2014-08-20

Digital Back End Development and Interference Mitigation Methods for Radio Telescopes with Phased-Array Feeds

Richard Allen Black
Brigham Young University - Provo

Digital Back End Development and Interference Mitigation Methods for Radio Telescopes with Phased-Array Feeds

Richard Black

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

Master of Science

Brian D. Jeffs, Chair
Karl F. Warnick
Neal K. Bangerter

Department of Electrical and Computer Engineering

Brigham Young University

August 2014

ABSTRACT

Digital Back End Development and Interference Mitigation Methods for
Radio Telescopes with Phased-Array Feeds

Richard Black
Department of Electrical and Computer Engineering, BYU
Master of Science

The Brigham Young University (BYU) Radio Astronomy group, in collaboration with Cornell University, the University of Massachusetts, and the National Radio Astronomy Observatory (NRAO), have in recent years developed and deployed PAF systems that demonstrated the advantages of PAFs for astronomy. However, these systems lacked the necessary bandwidth and acquisition times to be scientifically viable. This thesis outlines the development of a 20-MHz bandwidth system that can acquire for much longer periods of time and across much larger bandwidths than previous BYU systems. A report of the deployment of this system on the 305-meter reflector at the Arecibo Observatory in Puerto Rico is also summarized.

The Commonwealth Scientific and Industrial Research Organisation (CSIRO) is currently constructing a PAF-equipped synthesis imaging array named the Australian Square Kilometre Array Pathfinder (ASKAP) that offers great promise for widening FOVs and enhancing RFI mitigation techniques. Previous work in RFI mitigation has demonstrated effective cancellation for synthesis imaging arrays under the assumption that the processing bandwidth is narrowband and correlator dump times are short. However, these assumptions do not necessarily reflect real-world instrument limitations. This thesis explores simulated adaptive array cancellation algorithm effectiveness as applied on the ASKAP instrument given realistic bandwidths and correlator dump times. The results demonstrate that active RFI mitigation performed across long baselines is largely ineffectual.

Keywords: radio astronomy, phased-array feeds, radio-frequency interference, synthesis imaging arrays, interferometry, digital signal processing, australian square kilometre array pathfinder

ACKNOWLEDGMENTS

It goes without saying that this work could not have been done without the help of so many in my life. I would not be the same man, student, or engineer if not for the continual support of those around me.

My wife Karen has made enormous sacrifices to make this thesis a reality. Whether it be a read-through of my latest revision (a task she rarely enjoyed), an eye-glazing, one-sided brainstorming session, or a few extra hours of quiet sleep after a long night in front of the computer screen, she enabled me to complete this work. Without her constant support, love, and encouragement, my accomplishments would be far fewer.

My daughter Chloe, although only 10 months old, has made me the proudest father. She has brought a smile to my face during rough times and inspired me to become a better man and father. Although her well-intentioned efforts to help me typically involved smashing random keyboard keys, I know that without her, I would not be as motivated to excel.

My parents and sister merit more recognition than I can include here. My mother and father blazed the trail and paved the way for me to pursue an advanced degree. Their examples are those of paragons, causing me to grow into maturity both mentally and financially. Together with my sister, they at times provided me with much needed respite from a screaming baby, enabling me to focus on my research. Their lifetime of service has not gone unnoticed and is precious to me.

As for my adviser Brian Jeffs, I would not have the necessary academic confidence in my research without his astute observations and wise counsel. He has on numberless occasions given me much needed advise and correction. I always knew that his door was open to any question I had, regardless of its triviality. He along with committee members Karl Warnick and Neal Bangerter have brought me to a higher plane of understanding and academic maturity. For that, I am grateful.

This work was supported in part by the National Science Foundation, award number AST 1106218.

Table of Contents

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Phased-Array Feeds 1

1.2 Synthesis Imaging Arrays 2

1.3 Radio-Frequency Interference Mitigation 3

1.4 Problem Statement 3

1.5 Related Work 4

1.6 Thesis Contributions 6

1.7 Thesis Outline 7

2 Background 8

2.1 Phased-Array Feeds 8

2.1.1 PAF Beamforming 8

2.1.2 PAF Calibration Procedure 10

2.1.3 Sensitivity and System Noise Temperature 12

2.2 Synthesis Imaging Arrays 13

2.2.1 Synthesis Imaging Equation 13

2.3 Interference Mitigation Techniques 16

2.3.1 Subspace Projection 17
2.3.2 Cross-Subspace Projection 18
2.3.3 Motion and Bias Correction 20
2.3.4 Oblique Projection 21

3 Development of PAF Data Acquisition System 23

3.1 Introduction 23
3.2 Analog Receiver Cards 23
3.3 Processing Platform 24
3.4 CASPER Tools 26
3.5 Data Acquisition System 26
3.5.1 F-engine 27
3.5.2 Packetization 28
3.6 Packet Capture 31
3.7 Additional Operational Modes and System Specifications 33
3.8 Arecibo Deployment 34
3.8.1 Sensitivity Grid and System Temperature 34
3.8.2 Beam Patterns 35
3.8.3 Computed Mosaic 36
3.9 Conclusion 40

4 PAF-Equipped Interferometer Interference Cancellation Study 42

4.1 Introduction 42
4.2 Australian Square Kilometre Array Pathfinder 42
4.3 Description of Experiment 43
4.3.1 Simulation Model Details 45

4.4 Results 53
4.4.1 18-kHz Processing Bandwidth, 5-ms Dump Time 54
4.4.2 1-MHz Processing Bandwidth, 5-ms Dump Time 58
4.4.3 18-kHz Processing Bandwidth, 5-s Dump Time 58
4.5 Conclusions 61

5 Conclusions and Future Work **64**
5.1 Future Work 65

Bibliography **67**

A Operational Details for x64 System **72**
A.1 Introduction 72
A.2 Hardware 72
A.2.1 Analog Down Conversion Cards 72
A.2.2 ROACH-1 with x64 ADC 74
A.2.3 10-GbE Switch 75
A.2.4 Server PC 78
A.3 PC/ROACH Interface Setup 78
A.3.1 /etc/hosts 80
A.3.2 /etc/dnsmasq.conf 80
A.3.3 /etc/ethers 81
A.3.4 Starting dnsmasq 81
A.4 Running the Data Acquisition System 82
A.4.1 Python Dependencies 82
A.4.2 Gulp 84

A.4.3 Manual Interface 84
A.4.4 Network Interface 85
A.5 Firmware 90
A.5.1 x64 ADC 91
A.5.2 F-Engine 91
A.5.3 Bit Reduction 92
A.5.4 Data Filter 95
A.5.5 UDP Packetizer 95
A.6 Codes 96
A.6.1 correlator.m 97
A.6.2 batch_correlator.m 98
A.6.3 aggregate_grid.m 99
A.6.4 plot_histograms.m 99
A.6.5 plot_specs.m 99

List of Tables

3.1 x64 System FFT Specifications 28

3.2 x64 Packet Header 28

3.3 x64 Data Matrix 29

3.4 x64 System Specifications 33

3.5 Specifications for Arecibo Obervatory 34

3.6 Polarization A 3-dB Beamwidths 38

3.7 Polarization B 3-dB Beamwidths 39

3.8 M87 Facts 39

A.1 Assumed ROACH File Locations 79

A.2 Parameters for dnsmasq.conf 81

A.3 Available Python Commands 87

A.4 Socket-Changeable Parameters 89

A.5 PFB Block Parameters 93

A.6 Reorder Block Behavior 94

A.7 Sample x64 Packet Structure - Four Rows 97

A.8 Sample x64 Packet Structure - Eight Rows 97

List of Figures

2.1 PAFs are calibrated by steering the dish across a strong point source and computing the calibration vectors for each pointing. These are two such example calibration grids. The red dots indicate an "on" pointing, and the blue dots indicate an "off" pointing. The left grid estimates a noise covariance matrix before and after the grid. The right grid estimates it before every row. . . . 11

2.2 Sources of interest are modeled as being on the unit celestial sphere. Therefore, all vectors from the Earth-bound telescope to a point on the source of interest have unit-norm and can be wholly represented by the quantities p and q, which represent a set of coordinate axes that are superimposed onto the celestial sphere and centered at the center of the source of interest. . . . 14

2.3 Cross-subspace projection requires an array of auxiliary antennas that track a number of interferers, while the primary array tracks the source of interest. 19

3.1 A basic single-element feed receiver path traditionally includes a low-noise amplifier (LNA), a bandpass filter, a mixer, an analog-to-digital converter (ADC) and digital signal processing (DSP) modules. 24

3.2 A basic PAF receiver has all of the same elements as a traditional single-horn feed receiver repeated for each PAF element. The DSP block includes additional array signal processing functions such as beamforming and array covariance estimation. 24

3.3 16 of the above cards were used to filter, downconvert, and amplify the PAF signals prior to digitization. 25

3.4 The receiver cards used in conjunction with the x64 DAQ system take L-band RF signals and filter, amplify, and lower-sideband mix them to the second Nyquist zone prior to digitization. Diagram courtesy of BYU student Junming Diao. 25

3.5 The FPGA development board known as ROACH (blue PCB) was used to process PAF data after digitization by the x64 ADC board (green PCB). . . 26

3.6 The x64 system frequency channelizes the sampled data and selects user-defined frequency channels and signal inputs to stream user-defined protocol (UDP) packets to disk through a 10-GbE connection. 27

3.7 Each 18-bit/18-bit real/imaginary sample is sliced down to 8-bits/8-bits. The 8-bit values are computed by first selecting a 9-bit window (the desired eight bits followed by an additional LSB) and adding 1 to the LSB. This creates a rounded 8-bit quantity. 29

3.8 The F-Engine outputs eight 36-bit frequency samples per clock cycle. A 16-bit window of the outputted frequency channels is selected (see Figure A.16) and streamed to the packetizer. Only four of the eight 16-bit samples can be packetized per clock cycle, so the remaining four samples are saved into a FIFO buffer and packetized later. The dark circles represent signal concatenation of four 16-bit samples into a single 64-bit word. 30

3.9 Gulp uses two cores to control reading from the network and writing acquired data to disk. The reader thread writes its captured data into a large ring buffer. The buffer is then emptied by the writer thread and sent to disk. The arrows labeled "start" and "end" represent pointer variables that change with each operation on the ring buffer. 31

3.10 Gulp originally used a single pointer to mark a file boundary (left), which required that each file be about the size of the ring buffer. The addition of an array of file boundary pointers (right) enables the acquisition of multiple smaller files. The blue dots represent occupied bytes in the ring buffer that are awaiting transfer to disk, and the red dots represent occupied bytes that are next to be saved to disk. 32

3.11 After performing a 31×31 calibration grid acquisition on J2232+117, formed-beam sensitivities were computed for each pointing. The left and right plots are polarization-specific sensitivity grids. 35

3.12 The beam patterns for the Cornell feed were measured by applying the maximum SNR beamformer weights for a desired beam direction to every grid pointing covariance matrix. The black curves represent the 3-dB contours. . 37

3.13 M87 is an 8.3×6.6 arcminute source. The right image is one captured by the VLA at 1.5 GHz. The image dimensions are 42.7×42.7 arcminutes with 45 arcsecond resolution. Image courtesy of NASA/IPAC Extragalactic Database. The left image is the 5×5 grid mosaic of pointings around M87. Overlapping pixels were averaged together for the final image. Left image courtesy of Jay Brady. 40

4.1 The ASKAP radio telescope has 36 12-meter dishes that are each equipped with a 96-element dual-polarized PAF. (Left) Three of the 12-meter dishes pointed at zenith. (Right) One of the 192-element PAFs to be mounted on a dish. Photos courtesy of Australia Telescope National Facility (ATNF) and the Commonwealth Scientific and Industrial Research Organisation (CSIRO) 43

4.2 The ASKAP telescope will form 36 overlapping beams to produce a 30 square-degree field of view. The processing of a single beam consists of a beamformer at each PAF whose outputs are correlated with the outputs of each other PAF at the central correlator, producing the visibility function. The visibility function is then resampled onto a rectilinear grid, and the inverse 2D Fourier transform is performed. Image of Cygnus A courtesy of NRAO/AUI. 44

4.3 The ASKAP array is laid out so as to heavily oversample low spatial frequencies and collect high spatial frequencies as the Earth rotates. The right plot depicts the uv-plane for a signal of interest appearing at zenith. 45

4.4 XEphem simulates satellite orbits and astronomical source motion. M87 (yellow) and GSAT0102 (red) are shown here on Julian Date 2456781.93922. . . 46

4.5 Five seconds with one-millisecond position updates were simulated for source of interest M87 and source of interference GSAT0102. 47

4.6 Each PAF was modeled as a 19-element thickened-dipole array on ASKAP's 12-meter dish with $f/D = 0.5$, resulting in this maximum SNR boresight beampattern. 48

4.7 Two elements experience a relative phase shift when a plane wave arrives at an angle offset from zenith. 49

4.8 The propagation distance between feeds can be computed by projecting the relative baseline vectors onto the propagation vector. 51

4.9 Output INRs when mitigating RFI with an 18-kHz processing bandwidth and 5-ms correlator dump time. 55

4.10 Output SIRs when mitigating RFI with an 18-kHz processing bandwidth and 5-ms correlator dump time. 56

4.11 Output SINRs when mitigating RFI with an 18-kHz processing bandwidth and 5-ms correlator dump time. 57

4.12 As processing bandwidth increases, the interference becomes more decorrelated across the interferometer causing shallower null depths. A correlator with a 5-ms dump time was used to generate this plot. 58

4.13 Output INRs when mitigating RFI with a 1-MHz processing bandwidth and 5-ms correlator dump time. 59

4.14 Output SIRs when mitigating RFI with a 1-MHz processing bandwidth and 5-ms correlator dump time. 59

4.15 Output SINRs when mitigating RFI with a 1-MHz processing bandwidth and 5-ms correlator dump time. 60

4.16 Output INRs when mitigating RFI with an 18-kHz processing bandwidth and 5-s correlator dump time. 61

4.17 Output SIRs when mitigating RFI with an 18-kHz processing bandwidth and 5-second correlator dump time. 62

4.18 Output SINRs when mitigating RFI with an 18-kHz processing bandwidth and 5-second correlator dump time. 62

A.1 The x64 system is a back end that digitizes and frequency channelizes received element voltages and saves the resulting samples to disk. The system interfaces with a telescope M&C PC to streamline the entire process. 73

A.2 The x64 receiver cards filter, amplify, and mix the received RF signal down to a 20-MHz passband centered in the second Nyquist zone or 37.5 MHz. . . 73

A.3 Each receiver card processes four PAF element signals. The left image shows the front side of a receiver card with the RF input connectors on the left and IF output connectors on the right. The right image shows all of the cards housed in a rack-mountable cage. 74

A.4 Since the Agilent signal generators cannot generate enough power to supply 64 mixers, a series of amplifiers and splitters are used to distribute the LO signal. 75

A.5 The current BYU LO distribution network. 76

A.6 The LO distribution network and receiver cards requires +12V and +5V respectively. A rack-mountable box with two discrete supplies is used to accommodate this. 76

A.7 The x64 ADC attached to the ROACH board through two ZDOK connectors. 77

A.8 In order to sample in the second Nyquist zone, the ADC anti-aliasing filters must be deactivated. This can be done by removing the capacitors shown in the red box for all eight ADC chips. 77

A.9 The ROACH is programmed and maintained over a CAT5 Ethernet connection. The to-be-saved frequency channels are streamed to the host PC over a CX4/CX4 10-GbE connection. 77

A.10 The ADC needs a high-voltage signal to take it out of a reset mode. The ROACH FPGA firmware provides this signal across a single jumper cable. The left image shows the pin on the ROACH board that the jumper should be connected to, and the right image shows the ADC pin to connect the jumper to. 78

A.11 The switch has several 10-GbE ports that can forward packets to the appropriate PCs and convert CX4 to XFP. 78

A.12 The recommended PC used for the x64 system is a Dell PowerEdge C2100. . 79

A.13 The PC connects to the M&C PC (for socket communications) and ROACH board (for programming and maintenance) across CAT5 Ethernet connections. The PC receives packets from the ROACH across an XFP/SFP+ 10-GbE connection in a 10-GbE network mezzanine card. 79

A.14 The data samples come in demuxed by a factor of four and appear on 16 data lines out of the x64_adc yellow block. 92

A.15 The F-Engine uses the PFB FFT technique to compute frequency channels. It involves a PFB stage that mitigates the channel's spectral leakage, followed by a reordering stage that buffers up time samples for each input. These data are then frequency channelized using an FFT block. 93

A.16 This bit-reduction architecture is repeated for all possible 8-bit windows. The desired window is selected by specifying the value of the "lsb_select" register. 94

A.17 The 10-GbE core buffers up to 1024 64-bit words for packetization and transfer to a specified IP address. 96

Chapter 1

Introduction

At Bell Telephone Laboratories in 1930, a young engineer named Karl Jansky began surveying atmospheric sources of noise and their respective directions of arrival [1]. His observations identified a source of noise that originated from the same right ascension and declination angles (i.e., the same location in space) every day over the course of a year [2]. This simple discovery is widely cited as the birth of radio astronomy.

Radio astronomy, simply put, is the survey of naturally-occurring electromagnetic (EM) sources in space. These extraterrestrial sources are difficult to examine due to harsh observation conditions such as extremely low signal-to-noise ratio (SNR) and relatively strong man-made radio-frequency interference (RFI). Engineers have striven to address these problems by constructing lower-noise receivers and higher-gain antennas. Furthermore, observations can take hours to complete due to the narrow field of view (FOV) that accompanies high-gain antennas. In an attempt to expand telescope FOV, maximize feed SNR, and enhance RFI mitigation, engineers have recently begun developing phased-array feeds (PAFs) [3, 4].

1.1 Phased-Array Feeds

A phased-array feed (PAF) is a group of closely-spaced antenna elements that is placed in the focal plane of a large reflector dish in lieu of a single-element feed. The tight spacing of elements on a PAF allows one to linearly combine the array element voltages to shape and steer the illumination pattern on the dish. By adjusting the illumination pattern on the dish, the on-sky far-field beampattern can be controlled and modified, within the limitations of the dish's optics. This process is known as beamforming. Through beamforming, astronomers

are able to synthesize multiple simultaneous far-field beams, effectively widening the FOV of the feed.

In order to generate multiple beams, each PAF element requires its own receiver path and sampler. This greatly enlarges and complicates the receiver architecture. Additional challenges introduced by PAFs include beam-to-beam variations [5], additional noise due to mutual coupling between elements [6], and appreciably higher data rates.

A substantial amount of work has been done by Warnick et al. to minimize the effects of mutual coupling in a PAF [7, 8]. Beam-to-beam variations have also been addressed by Elmer with a dual-constraint beamformer [5]. And, in an effort to facilitate high data rate processing, a suite of open-source Field Programmable Gate Array (FPGA) modules and development boards have been produced by the Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) [9, 10].

Even with CASPER FPGA real-time processing modules, there is work yet to be done in order to meet the needs of astronomers. These needs include larger bandwidths, finer frequency resolution, and larger numbers of elements. Meeting these requirements requires state-of-the-art hardware, FPGA firmware, and computer software to process the resulting data rates.

1.2 Synthesis Imaging Arrays

Although PAFs are relatively new to the Radio Astronomy community, array processing techniques are commonplace, especially in synthesis imaging arrays. These consist of an array of separated feeds that are used to synthesize extremely fine resolution images, which are computed using the principle of interferometry [11].

There are many Radio Astronomy instruments that employ interferometry for image synthesis, including the Very Large Array (VLA) near Socorro, New Mexico [12], the Allen Telescope Array (ATA) near Redding, California [13], the Low Frequency Array for Radio astronomy (LOFAR) in the Netherlands [14], the Atacama Large Millimeter Array (ALMA) in Chile [15], the Australia Telescope Compact Array (ATCA) near Narrabri, Australia [16], the Westerbork Synthesis Radio Telescope (WSRT) in Westerbork, Netherlands [17], and

the Australian Square Kilometre Array Pathfinder (ASKAP) that is under construction in western Australia [18].

Interferometrs with single-element feeds are widely used, but they still suffer from the aforementioned challenges of poor FOV and RFI corruption. Some modern interferometers are attempting to compensate for these pitfalls by replacing the single-pixel feeds with PAFs [13, 18]. One such interferometer is the ASKAP telescope.

The ASKAP telescope is planned to employ 36 12-meter dishes with 192-element PAFs as the feeds [19]. This particular marriage of PAFs with an interferometer offers great promise of extremely fine resolution on the order of a few arcseconds (when operating around 1421 MHz) with a 30 square-degree FOV. Furthermore, it offers a richer variety of RFI mitigation techniques.

1.3 Radio-Frequency Interference Mitigation

RFI is becoming an increasing problem for radio telescopes, motivating the need for active mitigation algorithms. A conventional method used in removing interference is to estimate the array spatial signature of the RFI using principal component analysis (PCA) and to project the acquired signal onto the orthogonal complement of that estimate [20, 21]. This technique is commonly referred to as subspace projection, and it has mixed results when used in practice [22].

There have been several attempts to improve upon classical subspace projection including the use of auxiliary antennas [22, 23] and motion modeling [24]. Further work has been done to minimize the effects of RFI mitigation on the signal of interest through bias correction [23, 25], spectral bias removal [26, 27], and an oblique projection [28].

1.4 Problem Statement

In order to incorporate scientifically-viable PAFs onto radio telescopes, a significant amount of work must go into enhancing the processing capabilities of digital receivers and improving RFI mitigation techniques for real-world instruments. This thesis aims to address these problems.

To be scientifically viable, a PAF back end must be able to process large analog bandwidths and a large number of inputs. This, in turn, results in very high data rates. A 64-input system capable of processing 20 MHz of analog bandwidth and streaming to disk is presented herein.

With the incorporation of PAFs into a synthesis imaging array such as ASKAP, a re-evaluation of possible RFI mitigation techniques is needed. To some degree, interferometers are capable of mitigating interference without PAFs. With PAFs, however, a richer variety of techniques can be implemented and thus merits exploration. This thesis describes a study of various techniques as performed on the ASKAP-like instrument. The instrument has the same reflector and baseline geometries, but incorporates a theoretical low-gain reference antenna, variable correlator dump rates, and a simplified PAF model.

Lastly, all real-world radio telescopes do not operate under ideal conditions, which can influence the efficacy of RFI mitigation techniques. These non-ideal conditions include moderate processing bandwidths and motion-sensitive slow correlator dump rates.

1.5 Related Work

At Brigham Young University (BYU), previous graduate students Vikas Asthana, Taylor Webb, and Michael Elmer developed PAF digital back ends to support their growing arsenal of PAFs [29, 30, 5]. Asthana and Elmer developed a 20-input back end that could support an analog bandwidth of 450 kHz [29, 5]. The system was developed to test their 19-element single-polarization thickened-dipole PAF.

To handle the large number of inputs, the system digitized the received voltages across five PCs with four-input Adlink analog-to-digital converter (ADC) cards. Once an acquisition was completed, the collected data was streamed to a server PC for aggregation and array processing.

When the BYU group completed their first 19-element dual-polarization array, the previous system needed to be expanded. Webb and Elmer modified the 20-input system to support 40 inputs at the same bandwidth of 450 kHz by adding an additional ADC card to each of the five PCs [30, 5].

Both versions of the 450-kHz system were deployed at the Arecibo Observatory to test their PAFs [30, 5]. The purpose of the deployment was primarily to test PAF processing algorithms and provide proof-of-concept results. While the system produced accurate and stable results, it could not acquire for long periods of time, topping out at approximately 60 seconds of acquisition [5].

While the 450-kHz systems fulfilled the needs of the BYU group, the available bandwidth was far too small to be of use to astronomers. Furthermore, the data-rate limitations were too restricting to allow for long observations.

The ASKAP PAF backend that is to be deployed later this summer is able to process 384 MHz of analog bandwidth across 192 elements [19]. It is also able to form 36 simultaneous beams with its real-time beamformer and correlate in real time with 16-bit/16-bit real/imaginary quantization resolution. The APERture Tile In Focus (APERTIF) PAF receivers used on the WSRT process 300 MHz of bandwidth across 121 elements and form 37 simultaneous beams [31]. The Dominion Radio Astrophysical Observator (DRAO) PHased-Array feed Demonstrator (PHAD) processes 180 elements on variable bandwidths between 4 and 32 MHz [32]. For BYU to have a scientifically-viable PAF back end, a system with significantly more bandwidth and element support is needed.

Much work has also been done in developing interference mitigation techniques for interferometers. Raza et al. demonstrated that when the spatial signature of the interference is known across an interferometer, an orthogonal projection greatly attenuates the RFI [25]. They further showed that through the use of an "eigenfilter" in which the spatial signature of the interference is estimated (also known as subspace projection) the RFI can be canceled for sufficiently strong interference strength.

Jeffs et al. showed that the use of a reference antenna helps estimate the spatial signature of the interference causing the RFI to be canceled for weaker interference strengths [22]. Their results further showed that interferer motion causes shallower cancellation depths.

Leshem et al. provided an analysis of image quality after the application of projection-based algorithms for interferometers [21]. They demonstrated that residual interference is spatially white and thus will not amplify with integration. As such, images that are synthesized post-mitigation can be trusted as accurate with sufficient integration.

All of this work demonstrates very effective RFI mitigation for interferometers. However, these results hinge on a pair of crucial assumptions: (1) the processing sub-bands are sufficiently narrowband to ignore applied bulk time delays across the feeds, and (2) the correlator is able to stream correlation matrices at fast enough rates so that the effects of interferer motion is minimal across short time windows. Such assumptions are rational but do not accurately reflect the limitations of real-world radio telescope receivers.

1.6 Thesis Contributions

The contributions made by the author are summarized in the following list:

- Built upon previous student's FPGA frequency channelization digital signal processing (DSP) module to support three discrete Fourier transform (DFT) lengths.
- Augmented packet sniffer software to support multiple files and a larger RAM buffer.
- Set up and stress tested two high-performance server PCs with 10-GbE data link, real-time data streaming to disk, and FPGA control.
- Created control scripts to interface with Arecibo Observatory M&C system and to control acquisition system.
- Coded a MATLAB depacketizer and correlator to run on parallel cores.
- Created MATLAB codes to compute calibration sets, generate sensitivity grids, and plot formed beampatterns.
- Managed real-data experiment on Arecibo telescope.
- Made codes to model the ASKAP telescope with a theoretical reference antenna.
- Simulated RFI mitigation techniques on the ASKAP telescope using various bandwidths and correlator dump times.
- Analyzed relative performance of RFI mitigation on ASKAP-like instrument.
- Showed that active RFI mitigation is best performed using PAFs when processing moderate bandwidths.

- Showed that fast correlator dump times enhance moving RFI mitigation at the cost of subspace estimation error.
- Showed that active mitigation performed at the interferometer is seldom effective under real-world conditions.

1.7 Thesis Outline

Chapter 2 discusses the underlying theory of PAF beamforming, calibration, and sensitivity calculation. The basics of synthesis imaging using an interferometer are then established. Lastly, a selection of RFI mitigation techniques are discussed, along with motion considerations and bias correction.

Chapter 3 discusses the development of a 20-MHz digital signal processor and packetizer to work in conjunction with a 64-input front end. A summary of the system's performance during a deployment at the Arecibo Observatory in Puerto Rico is then given.

Chapter 4 explores various RFI mitigation techniques as simulated on the ASKAP telescope with a theoretical reference antenna. An analysis of the effects of processing bandwidth and correlator dump time is also given.

Chapter 5 offers a summary of the results described and suggests possible avenues for future work.

Chapter 2

Background

The purpose of this chapter is to provide a working knowledge of the basics regarding phased-array feeds (PAFs), synthesis imaging arrays, and interference mitigation techniques. The goal of this chapter is not to be exhaustive in derivations, but to provide a suitable reference for the remainder of the thesis.

This chapter will first discuss PAF beamforming, calibration, and sensitivity calculation. This will then be followed by a short summary of the synthesis imaging equation and its use in imaging arrays. We will conclude with an introduction to a few projection-based interference mitigation techniques, including a discussion on ways to correct signal corruption.

2.1 Phased-Array Feeds

A phased-array feed (PAF) is a group of closely-spaced elements placed in the focal plane of a large reflector dish. The constituent element radiation patterns can be coherently combined to change the illumination pattern on the dish. By doing so, the feed's far-field radiation pattern, or "beam," is shaped and steered, insomuch as the dish's optics allow.

Multiple simultaneous beams are often formed in order to survey a wide field of view (FOV). This section discusses beamforming methods for a single beam, the calibration procedure used to enable the formation of multiple simultaneous beams, and the effects of beamforming on sensitivity.

2.1.1 PAF Beamforming

In response to the electromagnetic field present at the dish's focal plane, each antenna element of the array produces a voltage signal consisting of a signal and a noise component.

The sampled voltage received across the array at time sample n is represented by the complex base-banded signal

$$\mathbf{x}[n] = \mathbf{x}_s[n] + \mathbf{x}_\eta[n], \tag{2.1}$$

where $\mathbf{x}_s[n]$ and $\mathbf{x}_\eta[n]$ are modeled as complex random vectors whose elements represent the signal and noise components of the corresponding received element voltage at time sample n. When the source of interest is a point source in direction θ_k, this signal model can be rewritten as

$$\mathbf{x}[n] = s[n]\mathbf{a}_k^s + \mathbf{x}_\eta[n], \tag{2.2}$$

where $s[n]$ is a random process that models the instantaneous baseband voltage of the source, and $\mathbf{a}_k^s$ is the normalized array response to a point source arriving from θ_k.

By applying a weighting to the received element voltages and summing them, the PAF's illumination pattern, and thus, the dish's far-field main beampattern can be steered and shaped. This process is known as beamforming [33]. The formed-beam voltage y_k for a beam pointing to θ_k is given by

$$y_k[n] = \mathbf{w}_k^H \mathbf{x}[n] \tag{2.3}$$

$$= s[n]\mathbf{w}_k^H \mathbf{a}_k^s + \mathbf{w}_k^H \mathbf{x}_\eta[n] \tag{2.4}$$

where $\mathbf{w}_k$ is the unit-norm weight vector required to steer a beam to θ_k, and ^{H} is the conjugate transpose operation.

There are several ways to select beamforming weights, each of which achieves a specific goal. One approach is to treat the beamformer as a finite impulse response (FIR) spatial filter and select weights that shape the main beam (inasmuch as the dish's optics allow) and mitigate sidelobe levels [33]. Other approaches use signal statistics to optimize beamformer weights.

One statistically-optimizing approach is to use the Weiner FIR filter, but this requires a reference signal to compute cross-correlations. Another approach is to select weights that

minimize the beamformer output variance subject to a set of linear constraints [34, 35]. This technique is known as linearly-constrained minimium-variance (LMCV) beamforming and was first introduced by Frost [36]. This technique, however, can require computationally-expensive numerical optimization. Weights can also be selected to maximize SNR as demonstrated by Monzingo et al. [37], but this requires *a priori* knowledge of signal and noise covariance matrices. In this thesis, we will only consider the maximum SNR beamformer weights due to their resulting high SNR and the availability of signal and noise covariances through PAF calibration.

