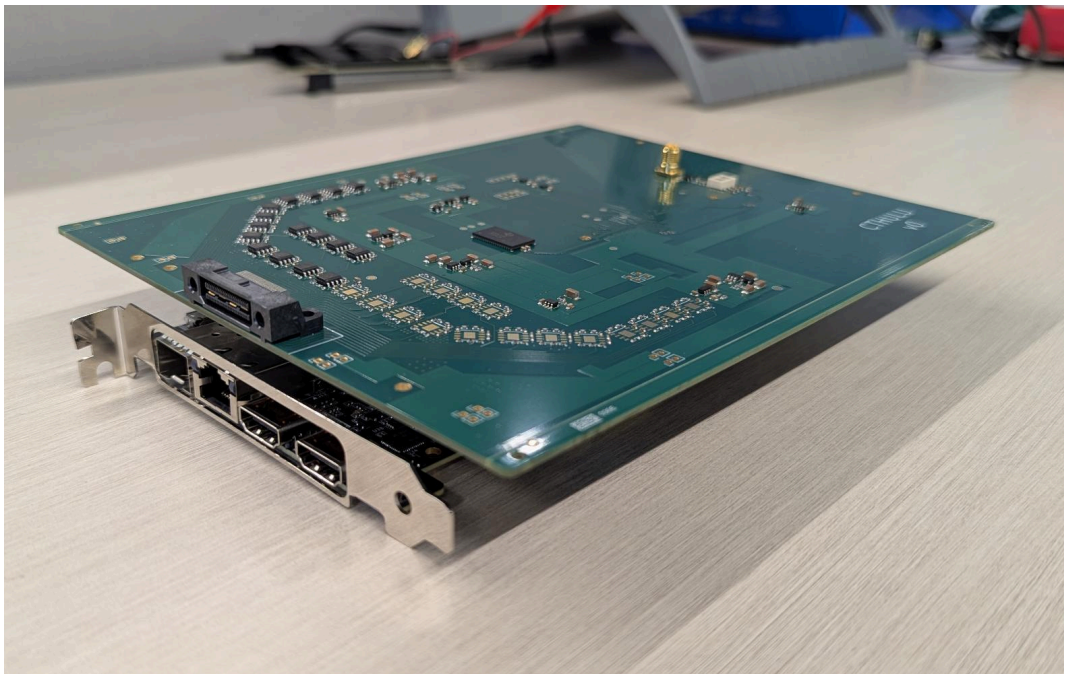
Final Report - CTHULHU

Team Members: Hazel Bershatsky, James Brennan, Alexi Carlone, Matt Geisel, Iris Kaucher, Julia Keadey

Advisor: Canek Fuentes Hernandez

Faculty Support: Gunar Schirner, Bryan Spring, Kai Zhang



CTHULHU's final product.

1. Abstract

Our project was developed for and with two Northeastern labs: Spring Lab, under the College of Science and the Embedded Systems Lab (ESL), under the College of Engineering. These labs aim to develop a biomedical imaging system to identify cancer in the human brain with cell-level resolution. The CTHULHU team has designed and built a system to drive a 32MHz bio-imaging laser and interpret the received data from its pulses. Existing solutions for high-bandwidth biomedical imaging either involve inferior data processing equipment or significantly higher costs; CTHULHU accomplishes more than was previously possible at a much lower price point (and with significantly less complexity, requiring only an off-the-shelf FPGA development board and one PCB).

At a high level, the architecture of our solution has three main parts: electronics, FPGA design, and software. Control flow starts from the software, where the user selects an imaging pattern and other parameters. This information flows to the FPGA, which calculates the required inputs to move the laser to specific positions in the field of view of the sample. At each position, the laser pulses (in 32 different subfrequencies) bounce off the desired cell and hit a silicon photomultiplier array (SiPM), which converts them into electrical signals. The electrical signals are handled within our own custom 6-layer PCB, designed to handle the processing and digitization of these raw signals. Once acquired, those signals are amplified by an array of fully differential amplifiers (FDAs) before being digitized by an ADC and fed back to the FPGA. The FPGA receives 32 channels' worth of data from each sample, performs further processing, and dispatches the data back to the host PC over a PCI-Express bus. There, the firmware receives the data and converts it into a format that can be rendered by a UI, which was initially designed by the ESL and modified to be compatible with this project.

