Amplitude Modulation Continuous Wave (AMCW) for Length Measurements

Mariana Reyes Holguín
EECS
MIT
Cambridge, USA
mareyesh@mit.edu

2nd Juan Luera
EECS and Physics
MIT
Cambridge, USA
j_luera@mit.edu

3rd Erin Zhang
EECS
MIT
Cambridge, USA
ewzhang@mit.edu

***Abstract*—This project focuses on implementing, Amplitude Modulated Continuous Wave (AMCW) a phase calculation method used to estimate length in LiDAR. This method works by sine-wave modulating a laser diode and measuring the phase shift of the reflected wave with respect to the reference wave by utlizing IQ demodulation and CORDIC. This work was able to successfully measure the length of 3 different wires.**

I. Background

A. LiDAR

Light Detection and Ranging (LiDAR) is a remote sensing technology that uses lasers to measure distance and sometimes velocity. It is frequently used to render 3D mappings of spaces that reflect the topology. The general principle is a laser is collimated and shone at an object, the time it takes for the laser to reflect off of the object and return to a photodiode is correlated to the distance traveled. There are various types of LiDAR, based on the laser type, goal, and phase measurement strategy. This project focuses on implementing the the phase calculation method of one of the LiDAR methods.

B. Selection of Strategy

a) Time-of-Flight (ToF): ToF LiDAR operates by sending short laser pulses and measuring the time it takes for the signal to return, calculating distance as $D = \frac{c \cdot t}{2}$. It is a well-established technology offering high resolution and long-range capabilities. The discrete return and full waveform variants provide versatility, with the latter delivering rich detail through backscattered signal analysis. However, ToF demands high-precision timing circuits for accurate measurements and is sensitive to noise and ambient light interference. Additionally, if made a full-waveform implementation, the processing results in significant data volume, making it more suitable for applications with sufficient computational resources.

b) Amplitude Modulation Continuous Wave (AMCW) LiDAR: This method continuously emits an amplitude-modulated wave and calculates the phase difference ($\Delta\phi$) between the emitted and received signals. Phase-Shift LiDAR achieves high resolution within its unambiguous range and requires simpler hardware compared to ToF. Nonetheless, its range is constrained by the modulating frequency ($D \propto \frac{1}{f_m}$) and is less effective in complex scenes involving multiple reflective objects due to potential signal overlap.

c) Frequency Modulated Continuous Wave (FMCW): FMCW employs a frequency-chirped laser and measures the beat frequency resulting from mixing emitted and received signals. This approach can simultaneously measure both distance and velocity and handles multiple reflections effectively by distinguishing beat frequencies. While FMCW offers high precision and resolution, it requires a laser capable of fast and accurate frequency modulation. Its implementation is more complex and requires advanced components, but its ability to manage overlapping distances makes it advantageous for intricate environments.

After comparing the technologies, we decided to go with a AMCW based approach. ToF might have been suitable due to the lower complexity, but the standard of quality of the hardware would put a hard constraint on us, which is simply not necessary. FMCW sounded very interesting, but more complex than what is suited for a proof-of-concept project, and does not necessarily higher performance for our purposes. AMCW LiDAR does have a range limitation and cannot discern multiple reflections well, but given this is a proof-of-concept project, we believe that is an acceptable trade-off.

II. Electronics System Design

This section elaborates on the process behind the electronics to implement a distance measurement LiDAR. We were not able to integrate the electronics with the digital design in time due to a sizing error in the board. However, the value of this section lies in the potential for this project to be expanded to perform an optic's based distance measurement.

A. Amplitude Modulated Continuous Wave (AMCW)

In phase shift LiDAR, we can calculate the distance based on the following equation: $D = \frac{\Phi * c}{4 * pi * f_m}$. It is important to select a modulating frequency that both enable sufficient range and does not limit resolution. One of the important metrics in this method is the maximum unambiguous range that can be measured using a particular modulating frequency. This value is determined by the equation $D_{max} = c/(2 * f_m)$ where c is the speed of light and f_m is the modulation frequency. Resolution can be found from the distance equation when reformulated: $\Delta D = \frac{\Delta\Phi * c}{4 * pi * f_m}$. This means our resolution depends on both the modulation frequency and the phase shift increment/resolution permitted by the ADC and oscillator.

