2016-01-01

Real-Time Beamformer Development and Analysis of Weak Signal Detection with Interference Mitigation for Phased-Array Feed Radio Astronomy

James Michael Brady
Brigham Young University - Provo

Real-Time Beamformer Development and Analysis of Weak Signal Detection with Interference Mitigation for Phased-Array Feed Radio Astronomy

James Michael Brady

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
Michael J. Wirthlin

Department of Electrical and Computer Engineering

Brigham Young University

January 2016

ABSTRACT

Real-Time Beamformer Development and Analysis of Weak Signal Detection with Interference Mitigation for Phased-Array Feed Radio Astronomy

James Michael Brady
Department of Electrical and Computer Engineering, BYU
Master of Science

In recent years, the Brigham Young University (BYU) Radio Astronomy Systems group has developed phased-array feeds and the data acquisition processing systems necessary to perform radio astronomy observations. This thesis describes the development and testing of a real-time digital beamforming system that reduces both the time required to process phased-array feed data and the disk space used to record this data compared to post-processing beamforming systems. A real-data experiment is also discussed in this thesis, which focuses on some of the data post-processing required for one of BYU's data acquisition systems.

Radio-frequency interference mitigation techniques for phased-array feed radio astronomy have been studied for several years, but the effect that these techniques have on weak-signal detection is not well understood. This thesis provides analysis of a simulated weak-source observation for the Green Bank 20-meter telescope and BYU 19 element phased-array feed with radio-frequency interference present. Interference mitigation techniques are shown to reduce the detectability of weak sources compared with the no interference case, but it is also shown that a weak source can be detected that would otherwise be masked by interference.

Keywords: radio astronomy, phased-array feeds, radio-frequency interference, real-time beamforming, digital signal processing, weak source detection

ACKNOWLEDGMENTS

My wife Deborah has done more to help me write this thesis than I could have thought possible. Her encouragement, patience, and support has been instrumental in helping me complete this work. Without her, this could not have been done. Thank you, thank you, and thank you again! My daughter Evangeline has also been a fantastic motivator for me. While she cannot yet speak, she has been such a source of strength for me, especially during the many long nights of research and writing. For her ever-present joy and love I am eternally grateful.

I also wish to say thank you to my parents and family, who have always been there for me when I needed them. My father has been my role model since I was little, and my drive to pursue my degree came from following in his educational footsteps. My wonderful mother has always been there for me, and helped to fill in holes in my understanding, especially in my English and writing courses. My brothers and sisters have each helped me in their own way, even if they do not realize it. Thank you all.

Lastly, I would like to thank my adviser Brian Jeffs who has been a source of so much knowledge; I don't know where I would be without him. Most of what I know about signal processing and radio astronomy has come from his mouth. I would also like to thank my other committee members, Karl Warnick and Mike Wirthlin. Their intellect and teachings have helped me become who I am today.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

For thousands of years, the heavens have been a source of mystery and curiosity. The Sun, Moon and stars were viewed as sources of power, diviners of the future, and echos of the past. More recently, our desire to understand the universe has driven new technology and pulled us into the future. While we reach out to touch moons, planets and even comets, our reach is limited by how fast we can travel. To understand the larger universe outside of our reach, we try to decode the information that the universe sends our way. This information is mostly sent in the form of electromagnetic (EM) radiation. EM radiation varies in energy (which is directly proportional to frequency) from the lowest frequency radio-waves through the visible spectrum and up to the highest energy gamma-rays.

While the radio telescopes used to detect these signals are very sensitive, scanning the sky to find new and interesting astronomical phenomena (such as pulsars [1], fast radio bursts [2], and neutral hydrogen [3]) can take a very long time. Additionally, the radio frequency spectrum is getting more crowded, and some interesting astronomical phenomena do not appear in frequency bands protected by the FCC [4]. Solutions to both of these problems are being addressed by the use of phased-array feeds (PAFs). A PAF can increase the field of view (FOV) of a telescope and allow new opportunities to apply signal processing algorithms which can mitigate radio-frequency interferance (RFI). In this thesis we will discuss signal processing instrumentation for PAFs (specifially a real-time beamformer) and a simulation of PAF radio astronomy in the presence of RFI to determine if interference mitigation techniques allow us to detect weak astronomical sources.

1.1 Radio Astronomy

Until the early 1900's, we could only observe visible light from the heavens, and our understanding of the universe was limited to the phenomena we could see with our eyes and aided by optical telescopes. Ever since Karl Jansky first discovered radio-waves coming from the Milky Way [5], we have been able to observe a much more interesting universe (see Figure 1.1). While the technologies used to view the visible universe has progressed over millenia from simply using our eyes up to the most advanced and largest optical telescopes used today, radio astronomy technology has advanced dramatically in just the past century. The first radio telescopes were only able to see very bright sources, such as Cassiopeia A and Cygnus A (both of which are used as calibration sources because they are so bright) [6]. The most sensitive telescopes today have observed objects twenty million times weaker than those first sources discovered [7]. Radio astronomy lets us see into the invisible universe, including black holes, pulsars, quasars and neutral hydrogen clouds, which are just a few of the many phenomena scientist now study using radio telescopes.

1.2 Radio Astronomy Systems

There are two major classifications for radio telescopes: single-dish telescopes and interferometric imaging array telescopes. A single-dish radio telescope generally consists of three major components: a reflector dish, a feed antenna and a receiver. Examples include the Robert C. Byrd Green Bank Telescope (GBT) [9], the Arecibo observatory [10], and the Effelsberg 100-m radio telescope [11]. The dish collects EM radiation and focuses it onto the feed where a receiver system amplifies the signals, processes, and records them. Interferometric array telescopes are a collection of many single-dish telescopes with relatively wide separation–such as the Very Large Array (VLA) in New Mexico [12] and the Atacama Large Millimeter Array (ALMA) in Chile [13]–or many dish-less aperture array antennas–such as the Low-Frequency Array (LOFAR) based in the Netherlands [14] and the Murchison Widefield Array (MWA) in Australia [15].

Very large reflector dishes, or arrays of smaller dishes, have the advantage of collecting more radiation energy, which allows them to be more sensitive to weak signals. However, even the largest telescopes require highly sensitive receiver systems to accurately record the signals