The time-averaged formed-beam SNR is given by

$$\begin{aligned} \mathrm{SNR}_k &= \frac{P_{s,k}}{P_{n,k}} \\ &= \frac{E[y_{s,k}^H y_{s,k}]}{E[y_{n,k}^H y_{n,k}]} \\ &= \frac{\mathbf{w}_k^H \mathbf{R}_s \mathbf{w}_k}{\mathbf{w}_k^H \mathbf{R}_n \mathbf{w}_k}, \end{aligned} \tag{2.5}$$

where $P_{s,k}$ is the kth beam signal power, $P_{n,k}$ is the kth beam noise power, $E[\cdot]$ is the expectation operator, $y_{s,k}$ is the kth formed-beam response to a signal, $y_{n,k}$ is the kth formed-beam response to noise, $\mathbf{R}_s$ is the signal covariance matrix, and $\mathbf{R}_n$ is the noise covariance matrix.

The weights that result in the maximum SNR are the solution to the generalized eigenvalue problem

$$\mathbf{R}_s \mathbf{w}_k = \lambda_{\max} \mathbf{R}_n \mathbf{w}_k, \tag{2.6}$$

where $\lambda_{\max}$ is the dominant eigenvalue of $\mathbf{R}_n^{-1}\mathbf{R}_s$ [33].

2.1.2 PAF Calibration Procedure

The generalized eigenvalue problem in (2.6) requires estimates of the signal and noise covariance matrices $\hat{\mathbf{R}}_s$ and $\hat{\mathbf{R}}_n$ respectively. In order to form beams in multiple directions, this must be repeated for each direction θ_k that a beam should point. The process of

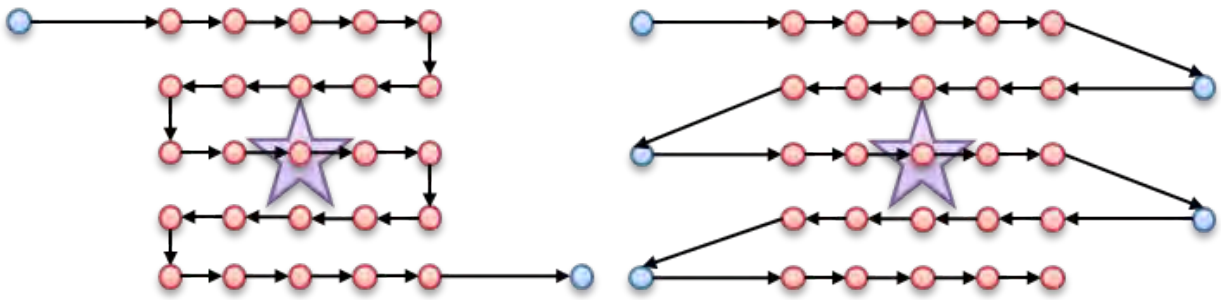


Figure 2.1: PAFs are calibrated by steering the dish across a strong point source and computing the calibration vectors for each pointing. These are two such example calibration grids. The red dots indicate an "on" pointing, and the blue dots indicate an "off" pointing. The left grid estimates a noise covariance matrix before and after the grid. The right grid estimates it before every row.

estimating the signal and noise covariance matrices for each desired pointing and calculating beamformer weights is known as PAF calibration [20, 38].

A PAF is calibrated by steering the dish to a bright, unresolved (i.e. point-like with regard to the beamwidth) source such as Cassiopeia A or Cygnus A [20]. A covariance matrix is estimated for each pointing in a grid pattern centered on the source. These covariance matrices are used to model $\mathbf{R}_s$ and are often denoted as $\hat{\mathbf{R}}_{\text{on},k}$. Another covariance matrix is then estimated by steering the dish off source. This matrix is often denoted as $\hat{\mathbf{R}}_{\text{off}}$ and models the noise covariance matrix $\hat{\mathbf{R}}_n$. The structure of the noise covariance matrix can change with each angle of elevation due to spillover noise structure variations [38], so $\hat{\mathbf{R}}_n$ may be re-calculated for every large change in angle of elevation. Figure 2.1 depicts two example calibration grids.

The calibration vector $\hat{\mathbf{v}}_k$ in the direction k is computed according to

$$\hat{\mathbf{v}}_k = \hat{\mathbf{R}}_n \mathbf{u}_k, \tag{2.7}$$

where $\mathbf{u}_k$ is the dominant eigenvector from the generalized eigenvalue problem,

$$\hat{\mathbf{R}}_{x,k} \mathbf{u}_k = \lambda_{\max} \hat{\mathbf{R}}_n \mathbf{u}_k, \tag{2.8}$$

where $\hat{\mathbf{R}}_{x,k}$ is the "on" pointing covariance matrix in the direction k.

Before using the PAF calibration vectors for observations, the maximum-SNR beamformer weights must be adjusted for local conditions. The adjusted weights are found by obtaining a local (in time and sky position) noise covariance estimate $\hat{\mathbf{R}}_\eta$ and applying its inverse to the set of calibration vectors according to

$$\mathbf{w}_k = \hat{\mathbf{R}}_\eta^{-1}\hat{\mathbf{v}}_k. \tag{2.9}$$

This is the solution to (2.6) when $\mathbf{R}_s$ is rank one, e.g. when the signal of interest is a point source.

2.1.3 Sensitivity and System Noise Temperature

Performance of the instrument, including dish feed, receiver, and beamformer is often evaluated by its sensitivity. A telescope's sensitivity is proportional to the inverse of the minimum flux density it can detect. The sensitivity for a receiving antenna system is defined by

$$S = \frac{A_e}{T_{sys}} \tag{2.10}$$

$$= \frac{k_b B}{S^{sig}}\text{SNR}, \tag{2.11}$$

where $A_e = \eta_{ant}A_p$ is the effective aperture area, η_{ant} is the antenna efficiency, A_p is the physical aperture area, T_{sys} is the system noise temperature, k_b is Boltzman's constant, B is the receiver bandwidth, and S^{sig} is the signal power density (W/m^2) in a single polarization [39]. Recall that SNR is dependent on beamforming weights according to

$$\text{SNR} = \frac{\mathbf{w}^H\mathbf{R}_s\mathbf{w}}{\mathbf{w}^H\mathbf{R}_n\mathbf{w}}.$$

Therefore, sensitivity varies with choice of beamforming weights. Because of this, the expressions "maximum-SNR beamformer" and "maximum-sensitivity beamformer" are often interchanged.

When system electronics contribute little noise, the overall sensitivity increases. With higher sensitivity, smaller signals can be detected with less dwell time. As such, it is a very crucial measurement to make with any PAF.

2.2 Synthesis Imaging Arrays

PAFs are still an up-and-coming instrument in Radio Astronomy, being introduced on single-dish telescopes experimentally over the last couple of decades [40, 41]. Instruments commissioned for astronomical science that use PAF feeds have only appeared in recent months [31, 18, 42]. While they offer many benefits for FOV and RFI mitigation, their imaging capabilities are limited to coarse resolutions. The tried-and-true technology for high-resolution imaging is known as a synthesis imaging array.

Synthesis imaging arrays operate on the principle of interferometry. Interferometry is the analysis of multiple composite signals to produce some meaningful information about the constituent signals. In the case of astronomical synthesis imaging arrays, samples in the spatial frequency domain, known to astronomers as the visibilities in the (u, v) plane, represent the signal-induced cross-correlations between antennas. This relationship is embodied by the synthesis imaging equation, described below.

2.2.1 Synthesis Imaging Equation

In the derivation of the synthesis imaging equation [11], consider axes, labeled p and q, that are superimposed on the unit celestial sphere and centered on a deep space object, as depicted in Figure 2.2. The vector $\mathbf{s}$ extends from the interferometer to a point on the (p, q) axes. Since $\mathbf{s}$ always extends from the center of the celestial sphere, all vectors $\mathbf{s}$ will have unit norm and can thus be represented by a three-dimensional vector defined as $(p, q, \sqrt{1 - p^2 - q^2})$. The vector $\mathbf{s}_0$ points to the center of the source of interest, and is given by the coordinates $(0, 0, 1)$.

The intensity of the source, $I(\mathbf{s})$, is defined as

$$I(\mathbf{s}) = E[|E(\mathbf{s})|^2], \tag{2.12}$$

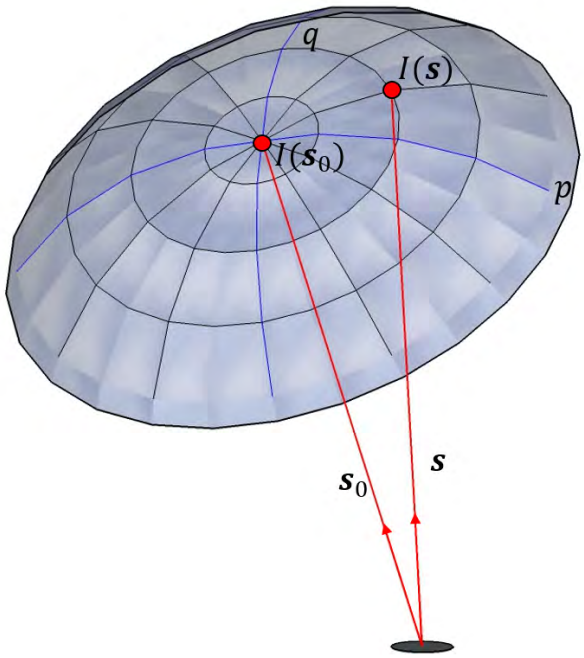


Figure 2.2: Sources of interest are modeled as being on the unit celestial sphere. Therefore, all vectors from the Earth-bound telescope to a point on the source of interest have unit-norm and can be wholly represented by the quantities p and q, which represent a set of coordinate axes that are superimposed onto the celestial sphere and centered at the center of the source of interest.

where $E(\mathbf{s})$ is the electric field at position $\mathbf{s}$. The signal received across the array is given by

$$\mathbf{y}(t) = \int_S A(\mathbf{s})E(\mathbf{s})e^{j(\Omega t+\boldsymbol{\phi}(\mathbf{s}))}d\mathbf{s} + \mathbf{n}(t), \tag{2.13}$$

where S is the celestial sphere, $A(\mathbf{s})$ is the antenna radiation pattern, Ω is the carrier radian frequency, $\mathbf{n}(t)$ is the noise received by each feed, and $\boldsymbol{\phi}(\mathbf{s})$ is the phase shift relative to a reference antenna.

To facilitate the comparison of the phase shift between different antennas, a relative coordinate system (u, v, w) is defined as the array baseline vectors given by

$$\lambda \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} - \begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} = \mathbf{r}_l - \mathbf{r}_m, \tag{2.14}$$

where λ is the observation wavelength, $[x_l, y_l, z_l]^T$ are the Cartesian coordinates of antenna l, and $[x_m, y_m, z_m]^T$ are the Cartesian coordinates of antenna m. The relative phase shift between antennas l and m is given by

$$\phi_l(\mathbf{s}) - \phi_m(\mathbf{s}) = -2\pi(\mathbf{s} - \mathbf{s}_0)^T(\mathbf{r}_l - \mathbf{r}_m). \tag{2.15}$$

The cross-correlation between these two antennas due to a point-source radiator in the imaging field at position (p, q) is therefore given as

$$\begin{aligned} R(\mathbf{r}_l, \mathbf{r}_m) &= R(u, v) \\ &= E[y_l(t)y_m^*(t)] \\ &\approx \iint_{-\infty}^{\infty} |A(p,q)|^2 I(p,q) e^{-j2\pi(up+vq)} dpdq. \end{aligned} \tag{2.16}$$

In this form, which has made a number of simplifying assumptions such as using co-planar antennas, the cross-correlations are recognized as the 2D Fourier transform of $|A(p,q)|^2 I(p,q)$. The intensity can therefore be solved by the inverse relationship

$$I(p,q) = \frac{1}{|A(p,q)|^2} \mathcal{F}_{2\mathrm{D}}^{-1}(R(u,v)), \tag{2.17}$$

where $\mathcal{F}_{2\mathrm{D}}^{-1}(\cdot)$ is the inverse 2D Fourier transform. This is known as the synthesis imaging equation. The quantity $R(u, v)$ is known to astronomers as the visibility function. Therefore, in order to produce accurate images, an accurate and uncorrupted estimate of the array covariance matrix must be found.

Having a large number of antennas with a variety of mutual spacings (the VLA has 271) yields more baselines $\mathbf{r}_l - \mathbf{r}_m$, and thus more $R(u, v)$ samples in the (u, v) plane. Also, by observing the same image field over 12 hours can exploit the Earth's rotation relative to the (u, v) plane to yield many additional unique baseline orientations and denser (u, v) samples. This greatly improves image quality.

2.3 Interference Mitigation Techniques

The presence of an interferer during observations alters the visibility function and thus corrupts synthesized images. This corruption can be minimized through the use of interference mitigation techniques.

There are many forms of interference mitigation including time gating, frequency gating, data-flagging, temporal adaptive filtering, subtraction, and orthogonal projection [22]. This section focuses on orthogonal projection techniques since they drive interference down to true nulls as opposed to becoming buried into the noise floor. Since radio astronomy signals typically have SNR less than unity, interference mitigation techniques must drive interference strength far below the noise floor to prevent corruption.

The goal of projection-based interference mitigation is to project the received signal onto a vector subspace that is orthogonal to the interference but that still roughly spans the signal space. Consider the signal model for a single stationary interferer,

$$\mathbf{x}[n] = \mathbf{a}_s x_s[n] + \mathbf{a}_i x_i[n] + \boldsymbol{\eta}[n], \tag{2.18}$$

where $\mathbf{a}_s$ is the normalized array response to a source of interest, $x_s[n]$ is a random process representing the instantaneous source amplitude, $\mathbf{a}_i$ is the normalized array response to an interferer, $x_i[n]$ is a random process representing the instantaneous interference amplitude, and $\boldsymbol{\eta}[n]$ is the random vector process representing the instantaneous noise amplitude. For an M-element array (PAF or imaging array of dishes), vectors $\mathbf{x}[n]$, $\mathbf{a}_n$, $\mathbf{a}_i$, and $\boldsymbol{\eta}[n]$ are $M \times 1$ and $\mathbf{R}_x$ is $M \times M$. All signals are thus representable as spanning subspaces of an M-dimensional vector space. Since $\mathbf{a}_s$ and $\mathbf{a}_i$ are the spatial signatures of the signal

and interference respectively they can be thought of as basis vectors for the 1-D signal and interference subspaces.

A projection matrix $\mathbf{P}$ can be applied to the received signal $\mathbf{x}[n]$ to cancel the interference if $\mathbf{P} \perp \mathbf{a}_i$ such that $\mathbf{P}\mathbf{a}_i = \mathbf{0}$ so

$$\mathbf{P}\mathbf{x}[n] = \mathbf{P}\mathbf{a}_s x_s[n] + \mathbf{P}\mathbf{a}_i x_i[n] + \mathbf{P}\boldsymbol{\eta}[n] \tag{2.19}$$

$$= \mathbf{P}\mathbf{a}_s x_s[n] + \mathbf{P}\boldsymbol{\eta}[n]. \tag{2.20}$$

Assuming that the source and interference are independent, the projection matrix can also be applied post-correlation using

$$\mathbf{P}\mathbf{R}_x\mathbf{P}^H = \mathbf{P}\mathbf{R}_s\mathbf{P}^H + \mathbf{P}\mathbf{R}_i\mathbf{P}^H + \mathbf{P}\mathbf{R}_n\mathbf{P}^H \tag{2.21}$$

$$= \mathbf{P}\mathbf{R}_s\mathbf{P}^H + \mathbf{P}\mathbf{R}_n\mathbf{P}^H. \tag{2.22}$$

The projection matrix can also be used to form new PAF beamformer weights to achieve the same effect [20, 24]:

$$\mathbf{w}_P = \mathbf{P}\mathbf{w}, \tag{2.23}$$

where $\mathbf{w}$ is the quiescent beamformer weight vector computed with some desired algorithm to achieve a nominal favorable response pattern in the absence of interference. The main challenge in interference mitigation is to find an accurate estimate of the interference subspace $\mathbf{R}_i$ so as to construct an effective projection matrix [21, 22, 43, 44]. Two algorithms for finding such a projection matrix, namely subspace projection and cross-subspace projection, are presented in the following sections.

2.3.1 Subspace Projection

The subpace projection algorithm estimates the array response subspace of an interferer using principal component analysis (PCA). The interference subspace for Q interferers, given that the interference-to-signal ratio (ISR) $\gg 1$, is estimated by finding the Q dominant eigenvectors of the total signal covariance matrix $\mathbf{R}_x$. The eigenvalue decomposition of the

$M \times M$ total signal covariance matrix $\mathbf{R}_x$ is given by

$$\mathbf{R}_x = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^H, \tag{2.24}$$

where $\mathbf{U}$ is an $M \times M$ matrix whose columns are the eigenvectors of $\mathbf{R}_x$, and $\mathbf{\Lambda}$ is an $M \times M$ diagonal matrix whose entries are the eigenvalues of $\mathbf{R}_x$. The $M \times M$ orthogonal projection matrix $\mathbf{P}$ is given by

$$\mathbf{P} = \mathbf{I} - \mathbf{U}_d\mathbf{U}_d^H, \tag{2.25}$$

where $\mathbf{I}$ is the $M \times M$ identity matrix, and $\mathbf{U}_d$ is the $M \times Q$ matrix of the Q dominant eigenvectors extracted from

$$\mathbf{U} = [\mathbf{U}_d|\mathbf{U}_r], \tag{2.26}$$

where $\mathbf{U}_r$ is the $M \times (M - Q)$ matrix of the remaining eigenvectors.

2.3.2 Cross-Subspace Projection

The cross-subspace projection algorithm is one of the most effective estimators of the interference subspace [22]. This improvement is the result of incorporating an auxiliary antenna to track the Q interferers providing a high-INR copy of the total interference. When correlating the received signal from the observing array with that found at the auxiliary, the interference subspace is found in the dominant eigenvectors.

Figure 2.3 depicts a configuration of primary and auxiliary dishes. The electric field acquired across the entire array is

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_p \\ \mathbf{y}_a \end{bmatrix}, \tag{2.27}$$

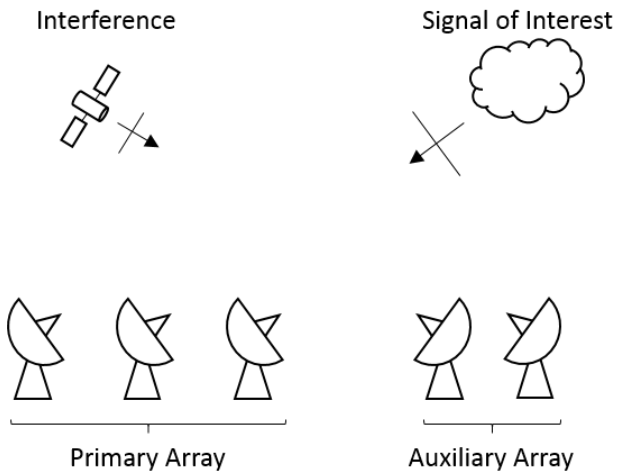


Figure 2.3: Cross-subspace projection requires an array of auxiliary antennas that track a number of interferers, while the primary array tracks the source of interest.

where $\mathbf{y}_p$ and $\mathbf{y}_a$ are the received electric fields at the primary and auxiliary arrays respectively. The total covariance matrix is given by

$$\begin{aligned}\mathbf{R}_y &= E[\mathbf{y}\mathbf{y}^H] \\ &= \begin{bmatrix} E[\mathbf{y}_p\mathbf{y}_p^H] & E[\mathbf{y}_p\mathbf{y}_a^H] \\ E[\mathbf{y}_a\mathbf{y}_p^H] & E[\mathbf{y}_a\mathbf{y}_a^H] \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_{pp} & \mathbf{R}_{pa} \\ \mathbf{R}_{ap} & \mathbf{R}_{aa} \end{bmatrix}. \end{aligned} \tag{2.28}$$

The singular value decomposition of $\mathbf{R}_{pa}$ is given by

$$\mathbf{R}_{pa} = \mathbf{U}\mathbf{S}\mathbf{V}^H. \tag{2.29}$$

Since the auxiliary array tracks the interference, its SIR is dramatically less than unity, causing the interference subspace to be spanned by the dominant eigenvectors from $\mathbf{U}$. The projection matrix is thus given by

$$\mathbf{P}_{\text{CSP}} = \mathbf{I} - \mathbf{U}_d\mathbf{U}_d^H, \tag{2.30}$$

where the columns of $\mathbf{U}_d$ are the M dominant eigenvectors of $\mathbf{U}$. The projection matrix can then be applied to the primary array covariance according to

$$\hat{\mathbf{R}}_s = \mathbf{P}_{\text{CSP}} \hat{\mathbf{R}}_{pp} \mathbf{P}_{\text{CSP}}^H. \tag{2.31}$$

2.3.3 Motion and Bias Correction

When an interferer moves relative to the pointing direction of the primary array, its spatial signature changes, causing the interference covariance estimate to smear. This smearing effect diminishes the efficacy of the projection matrix. To compensate, a subspace estimate is computed for each short-time integration window k across which the interference is approximately stationary. The total signal covariance across K short-time integration windows is given by

$$\hat{\mathbf{R}}_s = \frac{1}{K} \sum_{k=0}^{K-1} \mathbf{P}_k \hat{\mathbf{R}}_{x,k} \mathbf{P}_k^H \tag{2.32}$$

$$= \frac{1}{K} \sum_{k=0}^{K-1} \hat{\mathbf{R}}_{s,k}, \tag{2.33}$$

where $\mathbf{P}_k$ is the projection matrix for window k, and

$$\hat{\mathbf{R}}_{x,k} = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{x}[n+Nk] \mathbf{x}^H[n+Nk], \tag{2.34}$$

where N is the length of the short-time integration window.

Since the signal and interference subspaces are rarely orthogonal, the application of a projection matrix introduces a bias into the total signal covariance estimate [21]. The bias-corrected covariance matrix is given by

$$\hat{\mathbf{R}}_s = \text{unvec} \left\{ \mathbf{C}^\dagger \text{vec} \left\{ \frac{1}{K} \sum_{k=0}^{K-1} \hat{\mathbf{R}}_{s,k} \right\} \right\}, \text{ and} \tag{2.35}$$

$$\mathbf{C} = \sum_{k=0}^{K-1} \mathbf{P}_k^H \otimes \mathbf{P}_k, \tag{2.36}$$

where vec$\{\cdot\}$ creates an $M^2 \times 1$ vector by stacking the columns of an $M \times M$ matrix, † is the pseudo inverse operator, unvec$\{\cdot\}$ is the inverse operator of vec$\{\cdot\}$, and $\otimes$ is the Kronecker matrix product. The pseudo inverse is computed using the singular value decomposition with the smallest singular values set to zero and is used in lieu of a conventional inverse in the event that $\mathbf{C}$ is ill-conditioned, such as when there is insufficient interference motion [22].

2.3.4 Oblique Projection

As stated previously, the signal and interference subspaces are rarely fully orthogonal, causing a bias to be introduced into the visibility function when using orthogonal projections to mitigate RFI. An alternative approach to bias correction is to implement an oblique projection [45, 46, 28]. Such a projection will cancel the interference while preserving the response to the signal of interest. It will not, however, fully restore the array average beampattern, which bias correction does [20, 26, 27]. Oblique projection can be applied in the absence of interference motion.

In order to perform an oblique projection, the basis vectors for the signal and interference subspaces ($\mathbf{a}_s$ and $\mathbf{a}_i$) must be known. These are estimated using the techniques described in Sections 2.3.1 and 2.3.2. The signal subspace basis vector is a direct result from array calibration. It is given by the maximum SNR beamformer weights for a PAF, and it is found for an interferometer by applying the corrective phase gradients across the array after undergoing a self-cal.

Let $\mathbf{S}_s$ be the signal subspace and $\mathbf{S}_i$ be the interference subspace. The oblique projection operator $\mathbf{P}_{\text{ob}}$ is found by defining its range to be the signal subspace and its kernel to be the interference subspace, or

$$\mathbf{P}_{\text{ob}}\mathbf{S}_s = \mathbf{S}_s, \text{ and} \tag{2.37}$$

$$\mathbf{P}_{\text{ob}}\mathbf{S}_i = \mathbf{0}. \tag{2.38}$$

The oblique projection matrix is then given by

$$\mathbf{P}_{\text{obj}} = \mathbf{S}_s(\mathbf{S}_s^H \mathbf{P}_{\mathbf{S}_i}^{\perp} \mathbf{S}_s)^{-1} \mathbf{S}_s^H \mathbf{P}_{\mathbf{S}_i}^{\perp}, \tag{2.39}$$

where $\mathbf{P}_{\mathbf{S}_i}^{\perp}$ is the orthogonal projection matrix found using the techniques described in Sections 2.3.1 and 2.3.2.

Chapter 3

Development of PAF Data Acquisition System

3.1 Introduction

Compared to a single horn feed, phased-array feeds (PAF) require a fairly involved back-end receiver and processing architecture. PAFs have many antenna elements, each of which requires a dedicated analog receiver path. An example block diagram of a simplified traditional receiver for a single element is shown in Figure 3.1, and a PAF receiver is shown in Figure 3.2. The basic function of the receivers depicted in these two illustrations is to amplify, filter, downconvert, and digitize the received electric field.

The digitized signal can then be processed using digital signal processing (DSP) techniques. DSP techniques include frequency channelization, spectrometry, beamforming, correlating, pulsar despreading, detection processing, and data archiving. The development of a 20-MHz DSP module for data acquisition, known as the x64 DAQ system, for a 64-element PAF is presented in this chapter.

3.2 Analog Receiver Cards

While not directly relevant to the DSP system, an understanding of the analog signal path prior to digitization is necessary for downstream analysis. A set of 16 receiver cards with four receiver paths each were designed by prior students Vikas Asthana and Mike Elmer [29, 5]. Figure A.3 shows a photo of a fully-assembled receiver card.

Each receiver path takes an L-band RF signal (between 1200 and 1800 MHz) as input. This RF signal is then bandpass filtered, amplified, and lower-sideband down mixed, causing the spectrum to flip. After this first mixing stage, additional filtering and amplification takes place, followed by the second mixing stage. Lastly, the signal is further amplified and bandpass filtered to a 20-MHz passband centered at an intermediate frequency (IF) of 37.5

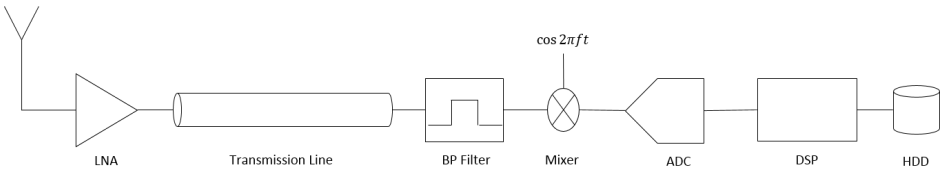


Figure 3.1: A basic single-element feed receiver path traditionally includes a low-noise amplifier (LNA), a bandpass filter, a mixer, an analog-to-digital converter (ADC) and digital signal processing (DSP) modules.

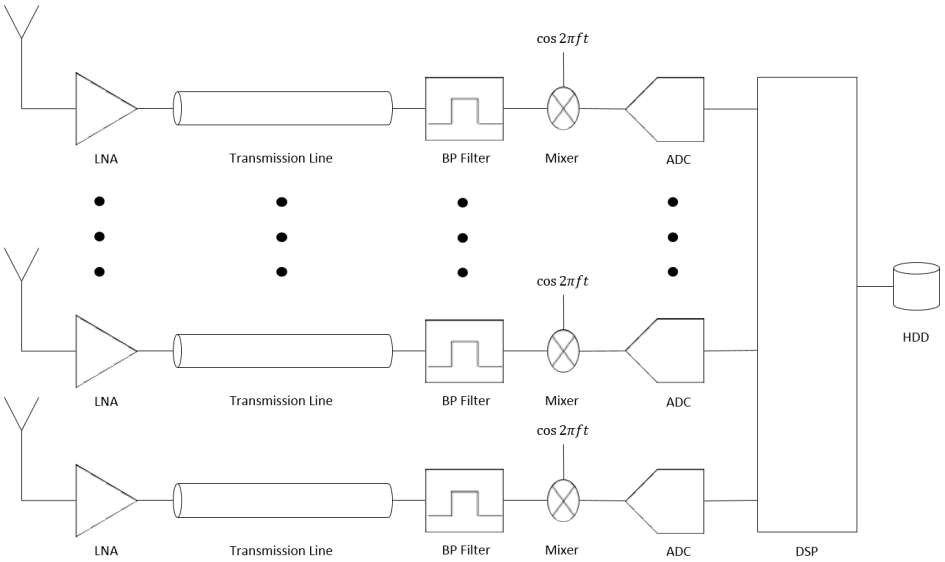


Figure 3.2: A basic PAF receiver has all of the same elements as a traditional single-horn feed receiver repeated for each PAF element. The DSP block includes additional array signal processing functions such as beamforming and array covariance estimation.

MHz and passed to the analog-to-digital converters (ADCs) for digitization. Figure A.2 depicts a block diagram of the entire analog receiver path. Since the signal is centered at 37.5 MHz, it is in the second Nyquist zone, which aliases to baseband upon digitization. This also causes the spectrum to flip again, returning it back to normal frequency order.

3.3 Processing Platform

The x64 DAQ system was developed on a UC Berkeley Reconfigurable Open-Architecture Computing Hardware (ROACH) board. Figure 3.5 shows a photo of a ROACH board and its companion ADC card known as the x64 ADC, which samples 64 inputs at 50 megasamples-per-second (Msps) with 12-bit resolution. The ADC then streams samples into a field pro-

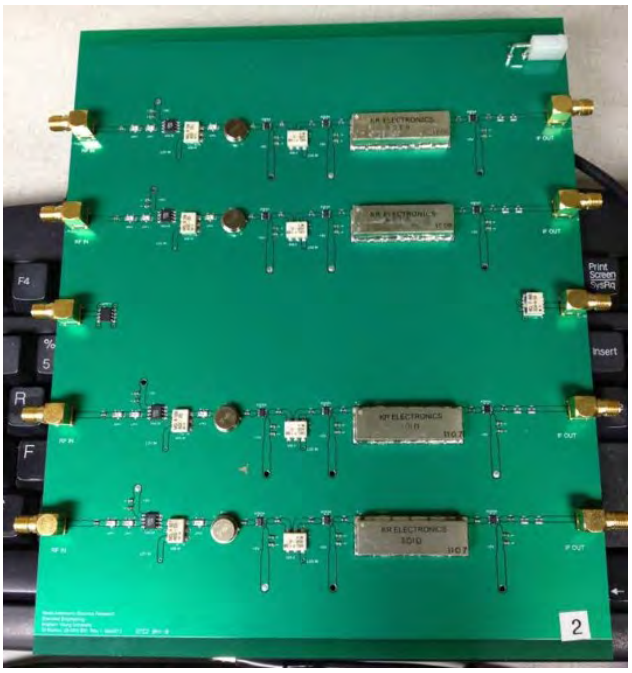


Figure 3.3: 16 of the above cards were used to filter, downconvert, and amplify the PAF signals prior to digitization.