Selecting the right frequencies is essential for a good performance. Based on the equations above, it is clear that a lower modulation frequency achieves a higher unambiguous range, however, it sacrifices precision. A higher modulation frequency achieves higher resolution, but sacrifices unambiguous range. Given this information and an in-depth surveying of parts online to determine reasonably priced and easily delivered options, we selected the range of 10MHz to 100MHz for the f_m. [1]

Unambiguous range:

- $D_{max} = c/(2 * 10 * 10^6) = 15$m
- $D_{max} = c/(2 * 100 * 10^6) = 1.5$m

Resolution:

- $\Delta\Phi = \frac{2*pi}{2^B} = \frac{2*pi}{2^14}$ 0.000384 radians
- $\Delta D = \frac{\Delta\Phi * c}{4*pi*10*10^6} = 0.916$ mm
- $\Delta D = \frac{\Delta\Phi * c}{4*pi*100*10^6} = 91.6$ µm

where $\Delta\Phi$ is the base change phase increment, B is the number of output bits, c is the speed of light, and ΔD is distance resolution. These are the bottlenecks of the hardware, but just as relevant are the limits of the digital logic implementation, which come from DAC and ADC rates, fixed point errors, and more.

B. Selecting parts for AMCW LiDAR

There were several considerations when selecting the parts for the physical implementation:

a) Laser Diode: :

- The wavelength needed to be in the visible spectrum due to safety regulations, we picked 650nm.
- Must be made for continuous wave outputs.
- A fast rise time and a high bandwidth to meet modulation capabilities, a couple of nanoseconds was our target.

b) Photodiode: :

- Responsivity at 650nm should be as high as possible to enhance SNR, we were aiming for at least 0.4 A/W.
- The rise time of the photodiode limits the detectable frequencies. We were aiming for a rise time below 5ns, however, only below 10ns was feasible in purchase.
- Low noise and low dark current photodiodes are highly preferable to improve the SNR.

c) Op-amp: :

- A high gain bandwidth (GBW) and low rise time op-amp is necessary to permit the high frequencies to be preserved when the returning signal is amplified.
- Low voltage and current noise is crucial in such an application to avoid excessive issues when amplifying.

[1]The lower bound bound was also influenced by the fact that the DACs and ADCs in the RFSoC don't perform well at low frequencies.

d) Collimator: :

- focal length $= fl = \dfrac{\phi_\perp/2}{\tan(\theta_\perp/2)}$ where $\phi_\perp$ is the desired perpendicular beam diameter and $\theta_\perp$ is the perpendicular beam divergence as specified by the diode datasheet.
- A large desired beam diameter reduces divergence
- Focal length for a 3mm beam diameter for our diode would be somewhere between 5mm and 7mm
- The aspheric lens diameter must be larger than the mayor axis, which is the greater of the two radii, to collimate.
- $NA_{lens} > NA_{diode}$ $sin(\dfrac{\theta_\perp}{2}) = sin(\frac{28}{2}) = 0.242$ but in the range of 0.199 to $0.275 \Rightarrow NA_{lens} > 0.275$

C. Designing the transmitter

The transmitter was responsible for converting a voltage from the DAC (digital-to-analog converter) to a proportional current for the laser diode. The DAC output range wasn't entirely quantified, but was in the range of 0 V to 2 V. The desired current was supposed to stay within the linear operating region of the MOSFET, so an output current of 0 mA to 100 mA was reasonable. The current for this circuit is regulated through feedback from a low-side current-sense resistor into the inverting input of the op-amp, a voltage to current converter.

A resistor at the output of the op-amp helps to isolate the op-amp from the MOSFET gate and proportion the current flowing into the MOSFET. Feedback components maintain output stability, reducing any noise from interference. Values from very similar specifications in a Texas Instruments document mostly translated over, except for our pre-set MOSFET choice.

