Mr. Ohm FPGA Subsystem

Final Report

Team 44 Sdmay23

Client: BetterBots Daniel Walker

Advisor: Nathan Neihart

Team Members/Roles:

- Raj Singh: Management/FPGA Design
- Jordan McGhee: FPGA Design
- Tyler Smith: ADC Design

Team Email: sdmay23-44@iastate.edu

Team Website: http://sdmay23-44.sd.ece.iastate.edu/

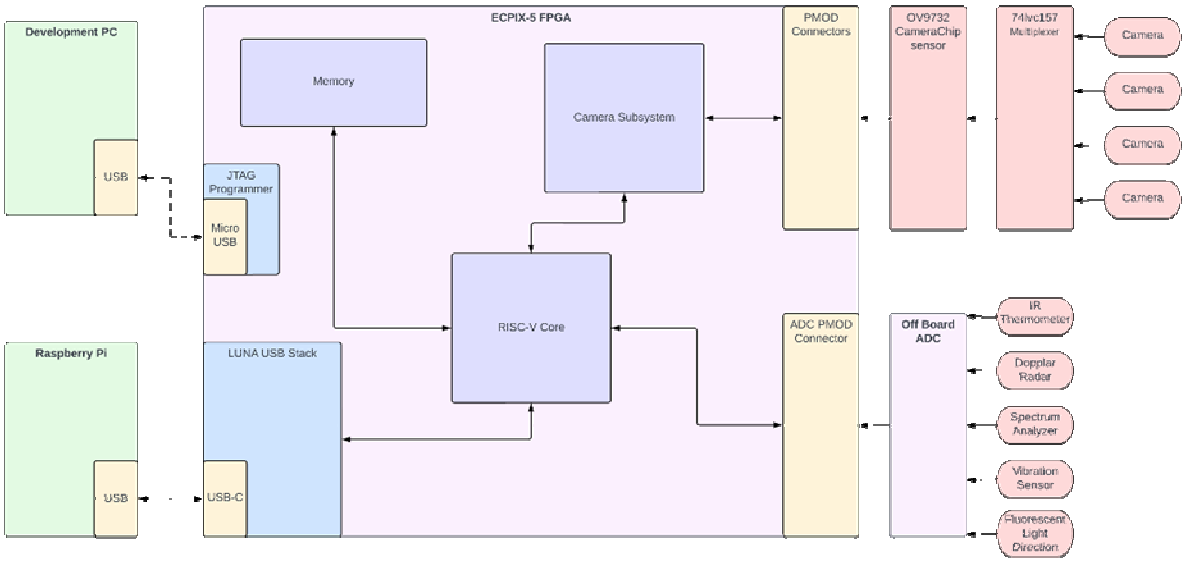


Table of Contents

Executive Summary

Development Standards & Practices Used

IEEE Standard VHDL Language synthesized through Python

IEEE Standard for Verilog synthesized through Python

Summary of Requirements

- ECPIX-5 FPGA core interface with the USB stack
- Create a development environment for interfacing with ECPIX-5 FPGA core
- Create a camera subsystem for interacting with Camera sensor (OV9732) ECPIX-5 FPGA core
- Create a block in LiteX for connecting ADC to SOC
- Develop off-board Analog-Digital Converter

Applicable Courses from Iowa State University Curriculum

- CPRE 281/288/381/480
- EE 201/230/330/333

New Skills/Knowledge acquired that was not taught in courses.

- Benefits and challenges of using USB based communication.
- VHDL generator frameworks called LiteX
- JTAG
- Serial Communication

Project Design

Overview

This project included two main fields of development, one side consisted largely of digital design working with a Lattice FPGA, while the other field required experience with analog circuity and PCB design. In order to accomplish this task our team split into two groups, one oversaw designing and implementing the FPGA portion and the other team working towards improving the ADC design.

FPGA

For the FPGA portion of the BetterBots robot, we were tasked with setting up the FPGA we were given with a system that included a processor, and two specialized digital blocks that served to connect an ADC and a camera system into the SOC. This system was intended to be accessible such that a raspberry pi could request information from a program running on the FPGA to control and receive information from the various peripherals on the board (those being the camera subsystem and ADC).

USB

One of the important constraints that we were given for this project was to ensure that the communication between the Raspberry PI and the Risc-V core was through USB. This constraint was to ensure there would be a high-speed communication method for streaming the video feed from the device.

Camera Subsystem

The camera subsystem was initially intended to take a stream of camera information, perform a set of operations on the data, and then write the result to memory to be accessed via through dma or from request through the processor. This part of the project was considered a lower priority when compared to the USB and ADC portions of the project and so its scope was reduced to writing a set stream of information into memory, simulating the output of its implementation.

ADC Wrapper

Within the ADC portion of the project there was an inherent engineering need for communicating the results of the ADC back out to the rest of the system. There were no requirements for this aspect of the project, but it arose out of our need for testing. This wrapper included a pipelined averaging circuit and a wishbone interface.

ADC

For the ADC portion of the project, we have been asked by our client to see if we can find a way to reduce the noise that is from the comparator inside the FPGA. The ADC is classified as a Delta Sigma variant which has many benefits for the application of embedded systems. The circuit is less reliant on its analog portion for better resolution making it ideal for being less costly. This allows it to improve its resolution even further than typical ADC, which makes it a lot more accurate between analog and digital readings. As well as having a very good bandwidth for stability which makes them a very desirable choice.

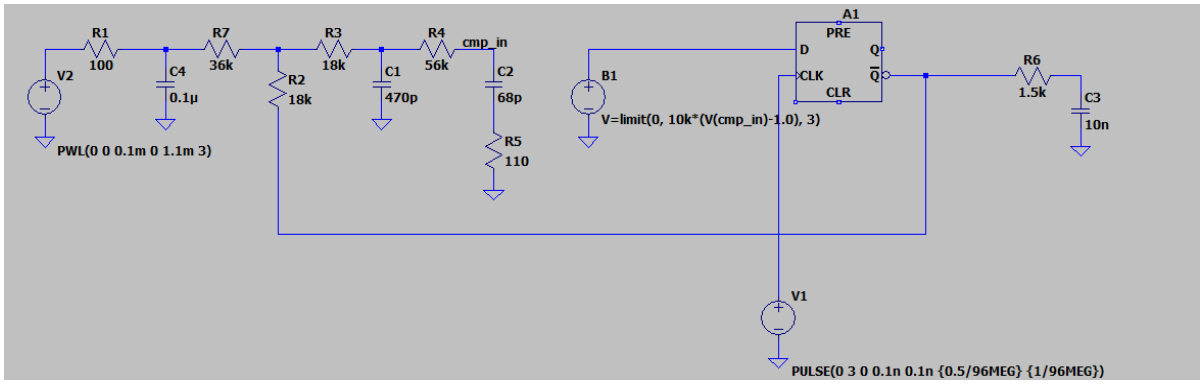
Detailed Design and Visual(s)

USB

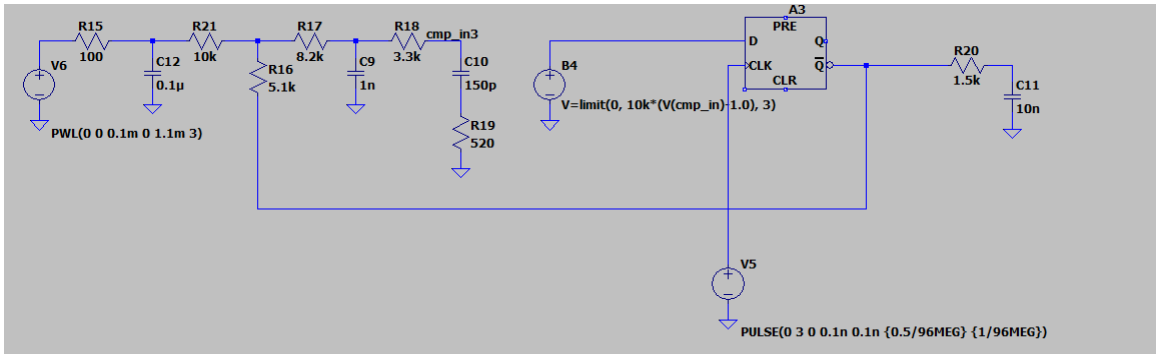
The FPGA portion of the project presented significant challenges, particularly in implementing USB communication. We initially intended to leverage existing projects due to the complexity of creating a USB controller from scratch. However, this approach resulted in numerous design iterations, largely due to compatibility issues with third-party libraries. Our first attempt involved using the Luna library to generate VHDL and wrapping it with LiteX. However, we encountered compatibility issues between the two libraries that made creating a custom wrapper difficult. In the second iteration, we attempted to use Butterstick, an open-source library that had an implementation that suited our needs. However, this approach failed due to poorly maintained dependencies that were incompatible with modern LiteX. Despite the lack of documentation, we persisted and ultimately incorporated the OrbTrace library, recommended by our client. With modifications, we successfully integrated this library into our project and were able to achieve our goal of communication through the USB-C port on the FPGA.