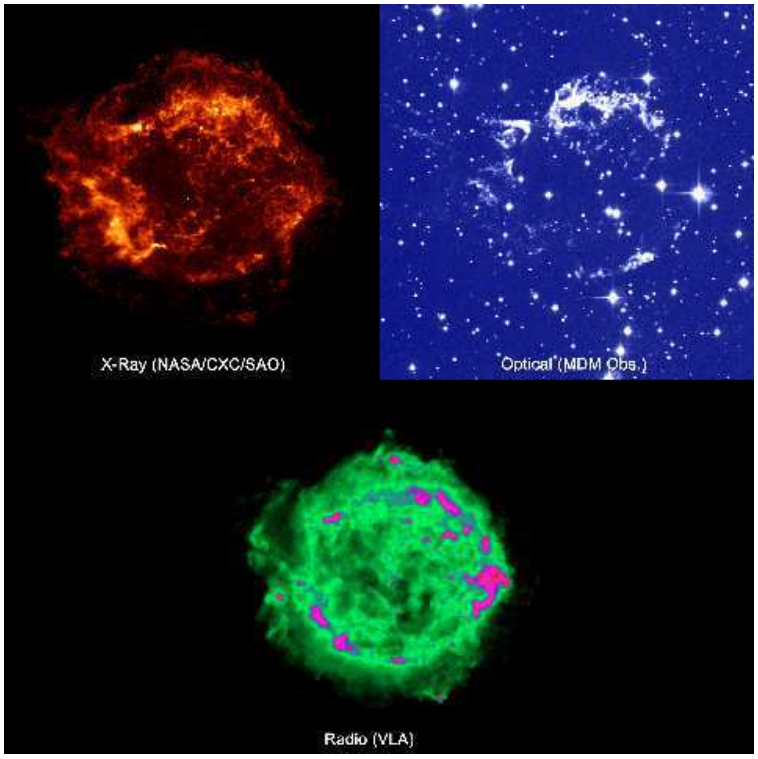


Figure 1.1: By observing different EM wavelengths, more information can be gathered from the supernova remnant Cassiopeia A. There is not much left of this star in the visible spectrum (top right), but the X-ray and radio images (top and bottom left) show the stellar matter expelled during the explosion. The tiny central spot visible in both the X-ray and radio images is possibly a neutron star or a black hole [8]. Image credits: NASA/CXC/SAO, MDM/R. Fesen, and NRAO/AUI/NFS

that are received. These receiver systems have advanced from Jansky's analog pen-and-paper system to digital systems built using high speed analog-to-digital converters (ADCs), field programmable gate arrays (FPGAs), and computers. As we improve these systems, astronomers can observe more of the universe by collecting more data at higher speeds, observing over wider bandwidths, and by applying advanced digital array signal processing techniques and algorithms. This thesis will focus primarily on single-dish instruments, but the principles are also applicable to interferometric imaging arrays.

1.3 Phased-Array Feeds

Most traditional single-dish radio telescopes use a single antenna. A phased-array feed (PAF) is built by combining many antennas into a single feed at the focus of the dish.

An array processing technique known as beamforming can be used with a PAF to digitally steer the telescope. A beam can be formed by generating a linear combination of signals from all of the array elements. This beam can be steered across a wider field of view (compared to a single horn feed) by adjusting the coefficients of the linear combination. This technique provides an opportunity to improve the amount of sky a single telescope can see at once. This thesis will focus primarily on telescopes with phased array feeds.

1.4 RFI Mitigation

Radio-frequency interference (RFI) is a problem for radio astronomers. Weak signals from space can easily be masked by other sources of EM radiation. Common sources of RFI include artificial satellites such as GPS or GLONASS, aircraft radar systems, and even microwave ovens [16]. There are a number of methods for reducing the effect of RFI on an observation. For example, using a PAF, RFI can be canceled by changing the shape of the beam to place a null at the location of the RFI. RFI mitigation techniques could prove to make RFI corrupted data usable, whereas now it is generally discarded.

1.5 Related Work

This thesis discusses the development and testing of a radio astronomy receiver system built using an FPGA-based hardware platform. FPGA-based spectrometers has been developed for use on both single-dish telescopes [17], and interferometric array telescopes [18]. FPGA-based correlators are also being developed and used for phased-array feed telescopes, both single-dish [19], and interferometric arrays [20] [21] [22]. There are also hybrid correlator systems that utilize both FPGAs and graphics processing units (GPUs) [23].

Phased-array feeds have been proposed for use in radio astronomy and continue to be developed. These systems include the 112 element APERTIF array for the Westerbork Synthesis Radio Telescope [24], the 96 element dual-pol ASKAP PAF [25], the fully cryogenic 19 element dual-pol Cornell PAF [26], the 19 element dual-pol NRAO/BYU PAF [27], and the 217 element PAF for the FAST telescope currently under construction [28].

Standard array processing techniques [29] can be applied to radio astronomy interferometry [30]. The beamforming technique was proposed in [31] and has been applied to PAF

radio astronomy [32] [33] [34] with demonstration of these techniques described in [35] [36]. Figures of merit for beamforming arrays are discussed in [37]. Radio-frequency interference techniques for PAF radio astronomy are discussed in [38] [39] and the performance of these techniques are discussed in [40] [41]. Improvements to some RFI mitigation techniques are given in [42] [43]. Oblique subspace projection (an RFI mitigation technique discussed in this thesis) [44] is based on the work of Behrens and Scharf [45].

1.6 Problem Statement

Phased-array feeds show significant promise for radio astronomy, and efforts continue to explore their performance. However, the performance increases associate with PAF radio astronomy (such as an increased field of view) do not come for free. As the number of elements and bandwidth requirements of a PAF increase, the required signal processing becomes more time consuming and difficult to implement. RFI mitigation techniques using PAFs show significant promise in their ability to cancel incoming interference that may mask a desired signal. However, the effect of these techniques on the full PAF-receiver system is not well known. In this thesis I describe an FPGA-based solution for real-time beamforming of a 64-element PAF with a bandwidth of 25 MHz. The effects of RFI mitigation on a typical radio astronomy observation are also explored in this thesis.

1.7 Thesis Contributions

The author's contributions in this thesis are listed below:

- Designed an FPGA-based FFT which has been used in multiple BYU radio astronomy receiver systems.
- Improved a previous student's real-time FPGA-based beamformer design, including extending it to multiple simultaneous beams.
- Incorporated the improved beamformer into a real-time digital receiver system capable of steering seven independent beams.
- Generated software to manage real-time data from the beamformer.

- Incorporated another university's real-time correlator into a real-time interference cancellation system based on the real-time beamformer.
- Generated software to implement an RFI mitigation algorithm in the correlator-beamformer interference cancellation system.
- Tested the validity of data gathered during an experiment at the Arecibo observatory.
- Generated a mosaic image using data gathered during the Arecibo experiment.
- Simulated a weak source observation of the Green Bank 20 meter dish in the presence of RFI.
- Studied the effects of RFI mitigation techniques on a weak source observation and showed that RFI mitigation techniques can allow weak sourced to be detected even in the presence of RFI.
- Provided a recommendation as to when an observation should be attempted with RFI present.
- Provided a recommendation to understand how well RFI mitigation techniques are preforming for long integration observations.

1.8 Thesis Outline

Chapter 2 discusses the theory behind radio astronomy receivers, PAFs, beamforming and RFI mitigation using subspace projection. Chapter 3 discusses the design and implementation of a real-time beamformer and real-time interference cancellation system. Chapter 4 includes the description of and results from a real data experiment at the Aricebo telescope. Chapter 5 discusses a study of weak source detection using a PAF in the presence of RFI and the effects that RFI mitigation techniques have on the receivers stability and sensitivity. Chapter 6 presents conclusions and recommendations based on the material covered in the previous chapters. Appendix A contains detailed instructions on the software that is used to operate the systems described in Chapter 3 and Appendix B describes an improvement to the correlator used in Chapter 4.

Chapter 2

Background

To help the reader gain a greater understanding of phased-array feed radio astronomy, we provide in this chapter a description of the hardware systems, signal processing, algorithms and theory that will be utilized in the remainder of this thesis. Section 2.1 discusses the hardware platforms used in this thesis and the software libraries and tools used to develop systems using those hardware platforms. These tools were developed by a collaborative research group known as the Collaboration for Astronomy Signal Processing and Electronics Research, or CASPER. Section 2.2 introduces that theory and algorithms that are used for PAF signal processing, and Section 2.3 describes how weak astronomical signals can be detected.