- Transfer function: $I_{out} = V_i/R_3$
- $R_3 = \dfrac{V_{iMax} - V_{iMin}}{I_{oMax} - I_{oMin}} = \dfrac{2V}{100mA} = 20\Omega$
- P_{max} through R_3: $P_{R3} = \frac{V_{iMax}^2}{R_3} = \frac{2}{20} = 0.2W$

Thus, typical 0.25W resistors should suffice.

Unfortunately, without any switching in the circuit, the MOSFET in the "on" mode and the current draw from the laser diode was much higher than expected (around 50 mA at some point) in an attempt to run it from 3.3 V instead of the recommended 2.6 V maximum. This was due to an oversight in designing the PCB before fully choosing the laser diode. Still, the circuit functioned to turn on the laser diode and emit light, but the MOSFET burned out before any modulation was possible.

D. Designing the receiver

The receiver was responsible for picking up the light signal through a photodiode that produced a current proportional to the light intensity. Since this light is modulated into a wave by the transmitter, the photodiode allows a sensor further away to pick up on this same signal. This current then needed to be converted into a voltage proportional to the current for input into the ADC (analog-to-digital converter). We selected a transimpedance amplifier for this purpose. The schematic for this is shown below:

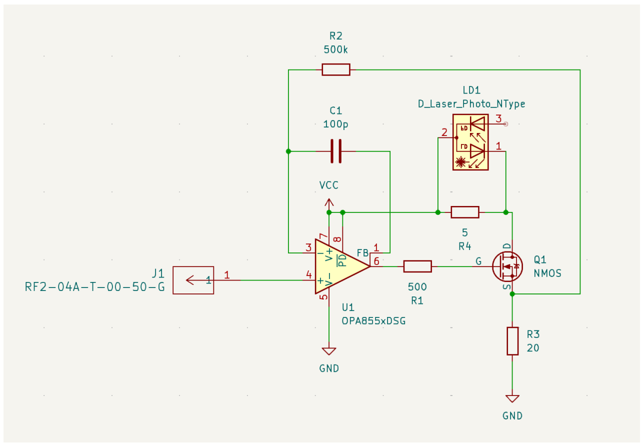


Fig. 1. Transmitter

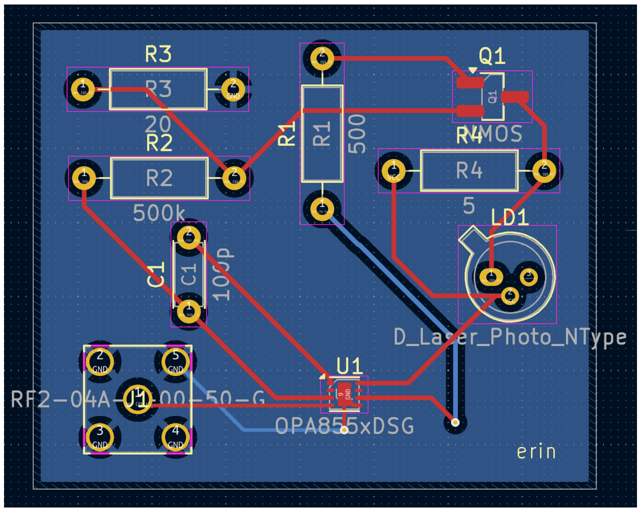


Fig. 2. Resulting PCB design

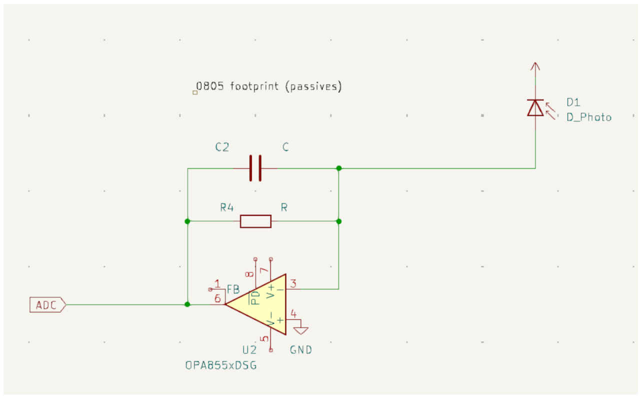


Fig. 3. Receiver

Once again, the op-amp had to have a high gain bandwidth with little noise, and unfortunately due to time constraints after selecting the parts, there was no PCB for the receiver and we were relegated to using through-hole mount parts and pre-milled boards. The op-amp that seemed to best meet our requirements ended up being the AD787, as our initial selection and order was for a surface mount part.

