Phased Array Digital Beamforming Algorithms and Applications

David Moyle Marsh
Brigham Young University

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/etd/7499

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Phased Array Digital Beamforming

Algorithms and Applications

David Moyle Marsh

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Karl F. Warnick, Chair
David G. Long
Brian Jeffs

Department of Electrical and Computer Engineering
Brigham Young University

Copyright © 2019 David Moyle Marsh
All Rights Reserved
ABSTRACT

Phased Array Digital Beamforming
Algorithms and Applications

David Moyle Marsh
Department of Electrical and Computer Engineering, BYU
Master of Science

With the expansion of unmanned aircraft system (UAS) technologies, there is a growing need for UAS Traffic Management (UTM) systems to promote safe operation and development. To be successful, these UTM systems must be able to detect and track multiple drones in the presence of clutter. This paper examines the implementation of different algorithms on a compact, X-band, frequency modulated continuous wave (FMCW) radar in an effort to enable more accurate detection and estimation of drones. Several algorithms were tested through post processing on actual radar data to determine their accuracy and usefulness for this system. A promising result was achieved through the application of pulse-Doppler processing. Post processing on recorded radar data showed that a moving target indicator successfully separated a target from clutter. An improvement was also noted for the implementation of phase comparison monopulse which accurately estimated angle of arrival (AOA) and required fewer computations than digital beamforming.

The second part of this thesis explains the work done on an adaptive broadband, real time beamformer for RF interference (RFI) mitigation. An effective communication system is reliable and can counteract the effects of jamming. Beamforming is an appropriate solution to RFI. To assist in this process FPGA firmware was developed to prepare signals for frequency domain beamforming. This system allows beamforming to be applied to 150 MHz of bandwidth. Future implementation will allow for signal reconstruction after beamforming and demodulation of a communication signal.

Keywords: Angle of Arrival, FMCW Radar, Phased Array Signal Processing, Doppler, Drones, Beamforming
ACKNOWLEDGMENTS

This work would not have been possible without the love and support from my amazing wife, Alexys. I want to give a huge thank you to my parents for always encouraging me and pushing me to be better. I am extremely grateful to my brothers and sisters for their incredible examples.

I am thankful to have had the opportunity to work with many wonderful students and people. I would like to give a special thanks to David Buck, Mitch Burnett, and Mark Ruzindana for their help, guidance, and friendship.

I would like to thank my committee members: Dr. David Long and Dr. Brian Jeffs for their help and advice. It has been a blessing to learn from them these past few years. I also want to thank my advisor Dr. Karl Warnick for all his support, assistance and advice over the course of my undergraduate and graduate degrees at BYU.
# TABLE OF CONTENTS

**LIST OF TABLES** ................................................................. vi

**LIST OF FIGURES** ............................................................. vii

**Chapter 1** *Introduction* .................................................... 1

1.1 Motivation .............................................................................. 1
1.1.1 Drone Detection .............................................................. 1
1.1.2 Communication Systems .................................................. 1
1.2 Literature Review ............................................................... 2
1.3 Contributions ......................................................................... 4
1.4 Thesis Outline ........................................................................ 4

**Chapter 2** *Background* ......................................................... 6

2.1 Phased Arrays ........................................................................ 6
2.1.1 Beamforming ................................................................. 7
2.2 FMCW Radar ......................................................................... 8
2.3 Radar Range Equation .......................................................... 10
2.4 Maximum Likelihood Estimation of Sinusoidal Parameters ............ 10
2.4.1 Frequency Estimation ....................................................... 11
2.4.2 Phase Estimation ............................................................ 12
2.5 Summary ............................................................................... 13

**Chapter 3** *Local Air Traffic Information System Radar* ................. 14

3.1 Introduction ............................................................................ 14
3.2 Radar Design ......................................................................... 14
3.3 Signal Processing Overview ................................................... 15
3.3.1 FFT, Correlation, and Averaging ........................................ 17
3.3.2 Coherence ........................................................................ 17
3.3.3 Sidelobe Reduction ........................................................... 19
3.4 Angle of Arrival Estimation ................................................... 20
3.4.1 AOA from Precomputed Weights ....................................... 20
3.4.2 Phase Comparison Monopulse .......................................... 21
3.4.3 MUSIC Algorithm .......................................................... 23
3.5 Target Number Estimation ..................................................... 27
3.5.1 Minimum Description Length (MDL) ................................. 27
3.5.2 Eigenvalue Gradients ....................................................... 28
3.5.3 Norm-Based Improved MDL (iMDL) .................................. 28
3.6 Target Detection ..................................................................... 31
3.6.1 Cell Averaging Constant False Alarm Rate (CA-CFAR) ....... 31
3.6.2 Doppler Processing .......................................................... 32
3.7 Additional Functionality ....................................................... 38
3.7.1 Inertial Measurement Unit ............................................... 38
Chapter 4 Adaptive Broadband Beamforming System for Naval Communications

4.1 Introduction
4.2 RF/IF Frontend
4.3 SNAP Board and Firmware
  4.3.1 F-Engine
  4.3.2 Bit Reduction
  4.3.3 Packetizer
  4.3.4 Results
4.4 Network Setup
4.5 Data Center Switch
4.6 Beamforming Backend

Chapter 5 Conclusion
5.1 Future Work

REFERENCES

Appendix A
  A.1 SNAP Board Startup
  A.2 SNAP Board Registers
  A.3 SNAP Board Notes
  A.4 SNAP Board Firmware Design
3.1  LATIS radar specifications. ................................................................. 15
3.2  Comparison of characteristics for different windows. ......................... 20

4.1  Different configurations for the Adaptive Broadband Beamforming System for Naval Communications project. The focus of the authors work was on the medium size array. 42
4.2  Header format for data packets. The header is the first word in the packet. 52
4.3  XID interpretation. The XID specifies which FFT bins the packet contains. 52
4.4  FID meaning. The FID specifies which antennas the packet data came from. 52
4.5  Packet Format for 6 antennas. Each packet contains 8 frequency bins from 6 antennas for 85 time windows. ......................................................... 53
4.6  ARP table from the SNAP board. IP and MAC addresses match those on the HPCs. 56
4.7  MAC address table for the Arista network switch. .............................. 56
LIST OF FIGURES

2.1 Uniform linear array. The geometry of the array influences the received signal at each antenna. A progressive phase shift of \( \frac{2\pi}{\lambda} d \sin \theta \) is applied to the incoming signal at each antenna. .......................................................... 7

2.2 Tx, Rx and resulting IF. Top: An FMCW chirp is transmitted and the returned signal is a delayed and attenuated chirp. When the transmitted signal is used to mix down the received chirp, a single frequency is the result (Bottom). The intermediate frequency depends on the time delay and corresponds to the targets range. ....................... 9

3.1 LATIS Radar. The system includes: a power supply, 1 transmit antenna, 4 receive antennas, RF board, IF board, ADC board, and MicroZed board with Zynq FPGA and dual core ARM CPU. .......................................................... 15

3.2 FMCW chirp from 10-10.25 GHz. This is a sampled spectrogram of the chirp after it is mixed down with a 9.75 GHz local oscillator. The doubled frequency from the mixer is faintly visible. .................................................. 16

3.3 Noncoherent data collected by the radar. For coherent data, each ADC sample on consecutive pulses should be the same as the previous pulse. In this figure the ADC samples show jitter, suggesting an incorrect sample offset. .......................... 18

3.4 Coherent data collected by the radar. This is an example of coherent data because the ADC values are consistent across multiple pulses. ................................. 19

3.5 AOA error (dB) vs SNR (dB). Increased SNR results in decreased error for phase comparison. .......................................................... 22

3.6 Example spatial spectrum of MUSIC algorithm. The AOA estimate produced by the MUSIC algorithm results from the sharp peak, providing a high resolution angular estimate. .................................................. 25

3.7 AOA estimates produced by the MUSIC algorithm. This data was recorded in an open field. The target moved from one side of the field to the other at different distances. The MUSIC algorithm accurately estimated the AOA. ........................................ 26

3.8 iMDL estimate of 2 targets where eigenvalues = 0.1444, 0.0761, 0.0006, 0.0002. The iMDL makes the inflection point easier to identify. The inflection point is represented by the point with the minimum norm, \( \Lambda_3 \). .................................................. 30

3.9 iMDL estimate of 1 target where eigenvalues = 3.8, 0.024, 0.007, 0.004. Successful estimation of the number of targets enables the MUSIC algorithm to perform accurately. .................................................. 30

3.10 Cell averaging constant false alarm rate (CA-CFAR) threshold. The CFAR threshold averages neighboring range bins to update the threshold value. The threshold is continuously changing to provide a constant probability of false alarm. .................................. 32

3.11 Range Doppler plot of a moving target in an environment with clutter. Clutter returns are seen in the 0 Hz frequency bin because it is relatively stationary. A moving target is visible on the right and separate from the clutter. .................................................. 34

3.12 Range Doppler plot of a moving target. The 0 Hz frequency bin has been zeroed out to remove the detection of clutter. The moving target is clearly visible and easily detected. .......................... 35
3.13 Target detection using CFAR and max power beamformer with clutter present. The clutter is detected 10 and 30 meters north of the radar. The data consists of a target running out to 40 m then turning around and running back. The target’s AOA estimates near 40 m are inaccurate. .................................................. 37

3.14 Target detection using Pulse Doppler and phase comparison in the presence of clutter. Pulse Doppler does not detect clutter returns and phase-comparison monopulse only uses the target return for AOA estimation. ........................................ 38

4.1 Adaptive Broadband Beamforming System for Naval Communications project setup. The rack holds the 4 HPCs, a 1 Gb switch, and a 40 Gb switch. ......................... 43

4.2 The IF frequency is in the 2nd Nyquist zone. The 50 MHz of transition band (25 MHz on each side) is discarded in the firmware by only keeping 96 out of the 128 frequency bins. .................................................. 44

4.3 The SNAP Board contains 3 ADCs, 2 10 GbE SFP+ ports, and an Xilinx Kintex7 FPGA. This board samples the signals, implements the F-engine, puts the data in specific packets, then sends the data to the HPCs. [1] ......................... 45

4.4 SNAP board firmware subsystems. After sampling, the data goes to the F-engine. Then the number of bits is reduced from 36 to 16. The packetizer splits the data into 12 packets with 8 bins in each. Then the two 10 GbE cores transmit the data to the network switch. .................................................. 46

4.5 Packetizer block diagram. The packetizer is responsible for creating 12 packets that each contain 8 frequency bins. The packetizer also maintains that the data rate transmitted by each 10 GbE core is above 7.2 Gbps. ................................. 47

4.6 Parallel to serial state machine. Used to break the data into 64-bit words and write to the jumbo FIFOs. The state machine controls the load and shift signals for the Casper block parallel to serial. .................................................. 48

4.7 Combiner state machine. This state machine takes the output of the bit reduction subsystem and assigns 8 frequency bins to 12 different FIFOs. ......................... 49

4.8 This is the state machine logic to load the 10 GbE cores and transmit the data to the HPCs. The data rate is maintained above 7.2 Gbps for each 10 GbE core by writing data for a set amount of clocks, and waiting for a set amount. ......................... 51

4.9 SNAP board packet data with an 81.25 MHz sine wave (bin 52) going into each antenna SMA. This is the average of the 85 time windows in a single packet. ......................... 54

4.10 Network setup for the Adaptive Broadband Beamforming System for Naval Communications project. Includes: 4 HPCs, 3 SNAP boards, 3 Raspberry Pis, 1 Arista switch, 1 TP-Link switch. .................................................. 55

4.11 Beamformer weights were calculated then multiplied with steering vectors at different angles. The simulated result is the normalized power of a beamformed signal at that angle. The max SNR weights were calculated assuming a SOI at 40 degrees and RFI at 0 degrees. .................................................. 58

4.12 Beamforming applied to SNAP board packets. Two beams were created. The weights on the left were set to all ones to maximize the power. The plot on the right used max SNR weights that treated the input as interference. The beamformer was successfully able to remove the interference as can be seen by the magnitude of the colorbars.  .... 58
A.1 Signal flow diagram for the sync generator. The sync generator starts the subsystems within the firmware design. A sync can be generated by writing a ‘1’ to the sync register, or by writing a ‘1’ to the arm register with the use of a 1 PPS signal. 67

A.2 Diagram of the firmware F-engine. The signals are sampled by the ADC then sent to a CASPER FFT wideband real. After the FFT the data is sent to bit reduction [2]. 68

A.3 Combiner subsystem. The combiner subsystem takes consecutive FFT outputs and concatenates them, creating a 192 bit data which is then sent to the packet setup subsystem. 68

A.4 Parallel to serial. The parallel to serial system takes 192 bit data and splits it up into 3 64 bit words which it then writes to a jumbo FIFO. 69

A.5 Packet setup subsystem. The packet setup subsystem uses a CASPER parallel to serial converter block to split the data up. 69

A.6 10 GbE core. This is the CASPER 10 GbE core. LEDs provide feedback on the connectivity and transmission of the core. 70

A.7 10 GbE data selection. The MUX selects the signals and data that goes to the GbE core. 71

A.8 Packet header. Shows the creation of the 64 bit header. For this case, this is the header for XID 0. The mcounr is 56 bits, XID is 4 bits, and FID is 4 bits. 71

A.9 Destination IP address. The MUX selects the destination IP address for each packet. The output of this goes to the 10 GbE core. The select line comes from the packetizer state machine. 72
1.1 Motivation

1.1.1 Drone Detection

The number of unmanned aircraft systems (UAS) is exploding and expected to reach over 7 million by the year 2020 [3]. With this expansion of UAS technology, there is a growing need for automated sense and avoid (SAA) technologies for collision avoidance. To assist in the development of these SAA technologies, creation of UAS Traffic Management (UTM) systems are needed, that enable safe low altitude operation. To be successful, these UTM systems must be able to detect and track multiple drones in the presence of clutter. The UAS must also transmit and receive position and navigational information with these UTM ground stations.