2.1 CASPER

The Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) is a worldwide group that work to provide open-source hardware, software and firmware to simplify the process of designing custom radio astronomy systems. Based at the University of California at Berkeley, this collaborative group includes members from laboratories and observatories such as NRAO, CSIRO, SKA-SA as well as many universities and other institutions.

Because many radio astronomy systems follow the same basic functional architecture design, CASPER works to generate basic building blocks that can be used to build a fully operational scientific instrument. These building blocks include the hardware platform with which the digital design will be implemented, IP cores that can be used to design the digital systems, and the software needed to compile these designs [46].

The CASPER hardware platforms used for this thesis include the ROACH (reconfigurable open architecture computing hardware) processing board and it's successor, the ROACH-2. The ROACH board is built around a Xilinx Virtex-5 series FPGA and also houses a number of FPGA interfaces and peripherals along with a CPU and CPU interfaces. The ROACH-2 includes a Virtex-6 series FPGA and more high-speed I/O interfaces than the ROACH.

Firmware developed by CASPER provide IP cores that perform many functions essential to radio astronomy DSP. These include the fast Fourier transfrom (FFT), analog-to-digital converter (ADC) interfaces, 10 Gigabit Ethernet (10GbE), and XAUI tranceivers (the 10 Gigabit Attachment Unit Interface XAUI is used to extend the interface between the MAC and PHY layer of 10GbE [47]). These cores are built as drop-in Simulink blocks, color coded to signify their purpose: "yellow blocks" are I/O interfaces (such as ADCs, shared registers and BRAMs) and "green blocks" implement algorithms (such as the FFT). Chapter 3 of this thesis describes a digital back-end system for a radio astronomy system that was designed using the tools provided by the CASPER group.

2.2 PAF Signal Processing

Using a phased array introduces additional complexity into the radio astronomy system. This complexity can be utilized to give a system greater flexibility. With a traditional horn feed, the beam shape is fixed and the radio telescope can only observe a signal that it is physically pointing at. A PAF allows us to electronically steer the beam, or multiple beams simultaneous beams, and control their shape. This greater flexibility allows astronomers to observe signals arriving from multiple directions simultaneously and to adjust the beam size, shape, and null placement to increase survey speed and perform spatial cancellation of interference. This section will discuss the theory and algorithms required to perform this digitally.

2.2.1 PAF Signal model

Assuming a narrowband M element PAF, the complex baseband data vector at time sample n is given as

$$\mathbf{x}[n] = \mathbf{v}s[n] + \boldsymbol{\eta}[n], \tag{2.1}$$

where $\mathbf{s}[n]$ is the signal of interest (SOI), $\mathbf{v}$ is the array response vector to a point source in the direction of the SOI, and $\boldsymbol{\eta}[n]$ is the total system noise seen at the array. For an M element array, $\mathbf{x}[n]$ is an $M \times 1$ complex vector. Assuming the signals $\mathbf{s}[n]$ and $\boldsymbol{\eta}[n]$ are independent zero mean wide-sense stationary random processes, the array covariance matrix is defined as

$$\begin{aligned} \mathbf{R} &= E\left\{\mathbf{x}[n]\mathbf{x}^H[n]\right\} \\ &= E\left\{\mathbf{v}s[n] + \boldsymbol{\eta}[n]\right\} \\ &= \mathbf{R}_s + \mathbf{R}_\eta, \end{aligned} \tag{2.2}$$

where $E\{\bullet\}$ denotes the expected value and $\mathbf{R}$ is $M \times M$. If we also assume the SOI is a point source with a power of σ_s^2, we see that

$$\mathbf{R}_s = \sigma_s^2 \mathbf{v}\mathbf{v}^H. \tag{2.3}$$

While Equation 2.2 represents the true array covariance matrix, we can only estimate this matrix during an observation. An estimate is obtained by integrating over some short time integration (STI) window of length N

$$\hat{\mathbf{R}} = \frac{1}{L}\sum_{n=0}^{N-1} \mathbf{x}[n]\mathbf{x}^H[n] = \frac{1}{L}\mathbf{X}\mathbf{X}^H, \tag{2.4}$$

where $\mathbf{X}$ is the $M \times N$ data matrix, with one column per sample, for an N-sample long STI data window.

2.2.2 Beamforming

As stated before, linear combinations of PAF element signals can be used to adjust the array's response to a signal. This process is called beamforming, and pattern response is controlled by the vector $\mathbf{w}$, called the beamformer coefficients or weights. The beamformed signal is given by [31]

$$y[n] = \mathbf{w}^H \mathbf{x}[n]. \tag{2.5}$$

We refer to this form as a "time-domain beamformer". A "post-correlation beamformer" can also be used to estimate the beamformed power within some STI by applying the beamformer weights to the estimated array covariance matrix

$$\begin{aligned} |\hat{y}|^2 &= \frac{1}{L}\mathbf{Y}\mathbf{Y}^H \\ &= \frac{1}{L}(\mathbf{w}^H\mathbf{X})(\mathbf{w}^H\mathbf{X})^H \\ &= \frac{1}{L}\mathbf{w}^H\mathbf{X}\mathbf{X}^H\mathbf{w} \\ &= \mathbf{w}^H\hat{\mathbf{R}}\mathbf{w}. \end{aligned} \tag{2.6}$$

Both time-domain and post-correlation beamforming will be used in this thesis.

2.2.3 Calibration

One of the challenges of beamforming is designing an effective set of beamformer weights. There are many ways to generate beamformer weights, and each method aims to achieve a different goal. Weights can be calculated to constrain the beamformer output variance using a linearly-constrained minimum variance (LCMV) beamformer [48], to minimize the computation required to generate the weights, as in the conjugate field match (CFM) beamformer [49], or to maximize SNR as in the max-SNR beamformer [50]. Most of these beamformers require knowledge of the array response vector $\mathbf{v}(\Omega)$, in every direction Ω where one wishes to steer the beampattern main lobe, and in every direction where sidelobe

or mainlobe respone constraints are the be placed. A PAF calibration procedure has been proposed in [32] to estimate these vectors.

To estimate the array response vector $\mathbf{v}$ for some direction Ω_k, the telescope is steered to this angular separation relative to a calibration source (common calibration sources include strong point-sources such as Cassiopeia A or Cygnus A). A signal plus noise covariance matrix $\hat{\mathbf{R}}_i$ is obtained at this location. The dish is then steered to another location away from the calibration source to obtain a noise-only covariance matrix $\hat{\mathbf{R}}_n$. The array response vector is estimated as

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

where $\mathbf{u}_{max}$ is given by the dominant solution to the generalized eigenvalue problem

$$\hat{\mathbf{R}}_n \mathbf{u}_m ax = \lambda_{max} \hat{\mathbf{R}}_n \mathbf{u}_{max}. \tag{2.8}$$

While a single array response vector is adequate to calculate a set of beamformer weights, many sets of weights are often desired to allow the PAF to steer to multiple locations. The calibration procedure is repeated for many directions in a grid pattern. These calibration girds are often vary large to thoroughly sample the field of view (FOV) of the feed. Chapter 4 describes an experiment where calibration grids of 225 and 961 different pointings were acquired in square grids of 15 and 31 rows and columns respectively.