Calculations:

Selecting the gain resistor (R_4 in the schematic shown):

$$R = \frac{V_{oMax} - V_{oMin}}{I_{iMax}} = \frac{1V - 0V}{0.002A} = 500\Omega \qquad (1)$$

The maximum output voltage and input current values come from what we know about the RFSoC we use (for voltage) and the below calculation:

The photodiode as a responsivity of 0.4A/W and the maximum output of the laser is 5mW, so the absolute maximum current output from the photodiode will be 0.002A (not even accounting for the quadratic light intensity decay over distance).

Maximum filtering capacitor based on bandwidth:

$$C \leq \frac{1}{2\pi Rf} \leq \frac{1}{2\pi 500(11.7MHz)} \leq 27.2pF = 22pF \qquad (2)$$

using standard capacitor values. This same calculation would require a capacitor under 7.8pF for the 40.5MHz signal.

Required op-amp GBW (gain bandwidth):

$$GBW > \frac{C_{in} + C}{2\pi RC^2} > \frac{5pF + 22pF}{2\pi(500)(22pF)^2} > 17MHz \qquad (3)$$

Note that the input capacitance value of 5pF comes from the datasheet. The 110MHz GBW of the AD797 is more than sufficient in this case.

III. Digital System Design

On a high level, the system consists of a DAC and ADC module, a Direct Digital Synthesis block (DDS), a demodulator, and a CORDIC implementation of $arctan(x)$. We implemented a lot these ourselves to allow for customization for the case of LiDAR, where we also needed the oscillators to drive the laser diode.

The general flow of our system is described here. One of the DDS modules produces a sine wave to feed to the DAC. The other DDS module produces both a sine wave and a cosine wave at almost exactly the same frequency as the first DDS module. These act as the reference wave, because they are assumed to operate almost perfectly in-phase. The DAC's input is physically routed to the ADC by a wire, such that the output of the ADC is a phase-shifted version of the original wave. The ADC's output is routed to the demodulator along with the reference wave, where they are mixed and filtered. In this case, the reference wave acts as the local oscillator in the demodulator, such that the output I/Q data encodes the phase difference between the reference wave and the input wave, which is our target measurement. The top 16 bits of the filtered output are then routed to a CORDIC $arctan(x)$ implementation which then produces and angle that is routed via a FIFO and DMA to the Processing System.