Initially, this system was envisioned to be used specifically in the field of brain cancer imaging and treatment: existing systems suffer from issues of low data resolution and possible side effects of treatment, while a laser-based system can accomplish both imaging and treatment tasks with much greater precision and safety. However, over the course of the project's development, it became apparent that CTHULHU can be applied to a wide range of biomedical applications, allowing for an increased channel count for a cheaper price than what is otherwise available. In addition, the accessible design of the hardware means that this project's work can easily be adapted to suit a wide range of potential experiments, especially when compared to the existing costly, closed-source solutions.

2. Introduction

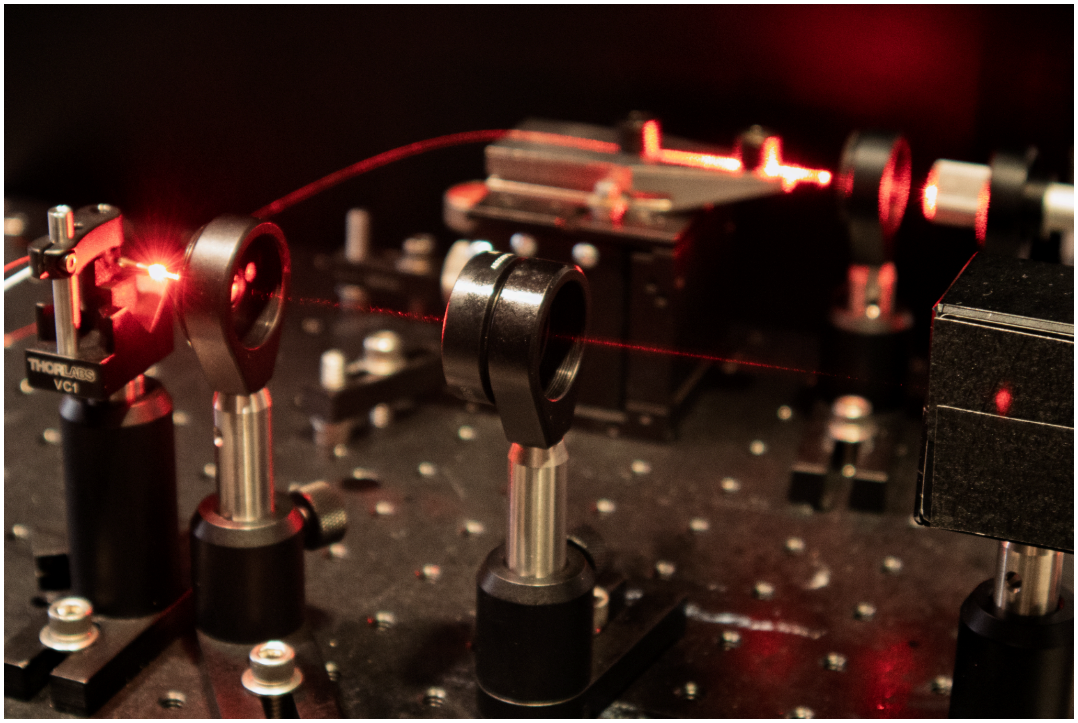


Figure 1: Spring Lab's imaging laser (Credit: Kai Zhang)

Spring Lab's and the Embedded Systems Lab (ESL)'s work is the motivation for our project. The labs have been working together to develop a system for cell microscopy in vitro at a very high resolution. This solution is important to improve the speed of brain cancer treatment, allowing for imaging in the patient instead of needing invasive surgeries and external inspection.