This thesis is primarily concerned with maximum-SNR beamforming. This beamformer is defined as [33]

$$\mathbf{w}_{\text{maxSNR,i}} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^H \mathbf{R}_s \mathbf{w}}{\mathbf{w}^H \mathbf{R}_n \mathbf{w}}, \tag{2.9}$$

which is solved using the generalized eigenvector problem

$$\hat{\mathbf{R}}_{s,i} \mathbf{w}_{\text{maxSNR,i}} = \lambda_{max} \hat{\mathbf{R}}_n \mathbf{w}_{\text{maxSNR}}. \tag{2.10}$$

2.2.4 Radio Camera Imaging

The idea for a "radio camera" was put forth by Steinberg [51] and involves using an array of antennas to gather more information about an object than a single antenna is able

to. This idea has been adapted and used with PAFs to generate images of astronomical objects by forming a beam at each calibration grid pointing, and treating the power in each beam as the image pixel intensity [52][53].

Once a calibration grid has been obtained and beamformer weights have been calculated for each pointing, a radio camera image can be produced by applying all sets of beamformer weights, $\mathbf{w}_k$, to the data collected from a single dish pointing. In so doing, images of astronomical objects can be formed much more quickly as the dish does not need to be physically steered to and integrate the received signal at each on-sky pixel location. However, care must be taken to ensure that the pixels generated from the radio camera are intelligently spaced within the PAF's FOV. A densely sampled calibration grid may not give better resolution than a calibration grid with greater separation between dish pointings (e.g. pointings separated by the half-power beam width (HPBW)), and would require much more computation to form the image in addition to the extra telescope time required to generate the calibration grid.

2.2.5 RFI Mitigation

Radio-frequency interference is EM radiation from sources other than the cosmic objects the astronomer is interested in. This interference can often mask the signals the astronomer *is* interested in. Common methods of reducing the impact of RFI are to excise the frequency band that is corrupted by RFI and discard it, or to discard (called blanking) corrupted time samples $\mathbf{x}[n]$ for intermittent RFI. PAF signal processing provides additional methods of RFI mitigation that do not necessarily require a loss of data. Subspace projection is a method that relies on projecting the received signal onto a vector subspace that does not contain the interference subspace. Suppose the received signal is of the form

$$\mathbf{x}[n] = \mathbf{v_s} x_s[n] + \mathbf{v_i} x_i[n] + \boldsymbol{\eta}[n], \tag{2.11}$$

where $x_s[n]$, x_i, and $\boldsymbol{\eta}[n]$ are the random processes representing the source, interference and noise amplitudes and $\mathbf{v_s}$ and $\mathbf{v_i}$ are the array response vectors to the signal and interference. A projection matrix $\mathbf{P}$ applied to the received signal can cancel the interference if $\mathbf{P} \perp \mathbf{v}_i$

such that $\mathbf{P}\mathbf{v}_i = 0$

$$\begin{aligned}\mathbf{P}\mathbf{x}[n] &= \mathbf{P}\mathbf{v_s}x_s[n] + \mathbf{P}\mathbf{v_i}x_i[n] + \mathbf{P}\boldsymbol{\eta}[n] \\ \mathbf{P}\mathbf{x}[n] &= \mathbf{P}\mathbf{v_s}x_s[n] + \mathbf{P}\boldsymbol{\eta}[n].\end{aligned} \tag{2.12}$$

This projection matrix can be used to modify the received signal $\mathbf{x}[n]$, the estimated correlation matrix $\hat{\mathbf{R}}$ or the beamformer weights $\mathbf{w}$. This can be seen by applying the projection matrix to the received signal in equations (2.5) and (2.6) and using some matrix algebra to rearrange the equations slightly:

$$\begin{aligned} y[n] &= \mathbf{w}^H\mathbf{P}\mathbf{x}[n] \\ &= \mathbf{w}^H\mathbf{x}_p[n], \end{aligned} \qquad \begin{aligned} y[n] &= \mathbf{w}^H\mathbf{P}\mathbf{x}[n] \\ &= (\mathbf{P}^H\mathbf{w})^H\mathbf{x}[n] \\ &= \mathbf{w}_p^H\mathbf{x}[n], \end{aligned} \tag{2.13}$$

$$\begin{aligned} |\hat{y}|^2 &= \frac{1}{L}\mathbf{Y}\mathbf{Y}^H \\ &= \frac{1}{L}(\mathbf{w}^H\mathbf{P}\mathbf{X})(\mathbf{w}^H\mathbf{P}\mathbf{X})^H \\ &= \frac{1}{L}\mathbf{w}^H(\mathbf{P}\mathbf{X}\mathbf{X}^H\mathbf{P}^H)\mathbf{w} \\ &= \mathbf{w}^H(\mathbf{P}\hat{\mathbf{R}}\mathbf{P}^H)\mathbf{w} \\ &= \mathbf{w}^H\hat{\mathbf{R}}_\mathbf{p}\mathbf{w}, \end{aligned} \qquad \begin{aligned} |\hat{y}|^2 &= \frac{1}{L}\mathbf{Y}\mathbf{Y}^H \\ &= \frac{1}{L}(\mathbf{w}^H\mathbf{P}\mathbf{X})(\mathbf{w}^H\mathbf{P}\mathbf{X})^H \\ &= \frac{1}{L}\mathbf{w}^H(\mathbf{P}\mathbf{X}\mathbf{X}^H\mathbf{P}^H)\mathbf{w} \\ &= (\mathbf{w}^H\mathbf{P})\hat{\mathbf{R}}(\mathbf{P}^H\mathbf{w}) \\ &= \mathbf{w}_p^H\hat{\mathbf{R}}\mathbf{w}_p, \end{aligned} \tag{2.14}$$

where $\mathbf{x}_p[n] = \mathbf{P}\mathbf{x}[n]$, $\mathbf{w}_p = \mathbf{P}^H\mathbf{w}$, and $\hat{\mathbf{R}}_p = \mathbf{P}\hat{\mathbf{R}}\mathbf{P}^H$.

This general approach is referred to as "subspace projection" interference canceling. It is preferred over variance minimization methods such as MVDR [29] in radio astronomy because it is "zero forcing," and produces very deep nulls. This is important because the SNR level may be many tens of dB below zero. The null must drive the RFI well below the noise floor [38].

Orthogonal Projection

The projection matrix $\mathbf{P}$ can be estimated using principal component analysis (PCA) [38]. Assuming the interfering signal power is much greater than that of the SOI, the subspace

spanned by Q interferers is estimated by finding the set of eigenvectors that correspond to the Q dominant eigenvalues of the total received covariance matrix $\mathbf{R}$. The eigenvalue decomposition of $\mathbf{R}$ is given by

$$\mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}, \tag{2.15}$$

where $\mathbf{U}$ is the $M \times M$ unitary matrix with eigenvectors of $\mathbf{R}$ as its columns and $\mathbf{\Lambda}$ is the $M \times M$ matrix whose diagonal entries are the eigenvalues of $\mathbf{R}$. The orthogonal projection matrix $\mathbf{P}$ is given by [38]

$$\mathbf{P} = \mathbf{I} - \mathbf{U_q}\mathbf{U_q}^H, \tag{2.16}$$

where $\mathbf{I}$ is the $M \times M$ identity matrix and $\mathbf{U_q}$ is the $M \times Q$ matrix comprised of the Q dominant eigenvectors of the matrix $\mathbf{U}$. Methods such as cross-subspace [39] projection include tracking the interference with an auxiliary antenna to produce a better estimate of the interference subspace.

Rarely is the interference subspace truly orthogonal to the signal subspace. Using the orthogonal projection given by (2.16) can introduce a bias into the received signal [40]. This bias should be corrected in order to preserve the true nature of the SOI. This bias correction can also preserve, on average, the beampattern sidelobe structure while tracking moving interferers [54].