ADC

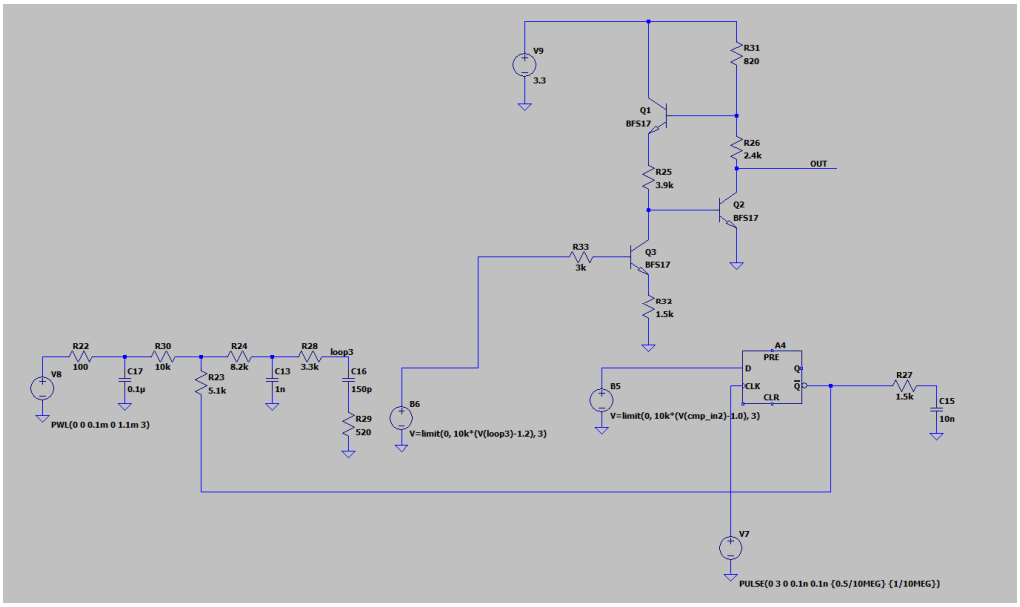
The ADC uses a delta-sigma variant which allows the incoming analog signal to be converted to a digital value the comparator of the FPGA can interpret. Our full version includes this circuit and includes an amplifier which is better in the sense that it can amplify the incoming signal making it easier for the FPGA to recognize. Our client restricted us to a discrete component amplifier which is great for recognizing AC signals such as audio or vibration but bad for DC signals. The DC signals are not accurate since component tolerances cause the DC bias voltage to change.



Original Delta Sigma Spice Schematic (Above)



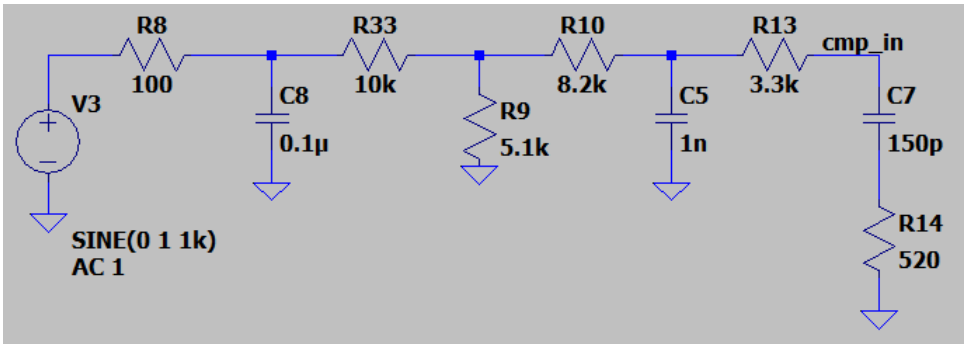
Delta Sigma Modified Component Value (Above)



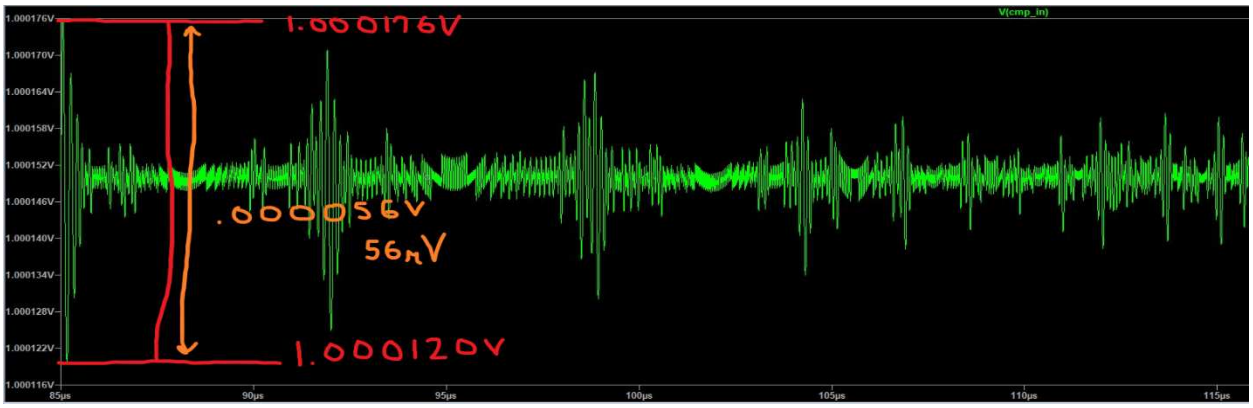
Delta Sigma Circuit with Amplifier (Above)

Passive Circuit

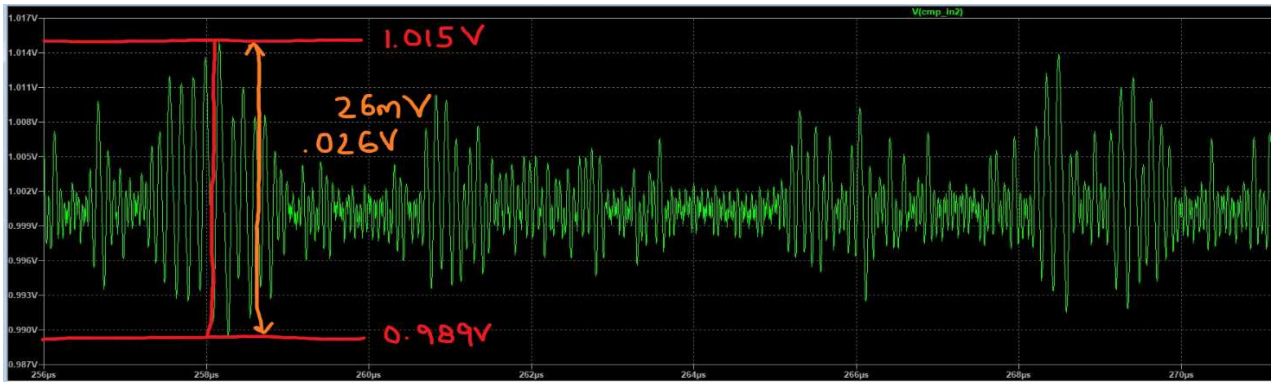
The already established part of the circuit mainly known as the delta-sigma ADC kept the same layout throughout the entirety of the project. We adjusted the delta sigma loop filter components of the passive circuit to compensate for the added amplifier which operated at a reduced frequency. For the record our original passive circuit operated at a frequency of 96MHz and now our new frequency is at 10MHz. This would change the location of the poles and zeros of the passive circuit. The poles and zero locations of the circuit were established with the aid of the Betterbots circuit descriptions and the frequency at which audio signals are interpreted at from the human ear.



Modified Delta-Sigma ADC Circuit



Voltage Swing of Delta Sigma (Above)



Voltage Swing with Amplifier

The two following waveforms above indicate the importance of increasing the gain of the output of the Delta-Sigma ADC. Without the amplifier the signal only swings in the range of microvolts whereas the one with the amplifier can increase its range into millivolts. This helps prevent the noise from the input of the comparators from drowning the signal and misreading the actual value.

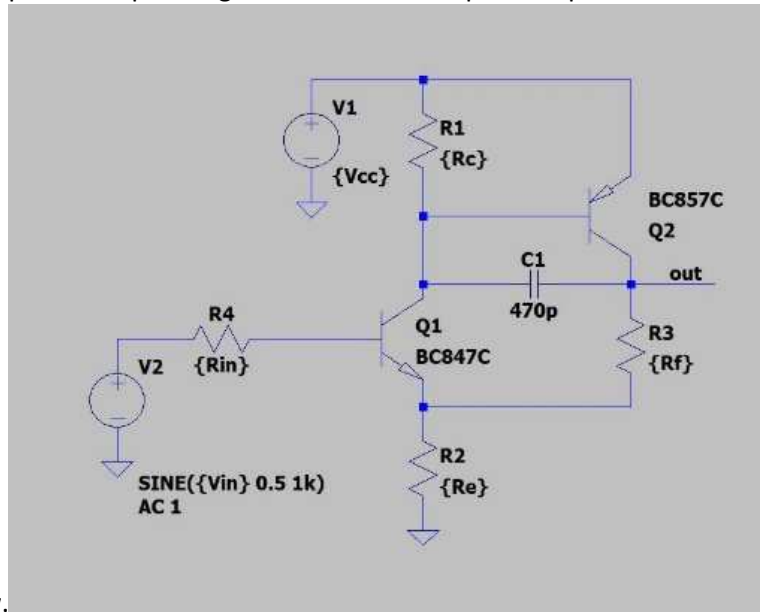
The following formulas were used to calculate the poles and zero of the delta-sigma. Please refer to the image above to reference each component. Pole1: 1/(2*pi*((R33||R9)+R10)*C5) = 14874.29Hz , Pole2: 1/(2*pi*((R33||R9)+R10+R13)*C7) = 75788.06Hz , Zero: =1/(2*pi*R14*C7) = 2MHz. These values were selected to ensure a zero analog input could result in an output above the FPGA comparator threshold with a 3.3V digital high FPGA feedback voltage. Full details about the formulas we used to dictate the position of poles and zeros can be viewed on the BetterBots website on page 15 of the document.