To enable this, Spring Lab has developed a system for multi-photon microscopy that allows for better-focused imaging in deeper tissue samples. They have also built an imaging laser (above), which operates at 32MHz and has programmable shapes per frame. On the biomedical side, fluorescent dyes which bond only to specific molecules allow for many imaging applications. Combining these fluorescent dyes with our imaging system can allow doctors to identify molecules in individual cells, including cancerous cells.

ESL have created an optical input system that takes in the output of the imaging laser (reflected off of the sample) and splits it into a 32-channel silicon photomultiplier (SiPM) array, outputting 32 channels of single-ended voltage signals to be analyzed. CTHULHU comes in to bring these systems to a usable point, analyzing data and offering an interface for doctors and researchers to use.

The CTHULHU team has designed and built a system to drive a 32MHz bio-imaging laser and interpret the received data from its pulses. Our solution consists of two parts: a custom designed PCB to deal with analog signals, and a pre-made FPGA (Artix UltraScale+) for data processing.

The FPGA takes in an imaging pattern and controls the laser position in that pattern. At each frame in this pattern, the PCB intakes 32 channels of data, each corresponding to a different frequency of light. Our PCB processes the data, sending them back to the FPGA. The FPGA packages these data together with the sample's relative location and forwards them to the host PC.

3. Problem Formulation

For existing laser technology to be useful to doctors in cell-specific targeting, a system must allow them to treat patients without being involved in low-level technical details. As we worked, we realized that our solution can be used for a much wider variety of applications, as it can function as a general-purpose 32-channel oscilloscope.

Analog-to-digital conversion at scale is a complex, expensive process, especially with large numbers of channels. Existing solutions are closed-source, hard to interface with, and costly to acquire and maintain. Compared to existing solutions, our work is far more affordable and easy to interface with. Making our work accessible means that it can be adapted for use in experiments throughout medicine.

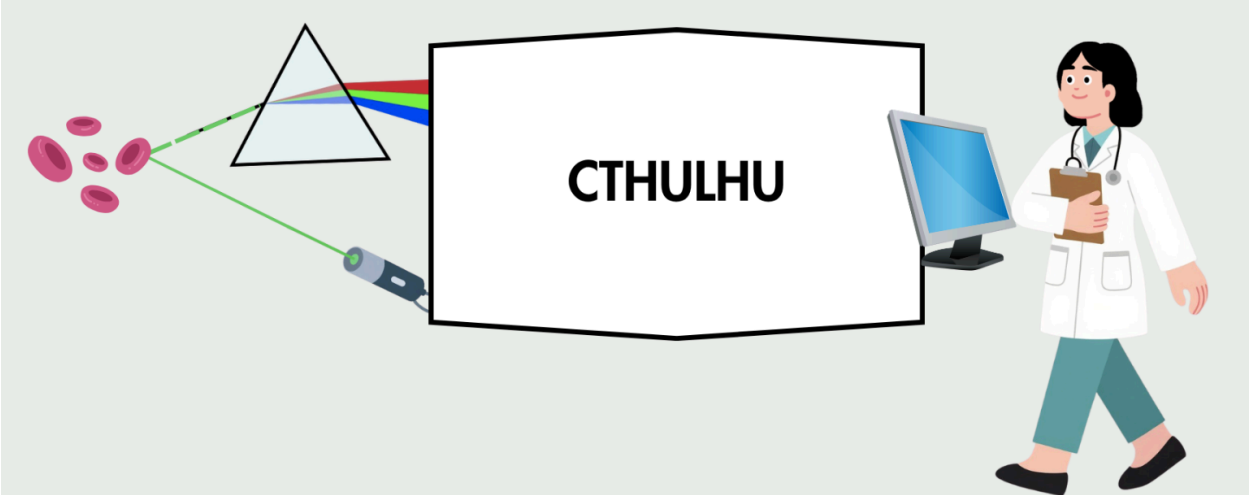


Figure 2: Solution graphic.

4. Design Approach

We decided to split the project into two main components, a custom PCB for analog processing and digitization, and an FPGA for control, signal processing, and a PC interface. In addition to this we also developed firmware to collect the data and expose an interface to the users of our project.