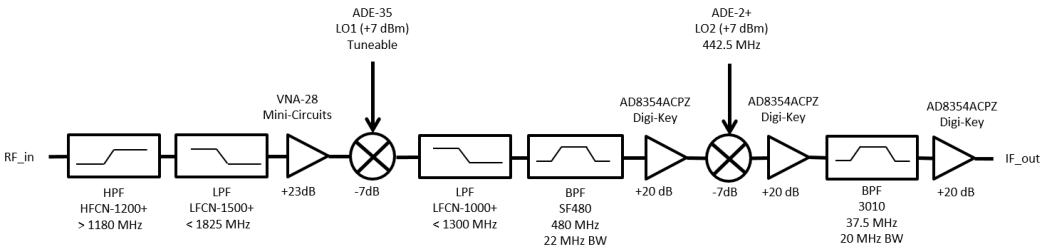


Figure 3.4: The receiver cards used in conjunction with the x64 DAQ system take L-band RF signals and filter, amplify, and lower-sideband mix them to the second Nyquist zone prior to digitization. Diagram courtesy of BYU student Junming Diao.

grammable gate array (FPGA) for processing. Processed data from the FPGA are then streamed across a 10-gigabit Ethernet (10-GbE) connection. The processing firmware was developed using Xilinx System Generator modules and open-source IP cores provided by the Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) group that developed the ROACH boards [9].

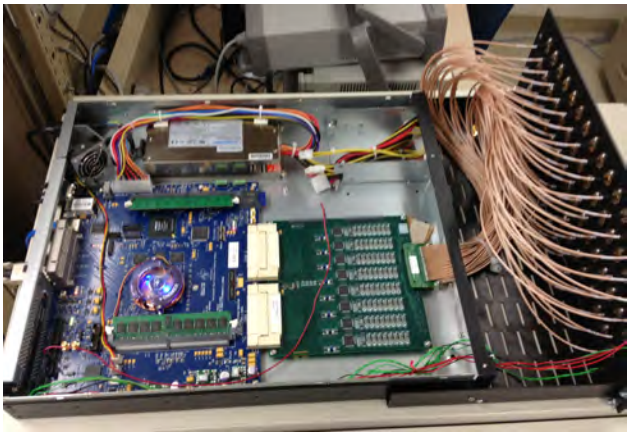


Figure 3.5: The FPGA development board known as ROACH (blue PCB) was used to process PAF data after digitization by the x64 ADC board (green PCB).

3.4 CASPER Tools

The CASPER group aims to enhance and simplify radio astronomy digital processors by providing open-source DSP modules for ROACH FPGA development [10]. They have developed libraries that perform several common operations such as a fast Fourier transforms (FFTs), polyphase filter banks (PFBs), and correlators [10]. The CASPER group also designs modules for communication with hardware such as ADCs, 10-GbE network interface controllers (NICs), and external registers. These modules can be used in tandem with Xilinx System Generator tools in the Simulink graphical development environment. A Simulink model can then be synthesized into an executable bitstream for download onto a ROACH board FPGA.

3.5 Data Acquisition System

The acquisition of 64 simultaneous 12-bit signals arriving at 50 Msps would require a data link running at 38.4 gigabits-per-second (Gbps), which far exceeds the write speed of high-speed random array of independent disks (RAID) drives. To reduce the data load, the x64 system frequency channelizes the data to allow the user to select an adjustable bandwidth and a user-selected number of the 64 inputs to stream to disk. When streaming all 64 ADC inputs, the maximum recordable bandwidth is 5.76 MHz for raw frequency-channelized data. Figure 3.6 demonstrates the top-level behavior of the x64 system. Other processing modes

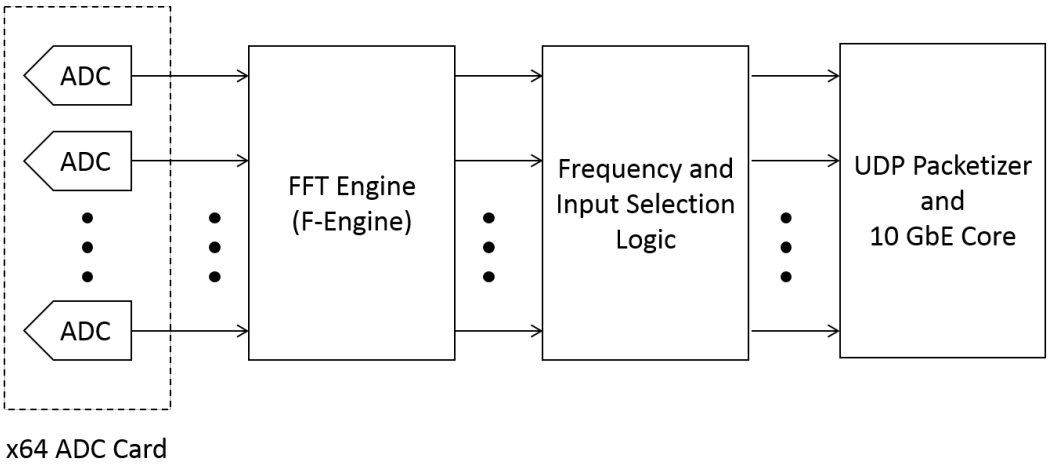


Figure 3.6: The x64 system frequency channelizes the sampled data and selects user-defined frequency channels and signal inputs to stream user-defined protocol (UDP) packets to disk through a 10-GbE connection.

such as real-time beamforming and correlation reduce the total data rate allowing the total 20-MHz analog bandwidth to be recordable.

3.5.1 F-engine

Due to the finite time window used to perform an FFT, estimated spectra suffer from spectral leakage. One approach to reduce this leakage is to use a PFB in tandem with an FFT [9]. This approach is commonly referred to as the PFB technique, the weighted overlap-add technique, and the window pre-sum-FFT technique. The PFB technique first windows a long data set (say 4*NFFT). The longer data window, coupled with the window taper, allows for narrower transition bands and lower sidelobes. The data window is then divided into FFT-length chunks and summed point-by-point. This effectively aliases the data set in the time-domain, which effects decimation in the frequency domain. Frequency channels are then produced by performing an FFT on the resulting data set.

The x64 system uses the decimation-by-four PFB technique with a Hamming window to significantly improve out-of-band signal rejection by driving sidelobes down and narrowing the transition bands at the expense of buffering additional time windows. This entire processing chain is known as the F-engine.

Table 3.1: x64 System FFT Specifications

FFT Length	# Channels	Channel Width	Maximum Data Rate*
256	128	195.31 kHz	10 Gbps
512	256	97.66 kHz	6.4 Gbps
1024	512	48.83 kHz	3.2 Gbps

* maximum data rates result from packet size limitations and 10-GbE capacity.

Table 3.2: x64 Packet Header

Bit Index in Header	63→54	53→44	43→41	40	39→37	36→34	33→32
Meaning	bin_start	bin_end	row_start	row_flag	col_start	col_end	fft_len

The F-engine is able to perform FFTs with lengths of 256, 512, and 1024 and exploits symmetry to output 128, 256, and 512 18-bit/18-bit real/imaginary frequency channels respectively. This corresponds to channel bandwidths of 195.31, 97.66, and 48.83 kHz respectively. Table 3.1 summarizes these specifications.

3.5.2 Packetization

Samples for a single time window are inserted into a user-defined protocol (UDP) packet for transfer. Each packet consists of a single-time window FFT output across user-specified frequency channels and input ports. The user may select ranges of frequency channels and input ports for streaming. These parameters are pre-pended to every packet in the form of a 64-bit header. Table 3.2 shows the information that the packet header specifies.

The input ports are indexed using a row-column scheme since the ADC and F-Engine firmware data path architectures demux the ADC outputs by a total factor of eight to create eight parallel bit streams. Table 3.3 illustrates the order in which signals are available at the output of the F-Engine. The rows of the table represent the eight outputs of the F-Engine. The F-Engine outputs all of the frequency channels for the first column then the second column, and so forth. In other words, ADC sample streams are multiplexed across eight serial data paths in the FPGA. The F-Engine and packetizer data structures are dependent on this.

Table 3.3: x64 Data Matrix

	Time →										
Available Inputs	0	8	2	10	4	12	6	14	0	8	...
	1	9	3	11	5	13	7	15	1	9	...
	16	24	18	26	20	28	22	30	16	24	...
	17	25	19	27	21	29	23	31	17	25	...
	32	40	34	42	36	44	38	46	32	40	...
	33	41	35	43	37	45	39	47	33	41	...
	48	56	50	58	52	60	54	62	48	56	...
	49	57	51	59	53	61	55	63	49	57	...

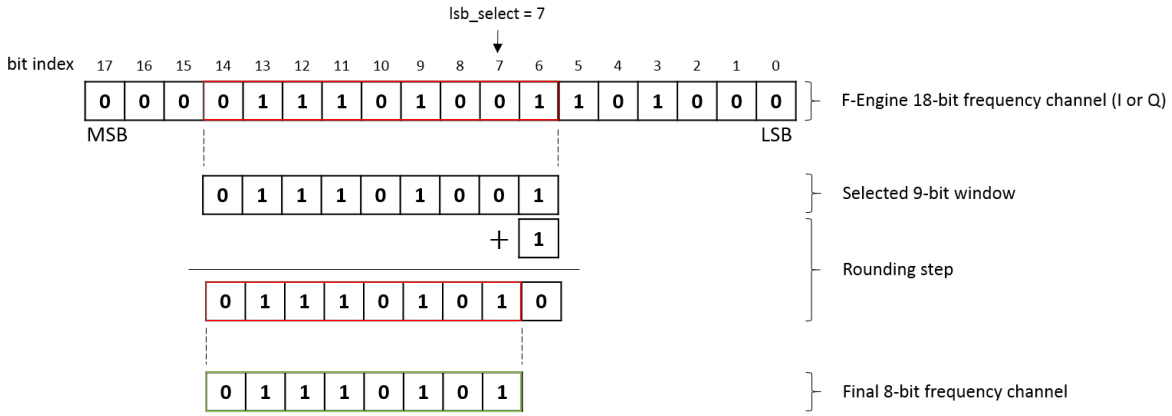


Figure 3.7: Each 18-bit/18-bit real/imaginary sample is sliced down to 8-bits/8-bits. The 8-bit values are computed by first selecting a 9-bit window (the desired eight bits followed by an additional LSB) and adding 1 to the LSB. This creates a rounded 8-bit quantity.

Prior to packetization, each frequency sample is sliced from 36 bits (18 bits real and 18 bits imaginary) to 16 bits (8 bits real and 8 bits imaginary). The user can select which eight bits (for both real and imaginary parts) are extracted by specifying the index of the 8-bit window least-significant bit (LSB). The 8-bit window is sliced using conventional slice blocks with rounding, which is performed by adding one to LSB-1. Figure A.16 depicts the bit-reduction step as performed on the real or imaginary part of a 36-bit frequency channel.

The packetizer can only accept a single 64-bit word every clock cycle and a total of 1024 64-bit words per packet, thus allowing for four 16-bit frequency samples per clock cycle and a single 64-bit header and 4092 frequency samples per time window. Every clock cycle, a total of eight frequency samples are available, only four of which can be packetized. To

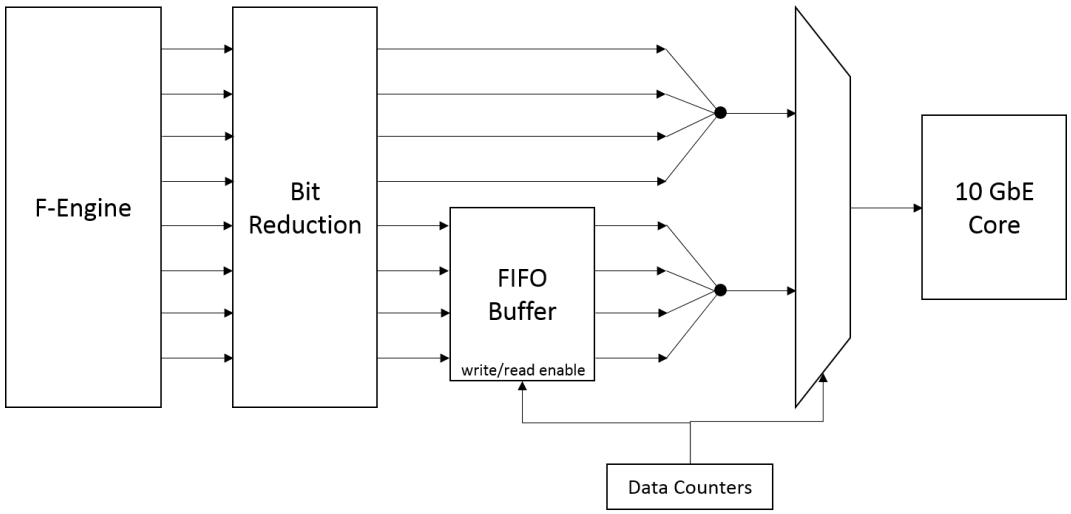


Figure 3.8: The F-Engine outputs eight 36-bit frequency samples per clock cycle. A 16-bit window of the outputted frequency channels is selected (see Figure A.16) and streamed to the packetizer. Only four of the eight 16-bit samples can be packetized per clock cycle, so the remaining four samples are saved into a FIFO buffer and packetized later. The dark circles represent signal concatenation of four 16-bit samples into a single 64-bit word.

save the remaining samples, a first-in-first-out (FIFO) buffer is used on four of the output data lines and is flushed into the packet I/O buffer when the number of frequency bins for the first four outputs have been packetized. Figure 3.8 illustrates how the FIFO is inserted into the data path. This results in a reduction of the total available bandwidth by two when acquiring more than four rows of input ports (from Table 3.3). To save all eight rows, the `row_flag` bit in Table 3.2 is set to 1. If the bit is set to 0, the four rows starting from `row_start` are saved.

The total bit rate depends on the time it takes to process a single time-window and the size of a packet. The bit rate in terms of bits per second into the 10-GbE core I/O interface is given as

$$\begin{aligned} r_b &= \frac{N_b f_s}{N_{\text{fft}}} \\ &= \frac{16(N_i N_c + 4) f_s}{N_{\text{fft}}}, \end{aligned} \tag{3.1}$$

where N_b is the number of bits in the packet, f_s is the ADC sample rate, N_{fft} is the length of the FFT operator, N_i is the number of selected inputs, and N_c is the number of selected

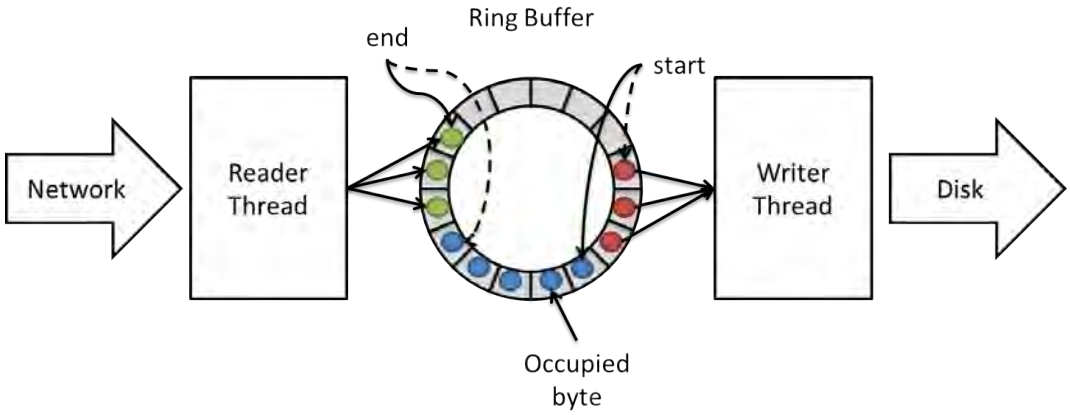


Figure 3.9: Gulp uses two cores to control reading from the network and writing acquired data to disk. The reader thread writes its captured data into a large ring buffer. The buffer is then emptied by the writer thread and sent to disk. The arrows labeled "start" and "end" represent pointer variables that change with each operation on the ring buffer.

frequency channels. The maximum achievable bit rates (seen in Table 3.1) are constrained by the maximum size of a packet, or 1024 64-bit words.

3.6 Packet Capture

To meet the demands of a high bit rate on the 10-GbE link, a high-end server PC and an efficient "packet sniffer" software utility are used. The PC is a Dell PowerEdge C2100, which is equipped with four Intel Xeon CPUs with four cores each, 192 GB of RAM, and two 11-TB striping RAID0 drives (i.e., a disk array that allows data to be segmented and written to multiple independent disks simultaneously). The system employed custom packet sniffing software "Gulp," which is an adaptation from code made available by Corey Satten at the University of Washington [47].

Satten's original code was designed to capture data on 1-GbE links with a 32-bit PC architecture. It utilizes multiple cores in a parallel processing mode to separate the nearly independent tasks of reading from the data link and writing the data to disk. It makes the tasks completely independent by using a large intermediate "ring buffer" or queue structure in main RAM memory. Figure 3.9 depicts the high-level operation of Gulp.

To meet the demands of a 10-GbE link, the ring buffer needs to be large. On a 32-bit architecture, a maximum of 4 GB of memory can be addressed, which makes 188 GB of

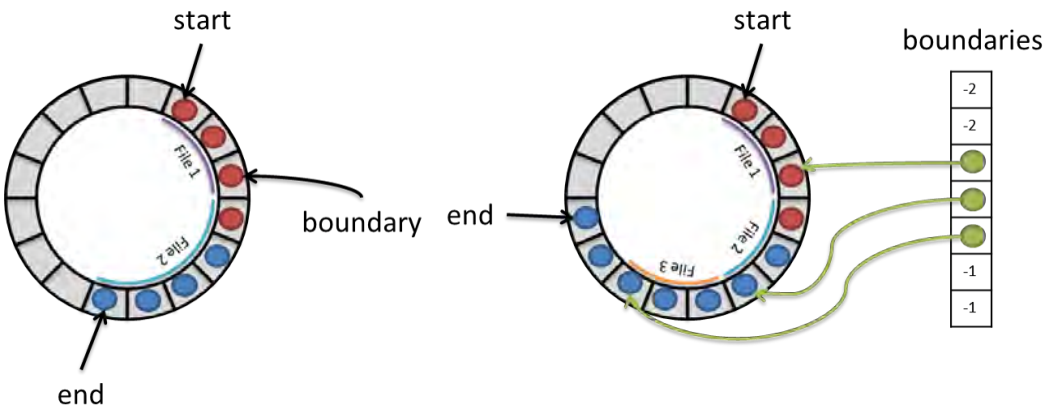


Figure 3.10: Gulp originally used a single pointer to mark a file boundary (left), which required that each file be about the size of the ring buffer. The addition of an array of file boundary pointers (right) enables the acquisition of multiple smaller files. The blue dots represent occupied bytes in the ring buffer that are awaiting transfer to disk, and the red dots represent occupied bytes that are next to be saved to disk.

RAM unusable. Gulp was therefore modified to operate on a 64-bit architecture and use `long long` data types.

To calibrate a PAF, hundreds of dish pointings must be acquired in a small time frame, resulting in hundreds of files per calibration. Gulp originally had a framework for multiple file acquisitions, but it required that the size of each file be on the order of the size of the ring buffer due to its use of a single pointer to mark when a new file should be saved. Gulp was modified to use multiple pointers to be able to save multiple files in a single ring buffer. Figure 3.10 illustrates this change.

To further facilitate link integrity, Gulp is run under ideal conditions: four cores are cleared of all processes prior to initialization; Gulp is forced to use the freed cores; and the NIC buffer size is maximized. Under these ideal conditions and using a heuristically-sized ring buffer of 170 GB, the modified Gulp packet sniffer can acquire on a 3.5-Gbps data link indefinitely. This equates to approximately 3.41 MHz of bandwidth when using a 512-length FFT across all 64 inputs. A bit rate of 5.9 Gbps ($\approx$ 5.76 MHz with 64 inputs and 512-length FFT) is sustainable for approximately ten minutes when buffer overflow occurs.

A Fusion-IO solid-state data cache card [48] was installed in the server PC with the intention of accelerating disk-write speeds. The card's drivers were never properly installed

Table 3.4: x64 System Specifications

x64 System Specifications	
Analog Receiver	
Observable RF Range	1180-1825 MHz
Available IF Bandwidth	20 MHz
Available Inputs	64
Total Receiver Gain	58-60 dB
Receiver 1-dB Compression Point	-128 dBm/Hz
Data Acquisition System	
Available Frequency Channel Resolutions	48.83 kHz, 97.66 kHz, 195.31 kHz
Maximum Recordable Bandwidth (64 inputs)	5.76 MHz
Maximum Recordable Bandwidth (40 inputs)	9.96 MHz
Frequency Channel Bit-Width	16 complex (8 real, 8 imaginary)
Real-Time Beamformer	
Number of Simultaneous Beams	7
Available Inputs for Beamforming	64
Frequency Channel Bit-Width	8 complex (4 real, 4 imaginary)
Number of Frequency Channels	256 (512-length FFT)
Beamformer Weight Bit-Width	8 complex (4 real, 4 imaginary)
Accumulation Length	N_{acc} samples
Formed Beam Power Bit-Width	32
Real-Time Correlator	
Frequency Channel Bit-Width	8 complex (4 real, 4 imaginary)
Available Inputs for Correlating	32
Number of Frequency Channels	1024 (2048-length FFT)
Cross-Correlation Sample Bit-Width	8 complex (4 real, 4 imaginary)
Accumulation Length	128*N_{acc} samples
Accumulated Cross-Correlation Sample Bit-Width	128 complex (64 real, 64 imaginary)

N_{acc} is a user-specified accumulation length scale factor.

so the achievable data rates cited previously could potentially be enhanced given proper installation and integration into Gulp.

3.7 Additional Operational Modes and System Specifications

This chapter thus far has only explored one of the operational modes of the x64 system, namely the stream-to-disk mode. A real-time beamformer and correlator were also developed by other BYU students and are thus not discussed in detail here. Performance specifications for the receiver cards and each operational mode of the x64 system are found

Table 3.5: Specifications for Arecibo Obervatory

Arecibo Specifications	
Main Reflector Shape	Fixed Spherical Cap
Dish Diameter	305 m (1000 ft)
Illuminated Area from Dome	213 m × 237 m
Azimuth Slew Rate	24°/min
Zenith Slew Rate	2.4°/min

in Table 3.4. The real-time beamformer was designed by Jay Brady, and the real-time correlator was implemented by visiting researcher Zhu Kai.

3.8 Arecibo Deployment

In late July 2013, the x64 system was deployed at the Arecibo Observatory and was tested with a 19-element dual-polarized PAF created by Cornell University [49]. Specifications for the Arecibo telescope are summarized in Table 3.5. The goal of the experiment was to measure critical performance parameters of the Cornell feed, such as sensitivity, system noise temperature, and beamwidths.

During the experiment, an as-of-yet unexplained problem caused measured power levels to be intermittently higher than possible (e.g. the resulting sensitivities were too large to be possible). Furthermore, several element LNAs failed after the dewar was de-pressurized. As such, the results presented in this section are those that were physically reasonable (e.g., resulting system noise temperature was higher than 35 K) and, thus, are considered true results.

3.8.1 Sensitivity Grid and System Temperature

Multiple calibration grids, as depicted earlier in Figure 2.1 (left), were performed to compute estimates for $\mathbf{R}_s$ and $\mathbf{R}_n$ and, consequently, an estimate for sensitivity. The covariance matrix for each grid pointing was computed using four seconds of accumulation. Figure 3.11 depicts a 31×31 grid of computed sensitivities in both polarizations for the source J2232+117, which has a flux density of 7.2 Jy.

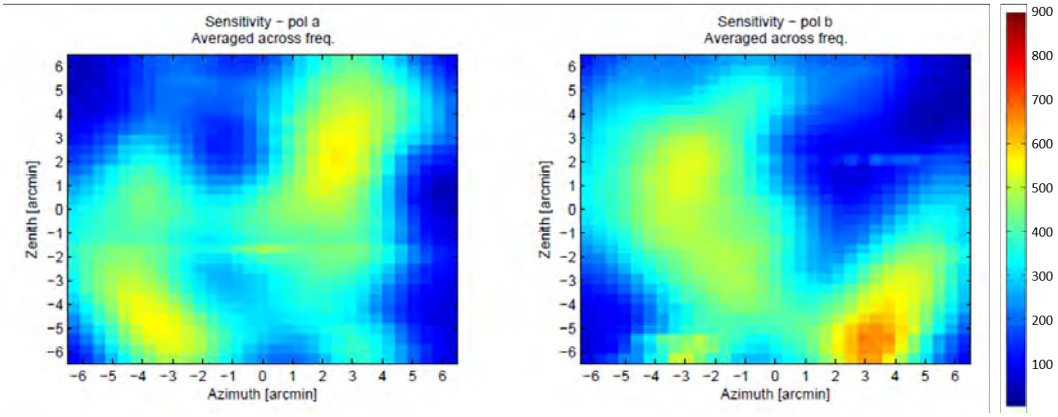


Figure 3.11: After performing a 31×31 calibration grid acquisition on J2232+117, formed-beam sensitivities were computed for each pointing. The left and right plots are polarization-specific sensitivity grids.

In these grids, there are several areas with very low sensitivity due to the failed elements in those locations. The most believable peak sensitivity is approximately 740 m^2/K. Given that the total illuminated area from the Gregorian dome is 39,648 m^2, the system noise temperature T_{sys}/η_{ap} is 53.6 K. This temperature is much higher than the seven-element cluster Arecibo L-Band Feed Array (ALFA) temperatures between 27 and 35 K [50].

3.8.2 Beam Patterns

The 3-dB beamwidth of a length L 1-D line aperture is given approximately by

$$\theta_3 = \frac{\alpha\lambda}{L} \quad \text{(rad)},$$

where α is a beamwidth factor that accounts for the aperture tapering function, and λ is the operating wavelength. Given the illuminated area from the Gregorian dome at Arecibo as stated in Table 3.5, the approximate best-case beamwidths for the Cornell feed operating at 1400 MHz with no taper (i.e. $\alpha = 1$) are expected to be

$$\theta_{\text{major}} = 3.46 \text{ arcmin, and} \tag{3.2}$$

$$\phi_{\text{minor}} = 3.11 \text{ arcmin,} \tag{3.3}$$

where θ_{major} is the 3-dB major axis beamwidth, and ϕ_{minor} is the 3-dB minor axis beamwidth.

A set of beampatterns and corresponding beamwidths was measured using the same calibration grid and source described in the previous section. Images of steered beams were computed by applying the maximum SNR beamformer weights for a single pointing to each acquired pointing according to

$$\text{beam}_i(j) = \mathbf{w}_i^H \mathbf{R}_{s,j} \mathbf{w}_i,$$

where i is the grid pointing index number for the desired beam, j is the pixel location in the image of the steered beam, $\mathbf{w}_i$ are the beamformer weights for grid pointing i, and $\mathbf{R}_{s,j}$ is the signal covariance matrix at grid pointing j. The measured beam patterns for each polarization are depicted in Figure 3.12.

The beamwidths for these patterns were found by fitting an ellipse to the 3-dB contour points and extracting the resulting major and minor axis lengths. Estimated beamwidths for beams formed in Figure 3.12 are shown in Tables 3.6 and 3.7. The center entries (i.e. 0 elevation and 0 cross-elevation) represent the boresight beam, which indicate that the boresight beam beamwidths for polarization A are approximately 3.80×3.45 arcminutes, and the boresight beam beamwidths for polarization B are approximately 3.60×3.31 arcminutes. In both polarizations, the beamwidths are larger than the theoretical widths in (3.2) and (3.3), indicating that the maximum SNR beamformer weights do not fully illuminate the dish.

3.8.3 Computed Mosaic

To further test the feed, a 5×5 grid of pointings was acquired around the elliptical galaxy Messier 87, also known as M87. Table 3.8 summarizes some facts about M87 [51].

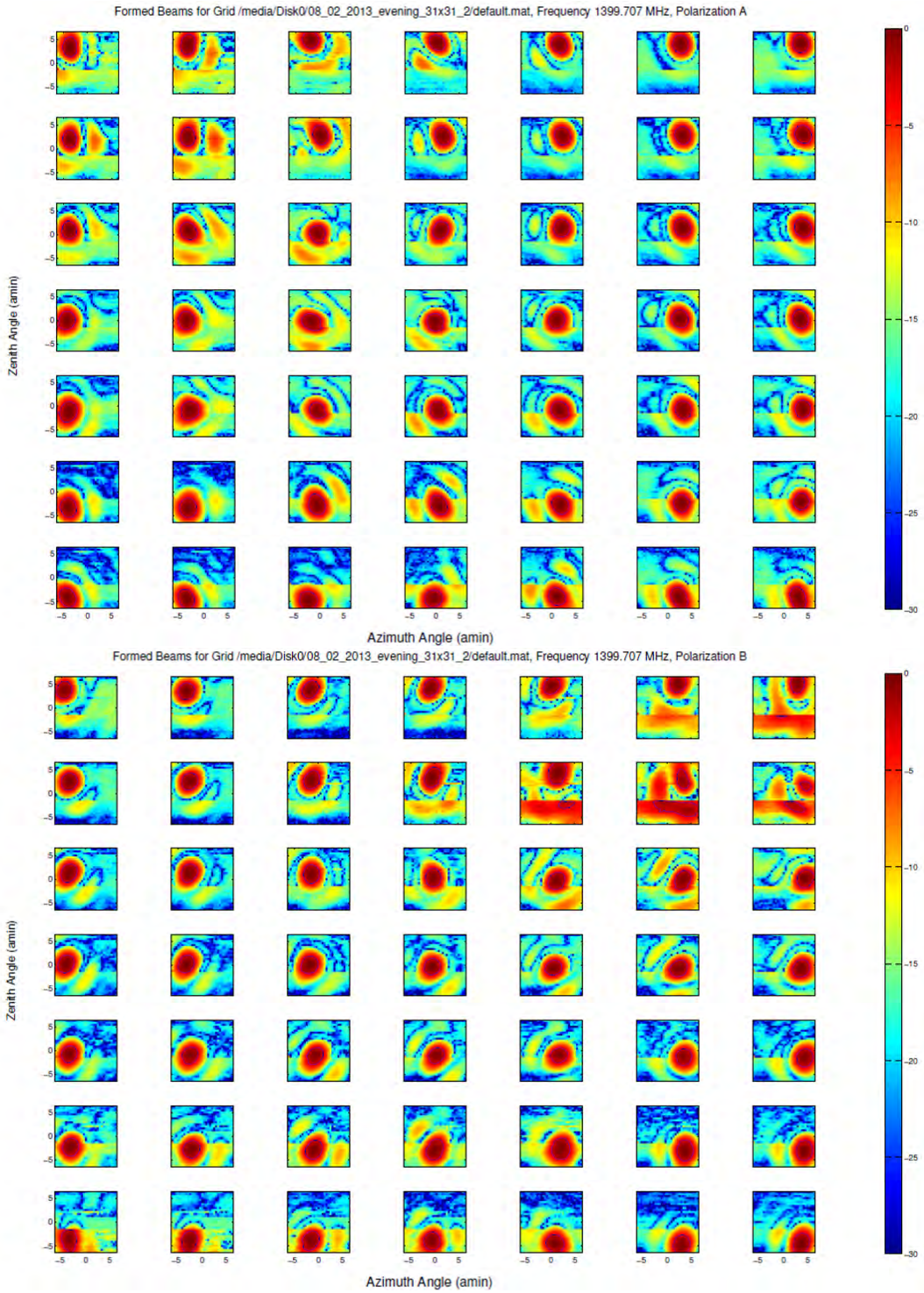


Figure 3.12: The beam patterns for the Cornell feed were measured by applying the maximum SNR beamformer weights for a desired beam direction to every grid pointing covariance matrix. The black curves represent the 3-dB contours.

