Open Source High-Speed Oscilloscope (OSHO)

Final Project Report

Team Members:
Timothy Bullock, Afnan Ali, Evan Hoffman, Umair Aslam, Zaeem Gauher

Faculty Advisor:
Jens-Peter Kaps

ECE493-001

Date of Submission: May 4th, 2019

George Mason University
4400 University Dr, Fairfax VA 22030

1. Executive Summary

Oscilloscopes are one of the most useful devices in engineering applications where time-varying electrical signals need to be measured, analyzed, and recorded. Some of these applications require the use of high performance oscilloscopes due to the high frequencies of the signals that need to be measured. However, many students, hobbyists, and small engineering firms cannot afford these high performance devices due to their costs. Therefore, we have proposed a low-cost, open-source, and high performance alternative to the commercially available devices. This Open-Source High Speed Oscilloscope (OSHO) provides a 500Mhz bandwidth along with a 1 GSPS sampling rate at the fraction of the relatively low cost of $587. The following report details this proposed solution through its entire development process.

The solution can be categorized by three main aspects including the analog front-end circuit for measuring and digitizing the signal, the FPGA datapath for processing and routing the digitized data, and the software design for creating a user-interface and plotting. The approach and requirements used in the derivation of this modular design are discussed in detail. Furthermore, not only the high-level design but also the detailed circuit schematics, VHDL datapath, and software implementation are presented through this report. The testing methodology is introduced and the results are discussed for each of the three project aspects including the discussion of any deviations due to the pandemic situation at the time of this report. This project has been developed under the supervision and guidance of Dr. Jens-Peter Kaps.

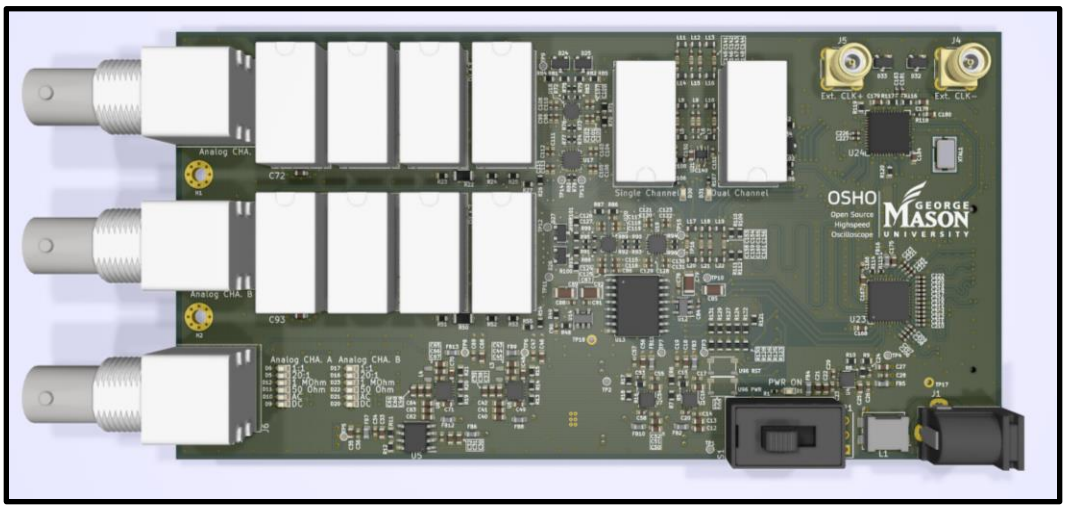


Figure 1: OSHO PCB CAD Model

2. Table of Contents

3. Problem and Solution Approach

3.1 Problem Statement

Digital oscilloscopes are indispensable tools for many engineering and scientific industries. Digital oscilloscopes “enable the user to debug, visualize and measure various signals,” which is especially crucial in lab settings where circuit testing and signal measurements are performed. However, in many applications such as RF design, high frequency signals can not be measured with standard low-cost oscilloscopes. This is especially a problem for students, hobbyists, and small engineering firms where funds are very limited. This not only a hindrance to the progress of their work, but also a detriment to education and innovation overall. Currently, oscilloscopes capable of high frequency analysis typically cost upwards of $6,000. Even with these high costs, these oscilloscopes often lack various features and usability aspects such as lack of external clock synchronization and a user friendly data download process. Although there are a few low cost options available on the market, their performance is extremely limited. Therefore, our project’s motivation is to create a low-cost, open-source, and high-speed oscilloscope solution that will be able to overcome these performance and financial limitations.

3.2 Proposed Design Solution

Our proposed solution is to create a system with three main components, an analog front end (in the form of a high speed custom PCB), a processing system (in the form of a MultiProcessor System on Chip (MPSoC) development board, and a user interface (in the form of of a web-based GUI). This solution Figure 2 below shows how this solution would interact with users, external inputs, and with itself in the form of the three main components of the system.

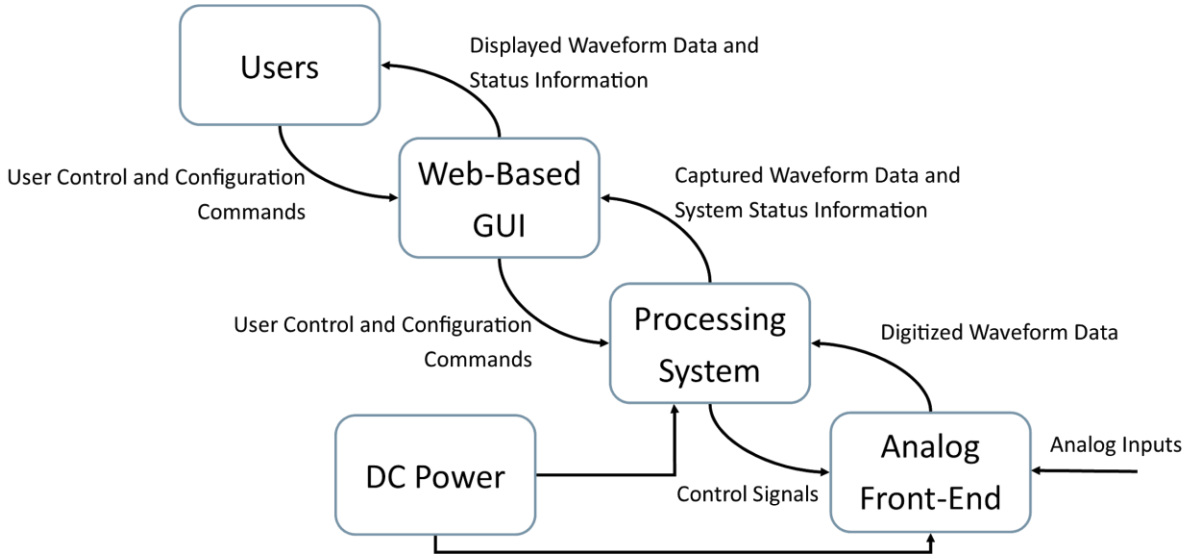


Figure 2: Proposed System Model

The analog front-end subsystem will primarily consist of the analog circuitry to precondition the incoming analog signals so that they may be optimally digitized by the ADC. The tasks that will be performed by the conditioning circuitry will include: attenuation, anti-aliasing filtration, variable gain amplification, coupling selection (AC or DC), DC offset selection, circuit overvoltage protection, and ADC clock generation/synchronization. Configurable aspects of this system such as DC offset will be configured through SPI commands from the processing subsystem. Once the analog inputs are conditioned properly, they will then be digitized by the ADC and sent to the processing subsystem. This front-end circuitry will be routed on a custom high-speed PCB that will be designed by our team. This board will be able to interface with the processing system through high and low speed mezzanine connectors.

The processing subsystem will consist of a MPSoC development board which includes programmable logic in the form of an FPGA as well as an ARM-based processor. The specific development board that will be used for this application will be the Avnet Ultra96-V2 which uses a Xilinx Zynq UltraScale+ MPSoC ZU3EG A484, has 2GB of LPDDR4 memory, and provides essential integrated peripherals such as USB3.0, an SD card slot, WiFi, and Mini DisplayPort. The programmable logic portion of this board will be used in conjunction with custom intellectual property (IP) blocks that will buffer the incoming raw digital data from the ADC and transform it to a standardized data packet format. These packets will then be sent to the system's main memory where processing can be conducted through an ARM processor that hosts a linux-based web server. The end user will be able to view and download waveform data and system status information as well as send configuration commands through this webserver.

The final foundational aspect of our preferred approach is a web-based GUI subsystem that will act as a client to the web server running on the Ultra96 board. This subsystem will act as the primary interface between the user and the overall system.

This subsystem will allow the user to enter system configuration commands (such as toggling between AC/DC coupling, configuring waveform triggers, etc) and download/display captured waveform data. This custom user interface should be responsive, intuitive, and effectively display captured waveform data. This aspect of the project will likely be programmed in Angular, and implemented incrementally, providing basic features at first, but adding more advanced features as time permits.

Providing a modular design proves to be the optimal solution to the problem because it will minimize cost while providing excellent analog capture performance. Additionally this approach will also provide a good basis for further open-source development.

This modular solution optimizes low-cost for multiple reasons. Much of the hardware cost will be absorbed by the fact that an external computer will be utilized for user interface. Furthermore, the front-end circuitry will be designed with cost-effective parts. For instance, the chosen ADC for this project is the HMCAD1511, which offers excellent performance for its price. Additionally, the effective price of the system is reduced if a compatible FPGA development board is already owned by the end user.

As stated earlier, this approach ensures that the system will be an excellent platform for future open source development. It will consist of open source software as well an open source development board, allowing the end users to customize it to their needs. The fact that the analog front-end is separate from the development board means that the front-end board could be used with other compatible MPSoC development boards (with minimal firmware porting). Additionally, the GUI for this system can also be customized and improved by users in an open source fashion.

3.3 Project Mission Requirements

Below is an outline of the project mission requirements for our project that were outlined at the beginning of the project. They served as an outline for what will define success in executing the project.

- The project shall design an oscilloscope that is an open source, low-cost alternative to commercially available oscilloscopes, and a high performance, feature rich alternative to existing open-source oscilloscopes.
- The project shall design a custom high-speed PCB that will easily interface with an Ultra96-V2 development board, as well as develop the supporting firmware and graphical user interface for the device.

3.4 System Operational Requirements

Below is an outline of the proposed system operational requirements for our solution that were outlined at the beginning of the project. They served as an outline for what will define a complete and functional system.

3.4.1 Input/Output Requirements

- The device shall have at least two analog input channels, one external clock input, and one external trigger input.
- The system will receive control and configuration commands as well as be able to responsively display captured data through a web client with an intuitive and responsive GUI.

3.4.2 External Interface Requirements

- The device will provide support for 1x and 10x passive probe inputs (50Ω and 1MΩ).
- Bayonet Neill–Concelman (BNC) connectors shall be used for the analog inputs, external clock input, and external trigger inputs.
- The system shall interface with a network capable computer through USB3.0 or WiFi.
- The system shall receive power from an external 5V DC power supply.

3.4.3 Functional Requirements

- The analog-to-digital converter (ADC) shall sample one input channel at 1 GSPS or two channels at 500 MSPS.
- The device will be able to measure analog inputs with a maximum input voltage of ±10V.
- The input analog circuitry shall achieve a 500 MHz bandwidth.
- The ADC shall be able to be configured to sample using either the FPGA clock or an external clock input (between 30 MHz and 1 GHz).
- The ADC output sample resolution shall be no less than 8 bits.
- The system's data capture shall have the ability to be triggered using both configurable edge triggers as well as a configurable external trigger input.

3.4.4 Technology and System-Wide Requirements

- The front-end device shall use a single 1GSPS ADC chip.
- The ADC data shall be processed and hosted on an onboard Linux web server using a Xilinx Zynq UltraScale+ multiprocessor systems-on-chip (MPSoC) aboard the Ultra96 Board.
- The analog front-end custom PCB should interface with the Ultra96 Board for data processing.
- Target FPGA development board shall have device driver firmware for interfacing with the ADC, and routing and storing ADC sample data in a memory device.
- Front-end programmable devices will be controlled using the Serial Peripheral Interface (SPI) or other serial protocol.
- The custom high-speed PCB and Ultra96 devices will interface with each other via the Ultra96's high-speed and low speed mezzanine connectors.
- The device should be low-cost ($600 or less).

3.5 Alternative Design Approaches

There are many possible solutions to the problem of providing a low-cost, high-speed, and feature-rich oscilloscope. Although the approach discussed above is the one that was determined to provide the best compromise between cost, performance, and features, it was still important to consider some alternative approaches at the beginning of this project. This ensured that our preferred approach was the optimal solution and as well as had backup approaches in case problems arose with our preferred approach. Alternative approaches that were considered are: using multiple ADCs, incorporating the MPSoC onto the same board as the analog front-end, and incorporating a display and physical controls as part of the device hardware.

3.5.1 One vs Multiple ADCs

In the development of our solution, having two analog input channels was listed as a key requirement as this provides a much more useful device. However, the issue with this is that there is no low-cost ADC that supports two channels at 1GSPS each. According to our preliminary research, the Analog Devices HMCAD1511 ($64) is the only low-cost ADC that supports 1GSPS [17]. This device can support multiple channels, but does not provide 1GSPS for each channel. Instead, the sampling rate is reduced immensely as more channels are utilized. This raised the question of whether multiple ADCs should be used to provide support for multiple analog inputs. It was concluded that due to cost limitations, this was not feasible. Due to this, we chose to utilize only one HMCAD1511 ADC, but offer a mode where the user can configure the analog front-end to handle two inputs at a lower sampling speed of 500MSPS. Additionally, data bandwidth issues were also cited as a reason to use lower sampling speeds with multiple input channels. However, if this proves to be overly complex and unexpectedly expensive, using separate ADCs for each channel may be reconsidered.

3.5.2 Using a MPSoC Development Board vs. a Single Board Solution

As the hardware for the Utra-96-V2 development board is open source, it was questioned whether or not this hardware should be incorporated into the front-end custom PCB to provide a more portable, single board solution. However, this was rejected in favor of using a development board that interfaces with the analog front-end through mezzanine connectors. This is because of two primary reasons. The first being that this provides unnecessary complexity to the hardware development and adds to the cost of production. Secondly, providing a single board solution would be a drawback to our target market of academics and hobbyists as they might only require the front-end device without our firmware for their specific application. Furthermore, they might prefer the multi-board solution so that the Utra-96 V2 remains reusable for different applications.

3.5.3 A Web-Based GUI vs. Physical Controls and On-Device Display

The last major alternative approach that was debated was the use of a graphical user interface vs physical controls and an incorporated display such as those in traditional bench oscilloscopes. It was decided that the web-based GUI solution should be favored over physical controls and on-device display. This was not only chosen because it minimizes the cost of the device, but also because it allows us to continually add more advanced controls to the device though software updates. Additionally, most users of this device would likely own a network capable computer which has a nicer display than any low-cost physical display we could include in our device. Furthermore, if a network connected device is used as the interface for this oscilloscope, it would ease the process for downloading captured data for external processing. However, the physical controls/display approach may prove a useful alternative for specific device controls for which a software approach may be too inconvenient.

3.6 Team Member Contributions

In order to successfully implement our chosen solution, each project team member was assigned specific responsibilities related to the project at the start of this project. Each of these assignments were devised so that they would align with the team member's abilities, interests, and experience. A summary of each member's roles and contributions is outlined below.

3.6.1 Afnan Ali

- Project Lead for GUI
    - Waveforms Live GUI development, debugging, testing
    - SimplePlotter development, debugging and testing

3.6.2 Umair Aslam

- Project lead for FPGA Logic Design
    - HACD firmware and Jupyter Notebook code debugging
    - HACD firmware testing and revision
    - OSHO Deserializer IP core development

3.6.3 Timothy Bullock

- Project Manager, responsible for major administrative aspects of project
- Project lead for printed circuit board layout and design
    - Revisions to analog front end circuitry
    - Selection of analog front end components and BOM generation
    - Layout and routing of high speed PCB
    - Responsible for component and PCB acquisition

3.6.4 Zaeem Gauher

- Project lead for Front-end Analog Circuit Design
    - HACD board testing and revision
    - Initial schematic design for analog front end circuit
    - PCB assembly of OSHO PCB

3.6.5 Evan Hoffman

- Project lead for Server development
    - Create server GUI workflow
    - Assist ultra96 server discovery

4. High Level Design

4.1 Level Zero Functional Decomposition

In order to provide a detailed overview of the system architecture for our solution, it is best to start with a functional decomposition of the system so that the system's functions can be related in a hierarchical manner. This decomposition will provide a top level overview of the system, then work downward to identify each of the main processes of the system, then continue downwards to identify the sub functions of each of these processes. The level zero decomposition provides a top level overview of the overall solution; it shows the overall system inputs and outputs. For our system, this is shown below in figure 3. It shows that the overall system will take in two analog inputs, an external clock input, an external trigger output, user commands, and DC power. The system then outputs status information and digitized waveform data.

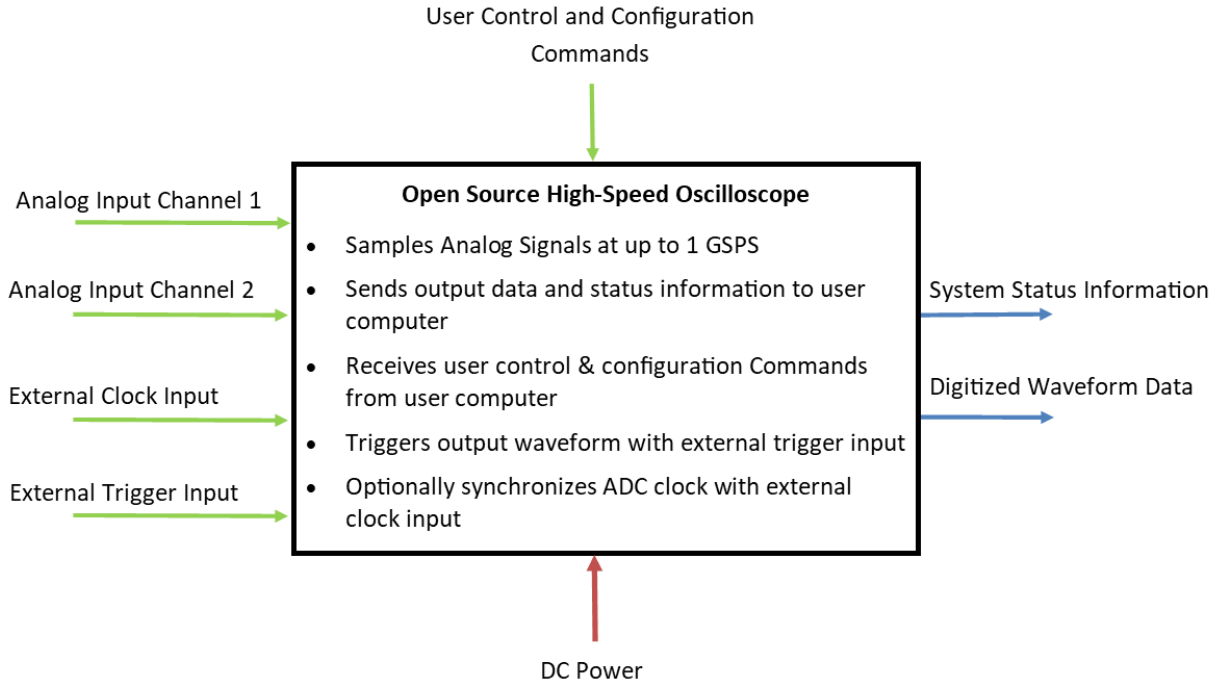


Figure 3: Level Zero Functional Decomposition of System

4.2 Level One Functional Decomposition

After the system is understood at the highest input/output level (level zero), the next step of functional decomposition is to identify the top level processes of the system. For our system this would include the analog front end power architecture, analog signal preconditioning, analog to digital conversion, ADC clock generation, data deserialization, data processing and hosting, and finally, display and interface processing. This is summarized in the level one diagram shown below (Figure 4). Once each of these main processes is identified at this level, they can then be further decomposed and discussed at the level two decomposition level. It is worth noting that from now on, the background color of each functional diagram will have a green, red, or blue background corresponding to which of the three main components each function is

located on. As shown below the green background will correspond to the analog front end PCB, red to the Ultra96 development board, and blue to the user's networked computer. Additionally, red arrows will correspond to the flow of power, green arrows, to the external inputs to the system, yellow arrows, to the flow of control via the serial peripheral interface (SPI), blue arrows will represent outputs to the system, and finally, gray arrows will reprepresent other internal data and control signals.

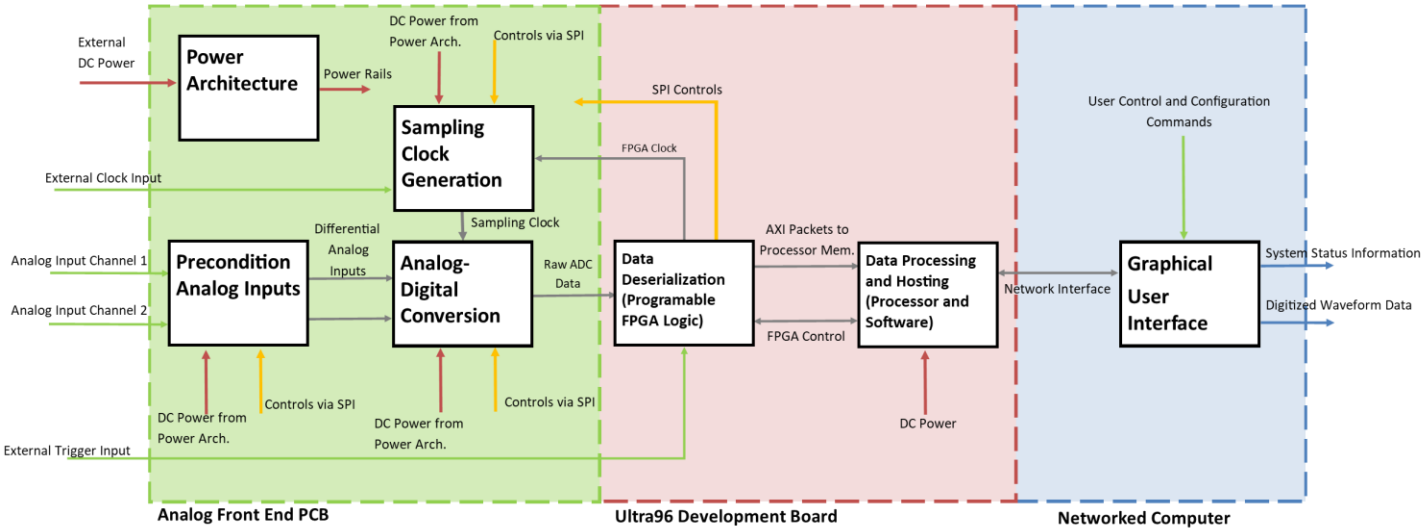


Figure 4: Level One Functional Decomposition of System

4.3 Level Two Functional Decomposition

Once the functionality of the system is understood at the level one demoposition level, the next step to providing a detailed overview to the system design is to take each of these top level processes and decompose them into their subprocesses. This is done for each of the top level processes shown in the level one functional architecture block diagram above (Figure 05). From each of these level two decompositions, we can then easily explain in great detail the hardware circuits, FPGA IPs, and software components that make up our solution's design; this will be done in section 5, Technical Design.

4.3.1 Analog Input Preconditioning Stage

The purpose of the analog input signal preconditioning stage/function is to take in the analog inputs and modify them so they can be most optimally digitized by the ADC. The functions that occur in this main process are: overvoltage protection, coupling selection, input impedance control, and offset generation, variable attenuation and amplification, and finally passing through a low pass anti-aliasing filter. The input signals are first passed though overvoltage protection to protect the remainder of the circuitry. These single ended signals are then modified by selecting DC or AC coupling, then put through a circuit to modify the input impedance of the analog inputs (creating either a 50Ω or 1MΩ input impedance as seen by the external circuitry). Next, the analog are attenuated so that the signals can fit within the full-scale range (FSR) of the ADC, and

depending on the effective DC component of the measured signals, the desired offset is added to the signal. Next, the signals are converted to a differential signal and are variably amplified so their amplitude more accurately fits the FSR of the ADC. Finally, the signals are sent through a low pass filter to reduce high frequency noise and limit the signals to the Shannon-Nyquist frequency dictated by the ADC maximum sampling rate. Configurable aspects of this system such as DC offset will be configured through SPI commands from the processing subsystem. This stage is located on the custom high-speed PCB that our team has designed.

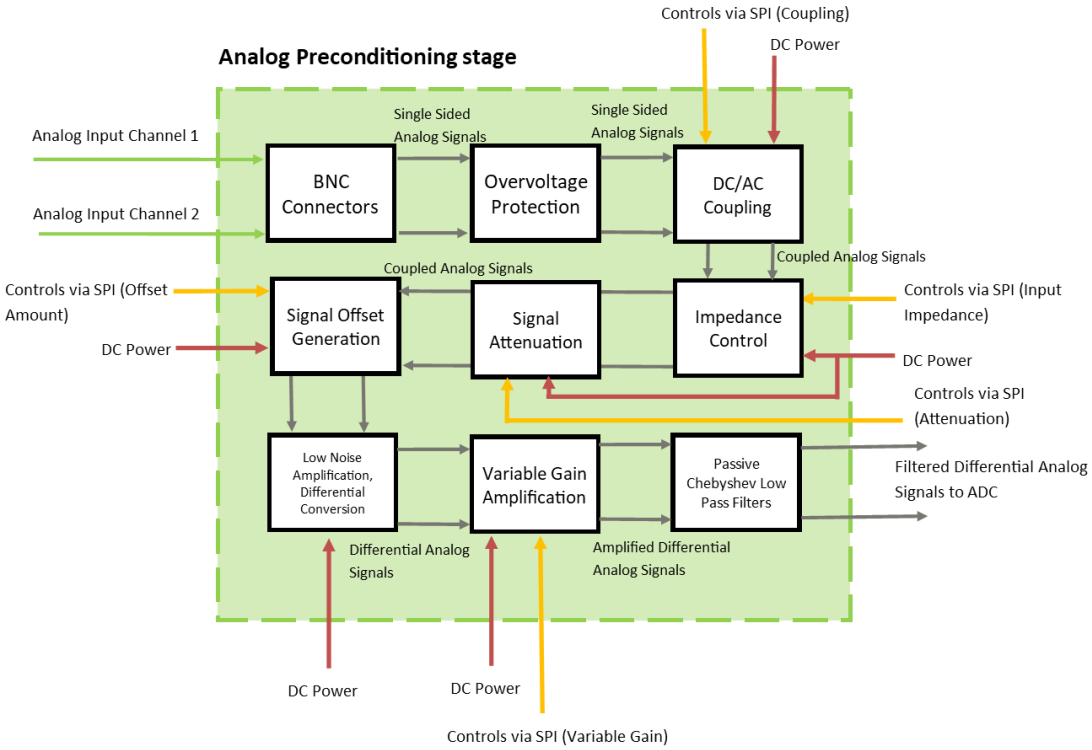


Figure 5: Level Two Decomposition: Analog Input Preconditioning Stage

4.3.2 Analog to Digital Conversion Stage

After the signals have been preconditioned, they are then sent to the analog to digital conversion stage/function. This stage is the simplest stage as it only consists of one main function and component, the high sampling speed ADC. This stage takes the preconditioned analog signals and outputs digital LVDS signals representing the digitized sample data. This digitized data is sent to the data deserialization stage in the programmable logic portion of the Ultra96 development board. This stage is also located on the custom high-speed PCB that has been designed by our team.

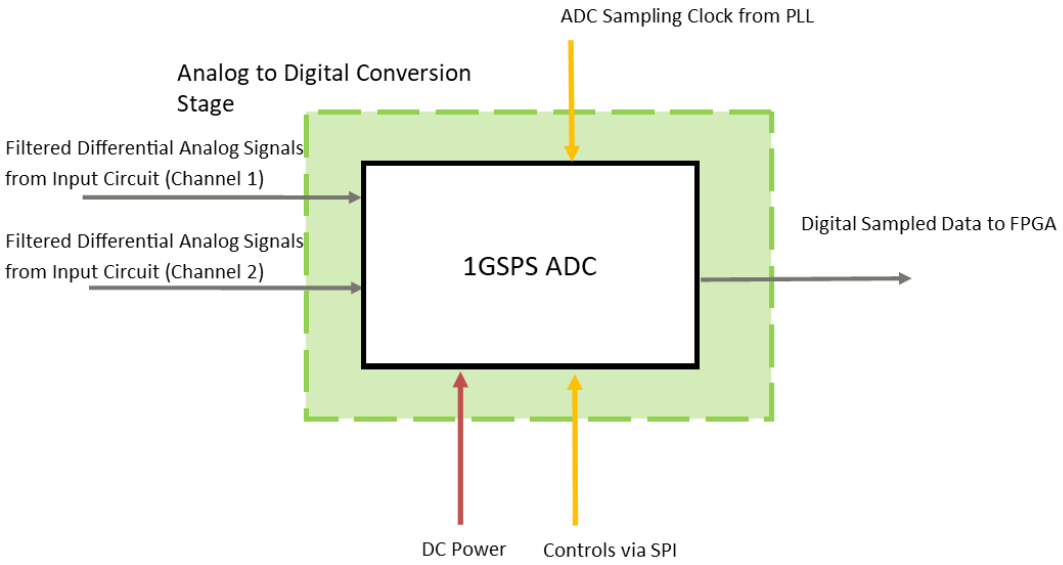


Figure 6: Level Two Decomposition: Analog to Digital Conversion Stage

4.3.3 ADC Sampling Clock Generation Stage

Another major function of the overall system is to generate the clock signal for the ADC to sample with. In this stage/function, either the FPGA clock, the external clock input, or a crystal oscillator reference are toggled between as an input into the phase locked loop (PLL) which matches or multiplies the frequency of the input signal to generate a low jitter clock signal for the ADC. This is the third stage/function that is located on the team's analog front end custom PCB.

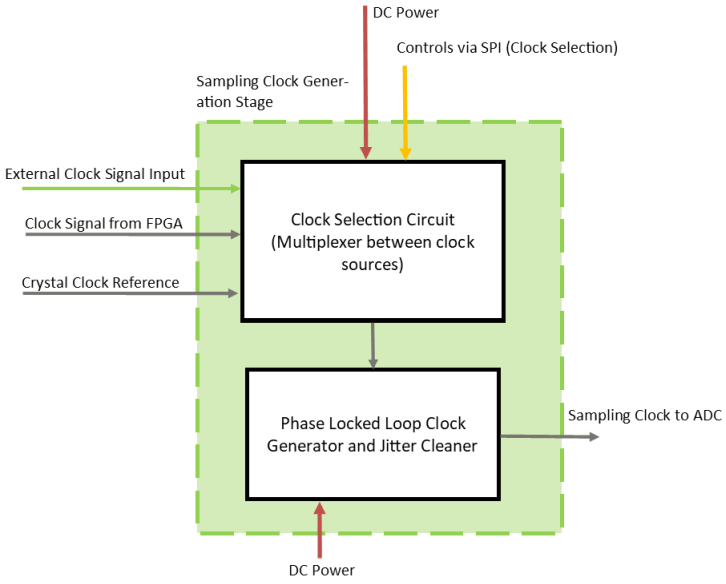


Figure 7: Level Two Decomposition: ADC Sampling Clock Generation Stage

4.3.4 Power Architecture Stage

The final stage that is located on the analog front end PCB is the power architecture stage. The purpose of this stage is to power the various components on the analog front end. In this stage/function DC power is provided from an external DC power supply. This power is then checked for overvoltage to protect the following circuitry, filtered to reduce common mode current and noise, and then regulated/converted in order create all of the required voltages for the remainder of the analog front end (such as 5V for the relays and input buffers, 3.3V for the PLL circuitry, etc.).

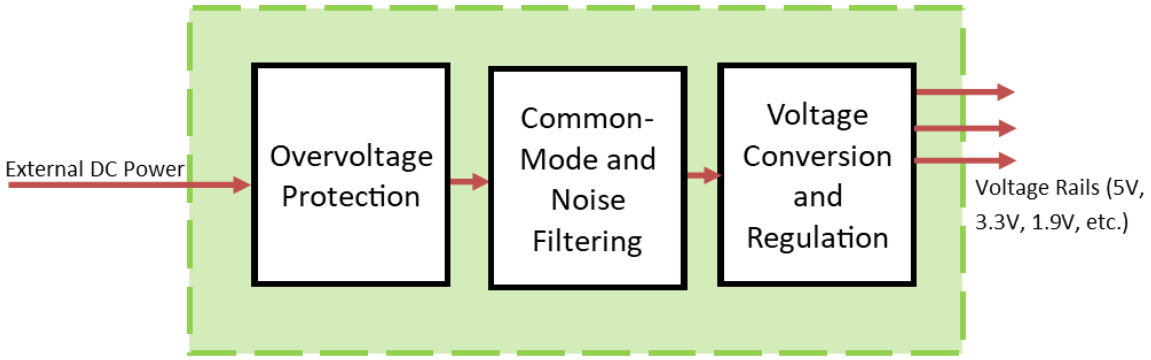


Figure 8: Level Two Decomposition: Power Architecture Stage

4.3.4 Data Deserialization Stage (FPGA Datapath)

The next stage/function of the system is the Data Deserialization Stage. The purpose of this stage is to receive data from the ADC, deserialize the data, and create 64-bit AXI packets which can then be loaded into main memory via direct memory access (DMA). This stage will also process the external trigger input in order to generate necessary control signals and stop the flow of digitized waveform data into memory. This stage will be implemented using the programmable logic (PL) portion of a MPSoC development board. More Specifically, this stage will be implemented on the Xilinx Zynq Ultrascale+ MPSoC ZUEG A484 that is on the Ultra96 V2 board and using custom and Xilinx provided Intellectual Property (IP) cores connected using the Advanced eXtensible Interface (AXI). The figure below shows the data deserialization stage (to the left) as well as the data hosting and processing stage to the right in order to show how the deserialized data gets sent to processor memory.

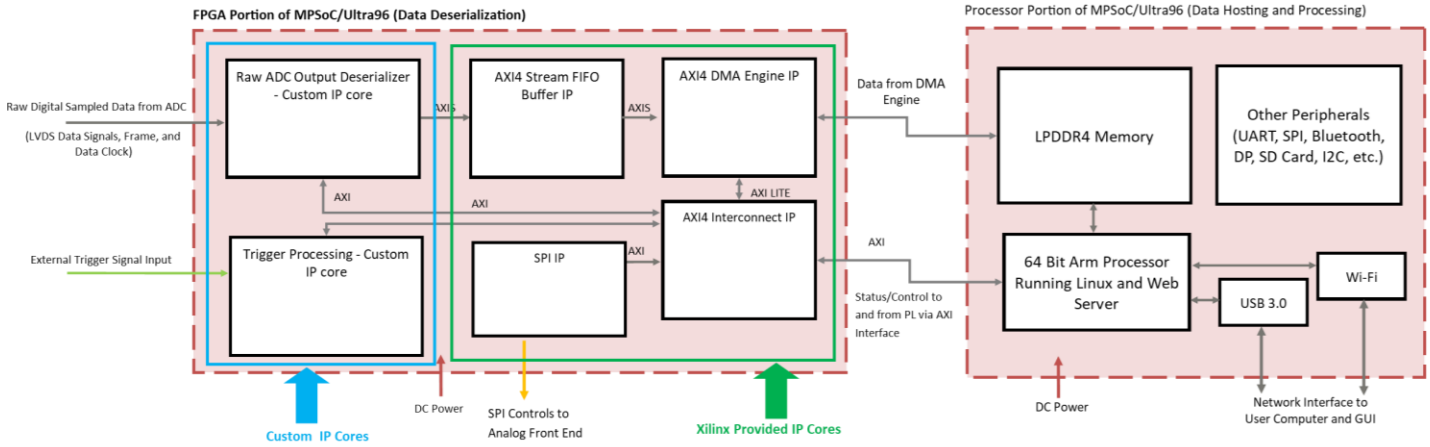


Figure 9: Level Two Decomposition: Data Buffering and Routing Stage

4.3.5 Data Processing and Hosting Stage (Server and Back-End Software)

After the output data from the adc is stored in main memory, the next thing that must happen to it is that it must be processed and hosted on a web server running on the ARM processor portion of the MPSoC. This ARM processor will be running a server which will host the web server that communicates with the user interface which will be implemented as a web client on a remote computer. This processor will also be running additional software in order to generate the commands to control the various components on the custom analog front-end via SPI, perform basic processing on the waveform data such as downsampling and converting the data into the desired protocol for the webserver, and finally to communicate with and control the PL portion of the chip via the AXI interface. This stage is shown on the right portion of the diagram below.

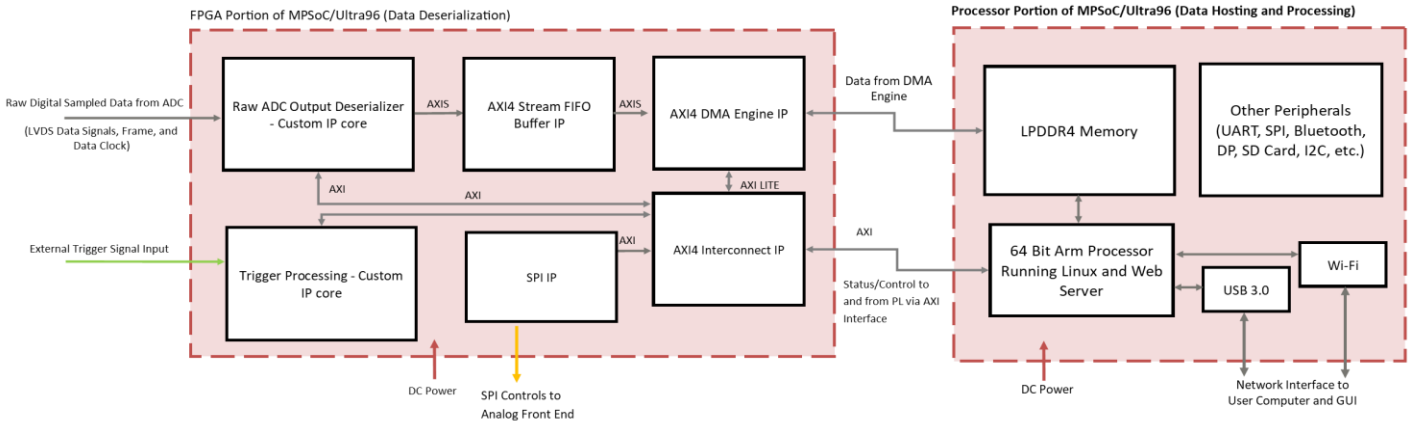


Figure 10: Level Two Decomposition: Data Processing and Hosting Stage

4.3.6 Graphical User Interface (GUI) Stage

The final stage/function that is required for our system is the user interface so that the user can control the system and view the digitized waveform data. This stage will consist of a webclient that will be running in a web browser running on the user's network capable computer. This web client will communicate with the server running on the Ultra96 in order to pass control and configuration information to the system and output waveform and status information.

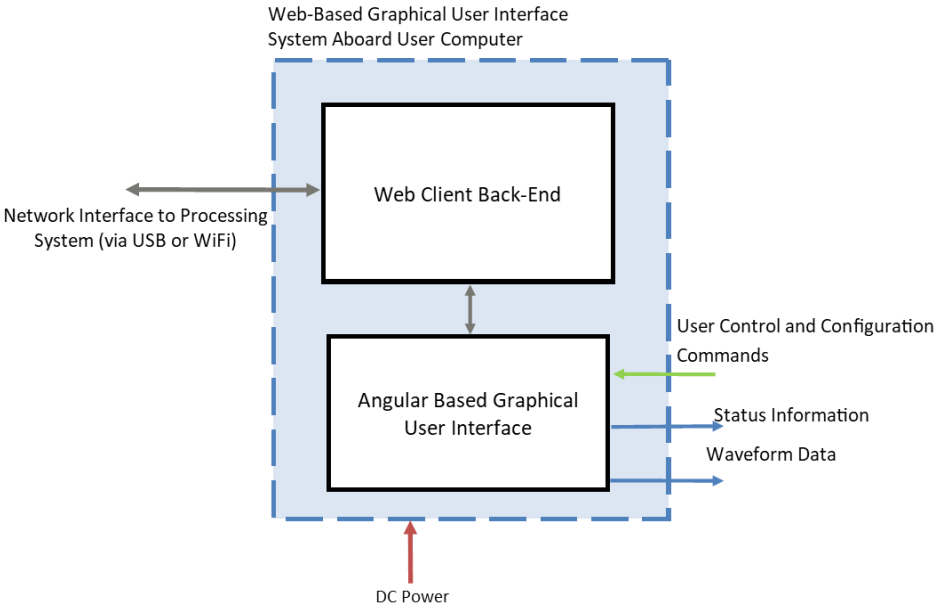


Figure 111: Level Two Decomposition: Graphical User Interface (GUI) Stage

4.4 Overall System Architecture

In Figure 12 below is a diagram of the main system components integrated into the overall system architecture. It can clearly be seen that the system will be divided into the three main subsystems: the analog front-end, the processing subsystem, and the web-based GUI. This diagram serves as the model in which data, power, and control flow throughout the system.

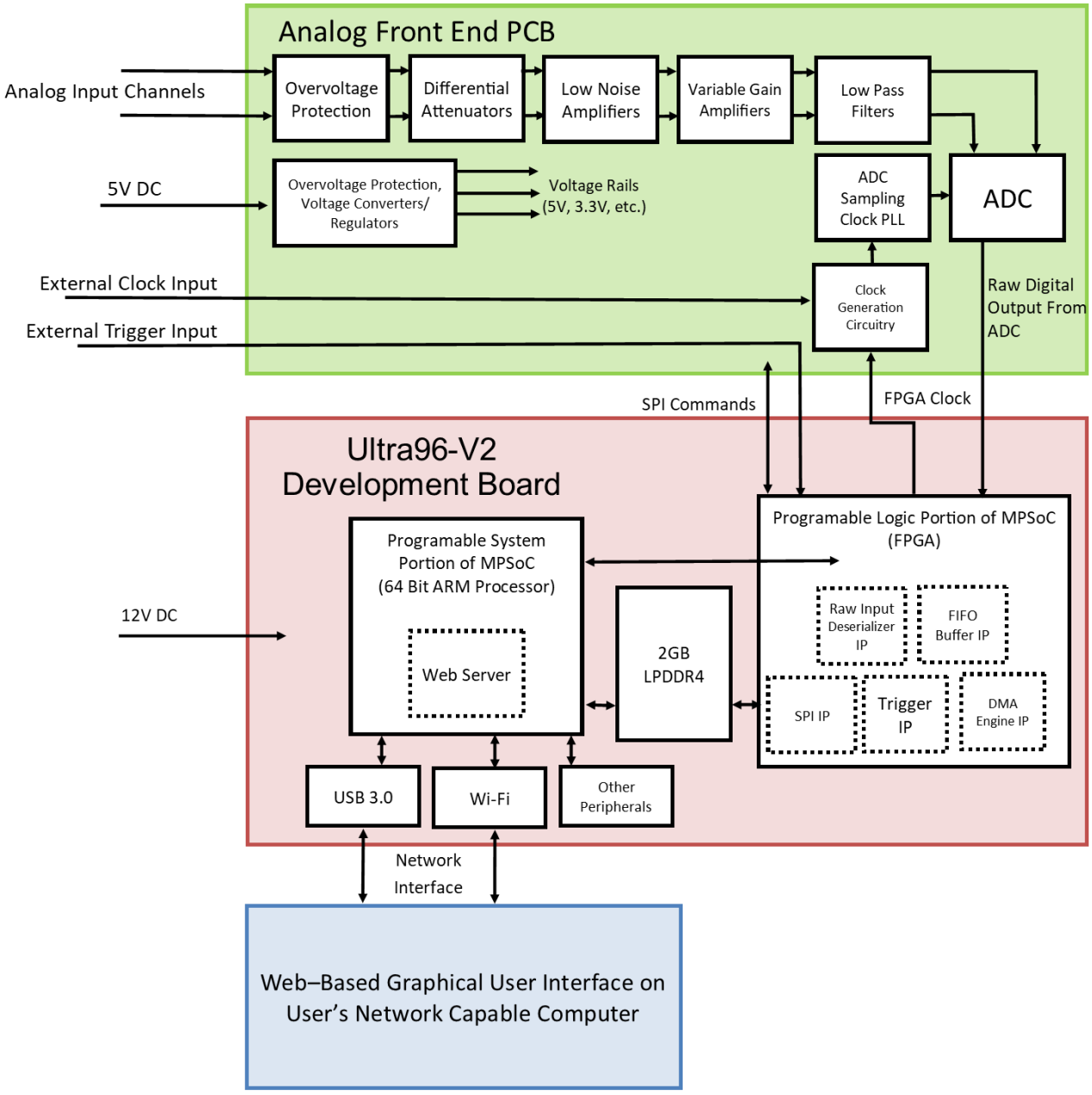


Figure 12: Overall System Architecture

4.5 Physical Architecture

An alternative representation of the system architecture is the physical architecture. The physical architecture consists of a hierarchical diagram that shows the main configuration items that make up the system. This includes major hardware and software components. This serves as a hierarchical overview of the major physical resources that will be required in our solution.

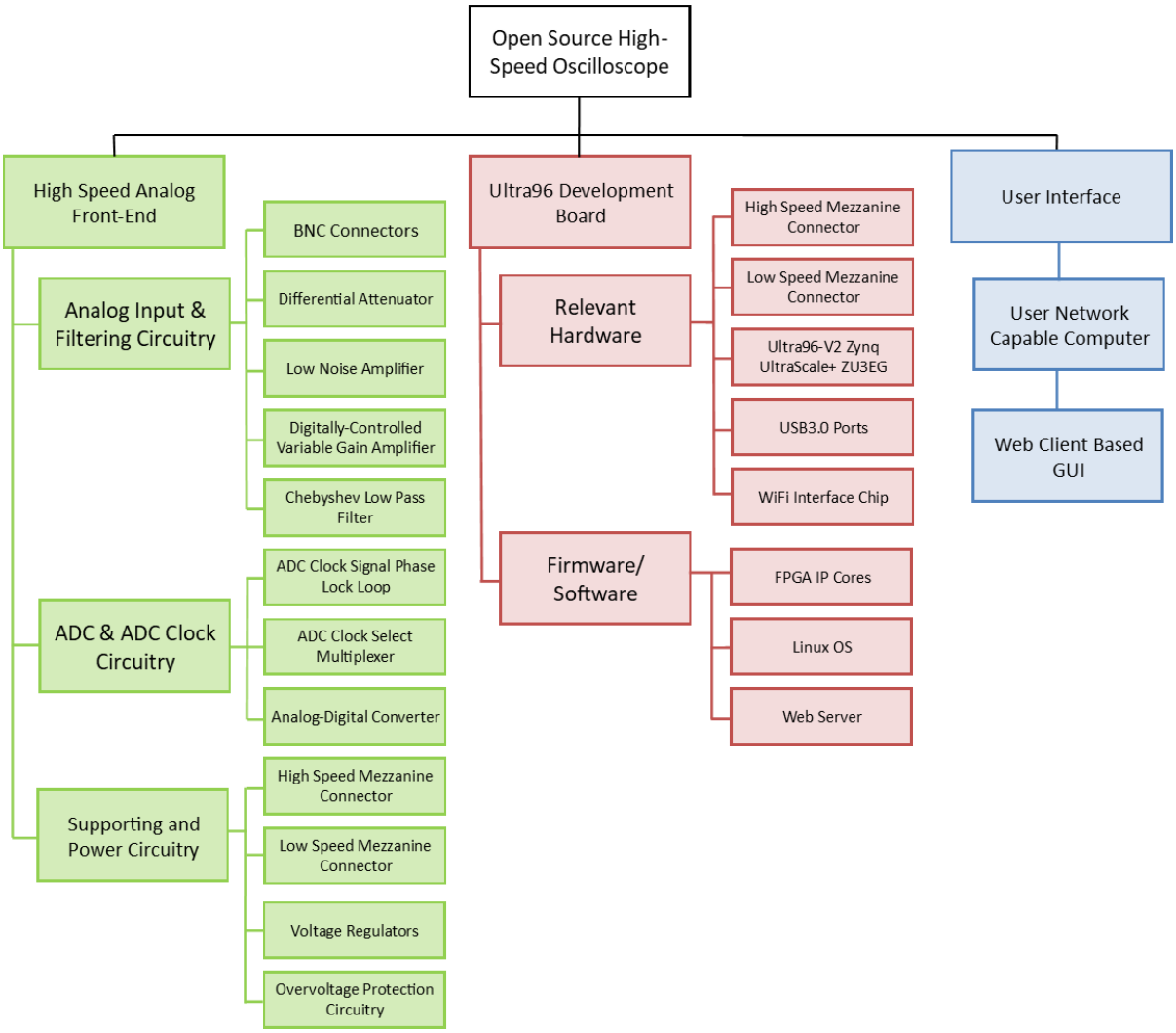


Figure 13: System Physical Architecture Architecture

5. Technical Design

5.1 Analog Front End

5.1.1 Discussion of Design

5.1.1.1 Power Architecture

The overall purpose of the power architecture of the analog front-end circuit is to essentially produce all the supply voltages required by various circuit elements used in the design. However, this process is a little more involved than it sounds. The input to the power circuitry is a 5V external power supply that can be received in two different ways. Through the use of a switch, the user can choose between a 5V external supply (through a barrel jack connector) or a 5V supply from the Ultra96-V2 board itself. It should be noted that the power supplied from Ultra96-V2 is only viable for a max current consumption of 3A. This 5V input is then filtered to remove any voltage spikes that might harm circuit elements using a common mode filter. High frequency noise is also removed throughout the overall circuit using ferrite beads. To further protect circuit elements, overvoltage protection is provided at the initial power supply input using an overvoltage protection controller. Lastly, various voltage regulators are used to produce any intermediate voltages and supply voltages for all chips in the circuit.

5.1.1.2 Analog Preconditioning Circuitry

It is extremely important to “condition” the input signals before they are digitized using the ADC. This not only protects the ADC from damage but also ensures that the signal does not contain unacceptable noise. There are various other aspects of signal conditioning that are discussed hereafter. The two analog input channels have a BNC connector interface and include gas discharge tubes at each input to protect against fast rising transients that could damage other chips. High speed relays are used to select between AC and DC coupling depending on if the DC component needs to be removed from signal. Another high speed relay is required for each signal to select between a 50 ohm and 1 Mohm impedance path. The 1 Mohm path is selected when a probe is being used to measure the signal. To increase the input signal range, a 20:1 pi attenuator is used to attenuate the signal. There is also a 1:1 path for each channel that can be selected for smaller signals using yet another high speed relay. The analog circuitry also contains a signal offset generation capability through the use of a digital potentiometer which is configured through Serial Peripheral Interface(SPI). Although many oscilloscopes use an analog potentiometer to control the offset, our design allows the user to control offset from the GUI itself. The signal preconditioning stage uses two amplifiers in each channel to amplify the signal such that their amplitude fits well within the full scale range of the ADC. This is done through the use of a low noise amplifier (LNA) and a variable gain amplifier (VGA). The LNA ensures that the signal is amplified without the noise being amplified along with it. The VGA (controlled via

SPI) fine tunes the gain/attenuation of the LNA output before it is sent to the ADC. Finally, the signals are sent through a 7th order Chebyshev low pass filter(LPF) to reduce high frequency noise and limit the signals to the Shannon-Nyquist frequency dictated by the ADC maximum sampling rate. The cutoff for the LPF in one channel mode is 500MHz as that is the bandwidth of the ADC. However, in two channel mode, the bandwidth is essentially split in half so two 250Mhz LPFs are utilized.

5.1.1.3 Sampling Clock Generation Circuitry

Another major function of the front-end circuit is to generate the clock signal for the ADC. With this circuitry, either the FPGA clock, the external clock, or the crystal oscillator is multiplexed as an input into the phase locked loop (PLL) which matches or multiplies the frequency of the input signal to generate a low jitter clock signal for the ADC. The PLL is configured through SPI by the software aspect of this project.

5.1.1.4 ADC Circuitry

After the signals have been preconditioned, they are then sent to the analog to digital converter. The ADC chip takes the preconditioned analog signals and outputs digital LVDS signals representing the digitized sample data. This data is sent to the data and buffering and routing stage on the FPGA board. The trace lengths of the digital LVDS outputs need to be matched on the PCB in high frequency applications for timing purposes.

5.1.1.5 Other Analog Front End Circuitry

There are a few other front-end analog circuitry elements that are worth noting. One of these elements is the bidirectional voltage level translator used to convert the 1.8V logic signals from the Ultra96-V2 to 3.3V logic levels for the various chips in the circuit and vice versa. These logic signals are mainly the Serial Peripheral Interface (SPI) signals used to configure elements such as the VGA, PLL, ADC, potentiometer, etc. There are also various LEDs used in the front-end circuit to indicate between AC vs. DC coupling, impedance, paths, etc. This not only improves user-experience but also makes debugging easier. Two important digital connectors used in the circuit are the high speed and low speed mezzanine connectors. Both of these connectors are used as an interface between the front-end PCB and the Ultra96-V2 board. The low speed connector is used as an interface for all control and SPI signals whereas the high speed connector is used to route the ADC data to the FPGA.

5.1.2 OSHO Board Schematics

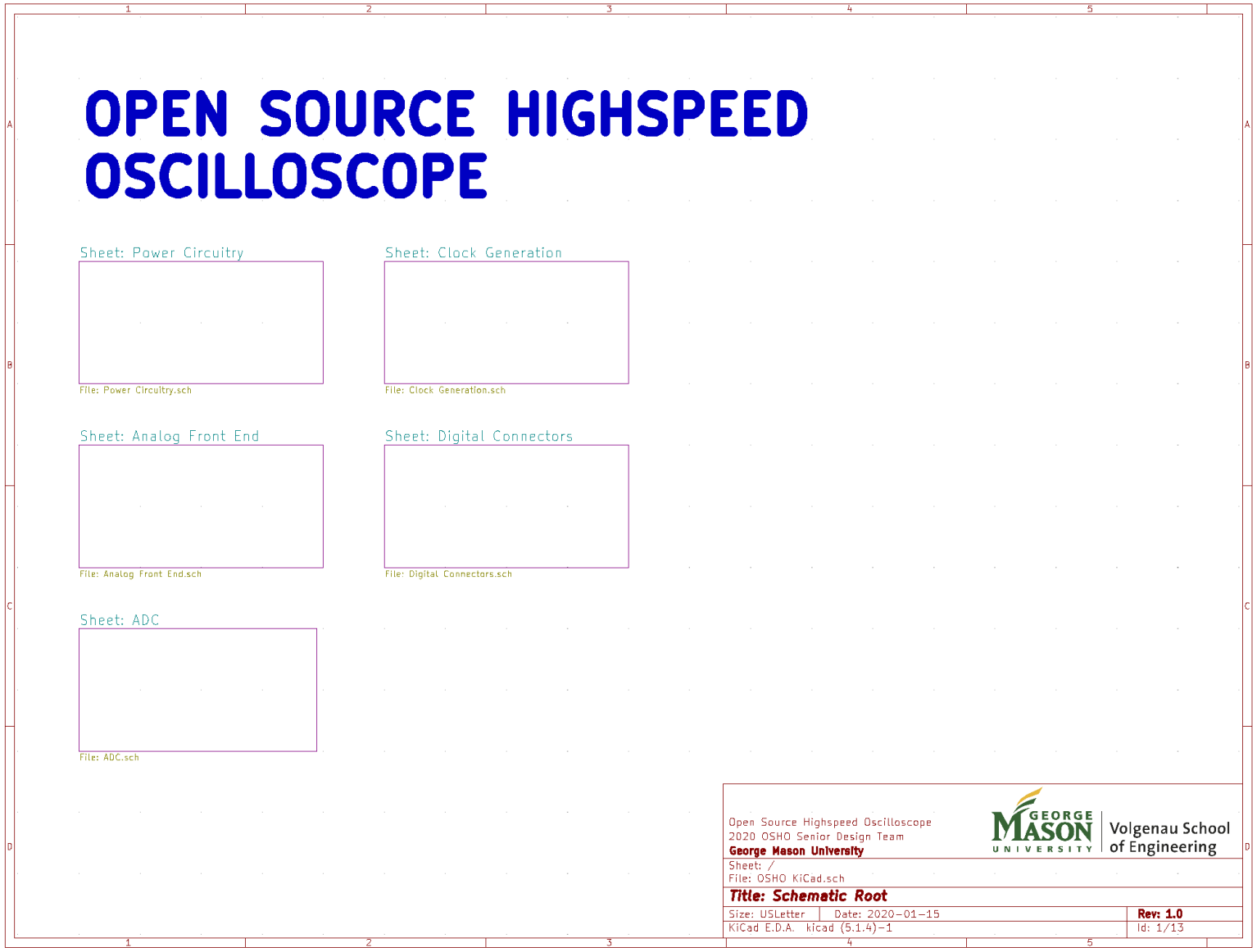

OPEN SOURCE HIGHSPEED OSCILLOSCOPE
Sheet: Power Circuitry
File: Power Circuitry.sch
Sheet: Clock Generation
File: Clock Generation.sch
Sheet: Analog Front End
File: Analog Front End.sch
Sheet: Digital Connectors
File: Digital Connectors.sch
Sheet: ADC
File: ADC.sch
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
GEORGE MASON UNIVERSITY
Volgenau School of Engineering
Sheet: /
File: OSHO KiCad.sch
Title: Schematic Root
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 1/13

Figure 14: Schematic Cover Page

5.1.2.1 Power Circuitry 1

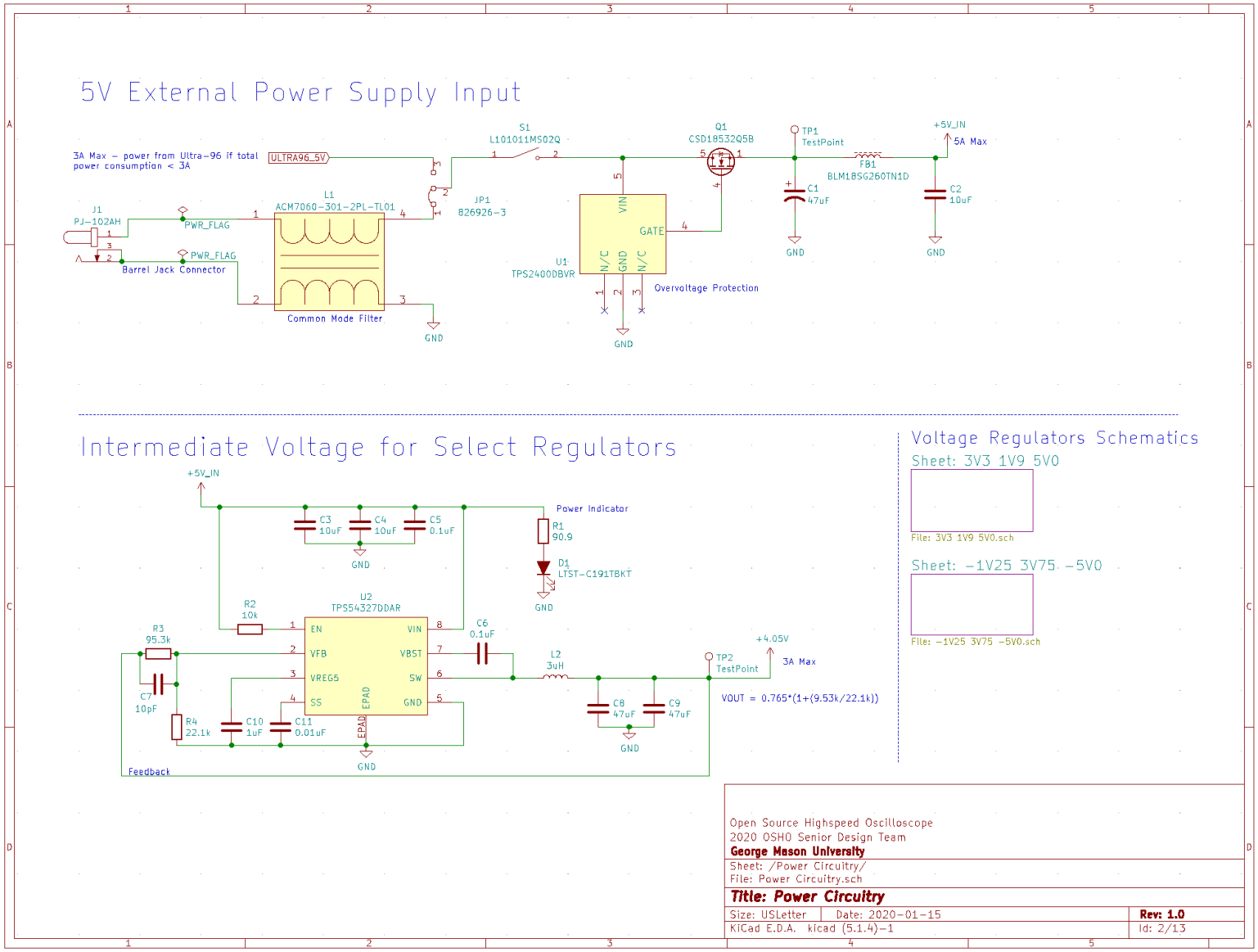

5V External Power Supply Input
3A Max - power from Ultra-96 if total power consumption < 3A
ULTRA96_5V
S1
L101011MS02Q
Q1
CSD18532Q5B
TP1
TestPoint
+5V_IN
5A Max
FB1
BLM18SG260TN1D
C1
47uF
C2
10uF
GND
J1
PJ-102AH
PWR_FLAG
Barrel Jack Connector
L1
ACM7060-301-2PL-TL01
Common Mode Filter
JP1
826926-3
U1
TPS2400DBVR
VIN
GATE
N/C
GND
Overvoltage Protection
Intermediate Voltage for Select Regulators
+5V_IN
C3
10uF
C4
10uF
C5
0.1uF
Power Indicator
R1
90.9
D1
LTST-C191TBKT
R2
10k
U2
TPS54327DDAR
EN
VFB
VREG5
SS
VIN
VBST
SW
GND
EPAD
R3
95.3k
C7
10pF
R4
22.1k
C10
1uF
C11
0.01uF
C6
0.1uF
L2
3uH
C8
47uF
C9
47uF
TP2
TestPoint
+4.05V
3A Max
VOUT = 0.765*(1+(9.53k/22.1k))
Feedback
Voltage Regulators Schematics
Sheet: 3V3 1V9 5V0
File: 3V3 1V9 5V0.sch
Sheet: -1V25 3V75 -5V0
File: -1V25 3V75 -5V0.sch
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Power Circuitry/
File: Power Circuitry.sch
Title: Power Circuitry
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 2/13

Figure 15: Power Architecture Schematics (Page 1 of 3)

5.1.2.2 Power Circuitry 2

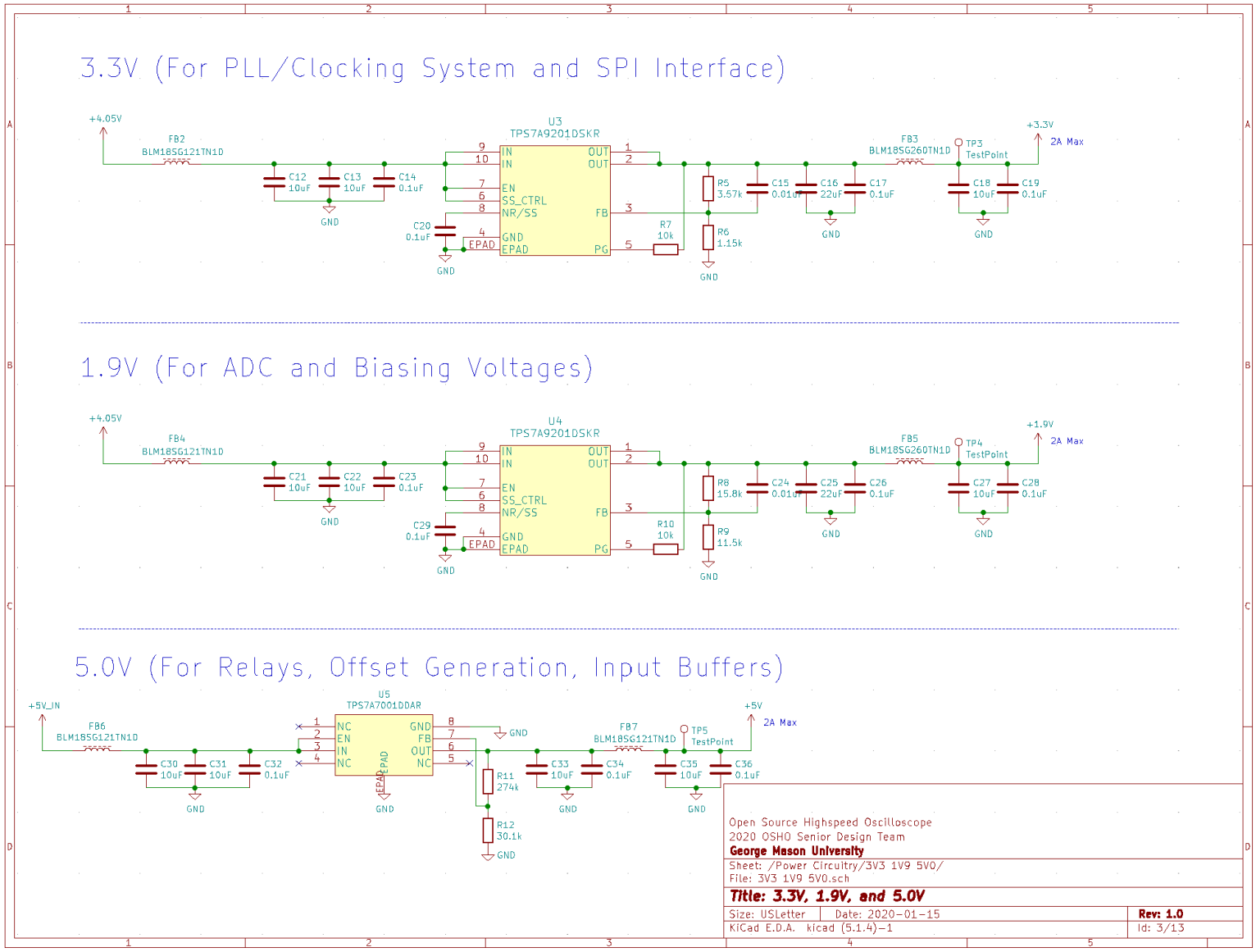

3.3V (For PLL/Clocking System and SPI Interface)
+4.05V
FB2
BLM18SG121TN1D
U3
TPS7A9201DSKR
IN
IN
EN
SS_CTRL
NR/SS
GND
EPAD
OUT
OUT
FB
PG
C12 10uF
C13 10uF
C14 0.1uF
C20 0.1uF
R5 3.57k
R6 1.15k
R7 10k
C15 0.01uF
C16 22uF
C17 0.1uF
C18 10uF
C19 0.1uF
FB3
BLM18SG260TN1D
TP3
TestPoint
+3.3V
2A Max
GND
1.9V (For ADC and Biasing Voltages)
+4.05V
FB4
BLM18SG121TN1D
U4
TPS7A9201DSKR
C21 10uF
C22 10uF
C23 0.1uF
C29 0.1uF
R8 15.8k
R9 11.5k
R10 10k
C24 0.01uF
C25 22uF
C26 0.1uF
C27 10uF
C28 0.1uF
FB5
BLM18SG260TN1D
TP4
TestPoint
+1.9V
2A Max
GND
5.0V (For Relays, Offset Generation, Input Buffers)
+5V_IN
FB6
BLM18SG121TN1D
U5
TPS7A7001DDAR
NC
EN
IN
NC
GND
FB
OUT
NC
EPAD
C30 10uF
C31 10uF
C32 0.1uF
R11 274k
R12 30.1k
C33 10uF
C34 0.1uF
C35 10uF
C36 0.1uF
FB7
BLM18SG121TN1D
TP5
TestPoint
+5V
2A Max
GND
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Power Circuitry/3V3 1V9 5V0/
File: 3V3 1V9 5V0.sch
Title: 3.3V, 1.9V, and 5.0V
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 3/13