Oblique Projection

If instead of an orthogonal projection we use an oblique projection, the signal subspace can be preserved. In order to perform an oblique projection, both the signal and interference subspaces must be known. The signal subspace can be produced using the calibration technique described in 2.2.3, and the interference subspace is estimated using PCA. The oblique projection matrix is formed by setting its range to be the signal subspace and its kernel to be the interference subspace. The oblique projection matrix is given by [44]

$$\mathbf{P}_{obl} = \mathbf{S}_s(\mathbf{S}_s^H\mathbf{P}^\perp\mathbf{S}_s)^{-1}\mathbf{S}_s^H\mathbf{P}^\perp, \tag{2.17}$$

where $\mathbf{S}_s$ is the signal subspace and $\mathbf{P}^{\perp}$ is the orthogonal projection matrix described in section 2.2.5. In the case of a single point source SOI, $\mathbf{S}_s = \frac{\mathbf{v}_s}{(\mathbf{v}_s^H \mathbf{V}_s)^{1/2}}$. This equation can be viewed as a correction which undoes the damage to the SOI subspace done by an orthogonal projection when the two subspaces are not actually orthogonal.

2.2.6 Performance Metrics

The primary metric of concern in this thesis is the system noise temperature, T_{sys}. This is given by [55]

$$T_{\text{sys}} = T_{\text{iso}} \frac{\mathbf{w}^H \mathbf{R}_n \mathbf{w}}{\mathbf{w}^H (\mathbf{R}_{iso} + \mathbf{R}_{loss}) \mathbf{w}}, \tag{2.18}$$

where T_{sys} and T_{iso} are the system temperature and isotropic noise environment temperature, respectively. R_{iso} is the covariance matrix of the isolated array response to spatially isotropic noise, and R_{loss} is due to antenna resistive losses. T_{sys} is one of the main factors to consider when determining how weak a signal the telescope can detect (see section 2.3). For observations with interference, the contribution to T_{sys} from RFI and the effects of subspace projection on the system noise can be calculated by recognizing that $\mathbf{R}_n$ is made up of the RFI covariance matrix $\mathbf{R}_i$ and the noise covariance matrix $\mathbf{R}_{noise}$.

$$\begin{aligned} T_{\text{sys}} &= T_{\text{iso}} \frac{\mathbf{w}^H \mathbf{R}_n \mathbf{w}}{\mathbf{w}^H (\mathbf{R}_{iso} + \mathbf{R}_{loss}) \mathbf{w}} \\ &= T_{\text{iso}} \frac{\mathbf{w}^H (\mathbf{R}_{noise} + \mathbf{R}_i) \mathbf{w}}{\mathbf{w}^H (\mathbf{R}_{iso} + \mathbf{R}_{loss}) \mathbf{w}} \\ T_{\text{sys,rfi}} + T_{\text{sys,noise}} &= T_{\text{iso}} \frac{\mathbf{w}^H \mathbf{R}_i \mathbf{w}}{\mathbf{w}^H (\mathbf{R}_{iso} + \mathbf{R}_{loss}) \mathbf{w}} + T_{\text{iso}} \frac{\mathbf{w}^H \mathbf{R}_{noise} \mathbf{w}}{\mathbf{w}^H (\mathbf{R}_{iso} + \mathbf{R}_{loss}) \mathbf{w}}, \end{aligned} \tag{2.19}$$

where $T_{\text{sys,rfi}}$ and $T_{\text{sys,noise}}$ are the contributions to T_{sys} from the interference and noise respectively. To calculate the effect of subspace projection, either the beamformer weights $\mathbf{w}$ or the covariance matrices $\mathbf{R}_i, \mathbf{R}_{noise}$ can be replaced by their post-subspace projection counterparts $\mathbf{w}_p, \mathbf{R}_{p,i}$, and $\mathbf{R}_{p,noise}$ respectively.

2.3 Weak Source Detection

In radio astronomy the SOI signal levels are typically many tens of dB below the noise floor corresponding to T_{sys}, even with the largest and most sensitive instruments. One

technique used to detect these weak sources is an on-off subtraction. To estimate the power in a weak source using a post-correlation beamformer, we require the signal correlation matrix $\mathbf{R}_s$. We first observe the SOI to obtain our "on pointing"

$$\mathbf{R}_{on} = \mathbf{R}_s + \mathbf{R}_n. \tag{2.20}$$

We then steer the telescope to a region of sky with no signal to collect an "off pointing"

$$\mathbf{R}_{off} = \mathbf{R}_n. \tag{2.21}$$

Subtracting the two will give us an estimate of the signal covariance from which we can determine the power received by the weak source

$$\hat{\mathbf{R}}_s = \mathbf{R}_{on} - \mathbf{R}_{off}. \tag{2.22}$$

This on-off subtraction can be performed for both broadband and narrow-band signals. However, for narrowband signals masked by broadband noise, only a single "on pointing" is necessary. The frequency bands outside the SOI bandwidth are used to form a virtual "off pointing" reference.

In practice we use instead $\hat{\mathbf{R}}_{on}$ and $\hat{\mathbf{R}}_{off}$ which are sample estimates over some integration window of length τ. In the next section, the relationship between τ, T_{sys}, and the limit of detectable signal power will be discussed.

2.3.1 Radiometer Stability

The receiver system for many radio astronomy instruments consists of five main components:

- A low-noise amplifier (LNA). This amplifies the weak signals received by the antenna to a more appropriate level for further processing. Because the noise figure of a cascaded system is most heavily influenced by the first stage, this amplifier is usually cooled to extremely low temperatures to reduce thermal noise.

- A mixer to bring the RF frequency band of interest down to a more convenient intermediate frequency (IF) band.
- An IF amplifier and filter to further groom the signal to the desired IF band.
- A square law detector which outputs a voltage proportional to the input power.
- An integrator which averages the detector output over some time interval τ.

The receiver system used to observe weak astronomical signals is a major factor in determining how weak of a signal we are able to detect. Each component in the receiver chain adds additional noise to the system [56], which can mask a weak SOI. Other sources of noise also effect to the overall system temperature T_{sys}. These include sky noise, spillover noise and losses in antenna, cables and other components [55] [32].

In order to detect a weak signal that produces an increase in antenna temperature of T_s, the variation of $\hat{T}_{\text{sys}}$ (the estimated T_{sys} over some long-term integration (LTI) window) must be sufficiently small so that we do not confuse a sudden increase in thermal noise with the SOI. The detrimental effects of a high T_{sys} can be overcome by sufficient integration, assuming wide-sense stationary statistics for $x_s[n], x_i[n]$ and $\eta[n]$. As N independent voltage samples of a signal with bandwidth β are averaged over a time τ, the variance of the voltage estimate is reduced by the factor $N = \beta\tau$. This leads to a relationship between the standard deviation of our estimate of T_{sys} (ΔT_{sys}), the true T_{sys} and the amount of integration [57]

$$\frac{\Delta T_{\text{sys}}}{T_{\text{sys}}} = \frac{1}{\sqrt{\beta\tau}}. \tag{2.23}$$

This equation assumes that there is no time-dependent gain variation of the receiver system. In the case that this is not true, additional variation is introduced to T_{sys}. Because any gain variation is independent of the noise variation, the two can be combined into

$$\Delta T = \sqrt{\Delta T_N^2 + \Delta T_G^2}, \tag{2.24}$$

where ΔT, ΔT_N, and ΔT_G are the total uncertainty in the system, the uncertainty due to noise, and the uncertainty due to gain variations respectively [56].