Radar is an attractive option that is capable of detection and drone monitoring. Radar works in most weather conditions, during the day or night, and can provide accurate range estimates. Previous work done to create an FMCW radar capable of detecting drones can be seen here, [4–7]. The work presented in this paper is an extension of that work. The previous projects aimed to reduce the size, weight and power (SWaP) of the radar in order to use it in flight on top of a drone. These constraints are not as strict with this UTM system because it is a ground based system. The overall goal is still to make the ground station movable and relatively inexpensive compared to current solutions. Throughout the remainder of this paper this radar project is referred to as the local air traffic information system or LATIS.

1.1.2 Communication Systems

Radio frequency interference (RFI) whether intentional (jamming) or unintentional is everywhere. The FCC tries to reduce the effects of unintentional RFI by managing the RF spectrum with allocated frequency bands. But for jamming and other hostile acts, robust systems must be
developed to mitigate or adapt to these adverse effects. One area where jamming is often seen is wireless communications. It has been shown in [8–10], that an effective way to remove RFI is by using phased arrays with digital beamforming. These previous projects applied beamforming to radio astronomy where the signal of interest (SOI) is very weak compared to the noise floor. The principles of beamforming also work in a communication system where the SOI is much stronger than the noise. The goal of this communication system project is to create an instrument for more reliable wireless communication in hostile environments by using an adaptive, broadband digital beamformer. The instrument consists of FPGAs and GPUs which handle the processing of the data. Throughout the remainder of this paper, this communication system project is referred to as the Adaptive Broadband Beamforming System for Naval Communications.

1.2 Literature Review

A common theme between the LATIS and the Adaptive Broadband Beamforming System for Naval Communications is the use of a phased array antenna with digital beamforming. The field of phased arrays and digital beamforming is mature and has material published in many books and papers. I cannot give an in depth review of these topics but I hope to give a brief summary and examples from some of the more comprehensive papers.

In 2002, Parker and Zimmermann discussed in [11] an overview of the theory and architecture of phased arrays. Phased arrays receive significant interest because of the ability to quickly form multiple beams. This can be accomplished using either passive or active arrays as explained by the authors. The LATIS and Adaptive Broadband Beamforming System for Naval Communications use active receiving arrays. Active arrays, though still expensive, are becoming increasingly affordable. With a decrease in the cost of active phased arrays and an increase in available computation power, phased arrays and beamforming can be implemented in more applications.

Van Veen and Buckley in [12] provide a wonderful overview to beamforming, or spatial filtering. They summarize that spatial filtering can separate two signals coming from different directions, even if they have the same frequency content. They point out that spatial filtering (beamforming) is similar to temporal filtering, which filters data from one element collected over a period of time. One such similarity is aliasing. The authors also mention that beamforming can be done on individual frequency channels which they call frequency domain beamforming. This is of
interest to the radio astronomy research group because frequency domain beamforming is applied in the Adaptive Broadband Beamforming System for Naval Communications project. Mention to, and references for, statistically optimal beamformers are given as well as applications that use phased arrays and beamforming.

One application of phased arrays and beamforming is radar. In 1985 Brookner explained in [13], that there are advantages of using an electronically scanned array (ESA), instead of a rotating dish. One of the main advantages, as explained by Brookner, is the update rate of the radar. For a mechanical dish, the update rate is limited by how fast the dish can turn. An ESA is not limited by physical motion because it can form its beams electronically. This allows an ESA to simultaneously track hundreds of targets by changing its beams in fractions of seconds. Another small advantage that a phased array has is it will never have a failed mechanical part.

Another application for phased arrays and beamforming is communications. In [14], Compton presents on an adaptive array that is used in a spread-spectrum communication system. This system used an LMS algorithm to track a desired signal while also nulling interference. Compton also discusses the importance of selecting signal waveforms so that the beamformer weights do not reduce the effectiveness of the communication system. In another paper [15], Adams, et al., examine the case where there are numerous interferers in the sidelobes of a communication system. Adaptively weighted beams are used to reduce the time of a link outage if these interferers cross over into the main beam. These papers show that beamforming can be successful when implemented in communication systems in the presence of interference.

An important topic within digital beamforming is choosing the beamformer weights. Beamformers can be placed into two main categories: data independent or statistically optimal. Data independent beamformers create a desired response through the application of FIR filter design. Statistically optimal beamformers chose their weights based on the statistics of the received data to minimize noise and RFI. The four main criteria for choosing optimal weights are mean square error (MSE), signal to noise ratio (SNR), maximum likelihood (ML), and minimum noise variance (MV) [17]. Some common optimal beamformers include the multiple sidelobe canceller (MSC), max SNR, and linearly constrained minimum variance (LCMV). These optimal beamformers require that the second order statistics are known. When these statistics are unknown or time varying, adaptive algorithms can be used. Two popular adaptive algorithms are the least mean
squares (LMS) and the recursive least squares (RLS) [12]. The weights of particular interest to the Adaptive Broadband Beamforming System for Naval Communications are the max SNR weights which are discussed in Section 2.1.1.

These are some of the relevant topics involving phased arrays and beamforming. The LATIS and Adaptive Broadband Beamforming System for Naval Communications build on these principles. In addition, the LATIS project builds on the previous work detailed in [4–7]. The Adaptive Broadband Beamforming System for Naval Communications shares components with previous projects explained in [8, 9].

1.3 Contributions

The author’s contributions to the LATIS project include

• Completed a design review of pulse-Doppler processing and verified the algorithm on real radar data in post processing.

• Implemented multiple angle of arrival estimation algorithms that have benefits such as higher resolution or faster computation.

• Added the capability for the radar to accurately estimate its position and heading information through the integration of an IMU and GPS.

The author’s contributions to the Adaptive Broadband Beamforming System for Naval Communications project include

• Designed FPGA firmware to sample, perform FFT, scale, packetize and transmit data in preparation for frequency domain beamforming on GPUs.

1.4 Thesis Outline

This thesis is organized as follows:

Chapter 2: Background, phased arrays, beamforming, FMCW radars.

Chapter 3: LATIS radar, signal processing algorithms for detection and estimation.
Chapter 4: Adaptive Broadband Beamforming System for Naval Communications, work done on a digital beamforming system for jamming mitigation.

Chapter 5: Conclusion, discuss important achievements and relevant future work.
CHAPTER 2. BACKGROUND

This chapter gives a background on important topics relevant to the LATIS and Adaptive Broadband Beamforming System for Naval Communications. These topics include phased arrays, digital beamforming, FMCW radars, and maximum likelihood estimation of sinusoidal parameters. Understanding the concepts in this chapter is important for the considerations made in this thesis.

2.1 Phased Arrays

A phased array is a group of antennas that can electronically steer an antenna beam by applying a phase shift at each element. Another advantage of phased arrays is that adding more elements to the array, creates a more directive beam. These two properties allow the implementation of phased arrays, where rotating dish antennas have been used in the past.

The simplest phased array configuration is the uniform linear array (ULA), where all the elements are equally spaced, as seen in Figure 2.1. The traditional method to examine a phased array beam pattern is to use the array factor method [18]. This method does not take into effect mutual coupling between elements and may not be accurate enough for high sensitivity applications but is useful in gaining simple intuition. By using the reciprocity theorem, insights gained from transmitting arrays can be applied to receiving arrays.

The following equations are slightly modified from [18] to match the geometry of Figure 2.1. For a ULA, the radiation pattern can be approximated by

\[
\vec{E}(\vec{r}) = \hat{E}_{\text{el}}(\vec{r})A(\theta, \phi),
\]

where \( \hat{E}_{\text{el}}(\vec{r}) \) is the individual element pattern and \( A(\theta, \phi) \) is the array factor defined by

\[
A(\theta, \phi) = \sum_{n=0}^{N-1} i_n e^{j kn d \sin \theta}.
\]
From this equation we can see that a signal coming from some angle $\theta$ has the same magnitude at each array element but has a different phase based on the antenna’s position.

2.1.1 Beamforming

The process of steering or scanning the phased array beam is called beamforming. Beamforming can be applied to transmit arrays or receive arrays. When applied to receive arrays, beamforming is nothing more than a weighted sum. The voltage output of a receiving array can be represented in vector form by

$$\mathbf{v} = \begin{bmatrix} v_1(t) & v_2(t) & \cdots & v_N(t) \end{bmatrix}^T.$$ 

Applying the beamformer weights results in the scalar output

$$v_{\text{out}} = \mathbf{w}^H \mathbf{v}. \quad (2.3)$$

As mentioned in Section 1.2, selecting the beamformer weights is important because it dictates the performance of the system. A powerful choice for beamformer weights is the max
SNR case. Assuming that the signal is uncorrelated with the noise, the correlation matrix, $\mathbf{R}$, can be broken up by

$$\mathbf{R} = \mathbf{R}_s + \mathbf{R}_n,$$

where $\mathbf{R}_s$ is the signal correlation matrix and $\mathbf{R}_n$ is the noise correlation matrix. After applying the beamformer weights, the SNR is represented by

$$\text{SNR} = \frac{w^H \mathbf{R}_s w}{w^H \mathbf{R}_n w}.$$  \hspace{1cm} (2.5)

The goal is to solve for weights that maximize Equation 2.5. The solution to maximizing a ratio of quadratic forms reduces to the solution of a generalized eigenvalue problem

$$\mathbf{R}_n^{-1} \mathbf{R}_s w = \lambda_{\text{max}} w,$$ \hspace{1cm} (2.6)

where the optimal weights equal the eigenvector that corresponds to the largest eigenvalue [18]. This is the method used in the Adaptive Broadband Beamforming System for Naval Communications to calculate and apply beamformer weights.

2.2 FMCW Radar

A frequency modulated continuous wave is a modulation scheme where the frequency changes over time and the transmitter is always on. Advantages of using an FMCW radar include being able to measure close targets, and having high accuracy range measurements. FMCW radars have different modulation schemes such as, sawtooth, triangular, rectangular, and staircase. The LATIS radar uses a sawtooth modulation pattern.

An echo return from a target looks like the original transmitted chirp but is delayed by $2R/c$. By using the transmit chirp to mix the receive echo, a beat frequency, $f_b$ (see Figure 2.2) is obtained. Applying knowledge of the slope of the chirp, $B/T$, where $B$ is the bandwidth and $T$ is the PRI, it can be shown that the range, $R$, to the target is related to the beat frequency by

$$R = \frac{cT f_b}{2B}. \hspace{1cm} (2.7)$$
Figure 2.2: Tx, Rx and resulting IF. Top: An FMCW chirp is transmitted and the returned signal is a delayed and attenuated chirp. When the transmitted signal is used to mix down the received chirp, a single frequency is the result (Bottom). The intermediate frequency depends on the time delay and corresponds to the targets range.