The first pole was put at 15KHz since most audio signals or noise in general becomes very hard to hear past this point. From the Betterbots website, the second pole needed to be located at a value 5 times greater than the first so 75KHz. The zero needed to be placed at the sampling rate divided by 5. Since our sampling frequency is 10MHz it was placed at 2MHz. From there we could start deciding on the actual resistor values. There was one thing to be assure as we were picking them. Since there was a feedback loop coming the comparator, we needed to perform a test where when we had no signal coming from the input, we had to be sure that it was below the threshold voltage to register as a zero at the output of the passive circuit. To elaborate and clarify, when the feedback is receiving a “high” value and the input is reading 0V the output will be interpreted as a logic “low”. If that condition was met and the frequencies were as well then resistors could be decided on.

An afterthought but never became an issue was parasitic capacitances. Although we never got to breadboarding due to time and resource constraints it was considered. We needed our capacitors in the circuit to be large enough to not be affected dramatically by the parasitics from the oscilloscope probe and breadboard itself. This would alter the values of the poles and zero. 10pF came from the probe and 2-20pF from the breadboard. These were accounted for when designing this circuit as well.

Active Circuit

The major contribution to the ADC portion of this project was incorporating an amplifier to the overall design. Its' purpose being to make the analog signal that was being interpreted by the comparator on board of the FPGA larger and thus easier to detect. This design started off with the attempt of incorporating a "series-feedback pair" amplifier. Pictured below.

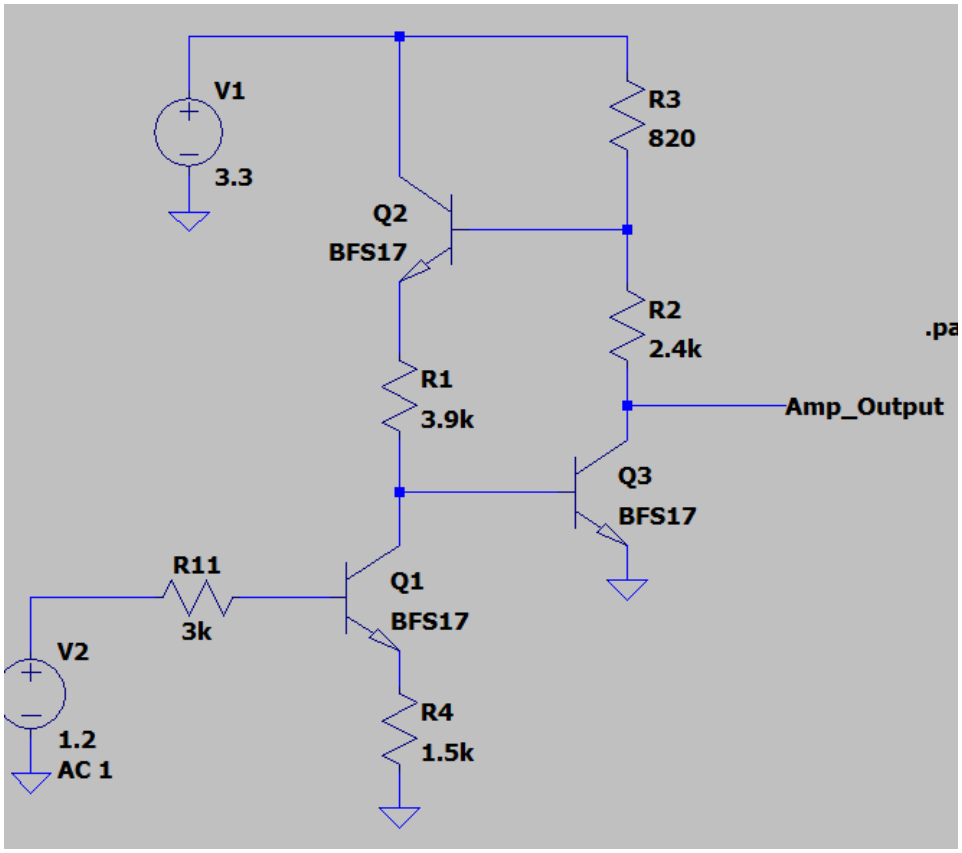


This was suggested by the client as a first attempt, and it was incorporated into another part of the robot. This amplifier is composed of two BJT's (Bipolar Junction Transistors, 4 resistors, and 1 power supply. This design quickly fell apart due to its poor stability and lack of bandwidth, which was very desirable to us. It was seen that as we incorporated the amplifier with the ADC circuit, we were trying to use its own internal pole in conjunction with the ADC's pole to stabilize it. The poles within the series feedback pair amplifier were found to be very complex and hard to control. One of the driving constraints for the amplifier was to achieve less than 10 degrees of phase shift at the delta signa sampling rate. We estimated that 10 degrees was the maximum amount of phase shift that would keep the delta sigma feedback stable. Otherwise, this would result in oscillation of the delta sigma system. Thus, why we lowered our 96MHz operating frequency to 10MHz.The phase would continue to drop drastically as we were observing its behavior. In addition to the transistors falling out of their linear region of operation which was needed for amplification purposes. This was regardless of our resistor's values which dictated the behavior of the individual BJTs, and amplifier circuit as overall poor. Thus, after heavy consideration, we decided we needed a new amplifier to suit our needs better.

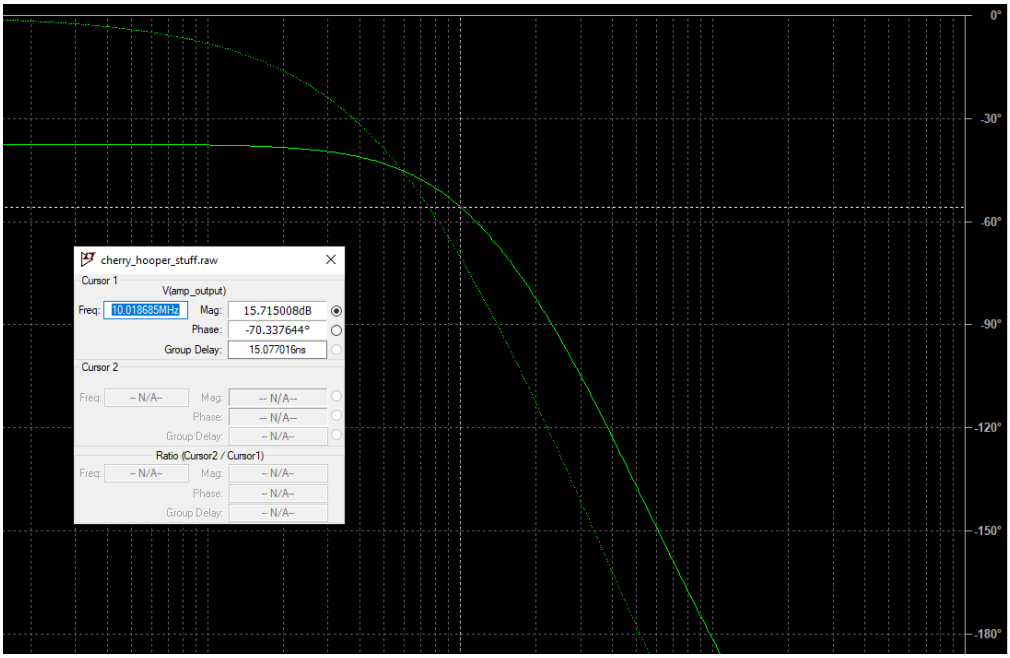
"Cherry Hooper" Amplifier Design

After talking with our client about the issues, he recommended another amplifier that was also in another part of the robot. The "Cherry Hooper" amplifier was brought to our attention and from our initial analysis it was deemed a good contender. Consisting of 3 BJTs, 5 resistors, and 1 power supply. This circuit encompassed the characteristic of having a large bandwidth which we were looking for. That

all being established and with a rough layout in front of us we could start setting specifications for the amplifier to meet. Our client wanted to see 20dB of gain and a bandwidth of 10MHz. One member of our group found this task challenging because they had no prior experience biasing a circuit to reach those specifications. The BJTs needed to be assumed to be in the “triode” region of operation or the linear region needed for amplification. This put in place voltages between the base and emitter of the transistors and would need to be considered when doing simple analysis like KVL (Kirchoff’s Voltage Law).



