DESIGNING A GUIDED AND AUTOMATIC ANALYSIS PROCESS FOR ANALOG AMPLIFIER REPONSE

by

Alex Chacko

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Electrical and Computer Engineering

2022

DESIGNING A GUIDED AND AUTOMATIC ANALYSIS PROCESS FOR ANALOG AMPLIFIER REPONSE

by

Alex Chacko

Approved: _____
Xxxx Xxxx, Highest Degree
Chair of the Department of Xxxx

Approved: _____
Xxxx Xxxx, Highest Degree
Dean of the College of Xxxx

Approved: _____
Xxxx Xxxx, Highest Degree
Dean of the College of Xxxx

Approved: _____
Louis F. Rossi, Ph.D.
Vice Provost for Graduate and Professional Education and
Dean of the Graduate College

ACKNOWLEDGMENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Chapter 1

INTRODUCTION

1.1 SLEDs Background

Infrared imaging and detection is used in various industries and services around the world. Since infrared light exists outside the human-visible light spectrum, it requires specialized hardware and testing procedures to ensure proper performance. Infrared applications generally require high speed and precision, which means that testing infrared systems is a vital step in ensuring their success.

A system must be "statically" and "dynamically" calibrated with an accurate infrared source. A static calibration only requires scientific test equipment to measure radiance. Dynamic calibration is much more involved and requires a throrough understanding of the system in question. To dynamically test, the system is placed under more realistic and changing conditions, such as having to react appropriately to moving objects or images. A common solution for this need is the Infrared Scene Projector (IRSP) which can be used to project infrared images and video to test a desired device.

IRSPs display realtime IR scenes that simulate true IR signals for their targets. Since infrared depends on physical information (temperature), simulating this as a signal requires specialized hardware. This hardware is used to turn commands from the users into analog display signals as well as read feedback from the display device and adjust its own parameters accordingly. This "hardware-in-the-loop" approach can allow a system to accomodate for various physical effects or issues within the display. Since infrared displays are analog devices, they cannot be precisely analyzed without some sort of readback and processing to view output. A typical system design is shown below.

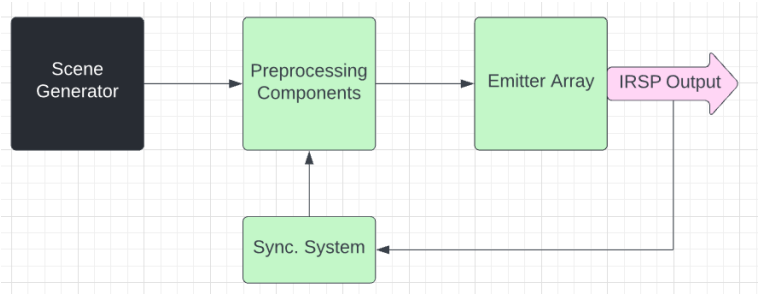


Figure 1.1: General IRSP architecture

Testing is one of the most important aspects of any system design, as it is the bottleneck for how many reliable systems can be built & validated. This work consists of a new approach to manually test the high-speed analog amplifier circuits within the system, as well as considerations & redesigns for the next generation of these devices.

1.2 Motivation

IRSPs can be characterized by multiple factors, of which two of the most important are frame rate and output temperature range. Frame rate can be described as the number of discrete outputs from the display device per second. The frame rate is directly tied to the rise time of individual pixels, which is fully dependent on the display architecture and physical properties. The temperature range represents different "colors" or ranges of IR light that the projector can display. An ideal device has both high refresh rate and high max temperature to overcome the challenges of temperature-based projection. [9]

Much of the IRSP market is dominated by resistor array devices. By heating up resistors (mapped to each pixel) to specific points with controlled current input, the device can simulate temperature output for an infrared image. The use of resistors limits the benefits of this approach, however. [4] Resistor arrays depend on raising temperatures to the actual desired temperature of the projected image. While innovative, resistors cannot dissipate heat quickly, limiting their rise time. The characteristics of the resistors are a hard limit on refresh rate, forcing most designs to stay under 500Hz. Additionally, higher projected temperatures will further increase the rise time, since

the resistors will have to physically cool down from this temperature level. By limiting the max temperature output, designers can ensure that pixel rise time will consistently stay within a certain range.

Since 2008, CVORG has been at the forefront of developing an Infrared LED-based IRSP with a full suite of custom hardware and software control. [2] The success of the current iteration of the system has allowed us to continue researching improvements and changes to the system that would further expand on the technology's potential. Our technology is now targeting a 1024 x 1024 display at 2KHz.

After successfully developing the technology to a certain point, it was considered stable enough to reproduce and iterate on. The process of building and testing each component of the system provides one with an intimate knowledge of the functions and requirements of individual hardware components, as well as limitations to be targeted later.

One of the most important links in the system's chain consists of the digital-to-analog conversion and subsequent amplification. These amplified analog signals are used to drive the display, making their functionality and speed a top priority. [7] As such, it was imperative to create reliable testing procedures for such components. The relevant systems, components & design methodology for this portion of the system will be described in detail below.

Chapter 2

SYSTEM OVERVIEW

Using a combination of digital commands and analog signals users can precisely command and utilize the display at the high frequencies required by the application. The system has been developed over the years to be user-friendly and configurable for many different applications. The associated technologies will be briefly explored below.

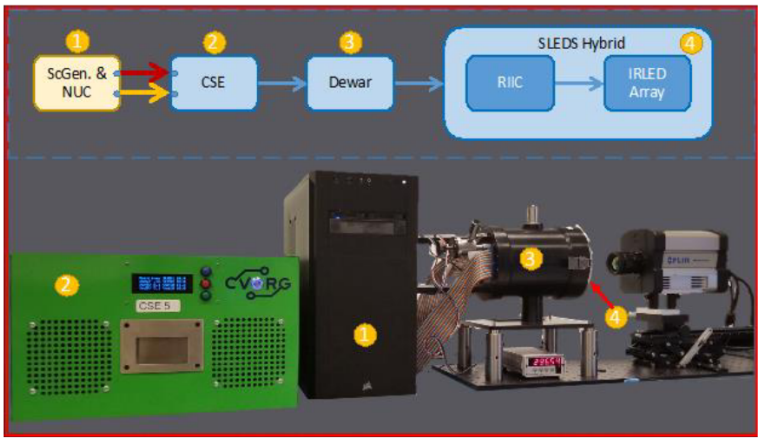


Figure 2.1: SLEDS IRSP Overview [6]

2.1 SLEDS Array

As mentioned before, infrared projection requires high-speed analog signals and full user control. The LED array itself is hybridized to a Read-In Integrated Circuit (RIIC), a 10V chip which is digitally instructed by a close-support unit. This hybridized unit in its temperature-controlled enclosure is referred to as the Dewar. The first iteration of this LED array named SLED (Superlattice Light-Emitting Diode System)

was constructed in 2008 with a resolution of 68 x 68 pixels. In this early stage, no driver had yet been developed to control the RIIC. [6]

To drive such a complex analog device, specialized electronics and communication are required to turn human commands into infrared signals. The first drive system for the SLED array was developed in 2011 and helped lay the groundwork for the next iteration of SLEDS, consisting of a 512x512 display and 48 micrometer pitch size.[6]

Further versions of SLEDS were created over the years to fulfill different requirements. TCSA (Two-Color SLEDS Array) was developed in 2016 and capable of driving the LEDs at two separate wavelength bands. NSLEDS (N3 Superlattice LED System) was also developed around this time. This array was capable of driving a 1024x1024 display for the first time. The next advancement, HDILED (High-Definition Infrared LED) was the first 2048x2048 display in the world when it was finished.

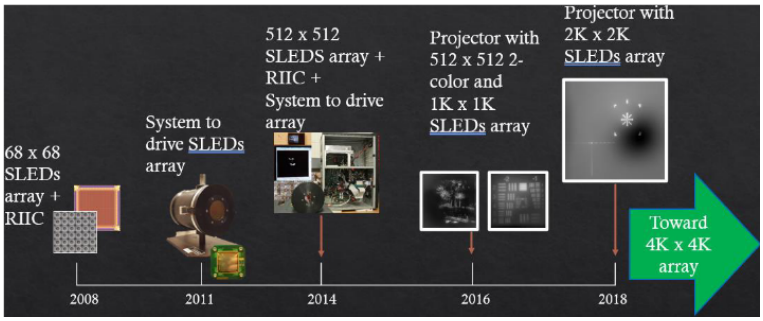


Figure 2.2: SLEDS Timeline [6]

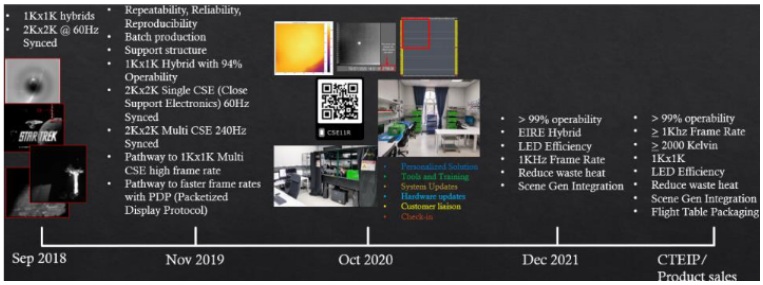


Figure 2.3: SLEDS Timeline, Part 2 [6]

2.2 RIIC Overview

The RIIC is a 10 volt IC that is hybridized with an array of IRLEDs to directly drive them using CMOS logic. A dual-circuit system allows the array to display at different dynamic ranges, a "weak gear" and "strong gear". Each pixel is driven by a

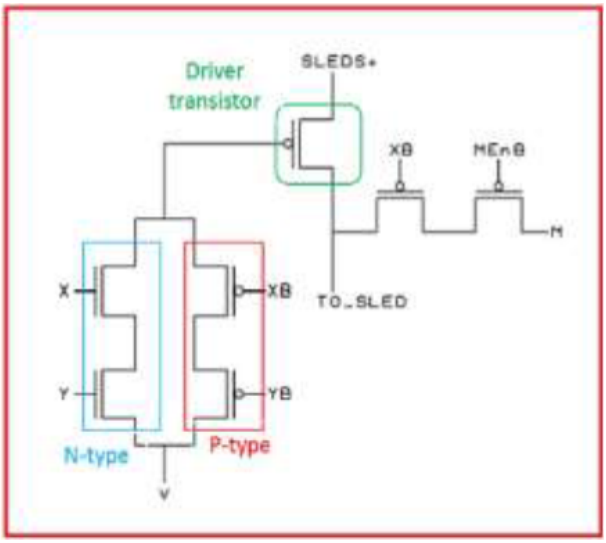


Figure 2.4: Single Pixel RIIC Schematic [5]

strong/weak pair of NMOS transistors, which do not take up as much space as PMOS and allow for greater resolution as a result.

The RIIC is divided into four quadrants that can be individually commanded. Due to the nature of the driver firmware that will be discussed later below, this quadrant approach allows great control over the display. [5] Each quadrant has 2-32 input channels depending on the desired configuration. Configuration is performed using an SPI register on the RIIC, shifting in data when commands are received.

2.3 CSE Overview

The RIIC/LED hybrid is driven by a close-support electronics system, or CSE. While the CSE was a response to the need for a RIIC driver, it is also the culmination of many years of work and development. The system architecture consists of an FPGA, DACs, Amps, and I/O boards. This array of hardware completes the loop from a user on any PC to the IRLED array, making the projector a viable product. This hardware

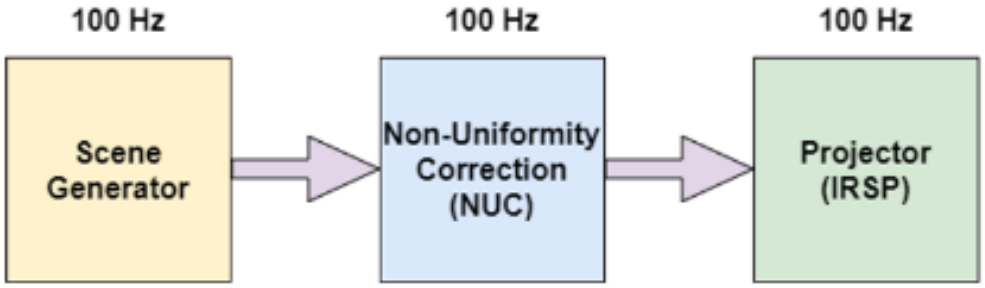
has been redesigned and iterated on for years, and is now being fully upgraded to a second revision consisting of smaller components and more efficient signal integrity strategies.

2.4 Scene Generation and NUCPC

The final ingredient necessary in the IRSP chain is the Scene Generator. While scene generation technology is far beyond the scope of my work, a general overview is necessary to understand the system. A scene generator is to the projector what a media player is to a television. Most of these scene generators are proprietary black boxes, documented by the creators to be sold to end users [6] Scene generators can communicate over nearly any physical interface, but the standards must be agreed upon between the manufacturer and user. A common display protocol is DVI, as it is fairly well documented and easy to understand. HDMI is another possible choice, but it is a more complex and less openly documented standard.

To generate a scene, a GPU is usually necessary to create images and transmit them over the chosen display protocol. The generator will use frames created on the GPU to send over to the IRSP. The next step is called "Non-Uniformity Correction," which is the process of analyzing predetermined output from the IRSP to determine radiance properties of each pixel. Pixel data is placed into a table used by the scene projector and used to accomodate for irregularities in the LEDs. These irregularites are nonlinear which is why they require analysis and writeback rather than simple algorithmic correction. Once the NUC process is complete, the desired images can be projected onto the display device.

Since our research group did not have access to a proprietary scene generator, they developed a pseudo-scene projector using what was dubbed a Non-Uniformity Correction PC (NUCPC). This NUCPC is connected to the CSE through HDMI, and image data is generated by an Nvidia graphics card[6].



All components of this process operate at the same speed and resolution.

bandwidth	**= *resolution* * *bits* * *fps***
resolution	: number of pixels (including blanking)
bits	: number of bits per pixel
fps	: frames per second
bandwidth	: bandwidth requirements in bits per second

Bandwidth requirements of a scene projector using a traditional display protocol.

Figure 2.5: Example scene generator in IR system [6]

Chapter 3

PDP VS. CONVENTIONAL DISPLAY PROTOCOLS

3.1 DVI & HDMI

Early scene generators used the DVI protocol, which is a simple and well understood standard. Unfortunately, they are also quite large connectors that did not fit well into the CSE design. For this reason, HDMI was chosen as the display protocol for NUCPC to CSE. HDMI uses the same electrical levels as DVI which kept it compatible with preexisting scene generator technology.

While they are good solutions, these display protocols are also rather limited. They are forced to remain compatible with even older protocols such as VGA. An example of typical display function is shown below. Video data is sent in sequential horizontal lines. Once the boundary of the display is reached, a "blanking period" is entered to reset the device's row/frame. The blanking period is recognized by a sync signal, which means that bandwidth and time are required for non-video data. If blanking periods were able to be reduced, it would inversely increase the potential frame rate.

Fixed frame-rate display protocols are also inherently limited. If a new frame is not necessary, the last frame will be retransmitted. Bandwidth is static which is useful for the reliability of consumer products, but less useful for a high-speed infrared projector. As such, utilizing the potential bandwidth of HDMI became a top priority and the foundation for the CSE firmware.

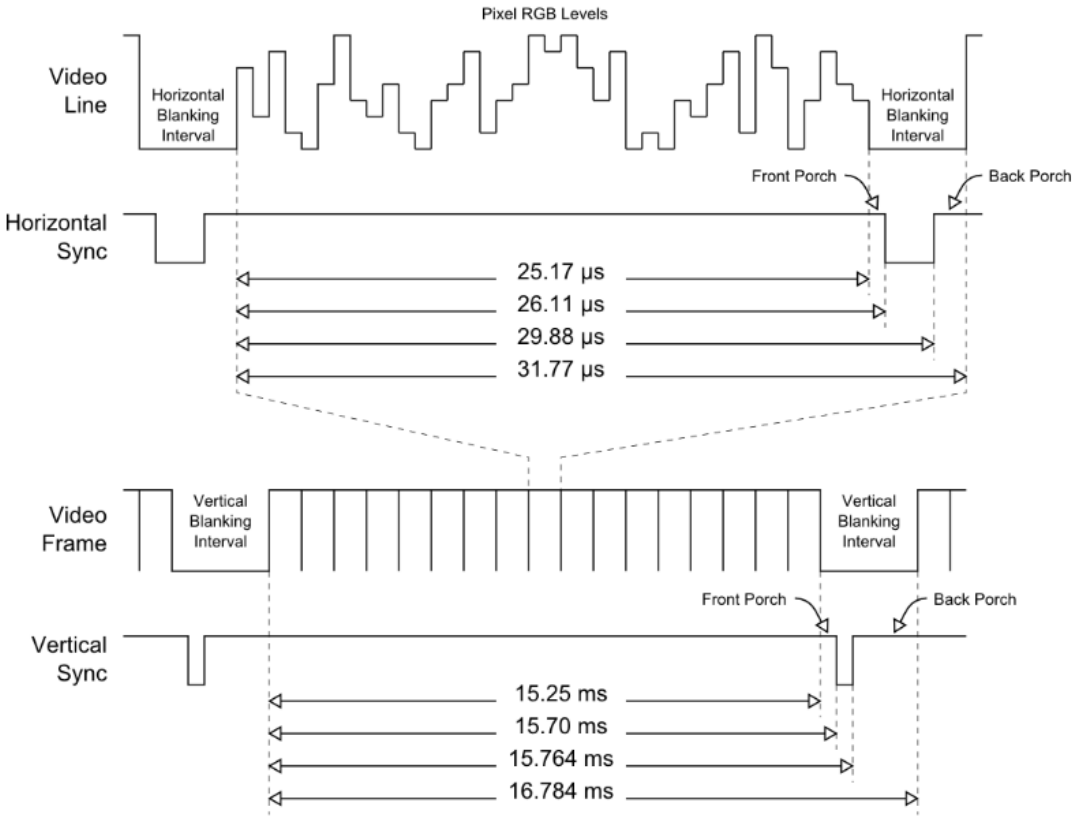


Figure 3.1: VGA display protocol timing diagram [1]

3.2 PDP

The Packetized Display Protocol (PDP) is the communication protocol for the CSE to LED array. The initial vision for PDP consisted of a "physical layer agnostic method of pixel data transmission capable of providing and addressing latency guarantees while maximizing bandwidth utilization" [3]. Unlike traditional display communication protocols, PDP does not operate on entire display frames, and instead splits any projected scene into regions known as sub-frames. These sub-frames can be written to individually, greatly increasing bandwidth utilization and allowing for high frame rates beyond any traditional HDMI connection.

Sub-frames are marked by the PDP firmware with packets denoting their x and y coordinates. PDP communication packets will then consist of a packet header for

draw location followed by the desired data. This divide-and-conquer strategy allows the NUCPC to only send frames when they require updates and therefore update relevant subframes at much higher frame rates than normally possible [6]. IRSP is an incredible application for this concept, as many scenes only have small regions that require updates. An example scene may contain a subframe changing at 200Hz, another at 50Hz, and yet another remaining static. Over a traditional display protocol, displaying any of these regions at the appropriate rates would be impossible due to the entire frames being sent at preset intervals. PDP, however, allows the NUCPC to utilize far more HDMI bandwidth to only send packets relevant to the subframes.