Another important concept in radar is range resolution. Range resolution is the minimum distance that two targets must be separated by to be detected by the radar. The range resolution for a FMCW radar is defined by the bandwidth of the chirp

\[ \Delta R = \frac{c}{2B}, \]  

(2.8)

where \( c \) is the speed of light.
2.3 Radar Range Equation

The SNR of a radar is defined by

\[ \text{SNR} = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 R^4 k_B T BF}, \]  

(2.9)

where \( \sigma \) is the radar cross section of the object. The SNR determines the maximum range of a target that can be detected. For a given SNR the maximum range is defined by

\[ R_{\text{max}} = \left[ \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 (\text{SNR}) k_B T BF} \right]^{1/4}. \]  

(2.10)

2.4 Maximum Likelihood Estimation of Sinusoidal Parameters

Maximum likelihood estimation is a technique that estimates unknown parameters given a distribution for a specific model. The model for our received signal comes from the implementation of the FMCW radar receiver board. The receiver board mixes the return echo by the original transmitted chirp which creates an intermediate frequency (IF). For a single point target in a noiseless environment the echo is the same as the transmitted chirp but with some amplitude attenuation and a time delay. The time delay which corresponds to the range of the target, causes the result of the IF signal to be a sinusoid with a single frequency. The model for our IF signal becomes a sinusoid with unknown parameters: amplitude, frequency, and phase [19]

\[ x_t = A \cos(\omega t - \phi). \]  

(2.11)

Now we adjust our model to account for multiple targets in a noisy environment. Each addition target contributes a sinusoid with an unknown amplitude, frequency, and phase. Our actual received signal is a summation of all the individual returns plus a noise element

\[ x_t = s_t(\theta) + n_t, \]  

(2.12)
where

\[ s_t(\theta) = \sum_{i=1}^{M} A_i \cos(w_i t - \phi_i) \]

\[ \theta = \begin{bmatrix} A_1 & \omega_1 & \phi_1 & \cdots & A_m & \omega_m & \phi_m \end{bmatrix}^T \]

and \( M \) equals the number of targets. The noise is independent, identically distributed Normal with zero mean and variance \( \sigma^2 \). \( n_t = N[0, \sigma^2] \). This means that the measurements of \( x_t \) are a sequence of i.i.d \( N[s_t(\theta), \sigma^2] \).

The pdf of a Gaussian random variable is

\[ f_X(x) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu_x)^2\right\} \]  

(2.13)

and joint density function for \( N \) random samples of \( x \) is

\[ f_\theta(x) = \prod_{i=1}^{N} f_\theta(x) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left\{-\frac{1}{2\sigma^2} \sum_{t=1}^{N} (x - s_t(\theta))^2\right\}. \]  

(2.14)

This yields a log-likelihood of

\[ L(\theta, x) = -\frac{N}{2} \ln 2\pi \sigma^2 - \frac{1}{2\sigma^2} \sum_{t=0}^{N-1} (x - s_t(\theta))^2. \]  

(2.15)

Expanding the last term results in

\[ L(\theta, x) = -\frac{N}{2} \ln(2\pi \sigma^2) - \frac{1}{2\sigma^2} \sum_{t=0}^{N-1} x_t^2 + \frac{1}{\sigma^2} \sum_{t=0}^{N-1} x_t s_t(\theta) - \frac{1}{2\sigma^2} \sum_{t=0}^{N-1} s_t^2(\theta). \]  

(2.16)

Equation (2.16) is the starting point for calculating frequency and phase estimates as described in Sections 2.4.1 and 2.4.2.

### 2.4.1 Frequency Estimation

The estimates of the frequency content of the return echoes are the values of \( \omega_i \) that maximize \( L \). The maximization can ignore the first two terms in Equation (2.16) because they have no dependence on \( \omega_i \). Examination of the final term provides insight into the limitations of estimating
the frequency and phase. Expanding this term results in

\[ \sum_{t=1}^{N} s_t^2(\theta) = \sum_{t=1}^{N} \left( \sum_{i=1}^{M} A_i \cos(w_i t - \phi_i) \right)^2 = \sum_{t=1}^{N} \sum_{i=1}^{M} \sum_{j=1}^{M} A_i \cos(w_i t - \phi_i) A_j \cos(w_j t - \phi_j). \] (2.17)

This can be rewritten as

\[ \sum_{t=1}^{N} \sum_{i=1}^{M} \sum_{j=1}^{M} \frac{A_i A_j}{2} (\cos((w_i + w_j)t - (\phi_i + \phi_j)) + \cos((w_i - w_j)t - (\phi_i - \phi_j))). \] (2.18)

When \( \omega_1 \neq \omega_2 \cdots \neq \omega_m \) this term is nearly constant at a value of

\[ \frac{N}{2} \sum_{i=1}^{M} A_i^2. \]

This means that the last term in Equation (2.16) can be ignored for the estimate of \( \omega \).

The maximization of the remaining term in \( L \) with respect to \( \omega \) becomes the estimation for the frequency

\[ \hat{\omega} = \arg \max_{\omega} \sum_{t=1}^{N} \sum_{i=1}^{M} x_i A_i \cos(w_i t - \phi_i) = \sum_{i=1}^{M} \text{Re} \left\{ \sum_{t=1}^{N} x_i e^{j(\omega t - \phi)} \right\}. \] (2.19)

To maximize \( \omega \), you must maximize all \( \omega_i \)

\[ \hat{\omega}_i = \arg \max_{\omega_i} X(\omega_i). \] (2.20)

The estimate of \( \omega \) can be used to determine the range of a target (see Section 2.2 and Equation (2.7)).

2.4.2 Phase Estimation

To estimate the phases of the received echoes, we again ignore the three terms in the log-likelihood function that do not depend on \( \theta \). We can then take the derivative of \( L \) with respect to \( \phi \)

\[ \frac{\partial L}{\partial \phi_i} = \frac{1}{\sigma^2} \sum_{t=1}^{N} x_i A_i \sin(\omega_i t - \phi_i). \] (2.21)
Setting the derivative equal to zero to find the maximum and simplifying results in

\[ 0 = \frac{A_i}{\sigma^2} \sum_{t=1}^{N} [x_t \sin(\omega_t t) \cos(\phi_i) - x_t \cos(\omega_t t) \sin(\phi_i)] \]  
(2.22)

\[ 0 = \cos(\phi_i) \sum_{t=1}^{N} x_t \sin(\omega_t t) - \sin(\phi_i) \sum_{t=1}^{N} x_t \cos(\omega_t t) \]  
(2.23)

\[ \frac{\sin(\hat{\phi}_i)}{\cos(\hat{\phi}_i)} = \frac{\sum_{t=1}^{N} x_t \sin(\omega_t t)}{\sum_{t=1}^{N} x_t \cos(\omega_t t)} \]  
(2.24)

\[ \hat{\phi}_i = \arctan \left( \frac{\sum_{t=1}^{N} x_t \sin(\omega_t t)}{\sum_{t=1}^{N} x_t \cos(\omega_t t)} \right). \]  
(2.25)

Since \( \cos(x) = Re \{ e^{jx} \} \) and \( \sin(x) = Im \{ e^{jx} \} \), Equation (2.25) reduces to

\[ \hat{\phi}_i = \arctan \left( \frac{Im \{ \sum_{t=1}^{N} x_t e^{j\omega_t t} \}}{Re \{ \sum_{t=1}^{N} x_t e^{j\omega_t t} \}} \right) = \arctan \left( \frac{Im \{ X(\omega_i) \}}{Re \{ X(\omega_i) \}} \right) = \arg \{ X(\hat{\omega}_i) \}. \]  
(2.26)

First, an estimate of the frequency, \( \omega_i \), is calculated. Then the frequency estimate is used in Equation (2.26) to calculate the phase, \( \phi_i \). The estimate of \( \phi \) can be used in the calculations of the angle of arrival.

### 2.5 Summary

Understanding the concepts presented in this chapter provides building blocks that are used throughout the remainder of this paper. By understanding the basics of an FMCW radar, an accurate signal model can be developed and algorithms that fit that model can be applied. A knowledge of phased arrays and beamforming provides a framework for the Adaptive Broadband Beamforming System for Naval Communications. Finally, familiarity with detection and estimation theory helps analyze and implement effective algorithms.
CHAPTER 3. LOCAL AIR TRAFFIC INFORMATION SYSTEM RADAR

3.1 Introduction

The LATIS radar builds on previous work done to create a compact, phased array radar for UAV detection. In 2014, Eck designed and fabricated PCB Vivaldi endfire antennas [7]. In 2015, the initial design and testing was done by Spencer [5]. Hardware improvements were made by Roberts [4] in 2017 and FPGA firmware development was done by Newmeyer [6]. With all this progress and a fully working radar, I collected useful radar data and tested different detection and estimation algorithms. My main focus for the LATIS radar was on the signal processing which resulted in the implementation of: phase comparison monopulse for AOA estimation, CFAR, MU-SIC, and post processing pulse-Doppler.

3.2 Radar Design

The hardware for the LATIS radar can be seen in Figure 3.1. The radar consists of 4 different boards: RF, IF, ADC, and MicroZed [4–7]. As explained in Section 2.2, the received echoes must be mixed with the original transmitted chirp to produce a single tone. The RF board generates the sawtooth FMCW chirp which operates from 10-10.25 GHz. Detailed specifications can be seen in Table 3.1. The RF board also performs the mixing before passing the signals to the IF board. The IF board amplifies the signals and applies an anti-aliasing filter in preparation for the ADC board. The MicroZed board performs all the detection and estimation.

Figure 3.2 shows a spectrogram of one period of the chirp, after it has passed through an analog mixer with a 9.75 GHz local oscillator. A result of using a non-ideal analog mixer, the doubled frequency is not completely filtered out and is also visible in the spectrogram.
Figure 3.1: LATIS Radar. The system includes: a power supply, 1 transmit antenna, 4 receive antennas, RF board, IF board, ADC board, and MicroZed board with Zynq FPGA and dual core ARM CPU.

Table 3.1: LATIS radar specifications.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center Frequency</td>
<td>10 GHz</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>250 MHz</td>
</tr>
<tr>
<td>Range Resolution</td>
<td>0.6 m</td>
</tr>
<tr>
<td>Pulse Period</td>
<td>2.048 ms</td>
</tr>
<tr>
<td>Sample Rate</td>
<td>2 MHz</td>
</tr>
<tr>
<td>Unambiguous Range</td>
<td>1200 m</td>
</tr>
<tr>
<td>Transmit Power</td>
<td>50 mW</td>
</tr>
</tbody>
</table>

3.3 Signal Processing Overview

The main goal of radar signal processing is to transform raw ADC data into target information such as number of targets, size, range, angle of arrival, and velocity. Engineers have developed
Figure 3.2: FMCW chirp from 10-10.25 GHz. This is a sampled spectrogram of the chirp after it is mixed down with a 9.75 GHz local oscillator. The doubled frequency from the mixer is faintly visible.

many algorithms to improve the detection of targets and the estimation of their parameters. Important aspects of an algorithm are to have stable, accurate, and precise results. The author has implemented several algorithms to determine if any could provide improvements to AOA estimation and detection for the LATIS radar.

The main tasks of the signal processing system consist of: (1) sampling the IF voltages using the ADCs, (2) taking the FFT of the data, (3) creating a correlation matrix using the data from the different channels, (4) averaging coherent correlation matrices across multiple pulses, (5) target detection, and (6) angle of arrival estimation.

After the FFT, correlation, and coherent averaging has been computed in the FPGA, the data is transported to the CPU where the signal processing algorithms can be applied. The radar signal processing areas that are addressed in this chapter include: angle of arrival estimation, estimation of the number of targets, and target detection. There is also a section on some added functionality that is useful for when multiple radar stations are networked together.
3.3.1 FFT, Correlation, and Averaging

As explained in Section 2.2, the range of a target can be calculated by the frequency of the return chirp mixed down by the transmit chirp (see Equation 2.7). Because the return echoes are happening at relatively the same time, they must be separated in the frequency domain to recover the beat frequency. The LATIS radar accomplishes this by performing a 4096-point FFT. The data is real which means the FFT output is conjugate symmetric and half of the output is discarded because it is redundant. The next step after taking the FFT of the ADC samples is to form a correlation matrix. This is done by first forming a vector $X$

$$X_i = [A_i \ B_i \ C_i \ D_i]^T,$$

(3.1)

where $A_i$, $B_i$, $C_i$, and $D_i$ are the values of the FFT at frequency bin $i$ for the four channels. The correlation matrix, $R_i$, is created by the following

$$R_i = E [X_i X_i^H].$$

(3.2)

This forms a $4 \times 4$ correlation matrix for each range bin which results in a $4 \times 4 \times 2048$ data cube for each pulse. By averaging multiple coherent correlation data cubes together, there is an increase in SNR as the noise floor is lowered. Having coherent data cubes implies two properties: the target is present, and the target is coherent over the averaging. The correlation matrix is useful in computing an angle of arrival estimate and is used by digital beamformer algorithms.

3.3.2 Coherence

Averaging becomes increasingly more effective when the radar is coherent. Coherent integration is when both the amplitude and the phase of the received signals add constructively and add in phase with each other. This enables the radar to use multiple pulses to average out the noise. One aspect of coherence is that the target coherence can dictate the coherent processing interval (CPI). That is, if the target is moving rapidly, it is not coherent for very long, whereas, a stationary target is always coherent.
Figure 3.3: Noncoherent data collected by the radar. For coherent data, each ADC sample on consecutive pulses should be the same as the previous pulse. In this figure the ADC samples show jitter, suggesting an incorrect sample offset.