The more important goal of biasing this circuit was determining the resistor values so that the transistors stay in the triode region of operation. Simply the resistor values could not be assumed with just one assumption. The resistors were calculated using: the power supply values, base-emitter voltage values, collector currents, and other pre-established valued resistors. These were decided to be our biasing parameters and would allow us to dictate the gain and bandwidth. The best way we went about this was sweeping the collector current of two of the transistors and the resistance value of one of the resistors. This was all done in LTSPICE XVII. When we did this, there were around 350 different runs. The first collector current sweep goes from 0.1-0.4mA with a 0.05mA step. The second collector current sweep goes from 0.4-0.7ma with a 0.05mA step. The resistor was swept from 900 ohms to 2.7k ohms with a 300-ohm step. Analyzing that many runs in one graph can be very hard by hand so with the help of the computer engineers in the group they developed a python code that aided this process. Eliminating any runs that were not near our 20dB gain and 10MHz bandwidth. This was done before we found out that LTSPICE had a command to do this for us. Up to that point we found a run with a gain of around 18.68dB and 10MHz bandwidth. This led to the following parameters: Icc6 = 0.65mA, Icc4 = 0.4mA, R3 = 831ohm, R2 = 2.4k, R1 = 3899ohm, R4 = 1497ohm. This wasn’t the 20dB we exactly wanted but was our design for a while and was what we went with to print for our PCB design.



Cherry Hooper – Phase Shift

We would later discuss with our client over email how we could increase the collector currents of one of the transistors to increase our gain and bandwidth. It was unfortunate that we didn't catch and address this before ordering the PCB. This also brought attention to using E-series value resistors which would alter our gain and bandwidth slightly which would have opposite effects of each other. If one increases, then the other decreases and vice versa. We did decide to repeat the simulation sweeps with a higher collector current range and found one that achieved the 20dB gain and a slightly higher bandwidth of 10MHz to compensate for the E-series resistors. From this point PCB assembly and testing were next.

ADC/FPGA Integration

The FPGA will use some of its onboard connectors to communicate with the ADC, off-board cameras, and the USB stack. All internal connections will be made possible using LiteX's built-in solution, the Wishbone bus. The FFT and ADC will need to be sampled at a very high speed, so we planned to accelerate the sample rate using the onboard PLL. The maximum theoretical sample our client wanted from us was 500 MHz, which is our target speed. For the camera subsystem, the client wanted three different modes for camera operation. Color segmentation, Motion detection, and passthrough. With the camera subsystem, due to its high memory usage, we opted to give the camera system its dedicated memory to free up resources on the Wishbone bus. Finally, we will have the RISC-V core, which can run generic code on board to offload some of the traditional tasks.

Block Diagram of Delta Sigma Modulator ADC (Analog)

For the ADC portion of the project, we have been asked by our client to see if we can find a way to reduce the noise that is from the comparator inside the FPGA. We are working on seeing if adding an amplifier to the current circuit is worth using because it can reduce the noise then. The problem we face is that the amplifier ruins the DC accuracy of the converter. If we pursue this path then we should seek to find ways to offset that inaccuracy. If not we are looking for something to fill the role of the amplifier. We are primarily focused on the analog modulator which is the primary part of the ADC shown above. It is the core of the converter and produces the bitstream that is needed for a digital signal. The only other

component of this ADC is a low pass filter since the average level of the bit stream is low and the noise surrounding it is high so the filter helps remove this.

Functionality

The off-board ADC will be used to convert the data collected from the surrounding environment via the various sensors. The data derived from the digitalized signals will then be used to communicate information to the various users, such as temperature.

Requirements and constraints

Type	Requirement	Notes
Functional	ECPIX-5 FPGA core interface with the USB stack	Building a USB client connected to a Raspberry PI zero
Functional	Build LiteX compatibility for ECPIX-5 FPGA core	This is to allow for LiteX to correctly flash to the FPGA
Experimental	Create a development environment for interfacing with ECPIX-5 FPGA core	Creating a docker build a development environment for the client that is maintainable
Functional	Better bots camera interface with ECPIX-5 FPGA core	Will require developing a camera subsystem that allows us to manipulate camera data before sending to the RISC-V core
Functional	Make a communication state machine for interfacing camera subsystem to RISC-V	This would need the actual camera subsystem designed, as well as throughput goals set up.
Functional	Implement sensor interface with ECPIX-5	Requires developing the analog sensors to collect the data for testing.
Functional	Analog-Digital Converter	Design/Modify ADC to quantify resolution of an audio signal to set binary values that can be used/interpreted by the FPGA

Engineering standards

- RISC-V ISA - RISC V architecture is a lot easier to modify as it's easier to learn, and the hardware is simpler.USB - Industry standard for communication between devices. Also provided on our FPGA
- SPI - Serial communication interface for embedded devices. Will be used for communication between sensors.
- I2C - Communication used for sensors

Areas of Concern and Development

As this is a tool geared towards programmers, there is a potentially infinite number of individual user needs (Support for some sensors, lower power applications, etc…), but given all of the core pieces, basic image detection, and analog sensor support, our design meets the needed user requirements. As of right now, the biggest concern is meeting the client's design requirements surrounding the ADC. Keeping in sync with the client to ensure the proper new design meets design requirements (Sampling speed, Transmission method). There are also concerns coming from the FPGA side as well, as is shown below, we are not getting the speed that we would expect from using the USB stack.

Design changes since December 2022

FPGA

There have been countless design changes since the end of the last semester on the FPGA front. These changes included alterations to the library used for the USB stack, the development board used for operating the ADC, and the scope of the camera subsystem. On the USB stack side, due to a lack of maturity and support for our selected method of synthesizing and flashing our SoC onto the FPGA, we ran into quite a few issues with incompatible dependencies. The framework we were using to generate our SoC was LiteX, a framework that none of our team members had experience with, and so it proved infeasible to make the required modifications to support the first two libraries required for integrating USB into the project. In the end, we found an example project named OrbTrace that we were able to modify to suit our needs. On the camera subsystem front, quite simply as time went on through the semester, and our issues with the USB stack persisted, we were unable to integrate some of the features that were originally planned. Finally, as we neared the end of the semester, we found that we were unable to verify the ADC operation using the original development board we were given (Lambda Concept ECPIX-5). This was caused by two issues, LiteX was not able to correctly place our differential inputs, meaning we were unable to use the internal comparator on the board which was critical to ensuring hi operation for our ADC. As a backup, we then tried imaging just digital part of the ADC onto the board using the Lattice Programmer, which was the original vendor tool synthesizing and programming the FPGA that we had. However, to use this tool for the specific FPGA we had, we needed to have specific licensing, which we were not able to acquire fast enough to be used in our project. In order to res resolve this issue, we ended up programming one of the development boards available in the (DE2-115 Altera dev board). Using this we were able to successfully capture the needed results.

ADC

The design changes that have occurred in the ADC portion this semester have been in the passive and active circuit. The passive is referring to the ADC circuit composed of passive components (resistors and capacitors). The active circuit refers to the amplifier circuit consisting of the BJTs and resistors.

To recap where we started with the ADC at the beginning of the semester we were trying to work with and develop the “series feedback” amplifier. This was short met with many pending issues that needed a lot of time and resources in order to find what “could be” a solution. The best answer was to find a new design that worked for the specifications we were looking for. The Cherry Hooper design was the major and most significant change made in the ADC portion of the project. This circuit would see many changes within itself throughout the semester including exchanging the BJT for a different one with a higher operating frequency. We started with the BC847C with a operating frequency of 100MHz, and changed it our for the BSF17C which had one of 3.2GHz. Other components like resistors would see changes as well in that circuit to achieve the gain and bandwidth we were looking for.

As for the passive circuit, the layout, or the “footprint” of it did not change and was not of concern until the later half of the semester. Once we established a bandwidth for the active circuit that result would determine how we gave values to the components in that circuit. This was also in conjunction with other values such as the threshold voltage and frequencies of input signals coming

from other parts of the robot. All together these allowed the proper and needed changes to create the final version of this circuit.

The Cherry Hooper design initially came from our client and keep in mind he threw it together in a short period of time. The circuit went under a few notable changes from that point on worth talking about. Just for comparison the clients circuit had a gain of about 14dB and a bandwidth of around 30MHz. Our circuit achieved a gain of 18.78dB and a bandwidth of 10MHz. Which was better in terms of bandwidth but was lacking in the region of gain. The layout of the circuit stayed the same, but a resistor was tacked on at the base of transistor Q1. This was to aid in stability of the circuit but in hindsight maybe was not needed since the circuit has a large natural bandwidth as it is. The other 2 changes were in the resistor values for biasing the circuit to keep all transistors in triode for amplification purposes. The other being the actual transistors themselves from the BC847C to the BFS17 due to its higher region of operating frequency. This falls in line with the changes since December of 2022 but a noteworthy remark to add since it was a part of the process of getting us to the end goal.

To know more about these changes and the details that followed please direct yourself to the “Detailed Design and Visual(s)” portion of the document.