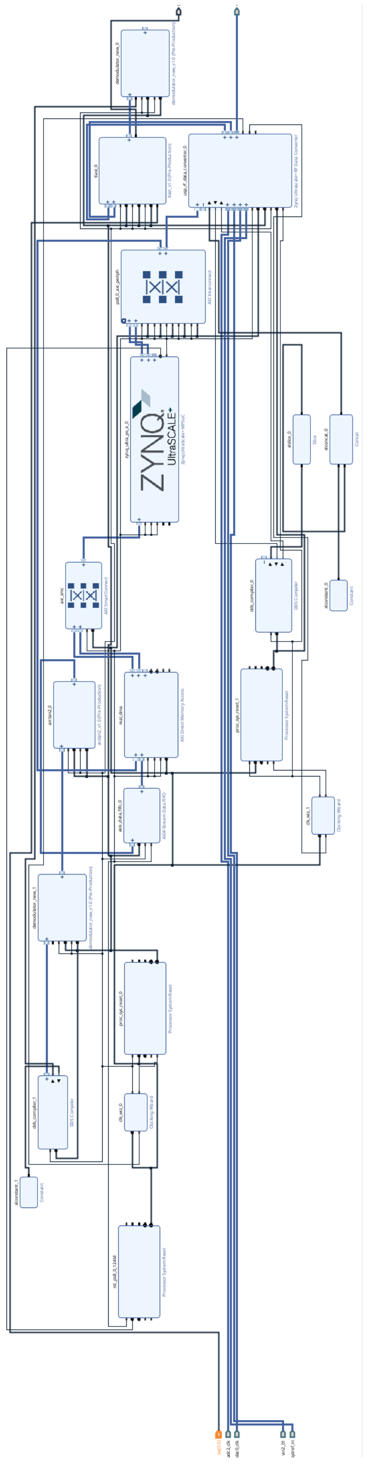


Fig. 4. Schematic of Vivado's Block Design

A. DDS

Direct digital synthesis is a method used by frequency synthesizers for creating waveforms from a reference clock. Vivado provides a highly capable DDS compiler IP which can produce both sine waves and cosine waves, can be configured for frequency modulation, allows for customization of bit width, phase granularity, and frequency. These properties made it a more suitable choice over a self-made look-up table step through.

The way the Phase Increment and Sin/Cos LUT DDS option works is it increments the phase periodically by a value that can either be fixed in the settings or fed into the AXI slave port. The frequency of the output sine wave depends on the reference clock f_{clk}, the bit width of the phase increment B, and the phase increment $\Delta\Phi$.

$$f_{out} = \Delta\Phi * \frac{f_{clk}}{2^B} \tag{4}$$

Given that our DAC and ADC were running on different clocks, 98.304 MHz and 147.456 MHz respectively, and we needed to mix the input of the DAC with the output of the ADC, we had to resort to making two DDS blocks. This meant we had two DDS blocks running on different clocks, attempting to produce the same frequency sine wave. This was achieved by plotting the equation above for both clock rates to find output frequencies within our desired range, that each DDS block can produce with a 16-bit integer phase increment value, and whose output frequencies minimally differ. We chose 11.7 MHz and 40.5 MHz. The difference was very minimal, in the order of 10^{-9}, so while the impact should not be very significant, this is an additional source of error that could limit the resolution of our system.

B. Demodulator: FIR and Fixed Point

We implemented a demodulator composed of a mixer and an FIR. The ADC's output is the signal into the demodulator and the reference wave acts as the local oscillator (LO) that it's mixed with.

These are the inputs to the demodulator, the wave in is a phase shifted version of the LO:

$$S_{in} = A * sin(\omega * t + \varphi) LO = A * sin(\omega * t) \tag{5}$$

These are mixed with the LO and the LO shifted 90°:

$$I = A*sin(\omega*t+\varphi)*A*sin(\omega*t), Q = A*sin(\omega*t+\varphi)*A*cos(\omega*t) \tag{6}$$

Simplifying with Ptolemy's Theorem:

$$I = \frac{A^2}{2}*(cos(\varphi)-cos(2*\omega*t+\varphi), Q = \frac{A^2}{2}*(sin(\varphi)-sin(2*\omega*t+\varphi) \tag{7}$$

After low-pass filtering with an FIR with a 10MHz cutoff frequency, the high frequency term is discarded:

$$I = \frac{A^2}{2} * cos(\varphi), Q = \frac{A^2}{2} * sin(\varphi) \tag{8}$$

The angle in the complex plane of this I/Q vector then has angle:

$$\varphi = \arctan(\frac{Q}{I}) = \arctan(\frac{\frac{A^2}{2} * sin(\varphi)}{\frac{A^2}{2} * cos(\varphi)}) = \varphi \qquad (9)$$

This is the phase difference between the LO and the input wave.