Table 1 Examples of PDP Packets

Name	Type ID	Type Specific Fields				
Ignored	0x0					
Draw Region	0x1	X start	X end	Y start	Y end	Data
Array Reset	0x2	Quad mask				

Figure 3.2: Example PDP Packet [6]

PDP can be used in two modes. The method of splitting and refreshing subframes is known as "streaming mode". Users may also opt to use it in a typical display mode where the entire frame is sent at a fixed refresh rate. This is known as "backwards compatibility mode". While there is nothing innovative about this mode, it is necessary to maintain compatibility with existing scene generation technology.

PDP was previously described as being "physical layer agnostic" [3]. The protocol can theoretically be implemented over any physical medium such as DVI or ethernet. The initial implementation utilized HDMI to maintain compatiblity with existing technology, but a future revision is moving the physical medium to SFP+. Improvements to the CSE hardware will be briefly discussed in a later section.

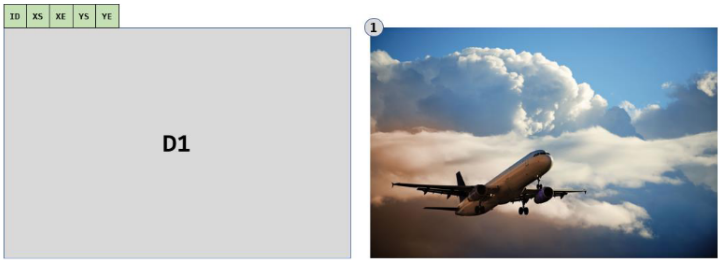


Figure 7 PDP Backwards Compatibility Mode

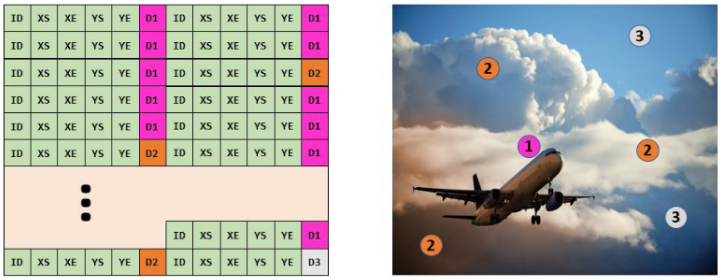


Figure 8 PDP Stream Mode

Figure 3.3: Packet organization in each mode [6]

Chapter 4

CSE ARCHITECTURE

4.1 CSE Hardware Overview

The CSE is comprised of two gigabit-speed input cards, a mainboard powered by an FPGA, eight DACs connected to the FPGA, eight amplifier cards (one for each DAC) and a pair of interface boards [6]. The input interface cards have HDMI I/O ports and connect to the NUCPC or external scene generator to receive input. Each interface card is connected to the main FPGA via FPGA Mezzanine Card (FMC) slots. The FPGA is host to a MicroBlaze soft processor and uses its eight remaining FMC slots for DACs. Each DAC outputs to an amplifier card, creating the analog driving inputs for the RIIC. All amplified signals are routed out through the interface board to be connected to the RIIC through ribbon cables.

4.2 Datapath

Once the scene generator provides video input, the HDMI cards decode and send the video stream to the FPGA. Since the FPGA uses a soft processor, it can be interfaced with in C to control registers [6]. The processed data is then distributed amongst the eight DAC slots in accordance with the packets defined by PDP. Each DAC card holds two dual-channel DAC components for a total of 32 channels. Once the digital input has been converted to analog, the amplifier cards boost the output and send it through the interface boards. Each interface board has power input from a PSU and power output for the DACs and Amps, and also contain boost components to convert the 2.5V amplified output to 5V for the RIIC to read.

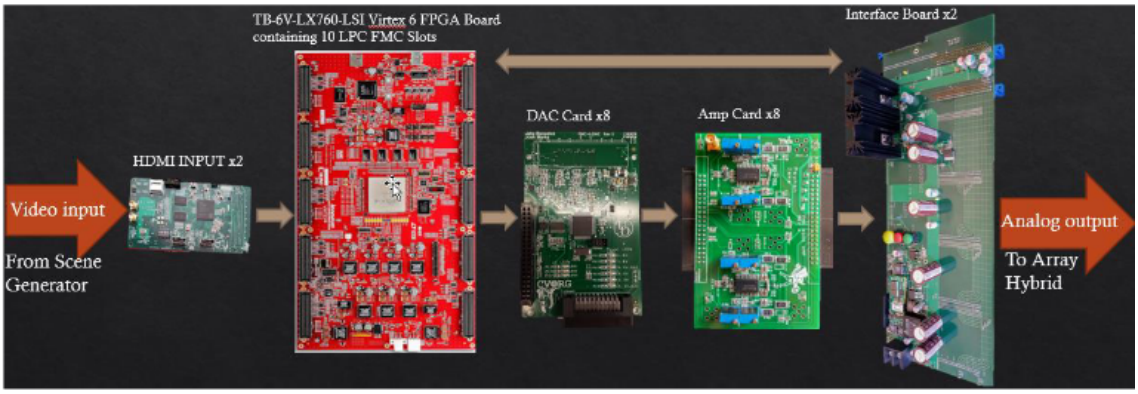


Figure 4.1: CSE Architecture Overview

4.3 Hardware Revisions & System Upgrade Plans

After many years of development, the CSE reached a level of maturity to be considered a cohesive product (internally dubbed Nessie). Due to ever-increasing requirements from customers and breakthroughs in the technology, the group has moved to target a multi-CSE platform capable of each driving a quadrant of an array (now known as Nessie2). By increasing the number of output machines, frame rates can be quadrupled and resolution can be doubled [6]. To compare, the previous system (CSE + SLEDS) would effectively be one quarter of this entire setup. The main challenge facing the system architecture is proper synchronization, as this is vital to ensure all quadrants are being controlled in the same time domain.

Another major change coming to the system architecture is the placement of all analog systems onto the Dewar enclosure rather than inside the CSE. This means the CSE will only handle digital video conversion, and send these values to amplifiers directly on the Dewar. The overall datapath will remain the same with a scene generator interfacing with the FPGA through an SFP+ interface. The FPGA will distribute video data amongst the DACs and then routed outside to the Dewar to be amplified and read. Plans for the Nessie2 amps based on lessons learned from the Nessie1 amps will be discussed later.

Chapter 5

CSE ANALOG COMPONENTS

5.1 Analog Circuit

In order to develop a better testing procedure for the current amplifier circuits as well as begin development of the next generation, some exploration of the initial designers' writing was required. Analog output from the CSE was addressed to each pixel of the IRSP. Due to the sensitive nature of digital-to-analog communications and the unique load requirements of the system the entire system architecture had to be measured and mapped out. To do so, the system was split into various sections representing gain and load stages. [9]. The first stage (DAC Buffer) was responsible for terminating the differential output of the DAC and turning it into a single-ended signal. The next stage was the Cable Driver, a gain stage which was responsible for driving interface board communication and ribbon cable signals. This stage provides some, but not all of the voltage gain required to drive the RIIC from 0-5V. The ribbon cables would output to the final stage known as the Dewar Driver. This stage provides the remainder of the gain required to drive the RIIC at 5V. The Dewar output stage was a 10nF load and required a 0-10V swing from the amplifiers.

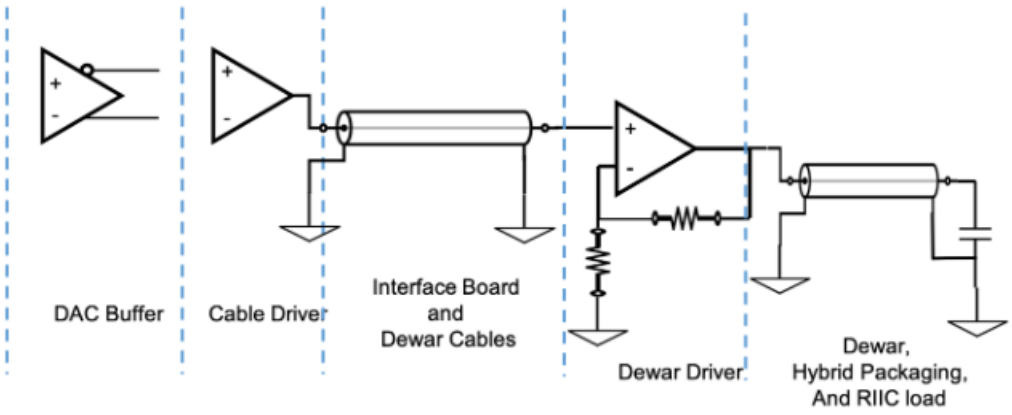


Figure 5.1: High-level schematic of the analog gain & impedance stages

5.2 Understanding the Requirements

To properly test, validate and repair the amp cards, one requires an understanding of the card's place in the system. The card was responsible for passing digital-to-analog converted signals with 16-bit resolution at high speeds, and any issues would be directly visible in the projected video. The amp card received inputs from a 16-bit DAC with a 250 Megasamples per second output [9]. For digital information to be properly decoded as an analog signal, the converter must settle on values at an acceptable speed with negligible margin of error. As bit resolution increases, the margin of error should decrease. Output voltage from the DACs has been previously tested and documented giving us a base for what voltages needed to be swept for validation.

5.3 Settling Time

In any digital-to-analog system, designers must account for physical characteristics of electronics. Every DAC has a delay between input and response, known as Delay Time. The DAC then enters a Slewing period, where it tries to approach the target current or voltage output as fast as possible (current level in the case of our specific DAC). As the DAC approaches the target, slope decreases and the DAC oscillates between over- and under-estimates of the value until finally settling on a DC output. While an ideal DAC will settle on a precise output, most DACs will continue

to oscillate between values if it is unable to discern their mean, especially in practical situations. The mathematical account of this phenomenon is called the Error Band.

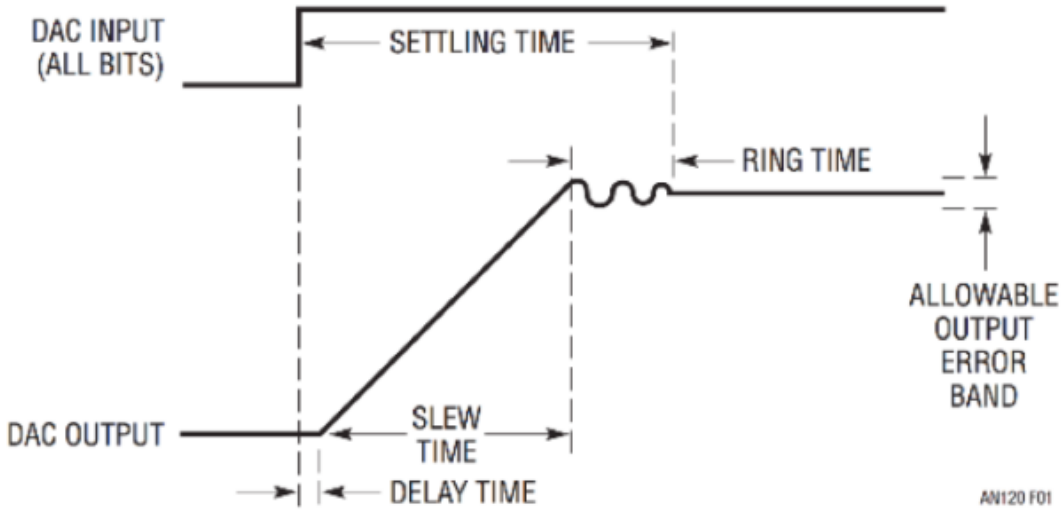


Figure 5.2: Example DAC output

5.4 Error Band

The maximum acceptable error band is notated as $\frac{Vout}{2^n}$ where Vout represents the maximum voltage output of the device and N represents the desired bit resolution. An 8-bit system with 4V max output, for example, has an error band of $\frac{4V}{2^8} = 0.0156V$ The settling time is the time it takes for our analog system to be bounded by this error band. A smaller error band makes calculating the final value more difficult for the device.

5.5 PCB

The amp card was designed to encompass the first and second stages (DAC Buffer/Cable Driver). Prototype and final PCB layouts are attached below. The load and bandwidth requirements meant that the amplifiers had to have a high slew rate (reflected as rise time in the Dewar) and bandwidth. These design choices were the

basis for the v2 redesign which will be discussed later. Each amp card contained two amplifiers, and each amplifier had their own ground planes which were unified at the DAC input and interface board output connectors. This is a common design choice meant to eliminate cross-talk between ground planes of sensitive analog components at high speeds. Each amplifier component was a quad amplifier, but only the first two were used in the design. While this decision was also made out of concern for cross-talk, the unusued pins wasted quite a bit of space and were one of the first motivations for a new PCB in Nessie2. The amplifier card was initially designed as a 2-layer PCB which proved to be insufficient for power delivery and signal integrity [9]. The board was redesigned to a 4-layer version and extensively tested to ensure the previous concerns were fixed.

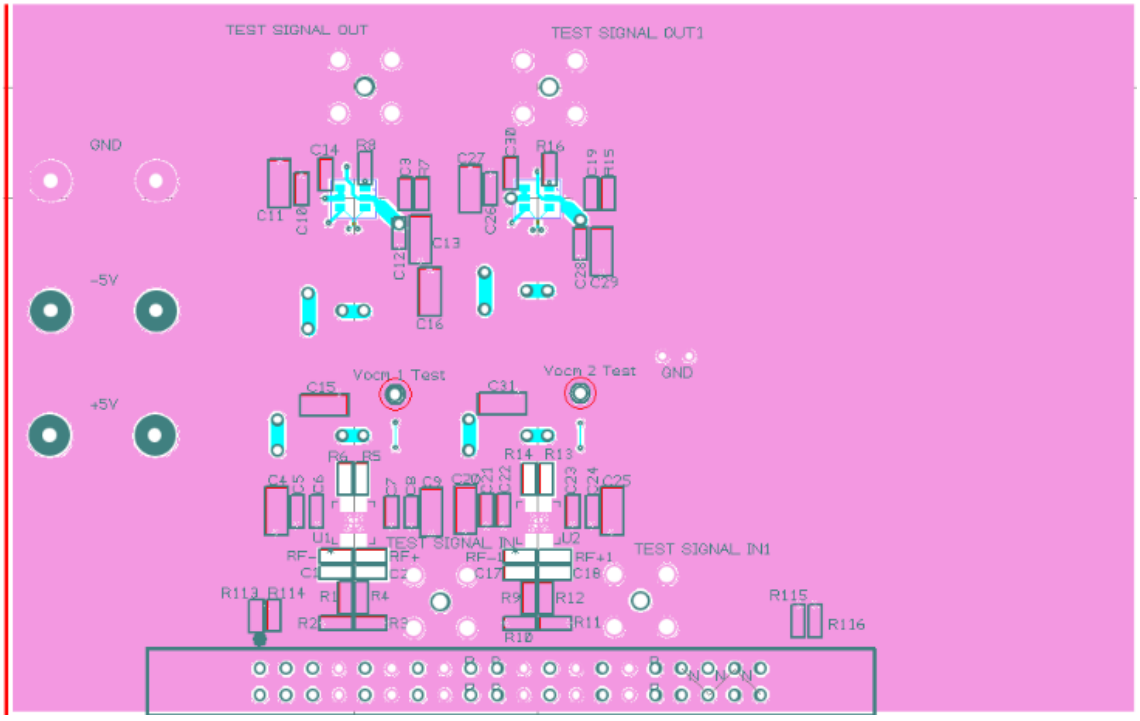


Figure 5.3: Prototype Amp Layout

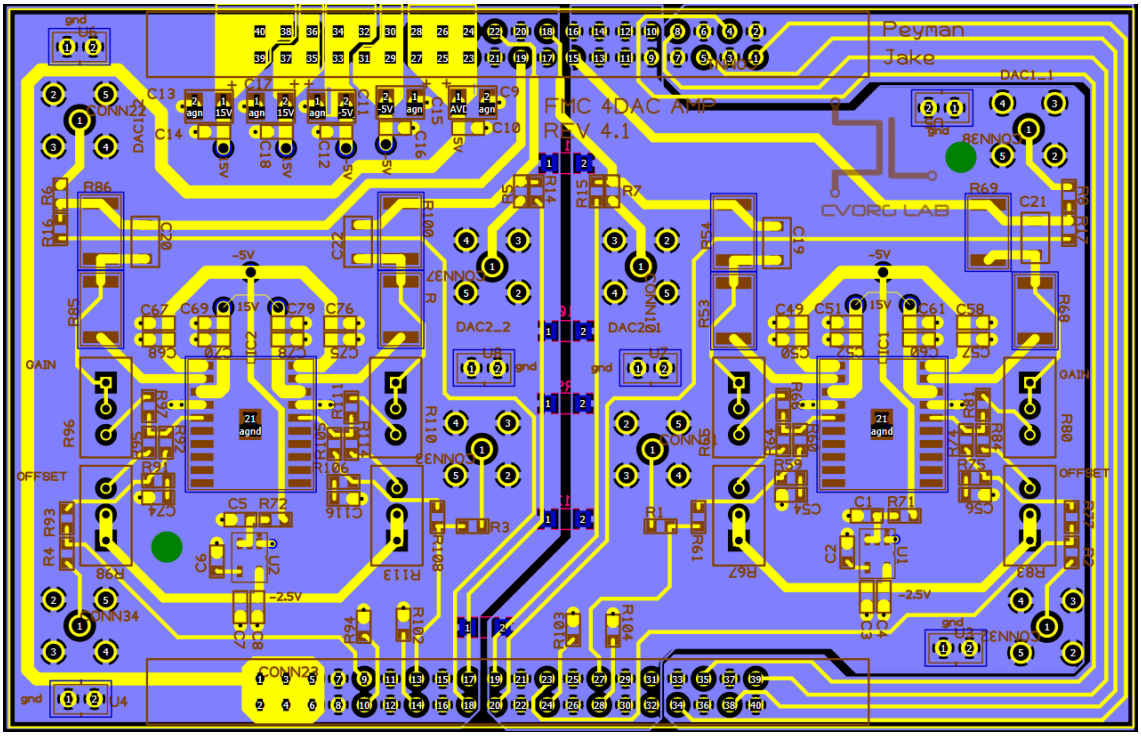


Figure 5.4: Final Amp Layout

5.6 Present Investigation and Review

Unfortunately, the original designer of the amplifier circuit did not design the testing procedure mentioned later. As such, there was no original documentation on the goals and expected outputs for the amplifier. There were multiple choices made in the PCB for theoretical future revisions that never happened. Some egregious examples include unpopulated coaxial inputs as well as a number of surface mount components. These component pads were placed with the desire to potentially investigate later. These further tests were never carried out and ended up amounting to quite a bit of wasted space on one of the system's key component boards. Due to the lack of test documentation, design methodology and stagnant development since its creation, the amp cards were targeted as a major oversight in the system's hardware that required new test procedures and a new design. Fortunately, having to use the poorly implemented test procedure allowed me to become familiar with the process and realize exactly what needed to change going forward. These new test procedures and design

reviews informed the development of the second generation of amp cards.

