Design and Implementation of an RFI Direction Finding System for SKA Applications

James Zekkai Middlemost Gowans

A dissertation submitted to the department of Electrical Engineering, University of Cape Town in fulfilment of the requirements for the degree of Masters of Science in Electrical Engineering

Supervisor:
Prof. A. J. Wilkinson.

October 7, 2020

University of Cape Town

The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

University of Cape Town

Declaration

1. I know the meaning of plagiarism and declare that all the work in the document, save for that which is properly acknowledged, is my own. own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.
5. This thesis/dissertation has been submitted to the Turnitin module (or equivalent similarity and originality checking software) and I confirm that my supervisor has seen my report and any concerns revealed by such have been resolved with my supervisor.

Signature:..... Signed by candidate .....
J. Z. M. Gowans

Date: 2019-05-26

Acknowledgments

I would like to thank my supervisors at the University of Cape Town, Prof Andrew Wilkinson, Prof Daniel O'Hagan and Prof Michael Inggs for defining this project, working with me through the implementation, and assisting during the write up.

The technical support and equipment, and FPGA design expertise provided by the SKA/MeerKAT digital backend team was indispensable. Specifically Jason Manley, Marc Welz, Andrew Martens, and Wes New: your assistance in the technical aspects of this project was amazing. From getting ROACH hardware, to setting up a Simulink design environment, debugging timing issues in compiles and resource optimization. I would not have been able to build the systems I did without your support. Marc specifically for working with me to integrate with the KATCP protocol and do data transfer between the FPGA and the computer.

Also from the SKA, thanks to Simon Norval for taking me to site on two occasions to conduct field trials and take real measurements. Your support of this project made it all the more worthwhile and enjoyable for me.

Thanks to Thomas Abbot for assisting with simulations of the field trials; this was key to generating a high degree of confidence in the field trials and the results.

The support of the SKA in general for allowing me to use their office space and drink their coffee is appreciated.

To my fellow students in the Radar and Remote Sensing Group at UCT, your assistance in setting up equipment is appreciated. Especially Stephen Paine for your indispensable assistance in field trials. Stephen, your time and technical knowledge helped greatly in making this project successful.

Abstract

For radio astronomy telescopes to be able to perform observations of weak signals from space, they need to operate in a radio-quiet environment. Any radio frequency interference (RFI) will interfere with the ability of the telescope to collect data. With the proliferation of electrical and electronic devices, RFI management is one of the major challenges facing radio astronomy reserves.

This thesis details the design, construction and testing of a system which is able to find the direction which a source of interference is coming from. User requirements for the system are captured, and one of the key requirement of the system is the ability to direction-find two classes of RFI: weak narrow-band continuous signals, and strong impulsive signals. Both of these classes of signals pose problems for radio telescopes. The primary focus of the thesis is implementing the algorithms to direction find those signals, and to evaluate whether the algorithms perform as expected on real RFI sources in the field.

An analysis of various prior direction finding techniques is done from the existing literature to select the most suitable technique for this system. A combination of phase interferometry and time difference of arrival is selected, due to their suitability for the classes of signals, the operating environment and the hardware that will be used. Simulations are done showing how the system should operate and to highlight potential challenges. A key challenge is around phase ambiguity, and special attention is paid to mitigating this.

After design and simulation, a full system is implemented containing a number of subsystems linked together. A four element deformed circular antenna array and RF front end pick up signals from the environment. These signals are digitised together in phase by fast analogue to digital converters (ADCs). The output of the ADCs goes into a field programmable gate array (FPGA) on a ROACH development board which does high speed DSP including Fourier transforms, spectrum cross correlations, accumulations, power detections and time domain capturing. The output of the Digital Signal Processing (DSP) done on the FGPA is received by a computer running a Python application which performs the final angle of arrival calculations in real time. The application has a mathematical model of the antenna array which it combines with the received baseline time difference or phase shift measurements to ascertain the direction of the signal source.

When designing and building the system, emphasis is put in making it flexible and reconfigurable, allowing it to be used with arbitrary array configurations or frequency ranges.

The system is first put together and tested in the lab using signal generators, noise sources and impulse generators. These signals are fed into the ROACH board to simulate an RF environment and hence ensure that the design is working as expected. Next, the system is made portable and taken out for field trials. The field trials demonstrate that the system is able to provide accurate tracks for a number of different RFI sources, both impulsive and narrowband. It is able to maintain a track over the full 360° field of view as required.

Contents

Glossary

ADC analogue to digital converter. 21, 23, 25–28, 35, 36, 43, 47–49, 51–53, 55, 56, 65–67, 69, 75

AoA angle of arrival. 1, 4, 11, 14, 15, 24, 61, 63

BRAM block random access memory. 51

CASPER Collaboration for Astronomy Signal Processing and Electronics Research. 49

DF direction finding. 1–5, 7, 15, 21, 23–26, 28, 33, 35, 40, 52, 55, 63–65, 69, 74, 76

DFT discrete Fourier transform. 33

DRAM Dynamic Random Access Memory. 47, 49, 52, 53, 55, 69

DSP digital signal processing. 1, 4, 23, 33, 47, 49–51, 55, 56, 64

FFT fast Fourier transform. 23, 24, 28, 47, 49, 51, 53, 56

FPGA Field Programmable Gate Array. 4, 21, 28, 47, 49–52, 55, 56, 61, 63, 64, 68, 69

GPIO general purpose input/output. 53

HDL Hardware Description Language. 49

JSON JavaScript Object Notation. 36

LED light emitting diode. 53

LNA low noise amplifier. 43

LPDA log-periodic dipole array. 69

LPF low pass filter. 43

POI probability of intercept. 2, 4

RF radio frequency. 1, 3, 4, 18, 21, 35, 46, 56, 57, 65–68, 75

RFI radio frequency interference. 1–3, 5, 21, 23, 27, 28, 33, 69, 70, 74

RHEL Red Hat Enterprise Linux. 49

RMS root-mean-square. 27, 37, 40, 64

ROACH Reconfigurable Open Architecture Computing Hardware. 3, 4, 21, 23, 47, 49, 57, 61, 69, 71, 74–76

SKA Square Kilometer Array. 1, 3, 4, 47, 49, 56

SNR signal to noise ratio. 28, 56, 59

UCT University of Cape Town. 74–76

UML unified modeling language. 61

Chapter 1

Introduction

1.1 Background

MeerKAT is a 64-dish radio telescope aimed at being the precursor to the full Square Kilometer Array (SKA) radio telescope. At the time of writing, the MeerKAT array is in the process of being deployed in the Northern Cape of South Africa. MeerKAT is designed to be a highly sensitive telescope, able to detect very weak signals. It will operate over a few bands in the 0.6 GHz to 15 GHz range. The majority of the dishes are concentrated in the core of the array, which is set to cover an area about 1 km in diameter.

Like any radio telescope, MeerKAT requires a radio frequency (RF)-quiet environment. The presence of radio frequency interference (RFI) on site would interfere with the ability of the telescope to collect data. At the very least, RFI would swamp the weak signals which the telescope is attempting to receive, or worse, strong RFI could destroy the sensitive amplifiers and digitisers designed to cope only with extremely low power signals. It is for this reason that the array is being deployed in the Karoo, away from built-up areas and the RF signals that go with such environments.

However, there is always the possibility that RFI will manifest. Although all equipment taken to site is tested beforehand for RFI, if equipment malfunctions or is not configured correctly or shielded correctly, RFI may be introduced. Additionally, electrical or electronic systems away from site in nearby towns or farms may inadvertently emit RFI. Sources such as communications systems or electronics typically emit well defined narrowband signals, while malfunctioning electronics could emit spark-like discharges producing very short bursts of broadband noise (impulsive RFI). As such, RFI management systems must be in place to aid in the detection and amelioration of RFI in order to allow the telescope to function optimally.

The purpose of this research is to design a device capable of detecting RFI and performing angle of arrival (AoA) parameter estimation on the detected signals. The process of estimating the AoA of a signal is known as direction finding (DF). Knowing the AoA will allow the SKA team to track down where rogue RFI signals are coming from and to silence them.

1.2 User Requirements

The following user requirements for the system to be designed and built were drawn up by three senior engineers at SKA SA: the Director of Science and Engineering, Prof Justin Jonas, Systems Engineer for Infrastructure, Carel van der Merwe and digital backend digital signal processing (DSP) specialist, Dr Jason Manley:

1. A system to perform direction-finding of both impulsive and narrowband continuous wave RFI sources is to be designed.

2. Key deliverables of this project are a software package and a thesis report.

3. The software package should have the following functions:
   a) It should take input from a correlator. This could either be time domain cross-correlation for impulsive sources or frequency domain cross-correlation for continuous sources.
   b) It should parse a configuration file which contains information about the array configuration and information about the output from the correlator.
   c) The data from the correlator should be used to ascertain the direction of detected signals.
   d) The software should be designed to fit into a system which has a 100% probability of intercept (POI)
4. The software should have the following user interface features:
   a) The user should be able both to interact with it on a computer while it's running and to modify its operation.
   b) A spectrogram plot of frequency vs amplitude should be displayed to the user to serve as a monitoring device of the RFI environment.
   c) The user should be able to select a band of interest from the spectrum.
   d) The direction should then be computed for the signal in the band that had been selected.
   e) The result of the DF should be presented to the user. An investigation must be done into the best way to present this information to the user.
   f) Where appropriate, additional meta-information should be displayed to the user, such as measurement accuracy or signal strength.
5. The system should be designed to find terrestrial RFI sources.
6. The system should be designed to be location independent. It could either be deployed to a fixed location or as a mobile device deployed on a vehicle.
7. This project should be able to interface easily with other systems requiring its data. Specifically, it should be designed to interface with and pass its data on to an allied project which is doing classification of RFI.

Figure 1.1: Photo of the MeerKAT core and a close up of an antenna. Src: [1]

8. The system should be real-time, where real-time is defined as having a latency in the order of a few seconds from receiving signals to displaying results to the user, as opposed to needing to post-process the data after taking measurements.

9. The hardware and software used should be in line with what is used at MeerKAT. This implies the Reconfigurable Open Architecture Computing Hardware (ROACH) platform for hardware, Python for backend software and JavaScript for front end software.

10. This system must operate in the context of the MeerKAT site, implying the following:
    a) In general, the RF environment is sparse. While there will be multiple simultaneous transmitters, it can be assumed that there will not be multiple narrow band transmitters with overlapping spectra, and that there will be at most one source of transients at a given time.
    b) The sources of the emissions will be relatively slow moving, up to the maximum speed of a vehicle on a dirt road; 60 km/h.

11. Once the software has been completed, its performance on real-life data should be quantified in the following way:
    a) A prototype 4-element antenna array should be connected to a 400 MHz bandwidth baseband digitiser and correlator. This wide bandwidth may compromise the dynamic range of the system but this is acceptable as in general we will be interested in the strongest signal.
    b) The correlator need not be real time for the demonstration.
    c) As the goal of this project is not to develop a hardware system, there is no specific requirement on receiver sensitivity or noise figure. Whatever the best available hardware is should be used for antennas, front-end receiver and digitiser.
    d) The performance of the hardware used should then be analysed.

12. Mitigation of the effects of performance degradation due to multipath is outside of the scope of this work.

13. The report produced should contain a theoretical analysis of the performance of the system, as well as an analysis of the performance of the prototype on site with real signals.

1.3 Requirements Review

The requirements as stated by the SKA are clear and understood. Some notable implications from the requirements:

- The system which will be designed here will not be immediately applicable to combating RFI at the MeerKAT site as this project is focused on the DF implementation, not hardware. Hence no down converter will be designed and consequently the frequency band being digitised by this system will not overlap with MeerKAT. MeerKAT has three bands of operation: 0.58 GHz - 1.015 GHz, 0.9 GHz - 1.67 GHz and 8 GHz - 14.5 GHz [2]. Future work is required to add a suitable RF front end to allow the system to operate in the MeerKAT band. The implication for this work is that it is necessary that this system be designed such that it is simple to reconfigure it for a different RF front end and frequency band.

- Based on the fact that the emitters will be either stationary or slow moving, it should be acceptable to have integration times in the order of 1 second.
- The requirement of a 100% POI will impose restrictions on what antenna array and receiver can be used, and this will in turn have restrictions on suitable DF algorithms.
- Seeing as this system is intended to be used by SKA and will require future work to make it production ready, the source code should be structured to allow collaborative development and hence will be store in Github in the following repositories.

Description	URL
This document	`https://github.com/jgowans/masters_writeup`
Ambiguity simulation	`https://github.com/jgowans/phase_ambiguity`
iADC calibration	`https://github.com/jgowans/iADC_calibration`
Direction finder front end	`https://github.com/jgowans/directionFinder_web`
ROACH firmware	`https://github.com/jgowans/correlation_plotter`
Direction finder backend	`https://github.com/jgowans/directionFinder_backend`

Table 1.1: Github repos for this project

1.4 Report Outline

Chapter 1 has so far contained the problem statement and user requirements of the system which needs to be designed and implemented.

Chapter 2 explores the existing literature around DF systems. This serves as the foundation for the system being designed here. The exploration focuses on direction-finding fundamentals and on a comparison of algorithms. It examines DF for both continuous narrowband signals and impulsive signals. It explores detection algorithms for impulsive sources, as well as calibration techniques and error calculations.

Chapter 3 contains the high level system design, showing what blocks need to be designed and how they will be linked together in order to build a complete system. Here, the system is broken into three main components: the RF front end, the ROACH digitiser and DSP engine, and the computer software for final AoA computation and logging. Additionally, the exact DF algorithm being used by the system will be defined and simulations of its performance run.

Chapter 4 discusses the antenna array and RF front end. It details designing and building a 4-element antenna array, including assembling and measuring low noise amplifiers and filters.

Chapter 5 discusses the implementation of the Field Programmable Gate Array (FPGA) firmware to do both data acquisition and the first stage of DSP. The focus is on designing the FPGA sub-system in simulink and then demonstrating it being functional both in simulink simulations and, most importantly, running on hardware in the lab to show successful processing of real signals.

Chapter 6 details Python code which implements the direction-finding algorithms. It requires interfacing the computer with the ROACH to read out data, creating a model of the antenna array, and estimating the AoA based on the received data and the array model. It also has to handle calibration, logging, and showing signal plots to give the operator insight into the signals being received.

Chapter 7 reports on the field trials of the DF system. There are two field trial sessions: one early in the project with a basic system to get a feel for the types of RFI data. Another with the final system doing real DF measurements and checking the accuracy of the system.

Chapter 8 contains conclusions indicating what has been achieved in the project and comments on the system which has been implemented. Also, scope for future work is looked at.

Chapter 2

Literature Review

In this chapter an exploration of the current literature around direction-finding is undertaken. It will first cover an overview of the fundamentals of antenna arrays and of signals in order to establish the concepts, terminology and mathematics required to understand specific DF techniques. Next, a variety of DF techniques will be analysed. The algorithm and principal of each technique will be presented, followed by the advantages and disadvantages of the technique. These advantages/disadvantages will be tied back to the user requirements defined in Chapter 1 to ascertain whether the technique is appropriate for this application. Where possible, existing systems which implement a particular technique will be cited as examples.

The purpose of this chapter is to gain sufficient information about the various DF techniques to make informed system design decisions in Chapter 3, as well as to implement accurate simulations.

2.1 Antenna Arrays

Let there be a far away transmitting source antenna (A), and a receiving antenna at the origin (B), and another close by receiving (C) at a position $\vec{x} = \begin{bmatrix} x & y & z \end{bmatrix}^\intercal$ offset from the receiver at the origin. The assumption is that the distance between the transmitter (A) and the receivers (B or C) is much greater than the distance between the two receivers (B and C); the array is small compared to the distance to the transmitter. If the signal received by the antenna at the origin (B) is $r(t)$ then the signal received by the offset antenna (C) can be be represented as $r(t, \vec{x})$, showing that the change in received signal by the offset receiver is a function of the position of the offset receiver. This model is shown graphically in Figure 2.1. As there is spacial difference between the two receivers, there is a time and phase delay between the signal seen by one compared to the other. This model contains both spacial and temporal information hece it is sufficient to be able to attain desired spacial information about the signal [3].

If we allow the source antenna to not be fixed but rather be placed at any direction in the far field, then the time delay relative to the origin for a signal arriving at the offset antenna is [4]

$$\tau(\phi, \theta) = \frac{1}{c} \left(x \cos(\theta) \cos(\phi) + y \sin(\theta) \cos(\phi) + z \sin(\phi) \right) \tag{2.1.1}$$

where θ is the azimuth angle of the source and ϕ is the elevation angle. Seeing as the transmitter is much further away from the receivers than the receivers are from each other, we assume that each receiver sees approximately the same angle to the transmitter. For a 2D space (with which this project is concerned) we let $\phi = 0$, $z = 0$ and hence simplify the time delay model to:

$$\tau(\theta) = \frac{1}{c} \left(x \cos(\theta) + y \sin(\theta) \right) \tag{2.1.2}$$

Naturally, the equivalent phase shift from the origin to the offset antenna (C) is

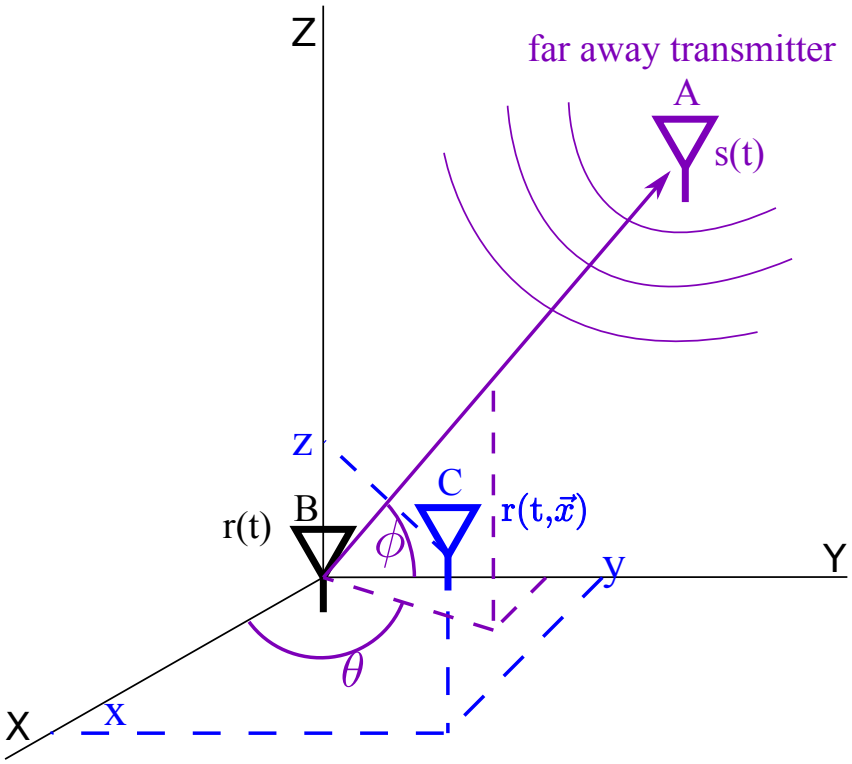


Figure 2.1: Diagram of propagation model used here. The far-field transmitter, A, producing signal s(t) is situated in the direction (θ, ϕ) from the two receivers. The receiver C, receiving r(t, $\vec{x}$) is at a position (x, y, z) offset from the reference receiver at the origin, B. Receivers A and B are close together compared to the distance to C, hence the angles θ and *phi* are approximately equal for both receivers.

$$\phi(\theta) = \omega\tau(\theta) = \frac{1}{\lambda}\left(x\cos(\theta) + y\sin(\theta)\right) \tag{2.1.3}$$

The observed time/phase difference refers to that of a single element. Array processing, however, requires modelling the output of the entire array. A highly useful way of expressing the output of an array is via the antenna array manifold vector[1]. The manifold vector consists of one complex term for each element in the array. The complex term refers to the relative amplitude and phase output of the element. As shown above, the time/phase response is a function of the position of the element. The amplitude is as a result of the gain of the antenna in the direction of arrival of the signal.

Assuming we now have an array of M elements, the array manifold vector for a signal source propagating from angle θ is: [6]

$$\vec{a}(\theta) = \vec{g}(\theta) \odot \exp\left\{-j\mathbf{X}^\intercal\vec{k}(\theta)\right\} \tag{2.1.4}$$

where

- $\vec{g}(\theta)$ is an M-dimensional vector of complex number expressing the gain and phase response of each element in the direction θ.
- $\odot$ is the Hadamard product which performs an entrywise multiplication of two matrices of equal sizes.
- $\mathbf{X}^\intercal$ is an $(M \times 2)$ matrix containing the x and y coordinates of each of the M elements of the form $\begin{bmatrix}\vec{x} & \vec{y}\end{bmatrix}^\intercal$. The coordinates used here will be measured in half-wavelengths to avoid frequency and speed of propagation factors in the equations.
- $\vec{k}(\theta)$ is the wavenumber vector given by $\vec{k}(\theta) = \pi\begin{bmatrix}\cos\theta & \sin\theta\end{bmatrix}^\intercal$. Graphically, this equates to a vector pointing in the direction of the source. The model assumes that the direction is the same from each element to the source. This is approximately true when the distance from the array to the source is much greater than the array size.

Typically all of the above vectors and matricies have terms for x, y and z location components as well as terms for the elevation angle, but for the purpose of this research the elements will all be located at the same elevation as we only intend to locate terrestrial signals. Hence, the equations have all been simplified here.

Furthermore, the $\vec{g}(\theta)$ term may be excluded if we assume omnidirectional elements. Although it is rare to deal with true 3D omnidirectional antennas, if the antenna is required to receive signals only in the azimuth plane, 2D omnidirectional antennas such as dipoles might very well be used in practice. This hence simplifies to:

$$\vec{a}(\theta) = \exp\left\{-j\mathbf{X}^\intercal\vec{k}(\theta)\right\} \tag{2.1.5}$$

or, more expressively:

$$\vec{a}(\theta) = \exp\left\{-j\begin{bmatrix}x_1 & y_1\\ x_2 & y_2\\ .., .. & \\ x_M & y_M\end{bmatrix}\begin{bmatrix}\pi\cos(\theta)\\ \pi\sin(\theta)\end{bmatrix}\right\} = \exp\left\{-j\pi\begin{bmatrix}x_1\cos(\theta) + y_1\sin(\theta)\\ x_2\cos(\theta) + y_2\sin(\theta))\\ ..\\ x_M\cos(\theta) + y_M\sin(\theta)\end{bmatrix}\right\} \tag{2.1.6}$$

[1] Also known as the steering vector or source-position vector or array response vector or DOA vector or direction vector [5]

Clearly, this is simply a vector of phase shifts introduced by each element in the array as a function of both the location of the element and the angle of the incident wave relative to some defined zero location and zero direction. Varying source direction corresponds to producing a curve in $\mathbf{C}^M$ space. A point on that curve corresponds to how the array will respond to a signal from that direction. This array manifold completely characterises the array [6]. For linear arrays there are additional details that allow simplification of the manifold vector as well as special properties possessed by the manifold of linear arrays. This will not be discussed here as the array used for this DF system is not likely to be linear.

An important takeaway from the array manifold is that it shows that for a known array with omnidirectional antennas at arbitrary x and y locations, receiving a signal from a certain direction, θ, the phase shift or equivalent time delay at each element is a vector which can be easily calculated. This is the basis around which the direction-finding algorithm for this project will be designed.

2.2 Direction Finding Techniques

A direction finder is a passive device able to ascertain the angle of arrival of a source of electromagnetic radiation. The objective of direction-finding is generally to find the location of non-cooperative emitters [4]. This is one of the fundamental activities of electronic defence as well as RFI hunting. The general structure of a direction-finding system is an antenna array connected to a receiver connected to a processing module connected to a display which then provides output to operators. Simple direction-finding systems were used as early as the start of the 20th century and have been continually evolving since then along with developments in communications and electronic warfare systems. Direction finding systems have applications in a variety of disciplines ranging from radar, sonar, surveillance, radio astronomy and even medical imaging. An optimal parameter estimator for angle of arrival is difficult to design and hence lots of research has been done into the topic over the last few decades [7].

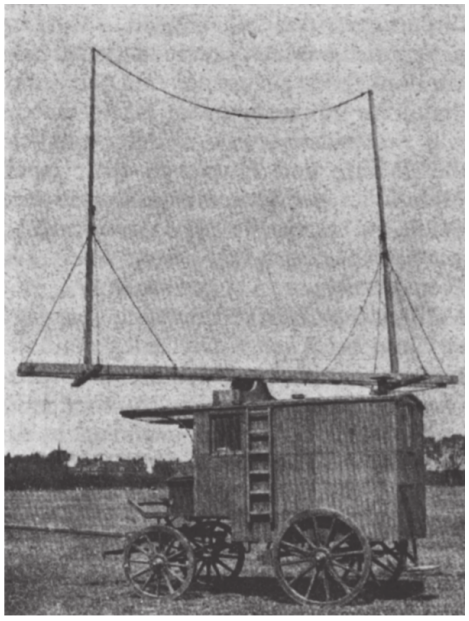
DF techniques are generally based on either amplitude comparison, time/phase comparison or statistical methods [8]. Various implementations of these classes of DF will now be explored.

Much of the following overview of types of direction-finding systems is based on discussions in the Electronic Warfare and Radar Systems Engineering Handbook [9].

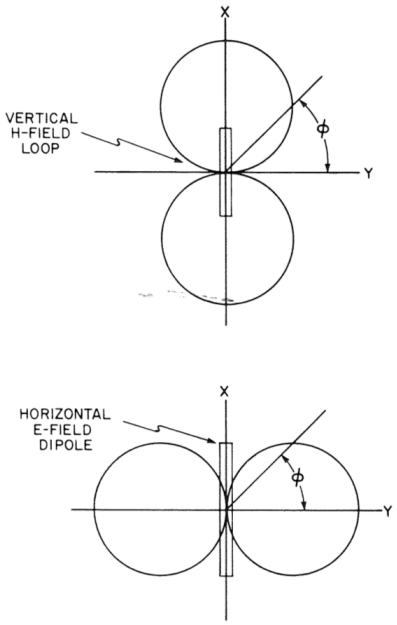
2.2.1 Scanned beam

This is a direct amplitude technique. An antenna with gain (non-uniform beam) is rotated and this causes the power output of the antenna to vary with rotation angle. This is the earliest form of radio direction-finding and was implemented in the early 1900's. One of the first uses of this DF technique was in a system developed by Marconi to locate the bearing to a ship which was transmitting a signal for navigation purposes [11]. A loop antenna was frequently used as it was a simple antenna to construct and the beam pattern has a sharp null. An example antenna and beam pattern are shown in Figure 2.2. As the change in antenna output power per degree rotated is highest around the null, the antenna was often aligned such that the signal of interest was in the null. If multiple signals originating from different bearings are present, the system cannot operate. While today it may be possible to put highly selective receivers on the output of the antenna, this technology did not exist when scanning beam DF systems were originally used.

Note that as the beam pattern is symmetric about both the x and y axis, there are in general four ambiguous points. However, if the antenna can be rotated such that the signal is in the



(a) Loop antenna used for direction-finding in 1918. Src: [10]



(b) Beam pattern of loop antenna or dipole. Note the sharp null. Src: [11]

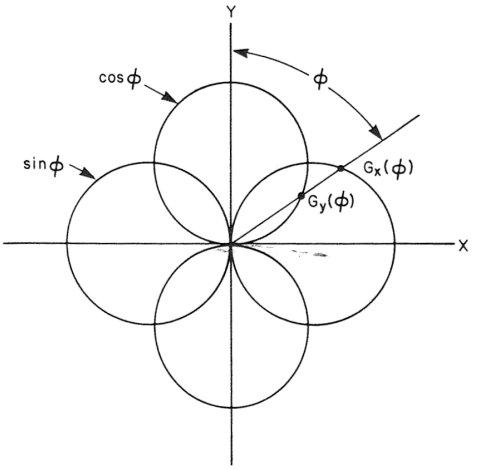
Figure 2.2: Loop antenna and beam pattern

null, there are then only two ambiguous points, or 180° ambiguity. This ambiguity can be resolved with an additional antenna element. Note also that this is not an instantaneous DF technique as it requires time for the antenna to be rotated. Hence, this technique is not suitable for finding transients.

2.2.2 Crossed Loop

The crossed loop technique is an amplitude comparison technique. As is shown in Figure 2.3, using two loop antennas perpendicular to one another produces two beam patterns: one being proportional to the sine of the angle and the other to the cosine of the angle. By comparing the signal power from each antenna, it is possible to ascertain the AoA. This system is instantaneous as only a single pulse is necessary to ascertain the ratio of antenna output power. However it has many ambiguities. Some ambiguities can be resolved with a sense antenna. For optimal ambiguity resolution, the crossed loop should be rotated such that signal of interest is located in the null of one of the loops and in the peak of the other. This resolves ambiguity but at the expense of the system not being real-time. As with the scanned beam, the crossed loop suffers from significant performance degradation arising from multipath, specifically ionospheric reflection. In the 1930's a marine radio direction-finder network using the crossed loop amplitude comparison technique was set up for marine navigation along the coast of Britain. Then, during World War II, rapid improvements to DF technology were made with the operating frequency of the systems being extended into the VHF and UHF band and with extensive DF networks being installed [11]. The tactical advantage which DF systems provided prompted significant research into DF during the war.

(a) Example of crossed loop antenna.



(b) Crossed loop antenna beam pattern showing difference in magnitude seen by each loop. Src: [11]

Figure 2.3: Crossed loop antenna and beam pattern

2.2.3 Adcock Array

The Adcock array was developed and patented in 1919 by British army engineer Frank Adcock [13]. Instead of using loop antennas, the Adcock array makes use of two orthogonally-orientated (crossed) pairs of monopole or dipole antennas. Typically a North-South pair and an East-West pair are used. These antenna pairs produce the same beam pattern as do loop antennas. As a loop antenna can be modeled by two vertical antennas on the sides and two horizontal antennas on the top and bottom of the loop, it should be clear that the loop is not polarisation selective: while the direct beam from the signal source may be vertically polarised, a reflected beam from atmospheric reflection may also be received and corrupt the signal output of the array. The Adcock array which uses only vertical elements maintains the same beam pattern as the loop but is not sensitive to horizontally polarised radiation hence offers better performance for a DF system. The antenna configuration of Adcock and Watson-Watt is shown in Figure 2.4. This array configuration was studied extensively in the 1930's and played a significant role in the electronic warfare of World War II [13]. One of the advantages of it at the time was that the output it provided allowed it to directly drive a CRT display, giving the user real-time visibility into what the system was receiving [11].

2.2.4 Watson-Watt Evaluation

This amplitude comparison technique is probably the most commonly used DF technique over the history of DF systems [4].

Ambiguity and multipath interference are two of the major difficulties which need to be overcome in direction-finding systems. In the mid 1920s, Sir Robert Watson-Watt developed an improved DF system based on the Adcock array configuration. As discussed above, the beam pattern of the output of an Adcock array is one cosine-shaped beam and one sine-shaped beam. By exploiting the trigonometric properties of these functions and adding an additional sense element, Watson-Watt developed a method to compute the angle of arrival from an array with this cos/sin property. This evaluation technique showed significant improvement in rejection of

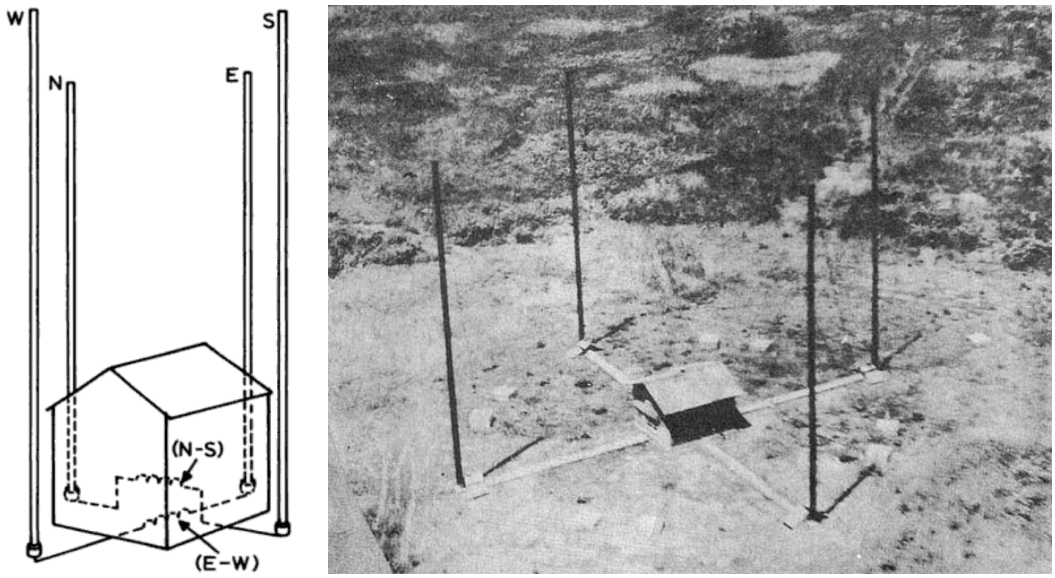
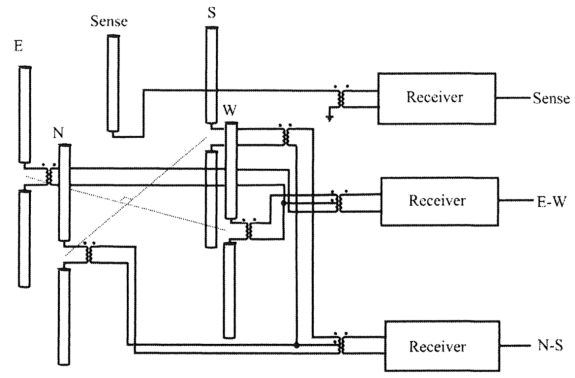
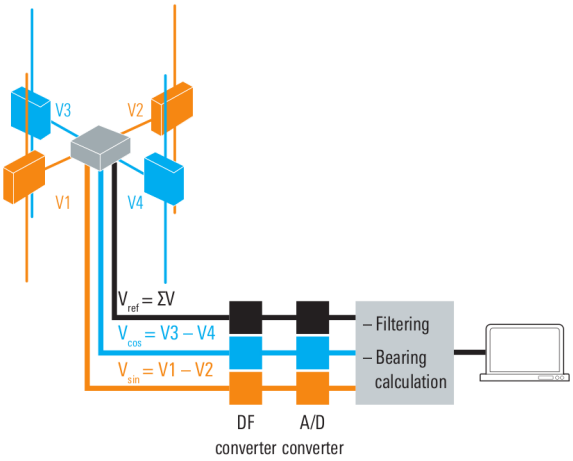


Figure 2.4: Left: Model for Adcock antenna showing N-S and E-W pairs. Right: Japanese implementation of array for 2 MHz direction-finding. Src: [12]

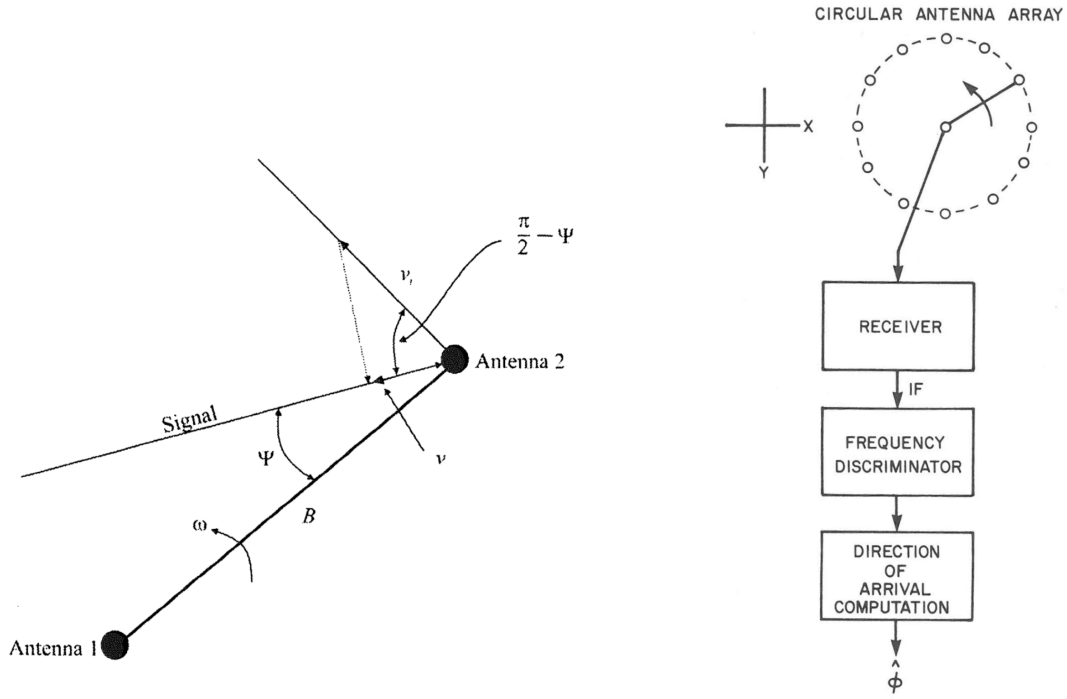


(a) View showing transformer configuration to produce required difference signals. Src: [14]



(b) Abstraced view of signal chain for digital processing. Src: [15]

Figure 2.5: Watson-Watt array processing technique using the Adcock array. Note that difference channels from crossed beams are formed.



(a) Model for Antenna 2 rotating around stationary Antenna 1 with angular velocity ω. Src: [4]

(b) Practical implementation switching antennas for pseudo-Doppler DF system. Src: [11]

Figure 2.6: Model for Doppler DF system

ionospheric reflections and made ambiguity resolution easier. In the case of Wattson-Watt, the beams are synthesised by forming sum and difference channels from broad beam antennas in an array.

The mathematics around this technique will not be explored in detail here, but a graphical view of the antenna and receiver structure can be seen in Figure 2.5. For a more detailed discussion of the mathematics behind the Watson-Watt DF algorithm, see Poisel [14]. For a simulation of Watson-Watt algorithm as well as a discussion around an ambiguity resolution implementation using a sense antenna see Pellejero [16].