Figure 16: Power Architecture Schematics (Page 2 of 3)

5.1.2.3 Power Circuitry 3

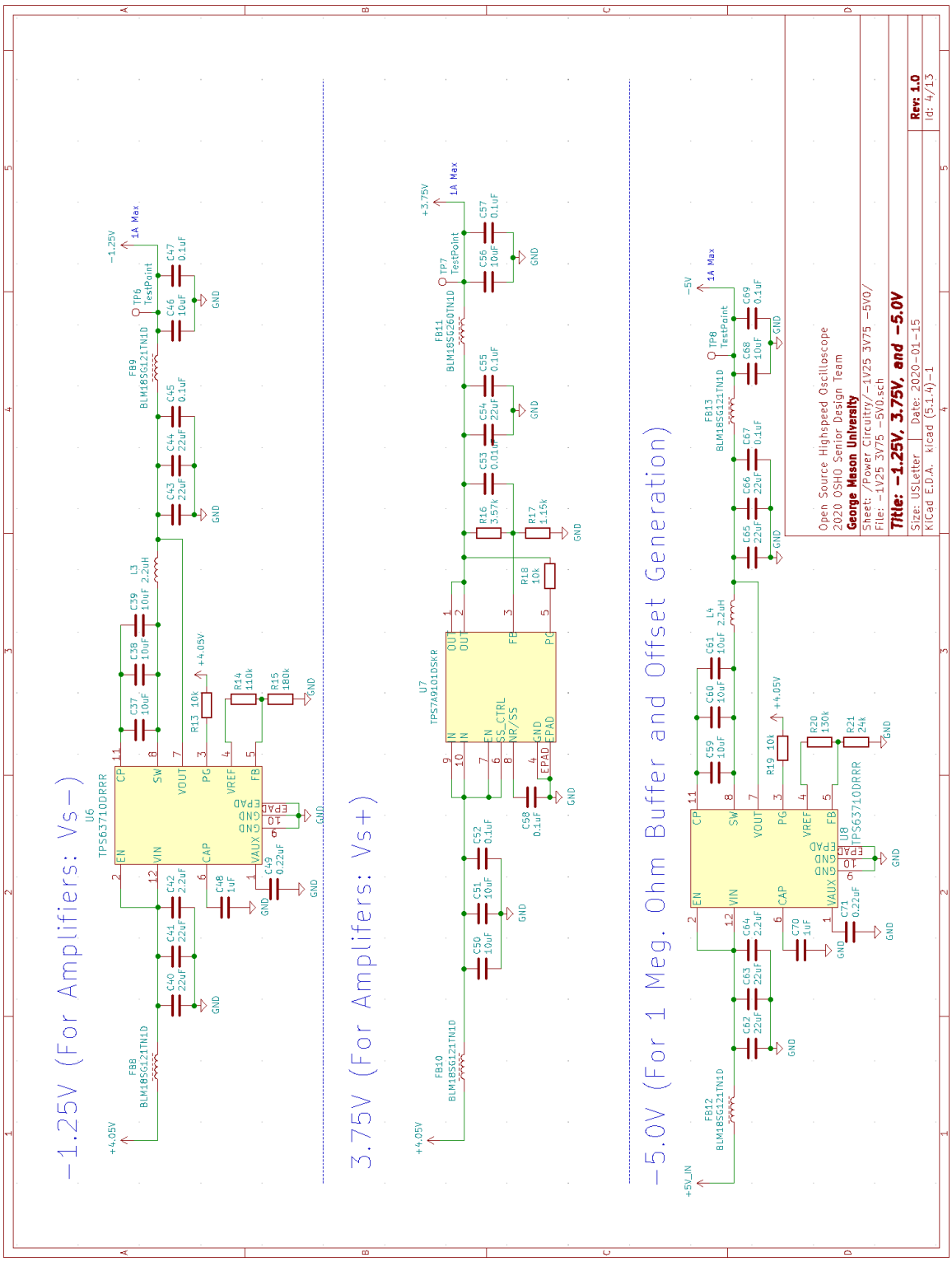

-1.25V (For Amplifiers: Vs-)
3.75V (For Amplifers: Vs+)
-5.0V (For 1 Meg. Ohm Buffer and Offset Generation)
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Power Circuitry/-1V25 3V75 -5V0/
File: -1V25 3V75 -5V0.sch
Title: -1.25V, 3.75V, and -5.0V
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 4/13

Figure 17: Power Architecture Schematics (Page 3 of 3)

5.1.2.3 Analog Front End

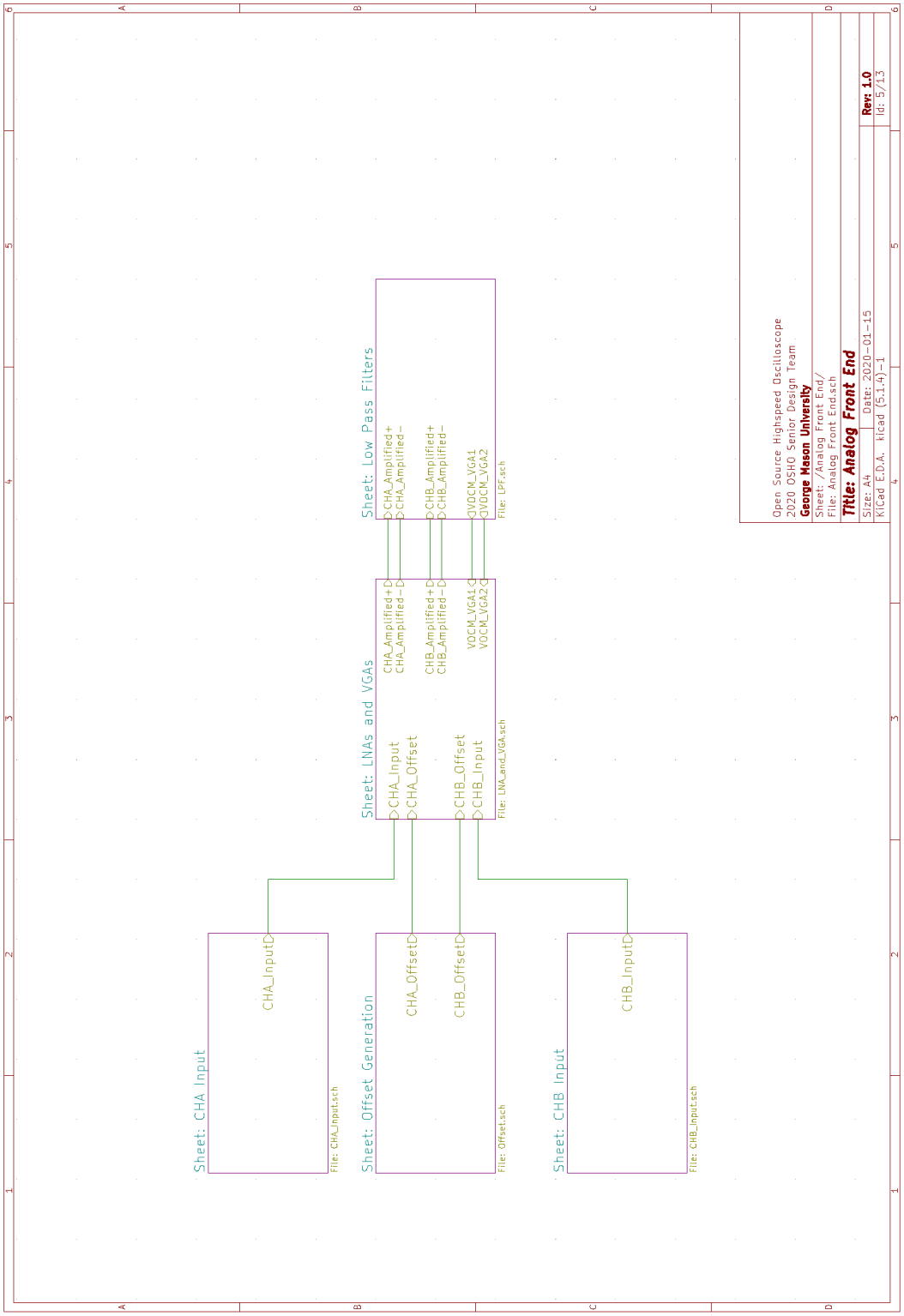

Sheet: CHA Input
CHA_InputD
File: CHA_Input.sch
Sheet: Offset Generation
CHA_OffsetD
CHB_OffsetD
File: Offset.sch
Sheet: CHB Input
CHB_InputD
File: CHB_Input.sch
Sheet: LNAs and VGAs
CHA_Input
CHA_Offset
CHB_Offset
CHB_Input
CHA_Amplified+
CHA_Amplified-
CHB_Amplified+
CHB_Amplified-
VOCM_VGA1
VOCM_VGA2
File: LNA_and_VGA.sch
Sheet: Low Pass Filters
CHA_Amplified+
CHA_Amplified-
CHB_Amplified+
CHB_Amplified-
VOCM_VGA1
VOCM_VGA2
File: LPF.sch
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Analog Front End/
File: Analog Front End.sch
Title: Analog Front End
Size: A4
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kiCad (5.1.4)-1
Id: 5/13

Figure 18: Analog Front End Schematics Hierarchical Page

5.1.2.4 Channel A Input Stage

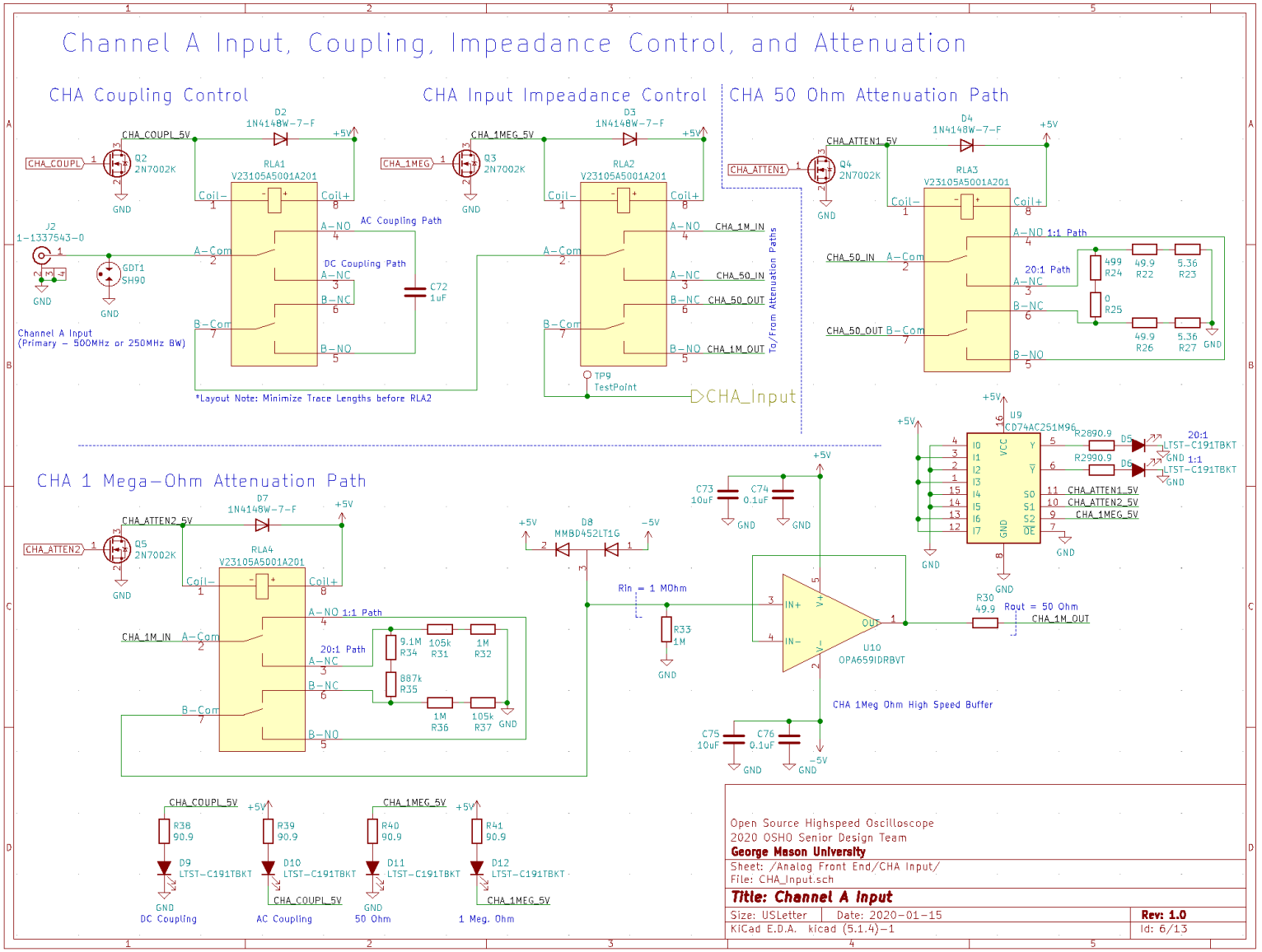

Channel A Input, Coupling, Impeadance Control, and Attenuation
CHA Coupling Control
CHA Input Impeadance Control
CHA 50 Ohm Attenuation Path
CHA 1 Mega-Ohm Attenuation Path
CHA_COUPL
CHA_COUPL_5V
CHA_1MEG
CHA_1MEG_5V
CHA_ATTEN1
CHA_ATTEN1_5V
CHA_ATTEN2
CHA_ATTEN2_5V
Q2 2N7002K
Q3 2N7002K
Q4 2N7002K
Q5 2N7002K
D2 1N4148W-7-F
D3 1N4148W-7-F
D4 1N4148W-7-F
D7 1N4148W-7-F
D8 MMBD452LT1G
RLA1 V23105A5001A201
RLA2 V23105A5001A201
RLA3 V23105A5001A201
RLA4 V23105A5001A201
J2 1-1337543-0
GDT1 SH90
Channel A Input
(Primary - 500MHz or 250MHz BW)
AC Coupling Path
DC Coupling Path
C72 1uF
*Layout Note: Minimize Trace Lengths before RLA2
To/From Attenuation Paths
TP9 TestPoint
CHA_Input
CHA_1M_IN
CHA_50_IN
CHA_50_OUT
CHA_1M_OUT
20:1 Path
1:1 Path
499 R24
49.9 R22
5.36 R23
0 R25
49.9 R26
5.36 R27
9.1M R34
105k R31
1M R32
887k R35
1M R36
105k R37
C73 10uF
C74 0.1uF
C75 10uF
C76 0.1uF
Rin = 1 MOhm
R33 1M
U10 OPA659IDRBVT
CHA 1Meg Ohm High Speed Buffer
R30 49.9
Rout = 50 Ohm
U9 CD74AC251M96
R2890.9
R2990.9
D5 LTST-C191TBKT
D6 LTST-C191TBKT
R38 90.9
R39 90.9
R40 90.9
R41 90.9
D9 LTST-C191TBKT
D10 LTST-C191TBKT
D11 LTST-C191TBKT
D12 LTST-C191TBKT
DC Coupling
AC Coupling
50 Ohm
1 Meg. Ohm
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Analog Front End/CHA Input/
File: CHA_Input.sch
Title: Channel A Input
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 6/13

Figure 19: Analog Input for Channel A Schematics

5.1.2.5 Analog Offset Generation

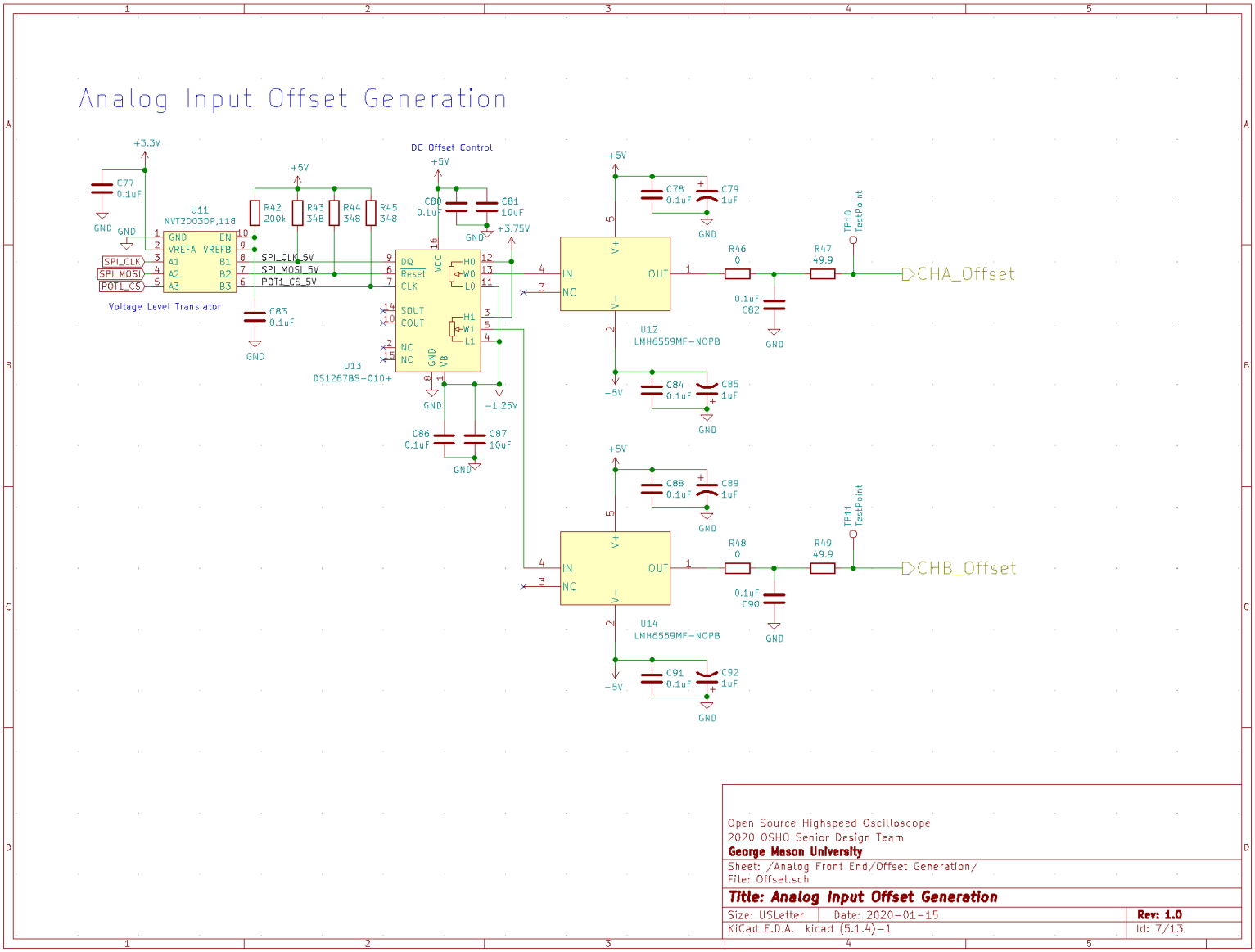

Analog Input Offset Generation
DC Offset Control
Voltage Level Translator
U11
NVT2003DP,118
U12
LMH6559MF-NOPB
U13
DS1267BS-010+
U14
LMH6559MF-NOPB
CHA_Offset
CHB_Offset
SPI_CLK
SPI_MOSI
POT1_CS
SPI_CLK_5V
SPI_MOSI_5V
POT1_CS_5V
+3.3V
+5V
+3.75V
-1.25V
-5V
GND
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Analog Front End/Offset Generation/
File: Offset.sch
Title: Analog Input Offset Generation
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 7/13

Figure 20: Analog Input Offset Generation Schematics

5.1.2.5 Channel B Input Stage

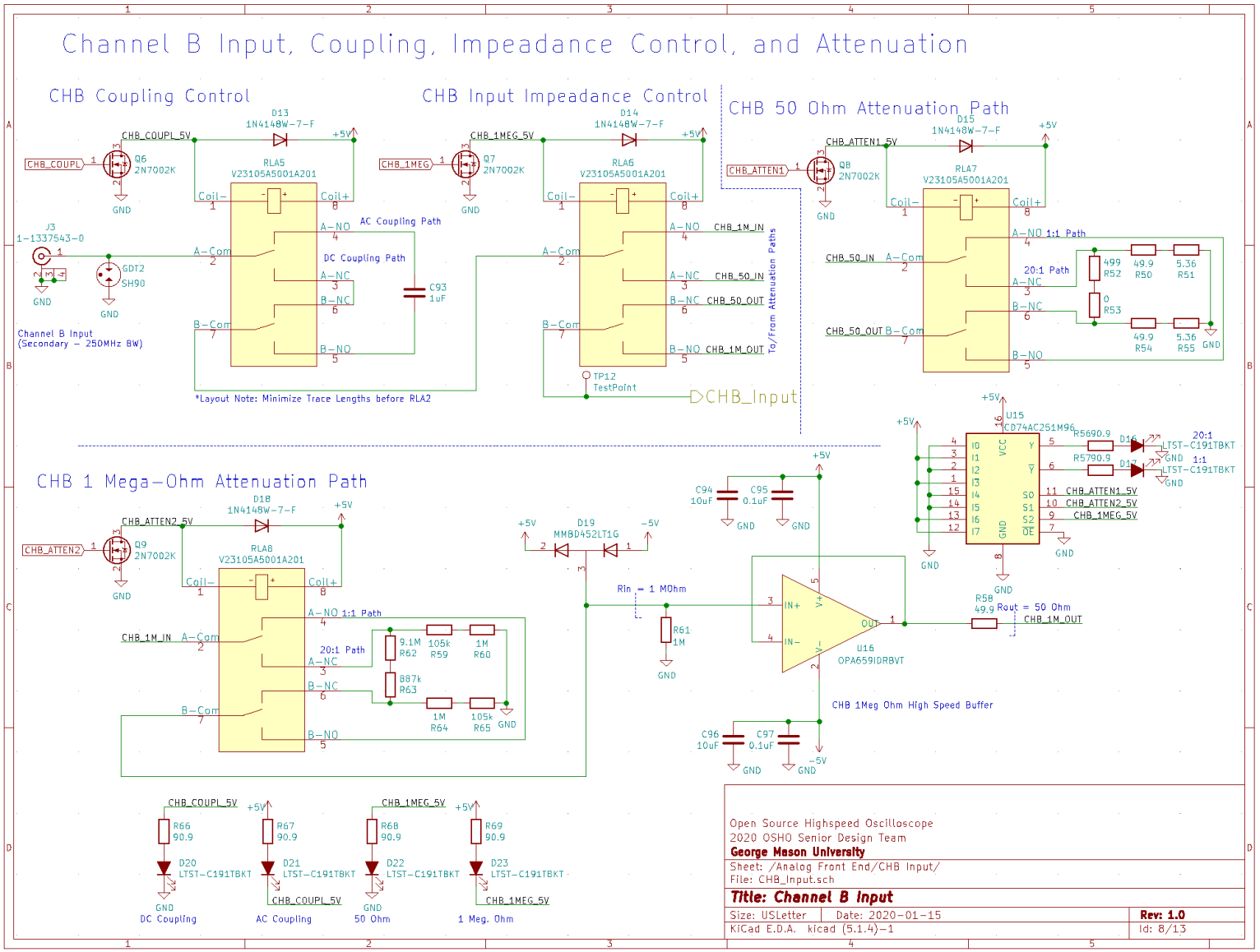

Channel B Input, Coupling, Impeadance Control, and Attenuation
CHB Coupling Control
CHB Input Impeadance Control
CHB 50 Ohm Attenuation Path
CHB 1 Mega-Ohm Attenuation Path
AC Coupling Path
DC Coupling Path
Channel B Input
(Secondary - 250MHz BW)
*Layout Note: Minimize Trace Lengths before RLA2
To/From Attenuation Paths
CHB_Input
20:1 Path
1:1 Path
CHB 1Meg Ohm High Speed Buffer
Rin = 1 MOhm
Rout = 50 Ohm
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Analog Front End/CHB Input/
File: CHB_Input.sch
Title: Channel B Input
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 8/13

Figure 21: Analog Input for Channel B Schematics

5.1.2.6 Low Noise and Variable Amplifiers

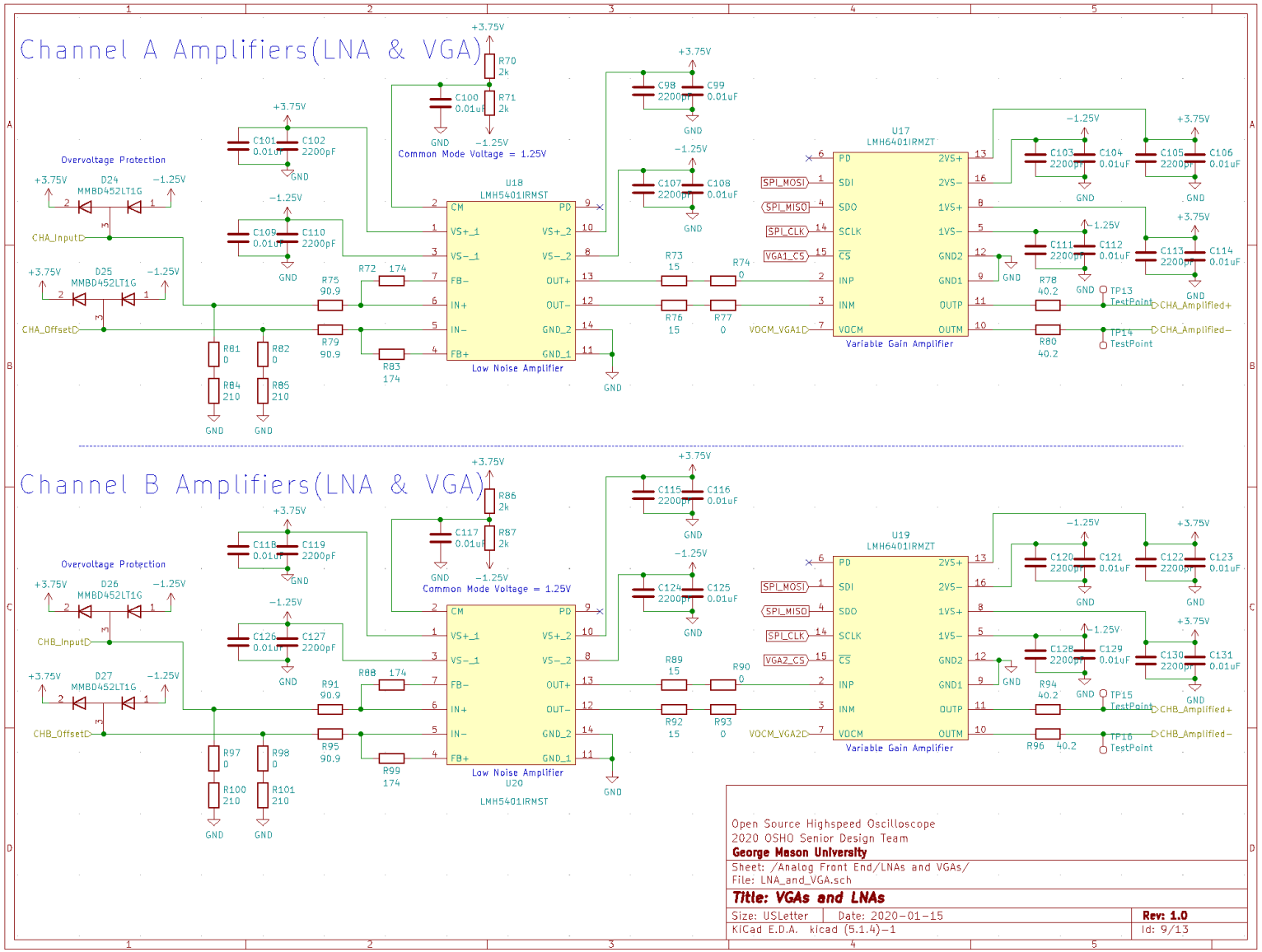

Channel A Amplifiers(LNA & VGA)
Overvoltage Protection
CHA_Input
CHA_Offset
D24
D25
MMBD452LT1G
Common Mode Voltage = 1.25V
U18
LMH5401IRMST
Low Noise Amplifier
U17
LMH6401IRMZT
Variable Gain Amplifier
SPI_MOSI
SPI_MISO
SPI_CLK
VGA1_CS
VOCM_VGA1
CHA_Amplified+
CHA_Amplified−
Channel B Amplifiers(LNA & VGA)
Overvoltage Protection
CHB_Input
CHB_Offset
D26
D27
MMBD452LT1G
Common Mode Voltage = 1.25V
U20
LMH5401IRMST
Low Noise Amplifier
U19
LMH6401IRMZT
Variable Gain Amplifier
VGA2_CS
VOCM_VGA2
CHB_Amplified+
CHB_Amplified−
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Analog Front End/LNAs and VGAs/
File: LNA_and_VGA.sch
Title: VGAs and LNAs
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 9/13

Figure 22: Low Noise Amplifiers and Variable Gain Amplifiers Schematics

5.1.2.7 Low Pass Filters

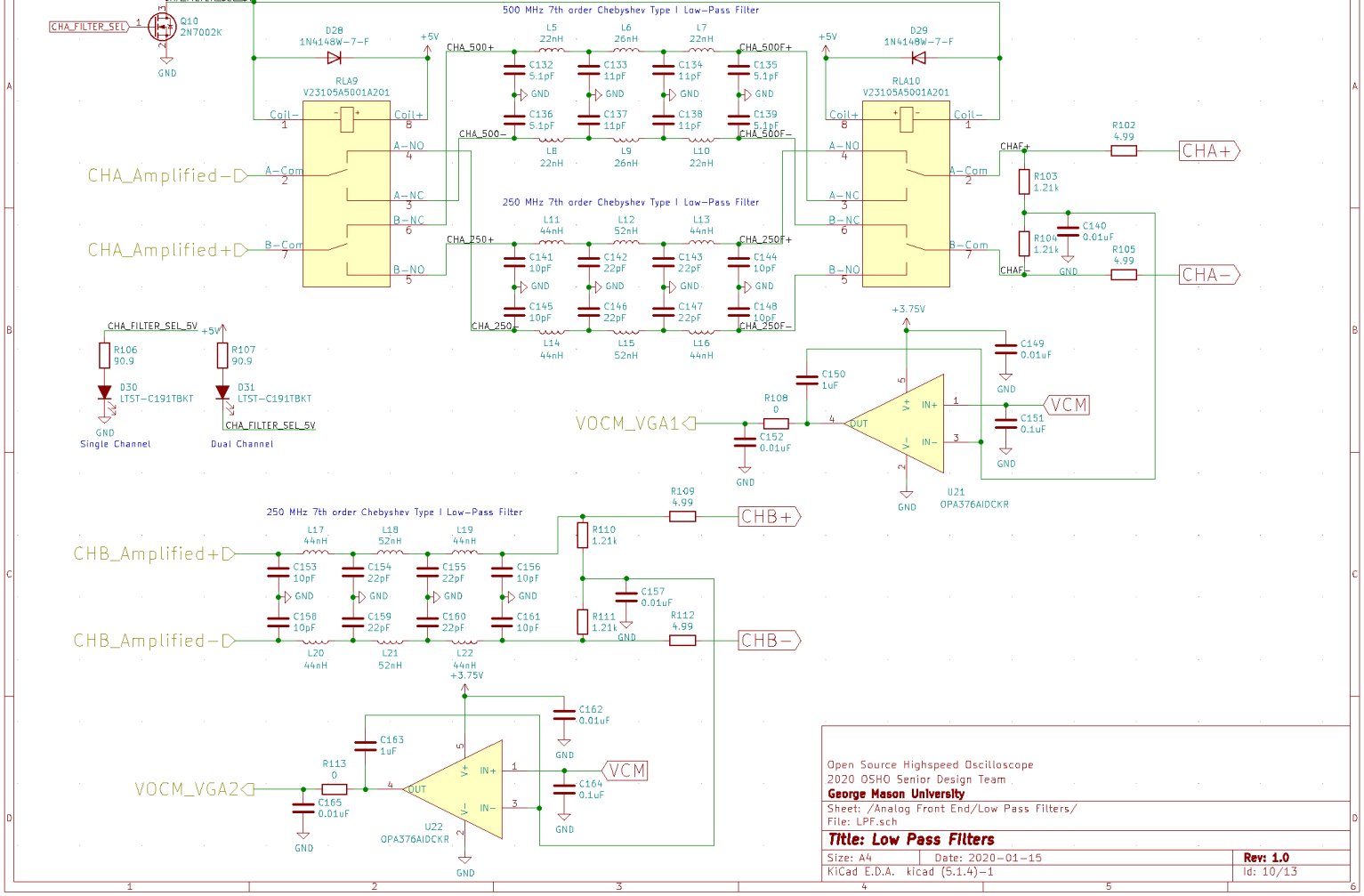

500 MHz 7th order Chebyshev Type I Low-Pass Filter
250 MHz 7th order Chebyshev Type I Low-Pass Filter
CHA_FILTER_SEL
Q10
2N7002K
D28
1N4148W-7-F
D29
1N4148W-7-F
+5V
GND
RLA9
V23105A5001A201
RLA10
V23105A5001A201
Coil+
Coil-
A-NO
A-Com
A-NC
B-NC
B-Com
B-NO
CHA_Amplified-
CHA_Amplified+
CHA_500+
CHA_500-
CHA_500F+
CHA_500F-
CHA_250+
CHA_250-
CHA_250F+
CHA_250F-
L5 22nH
L6 26nH
L7 22nH
L8 22nH
L9 26nH
L10 22nH
L11 44nH
L12 52nH
L13 44nH
L14 44nH
L15 52nH
L16 44nH
C132 5.1pF
C133 11pF
C134 11pF
C135 5.1pF
C136 5.1pF
C137 11pF
C138 11pF
C139 5.1pF
C141 10pF
C142 22pF
C143 22pF
C144 10pF
C145 10pF
C146 22pF
C147 22pF
C148 10pF
R102 4.99
R103 1.21k
R104 1.21k
R105 4.99
C140 0.01uF
CHA+
CHA-
CHA_FILTER_SEL_5V
R106 90.9
R107 90.9
D30 LTST-C191TBKT
D31 LTST-C191TBKT
Single Channel
Dual Channel
+3.75V
VOCM_VGA1
R108 0
C150 1uF
C152 0.01uF
C149 0.01uF
C151 0.1uF
VCM
U21
OPA376AIDCKR
250 MHz 7th order Chebyshev Type I Low-Pass Filter
CHB_Amplified+
CHB_Amplified-
L17 44nH
L18 52nH
L19 44nH
L20 44nH
L21 52nH
L22 44nH
C153 10pF
C154 22pF
C155 22pF
C156 10pF
C158 10pF
C159 22pF
C160 22pF
C161 10pF
R109 4.99
R110 1.21k
R111 1.21k
R112 4.99
C157 0.01uF
CHB+
CHB-
VOCM_VGA2
R113 0
C163 1uF
C165 0.01uF
C162 0.01uF
C164 0.1uF
U22
OPA376AIDCKR
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Analog Front End/Low Pass Filters/
File: LPF.sch
Title: Low Pass Filters
Size: A4
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 10/13

Figure 23: Low Pass Filters Schematics

5.1.2.8 Analog-Digital-Converter (ADC)

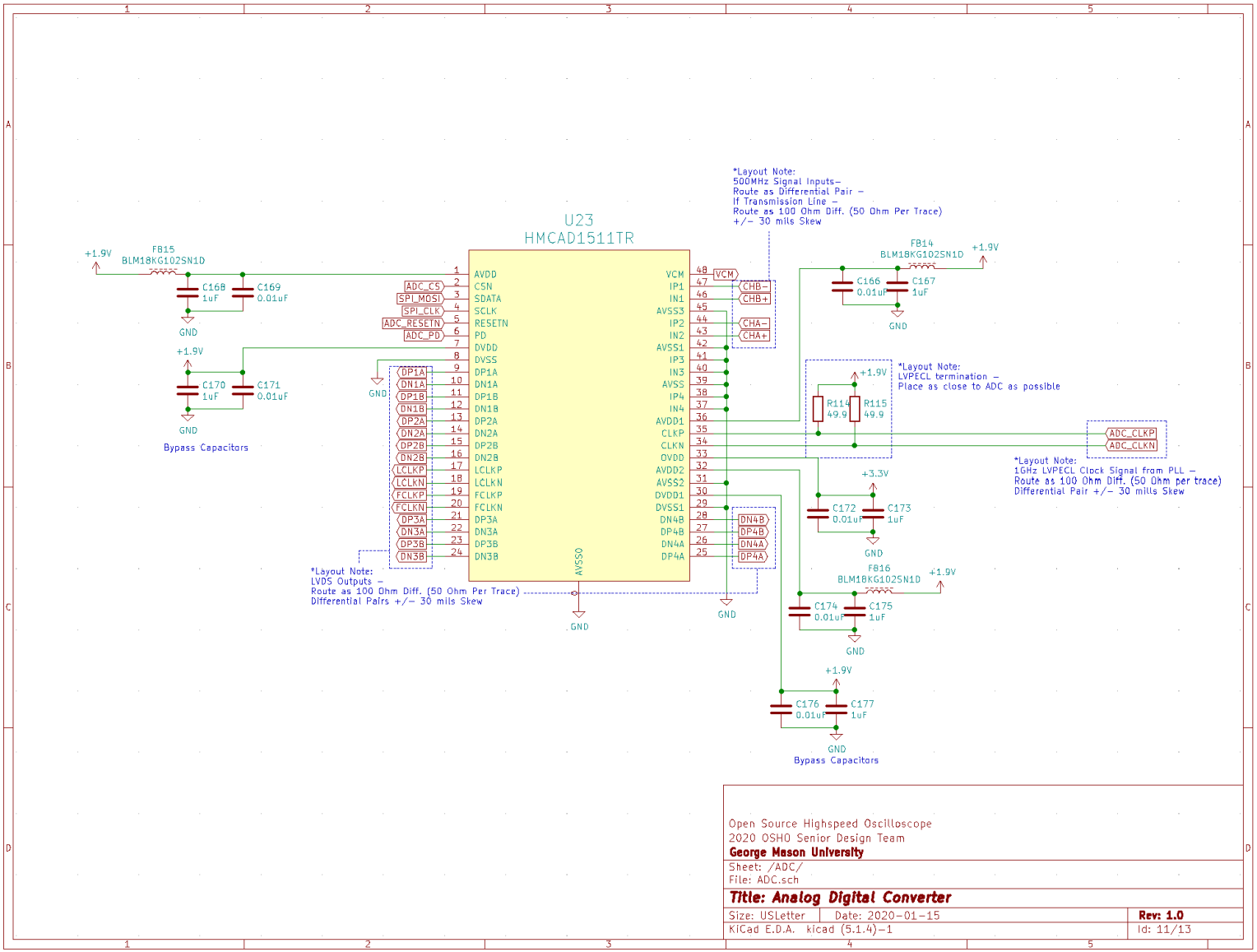

U23
HMCAD1511TR
*Layout Note:
500MHz Signal Inputs–
Route as Differential Pair –
If Transmission Line –
Route as 100 Ohm Diff. (50 Ohm Per Trace)
+/– 30 mils Skew
FB15
BLM18KG102SN1D
+1.9V
C168 1uF
C169 0.01uF
C170 1uF
C171 0.01uF
GND
Bypass Capacitors
ADC_CS
SPI_MOSI
SPI_CLK
ADC_RESETN
ADC_PD
AVDD
CSN
SDATA
SCLK
RESETN
PD
DVDD
DVSS
DP1A
DN1A
DP1B
DN1B
DP2A
DN2A
DP2B
DN2B
LCLKP
LCLKN
FCLKP
FCLKN
DP3A
DN3A
DP3B
DN3B
AVSS0
VCM
IP1
IN1
AVSS3
IP2
IN2
AVSS1
IP3
IN3
AVSS
IP4
IN4
AVDD1
CLKP
CLKN
OVDD
AVDD2
AVSS2
DVDD1
DVSS1
DN4B
DP4B
DN4A
DP4A
CHB–
CHB+
CHA–
CHA+
*Layout Note:
LVDS Outputs –
Route as 100 Ohm Diff. (50 Ohm Per Trace)
Differential Pairs +/– 30 mils Skew
FB14
BLM18KG102SN1D
C166 0.01uF
C167 1uF
R114 49.9
R115 49.9
*Layout Note:
LVPECL termination –
Place as close to ADC as possible
ADC_CLKP
ADC_CLKN
*Layout Note:
1GHz LVPECL Clock Signal from PLL –
Route as 100 Ohm Diff. (50 Ohm per trace)
Differential Pair +/– 30 mils Skew
+3.3V
C172 0.01uF
C173 1uF
FB16
BLM18KG102SN1D
C174 0.01uF
C175 1uF
C176 0.01uF
C177 1uF
Bypass Capacitors
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /ADC/
File: ADC.sch
Title: Analog Digital Converter
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 11/13

Figure 24: ADC Schematics

5.1.2.9 Sampling Clock Generation

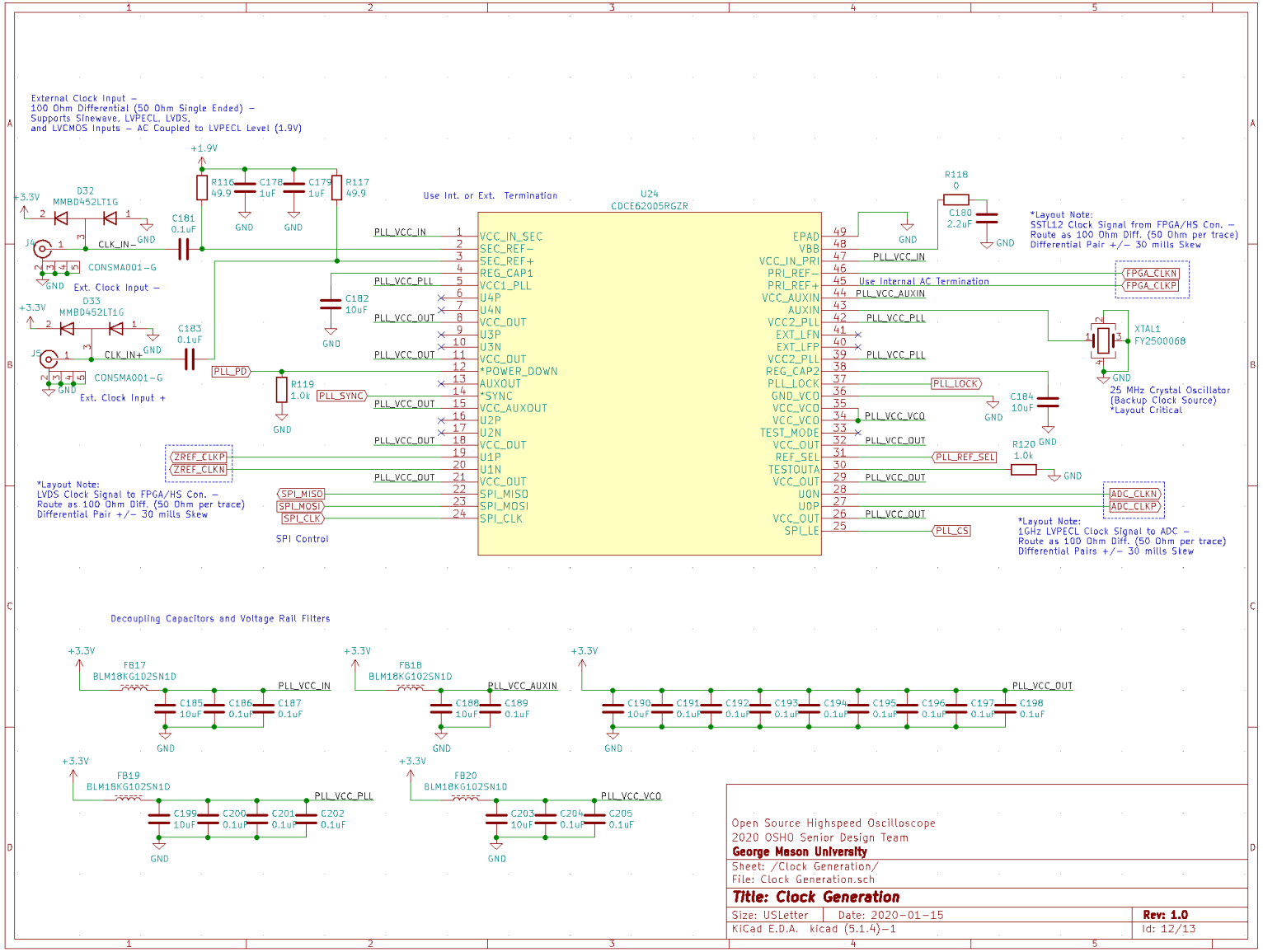

External Clock Input –
100 Ohm Differential (50 Ohm Single Ended) –
Supports Sinewave, LVPECL, LVDS,
and LVCMOS Inputs – AC Coupled to LVPECL Level (1.9V)
Use Int. or Ext. Termination
U24
CDCE62005RGZR
Ext. Clock Input –
Ext. Clock Input +
*Layout Note:
LVDS Clock Signal to FPGA/HS Con. –
Route as 100 Ohm Diff. (50 Ohm per trace)
Differential Pair +/– 30 mills Skew
SPI Control
*Layout Note:
SSTL12 Clock Signal from FPGA/HS Con. –
Route as 100 Ohm Diff. (50 Ohm per trace)
Differential Pair +/– 30 mills Skew
Use Internal AC Termination
25 MHz Crystal Oscillator
(Backup Clock Source)
*Layout Critical
*Layout Note:
1GHz LVPECL Clock Signal to ADC –
Route as 100 Ohm Diff. (50 Ohm per trace)
Differential Pairs +/– 30 mills Skew
Decoupling Capacitors and Voltage Rail Filters
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Clock Generation/
File: Clock Generation.sch
Title: Clock Generation
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 12/13

Figure 25: Sampling Clock Generation Schematics

5.1.2.10 Digital Connectors

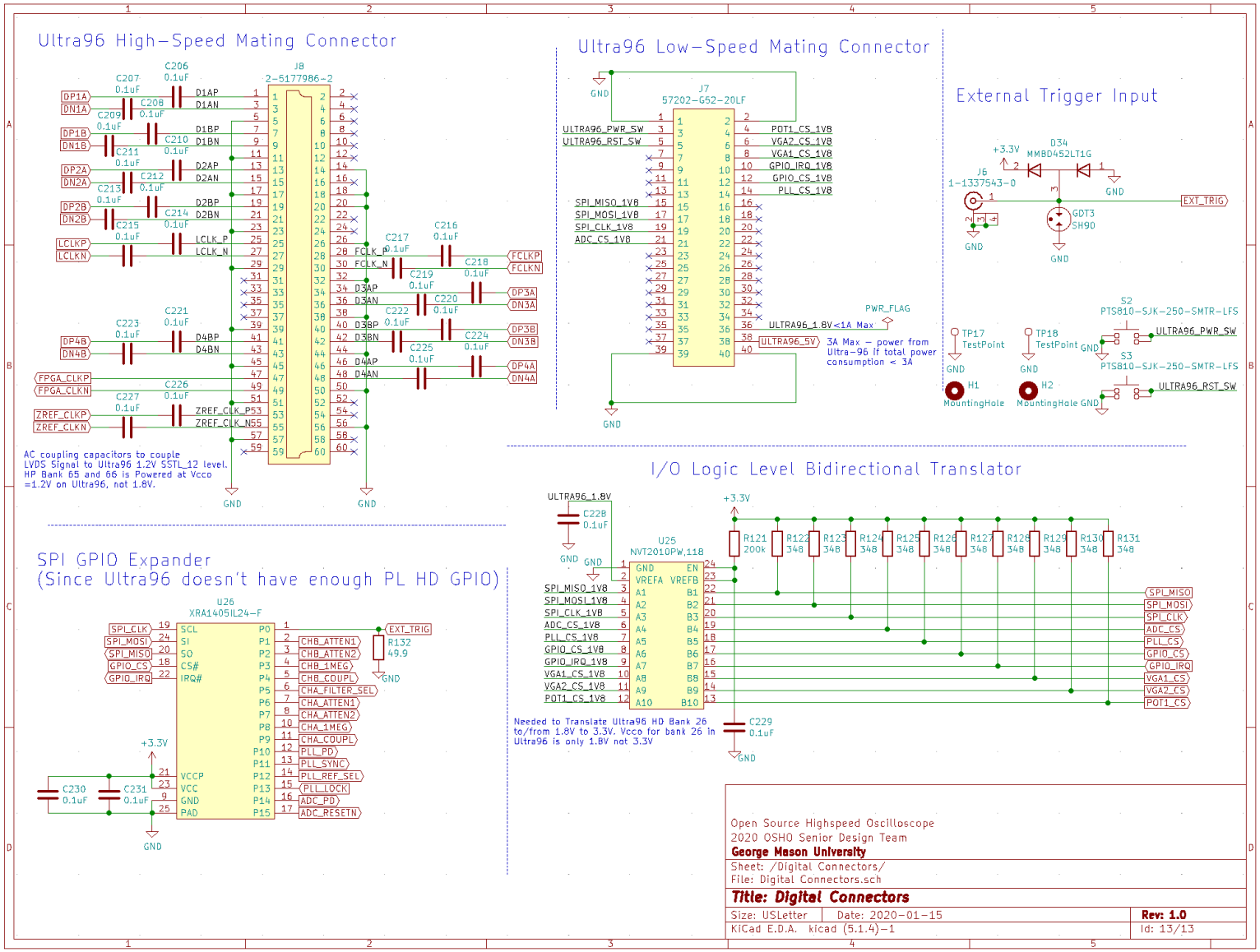

Ultra96 High-Speed Mating Connector
J8
2-5177986-2
AC coupling capacitors to couple LVDS Signal to Ultra96 1.2V SSTL_12 level. HP Bank 65 and 66 is Powered at Vcco =1.2V on Ultra96, not 1.8V.
Ultra96 Low-Speed Mating Connector
J7
57202-G52-20LF
PWR_FLAG
3A Max - power from Ultra-96 if total power consumption < 3A
External Trigger Input
J6
1-1337543-0
D34
MMBD452LT1G
GDT3
SH90
S2
PTS810-SJK-250-SMTR-LFS
S3
PTS810-SJK-250-SMTR-LFS
TP17
TestPoint
TP18
TestPoint
H1
MountingHole
H2
MountingHole
SPI GPIO Expander
(Since Ultra96 doesn't have enough PL HD GPIO)
U26
XRA1405IL24-F
I/O Logic Level Bidirectional Translator
U25
NVT2010PW,118
Needed to Translate Ultra96 HD Bank 26 to/from 1.8V to 3.3V. Vcco for bank 26 in Ultra96 is only 1.8V not 3.3V
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Digital Connectors/
File: Digital Connectors.sch
Title: Digital Connectors
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 13/13

Figure 26: Digital Connector Schematics

5.2 PCB Design

5.2.1 High Level Layout Approach

The printed circuit board layout was conducted in order to provide best electromagnetic, thermal, and efficient performance for the board. In order to accomplish this, considerations were taken in impedance matching, noise control, mixed signal design, heat dissipation, length matching, and physical constraints following PCB design best practices. The following sections provide a brief overview of these design principles and illustrate the final board design.

5.2.2 Controlled Impedance Design

Impedance controlled design refers to designing PCB traces with specific physical parameters such as trace width and dielectric values so that the trace has a specific characteristic impedance. As shown below in figure 27, a trace's characteristic impedance refers to the effective resistance and reactance per unit length. These values typically vary by depending on several factors including dielectric thickness, dielectric type, trace width, trace spacing, and even whether there is a solder mask on top of the trace. The values can be derived using electromagnetic equations, but are typically found using calculators as they are typically well established.

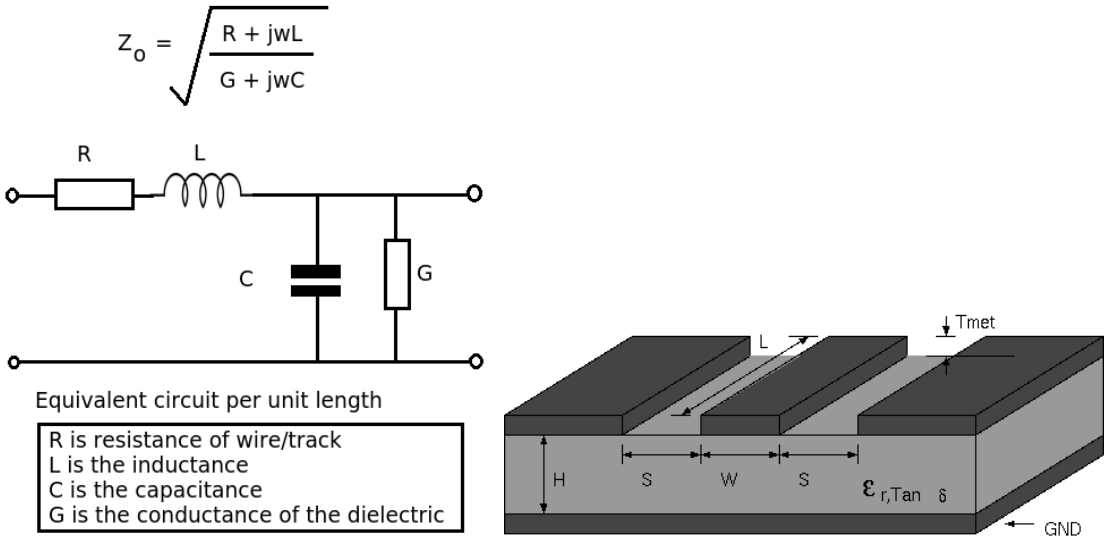


Figure 27: A PCB Coplanar Waveguide Trace and Equivalent Circuit

Controlled impedance design is critical when performing high speed and precision PCB layout as when any traces are longer than 1/10th of the minimum wavelength of the signals that the trace carries must be treated as a transmission line

with a characteristic impedance equal to the transmitter’s output impedance and the receiver's input impedance. Otherwise, if there is an impedance miss-match, signal power will be reflected back to the source transmitter causing a standing wave and a loss in signal integrity. Typically, in RF applications, traces are designed to a common characteristic impedance of 50 Ohms. In our PCB, high speed single-ended analog signals are designed to this impedance, and high speed differential signals (both analog and digital) are designed to an odd mode impedance of 100 Ohms. This was achieved by designing the widths and spacing of traces to match these values for chosen PCB stackup.

Originally these values were calculated using several different professional PCB calculators, however, it was discovered that these calculators do not account for the drop in characteristic impedance that via stitching and soldermasks create. Therefore an electromagnetic simulator was used to determine the appropriate width of the differential and single ended coplanar waveguides. A sample of these simulations can be seen below and were compared against the results of physical tests done by other electrical engineers. Using this method proved to be much more accurate in determining the characteristic impedance of the traces than just using PCB calculator tools.

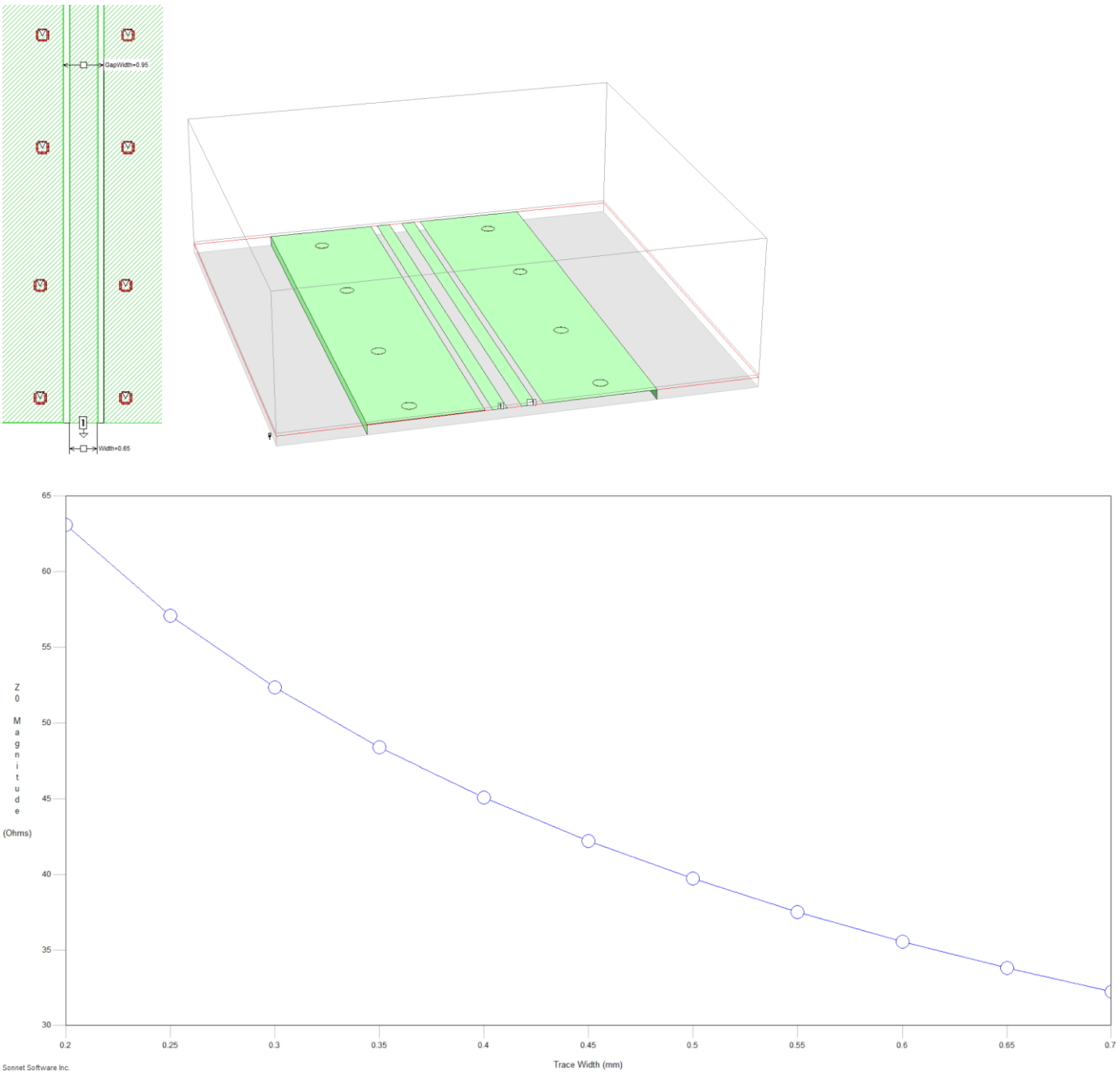


Figure 28: A PCB Single and Differential Coplanar Waveguide Trace Simulations for Selected Stackup (Single Ended - Top Left, Differential - Top Right, Simulation Result for Single Ended - Bottom)

5.2.3 Noise Control and Analog/Digital Separation

In mixed signal, precision circuit, and high speed PCB design, many considerations have to be taken into account in order to ensure that the circuit performs to specification. Otherwise electromagnetic interference (EMI) and other electromagnetic phenomena can have a great effect on the circuit performance. One main consideration is that the ground must be as low impedance as possible to create an accurate common ground voltage that is not affected by large return currents.This

typically means using a continuous ground plane. Hower, slits or separations may be used to protect large or noisy return currents from precision circuitry or to separate analog and digital planes in certain applications. An example of this can be seen below in figure 29 where the precision analog circuitry is protected from the large return current voltage drops in the ground plane using a slit.

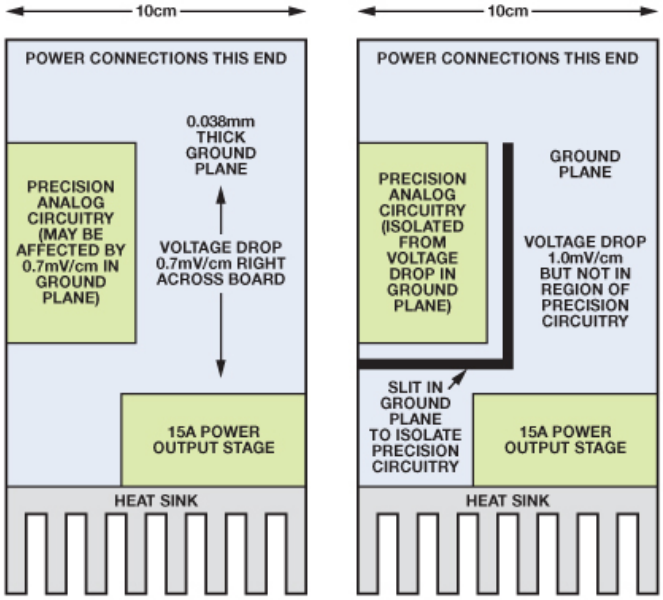


Figure 29: Example of Ground Plane Slit

Another important consideration is to ensure that return currents are taken into account. This includes properly separating the digital and analog components of a circuit, ensuring that traces do not go over ground reference plane discontinuities, and that return currents (both AC and DC) do not cause a ground loop like the one shown in the figure below. Ground loops make the circuit susceptible to noise as it acts like a large inductor where noise within the loop can influence the circuit.

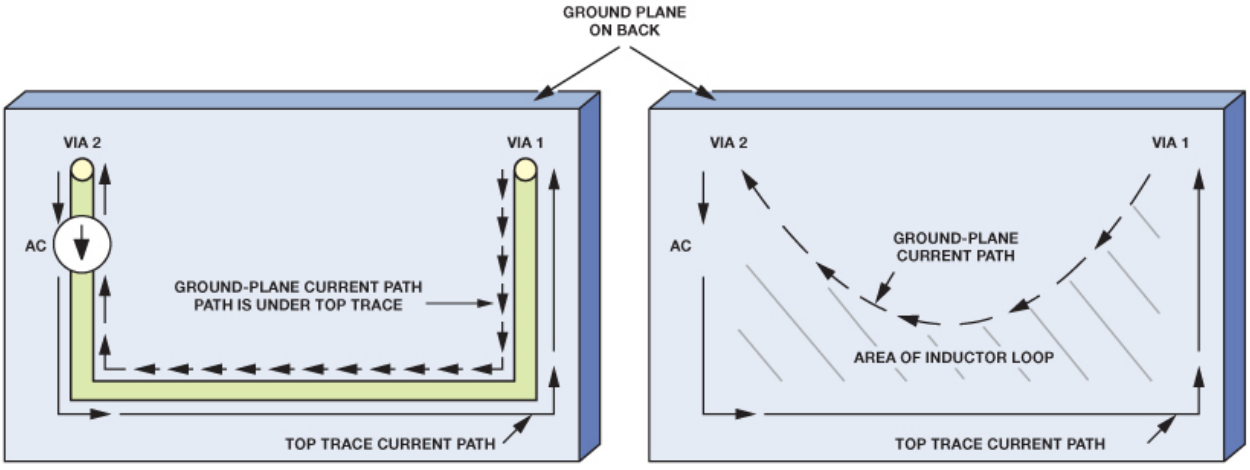


Figure 30: Example of Ground Loop

Another major consideration in the EMI aspect of mixed signal PCB design is proper separation between analog and digital circuits. This includes filtering between analog

and digital supplies with ferrite beads, the use of proper decoupling capacitors close to power pins, an seperation of digital and analog grounds (all though this can be avoided if components are placed appropriately in a star-ground formation). The figure below illustrates how a mixed signal component's power should be decoupled, and filtered to avoid digital noise from affecting the analog circuitry.

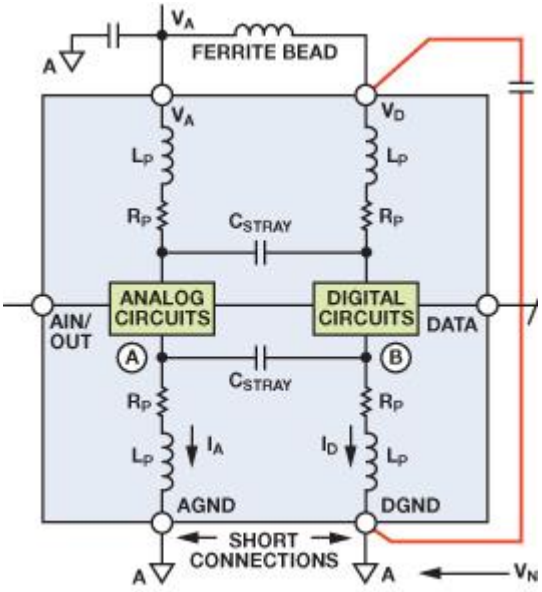


Figure 31: Analog and Digital Separation and Filtering for Mixed Signal Component

This list of EMI aspects is not meant to be a thorough overview of these concepts, but rather provide an overview of the types of EMI aspects that were taken into account when producing the final PCB layout. Whole textbooks can be written on this subject!

5.2.4 Differential Pairs and Length Matching

Another equally important aspect of high speed PCB design is trace length matching and differential pair tuning. This is because, as shown if figure ## below, any difference in length of high speed traces can cause a delay in the signal to arrive at its destination with respect to another signal. This can occur both within a differential pair (intra-pair) and across different pairs and races (inter-pair/trace). Ensuring that there is no skew is critical in the ADC output LVDS differential pairs because any deviation of the signal can cause misreads by the FPGA reception buffers, and is also important in the analog paths before the ADC because any artificial delay will cause each channel to be measured at different times. Additionally, if there is skew when the analog signals are differential, this can cause the analog signals to have errors at the receiver and be more susceptible to noise.

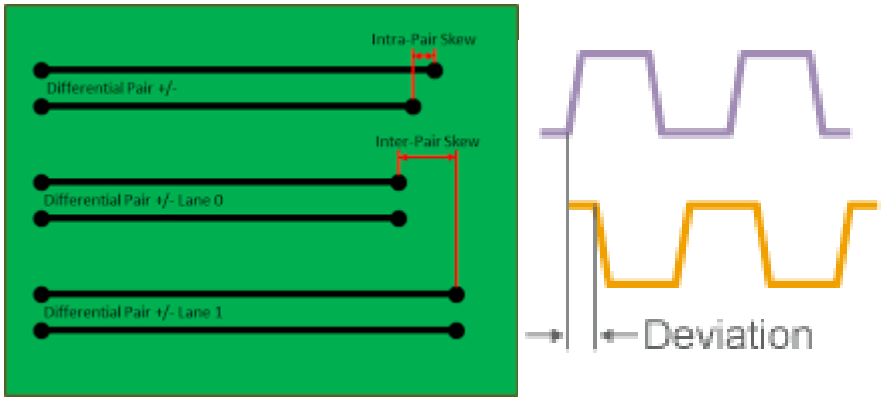


Figure 32: Intra-Pair and Inter-Pair/Trace Length Skew and example of resulting signal delay

To accommodate differences in pair lengths, differential and critical single ended traces were routed with serpentines, and inter-pair tuning adjustment following length matching best practices. An example of this on our board is shown below. All pairs that were length matched were matched to a skew of less than 0.05 mm.