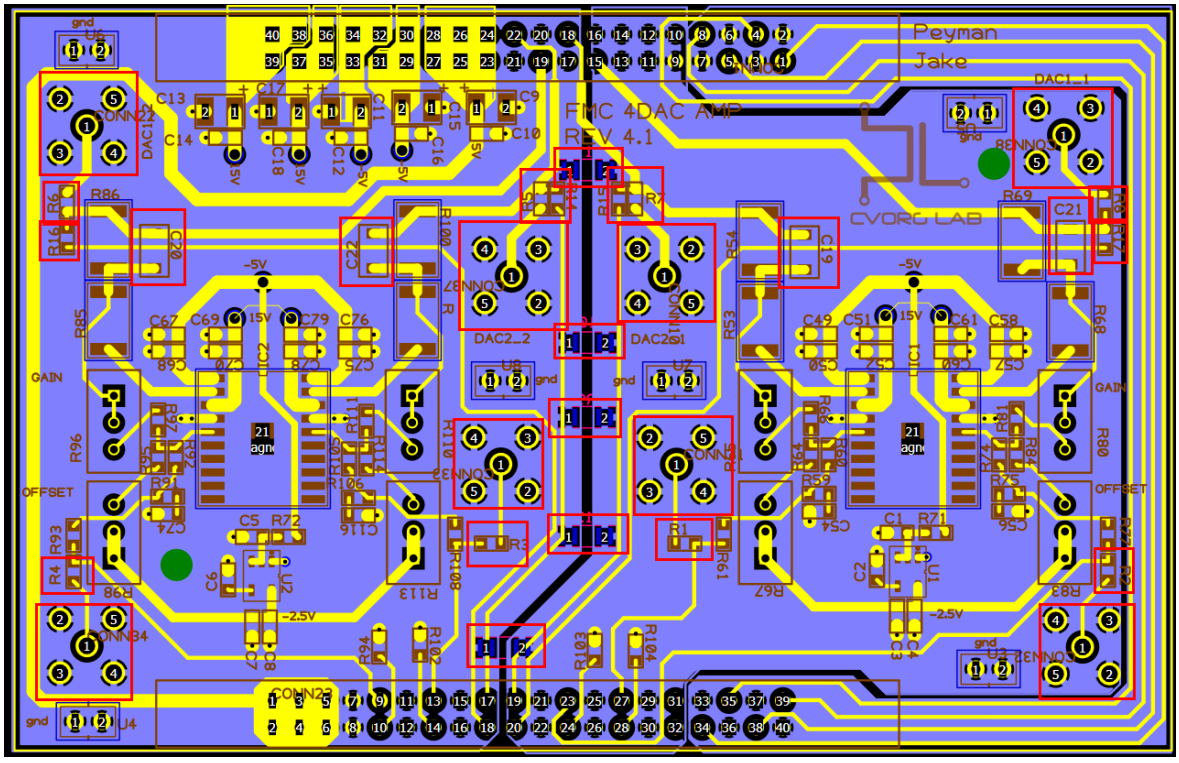


Figure 5.5: Unpopulated components on the amp card boxed in red. A great deal of space is wasted here

Chapter 6

PREVIOUS ITERATION OF AMPLIFIER TESTING

The first version of the test setup and procedure was not well documented or maintained. Due to hardware changes that were too costly to warrant a complete re-fabrication, every amp card had to first be electrically reworked. The post-rework test was a simple continuity and resistance check with a multimeter to make sure all new connections were stable. Amps were then hooked up to a function generator and power supply using jumper wires from their connectors. Input and output were viewed on an oscilloscope to ensure proper amplification. While this was supposed to be a sufficient emulation of their requirements in the system, the entire procedure was loosely documented and not well-enforced enough to ensure long term reproducability and reliability.

6.1 Electrical Rework

One of the first tasks I was given upon entering the group was to help rework each amp. The first issue was that the amplifier power rails (+15V/-5V) were not connected by default. Each amplifier's power rails had to be soldered to the power output using jumper wires. This first step already provided too much room for error, as jumper wires for power distribution would not necessarily be a better choice than simply integrating power lines into the PCB. Adding a person's hand solder in between increased the risks of cold solders or accidental shorts which should be considered unacceptable risks for power pins.

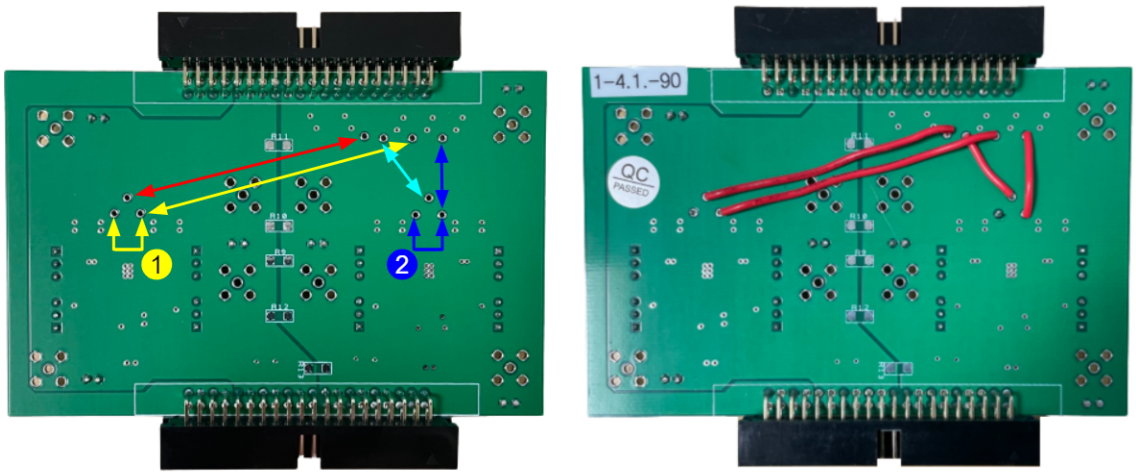


Figure 6.1: Rework of the power rails

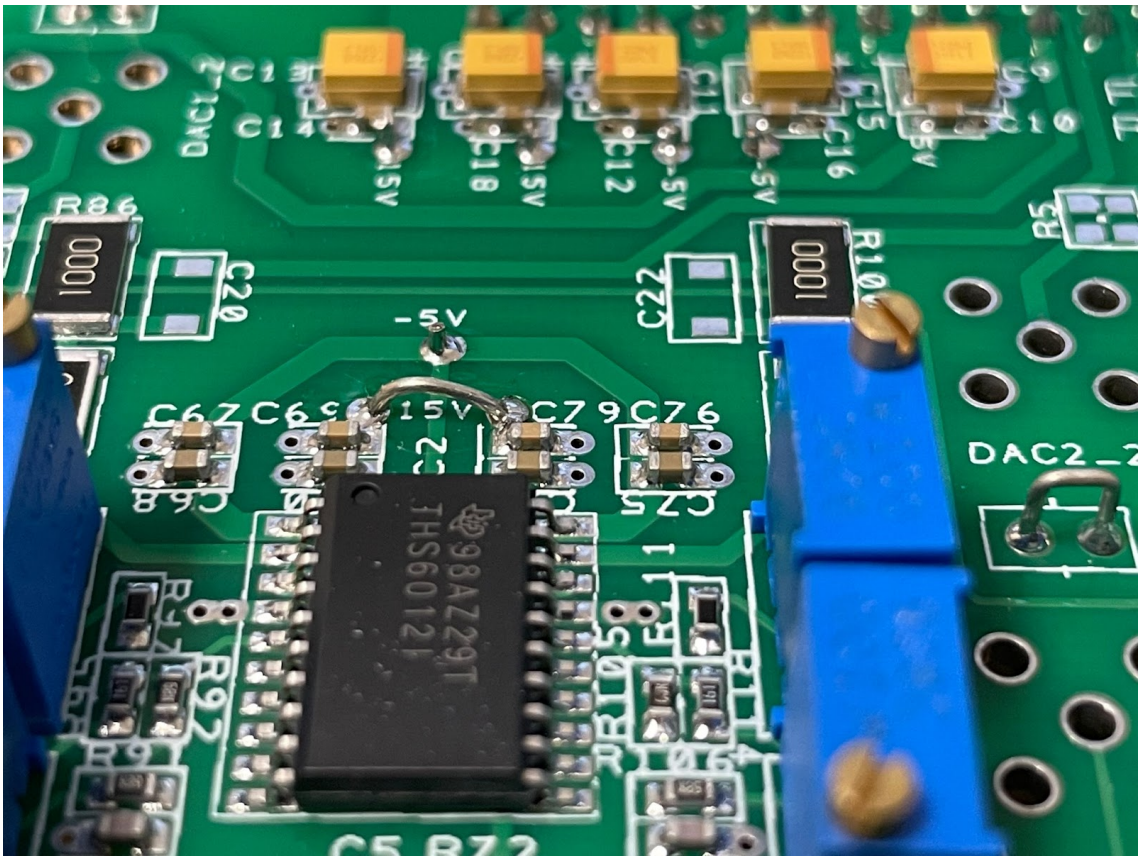


Figure 6.2: Power rails post-rework

6.2 Component Replacement

After bridging the power planes on the board, potentiometers and resistors had to be removed and replaced. The potentiometers were initially 100 ohms and used to control gain and offset of the amplified signals. These were replaced with 10 ohm potentiometers which was meant to allow more fine-tuned control over these values. The replaced resistors were also changed to new values to increase signal integrity. Once the rework was completed, it was to be given a brief electrical test (mentioned in the overview) to check the quality of all connections. While the continuity and resistance checks served their purpose, they were not sufficient for fully judging quality of the rework or identifying other issues early.

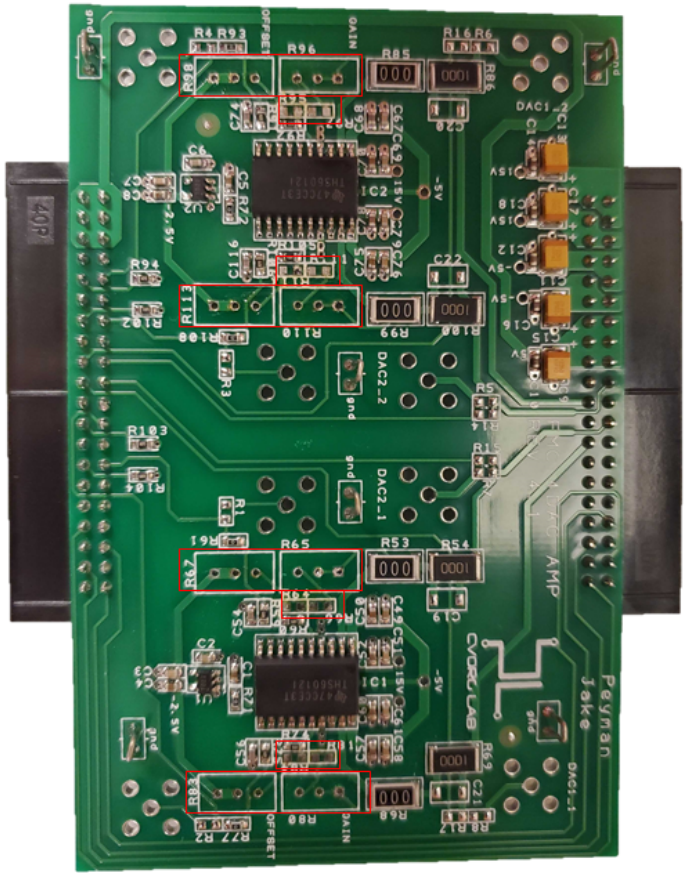


Figure 6.3: Components to be removed and replaced

The room for human error in this whole process proved to be a massive detriment. Despite a number of group members being experienced solderers, it was impossible to rework such a high volume of boards at a consistently high level of quality. The rework process introduced so many faulty cards that every post-rework card had to be "quality checked" by a senior group member to make sure the rework was acceptable. As usual, adding another intermediary step decreased the reliability of amps and the pool of boards we had available.

6.3 Electrical test

The first stage of the electrical test consisted of the electrical checks mentioned above. There was no enforced procedure beyond following instructions on a document, and a less experienced group member ran the risk of not fully understanding what they were testing. Without detailed instructions or an intimate knowledge of the purpose of the amp, this electrical check proved to be less and less useful.

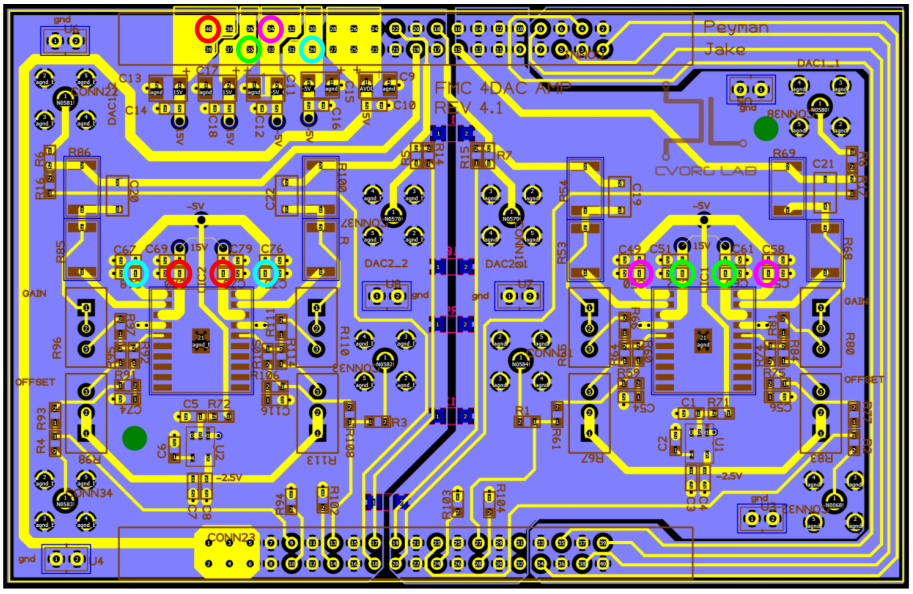


Figure 6.4: Official documentation for the continuity checks

The next testing stage involved hooking up instruments to simulate the type of load the amp card would have to support. This entire part of the procedure was the least well documented or understood and slowed down the validation process quite a

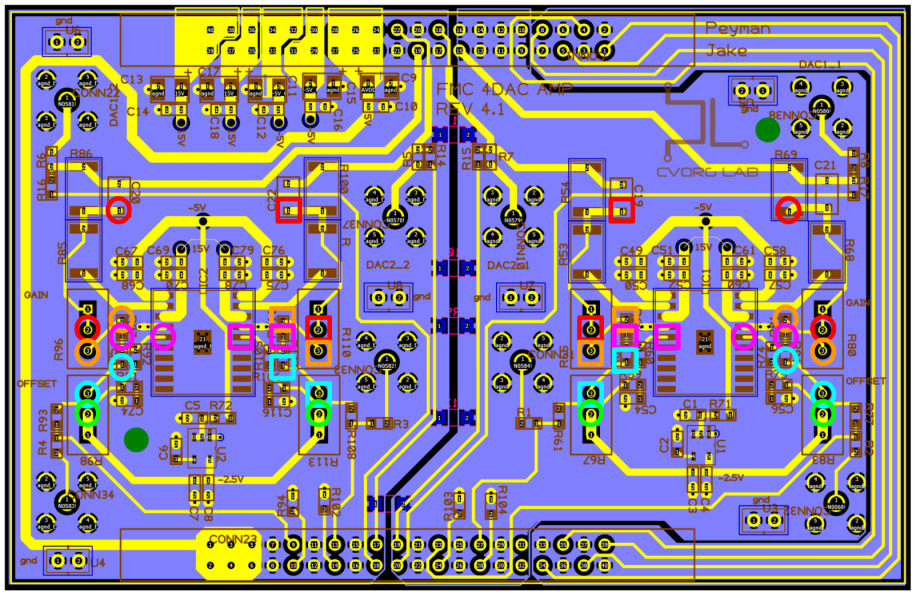


Figure 6.5: Official documentation for checking replaced components

bit. Simple breakout boards were hand-soldered for the amp connectors. Inputs were on the bottom connector and plugged into a function generator. Outputs were on the top connector and included pins for power connections (+15/-5) as well as output to the oscilloscope.

6.3.1 Function generator

The function generator settings used in the test were rather generic and barely modeled a realistic input for the amps in our system. Input was configured as a 1MHz square wave with an amplitude of 363mV and offset of 318mV. This function was also connected to an oscilliscope to compare against output. Using a high-speed square wave input was definitely a good idea, as a functional amplifier would output an equally square wave. Faulty components would not be able to replicate such sharp rise and settling time. The major downfall of this approach was that the function generator had to be individually tied to each of the card's four input channels, greatly increasing the amount of time a tester had to spend on moving pins and connectors. Having to manually set the wave up was also not the best idea, as this just added another area for the tester to make a mistake.

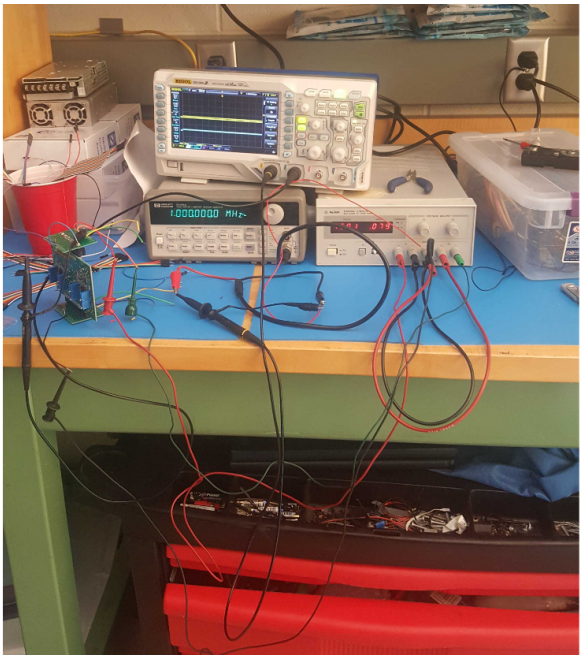
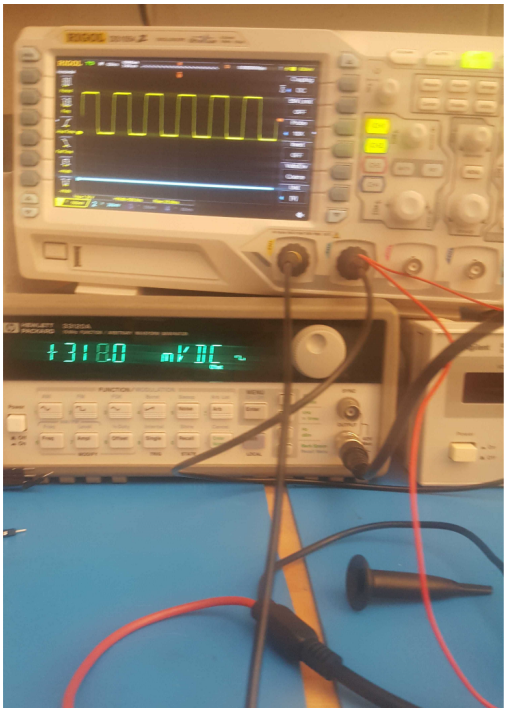
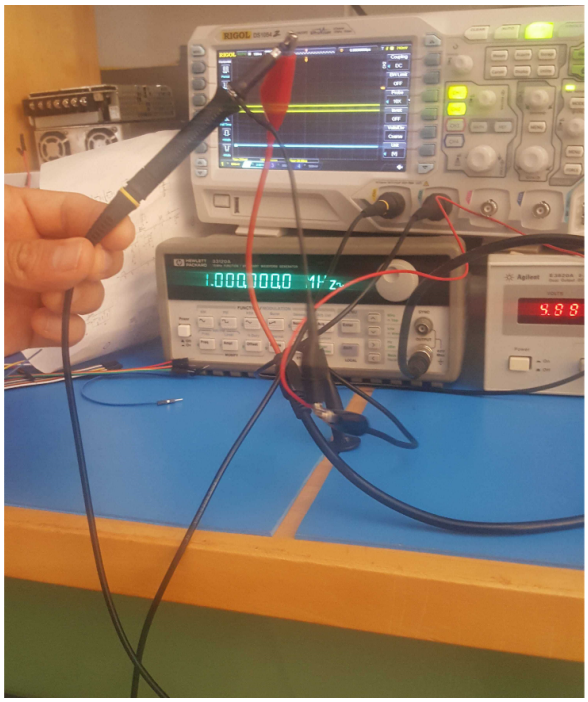


Figure 6.6: One of the only official pictures for a completed test setup



(a) Function generator settings



(b) Function generator to scope

Figure 6.7: Offical documentation for setting up the function generator. Insufficient to teach anybody new how to use the equipment.