Table 3.6: Polarization A 3-dB Beamwidths

Polarization A Major Axis Beamwidths

		Cross-Elevation Angle						
		-4.0	-2.8	-1.2	0	1.2	2.8	4.0
Elevation Angle	4.0	4.25	4.47	3.65	3.67	3.76	3.85	3.86
	2.8	3.74	4.05	4.23	3.93	3.86	3.81	3.96
	1.2	3.74	3.95	3.78	3.89	3.73	3.97	4.01
	0	4.05	3.85	4.55	3.80	3.81	3.91	3.94
	-1.2	0	4.06	3.74	3.86	4.06	3.88	3.74
	-2.8	4.15	4.16	4.43	4.40	4.03	3.66	3.53
	-4.0	4.27	4.26	3.92	3.83	3.92	4.75	5.00

Polarization A Minor Axis Beamwidths

		Cross-Elevation Angle						
		-4.0	-2.8	-1.2	0	1.2	2.8	4.0
Elevation Angle	4.0	2.97	3.01	3.35	2.95	3.18	3.35	3.33
	2.8	2.92	3.09	3.10	3.13	3.25	3.32	3.46
	1.2	3.18	3.26	3.26	3.18	3.42	3.30	3.45
	0	3.41	3.43	3.26	3.45	3.43	3.45	3.32
	-1.2	0	3.77	3.02	3.24	3.49	3.32	3.29
	-2.8	3.37	3.60	3.46	3.21	3.37	3.32	3.22
	-4.0	3.23	3.26	3.67	3.53	3.34	3.11	3.16

Measured in arcminutes corresponding to the beam directions depicted in Figure 3.12. Beamwidths of zero are the result of not being able to find a 3-dB contour or finding more than one. All angles are measured in arcminutes.

The mosaic was computed by first creating individual beamformer images $I_j(k)$ according to

$$I_j(k) = \mathbf{w}_k^H \hat{\mathbf{R}}_{s,j} \mathbf{w}_k,$$

where $I_j(k)$ is the image intensity at pixel k in the jth image, $\mathbf{w}_k$ are the beamformer weights for calibration grid pointing k, and $\hat{\mathbf{R}}_{s,j}$ is the signal covariance matrix given by

$$\hat{\mathbf{R}}_{s,j} = \hat{\mathbf{R}}_{\text{on},j} - \hat{\mathbf{R}}_{\text{off}}.$$

The individual images were then overlaid and all overlapping pixels were averaged to produce the final image depicted in Figure 3.13a. For comparison, an image of M87 from the VLA at L-band is presented with a similar FOV in Figure 3.13b. Both images

Table 3.7: Polarization B 3-dB Beamwidths

Polarization B Major Axis Beamwidths

		Cross-Elevation Angle						
		-4.0	-2.8	-1.2	0	1.2	2.8	4.0
Elevation Angle	4.0	3.72	3.67	4.04	3.84	3.95	3.87	0
	2.8	3.69	3.78	4.08	4.48	0	0	0
	1.2	4.12	4.03	3.95	3.97	3.95	4.30	3.98
	0	3.99	4.02	3.94	3.60	3.60	3.91	3.89
	-1.2	3.66	3.77	3.99	3.66	3.64	3.56	3.90
	-2.8	3.87	3.82	4.24	4.66	4.60	4.46	4.23
	-4.0	4.85	4.77	4.24	3.79	4.00	4.07	4.47

Polarization B Minor Axis Beamwidths

		Cross-Elevation Angle						
		-4.0	-2.8	-1.2	0	1.2	2.8	4.0
Elevation Angle	4.0	3.21	3.35	3.07	3.02	3.13	3.30	0
	2.8	3.35	3.30	3.13	3.11	0	0	0
	1.2	3.20	3.24	3.35	3.23	3.30	3.17	3.40
	0	3.23	3.17	3.42	3.31	3.07	3.28	3.45
	-1.2	3.17	3.20	3.08	2.99	3.05	3.30	3.45
	-2.8	3.34	3.46	3.27	3.38	3.84	3.46	3.47
	-4.0	3.09	3.25	3.30	3.45	3.59	3.43	3.28

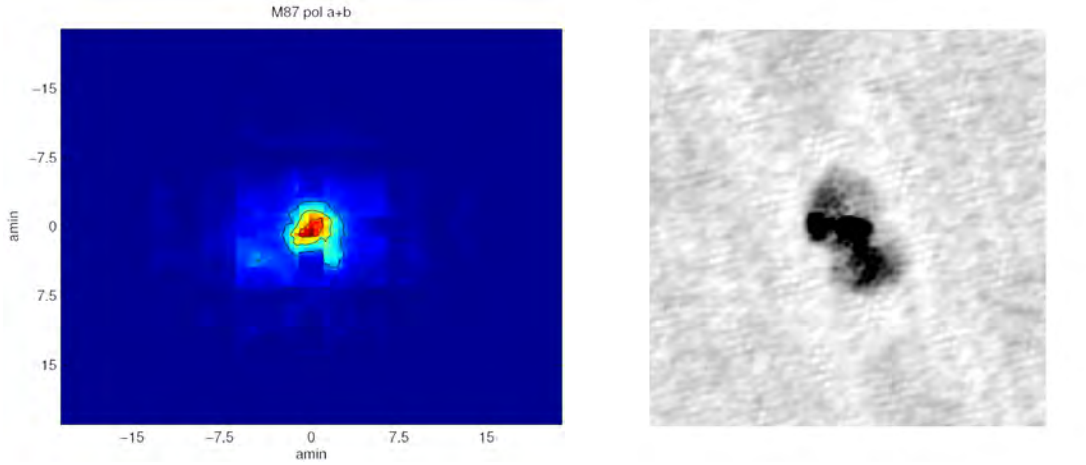
Measured in arcminutes corresponding to the beam directions depicted in Figure 3.12. Beamwidths of zero are the result of not being able to find a 3-dB contour or finding more than one. All angles are measured in arcminutes.

Table 3.8: M87 Facts

M87 Facts	
Galaxy Name	Messier 87
Equatorial Coordinates (Epoch J2000)	J1230+1223
Mean Distance*	54.345×10^6 light years
Major Diameter	8.3 arcminutes
Minor Diameter	6.6 arcminutes

* Computed from estimated distances from 66 literature sources.

of M87 have a high magnitude center with a small tail. The calculated major and minor diameters of M87 from Figure 3.13a are roughly 7.5 and 6.5 arcminutes respectively. The minor diameter is close to that given in Table 3.8, but the major diameter is off by about an arcminute, or about 1/3 of a beamwidth. This discrepancy is likely due to the relatively large beamwidth and consequently coarse spatial resolution of the telescope, whereas the



(a) Computed Mosaic of M87 **(b)** Reference Image of M87

Figure 3.13: M87 is an 8.3×6.6 arcminute source. The right image is one captured by the VLA at 1.5 GHz. The image dimensions are 42.7×42.7 arcminutes with 45 arcsecond resolution. Image courtesy of NASA/IPAC Extragalactic Database. The left image is the 5×5 grid mosaic of pointings around M87. Overlapping pixels were averaged together for the final image. Left image courtesy of Jay Brady.

VLA image benefited from large imaging baselines and fine resolution. However, the x64 system only needed to observe for a handful of minutes compared to the numerous hours needed by an interferometer such as the VLA.

3.9 Conclusion

To accommodate the ever-increasing bandwidth and number of elements for PAF systems under development, feed receivers must operate at higher sampling rates and incorporate more receiver paths. This chapter described the new x64 DAQ digital receiver system that can frequency channelize a 20-MHz bandwidth across 64 analog inputs and stream a portion of the resulting data to disk.

The x64 DAQ system FPGA firmware was developed using the CASPER ROACH environment. All 64 elements are frequency channelized to DFT lengths of 256, 512, and 1024 by using the F-Engine. Using a 512-length F-Engine, up to 59 frequency channels (or 5.76 MHz analog bandwidth) across all 64 ADC inputs can be streamed to disk for 10 minutes on a server PC. A 3.41-MHz bandwidth can be streamed indefinitely. When streaming only 40 ADC inputs with a 512-FFT, up to 9.96 MHz can be streamed to disk.

The system was deployed on the Arecibo Observatory to test the Cornell University PAF. During these tests, the system streamed frequency-channelized samples for PAF calibration, sensitivity grid synthesis, beam pattern analysis, and imaging. With exception to the aforementioned transient behavior, the system demonstrated proper operation and great promise for use in future experiments.

Chapter 4

PAF-Equipped Interferometer Interference Cancellation Study

4.1 Introduction

There has been a substantial amount of promising research on the topic of projection-based radio-frequency interference mitigation for single-element feed interferometers [21, 22, 23, 44]. While many have shown these techniques to be effective, they have made an overarching assumption of a narrow processing bandwidth and a rapid correlator dump rate. When these assumptions do not hold, the efficacy of projection-based RFI mitigation is lessened.

With the recent addition of PAFs to interferometers, a new class of RFI mitigation techniques becomes usable. Due to a relatively small aperture, PAFs can mitigate interference on relatively larger bandwidths and are more tolerant to interference motion than single-element feed interferometers, allowing for slower correlator dump rates. This chapter explores the performance of RFI mitigation techniques on the Australian Square Kilometre Array Pathfinder (ASKAP) instrument. Adaptive array cancellation algorithms can be applied at the PAF, the interferometer central correlator, or both. Each of these cases are studied in this chapter.

This chapter begins with a brief contextual discussion of the ASKAP instrument followed by a description of the simulation used to model the instrument and test various RFI mitigation techniques. The simulation is repeated for different bandwidths and correlator dump times, and results are then presented.

4.2 Australian Square Kilometre Array Pathfinder

The Australia Telescope National Facility is currently constructing an ambitious 36-dish interferometer equipped with 12-meter dishes and 96-element dual-polarized PAFs [18,

Figure 4.1: The ASKAP radio telescope has 36 12-meter dishes that are each equipped with a 96-element dual-polarized PAF. (Left) Three of the 12-meter dishes pointed at zenith. (Right) One of the 192-element PAFs to be mounted on a dish. Photos courtesy of Australia Telescope National Facility (ATNF) and the Commonwealth Scientific and Industrial Research Organisation (CSIRO)

19, 52]. This radio telescope is named the Australian Square Kilometre Array Pathfinder (ASKAP). Photographs of a dish and PAF to be used on ASKAP are shown in Figure 4.1. This phased-array approach allows astronomers to form 36 simultaneous, overlapping $1^\circ \times 1^\circ$ beams for a 30 square-degree field of view (FOV) [18]. With a maximum baseline of 6 km [52], it is also geared to form extremely high resolution images. A block diagram describing the imaging process for a single beam is depicted in Figure 4.2. Figure 4.3 shows the placement of the individual dishes on ASKAP. This distribution lends itself to heavy oversampling of the low spatial frequency samples but with a good range of non-repeating baseline vectors between antennas.

This new approach of using PAFs on an interferometer also provides a unique potential for improved interference mitigation techniques. The techniques shown in Chapter 2 were implemented in simulation at the PAF, central correlator, or both using the ASKAP geometry.

4.3 Description of Experiment

A MATLAB script was used to model a single beam for each of the 36 primary 12-meter dishes with a single theoretical 3.7-meter dish as an auxiliary. Under these parameters, the following interference mitigation techniques were tested (see Section 2.3):

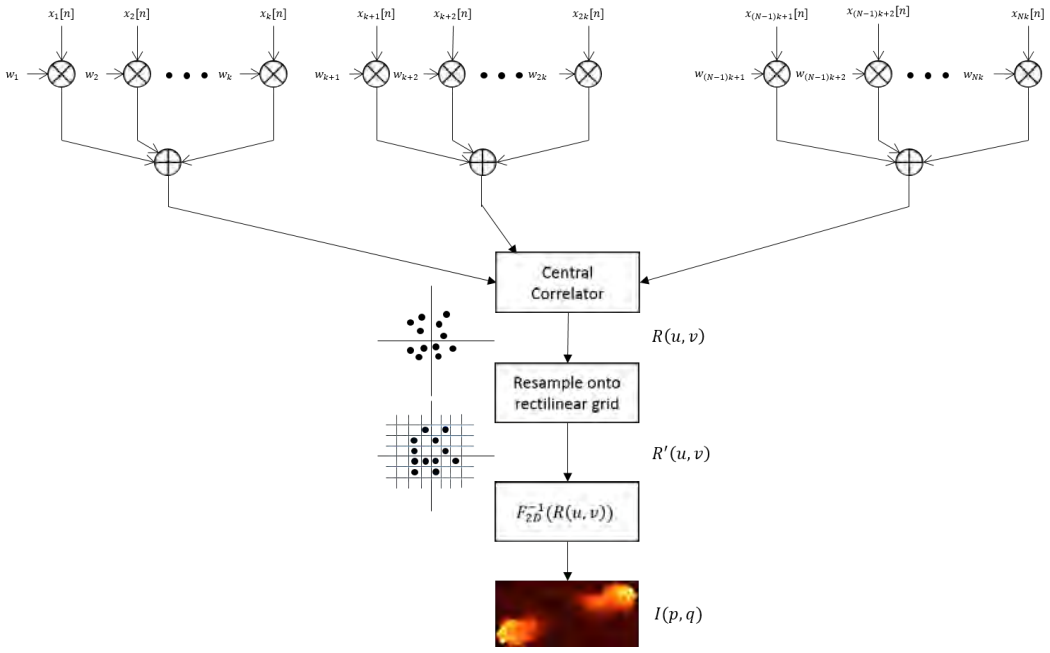


Figure 4.2: The ASKAP telescope will form 36 overlapping beams to produce a 30 square-degree field of view. The processing of a single beam consists of a beamformer at each PAF whose outputs are correlated with the outputs of each other PAF at the central correlator, producing the visibility function. The visibility function is then resampled onto a rectilinear grid, and the inverse 2D Fourier transform is performed. Image of Cygnus A courtesy of NRAO/AUI.

1. Subspace Projection at the PAF (SP-X)

2. Subspace Oblique Projection at the PAF (SOP-X)

3. Cross-Subspace Projection at the PAF (CSP-X)

4. Cross-Subspace Oblique Projection at the PAF (CSOP-X)

5. Subspace Projection at the Central Correlator (X-SP)

6. Cross-Subspace Projection at Central Correlator (X-CSP)

7. Cross-Subspace Projection at PAF and Central Correlator (CSP-CSP)

8. Cross-Subspace Oblique Projection at PAF and Cross-Subspace Projection at the Central Correlator (CSOP-CSP)

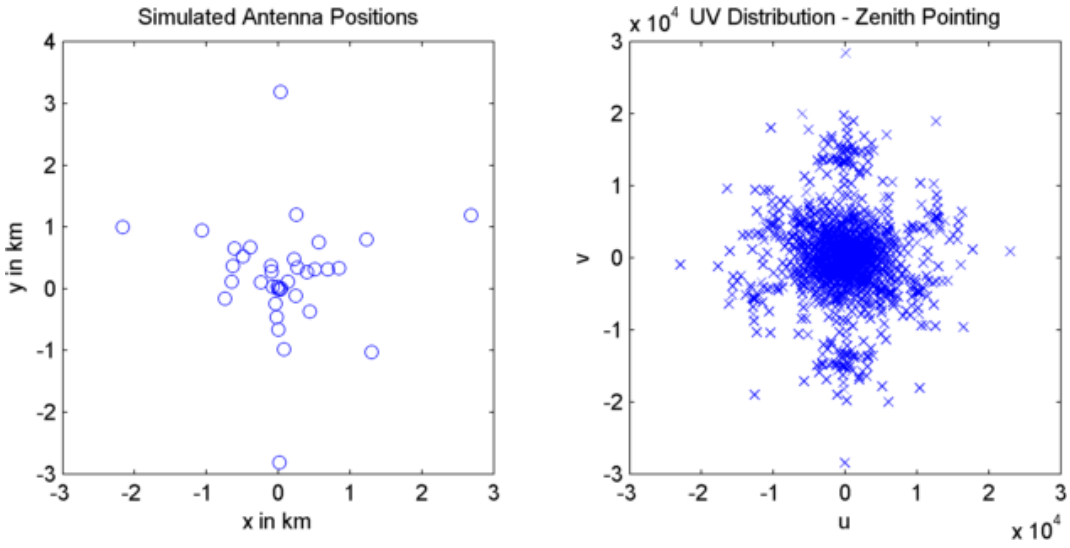


Figure 4.3: The ASKAP array is laid out so as to heavily oversample low spatial frequencies and collect high spatial frequencies as the Earth rotates. The right plot depicts the *uv*-plane for a signal of interest appearing at zenith.

Each technique will be identified using a pair of acronyms separated by a dash. The first acronym designates a mitigation technique that is performed at the PAF, and the second acronym designates a technique that is performed across the interferometer. The acronym "X" indicates that no mitigation was performed at that stage. This acronym format reflects signal propagation in that it is first beamformed at the PAF and then correlated across the interferometer (see Figure 4.2).

4.3.1 Simulation Model Details

There are several real-world considerations that must be made when simulating an interferometer with PAFs. In addition to those described in Chapter 2, the following considerations were incorporated into the simulation:

- Motion of extraterrestrial sources and orbiting satellites
- Reflector and feed gains
- Central correlator dump times
- Bulk time delay insertion

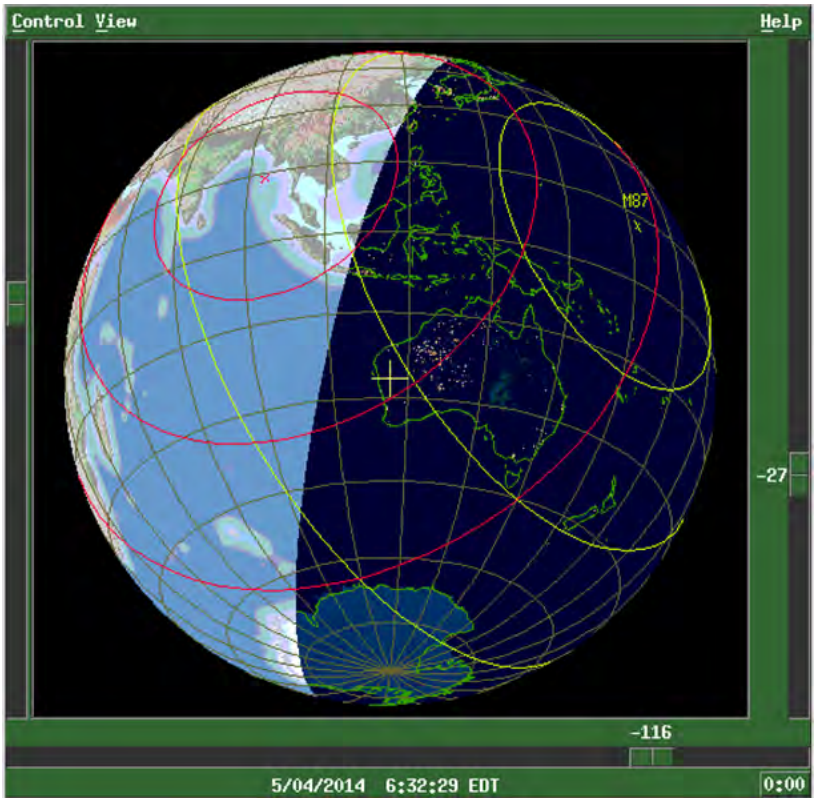


Figure 4.4: XEphem simulates satellite orbits and astronomical source motion. M87 (yellow) and GSAT0102 (red) are shown here on Julian Date 2456781.93922.

This section describes each of these aspects in detail and how they play a role in the simulation.

Source and Satellite Motion

The source of interest used in simulation was galaxy M87, and the interference was one of the four orbiting Galileo satellites. To simulate realistic orbital motion, the XEphem library of utilities was used [53]. A screenshot of XEphem modeling the orbits of M87 and the second Galileo satellite GSAT0102 is shown in Figure 4.4. XEphem can provide azimuthal and elevation tracking of extra-terrestrial sources and orbiting satellites given observational coordinates and a coordinated universal time (UTC). The software tracks by using two-line element sets (TLEs), which specify the Keplarian elements for orbiting sources. A time-series of azimuth and elevation angles can then be calculated and saved to disk.

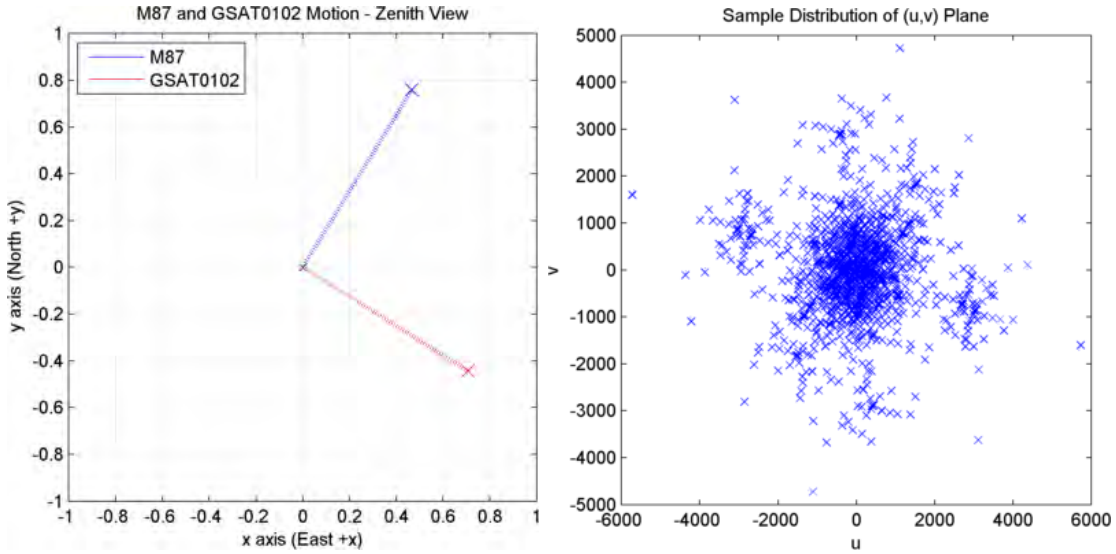


Figure 4.5: Five seconds with one-millisecond position updates were simulated for source of interest M87 and source of interference GSAT0102.

A total of five seconds was simulated using one-millisecond position updates. The interference is assumed to be stationary across each 1-ms step. A depiction of the motion for M87 and GSAT0102 as used in this simulation, as well as the accompanying sample coverage in the (u, v) plane is found in Figure 4.5.

Reflector and Feed Gains

The primary reflectors are modeled as 12-meter parabolic dishes, with 19-element thickened-dipole single-polarization PAFs as the feeds that are positioned at $f/D = 0.5$. This PAF does not match the 192-element checkerboard feed that will actually be mounted on the ASKAP reflectors, but it is arguably sufficient for modeling rough beampatterns. The actual ASKAP feed was not simulated since an accurate feed model was not available and would add significant computational burden to the simulation.

An electromagnetic PAF model created by Karl Warnick and students generates complex PAF steering vectors (e.g. $\mathbf{a}_s$ and $\mathbf{a}_i$ from (2.18)) from relative azimuth and elevation angles. The model incorporates several PAF and reflector-specific details. These details include element spacing, mutual coupling, active impedance matching, LNA noise and its

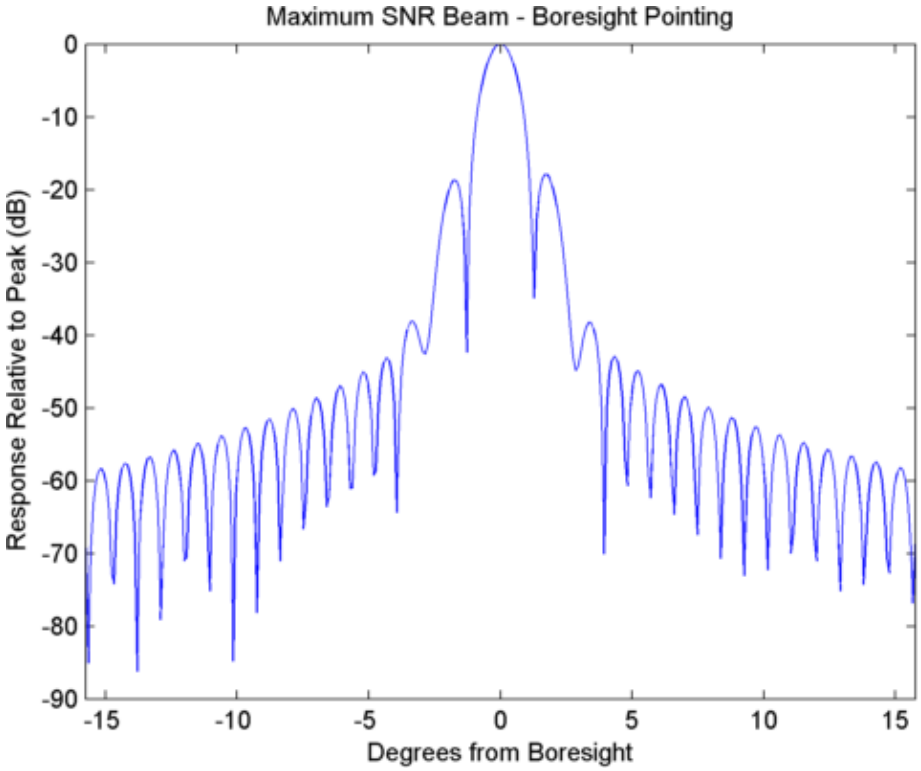


Figure 4.6: Each PAF was modeled as a 19-element thickened-dipole array on ASKAP's 12-meter dish with $f/D = 0.5$, resulting in this maximum SNR boresight beampattern.

recoupling, spillover noise, frequency dependencies, signal bandwidth, polarization, reflector diameter, and focal length.

Figure 4.6 illustrates an azimuthal slice of the maximum-SNR boresight beampattern as computed by the PAF model for one of the ASKAP 12-meter dishes using an 19-element thickened-dipole PAF. For the purposes of this simulation, each PAF used identical beampatterns, which were calibrated using a bright point source as described in Section 2.1.2.

The auxiliary reflector is modeled as a 3.7-meter parabolic dish with the same PAF used on the primary reflectors as the feed. This reflector was chosen to closely match a physical dish in current operation at the Parkes Observatory. It is highly unlikely that the fully-constructed telescope will have an auxiliary reflector with a PAF, so only the maximum SNR beamformer weights for a signal arriving at boresight are used.

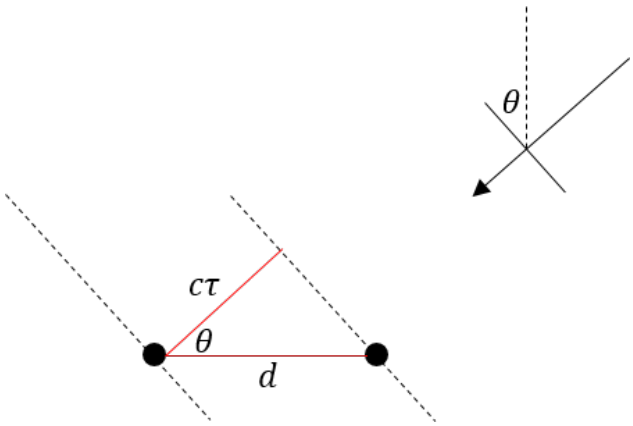


Figure 4.7: Two elements experience a relative phase shift when a plane wave arrives at an angle offset from zenith.

Correlator Dump Rates

The current version of the ASKAP central correlator integrates for five seconds before dumping. With such a setup, it is extremely difficult to cancel fast-moving interference at the central correlator (see Section 2.3.3). In order to effectively mitigate moving interference, a dump time on the order of a few milliseconds must be used at the central correlator.

The correlators used at the PAFs dump every two seconds, but to simplify code, they were modeled with the same dump rate as the central correlator, or five seconds. This parameter is not as crucial because the element baselines are so short.

To get an idea of why the PAF correlator dump times do not need to be short, we will examine the time delays caused by a differential change in angle. Consider the scenario depicted in Figure 4.7. Let θ be the plane wave angle of arrival and d be the baseline length between two elements. The time delay between the two elements is given by

$$\tau = \frac{d}{c}\cos\theta.$$

When the angle of arrival is slightly disturbed,

$$\begin{aligned}\Delta\tau &= \frac{d}{c}\sin\Delta\theta \\ &\approx \frac{d}{c}\Delta\theta.\end{aligned}$$

Therefore, with long baselines (i.e. d is large), the differential time delay becomes large and the spatial signature undergoes a significant change. With shorter baselines, however, the differential time delay is small and thus has little effect on spatial signatures.

In the simulation, a five-second dump time was modeled to reflect the real-world limitations of the central correlator. To illustrate the potential value of fast dump rates, a five-millisecond dump time was also modeled as a demonstration of the value of faster dump rates.

Bulk Time Delay Insertion

Radiation from far-field sources of interest is modeled as a plane wave. As such, the received signals at each feed are time-delayed copies of each other. The signal received at the mth feed is given by

$$\mathbf{x}_m^s[n] = s(nT_s - T_m)\mathbf{a}_s e^{j2\pi f_c n T_s}, \tag{4.1}$$

where $s(t)$ is the signal instantaneous amplitude, T_s is the sampling period, T_m is the plane wave time delay between antenna 0 and antenna m, $\mathbf{a}_s$ is the normalized feed array response to the source of interest, and f_c is the carrier frequency.

Delays across the interferometer are determined by the source's direction of arrival. The propagation distance between each antenna is calculated by projecting the relative baseline vectors $\mathbf{r}_m$ onto a propagation vector $\boldsymbol{\rho}$. The relative baseline vectors are given by

$$\mathbf{r}_m = \begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}, \tag{4.2}$$

where $[x_m \quad y_m \quad z_m]^T$ are the Cartesian coordinates of antenna m, and antenna $m = 0$ is used as the phase reference location. The propagation vector is found using spherical

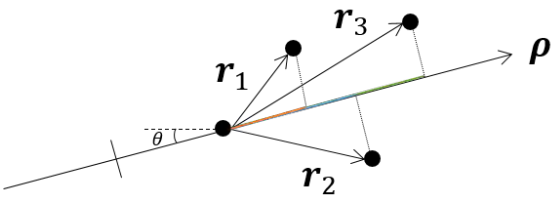


Figure 4.8: The propagation distance between feeds can be computed by projecting the relative baseline vectors onto the propagation vector.

coordinates given by

$$\boldsymbol{\rho} = \begin{bmatrix} \sin(\theta)\cos(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\theta) \end{bmatrix}, \tag{4.3}$$

where θ and ϕ are the elevation and azimuth angles respectively of the source of interest. Projecting the relative baseline vectors onto the unit-norm propagation vector yields

$$d_m = \mathbf{r}_m^T \boldsymbol{\rho}, \tag{4.4}$$

where d_m is the distance traveled by the plane wave between antenna 0 and antenna m. A graphical representation of this operation is depicted in Figure 4.8.