Ultimately there are two main functions of this system. The first is to process data from the analog signal into a format readable from the PC, and the second is to output control signals for the positioning of the laser. In addition, we need to combine this position data with the data received so that we are able to reinterpret the collected data correctly.

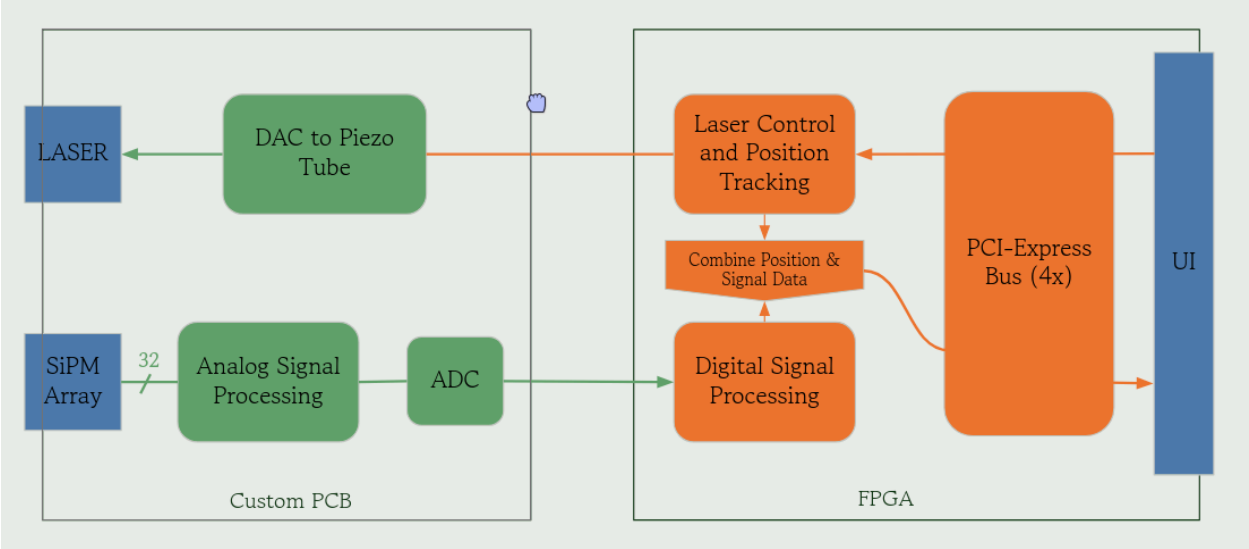


Figure 3: Block design of entire project.

4.1 Electrical Design

We designed a PCB that seats on top of the FPGA (connector shown in pink on the right). The PCB processes the 32 analog input signals, digitizes them, and passes them to the FPGA. This board required 6 layers to allow for the trace-out of the ADC and for good analog signal treatment (length matching, appropriate grounding).

The first important component on the PCB is the input RF connector, which will connect to the SiPM array output in 32 channels. From there, the signals go through a fully differential amplifier, each group of 4 which has individual power rails to amplify, filter, and convert the signal into the correct ADC input. Then, in the middle of the board is the ADC, which takes in all 32 channels and outputs 16 LVDS lines to the FPGA connector.