2.2.5 Doppler

By the principle of Doppler shift, moving an antenna towards a signal source increases the observed frequency while moving an antenna away from a source lowers the observed frequency. If an antenna is rotated around a central point (tracing out the circumference of a circle, the resultant Doppler shift is given by: [4]

$$\Delta f = \frac{B\omega}{c} f_c \sin(\Psi) \tag{2.2.1}$$

where B is the length of the antenna baseline, ω is the angular velocity of the antenna, f_c is the carrier frequency of the signal source and Ψ is instantaneous difference between the angle towards the signal source and the angle of the rotating antenna (this is shown graphically in Figure 2.6). By rotating one antenna around a reference antenna and measuring the received signal frequency difference, the Doppler shift can be measured and with Equation (2.2.1) the AoA, Ψ, calculated.

However, as an example, to DF a 100 MHz tone with a Doppler shift of 3 MHz, the antenna would need to be rotated with a tangential velocity of 10000 m/s [11]. For the technique to be successful, it is necessary to produce a relatively high percentage frequency shift change so that the change is noticeable. However, it is not practical to physically rotate an antenna at such a high speed. Hence, rather than physically rotating an antenna, what is done in practice is that a Doppler shift is synthesised by rapidly switching between sampling different elements of a large circular array. Typically between 12 and 30 elements are used for Doppler DF arrays. Figure 2.6 shows a typical implementation of a Doppler DF system using antenna switching.

As discussed by Jenkins [11], Doppler is not a high accuracy system due to the signal distortion introduced by switching the sampled antenna. This is why it is necessary to introduce a relatively large percentage Doppler to achieve usable accuracy for a Doppler-based DF system. More modern switching hardware would likely have cleaner switching and require a lower rotational rate, which would increase the resolution and accuracy of the technique. However, Doppler DF is not suited to locating transients for two reasons:

1. it is not an instantaneous technique due to there being some time required to switch between sampling all of the antennas. A typical time required to switch between all elements of an array may be 6 ms [15]. This is unsuitable for transients which may last only a few hundred nanoseconds.

2. it generally needs a narrowband signal which has a well defined carrier frequency so that the Doppler shift is well defined. Impulsive transients do not have this characteristic.

For a more detailed discussion of the mathematics behind Doppler DF, see the Rhode & Schwarz report [15].

2.2.6 Time Difference of Arrival

Time Difference of Arrival (TDoA) is a DF technique which is typically used for finding impulsive sources; sources which emit a pulse that exists for a short time duration where the pulse has a clearly defined start and end. Most commonly it is used for locating pulsed radar systems in the electronic warfare context [11].

For a two element array, the difference in time, δt in the arrival of a pulse at the elements is

$$\delta t = \frac{d \cos \theta}{c} \tag{2.2.2}$$

where d is the element spacing, θ is the AoA relative to the baseline, and c is the speed of light.

This technique requires the ability to measure the start of a pulse very accurately, or alternatively to be able to cross-correlate the whole pulse in the time domain. One approach to ascertain the time delay between two receives is to pass one of the signals through a system with a variable time delay and compute the cross correlation between the delayed signal and the reference signal. By sweeping the delay and looking for the peak of the correlation, the actual time delay may be found [17]. This works the in presence of uncorrelated noise.

As discussed by Gustafsson and Gunnarsson [18] one of the ways to compute the AoA from received signals is to construct hyperbolic functions for each receiver pair, and then find a position is a least-squares minima on the hyperbolic curves using various techniques which can achieve the Cramer-Rao lower bound (CRLB) for error. Additionally, computation for the time difference could equally be done in the frequency domain via a least-squares method which can also produce CRLB results [19].

The primary advantages of it are that it is independent of frequency and does not suffer from ambiguity when receiving impulsive signals [11].

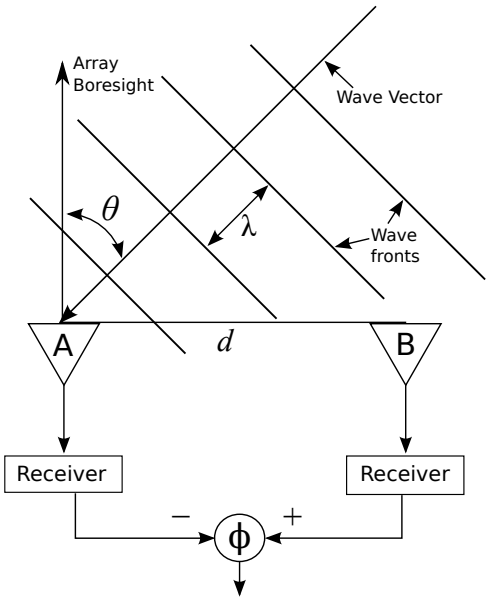


Figure 2.7: A simple two-element phase interferometer. The narrowband signal is arriving at the array at angle θ producing observed phase difference ϕ. Note that as the element spacing, d, is more than half a wavelength there is ambiguity.

2.2.7 Phase Interferometry

This technique is done by comparing the phase of the signal arriving at each element of an array. In general, it is a higher complexity technique and may suffer from ambiguity, but it can achieve comparatively high-accuracy direction measurements, often between 0.1° and 3° for real systems [9]. In comparison to amplitude systems, phase systems are more tolerant of multipath interference as additional compensation techniques can be applied [5]. The high complexity is introduced due to needing to perform real-time high-accuracy phase measurements of signals at multiple antennas and having to incorporate ambiguity-resolution algorithms. Also, phase-based systems often impose stringent requirements on the phase matching of the RF chain which requires careful design, measurement and calibration [20].

The simplest phase interferometry system is a two element array. This is shown graphically in Figure 2.7 where d is the element spacing, λ is the wavelength and θ is the AoA or difference between the boresight and wave vector angle. The phase difference, ϕ, at the output of the system is calculated as follows:

$$\phi = \frac{2\pi d \sin\theta}{\lambda} \tag{2.2.3}$$

Notes about this simple two element array:

1. The values of the sine function are only unique between $-90°$ and $90°$. Hence the output of the array is at best only unambiguous within a 180° field of view, requiring the antennas to be placed less than half a wavelength apart. A two-element phased-interferometry array cannot resolve this ambiguity. The best it can do is use antennas with directionality to reject signals from outside of its field of view (FOV).

2. If the element spacing is larger than $\frac{\lambda}{2}$ (as shown in the figure) the ambiguity gets worse. The unambiguous field of view for a 2-element array is: $\theta_{FOV} = 2\sin^{-1}(\frac{\lambda}{2d})$ [4]. Clearly, as d gets larger, θ_{FOV} gets smaller.

3. Although ambiguity (caused by a smaller θ_{FOV}) gets worse with a larger spacing, angular accuracy improves. This is because a certain FOV percentage error will equate to a lower angular error when the FOV decreases. Hence, there is a trade-off between ambiguity and accuracy.

The solution to this ambiguity problem is to use more than two elements, thereby constructing a multiple-baseline interferometer. Having more elements provides ambiguity resolving information, and also results in a lower error generally due to having more measurements.

There are generally three subclasses of phase interferometry systems [11]:

Continuous phase measurement: A single receiver takes in the signals from all elements and compares the instantaneous phase of the signals, typically using analogue multipliers. It is fast and relatively simple for two antennas. It becomes complex as the number of elements increases, typically has a limited field of view, is susceptible to interference, and so is unable to provide high accuracy.

Phase scanning correlation: Here, a phase shifter and correlator are used. The phase shifter is varied until the correlator produces a peak output. Again, this is simple for two antennas but becomes complex when multiple phase shifters need to be varied. It is also slow as the phase shift needs to be swept to find the peak correlator output.

Fourier transform method: The received signals are digitised and converted to frequency domain, where phase difference measurements are easily read from the spectrum. As this is a digital technique, spectral processing can be done; it provides 20 dB to 30 dB of additional sensitivity over the other techniques [9]. Also, working with digital signals in the frequency domain means that phase mismatches in the signal path can easily be compensated for with digital filtering, and adjacent frequency signals can be ignored.

Should the antenna array contain more than two elements, the system either needs to switch between pairs of antennas if it wants to maintain a simple two-input receiver, or it needs a more advanced receiver to sample all antennas simultaneously. Since computation is done in the digital domain, the Fourier transform method is easiest for accommodating more antenna inputs. Having all antennas processed by the receiver simultaneously allows for a faster system as no switching is required and also provides better performance as it can average multiple inputs at once. However, more inputs necessitates more computation and algorithmic complexity.

2.2.8 Subspace and Statistical Techniques

This class of techniques attempts to perform eigen analysis, decomposing the spectral or covariance matrix into eigen subspaces for noise and subspaces for signal. In some sense this equates to a beamforming algorithm which attempts to find an angle resulting in the strongest signal. Statistical methods generally involve taking snapshots of the signals received at each antenna and then calculating the covariance matrix of the received data [4].

The advantage of working with multiple subspaces for each signal is that it allows direction-finding to take place when there are multiple overlapping signals arriving at the array. Assume there are D narrowband signals all with unknown AoA parameter arriving at an M element array in the presence of uncorrelated noise, the problem space can be reduced from M-dimensional to D-dimensional [7].

It can be shown that the covariance matrix is diagonal for incoherent sources, non-diagonal and non-singular[1] for partially coherent sources, and non-diagonal and singular if at least one of the sources is coherent [4].

Examples of these techniques include [7]:

- Spectral MuSiC (Multiple Signal Classification) which gets multiple signal snapshots to produce an estimated spectral matrix, $\hat{\mathbf{S}}_{\mathbf{x}}$, and requires knowing the antenna array manifold vector. It can be applied to arbitrary arrays and performs best when the received signals are uncorrelated. The technique does not work when receiving multiple coherent signals.

- Root MuSiC, which is a special form of MuSiC allowing simplified polynomial equations provided linear arrays are used, and requires a lower antenna SNR than Spectral MuSiC.

- Least Squares and Total Least Squares ESPRIT (Estimation of Signal Parameter via Rotational Invariance Techniques) which constructs identical subarrays from uniform linear arrays, exploiting shift invariance properties of the array. It can be shown that the signal subspace eigenvectors are linear combinations of the array manifold vector. The RMS error for these techniques can be very close to the Cramer-Rao bound for estimation error [7].

These subspace techniques typically require knowledge of the number of unique signals, D, being received as they depend on being able to construct D subspaces. Seeing as coherent signals degrade performance, the techniques suffer significantly from multipath interference. These substrace techniques more more computationally complex than other techniques discussed here.

2.3 Geolocation

Geolocation refers to the process of finding the absolute position of a source, often in terms of a coordinate system like latitude/longitude/elevation. This information is often more useful than only knowing the direction which an emitter lies in. However, it can be shown that by having multiple DF stations, the process of triangulation may be used to geolocate an emitter from the direction bearings [4].

The relevance to this work of the above note on geolocation is that it is not necessary to attempt to design a system which can do geolocation natively. Rather, a DF system can be designed which can later be deployed to multiple coordinated sites in order to provide geolocation capabilities.

2.4 Summary

This chapter began with presenting the RF model that is used in this project of a far-field transmitter at some direction, and receivers offset from a reference origin. The antenna array manifold vector was then developed, showing how it is a useful tool in quantifying the expected time/phase shift of an arbitrary array by a signal arriving from a given direction. The chapter then proceeded to examine various existing direction finding techniques, both amplitude and phase techniques. The principal of operation was explained, and note was made of specific

[1] A singular matrix is one which is not invertible, meaning the determinant is 0

characteristics about the techniques that would make them suitable or unsuitable for this system given the user requirements. Finally, a brief note on the process of geolocation and its relation to direction finding was made.

Chapter 3

System Design

In this section the block level system design is presented and the direction-finding algorithms are identified. The design is based on the user requirements specified in Chapter 1 as well as the data and background gathered in Chapter 2. Simulations are run to show whether the algorithm performs as expected and also to highlight potential problems that need to be tackled.

3.1 Sub-systems

The DF system consists of the subsystems which are described here, as well as shown in block diagram view in Figure 3.1.

3.1.1 Antenna Array

The antenna array will receive the RFI signals. The user requirements state that a 360° field of view is necessary. Hence, a circular array is used. The arrays needs to be useful in direction-finding both narrowband and impulsive signals. This implies that a trade-off is required in terms of element spacing when designing the array. If the element spacing is too large, the array will experience high amounts of phase ambiguity and perform poorly at narrowband DF. If the element spacing is too small the time difference for impulsive signals will be difficult to measure. The elements should be non-dispersive[1] to facilitate keeping the structure of the impulsive signals. The number of elements to be used in the array will be explored later in Chapter 4.

3.1.2 Front End

The front end contains anti-aliasing filters and low noise amplifiers. For this project, a simple front end is used. There is no mixing or switching: the base band signal will be low pass filtered and amplified. The specifications of the filters and amplifiers are provided Chapter 4 based on the operating frequency band of the array and the sampling frequency of the analogue to digital converter (ADC)s. The design requires independent, identical RF chain for each element of the array.

3.1.3 ROACH

The ROACH board has the functions of doing analogue to digital conversion and the high-speed digital signal processing. The ROACH consists of a Virtex 5 FPGA and multiple ADC cards allowing all of the RF inputs will be digitsed independently. The purpose of the ROACH is to do real-time processing on the raw ADC samples, hence reducing the data rate to only signals of interest which can be sent to the computer and processed there.

[1] Phase response of the antenna should be linear across frequency.

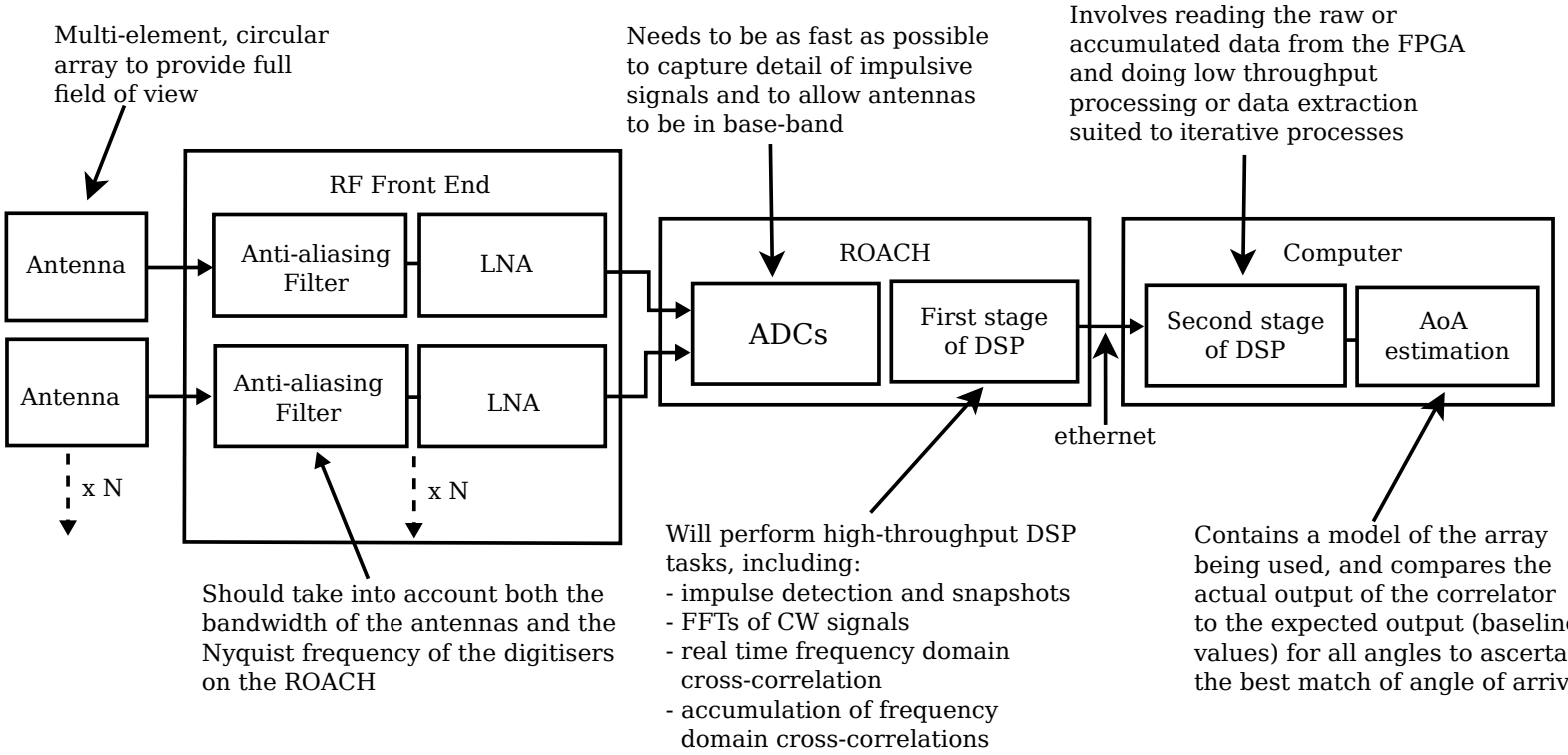


Figure 3.1: Block diagram view of the flow of signals through the proposed direction-finding system. The key function of the blocks has been noted on the diagram.

3.1.4 Computer Software

When the computer receives input from the ROACH its role is to do further DSP, specifically the final angle of arrival estimation. The computer needs to be able to log all of the data for future analysis and display the results of the direction-finding to the user real-time in the field.

3.2 Direction Finding Algorithms

With the above blocks in place, this section describes the selection of appropriate direction-finding algorithms which meet user requirements. If further refines what will be done in the first and second stage of DSP. Some additional requirements for the other blocks that arise from the selected algorithm are noted.

This section details how the selected direction-finding algorithms will be implemented on the system. A simulation of the algorithms and the system in operation written in Python follows in Section 3.3. The output of this simulation demonstrates that the algorithms are capable of carrying out the task.

There are two cases that the algorithms needs to deal with: weak narrowband signals and strong impulsive ones. The narrowband case naturally lends itself to frequency domain analysis since the information about narrowband continuous signals is concisely expressed in the frequency domain. The impulsive signals, however, are more naturally dealt with in the time domain as they are bound in time, existing only for a short duration, while being spread all over the frequency domain.

As such, the selected algorithms contain two distinct parts: a phase interferometry component designed to direction-find weak (below the noise floor) narrowband signals, and a time difference of arrival part designed to locate short-duration impulsive RFI that are sufficiently strong for power detection. Quantifying exactly how strong the impulsive signals need to be is hard. In general the higher the impulse that we're dealing with, the higher the probability of detection, and similarly the higher the threshold can be set to avoid false detection.[1]

3.2.1 Phase Interferometry

Phase Interferometry direction-finding of narrowband signals is done by phase comparison of the signal of interest as seen by each element in the array. The technique of phase interferometry has been extensively explored in the literature review. The process carried out by this DF system will be the following steps:

1. ADCs are used to *synchronously* sample the signal seen by each antenna. The ADCs need to be calibrated and have equal cable length to each antenna. Calibration procedure is discussed in Appendix B.

2. The output of each ADC is streamed into a real-time fast Fourier transform (FFT) capable of producing Fourier transforms of the observed signals as fast as they are captured. Real-time implies that the FFTs must stream their frequency channel outputs at the same rate as they get time domain data input.

3. Cross correlation between each pair of antennas is done to establish time or phase shift observed by the antenna baselines. In the discrete frequency domain this means multiplying

[1] In practice it was found that impulses with an energy that is three times that of the noise energy over the duration of the impulse were able to be detected reliably with very few false detection.

the value of each[1] frequency channel from one antenna by the complex conjugate of the value of the corresponding frequency channels from another antenna. For each pair of signals A and B where $A = (a+ib)$ and $B = (c+id)$ do $C = A\overline{B} = (ac+bd)+(bc-ad)i$. The output of this complex conjugate multiplication is a cross-correlation spectrum[2] where the phase in each frequency bin is the phase difference of the two input spectra.

4. Extract the signal of interest from the noise. For weak signals, the phase of the cross-correlation spectrum is corrupted by uncorrelated system noise. Attempting to read off phase difference from the output of a single FFT would produce poor results as the phase would be dominated with noise. However, assuming that the noise is additive zero mean, the noise component will sum incoherently. Taking advantage of this we add multiple instances of these cross-correlation spectra. The coherent signal component (observed phase difference difference of the signal seen by each antenna pair) sums coherently while the uncorrelated noise component sums incoherently. As such, the more spectra are summed the more the signal component stands out from the noise.

5. When sufficient baseline spectra have been summed, a snapshot of the result is taken and the phase of the channel of interest extracted from each baseline spectra. A M-dimensional vector of these observed baseline phase differences, where $M = \binom{N}{2}$ is the number of baselines for an N-element array.

6. For some value of K, an $M \times K$ matrix is constructed representing the theoretical array response at each of K sampled angles. Each column of the matrix is a vector of expected baseline phase shift for the AoA corresponding to that column. This is essentially a sampled antenna array manifold with K being the number of sample angles for which the manifold is calculated. K can be made as large as one wishes with a larger value meaning a higher resolution AoA estimation at the cost of more computational complexity.

7. For each column in the calculated manifold matrix, the sum of the squares of the difference between the observed phase shift vector and that column is calculated. The column with the lowest of these computed differences is selected as the AoA of the signal. If the vector $\vec{a}$ is the observed signal's baseline phase shifts and the vector $\vec{b}$ is a column of baseline phase shifts from the sampled manifold matrix with which to compare then the computed sum of squares difference d:

$$d = \sum_{i=0}^{k} \arg\{e^{ja_i} e^{-jb_i}\}^2 \tag{3.2.1}$$

where k is the number of terms in the vector; one term for each pair of baselines. A measure of confidence in the DF result can be given from the difference, d). The reason for using sum of squares here (as opposed to just sum of differences) is to make it less likely to select an angle where most of the phases match, but one is very different. This can happen as an incident angle may produce ambiguous phase measurements on some element pairs, but a large difference on another single pair. This ambiguous angle should

[1] It's not actually necessary to do each channel. If you know ahead of time the channel of interest the system need only accumulate that one, ignoring the rest. This reduces memory footprint, but means the entire spectrum cannot be read out and plotted or used for calibration, as well as making the system less dynamic/flexible.

[2] The spectra is referred to as the "cross correlation" spectra because complex conjugate multiplication in the frequency domain is the same as cross-correlation in the time domain

be rejected, and squaring the difference helps with this rejection. Further research can be done on the optimal difference calculation for various array geometries.

The difference equation given in Equation (3.2.1) can equally be expressed as a trigonometric calculation, and doing so can take advantage of the trig optimization of Python's numpy library as well as vector operation optimizations. Arctan is used to account for the fact that phase wraps around 2π:

$$d = \sum_{i=0}^{k} \text{atan2}(\sin(a_i - b_i), \cos(a_i - b_i))^2 \tag{3.2.2}$$

3.2.2 Time Domain Direction Finding

This technique is remarkably similar to phase interferometry. We are still capturing time domain signals from the antenna using ADCs that are sampling synchronously to be able to measure the received signal from the array. We are still comparing the value of the parameter to the theoretical response from the array at every angle in order to find the angle whose expected output most closely matches the observed signal. Hence much of the process described above will still apply.

The key difference is that instead of the parameter of interest being the phase difference of a frequency channel, the parameter of interest is the *time* difference of an impulsive signal at each antenna. As such it is necessary to capture time domain snapshots of the signals seen by each antenna and then do time domain cross-correlation to find the time domain offset of signals seen by each antenna. The peak of the cross-correlation output corresponds to the time difference.[1]

Time domain cross-correlation done by multiplying each point of one antenna's signal $\vec{a}$ by the corresponding points of a shifted version of another antenna's signal $\vec{b}$ for some shift k, producing the correlation $c_{ab}(k)$:

$$c_{ab}(k) = \sum_{n=0}^{N-1} a_n b_{n+k} \tag{3.2.3}$$

Additionally, as the time domain cross-correlation process is computationally expensive, it will only be done when a signal is detected. This implies that an impulse detector of some sort must be present and must be able to trigger the DF process.

Note that it would be possible to compute the time difference in the frequency domain as well, but that would require phase unwrapping across the spectrum hence we opt for the time domain approach. Phase unwrapping is necessary when the propagation time between the array elements is larger than one sample, as will be the case with this system.

Once the time difference for each pair of antennas is found, the vector of time differences is compared iteratively to sampled vectors from the array manifold, much like Equation (3.2.2). The difference how is that the computation is done in terms of time differences (measured vs computed from manifold vector) rather than phase differences.

[1] Strictly speaking the peak offset is actually the ADC clock sample difference, but seeing as the ADC is being clocked at a known rate it can easily be converted to time difference. The antenna array manifold constructed will thus be in the time domain rather than the frequency domain.

Upsampling for resolution

For impulsive signals for which time domain DF will be used, there is no issue of ambiguity as such signal are not periodic, unlike continuous narrowband signals. Instead, the issue is one of resolution. In order to get accurate time difference readings, a high time-resolution snapshot of the signals must be taken. The higher the time resolution of the signal, the more cross correlation points there are and the greater the accuracy of the time domain measurement. This means pushing the ADC faster: the faster the ADC sample rate and the higher the bandwidth, the better the resolution and accuracy. However, the sample rate of the ADC is generally a hard limit, and is a constrain that must be worked within, hence additional post-processing techniques are necessary to increase resolution to a useful value. Furthermore it may actually be undesirable to increase the sample rate, as that increases the errors caused by mis-matches between the various ADC cores or clock distribution.

Fortunately we can use upsampling to increase the resolution of the time domain cross-correlation provided that some requirements are met, and at the expense of additional computational complexity. The computational complexity comes both from the upsampling process as well as from the increased number of data points in the cross-correlation step.

The primary requirement is for a band-limited signal with no unwanted aliasing. The system designed here is doing base-band sampling which implies that this requirement is met provided a suitable low-pass filter is used in accordance with the Nyquist–Shannon sampling theorem. Sampling like this means that all of the information about the signal is captured in the samples. As the signal will thus be fully defined, smooth lines can be reconstructed between sampled points. A finder simple spacing can be obtained by interpolating more points (upsampling) between the sampled points. Note that this does not add any information, is simply increases the granularity at which a match can be found.

This technique of upsampling is used to compensate for the fact that we will be using an antenna array with relatively small dimensions[1], not typically suited to time domain direction-finding. A small antenna array is good for reducing phase ambiguity for frequency domain direction-finding but produces low time domain resolution; the sample spacing in the time domain needs to be fine enough to extract the correlation peak accurately, hence the need for upsampling.

It is worth noting here that part of the reason that a small array is less suitable for impulsive direction finding is that small imperfections in the real hardware have a much larger distortion when the time difference over an angle is low. These imperfections would include jitter in the ADCs, or mis-matches in the RF path. If the array is large then these imperfections are less impactful, as a change in angle of arrival results in a change of many more ADC samples.

3.3 Algorithm Simulations

3.3.1 Frequency Domain Simulations

In preparation for the implementation of the direction finding algorithms, a Python package called `DirectionFinder_backend`[2] is first written. This package has the code for an `AntennaArray` class which builds an array model from arbitrary `Antenna` objects. It can calculate and return a vector of the antenna array manifold for any given `AntennaArray` and required number of sample points. The structure of this key package is discussed in more detail in Chapter 6.

[1] Small in the sense that a signal will propagate over the array in the space of only a few ADC samples

[2] `https://github.com/jgowans/directionFinder_backend`

An additional package, `PhaseAmbiguity`[1] is written containing scripts for creating an `AntennaArray` object and then a matrix of sampled antenna array manifold vectors. It then selects a reference vector for some angle, θ_{ref}, and computing the root-mean-square (RMS) difference between the reference antenna array manifold vector and the vector from all other directions. The results of this difference computation are then plotted. What this is doing in practice is assuming that the signal is actually coming from the given reference angle, θ_{ref}, and checking how similar (aka: ambiguous) a signal from another comparison angle, θ_{comp}, looks for a range of comparison angles.

The `PhaseAmbiguity` simulator takes in as runtime arguments these parameters of frequency range, number of discrete frequency points, comparison θ_{comp} range, number of angle sample points, and the reference angle. This produces a 3-D plot with axes for frequency, comparison angle and RMS difference. Typically the 3-D plot is flattened to a 2-D image with colour used for the third dimension, rather than rendering a 3-D image. The type of data being plotted naturally lends itself to a colour plot view.

Ideally we would like 4-dimensional output: frequency, reference angle, comparison angle and RMS difference. However, this is difficult to visualise. Hence, two simulation approaches are used:

1. fix reference angle while varying frequency and comparison angle, or
2. fix frequency while vary reference and comparison angle.

These different approaches produce results with a different view on the data. One view shows a slice of the ambiguity performance over a range of frequencies when the RFI emitter is located in a set direction. The other shows the ambiguity performance at a set frequency for a range of source locations. The combination of the two views gives a more complete picture of the ambiguity of the system for different parameters.

3.3.2 Ambiguity Simulation Results

The plots in Figure 3.2 are of array ambiguity for circular antenna arrays ranging from 3 to 9 elements. For this application a circular array is preferable, as it provides equal angular resolution and accuracy over the full 360° field of view, which is a requirement for this application. As the number of elements increases the array radius is increased linearly to ensure equal spacing between elements is maintained. On the x-axis is the signal frequency ranging from 45 MHz to 400 MHz[2] corresponding to wavelengths of 6.6 m to 0.75 m. On the y-axis is the comparison angle from $-\pi$ radians to $+\pi$ radians.

The simulation is for a reference signal arriving from 0 radians and comparing how similar signals from other angles appear to the system. This is a theoretical performance for a signal in no noise. The ideal performance is that all signals from all angles other than 0 radians have a distinguishable phase difference. This is not always the case, for example consider the 4-element array at 300 MHz: there are strong points of ambiguity at 1.5 radians, -1.5 radians and $\pm\pi$ radians.

Two observations are made:

1. Arrays with an even number of elements suffer more from ambiguity than odd numbered arrays. This is due to there being lines of symmetry in even numbered circular arrays. For example, a 4-element uniformly sampled circular array is actually a square.

[1] `https://github.com/jgowans/phase_ambiguity`

[2] This frequency range was selected as it is the operating range of the ADCs that were used for this project

2. In general, the more elements you have, the lower the ambiguity. This is due to more baselines being able to resolve ambiguity.

The plots in Figure 3.3 show two examples of the other view of the data for 4 and 5 element arrays. Here both θ_{ref} and θ_{comp} are varied while f is fixed at 250 MHz corresponding to 1.2 m wavelength. The following observations are made:

1. The 4 element array has points of full indistinguishable ambiguity where the signal arriving from θ_{ref} and θ_{comp} are mathematically identical even though the angles are different. The 5 element array does not have true ambiguity due to no lines of symmetry in the array.

2. Although no full ambiguity, the 5 element arrays does have bands of angles where the observed signals look more similar than other bands. Hence some angles are "more ambiguous" than others, meaning that it requires less noise or measurement error to throw off the result. This implies that the performance of the array is not exactly uniform. The reason for this is that the array is not a true circle, it's a sampled circled with 5 sample points. More sample points will reduce the contrast of these bands and make performance more uniform.

3. The location of the bands of increased ambiguity is a function of the radius of the array measured in wavelengths. As such, the location of the bands will change with frequency.

Accumulation Gain Simulation Results

The other performance criterion of the frequency domain system is an ability to see weak signals below the noise floor. The contention is that by accumulating many data points, the noise component will sum to zero while the signal component will sum coherently.

To verify this, the simulation package is extended to add coherent signals in the presence of white noise at configurable signal to noise ratio (SNR) levels. The coherent signal has a defined frequency and phase, and snapshots of 2048 samples of it are added to 2048 samples of white noise. The result is Fourier transformed and the phase of the signal of interest read off the spectrum. The phase readings are accumulated as more and more snapshots are generated and measured. Phase error is measured after every snapshot / accumulation. The result is shown in Figure 3.4. From this figure it can be seen that accumulation decreases error: as more data points are accumulated, the more the real signal stands out of the noise. It also shows that the higher the SNR the faster the error drops off, hence stronger signals will become more accurate more quickly.

The structure of the simulation included many of the true limitations of the real system in order to make the result as meaningful as possible. This includes quantizing the input signal to 8-bits after being generated to simulate an ADC, as well as quantizing the output of the FFT to 32 bit fixed point signed integers to simulate the data bus widths that will be used in the FPGA.

This figure also demonstrates why it is difficult to quantify the performance of a DF system with a single "accuracy" number: the error is dependant on many factors, such as number of antenna elements, RFI signal strength and observation time.

3.3.3 Time Domain Simulations

The time domain system does not need either ambiguity or accumulations length simulations. The reason is that time domain DF of impulsive signals does not suffer from ambiguity, and

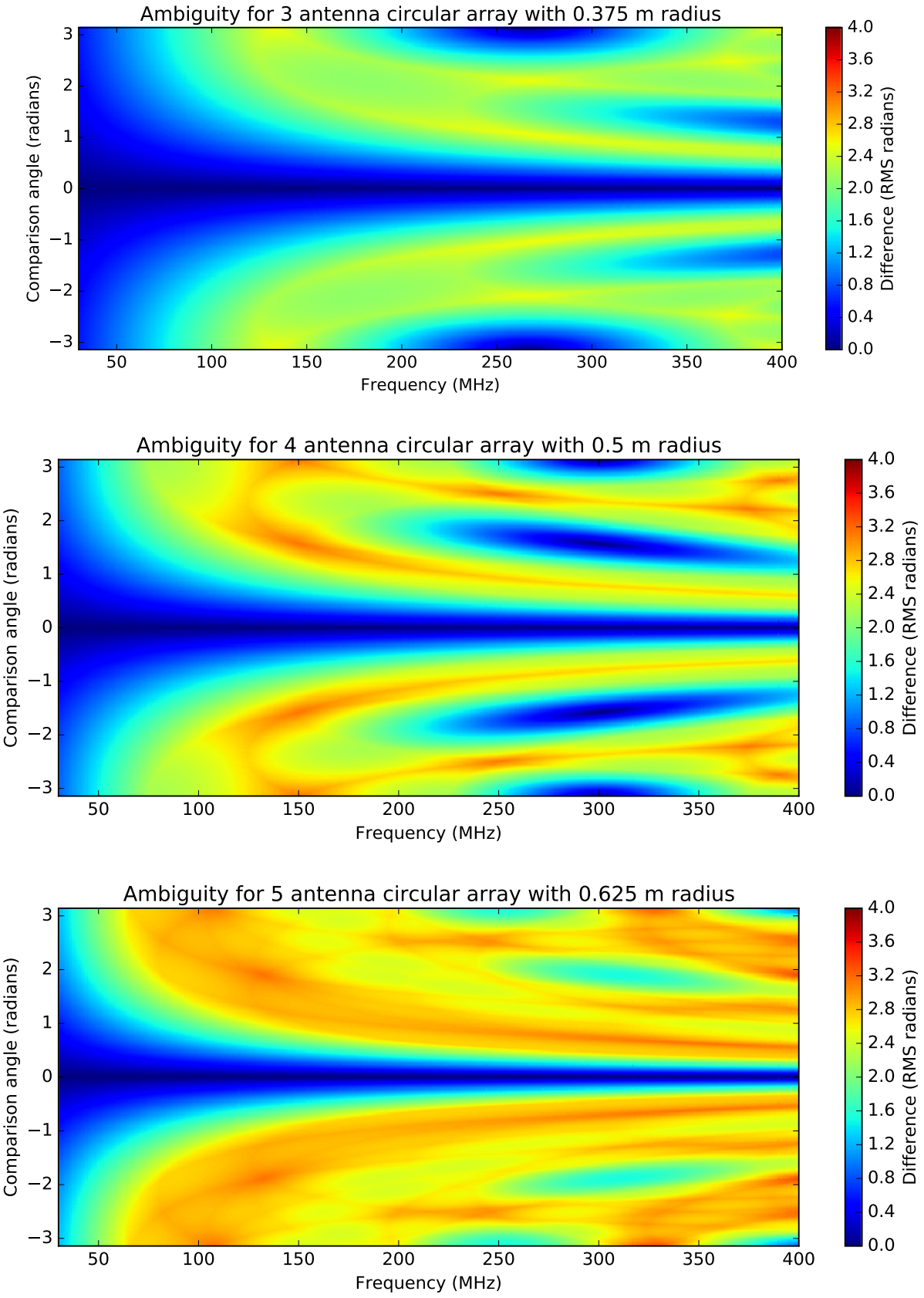


Figure 3.2: Ambiguity plots for various array sizes with the reference signal arriving at 0°.

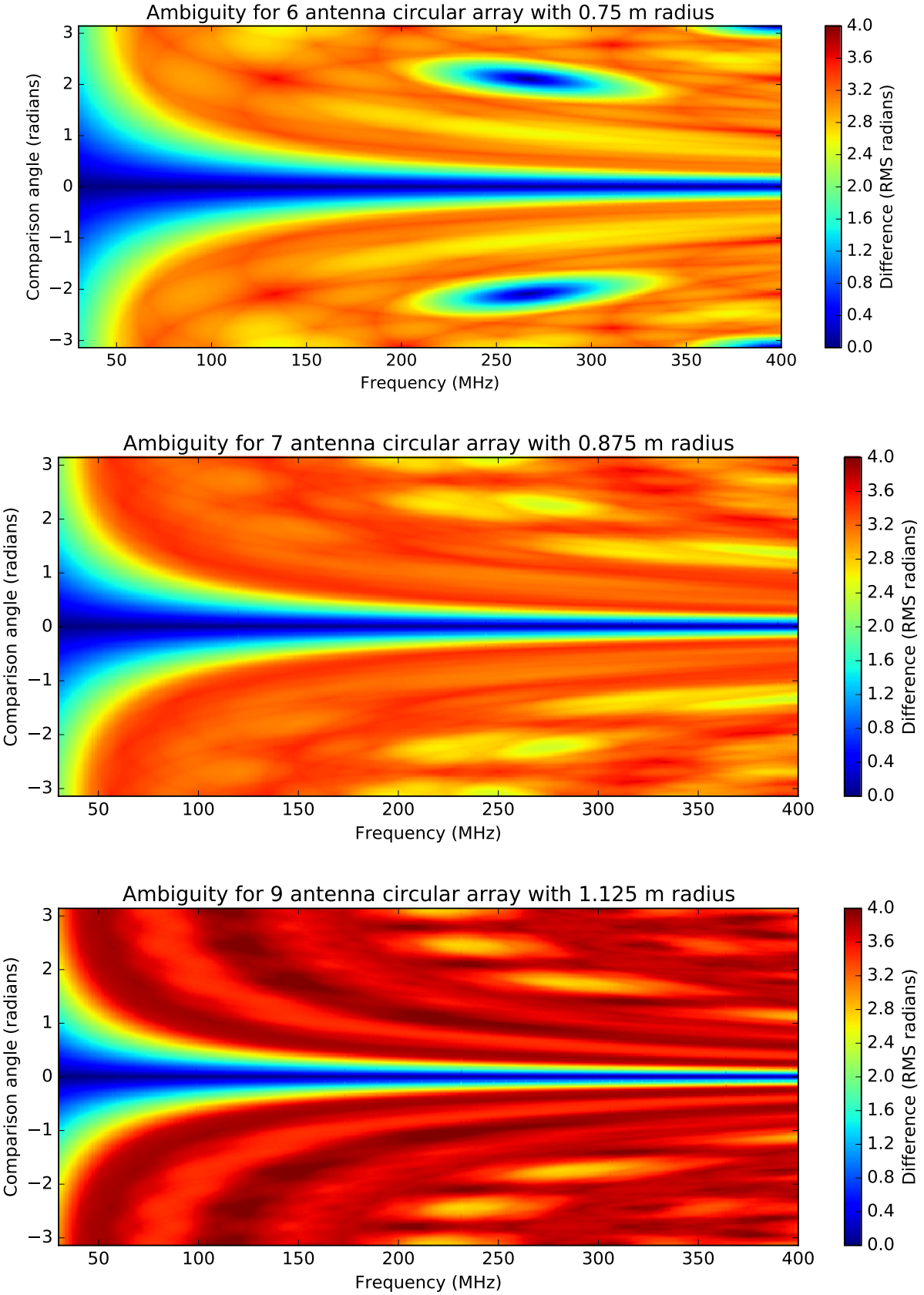


Figure 3.2: (cont'd) Ambiguity plots for various array sizes with the reference signal arriving at 0°.