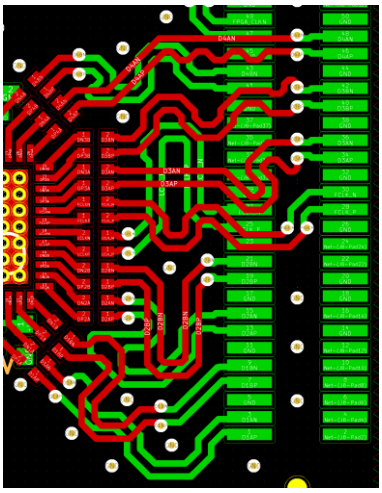


Figure 33: Differential Pair Length and Skew Tuning Example on ADC LVDS Outputs

5.2.5 Heat Dissipation

Another design consideration that was taken into account while performing routing and layout was heat dissipation. Power hungry chips such as the ADC and PLL which use a lot of power generate a lot of heat. This heat must be properly dissipated in order for the heat not to cause damage to the components. There are several methods of doing this but some of the methods that were employed in this design were using thermal pads attached to ground, using thermal vias attached to ground (as shown below in Figure 36), and using exposed thermal copper from which can be used to radiate thermal energy or even attach an additional heat sink to if needed (also shown below in Figure36).

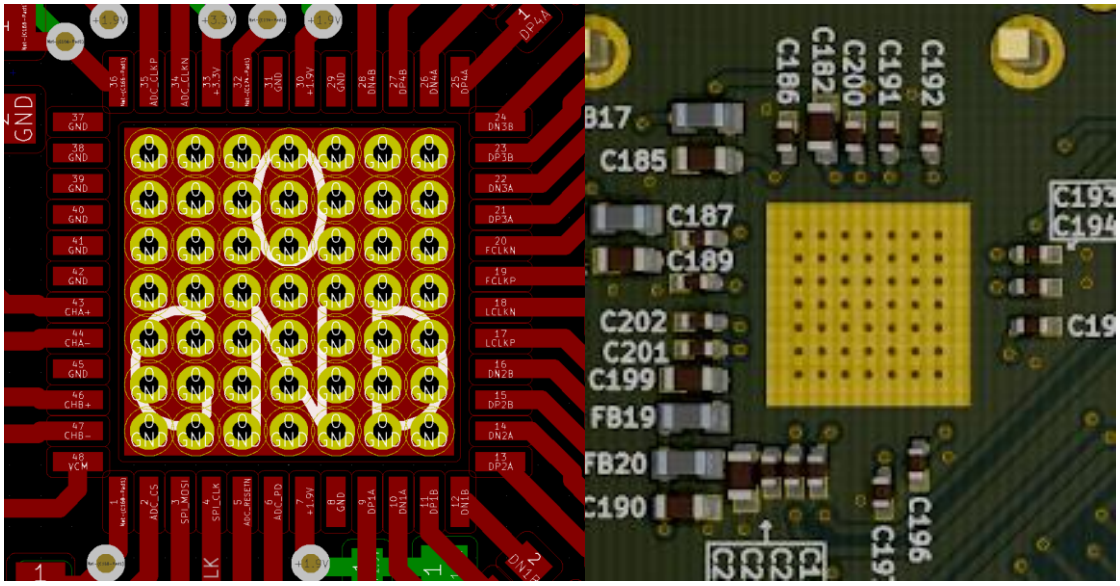
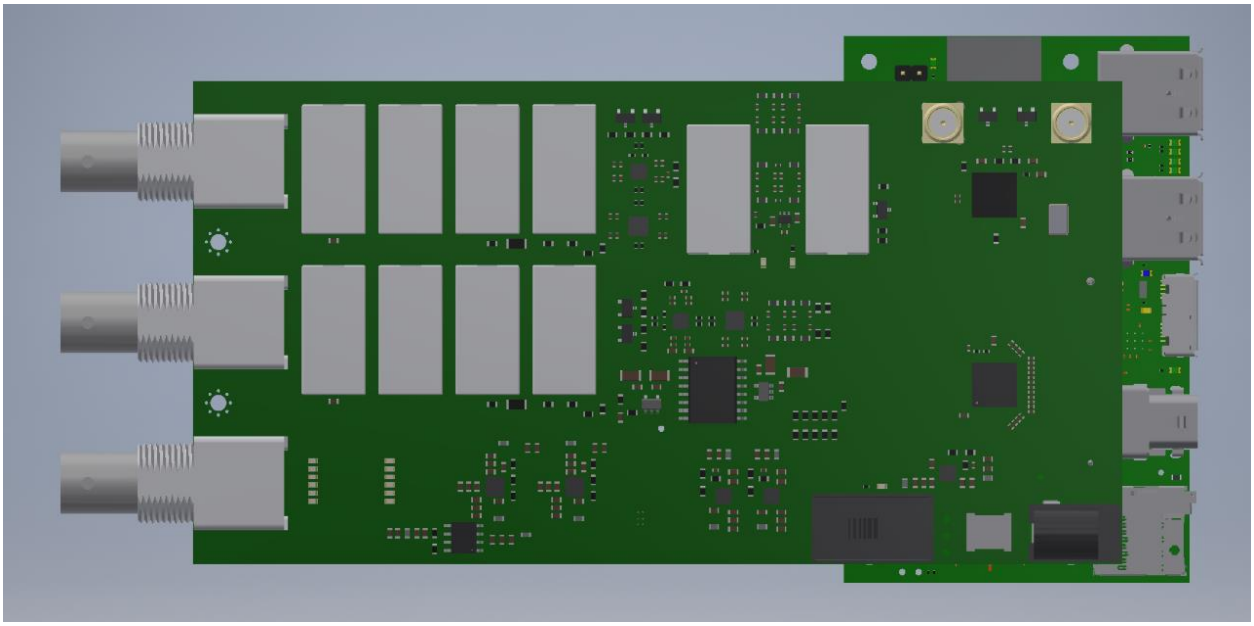


Figure 34: Thermal Vias and Exposed Thermal Copper Pad

5.2.6 Ultra 96 Design Constraints and Physical Layout Limitations

The last major design consideration that went into the board's design was the physical constraints of the board. This included location of Ultra96 connectors, location of Ultra96 switches and tall components, physical board size, digital noise consideration. In figure 37 below, shows the physical constraints of the ultra96, including connector position, size and location of components may cause clearance issues with our board. In figure 37 below, shows our board model mated with a model of the Ultra96 board to ensure that there were no clearance issues with our layout.



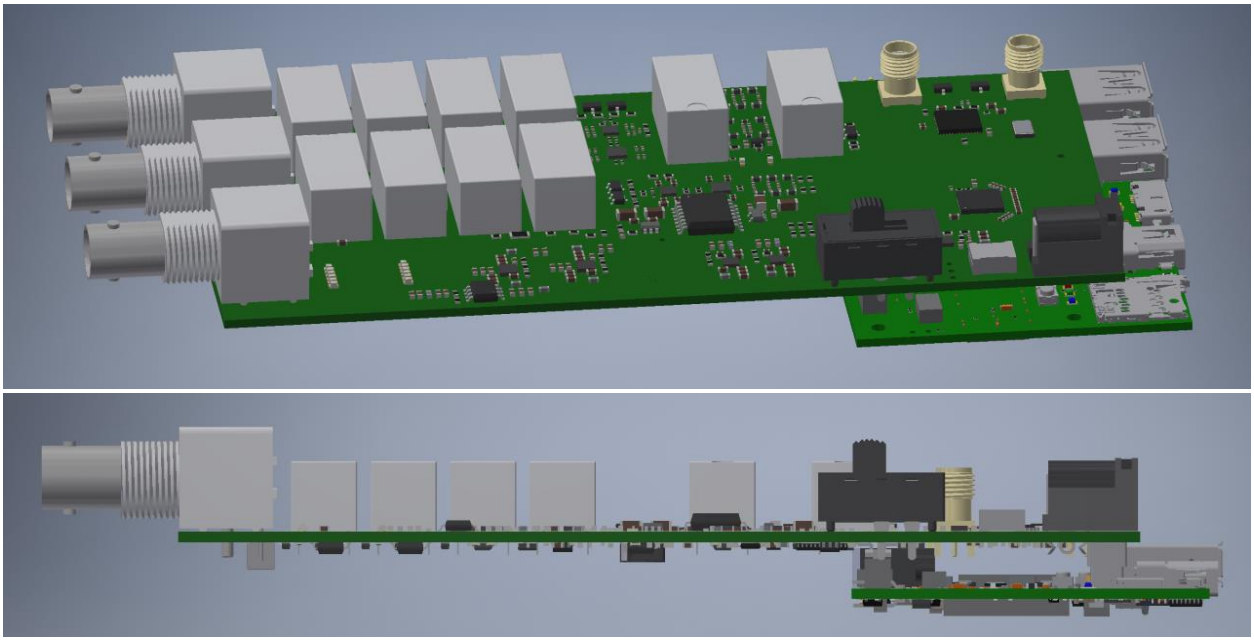


Figure 35: OSHO Board - Ultra96 Development Board Mating

5.2.7 Select PCB Layers

Below is a listing of select views that illustrate the final PCB design. They provide a clearcut representation of the design layers and board appearance. The final physical board design was 145m (5.7in) by 75mm (2.9in), and was designed on a standard 4 layer 0.062” FR4-Stackup. The top layer consisted mostly of sensitive analog signals on the left and noisy digital signals on the right. The upper-inner layer consisted of a continuous ground plane. The bottom layer consisted of several power planes for each power domain (3.3V, 5V, 1.9V, -1.25V, 3.75V, etc.). Finally, the bottom layer consists of mostly digital signals. Most digital signals that extend into the analog portion of the PCB are only active/noisy when not taking measurements.

5.2.7.1 Top Silk Screen View

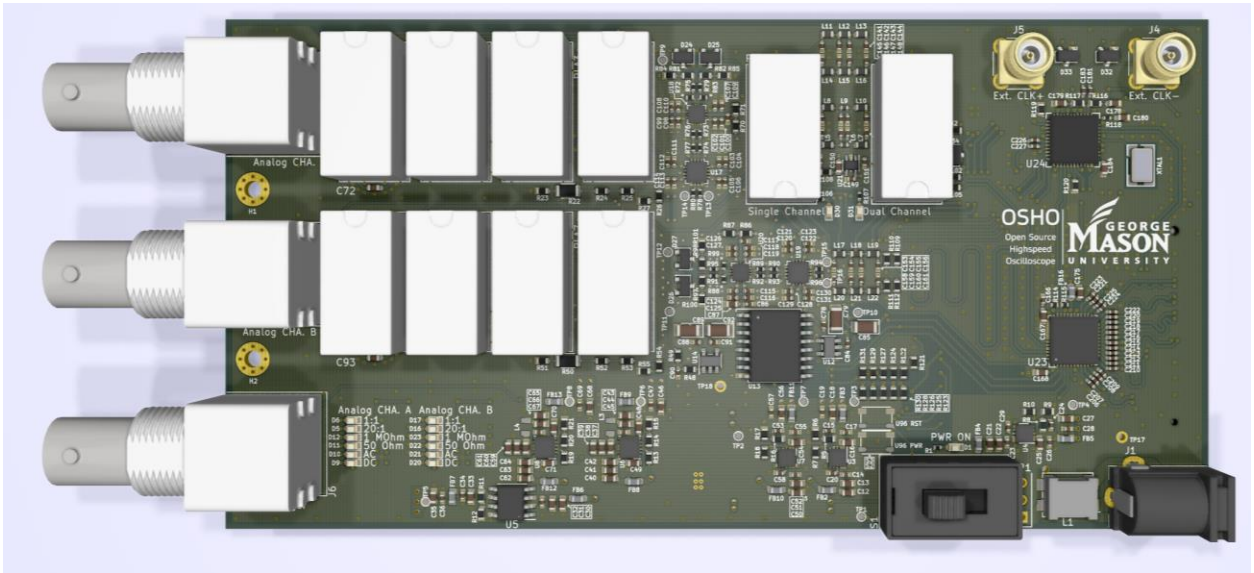


Figure 36: OSHO Board - Top Silk Screen View

5.2.7.2 Top Copper Layer

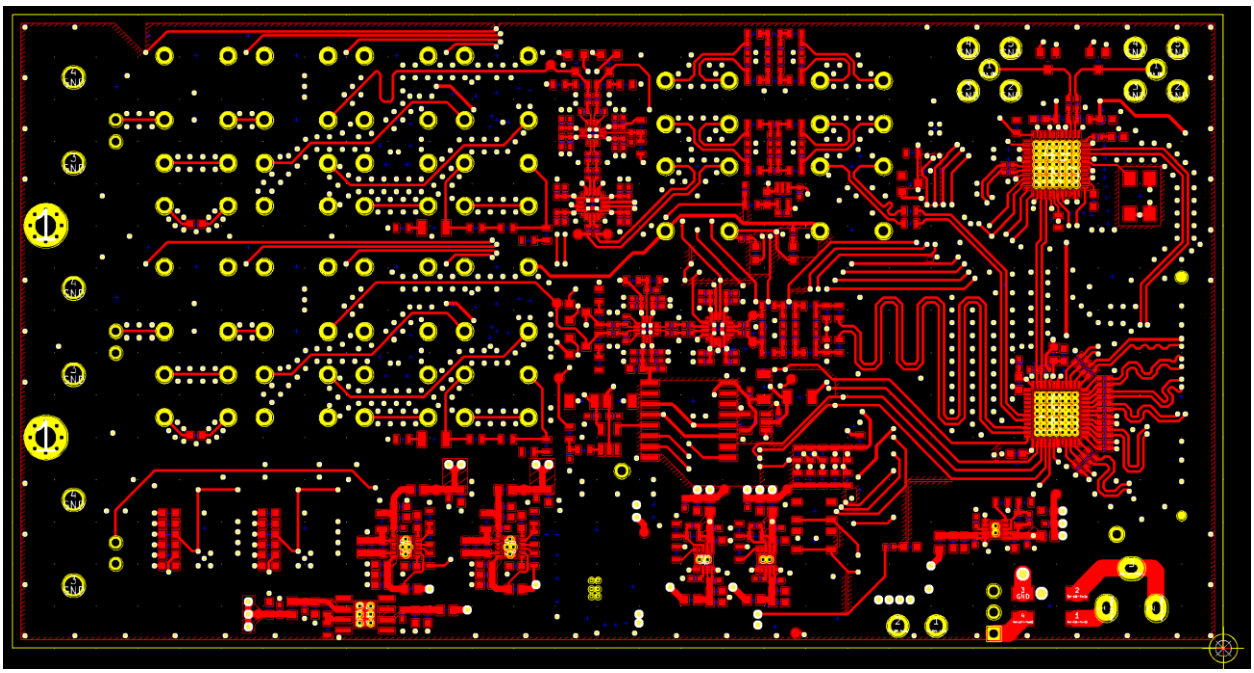


Figure 37: OSHO Board - Top Copper Layer

5.2.7.3 Top Inner Copper Layer (Ground Plane)

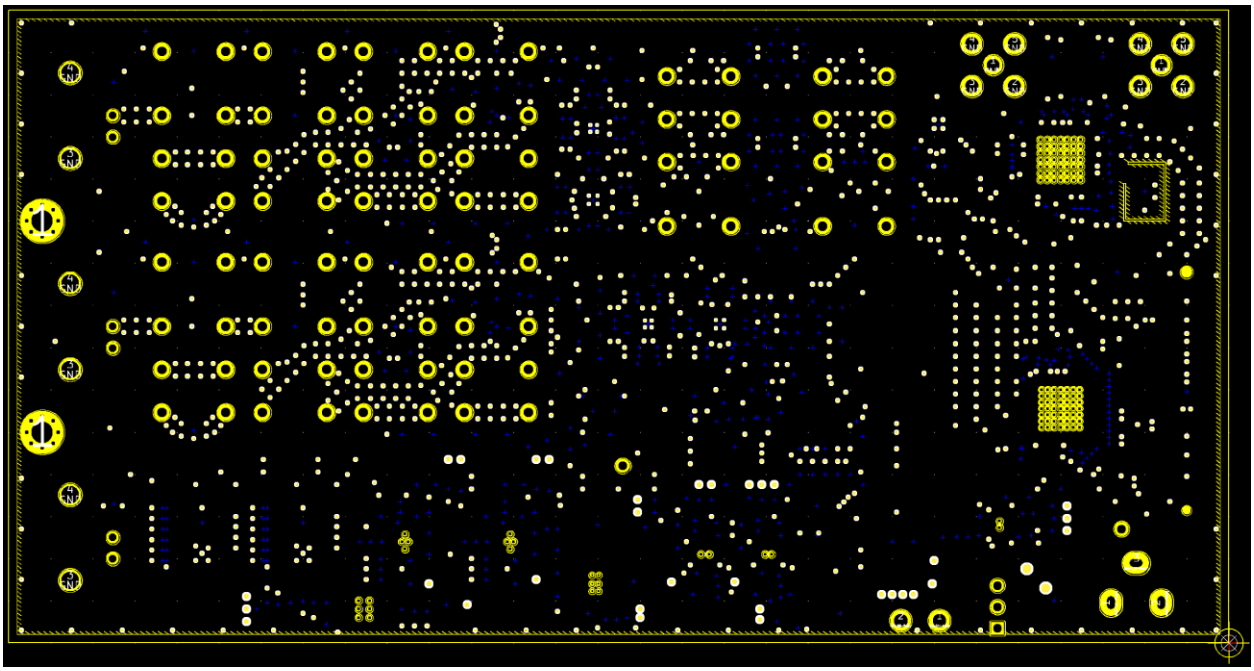


Figure 28: OSHO Board - Top Inner Copper Layer (Ground Plane)

5.2.7.4 Bottom Inner Copper Layer (Power Planes)

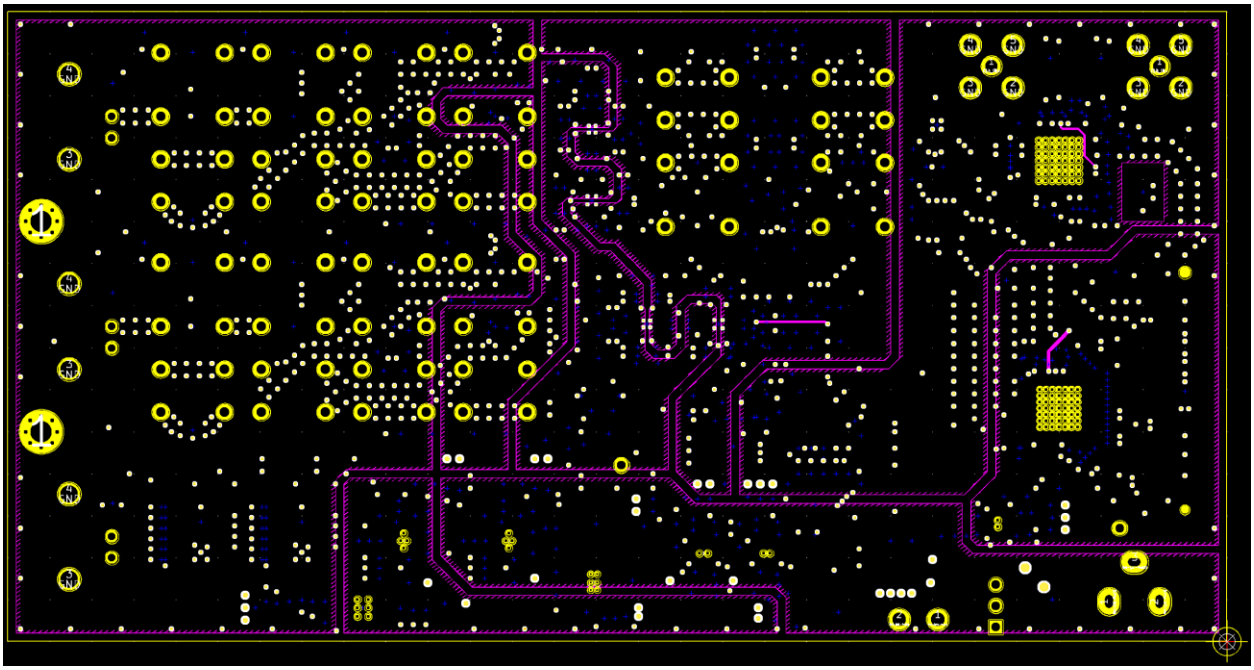


Figure 39: OSHO Board - Bottom Inner Copper Layer (Power Planes)

5.2.7.5 Bottom Copper Layer

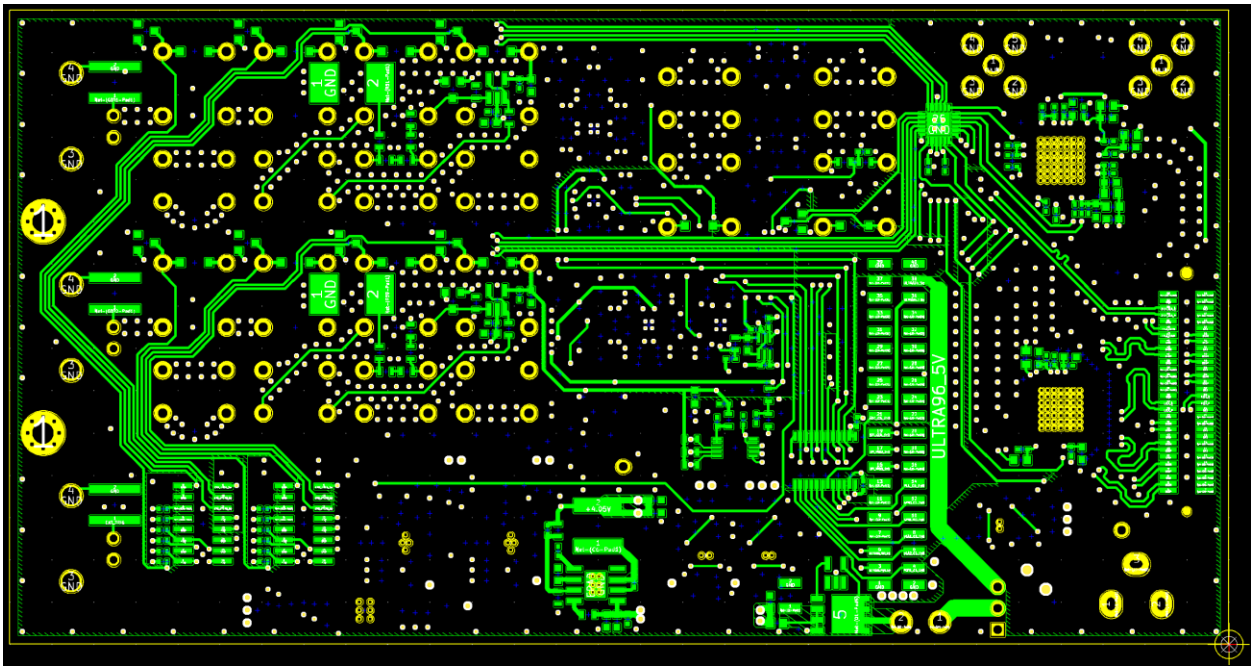


Figure 40: OSHO Board - Bottom Copper Layer

5.2.7.6 Bottom Silk Screen View (Flipped)

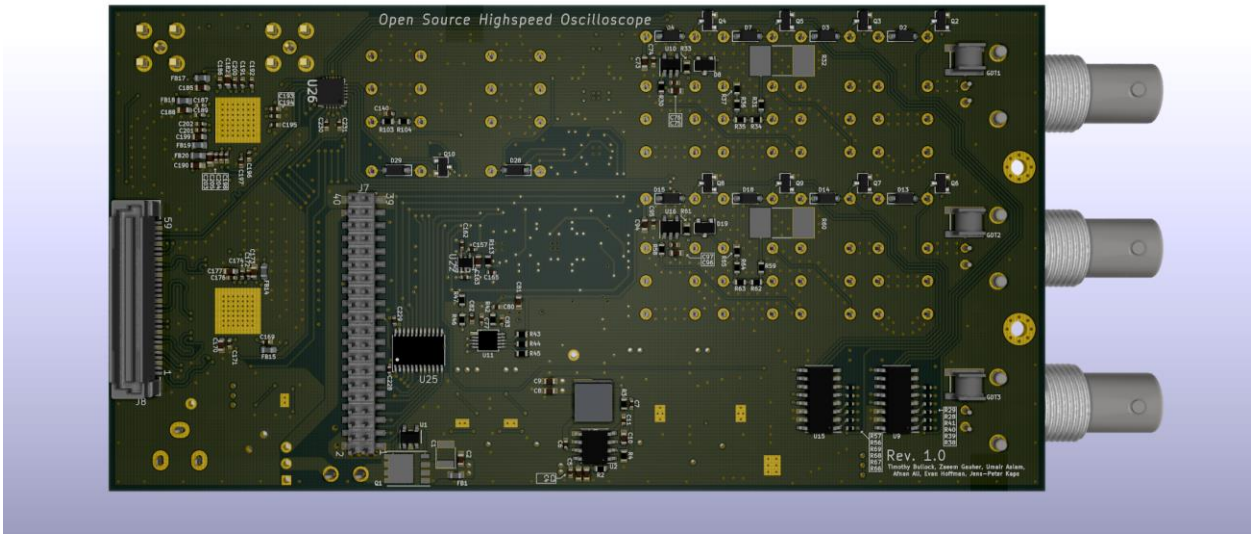


Figure 41: OSHO Board - Bottom Silk Screen View (Flipped)

5.3 FPGA Datapath Design

5.3.1 Datapath Overview

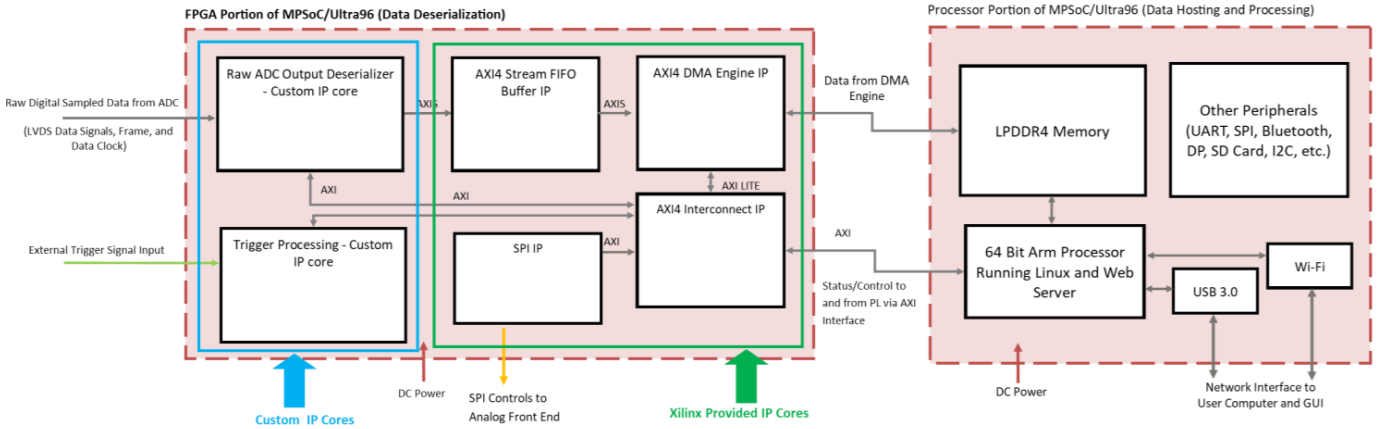


Figure 42: FPGA Datapath and the Processing System in Zynq Architecture

As shown in figure 42, the FPGA receives low-voltage differential signals (LVDS) from the ADC. These include the serial data bits, the frame clock and the bit clock. The frame clock (FClk) is a digitized and phase-shifted version of the ADC's sample clock while the high-speed bit clock (DCLK) is a 90° phase-shifted signal to the data. Data is valid at both edges of the bit clock leading to a double data-rate (DDR) interface. The low-voltage differential signals are buffered and then deserialized using the custom AXI IP core. This data is then passed through a FIFO (first-in, first-out) buffer for clock domain crossing from the ADC's sampling clock frequency to the global FPGA clock domain. The FIFO IP core uses the AXI stream protocol which does not need an address channel and is always used to write data in one direction. Therefore, the AXI Direct Memory Access (DMA) core is utilized for high-bandwidth direct memory access

between an AXI4-Stream target peripheral and the memory on the PS side. This data stored in memory is then accessed through the processor and transmitted to a client for plotting. Furthermore, a SPI IP core is used to send SPI signals to configure the ADC and the AXI Interconnect IP core is used to connect all the memory mapped AXI IP cores.

5.3.2 The Deserializer IP

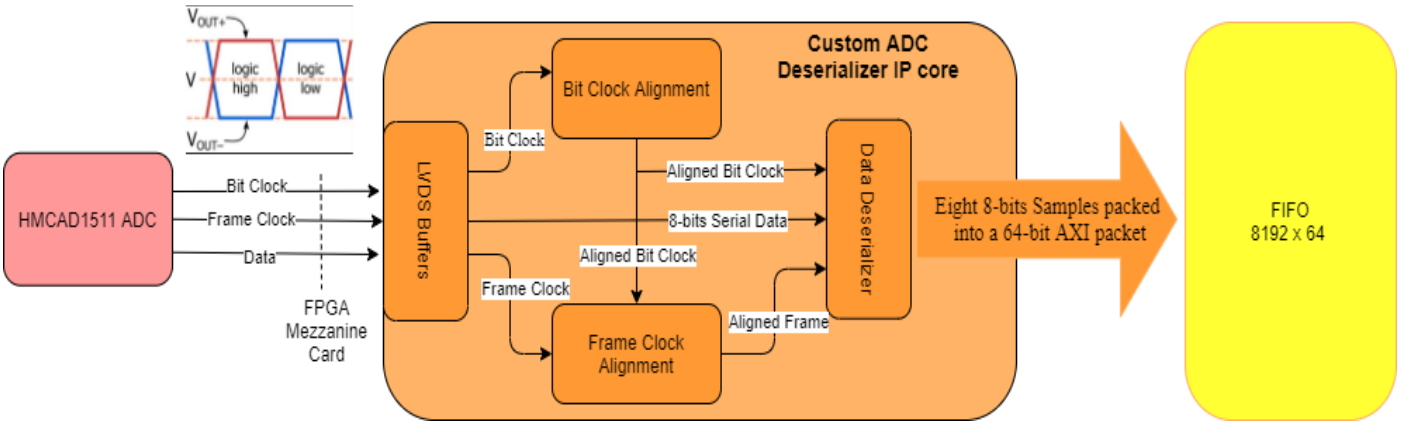


Figure 43: Modules within the custom ADC Deserializer IP Core

As shown in figure 43, the LVDS signals from the ADC are buffered into the FPGA. However, since the clocks and data pass through different routing resources, they lose their alignment that is required to correctly deserialize the data. Therefore, both the clocks need to be realigned. This is done by asserting the “re-align” signal to the FPGA. Once they have been aligned, the clocks are then used to deserialize the data. After deserialization, 8 samples, each with a resolution of 8-bits, are packed into a 64-bit AXI packet. This packet is then sent to the FIFO buffer for clock domain crossing as mentioned in the previous section.

5.3.3 Bit Clock Alignment

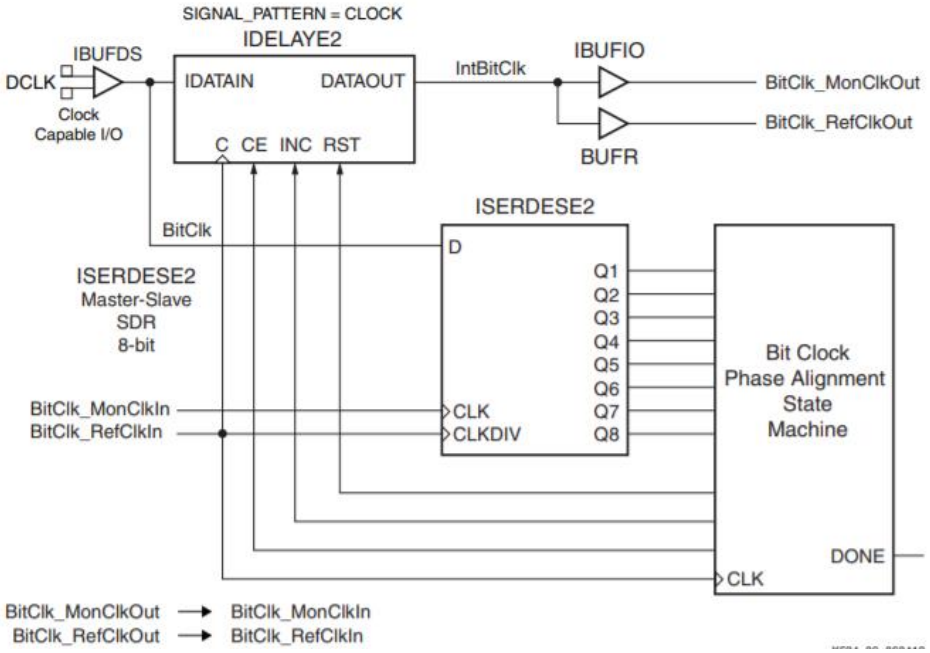


Figure 44: Bit Clock Alignment Setup

The bit clock (DCLK) from the ADC is routed through an IDELAYE2 used in variable mode to the input of a BUFIO and BUFR buffers. It also registers itself in the ISERDESE2 using a delayed version of itself as a clock. This technique is used to determine the position of the rising and falling edges of the bit clock. The Bit Clock Phase Alignment state machine monitors the ISERDESE2 outputs and the deserialized and parallel captured clock bits.

There are three possible ISERDESE2 output cases:

- The output data is random and changes on every clock cycle
- The output is all ‘1’s
- The output is all ‘0’s

The output data is random when the internal bit clock and the external bit clock are already phase-aligned and the random nature is caused by the clock jitter. In the other two cases, the IDELAYE2 delay amount is varied so that the output from the ISERDESE2 module becomes random and the bit clock is aligned to the original clock.

In addition, the ISERDESE2 module provided by Xilinx supports dual data rate- (DDR) mode signals and can deserialize 8-bit words. The module also includes a built-in bitslip operation that allows for the input data stream to be reordered. This operation can be used to train the ISERDESE2 module to lock onto an expected / training pattern output by the ADC as discussed in the next section.

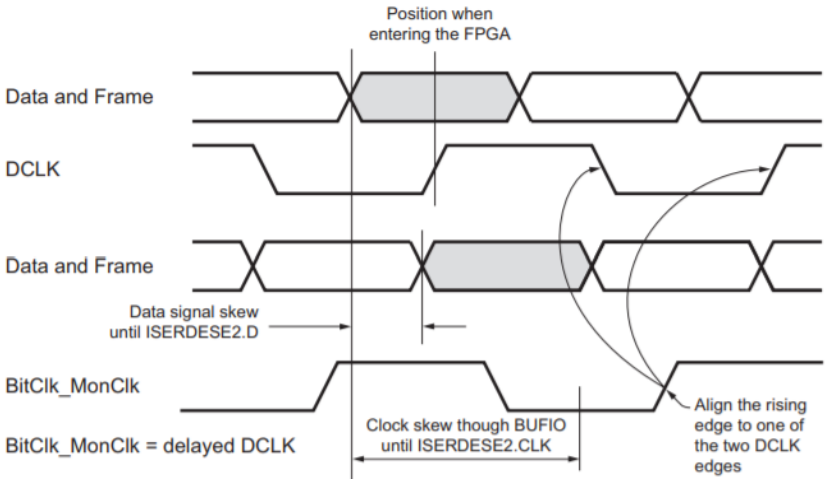


Figure 45: Clock Skew through the Buffers

5.3.4 Frame Clock Alignment

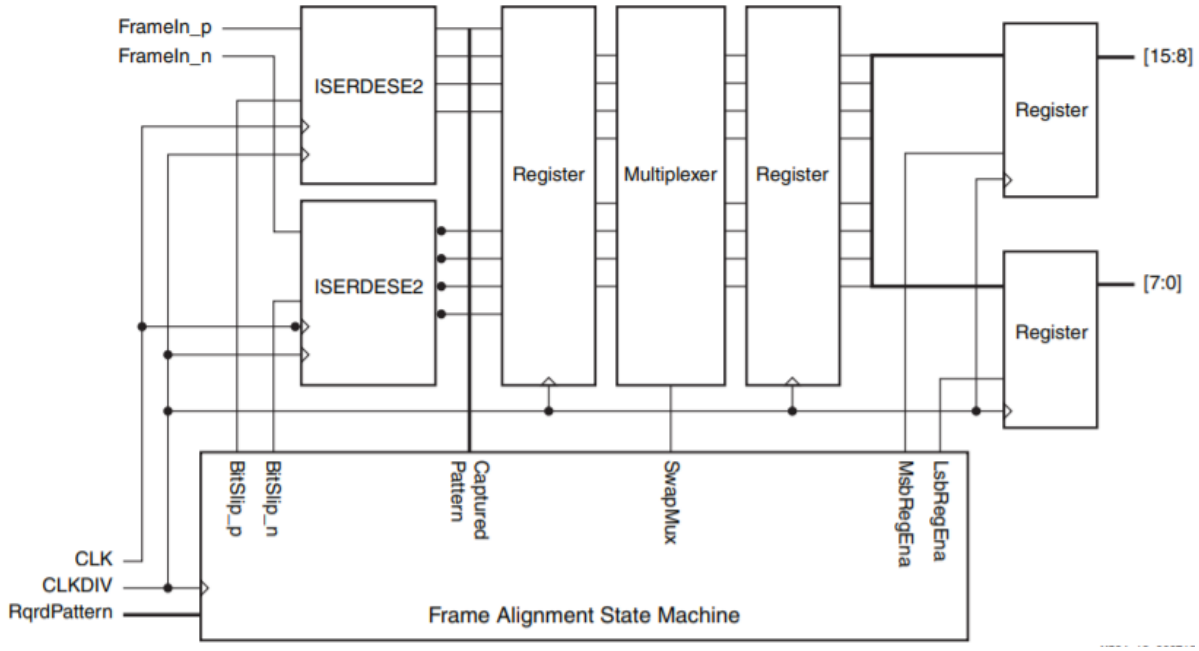


Figure 46: Frame Clock Alignment Block Diagram

After the bit clock (DCLK) has been properly aligned, the frame clock pattern discovery is begun. The LVDS frame clock from the ADC is a digitized version of the sampling clock that is phase aligned with the data. As shown in figure 46, the ISERDESE2 outputs are compared to a fixed value representing the expected frame clock pattern, which is 0xF0 for an 8-bit ADC. If the outputs of the ISERDESE2 do not match the expected value, a bitslip operation is carried out on the frame and data signals. When this output is finally equal to the programmed pattern, the bitslip operation is stopped and the data and frame clock signals within the FPGA are considered valid. Next, the received data is aligned because it is shifted with the frame

signal. Lastly, the bit clock and frame clock signals are used to capture and deserialize the data bits.

5.3.5 FPGA LVDS Data Inputs

The data from the ADC and the clock signals are read as LVDS inputs with internal termination in the Zynq SoC's I/O banks. Before being sent to the custom deserializer IP core, the differential buffer primitives are used to support the LVDS_25 I/O standard and the internal termination of the LVDS signals.

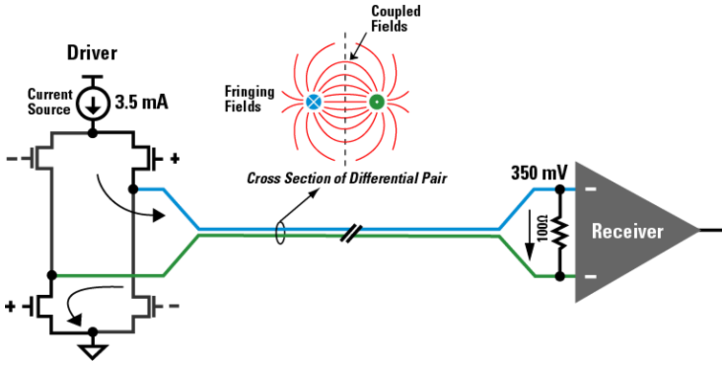


Figure 47: Basic LVDS circuit operation

Low-voltage differential signaling (LVDS) is a technical standard that specifies electrical characteristics of a differential, serial communication protocol. LVDS operates at low power and can run at very high speeds using inexpensive twisted-pair copper cables. Since, the current flows back to the driver in a loop, LVDS results in lower radiated emission (EMI) and rejects common-mode noise.

5.3.6 Processor

The Xilinx Zynq processing system is used as a standalone, low-level processor in the logic design to handle the cache, interrupts, exceptions and other features such as external I/Os and hardware peripherals. The Zynq processing system IP implemented in the Vivado datapath contains all the processor's configuration data such as clocking resources and the I/O mapping.

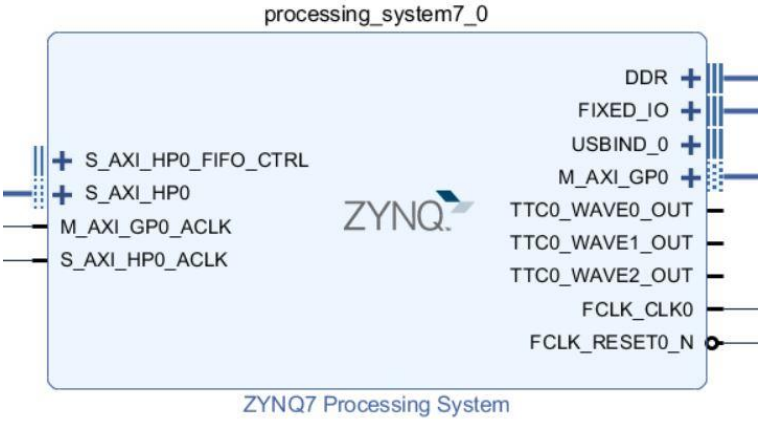


Figure 48: Block diagram of the ZYNQ7 processing system implemented in Vivado

5.4 Software Design

5.4.1 Software Overview

Waveforms live is a data visualization tool that works with digilent products like the OpenScope and the OpenLogger. This product allows for data visualization on any browser including phones. This progressive web application is written in a mixture of HTML, CSS, and Typescript. It also leverages the ionic framework and the cordova framework. Ionic is a software development kit that allows for cross platform usage of the application. Cordova is another framework that is also used for cross platform usage but specifically for mobile users.

Waveforms live was the base for our data visualization tool. With an already functional status that meets our design requirement of being able to perform high frequency analysis we figured it would be a good place to start as it would save more time than starting from scratch. Although the premise of saving time by using an already existing product as our base was valid we neglected to account for the time that it would take to become knowledgeable with the already existing code base. We also underestimated the amount of time it would take us to iteratively add to waveforms live. Not knowing what a large chunk of the source code did and how it functioned as well as not having a person who directly worked on this project before us led to a lot of lost time reading code/documentation and not fully understanding it. This also led to a lot of trial and error without much resolution.

As a result of lost time, the shift to isolationism, and the end of semester time crunch our final software for visualization was a python script that utilized matplotlib to take data and visualize it on a plot. This script took data samples from the hardware, translated them to representable values and then plotted them. We were also able to develop separate scripts that allow for zooming in and zooming out of the graph as well as showing the data values with a cursor. These separate scripts were tested in isolation but not able to be successfully integrated into one unified script. We also were able to modify the waveforms live GUI to have an additional button that began a websocket connection with a web server hosted by the Ultra 96. Once the web socket was connected we were able to transmit data back and forth. Unfortunately we were not able to fully control the source of the data nor start plotting the data. The data being pushed was from a counter function that incremented a data value by 1 and sent it to the connection.

We understand that the current state of the GUI is intermediate and there are next steps to be taken by whatever team continues this project. We think integrating the separate scripts into 1 is the next immediate step in the development of the GUI.

5.4.2 Software Models

Our original model (figure shown below) was made with the assumption of working with waveforms live. The goal was for us to have the GUI contained on the Ultra96 and have the user connect to it via the computer. Once connected via ssh and

ethernet cable the user would go to the localhost on their web browser and that would redirect them to the modified waveforms live instance running on the Ultra96. The user would be able to interact with the web application to plot and modify the capture settings for their data.

The new model for our modified visualization tool (shown below) is significantly simpler. This is because we shifted from using waveforms live with a websocket to running a python script that will run on the processor side of the Ultra96 which is also being used to access the data from the PCB via direct memory access.This data is taken from the DMA and then plotted. Buttons being clicked on the GUI result in visualization changes..

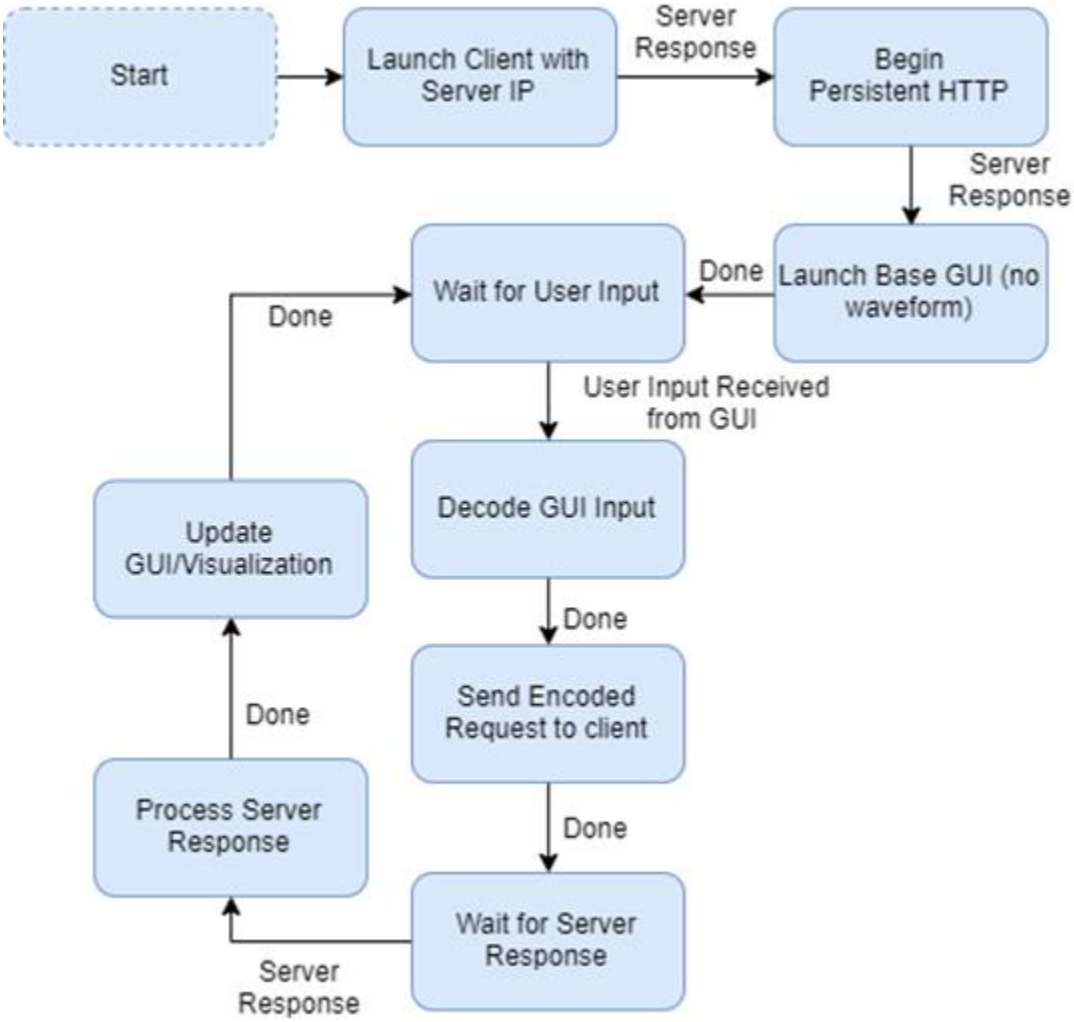


Figure 49: High Level GUI State Diagram Waveforms Live

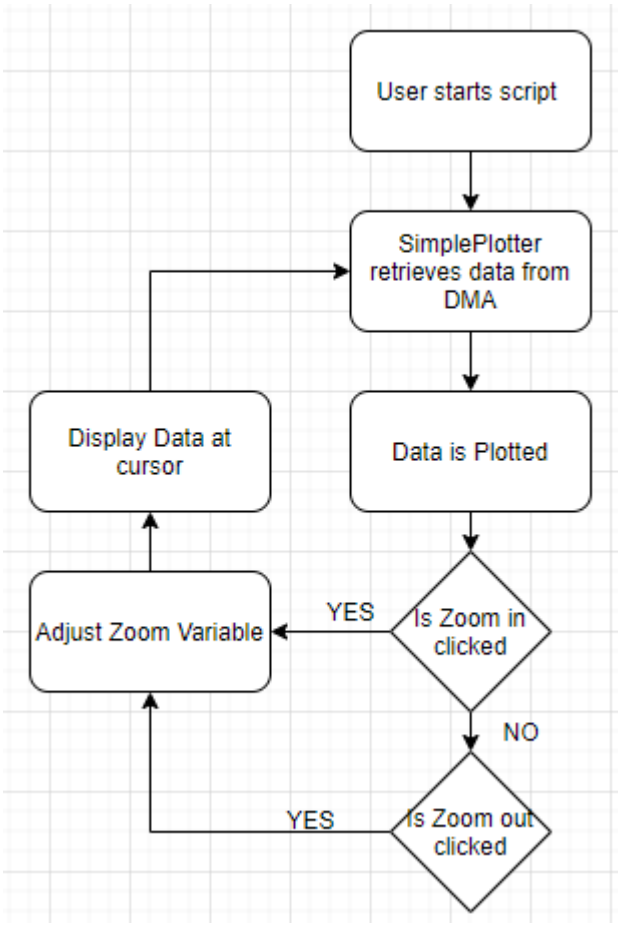


Figure 50: High Level GUI State Diagram SimplePlotter

.

5.4.3 Waveforms Live Design

As stated in previous sections the team ended up having 2 different GUIs partially implemented. The Waveforms Live GUI was designed to be built on top of the existing GUI that Digilent made. We added a websocket server and websocket client that were utilized to transfer data from the DMA to the GUI. We also added a specific button to the GUI to allow the user to select the Ultra96 as the hardware being used. We also added tooltips that provide additional information when the user hovers over the buttons we added.

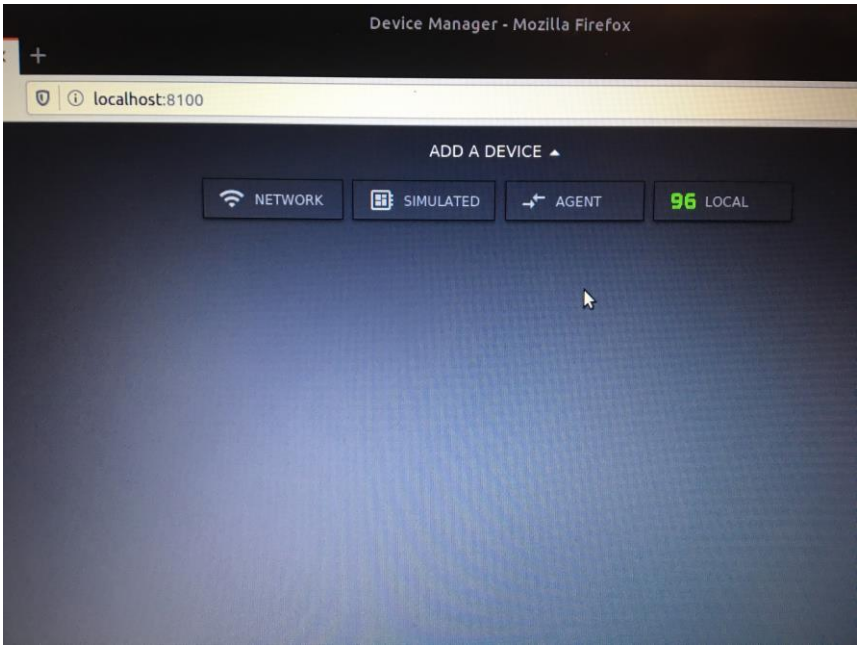


Figure 51: Custom Button Added for Ultra 96 Usage

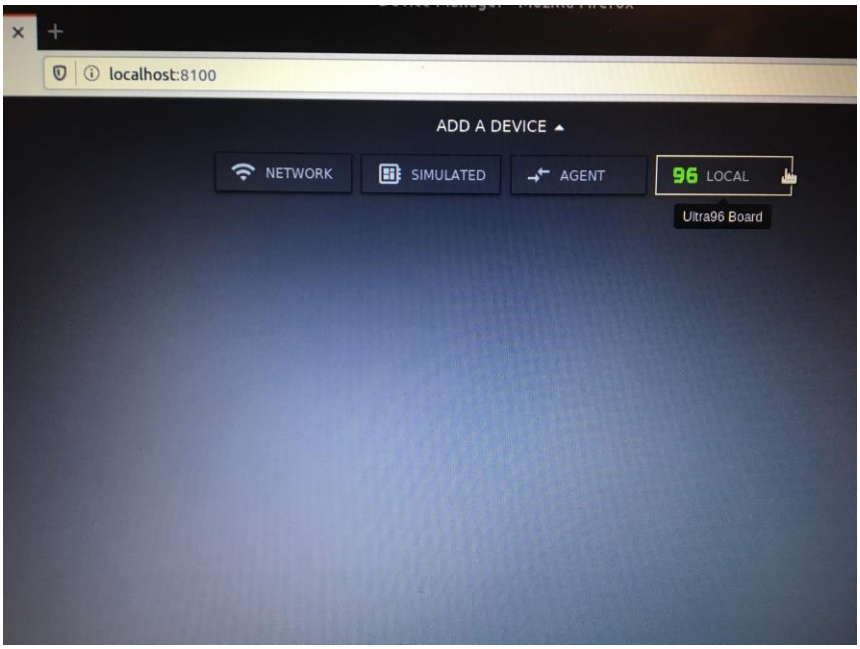


Figure 52: Tooltips added for Ultra96 button

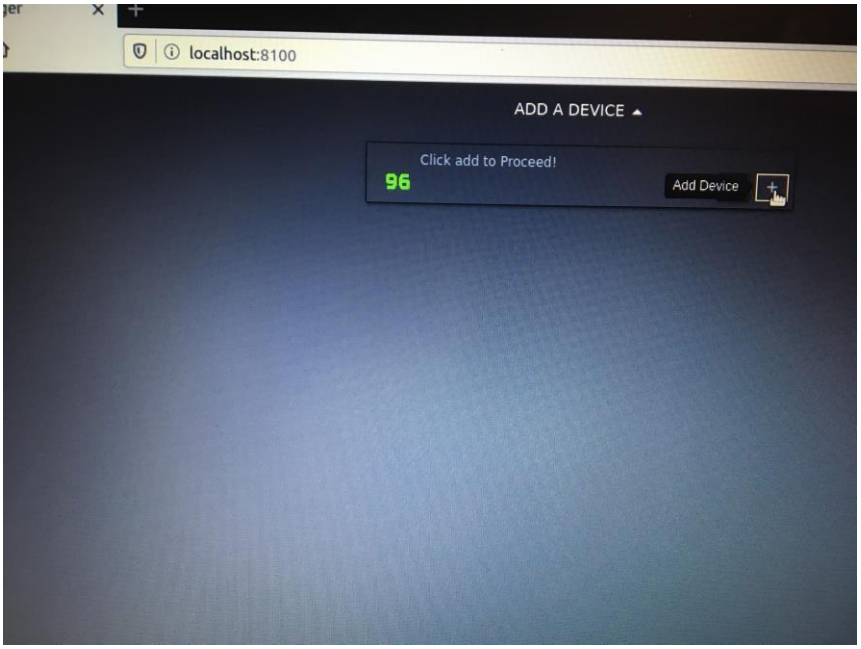


Figure 53: Confirmation screen for selecting Ultra 96

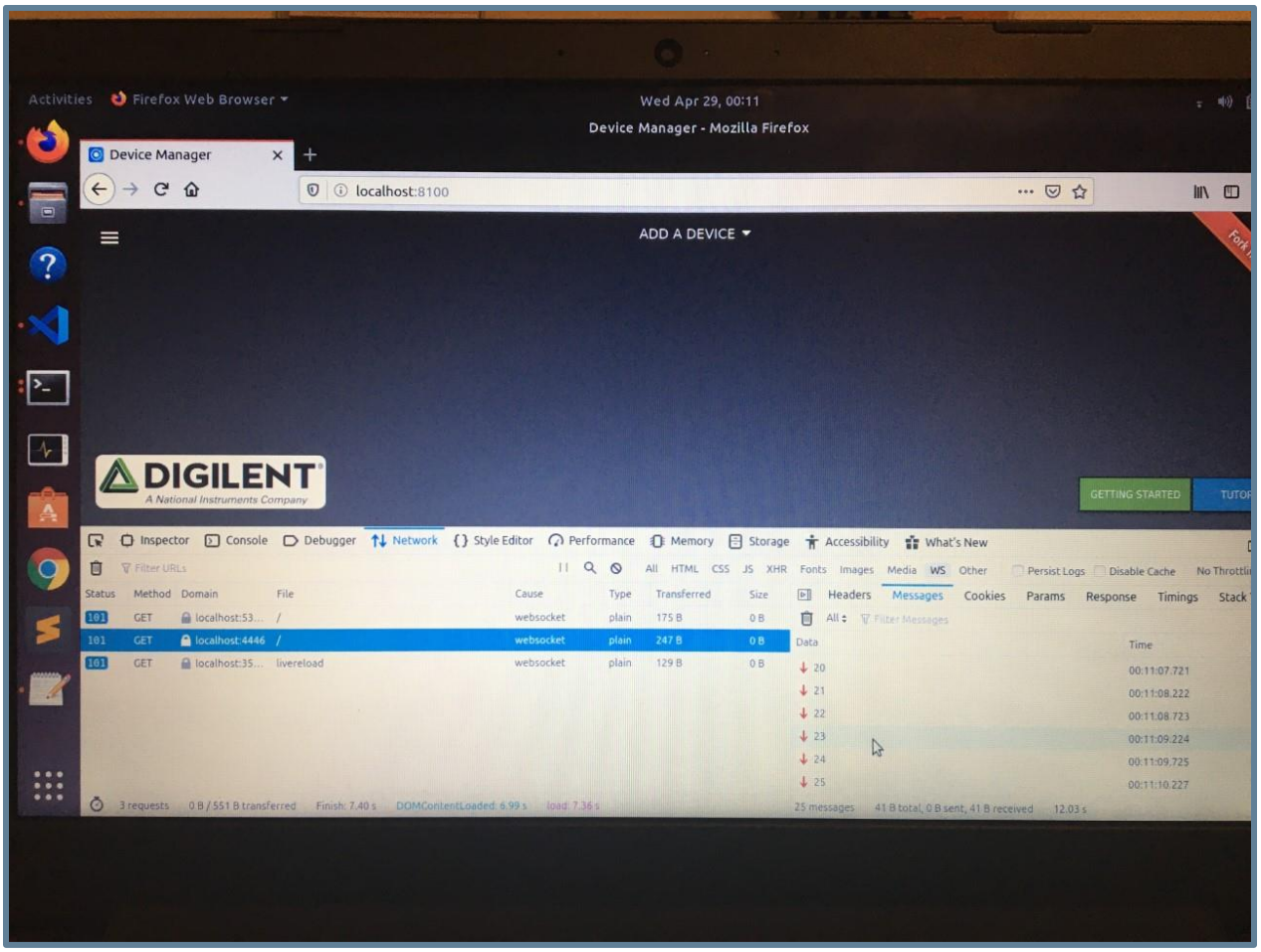


Figure 54: Websocket throwing out dummy data

5.4.6 Backup GUI

As stated in previous sections the team ended up having 2 different GUIs partially implemented. The SimplePlotter was implemented using Matplotlib, Tkinter, and numpy. These are standard libraries for data analysis. This GUI was created from the ground up because not knowing about the surrounding code for Waveforms Live that we did not write became prohibitive to us making progress on the GUI. Every addition we made required us to read through numerous different files to track down why the addition did not work or what proper approach we needed to use to implement the addition.The SimplePlotter was broken into different features that were supposed to be integrated into 1 uniform GUI. Unfortunately, because we started working on the SimplePlotter so late in the semester we were not able to integrate all the features. We were however able to create them in isolation and test them.

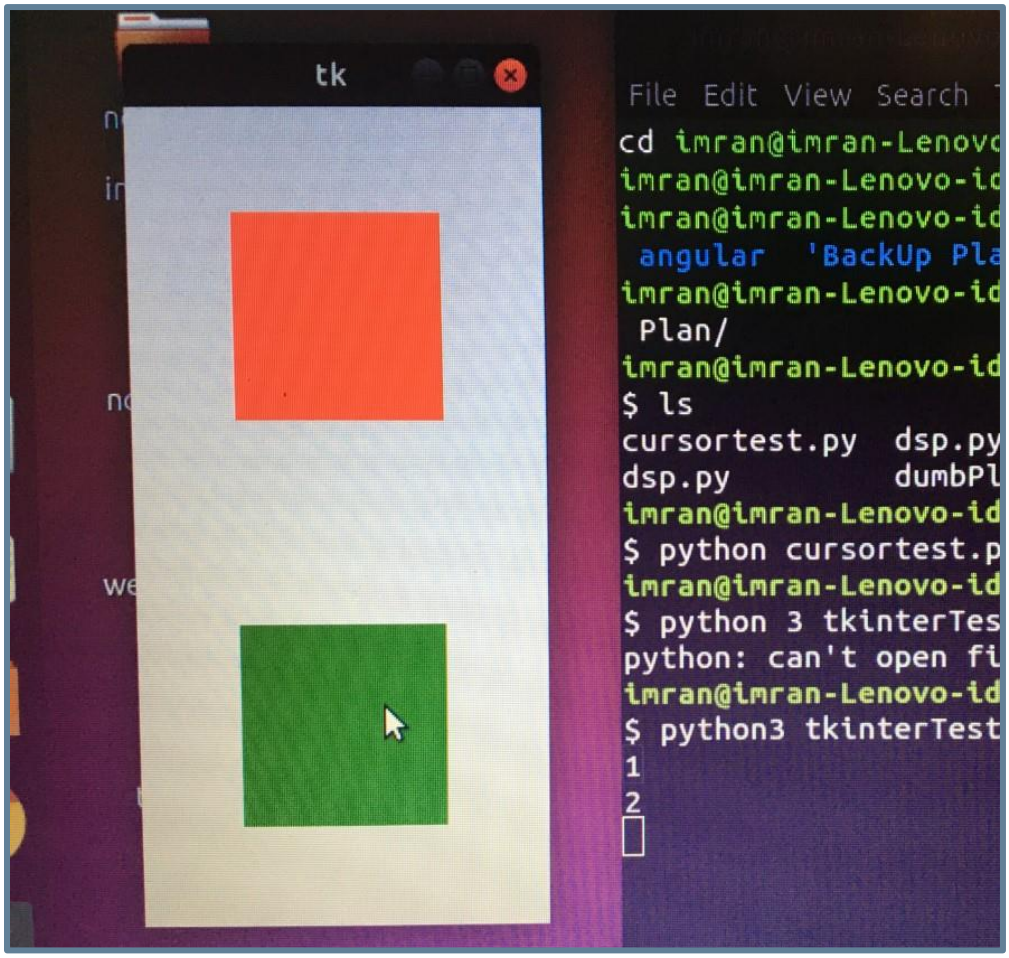


Figure 55: Buttons to adjust zoom level

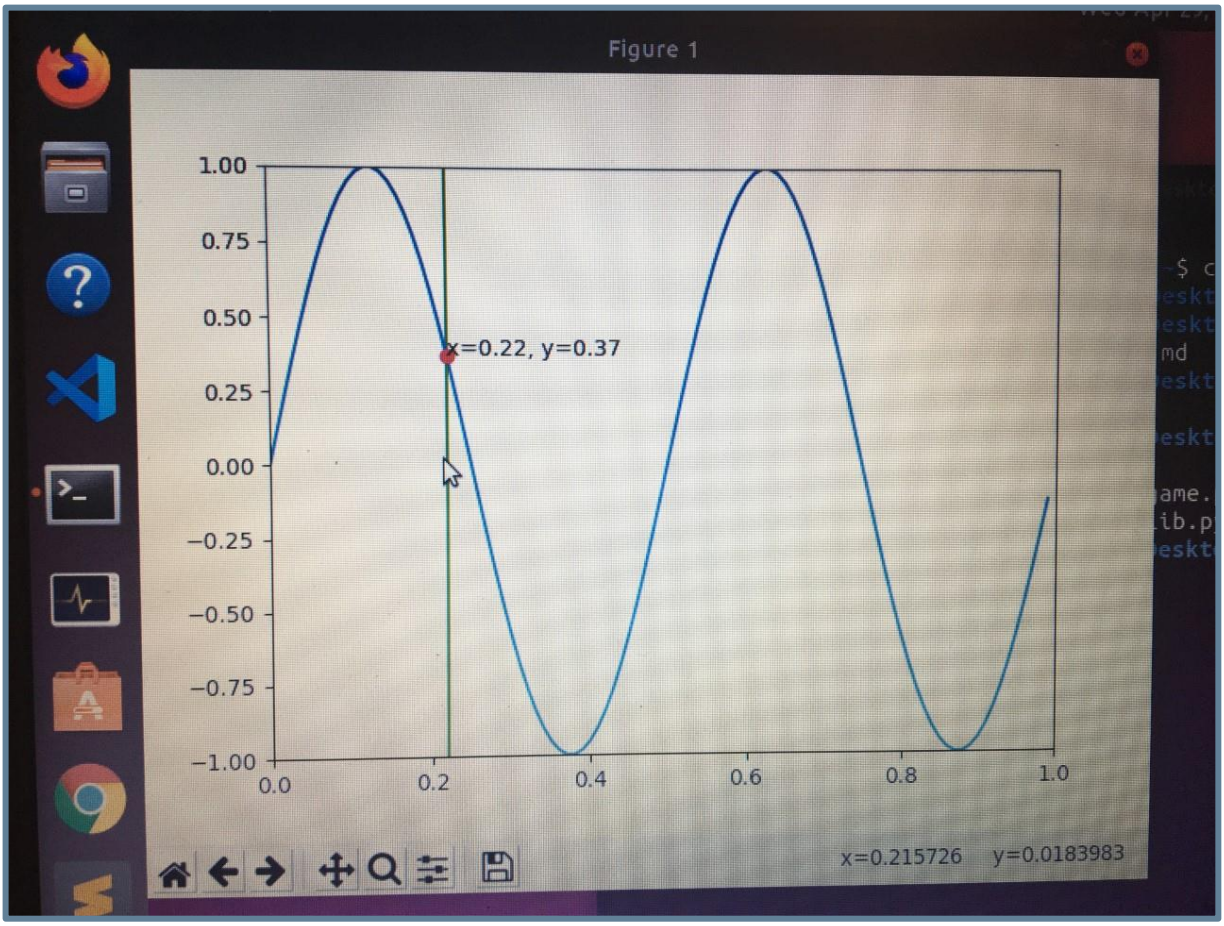


Figure 56: Tracing the waveform with a cursor and showing data values

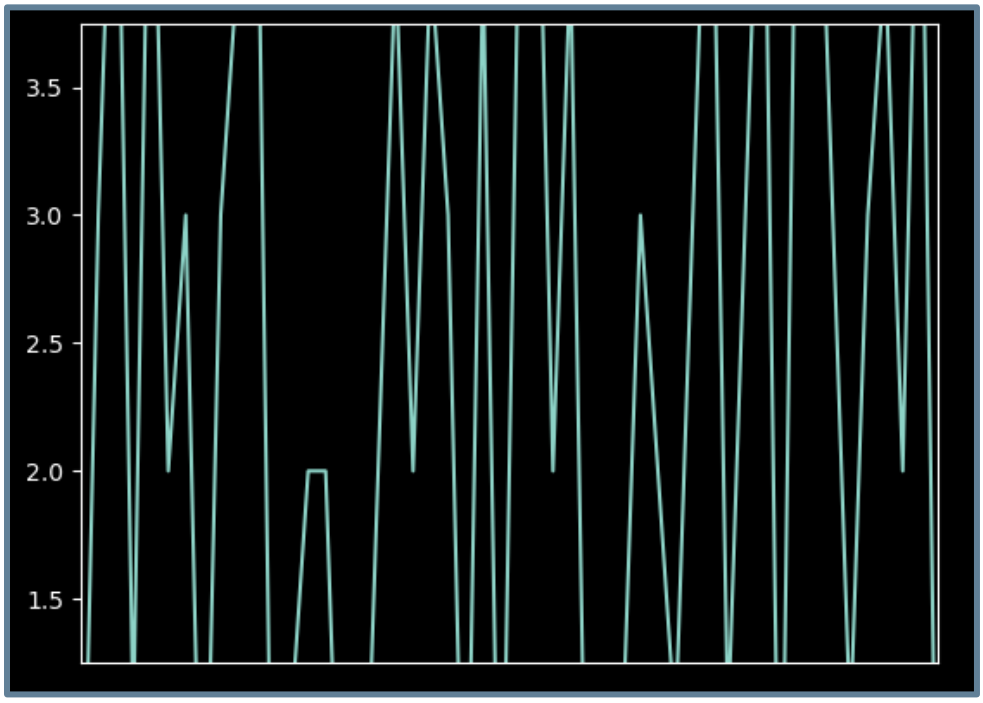


Figure 57: Plotting data values as they come in

6. Implementation, Experimentation, and Success Evaluation

6.1 Current Implementation Status

6.1.1 Analog Front End Implementation

Currently, the analog front-end schematics have been finalized. Although there were many changes discussed with our faculty supervisor, all of those changes have been implemented. The only aspect of this design that is left is testing the OSHO board and finding any errors in the circuit design. Only after these errors are found, a second version of the schematics can be created.