Results

FPGA Results

For the results on the FPGA portion of the design, we were able to meet our requirements for two of the larger design requirements, but the other two came up short. We were able to meet our design requirements for the memory subsystem portion of the project, where we created a digital block in LiteX that were to memory, simulating receiving camera information. To test this, we would run the LiteX BIOS and call the command mem_read supplied with the location and width of the memory section we allocated in memory. For correctness, we expect that each memory address corresponds to its index. As shown in the image below we read the desired sequence.

```
litex> mem_read 0x20000000 4096
Memory dump:
0x20000000  00 00 00 00 01 00 00 00 02 00 00 00 03 00 00 00  ................
0x20000010  04 00 00 00 05 00 00 00 06 00 00 00 07 00 00 00  ................
0x20000020  08 00 00 00 09 00 00 00 0a 00 00 00 0b 00 00 00  ................
0x20000030  0c 00 00 00 0d 00 00 00 0e 00 00 00 0f 00 00 00  ................
0x20000040  10 00 00 00 11 00 00 00 12 00 00 00 13 00 00 00  ................
0x20000050  14 00 00 00 15 00 00 00 16 00 00 00 17 00 00 00  ................
0x20000060  18 00 00 00 19 00 00 00 1a 00 00 00 1b 00 00 00  ................
0x20000070  1c 00 00 00 1d 00 00 00 1e 00 00 00 1f 00 00 00  ................
0x20000080  20 00 00 00 21 00 00 00 22 00 00 00 23 00 00 00   ...!..."...#...
0x20000090  24 00 00 00 25 00 00 00 26 00 00 00 27 00 00 00  $...%...&...'...
0x200000a0  28 00 00 00 29 00 00 00 2a 00 00 00 2b 00 00 00  (...)...*...+...
0x200000b0  2c 00 00 00 2d 00 00 00 2e 00 00 00 2f 00 00 00  ,...-......./...
0x200000c0  30 00 00 00 31 00 00 00 32 00 00 00 33 00 00 00  0...1...2...3...
0x200000d0  34 00 00 00 35 00 00 00 36 00 00 00 37 00 00 00  4...5...6...7...
0x200000e0  38 00 00 00 39 00 00 00 3a 00 00 00 3b 00 00 00  8...9...:...;...
0x200000f0  3c 00 00 00 3d 00 00 00 3e 00 00 00 3f 00 00 00  <...=...>...?...
0x20000100  40 00 00 00 41 00 00 00 42 00 00 00 43 00 00 00  @...A...B...C...
0x20000110  44 00 00 00 45 00 00 00 46 00 00 00 47 00 00 00  D...E...F...G...
0x20000120  48 00 00 00 49 00 00 00 4a 00 00 00 4b 00 00 00  H...I...J...K...
0x20000130  4c 00 00 00 4d 00 00 00 4e 00 00 00 4f 00 00 00  L...M...N...O...
0x20000140  50 00 00 00 51 00 00 00 52 00 00 00 53 00 00 00  P...Q...R...S...
0x20000150  54 00 00 00 55 00 00 00 56 00 00 00 57 00 00 00  T...U...V...W...
0x20000160  58 00 00 00 59 00 00 00 5a 00 00 00 5b 00 00 00  X...Y...Z...[...
0x20000170  5c 00 00 00 5d 00 00 00 5e 00 00 00 5f 00 00 00  \...]...^..._...
```

We, however, did not meet the requirements that we needed on the USB communication side of things. The purpose of the USB stack was to have a method of high-speed communication suitable for streaming video, however, when we tested out our throughput, we discovered that despite using the pins and the correct code from OrbTrace, we were only seeing a throughput of about 1.1KB/s. Which is significantly slower than what is needed for video streaming (On a scale of 10's MBs). We were able to discover that the root cause of this issue stemmed from a misconception in how the OrbTrace code placed the USB stack within the SoC, and instead of communicating using the USB stack, it was communicating using serial over the ULPI pins on the development board. From the moment this issue was discovered, time constraints prevented us from being able to fix the issue before the end of the semester.

```
Read 5712 in 500.19383430480957 ms
Bytes / Second = 11419.572990016515
root@BetterBots-Win:/home/betterbots/orbtrace#
```

ADC Results

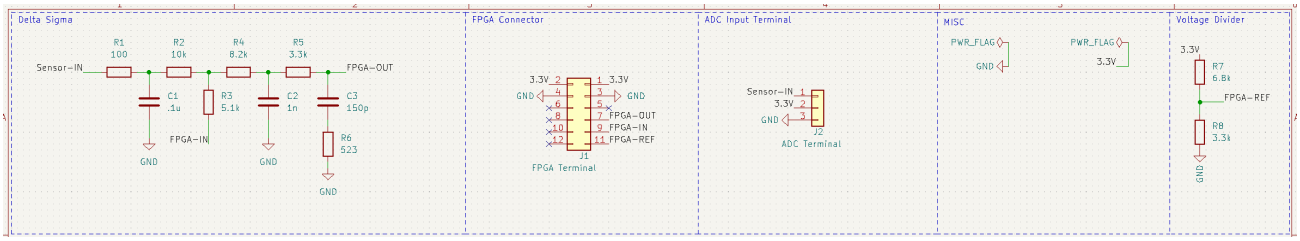
Board Design and Fabrication

We established a strong relationship with the client while designing the implementation boards to ensure that they met their design constraints. However, due to time and resource limitations, we only had the capacity to fabricate two sets of boards. To do this, we developed a final schematic for the ADC and determined the appropriate methodologies for interfacing with the FPGA.

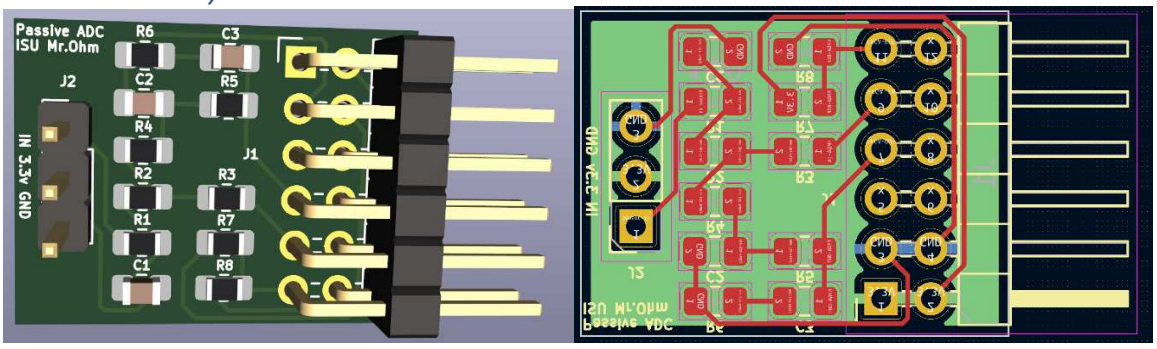
Of the two boards that were built, one for an active circuit and one for a passive circuit, both were green-lit after specifications and proper layout were met. In KiCad, schematics and board layouts were replicated from the ones that were from simulation in LTSPICE and to be fabricated. A data lane connection point between the FPGA and board was ensured. The diagram below highlights several key features:

- FPGA Reference Voltage: A voltage divider was used to set a reference voltage for the FPGA's comparator. This was essential for the FPGA to differentiate between digital 1s and 0s.
- FPGA Connector: A 2x06 male PMOD connector was used to directly connect the ADC with power and data lines.
- Delta Sigma: The final optimized design of the delta-sigma was used, utilizing E-series resistor values.
- Cherry Hopper Amplifier: The final optimized design of the cherry hopper amplifier was used, utilizing E-series resistor values.
- ADC Input Terminal: A 1x03 mail connector was included for easy testing and input of analog signals.