When \( n_p \) pulses are coherently integrated in the presence of white noise, the SNR gain is \( n_p \) times the SNR of one pulse [20], i.e.,

\[
\text{SNR}(n_p) = n_p \times \text{SNR}(1). \tag{3.3}
\]

The inherited radar had a logging function that was noncoherent, as evident in Figure 3.3. Figure 3.3 is a plot of the ADC values when sampling a single sinusoid across multiple pulses. If the data were coherent, there would be no ripple across adjacent pulses because coherence would ensue the same value for each sample as the previous sample. A coherent version can be seen in Figure 3.4. A timing offset was fixed in the logging function which now achieves coherence and provides reliable data for algorithmic testing.
Figure 3.4: Coherent data collected by the radar. This is an example of coherent data because the ADC values are consistent across multiple pulses.

### 3.3.3 Sidelobe Reduction

We want to maximize the power from a target in a specific FFT/Range bin. Maximizing this power makes detection easier. In [20], sidelobe reduction in an LFM Waveform is presented with an explanation on Taylor weighting. Taylor weighting is often applied in radar because it achieves the minimum mainlobe width for a given peak sidelobe level. It also achieves a lower average and peak straddle loss. This could be an improvement to the current radar because it currently does not implement windowing. The values in Table 3.2 were generated by the Matlab wvtool and show the tradeoffs of leakage factor, sidelobe level, and mainlobe width for different windows.
Table 3.2: Comparison of characteristics for different windows.

<table>
<thead>
<tr>
<th></th>
<th>Hamming (4096)</th>
<th>Taylor (4096, 4, -50)</th>
<th>Taylor (4096, 4, -200)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leakage factor</td>
<td>0.04%</td>
<td>0.02%</td>
<td>0%</td>
</tr>
<tr>
<td>Relative sidelobe attenuation</td>
<td>-42.7 dB</td>
<td>-44.7 dB</td>
<td>-69.6 dB</td>
</tr>
<tr>
<td>Mainlobe width (-3dB)</td>
<td>$6.1035 \times 10^{-4}$</td>
<td>$6.1035 \times 10^{-4}$</td>
<td>$7.9346 \times 10^{-4}$</td>
</tr>
</tbody>
</table>

There are some disadvantages to using a window. One of which is the decrease in SNR. Given a window $W[k]$ with $M$ points, the SNR loss is:

$$\text{SNR}_{\text{loss}} = \frac{\left[ \sum_{k=1}^{M} W[k] \right]^2}{M \sum_{k=1}^{M} W^2[k]}.$$  \hspace{1cm} (3.4)

For a Taylor window (4096, 4, -50), the SNR loss is 1.4 dB, which is considered an acceptable tradeoff.

### 3.4 Angle of Arrival Estimation

The radar must accurately estimate the angle of arrival so that the UTM systems can direct the drones away from other drones or objects. The following sections explain AOA algorithms that have been implemented and compared to determine effectiveness.

#### 3.4.1 AOA from Precomputed Weights

One method to estimating the angle of arrival is to compute a max power beamformer. The inherited radar uses 100 precomputed weights that are derived from the steering vectors explained in Section 3.4.3 (see Equation (3.8)). The power, $P$, for each angle is computed by

$$P_\theta = a^H_\theta R a_\theta. \hspace{1cm} (3.5)$$

The angle that produces the highest power is used as the estimate. Because there are only a discrete number of angles, interpolation can be used to get smoother transitions between angle estimates.
A disadvantage of this method is the requirement to sweep across all the angles. This is time consuming if high resolution is desired.

### 3.4.2 Phase Comparison Monopulse

Phase-comparison monopulse also called phase-interferometry is a technique used to accurately estimate the AOA of a target. This technique uses multiple antennas and the estimated phase information of the received signals. The name monopulse comes from the attribute that only one pulse is needed to calculate an estimate for the angle.

An interferometer uses the geometry of an antenna array and the phase differences to generate an estimate for AOA. When a target is at an incidence angle other than zero, the far field electromagnetic wave, acting as a plane wave, propagates different distances to the individual array elements, see Figure 2.1. Given the spatial frequency of $2\pi/\lambda$, the difference in phase between an electromagnetic wave observed by two antennas separated by distance $d$ is

$$\Delta \phi = \frac{2\pi}{\lambda} d \sin(\theta).$$  \hspace{1cm} (3.6)

If $\Delta \phi$ can be estimated, the AOA can be computed by

$$\theta = \sin^{-1}\left(\frac{\lambda \Delta \phi}{2\pi d}\right).$$  \hspace{1cm} (3.7)

To successfully extract accurate phase information, one of two things must happen. Either the signals are distinguishable in time, or the signals are separable in frequency. As explained in Section 2.4, the sampled IF signal in our radar system, is a summation of sinusoids. Therefore, it is not possible for this radar to distinguish the signals in time. The only remaining option is to try to separate those sinusoids by their frequency content. As described in Section 2.4.2, the phase is estimated by $\hat{\phi}_i = \arg(X(\phi_i))$.

As with all things, the addition of noise affects the performance and accuracy of this estimation. To view the effects of noise on AOA estimates the following simulation was run. Two signals were simulated with the same frequency but different phase, $\phi$, to represent the ADCs sampling the same range target (same beat frequency) at an angle of $\theta$ defined by Equation (3.7). Gaussian
Figure 3.5: AOA error (dB) vs SNR (dB). Increased SNR results in decreased error for phase comparison.

noise was added to the signals, and then the FFT was taken to determine the targets range. The phase of each signal was taken from the real and imaginary parts of the FFT bin that had the most power. An error was created by subtracting these two phases and comparing to the original angle. Simulation results comparing SNR and error can be seen in Figure 3.5.

To confirm the results of the ML estimator, another sine wave was added to one of the existing ones with a slightly different frequency but in the same FFT bin. This new sine wave had a different phase and, as expected, gave a drastically different result for the AOA. This motivated the exploration and implementation of the MUSIC algorithm (see Section 3.4.3) as a viable solution to correct this error.
Implementation

Phase comparison monopulse can be easily scaled to larger arrays because it does not scan across a range of angles making the computation time faster than other methods. The actual implementation is as simple as grabbing the difference in phase across each element from the correlation matrix and plugging that value into Equation (3.7). For a pair of antennas, the following two lines of code accomplish this

```c
float phase_diff_ab = atan2(ab_i, ab_r); //Antenna 2 to 1 (A to B)
float theta1 = asin((1/PI)*phase_diff_ab);
```

where $ab_r$ and $ab_i$ are the real and imaginary entries in the correlation matrix. Since the array has four elements, 3 pairs of adjacent antennas can be used to estimate the angle. These three estimates can then be averaged together to improve accuracy. If adjacent antennas are not used for the angle estimate, spatial aliasing can occur because the antennas are separated by $\lambda/2$.

3.4.3 MUSIC Algorithm

The Multiple Signal Classification (MUSIC) algorithm is a well known high resolution angle of arrival algorithm for narrow-band signals arriving at an array. This algorithm can estimate the angle of arrival for $p - 1$ targets, where $p$ is the number of array elements. The MUSIC algorithm uses the data collected by multiple antennas with known positions. The sampled signal is composed of the received signal and a noise component, $s_t = x_t + n_t$. The algorithm takes note that the observed signal contains a combination of a signal subspace and a noise subspace. The noise subspace does not contain any of the signal and is therefore orthogonal to the signal subspace. The MUSIC algorithm uses the orthogonality of these two subspaces as a basis for finding the angle of arrival. [21–23]

In a noise-free environment, the signal is spanned by the steering vector $a(\theta)$ defined by

$$
a(\theta) = \begin{bmatrix} 1 & e^{-jkd_1 \sin(\theta)} & e^{-jkd_2 \sin(\theta)} & e^{-jkd_3 \sin(\theta)} \end{bmatrix}^T, \quad (3.8)
$$

where $k$ is the wavenumber $2\pi/\lambda$ and $d_i$ is the difference in distance between each antenna in the array with the first, see Figure 2.1.
Ideally, the inner product between the signal subspace and the noise subspace is zero because of the orthogonal nature of the signals as seen in Equation (3.9).

\[ a^H R_n a = 0 \]  

(3.9)

Since the algorithm is applied to real world signals, the product in Equation (3.9) only approaches zero. Using the reciprocal of Equation (3.9) produces a spatial spectrum with sharp peaks at the angles where the noise is orthogonal to the ideal signal. The MUSIC algorithm searches through a set of possible steering vectors to find the one that is orthogonal to the noise subspace [21–23].

The noise subspace is not perfectly known, so an estimate must be found. This estimate is obtained by doing an eigen decomposition. The correlation matrix is decomposed into \( p \) (number of antennas) eigenvectors and eigenvalues. When \( k \) targets are present, the \( p - k \) smallest eigenvalues and eigenvectors correspond to the noise subspace.

Pros and cons of the MUSIC algorithm include

- Works for arbitrary antenna array configurations, as long as the positions are known
- Sensitive to sensor position, gain, and phase errors
- Computationally expensive to search through all \( \theta \).

Implementation

The FPGA produces a \( 4 \times 4 \) correlation matrix for each range bin. When a target is detected at a specific range, the correlation matrix is used to estimate the angle of arrival. The first step in implementing the MUSIC algorithm is to estimate the noise subspace. This is done by taking the eigen decomposition of the correlation matrix, which produces four eigenvalues with their corresponding eigenvectors. An estimate of the noise matrix, \( \hat{R}_n \), is formed by

\[ \hat{R}_n = V_{\text{noise}} V_{\text{noise}}^H, \]  

(3.10)

where \( V_{\text{noise}} \) is a \( 4 \times n \) matrix and \( n \) represents the number of noise eigenvalues. The columns of \( V_{\text{noise}} \) are the eigenvectors that correspond to the noise subspace. The resulting noise matrix is a
$4 \times 4$ matrix. There are several ways to estimate the number of signal eigenvalues and noise eigenvalues. One method is called the Minimum Description Length (MDL) criterion and is described in Section 3.5.1.

With the steering vector $a(\theta)$ and an estimate of the noise subspace, $\hat{R}_n$, the spatial spectrum is calculated by

$$S_n(\theta) = \frac{1}{a^H \hat{R}_n a}.$$  \hfill (3.11)

The estimated angle of arrival is taken by the value of $\theta$ that maximizes $S_n$.

An example of a spatial spectrum that used real radar data can be seen in Figure 3.6. For this test a corner reflector is placed 20 m away from the radar at an angle around $-30\text{deg}$.

![Figure 3.6: Example spatial spectrum of MUSIC algorithm. The AOA estimate produced by the MUSIC algorithm results from the sharp peak, providing a high resolution angular estimate.](image)
The MUSIC algorithm’s effectiveness is limited to uncorrelated signals. One improvement to the algorithm mentioned in [23] is to apply a transformation matrix $J$:

$$
J = \begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}.
$$

The transformation matrix is applied to the correlation matrix to create a new correlation matrix by

$$
R_{\text{new}} = R_{\text{old}} + JR_{\text{old}}^*J.
$$

(3.12)

When Equation (3.12) is applied to simulated data, a drastic improvement is seen in the AOA estimates.

Figure 3.7: AOA estimates produced by the MUSIC algorithm. This data was recorded in an open field. The target moved from one side of the field to the other at different distances. The MUSIC algorithm accurately estimated the AOA.
3.5 Target Number Estimation

For the MUSIC algorithm to produce an accurate and precise estimate of the AOA, the total number of targets in each range bin must be known.

3.5.1 Minimum Description Length (MDL)

One algorithm that estimates the number of targets is called the minimum description length (MDL). MDL uses the eigenvalues of the correlation matrix and maximizes the following log likelihood ratio [24–26]

$$L(\Theta) = -N \log \det R - \text{tr}[R^{-1}\hat{R}], \quad (3.13)$$

where $\hat{R}$ is the sample-covariance matrix

$$\hat{R} = \frac{1}{N} \sum_{i=1}^{N} x(t_i)x(t_i)^H.$$ 

By substituting in maximum likelihood estimates [24], the log likelihood ratio is reduced to

$$L(k) = (p - k)N \log \left( \frac{\prod_{i=k+1}^{p} l_i^{1/(p-k)}}{\frac{1}{p-k} \sum_{i=k+1}^{p} l_i} \right), \quad (3.14)$$

where $p$ is the number of array elements, $k$ is the number of targets and $l_i$ are the eigenvalues with $l_1 \geq l_2 \geq \cdots \geq l_p$. Using this maximum likelihood estimate and adding in the free parameter calculations, the resulting criterion is

$$\text{MDL}(k) = -L(k) + \frac{1}{2} k(2p - k) \log N, \quad (3.15)$$

where $N$ is the number of observations of the signal. For our radar, the correlation matrix is made from the average of 6 pulses, so $N = 6$. To solve the maximum likelihood estimate, $L$ should be maximized. The number of targets is the value of $k$ that minimizes the MDL.