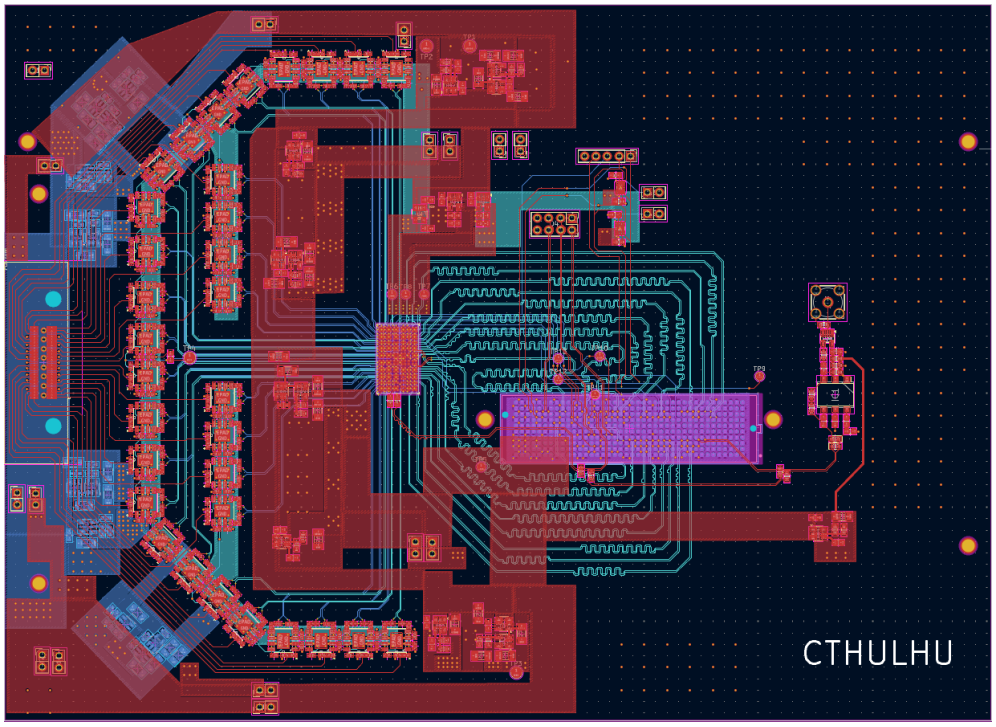


Figure 4: CTHULHU PCB layout.

4.2 FPGA Design

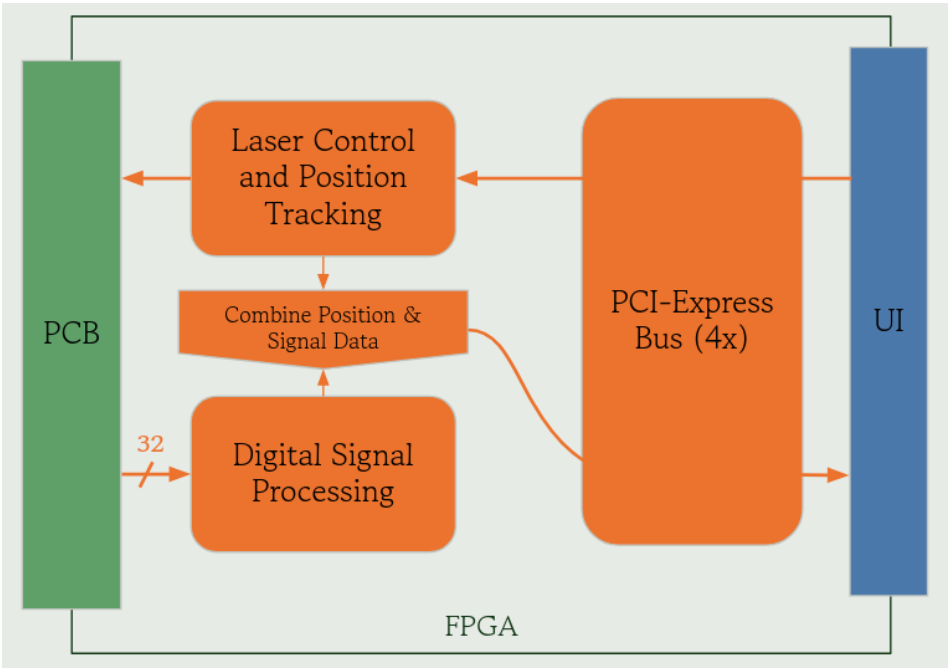


Figure 5: CTHULHU FPGA design.