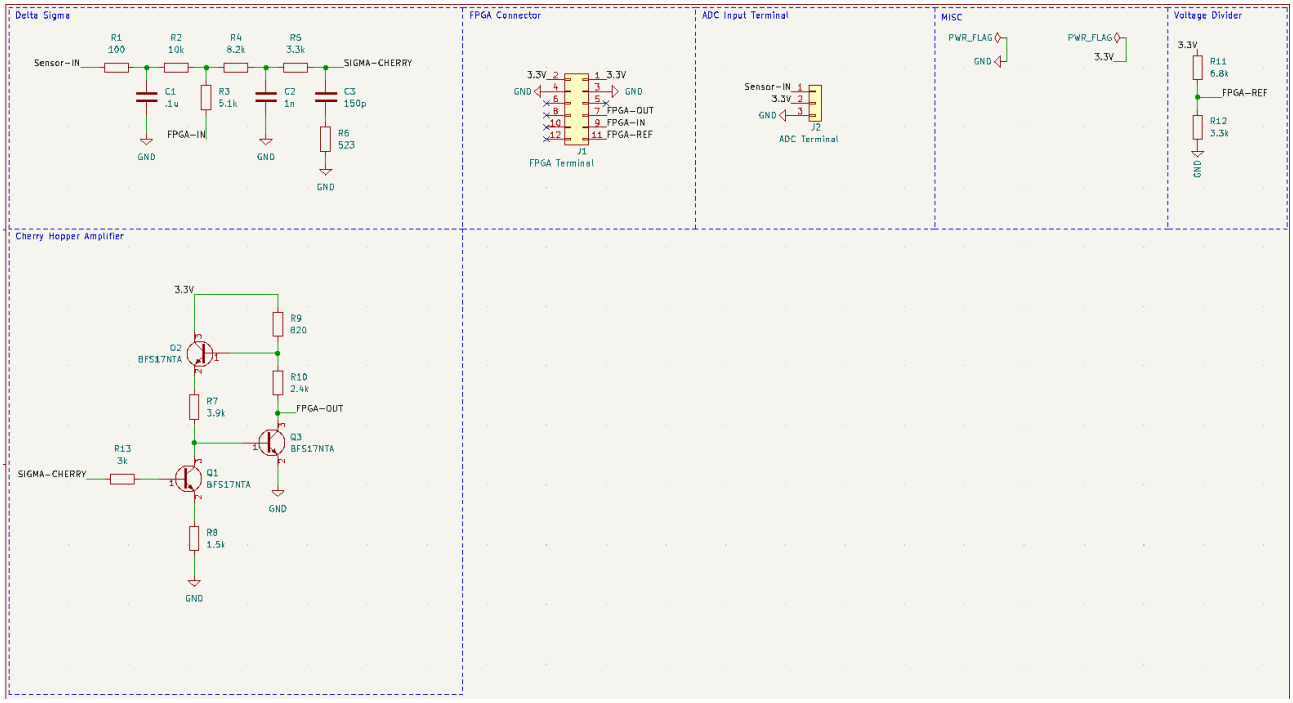
Passive Circuit Schematic



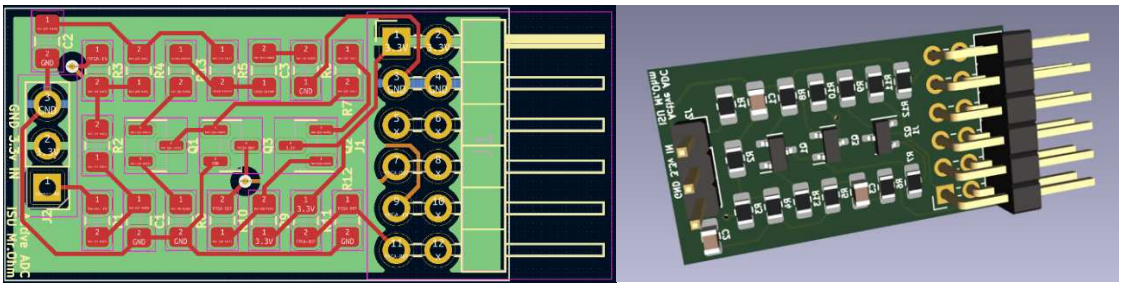
Passive Circuit Layout



Active Circuit Schematic



Active Circuit Layout



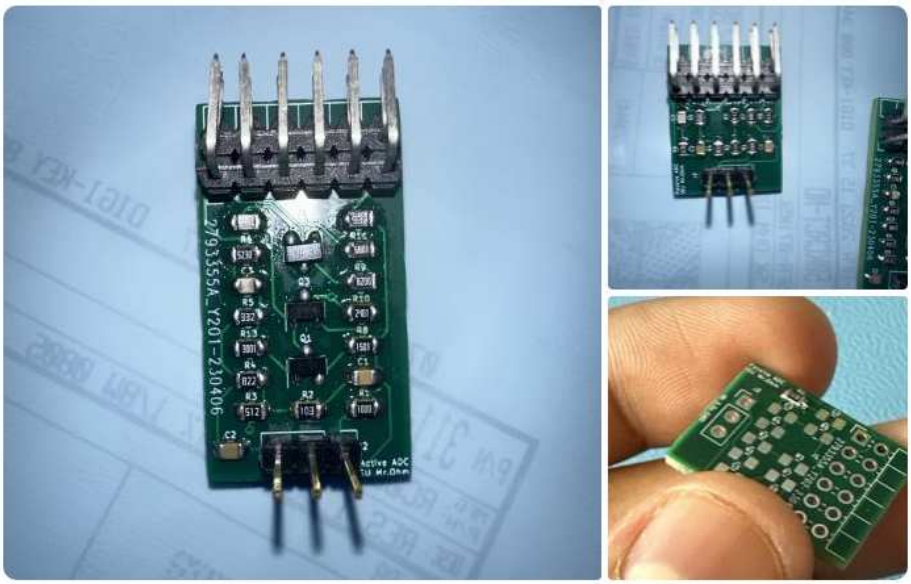
Board Completion

After completing the designs of both boards, Gerber files were generated and sent for fabrication. Upon receiving the boards and parts from the manufacturer, we faced a significant challenge in soldering the components onto the board due to their small size.

To address this challenge, we utilized several tactics, including:

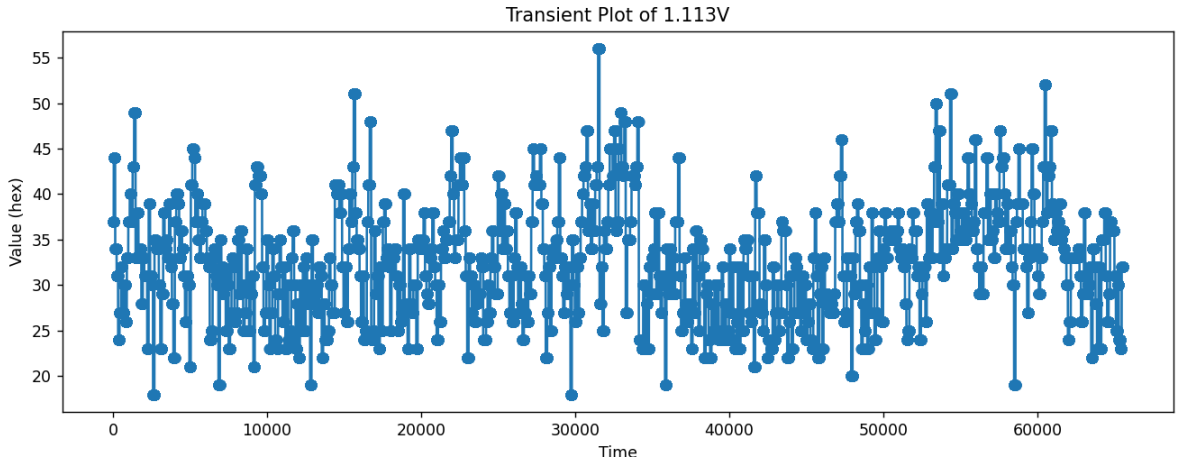
- Extended Component Pads - In the original design, we made sure to increase the size of the component pads to facilitate easier soldering since the components needed to be hand-soldered.
- Solder Paste - Solder paste allowed for easy application of solder onto each pad of the board, eliminating the need for traditional solder wire and other suboptimal soldering methods. This, in turn, allowed for the use of a Reflow oven.
- Reflow Oven - The Reflow oven eliminated the manual process of using a soldering iron to melt the solder paste. This was a crucial step in preventing us from damaging the board components due to less manual intervention.

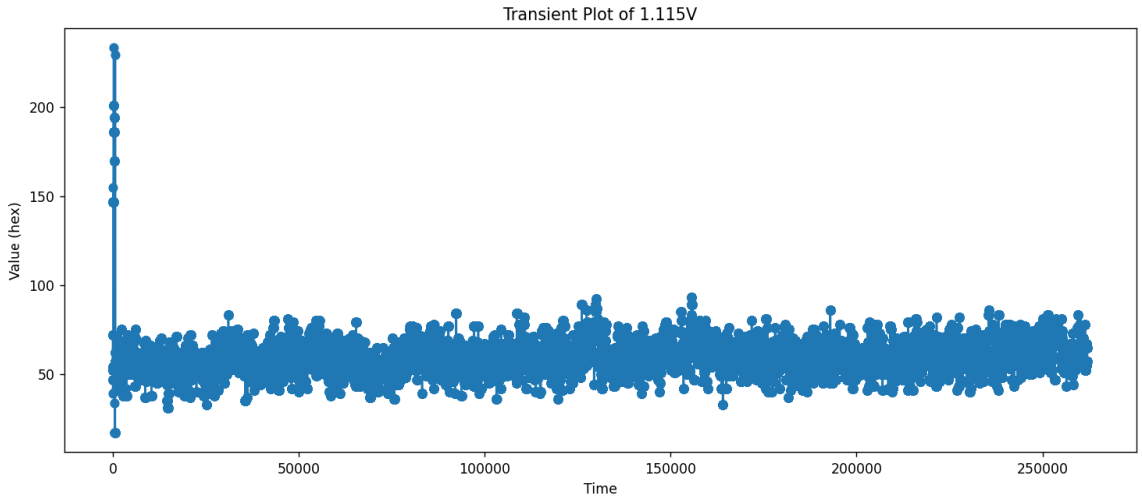
Based on my analysis of the ADC operation, it has been observed that there is a significant amount of variability in the output even when the input is stable. This indicates that the ADC is not performing optimally, as a stable input should ideally result in a stable output. This high variability can be a cause for concern, as it can lead to inaccurate readings and affect the overall performance of the system in which the ADC is being used. Further investigation into the potential causes of this issue is necessary in order to improve the performance of the ADC and ensure reliable and accurate operation.