6.1.2 PCB Layout Implementation

As soon as the analog front end schematics were finalized, work on the PCB layout began with first finalizing the selection of passive components, generating an initial bill of material for the project, and acquiring the component footprints required for the layout. Next, the layout of components was generated such that design considerations of EMI performance, proper grounding, manufacturability, and physical constraints were met. Most of this work was done over winter break. However, major changes to the analog front schematics end were made in the second half of this project, so much of this work had to be redone. Additionally, a second board layout was done in order to reduce the board from approximately 29 square inches to just under 17 square inches. These two setbacks delayed this project aspect significantly but the final layout was routed successfully with the design considerations listed above being met and the routing of differential and sensitive signals done properly. As of the time of this report, the analog front end PCB has been commercially printed and is being incrementally soldered and tested. Due to the unfortunate circumstances of COVID-19, the commercial printing and shipping of the PCB also experienced delays. However, team members have expressed interest in continuing with this aspect of the project after ECE493. Through the result of testing this first revision of the PCB, if any major design flaws or design changes for better EMI performance are required, these changes will be made after the completion of ECE493.

6.1.3 FPGA Datapath Implementation

Even though initial progress of the firmware development was slow because of a big gap in the background knowledge required to be able to understand all the resources and the VHDL code, the firmware development’s pace picked up in the second half of the project. A lot of progress was made and the firmware can now properly process, deserialize and plot digitized data from the ADC at up to 720 MHz sampling frequency. To thoroughly verify the functionality of the firmware, the digital logic design was first simulated in Xilinx Vivado. Next, the bitstream generated from the design was used to program the Xilinx Zynq-7000 Zedboard SoC, and tested using the

ADC Evaluation board. The output data was verified using internal logic analyzers (ILAs) within Vivado at sampling speeds of upto 100 MHz. To test the firmware at higher sampling speeds (100 MHz - 720 MHz), python code was written to access the ADC data in memory using an overlay and then plot it using matplotlib. This code was executed using the Jupyter Notebook server on the Zedboard.

6.1.4 Software Implementation

The software was originally supposed to be built on top of the Digilent Waveforms Live product. We began by reading the code base to become more familiar with what is happening at a high level.We also tried to learn javascript, HTML, and CSS to be more familiar with the typescript, HTML, and CSS being used. Once we had some level of understanding of the languages being used we added a button in the HTML and added functions that it activates on click in the Typescript. The project structure that Digilent used was not typical so finding code pieces and understanding it was difficult and delayed. Once we found someone who could help us understand the code and approach it we began to make more progress. The usage of the Ionic library however meant that there were a lot of reference files that were supposed to make the project easier to build. Having all these additional files made it difficult for us to locate dictionaries and objects that we needed to add to in order to allow the GUI to register our custom functions and buttons.

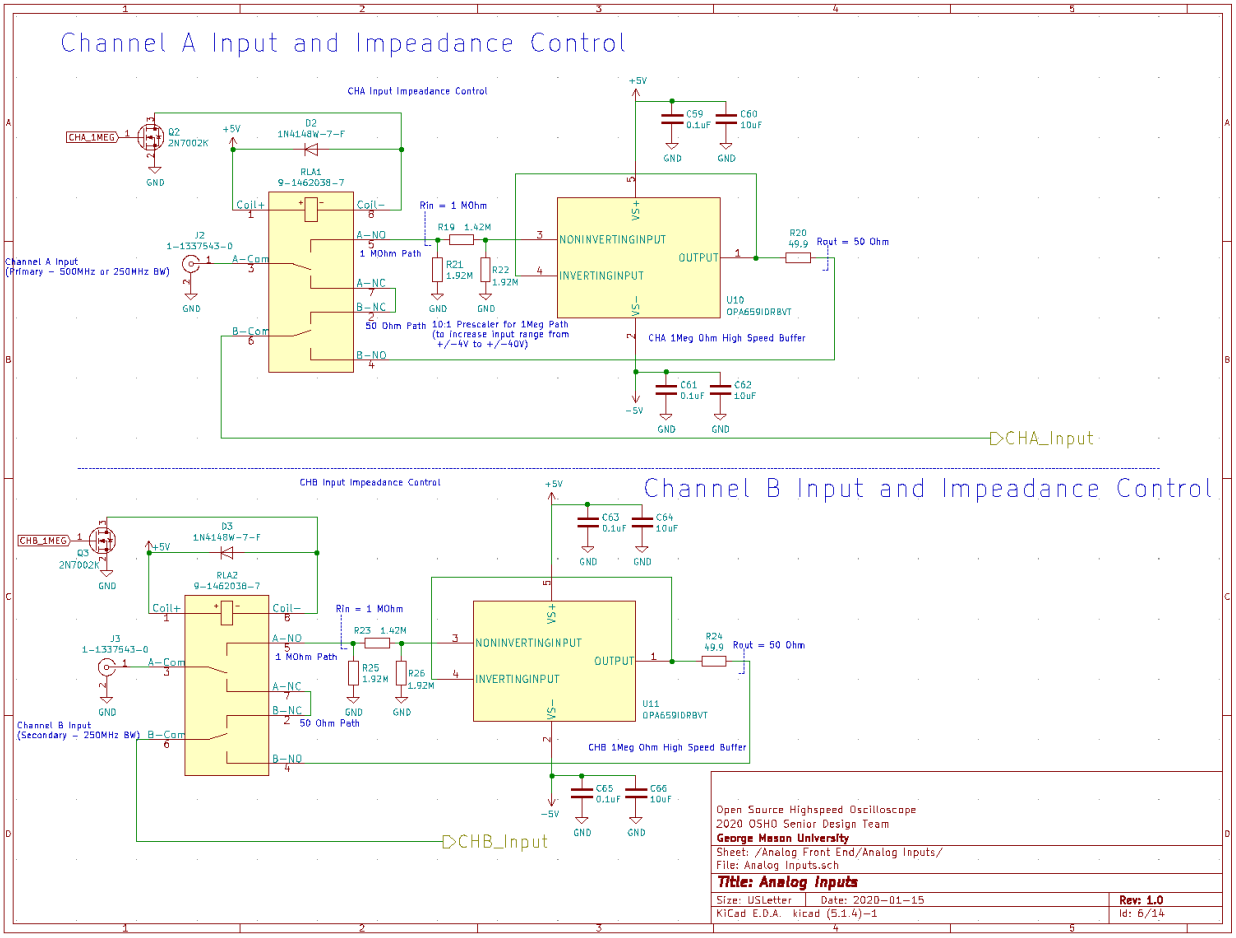
For our SimplePlotter we read up on the Matplotlib library and found sample code to create a plot. Then we looked at sample code to animate a plot so that we could have constant plotting instead of static plotting. With these sample codes we made modifications so that we could do so in 1 process. We then modified the code so that the plot would always plot left to right instead of just scrolling to the right. This allows for a more natural feel and prevented the visualization from moving when we had the same shape plotted on top of itself. Once we got the basic plotting feature we created separate files for the zoom feature and the data tracing feature. Tkinter requires python3 so to integrate we will need to upgrade our plotter and cursor follower to use python3. The separate files were created and tested until we got the basic functionality out of them and then we began trying to integrate them all into 1 file by giving each its own process that would run concurrently. Unfortunately we were not able to have the concurrent processes working. We think the error might have been synchronization of the plotting process not lining up with the input from the zoomin process. Additionally the control logic for some process requires a while loop to keep running whereas the library function being used in the process is not supposed to be looped.

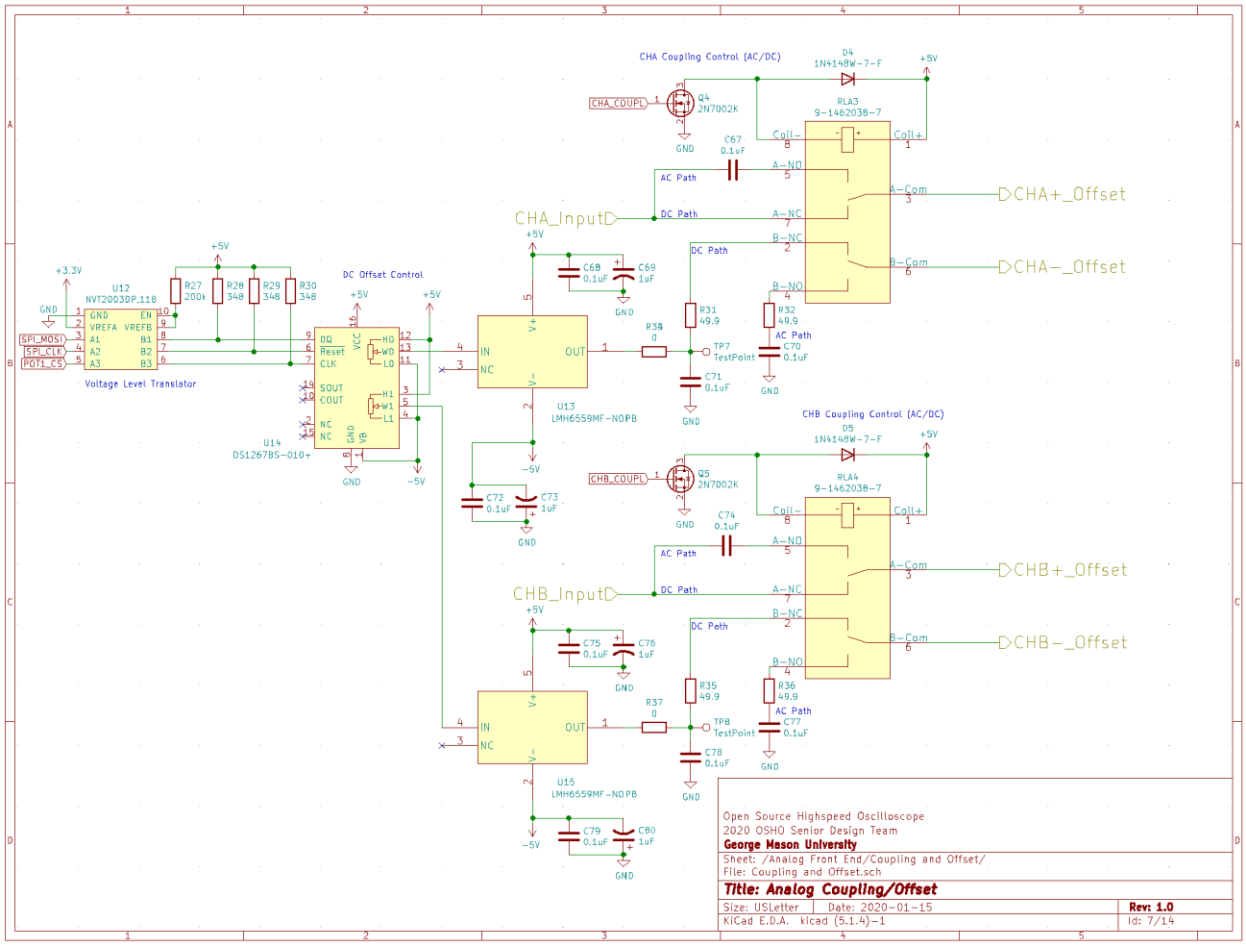
6.2 Design Changes Since ECE492 Design Document

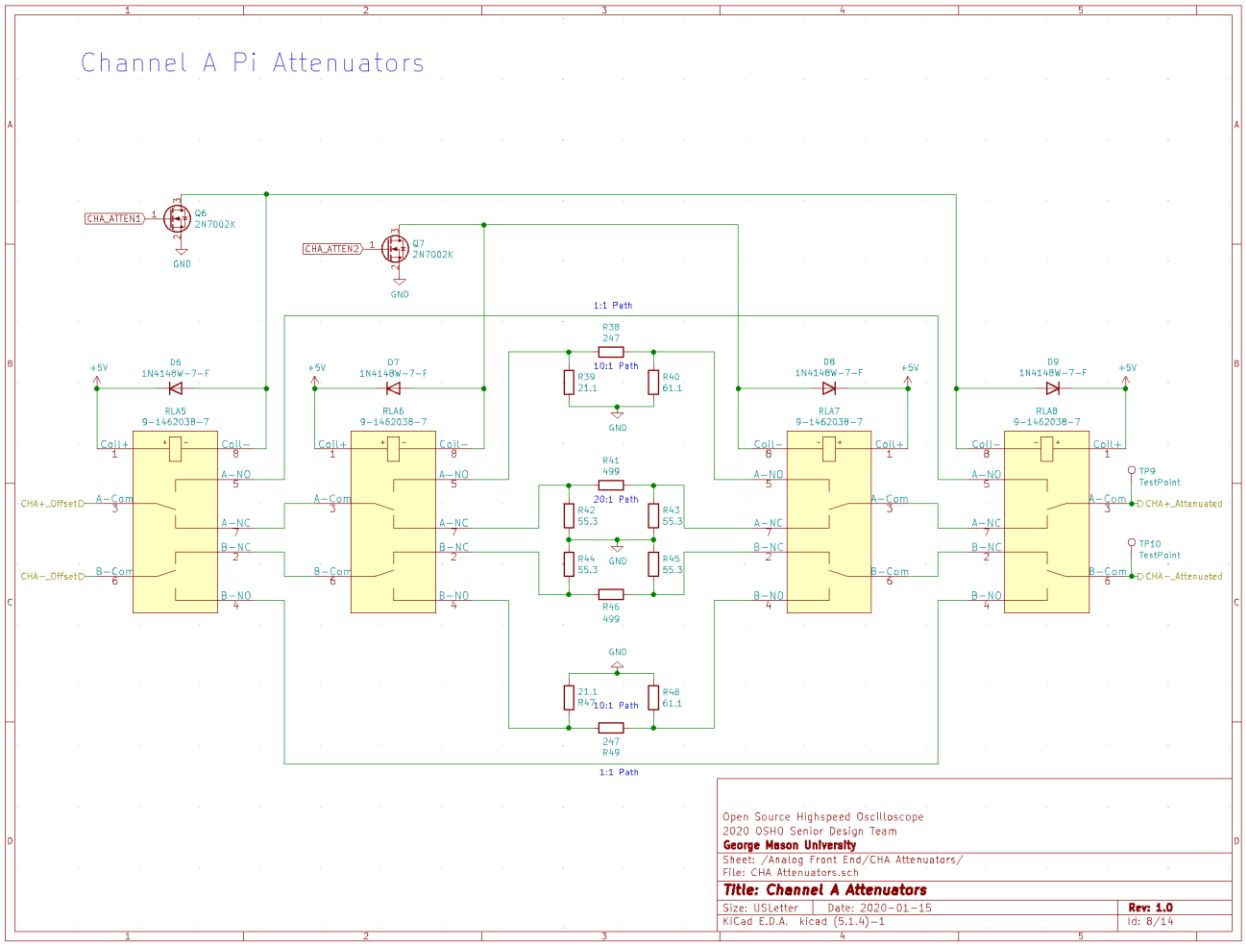
6.2.1 Analog Front End Circuitry

The analog front-end schematics were changed multiple times after the ECE 492 design presentation and report. These changes were in accordance with the guidelines given by our faculty supervisor. Some of these changes took place due to concerns for

cost whereas others were required for the correct operation of the circuit. The amount of relays used in our design was reduced by changing the design so that the cost and power consumption would decrease. Another major change is the fact that an impedance path for 1Mohm was added to comply with the probe connection. Few other components such as gas discharge tubes were added for protection against voltage spikes. LEDs were also added to indicate between AC or DC coupling, attenuation, channel mode, power, and the two impedance paths. A GPIO expander chip was added since the Ultra96-V2 Programmable Logic did not have enough pins to be used for SPI interface with all the required chips. Lastly, a voltage level translator was also added to convert the 1.8V logic from theUltra96-V2 to 3.3V logic which the chips in the circuit operate with. These Design Changes can be seen below:




CHA Coupling Control (AC/DC)
CHB Coupling Control (AC/DC)
DC Offset Control
Voltage Level Translator
CHA_Input
CHB_Input
CHA+_Offset
CHA-_Offset
CHB+_Offset
CHB-_Offset
AC Path
DC Path
U12
NVT2003DP,118
U13
LMH6559MF-NOPB
U14
DS1267BS-010+
U15
LMH6559MF-NOPB
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Analog Front End/Coupling and Offset/
File: Coupling and Offset.sch
Title: Analog Coupling/Offset
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 7/14


Channel A Pi Attenuators
CHA_ATTEN1
Q6
2N7002K
GND
CHA_ATTEN2
Q7
2N7002K
GND
1:1 Path
R38
247
10:1 Path
R39
21.1
R40
61.1
GND
D6
1N4148W-7-F
D7
1N4148W-7-F
D8
1N4148W-7-F
D9
1N4148W-7-F
+5V
RLA5
9-1462038-7
RLA6
9-1462038-7
RLA7
9-1462038-7
RLA8
9-1462038-7
Coil+
Coil-
A-NO
A-Com
A-NC
B-NC
B-Com
B-NO
CHA+_OffsetD
CHA-_OffsetD
R41
499
20:1 Path
R42
55.3
R43
55.3
R44
55.3
R45
55.3
R46
499
R47
21.1
10:1 Path
R48
61.1
R49
247
TP9
TestPoint
CHA+_Attenuated
TP10
TestPoint
CHA-_Attenuated
Open Source Highspeed Oscilloscope
2020 OSHO Senior Design Team
George Mason University
Sheet: /Analog Front End/CHA Attenuators/
File: CHA Attenuators.sch
Title: Channel A Attenuators
Size: USLetter
Date: 2020-01-15
Rev: 1.0
KiCad E.D.A. kicad (5.1.4)-1
Id: 8/14

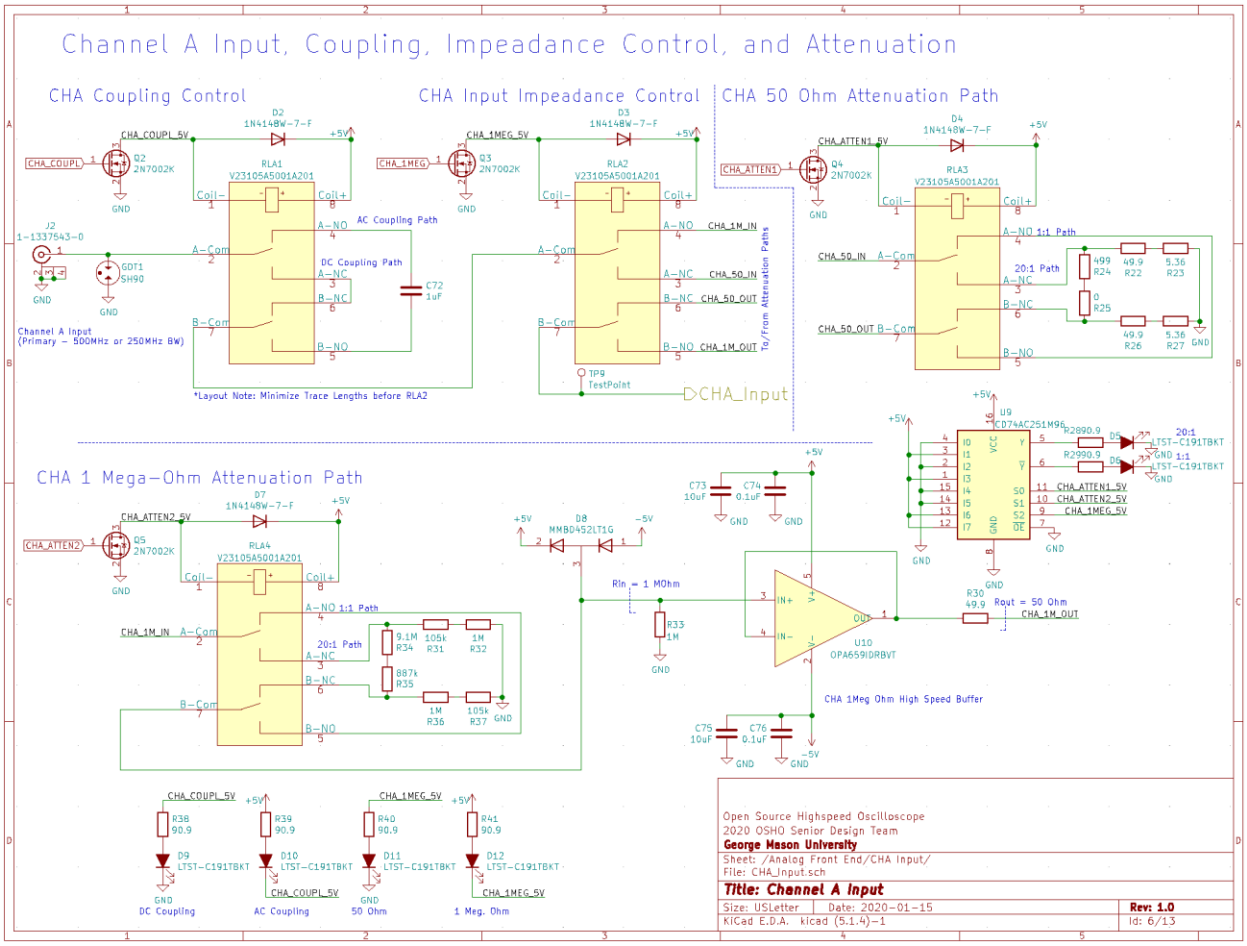


Figure 58. Analog Front End Changes to input impedance control, attenuation and offset (Before - Top 3 Sheets, and After - Last Sheet)

6.2.2 Backup GUI

Originally we thought we would be able to add to the Waveforms Live product. However, because the surrounding code of the Waveforms Live GUI was too time consuming and difficult to navigate/understand effectively we created the SimplePlotter as a backup GUI that we would be able to use so that the hardware team was not stuck waiting for us. This backup GUI was made with the mindset of being a much simple viable product that could achieve the basic task of plotting in a less elegant but equally effective way.

6.2.3 COVID-19 Project Related Scalebacks

Unfortunately, due to the circumstances of the coronavirus pandemic, many aspects had to be scaled in order to meet the additional constraints associated with the pandemic including limited production resources, meeting and working remotely, and delays in acquiring components. These changes mainly involved simplifying the project so that major project aspects would still be demonstrated but a complete integration and special features would be delayed. As such, due to the pandemic (as well as other

technical delays), the testing plans for our system, GUI plans, and scale back in the progress of the programmable logic portion of the project all had to be altered to achieve demonstrable results. In terms of the FPGA firmware, the overall design, once tested successfully on the Zedboard SoC, was supposed to be migrated to the Ultra96-V2 Xilinx Zynq UltraScale+ MPSoC. However, we decided to focus on improving the performance and the maximum clocking speed of the deserializer IP core first as this was a riskier aspect of the project that needed to be worked out first. Additionally, the trigger IP Core was also willingly delayed in favor of developing a new deserializer IP that could utilize dynamic calibration and operate at higher frequencies. Also, the trigger IP was seen as a more full scale integration feature so was put on hold till the end. As for the GUI, the plans for the Waveform's Live integration was swapped in favor of the backup simple GUI as mentioned above. Finally, the testing plan was revised so that we could demonstrate the more complex aspects of the project though unit testing, thereby proving the concept of this solution with managing the setbacks and delays of the project. With all of these changes, we think we made the best of an unfortunate situation, compensated the best we could for our technical delays, and provided a good scaled down proof of concept for the system.

6.3 Experimentation and Testing Plans

6.3.1 High Level Acceptance Testing

Due to the technical delays within the project and COVID-19 setbacks surrounding the project, we were not able to reach full integration with the proposed system solution. However, we thought it was still prudent to outline our initial plans for high level acceptance testing as it is what will define the system tests for the continuation of the project and outline a benchmark for what a successful complete implementation will perform. Below is an outline of the full integration testing plans we had outlined at the beginning of ECE493.

6.3.1.1 Waveform Comparison With Commercial Oscilloscope

The ability of the OSHO system to measure and record waveform data will be verified and the accuracy will be compared against a commercial oscilloscope. The overall device will be tested by applying a waveform to either of the analog inputs. These input waveforms will be varied between DC signals, sine waves, square waves, and triangular waves. Additionally, for each of the waveforms listed above, the input voltage levels will be changed from 0 V_{PP} to 20 V_{PP} in steps of five volts to confirm that the input voltage requirement is met. Furthermore, the signal input will be varied to the following frequencies: 100Hz, 100KHz, 1MHz, and 200 MHz.

First, the GUI will be first visually tested to ensure the system can accurately display these waveforms on the GUI. Screenshots of the GUI will be taken to record and demonstrate this functionality. Then during ten of these waveforms tests, the oscilloscope accuracy will be compared with a high-speed commercially available

oscilloscope. This will be done by measuring the same waveform with both the OSHO system and an 8-bit commercial 1GSPS oscilloscope. The waveform data from both be saved and converted to a .csv format for processing. Using Excel, each of the two waveform records will be aligned, and plotted on a superimposed graph showing the waveforms over time. The average difference between each waveform over time will also be calculated. This test will be deemed a success if the average error for each signal comparison is less than 5%.

6.3.1.2 Measured Frequency Sweep

In this test the bandwidth and frequency response of the OSHO system will be demonstrated in both single and dual channel modes. A function generator will be used to provide a $5V_{PP}$ sine wave to the device. A frequency sweep from 0Hz to 550MHz will be conducted in steps of 10MHz and the absence of aliasing shall be verified for the bandwidth of our device (500MHz). The Amplitude of the measured signal shall be recorded at each frequency and the results shall be presented in a graph showing measured amplitude vs. frequency. This will be repeated in dual channel mode where the frequency sweep will be conducted from 0Hz to 300MHz, as the device bandwidth is halved for dual channel mode (250. For this test to be deemed a success, the measured amplitude of the waveform shall not vary by more than ±3% of the ideal value of $5V_{PP}$ for 95% of the device's bandwidth (since the bandwidth is measured at the point of -3dB).

6.3.1.3 External Clock Input Verification

This test verifies and demonstrates the system’s external clock input and the system’s ability to synchronize ADC sampling with the external clock input. The external clock signal will be generated with a frequency synthesizer and will be used to test the function of the device at five different frequencies: 50MHz, 100MHz, 200MHz, 500MHz, and 1GHz. A very high-speed commercial oscilloscope (>5GSPS) will be used to measure the external clock signal and the ADC clock signal (measured at test points on board). The PLL synchronize operation will be applied to align the phase of the two clocks. The commercial oscilloscope data will be downloaded and the waveforms will be superimposed on the same graph to show the differences between the two clocks. This processing will be done in excel. This will be completed three times for each frequency and the average difference in both frequency and phase of the two clock signals will be recorded. This test will be a success if, for each test, the phase and frequency of each clock signal does not differ by more than 2%.

6.3.2 Unit Integration Testing

6.3.2.1 Analog Front End Testing

6.3.2.1.1 Power Architecture

The power circuitry on the OSHO PCB will be tested initially to ensure that all other chips in the circuit receive the correct supply voltage. This testing includes measuring the voltage at each test point (outputs of voltage regulators) and comparing them with the expected value. The measurements will be carried out through the use of a multimeter and then recorded in table format.

6.3.2.1.2 Input Coupling and Offset

This test will verify the coupling and offset functionality of the front-end circuit. Particularly, an arduino will be used to configure the digital potentiometer through SPI commands. For each potentiometer setting, the offset on the output waveform will be measured and recorded in table format. To test the AC-DC coupling capacity, the arduino will again be used to configure the high speed relay through SPI commands. Both the AC coupled and DC coupled waveforms will be recorded through an oscilloscope. The input and output waveforms will be captured and presented.

6.3.2.1.3 Attenuators

This test will verify the level of attenuation obtained from each of the three different attenuation paths in the front-end circuitry. For unit testing of the attenuators, an arduino microcontroller will be utilized to provide the SPI commands through the pin headers on the PCB. These SPI commands will configure the high-speed relays to choose each of the three attenuation paths one by one. For each attenuation path, the input and output waveform will be captured through the use of an oscilloscope. Furthermore, the peak to peak voltage values of the input and output waveforms will be recorded in table format. The attenuation factor will then be calculated for each of these paths and also recorded in the table. The waveforms used for testing will be generated through the use of a function generator and will range from 100Hz-500MHz in frequency. Furthermore, a DC voltage of +5V will also be tested for each of the three paths to ensure proper operation for DC and AC signals.

6.3.2.1.4 Low Noise Amplifier (LNA)

This test will verify the correct operation of the low noise amplifier. Input signals with a varying range of frequencies and amplitudes will be provided to the LNA and the relationship between input/output voltage (gain) will be recorded. A function generator and oscilloscope will be used to generate the test waveforms and measure the output respectively. Furthermore, the frequency response relationship will be plotted by increasing the input signal frequency from 100Hz to 500MHz incrementally. A graph with a logarithmic scale will be created displaying the gain (in dB) versus the input

frequency (Hz). This will allow a clear understanding of the voltage levels or frequency cut-offs where the output signal starts to saturate.

6.3.2.1.5 Variable Gain Amplifier (VGA)

In this test, the operation of the VGA will be verified by comparing input and output signals from the chip. For unit testing, an arduino will be used to configure the VGA to 5 unique gain settings. A function generator will be utilized to provide input signals with voltages ranging up to the max input rating of the VGA. The gain of the VGA will be recorded in a table format and verified for each different setting configured through SPI. A commercial oscilloscope will be used to record the input and output waveforms.

6.3.2.1.6 Phase-Locked Loop (PLL)

This particular test will verify the clock multiplier functionality of the phase-locked loop. An external differential clock will be provided to the reference input of the PLL. An arduino will be used to configure the PLL through SPI commands. The output frequency of the PLL clock signal will be measured through a commercial oscilloscope and recorded in table format. Five different reference frequencies will be tested.

6.3.2.2 VHDL Firmware Testing

Once the VHDL code had been debugged, it was simulated with some random serial data being sent to the firmware using the testbench. Next, the ADC was configured to send a test ramp signal which is the output of an 8-bit counter (0x00 to 0xFF). The ADC was sampled at incremental sampling frequencies and the output data of the deserializer IP core was viewed using Xilinx's internal logic analyzers (ILAs). To test analog input data, the Analog Discovery 2 kit was used to send various input signals to the ADC. Again, the output deserialized data viewed using the ILAs was compared to the known input data.

However, after making much progress over the winter break and being able to deserialize data at sampling frequencies of up to 150 MHz, no more headway was made towards the beginning of Spring 2020 and there was no improvement in the maximum sampling frequency. Since the Deserializer IP core uses static calibration (determines the delay required to get the frame pattern and then delays the data by the same amount), it was decided to try and make a new Deserializer IP core that would utilize dynamic calibration and could potentially achieve a higher sampling speed. This new OSHO IP core has been completed and though in theory it is better for deserialization as compared to the upgraded HACD deserializer IP core, it still needs some debugging. However, due to timing constraints created especially due to COVID-19, the debugging of this new OSHO Deserializer IP core as a viable solution was halted and testing of the HACD IP core was resumed.

After further modifications and optimizations to the HACD IP core, including modifying the operation of the DMA core so that it does not timeout, the firmware's operating frequency increased to 720 MHz.

Name	Slice LUTs (53200)	Slice Registers (106400)	F7 Muxes (26600)	Slice (1330 0)	LUT as Logic (53200)	LUT as Memory (17400)	LUT Flip Flop Pairs (53200)	Block RAM Tile (140)	Bonded IOB (200)	Bonded IOPADs (130)	IDELAYCTRL (4)	IBUFD S (192)	IDELAYE2/IDELAYE2_FINEDELAY (200)
> design_1_wrapper	12.91%	10.00%	0.26%	24.4...	11.08%	5.58%	7.61%	19.64%	13.00%	100.00%	25.00%	5.73%	0.50%

Figure 59: FPGA Utilization Report

6.3.2.3 Server and GUI Testing

For the Waveforms Live GUI we tested the websocket with sample data. We were able to get the sample data from the websocket to the GUI but were not able to control where it went. As a result the GUI would print the dummy data in the networking tab of the developers console but not display it anywhere else. We tried to modify the function that handled the sample data on the GUI side but were not knowledgeable enough in Typescript and progressive web app design to be effective in this approach. We asked an expert who works frequently with Typescript and progressive web apps but they were not able to understand the design approach that the original Waveforms Live team used as it was not standard. This made it difficult for our expert to help us. Once we hit this point of not knowing what to do or how to be impactful we tried to play with the code with little success. In order to continue being effective in our usage of time we made the SimplePlotter.

For the SimplePlotter GUI we broke the functionality down into different features that were created in separate files with the ultimate goal of integrating them into 1 file all together. Unfortunately we began the process of working on the SimplePlotter too late in the semester and were able to create separate features in separate files but not integrate them into 1 product.

6.4 Experimentation Validation and Testing Results

6.4.1 FPGA Firmware Test Results

First the firmware was simulated in Xilinx Vivado. As seen in the following figure, upon asserting the re-align input the state machine looks for and locks onto the expected frame pattern (0xF0).

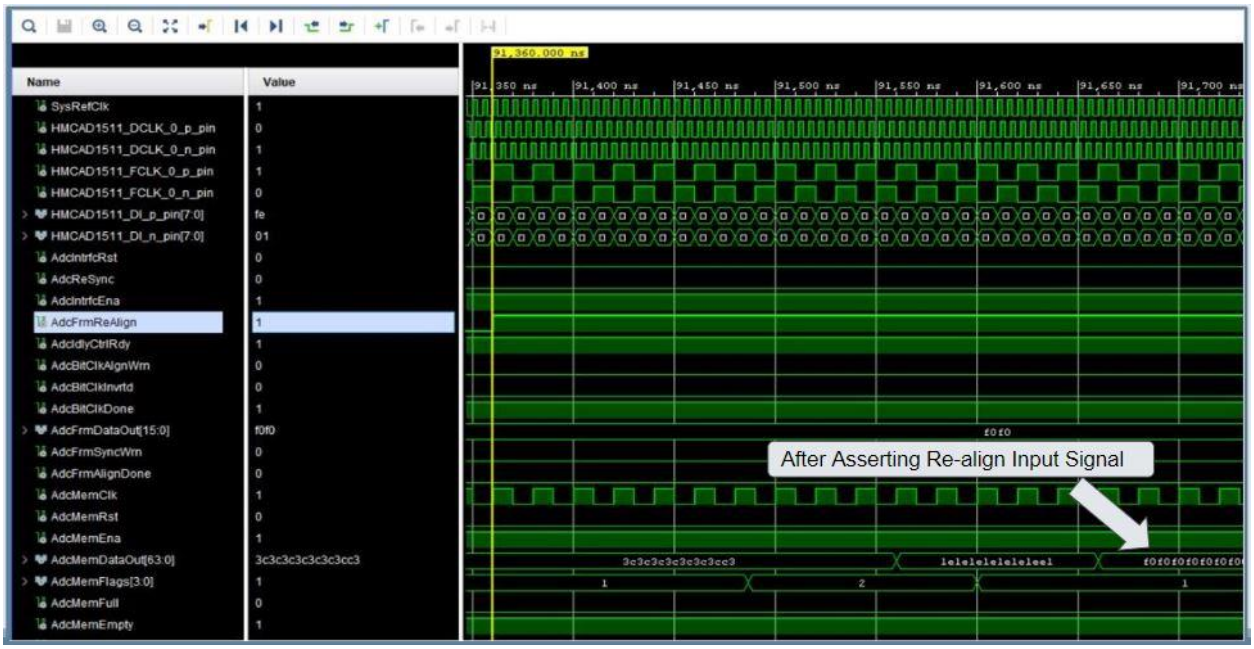


Figure 60: Deserializer IP Simulation Results

After verifying the clock alignment circuitry, the Zedboard SoC was programmed with the generated bitstream and tested with the ADC Evaluation test board. A PLL frequency synthesizer was used to provide the high-speed sampling clock to the ADC. Using python code and the Jupyter Notebook server, SPI commands were sent to configure the HMCAD1511 ADC to send a test ramp signal. As shown in the following figure, the deserializer IP core is successfully able to deserialize the digitized samples and provide a waveform that is identical to the input ramp signal.

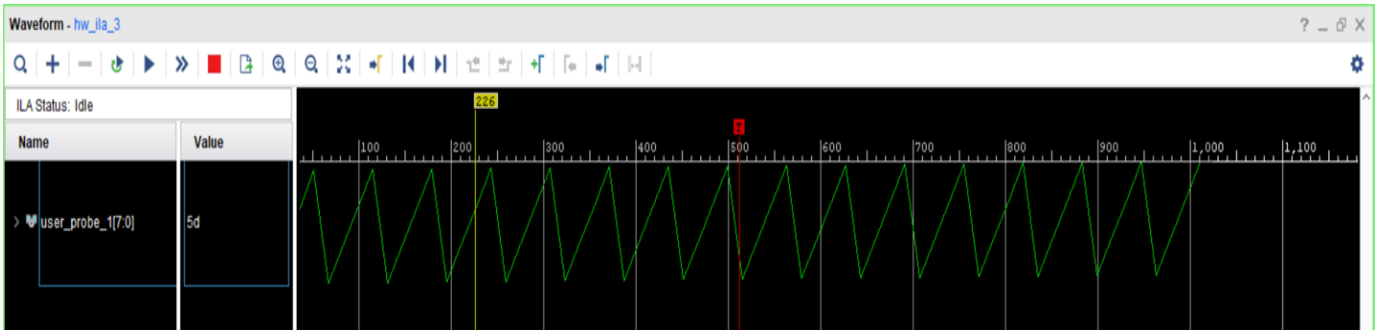


Figure 61: Vivado ILA Waveform

Since sampling frequencies of over 100 MHz exceed the ILA's Nyquist frequency, python code was written and run on the processor's Jupyter Notebook server. The code read data from the memory and plotted it using matplotlib. The following picture shows the result of reading the data (stored as 64-bit packets) from memory.

```
In [101]: print([hex(i) for i in listcapture])
['0x5f5f5f5f5f5f5f5f', '0x6060606060606060', '0x6161616161616161', '0x6262626262626262', '0x6363636363636363', '0x6464646464646
46464', '0x6565656565656565', '0x6666666666666666', '0x6767676767676767', '0x6868686868686868', '0x6969696969696969', '0x6a6a
6a6a6a6a6a6a', '0x6b6b6b6b6b6b6b6b', '0x6c6c6c6c6c6c6c6c', '0x6d6d6d6d6d6d6d6d', '0x6e6e6e6e6e6e6e6e', '0x6f6f6f6f6f6f6f6f',
'0x7070707070707070', '0x7171717171717171', '0x7272727272727272', '0x7373737373737373', '0x7474747474747474', '0x757575757575
7575', '0x7676767676767676', '0x7777777777777777', '0x7878787878787878', '0x7979797979797979', '0x7a7a7a7a7a7a7a7a', '0x7b7b7
b7b7b7b7b7b', '0x7c7c7c7c7c7c7c7c', '0x7d7d7d7d7d7d7d7d', '0x7e7e7e7e7e7e7e7e', '0x7f7f7f7f7f7f7f7f', '0x8080808080808080',
'0x8181818181818181', '0x8282828282828282', '0x8383838383838383', '0x8484848484848484', '0x8585858585858585', '0x868686868686
8686', '0x8787878787878787', '0x8888888888888888', '0x8989898989898989', '0x8a8a8a8a8a8a8a8a', '0x8b8b8b8b8b8b8b8b', '0x8c8c8
c8c8c8c8c8c', '0x8d8d8d8d8d8d8d8d', '0x8e8e8e8e8e8e8e8e', '0x8f8f8f8f8f8f8f8f', '0x9090909090909090', '0x9191919191919191',
'0x9292929292929292', '0x9393939393939393', '0x9494949494949494', '0x9595959595959595', '0x9696969696969696', '0x979797979797
9797', '0x9898989898989898', '0x9999999999999999', '0x9a9a9a9a9a9a9a9a', '0x9b9b9b9b9b9b9b9b', '0x9c9c9c9c9c9c9c9c', '0x9d9d9
d9d9d9d9d9d', '0x9e9e9e9e9e9e9e9e', '0x9f9f9f9f9f9f9f9f', '0xa0a0a0a0a0a0a0a0', '0xa1a1a1a1a1a1a1a1', '0xa2a2a2a2a2a2a2a2',
'0xa3a3a3a3a3a3a3a3', '0xa4a4a4a4a4a4a4a4', '0xa5a5a5a5a5a5a5a5', '0xa6a6a6a6a6a6a6a6', '0xa7a7a7a7a7a7a7a7', '0xa8a8a8a8a8a8
a8a8', '0xa9a9a9a9a9a9a9a9', '0xaaaaaaaaaaaaaaaa', '0xabababababababab', '0xacacacacacacacac', '0xadadadadadadadad', '0xaeaea
eaeaeaeaeae', '0xafafafafafafafaf', '0xb0b0b0b0b0b0b0b0', '0xb1b1b1b1b1b1b1b1', '0xb2b2b2b2b2b2b2b2', '0xb3b3b3b3b3b3b3b3',
'0xb4b4b4b4b4b4b4b4', '0xb5b5b5b5b5b5b5b5', '0xb6b6b6b6b6b6b6b6', '0xb7b7b7b7b7b7b7b7', '0xb8b8b8b8b8b8b8b8', '0xb9b9b9b9b9b9
b9b9', '0xbababababababaaa', '0xbbbbbbbbbbbbbbbb', '0xbcbcbcbcbcbcbcbc', '0xbdbdbdbdbdbdbdbd', '0xbebebebebebebebe', '0xbfbfb
fbfbfbfbfbf', '0xc0c0c0c0c0c0c0c0', '0xc1c1c1c1c1c1c1c1', '0xc2c2c2c2c2c2c2c2', '0xc3c3c3c3c3c3c3c3', '0xc4c4c4c4c4c4c4c4',
'0xc5c5c5c5c5c5c5c5', '0xc6c6c6c6c6c6c6c6', '0xc7c7c7c7c7c7c7c7', '0xc8c8c8c8c8c8c8c8', '0xc9c9c9c9c9c9c9c9', '0xcacacacacaca
caca', '0xcbcbcbcbcbcbcbcb', '0xcccccccccccccccc', '0xcdcdcdcdcdcdcdcd', '0xcecececececececе', '0xcfcfcfcfcfcfcfcf', '0xd0d0d
```

Figure 62: Digitized Data In Zynq PS Memory

The following pictures show the data plotted using matplotlib in Jupyter Notebook server. The plotted signals match the analog input signals that were sampled by the ADC verifying the successful operation of the firmware.

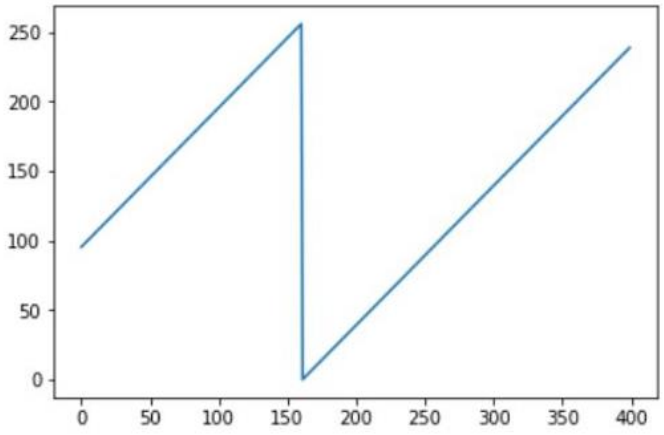


Figure 63 Ramp Signal Sampled at 550 MHz

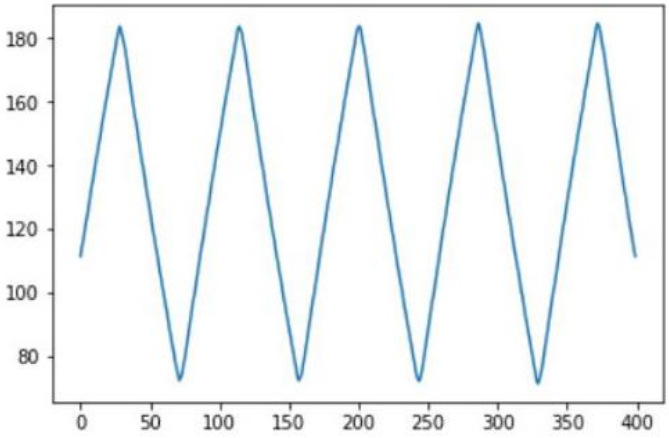


Figure 64: 1 MHz Triangular Input Signal from Analog Discovery 2 Sampled at 550 MHz

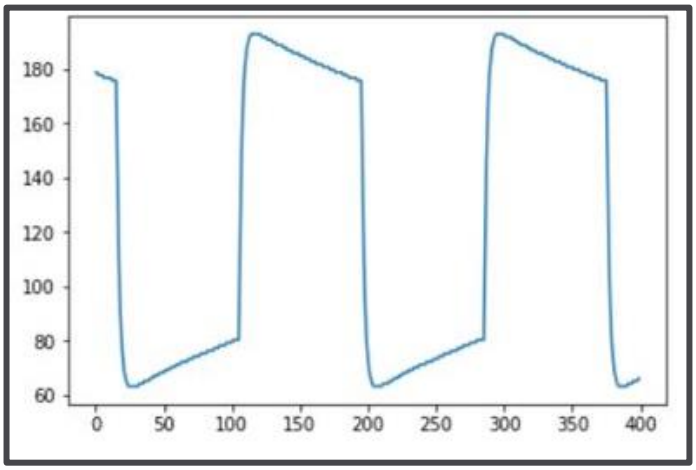


Figure 65: 500 kHz Square Wave Input from Analog Discovery 2 Sampled at 720 MHz

6.4.2 Data Visualization and GUI Test Results

Our Waveforms Live implementation of the GUI was not able to plot any data but we were able to send data back and forth with a websocket.

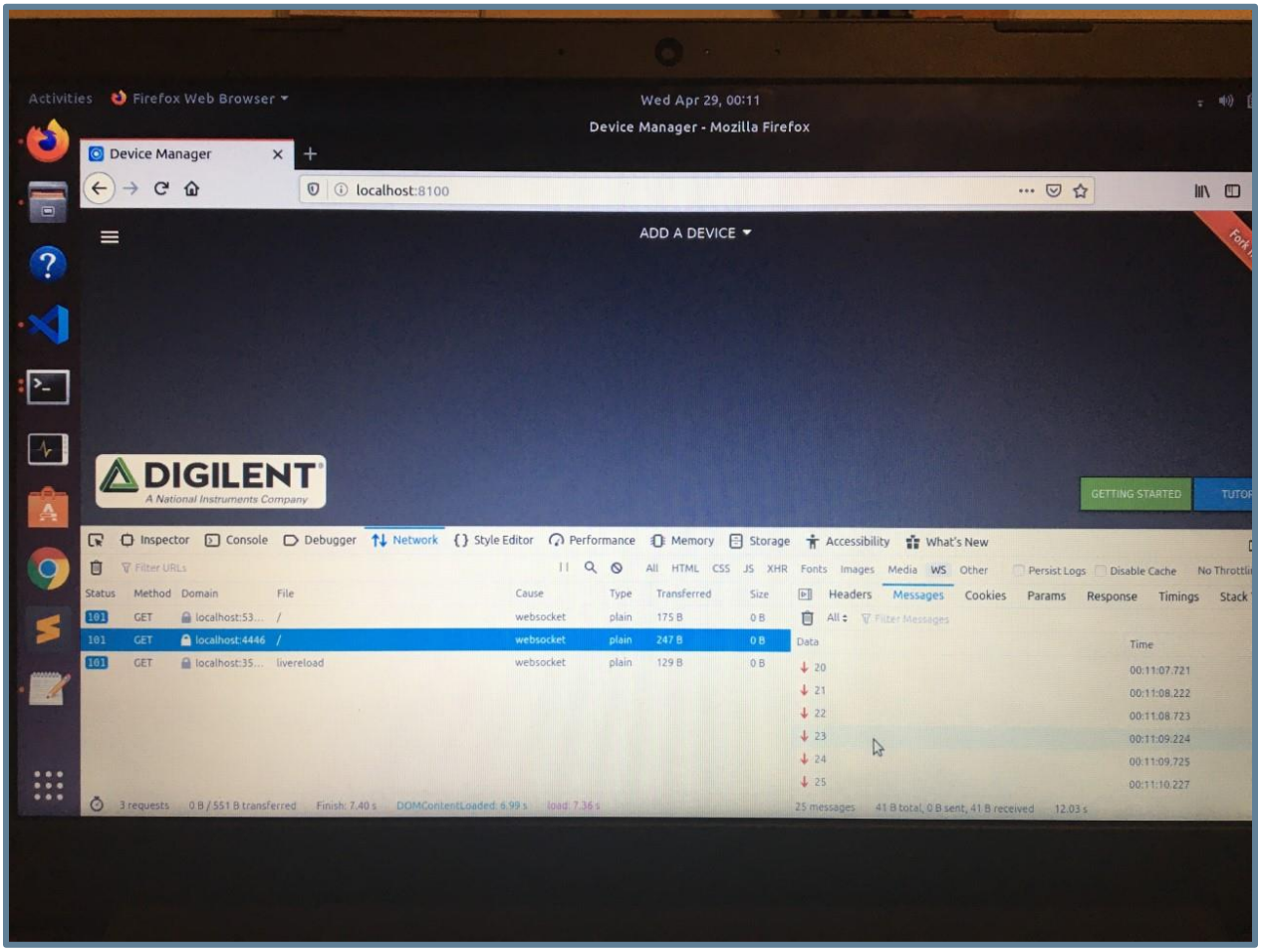


Figure 66: Data being passed through the websocket

The SimplePlotter was able to plot data but not with all the features in 1 product. We had success plotting data in Linux and Windows environments but when we took the same code to the jupyter notebook environment of the Zed board that the hardware team was using we were only able to plot one data point at a time. This means that we would have to rerun the script 50 times to plot 50 pieces of data. Although that was not how we intended the GUI to work we do not think that will affect the overall product as the final product is going to run on the Ultra96 which runs Linux.

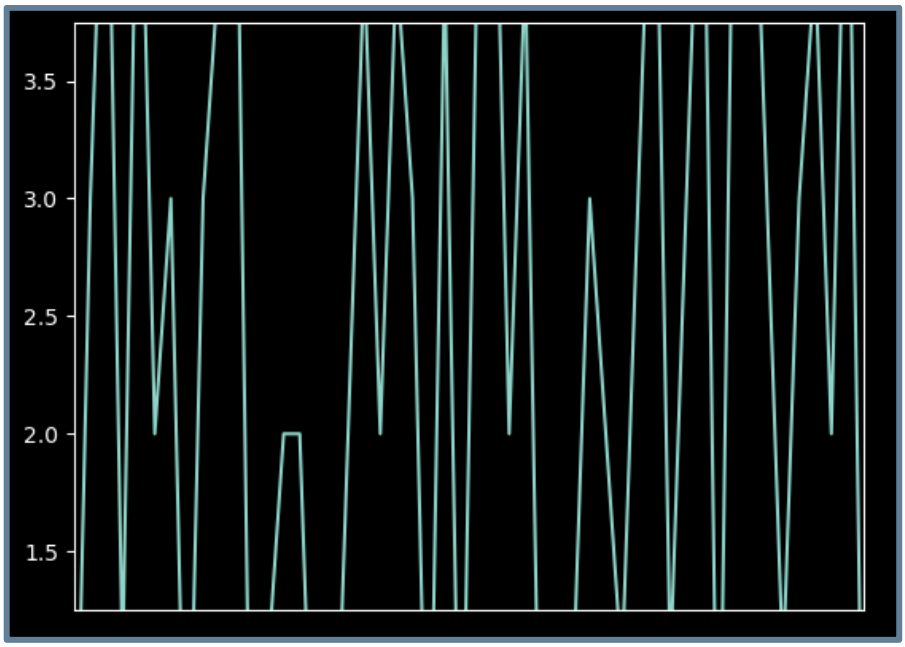


Figure 67: Successful plotting on a Windows and Linux environment

Figure 68: Single datapoint plotting on Jupyter Notebook

6.5 Solution Operational Requirements Analysis

Below is an analysis of whether our implemented solution meets all of the operational requirements as outlined in the beginning of the project.

6.5.1 Input/Output Requirements

Requirement: The device shall have at least two analog input channels, one external clock input, and one external trigger input.

Analysis: The system has two analog input channels, one differential clock input and one external trigger input, although the trigger input functionality has yet to be implemented in firmware and software. This feature can be added at a later date with further VHDL and software development. Therefore, this aspect of the project can be seen as successful.

Requirement: The system will receive control and configuration commands as well as be able to responsively display captured data through a web client with an intuitive and responsive GUI.

Analysis: The system currently only has a basic backup GUI so this requirement is only marginally implemented.

6.5.2 External Interface Requirements

Requirement: The device will provide support for 1x and 10x passive probe inputs (50Ω and 1MΩ).

Analysis: The system supports both 50Ω and 1MΩ probes with both 1x and 10x attenuation amounts. Therefore this requirement was successfully implemented.

Requirement: Bayonet Neill–Concelman (BNC) connectors shall be used for the analog inputs and external trigger inputs.

Analysis: The analog front end PCB has BNC connectors for each of these inputs, therefore this requirement was successfully implemented.

Requirement: The system shall interface with a network capable computer through USB3.0 or WiFi.

Analysis: The Ultra96-V2 implements network connection through both WiFi and USB3.0 (acting as a network adapter), therefore this requirement has been met.

Requirement: The system shall receive power from an external 5V DC power supply.

Analysis: The Analog Front End is powered by an external 5V DC power supply, therefore this requirement is successfully met.

6.5.3 Functional Requirements

Requirement: The analog-to-digital converter (ADC) shall sample one input channel at 1 GSPS or two channels at 500 MSPS.

Analysis: The analog-to-digital converter (ADC) samples one input channel at 1 GSPS or two channels at 500 MSPS. This requirement was successfully implemented.

Requirement: The device will be able to measure analog inputs with a maximum input voltage of ±10V.

Analysis: The analog front end is designed for an input of ±50V using 1MΩ input mode and ±10V using 50Ω input mode. Therefore this requirement has been met and exceeded.

Requirement: The input analog circuitry shall achieve a 500 MHz bandwidth.

Analysis: Channel A has a Bandwidth of 500 MHz in single channel mode and both channels have a bandwidth of 250Mhz in dual channel mode. Therefore, this requirement was successfully met.

Requirement: The ADC shall be able to be configured to sample using either the FPGA clock or an external clock input (between 30 MHz and 1 GHz).

Analysis: The analog front end has a differential SMA external clock input as well as an auxiliary crystal oscillator sampling clock reference, therefore this requirement was met and exceeded.

Requirement: The ADC output sample resolution shall be no less than 8 bits.

Analysis: The ADC resolution is 8 bits; this requirement was successfully met by our design.

Requirement: The system's data capture shall have the ability to be triggered using both configurable edge triggers as well as a configurable external trigger input.

Analysis: Trigger implementation is not currently implemented in software but is implemented in hardware, and therefore this requirement is not currently met. However this can be implemented in the future.

6.5.4 Technology and System-Wide Requirements

Requirement: The front-end device shall use a single 1GSPS ADC chip.

Analysis: The front-end board uses a single HMCAD1511TR which is a 1GSPS ADC; this requirement was successfully met.

Requirement: The ADC data shall be processed and hosted on an onboard Linux web server using a Xilinx Zynq UltraScale+ multiprocessor systems-on-chip (MPSoC) aboard the Ultra96 Board.

Analysis: The waveform data can currently be accessed via the Ultra96 development board but is limited in terms of full implementation, therefore this requirement is implemented marginally unsuccessfully.

Requirement: The analog front-end custom PCB should interface with the Ultra96 Board for data processing.
Analysis: The analog front-end custom PCB successfully mates with the Ulta96 Development Board, and thus this

Requirement: Target FPGA development board shall have device driver firmware for interfacing with the ADC, and routing and storing ADC sample data in a memory device.
Analysis: The FPGA firmware has had firmware developed and tested at ~750 MHz on the Zedboard/Easyboard setup. Therefore this requirement is mostly successful.

Requirement: Front-end programmable devices will be controlled using the Serial Peripheral Interface (SPI) or other serial protocol.
Analysis: Front-end programmable components are controlled using the Serial Peripheral Interface (SPI) and this requirement has been successfully met.

Requirement: The custom high-speed PCB and Ultra96 devices will interface with each other via the Ultra96's high-speed and low speed mezzanine connectors.
Analysis: The OSHO PCB uses both the Ultra96's low speed and high speed connectors, and therefore this requirement has been successfully met.

Requirement: The device should be low-cost ($600 or less).
Analysis: The device costs $587 including the cost of a Ultra96-V2 development board and therefore this requirement has been met.

6.6 Project Success Evaluation

6.6.1 Analog Front End and PCB

Overall, the analog front end design and PCB layout aspects of this project were very successful. The analog front-end schematics were finalized at the beginning of the Spring 2020 semester and the PCB layout has been completed to the exact specifications required by the project. The only aspect of analog front-end circuit that has not been completed is the soldering and testing of the board. Although this is an integral part of the project, it could not be completed due to COVID-19 lockdowns. Due to this restriction, we have declared that this aspect of the project is successful as allowed by the current situation.

6.6.2 FPGA Datapath and Firmware

Tremendous progress was made in terms of the operation of the firmware. The maximum operating frequency was increased from 50 MHz at the end of ECE 492 to 720 MHz as of now. The deserializer IP core is able to correctly deserialize data and the datapath then stores it in memory. The test results discussed in 6.4.1 verify the firmware's success. Despite the overall progress, however, the firmware was not able to reach its required frequency of operation of 1 GHz. The reason is that since the LVDS data is coming serially over 8 LVDS channels, the sampling frequency is reduced from 1 GHz to 125 MHz. And since the PL layer of the SoC is being clocked at 100 MHz, it cannot handle this frequency. However, the firmware was able to operate at up to 720 MHz which is very close to the maximum possible ADC sampling frequency of 800 MHz. One possible solution is to change the FCLK_CLK0 (PL clock) frequency and rebuild the FPGA's first stage boot loader. The same design should then be able to handle a 1 GHz input.

6.6.3 Software and GUI

As a whole we think that the GUI achieved the base task of plotting data but did not achieve all the separate features we initially intended on providing. The original goal was to have a plotting tool that integrates into Waveforms Live so that we could leverage all Oscilloscope functionality that Waveforms Live has in addition to adding functionality like triggers and AXI/SPI control. We invested a lot of time in trying to understand Waveforms Live and being able to control it because we knew that if the investment paid off we would end up with a higher quality product for visualizing the data. The knowledge gap was too much for the GUI team to overcome so all the time they spent on Waveforms live was essentially unfruitful. This forced the GUI team to create another plotter from the ground up so they could have full control over it and more easily integrate it together. This also allows the GUI team to not be empty handed in their deliverables. There was at least something they could show for the work they put in. We understand that the current implementation of the GUI is not in the best condition it could be but it does achieve the base goal of plotting. We think the next step for the GUI is integrating the different features into 1 base product and taking a 2nd look at waveforms live from someone who has a stronger Typescript/progressive web app background.

6.6.4 Overall Project

From the start, this project was a very challenging task given the project timeline and our previous experience. However, we have achieved a tremendous amount of progress on the project as a whole. Each part mentioned above was completed to the full extent possible. Although the original measure of success for this project was full integration and testing, this could not be completed due to the pandemic. Progress on some of these aspects is still scheduled to continue but at this point, the overall project is still considered a successful proof of concept given the COVID-19 situation and

technical setbacks we encountered. The project was a successful learning experience and interesting senior design project.

7. Administrative Project Aspects

7.1 Project Continuation and Future

As far as the analog front-end design is considered, the soldering and testing of the OSHO board will continue and any errors that are found will be fixed in the next version of the schematics and PCB layout. However, this testing cannot be completed without full access to a lab.

In case of the firmware, the next step would be to increase the FPGA's global clock speed so that it can handle and process input data of higher frequencies (> 720 MHz). Testing and debugging the new OSHO IP core will also allow for more accurate data since it only deserializes and outputs valid data when the clocks are properly aligned.

For the GUI we know the next steps of the project require integration of the 3 different SimplePlotter features. The other option for continuation on the GUI side is to re-approach with Waveforms Live by reading the code with an expert. This will allow better understanding of the underlying code so that the team can be knowledgeable enough to effectively add to it. The team needs to add a function that is run when the confirmation button is pressed to use the Ultra96. That function should start the websocket client that is already written and once the connection is made it should open the plotting tool of Waveforms Live. The team should also make another websocket server and client that handles the sending of SPI commands. This server should start when the Ultra 96 boots up (just like the data websocket server). The client should be launched once the plotting tool is launched. An AXI websocket client and server should also be created for doing the same communication but for the AXI commands.

Luckily, Team Members have expressed interest in continuing the project, and the open source nature of our project allows us to continue this project past the scope of ECE492 and ECE493. This project has real marketable and engineering value, and thus should be continued in the future. A Github page for the project has been setup (https://git.gmu.edu/tbulloc2/osho/-/tree/master/OSHO%20Hardware) for all project resources and will create a great site for contributors to add to the project in the future.

The future of our project in terms of our product retirement, maintenance, and disposal, our solution provided minimal impact as the only hardware solution that we provide that cannot be reused (like the Ultra96-V2) is the analog front end PCB, and software aspects can be updated in an open source fashion. At the end of the lifecycle of the PCB the board will need to be properly disposed of. PCBs are not biodegradable so they need to be recycled in the proper way through websites such as https://www.webuyics.com/scrap-pcb.htm. The parts can be desoldered if the user wishes to hold onto them. Because the GUI and FPGA Firmware is code it does not have a lifecycle defined by when it stops working but rather when the newest update

needs to be pushed. With this in mind the GUI will have a relatively short lifecycle as we know that there are improvements that still need to be made. Once the GUI is updated the appropriate course of action is to download the new update from where it is being stored and begin to use that version.

7.2 Project Challenges

7.2.1 Project Scope and Complexity

Although the analog front-end circuit design and VHDL code development were concepts that the team was familiar with, the amount of detail required in the designs of these was beyond the knowledge gained in undergraduate courses. Concepts such as noise filtering, impedance and trace matching, clock alignment, developing AXI IP cores, and others required extensive research.

The GUI design team was challenged with adding functionality to an already existing complex product that had to fully integrate into the already existing code. This was difficult as the team had no prior experience working with Typescript, HTML, CSS or the Ionic framework. To have the first Typescript project someone works on to be a fully functional oscilloscope that adds functionality to an already existing, complex, uncommented code base was more difficult than anticipated.

7.2.2 Design Change Delays

The design changes discussed in section 6.2 essentially required that the PCB layout needed to be redone which cost at least a couple weeks of delay in progress. On the software side, changing from the Waveforms Live GUI to the SimplePlotter resulted in us essentially losing all the time we spent on Waveforms Live and made it so a completely integrated GUI could no longer be achieved in the time remaining in the semester.

7.2.3 Problems with Existing Project Materials

Working on top of the existing Waveforms Live code base sounded like a great way to save time but ended up costing us more time than it saved. The code base was uncommented and not following the standard progressive web app design framework so we had to spend a lot of time digging around the different pieces of code to understand what was going on. The existing code base was also not commented which meant that the GUI team had to read the code, make an educated guess as to what was going on, try to make a change based on that educated guess and then, if it did not work, read through the error codes and different forums to figure out what was going wrong. This often meant that making 1 change took 3-4 iterations of code. As a result we spent a lot of time trying to achieve tasks that on the surface looked simple.

Furthermore, the python notebooks and deserializer IP Cores were unorganized and uncommented. To understand their work, meetings with old HACD team members were conducted.

7.3 Non-Planned Activities

7.3.1 Major Analog Front End Changes at Beginning of ECE493

Some of the major analog front-end design changes include the reduction of attenuation paths from three for each input channel to just two. Furthermore, 1 Mohm impedance path was added to the circuit so that probes could be used with the oscilloscope. Even more overvoltage and transient protection was added to the analog input channels. Due to these changes and some others, the PCB layout had to be restarted, thus, delaying the manufacturing process. The schematics that were designed before these changes took place are also presented in section 6.2.1.

7.3.2 Development of the New Custom AXI Deserializer IP Core

The HACD deserializer IP core utilizes static calibration to align the bit and frame clocks. At the required speed, the frame clock operates at 1 GHz. This gives a sampling window of 1 ns in an ideal case. Due to clock jitter, PCB trace length and clock skew/uncertainty, however, this window is further reduced and is too small to capture data with static calibration. Therefore, a new Deserializer IP core was made that would utilize dynamic calibration and could potentially achieve a higher sampling speed. This new OSHO IP core has been completed and was in the process of being debugged when the task was put on hold due to timing constraints introduced due to the unexpected new situation.

7.3.3 Switch from Waveforms Live to Backup GUI

As a result of having GUI progress stagnate the GUI team decided to start working on the SimplePlotter. We did not expect to have to create a whole new GUI but because the deadline for the project was fast approaching and progress on the GUI had stagnated we needed to make sure we had some way to plot data by the project delivery date. As such they made the SimplePlotter and stopped working on the Waveforms Live GUI.

7.3.4 Response of Project to COVID-19 Pandemic

Due to the COVID-19 Pandemic we experienced many delays in shipping for parts we ordered for the PCB. We also were no longer able to go to the ECE labs where we could have done PCB work with the proper tools. Additionally we could no longer meet in person with others to collaborate or seek help from experts. This forced us to find ways to collaborate digitally which are less effective and also makes it difficult to explain ideas fully at times.

7.4 OSHO PCB BOM and Solution Cost Breakdown

A complete listing of the OSHO PCB bill of materials can be found in Appendix C. The total cost of components per board at the ordered quantities (enough of each

component for three boards) is approximately $272, but will reduce if producing the solution in larger quantities. This corresponds to 46.3% of the total cost of our solution total of $587. The other costs in our solution include the Ultra96-V2 Development board ($249 or 42.4%) and the OSHO PCB itself ($66 or 11.2%). This summarized in the table and figure below.