6.3.2 Power supply

Setting up the power rails was a little bit less involved. The user only needed to attach +15V, -5V, and GND to the top breakout board. Connections were still made using extruding jumper wires and alligator clips, yet another point of concern when testing a large number of amps. There were hardly any examples of what a correct setup would look like in the testing document at this point.

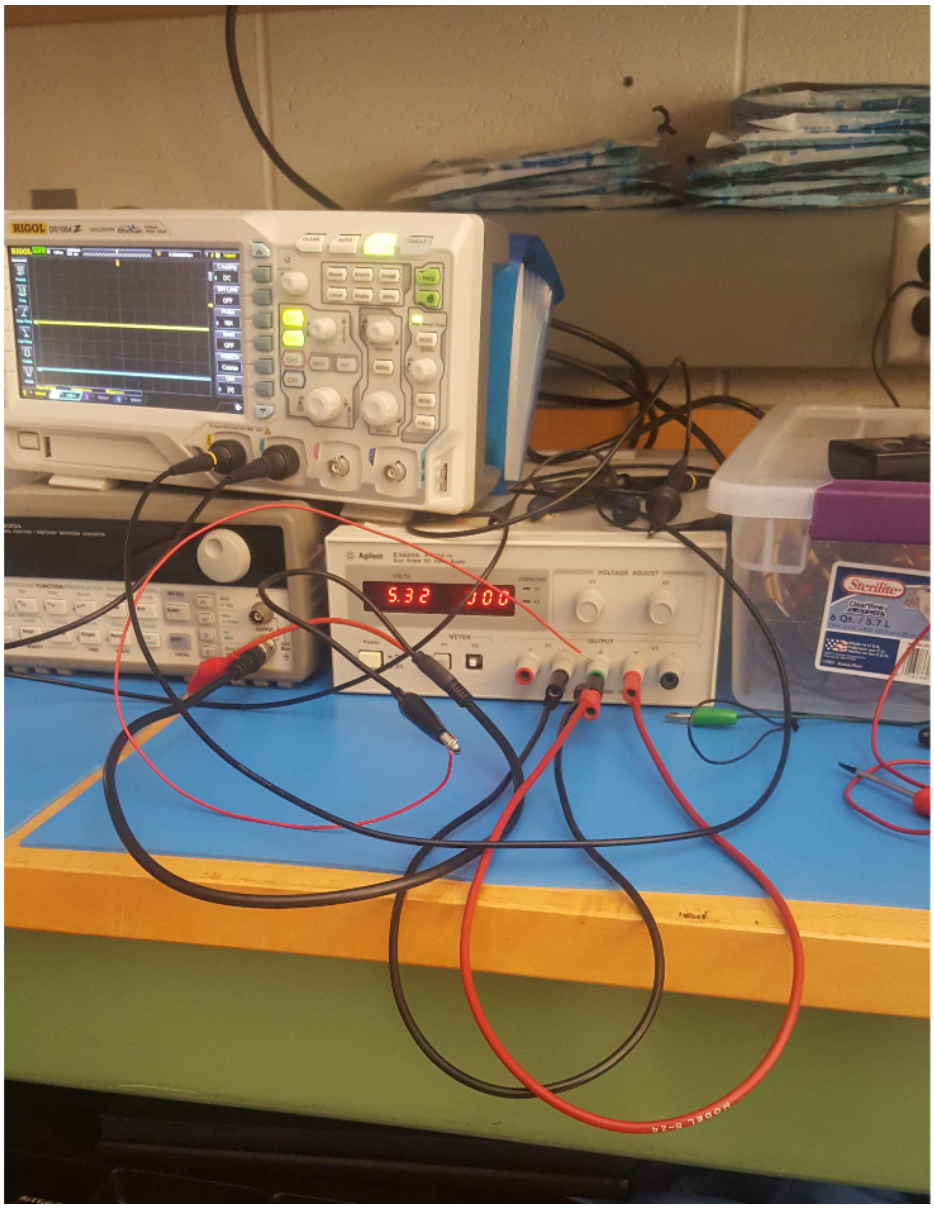


Figure 6.8: Official reference image for the amp power supply

6.3.3 Measuring Output

As mentioned before, output would be measured on an oscilloscope. Amplitude of input and output could be easily compared by human eyes and the next channel would be connected. This was a slow process that involved manyally re-attaching input and output alligator clips. If gain or offset needed to be tweaked, the tester was to turn their respective potentiometers and observe output. The move to 10 ohm potentiometers greatly decreased their effect on the circuit at all and ended up being more of a nuisance than a useful control signal.

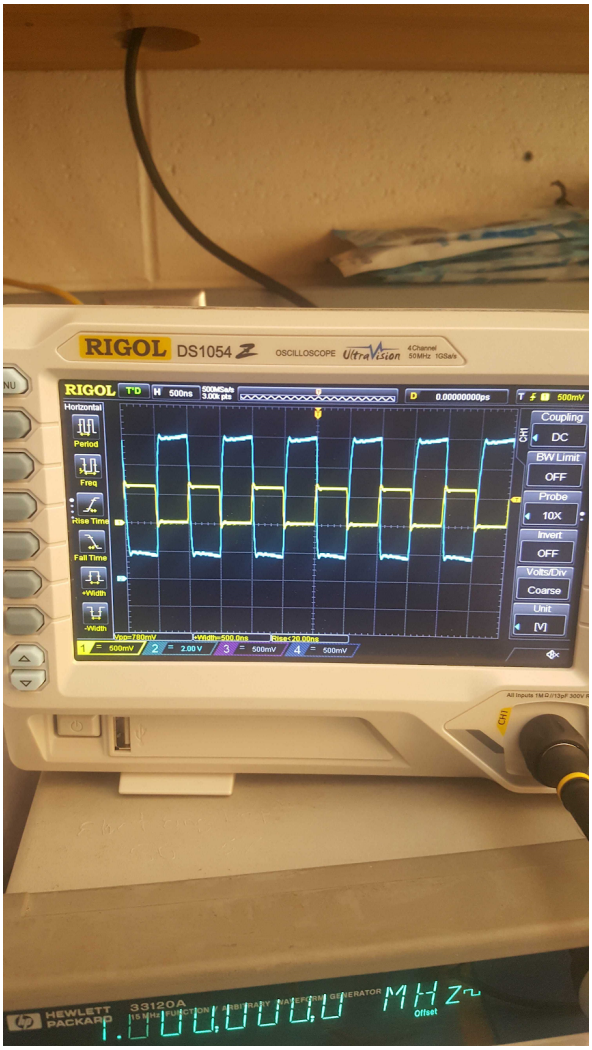


Figure 6.9: Official example of oscilloscope output. Yellow=input, blue=output

6.4 Lessons Learned

In conclusion, learning and using the amplifier rework and test procedures was a difficult task that left far too much room for interpretation or error. The amplifiers were crucial to the system's integrity but were not thoroughly tested or maintained. The process was slow, tedious and quite disconnected from the functionality of the system. Poor documentation meant that only a few group members truly understood the purpose and functionality of all steps involved. Swapping cards under test required disconnecting the card from its breakout boards that were already precariously grasped by alligator and probe clips from the instruments. The breakout board pins were not even labled, so it was difficult to determine if issues were due to the user or hardware. This procedure may have filled its need when first conceived, but proved to be unsustainable as the project scaled up. A careful look was required to understand what needed to be tested and what could be improved.

Chapter 7

DEVELOPMENT OF A FORMAL TESTING PROCEDURE

7.1 Initial considerations

The new procedure was desgined as a direct response to the previous one. Based on complaints and difficulties from all who had to use the setup, we compiled a list of the biggest concerns and focused the procedure design around them. These included:

- Manually changing input/output channels was slow and introduced too many steps
- Input & output were connected through a flimsy probe clip
- No way to separate amp power rails
- No indication if the amps were receiving sufficient power
- No data collection for comparison between cards
- No standards or documentation for gain/offset potentiometer tweaks

7.2 Design Choices

The first readily apparent issue lay in the fact that there was a great deal of manual setup being performed by testers which was not ideal for long-term or large-scale testing. Amp cards were delivered in relatively small batches and had to be reworked and tested as they came. Having to search for a procedure that was hard to understand was not a good solution. I opted to use Python drivers communicating over the two serial specifications (USB serial and VX11 over Ethernet) to automate the input generation and output measurement. By having direct software access to the wave being generated and the data collection, users were in control of a closed loop from beginning to end. The remainder of the issues remained in the lack of hardware support for testing. Input and output locations were unclear and pin placement was

only known to those who created the boards, unless testers wanted to open the PCB files and check themselves. By designing a pair of new PCBs to solve this issue, the test station had a clear set of instructions and devices to measure and capture data as was needed.

7.3 Software Control

Controlling the instruments available in our lab took a bit of research. While other group members had used some of the automation tools before, this was one of the first fully scripted hardware test setups in the lab. The communication protocols and code libraries required are explored below.

7.3.1 pySerial Overview

PySerial is a simple Python library to build and push serial commands over a variety of protocols. This library allows programmers to script serial access and build their own commands to be used for any device they require. For this application, I needed to build commands for a function generator. Other members of the group had previously automated this device and were able to provide documentation on the serial commands necessary. The benefit of a simple serial protocol was that it required no proprietary software and could be modified or extended for other devices. Initiating communication with a serial device is simple with pySerial as the programmer can simply can all serial ports and view device properties. In this specific application the serial device was the only one connected to the host computer, which allowed the device detection to be quite straightforward:

```
ports = list(serial.tools.list_ports.comports())
for port, desc, hwid in sorted(ports):
  print("{}: {} [{}]".format(port, desc, hwid))
for p in ports:
  print("USB Serial: {}".format(p.description))
  if "USB-SERIAL" in p.description:
      print("Found the FeelTech AWG")
```

```
        awg = feeltech_awg.FeelTechAWG(p[0])
assert awg is not None, "No AWG device found"
```

To understand the process of communicating with this device I traced example commands to their lowest levels. To command the device to do anything, requests first had to be packaged into a recognizable prefix. To generate a prefix, parameters must be packaged in a particular order:

```
assert rw in range(2), "R/W value for prefix must be 0 (read
 ) or 1 (write)"
prefix_rw = "W" if rw == 1 else "R"
assert channel in (1, 2), "Invalid channel, must be set to 1
  (main) or 2 (auxilary)."
prefix_channel = "M" if channel == 1 else "F"
valid_modifiers = ("W", "F", "A", "O", "D", "P", "N")
assert modifier in valid_modifiers, "Invalid cmd modifier (%
 s), must be in %s"%(modifier, valid_modifiers)
return prefix_rw + prefix_channel + modifier
```

This command prefix is then concatenated with a desired parameter based on the function being controlled. Turning on an output channel, for example, takes in an output flag and attaches it to the command prefix:

```
"""
output:  int -> 0 to turn channel off, 1 to turn channel on
"""
cmd = _generate_cmd_prefix(1, channel, "N") + str(output)
```

The command is now ready to be sent to the device. Using pySerial functions for read/write operations, the code is simple:

```
self.device.flushInput()
self.device.write((cmd_str + "\n").encode())
self.device.flush()
```

The straightforward nature of serial communications made this easy to debug and test. The function generator only needed to perform a few tasks (turn channels on/off,

set waveform parameters) the scope of my research into its functionality was limited. I believe, however, that this establishes a good base for future testing development. Many other instruments use specifications such as VISA which may be a better choice in the future.

7.3.2 VXI11 Overview

The VXI11 Communication Protocol is an architecture based on the VMEbus architecture meant for electronic communication. VXI11 in particular was built as an instrument communication spec [11]. VXI11 requires three channels and operates over Ethernet.

VXI11 Python libraries are available and make VXI communication simple. Similarly to serial, commands must be constructed by the programmer and packaged into strings sent over the VXI bus. For example, writing to the device makes a device-level system call to write the input data to a predetermined location. From the programmer's point of view, one only has to use the high-level functions to facilitate communication. There are, for example, functions for read(), write(), and ask(), which performs a write and then read. These functions are grouped and abstracted for use with this specific instrument. If, for example, the programmer wants to ask the device what a selected channel's probe ratio is, the code is as follows:

```
1  def get_probe_ratio(self, channel):
2    """
3    Returns the probe ratio for a specific channel
4    """
5    channel = self._interpret_channel(channel)
6    return float(self.query(':{0}:PROBe?'.format(channel)))
```

The string being pushed in the query function is a device-specific string that is outlined by the manufacturer [10]. Using the listed commands and an ethernet connection, the VXI11 bus will automatically detect when a valid device is connected to the local network through Ethernet. The IP address of the device is specified upon initialization and needs to be checked before running tests.

7.4 PCB Design Considerations

While only a limited amount of hardware was required to drive the setup, design considerations had to be made to ensure there were no issues with single integrity or connector placement. The main goals of the test PCBs were to:

- Provide users with a clear start and end point for signals
- Clearly label all relevant components
- Minimize manual steps to reduce error
- Provide instructions directly on the boards as well as official documentation

To accomplish this a thorough re-examination of the available equipment and possible solutions was required. To make the most of the previously mentioned measurement tools, simple shielded connectors were chosen as the entry and exit points for all signals. While these were not strictly necessary, standardizing connectors and overcompensating for signal integrity create stable foundations for future testing procedures.

Visual aids such as text and indicator displays were also necessary. It is impractical for a user to physically ensure power is being delivered when an active component could always sense voltage on its line. With new instructions, procedure and documentation, the visual aids were added purely as concessions for accessibility.

The final factor in the PCB design was proper electrical structure. The amplifier architecture provided separate ground and power lines for each amplifier component which housed four amplifiers each. In the system architecture, these ground planes and power lines were provided by the interface board (signaloutput) and DAC cards (input). Each of these ends unified the grounds. To mimic this architecture ground planes were added on both sides of the test setup to eliminate any risk of cross-talk or inaccurate results. The new testing hardware platform was built from the ground up as a representation of what the amps would be expected to do in the system.

7.5 Emulating DAC input

DAC output signals had previously been investigated with the purposes of limiting variations between analog response of individual channels. A long bout of experimentation yielded a list of voltage outputs from each DAC at different colors. As

mentioned before, each "color" band represents a range of values to represent desired brighness or intensity. These voltage levels provided a starting point for the voltage sweeps necessary to fully validate the amp cards. Results are recorded below:

Table 7.1: DACIN Value at Target Pixel Voltage

Voltage Level (V)	**DACIN Value (Color Counts)**
5	32767
4	26214
3	19660
2	13106
1	9000

Table 7.2: Voltage at DACIN=32767

Signal	**Channel**	**Min Voltage (V)**	**Max Voltage (V)**
vinn0	0	0.198	5.1
vinp0	1	0.200	5.16
vinn1	2	0.080	5.1
vinp1	3	0.139	5.12
vinn2	4	0.170	5.11
vinp2	5	0.142	5.08
vinn3	6	0.162	5.09
vinp3	7	0.140	5.07
vinn4	8	0.137	5.06
vinp4	9	0.206	5.1
vinn5	10	0.187	5.08
vinp5	11	0.140	5.07
vinn6	12	0.033	5.01
vinp6	13	0.063	5.02
vinn7	14	0.067	5.01
vinp7	15	0.087	5

Table 7.3: Voltage at DACIN=26214

Signal	**Channel**	**Min Voltage (V)**	**Max Voltage (V)**
vinn0	0	0.181	4.16
vinp0	1	0.185	4.21
vinn1	2	0.066	4.1
vinp1	3	0.120	4.17
vinn2	4	0.160	4.15
vinp2	5	0.130	4.11
vinn3	6	0.153	4.12
vinp3	7	0.133	4.1
vinn4	8	0.120	4.1
vinp4	9	0.196	4.16
vinn5	10	0.181	4.11
vinp5	11	0.130	4.1
vinn6	12	0.040	4.03
vinp6	13	0.060	4.04
vinn7	14	0.060	4.03
vinp7	15	0.070	4.02

Table 7.4: Voltage at DACIN=19660

Signal	**Channel**	**Min Voltage (V)**	**Max Voltage (V)**
vinn0	0	0.180	3.16
vinp0	1	0.186	3.19
vinn1	2	0.065	3.11
vinp1	3	0.122	3.15
vinn2	4	0.160	3.12
vinp2	5	0.126	3.11
vinn3	6	0.159	3.14
vinp3	7	0.129	3.11
vinn4	8	0.120	3.11
vinp4	9	0.187	3.15
vinn5	10	0.180	3.14
vinp5	11	0.133	3.11
vinn6	12	0.039	3.04
vinp6	13	0.060	3.06
vinn7	14	0.060	3.05
vinp7	15	0.075	3.04

Table 7.5: Voltage at DACIN=13106

Signal	**Channel**	**Min Voltage (V)**	**Max Voltage (V)**
vinn0	0	0.184	2.18
vinp0	1	0.188	2.2
vinn1	2	0.065	2.11
vinp1	3	0.122	2.15
vinn2	4	0.159	2.16
vinp2	5	0.126	2.14
vinn3	6	0.157	2.15
vinp3	7	0.128	2.15
vinn4	8	0.120	2.12
vinp4	9	0.191	2.18
vinn5	10	0.180	2.16
vinp5	11	0.135	2.13
vinn6	12	0.039	2.06
vinp6	13	0.060	2.07
vinn7	14	0.060	2.07
vinp7	15	0.066	2.07

Table 7.6: Voltage at DACIN=9000

Signal	**Channel**	**Min Voltage (V)**	**Max Voltage (V)**
vinn0	0	0.175	1.53
vinp0	1	0.175	1.54
vinn1	2	0.046	1.44
vinp1	3	0.100	1.49
vinn2	4	0.135	1.5
vinp2	5	0.103	1.48
vinn3	6	0.135	1.5
vinp3	7	0.105	1.48
vinn4	8	0.096	1.47
vinp4	9	0.185	1.53
vinn5	10	0.165	1.51
vinp5	11	0.110	1.47
vinn6	12	0.013	1.4
vinp6	13	0.036	1.41
vinn7	14	0.036	1.41
vinp7	15	0.049	1.41

Chapter 8

PROCEDURE DESIGN METHODOLOGY

8.1 Input Stage

The input stage of the amp testing procedure was the first to be targeted. Directing input to the amp card should be a simple task for the user. Since all four channels are used in parallel on the system, they can be connected to a unified signal source and displayed together on the oscilloscope for data collection. Using a mix of hardware and software support I developed an input procedure for the new test.