Once this has been done, the $p - k$ eigenvectors corresponding to the $p - k$ smallest eigenvalues can be used to form a noise matrix as explained above.
3.5.2 Eigenvalue Gradients

In [27], the gradients of the eigenvalues are used to estimate the number of targets. Given that the eigenvalue decomposition of the correlation matrix, where \( L \) is the number of antenna elements, results in eigenvalues

\[
\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq \cdots \geq \lambda_L
\]  

(3.16)

\[
\Delta \lambda = \frac{\lambda_1 - \lambda_L}{L-1}
\]  

(3.17)

\[
\Delta \lambda_i = \frac{\lambda_i - \lambda_{i+1}}{L-1}
\]  

for \( i = 1, \ldots, L-1 \).

(3.18)

Next you find all \( i \) that satisfy

\[
\Delta \lambda_i \leq \Delta \lambda.
\]  

(3.19)

Then take the \( i \) that is the first one of the last continuous block and the estimated signal number is \( p = i_0 - 1 \).

3.5.3 Norm-Based Improved MDL (iMDL)

A third method that I have implemented is called the norm-based improved MDL (iMDL) [28]. I continued to research more algorithms for target estimation because post processing on actual radar data led me to believe that the MDL (in our case) over estimates the number of sources. The eigenvalue gradient method performed well but is susceptible to noise.

The iMDL algorithm also starts with an eigenvalue decomposition as seen in Equation (3.16). The first step is to normalize the eigenvalues so that they appear on the interval \([0,1]\), with the largest being 1 and the smallest being 0. The indices are also also normalized on the interval \([0,1]\). These normalizations are shown in Equations (3.20) and (3.21).

\[
l_i = \frac{\lambda_i - \lambda_m}{\lambda_1 - \lambda_m} \quad \text{for } i = 1, \ldots, m.
\]  

(3.20)

\[
i^{(N)} = \frac{i - 1}{m - 1}.
\]  

(3.21)
The normalized eigenvalues are further modified by Equation (3.22). This non-linear operation is used to bend the curve of the eigenvalues.

\[ \lambda_i^{(N)} = \sqrt{1 - (1 - I_i)^E}, \quad (3.22) \]

where \( E \) is an integer and was optimally selected to be 5 by the authors in [28]. With these modified eigenvalues and indices, a vector, \( \Lambda_i \), is created as can be seen in Equation (3.23).

\[ \Lambda_i = \left[ \begin{array}{c} \lambda_i^{(N)} \\ i^{(N)} \end{array} \right] \quad (3.23) \]

iMDL estimation of the number of targets selects the smallest Euclidean norm. In a perfect environment the smallest eigenvalues are equal to the noise variance. Then the number of targets would be equal to the number of antennas subtracted by the number of eigenvalues that are the same. But because this is a real world application, this is not the case. The eigenvalues tend toward the noise variance but are not equal to each other. The iMDL algorithm looks for the transition from the largest \( p \) (number of targets) eigenvalues to the smallest \( m - p \) eigenvalues. This results in an inflection point. This inflection point is the smallest Euclidean norm which enables the estimation of \( p \)

\[ ||\Lambda_{p+1}|| < ||\Lambda_i|| \quad \text{for} \quad i \neq p + 1 \quad (3.24) \]

\[ \hat{p} = \arg\min_i ||\Lambda_i|| - 1. \quad (3.25) \]

An example of this is seen in Figure 3.8. The eigenvalues are 0.1444, 0.0761, 0.0006, and 0.0002. The Euclidean norm is displayed at each of the points. In this example the smallest norm which is the inflection point is seen at \( i = 3 \) and the number of targets is equal to two. Another example can be seen in Figure 3.9, where the eigenvalues are 3.8, 0.024, 0.007, and 0.004, the inflection point happens at \( i = 2 \), and the number of estimated targets is 1. With more antennas there are more eigenvalues which makes the inflection point easier to identify.
Figure 3.8: iMDL estimate of 2 targets where eigenvalues = 0.1444, 0.0761, 0.0006, 0.0002. The iMDL makes the inflection point easier to identify. The inflection point is represented by the point with the minimum norm, $\Lambda_3$.

Figure 3.9: iMDL estimate of 1 target where eigenvalues = 3.8, 0.024, 0.007, 0.004. Successful estimation of the number of targets enables the MUSIC algorithm to perform accurately.
3.6 Target Detection

One of the most fundamental tasks of a radar is to be able to detect targets accurately. This includes minimizing the probabilities of false alarm (reporting a detection when there is no target) and missed detection (not reporting a detection when there is a target).

3.6.1 Cell Averaging Constant False Alarm Rate (CA-CFAR)

A common method to achieve detection is to apply a threshold. One type of threshold algorithm is called constant false alarm rate (CFAR). The goal of a CFAR algorithm is to continuously adjust the threshold to maintain a constant probability of false alarm. A cell averaging CFAR creates an average at each range bin from a window of neighboring range bins. These averages are multiplied by the CFAR constant, $\alpha$. Reference [20] defines the calculation for $\alpha$ as

$$\alpha = N\left[P_{\text{FA}}^{-\frac{1}{N}} - 1\right],$$  \hspace{1cm} (3.26)

where $N$ is the number of data points that are averaged. $\alpha$ is chosen based on the choice for $P_{\text{FA}}$. An example of the CA-CFAR is seen in Figure 3.10.

The CA-CFAR algorithm was compared to a previously implemented ordered statistic CFAR (OS-CFAR) algorithm [4]. The CA-CFAR has poorer performance when multiple targets are close together. The OS-CFAR is implemented through the use of the Matlab sort function. The algorithm rearranges a window of range bins from smallest to largest values. One of the larger values is picked and multiplied by $\alpha$. To implement the OS-CFAR in C++ code for real time detection, it is crucial to use an effective sort algorithm. Quicksort is a popular sorting algorithm but in the OS-CFAR, the sorting is being done 64 times with only 2 out of the 32 values different. We can take advantage of this because insertion sort performs well when an array is nearly sorted. A C++ implementation of the OS-CFAR algorithm has been completed and, as expected, insertion sort performed faster than quicksort on the hardware.
Figure 3.10: Cell averaging constant false alarm rate (CA-CFAR) threshold. The CFAR threshold averages neighboring range bins to update the threshold value. The threshold is continuously changing to provide a constant probability of false alarm.

3.6.2 Doppler Processing

Another important method in radar to achieve detection of targets is Doppler processing. Doppler processing is a technique that uses Doppler information to detect targets and measure their velocity. This technique enables detection of targets when clutter is the dominant interference [20]. This has application to the LATIS project because deployment of the radar ground station in a city or other environment with large clutter returns (from buildings, trees and other objects) can drown out the return signal from the drone. Increasing the transmit power does not help with target detection in clutter because the power returned by the clutter is also increased. Doppler processing is the main technique for increasing the signal to clutter ratio (SCR). To increase the SCR, the signals must be separable in frequency. The moving target signal can be separated from the clutter returns in the frequency domain. Clutter is relatively stationary (wind moving trees) so it
contributes very low or no Doppler shift to the return echo. Thus we can apply Doppler processing to detect drones in the presence of large clutter returns.

Doppler processing has two general cases: moving target indication (MTI) and pulse-Doppler processing. MTI requires fewer computation but can only detect the presence of a moving target. MTI cannot identify if there is more than one target per range bin or what the velocity of that target is. The main advantage of an MTI is decreased complexity and fewer computations. However, pulse-Doppler processing also achieves detection of a target but can also measure Doppler shift, and from that estimate radial velocity. Another advantage of pulse-Doppler is that it can detect multiple targets, given that they are separated by enough Doppler to be resolved. These benefits come at a cost of increased computation and complexity [20].

An MTI is implemented using time data across multiple coherent pulses (slow time). Consecutive slow time range bins are filtered using a high pass filter. This attenuates the clutter which is centered around 0 Hz. Finally, a threshold is applied to the time data and a decision is made on detection.

Pulse-Doppler processing uses the same group of coherent range bins but computes the spectrum. This enables the estimation of Doppler shift frequencies which can be used to estimate the radial velocity of the target. By viewing the spectrum, multiple moving targets can be detected as long as they are moving faster than the clutter. Figure 3.11 shows a range-Doppler plot of a moving target in the presence of clutter. As expected, the clutter appears at 0 Hz because it is not moving and therefore contributes no Doppler shift to the return echo. To more easily see the moving targets, the low frequency bins are zeroed out removing the clutter. An example of the same range-Doppler plot but with the clutter removed can be seen in Figure 3.12.

In addition to the benefits listed above, pulse-Doppler processing is a form of coherent integration in slow-time, which results in a processing gain. Reference [20] states that the gain is $M$, where $M$ is the number of slow-time samples. If a window is used on the slow time data, the processing gain is reduced by the processing loss.

**Pulse-Doppler Design for future implementation**

As the current radar does not utilize pulse-Doppler processing, the following specifications when implemented could provide consistent detection of targets in clutter. The Doppler shift, $f_d$, ...
Figure 3.11: Range Doppler plot of a moving target in an environment with clutter. Clutter returns are seen in the 0 Hz frequency bin because it is relatively stationary. A moving target is visible on the right and separate from the clutter.

Introduced by a moving target is

\[ f_d = \frac{2v}{\lambda}, \]  

where \( v \) is the radial velocity of the target and \( \lambda \) is the wavelength. A target (drone) moving at 15 m/s (33.5 mph) introduces a Doppler shift of 1000Hz. To satisfy Nyquist and measure this frequency shift without aliasing, the pulse repetition frequency (PRF) must be greater than 2000Hz. This means that the pulse repetition interval (PRI) is no greater than 500 \( \mu s \), since \( PRI = \frac{1}{PRI} \). The current radar hardware is capable of generating a chirp with a bandwidth of 250 MHz and period of 500 \( \mu s \). We can also use an ADC that samples at 10 MHz. The ADC is able to generate 5000 samples per chirp. This gives the FFT output a resolution of 2000 Hz/Bin in fast time. The range resolution of an FMCW radar, as defined in Equation (2.8), is only dependent on the bandwidth, and is 0.6 meters. If the velocity of the target is 15 m/s and each range bin is 0.6
meters wide, it takes 40 ms for the target to cross range bins. In 40 ms the radar can chirp 80 times. This enables us to take at least a 64-point FFT across these coherent chirps in the same range bin.

With a 10 Msamp/s ADC the highest detectable unaliased frequency is 5 MHz. Using Equation (2.7) this corresponds to a range of 1500 meters. This system can avoid range ambiguity by applying an anti-aliasing filter before the ADCs.

The frequency resolution in slow time would be \( \frac{2000}{64} = 31.25 \text{Hz} \). A Doppler shift of 31.25 Hz corresponds to a velocity of 0.469 m/s (1 mph). Theoretically, after high pass filtering the data to remove the clutter, this radar should be able to pick out targets that are traveling from a few mph up to 33 mph.

Implementing pulse-Doppler processing in firmware introduces challenges. The main challenges include having enough resources to process the data, and having enough memory to store the results. The processing on the FPGA firmware includes taking four 4096-point FFTs every
chirp. These FFTs need to be done much faster, as the chirp is now only 500 µs. The output of these FFTs need to be stored for 64 consecutive chirps. Every 64 chirps, a 64-point FFT is taken across each range bin to calculate the Doppler. For Doppler processing across 64 chirps, the FFTs need to be done in 32 ms. Finally, the CPU needs to apply a threshold to determine if there is significant Doppler to detect a target. The main concern with implementing pulse-Doppler is the total number of FFTs that need to be taken and storing the data from 64 consecutive chirps.

Another important property of pulse-Doppler to consider is the existence of blind velocities. The data is being sampled in slow time at the PFR (2000 Hz) and then a high pass filter is applied. In the frequency domain the spectrum is periodic, therefore the velocities at and around multiplies of the PFR are filtered out. These are called blind velocities. For example, with a PRF of 2000 Hz, the first blind velocity occurs at 2000 Hz which corresponds to a velocity of 30 m/s (67 mph). Also, with this recommended design, any velocity above 15 m/s aliases/folds into the spectrum between ±PRF/2.

**Doppler Processing with phase comparison for direction of arrival**

This section combines the previous work done with phase comparison monopulse and pulse-Doppler processing. Figure 3.14 shows the application of the algorithm to real radar data collected at a park that contains clutter (trees, power lines, road).