Analog Front End Parts:	$272
Analog Front End PCB:	$66
Ultra96 Development Board:	$249
Total:	**$587**

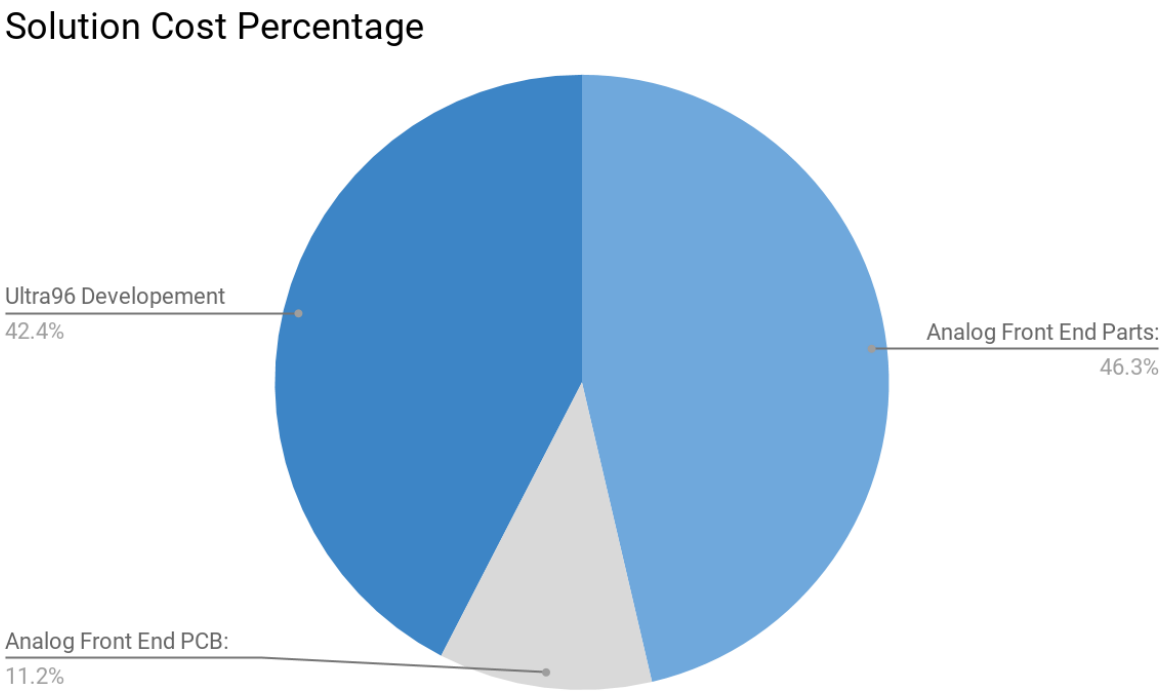


Figure 69: Cost Breakdown of Solution

7.6 Funds Spend

The total funds spent on the project can be broken down into the funds spent by Dr. Kaps and the funds spent by the team. The team spent a total of $260, and the total spent by Dr. Kaps total $2291. A further breakdown of these funds can be found below:

- Funds Spent by Dr. Kaps:
  - $1018 - Analog Front End Parts with enough for 3 boards and Spare
  - $516 - ADC Evaluation Board
  - $459 - Zedboard for Testing

    - $298- Two Ultra96 Development Boards
- Funds Spent by Team:
    - $225 - Three PCB Boards Professionally Printed
    - $35 - Frequency Synthesizer for Testing

7.7 Man-Hours Devoted to Project

The Table below summarized the total man hours that were devoted to the project over the course of two semesters. It is broken down into each main technical project aspect.

Project Area	***Total (# hours)***	***Approximate hours per week (Total/36 weeks)***
Analog Front-End Design	340	9
PCB Design	376	10
Firmware Development	370	10
Server and GUI Development	304	8
TOTAL	**1390**	

8. Lessons Learned

8.1 Additional Knowledge and Skills Acquired

Time allotment for component selection:

Once the overall high level circuit design is complete, one can feel a state of ease and think that the process is almost complete. However, this approach is completely incorrect. We learned that choosing the right components for every stage of the circuit is a very lengthy process. The price, performance, and availability of each chip/element needs to be analyzed. Not only this, the availability can change on a weekly basis so it is important to stay updated.

Importance of finalizing schematics before PCB design:

Tim had to do several iterations of PCB layout on KiCad due to the fact that the schematics were changed multiple times after meeting with our faculty supervisor. We should have met with him more often to ensure that the schematics were final before beginning PCB layout.

Reading Datasheets and Hardware User Guides:

Reading datasheets and hardware guides is a skill that is under-appreciated. There are many datasheets and guides that are incomplete/unclear and many that are extremely lengthy and detailed. Reading these documents and extracting the information we needed was a task that we were not prepared for before. However, after this experience, we have improved our ability to quickly find the information we need.

Other skills learned:

There are many other knowledge aspects/skills that we gained from this project including but that limited to: power architecture design, attenuator design, phase-locked loops, ferrite beads, chebyshev low pass filter design, and the signal conditioning process in general

The team also learned that it is very important to not overestimate your skills and abilities in some task that you have never done before, especially complex firmware design. Therefore, time taken for research and building up on background knowledge must also be taken into account when planning goals and the timeline for the project.

Also, although time can be saved by continuing a project, one must make sure they don't spend the saved time learning how to continue the work. It is sometimes best to aim for achieving a slightly less complex application so that you can actually achieve the goal. In addition, this year we tried to achieve a very complex GUI using tools that we have never worked with and as a whole felt very confused during the process. We had less than efficient/productive use of our time because we frequently had to look up different forums for how to do basic things. We were able to improve our typescript, HTML, CSS, python, and hardware/software codesign skills as a result working on the GUI of this project.

8.2 Team Experience

8.2.1 Teamwork and Team Environment

Through this project, everyone in the team learned valuable lessons about teamwork and the team environment. Since this was one of the longest team efforts for most of us, there were many situations that were a first time experience for us. Some of these lessons learned are highlighted below:

Shared vs. Individual responsibility:

Although shared responsibility seems more moral at first because everyone takes the blame for not completing a task, it is a sure way towards failure. If a task is assigned to the whole group, individuals think that someone else will complete it. Even if the task doesn't get completed, no individual feels any responsibility for the failure. Due to this, it is very important to break tasks down into manageable sections and assign them to individuals. Although the individuals can still ask the group for assistance, they are still responsible so there is extra motivation to complete that task.

Online communication can be better than physical meetings sometimes:

Even though we stayed on-topic in most weekly meetings, we realized that it was harder to get tasks accomplished while everyone is sitting in the same room. We learned that physical meetings should only be reserved for administrative tasks and all research and design should be accomplished in small pairs or individually. Any questions or concerns about specific topics were easily answered through email/text.

It is very important to frequently check in on teammates:

We learned that sometimes, it is hard for people to ask for help if they are stuck on a certain task. Due to this, it is important to frequently check in throughout the week and gauge not only their progress but mental state as well. This way, everyone stays engaged in the project and issues can be resolved quicker.

8.2.2 Project Management and Scheduling

Importance of scheduling meeting a week in advance:

It is very important to schedule and reserve a space for the next meeting during the current meeting. There were multiple times where we couldn't find a desirable location for a meeting because we did not reserve a place in advance.

Importance of sticking to Schedule:

It is very important to stick the deadlines as described in the Gantt chart. If one deadline is ignored. It has a snowball effect of delaying every other aspect of the project. It is also an easy way to be demotivated. Additionally we should have built in days for flexibility so if one aspect fell behind then we could catch up on certain days.

9. References

9.1 Overall Project References

[1] A. Wozneak, R. Nagpal, and R. Meruvia, “ECE - 492 Design Document.” 10-Dec-2018.

[2] 96Boards. (2019). Ultra96. [online] Available at: https://www.96boards.org/product/ultra96/ [Accessed 4 Oct. 2019].

[3] “Ultra96-V2 Development Board | Zedboard.” [Online]. Available: http://zedboard.org/product/ultra96-v2-development-board. [Accessed: 05-Dec-2019].

[4] TI, “50-Ohm 2-GHz Oscilloscope Front-end Reference Design TIDA-00826.,” TIDA-00826 50-Ohm 2-GHz Oscilloscope Front-end Reference Design | TI.com, Dec-2015. [Online]. Available: http://www.ti.com/tool/TIDA-00826. [Accessed: 05-Dec-2020].

[5] “HMCAD1511 Datasheet and Product Info | Analog Devices.” [Online]. Available: https://www.analog.com/en/products/hmcad1511.html. [Accessed: 12-Oct-2019].

9.2 Analog Front End References & Datasheets

[5] Texas Instruments, “LMH5401 8-GHz, Low-Noise, Low-Power, Fully-Differential Amplifier” LMH5401 datasheet.

[6] Texas Instruments, “LMH6401 DC to 4.5 GHz, Fully-Differential, Digital Variable-Gain Amplifier” LMH6401 datasheet.

[7] Texas Instruments, “CDCE62005 Four Output Clock Generator/Jitter Cleaner With Integrated Dual VCOs” CDCE62005 datasheet.

[8] “HMCAD1511 Datasheet and Product Info | Analog Devices.” [Online]. Available: https://www.analog.com/en/products/hmcad1511.html. [Accessed: 12-Oct-2019].

[9] Texas Instruments, “TPS2400 Overvoltage Protection Controller” TPS2400 datasheet.

[10] Texas Instruments, “TPS54327 3-A Output Single Synchronous Step-Down Switcher With Integrated FET” TPS54327 datasheet.

[11] Texas Instruments, “TPS7A92 2-A, High-Accuracy, Low-Noise LDO Voltage Regulator” TPS7A92 datasheet.

[12] Texas Instruments, “TPS7A7001 Very Low Input, Very Low Dropout, 2-Amp Regulator With Enable” TPS7A7001 datasheet.

[13] Texas Instruments, “TPS7A7001 Very Low Input, Very Low Dropout, 2-Amp Regulator With Enable” TPS7A7001 datasheet.

[13] Texas Instruments, “TPS63710 Low Noise Synchronous Inverting Buck Converter” TPS63710 datasheet.

[14] Texas Instruments, “TPS7A91 1-A, High-Accuracy, Low-Noise LDO Voltage Regulator” TPS7A91 datasheet.
[15] Texas Instruments, “CD74ACT251 8-Input Multiplexer, Three-State” CD74ACT251 datasheet.
[16] Texas Instruments, “OPA659 Wideband, Unity-Gain Stable, JFET-Input Operational Amplifier” OPA659 datasheet.
[17] NXP Semiconductor, “NVT2003/04/06 Bidirectional voltage-level translator for open-drain and push-pull applications” NVT2003/04/06 datasheet.
[18] Texas Instruments, “LMH6559 High-Speed, Closed-Loop Buffer” LMH6559 datasheet.
[19] Maxim Integrated, “DS1267B Dual Digital Potentiometer ” DS1267B datasheet.
[20] Texas Instruments, “OPAx376 Low-Noise, Low Quiescent Current, Precision Operational Amplifier e-trim Series ” OPAx376 datasheet.
[21] NXP Semiconductor, “NVT2008; NVT2010 Bidirectional voltage-level translator for open-drain and push-pull applications.” NVT2008 datasheet.
[22] Max Linear, “XRA1405 16-BIT SPI GPIO EXPANDER WITH INTEGRATED LEVEL SHIFTERS ” XRA1405 datasheet.

9.3 PCB References

[23] “A Practical Guide to High-Speed Printed-Circuit-Board Layout.” [Online]. Available: https://www.analog.com/en/analog-dialogue/articles/high-speed-printed-circuit-board-layout.html. [Accessed: 7-Jan-2020].
[24] “High Speed PCB Layout Techniques.” [Online]. Available: http://www.ti.com/lit/ml/slyp173/slyp173.pdf?ts=1588656387992. [Accessed: 7-Jan-2020].
[25] “SUCCESSFUL PCB GROUNDING WITH MIXED-SIGNAL CHIPS.” [Online]. Available: https://www.maximintegrated.com/en/design/technical-documents/tutorials/5/5450.html. [Accessed: 7-Jan-2020].
[26] “Grounding in mixed-signal systems demystified, Part 1.” [Online]. Available: http://www.ti.com/lit/an/slyt499/slyt499.pdf?ts=1588656599380. [Accessed: 7-Jan-2020].
[27] “Grounding in mixed-signal systems demystified, Part 2.” [Online]. Available: http://www.ti.com/lit/an/slyt512/slyt512.pdf?ts=1588656638497. [Accessed: 7-Jan-2020].
[28] “PCB Design Guidelines For Reduced EMI.” [Online]. Available: http://www.ti.com/lit/an/szza009/szza009.pdf?ts=1588656672557. [Accessed: 7-Jan-2020].
[29] “Grounding and Decoupling: Learn Basics Now and Save Yourself Much Grief Later! Part 1: Grounding.” [Online]. Available: http://www.ti.com/lit/an/szza009/szza009.pdf?ts=1588656672557. [Accessed: 7-Jan-2020].

[30] KiCad Org, “KiCad Documentation,” KiCad Docs. [Online]. Available: https://docs.kicad-pcb.org/. [Accessed: 05-Feb-2020].

[31] Yapo,, Ted “Towards a Multi-GHz Open-Source Sampling Oscilloscope,” Hackaday. [Online]. Available:https://cdn.hackaday.io/files/1672927157420928/ted-yapo-supercon-2019.pdf. [Accessed: 05-Feb-2020].

9.4 FPGA References

[32] “Intro to AXI Protocol: Understanding the AXI interface.” [Online]. Available: https://community.arm.com/developer/ip-products/system/b/soc-design-blog/posts/introduction-to-axi-protocol-understanding-the-axi-interface. [Accessed: 06-Dec-2019].

[33] “AXI4 Overview.” [Online]. Available: http://www.mrc.uidaho.edu/mrc/people/jff/EO_440/Handouts/AMBA Protocols/Xilinx Docs/XTECH_B_AXI4_Technical_Seminar.pdf. [Accessed: 05-Dec-2019].

[34] “7 Series FPGAs SelectIO Resources,” Xilinx, 08-May-2018. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug471_7Series_SelectIO.pdf. [Accessed: 05-Dec-2019].

[35] M. Defossez, “Serial LVDS High-Speed ADC Interface,” Xilinx, 20-Nov-2012. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp524-serial-lvds-adc-interface.pdf. [Accessed: 05-Dec-2019].

[36] M. Defossez, N. Sawyer, “LVDS Source Synchronous DDR Deserialization (up to 1,600 Mb/s),” Xilinx, 22-Jul-2016. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp1017-lvds-ddr-deserial.pdf. [Accessed: 05-Dec-2019].

[37] Open.uct.ac.za. 2015. [online] Available at: <https://open.uct.ac.za/bitstream/handle/11427/20046/thesis_ebe_2015_kemp_dayne_hilton.pdf?sequence=1&isAllowed=y> [Accessed 5 February 2020].

9.5 Software References

- https://matplotlib.org/3.2.1/api/index.html
- https://matplotlib.org/3.2.1/api/animation_api.html
- https://github.com/Digilent/waveforms-live
- https://www.codecademy.com/learn/paths/web-development
- https://matplotlib.org/3.1.1/gallery/misc/cursor_demo_sgskip.html
- https://docs.python.org/3/library/tk.html
- https://pythonbasics.org/tkinter-button/

- https://stackoverflow.com/questions/2408560/python-nonblocking-console-input

10. Appendix A: Project Proposal (ECE 492)

Open Source High-Speed Oscilloscope (OSHO)

Project Proposal

Team Members:
Timothy Bullock, Afnan Ali, Evan Hoffman, Umair Aslam, Zaeem Gauher

Faculty Advisor:
Jens-Peter Kaps

ECE492-001

Date of Submission: October 11th, 2019

George Mason University
4400 University Dr, Fairfax VA 22030

1. Executive Summary

High-speed oscilloscopes are very useful for many applications where electrical signals need to be measured. These tools can be used to measure, analyze, and display the waveforms of high frequency and low power analog signals with impressive precision. However, the downside of existing high frequency, commercially available oscilloscopes is that they are extremely expensive. Today, typical commercially available high-end oscilloscopes that have a bandwidth of 500-800 MHz cost upwards

of $6000. Furthermore, these devices can be quite difficult to use, and are surprisingly limited in certain aspects. For instance, downloading the captured data off of these devices for external processing in Matlab or Python is quite slow, and their built-in analog-to-digital-converter (ADC) cannot be synchronized to an external clock. Lower cost alternatives, such as entry-level commercial oscilloscopes or open source oscilloscopes, typically only offer bandwidths of up to 100 MHz before their prices significantly increase into the range of their more expensive counterparts.

To overcome this, our senior design group will be designing an open source high-speed oscilloscope that will provide a low-cost alternative to commercially available oscilloscopes, while also provide a higher-performance and feature rich alternative to the existing open-source solutions. This solution will be targeted towards academic and hobbyist communities, where funding is often a limitation, but high-sampling speed and bandwidth are needed [1]. Our solution will feature high-bandwidth, a high sample-rate ADC, a responsive and intuitive web-based graphical user interface (GUI), an ADC that can be synchronized to an external clock input, and an external trigger input.

This system will have three main components: A multiprocessor system on chip (MPSoC) development board which includes an FPGA and ARM-based processor, a high-speed analog front end with a custom PCB, and a web client GUI. The high-speed analog front end will provide the interface between the analog signal being analyzed and the FPGA, converting the signal into digital captured data. The MPSoC will buffer, route, and process the captured data, then host it on a web server for easy display and access by the web client. Lastly, the web client will provide the user with an intuitive and responsive graphical user interface to control the system and view the captured waveforms. Each of these components will be designed around the use of an 8-bit, one giga-samples-per-second (GSPS) ADC.

We plan to keep the overall cost of the product under $600, which is significantly less than the $6000 cost of other oscilloscopes at this performance level. The name we have chosen for this device is Open Source High-Speed Oscilloscope, or OSHO for short. Dr. Jens-Peter Kaps will be providing guidance on this project as he has experience guiding teams designing high-speed capture devices such as the GMU Logic Analyzer and the previous attempt at this project [2].

2. Problem Statement

2.1 Motivation and Identification of Need

Digital oscilloscopes are extremely useful tools for many engineering applications where electrical signals need to be measured and analyzed. Digital oscilloscopes “enable the user to debug, visualize and measure various signals,” and are an essential part of any engineering lab or project [3]. Yet, in many applications such as RF design, the signals that are being analyzed are too high frequency to be measured with standard low-cost oscilloscopes. In these applications, high performance oscilloscopes

with sufficient bandwidth and sampling-speeds are needed. The problem with this is that oscilloscopes with bandwidths greater than 500 MHz are extremely expensive. Even moderate performance oscilloscopes with bandwidths greater than 200 MHz can cost several hundreds to thousands of dollars. On top of this, even at these high prices, many of the commercially available devices can be limited in certain usability aspects and features. For example, downloading the captured data from these devices for external processing can be quite slow, and their built-in ADC cannot be synchronized to an external clock signal. Therefore, to overcome these limitations, it is our project's motivation to create a low-cost, open source, and high-speed alternative to existing oscilloscopes.

2.2 Market Review

As our project is bridging the gap between high-speed digital oscilloscopes and limited open source oscilloscopes, it is advantageous to first analyze these markets. As previously stated, typical commercially available oscilloscopes that have a bandwidth of 500-800 MHz cost upwards of $6000, can be quite difficult to use, and surprisingly limited in functionality [2]. The following table shows a sample of the cheapest commercially available oscilloscopes with a bandwidth above 500 MHz. Clearly, the price point of these devices is an obstacle to overcome in settings where funding is limited.

Table 1: Cheapest Oscilloscopes with a Bandwidth of 500MHz

Model	**Device Picture**	**Bandwidth (MHz)**	**Sampling Rate (GSPS)**	**# of Analog Channels**	**Sampling Resolution (Bits/Sam.)**	**Price ($)**

Rigol DS4052 [4]		500	4	2	12	5,999
Tektronix MDO3052 [5]		500	2.5	2	8	9,570
PicoTech PicoScope 6000 Series [6]		500	5	4	8	6,595
Keysight DSOX3052 A [7]		500	4	2	8	9,439

It should be noted however, that nearly all of the offerings at this bandwidth contain higher sampling speeds than we plan to offer. If you filter oscilloscopes by sampling speed, there are cheaper devices, but their bandwidth is typically limited to just 100 to 200 MHz. A sample of mainstream 1 GSPS oscilloscopes with the highest available bandwidth is shown in Table 2 below.

Table 2: Current 1 GSPS Oscilloscopes with Highest Bandwidths

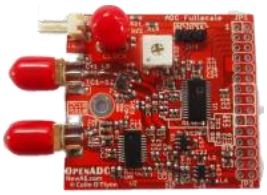
Model	**Device Picture**	**Bandwidth (MHz)**	**Sampling Rate (GSPS)**	**# of Analog Channels**	**Sampling Resolution (Bits/Sam.)**	**Price ($)**

Rigol DS1104Z-S Plus [8]		100	1	4	12	699
Tektronix DPO2022B [9]		200	1	2	8	2,520
PicoTech PicoScope 2000 Series [10]		100	1	2	8	679
PicoTech PicoScope 5000 Series [11]		200	1	2	8 to 16 (depending on mode)	1,945

The other market that needs to be analyzed is the market for open source oscilloscope hardware. Typically, the hardware from open-source oscilloscope projects have an average price point between $150 and $300, making it a much more affordable option. However, when even considering the fastest of these devices, their speeds come nowhere close to the specifications that our device will be designed to offer. A sample of these devices is shown below in Table 3. Our closest competitor would be Scopefun, which provides a bandwidth of 100 MHz and a top sampling speed of 500MSPS.

Table 2: Current Open Source Alternative Oscilloscopes

Model	**Device Picture**	**Bandwidth (MHz)**	**Sampling Rate (MSPS)**	**# of Analog Channels**	**Sampling Resolution (Bits/Sam.)**	**Price ($)**

ScopeFun [12]		100	500 (Single Channel) 250 (Dual Channel)	2	10	650
BitScope 10 [13]		100	40	2	8 or 12 (depending on mode)	245
Digilent OpenSCope MZ [14]		2	6.25	2	12	149
OpenADC [15]		40	105	1	10	137*

**Plus the cost of an FPGA Development Board*

3. Approach

3.1 Problem Analysis

3.1.1 Problems to be Addressed

In order to provide a successful solution to the problems described above, the designed system will have to overcome three main problems: The device hardware should be low-cost, the device should be high-performance with a high bandwidth and sampling speed, and the device should have features that other low-cost oscilloscopes do not conventionally have.

3.1.2 High Commercial Cost

The primary thing that needs to be overcome by our solution is the high cost of existing high speed oscilloscopes. As shown in Section 2.3, typical commercial digital oscilloscopes are extremely expensive, and open source oscilloscopes are functionally limited. Clearly, if our device can achieve a 500 MHz bandwidth with 1 GSPS at a final design price of under $600 (including the cost of an Ultra96 development board), our device would be a market leader. It would provide the same high bandwidth as extremely expensive oscilloscopes at a lower cost while being the fastest open source oscilloscope platform. This price will put our solution at an excellent price point given its significantly higher specifications. In order to accomplish this, we will be very selective with our design and component choices.

3.1.3 Bandwidth and Sampling Speed

The next thing that needs to be analyzed is the performance aspect of our solution. As shown above in section 2.3, there are low-cost alternatives that offer moderate performance, but nothing close to the speed required for many high-frequency applications. At a bandwidth of 500 MHz with a sampling rate of 1 GSPS, our solution will provide a good compromise between price and speed. At this speed, the system will be “powerful” enough to measure signals from high-speed applications such as RF signal analysis in the VHF to UHF range, or EM side-channel analysis. If we were to increase the sampling speed beyond this, it would substantially increase the cost of the device due to the unavailability of cheap ADCs with a higher sampling speed.

3.1.4 Special Features and Ease of Use

Finally, the last hurdle that should be overcome by our solution is the lack of particular features in low-cost and even some high-cost oscilloscopes. This includes three main things: the inability to synchronize the sampling clock to an external clock, the lack of a quick and easy transfer of sample data to an external computer for processing, and finally the lack of an external trigger input. All of these problems will be addressed in our solution as they do not add too much to the final price of the system, and they will make this device an extremely powerful tool.

3.2 Our Preferred Approach

3.2.1 A Modular Solution

To successfully overcome the cost, speed, and feature limitations associated with existing commercial and public-domain oscilloscopes, our team will be providing a modular solution that minimizes the cost of the various system components while still providing the features discussed earlier. Our preferred

approach to tackling this is to split the overall system into three modular subsystems: A web-based GUI subsystem, a processing subsystem, and an analog front-end subsystem. This is summarized in the External System Diagram below (Figure 1).

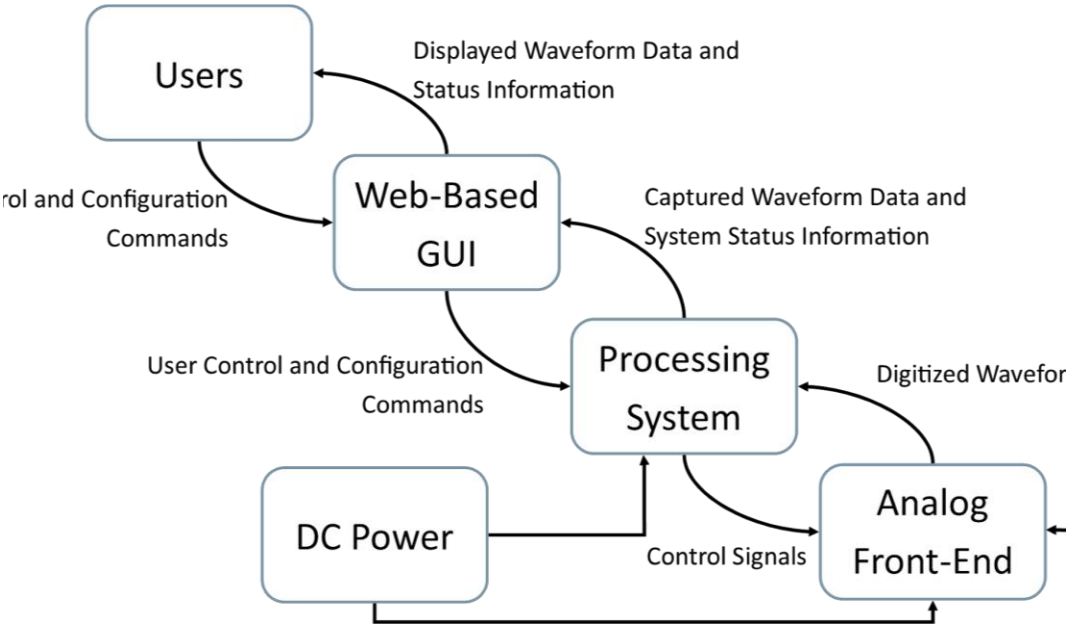


Figure 1. External System Diagram

3.2.2 The Analog Front-End

The analog front-end subsystem will primarily consist of the analog circuitry to precondition the incoming analog signals so that they may be optimally digitized by the ADC. The tasks that will be performed by the conditioning circuitry will include: attenuation, anti-aliasing filtration, variable gain amplification, coupling selection (AC or DC), DC offset selection, circuit overvoltage protection, and ADC clock generation/synchronization. Configurable aspects of this system such as DC offset will be configured through SPI commands from the processing subsystem. Once the analog inputs are conditioned properly, they will then be digitized by the ADC and sent to the processing subsystem. This front-end circuitry will be routed on a custom high-speed PCB that will be designed by our team. This board will be able to interface with the processing system through high and low speed mezzanine connectors.

3.2.3 The Processing Subsystem

The processing subsystem will consist of a MPSoC development board which includes programmable logic in the form of an FPGA as well as an ARM-based processor. The specific development board that will be used for this application will be the Avnet Ultra96-V2 which uses a Xilinx Zynq UltraScale+ MPSoC ZU3EG A484, has 2GB of LPDDR4 memory, and provides essential integrated peripherals such as USB3.0, an SD card slot, WiFi, and Mini DisplayPort [16]. The programmable logic portion of this board will be used in

conjunction with custom intellectual property (IP) blocks that will buffer the incoming raw digital data from the ADC and transform it to a standardized data packet format. These packets will then be sent to the system's main memory where processing can be conducted through an ARM processor that hosts a linux-based web server. The end user will be able to view and download waveform data and system status information as well as send configuration commands through this webserver.

3.2.4 The Web-Based GUI

The final foundational aspect of our preferred approach is a web-based GUI subsystem that will act as a client to the web server running on the Ultra96 board. This subsystem will act as the primary interface between the user and the overall system. This subsystem will allow the user to enter system configuration commands (such as toggling between AC/DC coupling, configuring waveform triggers, etc) and download/display captured waveform data. This custom user interface should be responsive, intuitive, and effectively display captured waveform data. This aspect of the project will likely be programmed in Angular, and implemented incrementally, providing basic features at first, but adding more advanced features as time permits.

3.2.5 Benefits of this Approach

Providing a modular design proves to be the optimal solution to the problem because it will minimize cost while providing excellent analog capture performance. Additionally this approach will also provide a good basis for further open-source development.

This modular solution optimizes low-cost for multiple reasons. Much of the hardware cost will be absorbed by the fact that an external computer will be utilized for user interface. Furthermore, the front-end circuitry will be designed with cost-effective parts. For instance, the chosen ADC for this project is the HMCAD1511, which offers excellent performance for its price [17]. Additionally, the effective price of the system is reduced if a compatible FPGA development board is already owned by the end user.

As stated earlier, this approach ensures that the system will be an excellent platform for future open source development. It will consist of open source software as well an open source development board, allowing the end users to customize it to their needs. The fact that the analog front-end is separate from the development board means that the front-end board could be used with other compatible MPSoC development boards (with minimal firmware porting). Additionally, the GUI for this system can also be customized and improved by users in an open source fashion.

3.3 Alternative Approaches

3.3.1 Overview

There are many possible solutions to the problem of providing a low-cost, high-speed, and feature-rich oscilloscope. Although the approach discussed above is the one that was determined to provide the best compromise between cost, performance, and features, it is still important to consider some alternative approaches. This ensures that our preferred approach is the optimal solution and provides us with backup approaches in case problems arise with our preferred approach. Alternative approaches that were considered are: using multiple ADCs, incorporating the MPSoC onto the same board as the analog front-end, and incorporating a display and physical controls as part of the device hardware.

3.3.2 One vs. Multiple ADCs

In the development of our solution, having two analog input channels was listed as a key requirement as this provides a much more useful device. However, the issue with this is that there is no low-cost ADC that supports two channels at 1GSPS each. According to our preliminary research, the Analog Devices HMCAD1511 ($64) is the only low-cost ADC that supports 1GSPS [17]. This device can support multiple channels, but does not provide 1GSPS for each channel. Instead, the sampling rate is reduced immensely as more channels are utilized. This raised the question of whether multiple ADCs should be used to provide support for multiple analog inputs. It was concluded that due to cost limitations, this was not feasible. Due to this, we chose to utilize only one HMCAD1511 ADC, but offer a mode where the user can configure the analog front-end to handle two inputs at a lower sampling speed of 500MSPS. Additionally, data bandwidth issues were also cited as a reason to use lower sampling speeds with multiple input channels. However, if this proves to be overly complex and unexpectedly expensive, using separate ADCs for each channel may be reconsidered.

3.3.3 Using a MPSoC Development Board vs. a Single Board Solution

As the hardware for the Utra-96-V2 development board is open source, it was questioned whether or not this hardware should be incorporated into the front-end custom PCB to provide a more portable, single board solution. However, this was rejected in favor of using a development board that interfaces with the analog front-end through mezzanine connectors. This is because of two primary reasons. The first being that this provides unnecessary complexity to the hardware development and adds to the cost of production. Secondly, providing a single board solution would be a drawback to our target market of academics and hobbyists as they might only require the front-end device without our firmware for

their specific application. Furthermore, they might prefer the multi-board solution so that the Utra-96 V2 remains reusable for different applications.

3.3.4 A Web-Based GUI vs. Physical Controls and On-Device Display

The last major alternative approach that was debated was the use of a graphical user interface vs physical controls and incorporated display such as those in traditional bench oscilloscopes. It was decided that the web-based GUI solution should be favored over physical controls and on-device display. This was not only chosen because it minimizes the cost of the device, but also because it allows us to continually add more advanced controls to the device though software updates. Additionally, most users of this device would likely own a network capable computer which has a nicer display than any low-cost physical display we could include in our device. Furthermore, if a network connected device is used as the interface for this oscilloscope, it would ease the process for downloading captured data for external processing. However, the physical controls/display approach may prove a useful alternative for specific device controls for which a software approach may be too inconvenient.

3.4 Introduction to Background Knowledge

3.4.1 Overview

In order to further justify the technical choices of our system, and effectively describe the system architecture and design, it is beneficial to first provide an overview of the background knowledge required to understand the various aspects of our solution. A brief overview of each aspect of our solution is provided below.

3.4.2 Oscilloscope Specifications

Although most oscilloscopes are used for a similar general purpose, their specifications greatly limit the applications in which they can be utilized. Some of these important specifications are explained in further detail below:

Bandwidth:

The bandwidth of an oscilloscope dictates the maximum frequency range that can be accurately measured by the device. High-speed, serial communication, and other complex signal applications require bandwidths of 500MHz or greater for accurate measurement.

Rise Time:

The rise time specification is very important for digital circuit applications. Rise time is defined as the time it takes for a signal to rise from 10% to 90% of its final value. This time can also be related to the bandwidth in the following manner: $Rise\ Time = 0.35 \div Bandwidth$ [18]. An oscilloscope

should have a fast enough rise time to capture rapid transitions in signals such as square waves and pulses in an accurate manner.

Sample Rate:

The sample rate of an oscilloscope (measured in samples/second) defines how often the device samples the signal. According to the Nyquist-Shannon Sampling Theorem, the sampling rate needs to be twice as fast as the highest frequency component of a signal in order to avoid aliasing. Thus, if a sampling rate of 1GSPS is used, the maximum input frequency should be limited to 500MHz.

Channel Resolution:

The resolution of the oscilloscope defines the granularity of the signal. If the ADC in the oscilloscope has an 8-bit resolution, this translates to $2^8 = 256$ digitized levels that each analog sample will be translated to [18]. An ADC with a resolution of 8 bits is sufficient for a low-cost oscilloscope application.

3.4.3 High-Speed Analog Front End

The front end signal measurement chain consists of many different analog subcomponents. Together, these elements transform the input signal into digital data that the back-end firmware can then process. These subcomponents are listed below:

Attenuator:

The attenutor's primary function is to reduce voltage, dissipate power, and improve impedance matching between devices such as amplifiers. Attenuators can be configured to adjust the amount of attenuation manually. The utilization of an attenuator is crucial in the front-end as it provides sufficient input amplitude adjustment to prevent saturation for large signal swings.

Low-Noise Amplifier (LNA):

Low noise amplifiers are used to amplify very low-power signals without negatively affecting the signal-to-noise ratio. By using a LNA close to the input source, the effects of noise in the following stages of the front-end stage can be greatly reduced. To ensure the maximum transfer of power from source to amplifier, the source impedance should match the input impedance of the LNA. This can be achieved through the attenuator mentioned earlier.

Variable Gain Amplifier (VGA):

A VGA is used to amplify input signals based on the gain parameter. The advantage of using a VGA is that the gain can easily be controlled through an interface such as SPI to ensure that the output fits within the full-scale

input range of the ADC. This prevents clipping of the digital output waveform.

Anti-Aliasing LPF:

An anti-aliasing low pass filter's function is to remove unwanted high frequency components from the input signal. This filter is crucial to ensure that the input to the ADC has a maximum frequency of half the sampling rate. This prevents aliasing as dictated by the Nyquist-Shannon Theorem.

Phase-locked loop(PLL):

A phase-locked loop is a voltage driven oscillator that receives a reference signal and outputs a signal with either a matched or multiplied frequency compared to the reference. The PLL also acts similar to a bandpass filter to remove high frequency jitter as well as low frequency VCO jitter from the clock signal [19]. The PLL will allow for the use of the FPGA clock in order to provide the clock input to the ADC. Similarly, it can also be utilized to sync the ADC clock with an external clock when that option is selected.

Analog to Digital Converter(ADC):

The ADC is one of the most crucial features of the oscilloscope as it determines the sampling rate, resolution, as well as bandwidth. The ADC receives analog signals from the signal conditioning stage which includes attenuation, amplification, and filtration. The analog signal is then sampled and digitized before being transferred to the back-end firmware for transferring the data into memory.

3.4.4 High-Speed PCB Design

When electrical components operate at high frequencies, the circuit performance becomes heavily dependent on the layout of the PCB. Certain aspects of PCB design such as trace lengths, thermal information, and component location are of increased importance for high frequency applications. Furthermore, power-supply bypassing needs to be implemented in order to minimize noise. This is achieved through the use of capacitors attached across the op-amp power supply and ground. Other aspects in high frequency applications which could cause major problems are parasitic capacitances of components, non-continuous ground plane configuration, long parallel traces, among other indirect effects. These will have to be taken into account when the PCB for the analog front end was designed.

3.4.5 FPGA Programmable Logic

The programmable logic on the FPGA refers to an array of interconnected digital subcircuits that can be configured for specific applications. This allows for a high level of flexibility. For the purpose of this project, the programmable logic will be configured to buffer the raw serialized ADC output data and store it into

the MPSoC's main memory. With this specific chip, the Advanced eXtensible Interface (AXI) will then be utilized to serve as an interface between the programmable logic and the ARM processor portion of the MPSoC.

3.4.6 Web Server

The Ultra96 board allows multiple ways to be connected to a network capable computer. Either the board can be connected to through WiFi or when plugged into a computer via USB, the Ultra96 is recognized as a network card. These network connections will easily allow a web server to be hosted on the Ultra96. A Web server is a program that uses established networking protocols to host files and data, as well as process and service client requests from networked computers. The Ultra96 comes with a pre-built Jupyter Notebook web server on board that can be configured using Python 3. However, if this web server is not able to meet our needs for the GUI, then we will use a different, more advanced web server such as Apache. For our system, a web server will host the data from the ADC and accept requests to configure the system.

3.4.7 Web Client & Graphical User Interface(GUI)

A web client is any sort of interface that allows you to communicate to a web server through a network connection. Having a web client with a GUI together allows for quick and simple communication between the user and the ultra96 without having to interface with the command line. A key component of this project is for the system to be very user-friendly and responsive. The web client and GUI created for this project will be made using Angular 2 for its responsiveness and compatibility with preexisting waveform simulators. Angular should allow the user to interact with the waveform through the GUI in realtime and watch it update with minimal delay.

3.5 Requirements Specification

3.5.1 Mission Requirements:

- The project shall design an oscilloscope that is an open source, low-cost alternative to commercially available oscilloscopes, and a performance, feature rich alternative to existing open-source oscilloscopes.
- The project shall design a custom high-speed PCB that will easily interface with an Ultra96-V2 development board, as well as develop the supporting firmware and graphical user interface for the device.

3.5.2 Operational Requirements:

- *Input/Output Requirements*
  - The device shall have at least two analog input channels, one external clock input, and one external trigger input.

- The system will receive control and configuration commands as well as be able to responsively display captured data through a web client with an intuitive and responsive GUI.

- *External Interface Requirements*
  - The device will provide support for 1x and 10x passive probe inputs (50Ω).
  - Bayonet Neill–Concelman (BNC) connectors shall be used for the analog inputs, external clock input, and external trigger inputs.
  - The system shall interface with a network capable computer through USB3.0 or WiFi.
  - The system shall receive power from an external 12V DC power supply.

- *Functional Requirements*
  - The analog-to-digital converter (ADC) shall sample one input channel at 1 GSPS or two channels at 500 MSPS.
  - The device will be able to measure analog inputs with a maximum input voltage of ±10V.
  - The input analog circuitry shall achieve a 500 MHz bandwidth.
  - The ADC shall be able to be configured to sample using either the FPGA clock or an external clock input (between 30 MHz and 1 GHz).
  - The ADC output sample resolution shall be no less than 8 bits.
  - The system's data capture shall have the ability to be triggered using both configurable edge triggers as well as a configurable external trigger input.

- *Technology and System-Wide Requirements*
  - The front-end device shall use a single 1GSPS ADC chip.
  - The ADC data shall be processed and hosted on an onboard Linux web server using a Xilinx Zynq UltraScale+ multiprocessor systems-on-chip (MPSoC) aboard the Ultra96 Board.
  - The analog front-end custom PCB should interface with the Ultra96 Board for data processing.
  - Target FPGA development board shall have device driver firmware for interfacing with the ADC, and routing and storing ADC sample data in a memory device.
  - Front-end programmable devices will be controlled using the Serial Peripheral Interface (SPI) or other serial protocol.
  - The custom high-speed PCB and Ultra96 devices will interface with each other via the Ultra96's high-speed and low speed mezzanine connectors.
  - The device should be low-cost ($600 or less).

4. System Design

4.1 System Functional Decomposition

4.1.1 Level Zero

In order to effectively design the system architecture of a system, it is best to start with a functional decomposition of a system so that the system functions can be related in a hierarchical manner. When functionally decomposing a system, it is best to start at a high level, and work downwards. Level zero provides a top level overview of the overall solution. It shows the overall system inputs and outputs. For our device, this is shown below in figure 2. It shows that the overall system will take in two analog inputs, an external clock input, an external trigger output, user commands, and DC power input. The system outputs status information and digitized waveform data.

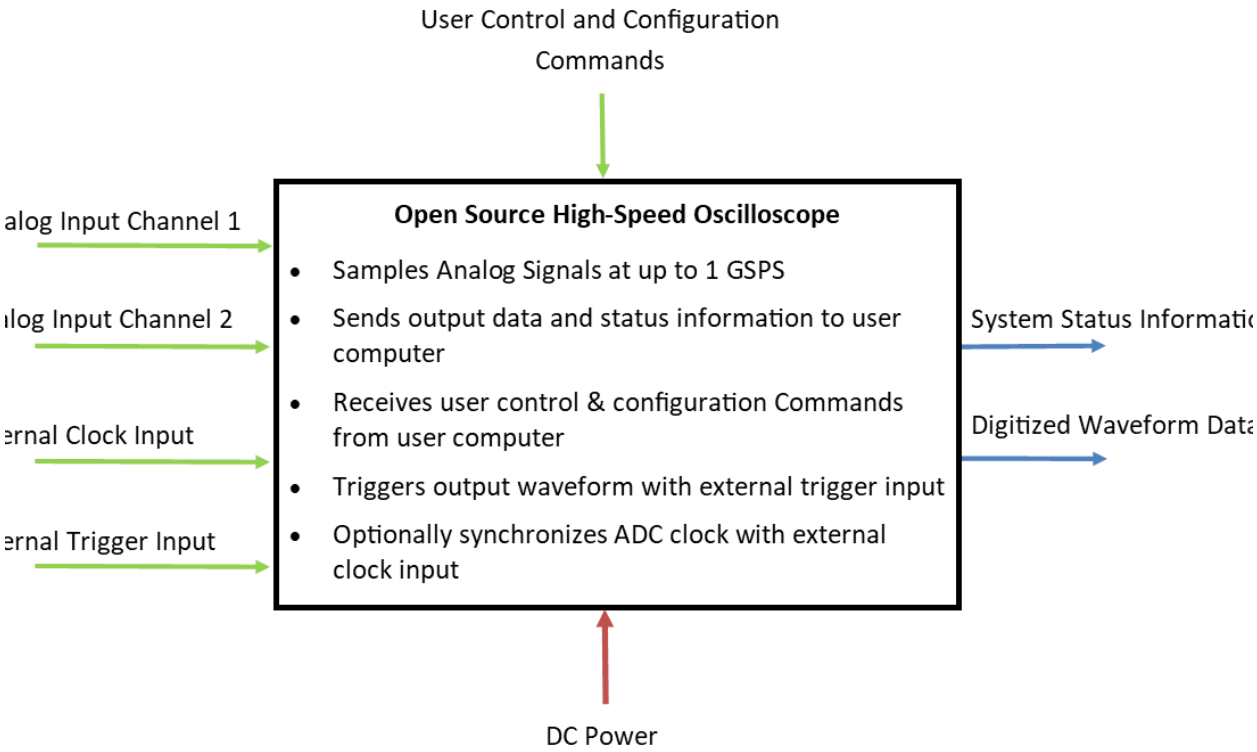


Figure 2. Level Zero Functional Architecture Block Diagram

4.2.2 Level One

After the system is understood at the highest input/output level (level zero), the next step of functional decomposition is to identify the top level functions of the system. For our system this would include analog signal preconditioning, analog to digital conversion, ADC clock selection/generation, raw data buffering and routing, processing and data hosting, and finally, display and interface processing. This is summarized in the level one diagram shown below (Figure 3).

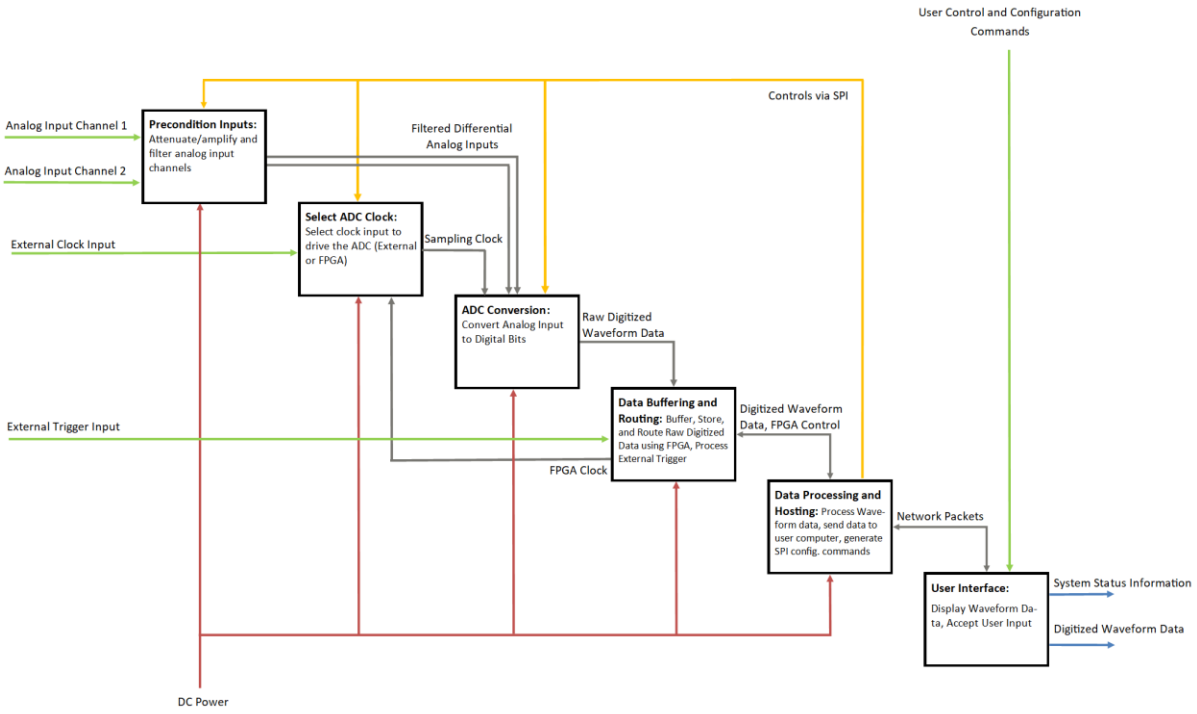


Figure 3. Level One Functional Architecture Block Diagram

4.2.4 Level Two

Once the functionality of the system is understood at the functions' level (level one), the next step to defining the system design is to take each of these top level functions and decompose them into their subprocesses. This is done for each of the top level functions shown in the level one functional architecture diagram above (Figure 3). It is worth noting that throughout the completion of the detailed system design that these level two subprocess blocks are subject to slight alterations.

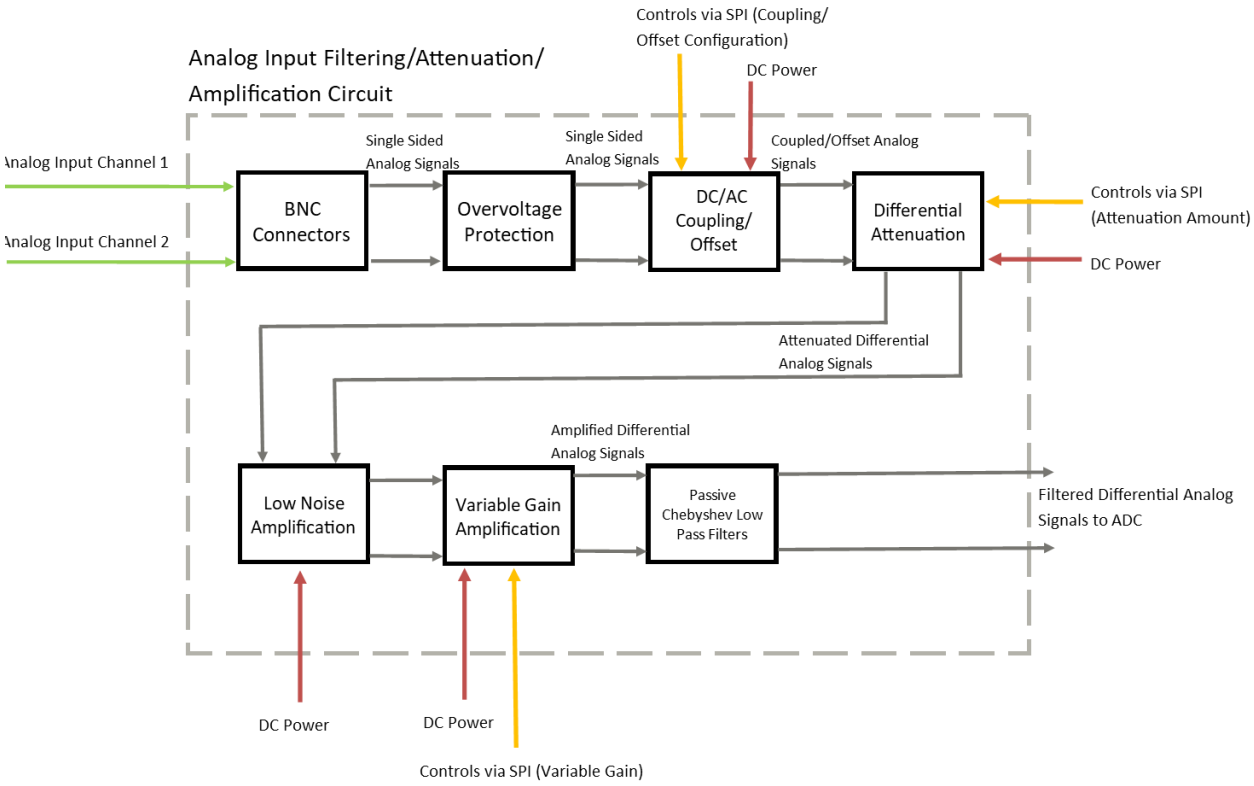


Figure 4. Level Two Functional Architecture Block Diagram - Analog Signal Input Preconditioning

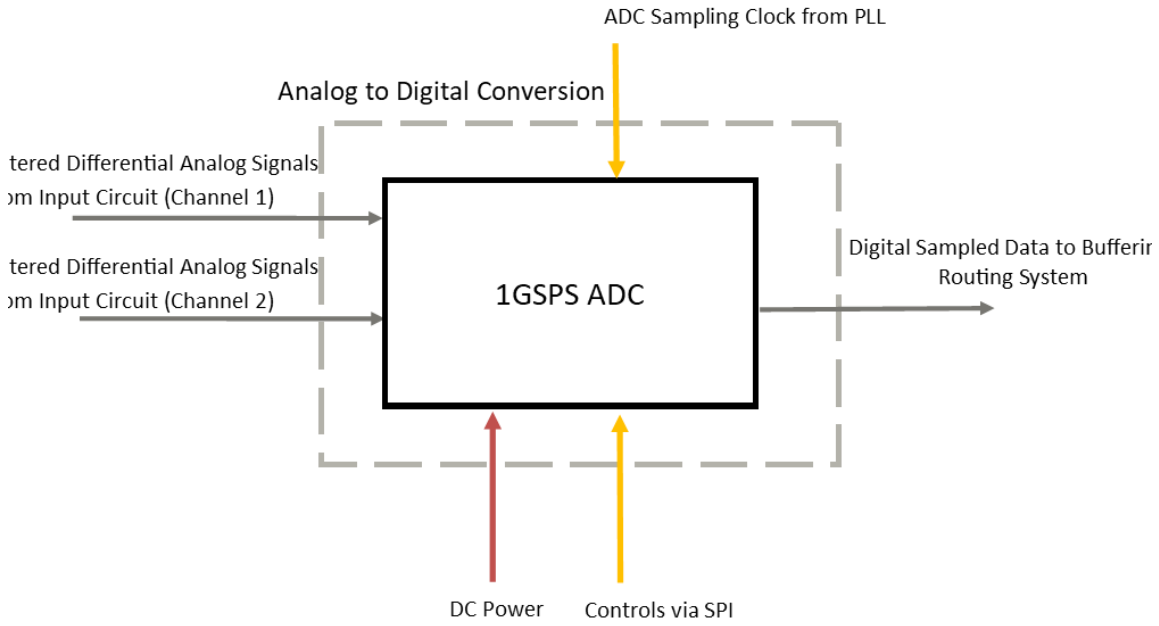


Figure 5. Level Two Functional Architecture Block Diagram - Analog Digital Conversion

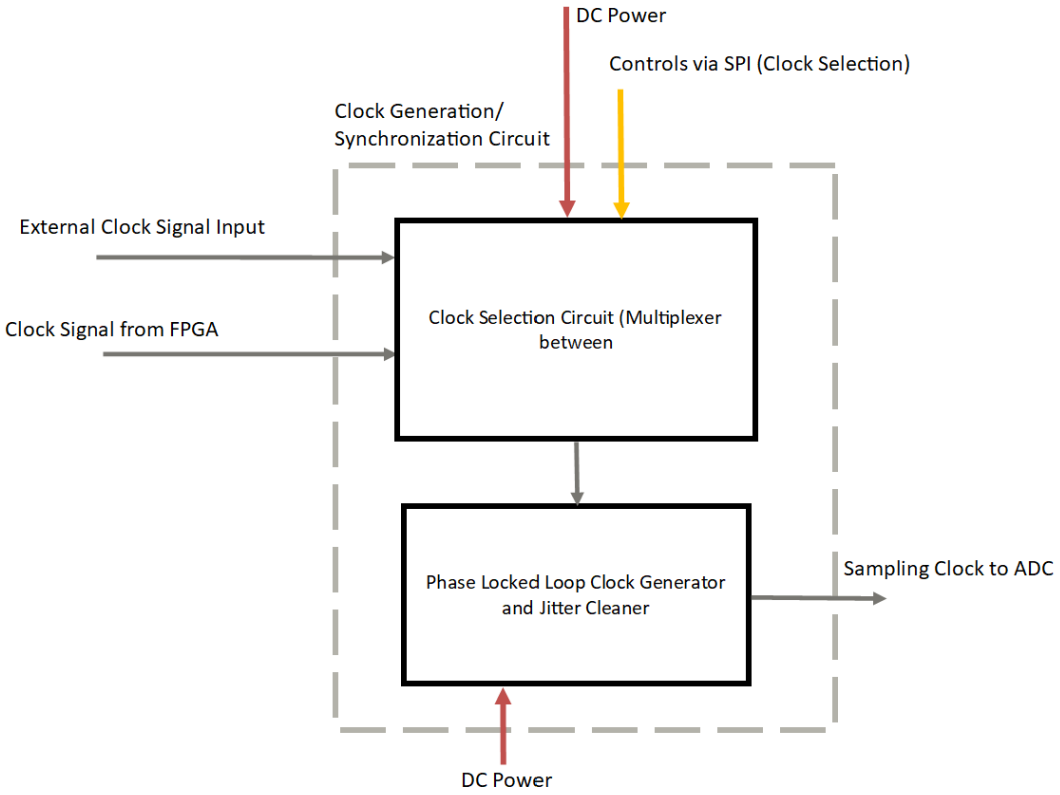


Figure 6. Level Two Functional Architecture Block Diagram - ADC Clock Generation

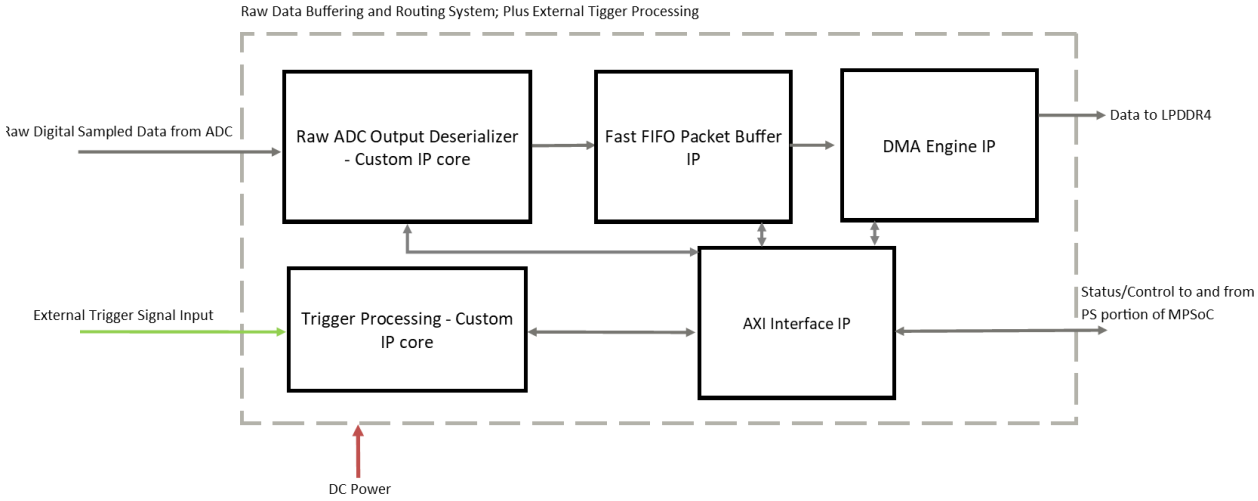


Figure 7. Level Two Functional Architecture Block Diagram - Raw Data Buffering and Routing

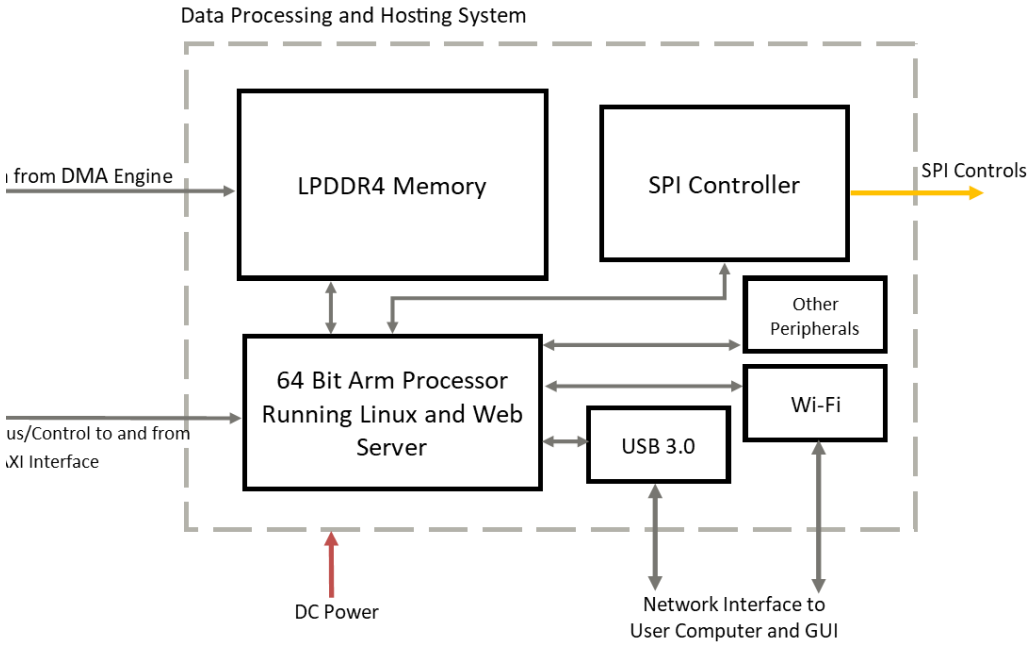


Figure 8. Level Two Functional Architecture Block Diagram - Processing and Data Hosting

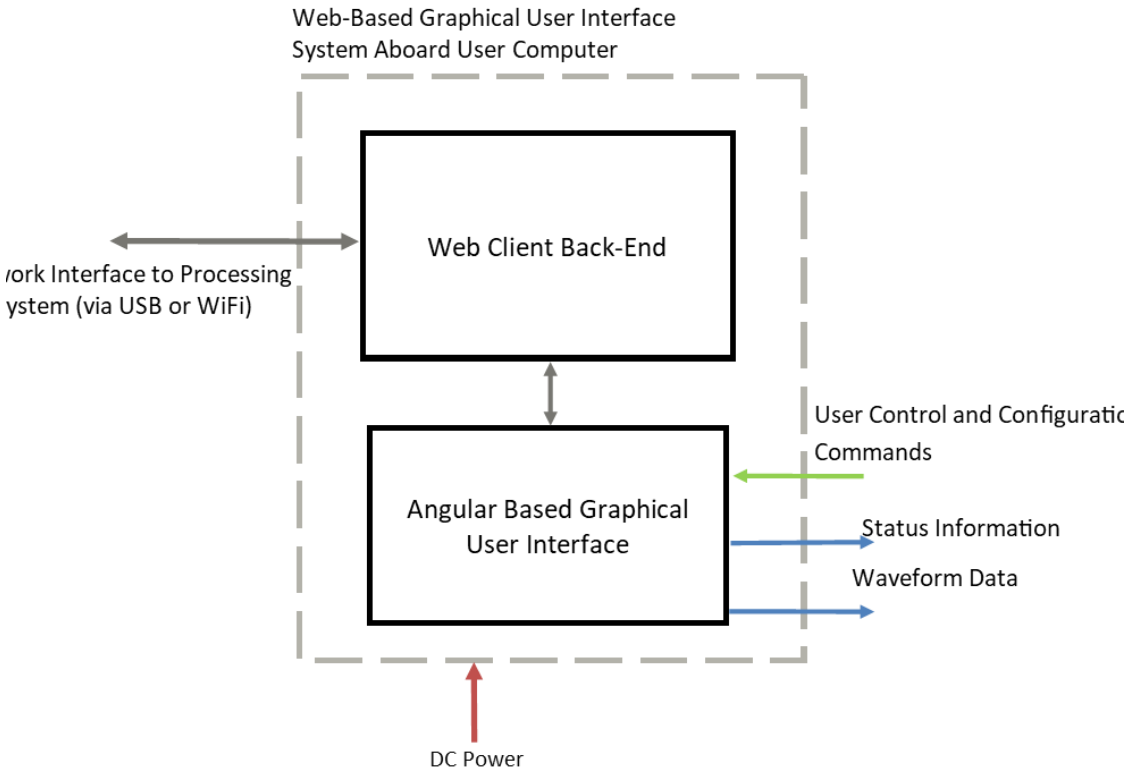


Figure 9. Level Two Functional Architecture Block Diagram - User Interface

4.2 System Architecture

4.2.1 Physical Architecture

The physical architecture consists of a hierarchical diagram that shows the main configuration items that make up the system. This includes major hardware and software components. With the exception of the Ultra96 development board, all of these components are currently generic as a specific bill of materials has not yet been determined. This serves as a hierarchical overview of the major physical resources that will be required to implement our solution.