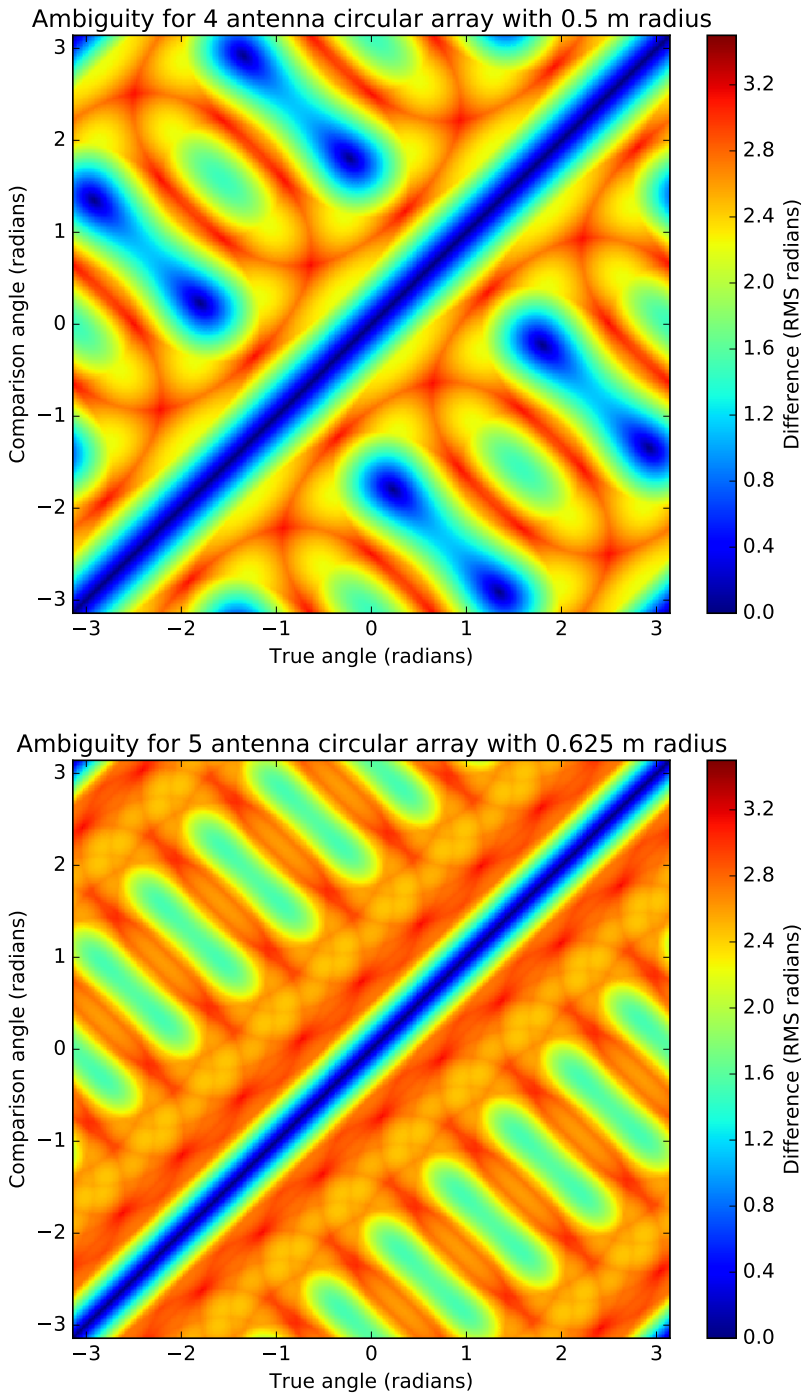


Figure 3.3: Ambiguity simulation for a 4 element (top) and 5 element (bottom) array. The simulation is for a fixed frequency / baseline length, with variables for the true angle of arrival and a comparison angle, with output showing how different a signal from the comparison angle looks to the true angle. Ambiguity gets worse when the difference goes to zero.

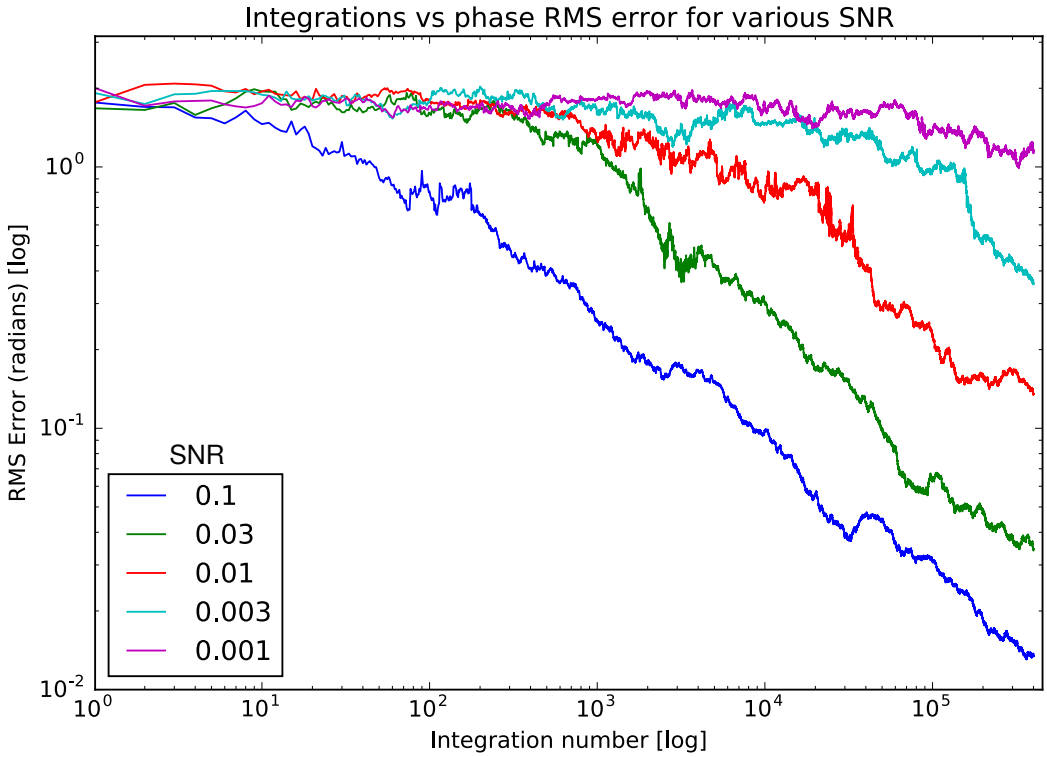


Figure 3.4: Number of accumulations (aka: integrations) vs output error for various input SNR levels. SNR here is defined as signal power divided by the noise power in a single FFT frequency channel for a 2k point FFT.

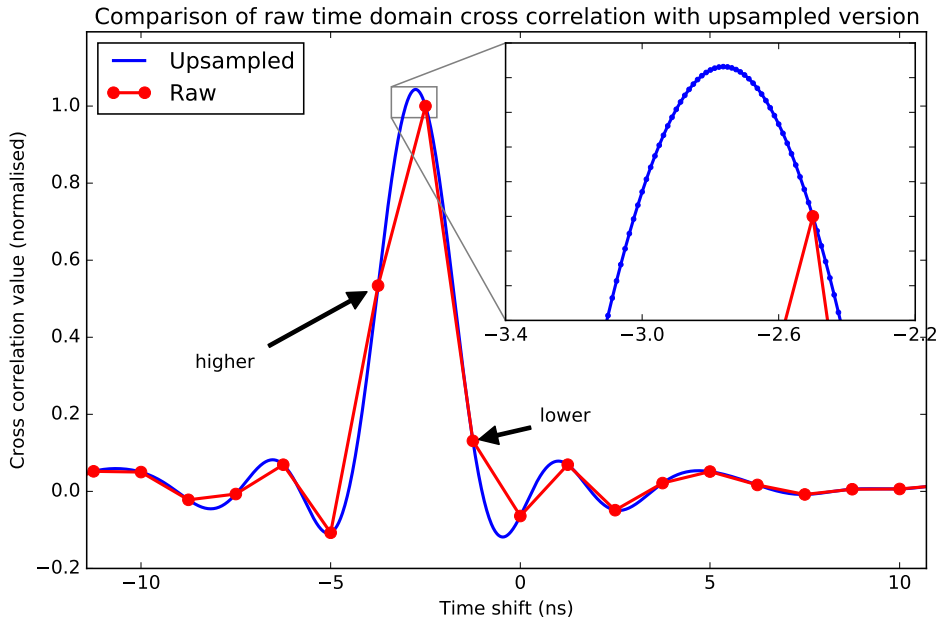


Figure 3.5: Original cross correlation output with upsampled version overlaid. Upsampling is done via the Fourier method, zero padding the frequency domain signal 100x before inverse Fourier transforming.

accumulation is not done for impulsive signals. What it does do, however, is upsampling. To demonstrate this, a simulation of Fourier method upsampling is done here to show how it works. The Fourier method, also known as interpolation, involves taking the discrete Fourier transform (DFT) of the received impulsive signal, zero padding the DFT output with $L \times F$ zeros where L is the length of the observed signal in number of samples and F is the factor that the signal should be upsampled by. Next the inverse Fourier transform of the padded signal is done to get it back to the time domain. It is permissible to zero pad a Frequency domain signal provided the received signal does not have any out-of-band or aliased components. This can be ensured with good analogue filtering.

It is worth noting that this interpolation can be done either on the raw signals or equivalently on the output of the cross correlation. This is important as there will be significantly less computation involved in interpolating the output of the cross correlation than the raw time domain signals. The reason is that correlation is only done over a small window of the raw signal, not over the whole signal. It is permissible to correlate only over a small window, as the correlation peak must lie somewhere within the interval of the propagation delay over the antenna array elements.

An example is shown in Figure 3.5. Here, the red signal with few datapoints is the original output of the time domain cross correlation. There is a clear peak data point in red, however the data point to the left is higher, implying that the real, continuous signal's peak is probably more to the left of the observed peak. When an upsampling is done and plotted in blue, the peak of the interpolated signal is more what would be expected indicating that the upsampled version has a more accurate peak due to the increased resolution.

Upsampling via DFT and zero padding is the method that was used in this research as it is straightforward to implement, and computationally efficient given the speed of the FFT library.

It also does not require any information about the signal or noise spectra. However there are more advanced techniques for TDoA estimation which are explored and compared in depth by Knapp and Carter [21]. Those techniques typically require running each received signal though a filter before performing the cross correlation, in order to extract a maximum likelihood (ML) estimator for the time delay. They are also able to be tuned to better handle multiple time delays such as in the case of multipath. Further research can be done on whether impulsive RFI is would benefit from one of being filtered by one of these processors.

3.4 Summary

This chapter has discussed how a system with an antenna array, an DF front end, digitisers and DSP can be used to DF both narrowband and impulsive RFI. The direction finding algorithms for narrow band and impulsive signals have been drawn up. They are similar in that they both observe a parameter of interest and compare it to sampled points on the computed antenna array manifold vector.

It has been shown how antenna geometry impacts ambiguity performance. In general, having more elements provides lower ambiguity, but an odd number of elements in a circular array is better than an even number. Additionally, it was discussed how the element spacing is a trade-off between frequency domain ambiguity and time domain resolution.

Accumulation has been shown to allow the system to see below the noise floor in the frequency domain, and interpolation used to re-sample a band limited signal to a finer sample spacing and hence improve the accuracy of the extraction of the peak in the correlation.

Simulations of the antenna array geometry, the accumulation gain and upsampling process have all been done to build confidence in the techniques.

Chapter 4

Antenna Array and RF Front End

This chapter details the design and characterisation of the first subsystem in the direction finder: the antennas and RF front end. The RF front end consists of low noise amplifiers, low pass filters, and cabling. As discussed in the previous chapters, the aspects of the array which need to be optimised for are:

- Minimizing the phase difference in the RF path for each antenna. The DF algorithm is based on phase/time difference measurements and hence path mismatches will result in errors. Some phase difference can be calibrated out but the calibration routine will be most robust when the initial error is low.
- Spacing of the elements in the circular array needs to trade off between too large an element spacing and too small a spacing. Too large will result in phase ambiguity, too small will result in difficulty measuring the impulse time difference as well as coupling between the elements. The compromise is to make the array as large as possible before phase ambiguity becomes a problem.

The ADCs which were available for use in this project influenced the antenna and operating frequency band choice. They are explored more in Chapter 5 but will be briefly noted here as justification for the decisions taken at this stage. The ADCs were two Atmel AT84AD001B dual-input cards which sample two inputs in phase. They were clocked up to 800 MHz. This implies four simultaneous inputs with a Nyquist frequency of 400 MHz.

The chapter will first discuss design and construction of the antenna array, then move on to the RF front end. Finally, characterisation of the amplitude and phase performance of the RF front end will then be done.

4.1 Antenna Array

As shown in the simulations in Chapter 3, it is preferable to use an odd number of elements in the circular array as the ambiguity is reduced when there are no lines of symmetry. A circular array is selected due to the equal coverage over the field of view.

However, as mentioned there is a constraint in the hardware that is being used for this project in that only four simultaneous inputs can be sampled. Options are hence to:

1. go down to three elements to have an odd number, leaving one unused ADC input.
2. use a five element array and build additional switching logic to selectively route four inputs to the ADCs
3. use a four element array, sampling all elements and attempt to find a way to mitigate the ambiguity introduced by the symmetry of a four element array.

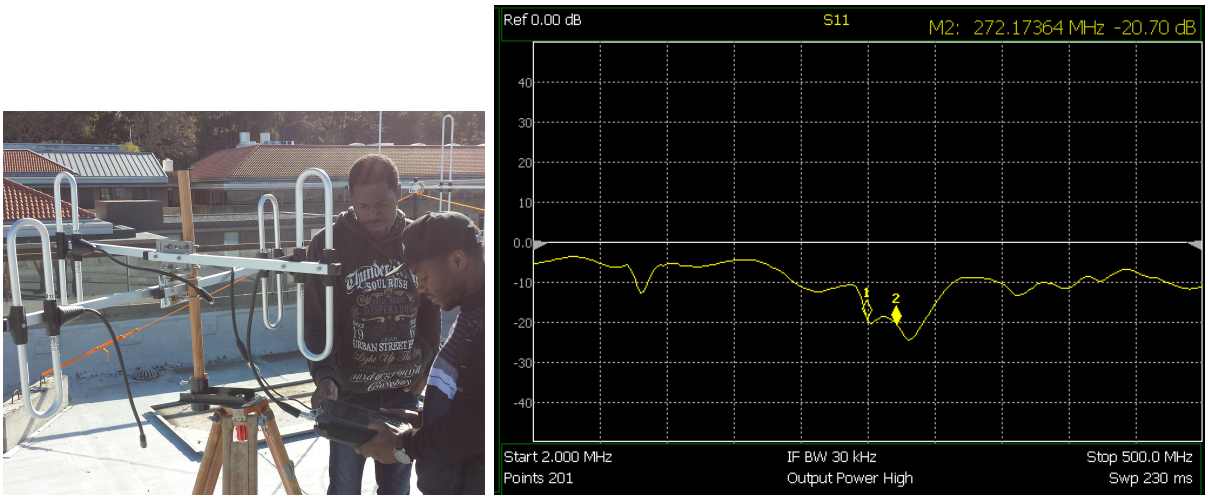


Figure 4.1: Array of four FD-250 folded dipoles being measured and the result of the S11 measurements showing acceptable performance between 200 MHz and 300 MHz.

Option 1 was ruled out as leaving an unused ADC input is wasting potential ambiguity-resolving input. The more input you have the better ambiguity can be resolved and the higher the accuracy of measurement.

Option 2 was ruled out for the complexity of building antenna switching circuitry and trying to dynamically route different inputs to the digitisers.

As such, the approach taken is to construct a four element array using all four inputs. However, the circular array will be *deformed* in order to remove the lines of symmetry.

Four FD250 folded dipole antennas with a center frequency of 250 MHz were purchased. This centre frequency was picked as it is around the middle of the 400 MHz ADC usable bandwidth. Measurements were taken and acceptable S11 performance in the band of interest was observed as shown in Figure 4.1. The antennas were mounted in a deformed circle. Deforming involved shortening one of the mounting booms, lengthening another one, and rotating a third boom by around 20°. The amount of deformation will be trade off between deviating too far from a circle and hence having performance that is not the same in all directions, or being too close to a circle and hence suffering from ambiguity. This project does not go into depth on array deformation strategies; the array design is proof of concept and the deformation we can do was limited by what could be easily changed on the booms what were supplied with the elements, but three different configuration were tried and the best performing was selected. There is scope for further research to explore optimal deforming for ambiguity reduction.

In order to be able model the array in simulations and field trials, it was necessary to assign coordinates for each element. A Python program[1] was written which takes as input measurements of all of the baselines and does a least square errors calculation to assign coordinates to each element. The algorithm of this calculator is discussed in detail in Appendix C but in brief, it first puts the two elements which are furthest apart on the x-axis, then does least-squares iteratively until the coordinates which best fit the measurement are found. The coordinates are then approximately centered on $(0, 0)$. The resulting coordinates are written out to a JavaScript Object Notation (JSON) file which can be read easily by the direction finding software discussed in Chapter 6. The array geometry calculator is able to calculate coordinates for arrays of any size, it needs only a map of baseline measurements. See Appendix C for more details.

[1] `https://github.com/jgowans/array-element-coordinate-calculator`

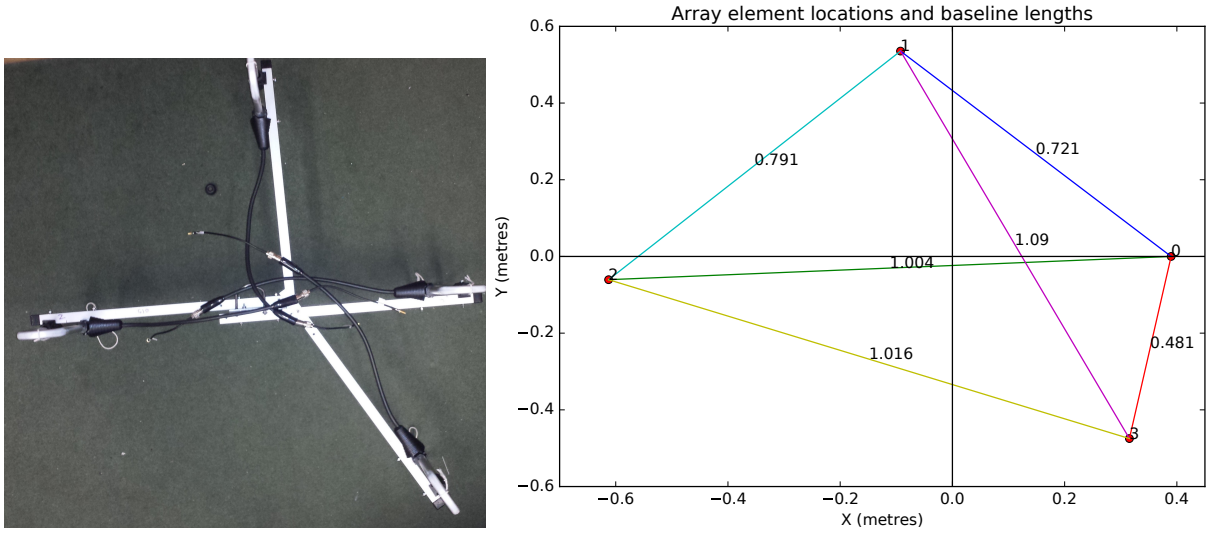


Figure 4.2: Top view of constructed deformed array with a plot of the calculated positions for each element next to it.

Measurements were taken for all of the baselines of the constructed array and fed into the calculator. A top down view of the real array as well as a plot of the calculated geometry produced by the script is show in Figure 4.2. It can be seen that the plotted array matches the geometry of the real one, indicating that the calculator is working as expected.

4.1.1 Ambiguity Performance

The array was deformed in order to ameliorate the ambiguity issue inherent in a square array. A 4-element circular array with equidistant elements is the same as a square. In order to know whether the deformation has improved ambiguity performance, the coordinates of the real constructed array are fed into the ambiguity simulator developed in Chapter 3. This allows for comparing of the deformed 4 element array against a standard 4 element array. The results are plotted in Figure 4.3 and Figure 4.4.

The two figures show that ambiguity has to a large extent been reduced by the deformation, especially at the centre frequency of the folded dipoles, 250 MHz. The dark blue points have mostly been eliminated, meaning that angles that before were difficult or impossible to tell apart now exhibit more distinct baseline phase shifts. This is at the expense of slightly non-uniform performance and less distinct peaks in phase difference (seen by dark reds) in some directions. This is a good trade-off and the deformed 4 element array is so far considered useable for the project.

An additional measurement of the performance of the various array configurations was done by simulating how susceptible the accuracy of the angle of arrival calculation is to corrupting noise on the input phase measurement. The simulation involves getting the ideal phase shift of each baseline in the array for some input angle, then corrupting all of the baseline phase shifts by some RMS error and re-running the resulting corrupted phase shifts thought the DF algorithm. The output angle is then compared with the original angle. This is done for all angles over 2π radians and the RMS error of the output angle calculated. This is done for a range of phase error corruption from 0 to 1 radians RMS, for antenna configurations of 3, 4, 5,

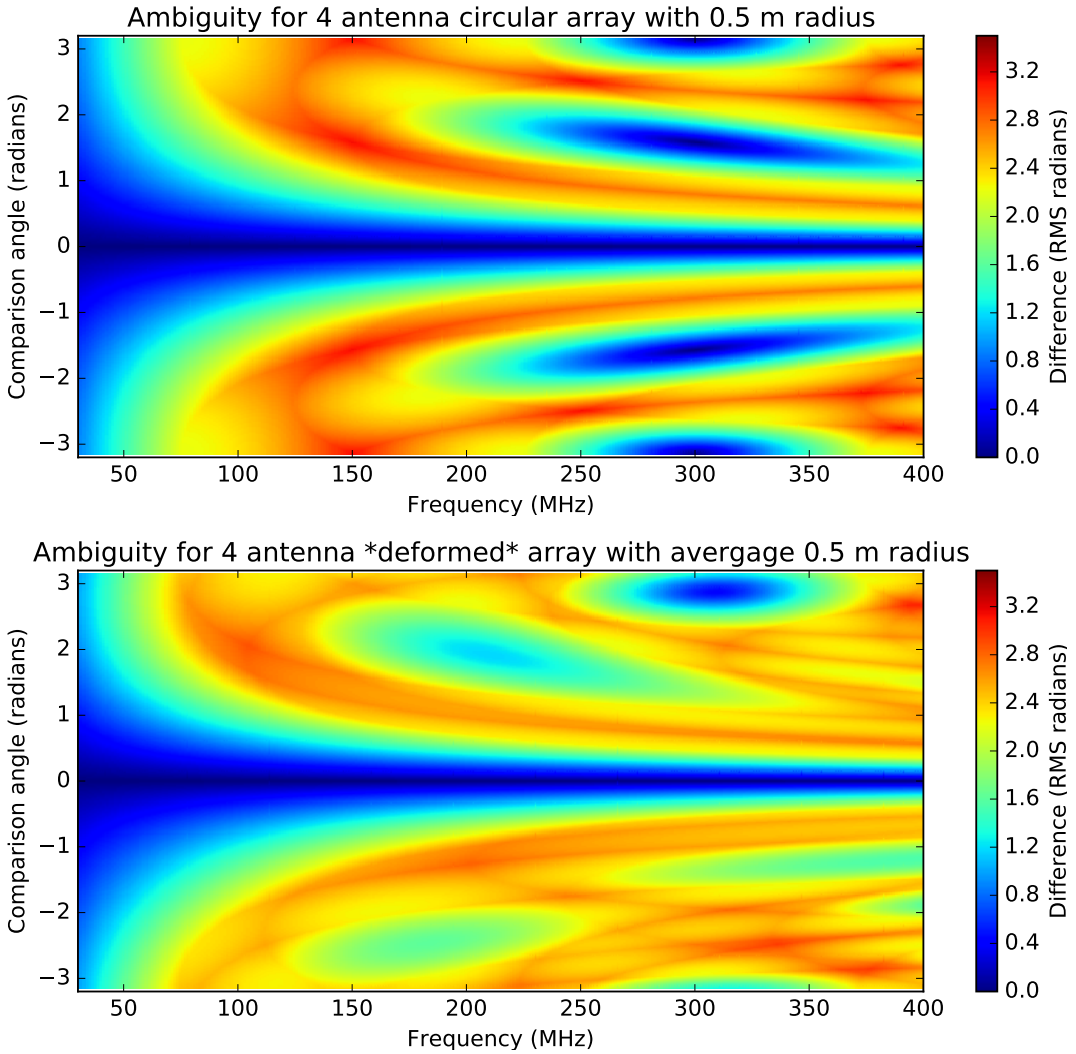


Figure 4.3: Ambiguity plots for 4 element square and deformed arrays with reference signal arriving at 0 radians.

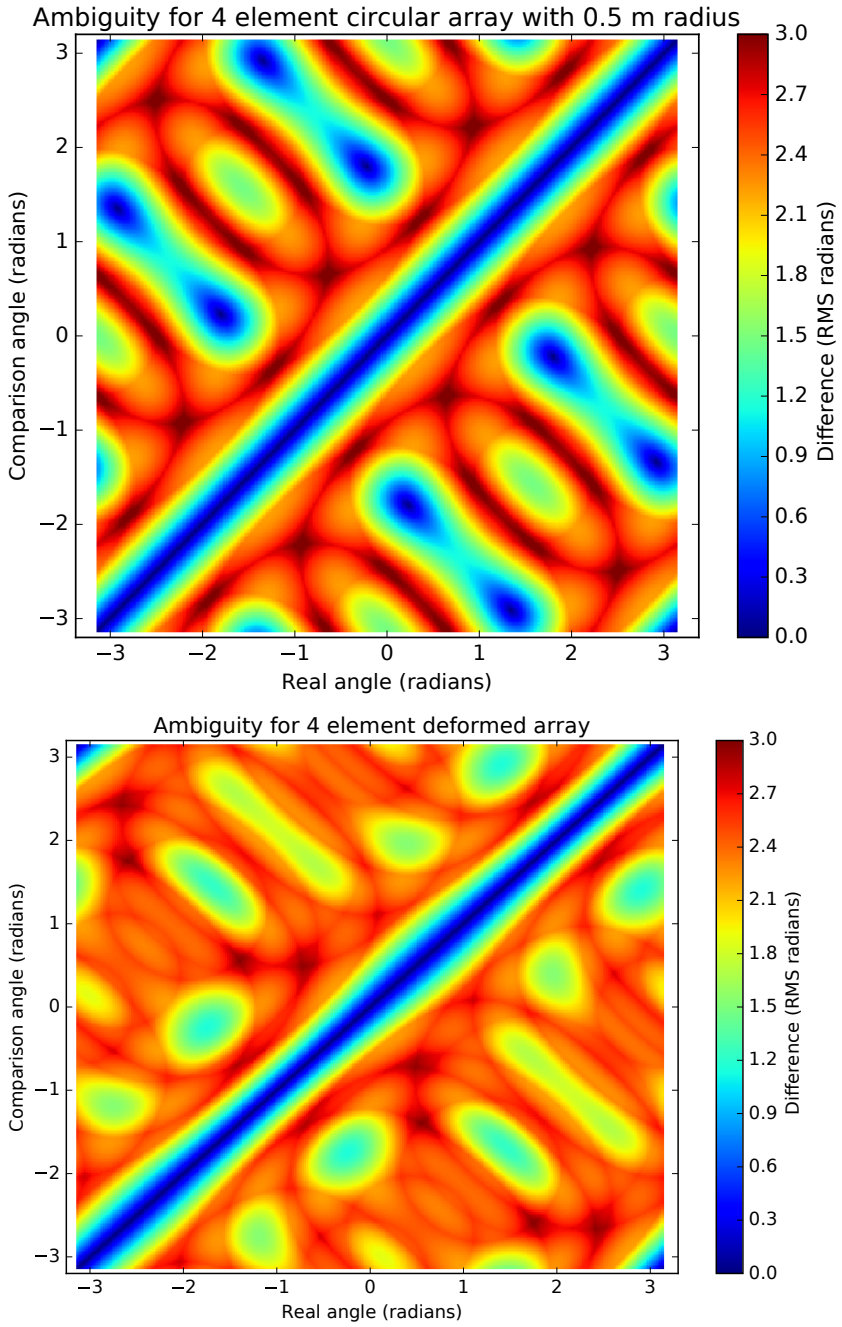


Figure 4.4: Ambiguity plots for 4 element square and deformed arrays at 250 MHz with varying reference and comparison angles.

6 and 7 elements as well as for the 4 element deformed configuration with has been constructed here. A script to perform this simulation was written[1] on top of the original simulator and array modeling package, and the results can be seen in Figure 4.5. From the figure we observe that the even numbered array configurations (4 and 6 elements) have a non-zero error right from the start due to the points of total ambiguity which they exhibit. The 3 element array becomes corrupted by noise very quickly and corruption increases fast due to not having as much information input as the larger arrays. The most interesting observation is that the 4 element deformed array performs only slightly worse than the 5 element array. This observation further indicates that the design decision of deforming a 4 element array rather than going for a 3 element array is a correct decision.

To see exactly how the output error arises in the square array and how it is mitigated by the deformed array, snapshots of DF performance at specific input RPM phase error values were taken. The performance of the square array is shown in Figure 4.6 and the deformed one in Figure 4.7. For angles along the x-axis, the theoretical array manifold was calculated, then corrupted with noise and fed back into the DF system to get the estimated angle of arrival (y-axis value). We see that the square array manifests output error earlier than the deformed one, and that the error congregates at the ambiguous points. For the deformed array it can be output error manifests only after larger input error and the output error is more evenly spread than the square array, indicating that the highly ambiguous points have been smoothed out or eliminated.

4.2 RF Front End

The RF front end, being the bridge between the antennas and the ADCs, is responsible for filtering and amplifying the signals. As discussed in Chapter 3 it is necessary for an interferometry system to have the phase and amplitude response of its RF front end components approximately matched. The ADCs have 400 MHz of usable bandwidth and the antennas have a centre frequency of 250 MHz. Hence, the filtering should cut off before 400 MHz and the amplification should occur around 250 MHz.

Suitable parts were purchased. The low noise amplifier (LNA)s which are used are the ZLF-500HLN from MiniCircuits. This part operates from 10 MHz to 500 MHz which is ideal for the application. The gain is approximately 21 dB across this band according to the datasheet, drawing up to 110 mA. The low pass filter (LPF)s which were purchased are VLF-225+ parts from MiniCircuits having a 3 dB point at 350 MHz, which defines the usable bandwidth of the RF front end.

The RF front end needs to be able to be taken out into the field and used while running on batteries. Two ZIPPY Compact 1000 mA h 3S 25C battery packs and 4 LM7815 regulators were acquired. Each amplifier has its own regulator to try to reduce any electrical coupling between the regulators through their power rails. The battery packs output 12.5 V when fully charged meaning a combined input to the 15 V regulators of 25 V. Hence the regulators are dropping 10 V at 100 mA or 1 W. The LM7815s can handle this dissipation provided a heat sink is connected, which was done. This power distribution circuitry was put onto veroboard.

All of the amplifiers and filters and the power distribution circuit were mounted to a wooden board. Care was taken to make sure that the low pass filters were securely and firmly attached to the board as there is a risk that the SMA connector could be damaged if bumped while in

[1] `https://github.com/jgowans/phase_ambiguity/blob/master/simulate_df_error_vs_visibility_error.py`

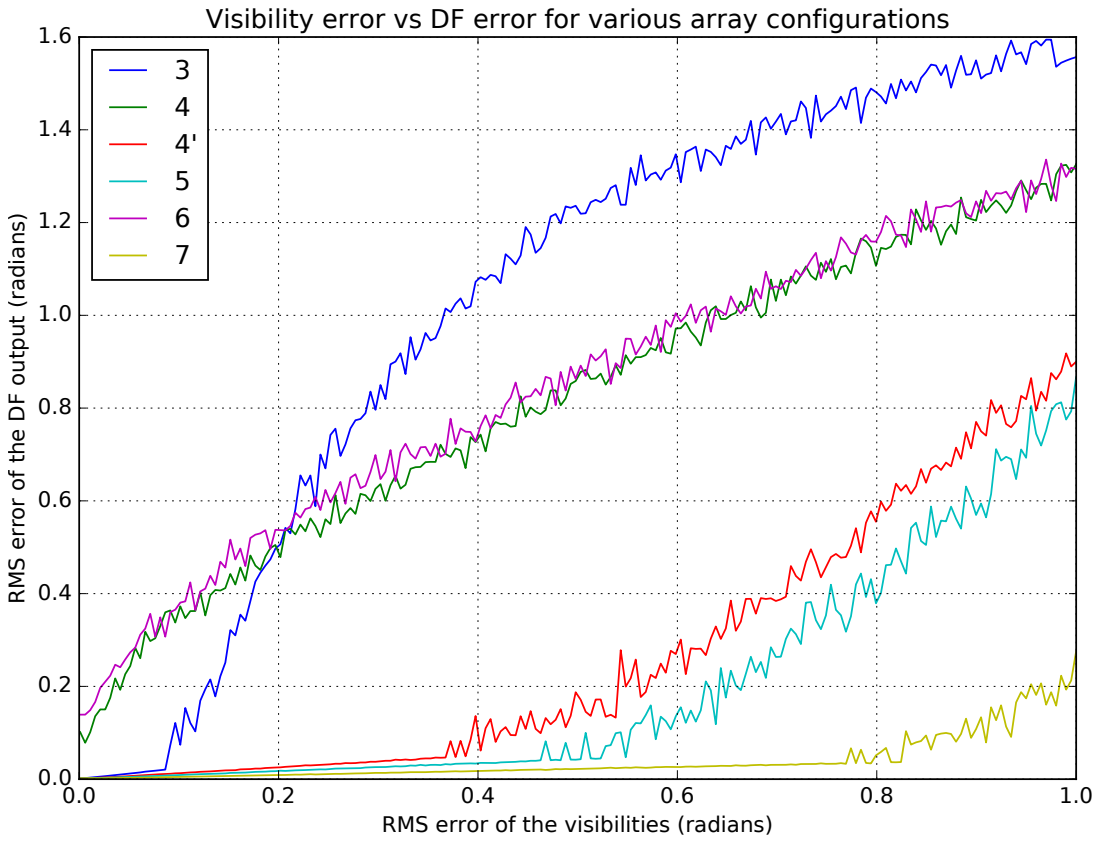


Figure 4.5: Susceptibility of arrays with different numbers of elements to input RMS phase error. The line labeled 4' represents the deformed 4 element array that has been constructed. A "visibility" here is the one pair of antennas; there are multiple visibilities for an array.

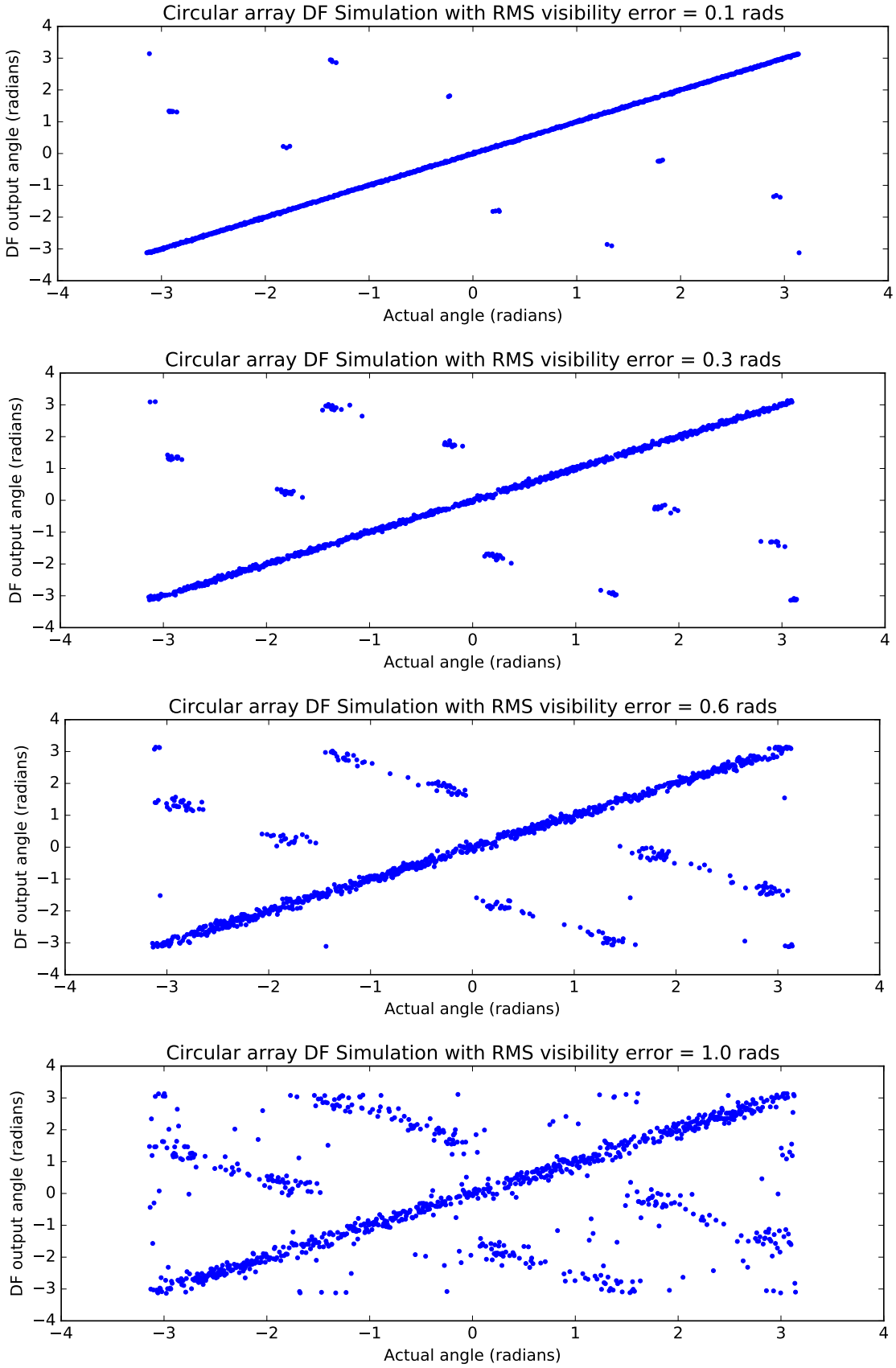


Figure 4.6: Four element *square* (evenly sampled circle) array's DF output vs simulated input when corrupted by increasing levels of noise. The dots off of the main straight line are where ambiguities introduced error.