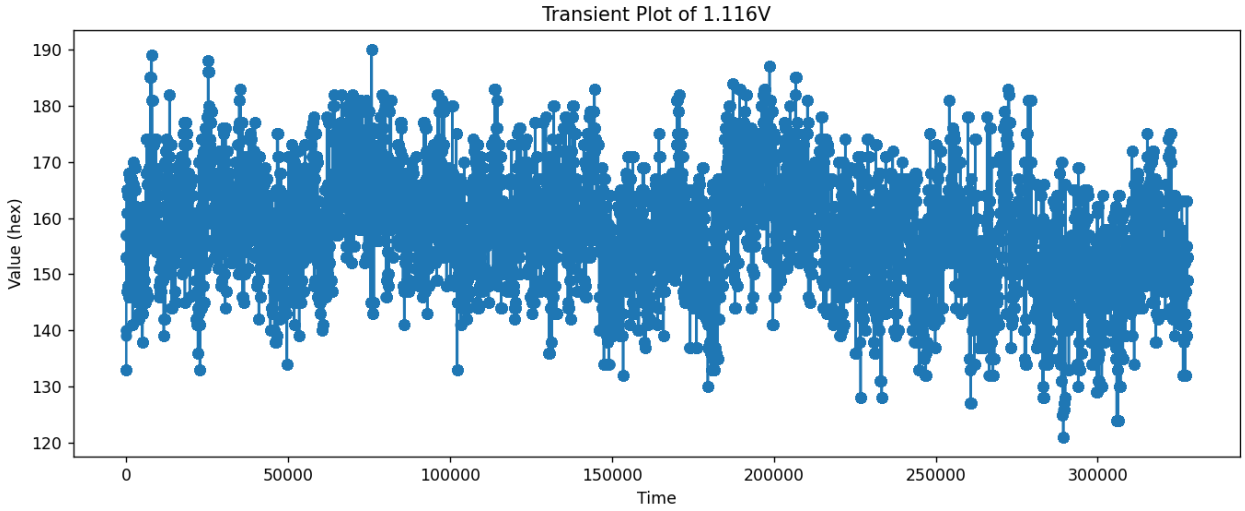


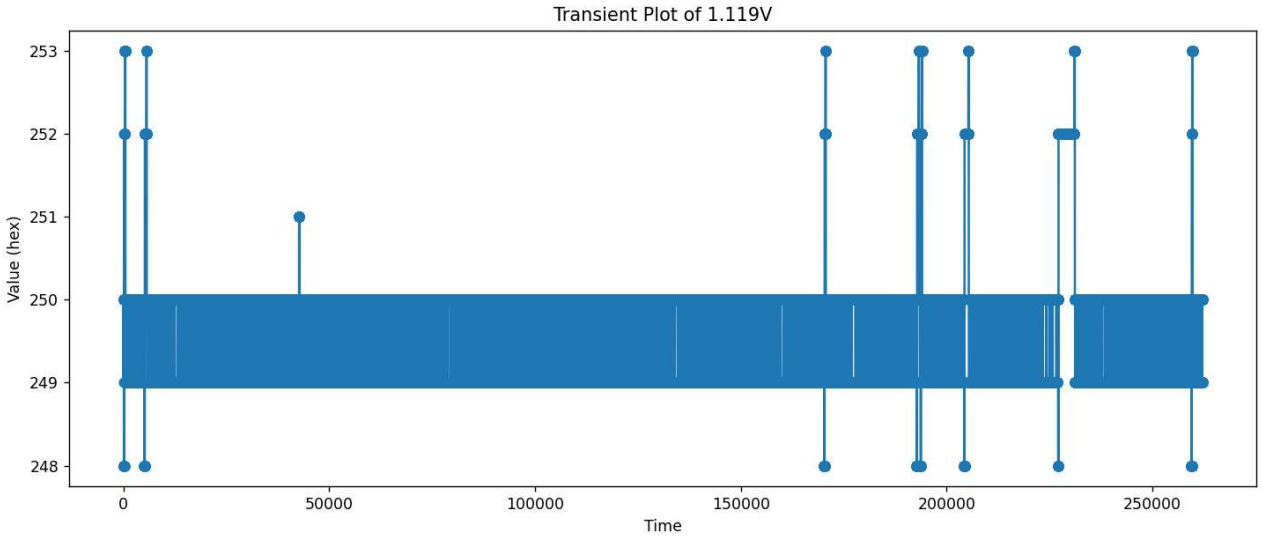
ADC DC stability Tests

On this page we have a plots of the active ADC value over time, given a stable DC input. We determined from these graphs that the average output over time stays relatively consistent, however there is a large spread within the short-term values of the ADC. We would typically see values shooting above and below the average by about +/- 17 units. Since we were using 8 bits of resolution, this would mean that our error $\frac{abs(m_{error} - V_{max})}{V_{max}}$, is approximately 6.67%.





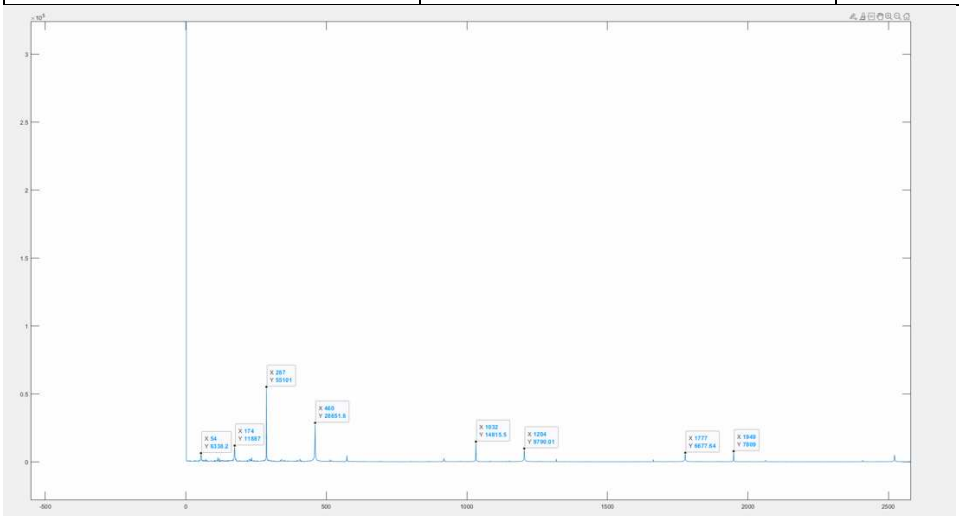

Transient Plot of 1.116V
Value (hex)
Time
190
180
170
160
150
140
130
120
0
50000
100000
150000
200000
250000
300000


Transient Plot of 1.119V
Value (hex)
Time
253
252
251
250
249
248
0
50000
100000
150000
200000
250000

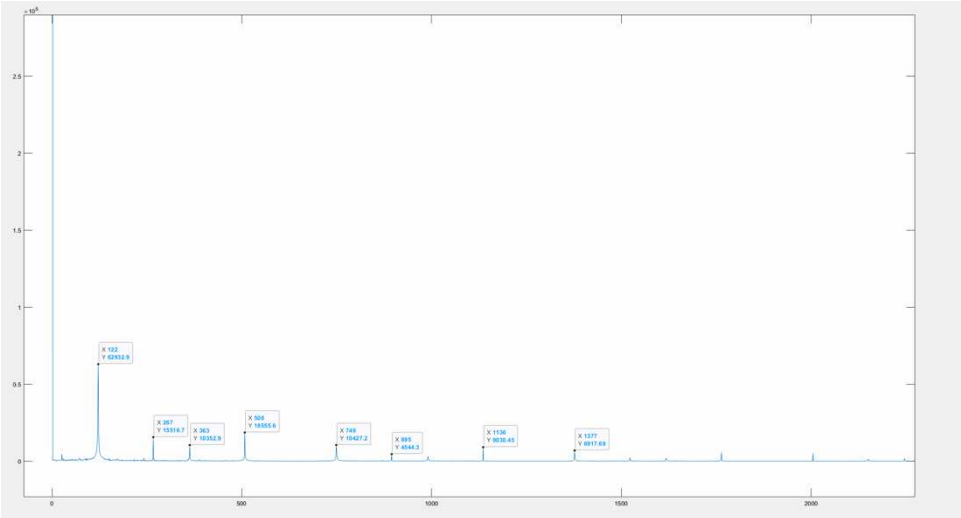
ADC AC Signal Measurements

Below we have three plots showing the FFT on 100,000 values of sine waves of varying frequency and amplitudes. Unfortunately, these graphs do not show a peak at each one's desired frequency. We believe that this is due to the fact the ADC tends to have a lot of noise and overlap between nearby voltages like shown in the previous set of graphs.

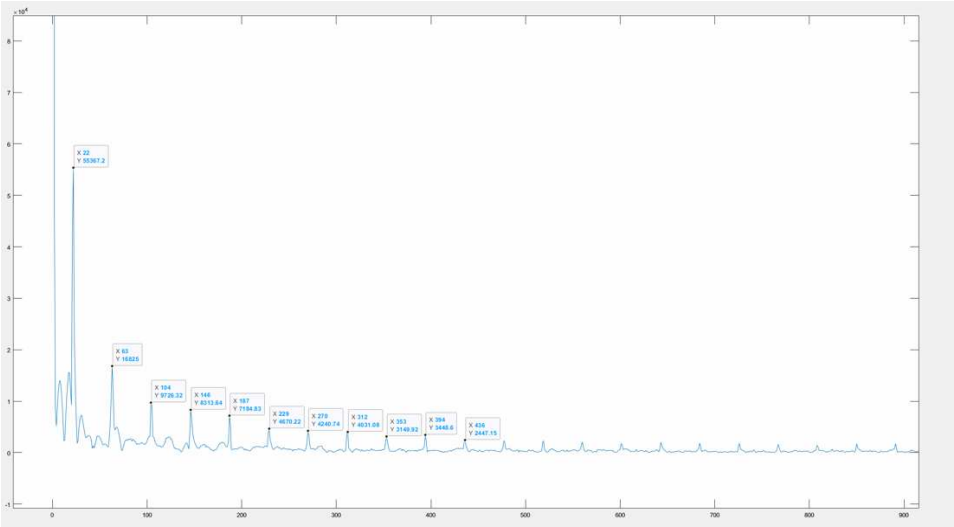
Frequency	Amplitude (Vpp)	DC Offset
2KHz	1.27 V	1.108V



Frequency	Amplitude (Vpp)	DC Offset
1KHz	0.47 V	1.108V



Frequency	Amplitude (Vpp)	DC Offset
60 Hz	0.25 V	1.108V



60Hz

Operation Manual

Overview

The intended use of our device is targeted towards low-level embedded systems programmers, with features that enable in-experienced developers to develop complex multi-sensor applications. In general, to use the device, the user should plug in the ADC of choice into PMOD4 on the development board, and then use generated C header file to find which addresses to query for information. If a program is loaded onto the board directly, it would also be possible to use the header file directly in the application being loaded.