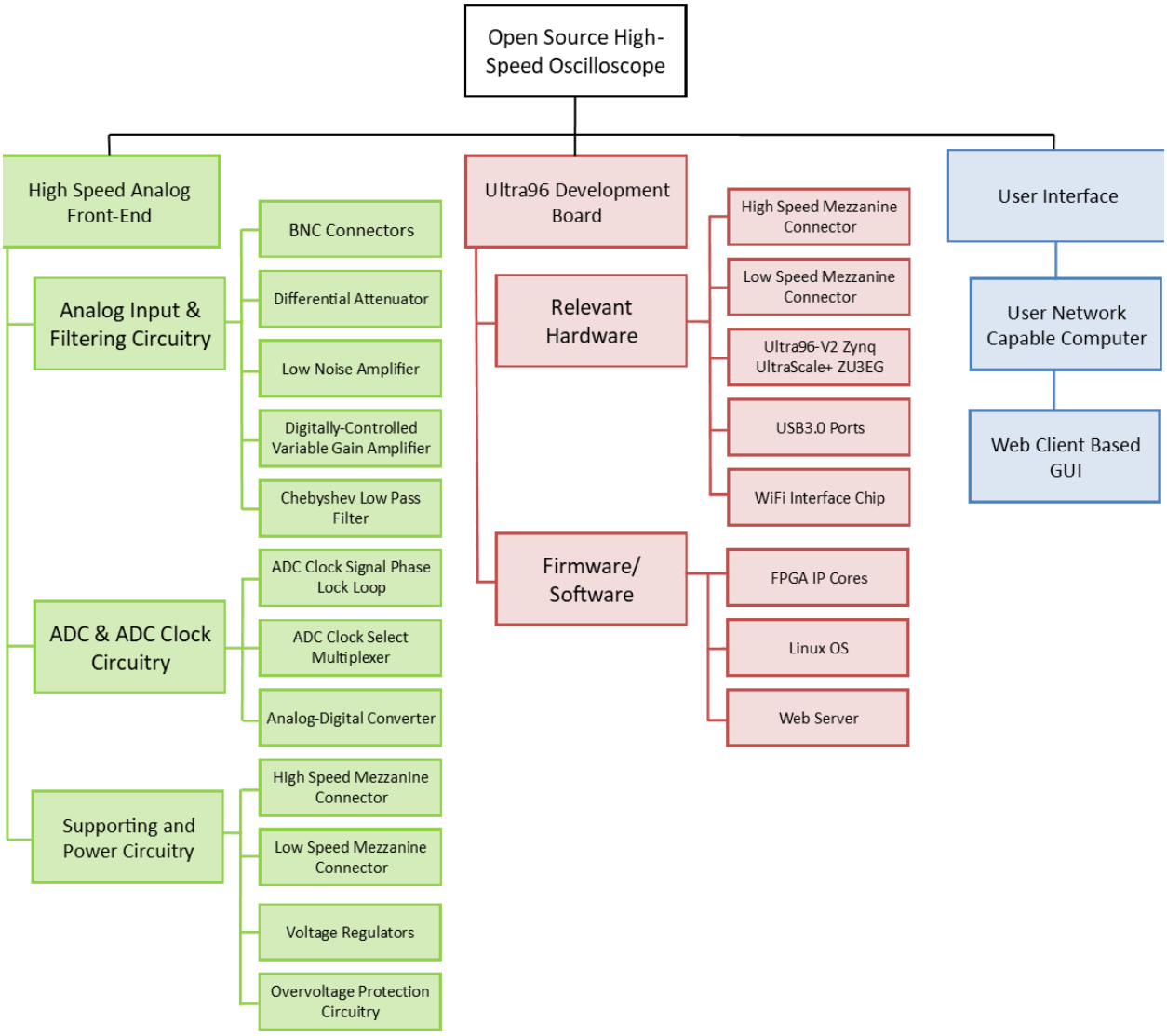


Figure 10. Physical Architecture

4.2.2 Overall System Architecture

In Figure 11 below is a diagram of the main system components integrated into the overall system architecture. It can clearly be seen that the

system will be divided into the three main subsystems that were outlined in our approach section: the analog front-end, the processing subsystem, and the web-based GUI. This diagram will serve as the model in which we plan to implement the flow of data, power, and control throughout the system

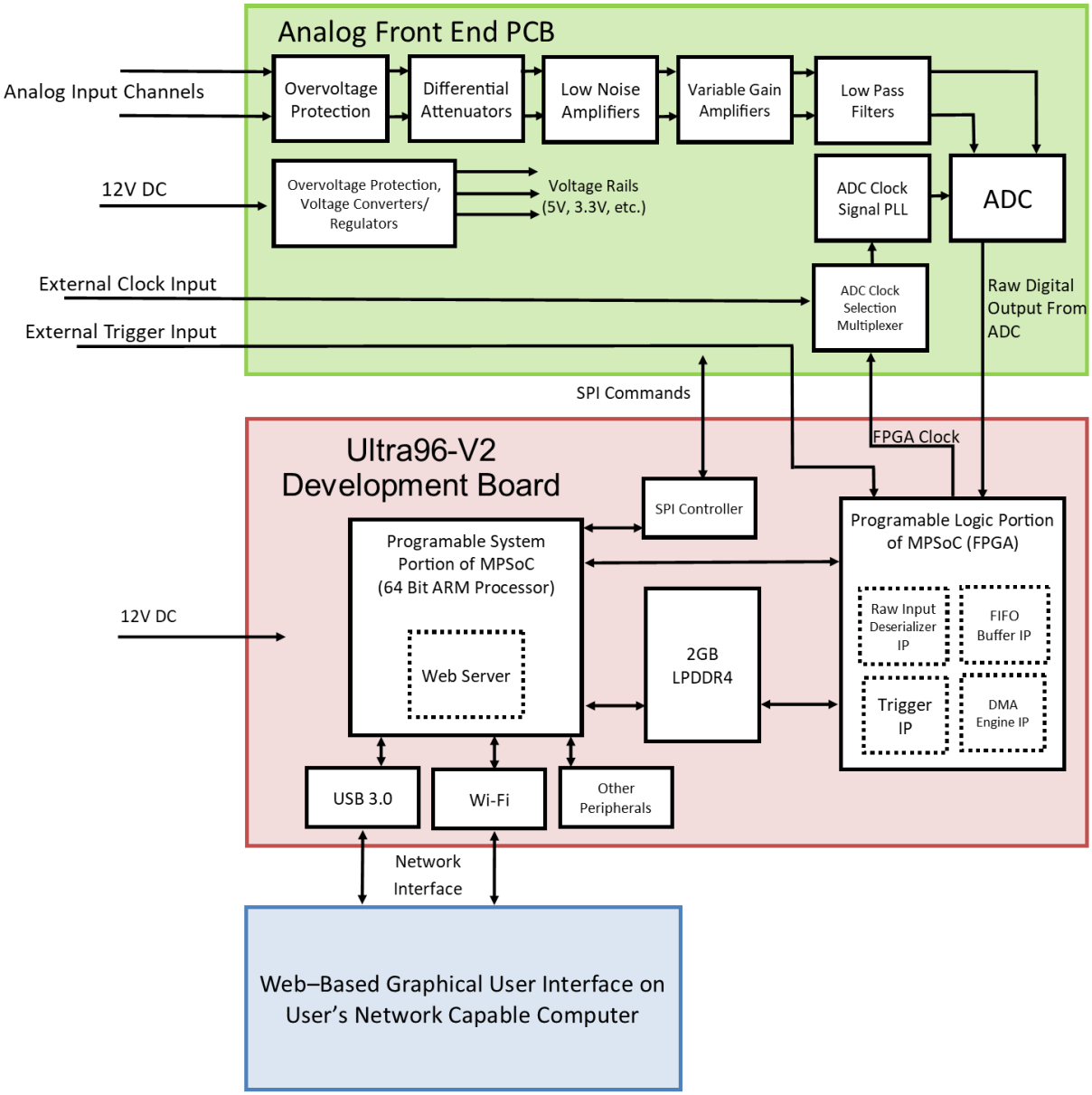


Figure 11. Overall System Architecture

5. Preliminary Experimentation and Testing Plan

5.1 Overview

In order to thoroughly verify the functionality of this device, testing will need to be conducted through two different approaches. The first approach is to perform testing at a white box level by examining the internal systems of the front-end board as well as the back-end firmware. This includes the analog circuitry on the board, the tracing

and layout of the board itself, and the IP cores used in the FPGA firmware. Much of this testing will be completed through the

The second approach is to perform testing at a black box level where the only signals being analyzed are the high level inputs and outputs of the system. The detailed list of testing plans for both approaches is presented below.

5.2 Internal Systems Testing

5.2.1 Attenuator

To verify the successful operation of the attenuator, a function generator will be used to provide AC inputs ranging from 100Hz-500MHz. The output voltage will be examined and the negative gain will be recorded at each frequency. Furthermore, a DC input voltage will be provided to the attenuator and the drop in voltage will be recorded again. This experiment will confirm the functionality of the attenuator for a wide range of input frequencies.

5.2.2 Low-Noise Amplifier (LNA)

Similar to the attenuator test, input signals with a varying range of frequencies and amplitudes will be provided to the LNA and the relationship between input/output voltage as well as the frequency response relationship will be plotted. This will allow a clear understanding of the voltage levels or frequency cut-offs where the output signal starts to saturate.

5.2.3 Variable Gain Amplifier (VGA)

The gain of the variable gain amplifier will be modified using SPI protocol and input signals of various frequencies will be provided through a function generator. The gain of the VGA will be verified through a commercial oscilloscope for frequencies up to 500MHz.

5.2.4 Phase-locked loop

An external clock shall be provided to the PLL and the clock multiplier will be adjusted through SPI. The output frequency of the PLL will be measured and verified.

5.2.5 Firmware testing

The HMCAD1511 ADC module will be used in conjunction with a function generator to provide a digitized waveform to the Zynq Zedboard SoC. This experiment will verify the success of the firmware in being able to display the waveform to a user on a computer. For preliminary tests, the Jupyter notebook web application will be utilized to view the waveforms. Other than the overall functionality of the firmware, testbenches will also be created in order to verify

each IP core. For the final design, an Ultra 96 board will be used instead of a Zynq Zedboard for running the back-end firmware.

5.3 High Level System Testing

5.3.1 External Trigger System

The external trigger system will be used to test if repetitive waveforms can be displayed in a steady manner for analyzation purposes. This will consist of applying an input signal to the analog input of the oscilloscope and verifying that the oscilloscope pauses data capture when an external trigger event occurs. This will be verified using a high-speed commercial oscilloscope by recording both the trigger event and the input signal.

5.3.2 Input variation

The overall device will be tested using both 1 and 2 analog inputs. The waveforms of these inputs will be varied between DC signal, sine waves, square waves, triangular waves, and more. The ability of the device to accurately display these waveforms on the GUI will be verified. The input voltage levels will be changed from 0 V_{PP} to 20 V_{PP} to confirm that the input voltage requirement is met.

5.3.3 Frequency Sweep

A function generator will be used to provide a periodic input signal to the device. A frequency sweep from 0Hz to 500MHz will be conducted and the absence of aliasing shall be verified.

5.3.4 Sampling rate

The external clock will be used to test the function of the device at different clock frequencies.

6. Preliminary Project Plan

6.1 Overview

The tasks we will be completing for ECE 492 will be divided into a hardware team and a software team. The hardware team will be focusing on testing and validation of the first version of the front-end analog circuitry. This includes cross-verifying the KiCad and Eagle schematics and ensuring the component models and values match. Furthermore, signal measurements will be taken directly from the board to ensure that the node voltages are equal to the specified values. After testing, certain electronic circuit components will be modified and a second version of the PCB will be created in KiCAD and sent out for printing. Besides this, the back-end firmware will also be modified so that the external trigger and clock functionalities can be added. For preliminary tests, the Zedboard will be utilized to verify the firmware, but then this will be ported to the Ultra96 development board.

The main task that the software team will be focused on will be the design and implementation of the web interface that will be used to display the waveforms that are collected by the hardware. A main component of this will be setting up a web server on the Ultra96 to function according to what we need for the interface to run properly. This will consist of configuring either the Jupyter notebook web server that is already on the board using Python 3 or creating another web server if necessary. We will also need to modify the open source Waveforms Live application

to be able to display and perform regular oscilloscope functions on the waveforms it receives.

In ECE 493, the main task for the hardware team will be to perform testing on the second version of the front-end PCB to ensure it functions correctly. The testing plan in section 5 will be followed to ensure each component as well as the entire system performs as designed. The main task for the software team will be to ensure that communication between the FPGA and the backend software runs smoothly and efficiently, creating near seamless interaction between the initial analog hardware and the graphical user interface.

6.2 Allocation of Responsibilities

The following is how the hardware and software teams will be divided, as well as a breakdown of the leads for each aspect of the project:

Hardware:

- Zaeem Gauher - Analog front-end circuitry validation and customization
- Timothy Bullock – Custom circuit and PCB design for analog front-end
- Umair Aslam – VHDL programming for the back-end firmware on the Ultra96

Software:

- Afnan Ali – Front end GUI design based on Waveforms Live Application
- Evan Hoffman – Backend software development for the web server/web client.

The **project manager** has the added responsibility of ensuring the timely completion of tasks assigned to each team member, and coordination with the faculty supervisor. Furthermore, the **team members** are responsible for completing their assigned tasks and deliverables by the agreed upon due date. Overall, each member of the team has a vital role to play in the success of this senior design project.

7. Potential Problems

7.1 Required Skills Training

There are multiple areas of knowledge as well as certain skills that will need to be learned to ensure successful completion of this project. Specifically, the following areas are of particular importance:

- Specifics of PCB design for high frequency analog applications
- Specific MPSoC interfaces and standards (AXI interfaces, BRAM, DMA Engine)
- GUI design using Angular

7.2 Risk Analysis

There are multiple risks associated with the development of this project. These risks are briefly discussed below:

- Digital signals have a finite speed at which they propagate through the PCB traces. If the trace lengths are mismatched, the measurements could potentially be highly inaccurate due to propagation delay. This is a high priority risk because a slight difference in trace lengths will reduce the effectiveness of the ADC. Furthermore, this issue is also very difficult to debug.
- PCB design errors are usually common which is why boards go through several revisions before being finalized. However, for this project, any errors will exceed the budget and time frame by an inadequate amount which is why this is a high priority risk. To reduce the chance of this happening, the PCB layout will be verified multiple times in KiCad before it is approved for printing.
- Electric component damage is also another probable risk associated with the testing of the front-end device. Since each integrated chip is being soldered by hand, the chances of burning out the chip due to incorrect pin connections or due to heat from soldering are quite high. Since this could increase the budget of the

project dramatically, extreme caution should be observed while handling the integrated chip components (especially the ADC).

- Many times, the ADC does not achieve the full resolution as specified in the data sheet. Due to noise, the ADC has a resolution related to the effective number of bits (ENOB). This would decrease the accuracy in the vertical scale of measurement. This risk is not given a high priority due to the fact that even a slightly lower resolution is still acceptable for application in which this device will be used.

8. Citations and References

[1] R. Beneder, M. Lechner and P. Schmitt, "Development of a low-cost, open-source measurement equipment for undergraduate courses dedicated to embedded systems," *IEEE EUROCON 2017 -17th International Conference on Smart Technologies*, Ohrid, 2017, pp. 187-192.

[2] A. Wozneak, R. Nagpal, and R. Meruvia, “ECE - 492 Design Document.” 10-Dec-2018.

[3] V. Niculescu and A. I. Liţă, “Open source oscilloscope for hobby users,” in 2015 14th RoEduNet International Conference - Networking in Education and Research (RoEduNet NER), 2015, pp. 203–207.

[4] “4000 Mixed Signal Oscilloscopes | RIGOL.” [Online]. Available: https://www.rigolna.com/products/digital-oscilloscopes/4000/. [Accessed: 12-Oct-2019].

[5] “Mixed Domain Oscilloscopes - MDO3000 Series Datasheet | Tektronix.” [Online]. Available: https://www.tek.com/datasheet/mixed-domain-oscilloscopes. [Accessed: 12-Oct-2019].

[6] “PC Oscilloscope, Data Logger & RF Products | Pico Technology.” [Online]. Available: https://www.picotech.com/. [Accessed: 12-Oct-2019].

[7] “DSOX3052A Oscilloscope: 500 MHz, 2 Channels | Keysight (formerly Agilent’s Electronic Measurement).” [Online]. Available: https://www.keysight.com/en/pdx-x201849-pn-DSOX3052A/oscilloscope-500-mhz-2-channels?cc=US&lc=eng. [Accessed: 12-Oct-2019].

[8] “1000Z Mixed Signal Oscilloscopes | RIGOL.” [Online]. Available: https://www.rigolna.com/products/digital-oscilloscopes/1000z/. [Accessed: 12-Oct-2019].

[9] “Untitled.” [Online]. Available: https://www.tequipment.net/TektronixDPO2022B.html. [Accessed: 12-Oct-2019].

[10] “PC Oscilloscope, Data Logger & RF Products | Pico Technology.” [Online]. Available: https://www.picotech.com/. [Accessed: 12-Oct-2019].

[11] “PC Oscilloscope, Data Logger & RF Products | Pico Technology.” [Online]. Available: https://www.picotech.com/. [Accessed: 12-Oct-2019].

[12] “ScopeFun - Open Source Oscilloscope.” [Online]. Available: https://www.scopefun.com/. [Accessed: 12-Oct-2019].

[13] “BitScope Mini Model 10 | World’s Smallest Mixed Signal PC Based USB Oscilloscope!” [Online]. Available: https://www.bitscope.com/product/BS10/. [Accessed: 12-Oct-2019].

[14] “OpenScope MZ: Open-source All-in-one Instrumentation,” *Digilent*. [Online]. Available: https://store.digilentinc.com/openscope-mz-open-source-all-in-one-instrumentation/. [Accessed: 12-Oct-2019].

[15] “OpenADC - NewAE Technology Inc.” [Online]. Available: http://store.newae.com/openadc/. [Accessed: 12-Oct-2019].

[16] 96Boards. (2019). Ultra96. [online] Available at: https://www.96boards.org/product/ultra96/ [Accessed 4 Oct. 2019].

[17] “HMCAD1511 Datasheet and Product Info | Analog Devices.” [Online]. Available: https://www.analog.com/en/products/hmcad1511.html. [Accessed: 12-Oct-2019].

[18] “12 THINGS TO CONSIDER WHEN CHOOSING AN OSCILLOSCOPE.” Tektronix, 2010.

[19] S. R. Al-Araji, K. A. Mezher and Q. Nasir, "First-Order Digital Phase Lock Loop with Continuous Locking," *2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks*, Madrid, 2013, pp. 414-417.

11. Appendix B: Design Document (ECE492)

Open Source High-Speed Oscilloscope (OSHO)

Design Document

Team Members:
Timothy Bullock, Afnan Ali, Evan Hoffman, Umair Aslam, Zaeem Gauher

Faculty Advisor:
Jens-Peter Kaps

ECE492-001

Date of Submission: December 6th, 2019

George Mason University
4400 University Dr, Fairfax VA 22030

1. Problem Statement

Digital oscilloscopes are extremely useful tools for many engineering applications where electrical signals need to be measured and analyzed. Digital oscilloscopes "enable the user to debug, visualize and measure various signals," and are an essential part of any engineering lab or project [3]. Yet, in many applications such as RF design, the signals that are being analyzed are too high frequency to be measured with standard low-cost oscilloscopes. In these applications, high performance oscilloscopes with sufficient bandwidth and sampling-speeds are needed. The problem with this is that oscilloscopes with bandwidths greater than 500 MHz are extremely expensive. Even moderate performance oscilloscopes with bandwidths greater than 200 MHz can cost several hundreds to thousands of dollars. On top of this, even at these high prices, many of the commercially available devices can be limited in certain usability aspects and features. For example, downloading the captured data from these devices for external processing can be quite slow, and their built-in ADC cannot be synchronized to an external clock signal. Therefore, to overcome these limitations, it is our project's motivation to create a low-cost, open source, and high-speed alternative to existing oscilloscopes.

2. System Requirement Specifications

2.1 Mission Requirements:

- The project shall design an oscilloscope that is an open source, low-cost alternative to commercially available oscilloscopes, and a performance, feature rich alternative to existing open-source oscilloscopes.
- The project shall design a custom high-speed PCB that will easily interface with an Ultra96-V2 development board, as well as develop the supporting firmware and graphical user interface for the device.

2.2 Operational Requirements:

2.2.1 Input/Output Requirements

- The device shall have at least two analog input channels, one external clock input, and one external trigger input.
- The system will receive control and configuration commands as well as be able to responsively display captured data through a web client with an intuitive and responsive GUI.

2.2.2 External Interface Requirements

- The device will provide support for 1x and 10x passive probe inputs (50Ω).
- Bayonet Neill–Concelman (BNC) connectors shall be used for the analog inputs, external clock input, and external trigger inputs.

- The system shall interface with a network capable computer through USB3.0 or WiFi.
- The system shall receive power from an external 12V DC power supply.

2.2.3 Functional Requirements

- The analog-to-digital converter (ADC) shall sample one input channel at 1 GSPS or two channels at 500 MSPS.
- The device will be able to measure analog inputs with a maximum input voltage of ±10V.
- The input analog circuitry shall achieve a 500 MHz bandwidth.
- The ADC shall be able to be configured to sample using either the FPGA clock or an external clock input (between 30 MHz and 1 GHz).
- The ADC output sample resolution shall be no less than 8 bits.
- The system's data capture shall have the ability to be triggered using both configurable edge triggers as well as a configurable external trigger input.

2.2.4 Technology and System-Wide Requirements

- The front-end device shall use a single 1GSPS ADC chip.
- The ADC data shall be processed and hosted on an onboard Linux web server using a Xilinx Zynq UltraScale+ multiprocessor systems-on-chip (MPSoC) aboard the Ultra96 Board.
- The analog front-end custom PCB should interface with the Ultra96 Board for data processing.
- Target FPGA development board shall have device driver firmware for interfacing with the ADC, and routing and storing ADC sample data in a memory device.
- Front-end programmable devices will be controlled using the Serial Peripheral Interface (SPI) or other serial protocol.
- The custom high-speed PCB and Ultra96 devices will interface with each other via the Ultra96's high-speed and low speed mezzanine connectors.
- The device should be low-cost ($600 or less).

3. System Decomposition & Architecture

3.1 Level Zero Decomposition

In order to provide a detailed overview of the system architecture for our solution, it is best to start with a functional decomposition of the system so that the system's functions can be related in a hierarchical manner. This decomposition will provide a top level overview of the system, then work downward to identify each of the main processes of the system, then continue downwards to identify the sub functions of each of these processes. The level zero decomposition provides a top level overview of the overall solution; it shows the overall system inputs and outputs. For our system, this is shown below in figure 1. It shows that the overall system will take in two analog inputs, an external clock input, an external trigger output, user commands, and DC power. The system then outputs status information and digitized waveform data.

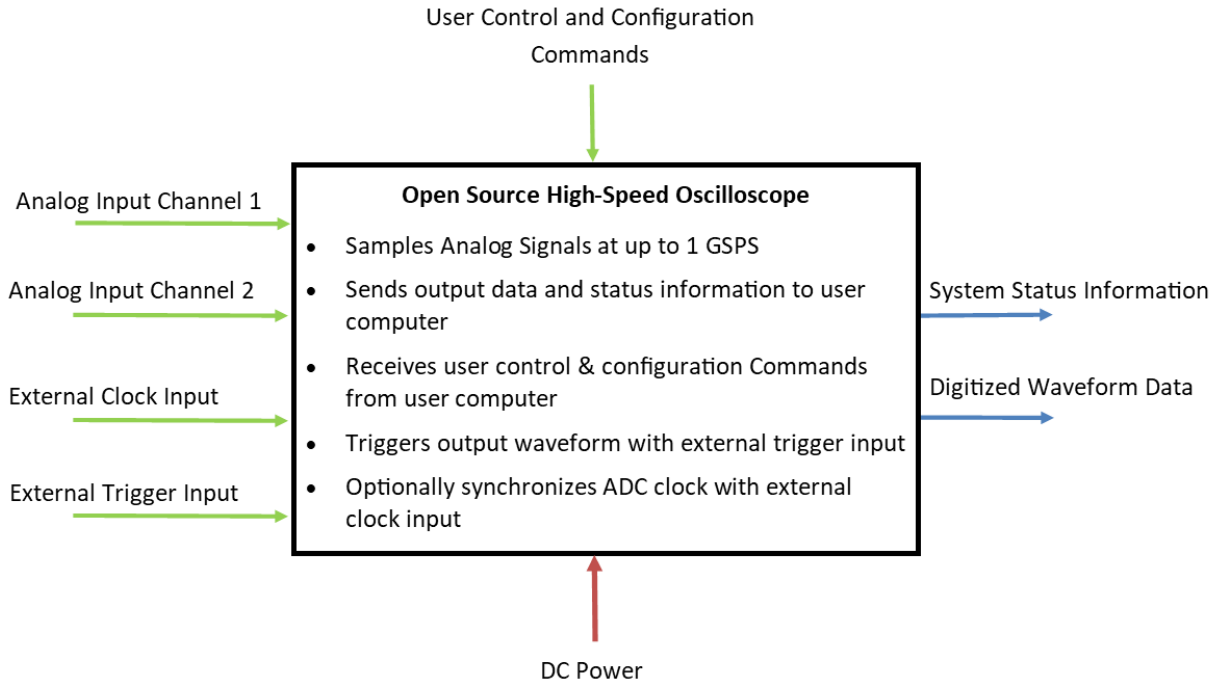


Figure 1. Level Zero Functional Architecture Block Diagram

3.2 Level One Decomposition

After the system is understood at the highest input/output level (level zero), the next step of functional decomposition is to identify the top level processes of the system. For our system this would include analog signal preconditioning, analog to digital conversion, ADC clock selection/generation, raw data buffering and routing, processing and data hosting, and finally, display and interface processing. This is summarized in the level one diagram shown below (Figure 2). Once each of these main processes is identified at this level, they can then be further decomposed and discussed at the level two decomposition level.

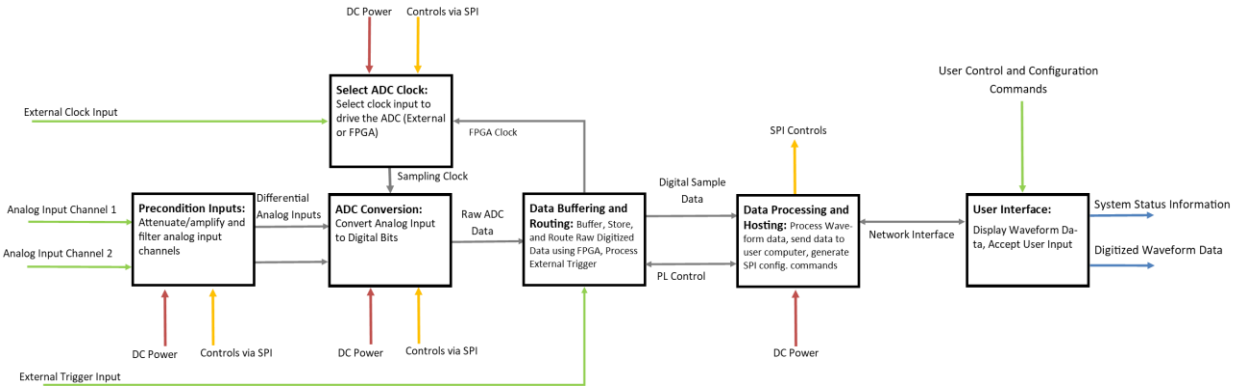


Figure 2. Level One Functional Architecture Block Diagram

3.3 Level Two Decomposition

Once the functionality of the system is understood at the level one demoposition level, the next step to providing a detailed overview to the system design is to take each of these top level processes and decompose them into their subprocesses. This is done for each of the top level processes shown in the level one functional architecture block diagram above (Figure 2). It is worth noting that throughout the completion of the detailed system design that these level two subprocess blocks are subject to slight alterations.

3.3.1 Analog Input Signal Preconditioning Stage/Function

The purpose of the analog input signal preconditioning stage/function is to take in the analog inputs and modify them so they can be most optimally digitized by the ADC. The functions that occur in this main process are: overvoltage protection, coupling and offset selection, variable attenuation and amplification, and finally passing through a low pass anti-aliasing filter. The input signals are first passed though overvoltage protection to protect the remainder of the circuitry. Then the signals are then modified by selecting DC or AC coupling, and the desired offset is added to the signal. Next, the signals are converted to a differential signal and are attenuated so that the signals can fit within the

full-scale range (FSR) of the ADC. Next, the signals are variably amplified so their amplitude more accurately fits the FSR of the ADC. Finally, the signals are sent through a low pass filter to reduce high frequency noise and limit the signals to the Shannon-Nyquist frequency dictated by the ADC maximum sampling rate. Configurable aspects of this system such as DC offset will be configured through SPI commands from the processing subsystem.This front-end circuitry will be routed on a custom high-speed PCB that will be designed by our team.

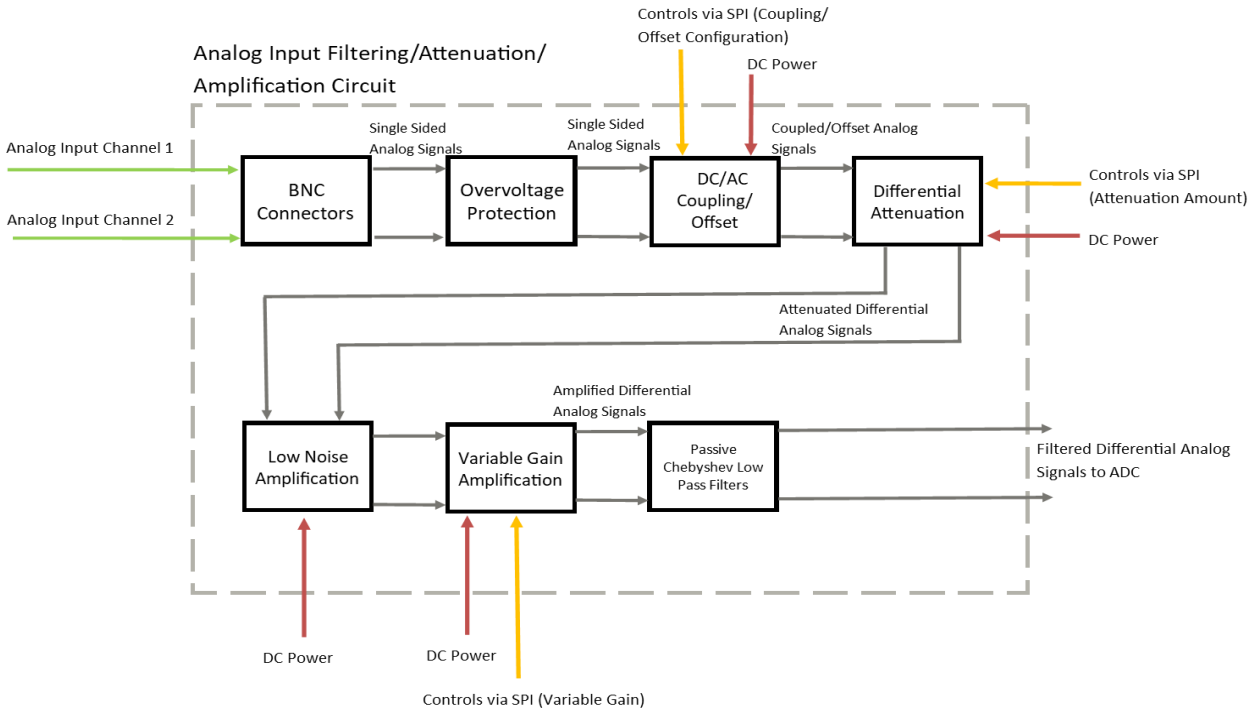


Figure 3. Level Two Functional Architecture Block Diagram - Analog Input Signal Preconditioning

3.3.2 Analog to Digital Conversion Stage/Function

After the signals have been preconditioned, they are then sent to the analog to digital conversion stage/function. This stage is the simplest stage as it only consists of one main function and component, the high sampling speed ADC. This stage takes the preconditioned analog signals and outputs a digital LVDS signals representing the digitized sample data. This data is sent to the data and buffering and routing stage. This stage will also be located on the custom high-speed PCB that will be designed by our team.

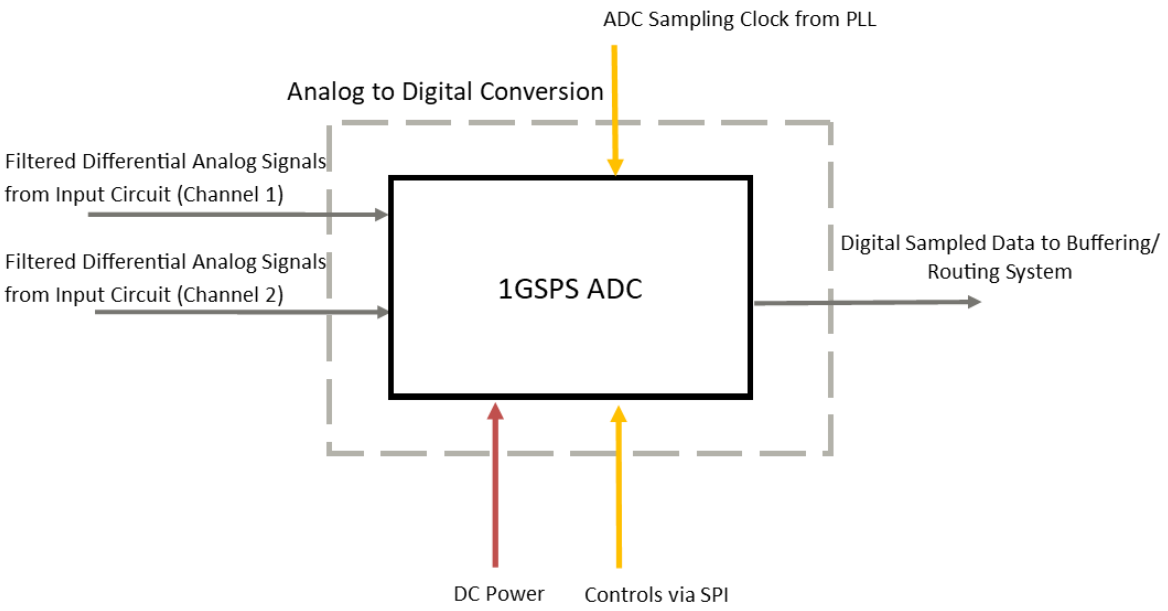


Figure 4. Level Two Functional Architecture Block Diagram - Analog Digital Conversion

3.3.3 ADC Sampling Clock Generation Stage/Function

Another major function of the overall system is to generate the clock signal for the ADC. In this stage/function, either the FPGA clock or the external clock is toggled between as an input into the phase locked loop (PLL) which matches or multiplies the frequency of the input signal to generate a low jitter clock signal for the ADC. This is the third stage/function that will be located on the custom PCB.

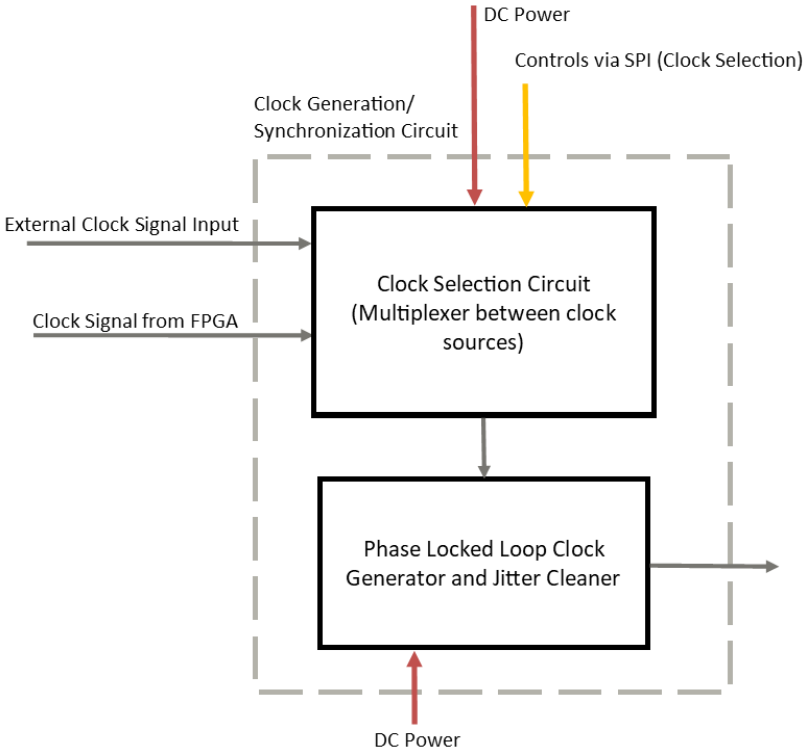


Figure 5. Level Two Functional Architecture Block Diagram - ADC Sampling Clock Generation

3.3.4 Data Buffering and Routing Stage/Function

The next stage/function of the system is the Data Buffering and Routing Stage. The purpose of this stage is to receive data from the ADC, deserialize the data, and create 64-bit AXI packets which can then be loaded into main memory via direct memory access (DMA). This stage will also process the external trigger input in order to generate necessary control signals and stop the flow of digitized waveform data into memory. This stage will be implemented using the programmable logic (PL) portion of a MPSoC development board. More Specifically, this stage will be implemented on the Xilinx Zynq Ultrascale+ MPSoC ZUEG A484 that is on the Ultra96 V2 board and using

custom and Xilinx provided Intellectual Property (IP) cores connected using the Advanced eXtensible Interface (AXI).

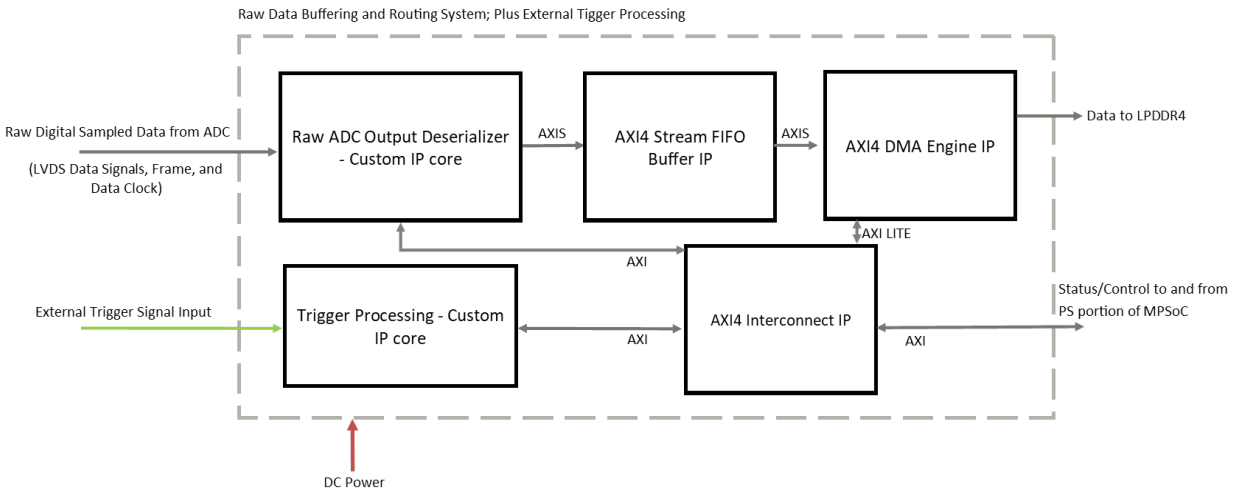


Figure 6. Level Two Functional Architecture Block Diagram - Data Buffering and Routing

3.3.5 Data Processing and Hosting System

After the output data from the adc is stored in main memory, the next thing that must happen to it is that it must be processed and hosted on a web server running on the ARM processor portion of the MPSoC. This ARM processor will be running a server which will host the web server that communicates with the user interface which will be implemented as a web client on a remote computer. This processor will also be running additional software in order to generate the commands to control the various components on the custom analog front-end via SPI, perform basic processing on the waveform data such as downsampling and converting the data into the desired protocol for the webserver, and finally to communicate with and control the PL portion of the chip via the AXI interface.

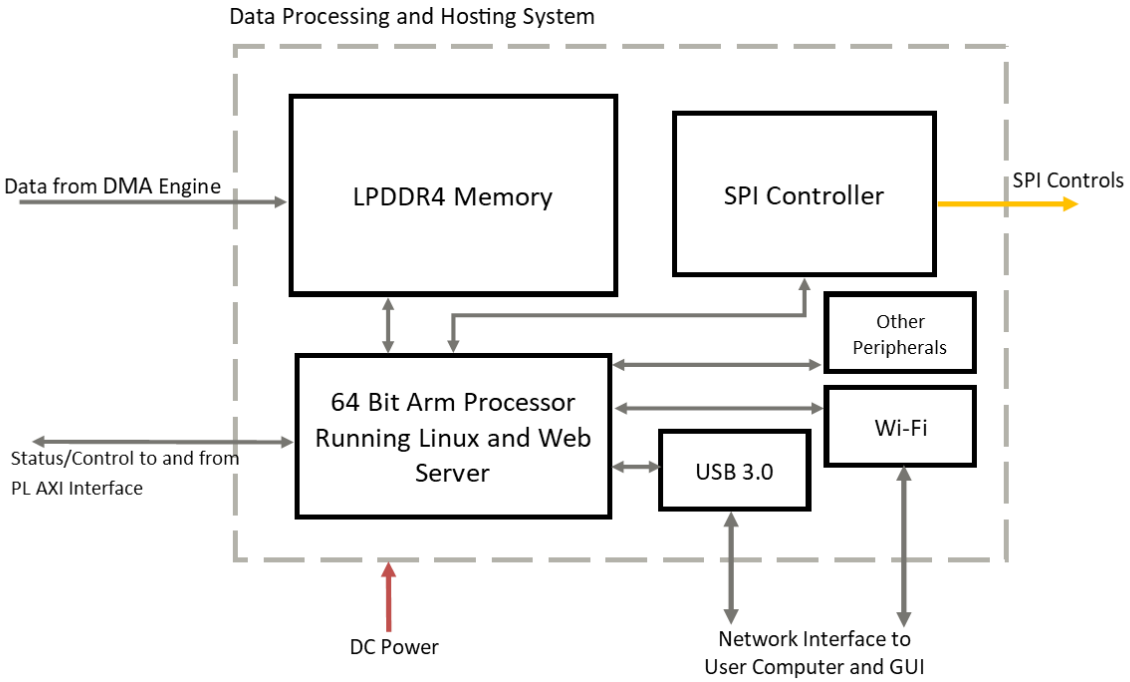


Figure 7. Level Two Functional Architecture Block Diagram - Processing and Data Hosting

3.3.6 User Interface

The final stage/function that is required for our system is the user interface so that the user can control the system and view the digitized waveform data. This stage will consist of a webclient that will be running in a web browser running on the user's network capable computer. This web client will communicate with the server running on the Ultra96 in order to pass control and configuration information to the system and output waveform and status information.

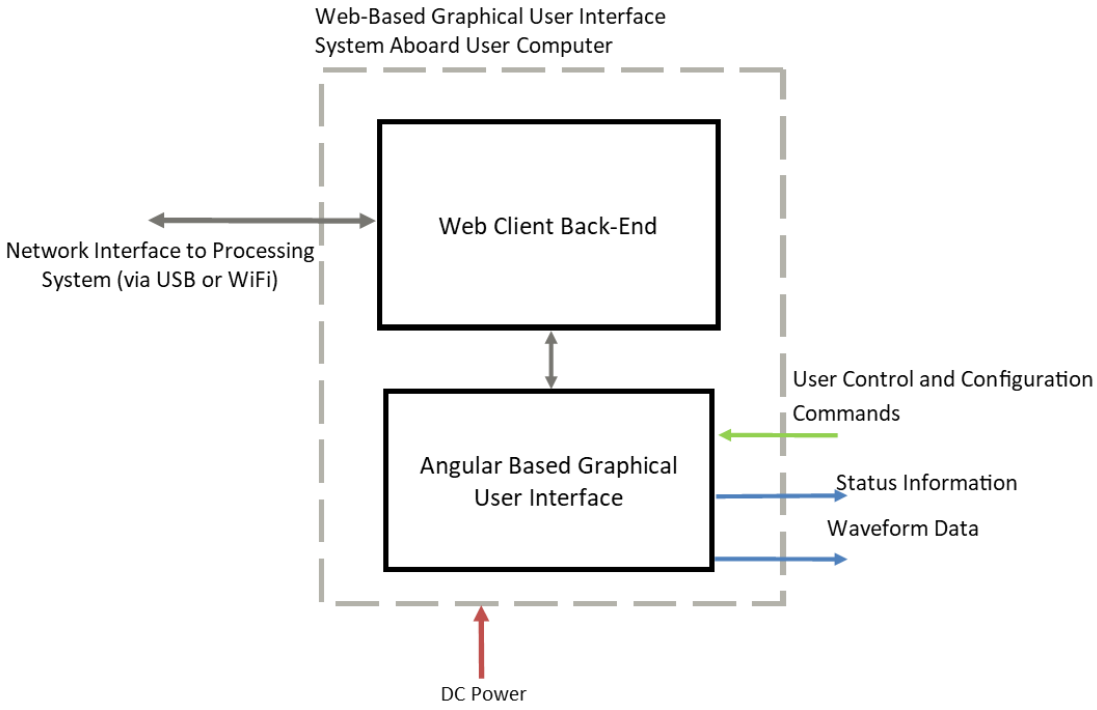


Figure 8. Level Two Functional Architecture Block Diagram - User Interface

3.4 Overall System Architecture

In Figure 9 below is a diagram of the main system components integrated into the overall system architecture. It can clearly be seen that the system will be divided into the three main subsystems: the analog front-end, the processing subsystem, and the web-based GUI. This diagram will serve as the model in which we plan to implement the flow of data, power, and control throughout the system.

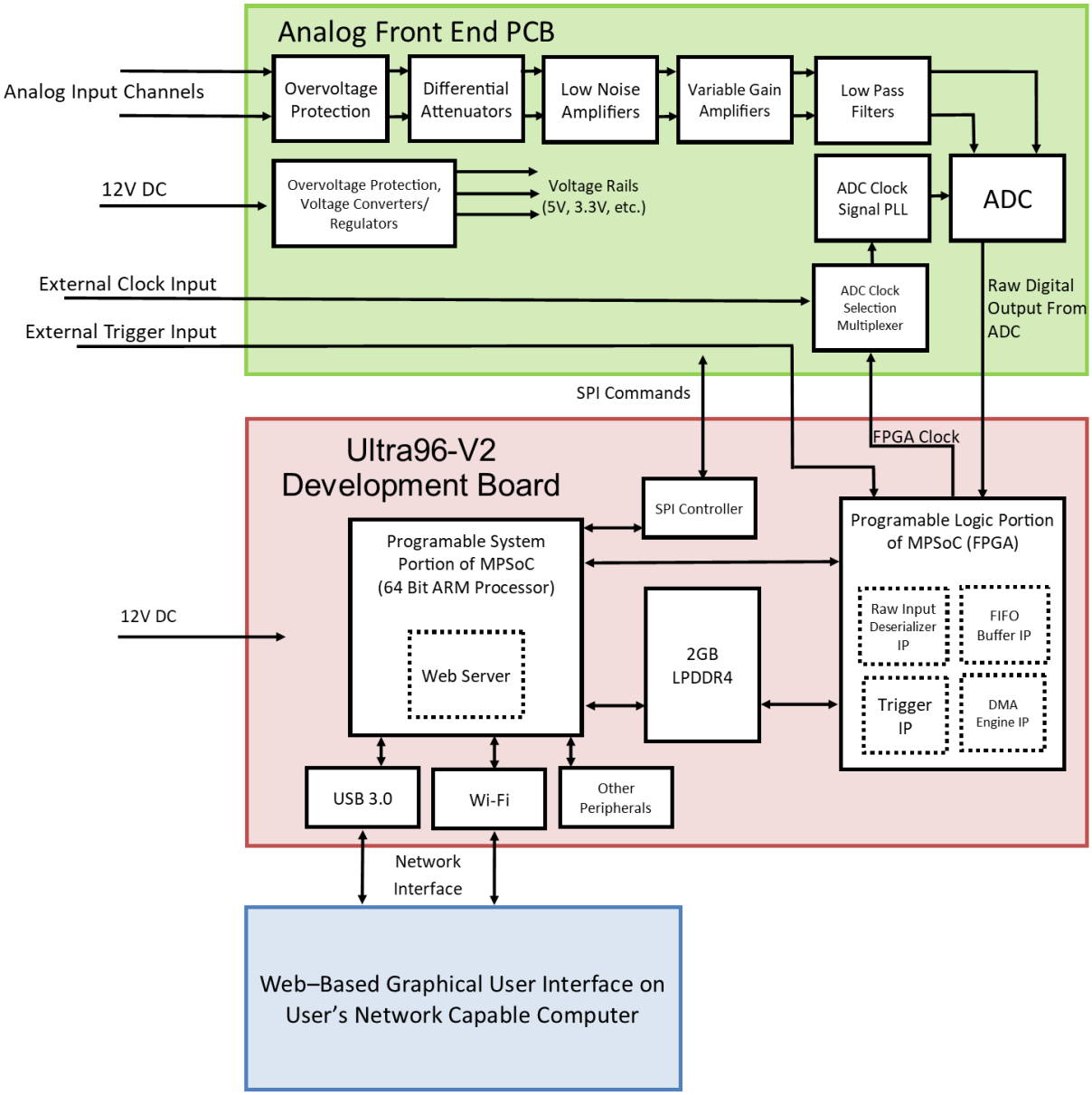


Figure 9. Overall System Architecture

3.5 Physical Architecture

The physical architecture consists of a hierarchical diagram that shows the main configuration items that make up the system. This includes major hardware and software components. This serves as a hierarchical overview of the major physical resources that will be required to implement our solution.

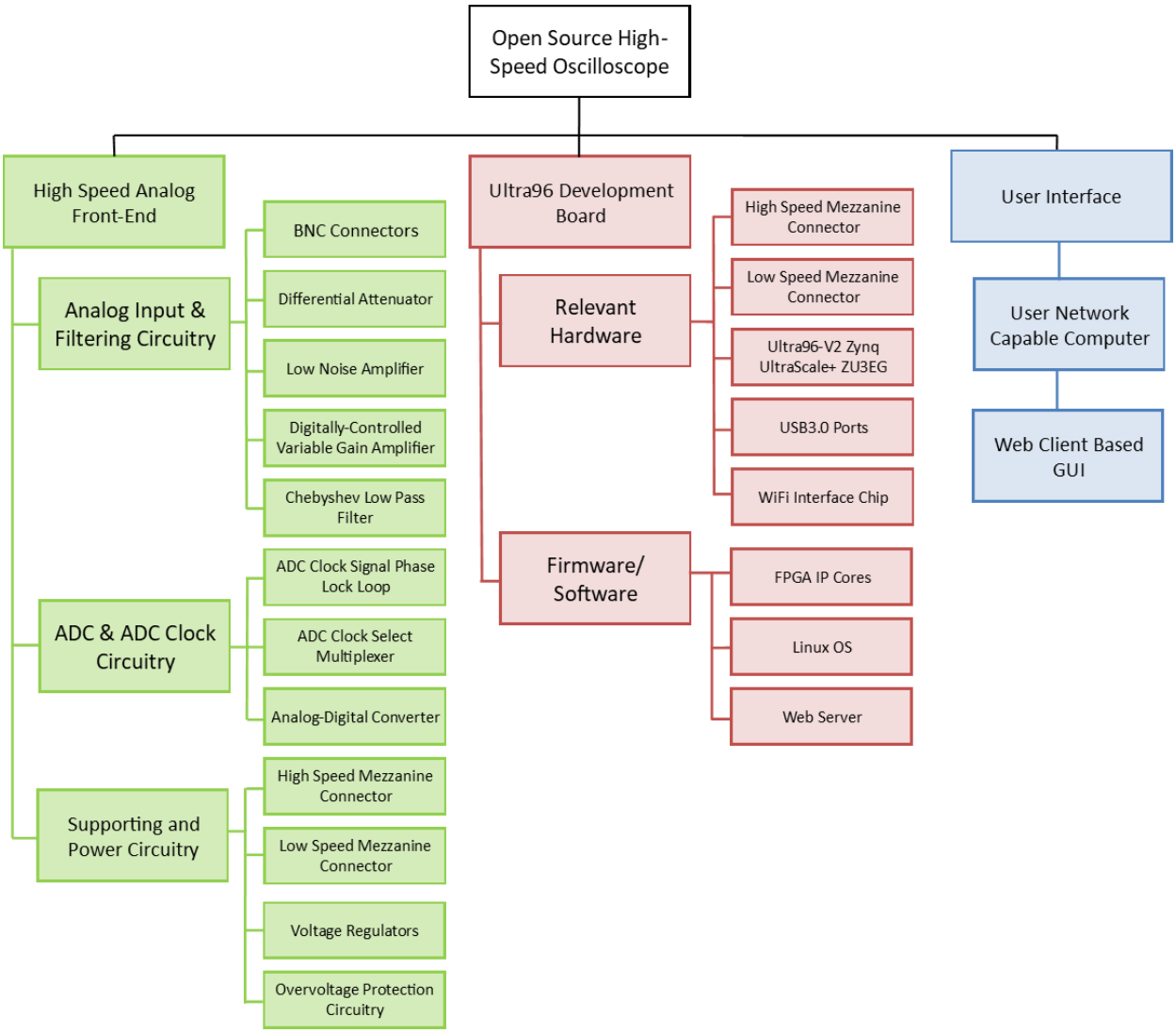


Figure 10. Physical Architecture

4. Background Knowledge Used in Design

4.1 Analog Front-End

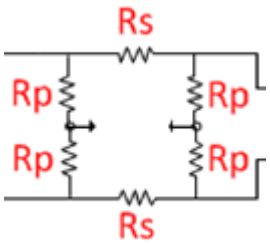
In order to understand the function of the overall analog front end circuit, it is important to understand the theory used in deriving our solution. The analog front end circuitry consists multiple subcomponents that comprise the overall signal measurement chain. These components include an attenuator, low noise amplifier (LNA), a variable gain amplifier (VGA), and an anti aliasing low pass filter (LPF). Furthermore, the analog to digital conversion system

consists of a phase locked loop and an 8-bit ADC. The background knowledge as well as the mathematical theory utilized to select and justify these components is described in further detail below.

4.1.1 Attenuator Design:

The attenutor's primary function is to reduce voltage, dissipate power, and improve impedance matching between devices such as amplifiers. Attenuators can be configured to adjust the amount of attenuation manually. The utilization of an attenuator is crucial in the front-end as it provides sufficient input amplitude adjustment to prevent saturation for large signal swings. Although there are multiple possible configurations of an attenuator, the Pi attenuator is best suited for applications where impedance matching is important. A circuit diagram of the differential Pi attenuator along with the calculations to design the resistance values are presented below. For the purpose of the OSHO project, two attenuator paths are required; a 10 to 1 as well as a 20 to 1 path are used to ensure that the input voltage specifications are upheld.

Differential Pi configuration attenuator circuit:



Relevant Equations used to design attenuator:

$$R_p = Z_0 * \frac{1 + A_T}{1 - A_T}$$

$$R_s = \frac{Z_0 R_p}{Z_0 + R_p} * \frac{1 - A_T}{A_T}$$

$$Z_0 = 50\Omega\ (Source\ and\ Load\ Impedance)$$

$$A_T: Attenuation\ (\frac{V}{V})$$

Resistance value calculations for 10:1 attenuation path:

10:1 Path

$$R_p = 50\Omega * \frac{1 + (\frac{1}{10})}{1 - (\frac{1}{10})}$$

$$\boldsymbol{R_p = 61.11\ \Omega}$$

$$R_s = \frac{50 * 61.11}{50 + 61.11}\Omega * \frac{1 - (\frac{1}{10})}{(\frac{1}{10})}$$

$$\boldsymbol{R_s = 247.49\ \Omega}$$

Resistance value calculations for 20:1 attenuation path:

20:1 Path

$$R_p = 50\Omega * \frac{1 + (\frac{1}{20})}{1 - (\frac{1}{20})}$$

$$\boldsymbol{R_p = 55.26\ \Omega}$$

$$R_s = \frac{50 * 55.26}{50 + 55.26}\Omega * \frac{1 - (\frac{1}{20})}{(\frac{1}{20})}$$

$$\boldsymbol{R_s = 498.75\ \Omega}$$

4.1.2 Low-Noise Amplifier (LNA):

Low noise amplifiers are used to amplify very low-power signals without negatively affecting the signal-to-noise ratio. By using a LNA close to the input source, the effects of noise in the following stages of the front-end stage can be greatly reduced. To ensure the maximum transfer of power from source to amplifier, the source impedance should match the input impedance of the LNA. This can be achieved through the attenuator mentioned earlier. The circuit for the low noise amplifier used in the OSHO front-end design is based off the reference design provided by Texas Instruments for the LMH5410 LNA. The specified voltage gain of 4V/V in the TI design is adequate for this application due to the fact that the VGA can be used to further fine tune the gain settings.

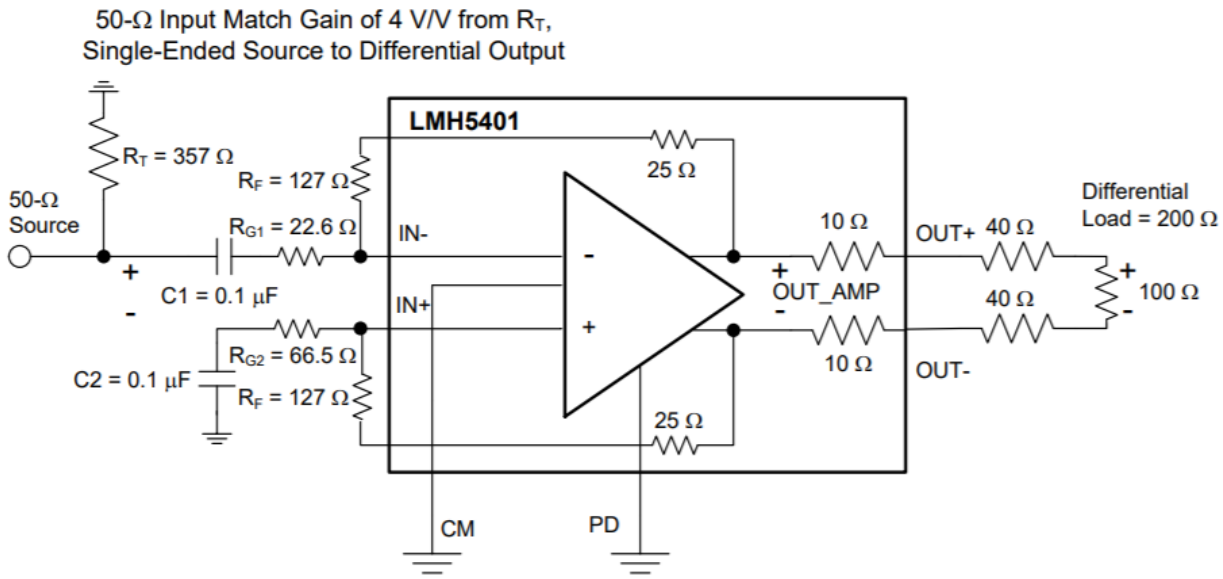


Figure 11. LMH5401 Circuit

The calculation process for picking the resistance values in this reference design is laid out in brief details below:

1) The resistance values for R_F is selected based on the choice of the user. There is great flexibility in choosing the R_F resistor since it is an external resistor.
2) Once R_F is chosen, the following equations are utilized to calculate the resistance values for R_{G1} and R_{G2}. The value for R_S used in these equations is equal to 50 ohms which is the source impedance.

$$R_{G1} = \frac{2\frac{R_F}{A_V} - R_S}{1 + \frac{R_S}{R_T}}$$

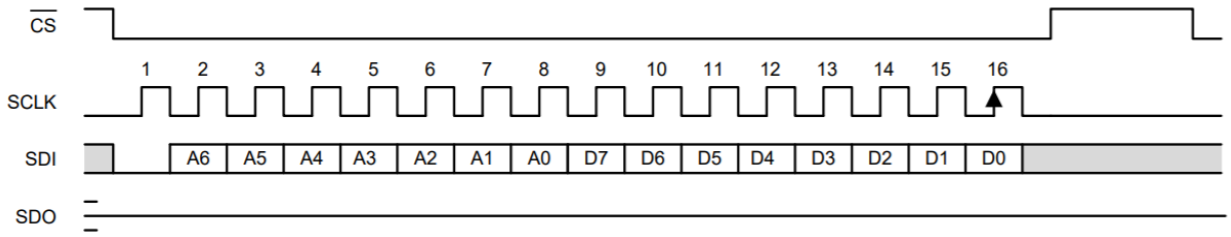
$$R_{G2} = \frac{2\frac{R_F}{A_V}}{1 + \frac{R_S}{R_T}}$$

4.1.3 Variable Gain Amplifier (VGA):

A VGA is used to amplify input signals based on the gain parameter. The advantage of using a VGA is that the gain can easily be controlled through an interface such as SPI to ensure that the output fits within the full-scale input range of the ADC. This prevents clipping of the digital output waveform. The LMH6401 VGA used in the OSHO design has a set of internal registers which can be read to or written from using a 4 pin SPI configuration. To enable transfer of data, an active low chip select pin is utilized. Serial data is loaded into/out of register every 16th clock cycle due to the fact that word length is 16 bits. The first 8 bits specify the address of the register and the next 8 bits are either

the data being loaded or read from the register. The SPI timing diagrams for read and write cycles for this VGA are presented below.

SPI Write Bus Cycle



SPI Read Bus Cycle

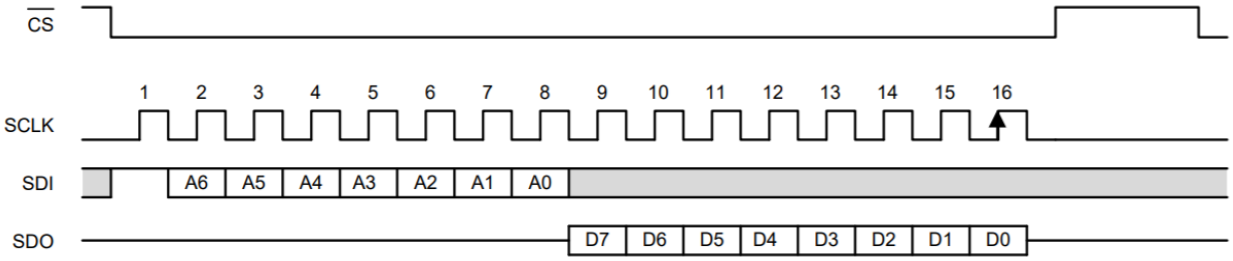


Figure 12. SPI Timing Diagram

4.1.4 Anti-Aliasing LPF:

An anti-aliasing low pass filter's function is to remove unwanted high frequency components from the input signal. This filter is crucial to ensure that the input to the ADC has a maximum frequency of half the sampling rate. This prevents aliasing as dictated by the Nyquist-Shannon Theorem. A Chebyshev LPF filter is used for in our design due to the fact that it provides the sharpest cut-off frequency by allowing a small ripple in the frequency response. For the purpose of this project, the values for the elements in the Chebyshev LPF were calculated using MATLAB software. The circuit designs as well as simulations for the the two chebyshev filters used in this project are presented later.

4.1.5 Phase-locked loop(PLL):

A phase-locked loop is a voltage driven oscillator that receives a reference signal and outputs a signal with either a matched or multiplied frequency compared to the reference. The PLL also acts similar to a bandpass filter to remove high frequency jitter as well as low frequency VCO jitter from the clock signal [19]. The PLL will allow for the use of the FPGA clock in order to provide the clock input to the ADC. Similarly, it can also be utilized to sync the ADC clock with an external clock when that option is selected. A functional block diagram of the PLL used for out front-end circuit is presented below:

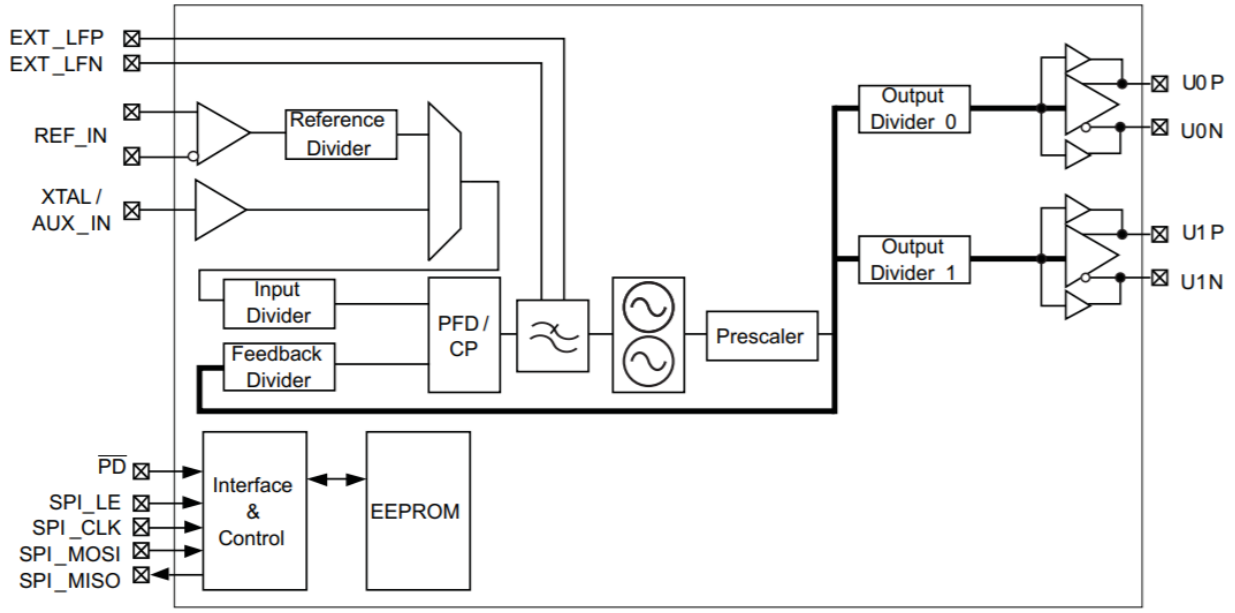


Figure 13. CDCE62002 PLL Block Diagram

In the diagram above, the interface and control block determines the status of the PLL when the device is powered based on the contents of the EEPROM. On the other hand, SPI commands can also be used to directly change the output of the PLL by writing directly to the device registers.

4.1.6 Analog to Digital Converter(ADC):

The ADC is one of the most crucial features of the oscilloscope as it determines the sampling rate, resolution, as well as bandwidth. The ADC receives analog signals from the signal conditioning stage which includes attenuation, amplification, and filtration. The analog signal is then sampled and digitized before being transferred to the back-end firmware for transferring the data into memory. Some of the important features of the ADC are detailed below:

Bandwidth:

The bandwidth of an ADC dictates the maximum frequency range that can be accurately measured by the device. High-speed, serial communication, and other complex signal applications require bandwidths of 500MHz or greater for accurate measurement.

Sample Rate:

The sample rate of an ADC (measured in samples/second) defines how often the device samples the signal. According to the Nyquist-Shannon Sampling Theorem, the sampling rate needs to be twice as fast as the highest frequency component of a signal in order to avoid aliasing. Thus, if a sampling rate of 1GSPS is used, the maximum input frequency should be limited to 500MHz.

Channel Resolution:

The resolution of the ADC defines the granularity of the signal. If the ADC in the oscilloscope has an 8-bit resolution, this translates to $2^8 = 256$ digitized levels

that each analog sample will be translated to. An ADC with a resolution of 8 bits is sufficient for a low-cost oscilloscope application.

The background knowledge for all important subcomponents of the OSHO front-end circuit are presented above. However, it is also important to understand the theory behind the function of the overall system.

4.2 FPGA Datapath and Firmware

4.2.1 Zynq Architecture

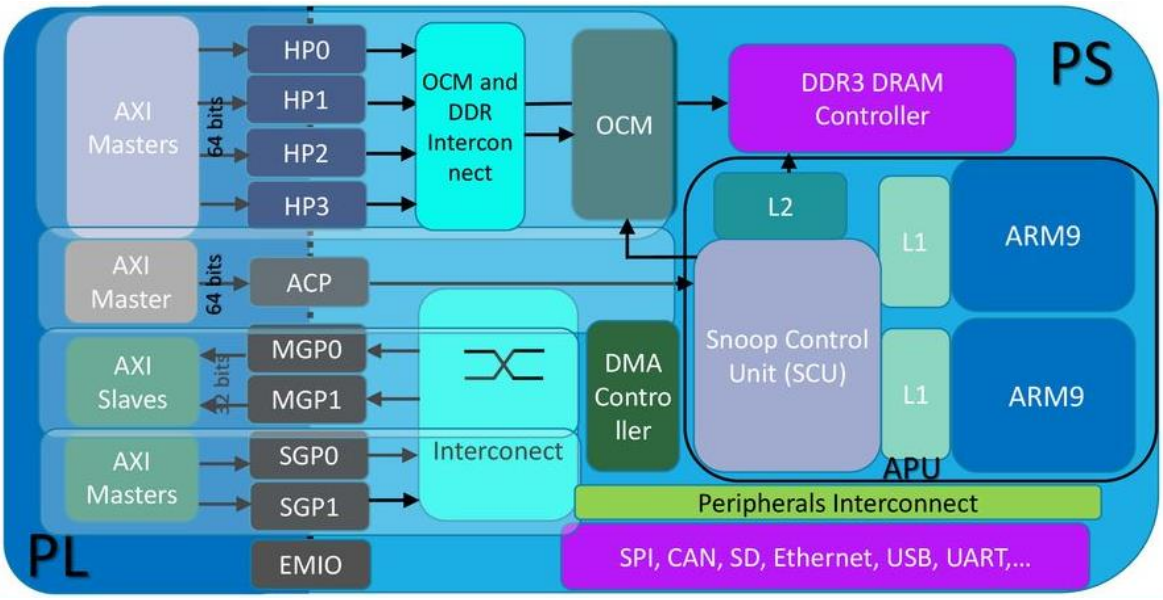


Figure 14. Zynq Architecture

A Field-Programmable Gate Array (FPGA) is the optimal choice for data intensive processing, massively parallel algorithms or applications dealing with a large number of inputs and outputs. In contrast, a microprocessor is better utilized for complicated decision making. However, both functionalities are often required in parallel for applications such as building a high-speed oscilloscope. Prior to the Zynq architecture, however, communication between the FPGA element and the Processing System (PS) was complex and this led to a collection of modules rather than an actual system. The Zynq architecture solves this problem by using the Advanced eXtensible Interface (AXI) standard for communication between the Programmable Logic (PL) layer and the PS layer. The PL layer contains the configurable logic blocks (CLBs) to implement any hardware functionality and can be used to extend the processing system, while the PS layer contains the ARM processor, I/O peripherals and integrated memory controllers. Utilizing the AXI standard also reduces latency and increases the overall performance of the system. Furthermore, PYNQ Linux is loaded on the Processing System (PS). It contains the Jupyter Notebook server which the user can use to interact with and control the system.