One of the key sources of error in our system was the overflow caused from the various arithmetic operations in the demodulator. The mixer starts with two 14-bit inputs, resulting in a 28 bit signal. This one is then fed into the 21 8-bit coefficients FIR, which performs a series of multiply and accumulate operations. Each of these adds to the bit width leading to a worst case of 39 bits. (see appendix, 3)

C. *CORDIC (arctan)*

For the arctan2 implementation we made a cordic using a Q1.15 lut table. We opted to write our own code rather than using an IP as it gave us more flexibility in how to manage the bits and bitflow. Furthermore, we wanted to add some complexity to our project as we felt that it didn't have enough verilog. First we wrote the code in python and tested cordic implementations. Once we understood the code we wrote it in system verilog and test benched it in Cocotb. (see appendix, 2)

IV. Results

As a proof of concept we attempted to measure the length of 3 different wires of lengths 15 cm, 50 cm, and 100 cm respectively. We used the relative phase to measure the length difference as a test of accuracy. We also tested two different frequencies, the idea behind this being that a higher frequency would provide greater resolution and a lower frequency provides range. In theory, our resolution for a 11.7 MHz frequency should've been quite small, almost negligible, but in practice it was around -/+ 5 cm for a theoretical range of 15 meters. As mentioned previously, we believe this noise was introduce by fixed point calculations and the FIR filter we implemented.

A. *Measurements at 11.7 MHz*

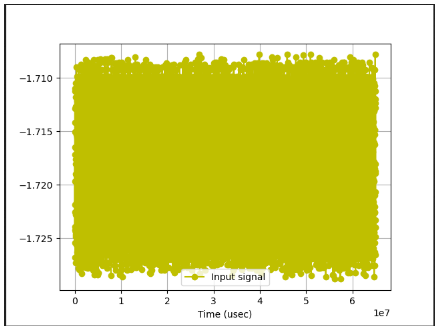


Fig. 5. Phase measurements for 15 cm coaxial cable at 11.7 MHz

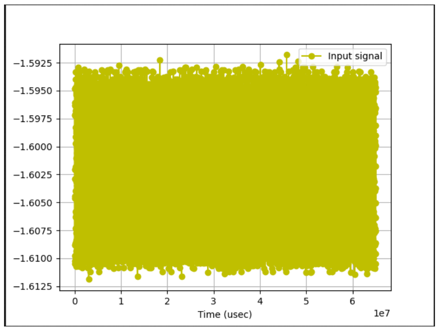


Fig. 6. Phase measurements for 50 cm coaxial cable at 11.7 MHz

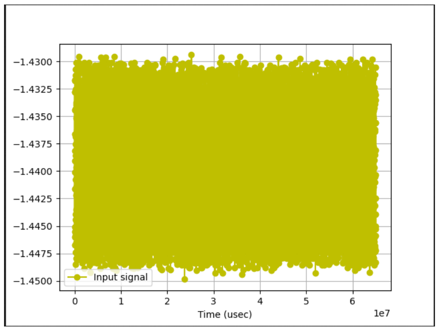
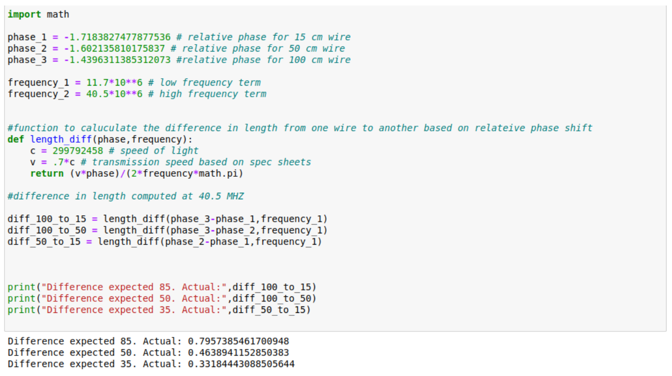


Fig. 7. Phase measurements for 100 cm coaxial cable at 11.7 MHz

For most measurements, we found around a +/- .015 radian error. This was far from desirable, so we implemented some averaging for our final estimates. This helped with accuracy and helped us get better results. While, we were a few cms of most times the actual measurements were always within the error boundaries, which was promising.