This algorithm implements the following processing steps. First, the FFT of the ADC data is taken on all four channels as normal for 64 chirps. Then another FFT is taken in slow time across all range bins. This can be seen in the range-Doppler Figure 3.11. By zeroing out the low frequency bins (Figure 3.12), only moving targets are visible. When an FFT is taken, the phase information is preserved. By extracting the phase from the peak values in the frequency domain, a comparison can be made between the four channels. Taking the difference of these phases enables a calculation of the AOA (see Section 3.4.2). This algorithm performs well when there are moving targets present. The tests show that it performs better than other algorithms when clutter is the primary interference.

The radar data for the following test is taken at a park containing trees and power lines within 40 m of the radar. Figure 3.13 is generated by using a CFAR algorithm for the detection process and a max power digital beamformer for the angle estimate. Two main clutter sources
can be seen at 10m and 30m north of the radar. Figure 3.14 is generated by combining pulse-Doppler to separate and detect the moving target from the clutter and phase comparison on the four channels to calculate the angle estimate. The benefits of using Doppler processing can be seen by comparing the two figures. First, the clutter at 10m and 30m is not detected by the pulse-Doppler processing. Second, when using the max power beamformer, the clutter near 30 meters appears to be more significant and affects the angle estimate. The pulse-Doppler angle estimates are consistent because they do not use the return from the clutter. Third, pulse-Doppler processing is able to detect the target closer to 40m in Figure 3.14 whereas the clutter overpowers the return and raises the CFAR threshold in Figure 3.13. This all comes at the cost of increased complexity and computation.

Figure 3.13: Target detection using CFAR and max power beamformer with clutter present. The clutter is detected 10 and 30 meters north of the radar. The data consists of a target running out to 40 m then turning around and running back. The target’s AOA estimates near 40 m are inaccurate.
Figure 3.14: Target detection using Pulse Doppler and phase comparison in the presence of clutter. Pulse Doppler does not detect clutter returns and phase-comparison monopulse only uses the target return for AOA estimation.

3.7 Additional Functionality

The goal of the LATIS project is to use multiple radar stations to form a network and create a reliable drone testing area. To establish a reliable testing area, it is important to know the location of each radar and the direction it is facing. These requirements introduced the addition of an inertial measurement unit (IMU) and a GPS module.

3.7.1 Inertial Measurement Unit

An IMU can create an electronic compass which can give the heading of the device from measured data. An IMU uses an accelerometer and a magnetometer to measure the gravitational forces and the magnetic fields acting on the chip. Both the accelerometer and magnetometer have three-axis measurements and return vectors for $x$, $y$, and $z$. These measurements can then be used to estimate the roll, pitch, and yaw with respect to magnetic north. The following equations come from [29].
The roll, $\phi$, is found by comparing the accelerometer data, $G$, in the y and z direction

$$\tan(\phi) = \left( \frac{G_y}{G_z} \right).$$  \hfill (3.28)

The pitch, $\theta$, accounts for the roll and is found with

$$\tan(\theta) = \left( \frac{-G_x}{G_y \sin(\phi) + G_z \cos(\phi)} \right).$$  \hfill (3.29)

Because the yaw, $\psi$, is with respect to magnetic north, the estimation uses the magnetic field measured by the magnetometer, $B = [B_x, B_y, B_z]^T$. The calculation also incorporates the roll and the pitch estimates. The yaw is calculated by

$$\tan(\psi) = \frac{(B_x - V_x) \sin(\phi) - (B_y - V_y) \cos(\phi)}{(B_x - V_x) \cos(\theta) + (B_y - V_y) \sin(\theta) \sin(\phi) + (B_z - V_z) \sin(\theta) \cos(\phi)},$$  \hfill (3.30)

where the vector $V = [V_x, V_y, V_z]^T$ represents hard-iron interference and/or sensor offset and is independent of orientation. We attempt to estimate and remove $V$ using the following calibration method from [30].

**Magnetometer Calibration**

$B_{geo}$ is the scalar geomagnetic field strength defined by $\sqrt{B_{geo_x}^2 + B_{geo_y}^2 + B_{geo_z}^2}$. In the presence of hard-iron interference the following equation is true

$$(B - V)^T (B - V) = B_{geo}^2.$$  \hfill (3.31)

Expanding this equation and replacing unknown values with estimates yields

$$\epsilon[i] = B_x[i]^2 + B_y[i]^2 + B_z[i]^2 - 2B_x[i]\hat{V}_x - 2B_y[i]\hat{V}_y - 2B_z[i]\hat{V}_z + \hat{V}_x^2 + \hat{V}_y^2 + \hat{V}_z^2 - \hat{B}_{geo}^2,$$  \hfill (3.32)

where $\epsilon[i]$ is the error for the i-th measurement. It is much simpler to express this in matrix notation as

$$\epsilon = Y - X\beta,$$  \hfill (3.33)
where \( Y, X, \) and \( \beta \) are defined as

\[
Y = \begin{bmatrix}
B_x[0]^2 + B_y[0]^2 + B_z[0]^2 \\
\cdots \\
B_x[M-1]^2 + B_y[M-1]^2 + B_z[M-1]^2
\end{bmatrix}
\]

\[
X = \begin{bmatrix}
B_x[0] & B_y[0] & B_z[0] & 1 \\
\cdots & \cdots & \cdots & 1 \\
B_x[M-1] & B_y[M-1] & B_z[M-1] & 1
\end{bmatrix}
\]

\[
\beta = \begin{bmatrix}
2\hat{\nu}_x \\
2\hat{\nu}_y \\
2\hat{\nu}_z \\
\hat{B}_{geo}^2 - \hat{\nu}_x^2 - \hat{\nu}_y^2 - \hat{\nu}_z^2
\end{bmatrix}
\]

with \( M \) equal to the number of samples. A least squares optimization is applied to minimize the mean squared error and solve for an estimate of \( \beta \). This optimization is computed by

\[
\beta = (X^TX)^{-1}X^TY. \tag{3.34}
\]

Because \( X \) and \( Y \) are known, this is an easy equation to solve. The hard-iron components, \( V \), are then taken from \( \beta \) and inserted into Equation (3.30). For this calibration method to work properly, the \( M \) samples should be taken at different roll and pitch angles. If all the samples are taken at the same orientation, this only estimates the sensor noise.

Using this calibration method with IMU data results in accurate compass headings. The accuracy of the results is determined by comparing IMU headings with those reported by a phone’s compass. The comparison shows that the IMU is accurate to within several degrees across a full range of yaw values. When estimating compass headings without the calibration, the results are not correct. The IMU was tested at multiple angles from 0-360 deg, and the estimate only re-
turned a few distinct values. This is a result of stronger unwanted magnetic fields dominating the measurements.

3.7.2 GPS

To increase the functionality of the UTM system, a GPS module has been added to the radar to provide location and timing data. The GPS location data works with the IMU data to provide an accurate representation of the field of view (FOV). Our current GPS module can provide accuracy to within a couple meters. To achieve highly accurate location data, real-time kinematic (RTK) positioning can be used. This is a technique that enhances the precision of GPS and enables the location accuracy to be within 2 centimeters.

The GPS is used to assist in synchronizing multiple ground stations. The GPS provides a pulse per second (PPS) that can be used to trigger events and it provides the coordinated universal time (UTC). We use the GPS to report the UTC and date. This enables the radar data to be transmitted with a time stamp. This allows the ground stations to receive and process data that has been recorded at the same time.
CHAPTER 4.  ADAPTIVE BROADBAND BEAMFORMING SYSTEM FOR NAVAL COMMUNICATIONS

4.1 Introduction

The goal of the Adaptive Broadband Beamforming System for Naval Communications is to create an instrument for more reliable wireless communication in hostile environments. To accomplish this goal, the following actions will be completed.

- Build a multichannel, broadband array signal processing platform
- Implement algorithms for real time beam steering and interference cancellation
- Demonstrate the platform and algorithms for improved communications in an RFI environment

Flexibility in the design allows for different number of antennas and frequency channels. The proposed configurations can be seen in Table 4.1.

Table 4.1: Different configurations for the Adaptive Broadband Beamforming System for Naval Communications project. The focus of the authors work was on the medium size array.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>High BW</th>
<th>Medium Size Array</th>
<th>Large Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of antennas</td>
<td>9</td>
<td>18</td>
<td>36</td>
</tr>
<tr>
<td>Beamforming bandwidth</td>
<td>300 MHz</td>
<td>150 MHz</td>
<td>75 MHz</td>
</tr>
<tr>
<td>ADC Sample Rate</td>
<td>800 MHz</td>
<td>400 MHz</td>
<td>200 MHz</td>
</tr>
<tr>
<td>Number of frequency channels</td>
<td>768</td>
<td>96</td>
<td>192</td>
</tr>
</tbody>
</table>

The Adaptive Broadband Beamforming System for Naval Communications project can be broken up into 3 main parts: (1) the analog frontend consisting of the antennas, RF/IF signals, (2) the firmware executed on the SNAP board, and (3) the beamforming backend performed on the HPCs.
My main focus for the Adaptive Broadband Beamforming System for Naval Communications has been designing the firmware executed on the SNAP board and testing the backend. My main contribution is building a working firmware design that prepared packets for frequency domain digital beamforming. These designs are based on the medium size array configuration. A brief description is given for the RF frontend and beamforming backend to give the reader a comprehensive overview of the system.

![Figure 4.1: Adaptive Broadband Beamforming System for Naval Communications project setup. The rack holds the 4 HPCs, a 1 Gb switch, and a 40 Gb switch.](image)

### 4.2 RF/IF Frontend

The frontend consists of patch antennas tuned to X band. The RF signal is in the frequency range 10.2-10.4 GHz. Once received, the signal is amplified with an LNA, filtered, and mixed down to IF. The resulting IF frequency is in the 2nd Nyquist Zone as seen in Figure 4.2.
Figure 4.2: The IF frequency is in the 2nd Nyquist zone. The 50 MHz of transition band (25 MHz on each side) is discarded in the firmware by only keeping 96 out of the 128 frequency bins.

4.3 SNAP Board and Firmware

The sampling of the IF frequency is done by the Smart Network ADC Processor (SNAP) board, see Figure 4.3. Each SNAP board contains three Analog Devices HMCAD1511 ADCs. Each ADC can sample 1, 2, or 4 inputs. This means that each SNAP board can handle up to 12 signal inputs. The SNAP board has two 10 Gigabit Ethernet (10 GbE) cores. These cores are the limiting factor in terms of data rate because the ADCs can sample up to 1000 MSPS and have a sampling resolution of 8 bits. The SNAP board contains a Xilinx Kintex7 FPGA (XC7K160T-2FFG676C).

The firmware on the FPGA was generated by using CASPER tools and Xilinx blocks within System Generator. Once the IF signals are sampled the FPGA implements an FFT to split the wideband communication signal into narrower sub bands. These sub bands are then transported from the SNAP board to the HPC via the 10GbE where they are used for beamforming.

For the modified medium size array configuration there are 6 antennas per SNAP board. The ADCs are sampling at 400 MHz to achieve beamforming on 150 MHz bandwidth. The FPGA is running at 200 MHz, so it recieves two ADC samples per clock. The firmware can be broken up into 4 main parts: the F-engine, bit reduction, packetizer, and 10 gigabit ethernet core. A block diagram of these subsystems can be seen in Figure 4.4. The packetizer can futher be broken up into three main parts as explained in Section 4.3.3.
Figure 4.3: The SNAP Board contains 3 ADCs, 2 10 GbE SFP+ ports, and an Xilinx Kintex7 FPGA. This board samples the signals, implements the F-engine, puts the data in specific packets, then sends the data to the HPCs. [1]

### 4.3.1 F-Engine

To accelerate the development and testing of the system, it was decided that the F-engine would only consist of an FFT instead of a polyphase filter bank (PFB). A PFB has benefits of reduced spectral leakage and scalloping loss but only allows for near perfect reconstruction. Using an FFT allows for perfect reconstruction of a signal.
A 256-point FFT is used and since the input to the FFT is real, we discard 128 bins because of symmetry. Out of the 128 remaining frequency bins, only 96 are kept. By discarding 32 bins we remove 50 MHz of bandwidth (25 MHz on each side). The FPGA is clocked at 200 MHz which means the input to the FFT gets 2 samples per clock. CASPER’s FFT wideband real takes advantage of this by implementing a biplex FFT that handles parallel data. The output of the FFT is one frequency bin per clock starting at bin 0 and ending with bin 127. The FFT is streaming so the output immediately after bin 127 is frequency bin 0 of the next time window.