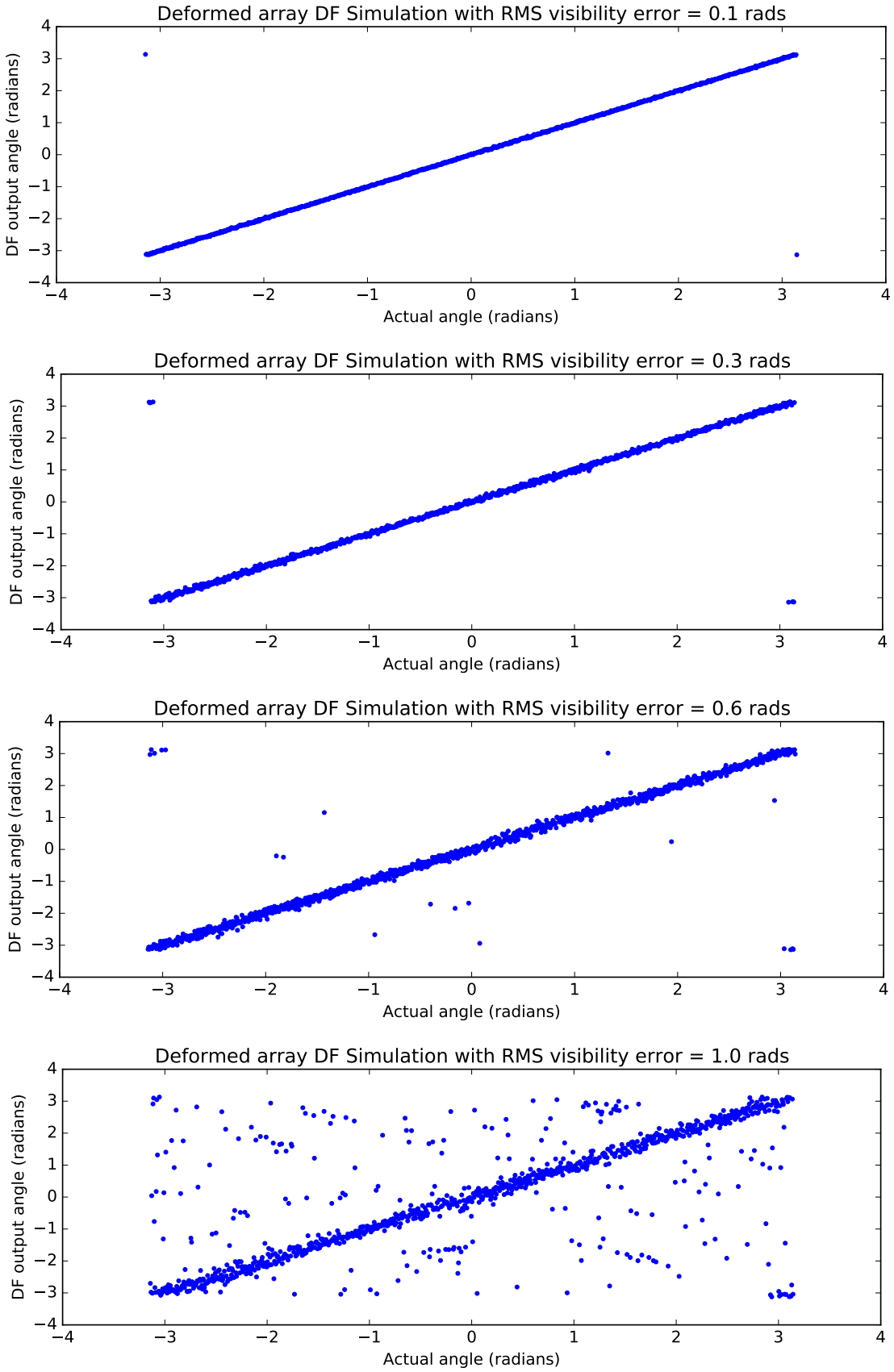


Figure 4.7: Four element *deformed* array's DF output vs simulated input when corrupted by increasing levels of noise. The dots off of the main straight line are where ambiguities introduced error.

the field. The circuit diagram for this board and the resulting hardware implementation for this RF front end are shown in Figure 4.8.

Finally, the cables need to be carefully matched as well. Provided the cables are of the same type, their velocity factors will be equal and it is hence only necessary to ensure that they are of the same length. Most of the cables are the same length except for those fixed to the antennas which are different by a few tens of centimetres. The DF software needs to be able to cater for these measured length mismatches.

The LPFs and LNAs may have phase mismatches due to manufacturing tolerances. In order to check whether the LNAs provide the expected gain and to check whether the phase matching of the RF front end is good, the whole subsystem of cables, amplifiers and filters was connected to a network analyser and S21 measurements[1] taken. The results are shown in Figure 4.9. It can be seen that the amplitude drops off rapidly at around 400 MHz providing necessary anti aliasing. At 350 MHz the difference between the in band signal and the aliased signal is almost −30 dB which is sufficient isolation to consider the system usable up to 350 MHz. The phase matching is good, within a degree or two over the whole band of operation.

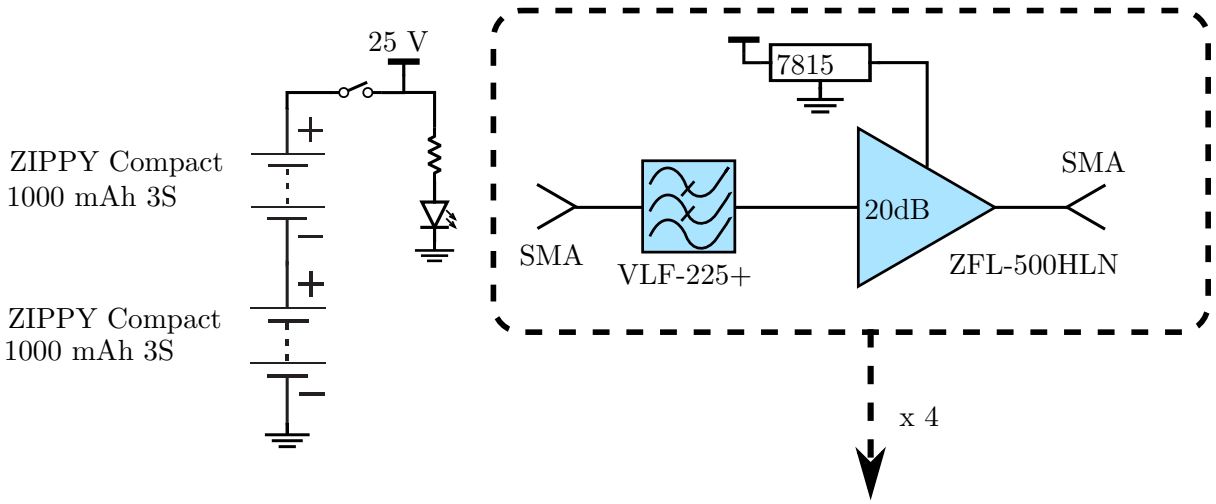
4.3 Summary

This chapter has detailed the design, construction and measurement of the antenna array and RF front end subsystem. It has been shown how a 4-element array can be deformed from a square to produce ambiguity performance that is not far off of what has achieved by a 5-element array. Code was written to be able to introduce a coordinate system to elements from baseline distance measurements.

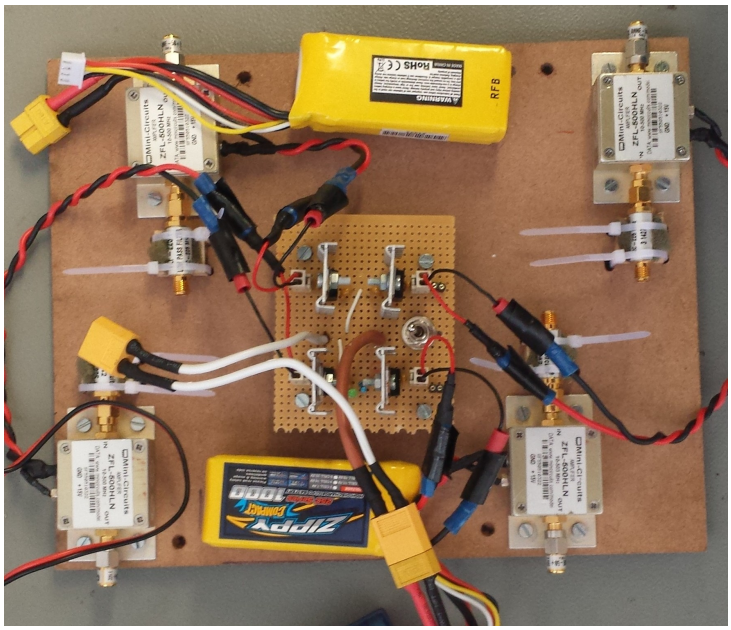
The RF front end was built to provide amplification in the band of interest and attenuation of out-of-band aliased signals. Mounting the circuitry on a board and making it battery-powered will allow it to be taken into the field easily.

Measurements of the performance of the RF front end show it should work as expected.

[1] S21 is a parameter produced by taking a measurement on a vector network analyser (VNA). The VNA inject a reference signal swept over frequency into the input of a 2-port system and measure the output at the other end. It then computes the transformation that the system applied to the signal at that frequency in terms of amplitude and phase shift. This input-to-output transformation is S21 and is hence an indication of how the system transforms the signal over frequency.



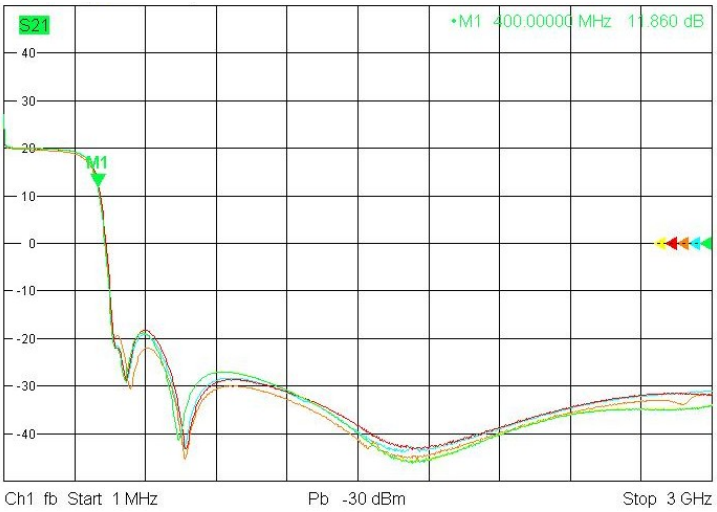
(a) Schematic of the RF front end circuitry.



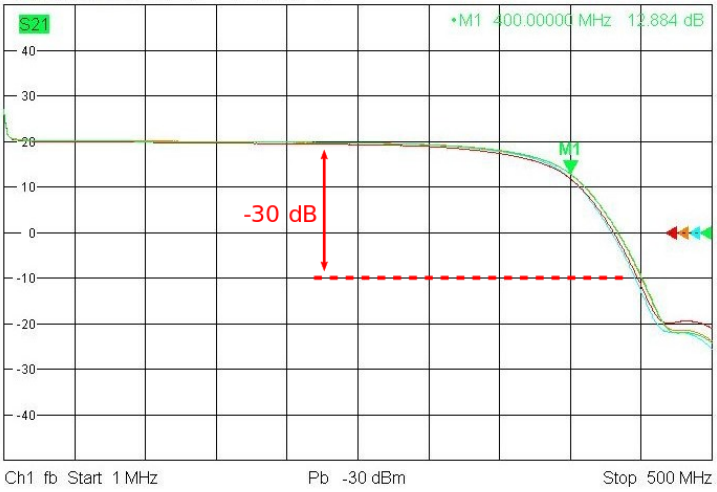
(b) Board containing amplifiers and filters in the corners, batteries velcroed down at the top and bottom, and regulators and connectors on the veroboard in the middle.

(c) Close up of a low pass filter connected to an amplifier, held down to make it more rugged in the field.

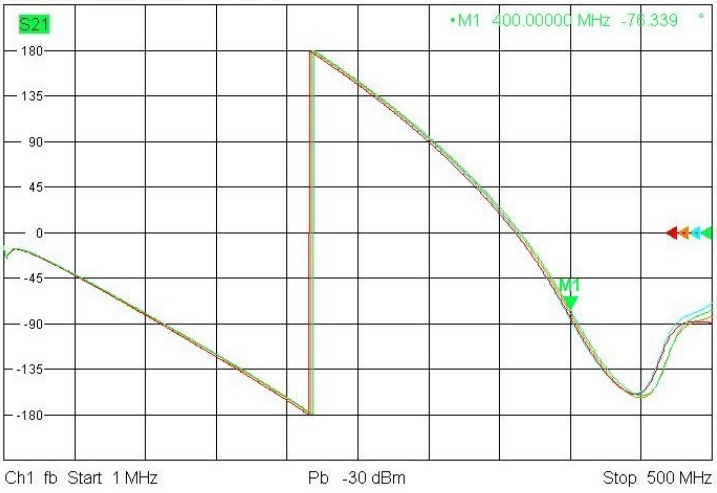
Figure 4.8: RF front end design and construction.



(a) Gain from 0 to 3 GHz in dB. Antialiasing is sufficient. The iADC is only receptive to signals up to 3 GHz so it's only necessary to look at filtering up to there.



(b) VNA measurements of S21 of RF chains showing antialiasing properties



(c) Phase from 0 to 500 MHz showing very good phase matching over operating frequency range below 350 MHz

Figure 4.9: S21 measurements of RF front end subsystem.

Chapter 5

FPGA Firmware

This Chapter details the design, implementation and testing of the FPGA firmware. The FPGA firmware is the second subsystem, responsible for taking in high speed data streams from the ADCs and doing the initial DSP operations. These operations include FFTs, spectrum cross correlations and accumulation in the frequency domain, and impulse detection and snapshotting in the time domain. Testing involved both simulating the data path as well as putting the firmware on the physical FPGA, sending input RF signals to it and analysing the output of the FPGA's DSP.

5.1 Hardware Specifications

As discussed in the user requirements, the FPGA processing board used here should be the same which is used by the SKA: the ROACH[1] board. This board has a Xilinx Virtex-5 FPGA on it, with connectors for Z-DOK+ cards for plugging in ADCs and a DDR2 Dynamic Random Access Memory (DRAM) connector for high volume[2] data storage. There is a PowerPC subsystem on the ROACH running Linux which provides a 1 Gbps Ethernet interface for programming and data readout from the FPGA memory. The PowerPC runs a daemon called tcpborphserver which speaks the KATCP protocol used to interact with the ROACH. There are other blocks on the ROACH board as well, but the ones listed here will be the ones utilised in this project.

There are various ADC cards compatible with the ROACH. The ones which were made available for this project are iADC cards. Each card has an Atmel/e2V AT84AD001B 8-bit dual ADC chip on it. It is designed for I/Q sampling but can be configured to sample both channels in phase - exactly what is needed here. The iADC can be clocked up to 1 GHz and must be externally supplied. The ADC feeds the clock signal through to the FPGA along with the sampled data with a demux factor of 4:1 meaning that for every 4 ADC clock cycles (4 samples), the FPGA gets one clock cycle with all 4 samples on the ADC data bus. This implies that the FPGA will run at exactly a quarter of the ADC speed, and needs to be able to handle 4 samples per channel times 4 channels = 16 samples every clock cycle.

As the ADC is 8-bit is has a dynamic range of $20\log(2^8 - 1)\text{dB} = 48\text{dB}$. If the amplitude of the received signal changes it may be necessary to change the front end filters and amplifiers to ensure it stays in the range of the ADC.

The clock source is a Valon 5007 dual synthesiser which is USB programmable from 137 MHz to 4400 MHz. Both ADCs need to be clocked exactly in phase so the Valon is fed into a Mini-Circuits ZESC2-11+ power splitter and then to the ADC cards.

The ROACH board with connected ADCs is shown in Figure 5.1. All of the components are in a metal case for RF shielding, although the cover has been removed for the photo.

[1] `https://casper.berkeley.edu/wiki/ROACH`
[2] Compared to the Virtex-5's on-chip storage

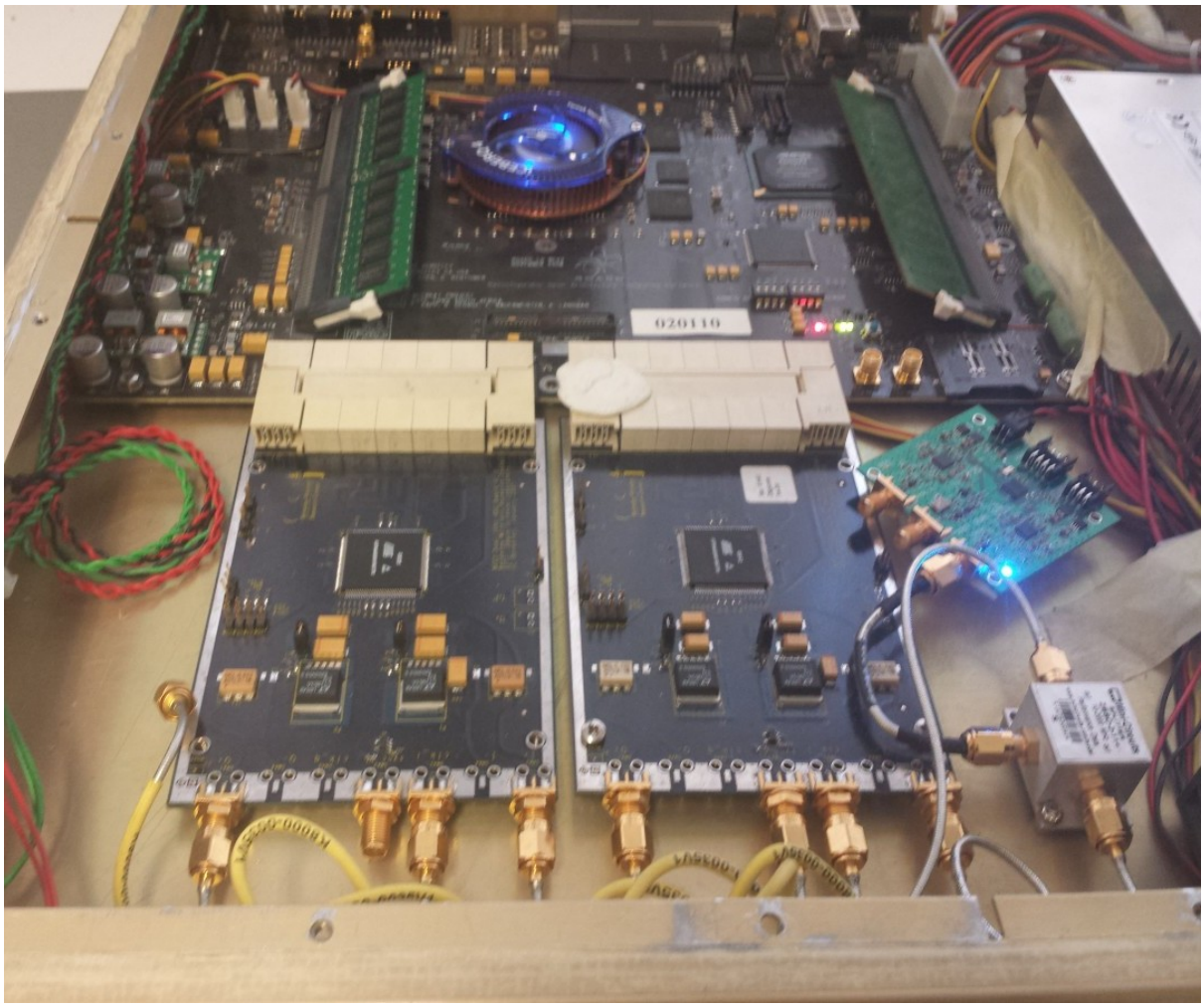


Figure 5.1: Photo of the ROACH board and ADCs. Bottom: Two Atmel AT84AD001B dual ADC cards with SMA connectors going in to them and connected via Z-DOK to the ROACH board. Bottom right: ADC clock synthesiser feeding into power splitter which in turn clocks each ADC board in phase. Top: main ROACH board. The Virtex-5 is under the blue fan. To its left is the ROACH's DRAM module. To the right of the fan is the PowerPC with its DRAM module.

5.1.1 Note on Quantization Noise

For an N bit ADC, the SNR due to quantization noise over the full sampling bandwidth is [22]:

$$\mathrm{SNR} = 6.02\mathrm{N} + 1.76\mathrm{dB} + 10\log_{10}\frac{f_s}{2\mathrm{BW}} \tag{5.1.1}$$

The ADC is use is 8-bit at $f_s = 800e6$. As will be seen later, the bandwidth of one channel in the 2048-point FFT is 200 kHz. This yields a SNR of 85 dB.

5.2 Firmware Development Toolchain

There are a few software tools necessary to develop for the ROACH. They are briefly listed here and detailed more in Appendix A.

Xilinx Integrated Synthesis Environment (ISE) 14.7 is needed for synthesising (compiling) Hardware Description Language (HDL) files to FPGA bitfiles. This proprietary tool is maintained by Xilinx and made available through the SKA. As well as this key task of compilation, it also has peripheral tools for debugging and optimisation which proved useful in getting the design to meet timing. More info in Appendix A. Although it also works on Debian/Ubuntu systems, ISE is most stable and only officially supported for Red Hat Enterprise Linux (RHEL). As such a CentOS system was used for FPGA development work as it's mostly the same as RHEL at the binary interface level.

Matlab R2012B with Simulink was installed, along with the `mlib-devel` plugin. The majority of the design work for ROACH done in this project is done at the block diagram level in Simulink. The `mlib-devel` plugin was originally created by Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) and is now contributed to extensively by the SKA. It contains a multitude of highly useful DSP blocks which are built on top of the Xilinx cores which can be wired up in Simulink. These range from relatively simple blocks like edge detectors to very complex ones like real-time FFT blocks and polyphase filterbank blocks. Additionally, blocks for interfacing with hardware like ADC cards, DRAM modules, PowerPC shared registers and generic IO are included. This includes the net definitions for the components on the ROACH board. It was necessary for this project to update some blocks during this project, hence a clone of this `mlib-devel` repository was created[1].

5.3 Design

There are a number of subsystems developed for the FPGA which will be discussed here. Most of these subsystems required extensive development, testing and tweaking. The full design is shown in Figure 5.2. It may be difficult to see all of the detail of the design, but the purpose of the figure is to give an indication of the structure, layout and data flow in the FPGA. The following subsections discuss the purpose of the key blocks in the design. A more detailed look at the composition and structure of some of the blocks can be found in Appendix A. That Appendix also discusses the challenges that were encountered during the FPGA design, and the optimisations that were done to get it working.

[1] `https://github.com/jgowans/mlib_devel`

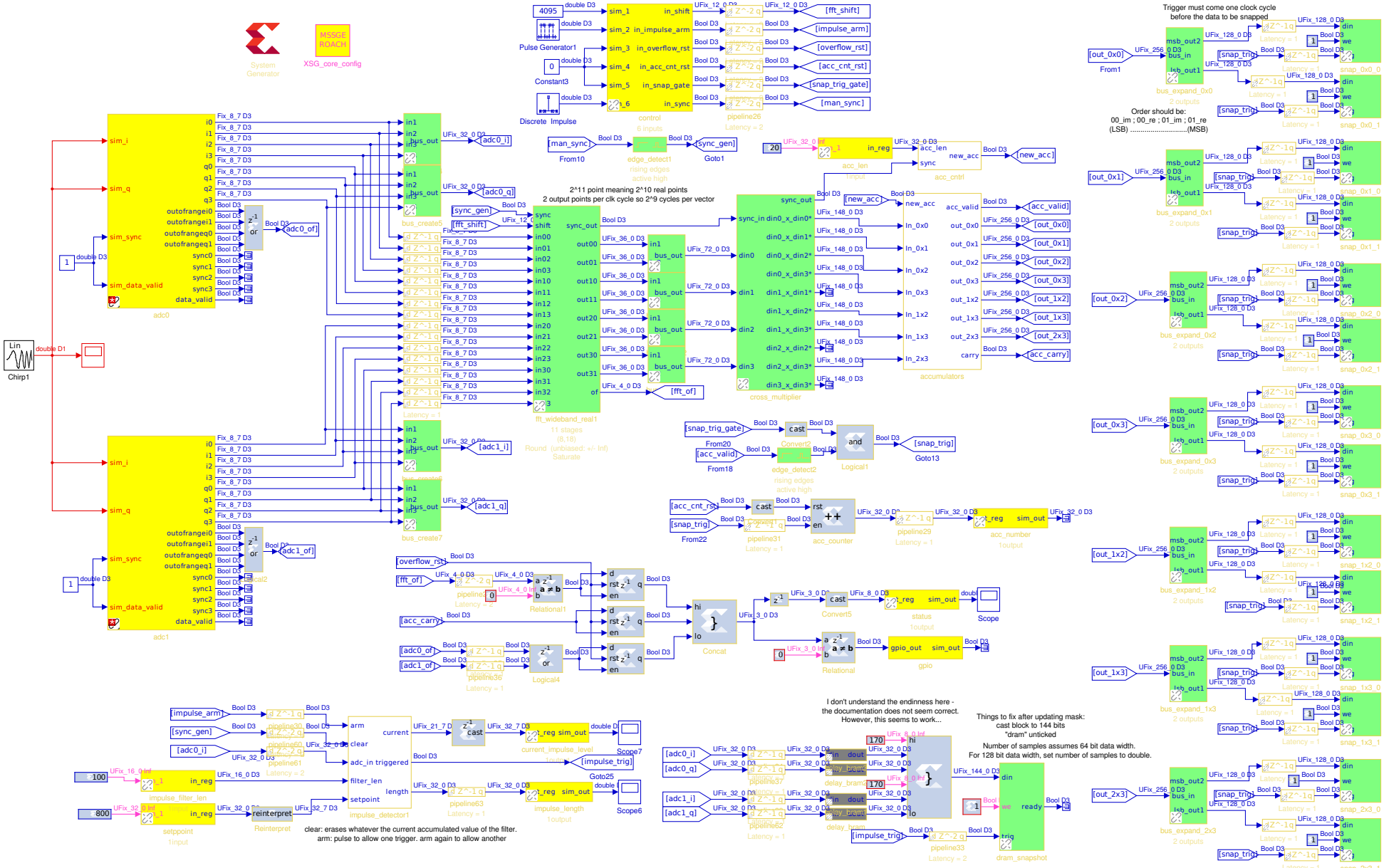


Figure 5.2: Full FPGA design in Simulink containing frequency and time DSP paths and control logic.

5.3.1 ADCs

The yellow blocks on the left represent the two dual ADCs providing 2×2 channels which will be referred to as channel 0, 1, 2 and 3. Each ADC has $2 \times 4 = 8$ buses for the 8-bit signed integer samples that are clocked in every clock cycle. As mentioned earlier the FPGA runs at a quarter of the ADC clock speed hence the ADC presents 4 samples per clock cycle. Also included are logic lines that indicate clipping. For simulation purposes, normal Simulink signal source blocks can be connected to the ADC, allowing the construction of arbitrarily complex input signals. The example shows a simple chirp connected. The output from the ADCs fans out to two distinct signals paths: a frequency domain path and a time domain path. These are now explored.

5.3.2 Frequency domain signal path

The frequency domain path starts with an FFT block, the first big green one, which performs real time FFTs on the data from the ADCs. Although it is one block it does four distinct Fourier transforms under the hood: one for each of the ADC channels. It is an 11 stage FFT meaning that it does $2^{11} = 2048$ points. It has an optimisation for real-valued signals in that it only outputs the positive half of the frequency spectrum in order to save bandwidth. The FFT has been configured to do full bit growth, meaning that at every stage another bit of data is added to the bus size. No bits are thrown away hence weak signals will be preserved. This is at the expense of significant data growth: each channel outputs two 36-bit complex numbers every clock cycle.

FFT output is fed into the second big green block, the Cross Multiplier. This does $\binom{4}{2} = 6$ cross multiplications, one for each baseline which will be referred to as 0x1, 0x2, 0x3, 1x2, 1x3 and 2x3. Additionally, the frequency domain autocorrelation of channel 0, 0x0, is taken to serve as a power spectrum analyser. The cross multiplication involves breaking the signal up into its real and imaginary components, taking the imaginary conjugate of one of the signals by inverting the sign of its complex component and then doing complex multiplication. For each pair of signals A and B where $A = (a + ib)$ and $B = (c + id)$ the FPGA will do $C = A\overline{B} = (ac + bd) + (bc - ad)i$. The cross multiplier does not throw any bits away meaning that the signals go from $2 \times 36 = 72$-bit complex numbers to 148 bits per channel.

The cross correlation signals then go to the Accumulators, which is the white block in the design. As discussed in Chapter 3, the cross correlations will be accumulated (aka: integrated) a number of times in order to allow weak signals to stand out of the noise. The accumulation length is runtime programmable via a control register which will be further discussed later. The accumulator works by using FPGA block random access memory (BRAM)M to produce a delay, and a DSP48E to add the new signal to the current one. The number of samples buffered in the BRAM is the number of points in the FFT spectrum: 2048. The accumulators allow signal growth from 74 bits per complex number to 96 bits. This implies a minimum of ≈ 4 million accumulations when the signals are full scale, although typically this can be orders of magnitude more as the system does not run with full scale input; it is dangerous the keep the ADCs at max power as a small increase in signal strength can cause the electronics to be damaged.

Each accumulated correlation value is wired to its own BRAM snapshot block, the array of small green blocks on the right. Once sufficient accumulations have been run, the snapshot blocks are triggered to take a snapshot of the correlation spectrums. The snapshots then record the full accumulated correlation spectrum of a channel, now 128 bits wide per frequency channel per correlation pair, to shared BRAM memory. This memory can then be read out from the FPGA via the PowerPC over Ethernet.

5.3.3 Time domain signal path

This path is comparatively simpler than the frequency domain path, seeing as the correlations are not being done on the FPGA and seeing as accumulation is not applicable for time domain DF.

The two main subsystems for the time domain path are shown at the bottom of the FPGA design. First is the impulse detector. This takes in the raw ADC data from channel 0, squares the values to get signal power, and then runs a rolling sum filter which provides the sum of the last N samples where N is some runtime programmable number typically between 10 and 1000 samples. This in effect produces an energy reading for the energy within the N-sample interval. If the rolling sum goes above a runtime programmable threshold the impulse detector considers an impulse to be present. It then records the raw time domain data of all 4 channels for as long as the impulse is present, plus some additional buffer at the start and end of the pulse. The snapshot is stored by the green block at the bottom of the design: the off-chip DRAM module. The starting buffer is attained by feeding a delayed version of the raw signal to the DRAM interface. The DRAM module was chosen as it can store a large amount of data, allowing multiple milliseconds of raw data to be stored if desired. Typically only a few microseconds of data is necessary to store an impulsive signal. As well as triggering the snapshot, the impulse detector will also store the length of the impulse which it detected, thus the computer can know much data to read out from the DRAM module when it collects the data. The DRAM data is also accessible through the PowerPC over Ethernet.

5.3.4 Control and Monitoring

The FPGA's runtime configuration and status can be interacted with via shared memory accessible in the address space of the PowerPC. This memory manifests as words of data registers on the FPGA. Slices of these registers are wired to different parts of the FPGA logic to either dynamically change configuration in the case of writable memory or to get status info in the case of readable memory. The registers are the small yellow blocks dotted around the design. The following are a few control and monitoring functions with descriptions which indicate more about how the design works:

- Writable integers for programming accumulation length, impulse detector rolling filter length and impulse detector setpoint. These are all separate 32-bit registers for runtime configuration of these numerical (as opposed to logical) values.

- Impulse detector arm. The impulse detector will only write a snapshot to DRAM if its threshold is breached while the arm latch is high. The latch is cleared when a snapshot is detected, and can be set again by the computer once the snapshot has been read out. The reason for this is to prevent the race condition where a new snapshot could partially overwrite an earlier one which is busy being read out by the computer.

- Snapshot trigger gate. Similar to the reasoning behind the impulse detector arming, this gate to all of the snapshot trigger blocks exists to prevent a new accumulated spectrum from being written on top of one that is being read out. The computer will gate the trigger while it's busy reading the snapshot blocks, and clear the gate once the readout is complete.

- Rolling sum level. This is the current value of the rolling sum filter. By watching the value of the filter for some time, an informed decision can be made about where to put

the setpoint for the impulse detector. Typically this would be a few standard deviations above the mean. Setting its value is a trade off between false positives and false negatives.

- Impulse length. The impulse detector records how long the impulse goes on for. The computer can poll this value to see how much data needs to be read from DRAM. By reading only the minimum data, the readout is faster and the time domain cross correlation is more accurate. The maximum recordable impulse length is a function of the installed DRAM module. For a 512 MB module at 4 x 800 Msps and 8-bit per sample it can store up to 160 milliseconds of raw time domain data. Pulses are typically below 1 millisecond.

- Overflow status latch and reset. The various parts of the system which can overflow feed a latched version of their overflow status lines into a register. Then, whenever a snapshot is read out the overflow register is examined as well so see if the data should be treated as valid or not. There are multiple bits so it can be seen which part of the system overflowed and configuration can be adjusted to remediate this overflow. For ADC overflow this would be by adding analogue attenuators. For accumulation overflow it would be by reducing the accumulation length. For FFT overflow this would be by changing the shift schedule. As well as going to a register to be read out by the computer, this overflow latch is also fed to a general purpose input/output (GPIO) register which drives an light emitting diode (LED) mounted on the ROACH case so that, should any overflow happen, it will immediately light the LED to inform the operator. The computer can acknowledge the occurrence of overflow by pulsing a reset line which will clear the latch.

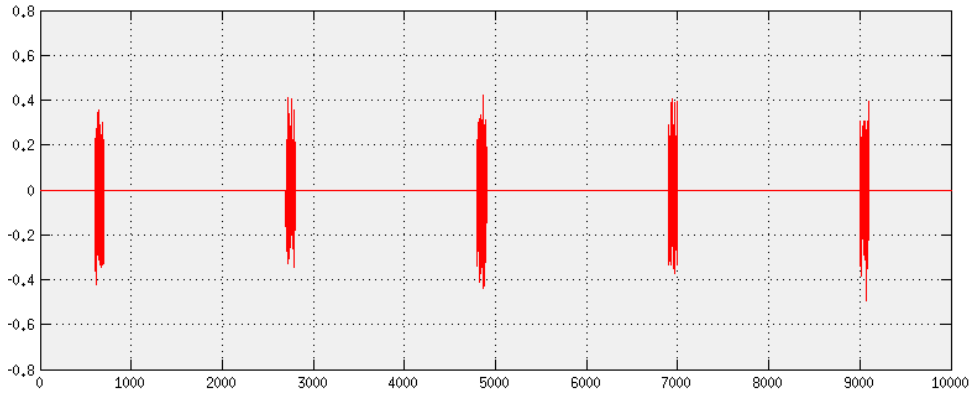
- Manual sync pulse. Data is streamed into the FFT block and frequency channels come out in order which eventually need to be stored in snapshot memory, lowest numbered channel at the start and highest channel at the end. As such, something needs to keep track of which channel is the first one. The sync pulse achieves this by resetting the FFT, and the FFT will then output the sync pulse at the same clock cycle as channel 0. This sync pulse is then propagated through the multipliers and into the accumulators, which ensures that the start of an accumulation is in phase with the start of a spectrum. This sync line is pulsed by the computer when it connects to initialize the data path.

- Accumulation counter. Every time an accumulation is completed and a new one started, this counter increments. The computer can use this to know when it's time to read out a new spectrum and also to ensure it's keeping up and reading out all snapshots.

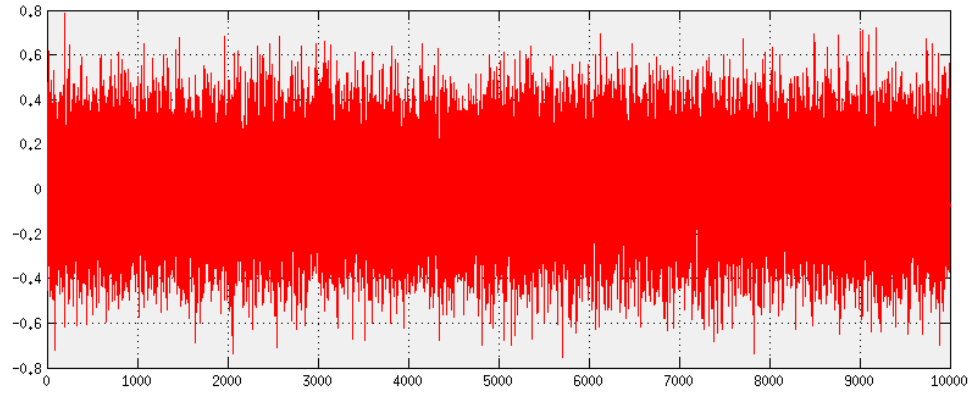
- FFT shift schedule. Sets which phases the FFT should shift data down by 1 bit and when it should allow bit growth. It ended up not being needed as it was possible to fit a full bit growth FFT on to the design. However, should a longer FFT or other additional logic be needed in future, it will be necessary to shrink the width of the data path and then use a shifting FFT.