B. *Measurements at 40.5 MHz*

As a means to get better measurements, we used a second frequency for our length estimations. While, none of the wires we had were long enough for length estimate correction strategies, as our longest wire of 1 m was still well enough within the maximum non-ambiguous range of a 40.5 MHz signal, in theory a higher frequency signal should have a better resolution.



```
import math

phase_1 = -1.7183827477877536 # relative phase for 15 cm wire
phase_2 = -1.602135810175837 # relative phase for 50 cm wire
phase_3 = -1.4396311385312073 #relative phase for 100 cm wire

frequency_1 = 11.7*10**6 # low frequency term
frequency_2 = 40.5*10**6 # high frequency term


#function to caluculate the difference in length from one wire to another based on relateive phase shift
def length_diff(phase,frequency):
    c = 299792458 # speed of light
    v = .7*c # transmission speed based on spec sheets
    return (v*phase)/(2*frequency*math.pi)

#difference in length computed at 40.5 MHZ

diff_100_to_15 = length_diff(phase_3-phase_1,frequency_1)
diff_100_to_50 = length_diff(phase_3-phase_2,frequency_1)
diff_50_to_15 = length_diff(phase_2-phase_1,frequency_1)



print("Difference expected 85. Actual:",diff_100_to_15)
print("Difference expected 50. Actual:",diff_100_to_50)
print("Difference expected 35. Actual:",diff_50_to_15)
```

```
Difference expected 85. Actual: 0.7957385461700948
Difference expected 50. Actual: 0.4638941152850383
Difference expected 35. Actual: 0.33184443088505644
```

Fig. 8. Length measurements at 11.7 MHz sine wave modulation

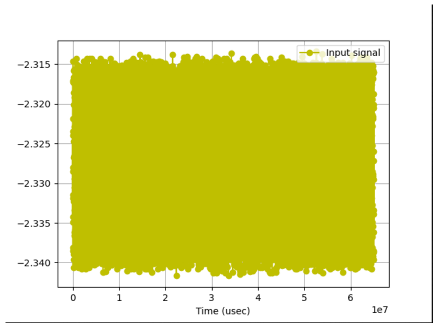


Fig. 9. Phase measurements for 15 cm coaxial cable at 40.5 MHz

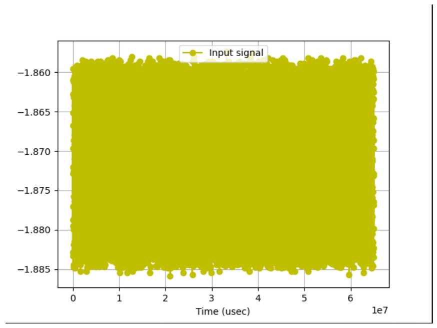


Fig. 10. Phase measurements for 50 cm coaxial cable at 40.5 MHz

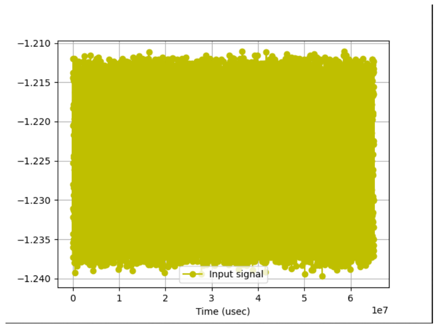
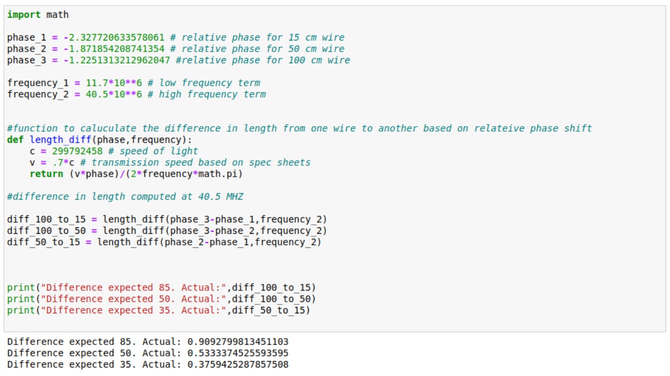