What's needed?

Device	Device Type	Notes
ECPIX-5	FPGA Dev Board	45F Lut model was used, but 85F would be compatible
Active/Passive ADC Plugin	Custom PCB	This will need to be hand assembled, relevant KiCad files provided
LiteX	Software Library	Found at - https://github.com/enjoy-digital/litex.git
Development Files	Software Library	Found at - https://github.com/rajsinghtech/Mr-Ohm-IowaState.git

ADC Operation

To operate or demo the ADC portion of this project, a few things will be needed. Starting with the passive circuit or also known as the actual ADC. You will need a power supply that can supply 3.3V in order to power the system. This would just be to power the system; in our case the ADC works in conjunction with the FPGA so that supplied our power. You will also need a function/waveform generator responsible for generating an input signal. This could be a ramp signal to test a range of values the ADC could convert to digital values. As well as a sine wave could work in this case. In our instance these inputs going into the ADC would be from other components of the robot such as the microphone array. For this to work proper entirely you would need a means of providing a digital feedback signal that decimates it. That is why something like an FPGA that has a comparator used for analog to digital conversion can provide that feedback.

For the active circuit things work essentially the same, where you would need the power supply of 3.3V. A device like the waveform generator that provides an input signal if you do not already have a device that provides an analog signal. Lastly, a way to provide digital feedback to the ADC portion of the circuit. This circuit behaves the same as the passive one besides the fact that it amplifies the analog input signal.

FPGA Operation

To ensure consistency across various setups and dependencies, we will use the provided docker image available on GitHub [Add Link] for all usage and imaging operations. This docker image is preloaded with open-source OSS-CAD-SUITE [Link], a LiteX installation, and a modified version of the OrbTrace library. To image, navigate to the modified OrbTrace project directory by entering cd ~/orbtrace in the terminal. Once in the directory, execute the command ./orbtrace orbtrace_builder.py -

-device 45F –platform ecpix5 –load –build –uart-name stream. The "--build" flag can be omitted if the design has already been built.

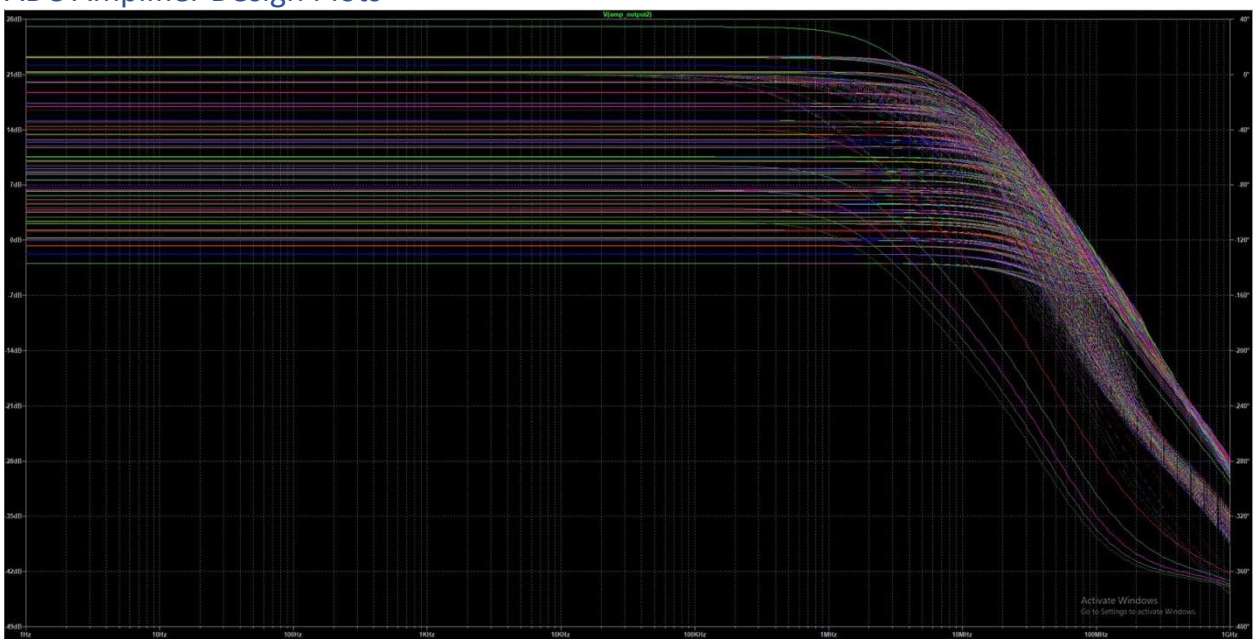
To interact with the SOC now running on the FPGA, run litex_term {serial port} from any directory. For instance, /dev/ttyUSB1 is an example serial port. You will be able to access the device's individual registers and memory from this point. Examine the generated header file from the build instructions to determine the CSR mapping.

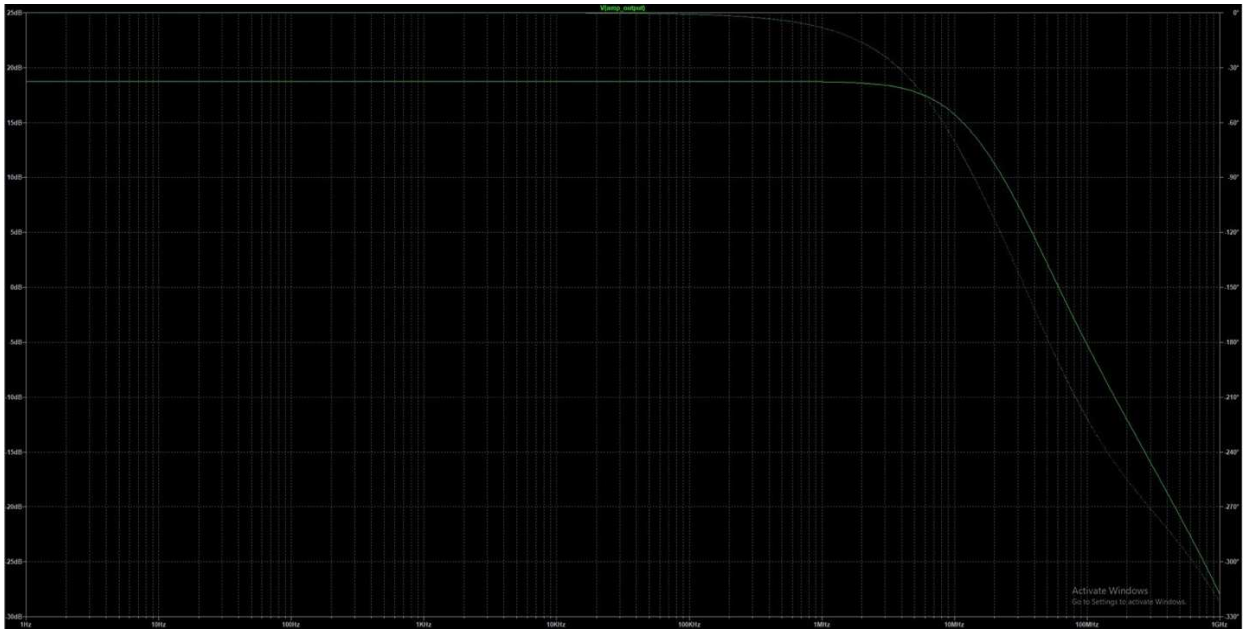
Relevant Links

- https://github.com/enjoy-digital/litex
- https://github.com/butterstick-fpga/butterstick-hardware
- https://github.com/orbcode/orbtrace
- https://www.betterbots.com/
- https://github.com/YosysHQ/oss-cad-suite-build/releases

Appendix

ADC Amplifier Design Plots





Work Cited

ADC Academic Paper #1: A. Roy and R. J. Baker, "A passive 2nd-order sigma-delta modulator for low-power analog-to-digital conversion," 2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS), College Station, TX, USA, 2014, pp. 595-598, doi: 10.1109/MWSCAS.2014.6908485.

ADC Academic Paper #2: [1]

D. Piši, “FPGA BASED DELTA-SIGMA ANALOG TO DIGITAL CONVERTER.” Accessed: Apr. 28, 2023. [Online]. Available: https://www.fekt.vut.cz/conf/EEICT/archiv/sborniky/EEICT_2012_sbornik/03doktorskeprojekty/02zpracovanisignaluobrazuadat/04-xpisid00.pdf