8.1.1 Function Generation

To implement a Python layer for the function generator I opted to use a FeelTech FY6900 Arbitrary Waveform Generator. This was a relatively cheap function generator

Figure 8.1: FeelTech FY6900

that fulfilled the tasks we needed to accomplish. With up to 60MHz sine wave output,

this device was more than enough to test the capabilities of the amplifiers. The DAC input the amps were designed to process were at frequencies far below this. For the first versions of the test I opted to use the original function generator settings until more could be determined about the input parameters. The code was simple and based on the previously listed libraries & protocols:

```
awg.set_on_off(1, 0)
awg.set_on_off(2, 0)

awg.set_waveform(1, 1)
awg.set_frequency(1, 1000000)
awg.set_amplitude(1, x[i])
awg.set_offset(1, 0.318)
awg.set_phase(1, 0)

# Turn channel 1 on
awg.set_on_off(1, 1)
```

For the automated test to save time, it was set to loop four times to collect plot data from the oscilloscope for all four amplifier output channels. In order for this automatic switching to work, hardware had to be designed to accept the signal and distribute it accordingly.

8.1.2 Bottom (Input) PCB

Major flaws in the original test setup included the lack of any shielded cable to transfer the input signal and the tester having to manually clip the input to each amp channel. Although we were not necessarily testing for strict signal integrity, it is better practice to use standard connectors for input rather than probe clips. Since the FeelTech generator used a BNC connector it was easiest to match this on the input card. BNC connectors are shielded coaxial connectors capable of reliably passing signals under 4GHz and less than 500V [8].

Separating the amplifier's input lines was a useless step that only served to complicate the process. To solve this I had to examine the amp's PCB layout to find

the relevant pins. The DAC inputs were differential pairs identified within the layout below: The input side of the board was fairly simple to draw and lay out. One of the

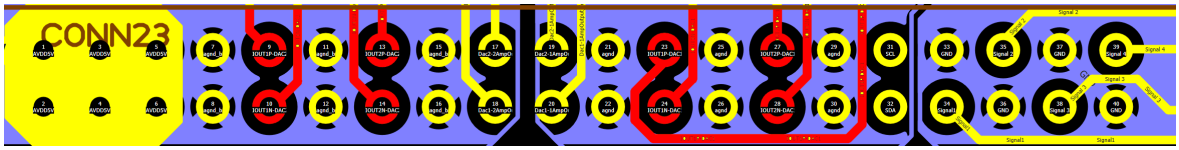


Figure 8.2: Differential input pins on the Amp PCB

amp's purposes was to turn the DAC's differential output into a single-ended output. To accomplish this, the amp card terminated the N-terminals of all incoming DAC signals through a 49 ohm resistor. Grounding the differential input was an acceptable

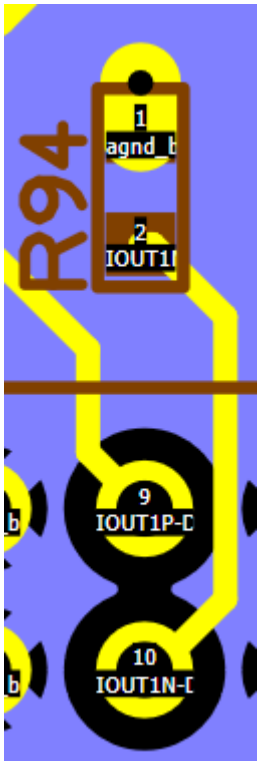


Figure 8.3: Terminating resistor for the N-terminal of the DAC output.

decision for the test setup. Only the P-lines of the signal would be processed by the circuit at all, so they were routed together and joined to a single BNC input in the test board. The layout followed shortly and included a ground plane to maximize signal

integrity. With such a limited number of relevant pins on the input side of the amp card, designing the test board was not an issue.

8.1.3 Input Conclusion

This section of the test setup resulted in a far simpler procedure. Having to only connect one output through a stable connection and clearly labeled board eliminated all possible confusion for testers.

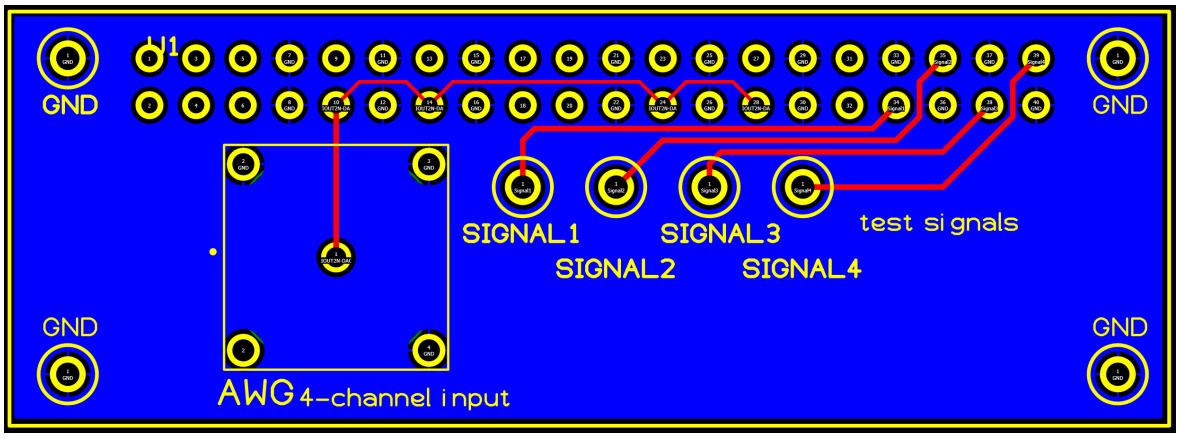


Figure 8.4: Input Test board layout

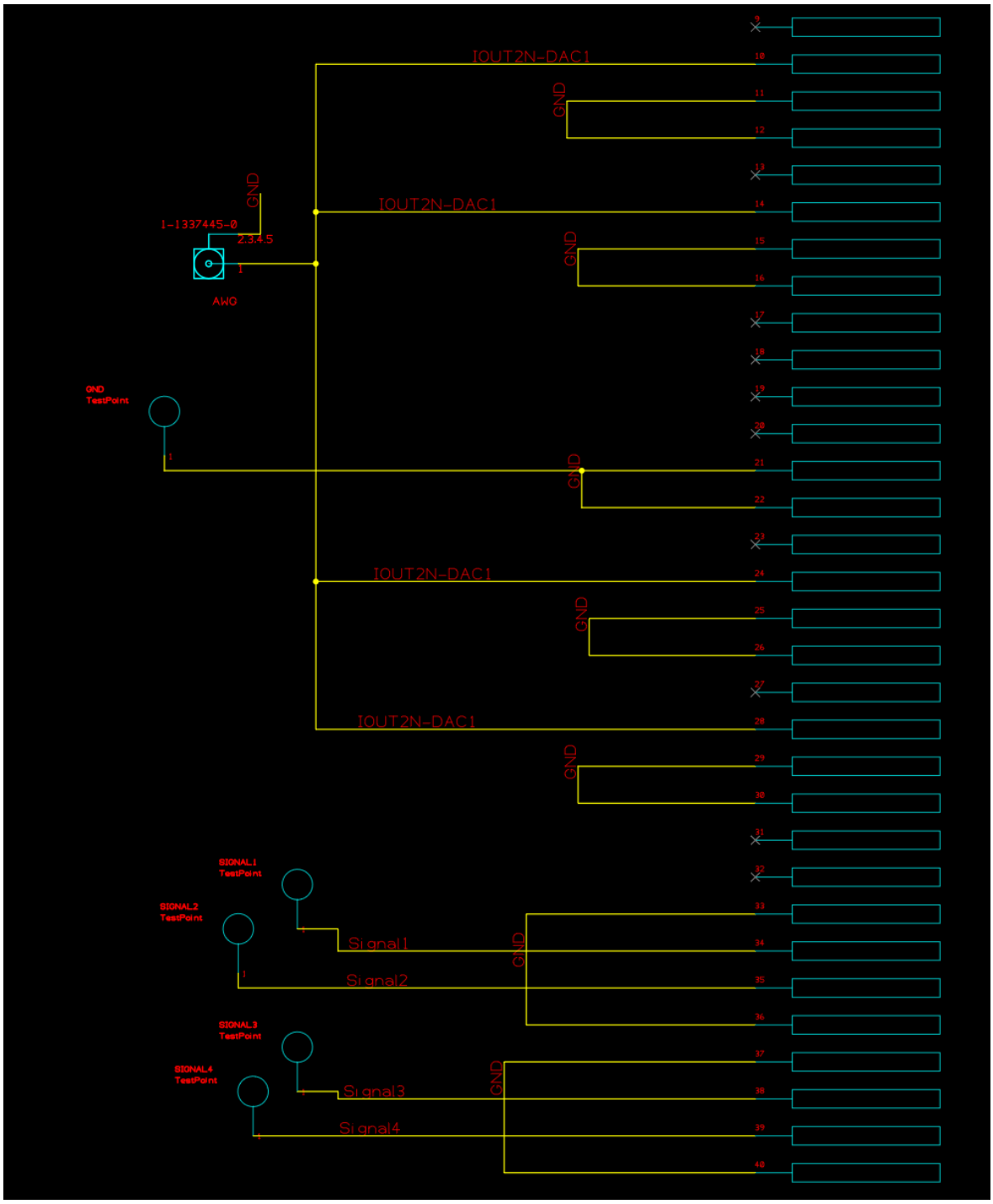


Figure 8.5: Input Test board schematic for joining P-inputs and grounding N-inputs

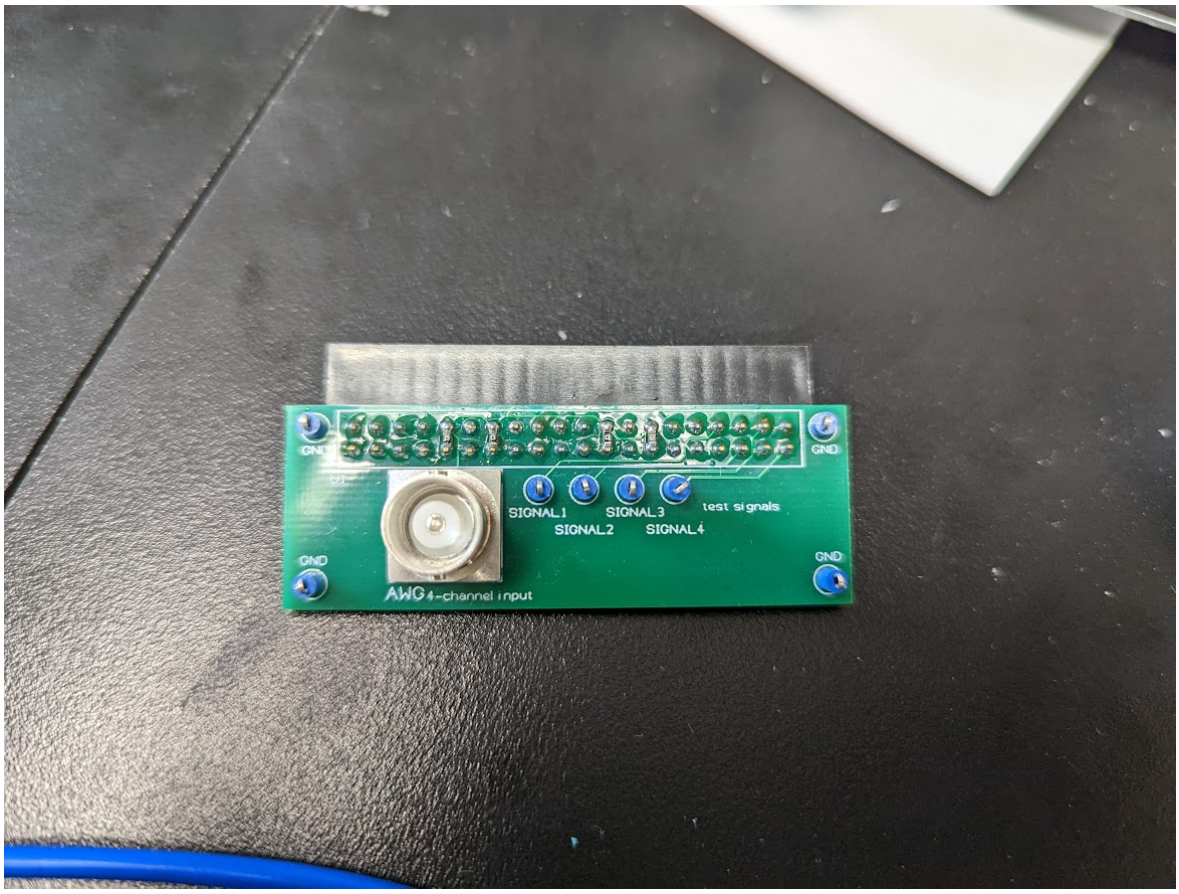


Figure 8.6: Final fabricated bottom test board

Figure 8.7: Oscilloscope with desired settings programmed

8.2 Output Stage

The output of the amp card required the testers to be able to view all output channels at once as well as collect and plot the oscilloscope's data. This once again required a mix of software and hardware tools to accomplish.

8.2.1 Top (Output) PCB

Before any data could be routed to the oscilloscope we needed to develop a control system for the top connector of the amp card. This connector housed a number of vital pins including power rail input and amplifier output. In the system, this connector would be attached to the interface board and transport signals through the ribbon cables. To appropriately test the functionality of all amp cards, users had to be able to fulfill a number of requirements:

- Ensure power was being properly delivered to both amplifiers
- Have the ability to independently toggle power to each amplifier unit
- Route output through shielded cable to oscilloscope

Each section of the PCB was therefore designed to address these issues. To indicate valid power delivery a simple LED indicator system was sufficient. Each amplifier had its own pin for power delivery and as such could be routed independently. A switch with three positions was used to control power delivery. When in neutral, power would be routed to both amplifiers A and B. When flipped left or right, the switch would direct power into only amplifier A or B respectively. The LED indicators were placed into voltage dividers. If enough voltage was being delivered on that line, the divider would pass voltage to light up the LED.

When moving on to the layout the main concern was for space and signal integrity. By keeping the board as symmetrical as possible and including another ground plane, the design was ensured to be safe from interference. The four output channels were available as BNC headers as well and could be directly connected to a 4-channel oscilloscope. The switch and LED components provided the most challenges in layout as they required quite a few traces. The final layout was compact and clearly labeled to show testers what they needed to use. Power delivery was accomplished through

probe clips to the power supply. The power supply is never changed or adjusted during the whole process, so the clips were not any issue when detatching and reattaching cards from the testing boards.