The time delay is calculated by dividing the propagation distance by the speed of propagation c yielding

$$\begin{aligned} T_m &= \frac{d_m}{c} \\ &= \frac{\mathbf{r}_m^T \boldsymbol{\rho}}{c}. \end{aligned} \tag{4.5}$$

Due to the large baselines of an interferometer, a signal of interest with a significant bandwidth will become decorrelated across the array, causing the image to become unfocused. The image is refocused by applying bulk time delay corrections across the array.

Consider the baseband equivalent signal model of (4.1),

$$\mathbf{x}_{m,bb}[n] = s(nT_s - T_m)\mathbf{a}_s. \tag{4.6}$$

To focus the signal of interest across the interferometer, a delay is inserted such that

$$\mathbf{x}_{m,bb}[n + T_m f_s] = s(nT_s)\mathbf{a}_s. \tag{4.7}$$

This ensures that the signal of interest $s(t)$ is correlated across the interferometer. However, this time delay correction causes the interference to become decorrelated across the array.

Consider the baseband signal model for an interferer,

$$\mathbf{x}^i_{m,bb}[n] = i(nT_s - T^i_m)\mathbf{a}_i, \tag{4.8}$$

where $i(t)$ is the interferer instantaneous amplitude, and T^i_m is the propagation delay of the interference plane wave across the interferometer. When a bulk time delay T_m is inserted to focus on the source of interest,

$$\mathbf{x}^i_{m,bb}[n + T_m] = i(nT_s - (T_m - T^i_m))\mathbf{a}_i. \tag{4.9}$$

This causes the interference to decorrelate across the array, threatening the efficacy of RFI mitigation albeit with some (though inadequate) effective RFI attenuation. However, if the processing bandwidth is sufficiently narrowband, this decorrelation effect is negligible.

In order to be considered narrowband, the signal processing bandwidth must meet the constraint

$$BW \ll \frac{c}{D}, \tag{4.10}$$

where c is the wave propagation speed, and D is the maximum baseline of the array. With ASKAP's 6-km maximum baseline, c/D is 50 kHz. This proves to be problematic for ASKAP since its correlators operate on 1-MHz and 18-kHz frequency channels, neither of which is

sufficiently narrow to neglect the decorrelation effect. Therefore, baseline-dependent decorrelation of the RFI signal is included in the simulation.

It is assumed that the bulk time delay correction step is performed perfectly. Therefore, only the decorrelating time delay $T_m - T_m^i$ is applied to the interference time series of baseband voltages. The delay is inserted by applying a phase ramp in the Fourier domain, or

$$i[n - (T_m - T_m^i)] = \mathcal{F}^{-1}\left\{\mathcal{F}\{i[n]\}e^{-j\omega f_s(T_m - T_m^i)}\right\}, \tag{4.11}$$

where $\mathcal{F}\{\cdot\}$ is the fast Fourier transform (FFT) operator, and f_s is the sampling frequency, which is equal to the processing bandwidth.

4.4 Results

The RFI mitigation algorithms listed in Section 4.3 were tested using an feed SNR of 0.87 dB with both 18-kHz and 1-MHz frequency channels to reflect the two operational modes supported by ASKAP. Furthermore, each mode was simulated using the true-to-life 5-s correlator dump time and a theoretical 5-ms dump time. Projection matrices were computed for each correlation dump, and the total survey time was five seconds. Furthermore, since the efficacy of RFI mitigation depends largely on the interference strength, the tests were performed for many different input interference-to-noise ratio (INR) levels.

Input power ratios are referenced at the feed, where the noise power is the average of the diagonal elements of the PAF electromagnetic model covariance matrix. The signal strength is adjusted to achieve the desired SNR. Lastly, the interference strength is defined as the average of the diagonal entries in the interference covariance matrix, or, in other words, the average interference power across all PAF elements. Since the interference strength varies with direction of arrival, the average received power across the motion of the interference is used.

The signal and interference time series were modeled as white complex Gaussian random processes. The noise time series was modeled as a zero-mean complex Gaussian random process with covariance given by the PAF electromagnetic model (see Section 4.3.1).

A total of ten Monte-Carlo trials were performed where each trial consisted of a complete five-second survey for each tested input INR level. The resulting power levels across the ten trials were then averaged together.

Signal-to-interference ratios (SIRs), INRs, and signal-to-interference-plus-noise ratios (SINRs) were computed using the estimated covariances found at the central correlator according to

$$\text{SIR} = \frac{\sum_{k=0}^{K-1} \text{tr}(\hat{\mathbf{R}}_k^s)}{\sum_{k=0}^{K-1} \text{tr}(\hat{\mathbf{R}}_k^i)}, \tag{4.12}$$

$$\text{INR} = \frac{\sum_{k=0}^{K-1} \text{tr}(\hat{\mathbf{R}}_k^i)}{\sum_{k=0}^{K-1} \text{tr}(\hat{\mathbf{R}}_k^n)}, \text{ and} \tag{4.13}$$

$$\text{SINR} = \frac{\sum_{k=0}^{K-1} \text{tr}(\hat{\mathbf{R}}_k^s)}{\sum_{k=0}^{K-1} \text{tr}(\hat{\mathbf{R}}_k^i + \hat{\mathbf{R}}_k^n)}, \tag{4.14}$$

where K is the total number of STI windows, $\text{tr}(\cdot)$ is the trace operator, and $\hat{\mathbf{R}}_k^s$ is the short-time mitigated signal covariance estimate given by

$$\hat{\mathbf{R}}_k^s = \frac{1}{N} \sum_{n=kN}^{(k+1)N-1} \mathbf{P}_k \mathbf{x}_s[n] \mathbf{x}_s^H[n] \mathbf{P}_k^H, \tag{4.15}$$

where $\mathbf{P}_k$ is the RFI-mitigating projection operator for the short-time window k as described in Section 2.3, and $\hat{\mathbf{R}}_k^i$ and $\hat{\mathbf{R}}_k^n$ are defined similarly. For the 5-ms scenario, $K = 1000$, and for the 5-s case, $K = 1$. All power estimates incorporate bias correction as described in 2.3.3 for the 5-ms dump time experiments if the algorithm mitigates at the central correlator. The correction is omitted for the 5-s tests since $\mathbf{C}$ is severely ill-conditioned.

The mitigation techniques will be analyzed by examining three criteria: (1) the lowest input INR level where the algorithm can estimate a usable interference subspace and at least partially cancel the interference, (2) the overall cancellation depth, and (3) the resulting signal integrity.

4.4.1 18-kHz Processing Bandwidth, 5-ms Dump Time

The output INR levels that resulted from RFI mitigation with a processing bandwidth of 18 kHz and a 5-ms correlator dump time are depicted in Figure 4.9. We begin by examining

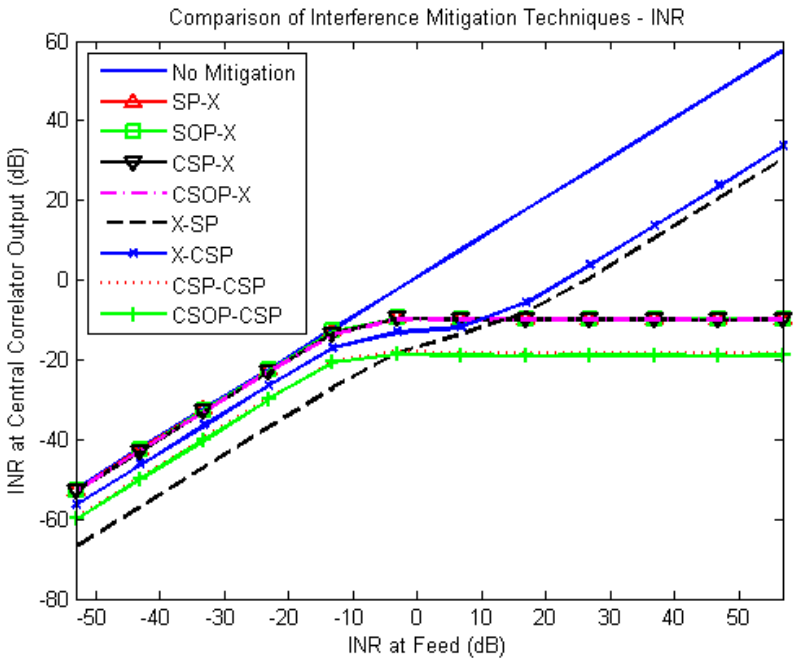


Figure 4.9: Output INRs when mitigating RFI with an 18-kHz processing bandwidth and 5-ms correlator dump time.

the lowest input INR where at least partial cancellation begins. In this sense, all techniques began to cancel the interference around an input INR of -10 dB.

Regarding cancellation depths, every algorithm, with exceptions of X-SP and X-CSP, eventually forced the output INR below -10 dB. Both the X-SP and X-CSP algorithms began to slope upwards and became parallel to the "No Mitigation" curve. This same behavior can be seen with every RFI mitigation technique when tested against high enough input INRs and can be thought as the limiting behavior, or maximum null depth, of the algorithm.

The signal integrity can be analyzed by examining the output SIR levels as plotted in Figure 4.10. Once again, all but the X-SP and X-CSP algorithms perform well and converge to a level of approximately 15 dB.

It is of particular interest that there is signal loss below input INR levels of -10 dB in the X-SP, X-CSP, CSP-CSP, and CSOP-CSP algorithms. Each of these algorithms performs mitigation at the central correlator. Signal cancellation occurs when the dominant eigenvector found using the techniques described in Chapter 2 is not orthogonal to the signal

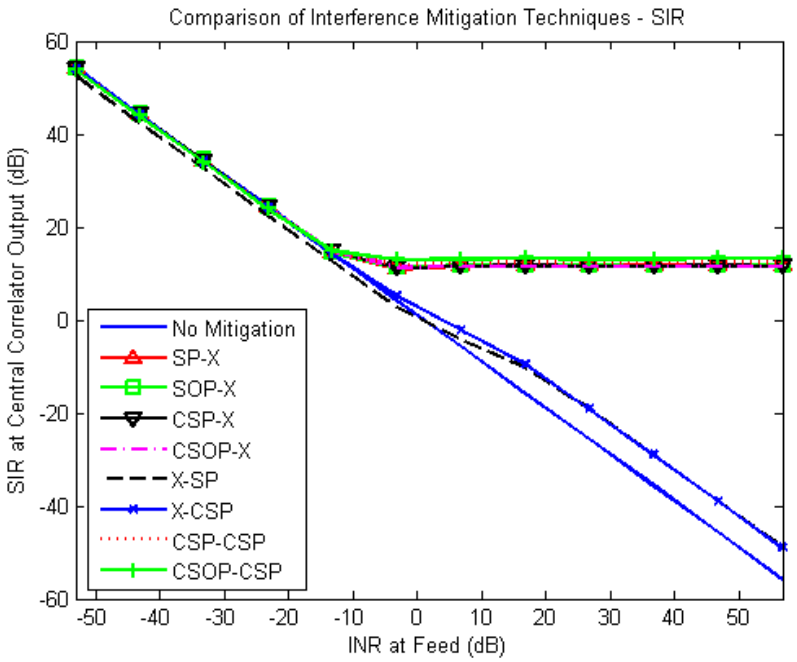


Figure 4.10: Output SIRs when mitigating RFI with an 18-kHz processing bandwidth and 5-ms correlator dump time.

of interest. A likely cause for this behavior is the attenuation of the interfering signal at the feed caused by its arrival in the sidelobes of the feed radiation pattern.

The multi-stage algorithms (i.e. CSP-CSP and CSOP-CSP) achieve a better SIR level than their single-stage counterparts. However, their modest increase is smaller than their decrease in INR. This would suggest that the multi-stage algorithm exhibits signal loss in addition to interference cancellation. This can be seen more clearly in the output SINR levels as plotted in Figure 4.11.

A perfect RFI mitigation algorithm would result in a constant SINR curve equal to the formed-beam SNR. Every algorithm except those that mitigate at the central correlator achieve an SINR close to 1.5 dB for all simulated input INR levels. The techniques that mitigate at the central correlator cancel the signal of interest at lower input INR levels. Surprisingly, the multi-stage algorithms created sustained signal loss across every tested input INR level. This is likely due to the efficiency of the PAF mitigation stage, which causes the signal subspace to be spanned by the dominant eigenvector, inhibiting interference subspace identification at the central correlator.

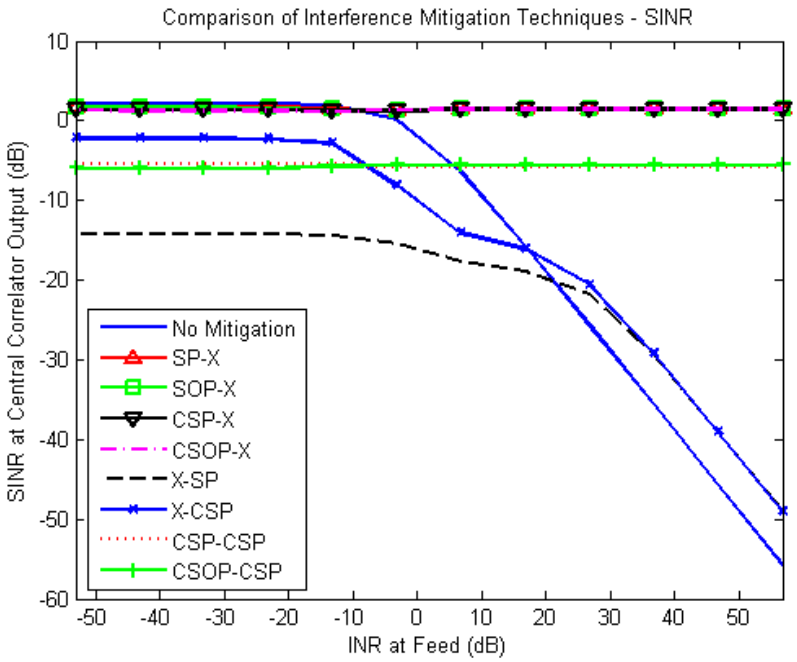


Figure 4.11: Output SINRs when mitigating RFI with an 18-kHz processing bandwidth and 5-ms correlator dump time.

With study, it is not clear which single algorithm is optimal since several exhibit superior performance. However, each of the superior algorithms mitigates exclusively at the PAF. As such, we conclude that, among those tested, the best technique for the 18-kHz bandwidth, 5-ms correlator dump time scenario is any algorithm that only mitigates at the PAF.

The shallow null depths that result from central correlator mitigation are surprising. The cause of this behavior is two-fold. First, sufficient interference motion along the long baselines causes an increase in the rank of the interference subspace, eroding the efficacy of RFI mitigation, which is removing only a rank-one subspace. This smearing effect can be compensated for with sufficiently short dump times at the expense of estimated subspace quality. Secondly, the decorrelation effect described in Section 4.3.1 is present since the processing bandwidth of 18 kHz is not narrowband according to the constraint described in (4.10). To further illustrate this behavior, Figure 4.12 shows the INR curves for the X-SP technique for various processing bandwidths. From these results, we conclude that the central correlator techniques improve with narrower processing bandwidths, as was expected.

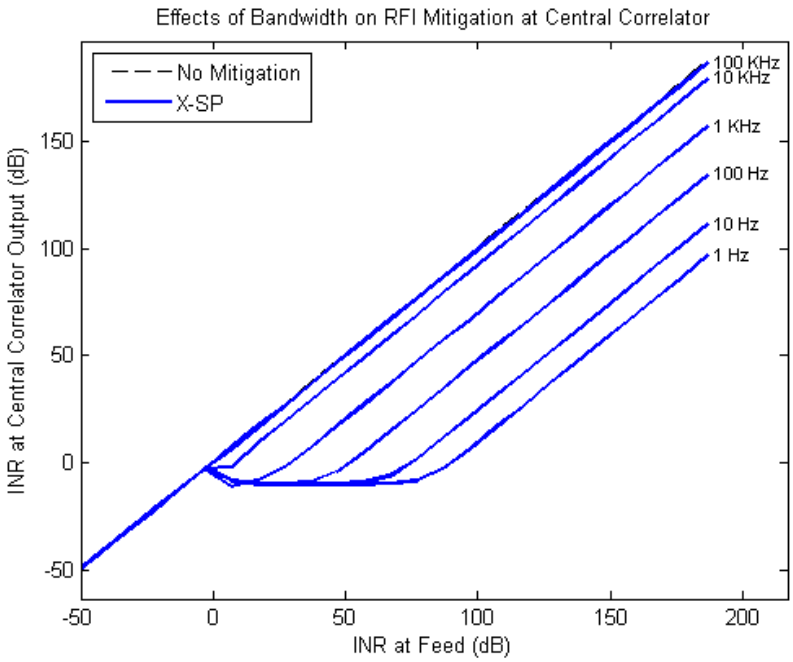


Figure 4.12: As processing bandwidth increases, the interference becomes more decorrelated across the interferometer causing shallower null depths. A correlator with a 5-ms dump time was used to generate this plot.

4.4.2 1-MHz Processing Bandwidth, 5-ms Dump Time

The output INR and SIR levels achieved with RFI mitigation on a 1-MHz channel are shown in Figures 4.13 and 4.14 respectively. The PAF techniques still perform on par with the 18-kHz tests, but, as demonstrated previously, the central correlator techniques suffer from almost non-existent null depths. Furthermore, the multi-stage algorithms do not exhibit deeper cancellation anymore, which is likely due to the now-ineffectual central correlator techniques. Lastly, the signal cancellation resulting from the multi-stage algorithms is now more pronounced.

The SINR levels are depicted in Figure 4.15. The central correlator techniques now do little to cancel RFI and instead cancel the signal of interest at lower input INR levels.

4.4.3 18-kHz Processing Bandwidth, 5-s Dump Time

When using a central correlator dump time of five seconds, the central correlator RFI mitigation techniques become slightly less effective due to the motion of the interference.

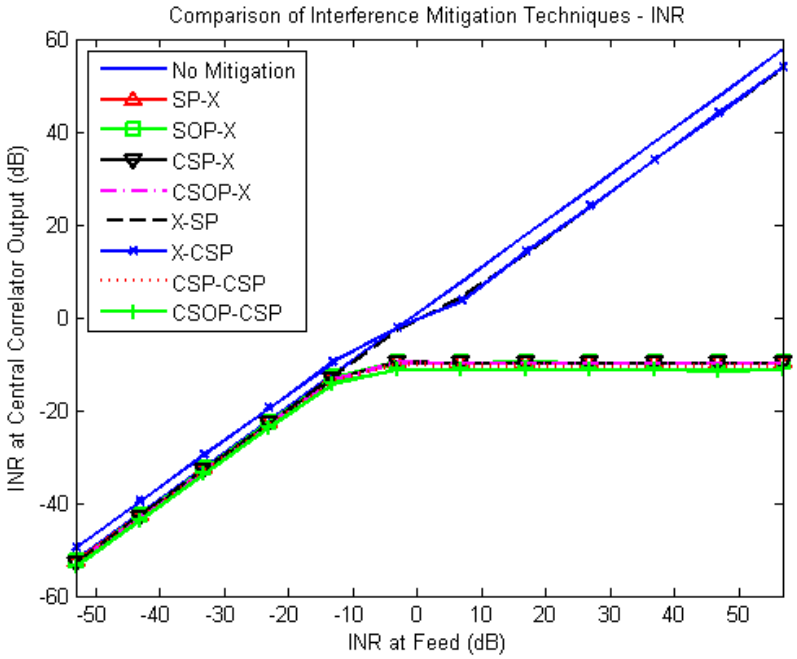


Figure 4.13: Output INRs when mitigating RFI with a 1-MHz processing bandwidth and 5-ms correlator dump time.

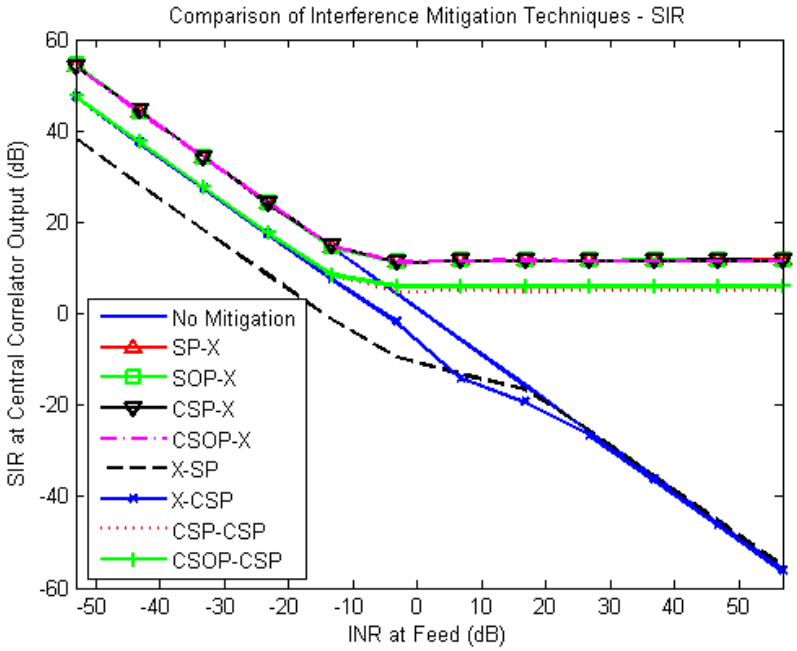


Figure 4.14: Output SIRs when mitigating RFI with a 1-MHz processing bandwidth and 5-ms correlator dump time.

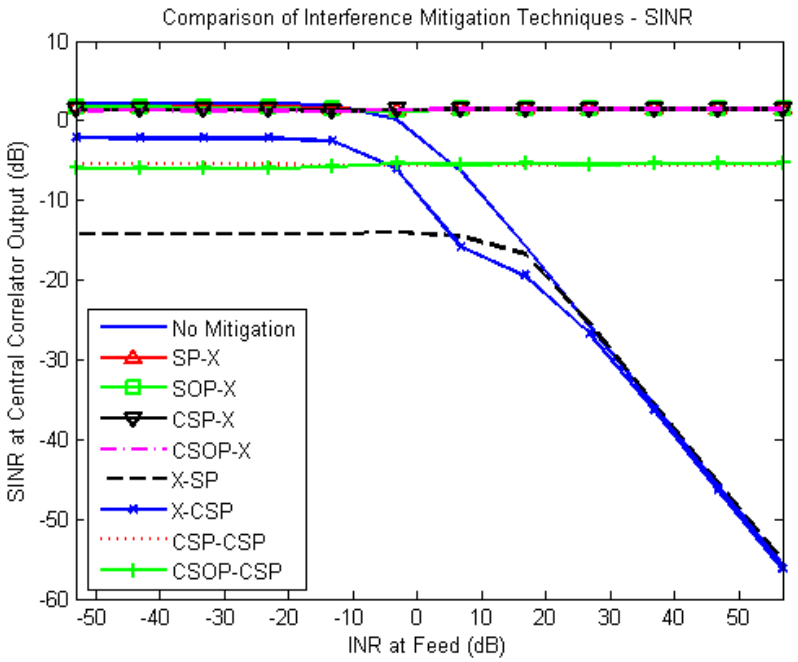


Figure 4.15: Output SINRs when mitigating RFI with a 1-MHz processing bandwidth and 5-ms correlator dump time.

However, the techniques performed at the PAF are still somewhat effective due to the short separations between the elements. Figure 4.16 shows the output INR levels when processing an 18-kHz frequency channel with a 5-s correlator dump time. As before, the central correlator techniques do not exhibit any significant cancellation, but there is a surprising spread of cancellation depths seen among the PAF techniques. Furthermore, there is deeper aggregate cancellation depth in the lower input INR levels. These are both likely caused by the lower subspace estimation error that results from a longer short-time integration window.

The techniques that begin canceling at the lowest input INR levels are CSOP-X and CSP-X. This aligns well with the results from the previous experiments and those presented by Jeffs et al. [22]. The algorithm with the deepest cancellation depths are once again the multi-stage algorithms, but they still suffer from undesirable signal loss.

One major difference between these results and those of the 5-ms experiments is the presence of a maximum null depth for the PAF techniques. In the 5-ms experiments, the PAF techniques were not tested at sufficiently high input INRs to see maximum null depths.

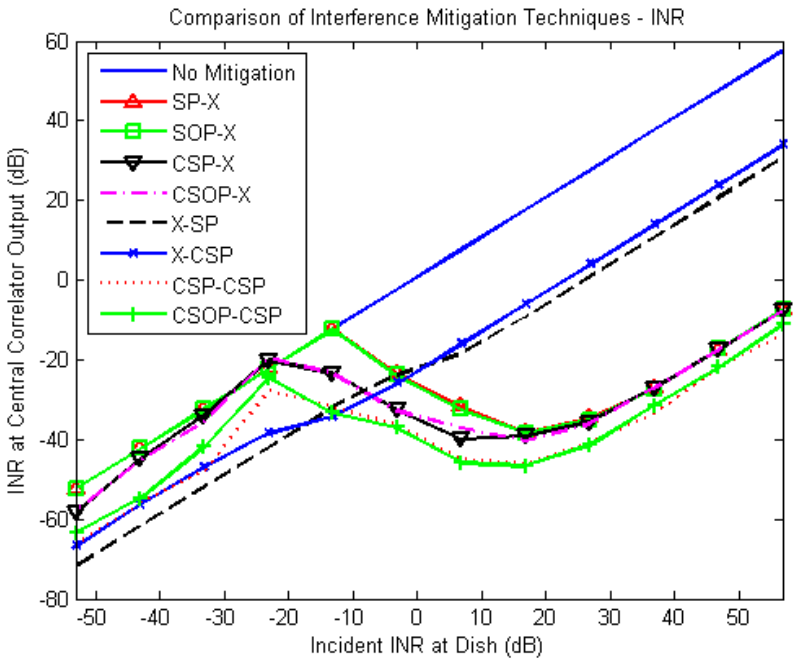


Figure 4.16: Output INRs when mitigating RFI with an 18-kHz processing bandwidth and 5-s correlator dump time.

Now, with the decreased correlator dump rate, the maximum null depths are visible, which demonstrates that there was sufficient RFI motion to weaken the PAF mitigation techniques.

Figure 4.17 shows the output SIR levels for this scenario. As seen with the INR curves in Figure 4.16, the CSOP-X and CSP-X algorithms exhibit the best performance. This indicates that there was not any signal suppression that accompanied the deeper cancellation. This can be further seen in the SINR levels, which are depicted in Figure 4.18.

This same experiment was performed for a 1-MHz processing bandwidth using a 5-s central correlator dump time, but the results yielded little-to-no new information. As such, they have been excluded from this thesis.

4.5 Conclusions

There has been a substantial amount of prior work in the literature that demonstrates effective mitigation techniques for an interferometer. However, up to this point, it has often been assumed that the processing bandwidths are narrow and the correlator can dump at a sufficiently high rate so as to minimize the effects of interference motion. This chapter

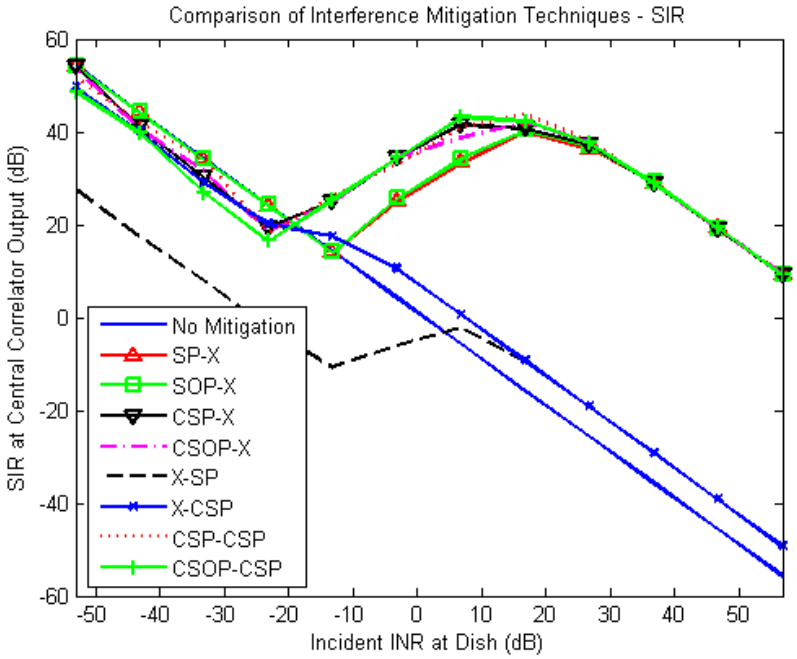


Figure 4.17: Output SIRs when mitigating RFI with an 18-kHz processing bandwidth and 5-second correlator dump time.

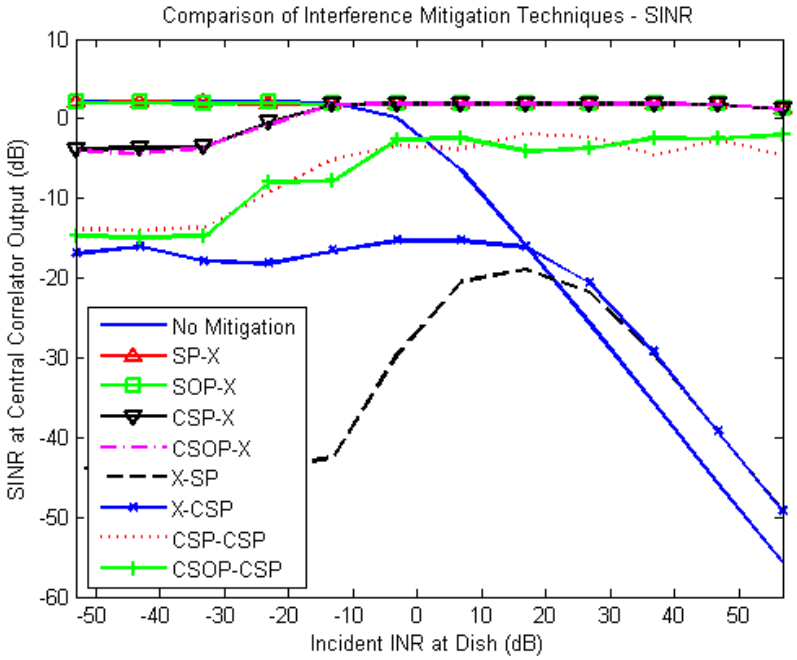


Figure 4.18: Output SINRs when mitigating RFI with an 18-kHz processing bandwidth and 5-second correlator dump time.

has demonstrated that the efficacy of mitigation techniques is highly sensitive to processing bandwidth and correlator dump times, especially when the mitigation takes place at the central correlator.