In order to detect a SOI with power (expressed in units of kelvin) T_s, we require that that $\Delta T < T_s$. This is generally extended to ensure a 5σ detection [58]

$$T_s > 5\sqrt{\Delta T_N^2 + \Delta T_G^2}. \tag{2.25}$$

Allan Deviation

To reduce the variance of the sample estimation error, the received signals are integrated for some LTI window. This reduction relies on the assumption that the system temperature is dominated by Gaussian white noise. If the system temperature begins to be dominated by some other factor, such as time-varying gain, our ability to reduce sample estimation error is diminished. To determine what factor is dominant in equation 2.24, the Allan deviation method can be applied to the sampled data. This method has been used to measure stability of atomic clocks [59], and can also been applied to the stability of microwave radiometers. The Allan deviation can be calculated for some sampling period τ_0 with N measurements of T_i [60]

$$\sigma_y(\tau_0) = \sqrt{\frac{\sum_{i=1}^{N-1}(T_{i+1} - T_i)^2}{2(N-1)}}. \tag{2.26}$$

To calculate the Allan deviation for various LTI lengths τ, n adjacent values are averaged so that $\tau = n\tau_0$, giving

$$\sigma_y(\tau_0) = \sqrt{\frac{\sum_{i=1}^{N-2n+1}(T_{i+2n} - 2T_{i+n} + T_i)^2}{2(\tau^2(N-2n+1)}}. \tag{2.27}$$

A log-log plot of the Allan deviation versus LTI length provides information as to what type of noise is dominating for various integration lengths. The Allan deviation is defined so that Gaussian white noise will produce a slope of -0.5. flicker noise, random-walk noise, and steady drift will have a slope of 0, 0.5 and 1 respectively. The "Allan time," defined as the minimum point of the Allan variance, provides the maximum integration length that can be used while still receiving the benefits of integration.

Chapter 3

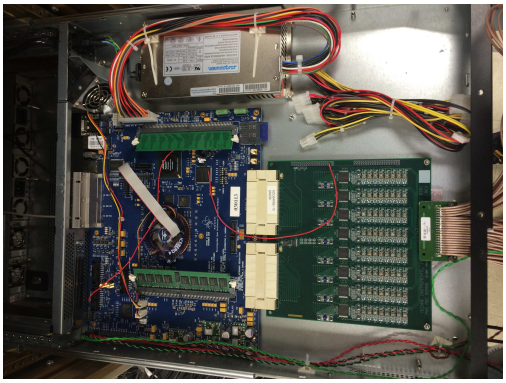
Development of the x64 Real-Time Beamformer

Some of the drawbacks of phased array feed (PAF) radio astronomy include the extra data load that comes with increasing the number of antenna elements, and the extra computational burden that comes with array signal processing. The x64 Real-Time Beamformer (xRTB) performs real-time beamforming which, when compared to post-correlation beamforming, has two major benefits: reduction of both computational burden and the amount of data to be stored on disk.

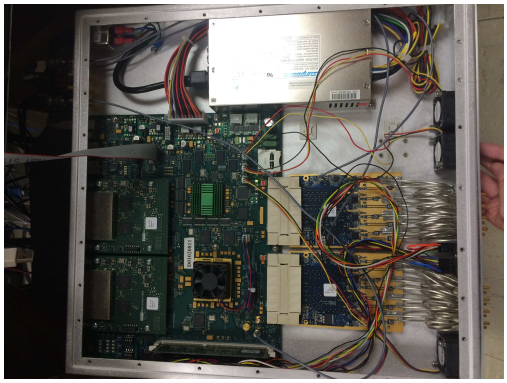
The xRTB was designed for the ROACH platform. The ROACH board is a Virtex-5 FPGA system that was designed by the UC Berkeley CASPER group to be used for high speed radio astronomy signal processing. Each ROACH board consists of a Virtex-5 FPGA and a PowerPC, along with numerous I/O interfaces including four 10Gb Ethernet and two Z-DOK connectors. Since the xRTB firmware was designed, a new ROACH-2 has been developed that includes a Virtex-6 FPGA and additional 10Gb Ethernet capacity. To interface with the ROACH's PowerPC, another computer is used as a host and communicates through a 1 GbE connection. This host computer can program the FPGA and read from and write to memory on board the ROACH that is shared with the FPGA. This shared memory can be used to monitor and control the FPGA and to transfer data between the FPGA and the host computer.

The xRTB (Figure 3.2(a)) generates beamformed data for up to 7 simultaneous beams. Using a 64-channel, 12-bit ADC (64ADCx64-12, developed by Rick Raffanti) the xRTB can operate using up to 64 channels sampled at 50 Megasamples per second (Msps). The beamformed data is saved to the host PC in a binary .mat file, and Matlab can be used to view the data in real time or to review the data at a later time. The xRTB can also be used as part of a Real Time Interference Cancellation (RTIC) system (Figure 3.2(b)). This

system requires multiple ROACH boards running simultaneously. The RTIC system uses the xRTB (one ROACH board) and the Real Time Correlator (two ROACH boards) along with MATLAB and Python scripts that perform the subspace projection and display output data.

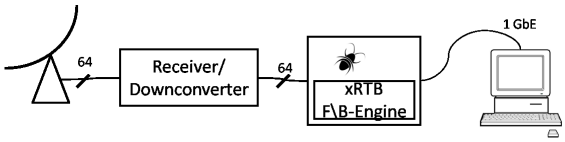


(a) Inside the ROACH enclosure.

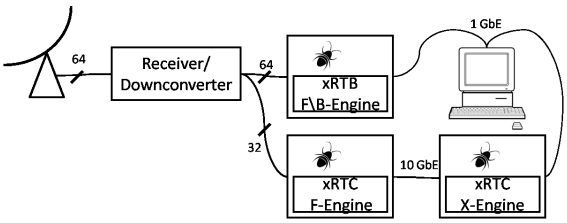


(b) Inside the ROACH-2 enclosure.

Figure 3.1: Both the ROACH and ROACH-2 boards were designed for high-speed FPGA-based radio astronomy signal processing. This ROACH board (a) is configured for the xRTB with a 64-channel 50 Msps ADC connected to the two Z-DOK connectors. This ROACH-2 (b) has two 16/8/4-channel 250/500/1000 Msps ADCs on the right and two 4x10GbE cards on the left.



(a) Block diagram of the xRTB system.



(b) Block diagram of the RTIC system.

Figure 3.2: In both the xRTB system (a) and the RTIC system (b), the PAF elements are connected to the receiver/downconverter boards and then sent to the ROACH boards. The xRTB uses a single ROACH connected to a PC via 1 GbE. The RTIC system uses three ROACHs, two of which communicate using, a 10 Gb communication connection via XAUI.

To operate the xRTB, the following steps are taken (in order):

- Calculate Beamformer weights for up to 7 beams
- Run the Beamformer using the generated weight files
- View the beamformed data

To operate the RTIC system, the following steps are taken (in order):

- Calculate initial Beamformer weights for 1 beam.
- Initialize F/X Engine.
- Initialize subspace projection function.
- Run the Beamformer.
- Start the Matlab function to view beamformed data.
- Using the F/X Engine, periodically save new correlation data.

See Appendix A for detailed operational notes.