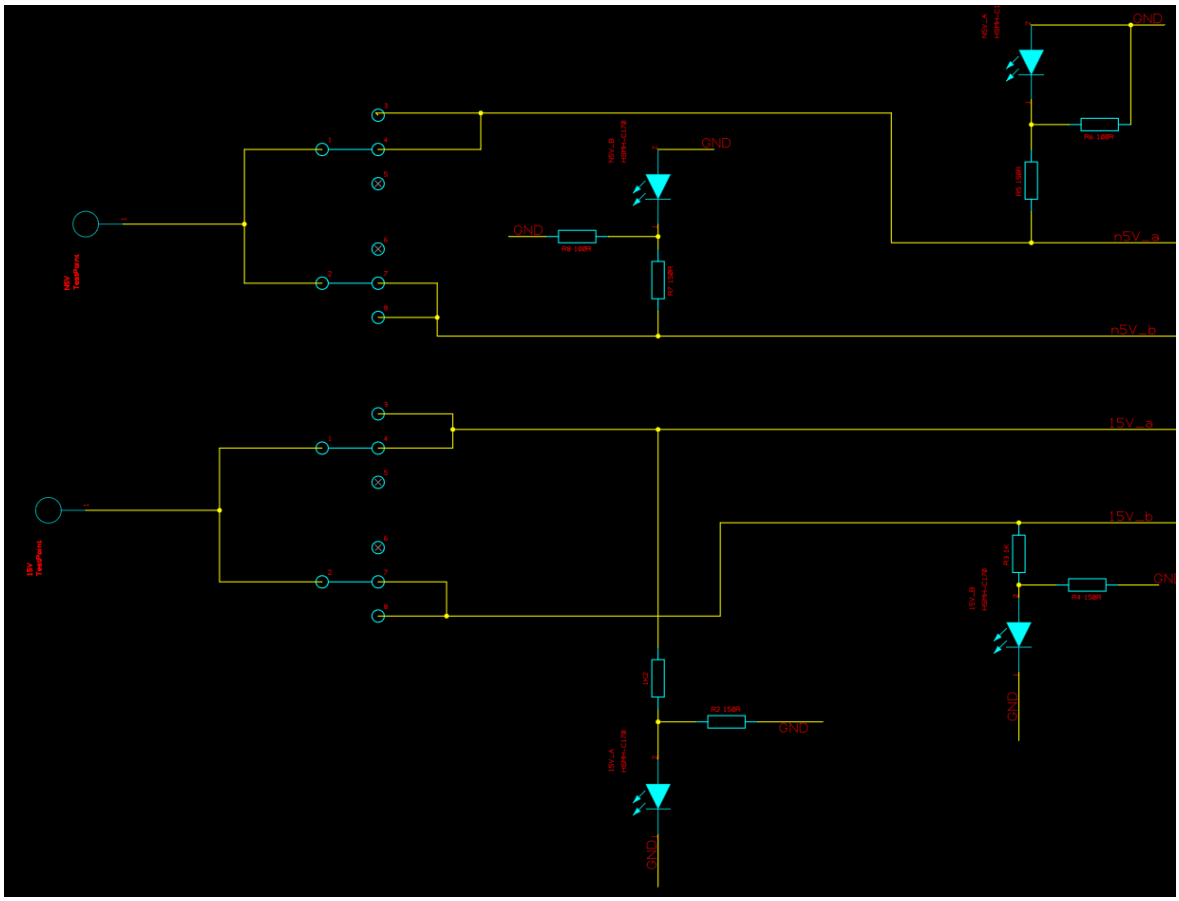


Figure 8.8: Power indicator & switching circuits for the -5V and +15V lines

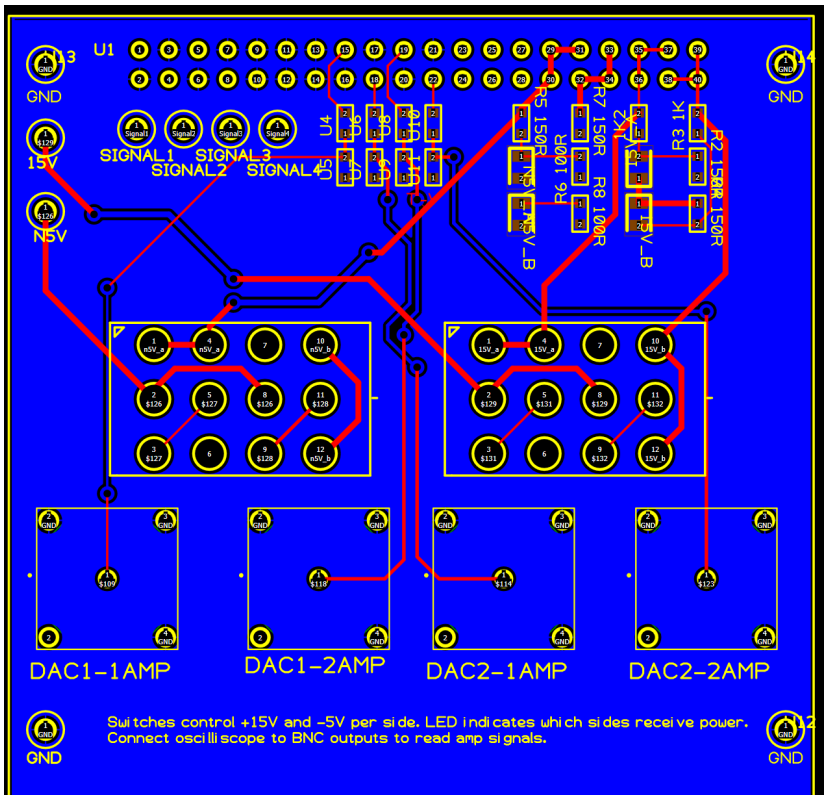


Figure 8.9: Finished layout of Top testing board

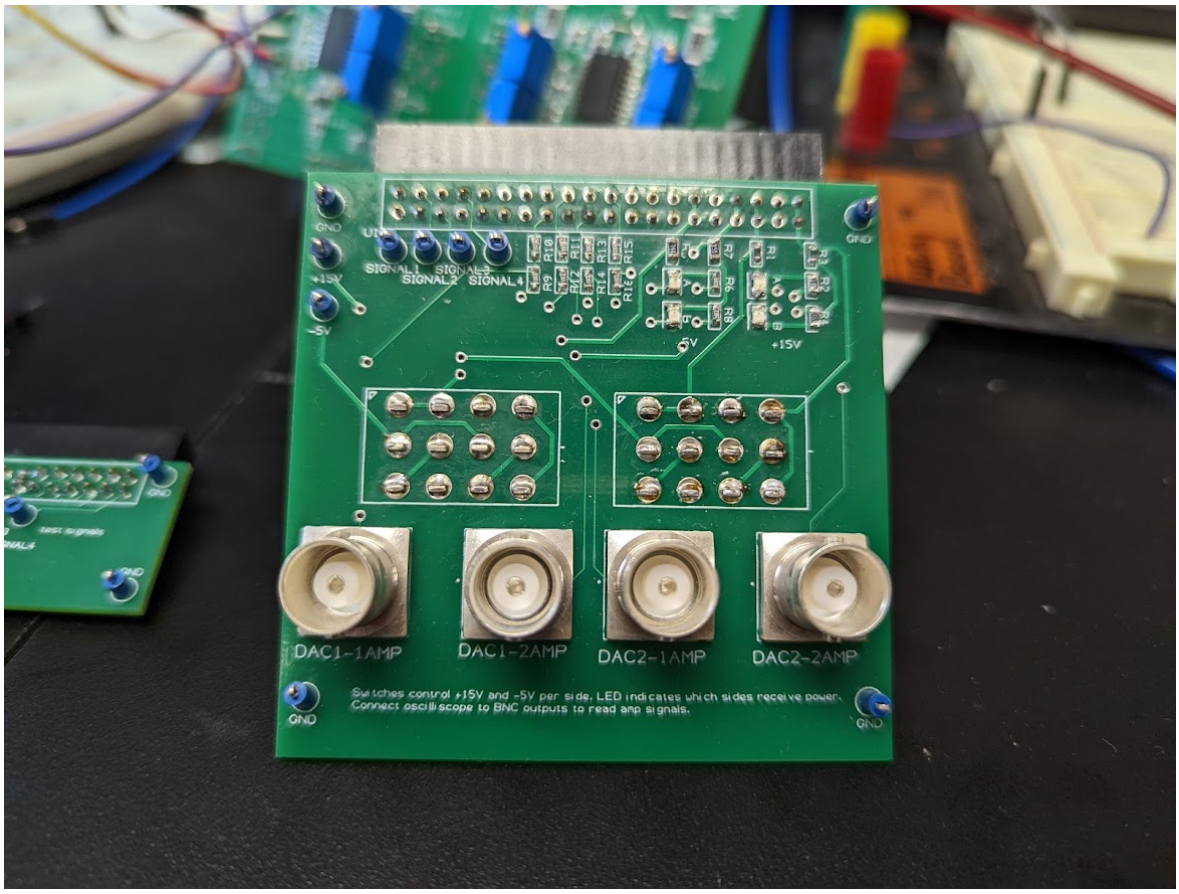
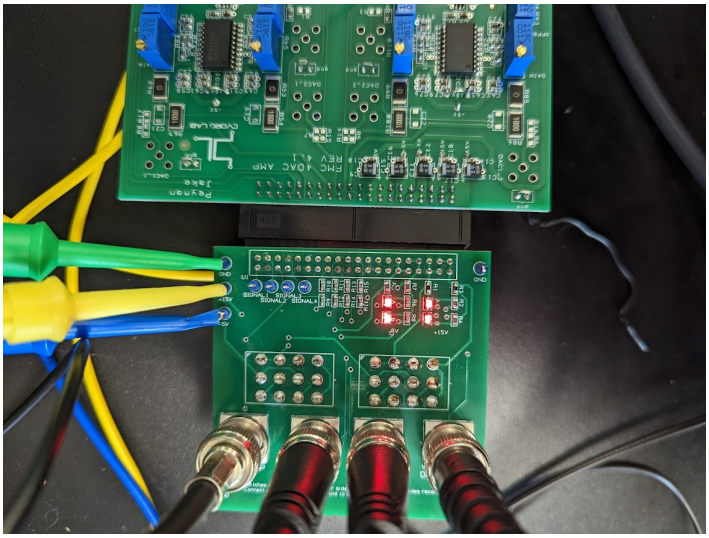
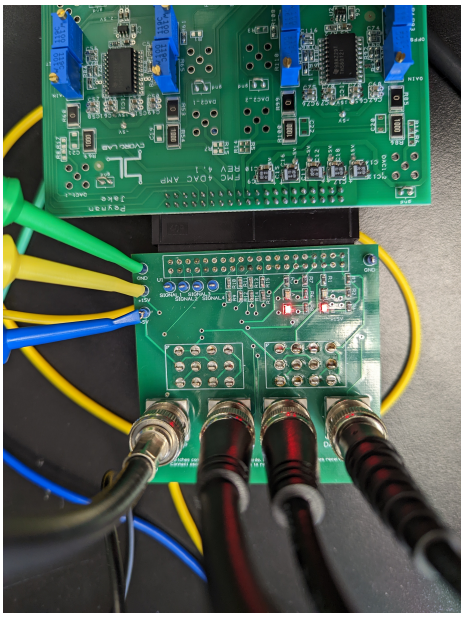


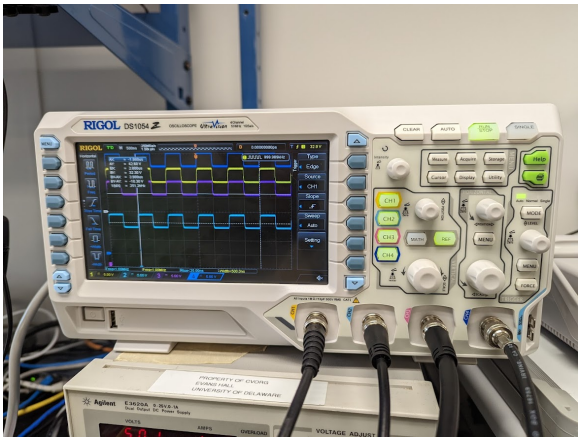
Figure 8.10: Final fabricated Top test board



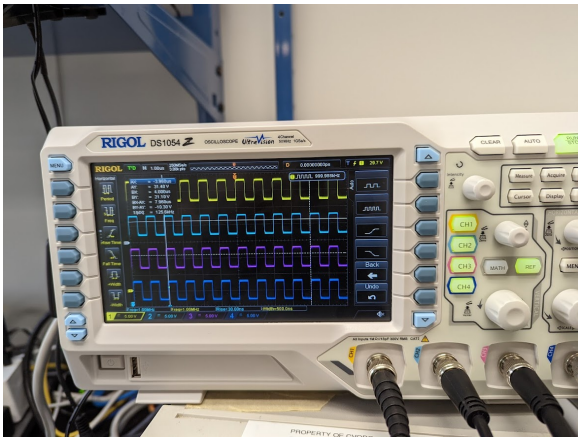
(a) Power LED indicators



(b) Only Amplifier A is on



(a) Before programming, scope channels are not aligned



(b) After programming, all signals are visible and ready for capture

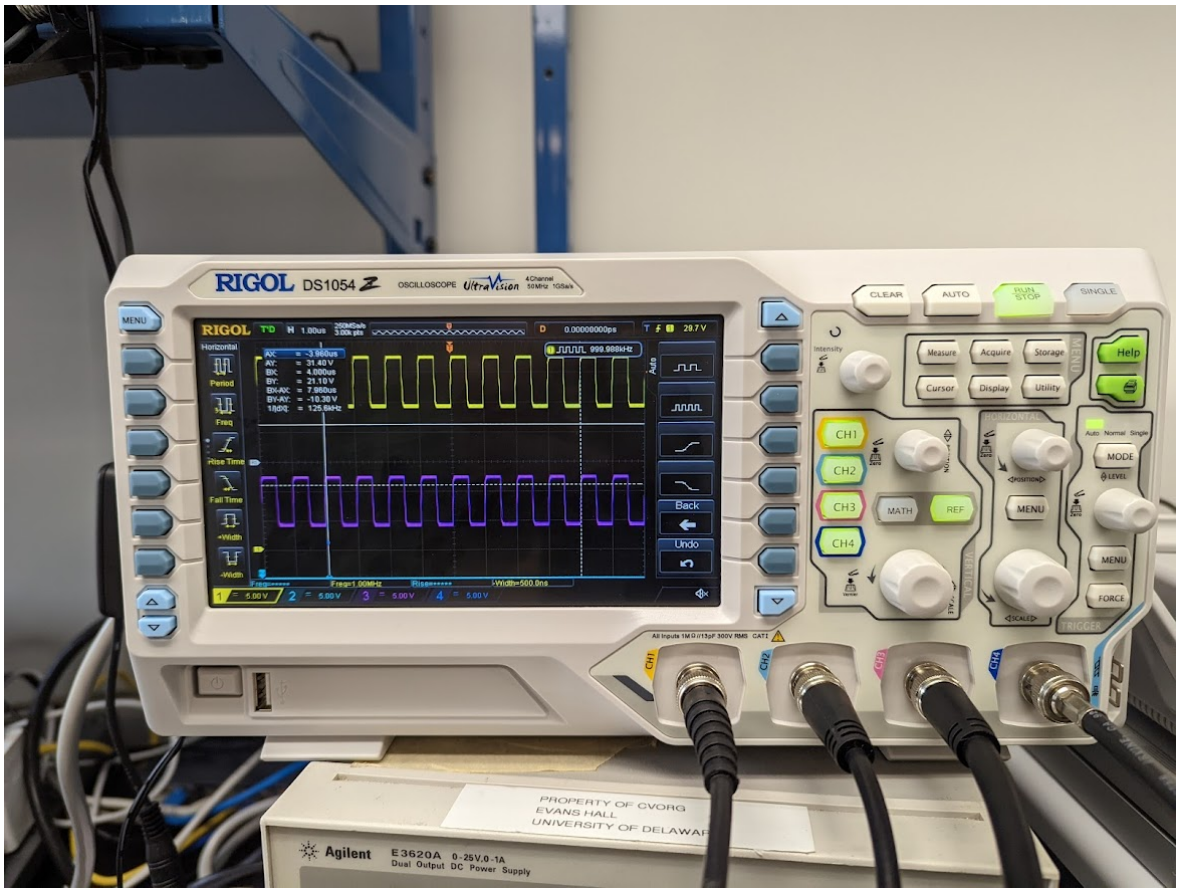


Figure 8.13: Scope output when only Side A is powered

8.3 Results

In order to reflect DACIN values I created arrays of amplitude values to be used by the AWG. By sweeping across the desired input range we could quickly gather data on the amplifiers without having to manually adjust inputs. The system operates with discrete sets of analog outputs and as such the measurements would be easy to gather within a predetermined window of time. The oscilloscope scale was fully programmable and allowed us to increase or decrease the number of samples on the screen. Boundaries were set according to the range of DACIN values. Testers could set the number of iterations or steps to increase the precision of a test. The first iterations of the procedure only used 5 steps, but further testing and development of the software allowed as many steps as desired.

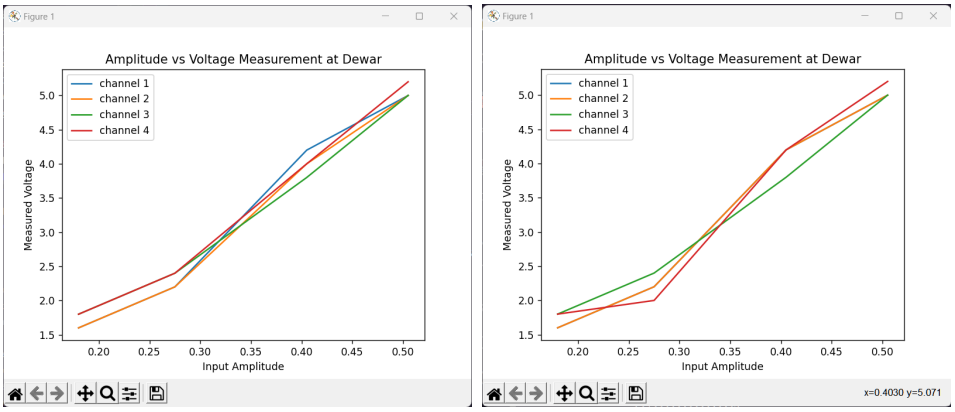
```
steps = 50
dacin_range = [1, 5.2]
x1 = [0] * steps
y1_1 = [0] * steps
y2_1 = [0] * steps
y3_1 = [0] * steps
y4_1 = [0] * steps
dacin_sweep([1, 5.2], steps, x1, y1_1, y2_1, y3_1, y4_1, awg
  , scope)
plt.plot(x1, y1_1, label="Channel 1")
plt.plot(x1, y2_1, label="Channel 2")
plt.plot(x1, y3_1, label="Channel 3")
plt.plot(x1, y4_1, label="Channel 4")
plt.xlabel("Input Amplitude")
plt.ylabel("Measured Voltage")
plt.title("Input Amplitude (V) vs Amplified Output (V)")
plt.legend()
# plt.show()
filename = num + ".png"
plt.savefig(filename)
plt.show()
```

```
def dacin_sweep(dacin_bounds, steps, x, y1, y2, y3, y4, awg,
  scope):
# turn off AWG
awg.set_on_off(1, 0)
awg.set_on_off(2, 0)
# set scope parameters & turn on displays
scope.display_channel("CHAN1", True)
scope.set_probe_ratio("CHAN1", 10)
scope.set_channel_scale("CHAN1", 10, False)
scope.set_channel_offset("CHAN1", -5)

scope.display_channel("CHAN2", True)
scope.set_probe_ratio("CHAN2", 10)
scope.set_channel_scale("CHAN2", 10, False)
scope.set_channel_offset("CHAN2", -10)

scope.display_channel("CHAN3", True)
scope.set_probe_ratio("CHAN3", 10)
scope.set_channel_scale("CHAN3", 10, False)
scope.set_channel_offset("CHAN3", -15)

scope.display_channel("CHAN4", True)
scope.set_probe_ratio("CHAN4", 10)
scope.set_channel_scale("CHAN4", 10, False)
scope.set_channel_offset("CHAN4", -20)
step = (dacin_bounds[1] - dacin_bounds[0]) / steps
nextval = dacin_bounds[0]
print(steps, "iterations with voltage step", step, "V")
with alive_bar(steps * 4, force_tty=True) as bar:
    for i in range(steps):
        x[i] = nextval
        nextval = nextval + step
        print(x[i], "V in")
        awg.set_waveform(1, 1)
        awg.set_frequency(1, 1000000)
```

```
35          awg.set_amplitude(1, x[i])
36          awg.set_offset(1, x[i] / 2)
37          awg.set_phase(1, 0)
38          awg.set_on_off(1, 1)
39          chan1 = scope.get_waveform_samples("CHAN1", "NORMal"
    )
40          chan2 = scope.get_waveform_samples("CHAN2", "NORMal"
    )
41          chan3 = scope.get_waveform_samples("CHAN3", "NORMal"
    )
42          chan4 = scope.get_waveform_samples("CHAN4", "NORMal"
    )
43          y1[i] = max(chan1) - min(chan1)
44          bar()
45          # print(y1[i])
46          y2[i] = max(chan2) - min(chan2)
47          bar()
48          # print(y2[i])
49          y3[i] = max(chan3) - min(chan3)
50          bar()
51          # print(y3[i])
52          y4[i] = max(chan4) - min(chan4)
53          bar()
54  # print(y4[i])
```

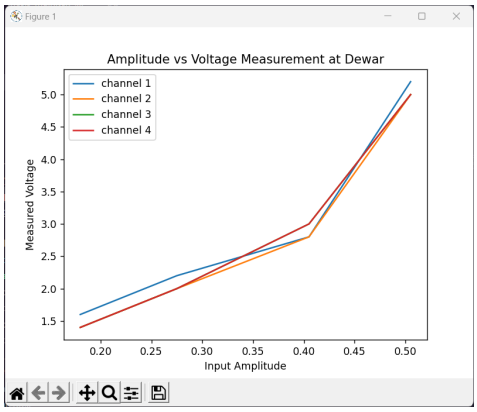
Using this simple script the equipment was automated to set parameters and measure, storing the data in arrays to be plotted with numpy later.

Initial data collection was only performed with five input points and five outputs. While rudimentary, this served as an effective foundation for understanding larger-scale measurements and intensive voltage sweeps. In general, good cards would respond linearly to increasing voltage input. Examples of results from different cards are shown below: The graphs make diagnosing and parsing issues significantly easier. For example, 8.14b shows a card that has somewhat irregular response across channels



(a) Best response

(b) Acceptable response



(c) Needs tuning

Figure 8.14: Results from three different amp sweeps

2 and 4. This can easily be pointed to as an issue with the signals' driver, amplifier B. Identified in this way, the cards can quickly be touched up and re-tested.