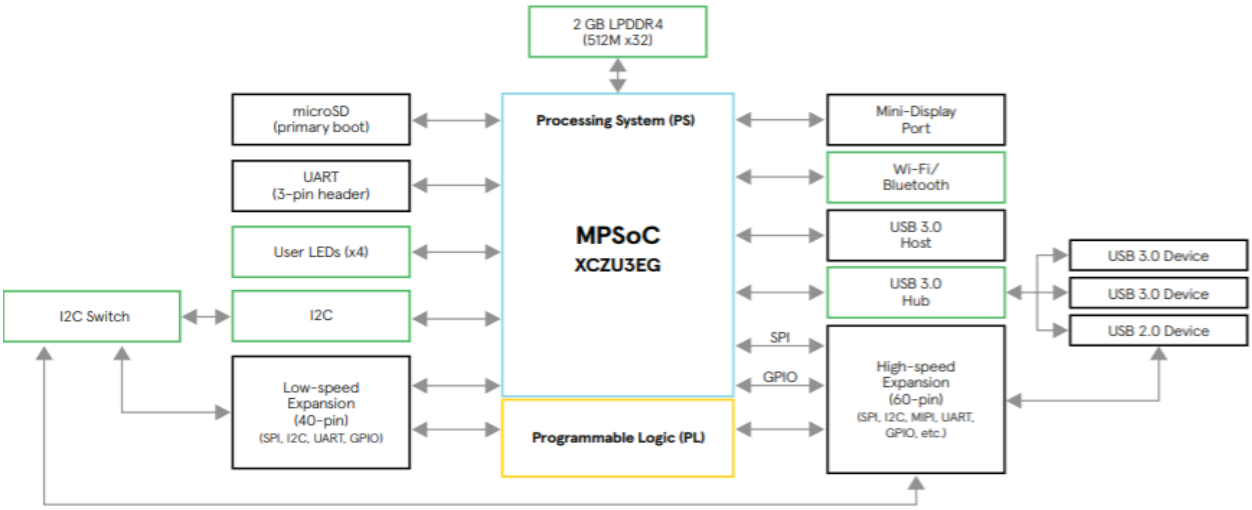
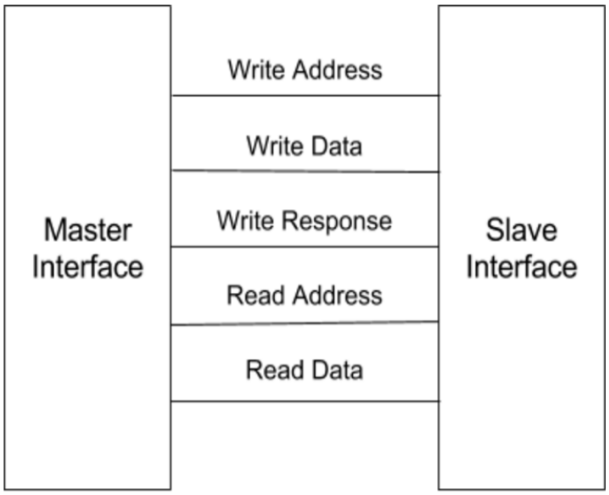


Figure 15. Ultra96 v2 Block Diagram

As shown in figure 14, the Ultra 96 v2 board used in this project has four AXI high-performance slave ports (HP0-HP3) and four AXI general-purpose ports (GP0-GP1) for PS-PL interfacing. It also provides 40-pin 96Boards low-speed expansion header and 60-pin 96Boards high-speed expansion header to interface with peripherals.

4.2.2 Advanced eXtensible Interface (AXI)



Channel connections between master and slave interfaces

Figure 16. AXI Channel Connections

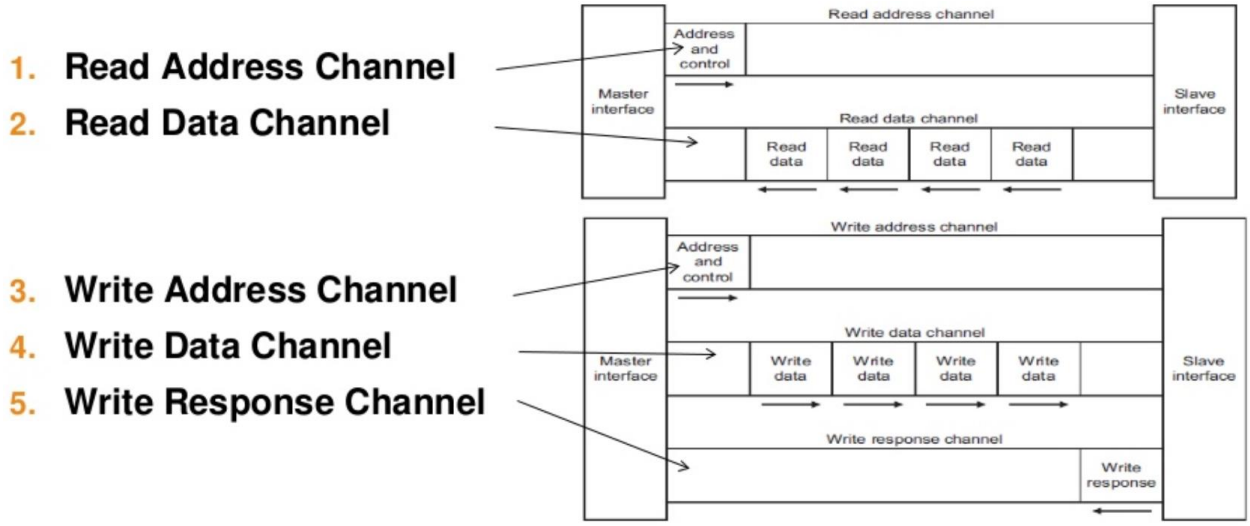


Figure 17. Basic AXI Signaling

The Advanced eXtensible Interface (AXI) is part of Advanced Microcontroller Bus Architecture (AMBA) and is a point to point interconnect designed for high-performance and high-speed microcontroller systems. The specifications of the protocol are:

- Before transmission of any control signal/address/data, both master and slave must extend their “hand” for a handshake via ready and valid signals.
- Separate phases exist for transmission of control signal/address and data.
- Separate channels exist for transmission of control signal/address and data.
- Burst type communication allows for continuous transfer of data.

In addition, Xilinx provides a library of AXI based Intellectual Property (IP) cores which are preconfigured logic functions. These IP cores have been validated and rigorously tested by Xilinx and have been optimized for Xilinx FPGAs.

4.2.3 FPGA Datapath

As shown in figure 6, the FPGA receives low-voltage differential signals (LVDS) from the ADC. These include the serial data bits, the frame clock and the bit clock. The frame clock (FClk) is a digitized and phase-shifted version of the ADC’s sample clock while the high-speed bit clock (DCLK) is a 90° phase-shifted signal to the data. In addition, these low-voltage differential signals need to be buffered and deserialized. They also have to be converted to the AXI format to be transferred to memory. Then, the data (AXI packet) is communicated to a first-in first-out (FIFO) buffer for clock domain crossing (CDC) from the ADC’s sampling clock frequency to the global FPGA clock domain. Lastly, the data is transferred to the PS memory.

4.3 Server and GUI

4.3.1 Server and Back-End Software

The Ultra96 board allows multiple ways to connect to a network capable computer. Either when the board is connected to a computer through WiFi or when plugged into a computer via USB, the Ultra96 is recognized as a network card. These network connections easily allow for a web server to be hosted on the Ultra96. A web server is a program that uses established networking protocols to host files and data, as well as process and service client requests from networked computers. The Ultra96 comes with a pre-built Jupyter Notebook web server on board that can be configured using Python 3. However, the Ultra96 also has an Apache web server installed on it as well which is more suitable for our needs since the code for WaveformsLive currently uses Apache Cordova.

4.3.2 GUI

By default, the Ultra96 runs PetaLinux. This is an OS often used for implementing embedded linux operations on xilinx products. With this linux distributions the user can connect the Ultra96 to a monitor through the mini display port and you essentially have a mini linux computer at your disposal with 2Gb of ram. In addition to being a small linux machine the Ultra96 is also a programmable microcontroller with GPIO pins that can be used to interface with the outside world. It is because of these 2 features that we plan on implementing a GUI that will serve the Ultra96. This GUI will be launched by connecting the Ultra96 to the computer and then connecting to its IP from your computer's browser. Once connected to your computer, the local version of a modified waveforms live will begin execution. The GUI that is launched is a modified version of WaveformsLive that includes the Ultra96 as an option for users to select. This allows the user to interact with an already known interface with some additional support for our specific project. None of the WaveformsLive GUI overall design is to be modified but rather we plan to change the way the data is being processed so that we can ensure the correct visualization of the Ultra96 data.

To run the GUI locally you must be on a linux environment and have the following software packages:

- NVM 0.35.1 (Node Version Manager)
- NPM 3.11.0 (Node Package Manager)
- Node 6.11.0 (Node.js)
- Cordova 4.2.0 (Framework for the Application)
- Ionic 2.2.2 (Frontend Software Development Toolkit)

Node Version Manager is a package that lets you select which version of NPM and Node your computer will use when building and launching the GUI. This package can be installed using this command:

wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.1/install.sh | bash

After that you can use NVM to install different versions of the Node and NPM with the command:

```
nvm install node v6.11.0
node -v
```

This should return “v6.11.0” which means you have correctly installed the packages. Node is an open source, cross platform javascript runtime environment that the machine needs to actually execute the javascript code. This runtime environment is built on Chrome’s V8 javascript engine. Node also utilizes a lightweight event-driven, non-blocking I/O model. Normally I/O bursts are blocking which prevent other things from happening while the I/O burst is processed. To get around this most systems need to use multiprocessing so that there can be multiple threads/processes running the I/O bursts. Node however, allows for non blocking I/O bursts which means that requests can be initiated and handled in parallel. This also removes the need for multiprocessing. A simplified diagram can be seen below.

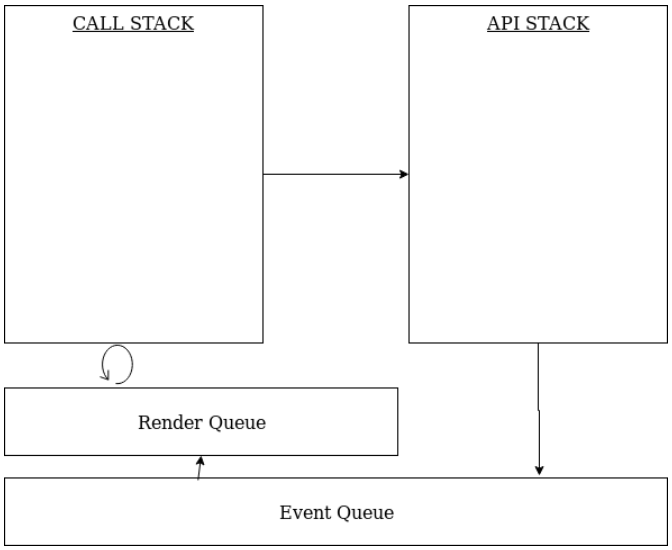


Figure 18. Call and API Stacks, and Render and Event Queues

The diagram above shows how if a function or chunk of code needs to be executed it would first be put on the stack where if it did not need to reference an API it would be immediately implemented. If a function in the call stack needed to be referenced with an API then it would be pushed into the API stack where it could then look up the necessary API. This allows for more functions to be put on the call stack in parallel and executed. Once the function has been looked up with an API it is then put into the event queue. This event queue holds the API calling function execution until the stack is empty. Once the stack is empty the API calling functions are pushed into the

render queue for rendering. The render queue will also check to make sure the call stack is still empty and if the stack is still empty the API function will be executed. This allows for the stack to not be clogged up with functions that need to do a lot of time consuming API references. This structure is meant to interleave API function executions in between regular function executions so that the user is able to interact with the GUI while something like a waveform is being drawn.

In order to run the GUI you must follow the steps below:

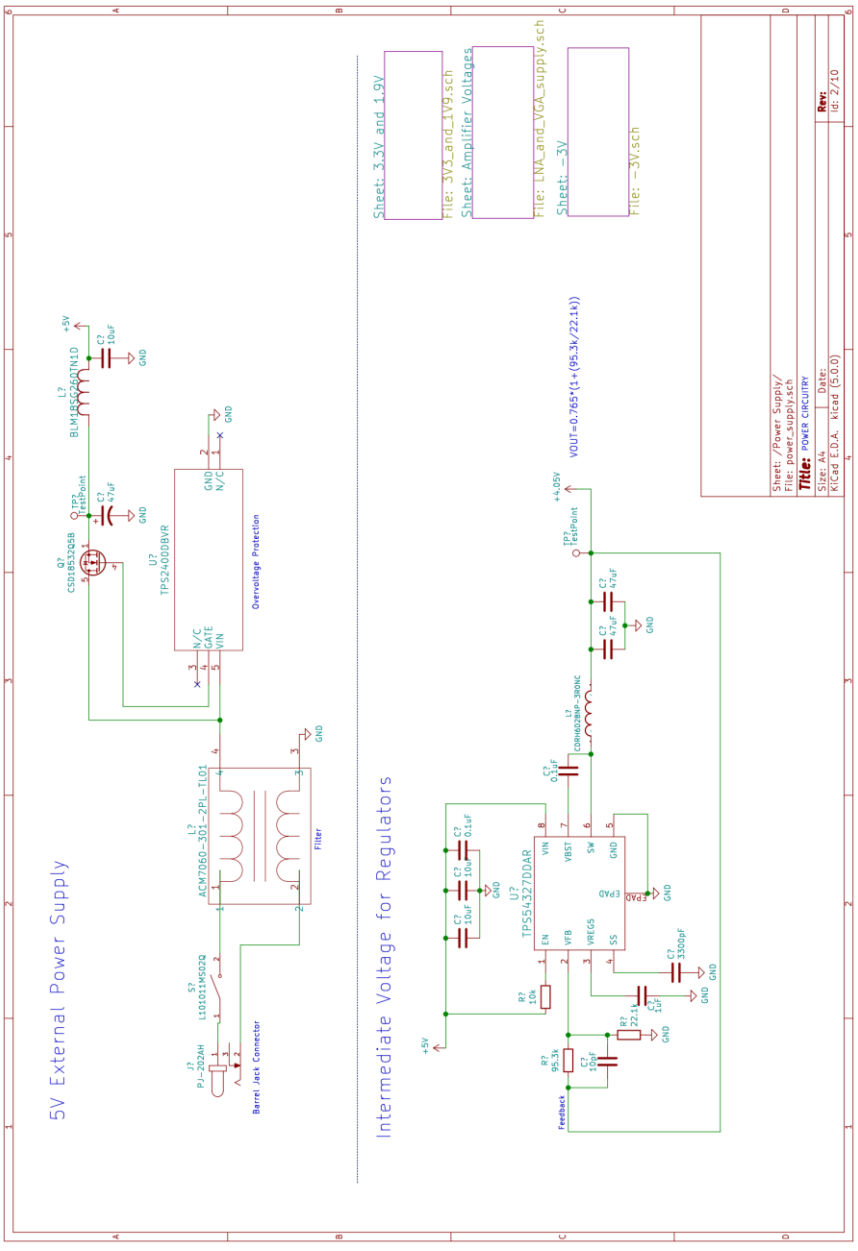
- ssh into the Ultra96 using “ssh xilinx@192.168.3.1” in your terminal.
- Change directory to the waveforms-live directory with “cd waveforms-live”
- Run “npm install”
- Run “ionic serve” (this command boots up a development server on the localhost, which in this case would be the Ultra96)

Once you have run the commands above the terminal will start launching the GUI. It takes about 30 seconds to have the GUI launched. Once launched you can go to your local computer’s web browser and enter “192.168.3.1:8100”. This will take you to the Ultra96 hosted GUI where you can start interfacing with the oscilloscope.

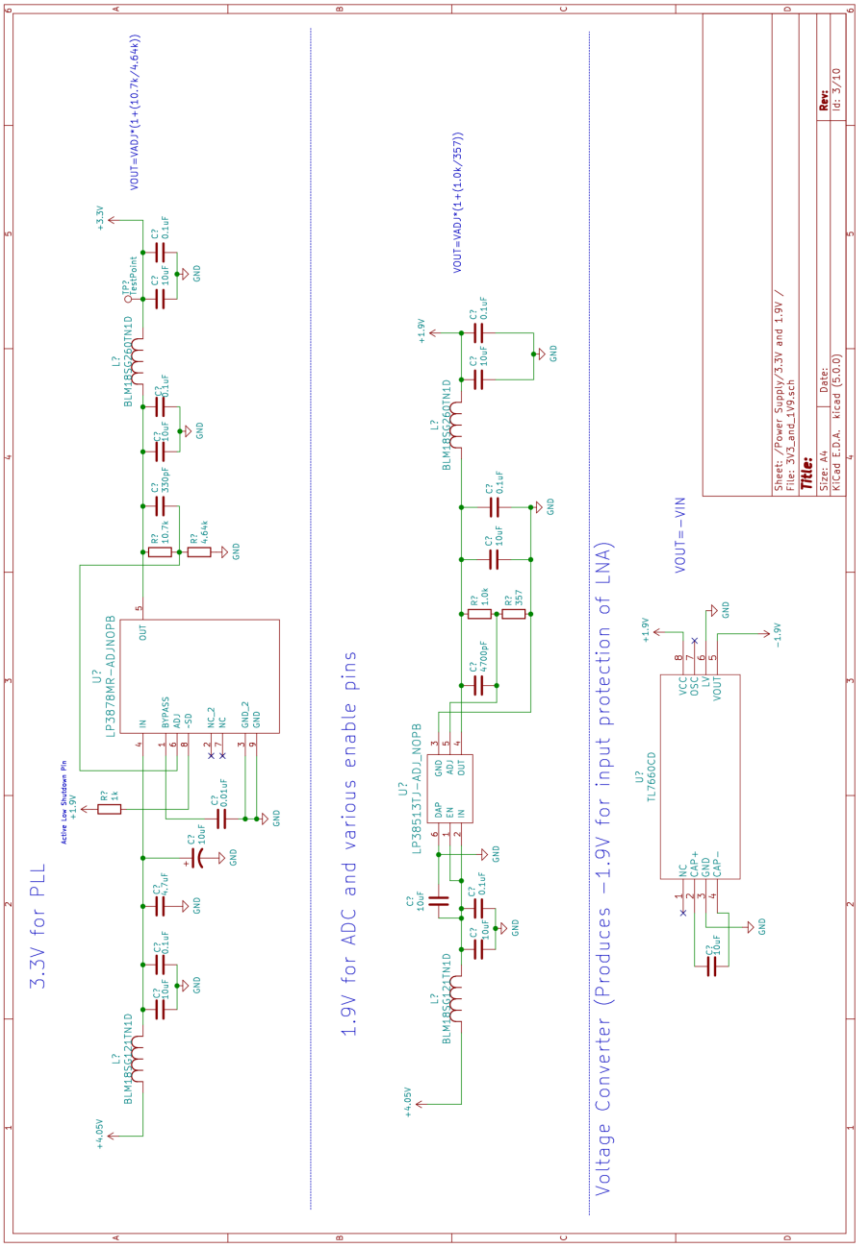
5. Detailed Design

5.1 Analog Front-End Schematics

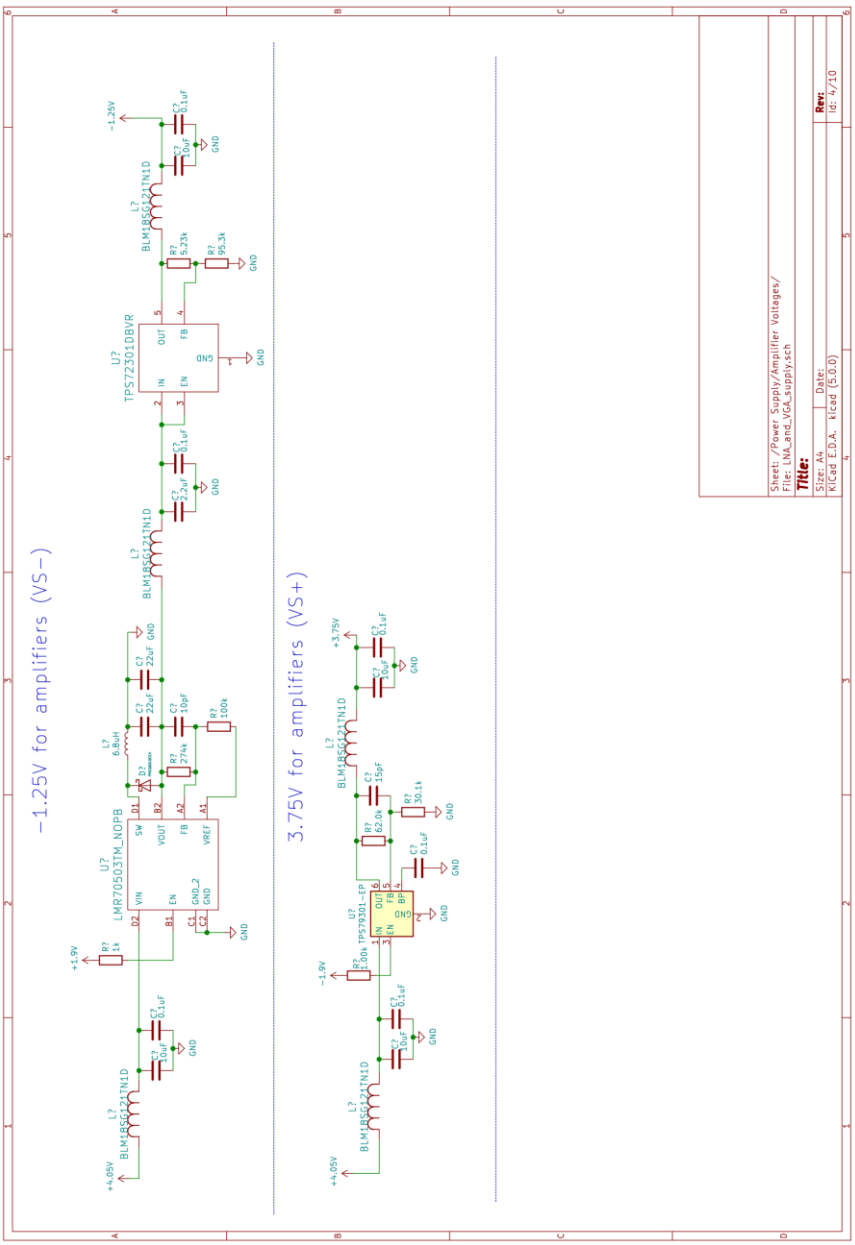
5.1.1 Power Circuitry 1



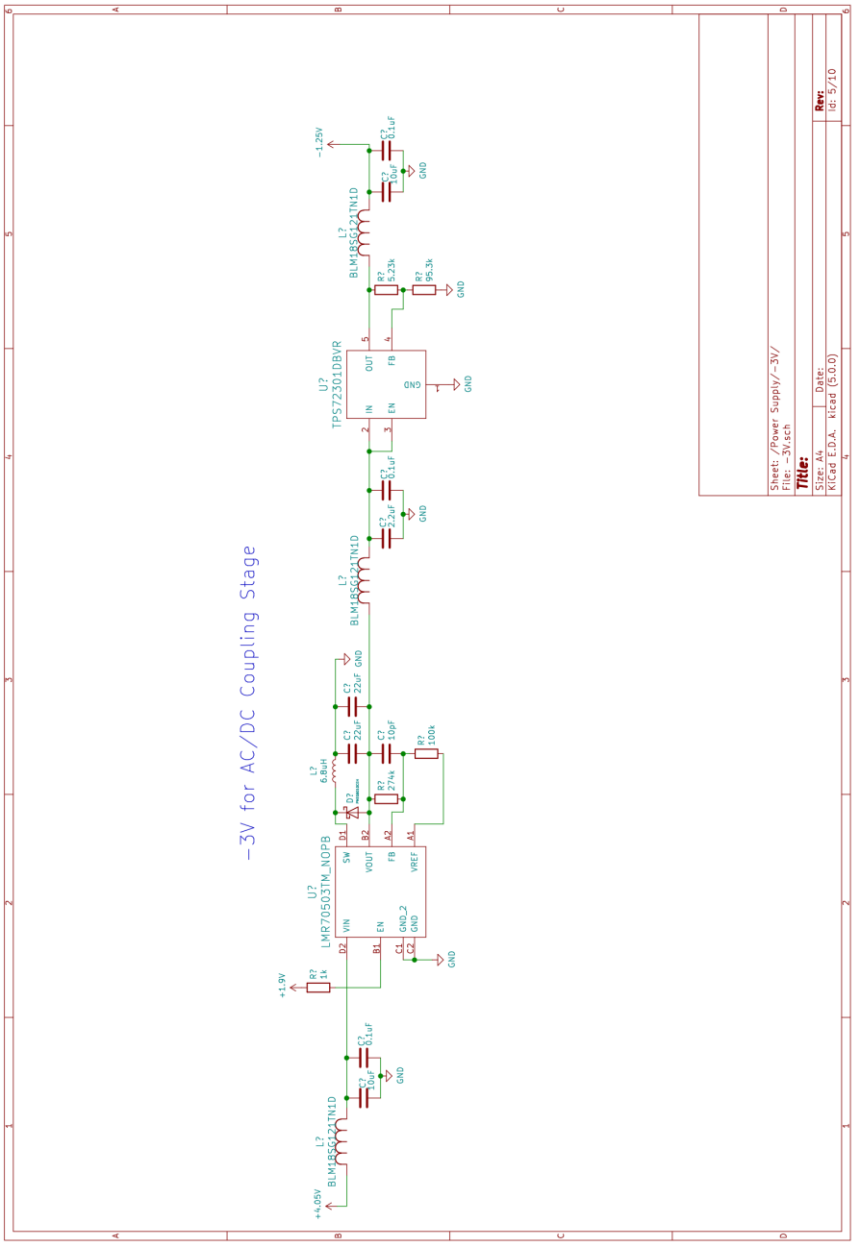
5.1.2 Power Circuitry 2



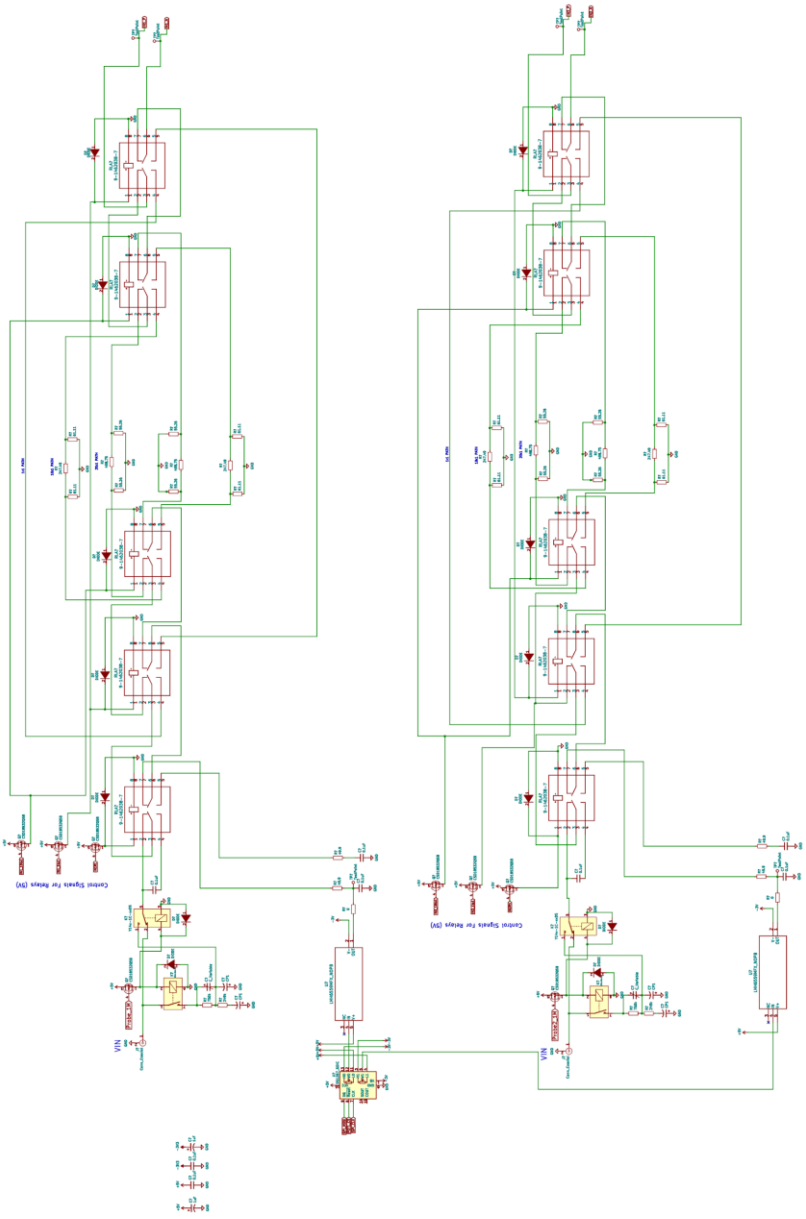
5.1.3 Power Circuitry 3



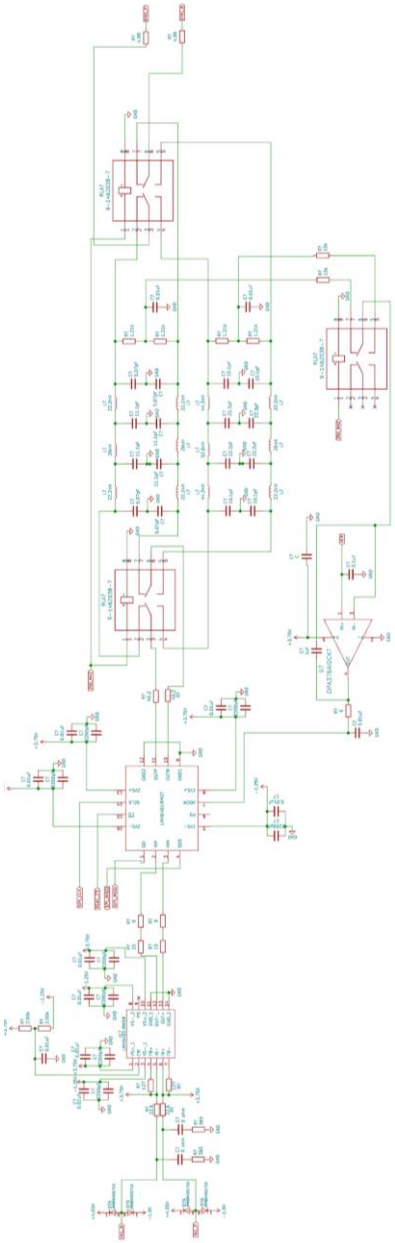
5.1.4 Power Circuitry 4



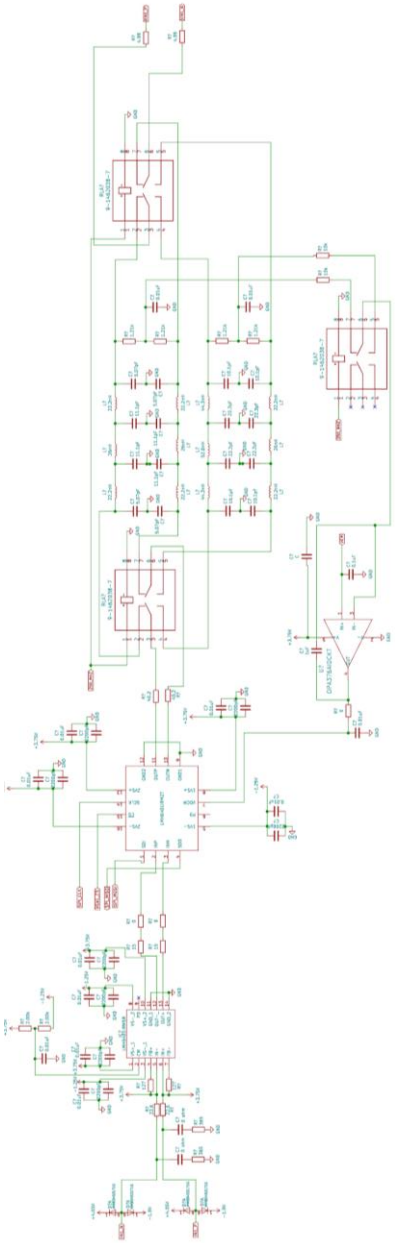
5.1.5 Input Attenuation Stage for Analog Inputs (note: page cropped for visibility)



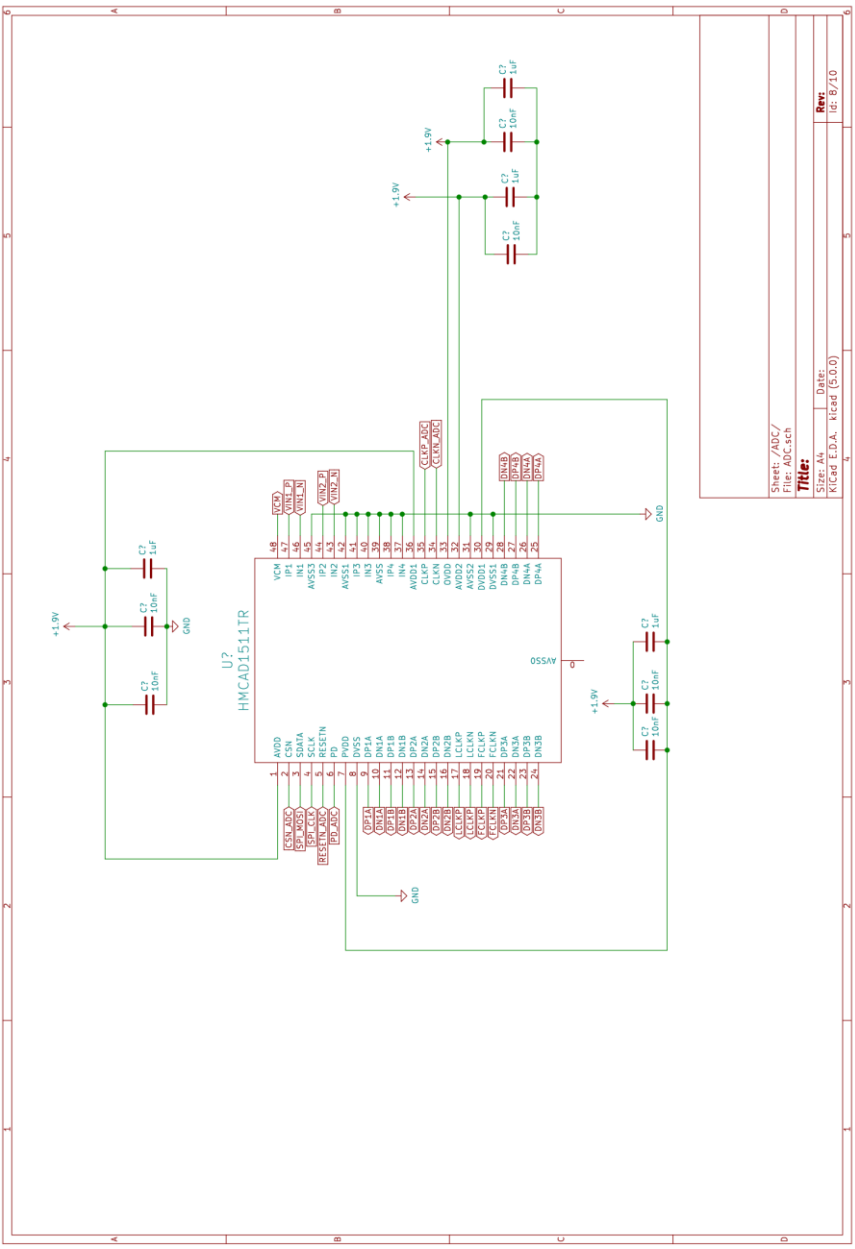
5.1.6 Amplification and Filtering Stage for Analog Input 1 (note: page cropped for visibility)



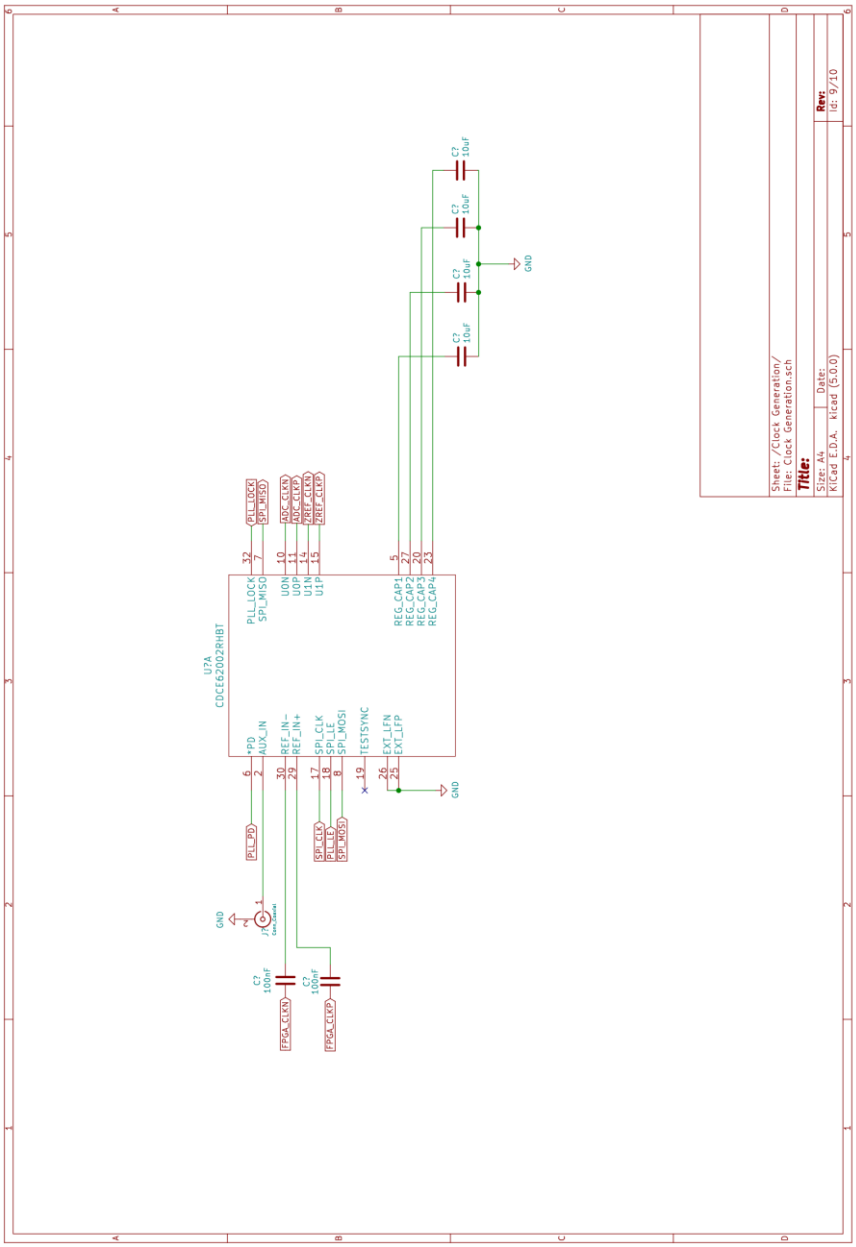
5.1.7 Amplification and Filtering Stage for Analog Input 2 (note: page cropped for visibility)



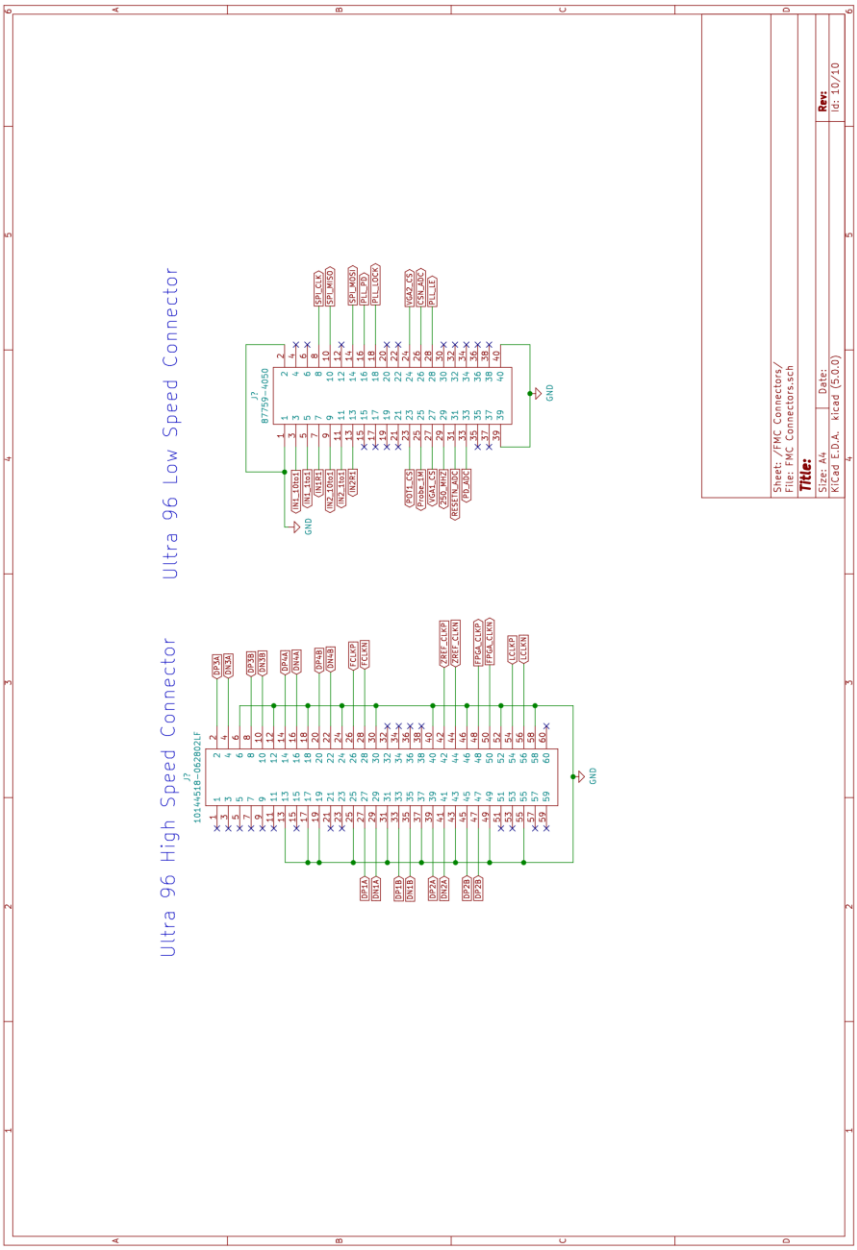
5.1.8 ADC Schematics



5.1.9 PLL Schematics



5.1.10 Ultra 96 SoC Connectors



5.2 Analog Front-End Component Selection

The component selection process for the major elements in the OSHO circuit consisted of comparing multiple different commercially available components. In order to select the ideal parts for this design, a compromise was made between the price as well as performance. The comparison between some of the top choices is listed below for each part (components in bold are the ones that were selected):

5.2.1 Switching Circuit Elements

ADG936 Wideband Switch	1.65V_{DC} – 2.75V_{DC} Power Supply Insertion Loss= 0.9dB (1GHz) Contact Form: Dual SPDT Absorptive and Reflective Options Available Low Power Consumption (1uW) Cost=$3.20
Teledyne Series High Speed Relay (A150-20-5)	5 V_{DC} Coil Voltage Insertion Loss=0.1dB (1GHz) Relay Contact Form: SPDT 1-20dB attenuation available (no need for external resistors) Operating Power<1W Cost= $71.74
Panasonic RE High Speed Relay (ARE104HC90)	4.5 V-$_{DC}$ Coil Voltage Insertion Loss ≅ 0.2dB (1GHz) Relay Contact Form: SPDT Operating Power= 200mW Cost= $5.96
Axicom High Speed Relay (IM43CGR)	**5 V_{DC} Coil Voltage** **Insertion Loss ≅ -0.33dB (900MHz)** **Relay Contact Form: DPDT** **Operating Power=100mW** **Cost= $2.93**

5.2.2 Phase Locked Loop

CDCE62005 (Clock Generator/Cleaner)	5 Configurable Outputs(differential) Output Frequency Range: 4.25MHz-1.175GHz SPI Interface Up to 3 reference inputs Phase Detector Frequency=40MHz Dual VCO Architecture Integrated EEPROM to store default settings Cost=$9.98
LMK0482(Clock Jitter Cleaner w/ dual loop PLLs)	14 differential clock outputs Dual Loop PLL Architecture Maximum Output Clock Frequency 3.1GHz SPI Interface Up to 3 reference inputs Phase Detector Rate=155MHz Dual Low Noise VCOs Cost=$19.59
CDCE62002 (Clock Generator/Cleaner)	**2 differential clock outputs** **Output Frequency Range: 10.94MHz-1.175GHz** **SPI Interface** **2 Reference Inputs(1MHz-500MHz)** **Phase Detector Frequency=40MHz** **Dual VCO Architecture** **Integrated EEPROM to store default settings** **Cost=$7.82**

5.2.3 Variable Gain Amplifier

Differential Variable Gain Amplifier **LMH6401IRMZR**	**Voltage Gain: -6dB to 26dB** **Voltage Gain Step Size: 1dB**

	Differential Input Impedance :100 Ω **Input Voltage Range=-5.5 to 5.5V** **Maximum Input Difference=2.1V** **SPI Interface** **Cost=$19.52**
Dual Channel Variable Gain Amplifier LMH6882SQE/NOPB	Voltage Gain: 30dB to -9dB Input Voltage Range=-0.6 to 5.5V Voltage Gain Step Size: 1dB Input Impedance: 50Ω or 75Ω SPI Interface Cost=$12.65
Dual Programmable Differential Amplifier LMH2832IRHAT	Voltage Gain: 26dB to 6dB Input Voltage Range=-0.5 to 5V Voltage Gain Step Size: 0.25dB Input Impedance: 100Ω SPI Interface Cost=$19.02

5.2.4 Low Noise Amplifier

LMH5401IRMST	**Input Voltage Noise= 1.25 nV/√Hz** **Slew Rate=17,500V/ns** **6GHz Bandwidth with 12dB voltage gain** **Quiescent Current: 55mA** **Ideal for DC and AC-coupled applications** **Cost=$15.32**
ADL5561	Input Voltage Noise= 2.1 nV/√Hz Slew Rate=9.8V/ns Max Voltage Gain: 15.5dB Gain Accuracy= ±0.15dB Quiescent Current: 40mA Cost=$9.02
ADL5569 (Dual Differential Amplifier)	Input Voltage Noise= 1.0 nV/√Hz a Slew Rate=24V/ns Max Voltage Gain:20dB Gain Accuracy= ±0.15dB Quiescent Current: 86mA per Amplifier Cost=$37.20

5.2.5 Analog to Digital Converter

ADC08D1520CIY	Sample Rate	1.5 GSPS

B/NOPB	Bandwidth	2 GHz
	Resolution	8 Bits
	ENOB	7.4 bits @748Mhz
	Price	$ 563.00

ADC08D1020CIY B/NOPB	Sample Rate	1GSPS
	Bandwidth	2.0 GHz
	Resolution	8 Bits
	ENOB	7.4 Bits @498MHz
	Price	$ 388.57

HMCAD1520	Sample Rate	1GSPS
	Bandwidth	700MHz
	Resolution	8 Bits
	ENOB	N/A
	Price	$ 113.18

HMCAD1511	**Sample Rate**	**1GSPS**
	Bandwidth	**650 MHz**
	Resolution	**8 Bits**
	ENOB	**7.9 Bits**
	Price	**$ 64.76**

5.3 FPGA Datapath Design

This project will leverage the preconfigured Xilinx IP cores for the majority of the datapath. This will lead to the completion of the task in an efficient and timely manner. As shown in the following figure, the ADC digitizes the input signal and outputs the 8-bit sample, along with the bit-clock and the frame-clock.

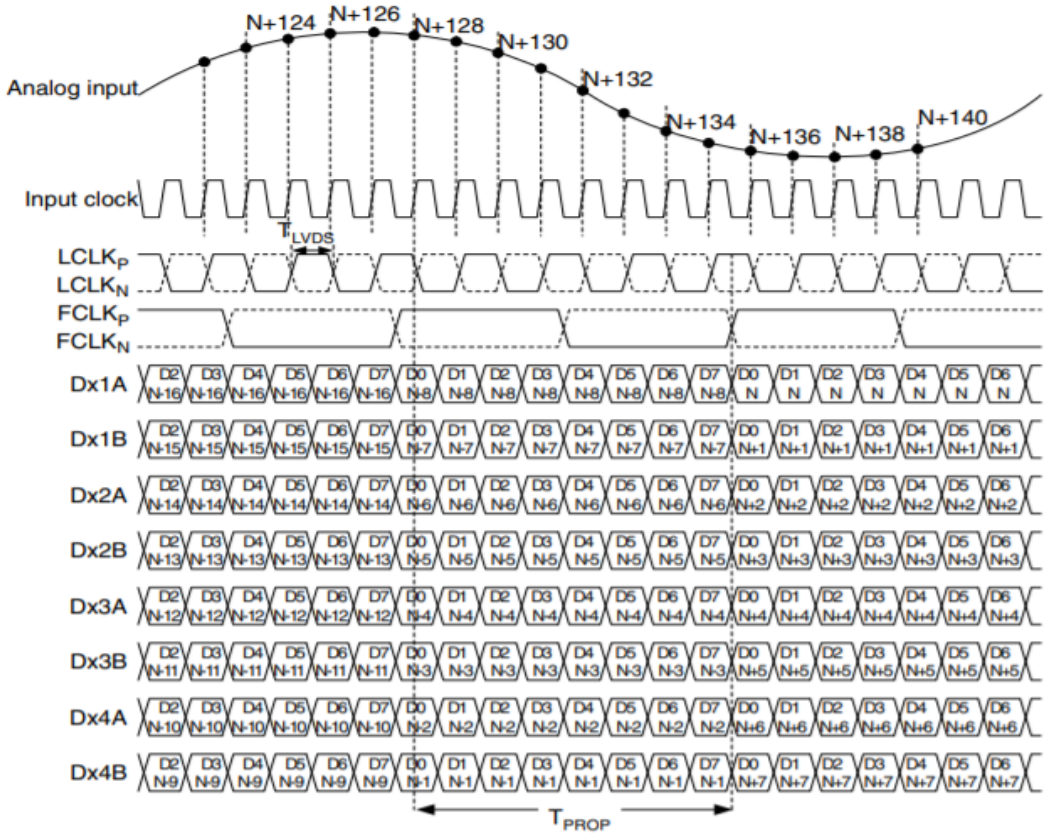


Figure 19. Single channel - LVDS timing 8-bit output of the HMCAD1511 ADC

Furthermore, the frame clock rising edge transitions are aligned with the framing ADC data bits (D0 and D7). This alignment helps a deserializer to correctly load parallel data after de-serialization. In addition, the bit clock is typically center-aligned and both clock edges are used to latch serial ADC data. Therefore, the bit clock is referred to as a double data rate (DDR) bit clock.

Within the Ultra96, when a bit is routed through a clock-capable I/O, BUFIO buffer, and/or BUFR clock buffer, it experiences a different amount of delay than the data and frame signals. Therefore, the phase relationship between the signals is lost. To compensate for this, the bit clock has to be realigned to the data and frame signals.

5.3.1 Bit Clock Alignment

Xilinx provides application notes on utilizing their IP cores for various needs. One such application note also discusses clock alignment. The complete bit clock alignment setup within the deserializer core is shown in the following figure.

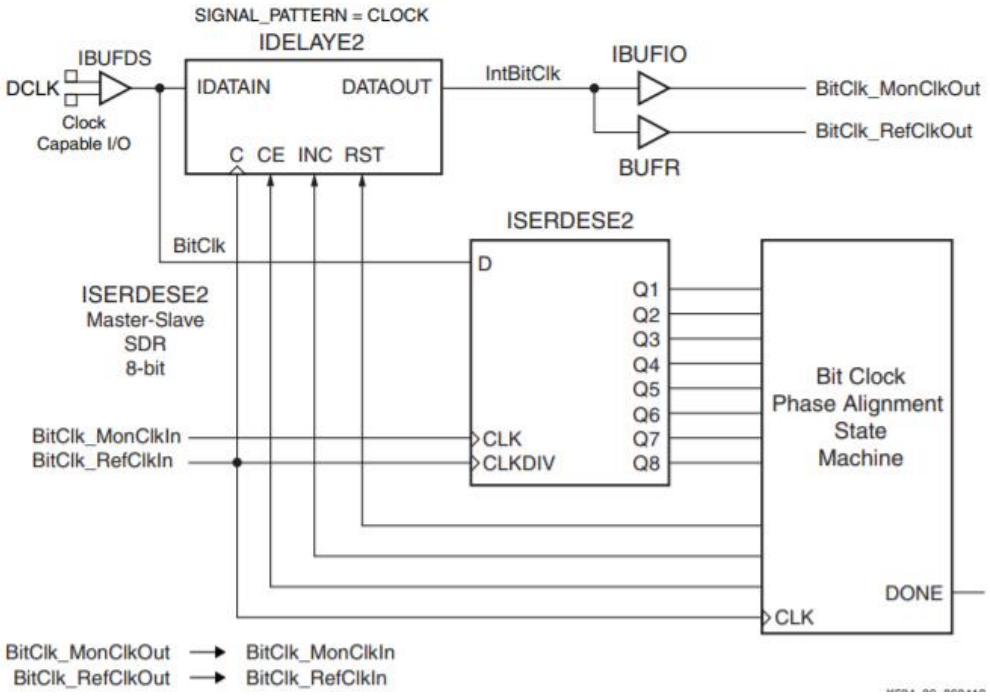


Figure 20. Bit Clock Alignment Setup

The bit clock (DCLK) from the ADC is routed through an IDELAYE2 core to two buffers. It is also sent to the D input of the ISERDESE2. This technique allows the determination of the position of the rising and falling edges of DCLK. The Bit Clock Phase Alignment state machine monitors the ISERDESE2 outputs and the deserialized and parallel captured clock bits. When all captured bits are equal (i.e., all 0s or all 1s), the state machine changes the delay of the IDELAYE2 core to align the internal clock to the external clock.

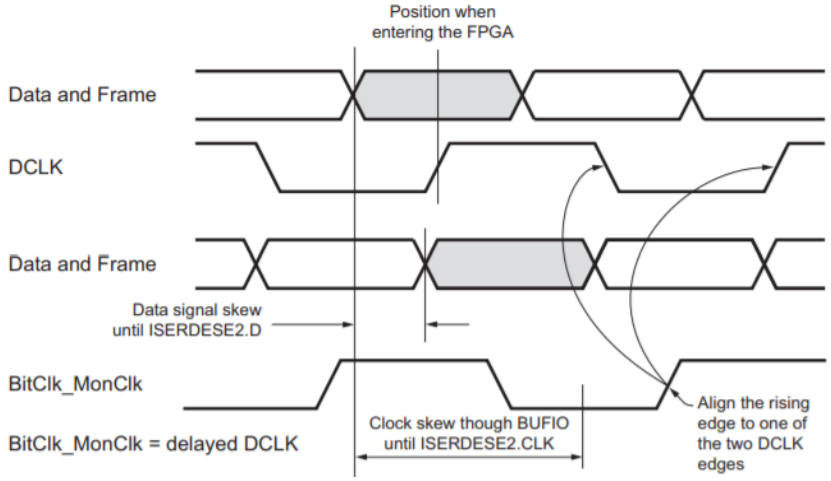


Figure 21. Clock Skew through the Buffers

5.3.2 Frame Clock Alignment

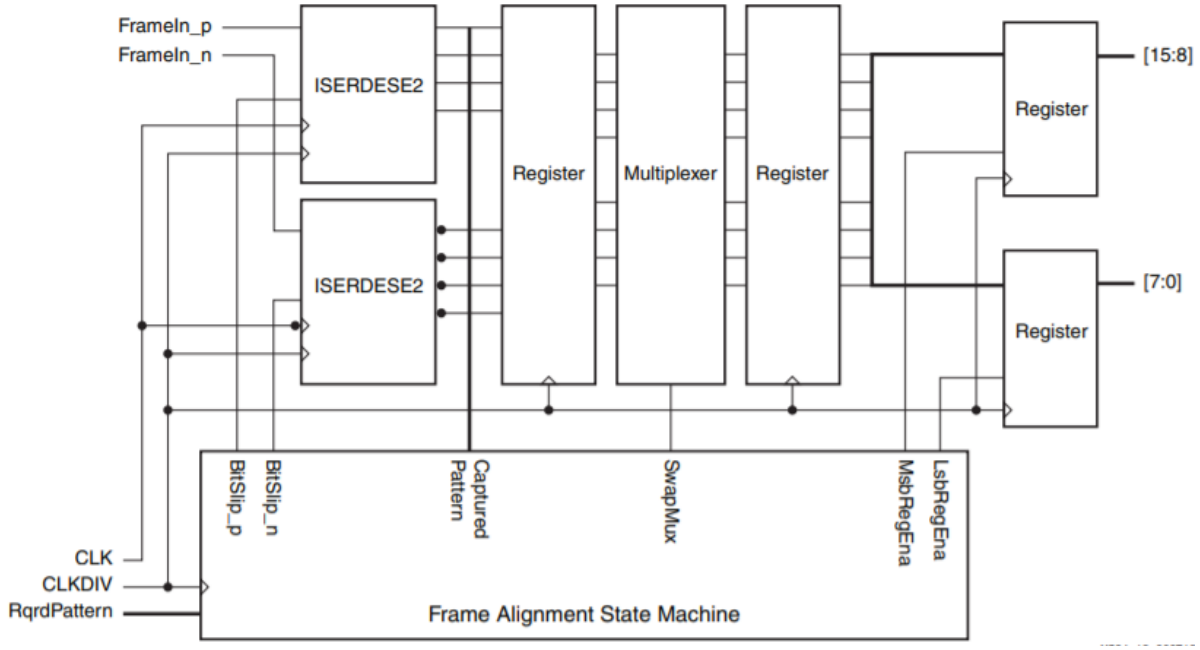


Figure 22. Frame Clock Alignment Block Diagram

After the bit clock (DCLK) has been properly aligned, the frame clock pattern discovery is begun. The LVDS frame clock from the ADC is a digitized version of the sampling clock that is phase aligned with the data. As shown in figure 22, the ISERDESE2's output is compared to a fixed value representing the expected frame clock pattern, which is "11110000" for an 8-bit ADC. If the output of the ISERDESE2 does not match the expected value, a bitslip operation is carried out on the frame and data signals. When this output is finally equal to the programmed pattern, the bitslip operation is stopped and the data and frame clock signals within the FPGA are considered valid. Next, the received data is aligned because it is shifted with the frame signal. Lastly, the bit clock and frame clock signals are used to capture and deserialize the data bits.

5.3.3 Post-Deserialization

After deserialization, the custom Deserializer IP core combines 8-bytes from the 8 input channels into a single AXI packet and sends it to the FIFO buffer. The FIFO buffer is being used for clock domain crossing from the ADC's sampling clock frequency to the global FPGA clock domain. Furthermore, the FIFO IP core uses the AXI stream protocol which does not need an address channel and is always used to write data in one direction. Therefore, a Xilinx AXI Direct Memory Access (DMA) core is utilized for high-bandwidth direct memory access between an AXI4-Stream target peripheral and the memory on the PS side. Lastly, an external trigger core will be added to the firmware. This will enable a simple AXI-stream source which will start storing and forwarding the data samples to the DMA block.

5.4 Software Design and Models

The ultimate goal is for us to have the GUI be wireless so that the user does not need to have the Ultra96 connected to their laptop to see the visualization. This would allow many people to see the same visualization. Also this result would be a more aesthetic and streamlined approach. For this ultimate goal to be achieved the Ultra96 needs to start up the server on boot so any computer on the same WIFI can type its IP into the browser to connect to it.Once the connection is made the user will use the GUI to generate control signals. These control signals will be sent back to the Ultra96 and processed. Once processed the Ultra96 will send back data to the GUI for it to be visualized. A diagram illustrating this process from the GUI and server side is shown below.

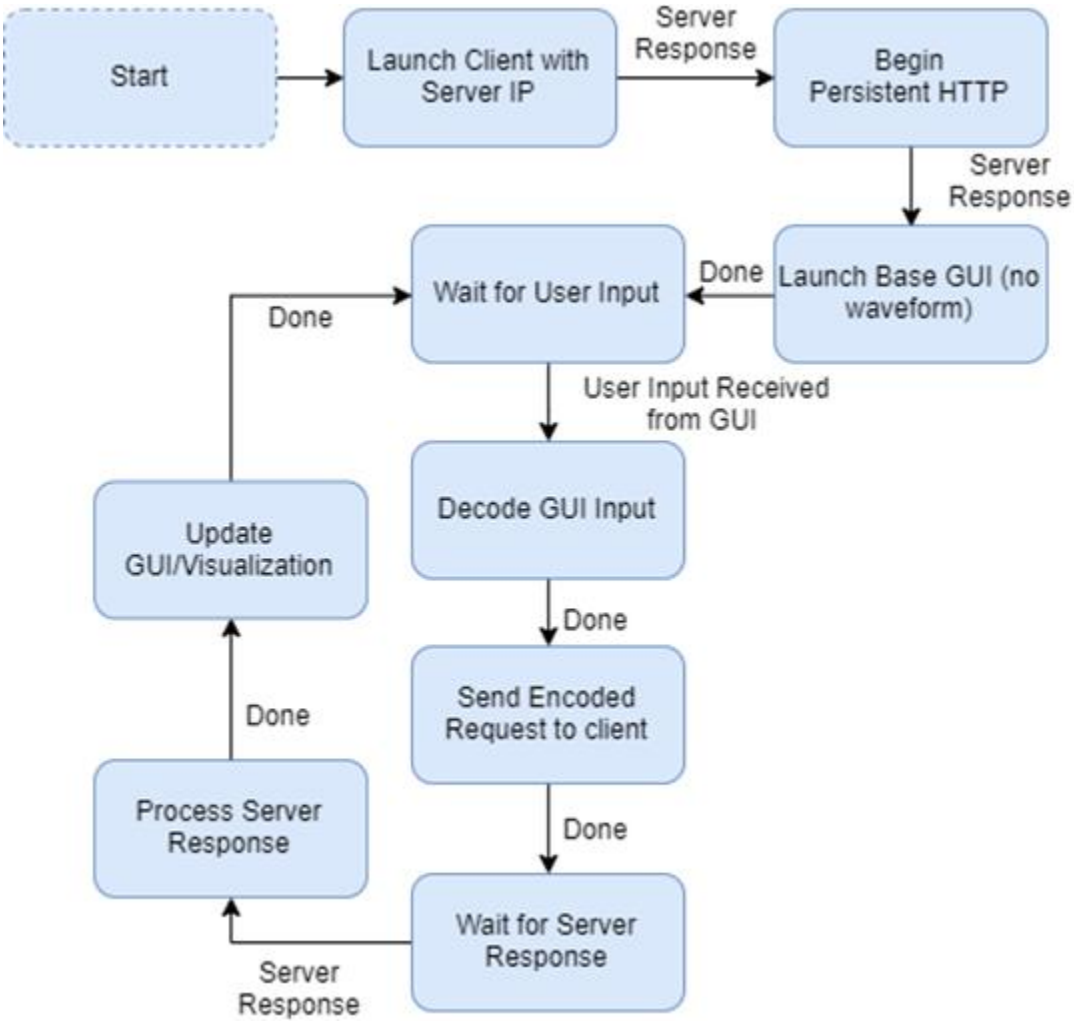


Figure 23. High Level GUI State Diagram

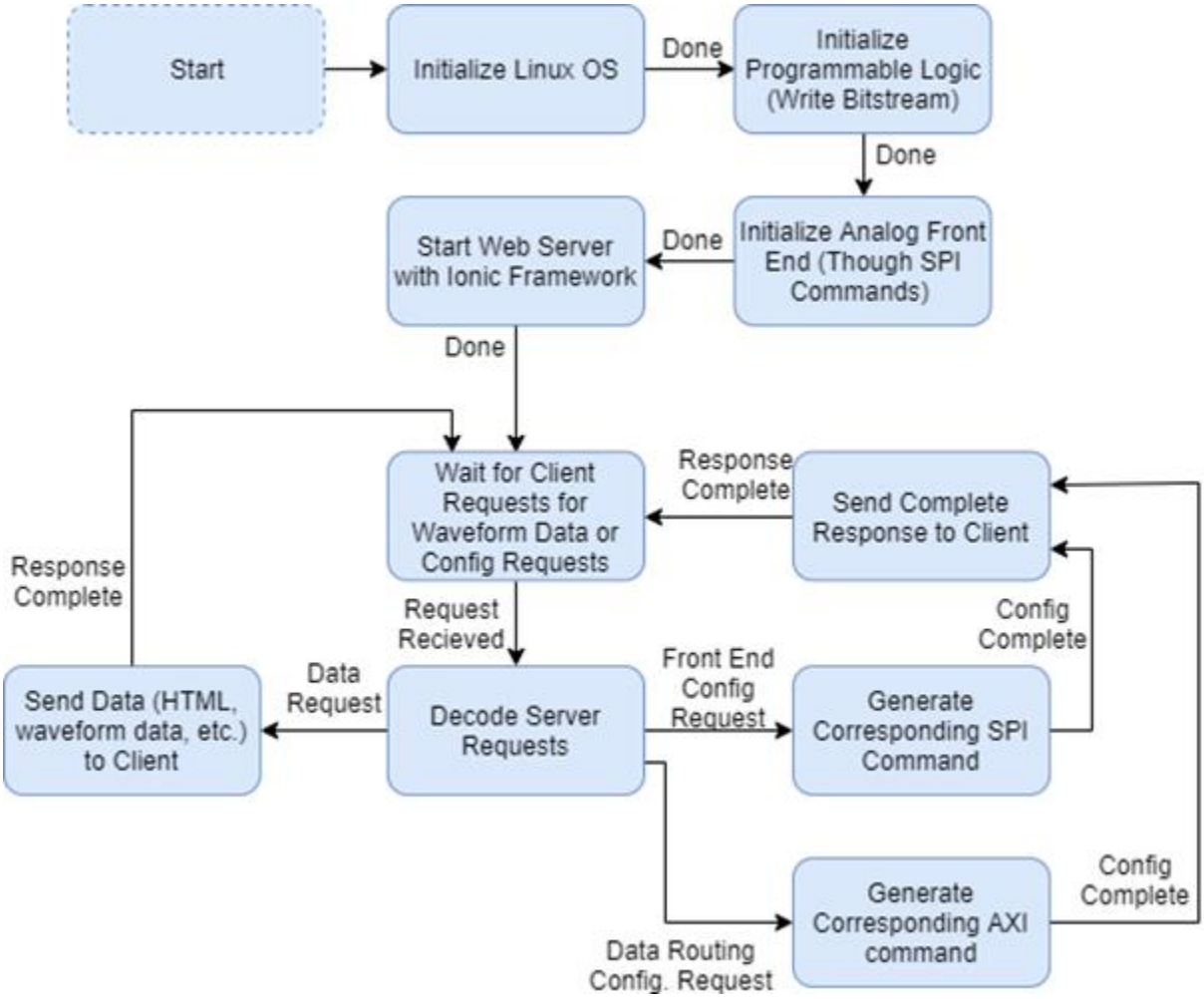


Figure 24. High Level Server and Backend Software State Diagram

This GUI is being designed with the Angular framework. This means that we will be using the Angular paradigm for the GUI design and will have higher flexibility due to the usage of a few key features.

Dependency Injection is one of these key features. Dependency injection allows for object dependencies to be put into an object that can be imported by the main object. This allows for objects and code to operate independent of its dependencies. The dependencies can be changed and updated without requiring the main object or code that rely on the dependency to need updating.

Another key feature is the use of typescript. GUIs can be made in a multitude of different languages but the Angular framework relies on the usage of typescript. This language can be thought of as a superset of javascript. The main difference between javascript and typescript is that type script is “heavily typed”. This means that javascript is like python where you do not need to declare types for values. This allows for easier code but can cause a lot of problems while running due to unpredicted interactions between variables of different types. Typescript requires every variable used to formally have a type. This prevents issues such as using the string “10” instead of the integer 10 in intermediate calculations. An example between variable declarations can be seen below.