5.4 Simulation

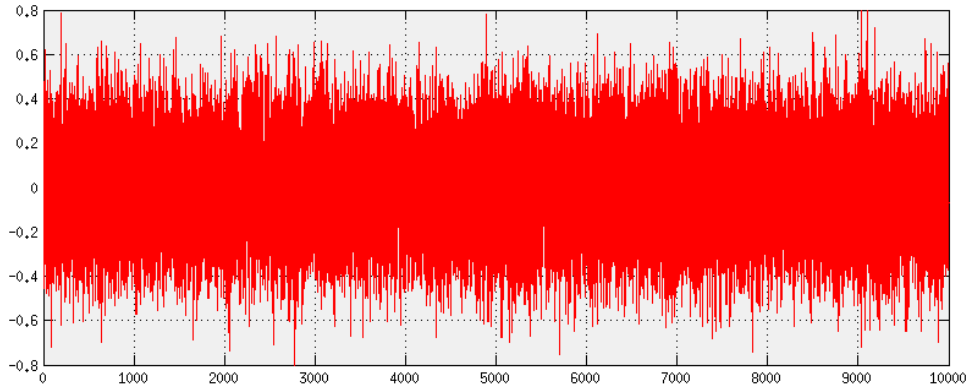
As the design is created in Simulink, the extensive simulation capabilities it provides can be used to validate that the design is behaving as intended. Simulation was used throughout the development process, for both the time and frequency domain signal path. For example, for the frequency domain path a repeating correlated chirp signal with a defined start and end frequency would be injected into the input, and the snapshot blocks would be examined to



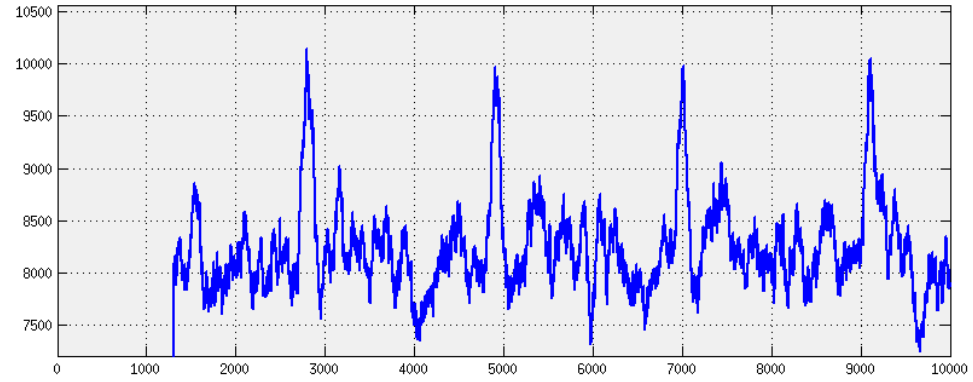
(a) Periodic 1 µs impulse at half of the power of the noise



(b) Additive Gaussian white noise



(c) Sum of impulse and noise. Impulse is not distinguishable by eye.



(d) Output of moving sum of magnitude filter. Additional noise power now clearly visible

Figure 5.3: Simulink simulation of how filter can detect impulse even below noise floor. X-axis is sample number, Y-axis is arbitrary amplitude units.

ensure that there was a flat power spectrum between those frequencies. For the time domain path, short impulses of noise in the presence of continual noise were simulated. Essentially this amounts to an increase in noise power for a brief interval. The output of the rolling average filter is monitored to ensure that it correctly detected a rise in power. An example of this is shown in Figure 5.3. This example has the extreme case where the impulse is actually smaller than the additive noise, yet the power detector is able to detect it and could trigger a capture of it. This example shows that the output of the detector does increase when input power goes up.

5.5 Design compilation

Over the course of this research, the design evolved from being a simple two-antenna time domain data collector to the full four-antenna time and frequency domain DSP subsystem described here. The full design is very large, consuming more than 95% of the available logic on the FPGA. Numerous optimisations were necessary to get it to fit in the available FPGA logic. These optimizations are detailed more in Appendix A.

Part of the compilation process involves defining the clock speed at which the FPGA should operate. The Xilinx tools then move logic around to ensure that all signals can propagate and be valid in the necessary time. Hence faster speeds are more difficult to compile as they make it more likely a path will not meet timing. The timing value that proved reliable for this design was 200 MHz. This implies an ADC clock speed of 800 MHz. The ADCs have a maximum clock speed of 1000 MHz so this speed is fine for them.

The ADC's 800 MHz sample rate implies a Nyquist frequency of 400 MHz which explains why 400 MHz has been used extensively in the previous sections for DF algorithm simulations and antenna design.

5.6 Lab Tests

In order to validate that the FPGA subsystem was working as expected, a hardware setup was created in the lab to emulate the sort of signals which the DF system would see in the field. The setup contained signal sources for both strong impulsive signals, as well as weak narrow band tone signals. The impulses and tones were split and fed to all ADC inputs. As well as getting the correlated signals, uncorrelated noise was added, allowing for comprehensive tests ensuring that the correlation and accumulation part of the DSP path could extract weak signals from noise. The physical setup in the lab is shown in Figure 5.4 with a diagram of the configuration in Figure 5.5. This hardware rig was used during development in conjunction with the Simulink simulations to ensure that the running design was operating as intended.

For the time domain path the tests involved arming the time domain capture, then manually triggering a one-shot impulse. The system successfully detected the impulse and captured it (as well as some start and stop buffer) on call four channels to DRAM. This signal could then be read out.

The frequency domain path tests involved producing a tone at a certain frequency and ensuring that accurate phase difference as seen by each input manifested after accumulation. The expected phase was set by feeding different cable lengths from the splitter to the ADCs. Two situations were tested: one where all cables were of equal lengths hence expecting to see 0° phase shift between all baselines and the other where one cable was longer than the rest, hence expecting some of the baselines to see a phase shift while others to be in phase. The expected

phase shift was checked by measuring the cables on a network analyser. These tests were done at different SNR levels to quantify not only that the observed phase shift was present but also how performance changed with SNR level and with accumulation length. Results for SNR levels of 0.1, 0.01 and 0.001 are shown in Figure 5.6. Note that these SNR levels are for the noise power in just the FFT frequency bin of the signal, not total noise power. Total noise power is order of magnitudes higher as the white noise has power across the whole spectrum. These results show that the system is behaving as expected in two key ways: First, in general, error diminishes as more samples are accumulated[1]. Second, the higher the SNR the faster the error drops off. Based on these results we can say that the system is able to operate all the way down to SNR levels of 0.01 provided integration time can span hundreds of thousands of samples, or a few seconds.

At SNR of 0.001 there is less than 1 bit of information for the signal on the 8-bit ADC. As such the system is relying largely on dithering to capture any correlated signal information. This noise dominance explains the stochastic behaviour of the output: the amount of correlated information that the signal can supply each sample is extremely low so the noise is able to alter the phase reading fairly easily. This is an indicator of the performance limitations of the system: given this hardware one would not be able to get reliable readings out at SNR=0.001 and it is advised not to drop below SNR=0.01 for a 1 million spectra accumulation.

5.7 Summary

This chapter described the hardware that has been used, which is all standard SKA hardware. The DSP signal paths in the FPGA design were then described, including the frequency domain cross-correlation and accumulation path, and the time domain rolling sum filter and impulse detection paths. It also discussed the control and monitoring blocks used to interface with a computer. Next, it was shown how the design has been tested and verified, using both Simulink simulations and a physical hardware setup in the lab. The hardware setup has RF sources for impulses, tones, and uncorrelated noise. As expected the system was able to detect and capture impulses, as well as measure the phase of tones which were well below the noise floor.

[1] As mentioned in Section 5.3.2 the accumulation is done by adding the array of frequency bin complex numbers to a tracking sum array, hence each accumulation is doing addition of vectors.

Figure 5.4: Equipment setup in the lab. Top to bottom: ROACH; four noise sources; RF splitters, combiners and attenuators; RF board with amplifiers and filters; impulse generator; trigger and gate for impulse generator; signal generator for narrow band signals

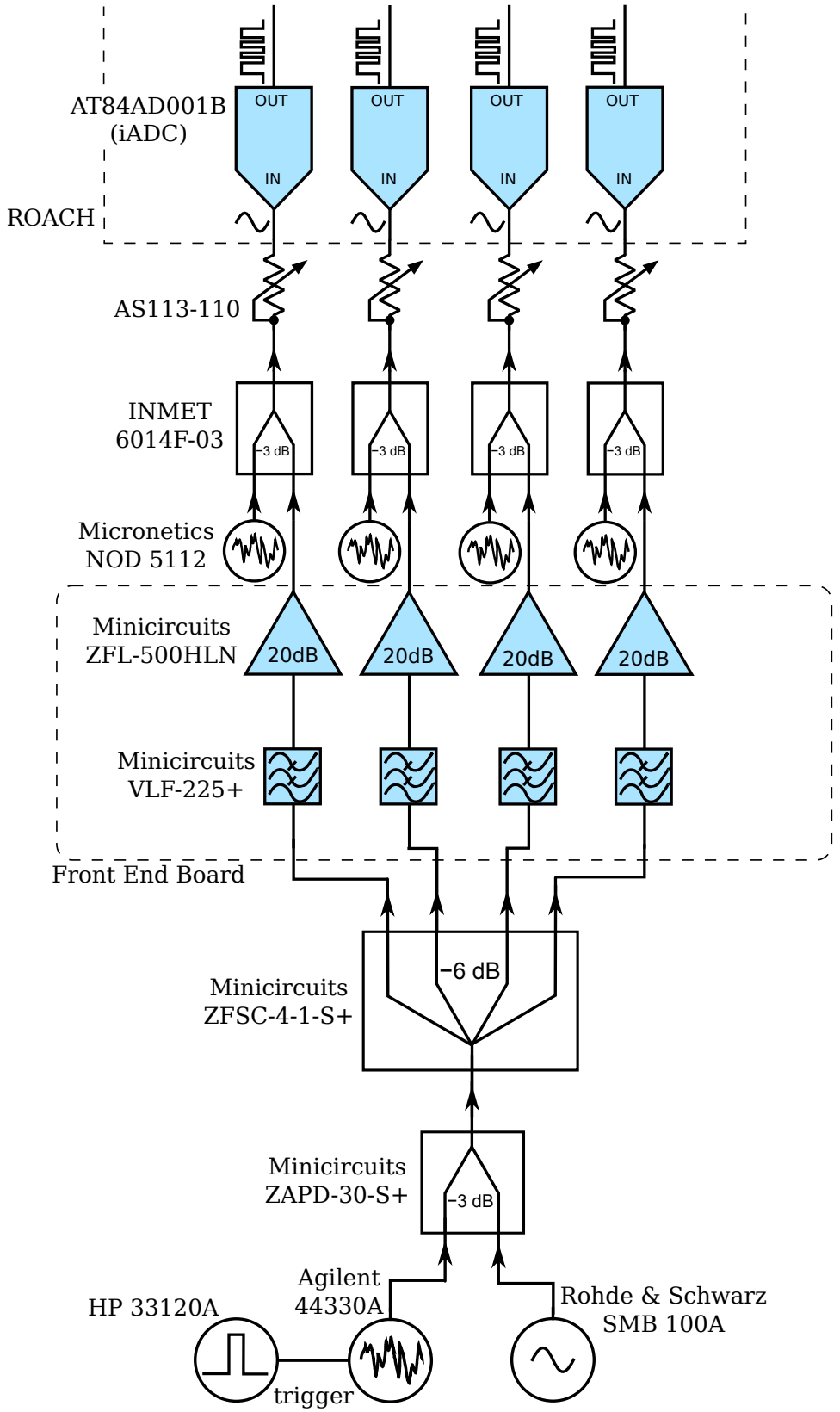


Figure 5.5: Diagram showing how the various components in Figure 5.4 are connected together.

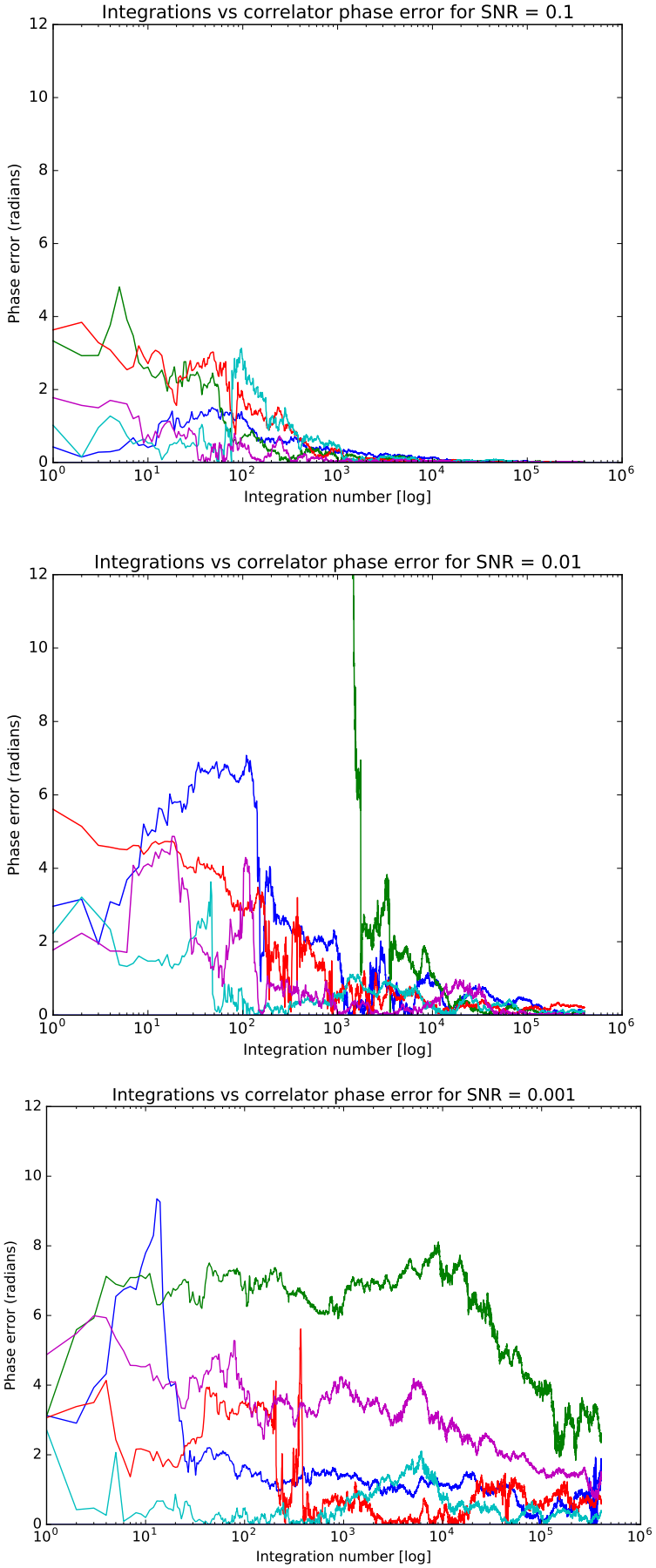


Figure 5.6: Plots for accumulation (aka: integration) length vs output error tests in SNRs of 0.1, 0.01 and 0.001. Each line on the graph is an independent iteration of the test. The colours are arbitrary.

Chapter 6

Software

Software was written in Python to run on a computer which is connected to the ROACH via Ethernet. The software has two broad functions: to interact with and provide an abstraction to the FPGA, and to perform angle of arrival estimation based both on the received signals and on the antenna model. This chapter looks at the structure of the software package and some results indicating it is working as expected.

6.1 Code Structure

The code had to be designed in accordance with good object-orientated methodologies in order to provide a useful, well defined and easily extendible interface to the various software components which need to interface with one another and with the correlator. As such, there was significant emphasis encapsulating logic into classes which mirrored the physical structure of the correlator and the antenna array with regard to modularising key components and writing reusable code.

The main package containing the array modeling, the ROACH interface and the AoA estimation is `DirectionFinder-backend`[1]. The structure of the classes in this package is shown as a unified modeling language (UML) diagram in Figure 6.1. The root class, `DirectionFinder` is composed of an `AntennaArray` and a `Correlator`.

The `AntennaArray` is initialised from an array geometry file that has been produced by the element coordinate calculator discussed in Chapter 3. It is able to return expected baseline phase shifts or time delays for all antenna pairs at any angle. The returned values are used to get the theoretical array response for a given angle at a specific frequency. Due to the modular structure of the `AntennArray` class, it can be used to simulate and provide information about an antenna array with any number of elements in any configuration. This is an example of the general purpose nature of the application which is being developed here.

The `Correlator` provides an interface to processed data from the FPGA. It is able to fetch both frequency domain cross correlations from the snapshot blocks as well as time domain snapshots which it gets by reading raw time domain data and doing in-memory time domain cross correlations. The `Correlator` also contain a `ControlRegister` which abstracts the raw bit or word read/writes necessary to interact with the status and configuration registers. Once again the software has been designed with generality in mind: any number of baselines can be read out, having spectrums with any number of points for arbitrary start/stop frequencies. These are initialisation configuration parameters of the classes which provide the various abstractions.

The software application is launched by a Python executable which takes command line arguments that define the IP address of the ROACH, the path to the configuration file for the ROACH, the frequency domain start/stop frequencies, the frequency or frequency range

[1] `https://github.com/jgowans/directionFinder_backend`

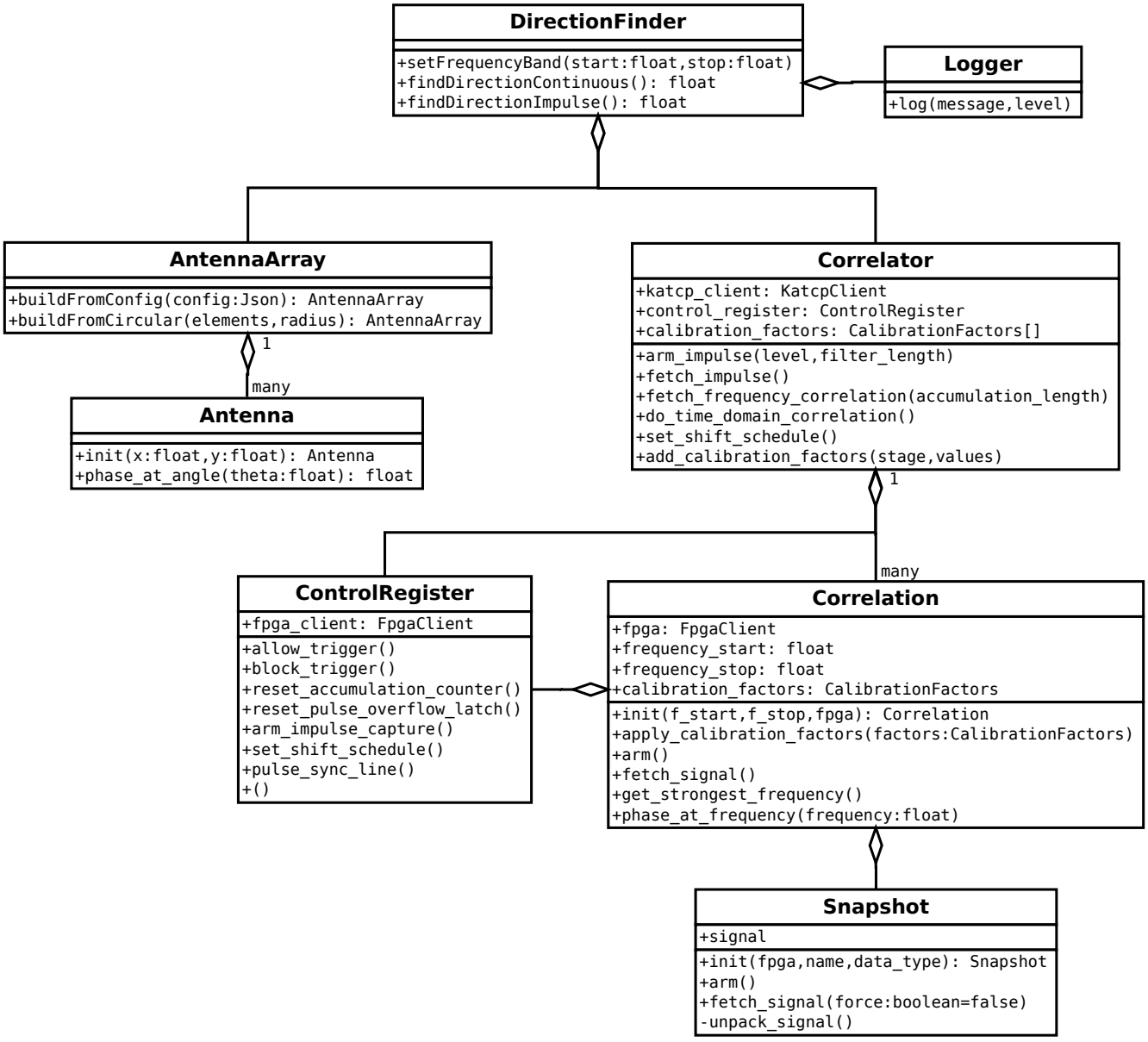


Figure 6.1: UML class diagram of software structure in `DirectionFinder-backend`.

which should be monitored for direction-finding, impulse setpoint, accumulation length, and a comment string to label the observation which is being made. Once running, the software will monitor for impulses and it will continuously DF the narrow band signal. The raw baseline measurements as well as the calculated AoA are both printed to the screen and are written to a log file in a format which can easily be post-processed for plotting. Warnings such as overflow are printed to the screen.

The following sections explain how the software goes about ascertaining angle of arrival for time and frequency domain signals.

6.2 Frequency Domain Direction Finding

The frequency domain DF algorithm assumes that the signal of interest may be below the noise floor. As such, no power detection is done; the direction finding process runs continuously.

When the application starts up, it reads in the array configuration file and constructs an `AntennaArray` object from it. It then samples the antenna array manifold vector at 0.36° intervals, corresponding to 1000 baseline phase shift vectors. These are stored in a hash in memory. The value of 1000 angular samples corresponds to the angular resolution of the system. A higher value gives more resolution but at the cost of computational complexity as more comparisons are required. For this system the time to do the spectra accumulation dominated the processing time so selecting a large value of 1000 had a negligible contribution to execution time, although this resolution could be tuned to a lower value more in line with the accuracy of the rest of the system.

Next, the ROACH is initialised by writing to the accumulation length (`acc_len`) register and pulsing the `sync` line. The accumulation length value is set based on user input, typically it will be around 1 second. Instances of the `Snapshot` class are then constructed, one instance for each baseline cross-correlation BRAM snapshot on the FPGA. These snapshot blocks are all armed and, once armed, the snapshot trigger is un-gated. The system sits in a tight loop watching the accumulation counter. When the accumulation counter ticks, the snapshot trigger is gated and all snapshot values read out. Calibration factors (discussed more later) are then applied to the signals. The strongest signal in the frequency interval of interest is found and the phase shifts, as seen by each baseline correlation, are fetched and stored in a map of baseline name (eg: "1x3") to phase shift. Finally, the `DirectionFinder` iterates through all 1000 simulated angles, comparing the simulated baseline phase shifts to the observed baseline phase shifts and it selects the angle whose simulated values most closely match the observed values. In pseudo code:

```
AntennaArray antenna_array = new AntennaArray(config_file)
Correlator correlator = new Correlator(...)

while True:
  Correlation correlation = correlator.get_correlation() # blocking
  Float frequency = correlation.strongest_signal_in_range(f_start, f_stop)
  Map<Baseline, Float> observation = correlation.baseline_phase_shifts_at(frequency)

  Float best_angle = 0;
  Float lowest_difference = INT_MAX;

  for angle in antenna_array.sampled_angles
    Float difference = antenna_array.manifold_at(frequency, angle).compare(observation)
    if difference < lowest_difference:
```

```
15        best_angle = angle;
16        lowest_difference = difference
17    write_result(best_angle)
```

The `compare` method computes the difference between vectors by the RMS of the difference of their terms. When the difference d is computed for two vectors $\vec{a}$ and $\vec{b}$ with k terms, it uses arctan to account for the fact that phase wraps around 2π as mentioned previously in Equation (3.2.2):

$$d = \left[\sum_{i=0}^{k} \text{atan2}(\sin(a_i - b_i), \cos(a_i - b_i))^2\right]^{1/2} \tag{6.2.1}$$

6.3 Time Domain Direction Finding

As mentioned in Chapter 3, the algorithm behind time domain DF is remarkably similar to frequency domain DF. The main difference is that the comparison method uses baseline time delays rather than phase shifts.

One important difference is that where the frequency domain cross-correlations are calculated using DSP on the FPGA, the time domain cross-correlations need to be calculated on the computer. This is not a hard requirement, but the implementation of the FPGA design did not cater for time domain cross-correlations which can be a lot trickier to do in hardware since the length of the pulse is dynamic.

As mentioned in Equation (3.2.3), the time domain cross-correlation is implemented, as defined, by multiplying each point of one antenna's signal $\vec{a}$ by the corresponding points of a shifted version of another antenna's signal $\vec{b}$ for some shift k, producing the correlation $c_{ab}(k)$:

$$c_{ab}(k) = \sum_{n=0}^{N-1} a_n b_{n+k} \tag{6.3.1}$$

The range of values for k will be picked to span an interval a bit larger than the propagation time of a signal over the whole array. For the array being used here this will correspond to $k \in [-100, 100]$, $k \in \mathbb{Z}$[1]. To make this possible, $\vec{b}$ is zero-padded with 100 zeros on each end[2]. Hence, in Equation (6.3.1) the value of N is the number of points in the shorter, non-padded vector $\vec{a}$.

Once an array of c values for various k values has been generated, the c array is upsampled using the Fourier method discussed in Chapter 3. Upsampling of large signals can consume lots of CPU cycles and slow down the DF system. However by upsampling after cross-correlation we take advantage of the property that upsampling the raw time domain signals has the same effect as upsampling the output of the time domain cross-correlation. While impulses of even a few microseconds will be tens of thousands of ADC samples (which can be expensive to upsample), the time domain cross-correlation output will be only 200 samples as it's limited to the correlation interval.

[1] This is about 10 times more than is necessary, but seeing as the correlation output will be upsampled the signal should not be short, or the frequency resolution available in the upsampling process will be limited, leading to an inaccurate output.

[2] Zero padding one of the samples would increase the bias for the cross correlation at zero shift. As the shift goes further away from zero, more data points will be multiplied by the padding zeroes, decreasing the summed cross correlation value. It may be preferable to rather truncate one of the samples instead of zero padding

Once the upsampling is complete, calibration factors are applied to c, then the maximum value of the upsampled calibrated c is found and the corresponding time shift noted. This is done for the snapshot of each combination of pairs of antennas, a and b. The result is a map of baselines to observed time differences seen by that baseline.

The same algorithm as the frequency domain DF is then used to find the angle where the simulated baseline time delays most closely match the observed ones. That best-matching angle is then chosen as the angle of arrival.

6.4 Calibration

Although efforts have been made to keep the RF chains for each antenna's time/phase delay matched, it hasn't always been possible to do so. Small errors in measured phase/time shifts can throw off the measurement results. Examples of where mismatches can occur are:

- Antenna cables: the antennas which were purchased for this project do not have the same length cable length coming out of them. This means that although the antennas may receive a signal with a certain phase difference, a different phase difference is created by the cables as the propagation delay is different.
- RF front end: the RF components and the connecting cables from the antennas to the ADCs may not be matched. Measurements in earlier chapters have shown that the difference in this setup is low, but it is worth catering for this mismatch anyway for future systems which may have RF front ends with more significant mismatches.
- ADC cores: The ADCs should be clocked exactly in phase, but mismatches in clock distribution or internal ADC characteristics cause the actual samples not to be exactly in phase.

Following is a brief description of how these mismatches were measured and calibrated out in software.

6.4.1 Antenna cable lengths

It is difficult to empirically measure the differences in signal propagation delay through the antennas as delayed by their cables. To cater for cable length mismatches in a generic way, the direction finding system takes as input a file specifying the measured cable lengths coming out each of the antennas as well as specifying the velocity factors of the cable. The velocity factor is a property of the specific cable type in use. In this case, the RG214 50 Ω cables used here have velocity factor of 0.6 as per their data sheet. Table 6.1 shows the measured length of cables from each antenna. The software then calculates the resultant time delay t as follows:

$$t = \frac{l}{\mathrm{vf} \times c} \tag{6.4.1}$$

where l is the length of the cable, vf is the velocity factor and c the speed of light. Phase shift at a particular frequency can easily be deduced from the time delay.

To verify the implementation of this technique in a laboratory test, all ADC channels were connected to the same noise source via equal-length cables except channel 0 which had a longer cable. The difference in propagation delay between the longer and shorter cables was checked on a network analyser. Figure 6.2 indicates that cable length response was flat and that the

Antenna Number	Cable Length (m)
0	0.557
1	0.566
2	0.510
3	0.590

Table 6.1: Lengths of cables coming out of antennas

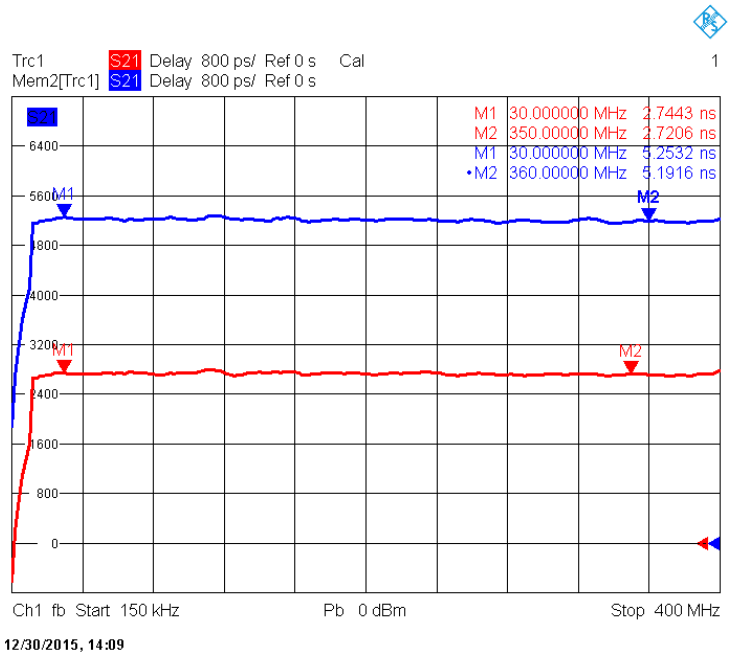


Figure 6.2: Two different length cables on VNA doing time measurement showing propagation delay difference of 2.49 ns

actual difference between the propagation delays is 2.49 ns. Next, the DF system was used to do a time domain cross correlation before and after calibration factors were applied. Table 6.2 indicates that after cable length calibration factors had been applied, the system's measurements accurately reflected actual time differences and it correctly identified that the signals were arriving in phase.

6.4.2 RF front end mismatches

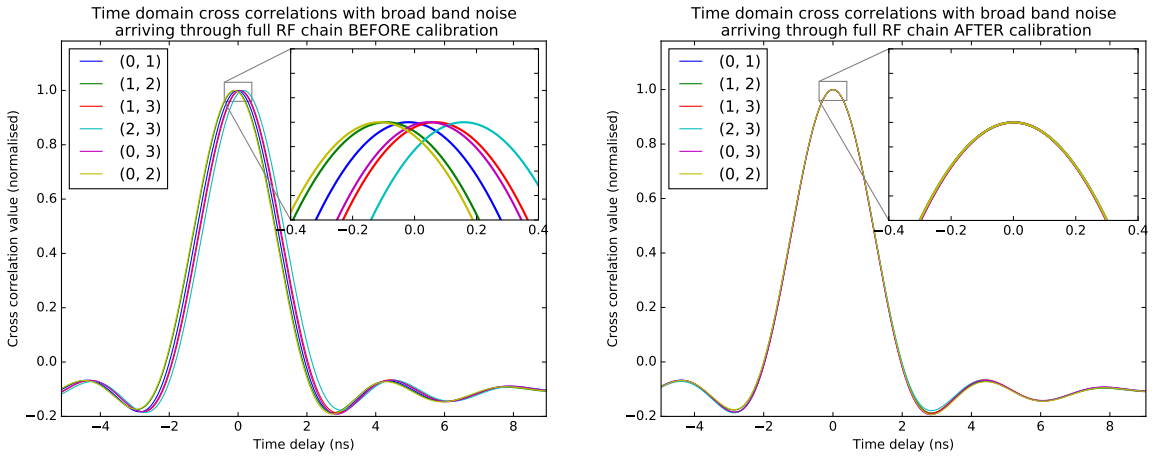
The next class of time/phase mismatches is from the rest of the RF front end: filters, amplifiers, connecting cables and ADCs. This is different to the antenna cables because the mismatches here can be measured by injecting signals into the system. This is how mismatches through the remainder of the RF chain were measured:

- For the time domain, broadband noise was injected into the start of the RF chain, raw data captured from the ADC and cross-correlated to find time differences introduced by each input path. The offset of the correlation peak from 0 is the time delay error for that baseline.

- For the frequency domain, iteratively through each frequency channel, a tone centered in

Baseline	Uncalibrated (ns)	Calibrated (ns)
0x1	2.492	0.002
0x2	2.489	-0.002
0x3	2.495	0.002
1x2	-0.001	-0.002
1x3	0.001	0.000
2x3	0.005	0.005

Table 6.2: Time domain correlation peak position for each baseline before and after cable length calibration factors were applied. Channel 0 had a longer cable length which was the network analyser showed was 2.49 ns and the correlator produced the same result. ADC sample period: 1.25 ns. Upsampled correlation step size: 1 ps



(a) Before calibration

(b) After calibration

Figure 6.3: Time domain cross correlation plots of each baseline for broadband noise injected in phase before and after calibration.

the frequency channel was injected from a signal generator exactly in phase to each signal path. The phase differences as seen by the output of the accumulator on the ROACH were recorded. These phase differences were the error for the baseline in that frequency channel.

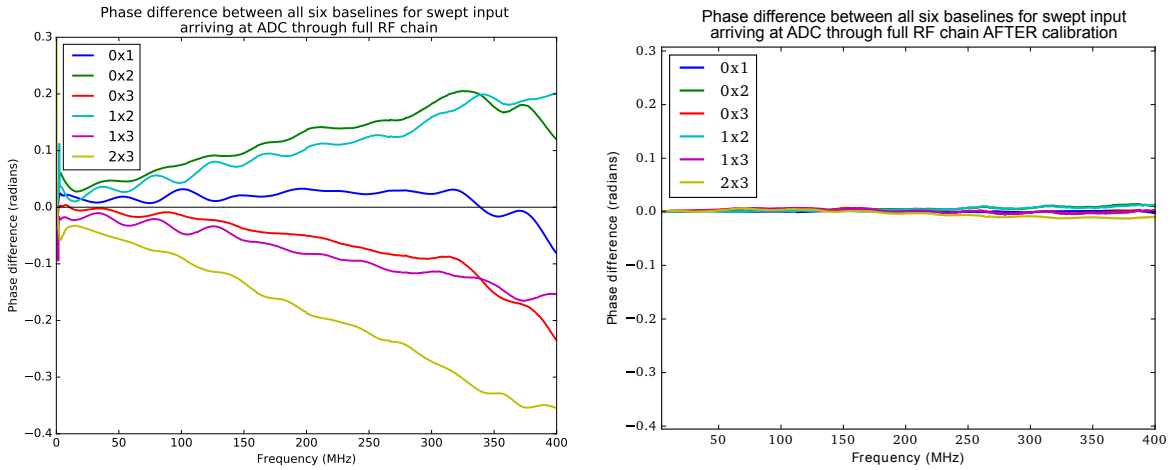
These two experiments produced two calibration files, one for broadband time delay and one for phase shift at each specific channel. These two files are then used by direction finding code to subtract the time or phase offset introduced by the RF chain.

Figure 6.4 shows the time domain correlation peaks before and after calibration factors are applied, indicating that after calibration the peaks correctly align when broadband noise is injected in phase.

Figure 6.4 shows how phase offsets due to differing and non-linear phase performance across frequency is significantly reduced by applying frequency domain calibration factors.

6.4.3 Non-ideal antenna elements

A further source of error would be if the antenna array elements do not have a uniform phase response in all direction. In it's full form the array manifold vector from Equation (2.1.4) does



(a) Before calibration

(b) After calibration

Figure 6.4: Phase shifts through full RF front end of signal being injected exactly in phase before and after calibration.

make provision for non-ideal antenna elements via the $\vec{g}(\theta)$ term. For simplicity in this research we have not attempted to quantify the array element response for the folded dipoles, assuming that they have an identical phase response from all angles, and that each element has the same response as each other element. Further research could devise a calibration technique which quantifies the $\vec{g}(\theta)$ term.

Furthermore, amplitude has been ignored; the interferometer only considers phase. Further research could use the observed amplitude of a calibration signal from various directions to quantify element shadowing, and bias away from shadowed elements.

6.5 Summary

This chapter has discussed the structure of the direction finding software package, as well as how the angle of arrival estimation calculations were implemented. It started by showing how the class structure mirrors that of the physical system, providing abstractions around antenna array models, FPGA interactions and FPGA data outputs. Emphasis was placed on making the algorithms reconfigurable for any number of inputs or frequency range.

The frequency domain and time domain angle of arrival estimations were both done in a similar way, by correlating the observed baseline time/phase differences with the theoretical time/phase differences for each possible angle of arrival and finding the one which most closely matches.

Finally, the calibration section discussed potential sources of error in the analogue components of the system. It showed how those sources of error were measured and made available to the direction finding code so that they could be subtracted out to provide a more accurate calibrated measurement of the real signals.

Chapter 7

Field Trials

In order to test the operation of the complete system it was necessary to conduct field trials where the DF system was taken from the lab, made into a portable self-contained unit, and tested on real signals in the field. There were two field trials conducted: the first was done at the MeerKAT site early in the project when the system could capture from two channels in the time domain only. The second was at the end of the project with the full four channel time and frequency domain system. The objective of the first field trial was to test the data capturing ability of the ROACH as well as to get samples of what real impulsive RFI looks like. The objective of the second field trial was to test the performance and accuracy of the complete system. This chapter looks briefly at the results of the first field trial and the implications of the types of impulsive RFI that were captured. It then looks at the full system that was tested at the end of the project, tracking both weak narrow band as well as strong impulsive signals.