8.4 Further Testing

Adding iterations yielded smooth results. By iterating through the voltage bounds with a fine grain the cards are thoroughly tested within all normal voltage levels they will be expected to encounter in the system. The goal of adding more iterations is to approach a continuous measurement that reliably reports the behavior of the circuit at any input. Adding iterations is a useful measurement tool, but comes with the caveat of making the program take much longer to run. The oscilloscope is slow to report back its channel measurements and cannot scale well with multiple

outputs.

8.5 Testing Conclusions

The testing procedure was sufficiently automated and documented at this point. While the procedure certainly left room for improvement, fixing test procedures is an iterative process that requires experience with the existing procedures to improve. A major area of concern was the time taken for testing by increasing iterations. The time lapsed did not appear to scale linearly and adding more iterations would increase the time taken by far more than a simple scalar. To improve on this the oscilloscope must be replaced entirely by an analog or digital device that can perform fast measurements without extra issues. The analysis process is quite straightforward (Plot Vpp over increasing voltage) and can certainly be performed by a faster component. Alternatively, four oscilloscopes could be stacked to measure and report on each channel independently, although this could lead to issues and inconsistencies. The oscilloscope programming is entirely sequential and therefore not ideal for parallel measurements.

This testing model led to multiple developments and discoveries that will be useful when creating and testing the next generation of amplifiers. Using the automation and PCB design strategies outlined here will allow us to implement features such as self-tuning gain and doubling the number of channels.

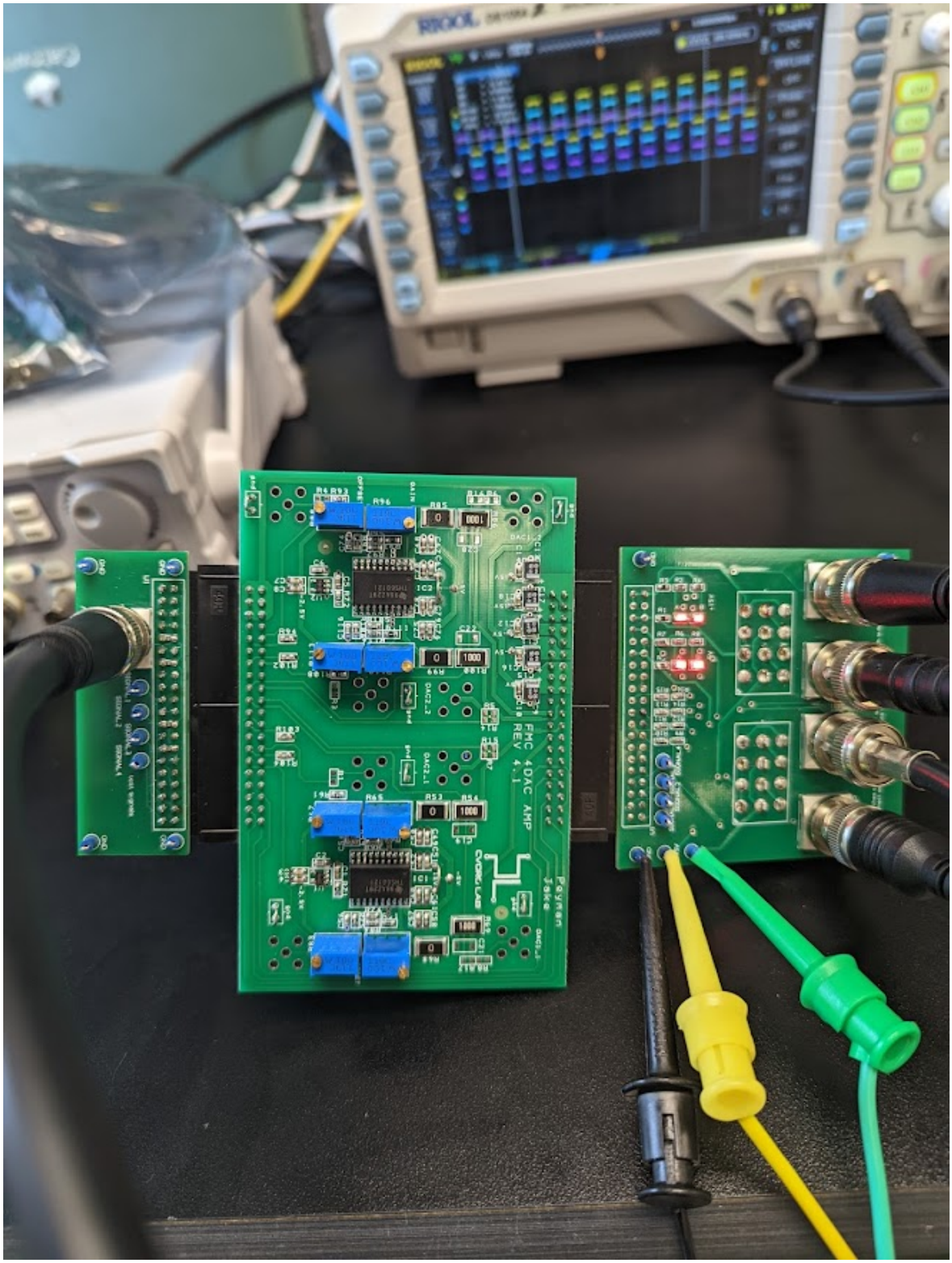


Figure 8.15: Full testing setup powered on

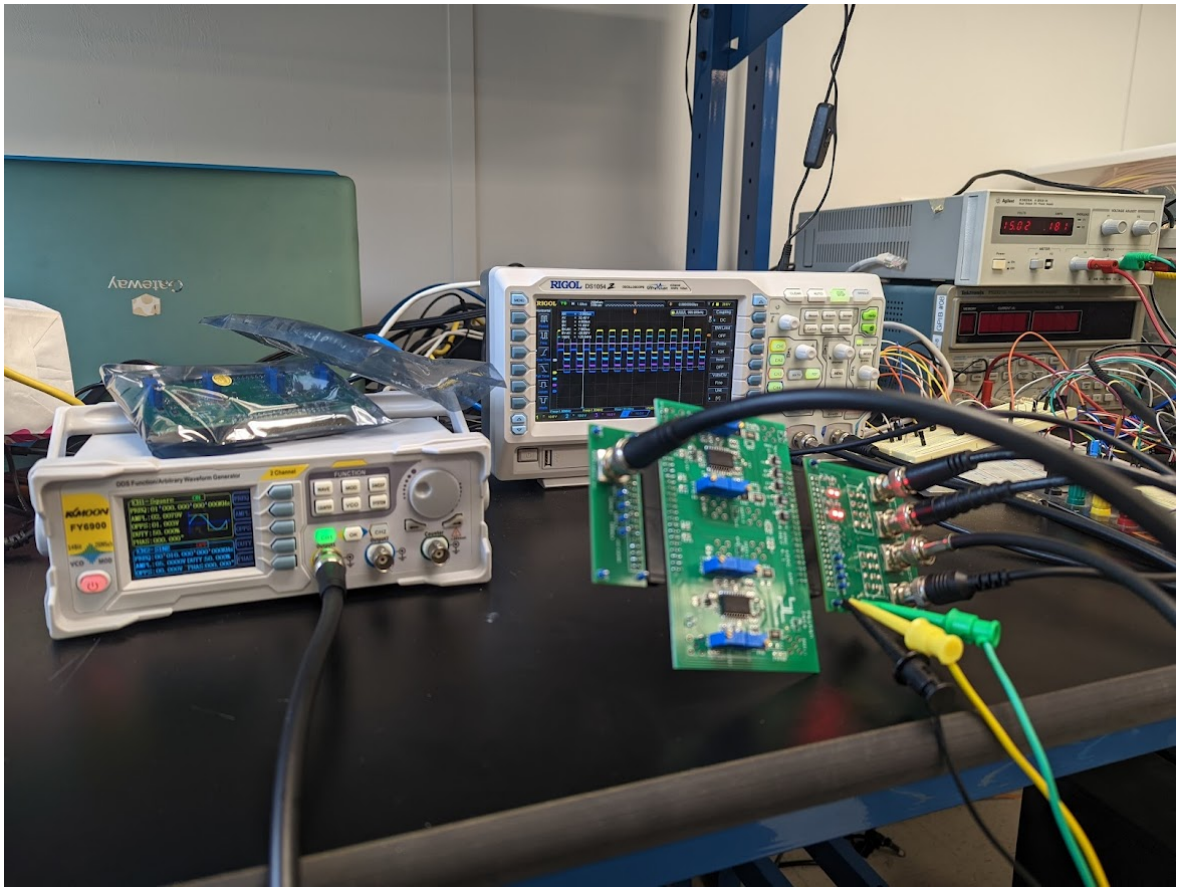
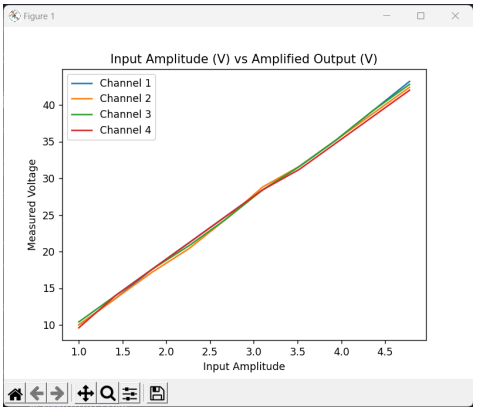


Figure 8.16: Measurements in progress



(a) 10x iterations results on a good card

```
Amp card #: 1-4.1-184
Finding ports:
COM4: USB-SERIAL CH340 (COM4) [USB VID:PID=1A86:7523 SER= LOCATION=1-2]
USB Serial: USB-SERIAL CH340 (COM4)
Found the FeelTech AWG
Connected to:  RIGOL TECHNOLOGIES,DS1054Z,DS1ZA231501834,00.04.04.SP4
Testing card  1-4.1-184
0.42000000000000004
| 40/40 [100%] in 1:29.1 (0.45/s)
```

(b) Terminal output for test run

Figure 8.17: Data collected & terminal output with 10 iterations

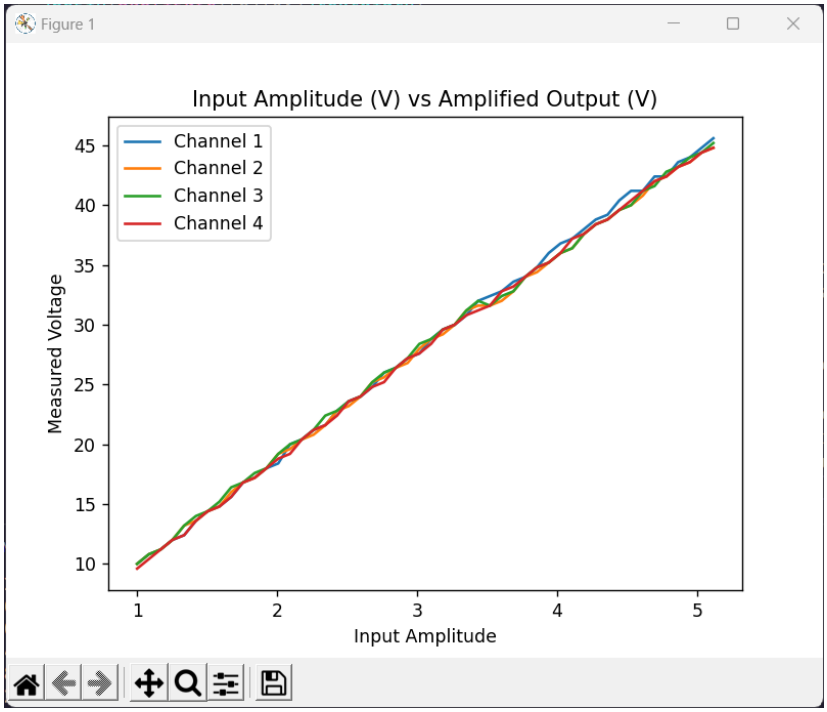


Figure 8.18: Results from 50x iterations

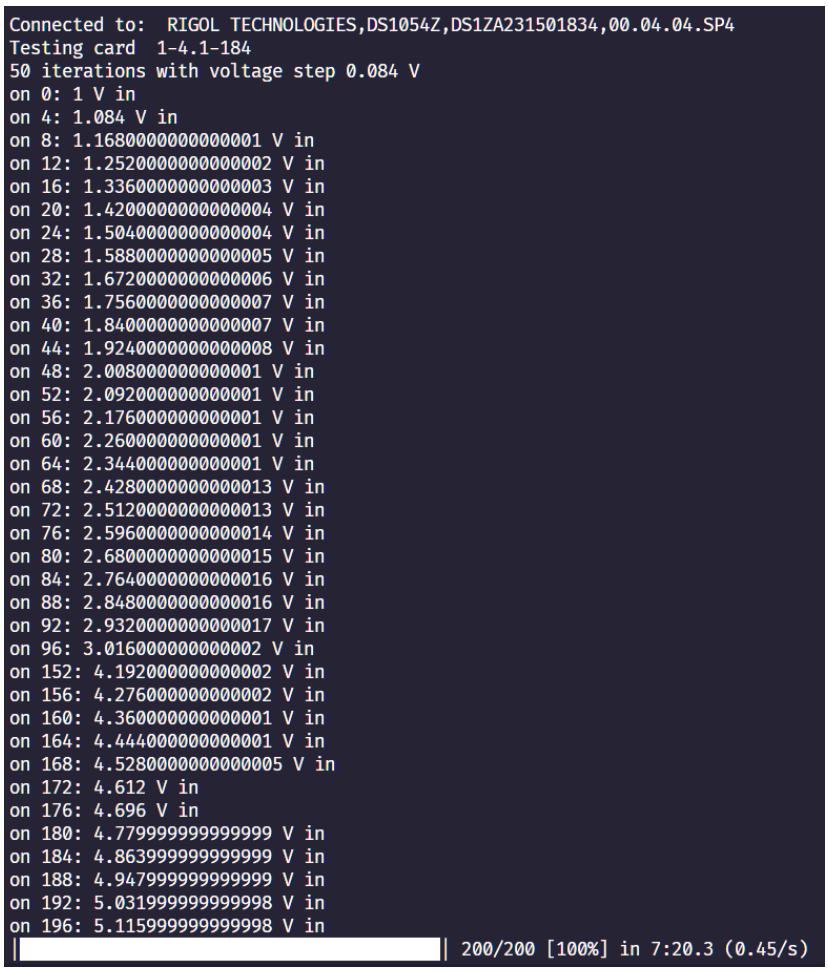


Figure 8.19: Revised terminal output with 50 iterations

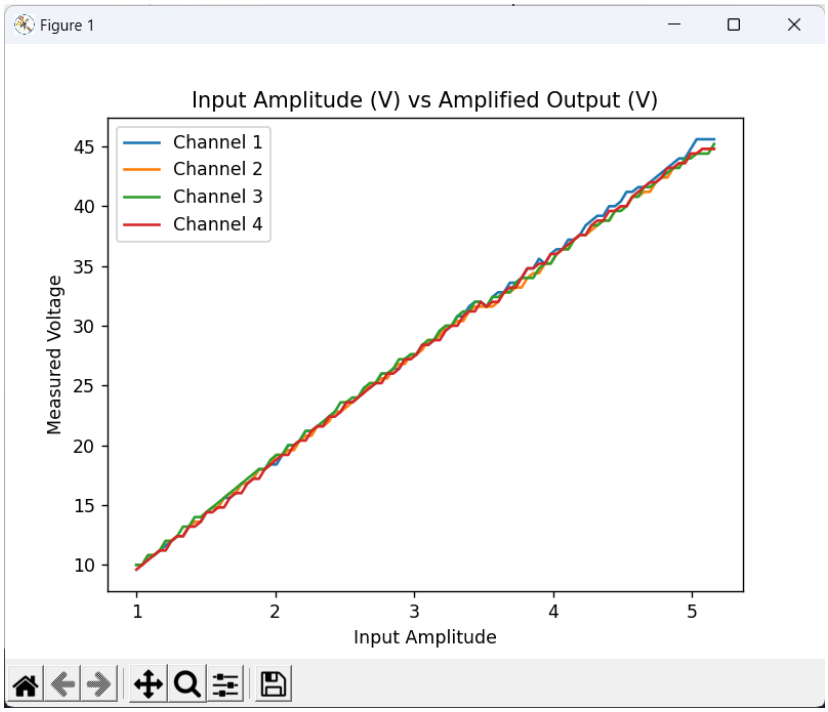


Figure 8.20: Output measurement with 100 iterations

```
on 316: 4.317999999999991 V in
on 320: 4.3599999999999905 V in
on 324: 4.40199999999999 V in
on 328: 4.44399999999999 V in
on 332: 4.48599999999999 V in
on 336: 4.52799999999999 V in
on 340: 4.56999999999999 V in
on 344: 4.611999999999989 V in
on 348: 4.653999999999989 V in
on 388: 5.073999999999987 V in
on 392: 5.115999999999987 V in
on 396: 5.157999999999987 V in
|                                        | 400/400 [100%] in 14:36.0 (0.46/s)
```

Figure 8.21: Terminal output after 100 iterations

Chapter 9

AMPLIFIER REVISION DESIGN

The investigation of the first generation amp design and testing gave the team useful insight into what needed to change for the next hardware revision. An evaluation of the goals of Nessie2 provided the basis for the redesign. By exploring a number of hardware architecture choices we begin to rebuild the amplifier circuit from the beginning.

9.1 Nessie2 Overview

As capabilities of the CSEs and SLEDS arrays have increased, so too have requests from customers. What started as a project to drive a single 512x512 LED array has turned into a deep investigation to the physics and hardware constraints involved with increasing frame rates and resolutions. The eventual goal is a system able to drive a 2Kx2K array at 2kHz [6].

To move toward this new goal a distributed architecture using smaller CSEs was envisioned. Each CSE would be capable of driving twice as many analog channels on a quadrant of the new 2Kx2K array. The array would therefore be driven by four synchronized systems. To accomplish synchronization and analog integrity at this scale required a complete remodeling of the system firmware and hardware architecture. A prototype "Multi-CSE" system has been constructed as a proof-of-concept. In short, this system broke the Dewar pins out to routing boards that accepted all four cable outputs from a CSE each. Using the full bandwidth of a system greatly increased potential resolution and frame rate such that no system was driving more than a normal SLEDS array's worth of pixels. More details on the Multi-CSE prototype and Nessie2 architecture can be found in Christopher Jackson's doctoral dissertation [6].

The biggest change directly relevant to the analog design was that the main system would only output unamplified DAC signals along new coaxial cables to the Dewar. The Dewar's main input board will contain cable receptacles and signal paths to mimic the old interface board. Included in this main input board will reside the new amplifiers. Moving this processing stage outside the system allowed us to consider many more functions for the amplifiers due to their proximity to the RIIC, which will be explored below.