When processing moderate processing bandwidths, any central correlator RFI mitigation performed across large baselines becomes very ineffective. Couple that with the severe motion smearing that accompanies a slow correlator dump time, and active central correlator mitigation becomes a fruitless exercise. However, mitigation techniques performed at a PAF are able to mitigate at much higher bandwidths and are resilient to motion smearing due to their short baselines.

For either of the ASKAP telescope's frequency channel bandwidths and a correlator that dumps every five seconds, the best active RFI mitigation technique is to cancel at the PAF with an interference-tracking reference antenna. The use of the reference antenna enables the technique to cancel low-power interference. The cross-subspace oblique projection did not demonstrate any significant superiority over conventional cross-subspace projection in this scenario. However, it is likely that the oblique projection operation will perform better on average across several observation scenarios since it guarantees signal integrity without having to resort to the sometimes ill-conditioned bias-correction step.

Chapter 5

Conclusions and Future Work

Radio astronomy observations have long been plagued by harsh observation conditions of poor signal-to-noise ratios (SNRs) and ubiquitous radio-frequency interference (RFI). Furthermore, radio astronomy instruments have traditionally used single-element feeds resulting in a narrow field of view (FOV). FOV limitations and RFI can be mitigated when using phased-array feeds. A phased-array feed, however, introduces significant receiver complexity and substantially higher data rates.

Using open-source FPGA resources from the Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) facilitates the real-time processing of parallel high data rate streams. Using these modules, a data acquisition system was developed to process a bandwidth of 20 MHz across 64 analog inputs and stream the data to disk. The system could produce frequency channels with 16 bits real and imaginary precision and channel bandwidths of 195.31, 97.66, and 48.33 kHz. When streaming 64 ADC inputs to disk, up to 3.41 MHz of analog bandwidth could be acquired indefinitely while 5.76 MHz could be acquired for 10 minutes. When streaming only 40 ADC inputs, up to 9.96 MHz of analog bandwidth was acquirable.

This custom FPGA-based system was tested using a phased-array feed at the Arecibo Observatory by performing calibrations, forming sensitivity grids, measuring beampatterns, and imaging an extraterrestrial source. While the system exhibited an intermittent bug, it still produced believable results that were repeatable. As such, the system shows great promise both as a springboard into higher-bandwidth systems and as a platform for future astronomical surveys.

PAFs can also enhance the interference mitigation capabilities of an interferometer. The Australian Square Kilometre Array Pathfinder (ASKAP) is currently under construction

and will boast a 36-dish synthesis imaging array where the feed of each dish will be a phased-array feed. When canceling interference on this instrument, it is best to exclusively mitigate at the feeds with a reference antenna. The techniques tested that operate at the interferometer central correlator are ineffective due to the long correlator dump times and large processing bandwidths.

5.1 Future Work

The 20-MHz acquisition system described in this thesis is a predecessor to a 150-MHz system named FLAG, or Focal L-band Array for the GBT, which is currently under development. The real-time processing will shift from data streaming to real-time beamforming and correlating. With such a large bandwidth and the need for frequency channel widths to be as small as 10 kHz, a major amount of parallel processing must be performed. To meet this demand, the 150-MHz system will shift away from CASPER FPGA tools and use graphics processing units (GPUs) for most real-time digital signal processing. A second-generation ROACH board (ROACH-II) will still be used to perform coarse frequency channelization using the polyphase filter bank technique described in Section 3.5.1, but any remaining signal processing such as fine polyphase filter bank channelization and correlations will take place on GPUs. As such, a great deal of work must be dedicated to porting the FPGA modules to the GPUs.

The ASKAP RFI mitigation simulations yet lack several real-world considerations that could further affect interference excision performance. These include (1) dynamic range constraints due to bit-widths, (2) the presence of multiple sources of interference, (3) dish-to-dish beampattern variations due to pointing and beamformer calibration error, (4) a detailed electromagnetic PAF model of the true ASKAP 192-element feed, and (5) average-case analysis for various directions of arrival of both signal and interference. These additions would improve accuracy in the simulation results and help them better reflect real-world conditions.

Naturally, it is of great interest to verify any simulation using a real-world experiment. The ASKAP telescope currently has six operational dishes with first-generation PAFs, and thus is nearly capable of verifying the general behaviors described in Chapter 4. However, the

effects of bandwidth will not be easily verified until the largest baselines are established. We along with our colleagues at the Commonwealth Scientific and Industrial Research Organisation (CSIRO) are now collecting data with a 3.7-m reference antenna and an ASKAP-like 12-m Patriot dish and 192-element PAF at the Parkes ASKAP Test-Platform Facility in Australia. Early tests with the SOP-X algorithm are promising.

The simulations from Chapter 4 only considered the diagonal elements of the mitigated covariance matrices. However, the samples of interest are the off-diagonal elements. Due to the bulk time delay correction step, the interference becomes decorrelated causing the off-diagonal elements of the interference covariance matrix to have low magnitude. The largest baselines cross-correlations will be the most decorrelated, but the shorter baselines could still have significant energy to corrupt images. As such, an analysis is needed of whether the decorrelation of the interference due to bulk time correction is sufficient to dismiss the need of further RFI mitigation.

Lastly, the use of a PAF in the reference antenna can also be explored further. If the reference antenna formed multiple beams around the interference, it is possible that the effects of the interference motion could be further mitigated without needing to resort to faster correlator dump rates.

Bibliography

[1] W. Imbriale, "Introduction to 'electrical disturbances apparently of extraterrestrial origin'," *Proceedings of the IEEE*, vol. 86, no. 7, pp. 1507–1509, July 1998. 1

[2] K. Jansky, "Electrical disturbances apparently of extraterrestrial origin," *Proceedings of the IEEE*, vol. 86, no. 7, pp. 1510–1515, July 1998. 1

[3] A. Van Ardenne, J. Bregman, W. Van Cappellen, G. Kant, and J. de Vaate, "Extending the field of view with phased array techniques: Results of european ska research," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1531–1542, Aug 2009. 1

[4] K. F. Warnick, B. Jeffs, J. Landon, J. Waldron, D. Jones, J. Fisher, and R. Norrod, "Phased array antenna design and characterization for next-generation radio telescopes," in *Antenna Technology, 2009. iWAT 2009. IEEE International Workshop on*, March 2009, pp. 1–4. 1

[5] M. J. Elmer, "Improved methods for phased array feed beamforming in single dish radio astronomy," Ph.D. dissertation, Brigham Young University, 2012. 2, 4, 5, 23

[6] K. F. Warnick and B. Jeffs, "Efficiencies and system temperature for a beamforming array," *Antennas and Wireless Propagation Letters, IEEE*, vol. 7, pp. 565–568, 2008. 2

[7] K. F. Warnick, B. Woestenburg, L. Belostotski, and P. Russer, "Minimizing the noise penalty due to mutual coupling for a receiving array," *Antennas and Propagation, IEEE Transactions on*, vol. 57, no. 6, pp. 1634–1644, June 2009. 2

[8] K. F. Warnick and M. Jensen, "Effects of mutual coupling on interference mitigation with a focal plane array," *Antennas and Propagation, IEEE Transactions on*, vol. 53, no. 8, pp. 2490–2498, Aug 2005. 2

[9] A. Parsons, D. Backer, H. Chen, P. Droz, T. Filiba, J. Manley, D. MacMahon, P. McMahon, A. Parsa, A. Siemion, D. Werthimer, and M. Wright, "A scalable correlator architecture based on modular fpga hardware, reusable gateware, and data packetization," *Publications of the Astronomy Society of the Pacific*, vol. 120, no. 873, pp. 1207–1221, 2008. 2, 25, 27

[10] A. Parsons, D. Backer, C. Chang, D. Chapman, H. Chen, P. Crescini, C. de Jesus, C. Dick, P. Droz, D. MacMahon, K. Meder, J. Mock, V. Nagpal, B. Nikolic, A. Parsa, B. Richards, A. Siemion, J. Wawrzynek, D. Werthimer, and M. Wright, "Petaop/second fpga signal processing for seti and radio astronomy," in *Signals, Systems and Computers, 2006. ACSSC '06. Fortieth Asilomar Conference on*, 2006, pp. 2031–2035. 2, 26

[11] A. Thompson, J. Moran, and G. S. Jr., *Interferometry and Synthesis in Radio Astronomy*, 2nd ed. Wiley-VCH, 2001. 2, 13

[12] P. Napier, A. Thompson, and R. Ekers, "The very large array: Design and performance of a modern synthesis radio telescope," *Proceedings of the IEEE*, vol. 71, no. 11, pp. 1295–1320, Nov 1983. 2

[13] J. Welch, D. Backer, L. Blitz, D. Bock, G. Bower, C. Cheng, S. Croft, M. Dexter, G. Engargiola, E. Fields, J. Forster, C. Gutierrez-Kraybill, C. Heiles, T. Helfer, S. Jorgensen, G. Keating, J. Lugten, D. MacMahon, O. Milgrome, D. Thornton, L. Urry, J. van Leeuwen, D. Werthimer, P. Williams, M. Wright, J. Tarter, R. Ackermann, S. Atkinson, P. Backus, W. Barott, T. Bradford, M. Davis, D. DeBoer, J. Dreher, G. Harp, J. Jordan, T. Kilsdonk, T. Pierson, K. Randall, J. Ross, S. Shostak, M. Fleming, C. Cork, A. Vitouchkine, N. Wadefalk, and S. Weinreb, "The allen telescope array: The first widefield, panchromatic, snapshot radio camera for radio astronomy and seti," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1438–1447, Aug 2009. 2, 3

[14] A. Gunst and M. Bentum, "The lofar phased array telescope system," in *Phased Array Systems and Technology (ARRAY), 2010 IEEE International Symposium on*, Oct 2010, pp. 632–639. 2

[15] T. L. Wilson, "The atacama large millimeter array," in *Infrared and Millimeter Waves, 2007 and the 2007 15th International Conference on Terahertz Electronics. IRMMW-THz. Joint 32nd International Conference on*, Sept 2007, pp. 591–593. 2

[16] T. Wong, "The Australia Telescope Millimetre-Wave Upgrade," in *SFChem 2002: Chemistry as a Diagnostic of Star Formation*, C. L. Curry and M. Fich, Eds., 2003, p. 452. 2

[17] J. Baars, J. van der Brugge, J. L.Casse, J. P. Hamaker, L. H. Sondaar, J. J. Visser, and K. Wellington, "The synthesis radio telescope at westerbork," *Proceedings of the IEEE*, vol. 61, no. 9, pp. 1258–1266, Sept 1973. 2

[18] D. DeBoer, R. Gough, J. Bunton, T. Cornwell, R. Beresford, S. Johnston, I. Feain, A. Schinckel, C. Jackson, M. Kesteven, A. Chippendale, G. Hampson, J. O'Sullivan, S. Hay, C. Jacka, T. Sweetnam, M. Storey, L. Ball, and B. Boyle, "Australian ska pathfinder: A high-dynamic range wide-field of view survey telescope," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1507–1521, Aug 2009. 3, 13, 43

[19] J. Bunton, G. Hampson, A. Brown, J. Pathikulangara, J. Tuthill, L. Souza, J. Joseph, T. Bateman, and S. Neuhold, "Askap beamformer," in *General Assembly and Scientific Symposium, 2011 XXXth URSI*, Aug 2011, pp. 1–4. 3, 5, 43

[20] B. Jeffs, K. F. Warnick, J. Landon, J. Waldron, D. Jones, J. Fisher, and R. Norrod, "Signal processing for phased array feeds in radio astronomical telescopes," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 2, no. 5, pp. 635–646, 2008. 3, 11, 17, 21

[21] A. Leshem and A. van der Veen, "Radio-astronomical imaging in the presence of strong radio interference," *Information Theory, IEEE Transactions on*, vol. 46, no. 5, pp. 1730–1747, 2000. 3, 5, 17, 20, 42

[22] B. Jeffs, L. Li, and K. F. Warnick, "Auxiliary antenna-assisted interference mitigation for radio astronomy arrays," *Signal Processing, IEEE Transactions on*, vol. 53, no. 2, pp. 439–451, 2005. 3, 5, 16, 17, 18, 21, 42, 60

[23] A. van der Veen and A. Boonstra, "Spatial filtering of rf interference in radio astronomy using a reference antenna," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 2, May 2004, pp. ii–189–92 vol.2. 3, 42

[24] J. Landon, B. Jeffs, and K. F. Warnick, "Model-based subspace projection beamforming for deep interference nulling," *Signal Processing, IEEE Transactions on*, vol. 60, no. 3, pp. 1215–1228, 2012. 3, 17

[25] J. Raza, A. Boonstra, and A. van der Veen, "Spatial filtering of rf interference in radio astronomy," *Signal Processing Letters, IEEE*, vol. 9, no. 2, pp. 64–67, Feb 2002. 3, 5

[26] B. Jeffs and K. F. Warnick, "Bias corrected psd estimation with an interference canceling array," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 2, April 2007, pp. II–1145–II–1148. 3, 21

[27] ——, "Bias corrected psd estimation for an adaptive array with moving interference," *Signal Processing, IEEE Transactions on*, vol. 56, no. 7, pp. 3108–3121, July 2008. 3, 21

[28] G. Hellbourg, "Radio frequency interference spatial processing for modern radio telescopes," Ph.D. dissertation, University of Orléans, 2014. 3, 21

[29] V. Asthana, "Development of l-band down converter boards and real-time digital backend for phased array feeds," Master's thesis, Brigham Young University, 2012. 4, 23

[30] T. Webb, "Design and polarimetric calibration of dual-polarized phased array feeds for radio astronomy," Master's thesis, Brigham Young University, 2012. 4, 5

[31] W. Van Cappellen and L. Bakker, "Apertif: Phased array feeds for the westerbork synthesis radio telescope," in *Phased Array Systems and Technology (ARRAY), 2010 IEEE International Symposium on*, Oct 2010, pp. 640–647. 5, 13

[32] B. Veidt, T. Burgess, R. Messing, G. Hovey, and R. Smegal, "The drao phased array feed demonstrator: Recent results," in *Antenna Technology and Applied Electromagnetics and the Canadian Radio Science Meeting, 2009. ANTEM/URSI 2009. 13th International Symposium on*, Feb 2009, pp. 1–4. 5

[33] B. Van Veen and K. Buckley, "Beamforming: a versatile approach to spatial filtering," *ASSP Magazine, IEEE*, vol. 5, no. 2, pp. 4–24, 1988. 9, 10

[34] K. Buckley, "Spatial/spectral filtering with linearly constrained minimum variance beamformers," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, no. 3, pp. 249–266, Mar 1987. 10

[35] K. Takao, M. Fujita, and T. Nishi, "An adaptive antenna array under directional constraint," *Antennas and Propagation, IEEE Transactions on*, vol. 24, no. 5, pp. 662–669, Sep 1976. 10

[36] I. Frost, O.L., "An algorithm for linearly constrained adaptive array processing," *Proceedings of the IEEE*, vol. 60, no. 8, pp. 926–935, Aug 1972. 10

[37] R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*, 2nd ed. SciTech Publishing, 2011. 10

[38] M. Elmer, B. Jeffs, K. F. Warnick, J. Fisher, and R. Norrod, "Beamformer design methods for radio astronomical phased array feeds," *Antennas and Propagation, IEEE Transactions on*, vol. 60, no. 2, pp. 903–914, 2012. 11

[39] K. F. Warnick and B. Jeffs, "Efficiencies and system temperature for a beamforming array," *Antennas and Wireless Propagation Letters, IEEE*, vol. 7, pp. 565–568, 2008. 12

[40] J. Welch, D. Backer, L. Blitz, D. Bock, G. Bower, C. Cheng, S. Croft, M. Dexter, G. Engargiola, E. Fields, J. Forster, C. Gutierrez-Kraybill, C. Heiles, T. Helfer, S. Jorgensen, G. Keating, J. Lugten, D. MacMahon, O. Milgrome, D. Thornton, L. Urry, J. van Leeuwen, D. Werthimer, P. Williams, M. Wright, J. Tarter, R. Ackermann, S. Atkinson, P. Backus, W. Barott, T. Bradford, M. Davis, D. DeBoer, J. Dreher, G. Harp, J. Jordan, T. Kilsdonk, T. Pierson, K. Randall, J. Ross, S. Shostak, M. Fleming, C. Cork, A. Vitouchkine, N. Wadefalk, and S. Weinreb, "The allen telescope array: The first widefield, panchromatic, snapshot radio camera for radio astronomy and seti," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1438–1447, Aug 2009. 13

[41] K. F. Warnick, B. Jeffs, J. Landon, J. Waldron, D. Jones, J. Fisher, and R. Norrod, "Beamforming and imaging with the byu/nrao l-band 19-element phased array feed," in *Antenna Technology and Applied Electromagnetics and the Canadian Radio Science Meeting, 2009. ANTEM/URSI 2009. 13th International Symposium on*, Feb 2009, pp. 1–4. 13

[42] W. Van Cappellen, J. de Vaate, K. Warnick, B. Veidt, R. Gough, C. Jackson, and N. Roddis, "Phased array feeds for the square kilometre array," in *General Assembly and Scientific Symposium, 2011 XXXth URSI*, Aug 2011, pp. 1–4. 13

[43] J. Raza, A. Boonstra, and A. van der Veen, "Spatial filtering of rf interference in radio astronomy," *Signal Processing Letters, IEEE*, vol. 9, no. 2, pp. 64–67, 2002. 17

[44] P. Fridman and W. Baan, "Rfi mitigation methods in radio astronomy," *Astronomy and Astrophysics*, vol. 378, no. 1, pp. 327–344, 2001. 17, 42

[45] R. Behrens and L. Scharf, "Signal processing applications of oblique projection operators," *Signal Processing, IEEE Transactions on*, vol. 42, no. 6, pp. 1413–1424, Jun 1994. 21

[46] G. Hellbourg, R. Weber, C. Capdessus, and A. Boonstra, "Oblique projection beamforming for rfi mitigation in radio astronomy," in *Statistical Signal Processing Workshop (SSP), 2012 IEEE*, Aug 2012, pp. 93–96. 21

[47] C. Satten. (2008, March) Lossless gigabit remote packet capture with linux. [Online]. Available: http://staff.washington.edu/corey/gulp/ 31

[48] (2014, August) Fusion-io: A sandisk company. [Online]. Available: http://www.fusionio.com/ 32

[49] G. Cortes-Medellin, A. Viswash, S. Parsley, D. B. Campbell, P. Perilat, R. Black, J. Brady, K. Warnick, and B. Jeffs, "Fully cryogenic phased array camera prototype," in *IEEE International Symposium on Antennas and Propagation and USNC-URSI National Radio Science Meeting*, 2014. 34

[50] (2014, June) Alfa performance paramters. [Online]. Available: http://www.naic.edu/alfa/performance/performance.shtml 35

[51] (2014, April) Nasa/ipac extragalactic database. [Online]. Available: http://ned.ipac.caltech.edu/cgi-bin/nph-objsearch?objname=M+87&img_stamp=yes&extend=no 36

[52] S. Johnston, I. J. Feain, and N. Gupta, "Science with the Australian Square Kilometre Array Pathfinder (ASKAP)," in *The Low-Frequency Radio Universe*, ser. Astronomical Society of the Pacific Conference Series, D. J. Saikia, D. A. Green, Y. Gupta, and T. Venturi, Eds., vol. 407, Sept. 2009, p. 446. 43

[53] (2014, August) Xephem: The serious interactive astronomical software ephemeris. [Online]. Available: http://www.clearskyinstitute.com/xephem/ 46

Appendix A

Operational Details for x64 System

A.1 Introduction

The x64 system is BYU's first scientifically-viable phased-array feed digital receiver and is geared to be used in many future experiments. In order to preserve the usability of the system in a student environment, this appendix provides operational instructions for future users. All codes and documents referenced herein can be found on the Radio Astronomy wiki at "https://radioastron.groups.et.byu.net/" under "x64_system."

The x64 system is a digital back end for phased-array feeds (PAFs). It supports an L-band (1180-1825 MHz) PAF with up to 64 elements across a 20-MHz analog bandwidth. After digitization, there are three possible modes of operation: (1) data acquisition, (2) real-time beamforming, and (3) real-time correlating. This appendix focuses on the data acquisition mode of operation.

There are five major sections to this appendix: (1) a listing of all hardware and interconnections, (2) an outlining of how the PC communicates with the ROACH, (3) a description of how to control the x64 system manually and via a telescope M&C PC, (4) an explanation of the FPGA firmware, and (5) a summary of all operationally-critical post-processing codes.

A.2 Hardware

A top-view of the x64 system when connected to a PAF with its corresponding receiver of electronics is depicted in Figure A.1. The system consists of the following components:

- 16 analog down-conversion cards
- ROACH-1 development board and companion x64 ADC card
- Fujitsu XG2000C 10-GbE switch
- Dell PowerEdge C2100 server PC

This section will provide a technical description of each of these components.

A.2.1 Analog Down Conversion Cards

The purpose of the analog down conversion cards is to mix, filter, and amplify the received RF PAF signals to a 20-MHz bandpass signal centered at 37.5 MHz. A block diagram for the down-conversion electronics is depicted in Figure A.2. The 64 signal paths are

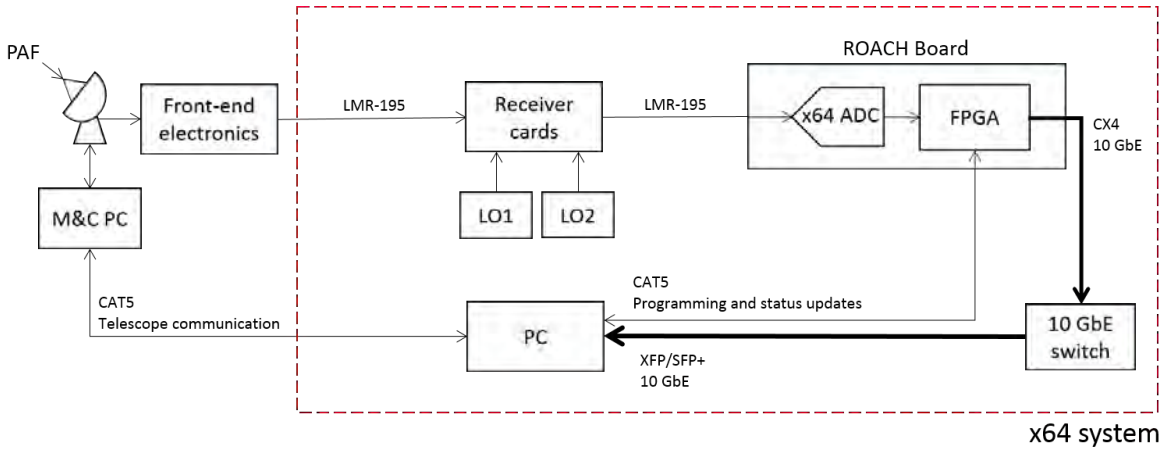


Figure A.1: The x64 system is a back end that digitizes and frequency channelizes received element voltages and saves the resulting samples to disk. The system interfaces with a telescope M&C PC to streamline the entire process.

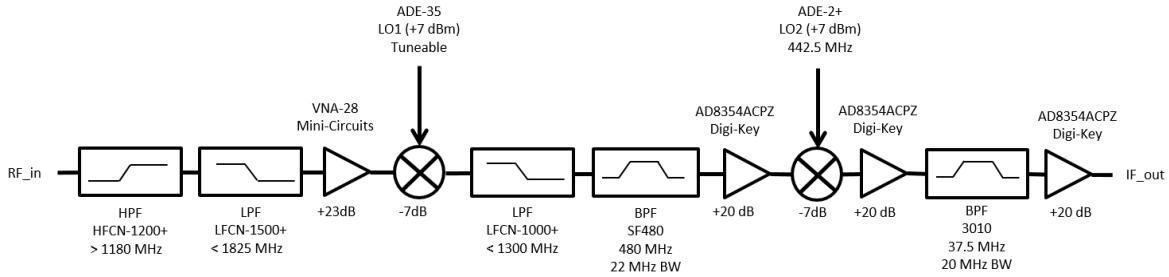


Figure A.2: The x64 receiver cards filter, amplify, and mix the received RF signal down to a 20-MHz passband centered in the second Nyquist zone or 37.5 MHz.

implemented on 16 printed circuit board (PCB) cards, each of which houses four independent signal paths. Pictures of one of these cards and all 16 cards (and some backups) in their cage are shown in Figure A.3.

Each card requires a +5V supply to power the on-board amplifiers. The cards expect an L-band signal between 1180 and 1825 MHz as input into the SMA connectors on the left. The resulting 20-MHz IF signal is outputted through the right connectors. The center connectors on the left and right sides of the cards are for the LO inputs.

The LO signals in each path need to be 7-dBm sinusoidal signals, which means that the LO inputs on each card, which are split by four, need 13 dBm. The first LO signal is tuneable allowing the user to select which 20-MHz down-converted window is outputted. The second LO is fixed at 442.5 MHz.

If the the 20-MHz window needs to be centered at 1600 MHz, for example, the first LO should be set to 2080 MHz. In general, if the desired center frequency is f_{center} (in MHz),

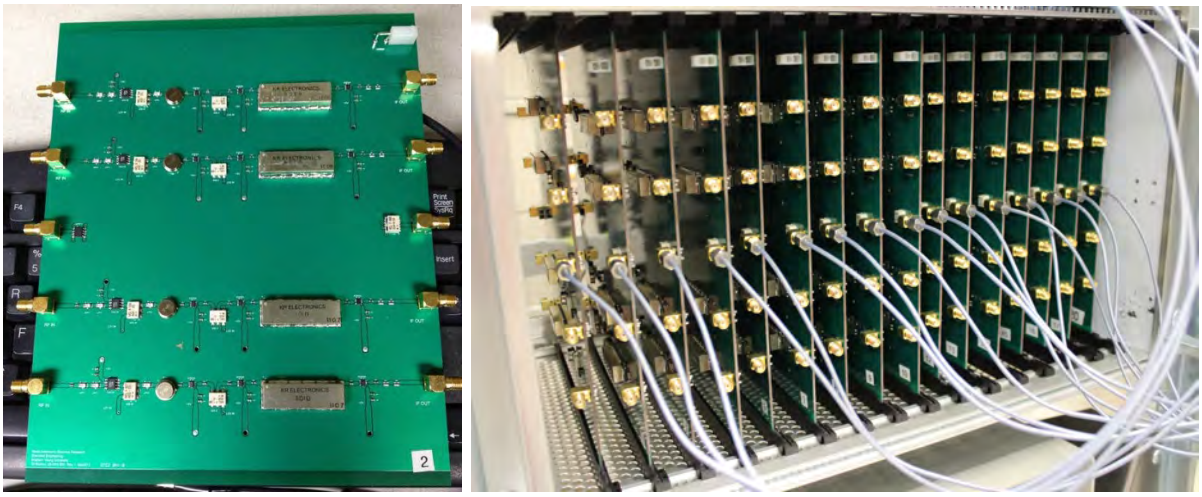


Figure A.3: Each receiver card processes four PAF element signals. The left image shows the front side of a receiver card with the RF input connectors on the left and IF output connectors on the right. The right image shows all of the cards housed in a rack-mountable cage.

then the first LO signal, LO_1 (in MHz) should be set to

$$\mathrm{LO}_1 = f_{\mathrm{center}} + \mathrm{LO}_2 + 37.5, \tag{A.1}$$

where $\mathrm{LO}_2 = 442.5$.

The LO signals are generated using Agilent 8848D signal generators and are distributed to all 16 cards using an LO distribution network. A block diagram of the network is shown in Figure A.4.

The Agilent generators are only able to generate signal powers up to 13 dBm, which is insufficient when split across 16 cards. To compensate, a cascading system of amplifiers and splitters are inserted into the LO distribution network. A picture of the implemented LO distribution network is shown in Figure A.5.

The LO amplifiers require +12V, and so another power supply (in addition to the +5V one) is used. Pictures of the +12V/+5V power supplies are shown in Figure A.6.

The cables coming out of the splitters are carrying high frequency signals and should therefore be short runs on RF-rated cables. The cables used at BYU are 2-ft minibend SMA-M/SMA-M jumpers.

A.2.2 ROACH-1 with x64 ADC

The 64 outputs of the 16 receiver cards are simultaneously digitized at 50 Megasamples per second (Msps) to 12-bit samples by the x64 ADC card, shown in Figure A.7. The card as manufactured has 25-MHz lowpass anti-aliasing filters. If these filters are not deactivated, the receiver cards' outputted bandpass signals centered at 37.5 MHz will be excised. By removing a single capacitor on each ADC signal path, this filter is effectively deactivated. The capacitor that needs to be removed is shown in Figure A.8.

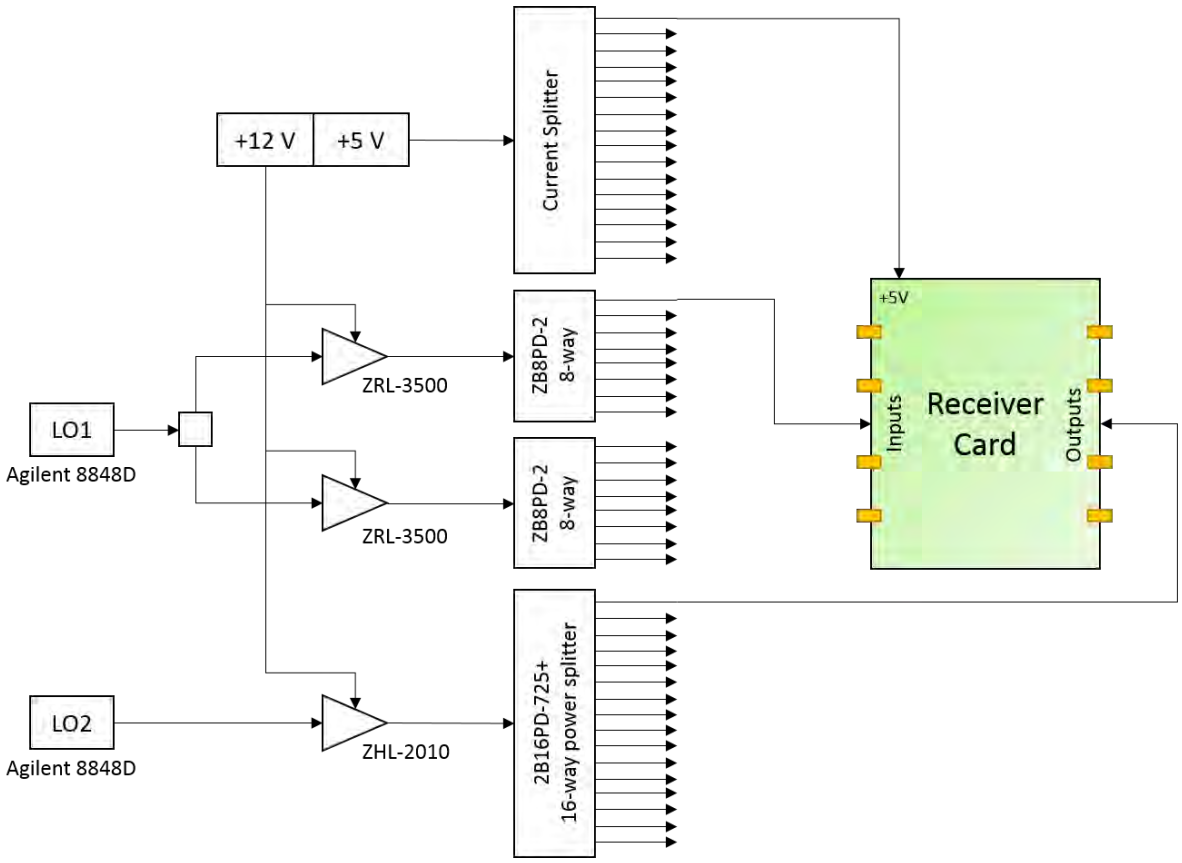


Figure A.4: Since the Agilent signal generators cannot generate enough power to supply 64 mixers, a series of amplifiers and splitters are used to distribute the LO signal.