7.1 First trial: Sample Impulsive RFI

A few months after the start of the project, the system consisted of a simple two-element antenna array and a simple ROACH design. The ROACH streamed input from a single two-channel ADC, did threshold detection and stored a small snapshot of raw time domain ADC samples to FPGA Block RAM (BRAM). Computer code was at this point only able to pull raw samples, save them, and do in memory time domain cross-correlation.

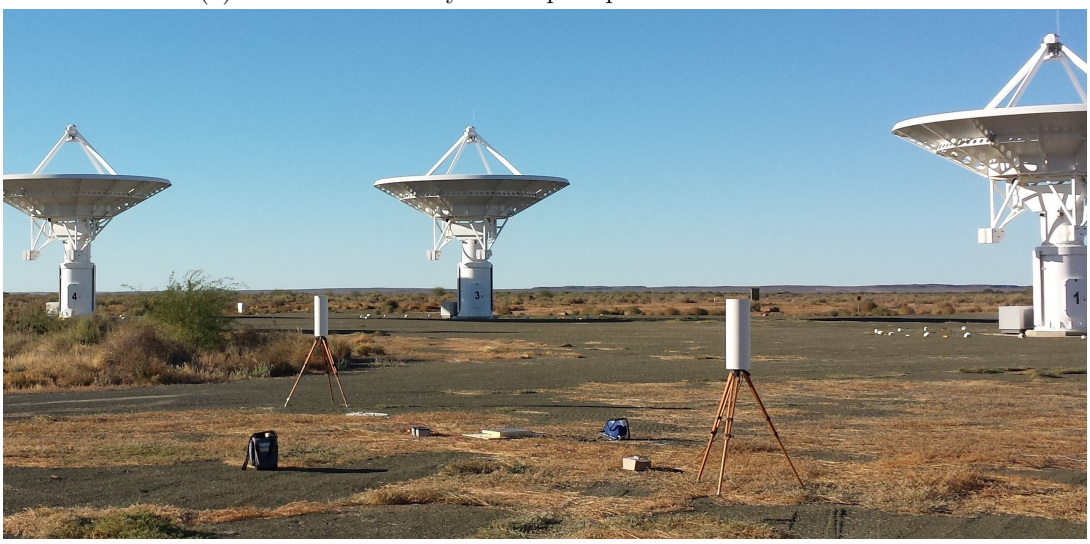
This system was taken to the SKA's MeerKAT site in the Karoo with the objective of getting some snapshots of RFI to see what characteristics the impulses had as well as to attempt impulsive RFI hunting. Two types of antenna were used: printed log-periodic dipole array (LPDA) antennas and omni-directional conical antennas. The two antenna types and the setup and operation are shown in Figure 7.1. Simple power detection was done by FPGA firmware which was designed to detect when the received signal amplitude went above a threshold,[1] and then to capturing the pulse to DRAM. Once captured, detection was paused and the computer notified of the pulse. The computer would then read out the samples and re-enable detection when finished. Examples of captured signals of unknown origin are shown in Figure 7.2 and signals of known origin in Figure 7.3. From this, we can see that there are almost no characteristics that all of the impulsive signals had in common. They varied in terms of all key pulse classification parameters: pulse length, shape, frequency content and repetition pattern. This fact contributed to the decision to do power detection for the impulsive signals. Since it would not be practical to attempt any form of matched filtering when hunting pulses that are of unknown origin, all that can be done is to look for a burst of energy in the time domain.

Direction finding from the raw time domain signals was attempted on the computer by doing the time domain cross-correlation process described earlier. However, these attempts to hunt for

[1] Threshold was set by sampling the ambient signal amplitude for a few minutes and defining the threshold two standard deviations above ambient.

RFI using the two-element system did not prove successful due to the 180° ambiguity inherent in a two-element design and due to calibration techniques not having yet been developed at that early stage of the project. Also, at that stage the system needed to be plugged in to run and hence was constrained to be close to a power source.

(a) Two element array of compact printed LPDA antennas.



(b) Two element array of EM-6916 Omni-directional conical antennas.

(c) Laptop linked via Ethernet to the ROACH, processing collected data in real time.

Figure 7.1: Setup for first field tests at the SKA's MeerKAT site in the Karoo. Early 2014.

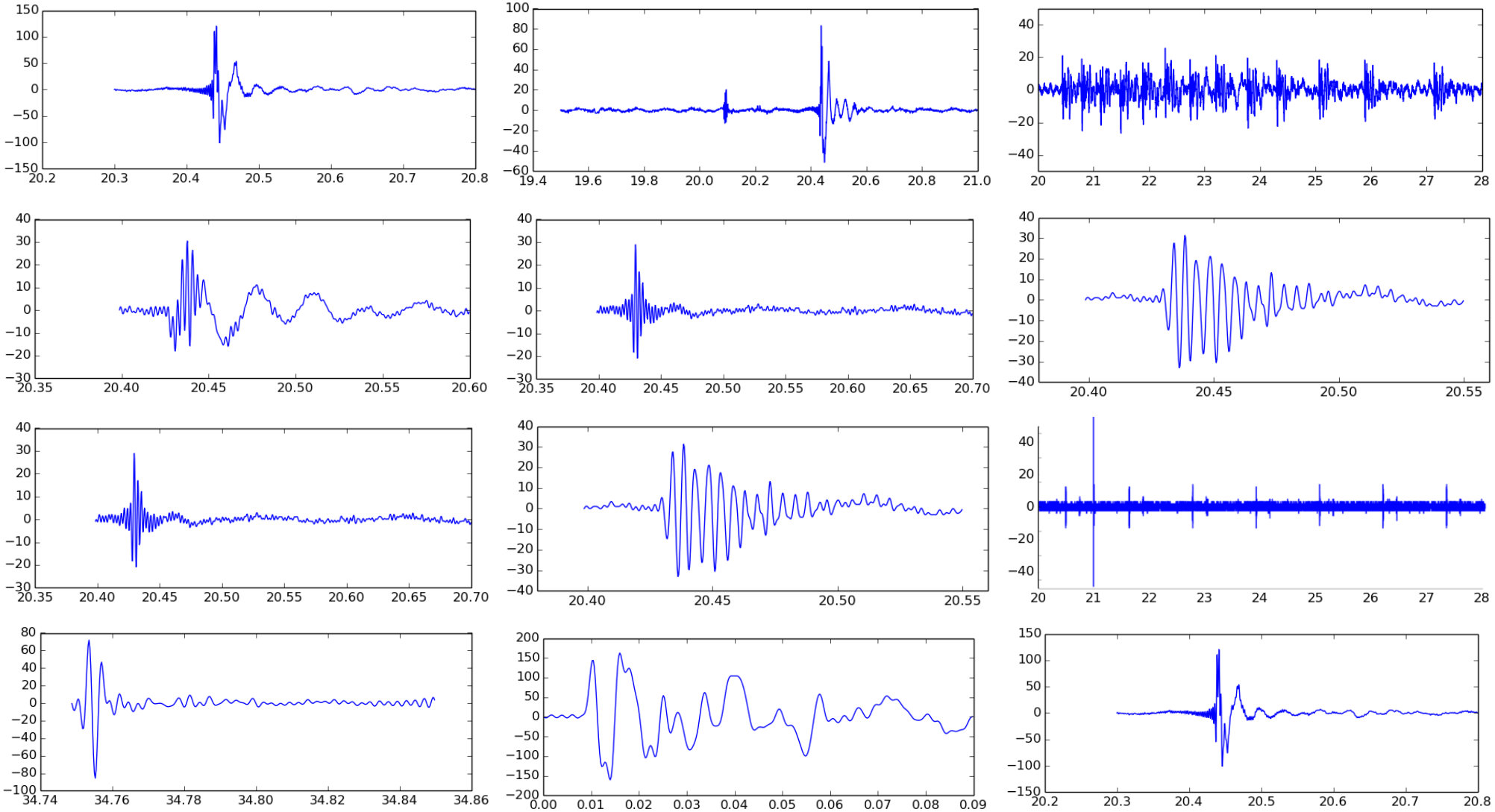
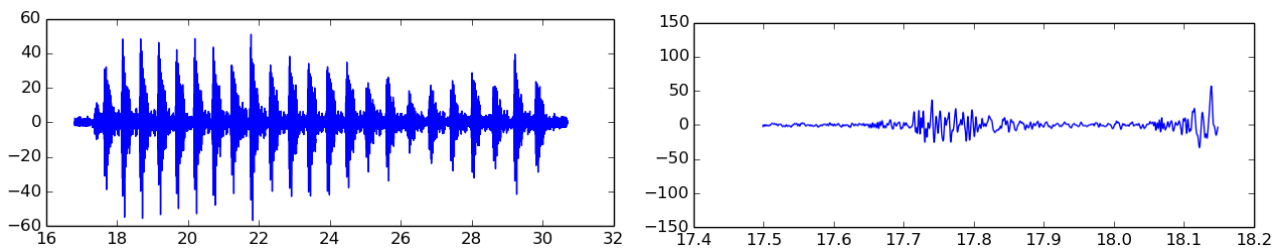
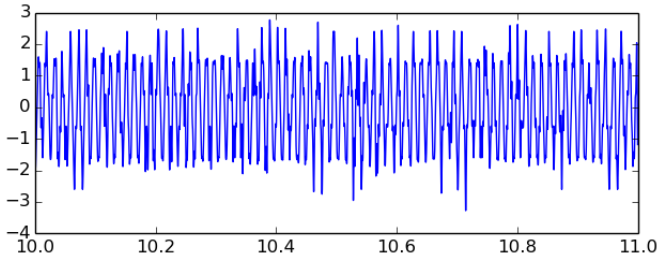


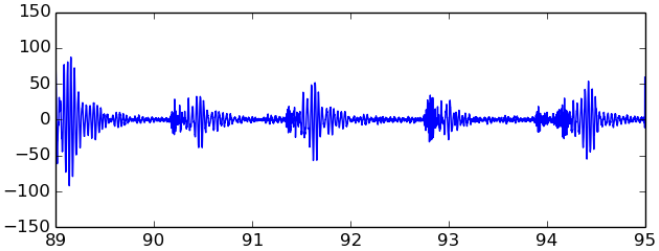
Figure 7.2: Selection of impulses collected at MeerKAT site with UNKNOWN origin. Construction was taking place on site at the time. Plots are time domain. X-axis is time in microseconds. Y-axis is ADC output number of 8-bit ADC



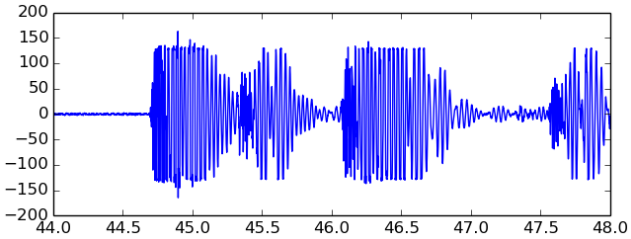
(a) Bakkie starting up

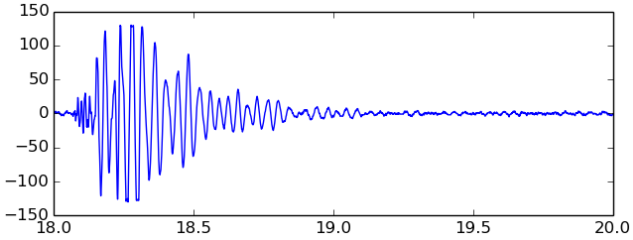


(b) Keying the 70 MHz two way radio

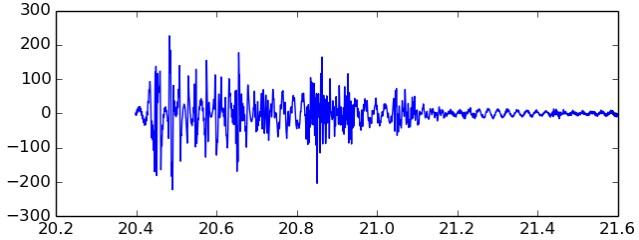


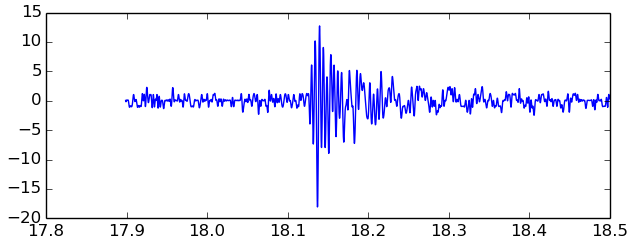
(c) Unplugging laptop charger

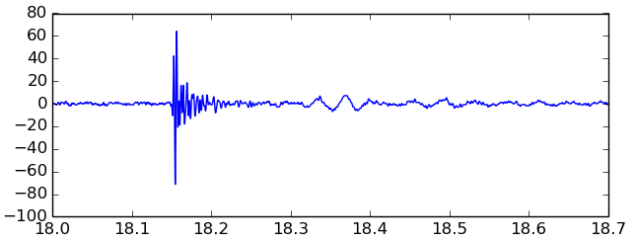




(d) Plugging in laptop charger







(e) Refrigeration compressor turning on

Figure 7.3: Selection of impulses collected at MeerKAT site with KNOWN origin. Plots are time domain. X-axis is time in microseconds. Y-axis is ADC output number of 8-bit ADC

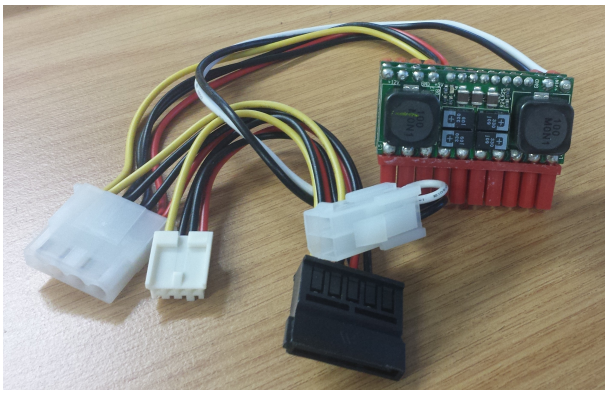


Figure 7.4: Mini-Box PicoPSU which plugs into the ROACH motherboard allowing the ROACH to run directly from a battery.

7.2 Second trial: Final DF system

Once the full DF system described in the previous chapters had been completed, the second set of field trials could take place. This was done at the University of Cape Town (UCT) sports field where we could more freely generate RFI. The tests involved setting the system up in the center of the field and walking various RFI sources around the field in a circle and measuring how well they were tracked. This section first explains how the system was made fully portable, then it proceeds to show the setup and discuss how the measurements were taken, and finally it looks at the results for each signal source that was trialed.

7.2.1 Portability

It was necessary to power the ROACH from a battery in order to allow it to be portable for taking out into the open field. Initially the plan was to power it from an inverter running off of a battery, however in order to reduce switching noise emissions, a battery powered ATX power supply was used instead. The ATX PSU is a Mini-Box PicoPSU-80-WI-32V which runs directly from a 12 V battery. It can output 80 W which is more than enough to run the ROACH; testing in the lab showed that the ROACH pulled 3.1 A at 12 V which is 37 W. To connect it, the traditional mains-powered ATX power supply was disconnected from the motherboard and this module plugged into the motherboard. This is shown in Figure 7.4.

The battery used to power the ROACH and laptop in the field was a ROYAL 1150K 105 A h deep cycle calcium battery. As the ROACH draws 3.1 A at 12 V, meaning a running time of $\frac{105}{3.1} = 34$ h. To maintain battery lifespan it is advised to not run down below 30 % of capacity. Even so, that's more than 20 h of runtime in the field, which is more than enough.

7.2.2 Setup and Test Procedure

These field trials involved having a person walk various signal sources around the DF system. Calculations and simulations were done to model interference due to ground reflection multipath and error due to non-flat wave fronts. These simulations showed that in order to minimize both of these possible error sources the radius which should be used when walking around the DF station should be 35 m and the height above ground of both the DF antennas and the transmitter should be 2 m. The simulations which produced these dimensions are detailed in Appendix D.

Figure 7.5: Setup on UCT sports field focused on receiver and front end. From top to bottom: four element antenna array, SMA cables into RF front end board, cables into ROACH (shiny box under the laptop). ROACH running directly off of 12 V battery, with the battery in the red and black box (right). Blue inverter for charging laptop between tests. Grey box on the left is R&S spectrum analyser for checking the power level coming out of the LNAs before connecting to ROACH ADCs.

Figure 7.6: Photo on UCT sports field (left) showing how the transmitter was carried around, with both it and the receiving antennas at a height of approximately 2 m. The PCB in (right) is the Valon synthesiser. The shield of the USB cable going down and the wire coming out of the SMA port pointing up act as a quarter-wave dipole.

The antennas and LNAs were attached to a tripod and set up in the center of the field. Initial power measurements were done using a spectrum analyser to measure environmental noise and to set attenuators appropriately. The ROACH and laptop were set up under the tripod, with a shielded Ethernet cable connecting them as shown in Figure 7.5. Various transmitters (each has its own subsection following this) were attached to a wood pole so that they could easily be carried when walking around the field. One of these transmitters is shown in Figure 7.6. A person carried the transmitter, walking slowly while keeping a GPS logger on him. The GPS logger was a mobile phone which took a timestamped GPS reading every 1 second and wrote it to a CSV file on the phone. The plot of the route walked over the course of the measurements is shown in Figure 7.7. Before doing the field trial for each transmitter, the spectrum analyser was used to figure out the transmit frequency of each transmitter individually. In each instance the DF system was configured to lock on to the strongest observed frequency in a small range (a few channels) around the peak of the particular transmitter. The field trial for the transmitter could then be carried out as described above. Once complete, the readings from the GPS logger were converted to angle measurements by running each time/coordinate reading along with the fixed coordinates of the DF station through a Python library that converted the coordinate pair into an angle. Those time/angle readings from the GPS logger were then plotted on top of the corresponding time/angle reading from the DF system to compare how well the DF system tracked the transmitter. The plots of the track of each transmitter, as well as a short discussion about each plot now follow.

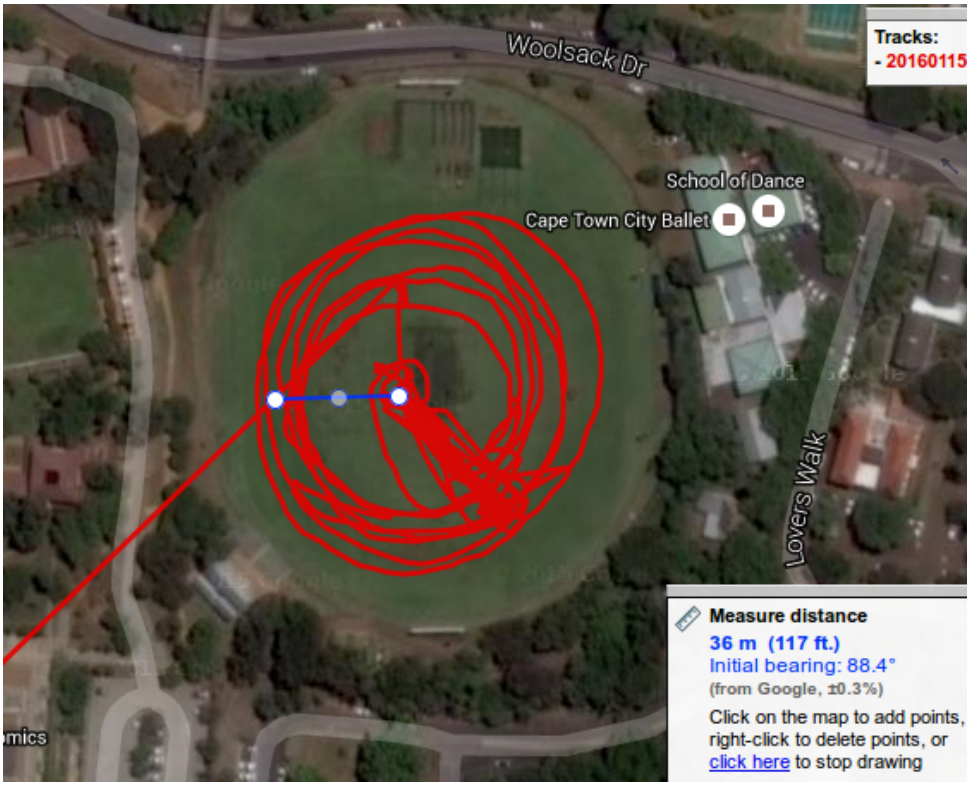


Figure 7.7: Route of the person walking with the transmitters during the field trials, according to GPS logger. The curve is a smoothed collection of timestamped lat/long coordinates

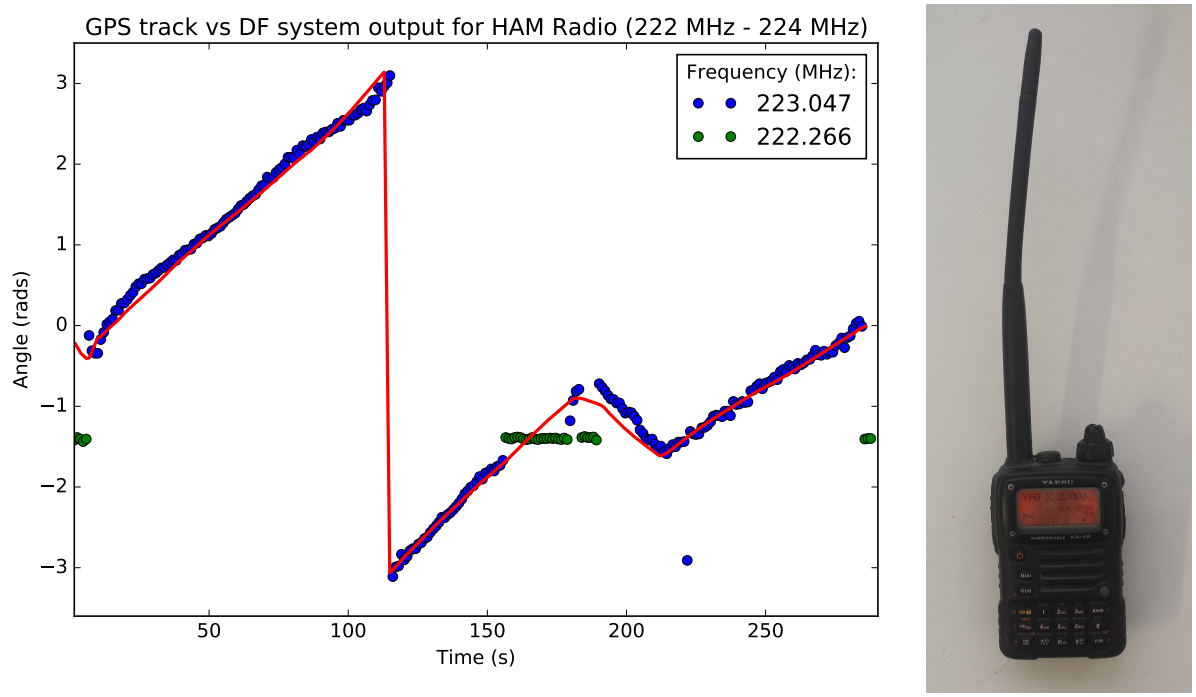


Figure 7.8: DF results for HAM radio tracking on UCTT sports field. Red: GPS track of real angle. Blue: track of HAM radio. Green line in center: track of adjacent signal when DF system lost the HAM radio.

7.2.3 HAM radio

The first source which was trialed was a portable HAM radio, a Yaesu VX-7R, transmitting 17 dBm at 223.0 MHz. The track for this source is shown in Figure 7.8 indicating that in general the signal was tracked very well. Midway though the trials the person carrying the HAM radio accidentally released the transmit button, causing the DF system to lock on to a near-by strong signal at 222.2 MHz. This loss of transmitter was noticed immediately by the DF station operator as the results were being displayed real-time on screen. As such the operator could tell the person carrying the transmitter to push the transmit button and re-walk the last few tens of meters. On the plot this manifests as a blip in the middle of the observation.

7.2.4 Raspberry Pi

As the HAM radio transmitted a strong signal, a different device was necessary to contrast performance under weak signal conditions. For this, a Raspberry Pi had an application called `fm-transmitter` installed on it allowing the Pi to broadcast an FM radio station by toggling a GPIO pin. The FM carrier frequency can be set well above the usual FM band, and for this test the carrier was set to 241.3 MHz. The modulating source was a silent sound file, thus produce a continuous RF tone. To create a quarter wavelength antenna with vertical polarisation, a 0.3 m length of wire was connected to the Pi's toggling pin, with the shielding of the USB power cable used as the other half of the dipole. With the GPIO pin toggling at such a high frequency, only about 100 mV peak-to-peak voltage was present. This voltage driven into the approximately 75 Ω dipole means a effective isotropic radiated power (EIRP) of −18 dBm[1] and

[1] 100 mV peak-to-peak is 35 mV RMS hence power is $P = v^2/r = 0.035^2/75 = 16\text{e}{-6}$ w which is −18 dBm

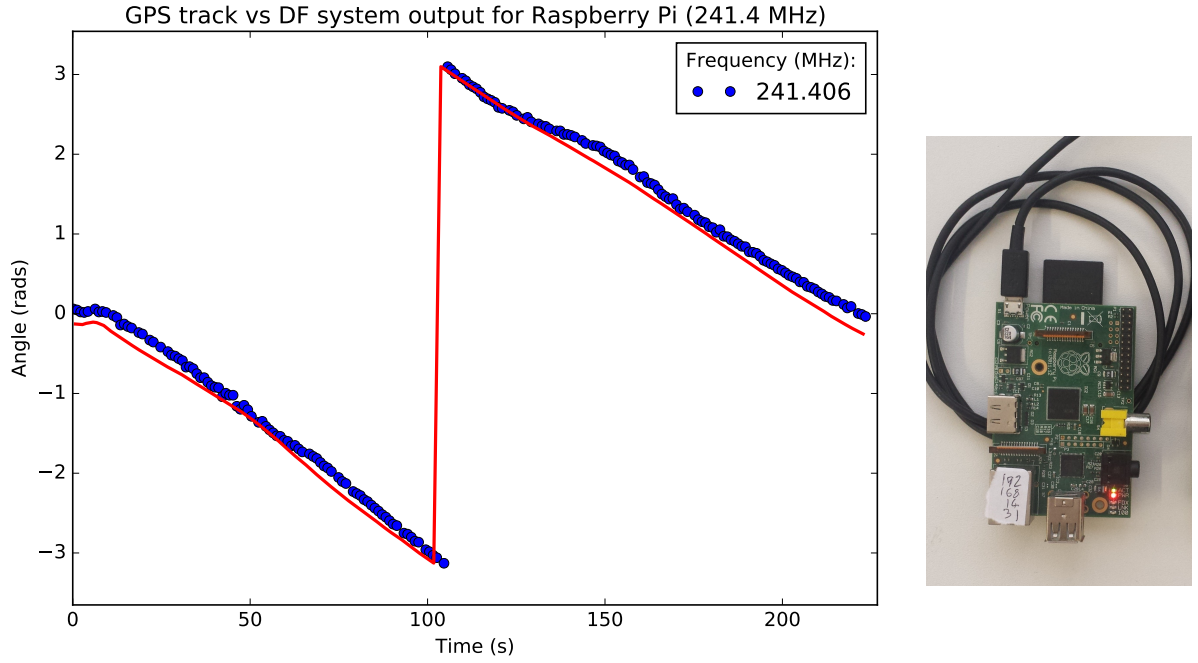


Figure 7.9: DF results for Raspberry Pi tracking on UCT sports field. Red: GPS track of real angle. Blue: DF system's track of Raspberry Pi.

received [1] power of −69 dBm. Despite this much lower signal strength, Figure 7.9 shows that the DF system was still able to track the Pi remarkably well, with an error only marginally higher than the HAM radio track.

7.2.5 Valon Synthesiser

The final narrow-band transmitter which was tracked was a Valon synthesiser, the same device which is used internally in the ROACH to generate a clock signal which clocks the FPGA and ADCs at a precise and programmable frequency. The Valon was configured to generate a frequency higher than the HAM or Pi: 257.0 MHz. Figure 7.10 shows that the result is similar to that of the HAM radio and Pi in that it was tracked very well. One interesting thing to note was that this frequency was right in the band of another fixed transmitter and while the device was being walked around the field, when it got close to being in line with the fixed transmitter its angle was "pulled" to point to the fixed transmitter. This illustrates one of the challenges of having multiple transmitters in or close to each other's frequency channel: interference. The results of the tracking and the spectrum in Figure 7.10 indicate that there is another high power transmitter in the same band. This is quite possibly a TV or radio station that either has harmonics in the band, or has been aliased down. Although efforts have been taken to block aliased signals before the LNA, it is still possible that powerful transmitters can couple in. Separation of multiple signal on the same frequency requires different algorithms and more antennas and is a topic reserved for future work.

[1] $P_r = P_t G_t G_r \left(\frac{\lambda}{4\pi R}\right)^2 = 16\,\mu\text{W} \times 1 \times 1 \left(\frac{1.2\,\text{m}}{4\pi \times 35\,\text{m}}\right)^2 = 119\,\text{pW} = -69\,\text{dBm}$

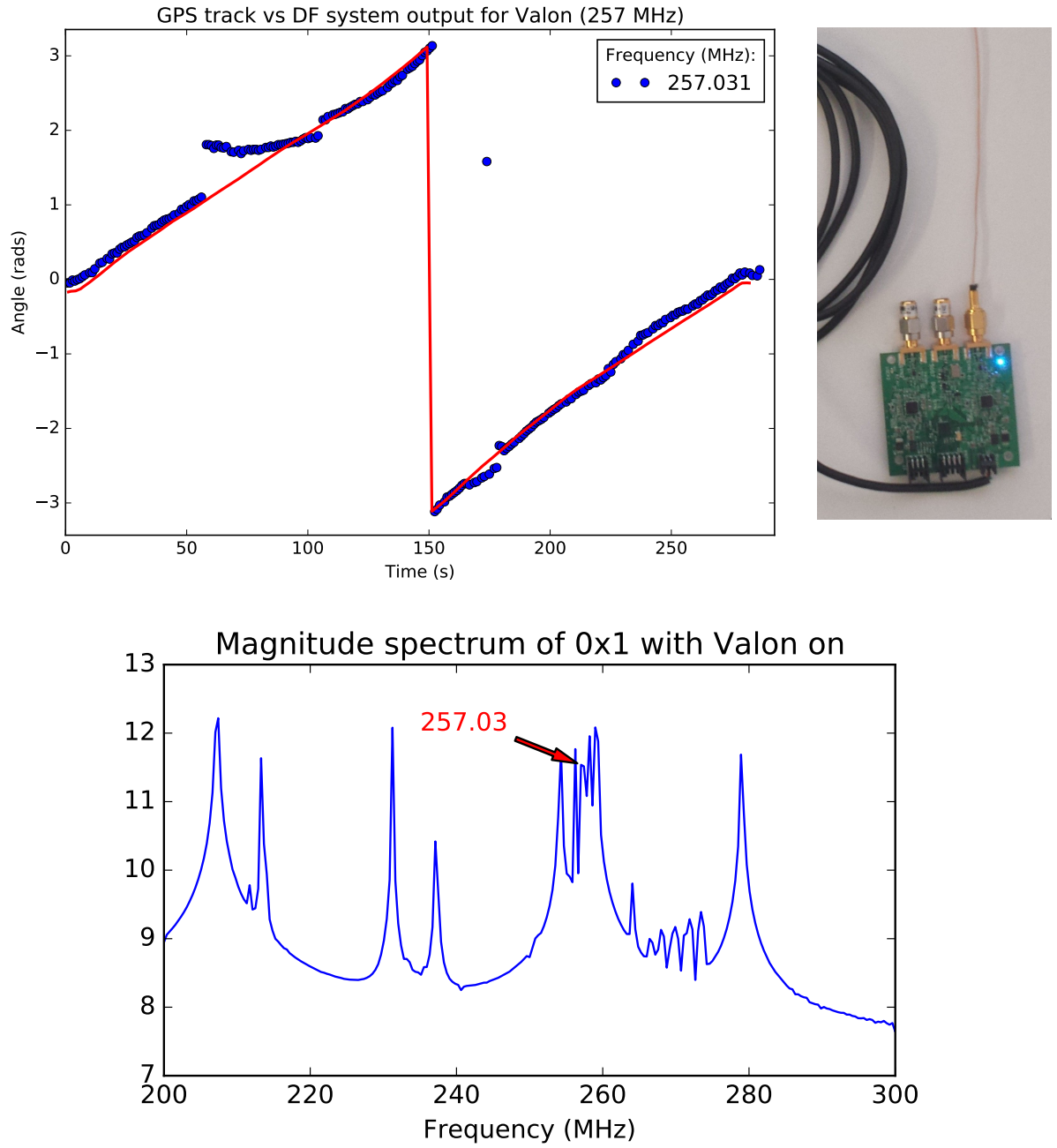


Figure 7.10: Top: DF results for Valon tracking on UCT sports field. Red: GPS track of real angle. Blue: DF system's output. Bottom: cross-correlation spectrum showing how the 257 MHz Valon Tx frequency was right on top of some other broad transmitter or set of transmitters which caused AoA distortion at some angles. Y-axis power is arbitrary log units.

7.2.6 Spark Generator

In order to test the ability of the DF system to track an impulsive signal, a spark lighter for lighting gas fires was used as an impulse generator. It has a small 2 mm spark gap which is triggered when a button is pressed. The spark generator was walked around the field with the button being pressed about once every 2 seconds, which is approximately how long it takes the computer to read out and process a pulse. The DF system was constantly monitoring the time domain signal's power with a rolling sum window (as described earlier) and would trigger a time-domain snapshot of all channels when the energy went above a threshold. The raw time domain data was then read out run through the DF algorithms. The results are plotted in Figure 7.11. These initial results show a poor tracking of the impulse source. This is partially expected due to the presence of very strong correlated signals such as radio and TV transmitters. Although these are mostly out of the band of the antennas, they are still strong enough to drown out weak signals like the spark generator for time domain correlation.

In an attempt to mitigate the impact of these strong correlated signals two forms of post-process filtering were done in software. Firstly, a bandpass filter was applied to select only signals which are in the band of the antennas. The spectra before and after filtering are shown in Figure 7.12. It can be seen from this figure that after the bandpass filtering the strong signals out of the band of the antenna have been significantly reduced in amplitude. Secondly, time domain threshold filtering was done to notch out the part of the time domain signal when it went below a threshold. These two types of filtering were done in an attempt to simulate a more RF-quiet environment. Time domain plots compare the signal before and after filtering in Figure 7.13. It can be seen how the ambient signals between the main pulse chain have been removed.It can be seen how the ambient signals between the main pulse chain have been removed. This two-step post-processing filtering was done to all of the captured time domain data. The data was then run through the DF algorithms once again and new plots generated. This final result for this post-processed data is shown in Figure 7.14 where it is clear that the track is much better, although still not perfect. The plot shows a significantly improved track to the raw data shown in Figure 7.11. The track is good at the start and end of the test. Towards the middle not many detections occurred but it is not known what caused the missing detections. It is possible that the orientation of the transmitter changed, or some form of destructive interference, or the power detection element was shadowed. It must be stressed that this time domain DF system was operating in a hostile environment. The system is designed to operate in a radio astronomy reserve which is generally radio quiet in the bands of interest and hence would not have its cross-correlations skewed by ambient signals.

7.3 Direction Finding Accuracy

In order to quantify the accuracy of the direction finding during these field trials, the error between the DF angle vs and GPS angle was computed for each DF output point. Next, the mean and standard deviation of this array of errors errors was calculated. The results are shown in Table 7.1. Note that there is a column for "Detected points" and "Usable points." This is because not all points recorded on the plot were valid: some points were distorted by other transmitters. Any points with an error lower than 20° for the narrow band transmitters or 35° for the impulse sorce were considered usable; the others were dropped from the error calculation. This gives us both an indication of how often a valid point is generated, and how accurate that point is when it is valid. Some notes about the results follow.