4.2.1 LVDS Interface

The LVDS interface is the primary input interface for the FPGA. Signals come in along 16 LVDS lines from the ADC. Along with these lines there is a data bit clock and a frame clock. We selected a serialization factor of 12 for the ADC. As an input, we used the SelectIO IP block from AMD to handle differential signals, impedance matching, LVDS standards, and constraints on the input lines. Because this block is standardized, it does not quite fit our use case fully. This block spits out 4 bits at a time in parallel, along with a clock synchronized to this. We need to do some additional processing here to get this to a usable form, but to understand that we first need to understand how the ADC packages and sends serialized signals.

In 32 channel mode, the ADC multiplexes the signals over the LVDS lines. It sends one frame at a time switching between ADC inputs. As an example, over the LVDS lines, we get 12 bits (least significant bit first) from one ADC while the frame clock is high and the bit clock is pulsing for each bit; let's call this the input from ADC line "A". Then, the frame clock will go low and we will receive 12 more bits as the input from ADC line "B". The ADC will continue to follow this pattern sending serialized, multiplexed, data over each ADC line in the pattern A, B, A, B.

Ideally, we want to be able to process an entire frame at once in parallel. To achieve this, we need to perform a partial deserialization step, and then a demultiplexing step. First, we need to partially deserialize the rest of the data. From the SelectIO block, we receive 4 bits at a time. We receive three chunks of ADC frame A, then three chunks of ADC frame B. To reconstruct the 12 bit chunks we expect, the partial deserialization step saves intermediary values, and spits out the entire 12 bits of an ADC frame every 3 clock cycles. This still outputs in the pattern A, B, A, B, but we get the correct 12 bit frames instead of 4 bit chunks. We have one of these for each LVDS wire, so there are still 16 channels of multiplexed data. We still need some way to rectify this.

Finally, we do a similar step to demultiplex the remaining wires into the correct 32 channels. This lets us output all of the data for a frame in serial to the Framinator. Initially, we are getting data at about 720 MHz. The SelectIO block steps this down by a magnitude of 4 to 180 MHz. Partial Deserialization steps the signal down by another factor of 3, to 60 MHz, and then the Demultiplexer divides the clock rate by 2, to get 30 MHz. This makes sense because the laser runs at this speed, so we ultimately get one frame at roughly the speed of the laser pulses.

4.2.2 Framinator

The Framinator is the name given to the FPGA component that combines the signal data (received from the LVDS interface) with the correlated position, so that the data can be reconstructed in software. Its purpose is to act as a buffer across a crucial clock boundary: the one between the LVDS interface (taking data at 30MHz/channel) and the internal chip fabric

(operating at 100MHz). It also formats the LVDS and position data so that it can be transmitted within the FPGA to the PCI-Express interface via an AXI4-Stream. The information is packetized into two 256-bit packets, which are then reconstructed on the host PC.

4.2.3 Position Control Unit

The Position Control Unit (PCU) takes in several parameters from the user and uses them to build frames of 8000 coordinate samples. These frames are generated at a rate of 20fps and samples are outputted to the PCB at a fixed rate of 160kHz. This rate is controlled by a SPI controller ensuring it is consistent at all times. The rate of 160kHz is important as it ensures all readings are taken at the correct position and allows the Framinator to match incoming signal data with the outgoing position data. The PCU is designed in such a way that allows for the user to have a lot of control over the shape the frame will take, allowing for flexibility in the future UI implementation.