3.1 xRTB Design

The xRTB was designed using a combination of VHDL and Simulink, and relies heavily on the CASPER (Collaboration for Astronomy Signal Processing and Electronics Research) toolflow. The CASPER toolflow includes a number of libraries and Simulink blocks that have been designed and optimized for radio astronomy systems implemented on the Virtex-based ROACH boards. These libraries include high-speed ADC and I/O interfaces, efficient FFT implementations and polyphase filter banks. The CASPER toolflow also includes the software necessary to communicate with the ROACH board from a host PC through a 1Gb Ethernet link.

There are two major components in the xRTB design: the F-Engine and the B-Engine. These two components perform the bulk of the computation and take up a majority of the FPGA fabric. Four minor subsystems handle control logic, pre-beamforming data reduction,

B-engine coefficient updating and the x64 ADC. The F-Engine, minor subsystems, and control software were designed by this author, using other CASPER designs as a framework for the F-Engine. A preliminary beamformer was designed by BYU student Megan Fuller. This was then expanded, incorporated into the xRTB system, and tested by this author. A description of the development and testing of these components follows.

3.1.1 xRTB F-Engine:

The purpose of the F-Engine is to frequency channelize the data coming in on each of the 64 ADC inputs. With an ADC sample rate of 50 MHz and an FFT length of 512, this F-Engine produces 256 frequency channels with a bandwidth of 97.7 kHz each. This frequency channelized data is then passed to a broadband beamformer contained in the B-Engine.

The F-Engine for the xRTB was designed using the polyphase filter bank (PFB) technique (see [61], [62]) to reduce spectral leakage. Conceptually, this techniques involves passing a signal through a bank of low-pass polyphase filters which are frequency shifted appropriately by an FFT [63]. This produces many bandpass filters that perform the frequency channelization and results in much less spectral leakage than a standard FFT alone. To implement this technique on the ROACH hardware, a number of time windows are buffered up, filtered, and added point-by-point to produce a single filtered window of data that is then passed to the FFT [61]. The two downsides to this technique are the extra time needed to fill the buffers and the extra FPGA resources consumed by the PFB. Implemented on an FPGA, the PFB technique generally uses 50% more resources than a standard FFT [61].

3.1.2 xRTB B-Engine:

The B-Engine performs the real-time beamforming and the spectral power accumulation. The resulting beamformed spectra are stored in a shared BRAM to be read by the ROACH host PC.

At the heart of the B-Engine is a single 64 input beamformer. A block diagram of this beamformer is shown in Figure 3.3. In comparing this block diagram to the beamformer equation given in section 2.2.2

$$y[n] = \mathbf{w}^H \mathbf{x}[n], \tag{3.1}$$

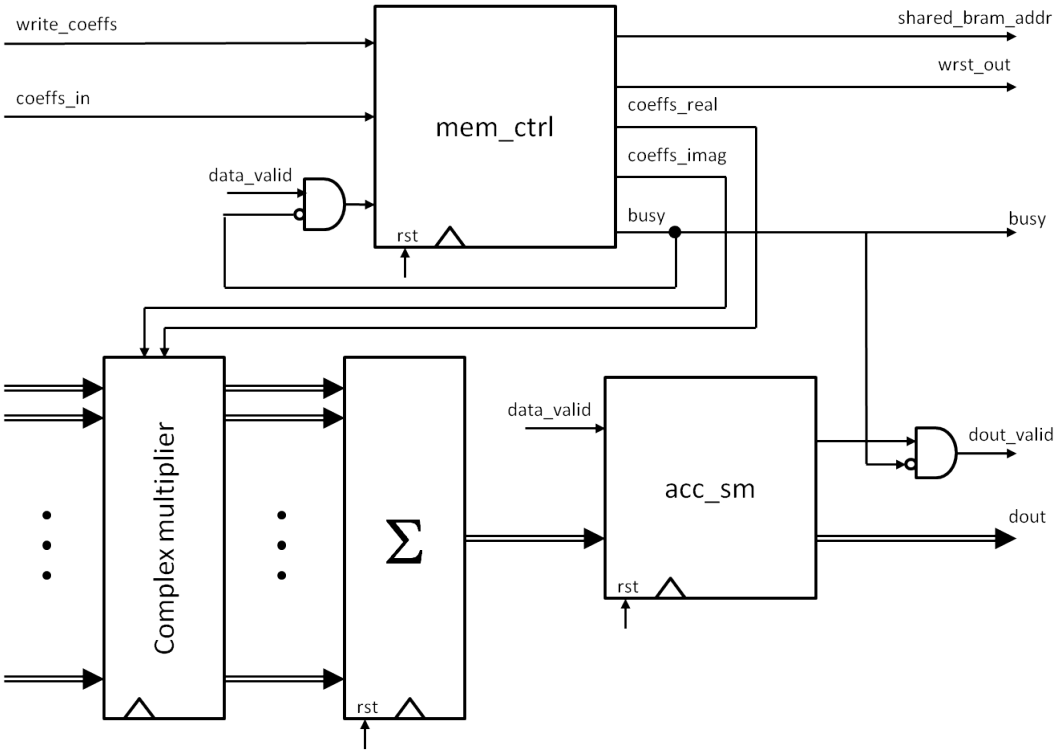


Figure 3.3: Block diagram of the B-Engine.

recall that a vector multiplication operation can be described as

$$\mathbf{ab} = \sum_i a_i b_i. \tag{3.2}$$

The block diagram shows the beamformer coefficients $\mathbf{w}$ (stored in memory within the mem_ctrl block) multiplied by the array data vector $\mathbf{x}[n]$ (in the complex multiplier block), added together (in the block marked Σ), and passed to an accumulator (acc_sm). Because the frequency channelized data streams into the B-Engine one channel at a time, this single narrowband beamformer can be used to beamform all frequency channels by simply changing the coefficients appropriately as each new frequency channel arrives at the input to the beamformer.

The majority of the B-Engine was written in VHDL and contains two Xilinx IP cores generated using Xilinx Core Generator. The main components of the B-Engine are the compelx multiplier (from CoreGen), the memory control module (mem_ctrl.vhdl) and