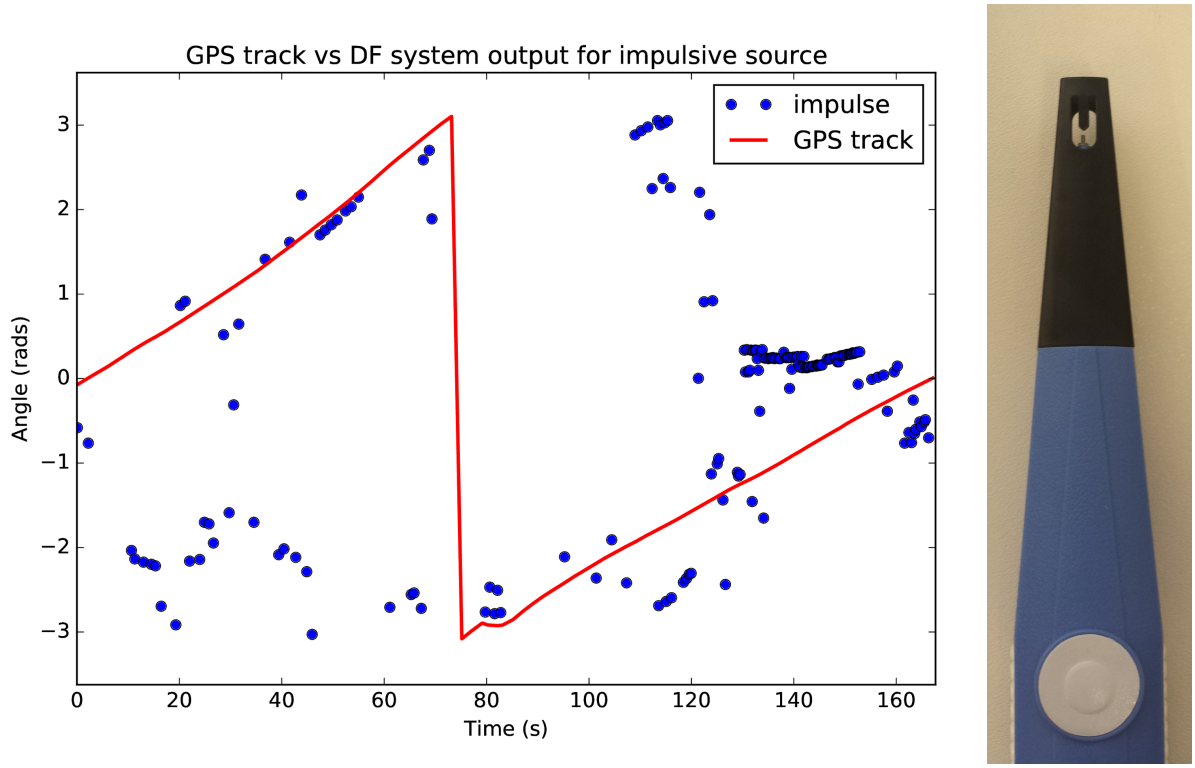
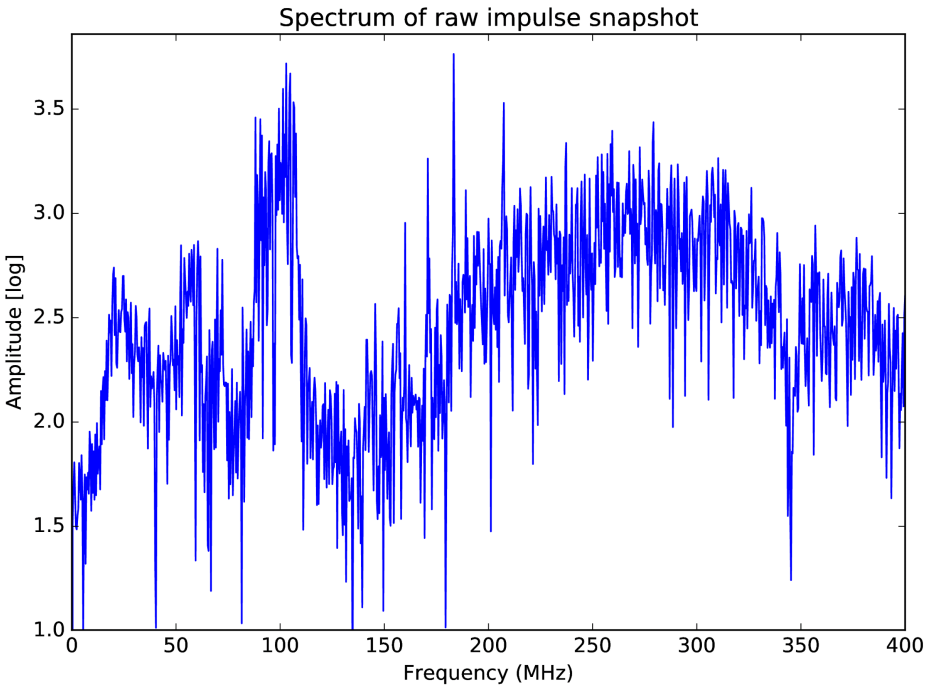
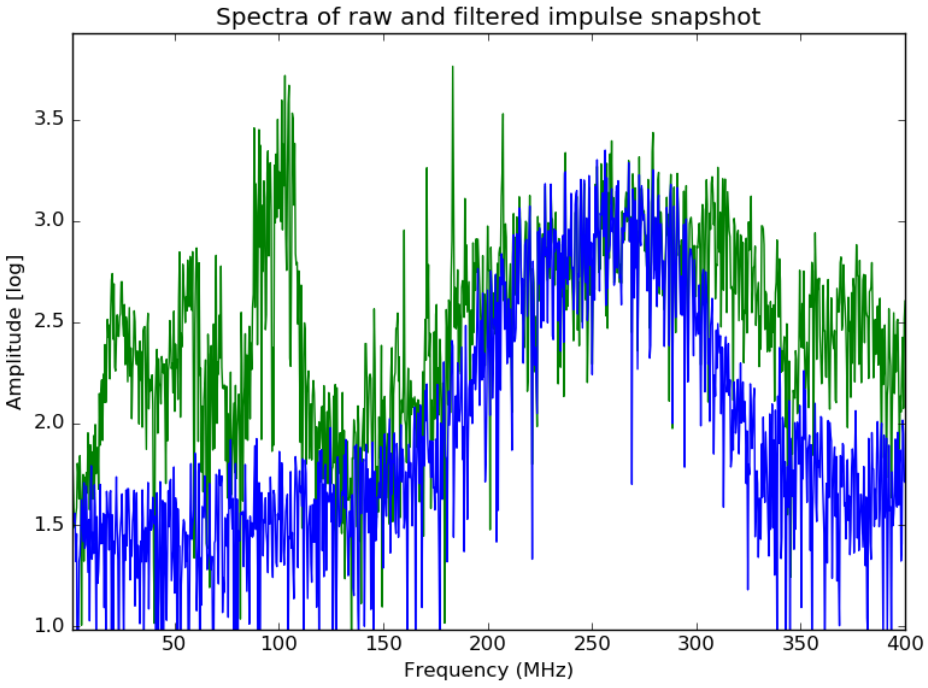


Figure 7.11: Left: Track on UCT sports field of impulsive spark generator showing fairly poor tracking performance. This is *before* the data had been cleaned up with post-processing. Right: impulse generating lighter showing small spark gap.



(a) Spectra of the raw impulse showing strong signals in the 30 MHz to 130 MHz leaking in, along with the antenna main pass band from 200 MHz to 330 MHz.



(b) Raw spectrum (green) from Figure 7.12a overlaid with bandpass filtered version of the signal signal (blue).

Figure 7.12: Spectra before and after bandpass sampling of one captured impulse to select only signals received in the main band of the antenna.

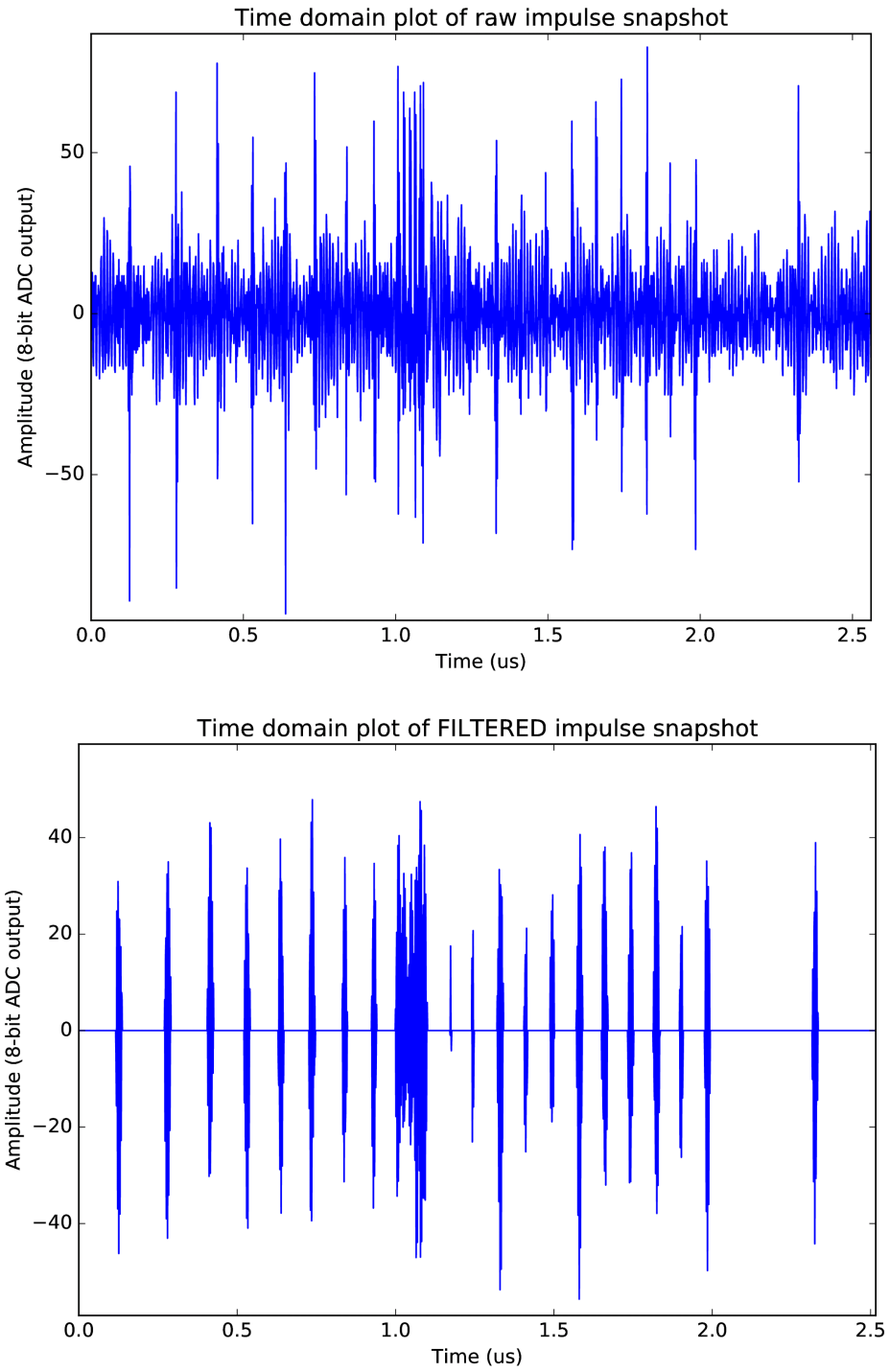


Figure 7.13: Time domain plots of one captured pulse before and after time domain threshold filtering. One spark actually produces this very rapid pulse chain, each ssub-pulse being a few nanoseconds long, and spaced apart by a few dozen nanoseconds. X-axis is time in microseconds.

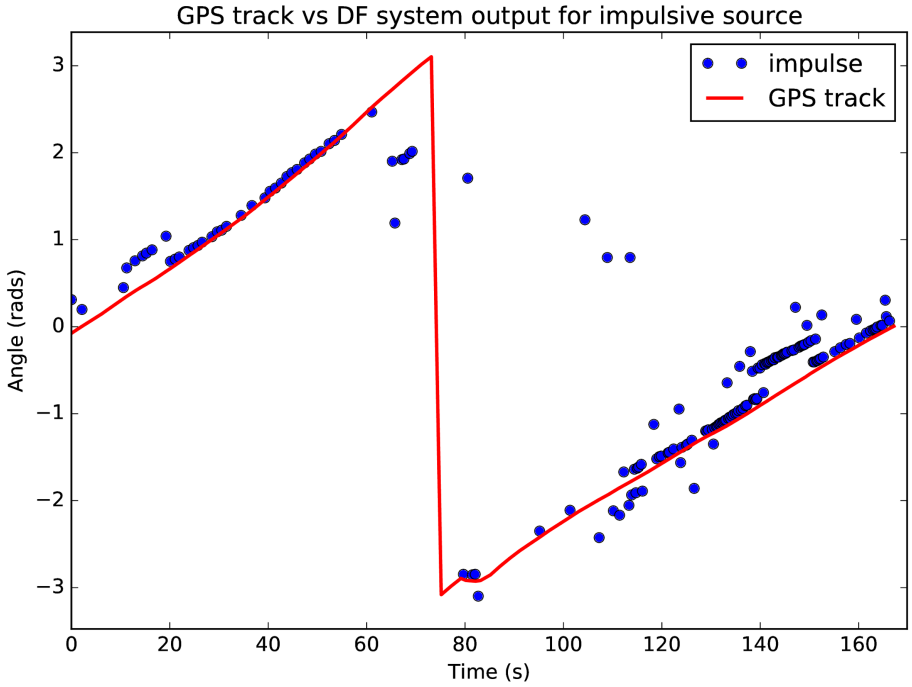


Figure 7.14: Track of the impulsive spark generator *after post-processing the data to clean up ambient noise.*

1. The number of valid points for the HAM radio and Raspberry Pi are both 99% which is an encouraging result. In general, when the system produced a measurement the measurement corresponded to the device. The validity rate for tracking the Valon is slightly lower, but we understand that this is due to it being right in the middle of the band of another strong transmitter which occasionally distorted the signal.

2. The mean for the two strong transmitters (HAM radio and Valon) is below 5°. The mean for the weaker transmitter (Raspberry Pi toggling a pin) is almost double. This could indicate that stronger transmitters have a lower error, which would be expected.

3. The post processing filtering of the spark generator time domain data made a substantial difference. The main benefit of excluding the other strong transmitters from the time domain data was that the detected points were no longer as distorted by other transmitters, but became valid DF results for the spark generator. This is reflected in the change from 25% valid points to 89% valid points. The post processing also significantly improved the error mean. The interpretation of this is that were were initially multiple correlations peaks: the erroneous (caused by sources other than the spark generator) peak was often strongest and hence selected, while the correct peak was skewed. Filtering caused the erroneous peaks to be suppressed, causing the correct one to be selected. Additionally, the skew of the correct peak was reduced so when it was selected it was more accurate.

4. The error mean and standard deviation for the spark generator are an order of magnitude larger than that of the narrow band sources. This is somewhat expected as the amount of signal energy available from the spark is lower than the accumulated narrow band signals, hence the noise is able to skew it more.

5. The accuracy measurements relied on comparing DF results to GPS. However, it is known that GPS is not perfect. There is an error, which can be around 5 m in open sky, which could correspond to 2% this case. Typically GPS also has smoothing filters hence may not provided a point-to-point accuracy.

Device	Detected points	Usable points	Error mean	Error std dev
HAM radio	224	221 (99%)	4.2	3.7
Raspberry Pi	175	173 (99%)	7.5	3.4
Valon	223	203 (91%)	4.5	2.7
Spark (unprocessed)	170	46 (27%)	17.0	9.2
Spark (processed)	170	151 (89%)	10.3	9.1

Table 7.1: Summary of the accuracy results of the devices under test. Mean and standard deviation are given in degrees.

As has been discussed before, the accurate of a DF system is heavily influenced by the environment in which it operates. As such, it is difficult to provide a fair and useful comparison of these results to other DF systems presented in literature or industry. Further work will be required to develop and implement a test procedure which this and other systems can be subjected to in order to provide comparable accuracy results. There are test procedures available in industry, although typically they focus on location narrow band sources or continuous sources. It may be necessary to extend these to incorporate a test for impulsive RFI.

To provide ballpark indications, the RhoTheta RT-300 DF system has a specification of 5° error, and the Rohde-Schwarz ADD10 is specified to 1° RMS error typical. The system developed here appears to be in the same order of magnitude.

For future field trials, the impact of GPS inaccuracy should also be tackled. This could be done either by significantly increasing the radius of the travelled circle, or by using a different reference technique, such as optical tracking.

7.4 Summary

This chapter has looked at the results obtained from two sets of field trials. The first trial at the SKA reserve was done with an early prototype version of that system that only had two antennas and a simple signal path. The test captured real impulses and showed that since there was little to no similarity between different types of impulsive signals, power detection should be used in time-domain capture. Techniques like matched filtering for detection would not be successful unless the exact signal being hunted for was known up front.

The second trial was of the complete system that has been built up and described in previous chapters. The trial was done at the UCT sports field and was done by tracking various narrow-band transmitters as well as an impulse source with both the DF system and a reference GPS tracker. The trial showed that the DF system performed well at tracking narrow band signals, even weak ones in the presence of other signals. The tracking of impulsive signals was shown to be acceptable after a cleaning up of the data which was necessitated by the environment being noisy. Some techniques for clean-up were developed, but further research can be done about how best to extract the impulse train of interest, and whether this can be done in hardware.

Chapter 8

Conclusions

This project has detailed the systematic design and construction of a direction finding system, able to locate both strong impulsive signals and weak narrow band signals.

The project began with defining the problem statement and gathering user requirements in Chapter 1. Some notable requirements of the DF system are that it should be able to locate both narrow-band continuous signals as well as broad-band impulses, should have a 360° field of view, should use hardware and software that fit into the SKA ecosystem, and that the system be a prototype with extensible DF algorithms rather than concentrate on any specific frequency band.

A review of the relevant mathematics and DF techniques from prior literature was then done in Chapter 2. It emerged that the most suitable techniques would be correlative phase interferometry for narrow-band signals, and time difference of arrival (TDoA) for impulsive signals. Phase interferometry was selected as it would be able to make use of the processing power of the SKA hardware to produce digital gain and see signals below the noise floor of the DF system. TDoA was selected due to its robustness in being able to DF any sort of strong broadband pulse.

A block level design for our system was drawn up in Chapter 3 to show how the system would be subdivided into the following distinct subsystems. First, an antenna array would pick up RFI signals which will pass through an RF front end for filtering and amplification. Next, digitisers in the ROACH get the signals into the FPGA where the first stage of high-speed DSP is done. Finally, the processed signals are read out from the ROACH onto a computer. It is here that the angle of arrival (AoA) estimation algorithms are executed, and where the user will interface with the system. This modular design would allow the system to be modified easily in future to, for example, upgrade the antenna array and RF front end to work at a higher frequency that would overlap with the MeerKAT telescope's band of interest. Chapter 3 also looked at results of simulations which were done to compare the phase asmbiguity of circular arrays with varying numbers of elements. The outcome was that, in general, the more elements the array has the lower the error introduced by ambiguity. It also showed that circular arrays with odd numbers of elements have significantly less phase ambiguity than those with even numbers of elements.

Construction of the system was described in Chapter 4 beginning with the antenna array and RF front end. The array was built from four dipoles with a center frequency of 250 MHz. Generally circular arrays with even numbers of elements should be avoided. However the hardware used here was limited to only four simultaneous ADC inputs. Despite this, it was found that by deforming the four element circular array so that it was no longer on a perfect circle, the ambiguity issue was improved significantly. The performance approached that of a five element array. For phase-based direction finding it is best to have a compact array (elements spaced less than a wavelength apart) to minimize ambiguity. However, for TDoA it's best to have a more widely spaced array to make the propagation delays between the elements noticeable in

the time domain. A compromise was made here to find dimensions that worked for both, but the fact that these two requirements are at odds with each other indicates that trying to design a signal system to DF both narrow band and impulsive signals will lead to worse performance than if the systems were independent. Chapter 4 also discussed the RF front end which was built, consisting of four individual RF chains. Each has an amplifier and low pass anti-aliasing filter, along with cabling. Profiling of these RF chains showed that they were relatively well phase matched and provided the expected gain of 20 dB.

Chapter 5 detailed the next phase of the system implementation, the FPGA firmware design for the ROACH. There are two main signal paths in the FPGA: one for the frequency domain signals and the other for time domain signals. The frequency domain path contained a Fourier transform, spectrum cross-multiplier, accumulators and snapshot blocks. The Fourier transform has the ability to receive multiple inputs simultaneously, and allows phase differences at a given frequency to be easily read. Accumulators allow for many cross-correlations to be added together, enabling weak correlated signals to stand out from uncorrelated noise. The time domain signal path contained a power detector and high-capacity snapshot block able to detect and capture impulses when they occur. Lack of overlap in these DSP paths (both in how processing is done and how detections are made) further indicates that coupling narrow band DF and impulsive DF in a single system does not have much advantage over running independent systems. Nevertheless, the lab testing which was done demonstrated that both DSP paths performed well. The lab testing emulated environmental signals by introducing configurable levels of uncorrelated noise in each channel, and generating both correlated narrow band and impulsive signals for capture and processing. The frequency domain path successfully demonstrated using accumulation to produce gain, and the time domain path demonstrated detection of impulses. The control and monitoring logic in the FPGA made it a general purpose runtime programmable system.

The final part of the design and implementation was described in Chapter 6 involving the software interface and direction finding algorithms. The software is a Python package which has been structured in such a way to make if easy to adapt to correlators of different configurations with regards to number of elements or frequency range. The direction finding algorithms work by first building a model of the theoretical response of the array for every angle, and then comparing the real observed signals with the model, finding the angle whose simulated response best matches the observed signals. The response is baseline phase shifts in the case of narrow band direction finding, and time delays in the case of impulse direction finding. This approach is general purpose as any array configuration can be modeled. Additionally, the software catered for injecting calibration factors to offset analogue mismatches. Measurements on the hardware were taken to find calibration factors, and tests showed that calibration successfully lowered the observed error.

The last chapter was Chapter 7 which showed results from two field trials that were carried out. The first trial was early in the project with only a rudimentary two-element setup to test capture of impulses. It obtained useful data on the characteristics of the pulses, but concluded that more elements or a wider spacing was needed to reliably track impulsive signals. The final fields trials on the full DF system involved having it track various transmitters and comparing its results with a GPS logger which had been running simultaneously. It was shown that in general the DF system tracked the sources very well, despite being in a hostile RF environment. The track of the impulsive source was initially not good due to the presence of many strong correlated signals, but after the recorded signals were cleaned up in post-processing the tracking was significantly better. This demonstrated that the algorithms and implementation were able to meet the requirements.

Overall, this project successfully developed a complete direction finding system that met with user requirements. It was successfully tested both in the lab and in the field. Systems design was implemented throughout, along with research into antenna ambiguity, RF chains, ADCs, FPGAs and modular software design. The system was designed to be flexible and reconfigurable to meet the eventual frequency requirements for the SKA. Although the system managed to perform well in both narrow band and impulsive DF, these two requirements were often at odds with each other. They will conflict even more at higher frequencies when the antenna array becomes more compact.

8.1 Future Work

While this project implemented a full proof of concept system, there are still further developments required to get it to be production ready for usable in on-site RFI hunting. The main categories required are:

1. Investigate the viability of splitting the system into independent systems for narrow band continuous RFI hunting and impulsive RFI hunting. Alternatively investigate converting the TDoA sub-system to operate in the frequency domain, using phase unwrapping and whole spectrum cross correlation.

2. Investigate the usefulness of the amplitude parameter of the various frequency domain cross correlations. This system only made use of phase; amplitude was dropped. Amplitude could possibly be an indication of the degree of confidence in the accuracy of the cross correlation measurement, as lower amplitudes of one pair of antennas vs another pair could be an indication of destructive interference or element shadowing.

3. Compare how the computed array manifold vector values match up to the measured ones for a real supplied calibration signal. The field trail results here show that the algorithm does work as expected, but it will be informative to know close the matches are, and also how this varies if various types of interference are intentionally introduced.

4. Extend the system's operating frequency range into the gigahertz range for MeerKAT support. Investigate what what the best band to capture various classes of impulsive signals is. It may well be that they are generally strongest at lower frequencies.

5. Move the design to a larger FPGA which is able to do the array manifold matching or time domain correlation on-chip for higher performance.

6. Develop and carry out a test procedure which is able to give fair and accurate comparisons between the accuracy of this and other systems.

Appendix A

ROACH Development

A.1 Compiling

The ROACH firmware design developed during this thesis was iterated on and compiled more than 100 times as functionality was added and experimentation was done. The design is very large, using over 95% of the available logic on the Virtex 5 FPGA. This has meant that it is a difficult design to compile, often failing to meet timing requirements initially, and hence requiring optimisation. Running at 200 MHz means that a signal needs to get from the data source element to the data end element through any intermediate logic within 5 ns. There are two components to the propagation time of a signal inside the FPGA. First is propagation delay through logic and the second is propagation delay through routing: when the design is very large routing delay goes up as logic gets spread further apart on the chip. Second, when there are lots of gates which a signal has to go through logic delay goes up.

The data signal needs to arrive at the data end element a short time before the clock signal, and it needs to remain valid for some short time after the clock edge. This first amount of time is known as setup time and the time after the edge is known as hold time. If the data propogration delay is too large, setup will not be met. If clock skew is too large, hold time will not be met. When timing fails, Xilinx Sysgen reports which net has failed, between which source and end elements timing has failed, which logic blocks the data line passed through, where they are physically located on the FPGA, how much of the delay was caused by logic propagation and how much routing propagation.

If the source and destination elements are far apart, timing will be dominated by routing. The designer may attempt to use tools like FloorPlanner to move the elements around on the chip to get them close together. For a design like this using about 90% of the logic, there is almost no room to adjust and this becomes very challenging. An example of a difficult design constraint is the DRAM clock fanout, shown in Figure A.1, where it can be seen that a signal has a high fanout from the center of the chip to border IO ports. A potential solution to difficult timing requirements is inserting a dummy latch in the data path so that the data will be latched in somewhere between the original source and destination elements. This is essentially increasing the latency of the pipeline. This is not a easy solution though: often for logic constrained design, adding a latch to a wide data pipeline will make even more data paths fail timing as there is less logic available for optimizing logic placement. It's necessary to carefully consider exactly where and why timing is failing before attempting to fix. Trying to fix timing errors is painful and slow, considering that a compile can take over 6 hours before throwing a timing error. As such, it is often necessary or advantageous to pull out unnecessary logic or optimize the internals of library blocks in order to get a compliant design. Some of these optimizations and new blocks will be discussed in the remainder of this appendix.

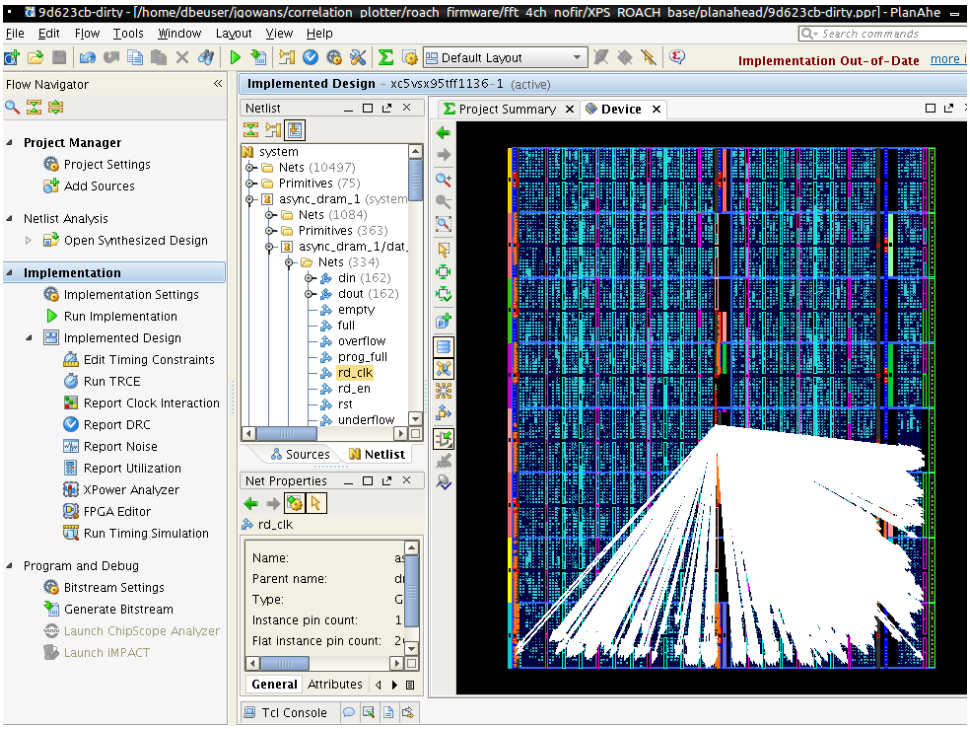


Figure A.1: Xilinx ISE Plan Ahead output showing DRAM clock fanout when design compile failed.

A.2 Vector Accumulators

The vector accumulator (VACC) was a block that I designed for use in this thesis. Its job is quite simply to take in a decimal number every clock cycle and add it to an element at a position in an array of numbers, then move on to the next element, and after doing an add for each element loop back to the start of the array. For example, for a 2048-element vector V, and clock cycle N, with an input value x:

$$V_{N \mod 2048} = V_{N \mod 2048} + x \tag{A.2.1}$$

The VACC takes in an element of the stream decimal numbers every clock cycle and feed out a running total, as well as providing a reset line. There was already a VACC in the standard library, but it used logic slices for arithmetic operations, and hence used a lot of the available logic. This design used many VACCs, hence it was worth the engineering time to design a use-case specific optimized block using the DSP48E modules on the Virtex 5. The DSP48E is a multi-purpose DSP block, and the Virtex 5 has many of them. As seen in Figure A.2 it can be configured to do 48-bit addition. Hence a VACC can be made using two of these and consuming little logic. The design for the VACC using the DSP48E can be seen in Figure A.3.

The input to the VACC is a *stream* of complex numbers from the cross correlator block. The stream is 2048 samples long, and each sample is a complex number consisting of a 37-bit real part and a 37-bit imaginary part. Each 37-bit number is a signed decimal number. After the stream is finished, it repeats for the next FFT, and continues this pattern indefinitely. The DSP48E (as the name implies) deals in 48-bit signed values. Hence, there is a limit to how much it can accumulate before it overflows and the data is lost. There is a BRAM block in the feedback loop which can buffer 2048 numbers in a circular buffer and which acts as the storage

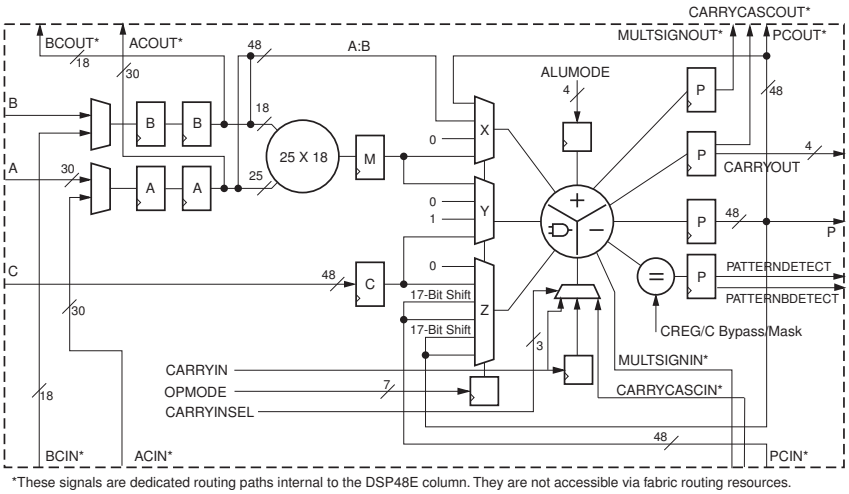


Figure A.2: Internal block diagram of the DSP48E.

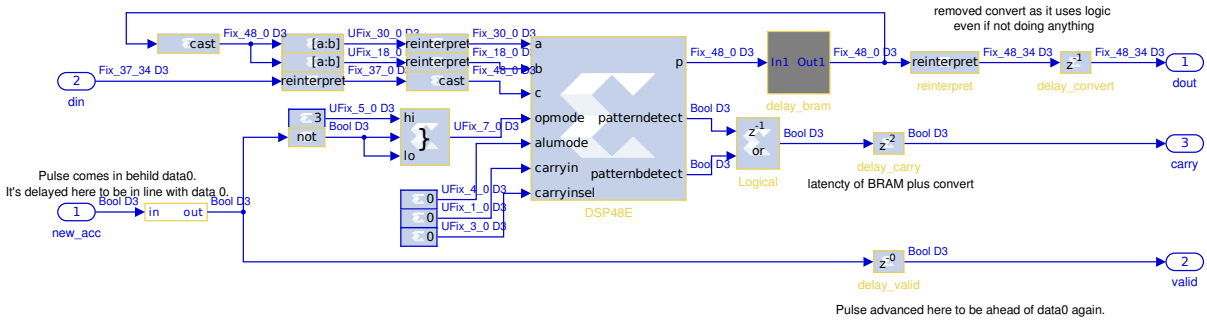


Figure A.3: Internals of the VACC which was design for this project using the DSP48E core.

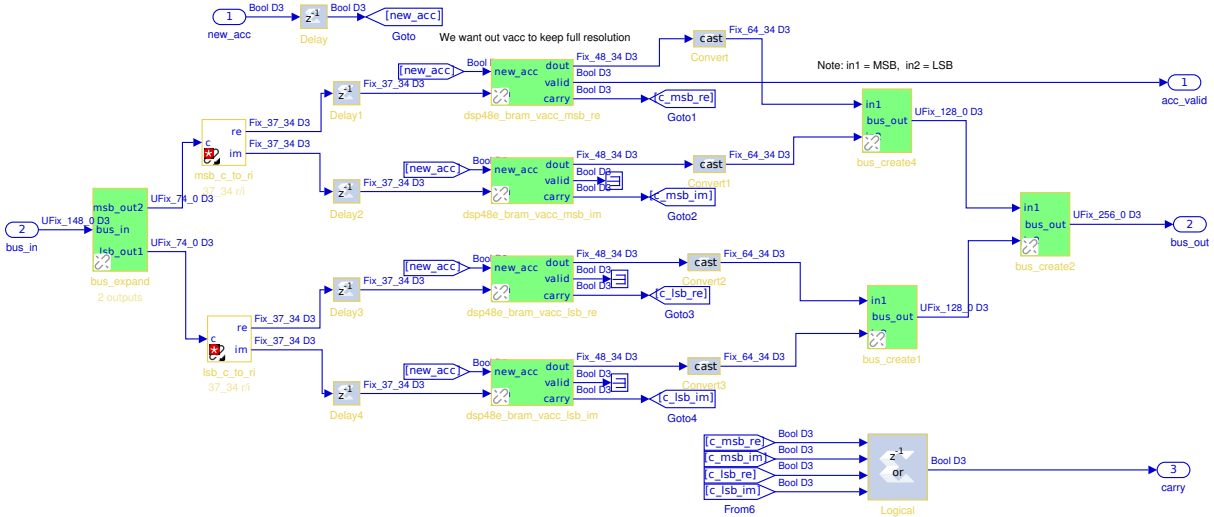


Figure A.4: Internals of dual-sample complex number accumulator, showing how the two samples are first split up, then broken into real and imaginary components, run through green VACC blocks (internals represented above) and the results combined back together.

mechanism for the vector. In order to preserve maximum data fidelity, no truncation happens before the signal is fed into the VACC. *This is key to ensuring that weak signals can be recovered from uncorrelated noise.*

Lab tests of feeding a strong tone right in the middle of a frequency bin (specifically 123.047 MHz) demonstrated that typical overflow will be after a 1 second accumulation time, corresponding to 400 000 accumulations. This should be the typical read out time that the consumer of the VACC should expect.

The design is such that 48 bits are continuously accumulated. After the accumulation has run for a configurable number of iterations, the most significant 32 bits are sliced off and snapped. By accumulating 48 bits, no data is thrown away until the snap. Figure A.4 shows how the DSP48E-based VACCs are used in the full accumulator subsystem. There is one of these accumulators for each cross correlation output (eg: 0x1).

A.3 Using the DRAM

DRAM has large amounts of storage, necessary for dumping large time domain snapshots. However, in order to be usable, the DRAM module needs to be clocked faster than FPGA fabric in order to account for DRAM module overheads (such as data refreshes and internal accounting). For this design, it was necessary to run the RAM module at 266 MHz. This fast speed and high bandwidth buses makes routing difficult and hence DRAM design are significantly harder to compile.

A.4 Cross Multiplier

After the FFT, each antenna combination is multiplied together, one being the original signal and one being the complex conjugate. This is somewhat equivalent to dividing the complex numbers, where the key output is the phase difference between the two antennas.

Some optimisations were done here: these are fairly large multipliers. Each pair of antennas requires an 18 bit multiplier for the real and imaginary components, for both simultaneous channels. This means 4 18-bit multipliers for 10 combinations. 40 x 18-bit multipliers is a lot of hardware! To mitigate this, I made a change to the complex multiplier block to allow selecting of DSP48E for multipliers. This change was committed back into the central code repo for all to use.

The output of a 18_17 x 18_17 multiply is a 37_34 number[1] which explains the 37-bit input to the accumulators.

[1] Notation for these numbers is a fixed pointed integer X_Y where X is the total number of bits and Y is the number of bits used for the decimal place with the rest being used for the integer.

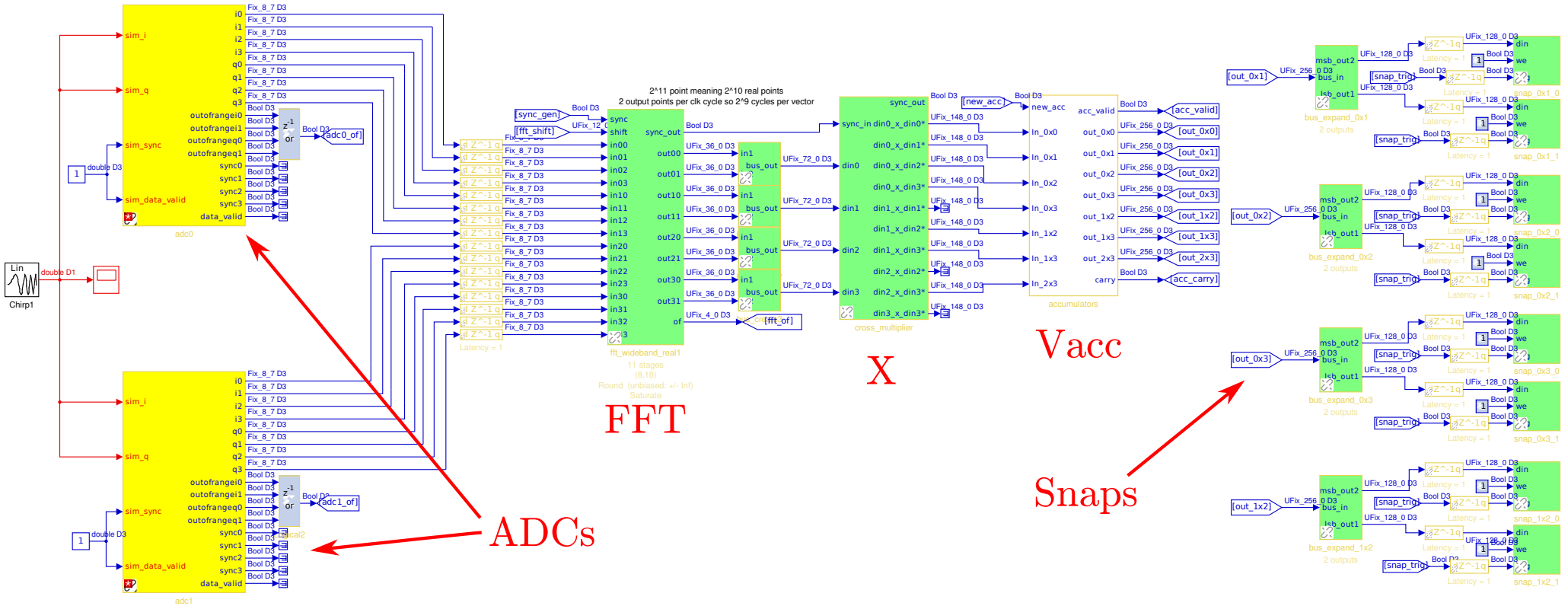


Figure A.5: Signal flow through frequency domain processing chain. Two dual ADCs into 2K point FFT into cross correlator into vector accumulators into snapshots. Note how the bus widths increase from 8 to 36 to 148 to 256 bits.