### 4.3.2 Bit Reduction

The output of the FFT is 36 bits per clock for each antenna for a total of $36 \times 6 = 216$ bits per clock. This yields a bit rate of $200 \text{ Mclock/s} \times 216 \text{ b/clk} = 43.2 \text{ Gbps}$. This is too high since there are only two 10 GbE ports on each SNAP board. To fix this, the data is passed through a bit reduction subsystem. This subsystem takes the 36 bits (18/18, real/imag) and reduces them to 16 bits (8/8, real/imag). The user can select which 8 bits to keep by writing to the register $\text{lsb\_select}$ (see A.2). The slicing of the 8 bits includes rounding. This reduces the data rate to $200 \text{ Mclock/s} \times 96 \text{ b/clk} = 19.2 \text{ Gbps}$. The two 10 GbE cores could handle this but it is better to operate at slightly lower, so we only keep 96 out of the 128 frequency bins. This results in a final bit rate of $200 \text{ Mclock/s} \times 96 \text{ b/clk} \times 96/128 = 14.4 \text{ Gbps}$. Each 10 GbE must handle at least 7.2
Gbps to keep up with the data rate coming out of the FFT. The bit reduction is implemented as explained and designed in [2].

4.3.3 Packetizer

![Packetizer block diagram](image)

Figure 4.5: Packetizer block diagram. The packetizer is responsible for creating 12 packets that each contain 8 frequency bins. The packetizer also maintains that the data rate transmitted by each 10 GbE core is above 7.2 Gbps.

The packetizer comes after the Bit Reduction subsystem (Section 4.3.1) and consists of 3 main parts: splitting the 96 frequency bins into 12 packets containing 8 bins each, organizing the data into 64-bit words, writing these packets to the 10 GbE core, and transmitting them to the HPCs.

Combiner

Coming out of the FFT/Bit Reduction is 16 bits per frequency bin for each antenna. For 6 antennas these bits are concatenated together which equals 96 bits/clock. This is equivalent to 1.5 words (word = 64 bits). When writing data to the 10 GbE core, the data must be a 64-bit word. For convenience in formatting the data into 64-bit words, every two clocks the data is concatenated together to form 192-bit data. This is done by toggling the data between two registers and concatenating every other clock. The state machine in Figure 4.7 controls the enable signal for these registers. This state machine also splits up the data into 12 192-bit FIFOs. Each FIFO
contains 4 192-bit elements which is 8 frequency bins. The state machine also discards the first and the last 16 frequency bins.

**Parallel to Serial**

The next main part of the packetizer takes the data from the 192-bit FIFOs and formats it into 3 64-bit words. When data is written to one of the twelve 192-bit FIFOs, the state machine in Figure 4.6 is triggered. This state machine loads the 192-bit data into a CASPER block then shifts 64-bit words on three consecutive clocks. As the 64-bit words are shifted out they are written to a jumbo FIFO. For each time window, each jumbo FIFO receives 12 words of data.

The 10 GbE core transmits UDP over IPv4 packets. Each packet is wrapped in an Ethernet frame. The Ethernet frame requires 38 bytes, IPv4 requires 20 bytes, and UDP requires 16 bytes. So for each packet of data sent, the overhead cost is at least 74 bytes. To maximize data throughput, maximum packet length should be used. The 10 GbE core can send a maximum packet length of 1024 words. This motivates us to use packets of size 1021. There are 12 jumbo packets each containing 8 frequency bins. This means that 12 64-bit FIFOs are filled up with 1020 entries.

![Figure 4.6: Parallel to serial state machine. Used to break the data into 64-bit words and write to the jumbo FIFOs. The state machine controls the load and shift signals for the Casper block parallel to serial.](image-url)
Figure 4.7: Combiner state machine. This state machine takes the output of the bit reduction subsystem and assigns 8 frequency bins to 12 different FIFOs.
Transmitter

The data from the 12 jumbo FIFOs must be transmitted by the 10 GbE core one at a time and maintain a transmission data rate of at least 7.2 Gbps. This step uses the state machine in Figure 4.8 to enable reading from the jumbo FIFOs and writing to the 10 GbE core. It also handles selecting the data line that feeds the GbE core through the use of a MUX.

This state machine’s start is triggered as soon as the first jumbo FIFO contains 1020 elements. WRITE1 starts writing to the 10 GbE core and writes for 1021 consecutive clocks. A MUX selects which data appears at the GbE core input data port. A one word header and 1020 words of data are written to the core, creating a jumbo packet. After the 1021 words are written, an end of file (eof) signal is triggered to start transmission of the data from the core to the switch. We then wait 790 clocks for part of the GbE buffer to empty before we start filling it up again. This results in a data rate of \(\frac{1020}{(1021 + 790)} \times 200 \times 10^6 \times 64 = 7.209\) Gbps. This process continues until all 6 FIFOs have been written to the core and transmitted (6 FIFOs per 2 GbE cores). Then the state machine returns to the INIT state and waits for the first FIFO to have another 1020 words because the 10 GbE core is running slightly faster than the data coming out of the FFT.

Jumbo FIFO1 is sent to 10 GbE0, then jumbo FIFO2 is sent to 10 GbE1 and so on, alternating until all packets have been transmitted.

Packet Format

The packet header is formatted as seen in Table 4.2. The mcount is a number that indicates which collection time the packet corresponds to. Each of the 12 jumbo packets has the same mcount. Then mcount is incremented for the next set of 12 packets. Mcount is generated with an enabled counter, so it rolls over if the SNAP board is on for a long enough period of time. The XID is the GPU core that the packet is sent to (0-11). The 12 XIDs are hard coded into the firmware design with the first packet having an XID of 0, the second packet has XID 1 and so on. The FID represents which SNAP board the packet came from (0-2). This value is the same for all packets on the same SNAP board and is set by writing to the register FID. The FID allows the HPC to know which antennas the data came from (see Table 4.4). The XID and FID are summarized in Tables 4.3 and 4.4.
Figure 4.8: This is the state machine logic to load the 10 GbE cores and transmit the data to the HPCs. The data rate is maintained above 7.2 Gbps for each 10 GbE core by writing data for a set amount of clocks, and waiting for a set amount.
Table 4.2: Header format for data packets. The header is the first word in the packet.

<table>
<thead>
<tr>
<th>Bit Index</th>
<th>63 → 8</th>
<th>7 → 4</th>
<th>3 → 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>mcount</td>
<td>XID</td>
<td>FID</td>
</tr>
</tbody>
</table>

Table 4.3: XID interpretation. The XID specifies which FFT bins the packet contains.

<table>
<thead>
<tr>
<th>XID</th>
<th>Bins</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16-23</td>
</tr>
<tr>
<td>1</td>
<td>24-31</td>
</tr>
<tr>
<td>2</td>
<td>32-39</td>
</tr>
<tr>
<td>3</td>
<td>40-47</td>
</tr>
<tr>
<td>4</td>
<td>48-55</td>
</tr>
<tr>
<td>5</td>
<td>56-63</td>
</tr>
<tr>
<td>6</td>
<td>64-71</td>
</tr>
<tr>
<td>7</td>
<td>72-79</td>
</tr>
<tr>
<td>8</td>
<td>80-87</td>
</tr>
<tr>
<td>9</td>
<td>88-95</td>
</tr>
<tr>
<td>10</td>
<td>96-103</td>
</tr>
<tr>
<td>11</td>
<td>104-111</td>
</tr>
</tbody>
</table>

Table 4.4: FID meaning. The FID specifies which antennas the packet data came from.

<table>
<thead>
<tr>
<th>FID</th>
<th>Antennas</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1→6</td>
</tr>
<tr>
<td>1</td>
<td>7→12</td>
</tr>
<tr>
<td>2</td>
<td>13→18</td>
</tr>
</tbody>
</table>

Each data packet has a format as seen in Table 4.5. With 1.5 words per bin and 8 bins per time window, 85 time windows are sent to achieve jumbo packets of length $12 \times 85 = 1020$ words plus a one word header.

**Starting the Firmware**

The firmware design relies on a single sync pulse to start the FFT and the combiner state machine. This sync pulse is generated by writing a ‘1’ to the sync register. A python script that runs on the HPC connects to the Raspberry Pi and programs the FPGA with a .fpg file. The script then initializes the ADCs to the correct sampling rate and number of channels. It sets up the two 10 GbE cores including the destination IP address and port. Finally, it triggers the sync pulse by writing to the sync register.

In addition, there is an “ARM” register that is used with a 1 PPS signal that can generate a starting pulse. This allows for multiple SNAP boards to be synchronized and to start recording data and sending packets at the same time. To make this possible, all the SNAP boards are connected to the same 10 MHz reference and 1 PPS. The ARM/PPS signal is “OR”ed with the sync signal to allow the user to run a SNAP board independently or as a group without having to change the firmware.
Table 4.5: Packet Format for 6 antennas. Each packet contains 8 frequency bins from 6 antennas for 85 time windows.

<table>
<thead>
<tr>
<th>$t_1$</th>
<th>$t_{85}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>63 → 48</td>
<td>47 → 32</td>
</tr>
<tr>
<td>31 → 16</td>
<td>15 → 0</td>
</tr>
<tr>
<td>Antenna 1, Bin 0</td>
<td>Antenna 2, Bin 0</td>
</tr>
<tr>
<td>Antenna 5, Bin 0</td>
<td>Antenna 6, Bin 0</td>
</tr>
<tr>
<td>Antenna 3, Bin 1</td>
<td>Antenna 4, Bin 1</td>
</tr>
<tr>
<td>Antenna 1, Bin 2</td>
<td>Antenna 2, Bin 2</td>
</tr>
<tr>
<td>Antenna 5, Bin 2</td>
<td>Antenna 6, Bin 2</td>
</tr>
<tr>
<td>Antenna 3, Bin 3</td>
<td>Antenna 4, Bin 3</td>
</tr>
<tr>
<td>Antenna 1, Bin 4</td>
<td>Antenna 2, Bin 4</td>
</tr>
<tr>
<td>Antenna 5, Bin 4</td>
<td>Antenna 6, Bin 4</td>
</tr>
<tr>
<td>Antenna 3, Bin 5</td>
<td>Antenna 4, Bin 5</td>
</tr>
<tr>
<td>Antenna 1, Bin 6</td>
<td>Antenna 2, Bin 6</td>
</tr>
<tr>
<td>Antenna 5, Bin 6</td>
<td>Antenna 6, Bin 6</td>
</tr>
<tr>
<td>Antenna 3, Bin 7</td>
<td>Antenna 4, Bin 7</td>
</tr>
</tbody>
</table>

4.3.4 Results

It is important for the firmware to generate correct FFT bin outputs and to correctly packe-tize them for the HPCs. To test this I used a signal generator to produce a 81.25 MHz sine wave (Bin 52). This sine wave was split into 6 copies and fed into the SMA connectors on the SNAP board. I started the firmware and packets were being sent to the HPCs. Using Wireshark, a packet analyzing tool, I captured and saved the packet data. To get a full spectrum I took one packet with each of the 12 XIDs. Using Matlab I parsed each of the 12 packets and averaged over the 85 time windows. Figure 4.9 shows the results and as can be seen matches the expected output of the FFT.
Figure 4.9: SNAP board packet data with an 81.25 MHz sine wave (bin 52) going into each antenna SMA. This is the average of the 85 time windows in a single packet.

### 4.4 Network Setup

Figure 4.10 shows the setup for all the networking. The Figure includes the IP addresses for all the used ports. All data related IP addresses have the prefix 10.17.16.0/24 while the management related IP addresses use the prefix 192.168.4.0/24. The interfaces with 10.2.117.(85-88) go out to BYU’s network. Some relevant files include:

```
\etc\sysconfig\network-scripts\ifcfg-enp59s0
\etc\hosts
\etc\dnsmasq.conf
```

### 4.5 Data Center Switch

An Arista DCS-7050QX-32-R data center switch is used to route the packets to their desired location. This allows each HPC to get the same FFT bins from each SNAP board. When each
Figure 4.10: Network setup for the Adaptive Broadband Beamforming System for Naval Communications project. Includes: 4 HPCs, 3 SNAP boards, 3 Raspberry Pis, 1 Arista switch, 1 TP-Link switch.

packet is transmitted by the SNAP board, the firmware wraps the data with a header that contains the source and destination MAC address, IP address, and port. The packetizer was designed so that each of the 12 packets has a different destination IP address which can be changed by writing to software registers in python. These IP addresses also affect the destination MAC addresses by the use of an address resolution protocol (ARP) table. The ARP table associates a destination IP address with a destination MAC address. This table can be set in the python code which is then implemented in the firmware. Doing this requires you to take the MAC address from the HPC NIC ports and code them into the python script. When finished, the SNAP board ARP table should look similar to Table 4.6 (with 16 entries), where the MAC Addresses are from the NIC interfaces on the HPC and found using the command `ifconfig` at the command line.