Javascript:

Var voltage = getVoltage();

Typescript:
Let voltage : number = getVoltage();

The javascript does not verify the data type of the voltage we are fetching. This means that it is possible to store a string in the variable voltage. This would result in errors on runtime without much explanation. The typescript requires that voltage be of type number and will not let the code run if the getVoltage function has the possibility to return anything else. This prevents unexplained errors on runtime by raising them during compilation.

Another Angular feature that will be used is RxJS. This reactive programming library allows for the developer to execute operations on streams of data instead of waiting for the data to arrive and then operating on it. This will allow for a much more responsive GUI. The Angular paradigm revolves around being modular. This means that most objects in Angular are made so that they can have other objects inside of them. Having this nested structure of objects is very powerful because once an object is made it can become a building block for a more complex object. Additionally, with the dependency injection, objects are independent of their dependencies so if an object that is a building block of a more complex object is updated there is no need to update the more complex object as well.

With the Ionic framework that Angular easily integrates with, this GUI will also be available on mobile iOS and Android devices. Despite having a screen of a completely different screen, the GUI will still work because Angular allows for each component to have its own HTML and CSS file that control they display on various devices.

By breaking down the GUI into its graphical, computational, and hardware-dependent components the GUI can be much more modular and cleanly implemented. Below is a breakdown of some services and hardware dependent components that this GUI will rely on. Many of these components already exist in the WaveformsLive code but need to be modified to also provide support for the OSHO.

- Utility.service.ts:
  - Determines the proper prefix for the measurement (G,M,K,m,u,n)
  - Creates name for the logging device being used for the rest of the code
  - Add OSHO to possible device names for logging and rest of code
- Ui-helper-service.ts:
  - Determines when to disable and enable buttons on the oscilloscope control panel
  - Returns an error message when the user tries to interact with the disabled object
- Tooltip.service.ts:
  - Creates error messages by referencing a master dictionary of potential failures.
- Toast.service.ts:
  - Creates toast notifications (non clickable notification on bottom of screen) with variable time and message to notify user for specific event occurrences

- Storage.service.ts:
    - Manages interactions with backend SQL database
    - Gets data from SQL
    - Saves data to SQL
    - Removes data from SQL
    - Removes all data from storage
    - Saves listener settings for components
    - Loads listener settings for components
- Settings.Service.ts:
    - Loads device firmware onto data logging device
    - Saves local copy of data and log files
    - Sets up timeouts for data logging device
    - Exports CSV of data
    - Add OSHO firmware to list of firmwares to load
    - Create non AWS hosted firmware path handling
- Scaling.service.ts:
    - Manages Unit conversions for waveform display
- Logger-plot.service.ts:
    - Maintains data for time and voltage captures
    - Maintains plotting data
    - Manages data visualization window shrinking
    - Draws waveform
    - Updates divisions in for oscilloscope
- Loading.service.ts:
    - Displays loading messages
    - Add more messages to ensure user knows what operations are being performed
    - Create more GMU oriented branding for product
- Export.service.ts:
    - Creates exports for png and csv
- Device-data-transfer.service.ts:
    - Sets trigger levels for data capture
    - Selects trigger source
    - Selects data capture channels
- Unit-format.pipe.ts:
    - Manages unit conversion and final value display
    - Modify to allow for displaying of full VPP value and increased bandwidth
- Device-manager-page.ts:
    - Maintains the different hardwares user can select
    - Add OSHO as option for collecting data from
    - Add OSHO firmware reference with non AWS path handling
- Device-manager.model.ts:
    - Manages plotting of waveform through WiFi connections
    - Implement OSHO WiFi configuration
- Pages/logger:

- Maintains all data logging hardware and their respective configurations and displays
- Add OSHO data logging device
- Create configuration file for OSHO communication
- Create HTML file for OSHO GUI displaying
- Create SCSS file for OSHO display customization
- Create OSHO Module for all parts of the code to reference for OSHO usage

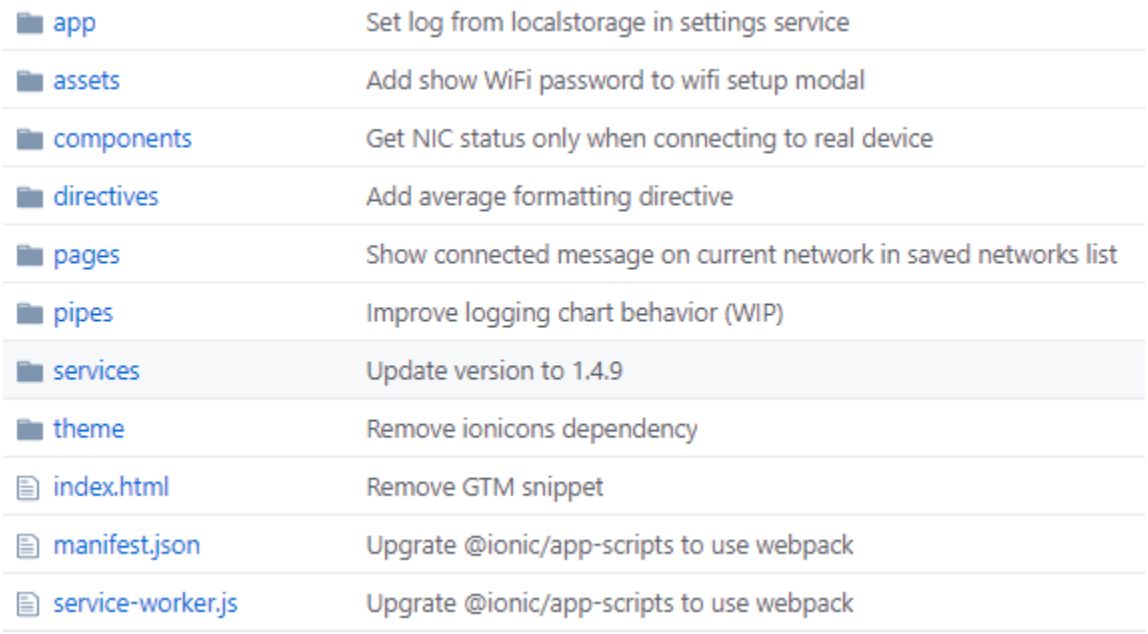


Figure 25. Hierarchy of code project broken into main components of project

6. Prototyping & Early Testing Progress Report

6.1 Analog Front-End HACD Board Testing

The preliminary testing of the power circuitry on the HACD PCB uncovered multiple errors in design and implementation. Some of these errors are listed below:

- Ferrite bead resistance mismatch
- Negative voltage regulator used instead of positive regulator

Furthermore, these errors were corrected by ordering new components and soldering them onto the board. On the other hand, some early prototyping progress has also been made on the OSHO circuit design. The newly designed attenuator and Chebyshev LPF circuits were simulated in PSPICE and the results are presented below.

6.1.1 10:1 Attenuator Path Simulation

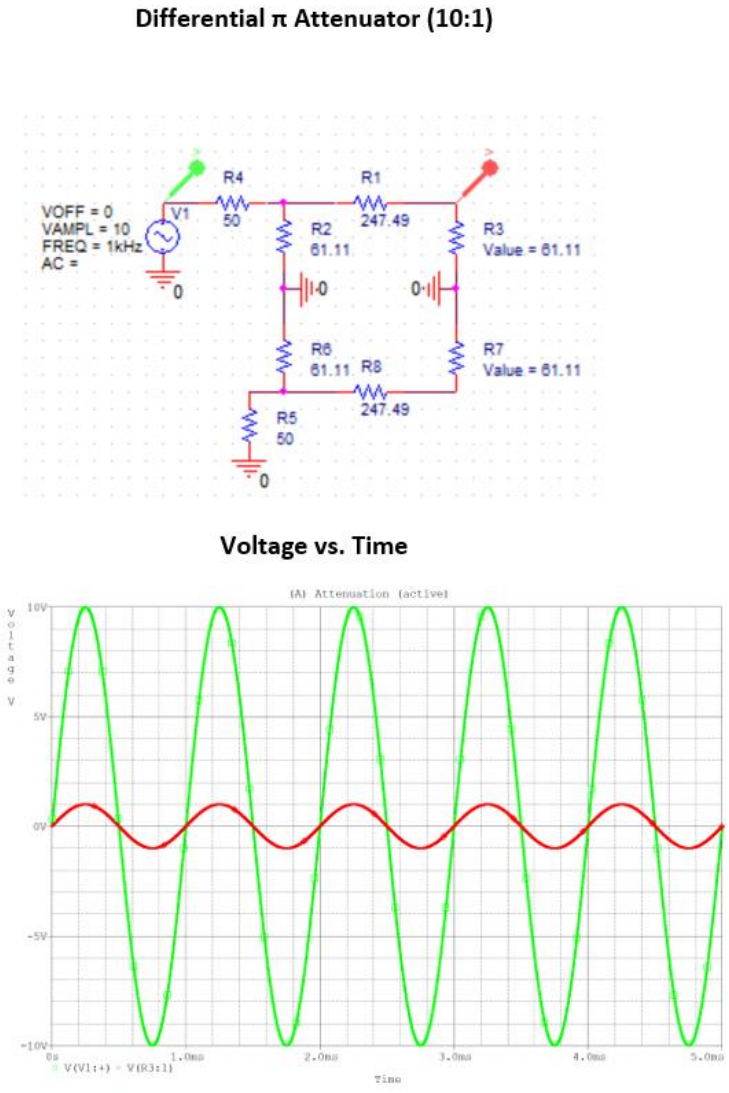


Figure 26. 10:1 Differential Pi Attenuation Simulation

6.1.2 20:1 Attenuator Path Simulation

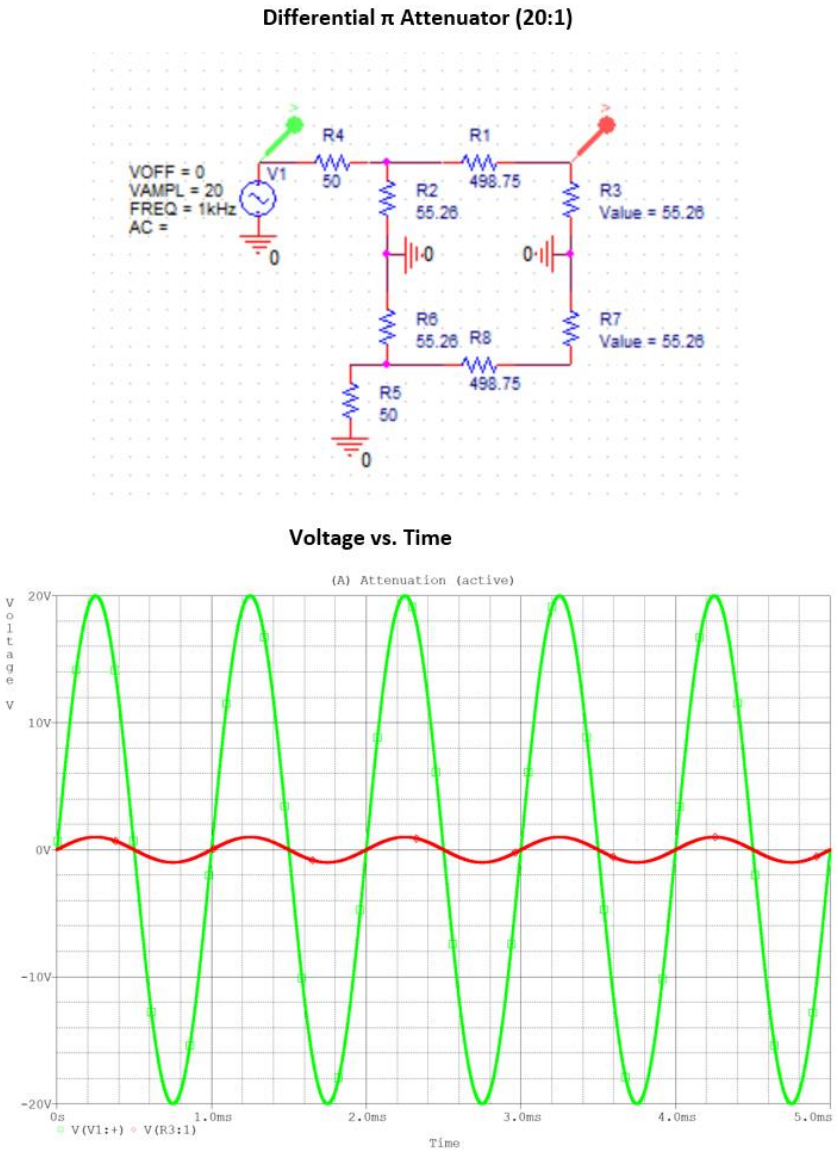


Figure 27. 20:1 Differential Pi Attenuation Simulation

6.1.3 LPF simulation (500MHz cutoff frequency)

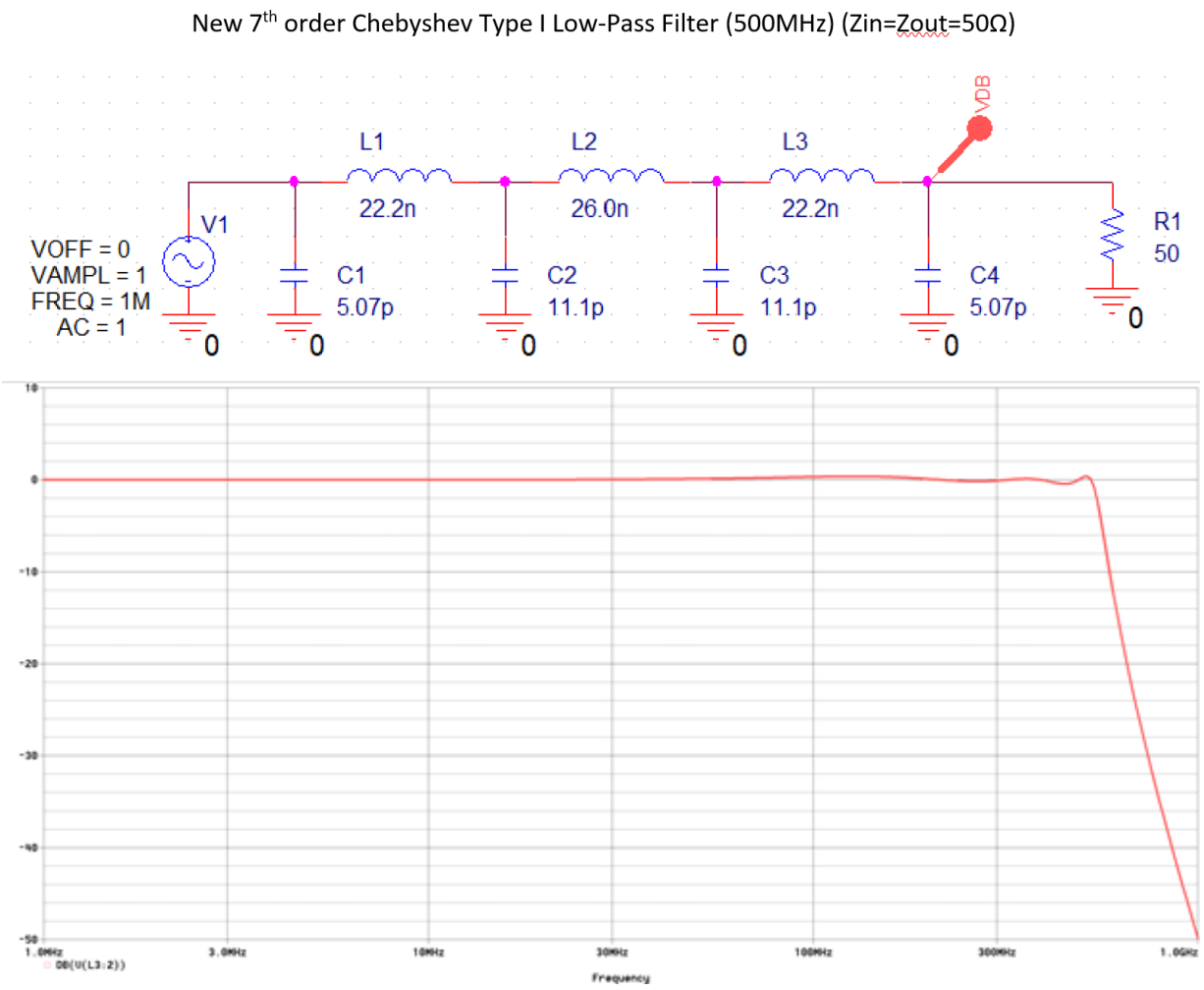


Figure 28. 500MHz Chebyshev Low Pass Filter Simulation

6.1.4 LPF simulation (250 MHz cutoff frequency)

New 7th order Chebyshev Type I Low-Pass Filter (250MHz) (Zin=Zout=50Ω)

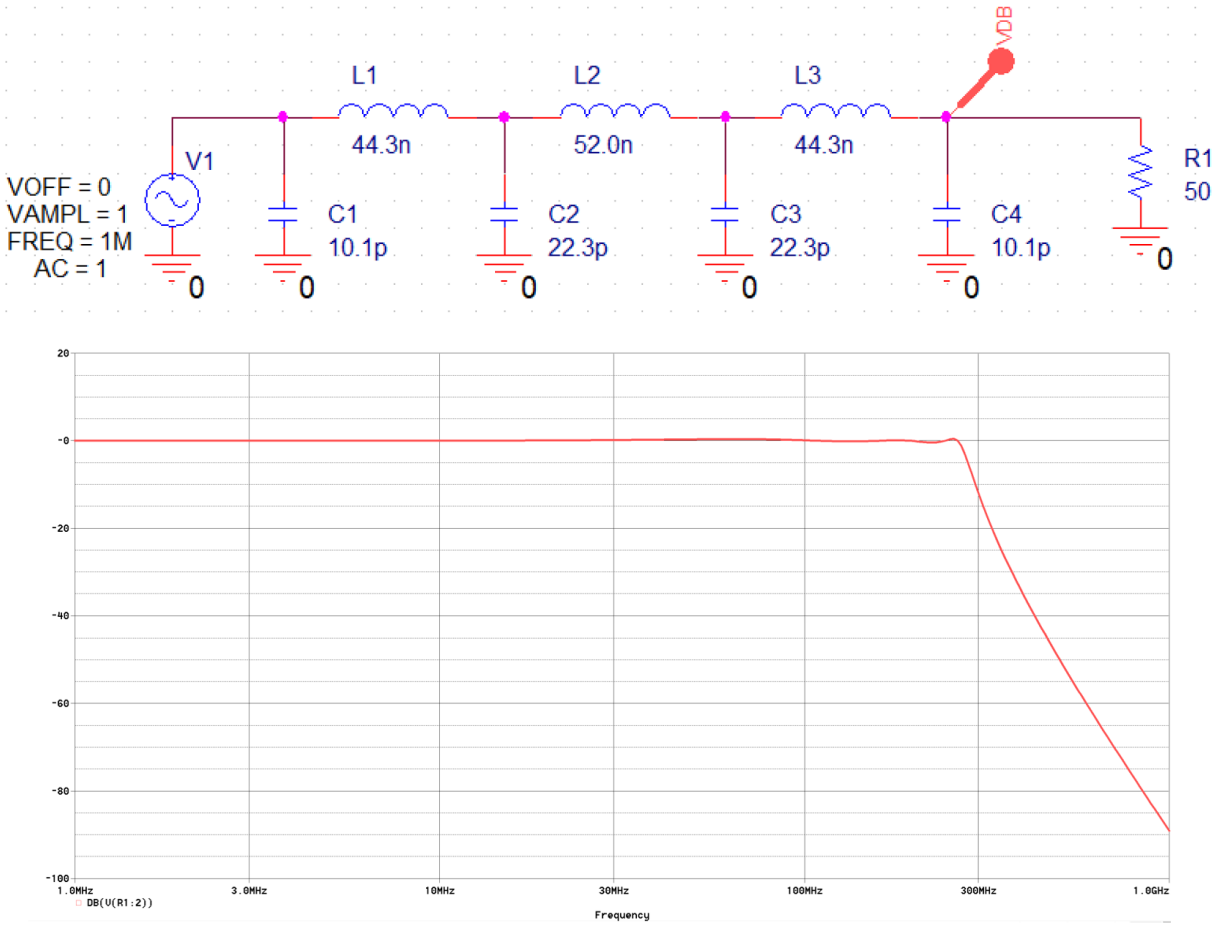


Figure 29. 500MHz Chebyshev Low Pass Filter Simulation

6.2 VHDL Firmware Testing Progress

6.2.1 Vivado Project and Xilinx Zedboard Testing

The VHDL firmware has been successfully debugged and implemented. The errors discovered in the block diagram and the xdc file have also been fixed. After generating the bitstream, the firmware was loaded onto the Zedboard to test it using the Easyboard. A 50 MHz clock signal was provided from the Analog Discovery 2, and the HMCAD1511 was programmed using Serial Peripheral Interface (SPI) commands to output a test ramp signal serially using LVDS; this ramp pattern goes from 0x00 to 0xFF. The deserialized data measured using Xilinx's Internal Logic Analyzer (ILA) is shown in the following figure.

Figure 30. Test Ramp Signal Waveform

After verifying the deserializer IP's and the firmware's functionality, the Jupyter notebook code was modified to configure the HMCAD1511 into single channel mode with an external input. An 800 kHz square wave was generated using a waveform generator and input to the ADC. The ILA waveform is shown in the following figure.

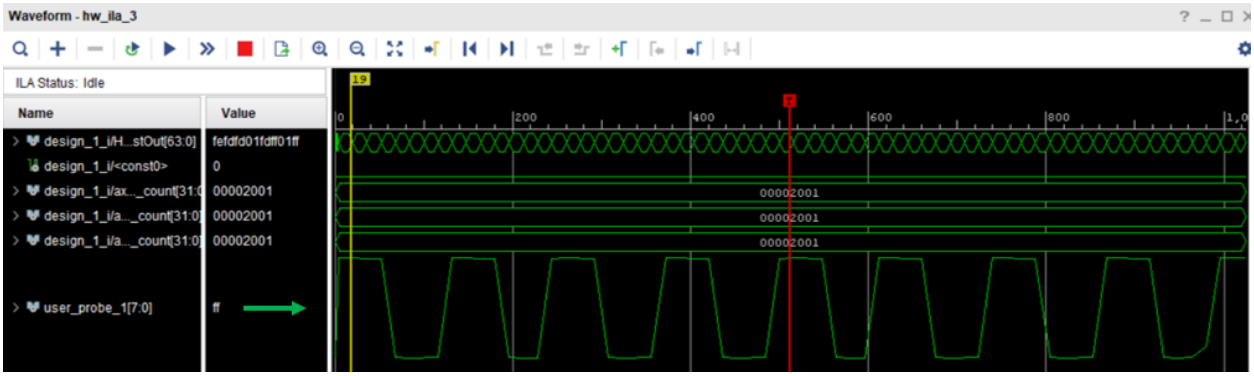


Figure 31. 800 kHz Square Wave Signal Waveform

6.2.2 Jupyter Notebook Prototyping Progress

After verifying the firmware's functionality, the Jupyter notebook was run to extract the ADC data and plot it using the Matplotlib library. The Jupyter notebook code obtained from the previous HACD group was out of order, without comment and very time consuming to review and debug. Furthermore, due to inconsistent SPI communication with the Easyboard, the output was not exactly as expected. In addition, the data from the ADC is converted from 8-bit samples to a 64-bit AXI packet. This 64-bit packet is then read from memory and results in the waveform plotting taking a while. The current output of the waveform (shown below) is not an ideal ramp signal but proves that we can get the data from memory and plot it. This is a key component of this project. It can be noted that we are currently experiencing some clipping and flatlining but the overall shape is still represented.

We plan on taking this proof of concept further during the winter break and ECE 493 by making the conversion from 8 bit packets to plotted values more efficient so there is less time in between visualizations. We will also be working to understand how we can take this proof of concept and integrate it with the GUI and server so that both the GUI code and the ADC send data in the proper formats. Thus far, the Jupyter notebook code was used to extract and plot the test ramp signal. However, during the winter break and ECE 493, we plan on testing more dynamic waveforms as discussed in section 7.

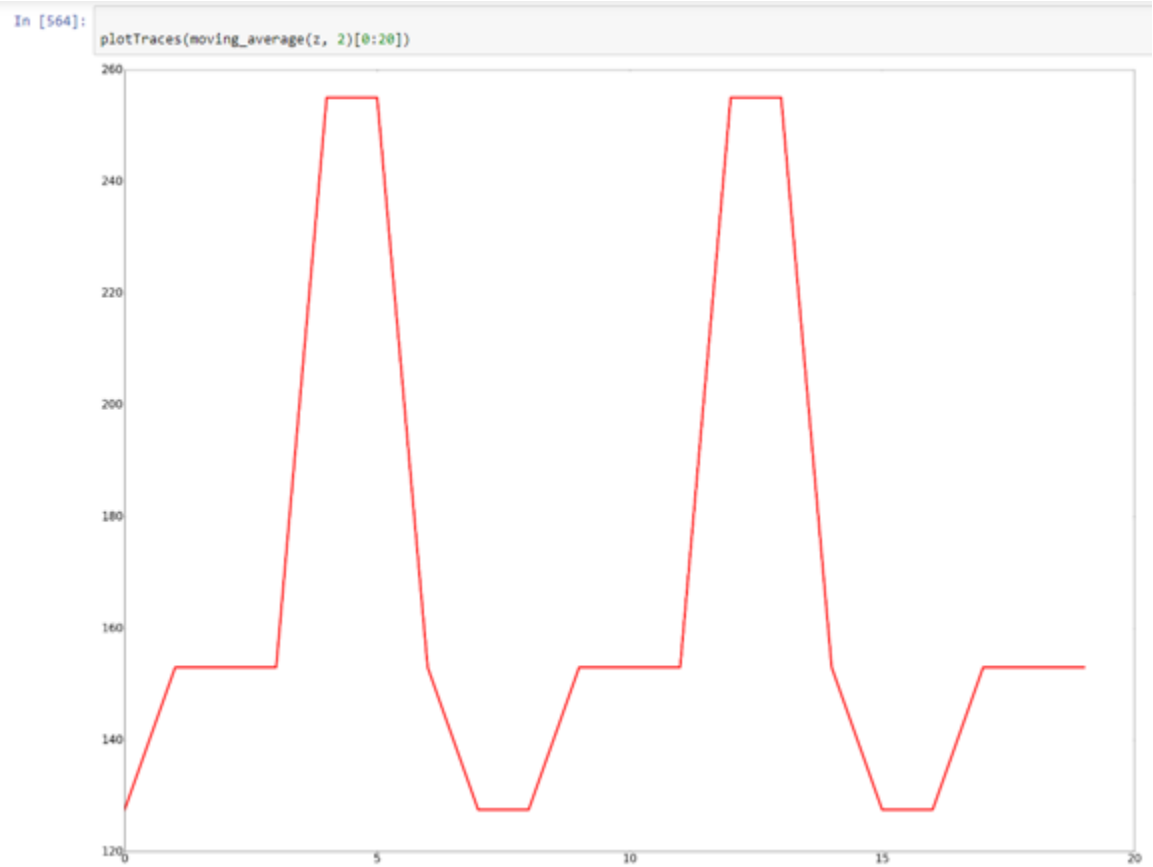


Figure 32. Ramp Pattern Plotted in Jupyter Notebook

6.3 Software Development & Waveforms Live Cloning

As of now, we have successfully cloned a copy of the WaveformsLive repo and were able to run a local version of this on the Ultra96. This means that the entire WaveformsLive GUI is currently set up on the Ultra96 and will run if the “ionic serve” command is run in the waveforms-live directory and then you type the IP into the address bar of any web browser.

For the GUI we have been reading through the current GUI code to better understand how to modify and add to it without removing OpenScope functionality. It is important to us that this product be a continuation of the previous product and not a new one all together. In addition to this we are also learning typescript so we can know the language this project is written in.

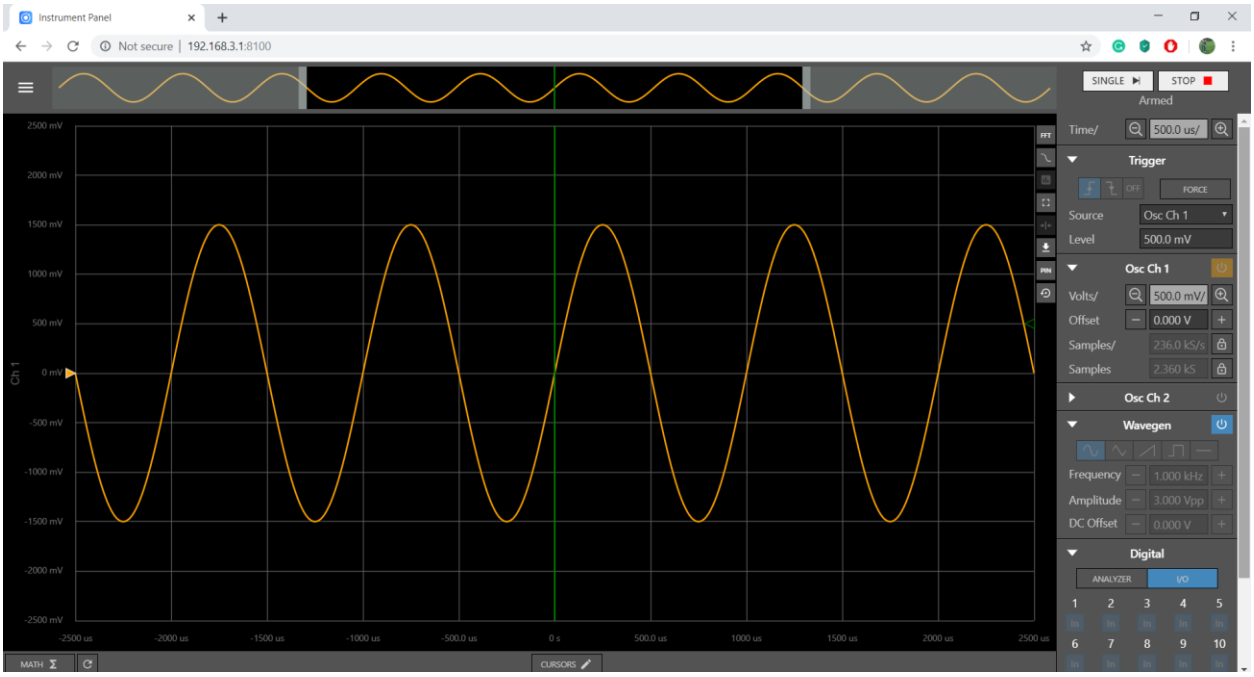


Figure 33. Sample waveform from locally hosted WaveformsLive on Ultra96

7. Testing Plan for ECE493

7.1 Analog Front-End Testing

To verify the proper operation of the analog front-end, several key components of the circuit will be tested individually. After the operation of these individual components is verified, the system will be tested at a black box level to demonstrate the proper input/output functionality. This comprehensive testing plan is presented in section 7.4 of this document. These components are listed below:

7.1.1 Attenuator

To verify the successful operation of the attenuator, a function generator will be used to provide AC inputs ranging from 100Hz-500MHz. The output voltage will be examined and the negative gain will be recorded at each frequency. Furthermore, a DC input voltage will be provided to the attenuator and the drop in voltage will be recorded again. This experiment will confirm the functionality of the attenuator for a wide range of input frequencies.

7.1.2 Low-Noise Amplifier (LNA)

Similar to the attenuator test, input signals with a varying range of frequencies and amplitudes will be provided to the LNA and the relationship between input/output voltage as well as the frequency response relationship will be plotted. This will allow a clear understanding of the voltage levels or frequency cut-offs where the output signal starts to saturate.

7.1.3 Variable Gain Amplifier (VGA)

The gain of the variable gain amplifier will be modified using SPI protocol and input signals of various frequencies will be provided through a function generator. The gain of the VGA will be verified through a commercial oscilloscope for frequencies up to 500MHz.

7.1.4 Phase-locked loop

An external clock shall be provided to the PLL and the clock multiplier will be adjusted through SPI. The output frequency of the PLL clock signal will be measured and verified.

7.2 VHDL Firmware Testing

7.2.1 Pynq Linux Port Testing

To test the successful porting of Pynq Linux onto the Ultra96-v2 board, a simple Pynq overlay will be created with an AXI GPIO peripheral to control the onboard switches and LEDs. This will verify that the system can load a bit file onto the programmable layer of the FPGA, and will validate the successful boot of linux OS. The test is based off of a similar test previously performed on the Xilinx Zedboard.

7.2.2 Firmware Testing

Having verified the firmware's operation at 50 MHz using the Zedboard and the Analog Discovery 2 for the clock input, the next stage would be to make sure the firmware works properly at the required 1 GHz speed. An external fractional-n PLL frequency synthesizer will be used for this purpose. Next the Ultra96-v2 board will be programmed with the OSHO VHDL firmware and then tested using the Easyboard. The firmware's functionality will be verified using

the ILAs. The HMCAD1511 Easyboard will be used to sample a known signal and send data to the Ultra96 board, which will then be checked against the expected result.

7.2.3 Jupyter Notebook Testing

The Jupyter notebook needs to be optimized to obtain data from the Ultra96 board's memory, process it and plot the resulting data using Matplotlib. This will be a backup for the GUI.

7.3 Server Testing & GUI Testing

To ensure that the GUI is successful and we never ruin the GUI version on the Ultra96 we will be testing the GUI locally on our laptops and once we have a final version we will port it to the Ultra96. The goal is to test each new feature modularly. To achieve this we will be adding small parts to our overall GUI on the version on our laptops, verifying that works, then porting the code to the Ultra96. The first feature to test is the removal of the "clone me" tag on the home page when the GUI is launched. This is a quick change as it is just graphical and can be quickly tested on our computers.

The next testing step would be to add the Ultra96 to the list of devices that can be used and configuring the device. To test the success of this we will be launching the local WaveformsLive from our laptop and seeing if the GUI crashes when we select the Ultra96. If this results in a crash we plan to follow the debugging trace to understand where the break occurs and resolve this issue. With the addition of the new device we will be running through the tutorial and signal generation with this device to ensure a successful feature addition.

Once we know that the Ultra96 option is successfully implemented we can push to the repository from our laptops and pull that onto the Ultra96 to verify functionality from there as well. With that feature tested we will need to use the OpenScope to test how data transfers from one of the known working devices. With this we will get a better understanding of the method of data visualization and can test the OpenScope on the Ultra96 mode to understand the difference between the devices that we need to be cognizant of. With those differences in mind we will test the waveform visualization with a sample data file on the Ultra96 and ensure that the sample waveform can be visualized correctly. Modifications to the waveform calculation will be made until we can get successful visualization of the sample waveform on the Ultra96 mode. The different sample waveforms we will be using will be solid ground, solid 5V, a 5MGhz/50MGhz/100MGhz/200MGhz,500MGhz sine/square wave with Vpp of 5V, and 5MGhz/50MGhz/100MGhz/200MGhz,500MGhz sine/square wave with Vpp of 10V. This will give us an understanding of the performance on simple waveforms at various speeds and voltage ranges. Once that is done we can move onto testing data from a live circuit. We will do this by making a simple circuit with a resistor and LED to verify the DC functionality. With the DC functionality tested we will use an RC circuit to test the AC functionality. We will go through a 286 Lab with this device to help verify with a realistic scenario.

If at any point in time the testing seems to be hitting a wall or progress cannot be made we plan on reaching out to our contacts that work as full time GUI developers to get their insight into the problem.

7.4 High-Level Overall System Testing

7.4.1 Input variation

The overall device will be tested using both 1 and 2 analog inputs. The waveforms of these inputs will be varied between DC signal, sine waves, square waves, triangular waves, and more. The ability of the device to accurately display these waveforms on the GUI will be verified. The input voltage levels will be changed from 0 V_{PP} to 20 V_{PP} to confirm that the input voltage requirement is met. The results from this test will be compared at a high-speed commercially available oscilloscope.

7.4.2 Frequency Sweep

A function generator will be used to provide a periodic input signal to the device. A frequency sweep from 0Hz to 500MHz will be conducted and the absence of aliasing shall be verified for the bandwidth of our device. This will repeated in dual channel mode where the frequency sweep will be conducted from 0Hz to 500MHz.

7.4.3 External Trigger System

The external trigger system will be used to test if repetitive waveforms can be displayed in a steady manner for analyzation purposes. This will consist of applying an input signal to the analog input of the oscilloscope and verifying that the oscilloscope pauses data capture when an external trigger event occurs. This will be verified using a high-speed commercial oscilloscope by recording both the trigger event and the input signal.

7.4.4 External Clock Input

The external clock signal generated with a frequency synthesizer will be used to test the function of the device at different clock frequencies. A high-speed commercial oscilloscope will be used to measure the external clock signal, the input signal, and the ADC sampling clock will be measured in order to verify that the ADC sampling clock will be synchronized with the external clock input.

8. Task Allocations for Remainder of Project

8.1 Analog Front-End

Due to the fact that the preliminary design and component selection of the OSHO analog front-end circuit is already complete, the task allocation for the remainder of the project are listed below. The persons responsible for the task allocation listed above are Zaeem Gauher and Umair Aslam. They will serve as the lead and the backup respectively.

- Complete power circuitry testing of the HACD front-end board
- Test the overall functionality of the HACD board after the power stage is finalized
- Solder power circuitry components on the OSHO board once manufactured
- Test and verify the repopulated power circuitry on OSHO PCB

- Solder the components that comprise the signal measurement chain and execute the testing procedure as specified in section 7.
- Test the interface between the front-end PCB and the Ultra-96 board to verify correct transfer of signals.
- Change schematics as well as select new components if necessary to produce a final version of the OSHO board.

8.2 PCB Design

After the analog front-end schematics are finalized this semester, the initial design of the custom PCB for the analog front will be conducted as early as possible to allow for a possible second revision and adequate time for testing. This will likely be conducted over winter break so that time is not wasted while waiting for the board is commercially printed. The lead for this aspect of the project will be Timothy Bullock and the backup for the project will be Zaeem Gauher. The following tasks are the tasks that are expected to make up this part of the project:

- Creation and acquisition of component footprints and three-dimensional models
- Initial planning and layout of PCB layers and overall high level layout
- Completion of preliminary component layout and trace routing
- Revision of the initial design after rules checking and advice of project advisor and PCB layout experts.
- Final rules checking, inspection, and design validation
- Commercial printing and in-house population of components
- Potential revision of the design if design issues are found in first revision

8.3 FPGA & Firmware Development

Since the Easyboard's onboard oscillator and PLL were non-functional, the firmware was tested using Analog Discovery 2's 50 MHz clock. The next step would be to verify the firmware works at its required 1 GHz speed. For this, an external fractional-n PLL frequency synthesizer will be used. The firmware will most probably need further optimization to be able to meet the timing constraints at this speed. The lead for this part of the project is Umair Aslam and the backup for this aspect of the project is Timothy Bullock. The remaining tasks for firmware development are:

- Modifying Jupyter Notebook code to send the appropriate SPI commands to configure the ADC to operate in single mode at its required full speed using a 1 GHz frequency synthesizer
- Testing the Zedboard at 1 GHz
- Porting Pynq Linux onto the Ultra96 board
- Programming the Ultra96-v2 with the OSHO firmware
- Adding an external trigger processing IP core to the datapath design
- Final design validation, verification and testing

8.4 Server Back-End & GUI Web Client Development

The following list is a review are the remaining tasks we need to complete in order to completely develop the web server and GUI design using the Ultra96. The lead for the GUI portion of the project is Afnan Ali, and the backup is Evan Hoffman.

Conversely, as these aspects of the project are very interdependent, the lead for the server back-end portion of the project is Evan Hoffman and the backup is Afnan Ali.

- Solve timeout issue on the Ultra96 when logged in for an SSH session
- Edit configuration file for the Ultra96 so that upon startup, the web application will immediately be launched so that the user will not have to SSH into the device in order to get it started
- Determine the protocol that WaveformsLive uses to receive data
- Send test data to our cloned version of the WaveformsLive application in order to verify that we are using the correct protocol and that this will be the correct protocol for our firmware to use.
- Make basic interface modifications to the user interface of the WaveformsLive application running on the Ultra96
- Implementation of the SPI commands on the Ultra96 and verify that these commands are able to interact with our analog frontend. Use MSP430 Launchpad to test the SPI signals
- Modification of Waveforms Live and server to incorporate these control and configuration aspects into the GUI
- Modification of Waveforms Live and server to visualize data from OSHO board

9. Schedule for Remainder of Project

With the task allocations outlined in section 8, we can now create a schedule for the remainder of the project that incorporates major tasks in each aspect of the project, significant project milestones, and project deliverables. A Gantt chart for the remainder of the project including the last 3 weeks of ECE492, winter break, and ECE 493 is shown in the subsequent figures.

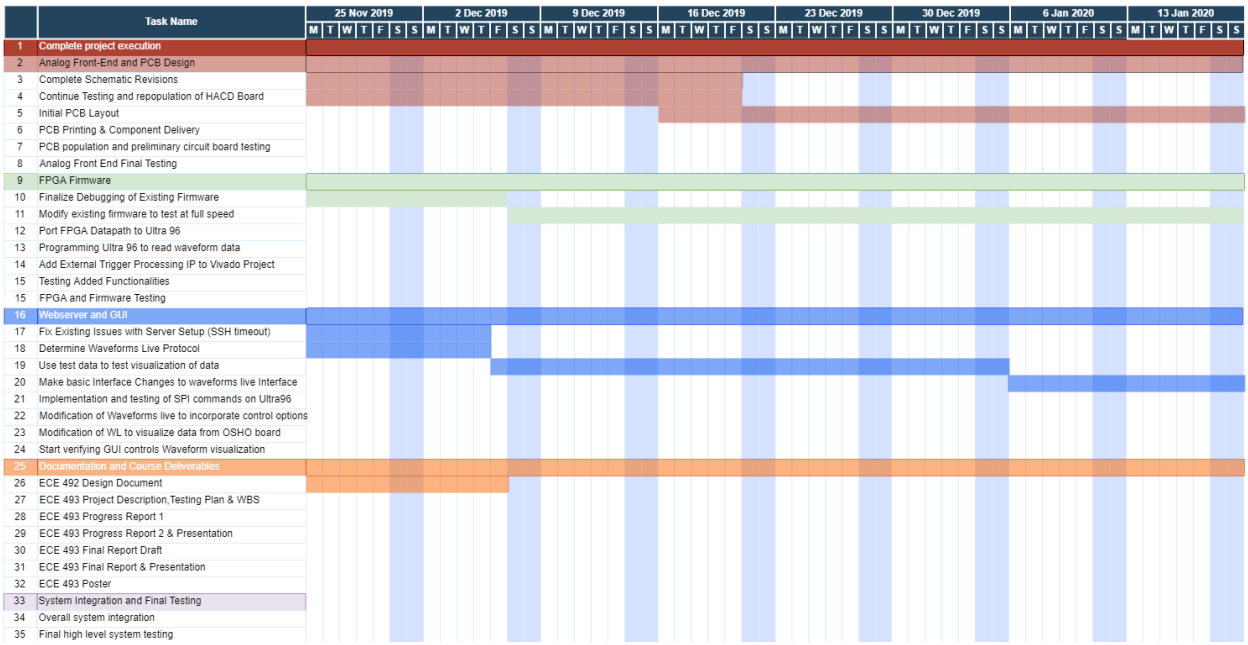


Figure 34. Gantt for the Remainder of the Project: Now - Jan 19.

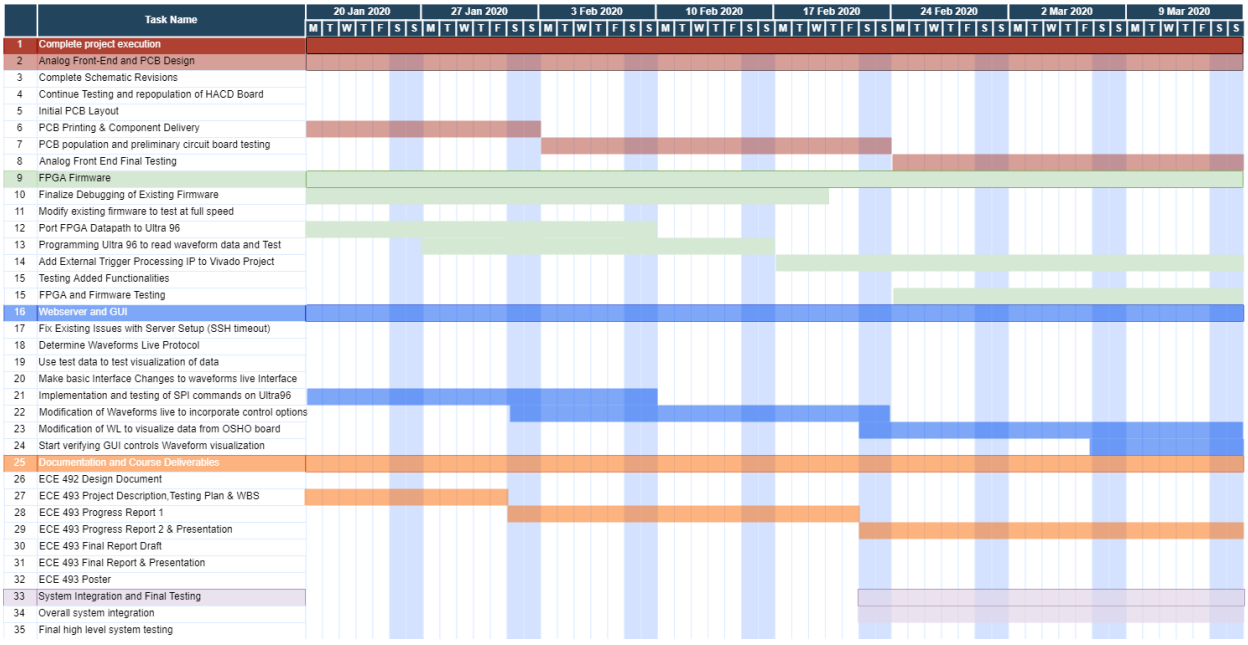


Figure 35. Gantt for the Remainder of the Project: Jan 20 - Mar 16.

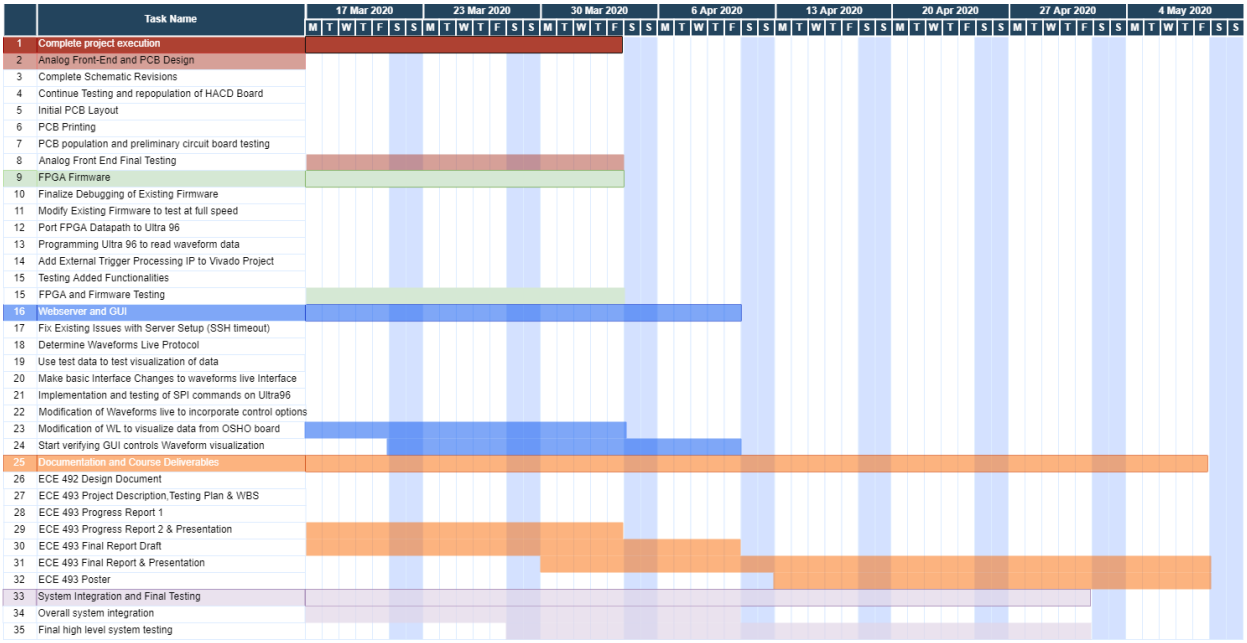


Figure 36. Gantt for the Remainder of the Project: Mar 17 - Completion.

10. References

[1] A. Wozneak, R. Nagpal, and R. Meruvia, “ECE - 492 Design Document.” 10-Dec-2018.

[2] 96Boards. (2019). Ultra96. [online] Available at: https://www.96boards.org/product/ultra96/ [Accessed 4 Oct. 2019].

[3] “HMCAD1511 Datasheet and Product Info | Analog Devices.” [Online]. Available: https://www.analog.com/en/products/hmcad1511.html. [Accessed: 12-Oct-2019].

[4] C. Sisterna, “Zynq Architecture 7-Series FPGA Architecture,” International Centre for Theoretical Physics. [Online]. Available: http://indico.ictp.it/event/8342/session/10/contribution/68/material/slides/0.pdf. [Accessed: 03-Dec-2019].

[5] “Ultra96-V2 Development Board | Zedboard.” [Online]. Available: http://zedboard.org/product/ultra96-v2-development-board. [Accessed: 05-Dec-2019].

[6] “Intro to AXI Protocol: Understanding the AXI interface.” [Online]. Available: https://community.arm.com/developer/ip-products/system/b/soc-design-blog/posts/introduction-to-axi-protocol-understanding-the-axi-interface. [Accessed: 06-Dec-2019].

[7] “AXI4 Overview.” [Online]. Available: http://www.mrc.uidaho.edu/mrc/people/jff/EO_440/Handouts/AMBA Protocols/Xilinx Docs/XTECH_B_AXI4_Technical_Seminar.pdf. [Accessed: 05-Dec-2019].

[8] “7 Series FPGAs SelectIO Resources,” Xilinx, 08-May-2018. [Online]. Available: https://www.xilinx.com/support/documentation/user_guides/ug471_7Series_SelectIO.pdf. [Accessed: 05-Dec-2019].
[9] M. Defossez, “Serial LVDS High-Speed ADC Interface,” Xilinx, 20-Nov-2012. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp524-serial-lvds-adc-interface.pdf. [Accessed: 05-Dec-2019].
[10] M. Defossez, N. Sawyer, “LVDS Source Synchronous DDR Deserialization (up to 1,600 Mb/s),” Xilinx, 22-Jul-2016. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp1017-lvds-ddr-deserial.pdf. [Accessed: 05-Dec-2019].
[11] N. Sawyer, “LVDS Source Synchronous 7:1 Serialization and Deserialization Using Clock Multiplication,” Xilinx, 18-Jul-2018. [Online]. Available: https://www.xilinx.com/support/documentation/application_notes/xapp585-lvds-source-synch-serdes-clock-multiplication.pdf. [Accessed: 05-Dec-2019].
[12] Digilent, “Digilent/waveforms-live,” *GitHub*, 08-Oct-2019. [Online]. Available: https://github.com/Digilent/waveforms-live. [Accessed: 07-Dec-2019].
[13] P. Patel, “What exactly is Node.js?,” freeCodeCamp.org, 25-Jun-2019. [Online]. Available: https://www.freecodecamp.org/news/what-exactly-is-node-js-ae36e97449f5/. [Accessed: 07-Dec-2019].
[14] “React vs Angular: An In-depth Comparison,” *SitePoint*, 30-Jan-2019. [Online]. Available: https://www.sitepoint.com/react-vs-angular/. [Accessed: 07-Dec-2019].
[15] O. Temitope, “Dependency Injection Explained in Plain English,” Codementor. [Online]. Available: https://www.codementor.io/olotintemitope/dependency-injection-explained-in-plain-english-b24hippx7. [Accessed: 07-Dec-2019].
[16] Sk, “How To Install NodeJS On Linux,” OSTechNix, 25-Nov-2019. [Online]. Available: https://www.ostechnix.com/install-node-js-linux/. [Accessed: 07-Dec-2019].
[17] Texas Instruments, “LMH5401 8-GHz, Low-Noise, Low-Power, Fully-Differential Amplifier” LMH5401 datasheet.
[18] Texas Instruments, “LMH6401 DC to 4.5 GHz, Fully-Differential, Digital Variable-Gain Amplifier” LMH6401 datasheet.
[19] Texas Instruments, “IM01CGR High Speed RF Relay” IM01CGR datasheet.
[20] TE Connectivity, “High Speed Multi-Mode 8-Bit 30 MSPS to 1 GSPS A/D Converter” HMCAD1511 datasheet.
[21] Texas Instruments, “CDCE62002 Four Output Clock Generator/Jitter Cleaner With Integrated Dual VCOs” CDCE62002 datasheet.
[22] Texas Instruments, “LP3878-ADJ Micropower 800-mA Low-Noise "Ceramic Stable" Adjustable Voltage Regulator for 1-V to 5-V Applications ” LP3878-ADJ datasheet.

[22] Texas Instruments, “LP38513-ADJ 3A Fast-Transient Response Adjustable Low-Dropout Linear Voltage Regulator” LP38513-ADJ 3A datasheet.
[22] Texas Instruments, “TPS54327 3-A Output Single Synchronous Step-Down Switcher With Integrated FET ” TPS54327 datasheet.
[23] Texas Instruments, “TL7660 CMOS VOLTAGE CONVERTER” TL7660 datasheet.
[24] Texas Instruments, “LMR70503 SIMPLE SWITCHER Buck-Boost Converter For Negative Output Voltage in µSMD ” LMR70503 datasheet.
[25] Texas Instruments, “TPS79301 Low-Noise, High PSRR, RF, 200-mA Low-Dropout Linear Regulators in NanoStar™ Wafer Chip Scale and SOT-23 ” TPS79301 datasheet.
[26] Texas Instruments, “TPS72301 200-mA, Low-Noise, High-PSRR, Negative Output Low-Dropout Linear Regulators” TPS72301 datasheet.
[27] Dallas Semiconductor, “DS1267 Dual Digital Potentiometer Chip” DS1267 datasheet.

12. Appendix C: OSHO PCB Bill of Materials

Reference	Value	Manufacturer Part #	Quantity Per Board	Quantity Per 3 Boards	Price Per 1	Price Per 10	Price Per 100	Total Per Component (At Ordered Price Point)
C1	47uF	TAJB476M010TNJ	1	3	$0.28	$0.280	$0.183	$0.28
> C2-C4, C12, C13, C18, C21, C22, C27, C30, C31, C33, C35, C37-C39, C46, C50, C51, C56, C59-C61, C68, C73, C75, C81, C87, C94, C96, C182, C184, C185, C188, C190, C199, C203	10uF	GRM188R61A106KE69J	37	111	$0.18	$0.122	$0.062	$2.29
> C5, C6, C14, C17, C19, C20, C23, C26, C28, C29, C32, C34, C36, C45, C47, C52, C55, C57, C58, C67, C69,	0.1uF	CC0402KRX7R7BB104	81	243	$0.10	$0.024	$0.009	$0.73

C74, C76-C78, C80, C82-C84, C86, C88, C90, C91, C95, C97, C151, C164, C181, C183, C186, C187, C189, C191-C198, C200-C202, C204-C231								
> C7, C141, C144, C145, C148, C153, C156, C158, C161	10pF	CC0402KRNPO9BN100	9	27	$0.10	$0.019	$0.009	$0.08
> C8, C9	47uF	EMK107ABJ475KA-T	2	6	$0.26	$0.121	$0.082	$0.24
> C10, C48, C70, C72, C93, C150, C163, C167, C168, C170, C173, C175, C177-C179	1uF	EMK107B7105KA-T	15	45	$0.12	$0.043	$0.029	$0.44
> C11, C15, C24, C53, C99-C101, C104, C106, C108, C109, C112, C114, C116-C118, C121, C123, C125, C126, C129, C131, C140, C149, C152, C157, C162, C165, C166, C169, C171, C172, C174, C176	0.01uF	CC0402KRX7R9BB103	34	102	$0.10	$0.018	$0.009	$0.31
> C16, C25, C40, C41, C43, C44, C54, C62, C63, C65, C66	22uF	GRM188R61A226ME15D	11	33	$0.34	$0.236	$0.133	$1.46
> C42, C64, C180	2.2uF	LMK107BJ225KAHT	3	9	$0.15	$0.062	$0.042	$0.19
> C49, C71	0.22uF	TMK107B7224KA-T	2	6	$0.11	$0.037	$0.025	$0.07
> C79, C85, C89, C92	1uF	TAJA105K016RNJ	4	12	$0.31	$0.216	$0.117	$0.86
> C98, C102, C103, C105, C107, C110, C111, C113, C115, C119, C120, C122, C124, C127, C128, C130	2200pF	GRM155R71H222KA01D	16	48	$0.10	$0.029	$0.013	$0.21
> C132, C135, C136, C139	5.1pF	CC0402CRNPO9BN5R1	4	12	$0.10	$0.020	$0.009	$0.04
> C133, C134, C137, C138	11pF	0402N110J500CT	4	12	$0.10	$0.038	$0.020	$0.08
> C142, C143, C146, C147, C154, C155, C159, C160	22pF	0402N220G500CT	8	24	$0.10	$0.038	$0.017	$0.14

> D1, D5, D6, D9-D12, D16, D17, D20-D23, D30, D31	LTST-C191TBKT	LTST-C191TBKT	15	45	$0.48	$0.260	$0.122	$1.83
> D2-D4, D7, D13-D15, D18, D28, D29	1N4148W-7-F	1N4148W-7-F	10	30	$0.16	$0.150	$0.053	$0.53
> D8, D19, D24-D27, D32-D34	MMBD452LT1G	MMBD452LT1G	9	27	$0.38	$0.246	$0.106	$0.95
> FB1, FB3, FB5, FB11	BLM18SG260TN1D	BLM18SG260TN1D	4	12	$0.13	$0.075	$0.051	$0.30
> FB2, FB4, FB6-FB10, FB12, FB13	BLM18SG121TN1D	BLM18SG121TN1D	9	27	$0.13	$0.075	$0.051	$0.68
> FB14-FB20	BLM18KG102SN1D	BLM18KG102SN1D	7	21	$0.10	$0.067	$0.036	$0.25
> GDT1-GDT3	SH90	SH90	3	9	$1.47	$1.250	$0.967	$3.75
J1	PJ-202AH	PJ102AH	1	3	$0.75	$0.566	$0.495	$0.75
> J2, J3, J6	1-1337543-0	1-1337543-0	3	9	$1.45	$1.450	$1.190	$4.35
> J4, J5	CONSMA001-G	CONSMA001-G	2	6	3.14	2.9	2.73	$6.28
J7	57202-G52-20LF	57202-G52-20LF	1	3	$4.95	$4.550	$3.960	$4.95
J8	2-5177986-2	2-5177986-2	1	3	$4.98	$4.180	$3.980	$4.98
JP1	826926-3	826926-3	1	3	$0.27	$0.235	$0.182	$0.27
L1	ACM7060-301-2PL-TL01	ACM7060-301-2PL-TL01	1	3	$2.03	$1.530	$1.400	$2.03
L2	3uH	SRN6028-3R0Y	1	3	$0.39	$0.271	$0.246	$0.39
> L3, L4	2.2uH	LQM21PN2R2MCHD	2	6	$0.31	$0.276	$0.189	$0.55
> L5, L7, L8, L10	22nH	LQW18AN22NG80D	4	12	$0.24	$0.210	$0.140	$0.84
> L6, L9	26nH	LQW15AN26NG80D	2	6	$0.26	$0.223	$0.149	$0.45
> L11, L13, L14, L16, L17, L19, L20, L22	44nH	LQW18AN44NG80D	8	24	$0.24	$0.210	$0.140	$1.68
> L12, L15, L18, L21	52nH	LQW18AN52NG80D	4	12	$0.24	$0.210	$0.140	$0.84
Q1	CSD18532Q5B	CSD18532Q5B	1	3	$2.37	$2.010	$1.610	$2.37
> Q2-Q10	2N7002K	2N7002KT7G	9	27	$0.17	$0.160	$0.057	$0.51
> R1, R28, R29, R38-R41, R56, R57, R66-R69, R75, R79, R91, R95, R106, R107	90.9	RC0402FR-0790R9L	19	57	$0.10	$0.012	$0.004	$0.08
> R2, R7, R10, R13, R18, R19	10k	RC0603FR-0710KL	6	18	$0.10	$0.018	$0.006	$0.04
R3	95.3k	RC0603FR-0795K3L	1	3	$0.10	$0.018	$0.006	$0.01
R4	22.1k	RC0603FR-1022K1L	1	3	$0.10	$0.018	$0.006	$0.01
> R5, R16	3.57k	CR0603-FX-3571ELF	2	6	$0.10	$0.010	$0.006	$0.01
> R6, R17	1.15k	RC0603FR-071K15L	2	6	$0.10	$0.018	$0.006	$0.01

R8	15.8k	RC0603FR-0715K8L	1	3	$0.10	$0.018	$0.006	$0.01
R9	11.5k	RE0603FRE0711K5L	1	3	$0.10	$0.018	$0.006	$0.01
R1 1	274k	RC0603FR-07274KL	1	3	$0.10	$0.018	$0.006	$0.01
R1 2	30.1k	RC0603FR-0730K1L	1	3	$0.10	$0.018	$0.006	$0.01
R1 4	110k	RC0603FR-07110KL	1	3	$0.10	$0.018	$0.006	$0.01
R1 5	180k	RC0603FR-07180KL	1	3	$0.10	$0.018	$0.006	$0.01
R2 0	130k	RC0603FR-07130KL	1	3	$0.10	$0.018	$0.006	$0.01
R2 1	24k	CR0603-FX-2402ELF	1	3	$0.10	$0.011	$0.006	$0.01
R22, R50	49.9	HRG3216P-49R9-D-T1	2	6	0.96	0.745	0.584	$1.49
> R24, R52	499	RC0603FR-07499RL	2	6	$0.10	$0.018	$0.006	$0.01
> R25, R46, R48, R53, R81, R82, R97, R98, R108, R113	0	RC0603JR-070RL	10	30	$0.10	$0.015	$0.005	$0.05
> R23, R27, R51, R55	5.36	RC0603FR-075R36L	4	12	$0.15	$0.035	$0.011	$0.04
> R26, R30, R47, R49, R54, R58, R116, R117, R132	49.9	RC0603FR-0749R9L	9	27	$0.10	$0.018	$0.006	$0.05
> R32, R60	1M	35401M0JT	2	6	$0.98	$0.837	$0.590	$1.67
> R33, R36, R61, R64	1M	RC0603FR-071ML	4	12	$0.10	$0.018	$0.006	$0.02
> R34, R62	9.1M	RC0603FR-079M1L	2	6	$0.10	$0.018	$0.006	$0.01
> R35, R63	887k	RC0603FR-07887KL	2	6	$0.10	$0.018	$0.006	$0.01
> R31, R37, R59, R65	105k	AC0603FR-07105KL	4	12	$0.10	$0.022	$0.008	$0.03
> R42, R121	200k	RC0603FR-10200KL	2	6	$0.10	$0.018	$0.006	$0.01
> R43-R45, R122-R131	348	RC0603FR-07348RL	13	39	$0.10	$0.018	$0.006	$0.08
> R70, R71, R86, R87	2k	RC0603FR-072KL	4	12	$0.10	$0.018	$0.006	$0.02
> R74, R77, R90, R93, R118	0	RC0402JR-130RL	5	15	$0.10	$0.010	$0.004	$0.02
> R73, R76, R89, R92	15	RC0402FR-1315RL	4	12	$0.10	$0.012	$0.004	$0.02
> R78, R80, R94, R96	40.2	RC0402FR-0740R2L	4	12	$0.10	$0.012	$0.004	$0.02
> R72, R83, R88, R99	174	RK73H1ETTP1740F	4	12	$0.10	$0.025	$0.010	$0.04
> R84, R85, R100, R101	210	RC0603FR-07210RL	4	12	$0.10	$0.018	$0.006	$0.02

> R103, R104, R110, R111	1.21k	CR0603-FX-1211ELF	4	12	$0.10	$0.017	$0.006	$0.02
> R102, R105, R109, R112	4.99	RK73H1JTTD4R99F	4	12	$0.10	$0.021	$0.008	$0.03
> R114, R115	49.9	RC0402FR-0749R9L	2	6	$0.10	$0.012	$0.004	$0.01
> R119, R120	1.0k	RC0603FR-071KL	2	6	$0.10	$0.018	$0.006	$0.01
> RLA1-RLA10	V23105A5001A201	V23105A5001A201	10	30	$2.62	$2.450	$1.960	$24.50
S1	L101011MS02Q	L101011MS02Q	1	3	$2.03	$1.970	$1.630	$2.03
> S2, S3	PTS810-SJK-250-SMTR-LFS	PTS810SJK250SMTRLFS	2	6	$0.17	$0.165	$0.165	$0.33
> TP17, TP18	TestPoint	5-146850-1	2	6	$0.10	$0.046	$0.046	$0.09
U1	TPS2400DBVR	TPS2400DBVR	1	3	$2.07	$1.760	$1.410	$2.07
U2	TPS54327DDAR	TPS54327DDAR	1	3	$1.48	$1.250	$0.904	$1.48
> U3, U4	TPS7A9201DSKR	TPS7A9201DSKR	2	6	$2.06	$1.750	$1.400	$4.12
U5	TPS7A7001DDAR	TPS7A7001DDAR	1	3	$1.81	$1.540	$1.110	$1.81
> U6, U8	TPS63710DRRR	TPS63710DRRR	2	6	$2.49	$2.110	$1.690	$4.98
U7	TPS7A9101DSKR	TPS7A9101DSKR	1	3	$1.76	$1.500	$1.200	$1.76
> U9, U15	CD74AC251M96	CD74AC251M96	2	6	$0.89	$0.740	$0.478	$1.78
> U10, U16	OPA659IDBVT	OPA659IDBVT	2	6	$7.11	$6.430	$5.220	$14.22
U11	NVT2003DP,118	NVT2003DP,118	1	3	$0.87	$0.721	$0.465	$0.87
> U12, U14	LMH6559MF-NOPB	LMH6559MF/NOPB	2	6	$2.73	$2.380	$1.960	$5.46
U13	DS1267BS-010+	DS1267BS-010+T/R	1	3	$5.39	$4.870	$3.870	$5.39
> U17, U19	LMH6401IRMZT	LMH6401IRMZT	2	6	$19.52	$18.010	$17.200	$39.04
> U18, U20	LMH5401IRMST	LMH5401IRMST	2	6	$14.98	$13.780	$11.390	$29.96
> U21, U22	OPA376AIDCKR	OPA376AIDCKR	2	6	$1.69	$1.430	$1.030	$3.38
U23	HMCAD1511TR	HMCAD1511TR	1	3	$64.76	$63.740	$61.520	$64.76
U24	CDCE62005RGZR	CDCE62005RGZR	1	3	$9.98	$9.030	$7.490	$9.98
U25	NVT2010PW,118	NVT2010PW,118	1	3	$0.97	$0.820	$0.630	$0.97
U26	XRA1405IL24-F	XRA1405IL24-F	1	3	$1.93	$1.560	$1.250	$1.93
XTAL1	FY2500068	FY2500068	1	3	$0.73	$0.614	$0.513	$0.73
							Total Per Board:	**$272.49**