The ADC streams the digitized samples to the ROACH-1 development board, which houses a Virtex 5 FPGA. When programmed, the FPGA processes the digitized samples and streams the resulting data through a 10-GbE connection. A CX4-to-CX4 2-m cable connects the ROACH and the 10-GbE switch for data transfer.

The ROACH is programmed and managed using a CAT5-Ethernet connection. The PC network connections made to the ROACH are summarized in Figure A.9. The x64 ADC has a reset signal that is low-enabled, meaning that a no-voltage signal holds the card in reset. A high signal can be sent to the ADC reset pin from the ROACH GPIO pins. To connect the appropriate GPIO pin to the ADC reset pin, a jumper cable must be installed between the ROACH and ADC card, as shown in Figure A.10.

A.2.3 10-GbE Switch

The Fujitsu XG2000C 10-GbE switch has 16 CX4 and four XFP connections. A picture of the switch is shown in Figure A.11. The switch, in the x64 configuration, only serves as a media converter. The ROACH-1 uses the CX4 10-GbE connection to stream packets to the switch. The packets are then forwarded from the switch to the host PC over an XFP-to-SFP+ cable.

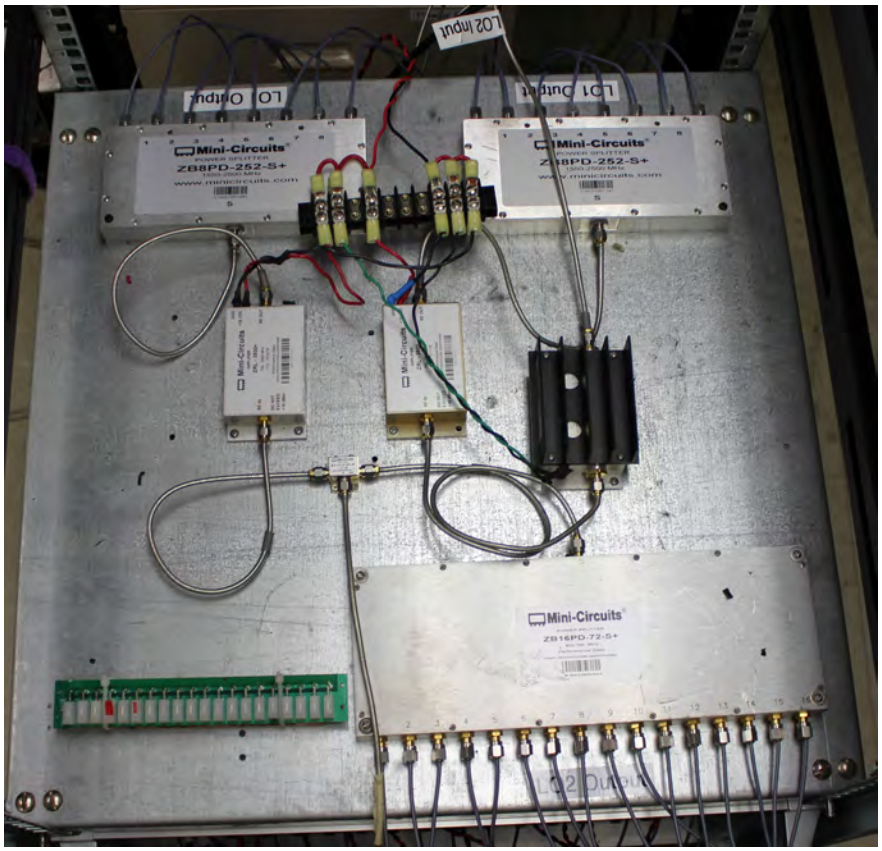


Figure A.5: The current BYU LO distribution network.

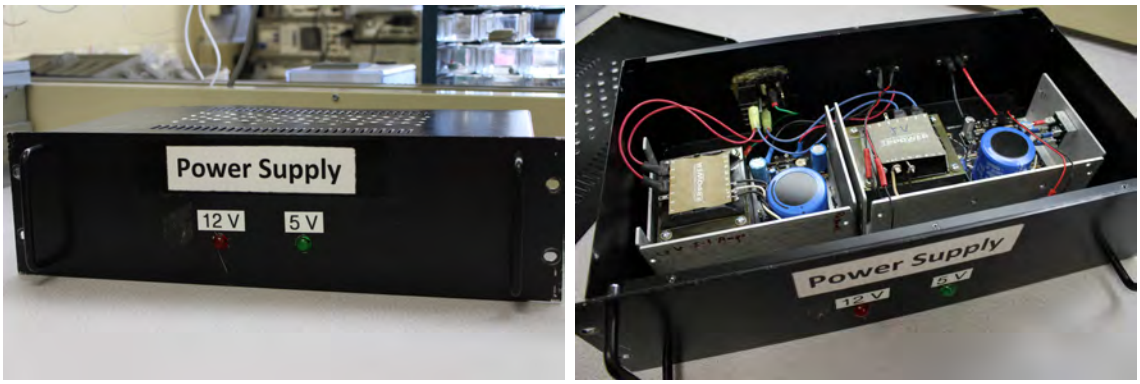


Figure A.6: The LO distribution network and receiver cards requires +12V and +5V respectively. A rack-mountable box with two discrete supplies is used to accommodate this.

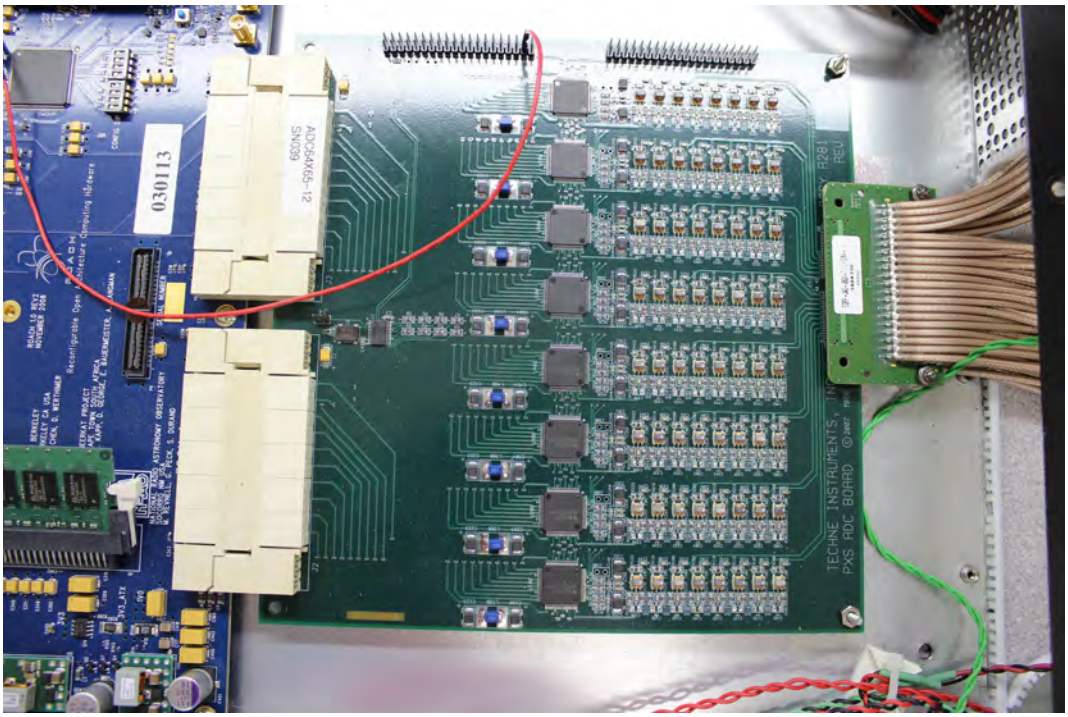


Figure A.7: The x64 ADC attached to the ROACH board through two ZDOK connectors.

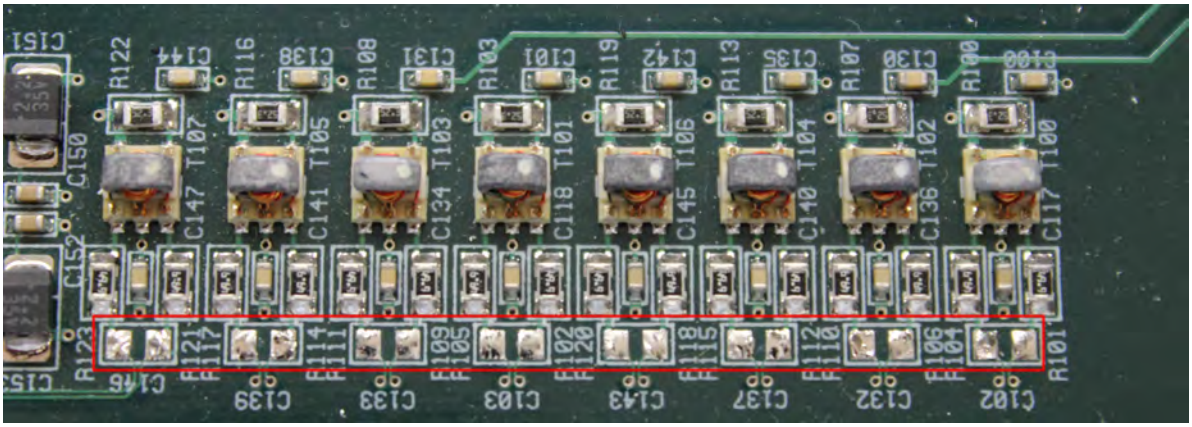


Figure A.8: In order to sample in the second Nyquist zone, the ADC anti-aliasing filters must be deactivated. This can be done by removing the capacitors shown in the red box for all eight ADC chips.

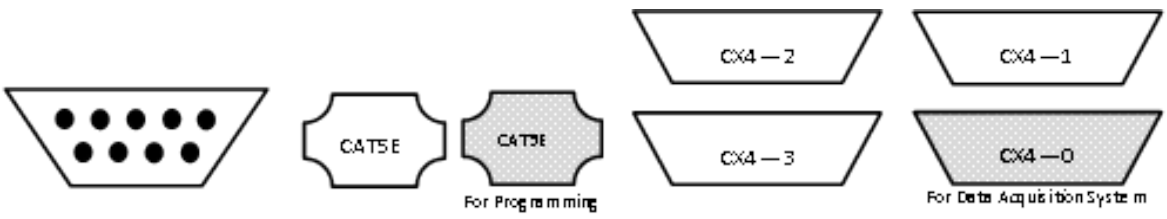


Figure A.9: The ROACH is programmed and maintained over a CAT5 Ethernet connection. The to-be-saved frequency channels are streamed to the host PC over a CX4/CX4 10-GbE connection.

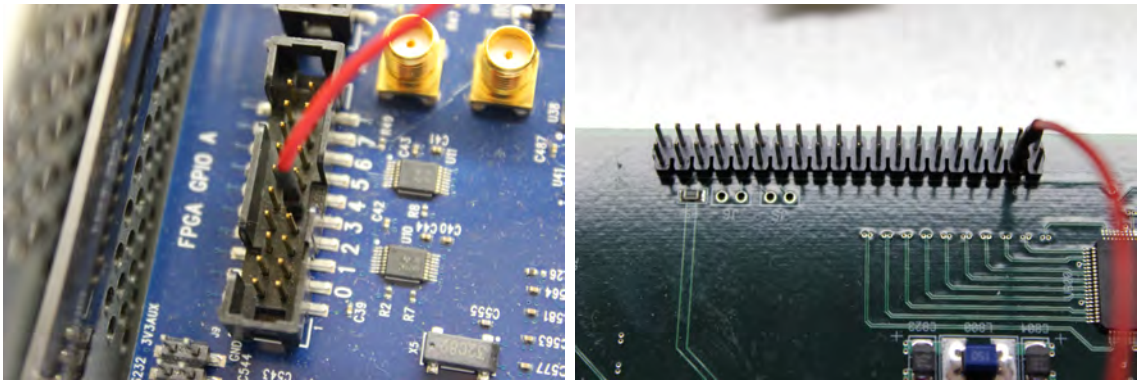


Figure A.10: The ADC needs a high-voltage signal to take it out of a reset mode. The ROACH FPGA firmware provides this signal across a single jumper cable. The left image shows the pin on the ROACH board that the jumper should be connected to, and the right image shows the ADC pin to connect the jumper to.

Figure A.11: The switch has several 10-GbE ports that can forward packets to the appropriate PCs and convert CX4 to XFP.

The Fujitsu switch can easily overheat and cease functioning. Given time to cool down will fix this problem. Further documentation about the Fujitstu switch can be found in "xg2000_user-guide.pdf."

A.2.4 Server PC

The recommended server PC to be used is a Dell PowerEdge C2100, shown in Figure A.12. The PowerEdge PC has four 3.07-GHz Intel Xeon CPUs with four cores each, 192 GB of RAM, an Intel 10-GbE SFI/SFP+ mezzanine network card, and 12 2-TB hot-swappable SATA hard drives configured into two 11-TB striping RAID0 virtual drives. While not yet properly configured, a FusionIO ioDrive II 785-GB solid-state drive can be installed to enhance disk-write speeds. The PC uses the Ubuntu 11.04 operating system.

The PC is used to communicate with the telescope system, control the ROACH, and capture data packets from the FPGA. A diagram of the connections that need to be made for these purposes is shown in Figure A.13.

A.3 PC/ROACH Interface Setup

The first step in setting up the PC/ROACH interface is to get the necessary kernel image and file system. The Linux kernel image can be found in the "ROACH/Kernel Image"

Figure A.12: The recommended PC used for the x64 system is a Dell PowerEdge C2100.

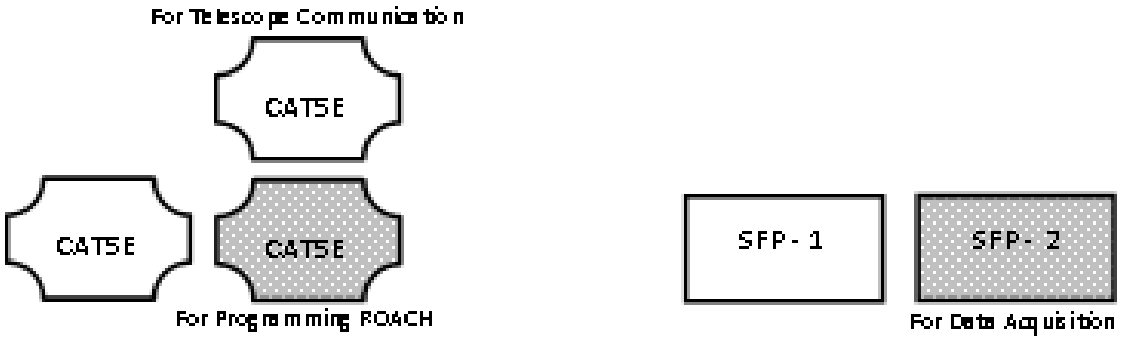


Figure A.13: The PC connects to the M&C PC (for socket communications) and ROACH board (for programming and maintenance) across CAT5 Ethernet connections. The PC receives packets from the ROACH across an XFP/SFP+ 10-GbE connection in a 10-GbE network mezzanine card.

directory in the code repository, and the file system tarball can be found in the "ROACH/File System" directory.

For convenience, it is assumed that a symbolic link is connected to the kernel image file. This link can be made using the following code:

```
$ ln -s uImage-20110812-mmcomitfix uImage
```

This section also assumes that the kernel image symbolic link and extracted file system are in the directories specified in Table A.1. Lastly, the image file must have full permissions. This can be done by typing the following in a terminal:

```
$ chmod 777 uImage-20110812-mmcomitfix
```

Table A.1: Assumed ROACH File Locations

File(s)	Directory
uImage-20110812-mmcomitfix	/tftpboot/
Contents of filesystem_etch_2009-11-30.tar.bz	/srv/roach_boot/

The ROACH boards have a PowerPC that runs a Linux box, which boots off the host PC. The ROACH acts as a network client seeking an IP address and file system. The PC runs a Dynamic Host Configuration Protocol (DHCP) server named "dnsmasq" that assigns IP addresses and provides a TFTP server for clients needing to network boot.

In order to install dnsmasq, type the following command at a terminal:

```
$ sudo apt-get install dnsmasq
```

Once installed, the server must be configured. To do this, the following files must be modified:

1. `/etc/hosts`
2. `/etc/dnsmasq.conf`
3. `/etc/ethers`

The following sections outline how the dnsmasq configuration files can be modified and how dnsmasq is started up.

A.3.1 /etc/hosts

The "hosts" file lists mappings of ROACH IP address to hostnames. This makes it possible to reference different ROACH boards by an alias instead of an IP address. An example of what might be included in this file is as follows:

```
169.254.145.11  roach1  roach1.ee.byu.edu
169.254.145.12  roach2  roach2.ee.byu.edu
```

For example, this connects the IP address "169.254.145.11" to the hostnames "roach1" and "roach1.ee.byu.edu."

A.3.2 /etc/dnsmasq.conf

This configuration file lists commands that would normally be specified at the command line when running dnsmasq. The following outlines an example configuration file that sets up dnsmasq to listen on the network interface "eth1" and designates a filesystem and TFTP server on the host PC:

```
interface = eth1
dhcp-range = 169.254.145.100, 169.254.145.200, 12h
read-ethers
dhcp-option = 42, 0.0.0.0
dhcp-option = 17, 169.254.145.1:/srv/roach_boot/etch_devel
bind-interfaces
dhcp-boot = uImage
enable-tftp
tftp-root = /tftpboot
```

Table A.2: Parameters for dnsmasq.conf

Command	Description
`interface`	Specifies the network interface from which the DHCP server will listen for connection requests
`dhcp-range`	Specifies the range of leaseable IP addresses and how long a lease can be held
`read-ethers`	Makes dnsmasq read the "/etc/ethers" file
`dhcp-option`	Specifies additional options for dnsmasq (see RFC 2132 for options)
`bind-interfaces`	Makes the server-client connection exclusive
`dhcp-boot`	Designates the Linux kernel image file
`enable-tftp`	Enables the TFTP server
`tftp-root`	Specifies location of the Linux kernel image file on the server PC

A description of each of these commands is summarized in Table A.2. The two DHCP options that are used in dnsmasq are 17 and 42. A description of all DHCP options can be found in RFC 2132. Option 17 specifies the root path for the client. The address of "169.254.145.1" is the server PC's IP. Option 42 lists IP addresses for available network time protocol (NTP) servers. An address of "0.0.0.0" represents the machine running dnsmasq.

A.3.3 /etc/ethers

The "ethers" file links media access control (MAC) addresses to IP addresses for the DHCP server. When a client with a MAC address that is specified in this file attempts to connect to the server, the specified IP address will be assigned in lieu of a random one. An example file is:

```
02:00:00:03:01:11  169.254.145.11
02:00:00:03:01:12  169.254.145.12
```

The left entries are MAC addresses of clients, and the right entries are the designated IPs for those MAC addresses. This is useful because it establishes a fixed IP address for each ROACH board, making it easier to reference.

A.3.4 Starting dnsmasq

The dnsmasq server is started by typing "`dnsmasq`" in a terminal. If any changes are made to the configuration files after dnsmasq has been started, the server can be restarted using the following command:

```
$ /etc/init.d/dnsmasq restart
```

Lastly, in order to make the filesystem specified in "/etc/dnsmasq.conf" (see Section A.3.2) available to the ROACH boards, the network file system (NFS) utility must be installed and configured as well. This is installed by entering the following at a terminal:

```
$ sudo apt-get install nfs-kernel-server nfs-common
```

Once that is completed, the "/etc/exports" file must be appended with the following line (shown here as two lines):

```
/srv/roach_boot 169.254.145.0/255.255.255.0
   (rw,subtree_check,no_root_squash,insecure)
```

The file system is then made available to the ROACH by entering the following code at a terminal:

```
$ exportfs -a
```

This will put the file system into the export list. This list can be checked by typing the following:

```
$ showmount -e
```

The file system directory should appear. If an error occurs, then the NFS utility is not running and must be started. This is done with this line:

```
$ service nfs start
```

Once this is done, the ROACH will be able to boot from the PC. The boot process can be monitored by connecting a serial cable between the ROACH and PC and opening the `minicom` tool. This utility can be installed by typing the following in a terminal:

```
$ sudo apt-get install minicom
```

It can then be run by entering the following in a terminal:

```
$ minicom
```

A.4 Running the Data Acquisition System

The data acquisition system (DAQ) streams a selection of frequency channelized samples for a PAF to disk for offline signal processing. The high-level description of the system is summarized in Chapter 3.

There are two ways to interface with the DAQ system: manually or through a TCP/IP socket protocol. This section focuses on setting up and running the system in both of these modes.

A.4.1 Python Dependencies

The DAQ system is controlled using Python scripts on the host PC described in Section A.2.4. These scripts depend on the following publicly available libraries:

1. setuptools*
2. dev*

3. numpy*
4. katcp
5. corr
6. aipy
7. pyephem
8. iniparse
9. construct
10. spead
11. h5py*
12. matplotlib*
13. bitstring+

The majority of these libraries can be installed using the "apt-get" utility by typing the following in a terminal, replacing `<library_name>` with the library to be installed:

```
$ sudo apt-get install python-<library_name>
```

The libraries marked with * cannot be installed using "apt-get" and can be found in a ".tar.gz" archive format at "pypi.python.org/pypi/⟨library_name⟩." Once the archive has been downloaded, extract the contents, navigate to the created directory, and type the following:

```
$ python setup.py install
```

The libraries marked with + cannot be installed using either of the above methods. They use ".zip" archives and can be extracted using the "unzip" utility, which can be installed using "apt-get" as follows:

```
$ sudo apt-get install unzip
```

The archive can then be extracted by typing the following:

```
$ unzip bitstring_3.0.2.zip
```

A.4.2 Gulp

"Gulp" is an open-source optimized packet sniffer written in C. It uses multiple cores and a ring buffer to divide the workload and buffer the data (see Section 3.6). The DAQ system uses Gulp to capture each sample spectrum onto the PC hard drives.

A copy of the Gulp C code is located in the "Gulp" directory in the code repository. This version is modified from the Satton's original code so that it now can work with 64-bit environments, can acquire multiple smaller files, and can be controlled from external software using system interrupts and signals. For lower-level details about the Gulp code, refer to "Gulp Code Documentation.pdf" in the "Gulp" directory.

To make Gulp run the most efficiently, the "schedtool" utility is used to set process core affinities. This can be installed using the "apt-get" utility.

A.4.3 Manual Interface

The DAQ can be manually controlled using the Python script `daq_manual.py`. This script defines a "DAQ" object that serves as a handle into the FPGA firmware. In this section, it is assumed that this script is used in the Python terminal "ipython," which can be installed at the command line using the "apt-get" utility.

Before starting ipython, navigate to the directory where `daq_manual.py` is located. Once ipython is started, the DAQ object can instantiated using the following code:

```
In [1]: import daq_manual
In [2]: a = daq_manual.DAQ('roach1')
```

The segment `'roach1'` should be replaced with the desired ROACH board's IP address or hostname, which was designated in "/etc/hosts" (see Section A.3.1).

Once the DAQ object has been created, a usage statement will appear describing the functions that control the DAQ system. To acquire, use the "grab_data" command. In general, this command is run as follows:

```
a.grab_data(bin_start, bin_end, row_start, row_end,...
            col_start, col_end, fft_len, num_packets, lsb_select)
```

The `bin_start` and `bin_end` parameters specify the frequency channels to be captured, `row_start` and `row_end` specify the input rows to be streamed, and `col_start` and `col_end` specify the input columns to be streamed (see Table 3.3). The `fft_len` documents the length of the FFT performed on the FPGA. **Caution:** this parameter does not change the length of the FFT and is used to verify data rates. The `num_packets` argument specifies the number of packets to capture and `lsb_select` sets the LSB of the 8/8-bit window (see Figure A.16).

For example, the following code commands the ROACH firmware to acquire frequency bins 0-20 for input rows 0-3 and input columns 2-3 with an LSB of 7 across 1000 packets:

```
In [3]: a.grab_data(0, 20, 0, 3, 2, 3, 512, 1000, 7)
```

After this command is issued, packets will appear on the network. If Gulp is not running at this stage, those packets will be lost. Gulp can be started in a separate terminal using the following code:

```
schedtool -F -p 99 -a 0xf -e /gulp -i eth2 -f 'udp' -r 1024 > out.bin
```

This starts up Gulp to acquire UDP packets ("-f 'udp'") from interface "eth2" ("-i eth2") into a 1024-MB ring buffer ("-r 1024") and dumps the packets into the "out.bin" file ("⟨ out.bin"). The schedtool utility forces Gulp to run in FIFO mode ("-F"), meaning that no other process will start on the cores Gulp uses until it terminates. The utility also makes Gulp be the highest priority process on its cores ("-p 99") and sets the CPU affinity to the first four cores ("-a 0xf").

Once Gulp is running, run the "grab_data" command from ipython and packets will stream and be captured. After all the packets have captured, Gulp can be terminated by sending the CTRL+C signal twice.

A.4.4 Network Interface

The DAQ system is controlled over the network using a TCP/IP socket interface, implemented in a series of three Python scripts. These scripts are (1) `byu_slave.py`, (2) `msg_parse.py`, and (3) `res_manage.py`. The network interface can be started by typing the following code at a terminal:

```
python byu_slave.py -c [config file] -l [log file]
```

The system uses a configuration file to customize the system for different telescope M&C interfaces. When running, all console outputs are saved to a log file.

The configuration file is an XML file that specifies socket parameters, antenna-cable mappings, socket message templates, ROACH parameters, and bitstreams. The root tag of the file is named "Telescope" and has five children: (1) header, (2) socket, (3) configuration, (4) msg_parse, (5) res_manage. The order of the children tags is not important. An example of the root tag with its children is depicted in the following code:

```
<Telescope>
    <header>
        ...
    </header>
    <socket>
        ...
    </socket>
    <configuration>
        ...
    </configuration>
    <msg_parse>
        ...
    </msg_parse>
    <res_manage>
        ...
    </res_manage>
</Telescope>
```

The following sections will explain what each child tag and their respective sub-trees do to customize the x64 system.

header

The "header" tag documents user-specific meta data. These entries do not affect the operation of the x64 system but rather provide contextual information about the configuration file.

There aren't any required header tags, but it is recommended that the observation telescope, configuration file author and creation date, and feed be specified. It can also be used to document additional notes. An example header tree is shown below:

```
<header>
    <telescope>Arecibo Observatory</telescope>
    <feed>AO19</feed>
    <author>Richard Black</author>
    <date>Aug 2, 2013</date>
    <notes>Sample configuration file</notes>
</header>
```

socket

The parameters needed for socket communication are specified under the "socket" tag. The IP addresses and fabric ports for the host and M&C PC are specified here. Additionally, a "send_error" tag specifies a boolean (a 0 or 1) that informs the x64 system that the M&C PC expects an error message. An example socket tree is shown below:

```
<socket>
    <my_ip>10.9.131.1</my_ip>
    <my_port>6000</my_port>
    <their_ip>10.10.132.2</my_ip>
    <their_port>6000</their_port>
    <send_error>1</send_error>
</socket>
```

configuration

The "configuration" tag contains a table that connects cable numbers, ADC input numbers, and PAF element numbers. This tag does not affect the operation of the x64 system but does log this mapping for use in post-processing codes.

The codes found in the code repository expect a three-column table representing the ADC input numbers, cable numbers, and feed element numbers. Each row is placed on its own line, and each column is delimited by variable white space. An example of a configuration tag is shown below:

```
<configuration>
    ADC Cable Dipole
    1   1     7a
    2   2     13b
    17  3     2a
```

Table A.3: Available Python Commands

Command Name	Description
`daq_start`	Starts the DAQ system by programming the FPGA and calibrating the ADC.
`daq_scan`	Creates a new file and acquires for specified accumulation length (no integration occurs).
`daq_setup`	Changes system parameters.
`daq_spec`	Acquires "num_specs" files for specified accumulation length. Postpends "_spec_⟨num⟩" to each filename, where ⟨num⟩ is a file counter.
`daq_end`	Closes the socket.

```
    18  4     2b
    ... ...   ...
</configuration>
```

msg_parse

The "msg_parse" section defines socket messages and what commands are run when said messages are received. Each message is defined using a "message" tag with attributes of (1) "name," (2) "ack," (3) "err," and (4) "cmd." An example "message" tag is shown below:

```
<msg name="INIT" ack="1" err="1" cmd="daq_start">
    ...
</msg>
```

The "name" attribute is meta data, "ack" is a boolean that specifies whether the M&C system expects an acknowledge message, "err" is a boolean that specifies whether or not the M&C system wants error messages through the socket, and "cmd" is the name of the Python command to be run. Available commands are enumerated in Table A.3. There are additional commands that have yet to be implemented. These are:

1. `bf`
2. `bf_coeff`
3. `spec`
4. `x`

These, along with any unsupported message, will result in a returned error.

The contents of the "msg" tag contain a message structure, formatting instructions for acknowledgments and errors, acknowledgement timing, and parameter names that can be set by the message. An example of a complete "msg" tag is shown below:

```
<msg name="SETUP" ack="1" err="1" cmd="daq_setup">
    SETUP( [A-Z_]*=[A-Za-z.+0-9]+)*
    <ack_format>setup ok</ack_format>
    <err_format>setup err %s</err_format>
    <ack_timing>after</ack_timing>
    <param name="num_secs">TM_SECS=(?#)[0-9]+(.[0-9]+)?(?#)</param>
    <param name="lsb_select">LSB_SEL=(?#)[0-9]+(?#)</param>
</msg>
```

This example will configure the x64 system to run the "daq_setup" command when a message of the regular expression format "`SETUP( [A-Z_]*=[A-Za-z.+0-9]+)*`" is received. Examples of this kind of message include the following:

```
SETUP N_EL=31 N_XEL=31 ZA_STEP=0.5 AZ_STEP=0.5 TM_SECS=4 SRC=J2123+250
SETUP N_EL=15 N_XEL=15 TM_SECS=1 LSB_SEL=9 SRC=CAS-A
SETUP SRC=moon LSB_SEL=6
```

The "ack_format" tag defines the message that will be returned as an acknowledgement, and the "ack_timing" tag specifies when the acknowledge should be sent ("before" or "after"). If the parent tag's "ack" parameter is set to 0, these tags can be omitted.