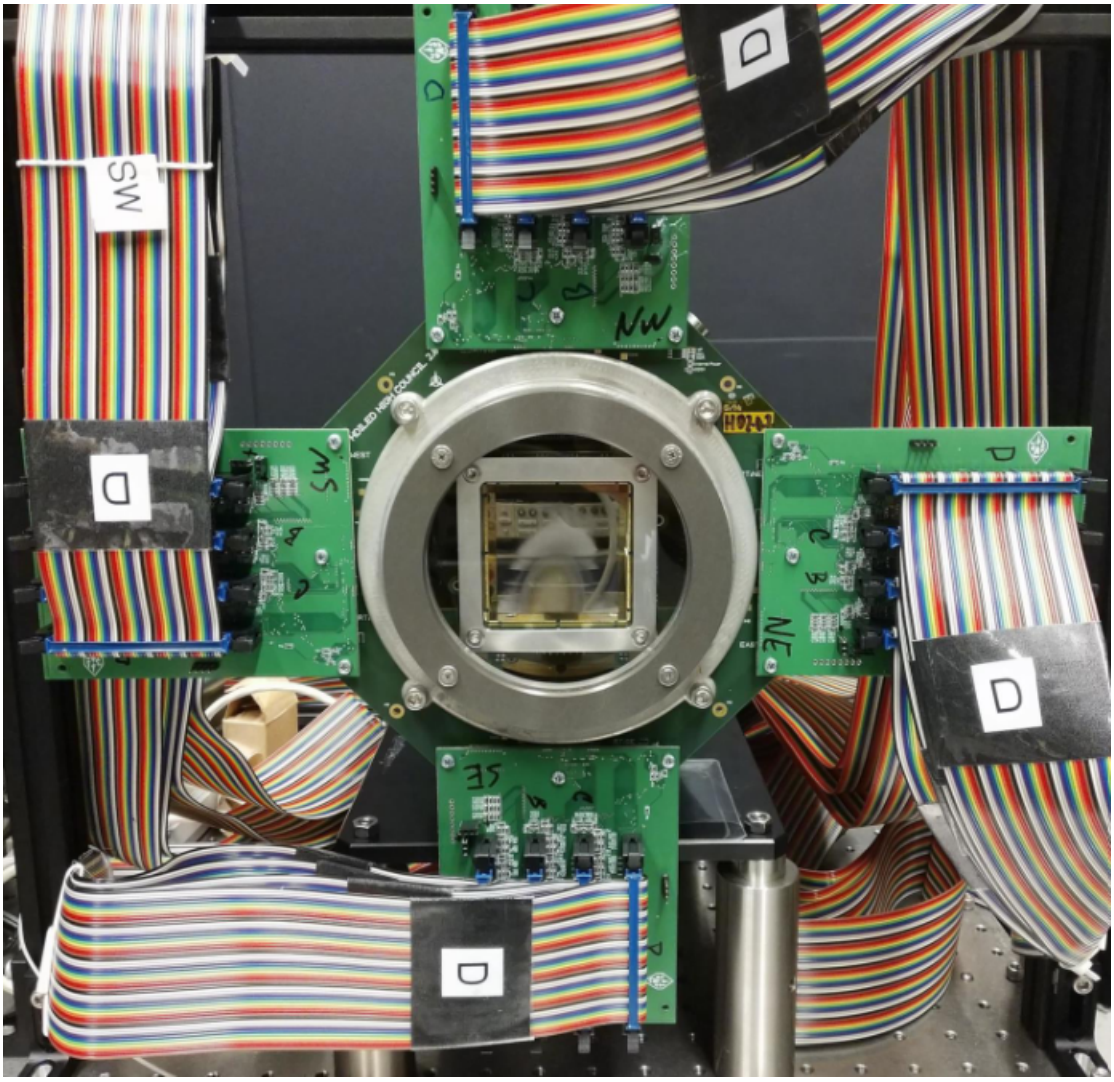


Figure 9.1: Multi-CSE Prototype System [6]

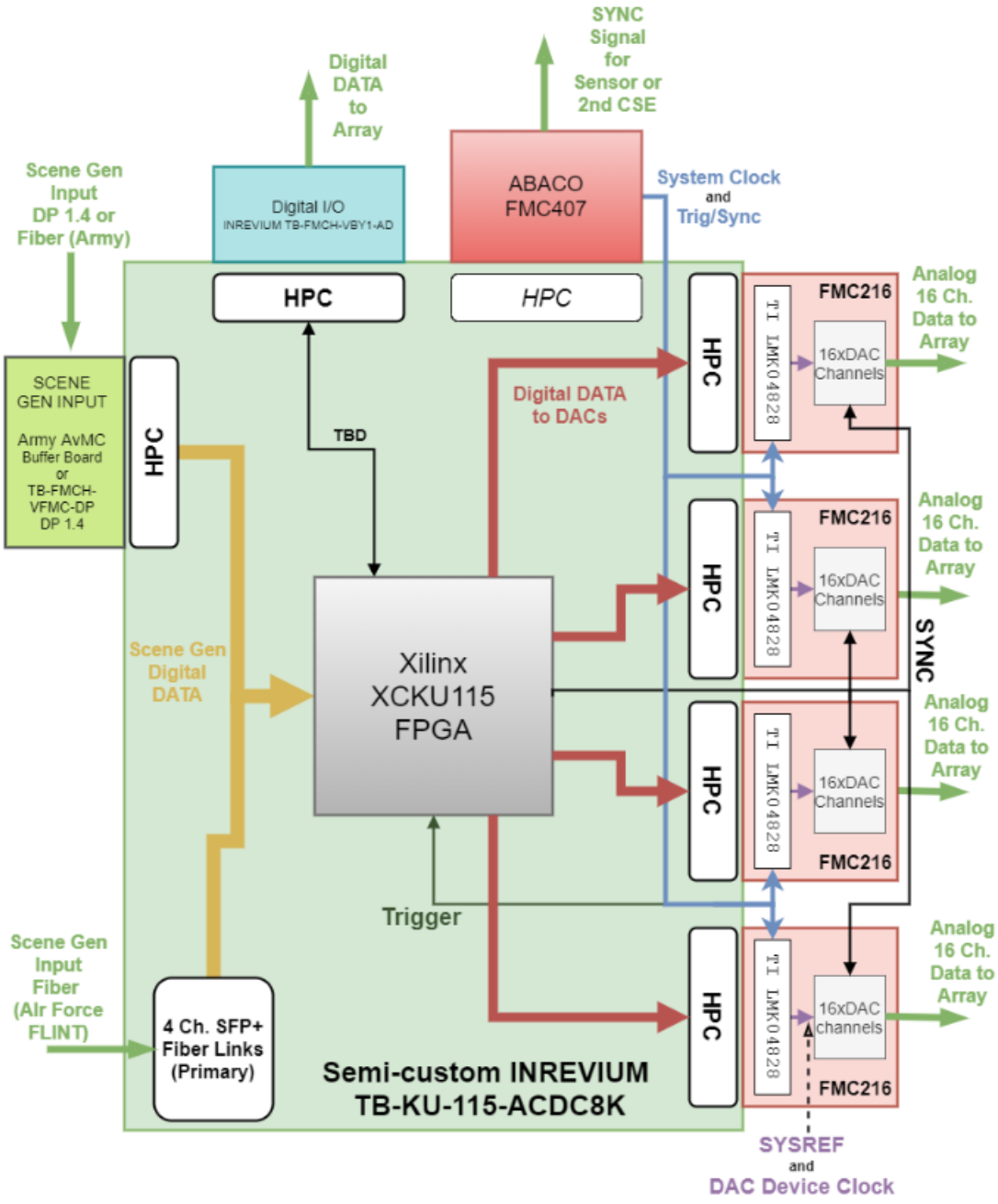


Figure 9.2: Nessie2 System Architecture. Of note are the external ports for pre-amplified DAC output [6].

9.2 Objectives

The goal of the new amplifiers is twofold. The new system hardware already requires new amplifier cards as the current ones are too big. The hardware is due

for a revision thanks to the excess of unpopulated components, empty connectors and unexplained design choices that waste space and money on the PCB. Additionally, due to the new increased resolution and frame rate, the system will have to be able to monitor and tune the amplifiers in order to maintain consistent output. These cards have to have some level of analog analysis and response to appropriately self-tune in conjunction with providing signals of the same (hopefully better) quality as the first generation. The self-tuning process is a response to the useless gain and offset potentiometers on the current design that do not affect gain or offset in any meaningful way. This is similar to the corrective role of the PC in creating NUC tables, but is system-level and intended to compensate for electrical differences in the large number of channels. An abstract architecture is as follows:

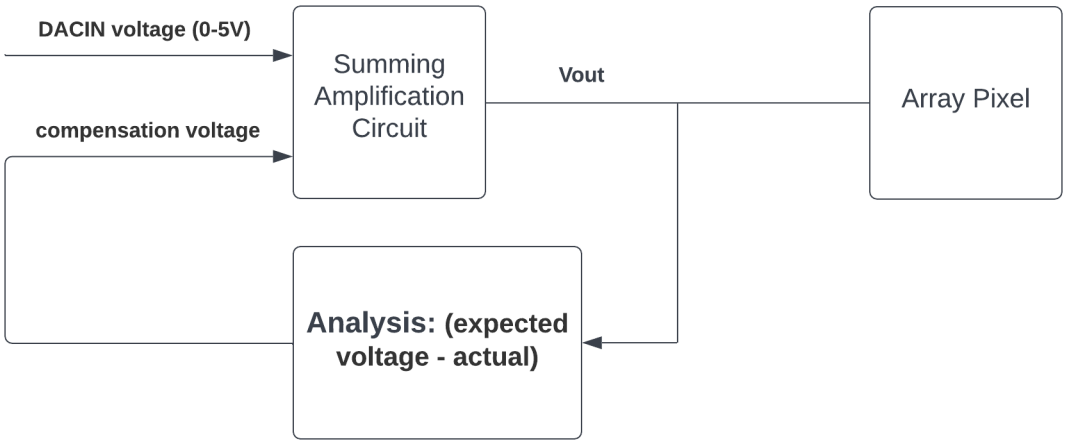


Figure 9.3: Overview of the new analog component

9.3 Amplifier Design Choices

To properly analyze and respond to input, the system had to have some sort of analog probe point to read amplifier output. The simplest solution is to use an ADC to convert the reading into digital data. Once the voltage reading has been converted, a microcontroller or other appropriately small digital component should be able to read and react to this input. The result of the calculation can then be directly output

as a voltage through a DAC, where the microcontroller will tell the DAC how much extra voltage to add to the summing amplifier component. By implementing a small subsystem that can automatically perform this task we remove the need for manual gain/offset controls and further ensure reliability for the new system.

Working on the previous test setup gave me a foundation for understanding real-time measurements and analog output. Since each channel can be individually tuned regardless of the others, the best architecture should utilize some level of parallelism to work through all 64 channels as quickly as possible.

9.3.1 Parallelized Architecture

The new Nessie2 system will have 64 analog channels. This in itself creates an interesting problem for the analog designers, as all 64 channels must be accessible for reads to our data processing system. An initial design consisted of placing 32 ADCs on each side of the amplifier board and having the microcontroller probe them. Many microcontrollers come with multiple ADCs built-in but are usually not able to read from all of them at once. In the interest of pure parallelism, discrete ADCs on every single line would be the most effective solution to the problem. If individual components are going to be placed on every line rather than switching through them with an extra multiplexing layer, an FPGA would be the most effective way to analyze and respond to ADC inputs. The FPGA could be split into as many cores as required (64 in this case) and programmed to self-tune all channels at once when given a start signal. As the main controller, the FPGA would simply have to know what the targeted pixel voltage is (based on the DACIN value) and compare it to the amplifier output. If the amplifier is struggling to meet the target, the FPGA would command its channel's DAC to output compensation voltage that would be added to the DAC input through a summing amplifier. A parallelized architecture would also make testing simpler as individual cores or channels could be examined for issues.

While this would be an effective strategy for parallelzing the workload, discrete ADCs are prohibitively expensive and unavailable for purchase from many vendors

due to the chip shortages at the time of writing. The amplifier additionally has size constraints at the moment that prevent us from adding too many components to the board. Hardware-level parallelism is always a balancing act between speeding up the task and adding more and more hardware to the design. While we believe a fully parallel amplifier tuning architecture is the best eventual goal, testing and development will for now be confined to a smaller scale microcontroller setup.

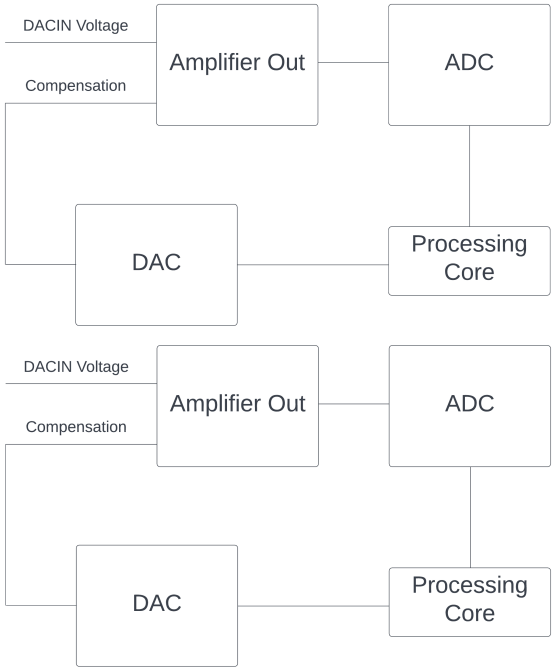


Figure 9.4: Block diagram of potential parallized architecture with 2 channels. Scalable to the physical limits of the FPGA.

9.3.2 Sequential Architecture

To establish an effective parallel architecture in the future we must first solve the problem sequentially. To accomplish this a similar strategy will be taken of digitally analyzing input and outputting results to a DAC for summing. When performed sequentially, however, the large number of channels requires the designers to figure out an addressing or switching scheme to read from/write to all 64 channels. To accomplish this the channels can be grouped into multiplexing ICs which allow selection of a single

input based on a select signal from a digital device. The main controller will be able to request a read from a specific channel which will activate select signals in a way that connects the selected channel to a single analog input. The main benefit of this approach is massively reduced cost and board size, as mux chips are small and the single ADC + DAC required are no longer an issue of cost. By testing this approach we plan to benchmark measurements and see if analysis can be carried out in a timely manner. Embedded solutions require clever programming and a full understanding of the hardware constraints.

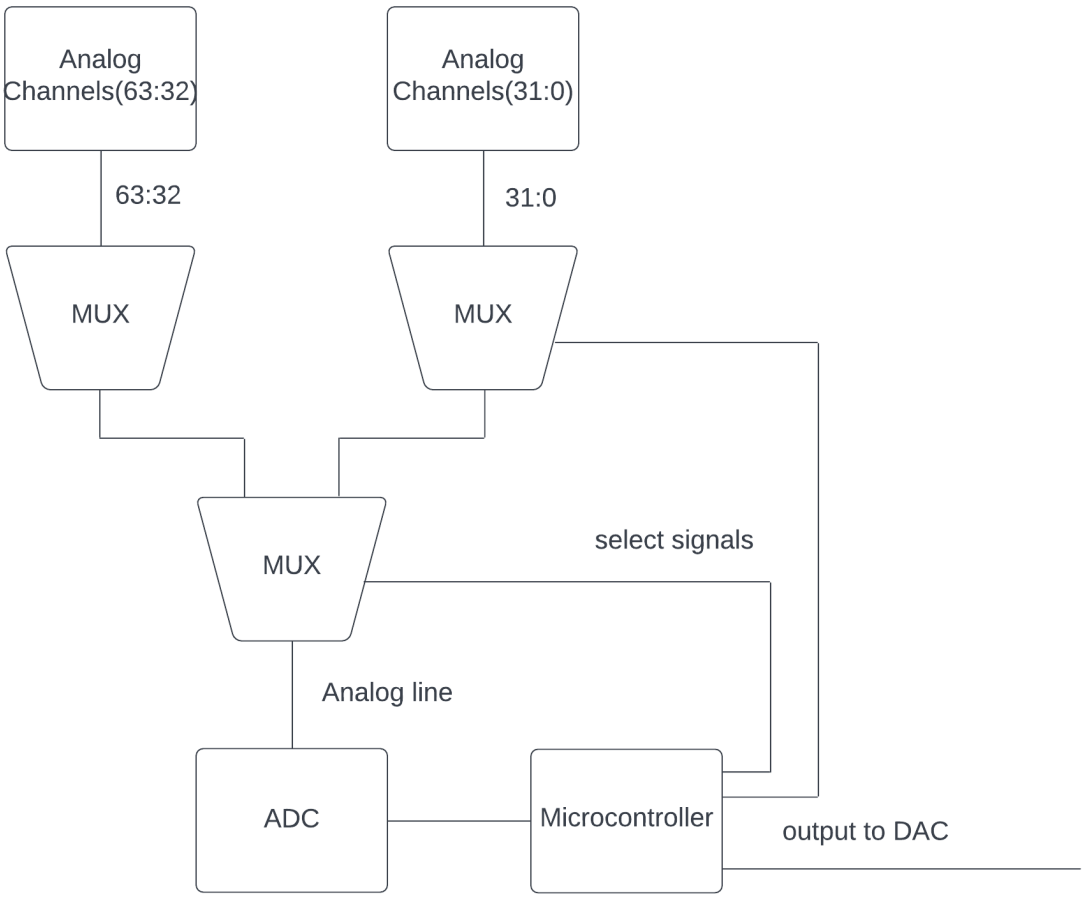


Figure 9.5: Microcontroller with multiplexed addressing

REFERENCES

[1] Devon Andrade. Basic vga controller design example. *Altera*, 2015.

[2] Peyman Barakhshan. Thermal performance characterization of a 512x512 mid-wave infrared super lattice light emitting diode projector. Master's thesis, University of Delaware, 2016.

[3] T. Browning, C. Jackson, R. Houser, A. Landwehr, H. Ahmed, and F.E. Kiamilev. A modular platform for rapid irsp development. *IEEE Photonics*, 11(3), 2019.

[4] G. Franks, J. Laveigne, T. Danielson, S. McHugh, J. Lannon, and S. Goodwin. Development of an ultra-high temperature infrared scene projector at santa barbara infrared inc. *SPIE DEFENSE + SECURITY*, 9452, 2015.

[5] Miguel Hernandez. Data collection and analysis of read-in integrated circuits designed to drive arrays of infrared light emitting diodes using a scalable and modular testing platform for infrared scene projector. Master's thesis, University of Delaware, 2016.

[6] Christopher Jackson. Hardware and close support electronics architectures for enabling a packetized display protocol on irled scene projectors. PhD Dissertation, University of Delaware, 2021.

[7] Tianne Lassiter. Scalable board architecture design and mechanical adaptations for infrared scene projector systems. Master's thesis, University of Delaware, 2019.

[8] Thomas H. Lee. *Planar microwave engineering: A Practical Guide to Theory, Measurement, and Circuits, Volume 1.* Cambridge University Press, 2004.

[9] Joshua Marks. Abutted irled infrared scene projector design and their characterization: A tale of strife and semiconductors. PhD Dissertation, University of Delaware, 2019.

[10] Rigol. *DS1000Z Programmer's Manual*, 2019.

[11] Inc. VXIbus Consortium. *VXI-1 Revision 4.0 System Specification*, 2010.

Appendix A

TITLE OF APPENDIX

This is the information for the first appendix, Appendix A. Copy the base file, appA.tex, for each additional appendix needed such as appB.tex, appC.tex, etc. Modify the main base file to include each additional appendix file.

If there is only one appendix, then modify the main file to only use app.tex instead of appA.tex.