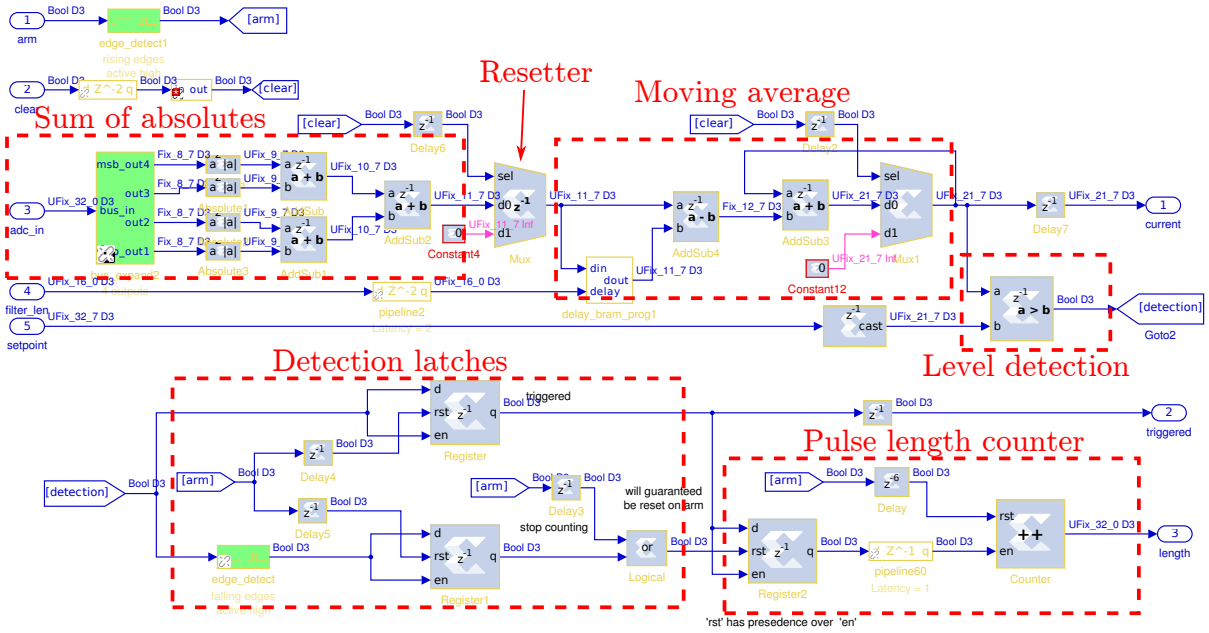


Figure A.6: Internals of the impulse capture subsystem. Boxes show the various logical components in the pulse detection and pulse length measurement.

A.5 Impulse detection

The application of the impulse detection for storing time domain snapshots has been discussed earlier in the body of the thesis. The impulse detector and pulse length counter is a subsystem that was designed for this thesis. It consists of a few parts. First, the sum of the absolute values of the ADC samples is taken. There are four samples that need to be summed as there are four ADC samples per FPGA clock cycle. Second, this sum-of-absolutes is fed into a rolling-sum filter with a runtime-programmable length. The output of the rolling sum filter goes into a level detection block with a runtime-programmable threshold. When a detection happens, the detection latches begin counting samples and storing the ADC samples to DRAM. The pulse length counter will continue clocking and the DRAM will continue recording until the level detection drops below the threshold. The pulse length can then be read out at runtime, allowing the computer to read out exactly the pulse from DRAM. This subsystem is show in Figure A.6

Appendix B

ADC Calibration

There are a few areas where calibration needs to take place in order to ensure that the system performs as expected. What will be discussed here is work on calibrating the ADCs and calibrating the phase of the RF chain from the antennas to the output of the correlator. This RF chain calibration will be done in two stages:

- firstly, the path starting at the input of the LNAs will be calibrated as this can be easily done in the lab by injecting a phase coherent tone into all LNAs simultaneously.
- secondly, the entire RF beginning at the antennas will be calibrated by taking the system out into the field and transmitting tones from known positions

B.1 iADC

The iADCs suffer from a few mismatches between the cores which should be calibrated out for optimal performance. These include:

1. The offsets of each core should be set to 0. Each core can have its offset adjusted from -31.75 LSB to 31.75 LSB in steps of 0.25 LSB.
2. The gains of the cores should be made equal. The gain of each core can be adjusted from -1.5 dB to 1.5 dB in steps of 0.011 dB.
3. The sampling delay between the cores should be adjusted such that the sampling time between each core is as intended. This could be a phase difference of 0 for sampling independent signals in phase, or $\pi/2$ for I/Q sampling or π for interleaved sampling of a single stream.
4. There are other parameters which can also be calibrated such as the data ready delay adjustment or the gain compensation, but these are not very relevant for this work so will be ignored. Only the analogue gain, offset and sample delay will be calibrated for this work.

The iADC can be calibrated either in interleaved or independent mode. The algorithms are the same for each except for where the data is sourced from. Interleaved calibration sources all of its data from a single RF input, then splits the data up into what was read by each core and does core calibration. Independent calibration takes one stream of data samples from one of the RF inputs and another stream for the other input. Calibration is then done on this. In independent mode, care should be taken to ensure that the signal at each port is exactly the same in terms of amplitude and phase. The calibration process attempts to force these parameters to be equal.

Table 7-8. Digital Output Coding (Nominal Setting)

Differential Analog Input	Voltage Level	Digital Output I or Q (Binary Coding)	Out-of-range Bit
> 250 mV	> Positive full-scale + 1/2 LSB	1 1 1 1 1 1 1 1	1
250 mV 248 mV	Positive full-scale + 1/2 LSB Positive full-scale - 1/2 LSB	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0	0 0
1 mV -1 mV	Bipolar zero + 1/2 LSB Bipolar zero - 1/2 LSB	1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	0 0
-248 mV -250 mV	Negative full-scale + 1/2 LSB Negative full-scale - 1/2 LSB	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0	0 0
< -250 mV	< Negative full-scale - 1/2 LSB	0 0 0 0 0 0 0 0	1

Figure B.1: ADC input differential voltage vs output binary number. Src: iADC datasheet

	Uncalibrated	Calibrated
ADC0-I DC offset (bits)	−0·75	−0·04
ADC0-Q DC offset (bits)	−0·55	−0·09
ADC1-I DC offset (bits)	−0·81	−0·02
ADC1-Q DC offset (bits)	−0·44	0·01
ADC0 phase mismatch (radians)	−0·0550	0·0007
ADC1 phase mismatch (radians)	−0·0336	0·0063

Table B.1: Performance improvements in ADC after calibration of DC offset and core sampling delay

The actual interpretation of the output of the ADC (as the ADC datasheet intends the binary number to be interpreted) is a number which oscillates between -1 and 1 depending on whether the differential voltage is slightly positive or slightly negative, even in the presence of no signal. It does not have a concept of 0. This is shown in fig. B.1. However, the data structures of the FPGA gateware certainly do have a concept of 0. As a result, they incorrectly interpret this number as -1 to 0. This causes the signal to have a spurious DC offset. To compensate for this, the iADC is calibrated by applying a positive offset to the output number such that it is centered around 0 and hence does not have a DC offset.

B.2 Phase mismatch correction

One of the ways that the timing mismatch in the ADC cores manifests is by the introduction of spurious phase differences when the same signal is injected into all of the channels. In order to calibrate this out, a calibration routine was developed. The hardware is set up so that a *single* noise generator is feeding the same broad-band noise source to all ADC channels. In theory, this means that there should be zero measured phase difference between the signal seen by each ADC core. However, this is not the case, as can be seen from the cross correlation phase measurements shown in Figure B.2. That figure shows that there are slopes caused by sampling phase error in the ADC. The calibration routine generates this plot, reads the slope of each line, calculates whether each core needs to be phase shifted forward or backward, and then updates the calibration registers on the ADCs. This process is repeated until the cores are sampling in phase. After calibration was complete, all of the phase shifts were below 0.05 radians.

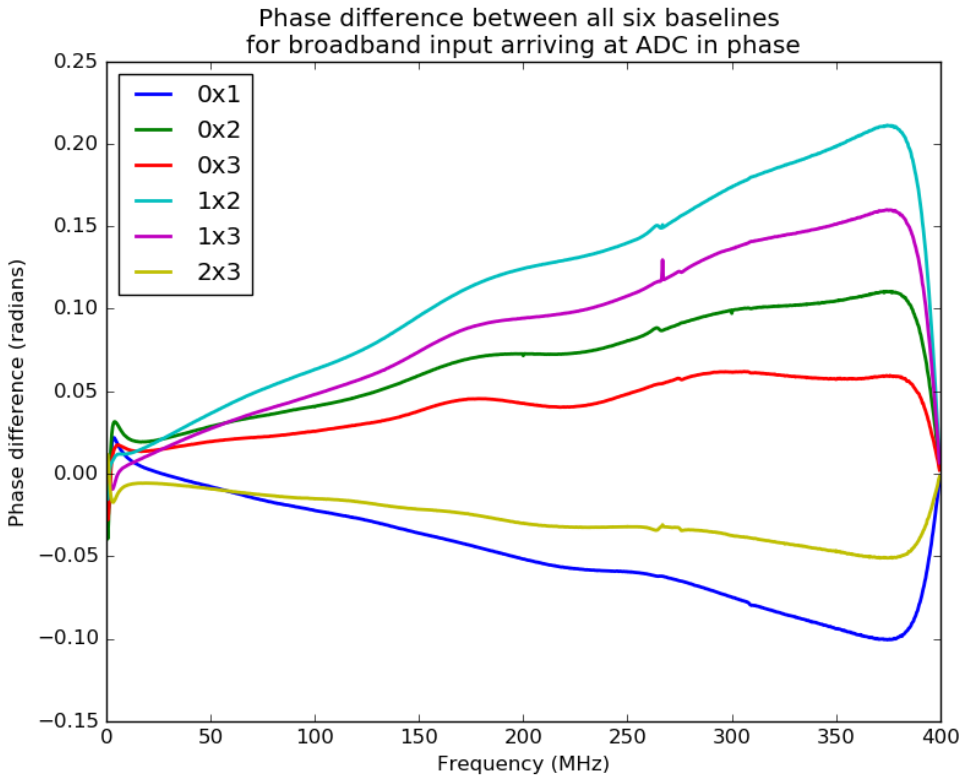
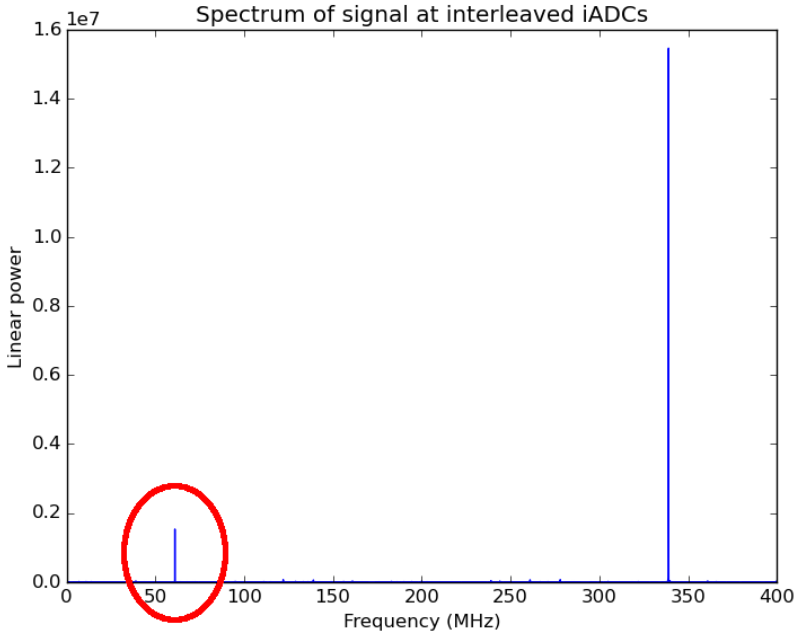


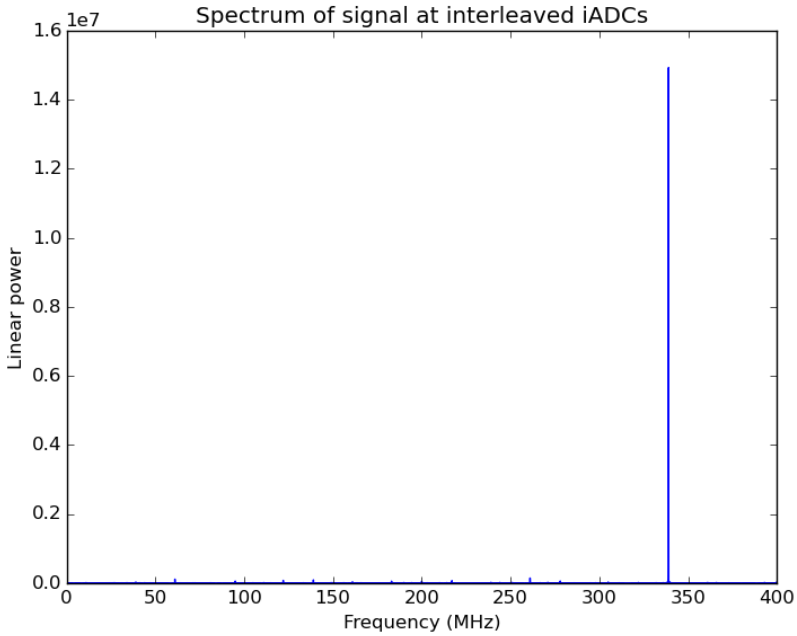
Figure B.2: Phases before calibration

The other way in which timing offsets manifest as spurious signals is when the cores are configured to operate in interleaved mode. Although this wasn't the mode that the ADCs ended up being used for in this thesis, they were originally used this way during initial testing so it is worth mentioning. When operating interleaved, we expect the one core to sample *exactly* in between the samples of the other cores. If this is not the case, a spurious aliased signal will be produced. The same soft or calibration routing as discussed earlier was developed for this form of calibration. Here, a single tone is injected, and the calibration routine measures the magnitude of the aliased signal. The calibration routine progressively adjusts the core phase offset register until the aliased signal is at a minimum. This can be seen in Figure B.3: before calibration there is a strong alias signal, after calibration it is gone.

All of the calibration routines described here work by pulling large time domain ADC data snapshots from the FPGA's DRAM onto a computer, and doing DSP algorithms such as FFTs and gradient calculation using Python code on the computer. The code for the ADC calibration routines is available: `https://github.com/jgowans/iADC_calibration`



(a) Spectrum BEFORE calibration



(b) Spectrum AFTER calibration

Figure B.3: Comparison between spectrum of interleaved iADCs showing how intermodulation product is removed (or significantly reduced) by calibration

Appendix C

Array Element Coordinate Calculator

The code discussed in this section is available here:
`https://github.com/jgowans/array-element-coordinate-calculator`.

The software developed in the thesis for direction finding now requires as an input a list of coordinates of the antenna array being used. These coordinates will be used to create a model of the array.

While it is trivial to measure the distance between elements of an array, it is not trivial to calculate absolute coordinates for elements. The process of trilateration can be used to ascertain the coordinates of elements by ascertaining the points of intersections of circles formed from the baselines of the array. This involves a fair amount of geometry and trigonometry, and while not necessarily difficult is certainly laborious.

As the system developed here needs to be easily reconfigurable for different arrays, it is important that the user does not have to go through the manual process of calculating array element coordinates when a different array is used of if the array geometry is changed. This process should be made as simple as possible. For this reason, a small software package has been written which does the job of converting between the measured baselines and the coordinates of the elements.

Obviously, the introduction of a coordinate system requires some assumptions, as there are infinitely many ways in which a shape can be placed onto a coordinate system. The process which the Python software package follows (with some small code example snippets) is:

- Assume that element 0 of the array is at the origin, $(0, 0)$
- Assume that element 1 is at location $(d, 0)$, where d is the baseline distance measured between element 0 and element 1.

```
p0 = Point(0, 0)
p1 = Point(distance_matrix_target[0][1], 0)
```

- For all other elements:
  - create a circle with centre at element 0 and radius the distance between element 0 and that element
  - create a circle with centre at element 1 and radius the distance between element 1 and that element
  - find the points corresponding to the intersection of the circles and select the point with the positive y component (there should be only one) as the location of the element

```
def initial_trilateration(p0, p1, distance_matrix):
    points = [p0, p1]
    for target_number in range(2, distance_matrix.shape[0]):
        d0t = distance_matrix[0][target_number]
        c0t = Circle(p0, d0t)
        d1t = distance_matrix[1][target_number]
        c1t = Circle(p1, d1t)
        points.append(c0t.positive_intersection_with(c1t))
    return points
```

- Now that preliminary coordinates have been calculated for each element, repeat the process a few times for all elements, selecting the centroid of the intersection points as the location of the element. This is done to try to mitigate measurement error.

```
def trilateration_phase(points_in, distance_matrix):
    points_out = [points_in[0]]  # force point 0 to stay at origin
    for target_idx in range(1, len(points_in)):
        approximations = []  # points which are approximate 'target'
        # we want pairs of circles which intersect at target. Will have circle A and circle B.
        for source_a_idx, source_b_idx in itertools.combinations(range(0, len(points_in)), 2):
            # ensure the centre of the circle isn't target.
            # It must be defined by another point and target
            if (source_a_idx != target_idx) and (source_b_idx != target_idx):
                a_center = points_in[source_a_idx]
                a_rad = distance_matrix[source_a_idx][target_idx]
                a = Circle(a_center, a_rad)
                b_center = points_in[source_b_idx]
                b_rad = distance_matrix[source_b_idx][target_idx]
                b = Circle(b_center, b_rad)
                intersections = a.intersection_with(b)
                approximations.append(points_in[target_idx].get_closest(intersections))
        approximation = get_centroid(approximations)
        points_out.append(approximation)
    return points_out
```

- Remove the offset caused by forcing element 0 to the origin. Rather place the origin in the centroid of the array.

```
def remove_origin_offset(points):
    centroid = get_centroid(points)
    points = [point - centroid for point in points]
    # now rotate
    theta = np.arctan2(points[0].y, points[0].x)
    theta = -theta  # want to rotate backwards
    rot_matrix = np.array([[np.cos(theta), -np.sin(theta)],
                           [np.sin(theta), np.cos(theta)]])
    rotated_points = []
    for point in points:
        point_vector = np.array([[point.x], [point.y]])
        rotated_point_vector = np.dot(rot_matrix, point_vector)
        rotated_points.append(Point(rotated_point_vector[0][0],
                                    rotated_point_vector[1][0]))
    return rotated_points
```

- Output the coordinates to the terminal in a format which can be copied and pasted into an array configuration file.
- Output a list of measured baselines and final baselines to the terminal to indicate how close the software was able to get the calculated baselines to the measured baselines. The user can look at the percentage errors to see if they want to retake measurements.

Appendix D

Field Trials dimensions

In order to ascertain a useful radius to use when walking the transmitters around the DF system during the field trials, it was informative to do some simulations. As a reminder, the GPS track showing the radius of the transmitters around the receiver in the center of the field is shown in Figure D.1. The simulations aim to inform the expected error from both ground reflections and non-flat wave fronts. Ground reflections are the multi-path from a reflected ray bouncing off of the ground, and non-flat wave fronts are due to the transmitter not being significantly far away from the receiving array so that the circular propagation may cause a delay between when different elements of the receiving array pick up the signal. The objective of the simulations is to ensure that the selected transmitter distance (radius) from the receiver has acceptably low error from these two error sources.

Assume there are two receiving antennas, RxA and RxB. These would be two elements of the DF system. Let RxA be some distance R from the transmitter, and RxB be 1 m farther away from the transmitter, $R + 1$. Let the transmitter, Tx, be producing signal $s(t)$.

Recall that the equation for a received direct signal is:

$$r(t) = f_a(\theta_a) f_b(\theta_b) s(t) \frac{\lambda}{4\pi R} \exp\left\{-j2\pi \frac{R}{\lambda}\right\} \tag{D.0.1}$$

where $f_a(\theta_a)$ and $f_b(\theta_b)$ are the beam patterns of antenna Tx and RxA (or equally for RxB) respectively. For folded dipoles, we will assume them to be cosine beams. If we keep the transmitter and receivers at the same height then these terms can be set to 1. There are two signals paths, one from the direct beam, R and one from the reflected beam R'. A reflected beam has a ground reflection coefficient $\rho e^{j\phi}$ [23].

$$\rho e^{j\phi} = \frac{(\kappa - j\chi)\sin\theta - \sqrt{(\kappa - j\chi) - \cos^2\theta}}{(\kappa - j\chi)\sin\theta + \sqrt{(\kappa - j\chi) - \cos^2\theta}} \tag{D.0.2}$$

where $\chi = \alpha/\omega\epsilon_0$ with α as the soil conductivity and ω being angular frequency, κ the soil dielectric constant and θ grazing angle

Now we have $r'(t)$ which is the received signal as a result of the longer path length: R' as well as the ground reflection coefficient: $\rho e^{j\phi}$

$$r'(t) = \rho e^{j\phi} s(t) \frac{\lambda}{4\pi R'} \exp\left\{-j2\pi \frac{R'}{\lambda}\right\} \tag{D.0.3}$$

The final signal arriving is the combination: $r(t) + r'(t)$. Note in the worst case, where θ is very small (large distance, small elevation), $R \approx R'$ and $\rho e^{j\phi} \approx -1$. This equates to complete destructive interference and should be avoided.

We want to know what range and elevation would produce results which suffer least from ground reflection interference. Most critically, we want the phase of combined signal to be as

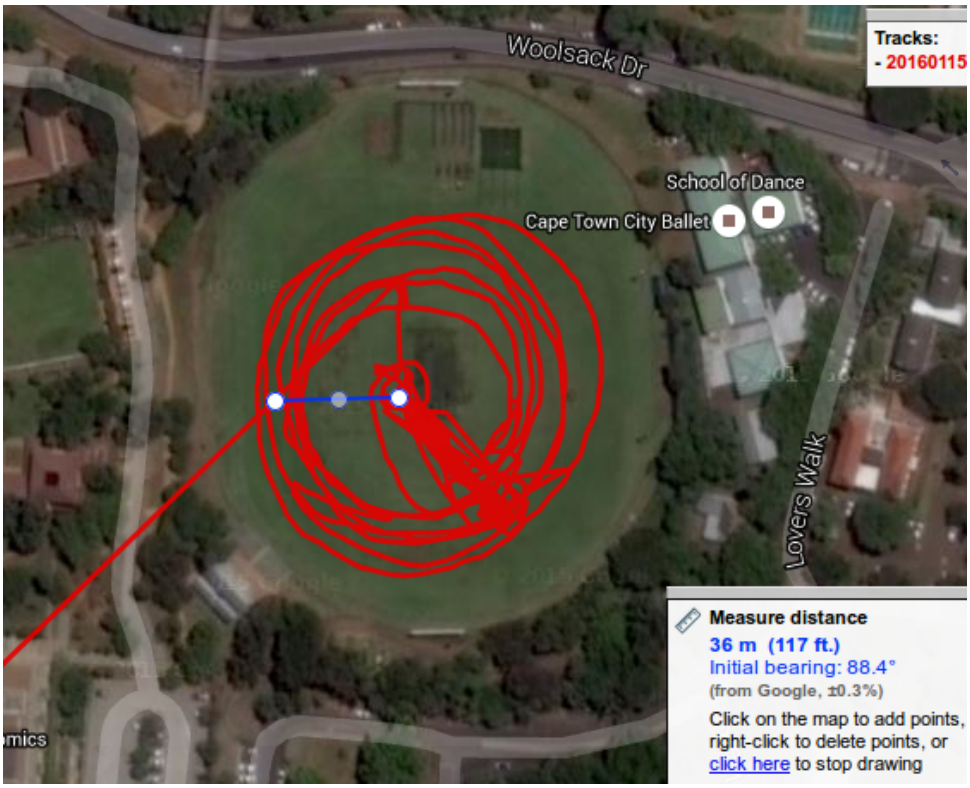
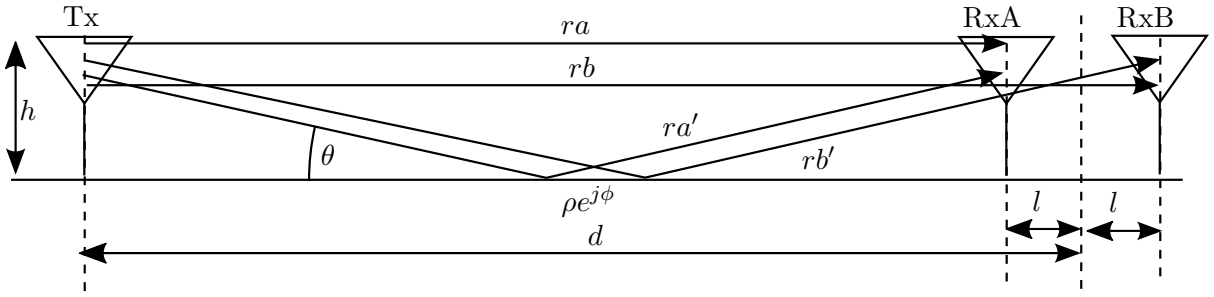


Figure D.1: GPS track of transmitters around the receiver.

close as possible (within a few percent error) of the ideal case. Assumptions: Tx and Rx heights are the same. The distance between RxA and RxB is 1 m. This is representative of the array which was constructed in this thesis. A simulation was run for height $h \in [0.5, 5]$ and radius $R \in [5, 50]$ Algorithm:

1. For a given R
   a) compute the ideal received signal for RxA and RxB with no ground reflection. Friis only.
   b) Store this ideal amplitude and phase difference.
   c) for a given h:
      i. compute θ, the grazing angle and R' the longer path length
      ii. compute the resulting $\rho e^{j\phi}$ from θ
      iii. compute the received signal from the reflection which is a function of the path length multiplied by the ground reflection coefficient.
      iv. add the direct and reflected.
      v. add a data point to the plot: difference between ideal and actual amplitude. difference between ideal and actual phase.

A model of what is being simulated is shown in Figure D.2a, and the results of the simulation shown in the remainder of Figure D.2. The x-axis and y-axis define the geometry of the setup (distance and height) and the z-axis (colour) of the figures shows the resultant phase shift and



(a) Model being simulated. Ideal case is the phase difference between receiving elements from ra and rb paths only. Real case is phase difference due to all four paths with ground reflection coefficient $\rho e^{j\phi}$

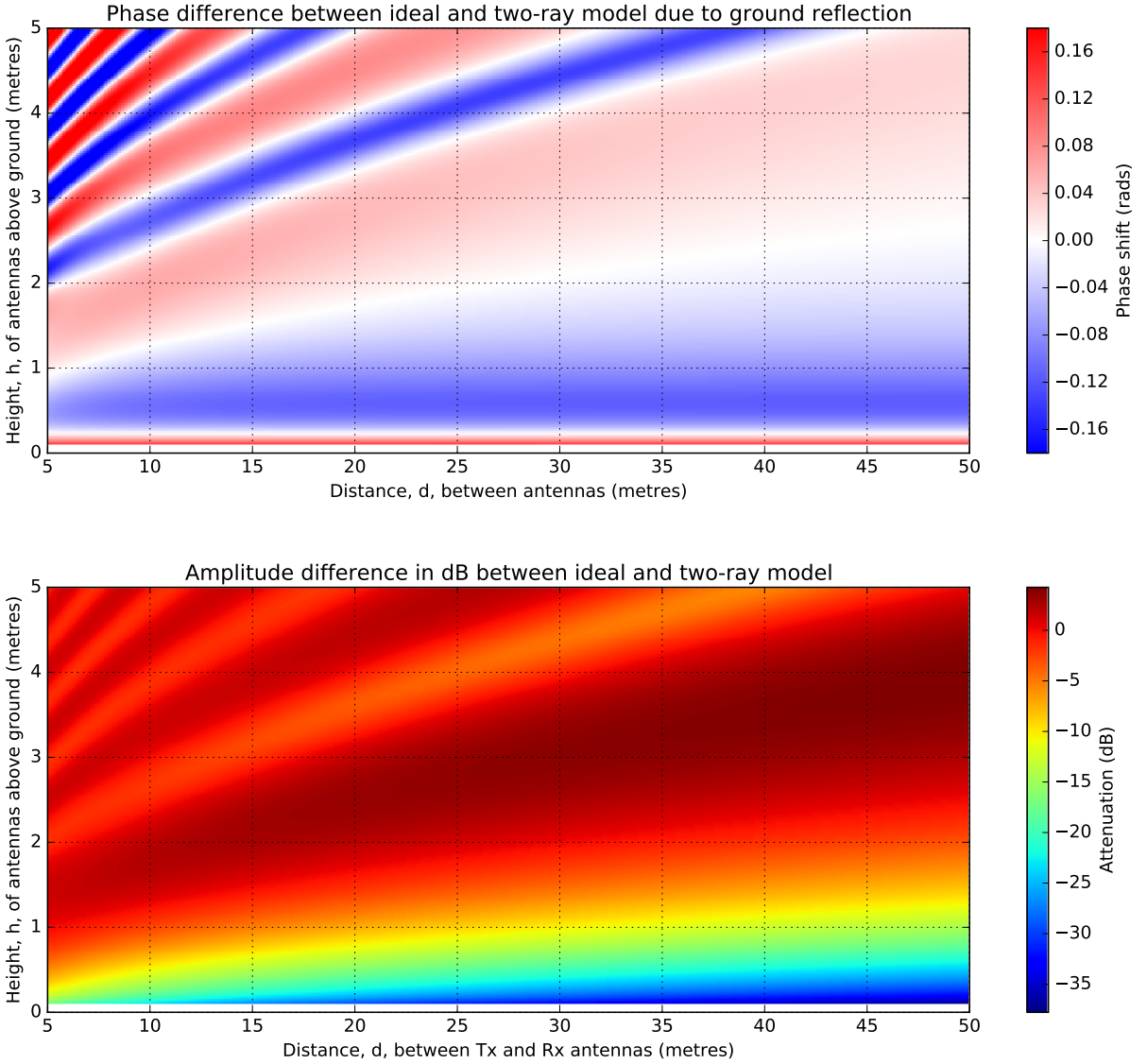


Figure D.2: Simulation of change in antenna array response due to ground reflection multipath. (Note: the colour label incorrectly uses "attenuation" and negative values. It should be gain and negative values, or alternatively attenuation and positive values.)

amplitude change induced by the reflected beam. It can be seen that below 3 m elevation, it looks like the phase becomes fairly uniform with distance after about 25 m. Above 1.5 m elevation the amplitude degradation is below 5 dB so no need to be concerned.

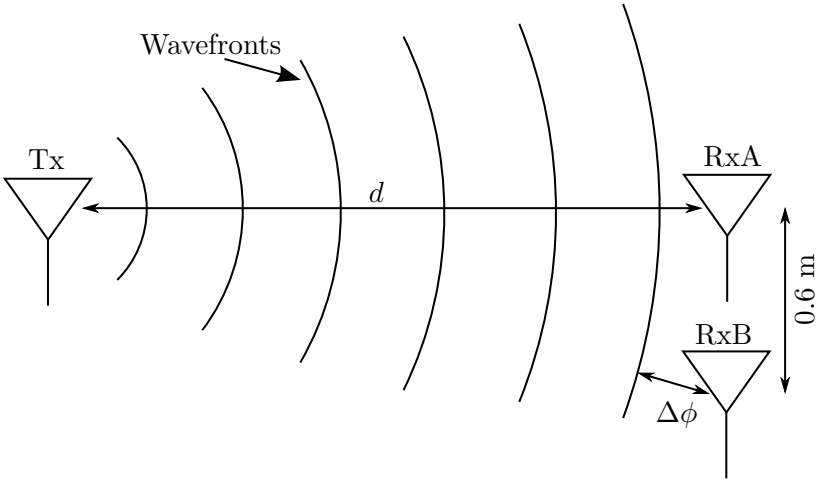
Note that there was no indication during the field trials whether ground reflections did actually occur. The calculations done here were more to minimize the reflections if there were any, and to have something to back up the choice of antenna array elevation and radius being walked.

The other form of error mentioned is due to non-flat wave fronts. Waves impinging in an array are often modelled as as having flat wave fronts, but this is only true when the transmitter is infinitely far from the receiver. In practice, there is some curvature to the wave fronts. The closer the transmitter is to the receiving array the more apparent this becomes. As such, the transmitter needs to be far enough that the error is below a suitable threshold. Again, a simulation was done to to help decide on a suitable distance.

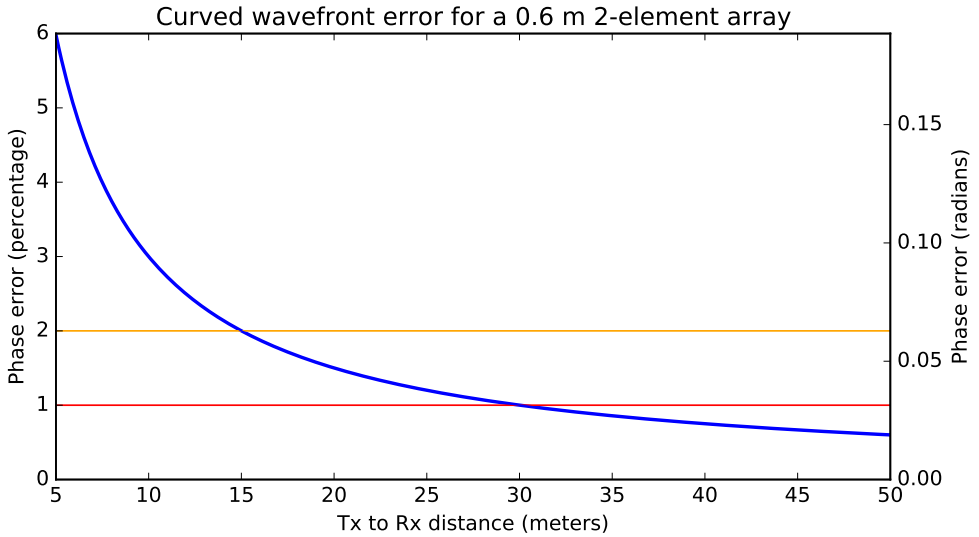
The model being simulated is the same as before, but now the top-down view is of interest. This model and the results of the simulation can be seen in Figure D.3 showing how a lateral offset can result in an unintended phase shift. The results of the simulation of phase shift percentage error vs distance d are shown. It can be seen that after 30 m the error drops below 1%.

After considering both of these simulations, it was considered suitable to use a distance (radius) of 35 m and a Tx/Rx height of 2 m off the ground. This geometry is what was used for the field trials at the UCT sports field, described earlier in the body of the thesis.

The code for the simulations discussed in this appendix can be found in the following repo: `https://github.com/jgowans/phase_ambiguity`



(a) Model being simulated showing how non-flat wave-fronts impinging on an antenna array can cause a phase shift between when one antenna receives the signal and when the other does.



(b) Results of the simulation showing how percentage phase error drops off as distance increases.

Figure D.3: Non-flat wave-front simulations.

Bibliography

[1] SKA South Africa. (2015, November) SKA SA Official Website. [Online]. Available: http://www.ska.ac.za

[2] ——. (2016, November) SKA SA Official MeerKAT Fact Sheet. [Online]. Available: http://www.ska.ac.za/wp-content/uploads/2016/07/meerkat-fact-sheet-2016.pdf

[3] H. Krim and M. Viberg, "Two decades of array signal processing research: the parametric approach," *Signal Processing Magazine, IEEE*, vol. 13, no. 4, pp. 67–94, 1996.

[4] R. Poisel, *Electronic warfare target location methods.* Artech House, 2012.

[5] S. V. Schell and W. A. Gardner, "18 high-resolution direction finding," *Handbook of statistics*, vol. 10, pp. 755–817, 1993.

[6] I. Dacos and A. Manikas, "Estimating the manifold parameters of one-dimensional arrays of sensors," *Journal of the Franklin Institute*, vol. 332, no. 3, pp. 307–332, 1995.

[7] H. L. Van Trees, *Detection, estimation, and modulation theory, optimum array processing.* John Wiley & Sons, 2004.

[8] T. E. Tuncer and B. Friedlander, *Classical and modern direction-of-arrival estimation.* Academic Press, 2009.

[9] Naval Air Warfare Center Weapons Div Point Mugu CA, *Electronic warfare and radar systems engineering handbook.* Naval Air Warfare Center Weapons Division, 2012.

[10] R. Grabau and K. Pfaff, "Funkpeiltechnik," *Stuttgart: Franckh'sche Verlagshandlung, W. Keller & Co*, pp. 350–351, 1989.

[11] H. H. Jenkins, *Small-aperture radio direction-finding.* Artech House, 1991.

[12] P. Clancey. Japanese radio communications and radio intelligence. HyperWar. Accesses 2015-09-09. [Online]. Available: http://ftp.ibiblio.org/hyperwar/USN/ref/KYE/CINCPAC-5-45/index.html

[13] P. J. Gething, *Radio direction finding and superresolution.* IET, 1991, no. 33.

[14] R. A. Poisel, *Introduction to communication electronic warfare systems.* Artech House, Inc., 2008.

[15] Radiomonitoring and R. Catalog, "Introduction into theory of direction finding," Rhode & Schwarz, Tech. Rep., 2011/2012. [Online]. Available: https://cdn.rohde-schwarz.com/us/campaigns_2/a_d/Introduction-Into-Theory-of-Direction-Finding.pdf

[16] I. Pellejero. Adcock/watson-watt radio direction finding. Accessed 2015-09-01. [Online]. Available: http://www.ipellejero.es/tecnico/adcock/english.php

[17] I. of Electrical and E. Engineers., "Coherence and time delay estimation," *Proceedings of the IEEE.*, vol. 75, no. 2.

[18] F. Gustafsson and F. Gunnarsson, "Positioning using time-difference of arrival measurements," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, vol. 6. IEEE, 2003, pp. VI–553.

[19] H. C. So, Y. T. Chan, and F. K. W. Chan, "Closed-form formulae for time-difference-of-arrival estimation," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2614–2620, 2008.

[20] D. C. Schleher, *Electronic warfare in the information age*. Artech House, Inc., 1999.

[21] C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 4, pp. 320–327, 1976.

[22] W. Kester, "Mt-001: Taking the mystery out of the infamous formula," snr= 6.02 n+ 1.76 db," and why you should care," *REV. 0*, pp. 10–03, 2005.

[23] R. Collin, *Antennas and radiowave propagation*, ser. McGraw-Hill series in electrical and computer engineering. McGraw-Hill Higher Education, 1985. [Online]. Available: https://books.google.co.za/books?id=pgJTAAAAMAAJ