The "err_format" tag, similar to "ack_format," specifies how any error message should be structured. It allows for a single string to be inserted in the message by using the `scanf` delimiter "%s." Using the example above with an error message of "Unsupported Operation," the following message would be sent through the socket:

```
setup err Unsupported Operation
```

If the parent tag's "err" parameter is set to 0, this tag can be omitted.

System parameters can be changed through socket messages by using "param" tags in "msg." The above example allows for changes to the "num_secs" and "lsb_select" parameters. The contents of a "param" tag should contain a regular expression for a message chunk that will change the specified parameter. The delimiters "(?#)" surround the regular expression that represents the new parameter value. For example, to let the "num_secs" parameter change when "TM_SECS" appears in the body of the received socket message, and the parameter will be an integer, the following would be used:

```
<param name="num_secs">TM_SECS=(?#)[0-9]+(?#)</param>
```

The parameters that can be changed through socket messages are enumerated in Table A.4.

res_manage

The "res_manage" tag contains information about the ROACH resources, FPGA bitstreams, 10-GbE network settings, and Gulp parameters. It has four children tags, as shown in the following code segment:

Table A.4: Socket-Changeable Parameters

Parameter	Description
bin_start	Starting frequency bin index (starts from 0)
bin_end	Ending frequency bin index
row_start	Starting input row index (starts from 0)
row_end	Ending input row index
col_start	Starting input column index (starts from 0)
col_end	Ending input column index
lsb_select	Index of LSB for 8-bit/8-bit data window (0 - 10)
num_secs	Number of seconds to acquire for
num_specs	Number of files to acquire when in spectrometer mode

```
<res_manage>
    <resources>
        ...
    </resources>
    <configuration>
        ...
    </configuration>
    <gulp>
        ...
    </gulp>
</res_manage>
```

The "resources" child tag lists all of the ROACH boards that will be used in the x64 system and its corresponding IP address or hostname. For example, if two ROACH boards will be used, then the "resources" tag would look like the following:

```
<resources>
    <roach name="roach1">10.0.1.1</roach>
    <roach name="roach2">10.0.1.2</roach>
</resources>
```

The name attribute for each "roach" tag is used later in the "configuration" section to connect ROACH boards to modes of operation.

The "configuration" tag contains "process" children that connect ROACH boards to their corresponding bitstreams. Three processes are currently supported: (1) "daq," (2) "bf," (3) "x." These correspond to the stream-to-disk, beamformer, and correlator modes respectively. An example of a "configuration" tag is shown below:

```
<configuration>
    <process name="daq">
        ...
    </process>
```

```
        <process name="bf">
            ...
        </process>
        <process name="x">
            ...
        </process>
    </configuration>
```

Each "process" tag must specify the names of the bitstreams and the ROACH board to be used. It may also set any process-specific parameters to default values. Each "bitstream" tag has an "fft" attribute that indicates what FFT length will be used in the process. For example, the "daq" process could look like the following:

```
    <process name="daq">
        <bitstream fft="256">x64daq_256.bof</bitstream>
        <bitstream fft="512">x64daq_512.bof</bitstream>
        <bitstream fft="1024">x64daq_1024.bof</bitstream>
        <roach>roach1</roach>
        <params>
            <param name="bin_start">100</param>
            <param name="bin_end">150</param>
            <param name="row_start">0</param>
            <param name="row_end">7</param>
            <param name="col_start">0</param>
            <param name="col_end">4</param>
            <param name="fft_length">512</param>
            <param name="lsb_select">7</param>
            <param name="num_secs">5</param>
            <param name="num_specs">1000</param>
        </params>
        <ip>10.0.0.30</ip>
    </process>
```

Note that the "fft_length" parameter determines which bitstream will be used and that changing this parameter through the socket will not change the bitstream. Also note that an additional tag named "ip" is included in the above example. This is specific to the "daq" process and specifies the destination IP address for the outputted 10-GbE packets.

A.5 Firmware

The following files are used to generate the DAQ system's bitstreams:

1. `model_daq.mdl`
2. `data_acq_16bit_in2.vhd`
3. `data_acq_16bit_in2_config.m`

4. `fifo_16x64_fwft_bram.vhd`

5. `fifo_16x64_fwft_bram.ngc`

The `.mdl` file is the main, top-level Simulink model. The `data_acq_16bit_in2.vhd` is the VHDL code used for the input selection logic. The `data_acq_16bit_in2_config.m` file is the configuration file used to initialize the generic values and provide error checking for the `data_acq_16bit_in2.vhd` file. The `fifo_16x64_fwft_bram.vhd` file provides the structure for a BRAM module used by `data_acq_16bit_in2_config.m`. The `fifo_16x64_fwft_bram.ngc` file provides a compiled version of the VHDL entity. The majority of changes that will ever need to be made in this model will be to the `.mdl` file.

The following sections will breakdown the model_daq.mdl file by examining (1) the x64 ADC, (2) the F-Engine, (3) the bit reduction logic, (4) the data filter, and (5) the UDP packetizer and 10-GbE core.

A.5.1 x64 ADC

There are two important aspects of the x64 ADC card to understand as a developer: the physical operation of the card and the digital handle to the ADC outputs (yellow block). Physically, the card simultaneously samples 64 inputs on eight 12-bit ADC chips at 50 Msps. Chip 1 samples inputs zero through seven, and chip k samples inputs $8(k-1)$ through $8k-1$. After sampling, the input signals are serialized and sent across two ZDOK connectors to the FPGA at 300 MHz.

The digital handle to the ADC outputs is embodied in a yellow block, depicted in Figure A.14. There are 16 data lines outputted from the ADC yellow block, which is 1/4 of the total ADC outputs. The ROACH is clocked at 200 MHz, or 4x the ADC clock speed, enabling each data line of the yellow block to service four of the ADC inputs. Line 0 services inputs zero through four, and the kth line services inputs $4(k-1)$ through $4k-1$. These lines connect to the F-Engine inputs.

A.5.2 F-Engine

The F-Engine performs the PFB FFT technique to obtain frequency channels. A screenshot of the F-Engine in the Simulink model is shown in Figure A.15. There are three distinct stages in the F-Engine: (1) the FIR polyphase filter bank, (2) reordering logic, and (3) FFT block.

The filter bank pre-filters the incoming data to reduce spectral leakage (see Section 3.5.1). Default parameters for the PFB block are enumerated in Table A.5.

The reordering logic undoes the demuxing caused by the ADC yellow block to create a set of time series as required by the FFT block. To do this, the reorder block buffers up 4*NFFT points on each data line and outputs every fourth sample. This reordering scheme for a single data line is depicted in Table A.6.

Lastly, the FFT block computes frequency channels for four separate inputs and outputs the positive channels on two data lines. This adds another level of demuxing (by a factor of two), creating the output structure seen in Table 3.3.

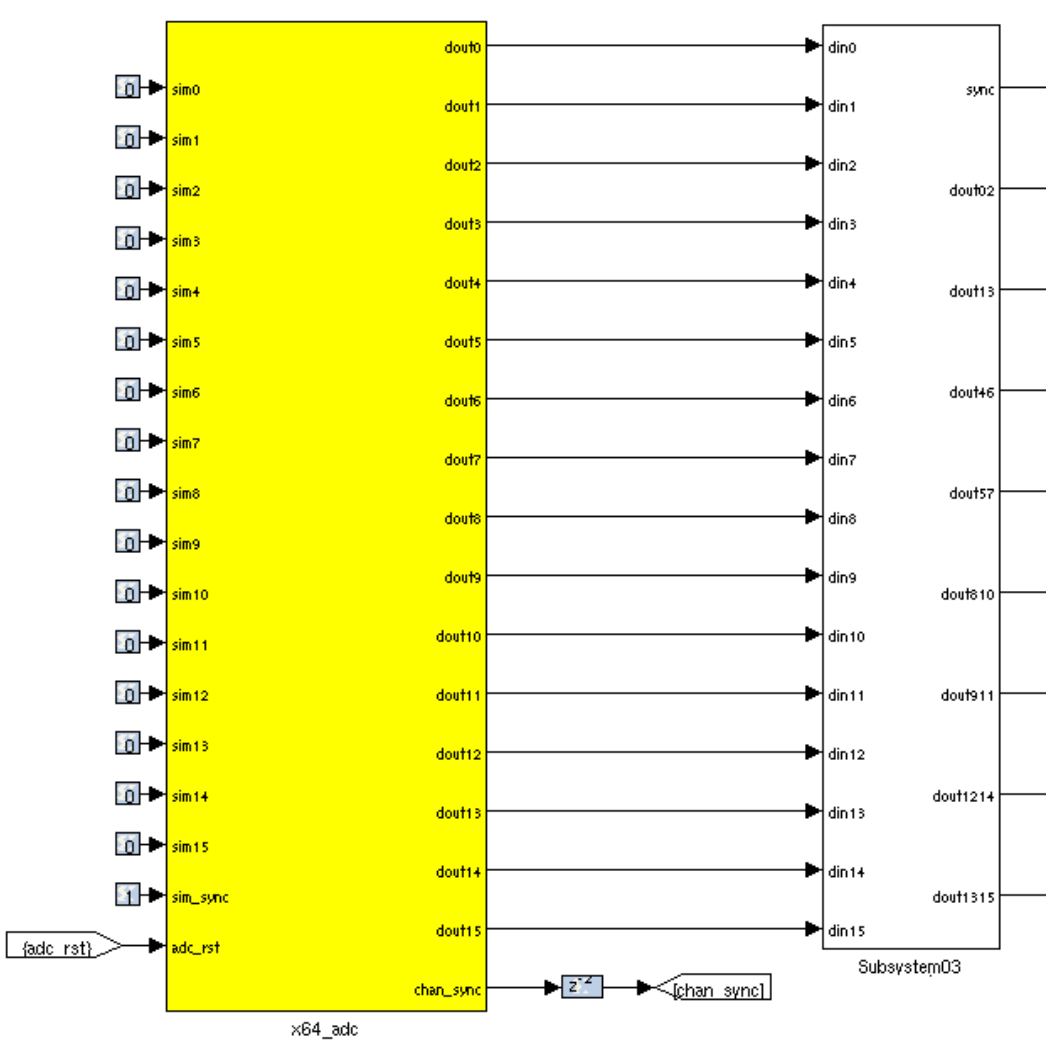


Figure A.14: The data samples come in demuxed by a factor of four and appear on 16 data lines out of the x64_adc yellow block.

A.5.3 Bit Reduction

The samples coming out of the F-Engine are 36-bit complex values (18 bits real and 18 bits imaginary). This is cut down to 16-bit complex samples (8 bits real and 8 bits imaginary) using the bit reduction logic. A screenshot of this subsystem is depicted in Figure A.16. This subsystem splits the incoming 36-bit samples into their real and imaginary components and slices them down to 9 bits each. A value of 1 is then added to the LSB and sliced to the 8 MSBs, effecting rounding (see Section 3.5.2).

Every possible window is computed and fed into a multiplexer with a select signal set from the register named `lsb_select`. The outputted samples are then forwarded to the data filter.

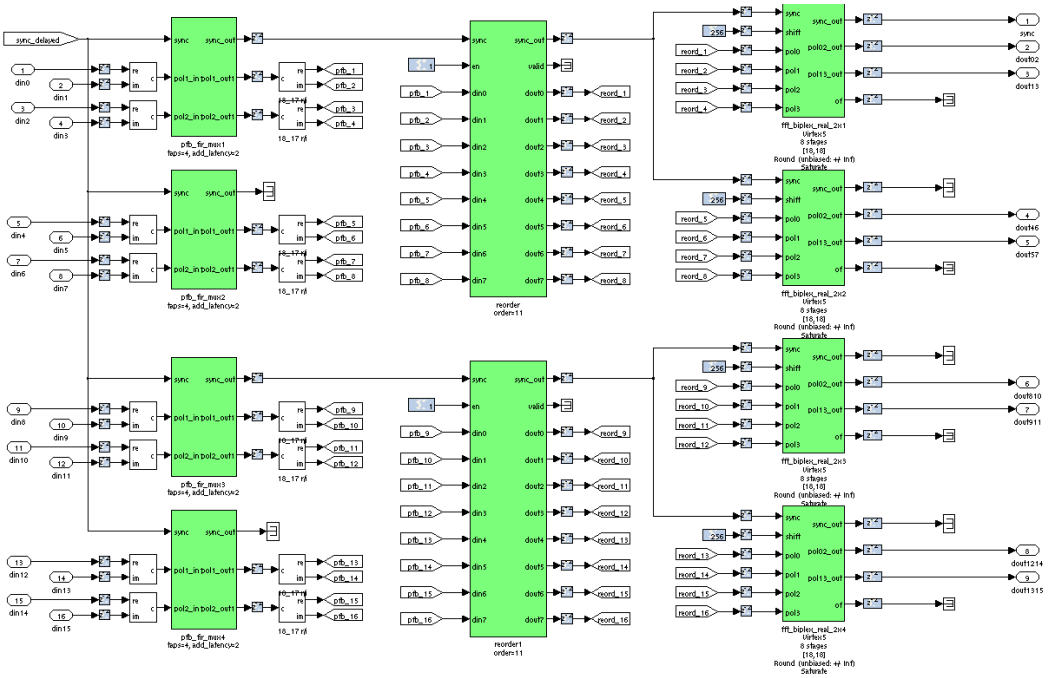


Figure A.15: The F-Engine uses the PFB FFT technique to compute frequency channels. It involves a PFB stage that mitigates the channel's spectral leakage, followed by a reordering stage that buffers up time samples for each input. These data are then frequency channelized using an FFT block.

Table A.5: PFB Block Parameters

Mask Parameter	**Value**
Size of PFB ($2^?$)	8, 9, or 10
Total Number of Taps	4
Windowing Function	Bartlett*
Number of Simultaneous Inputs	2
Make Biplex	True
Input Bitwidth	12
Output Bitwidth	18
Coefficient Bitwidth	18
Use Distributed Memory for Coeffs	False
Add Latency	2
Mult Latency	2
BRAM Latency	3
Fanout Latency	1
Quantization Behavior	Round (unbiased: +/− Inf)
Bin Width Scaling	1
Multiplier Specification	2
Share Coefficients?	True

* This is a reasonable window, but a window that results in equal-level sidelobes is recommended.

Table A.6: Reorder Block Behavior

Time →									
$x_0[0]$	$x_1[0]$	$x_2[0]$	$x_3[0]$	$x_0[1]$	...	$x_3[\text{NFFT}/4-1]$	$x_0[\text{NFFT}/4]$	...	$x_3[\text{NFFT}-1]$
↓									
Time →									
$x_0[0]$	$x_0[1]$	$x_0[2]$	$x_0[3]$	$x_0[4]$	...	$x_0[\text{NFFT}]$	$x_1[0]$	...	$x_3[\text{NFFT}-1]$

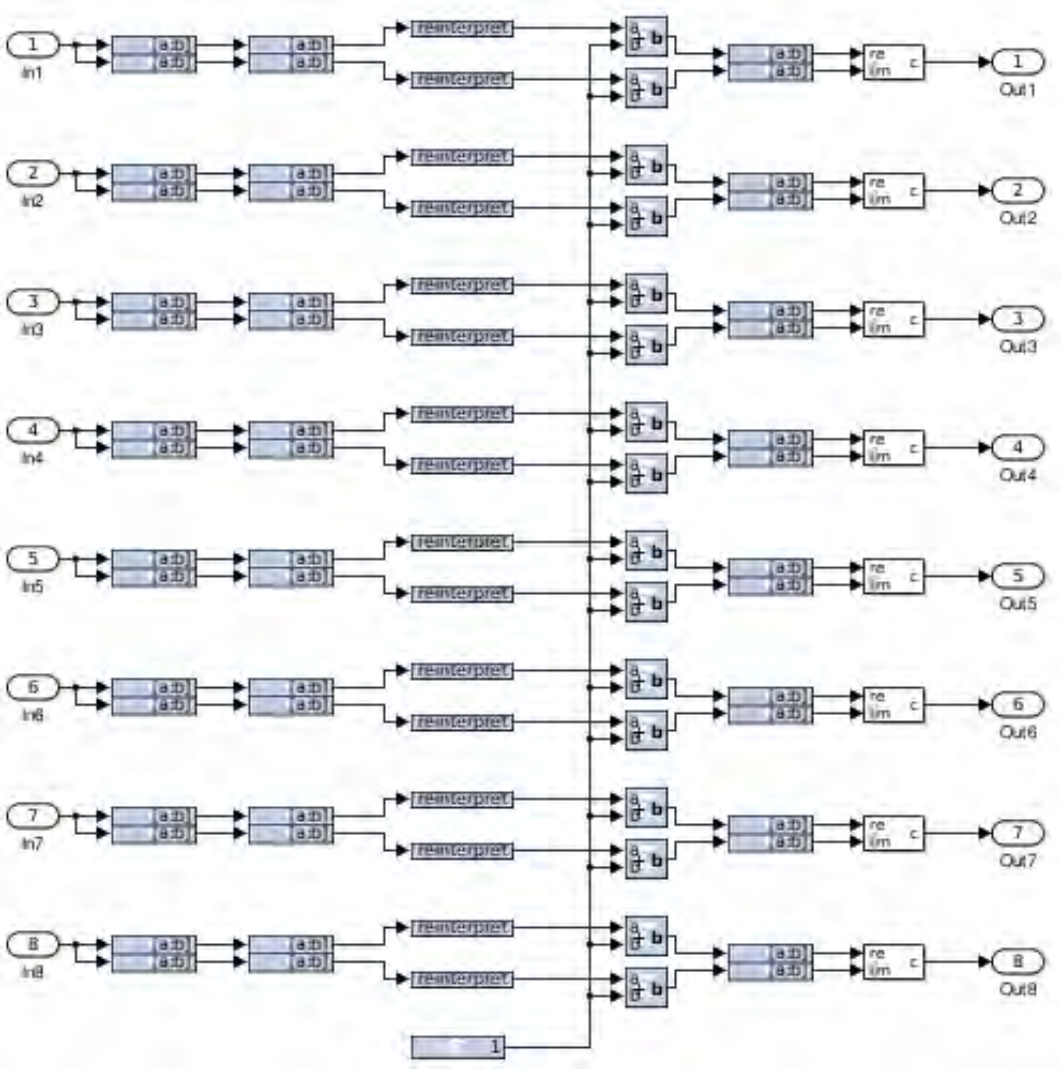


Figure A.16: This bit-reduction architecture is repeated for all possible 8-bit windows. The desired window is selected by specifying the value of the "lsb_select" register.

A.5.4 Data Filter

The frequency data samples outputted from the bit reduction logic is then run through a data filter. This operation controls the packetizer's "valid" control signal and should not be confused with conventional filtering operations that process signals. The data filter consists of a VHDL module that uses user parameters to select a range of frequency bins, among other things, for streaming. The parameters that can be set by the user are as follows:

- `bin_start`
- `bin_end`
- `input_line_start`
- `input_line_end`
- `input_time_start`
- `input_time_end`
- `num_packets`

These parameters can be set during operation by writing to software registers. The names of the registers are the same as the parameters that they represent.

The `freq_start` and `freq_end` registers specify a frequency bin window. Depending on which length FFT is being programmed (256, 512, or 1024), the bit slicing blocks that follow the register outputs must have the width parameters changed to 7, 8, or 9.

The `input_line_start` and `input_line_end` registers specify a range of "rows." Similarly, the `input_time_start` and `input_time_end` registers specify a range of "columns."

Lastly, the `num_packets` register specifies the number of packets to capture. What is meant by a packet is a single time-sample of selected frequency channels across selected rows and columns. Thus the number of packets is equivalent to the desired number of time samples of frequency channels.

There is one additional software register that plays an important role in the acquisition process. The `go` register signals the system to begin sending packets over the 10-GbE link.

Four of the eight outputted streams are sent to the UDP packetizer for 10-GbE transfer, and the other four are saved into a FIFO buffer. This is due to limitations of the 10-GbE core, being able to buffer up a single 64-bit word every clock cycle.

The raw VHDL for this module can be found in `data_acq_16bit_in2.vhd`, and its configuration file is `daq_acq_16bit_in2_config.m`.

A.5.5 UDP Packetizer

The 16-bit frequency samples from the F-Engine are taken to a 10-GbE yellow block for packetization and transfer, as shown in Figure A.17. Each clock cycle, a single 64-bit word consisting of four 16-bit frequency samples is buffered up in the 10-GbE core via the "tx_data" port if the "tx_valid" signal is held high. The packet is sent to the IP address specified at the "tx_dest_ip" port the cycle after the "tx_end_of_frame" signal is asserted.

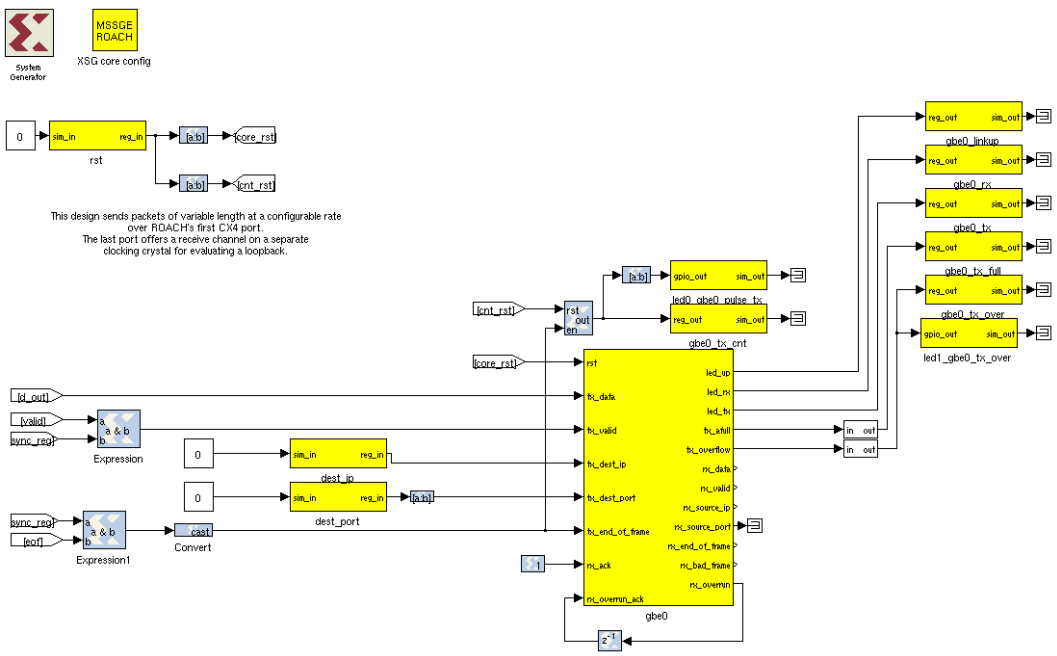


Figure A.17: The 10-GbE core buffers up to 1024 64-bit words for packetization and transfer to a specified IP address.

The 10-GbE buffer can only support up to 1024 64-bit words per packet (these are jumbo packets). If 1025 or more words are buffered before transmission, the "tx_overflow" signal will assert, and the core will lock up, preventing any additional packetization.

Each 64-bit word represents a frequency bin for four rows of a single column. The next word is the next frequency bin for the same four elements. The next four rows are then stored in the same manner. This continues until every column has been treated.

If a capture contained rows 0-3, columns 2-4, and bins 0-9, then the packet structure depicted in Table A.7 would result. If eight rows are requested, then the top four rows of a column are packetized first followed immediately by the bottom four. For example, a capture containing rows 0-8, columns 5-6, and bins 20-45 would have the packet structure shown in Table A.8. Recall that each packet is prefaced with an x64 header, shown in Table 3.2.

A.6 Codes

There are many codes that are used to further process the data received from the x64 acquisition system. This section will focus on the subset of MATLAB codes that are necessary to unpack, correlate, and aggregate packet data. These codes include:

- `correlator.m`
- `batch_correlator.m`
- `aggregate_grid.m`

There are also diagnostic codes that examine raw channelized voltages for spectrometer mode and histogram plotting. These codes will also be addressed and include the following:

Table A.7: Sample x64 Packet Structure - Four Rows

63 downto 48	47 downto 32	31 downto 16	15 downto 0
Element 2, Bin 0	Element 3, Bin 0	Element 18, Bin 0	Element 19, Bin 0
Element 2, Bin 1	Element 3, Bin 1	Element 18, Bin 1	Element 19, Bin 1
⋮	⋮	⋮	⋮
Element 2, Bin 9	Element 3, Bin 9	Element 18, Bin 9	Element 19, Bin 9
Element 10, Bin 0	Element 11, Bin 0	Element 26, Bin 0	Element 27, Bin 0
⋮	⋮	⋮	⋮
Element 10, Bin 9	Element 11, Bin 9	Element 26, Bin 9	Element 27, Bin 9
Element 4, Bin 0	Element 5, Bin 0	Element 20, Bin 0	Element 21, Bin 0
⋮	⋮	⋮	⋮
Element 4, Bin 9	Element 5, Bin 9	Element 20, Bin 9	Element 21, Bin 9

Table A.8: Sample x64 Packet Structure - Eight Rows

63 downto 48	47 downto 32	31 downto 16	15 downto 0
Element 12, Bin 20	Element 13, Bin 20	Element 28, Bin 20	Element 29, Bin 20
Element 12, Bin 21	Element 13, Bin 21	Element 28, Bin 21	Element 29, Bin 21
⋮	⋮	⋮	⋮
Element 12, Bin 45	Element 13, Bin 45	Element 28, Bin 45	Element 29, Bin 45
Element 44, Bin 20	Element 45, Bin 20	Element 60, Bin 20	Element 61, Bin 20
⋮	⋮	⋮	⋮
Element 44, Bin 45	Element 45, Bin 45	Element 60, Bin 45	Element 61, Bin 45
Element 6, Bin 20	Element 7, Bin 20	Element 22, Bin 20	Element 23, Bin 20
⋮	⋮	⋮	⋮
Element 6, Bin 45	Element 7, Bin 45	Element 22, Bin 45	Element 23, Bin 45
Element 38, Bin 20	Element 39, Bin 20	Element 54, Bin 20	Element 55, Bin 20
⋮	⋮	⋮	⋮
Element 38, Bin 45	Element 39, Bin 45	Element 54, Bin 45	Element 55, Bin 45

- `plot_histograms.m`
- `plot_specs.m`

All of these codes can be found in the code archive. They are found in "Codes" folder.

A.6.1 correlator.m

This MATLAB script parses a ".bin" file that was captured using Gulp. It returns an estimated covariance matrix, which is integrated over all available packets in the provided

file. It also can correlate over a variable number of packets and starting at a user-specified packet number.

The correlator is run across all packets using the following command:

```
[Rx, params] = correlator(FILENAME)
```

The variable `Rx` is the resulting estimated covariance matrix with dimensions of $N_p \times N_p \times N_c$, where N_p is the number of PAF elements, and N_c is the number of frequency channels. `FILENAME` is the name of the ".bin" file with the raw samples.

The outputted `params` variable is an array of capture parameters, which are extracted from the x64 header. The first two entries are the starting and ending bin indices (one-indexed) followed by the starting and ending rows and columns. The last entry is the length of the FFT used.

The correlator can be operated on a packet offset from the start by using the following code:

```
[Rx, params] = correlator(FILENAME, 'Starting Packet', S)
```

The argument `S` is the index of the first packet to correlate (one-indexed). The, and `N` is the number of packets that will be integrated.

A variable number of packets can be processed by using the following code:

```
[Rx, params] = correlator(FILENAME, 'Number Packets', N)
```

The argument `N` is the number of packets to be captured.

Lastly, a variable number of frequency bins can be processed by specifying the argument `B` in the following code:

```
[Rx, params] = correlator(FILENAME, 'Number Bins', B)
```

A.6.2 batch_correlator.m

The batch correlator code runs the correlator on all ".bin" files found in a single directory and saves the results into ".mat" files. All of the optional arguments that could be specified in the "correlator.m" script can be specified in "batch_correlator.m."

The batch correlator can be run using the following code:

```
batch_correlator(DIR_NAME)
```

The argument `DIR_NAME` specifies the directory which has the files to be correlated.

The batch correlator by default does not overwrite previously correlated files. To configure the code to overwrite ".mat" files, the following code is used:

```
batch_correlator(DIR_NAME, 'Overwrite')
```

Note that if any correlator-specific arguments are specified, the 'Overwrite' should come after those. For example, if a variable number of frequency bins are to be correlated and the previous correlations should be overwrittten, the following code is used:

```
batch_correlator(DIR_NAME, 'Number Bins', 5, 'Overwrite')
```

The batch correlator can also postpend a string to the end of each outputted file. This is done using the following code:

```
batch_correlator(DIR_NAME, 'Save Postfix', POST_STR)
```

The argument `POST_STR` is the string to be postpended to each outputted ".mat" file.

A.6.3 aggregate_grid.m

This code is used to aggregate correlated files with an acquisition log file. The script parses the log file through regular expressions. Currently, these expressions are intended for Arecibo acquisitions. To configure the code for other telescope message protocols, the regular expressions should be modified. In theory, this should be the only change necessary.

The script is run by specifying a log file name `LOG_FILENAME` using the following code:

```
D = aggregate_grid(LOG_FILENAME)
```

The outputted variable `D` is a struct containing the log file's meta data and correlation matrices. This struct is also saved to a file "default.mat," in the same directory as the constituent ".mat" files. To change the outputted file name, the following code is used:

```
D = aggregate_grid(LOG_FILENAME, 'Save Filename', SAVE_FILENAME)
```

The argument `SAVE_FILENAME` is the new name of the outputted file.

The aggregating code can also translate frequency bin indicies to absolute frequencies by specifying the center frequency of the surveyed passband. This is done using the following code:

```
D = aggregate_grid(LOG_FILENAME, 'Center Frequency', CENTER_FREQ)
```

The argument `CENTER_FREQ` is the center frequency in MHz.

A.6.4 plot_histograms.m

This script plots the first 2000 voltage samples of all 64 inputs into a histogram plot. At the moment, the script only displays the real part of the complex samples. This script is run using the following code:

```
plot_histogram(RAW_FILENAME)
```

The `RAW_FILENAME` argument is the file name for the ".bin" file to be analyzed.

Note: this script will not work without the `examine_raw_data.m` code in the same directory. This code parses raw data files and outputs voltage samples.

A.6.5 plot_specs.m

This code searches a directory for the most recent acquired file and plots its corresponding short-time integrated spectrum. The script allows the user to specify the source directory, analog center frequency, and accumulation length. Further, a cable and antenna ordering can be specified to make the spectrometer plots more meaningful.

The script is easily run from the MATLAB command line use the following code:

```
plot_specs
```

The script needs the `parse_data_x64` code to run.