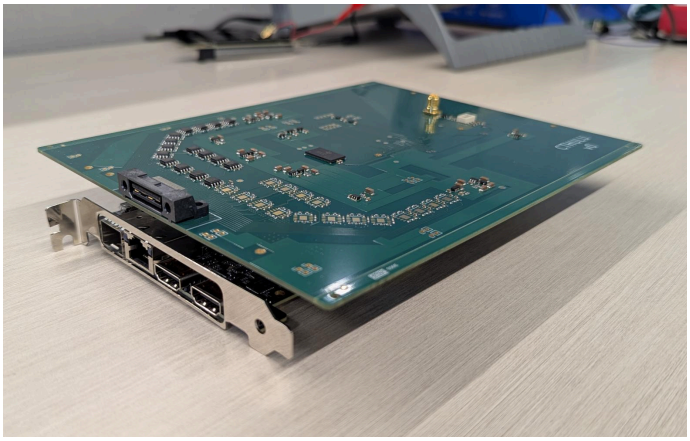
4.2.4 PCIE Interface

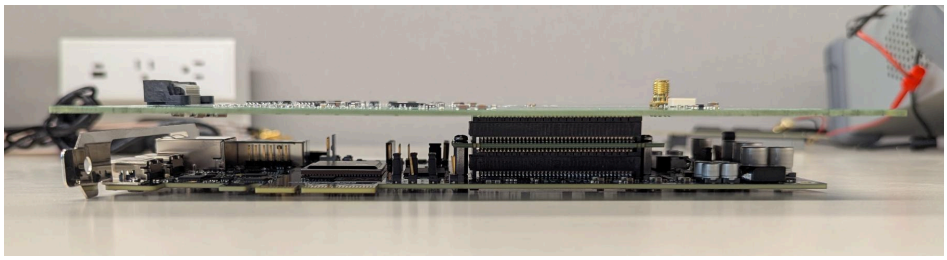
The PCI-Express interface uses an open-source XDMA driver to communicate between the PCIe DMA and the C++ firmware library. It receives 256-bit-wide packets from the Framinator into the stream buffer, which are marked with TLAST every couple of packets to transfer the data at an appropriate rate without losses. The PCI-Express interface also streams the control parameters for the PCU via an 256-bit-wide AXI4-Stream interface. In addition, the PCI-Express interface connects via an AXI-Lite interface to a SPI FIFO, which sends the initializing register writes in succession to bring up the ADC.

4.3 Firmware Design

A C++ library is used to abstract the XDMA PCIe interface into something usable, communicating with the driver through the file system. It takes the packets sent from the Framinator and reconstructs them, placing them in single lines in a CSV file. It also can stream all the necessary parameters to the PCU by taking a struct and consistently unpacking it into a data stream to send to the PCU. In addition, with a single command the firmware can send the initializing register writes through SPI to bring up the ADC and check that the ADC is initialized via the XDMA file system.

5. Parts and Implementation





We were able to ship our board fully assembled from JLC, and it seated on top of the FPGA as shown in the above images. This allowed us to individually test 1) the input data path through the connector on the PCB, through the FDAs and ADC, and coming in to the LVDS input lines on the FPGA, and 2) the FPGA data path, where the LVDS interface is given data and can be visualized on the host PC.

6. Cost Analysis

Below is a table of how much we spent on components:

Differential Amplifiers	$337.32
ADC	$486.35
Board Fabrication	$172.12
PCB Assembly & Other Components	$351.35
Testing	$33.31
PCB Total	$1380.45
FPGA Total	$699.00
Subtotal	**$2079.45**
Tariffs	$621.20
TOTAL	**$2700.65**

The capstone budget was spent on the ADC and some of the differential amplifiers, and the rest of the budget was supplemented by the labs we worked with. While this all may seem like quite a lot, subtracting out tariffs, and accounting for the fact that we populated 2 PCBs, the final product we created ended up being around $1400 to manufacture. Compared to already existing industry standard solutions, this is a significant magnitude less. The previous solution used by the lab, which has support for half the channels and has been described to us as “wildly inconsistent” cost about $10k for one board. Furthermore, the other solutions that were being considered as an alternative to our project cost around $30k per system, and would require the lab to work with CAEN (the company manufacturing this board) for any modifications made to the process.

7. Conclusion

Throughout this project, we were able to work with two northeastern research labs to develop a cost-effective solution for controlling a brain imaging laser. We designed and constructed an efficient solution for 32 channels of data collection, digitization, and processing. We also realized that not only is our board delivering an unbeatable cost to functionality ratio, but it also has a much wider scope than initially planned. We realized that we essentially built an open source 32 channel oscilloscope, specifically tuned for biomedical systems and real time processing.