The Arista switch uses layer 2 switching which means that packets are sent to specific switch ports based on destination MAC addresses. To configure the switch, MAC addresses that match the destination MAC addresses need to be added to the MAC address table. The MAC address table associates a MAC address with a specific port on the switch. The MAC address table is dynamically updated, but for more reliable data transfer we specify static addresses. The reason
is that dynamic addresses only stay in the MAC address table for a short time (~5 minutes) and if
the table is not continuously updated, the packets are not switched correctly.

To summarize the FPGA and switch operation: if a packet has a desired destination of
10.17.16.4 (which is the IP address for enp59s0 on the HPC), enp59s0 must be connected to one
of the switch’s ports (Ex. Et5/1). The ARP table in the firmware assigns a destination MAC
address of 24:8a:07:a3:fc:5c. Then we assign a static MAC address of 24:8a:07:a3:fc:5c to the
interface Et5/1. Now all packets with a destination IP of 10.17.16.4 are switched to port Et5/1
which is connected to the HPC’s enp59s0. The switch’s command line interface (CLI) syntax to
accomplish this behavior is:

localhost: enable
localhost# config
localhost(config)# mac address-table static 248a.07a3.fc5c vlan 1 interface Eth 5/1.

When configured, the MAC address table should look similar to Table 4.7, depending on
the ports that the SFP+ cables are plugged into. A completed table would have 12 static entries, one
for each destination IP. The two DYNAMIC addresses are associated with ports that are connected
to the SNAP board’s 10 GbE interface.

Table 4.7: MAC address table for the Arista network switch.

<table>
<thead>
<tr>
<th>Vlan</th>
<th>MAC Address</th>
<th>Type</th>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>248a.07a3.fc5c</td>
<td>STATIC</td>
<td>Et5/1</td>
</tr>
<tr>
<td>1</td>
<td>248a.07a3.fc5d</td>
<td>STATIC</td>
<td>Et5/2</td>
</tr>
<tr>
<td>1</td>
<td>506b.4b6d.5bb0</td>
<td>STATIC</td>
<td>Et5/3</td>
</tr>
<tr>
<td>1</td>
<td>506b.4b6d.5bb1</td>
<td>STATIC</td>
<td>Et5/4</td>
</tr>
<tr>
<td>1</td>
<td>0202.0000.0014</td>
<td>DYNAMIC</td>
<td>Et9/1</td>
</tr>
<tr>
<td>1</td>
<td>0202.0000.0015</td>
<td>DYNAMIC</td>
<td>Et9/2</td>
</tr>
</tbody>
</table>
4.6 Beamforming Backend

Once the packets of data go through the network switch, they arrive at the HPCs. The data is then captured using High Availability Shared Pipeline Engine (HASHPIPE) [8, 9]. HASHPIPE uses plugins that reorder, correlate, and beamform the data.

The final demo for this project consists of the following: an antenna array receives a wideband communication signal coming from some angle while simultaneously receiving interference coming from a different angle. The data is sampled, split up into narrowband sub channels, beamforming is applied to remove interference, then the signal is reconstructed and demodulated. Up to this point in the project we can: sample signals, put 85 time windows into packets with 8 frequency bins each, capture the data with HASHPIPE, apply beamforming, and write the result to a file.

To test the system with the beamformer, two different weights were generated. The first beam’s weights were set as all ones to maximize the power of a signal arriving at 0 degrees. For the second beam, max SNR weights were generated assuming that the signal of interest (SOI) was coming from 40 degrees and the RFI was arriving at 0 degrees. A plot of the power (dB) of an arbitrary signal after beamforming vs the signal’s angle of arrival can be seen in Figure 4.11. These specific weights were chosen so that one beam would maximize the power of a signal arriving at 0 degrees, and the other beam would minimize that power. These were convenient weights to choose because the input into the SNAP board had characteristics of a signal arriving at an antenna array from bore sight (0 degrees). Figure 4.12 shows the results of the beamformed power using both weights. As can be seen by the values of the colorbars, the max SNR beam was successfully able to attenuate the signal.

After beamforming is performed on the selected frequency channels, it is desired to recover the transmitted data. In order for this to happen, the signal must be reconstructed from the output of the beamformer, then it must be demodulated. A fourth HPC is responsible for the reconstruction and demodulation of the signal.
Figure 4.11: Beamformer weights were calculated then multiplied with steering vectors at different angles. The simulated result is the normalized power of a beamformed signal at that angle. The max SNR weights were calculated assuming a SOI at 40 degrees and RFI at 0 degrees.

Figure 4.12: Beamforming applied to SNAP board packets. Two beams were created. The weights on the left were set to all ones to maximize the power. The plot on the right used max SNR weights that treated the input as interference. The beamformer was successfully able to remove the interference as can be seen by the magnitude of the colorbars.
The next several years will see a large increase in UAS flights. UTM systems that can detect and track multiple drones will be invaluable in the testing and development of effective sense and avoid algorithms. This will further enable the development of UAS technology.

Implementation of phase comparison monopulse allows for faster AOA estimation. This is useful for larger antenna arrays, because it does not require a sweep across all angles. Computing the AOA estimate faster provides more time for other types of processing to take place.

Implementation of the MUSIC algorithm could prove invaluable when there are multiple targets. This algorithm is capable of detecting multiple targets in the same range bin and estimating their AOA.

Finally, implementing pulse-Doppler processing would be exciting in future hardware, because it is capable of clutter reduction and detecting moving targets. Post processing on actual radar data has shown that this can be a successful technique in drone detection.

With these algorithms implemented, the LATIS radar is able to provide more accurate target detection and estimation for use in UTM systems to help them be more effective in tracking and path planning.

The Adaptive Broadband Beamforming System for Naval Communications project has successfully implemented a digital beamforming backend. This includes sampling a signal at 400 MHz and taking the FFT. Then creating 12 packets with eight frequency bins in each for 85 time windows. These packets are sent to three separate HPCs, where they are collected by HASHPIPE and processed by a beamforming plugin. The beamformed output is then written to a file.
5.1 Future Work

As the Adaptive Broadband Beamforming System for Naval Communications and LATIS projects are ongoing, the future work may be tasks that are currently being accomplished or will be attempted in the near future.

The Adaptive Broadband Beamforming System for Naval Communications is progressing in its development. One of the main tasks remaining is the demodulation of the communication signal. The beamformer output from three HPCs will be sent to the fourth HPC. This HPC will reconstruct the time domain signal through the use of an IFFT. Demodulation of the time domain signal can then be implemented. A future improvement could include, replacing the FFT in the firmware with an oversampled PFB. This would give benefits like improved spectral leakage and scalloping loss, while still allowing for perfect reconstruction of the signal.

When the medium size array is complete and functioning, work can be done on the high bandwidth array as seen in Table 4.1. This configuration would double the bandwidth of the system while using half of the antennas.

For the LATIS project, when the ADC board for the 16 antennas is complete, the various algorithms will need to be implemented and tested. Testing these algorithms can determine which ones perform best for this system.
REFERENCES


APPENDIX A.

This appendix covers topics relating to the SNAP board. Topics include starting the firmware, using the firmware’s registers, helpful hints, and System Generator designs.

A.1 SNAP Board Startup

The following command line instructions can be used to start the SNAP board firmware.

you@clyde$ conda activate casper-dev
you@clyde$ ipython

Example running all SNAP boards from ipython

%run <python script> -b <fpg file>

In[1]: %run snap_all.py -b no_pfb_6_ant/outputs/Latest.fpg
In[2]: snap1.write_int('arm', 1); snap2.write_int('arm', 1);
...snap3.write_int('arm', 1)

A.2 SNAP Board Registers

The following list contains the registers that are currently implemented in the SNAP board firmware. For each register, a description is given and instructions on how to use it are listed.

lsb_select - This register sets which bit will be the LSB coming out of the bit reduction. Valid values include 0-10. snap1.write_int('lsb_select', 7)

arm - This register is used to start the firmware with a 1 PPS signal. When running multiple SNAP boards connect a 1 PPS signal to the 2nd SMA (see Figure 4.3). Then write a ‘1’ to the register. snap1.write_int('arm', 1)
sync - The sync register can also be used to start the firmware. This is convenient when you only want to run one SNAP board. This does not require a 1 PPS signal. To start the firmware with the sync: snap1.write_int('sync', 1); time.sleep(1); snap1.write_int('sync', 0);

dest_ip_pack(1-12) - These 12 registers select the destination IP address for the 12 XID packets. ip_base = 10*(2**24) + 17*(2**16) + 16*(2**8).
snap1.write_int('dest_ip_pack1',ip_base+4)

FID - This specifies which SNAP board the packet comes from. Valid values are: 0,1,2. Each SNAP board gets a unique FID. snap1.write_int('FID', 0)

rst and rst1 - These registers reset the 10 GbE cores. They are 2 bit registers. One bit resets the core and the other resets a debug counter. snap1.write_int('rst', 3);
snap1.write_int('rst', 0)

dest_port(0-1) - This sets the destination UDP port.
snap1.write_int('dest_port0', 60000)

packet_trigger - This register can be used to stop the 10 GbE cores from transmitting. Set this value high before starting the firmware. To stop transmitting write:
snap1.write_int('packet_trigger', 0). To start transmitting again simply write:
snap1.write_int('packet_trigger', 1)

mcnt_rst - This can be used to reset the mcount. Useful when testing HASHPIPE because it expects mcount 0. The mcount reset is synced to the 1 PPS. To use it write:
snap1.write_int('mcnt_rst', 1); snap2.write_int('mcnt_rst', 1);
...snap3.write_int('mcnt_rst', 1); time.sleep(.9); snap1.write_int('mcnt_rst', 0);
...snap2.write_int('mcnt_rst', 0); snap3.write_int('mcnt_rst', 0);

A.3 SNAP Board Notes

The following is a list of set backs and lessons learned while working with the SNAP board.

- I had to change to a 3 Amp power supply. The 2 Amp one did not provide enough power to the Raspberry Pi (causing it to reboot) and failed while trying to initialize the ADC chips. Under normal operation the SNAP board draws around 1.7 Amps at 12 Volts.
• The packetizer sends jumbo packets. If you do not configure the network to receive these jumbo packets they won’t be captured. In /etc/sysconfig/network-scripts/ifcfg-enp59s0 add the line MTU=9000. Then type on the command line ifdown enp59s0, and ifup enp59s0 so that the changes made will be placed into effect. You must do this for each of the interfaces being used.

• To clock the FPGA at 200 MHz you need to set the User IP Clock source to adc0_clk within the SNAP block in Simulink. If you select sys_clk, the design is compiled with a clock rate of 100 MHz even if you have specified a different User IP Clock Rate.

• All communication with the SNAP board is done through the Raspberry Pi 3 B+. When you turn on the SNAP board, you can have the FPGA boot from flash or have the Pi program it with Python.

• The Raspberry Pi 3 B+ image from CASPER is around 16 GB. This won’t fit on a 16 GB SD card. The Pi is configured to PXE boot from Clyde, so that the SD card is no longer needed.

• CASPER blocks within Simulink require the Fixed-point designer toolbox in Matlab. Without that package, errors will occur when trying to simulate most designs.

• When connecting a SFP+ cable to the SNAP board and HPC, an LED should come light up on the HPC indicating a good connection. If this does not happen, there may be a bad cable/port or possibly the wrong brand of cable.

• When you first get the SNAP boards, the power controller chips are not programmed. You must use a Texas Instruments programmer to program these. Instructions are on the website SNAP board bringup.

A.4 SNAP Board Firmware Design

The following Figures are screen shots of the firmware design from System Generator.
Figure A.1: Signal flow diagram for the sync generator. The sync generator starts the subsystems within the firmware design. A sync can be generated by writing a ‘1’ to the sync register, or by writing a ‘1’ to the arm register with the use of a 1 PPS signal.
Figure A.2: Diagram of the firmware F-engine. The signals are sampled by the ADC then sent to a CASPER FFT wideband real. After the FFT the data is sent to bit reduction [2].

Figure A.3: Combiner subsystem. The combiner subsystem takes consecutive FFT outputs and concatenates them, creating a 192 bit data which is then sent to the packet setup subsystem.
Figure A.4: Parallel to serial. The parallel to serial system takes 192 bit data and splits it up into 3 64 bit words which it then writes to a jumbo FIFO.

Figure A.5: Packet setup subsystem. The packet setup subsystem uses a CASPER parallel to serial converter block to split the data up.
Figure A.6: 10 GbE core. This is the CASPER 10 GbE core. LEDs provide feedback on the connectivity and transmission of the core.
Figure A.7: 10 GbE data selection. The MUX selects the signals and data that goes to the GbE core.

Figure A.8: Packet header. Shows the creation of the 64 bit header. For this case, this is the header for XID 0. The mcount is 56 bits, XID is 4 bits, and FID is 4 bits.
Figure A.9: Destination IP address. The MUX selects the destination IP address for each packet. The output of this goes to the 10 GbE core. The select line comes from the packetizer state machine.