Fig. 11. Phase measurements for 100 cm coaxial cable at 40.5 MHz



```
import math

phase_1 = -2.327720633578061 # relative phase for 15 cm wire
phase_2 = -1.871854208741354 # relative phase for 50 cm wire
phase_3 = -1.2251313212962047 #relative phase for 100 cm wire

frequency_1 = 11.7*10**6 # low frequency term
frequency_2 = 40.5*10**6 # high frequency term


#function to caluculate the difference in length from one wire to another based on relateive phase shift
def length_diff(phase,frequency):
    c = 299792458 # speed of light
    v = .7*c # transmission speed based on spec sheets
    return (v*phase)/(2*frequency*math.pi)

#difference in length computed at 40.5 MHZ

diff_100_to_15 = length_diff(phase_3-phase_1,frequency_2)
diff_100_to_50 = length_diff(phase_3-phase_2,frequency_2)
diff_50_to_15 = length_diff(phase_2-phase_1,frequency_2)



print("Difference expected 85. Actual:",diff_100_to_15)
print("Difference expected 50. Actual:",diff_100_to_50)
print("Difference expected 35. Actual:",diff_50_to_15)
```

```
Difference expected 85. Actual: 0.9092799813451103
Difference expected 50. Actual: 0.5333374525593595
Difference expected 35. Actual: 0.3759425287857508
```

Fig. 12. Length measurements at 40.5 MHz sine wave modulation

After implementation we found marginal improvements of around 0.5 cm. Admittedly, this was to be expected due to the amount of noise that was introduced into our circuit. However, the fact we saw improvements shows promise that with a cleaner implementation we could have very accurate measurements. Its important to note, we were severely limited by the switching rate of the photodiode which is one of the reasons our high frequency wasn't higher. If we had anticipated that our switching FET would burn we could've tested higher frequencies and potentially gotten better results.

V. Conclusion

This work explored the implementation of AMCW and provided starting guidance on how this system could be expanded to a small LiDAR system used for measuring distances. While its hard to generalize our results due to the large margin in error, we believe this implementation has a future in high speed LiDAR implementation. By leveraging the AXIS4 Stream protocol, the parallelization provided by FPGA's, and the ultra-fast rf blocks in the rfsoc, we were able to create a system that was able to measure a real-time phase shift at clock speeds (throughput).

VI. Future Work

The main thing we'd like to improve on is our accuracy. One idea we had was re-implementing our FIRs and using Vivado's instead. Intially we liked the flexibility of writing our own AXIS protocols, this soon proved to be a fatal-flaw as it introduced many flaws that made us waste considerable time. We also propose a swith to a strategy that is more dependent on automatic calibration than a need for components to match clock rates and phases. This would consist of using the built-in mixers of the ADC and DAC, along with a DDS module, knowing that their phases with respect to each other might vary, and then calibrating by sending, receiving, and comparing an expected measurement; this is a more robust approach. Another direction for improvement is iterating on the hardware. Unfortunately, despite our research, we made sizing errors in the hardware. With more time we'd iterate on the design and hopefully have a working module.

References

[1] Instruments, Texas. "Https://Www.Ti.Com/Lit/an/Sboa327a/Sboa327a.Pdf?Ts=1732651
." TI Incorporated, Aug. 2021.
[2] Instruments, Texas. LIDAR-Pulsed Time-of-Flight Reference Design Using High-Speed Data Converters, TI Incorporated, Dallas, Texas, 2017.
[3] Instruments, Texas. Voltage-to-Current (V-I) Converter Circuit with MOSFET, Aug. 2021.
[4] Kuisma, Mikael. "I/Q Data for Dummies." I/Q Data for Dummies, Mar. 2023, whiteboard.ping.se/SDR/IQ.
[5] Paschotta, Dr. Rudiger. "Laser Diodes." – Semiconductor, Gain, Index Guiding, High Power, RP Photonics AG, 10 Dec. 2024, www.rp-photonics.com/laser_diodes.html.
[6] Yan, Peijia. "High Precision Hybrid Pulse and Phase-Shift Laser Ranging System ." Concordia University, Oct. 2018.

Appendix

1) For demonstration: Watch on YouTube
2) For relevant code, click here: GitHub Repository
3) 39 bit worst case isn't neccesarily true in practice. We found that the FIR in general added less bits than expected. Furthermore, due to signedness the worst case was closer to 35 bits, but in practice it wouldn't always overflow. However, it wasn't precise enough and introduced undesired noise.