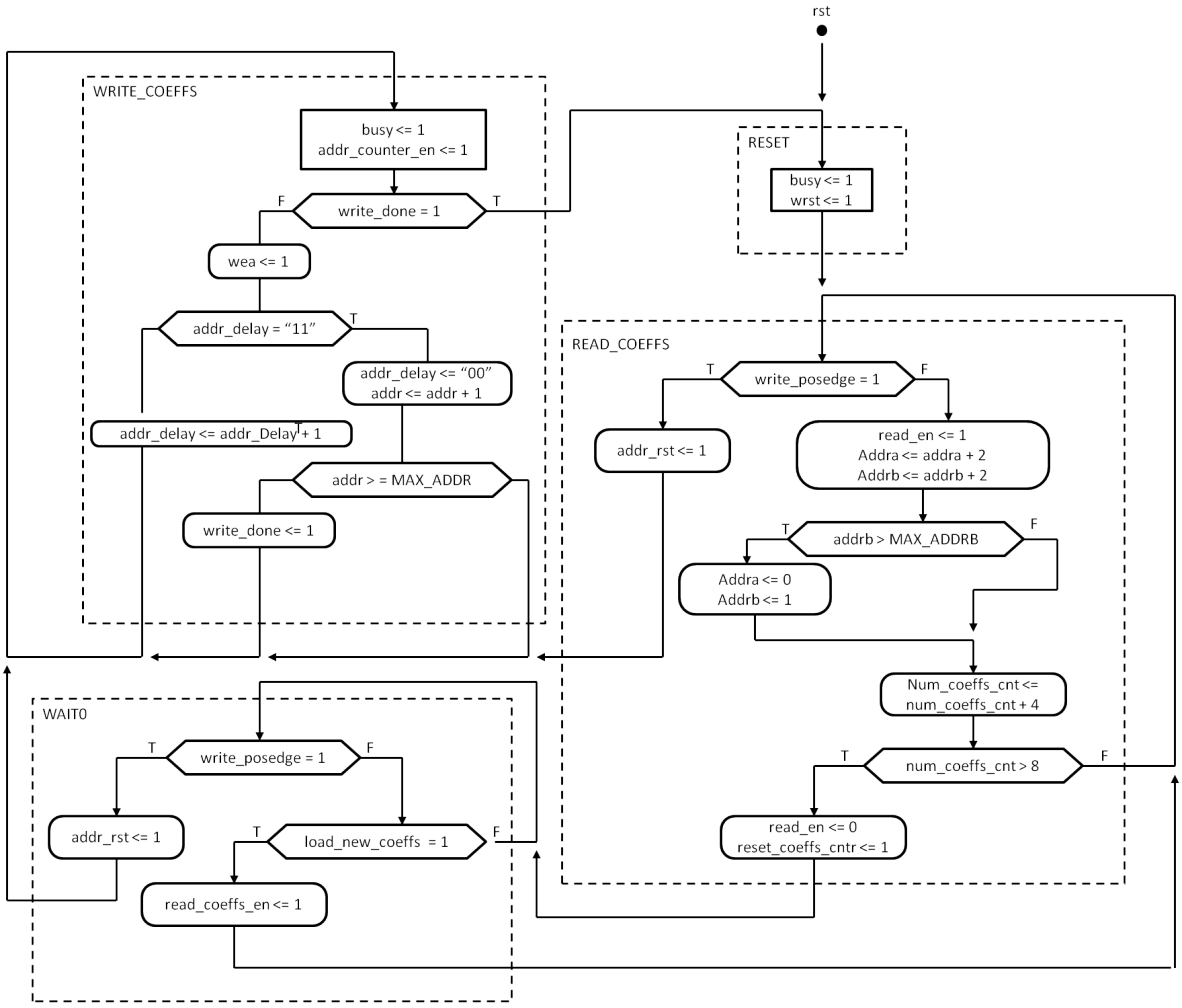


Figure 3.4: ASM chart for the mem_ctrl block.

the accumulation module (acc_sm.vhdl). These components are all included in the VHDL component named "beamformer". This component is incorporated into the larger xRTB design using a Simulink black box module which is surrounded by the requisite control logic. The magnitude of the final complex output of the B-Engine is squared and added in an accumulator. This accumulator periodically writes to a shared BRAM that can be read from the host PC to retrieve the accumulated beamformed power spectral data.

The memory control module is the most complex part of the B-Engine. This module controls the writing, storing, and reading of the beamformer coefficients. Its basic operation is to cycle through each coefficient and send it to the complex multiplier. This requires a state

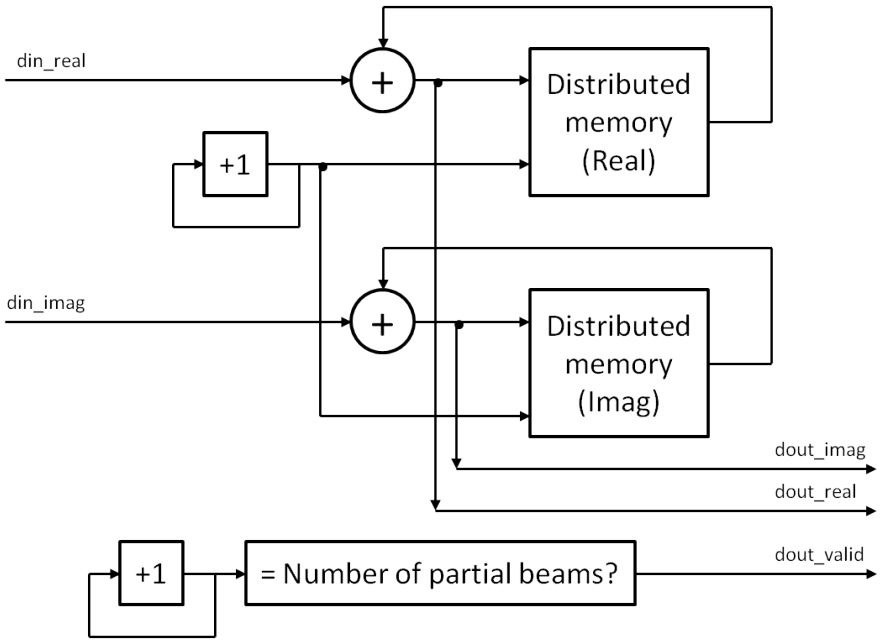


Figure 3.5: Block diagram of the B-Engine partial beam accumulator.

machine to control coefficient BRAM read address and when the values stored in this BRAM are sent to the multiplier. In addition to simply reading stored data, the memory control module can update the coefficients stored in its BRAM. The coefficients to be written to this BRAM are stored in a shared BRAM in the top-level Simulink xRTB model. Reading from this BRAM and writing to the memory control module BRAM is controlled by the memory control module state machine. The memory control module state machine is shown in the ASM chart in Figure 3.4.

The accumulation module completes the summation started by the initial adder. Because data from all antenna elements is not available to the B-Engine simultaneously, the multiplier and adder are not able to calculate a beamformed output for all antenna elements at once. Instead, they compute "partial beams" which are accumulated until the full beam has been formed. When a full beam has been formed, the data is marked as valid and sent out of the B-Engine to be accumulated in the top-level power accumulator. A block diagram of the partial beam accumulator is shown in Figure 3.5.

3.1.3 xRTB Minor subsystems:

The simplest minor subsystem is the x64 ADC. Because it is a CASPER library yellow block, it only needs to be dropped into the simulink model and connected to the F-engine.

The control system is somewhat more complex. This system consists of a few 32-bit shared registers that can be written to and read from by the host PC. These registers are used to save parameters such as the accumulation length, the dynamic range selection window, and input data for beamformer testing. One of these registers (named 'ctrl') is the main control register that generates the synchronization pulse that keeps each of the other subsystems running in lock-step with each other. This register also controls global and ADC reset signals as well as the signals needed to update beamformer coefficients for each of the seven beams.

The amount of data coming out the F-Engine is too much for the B-Engine to handle, so it must be reduced. To reduce the amount of FPGA fabric required by the B-Engine, the 36-bit complex output (18 bits real, 18 bits imaginary) of the F-Engine is reduced to an 8-bit (4 real, 4 imaginary) complex value. This reduction is done by rounding and slicing each of the real and imaginary lines to 4 bits. To avoid introducing a bias into this rounded output, half of the LSB is added before truncation [64]. This is done by slicing off the desired 4-bit window plus the next least-significant bit and adding one. Then the LSB of this intermediate value (the extra 5th bit) is dropped and the remaining 4-bit value is sent to the B-Engine. While this slicing reduces the dynamic range of the beamformer, the FPGA real-estate that is saved is critical for this design. The full 7-beam xRTB firmware occupies more than 95% of the available Virtex-5 resources. Without this data reduction, even a single-beam beamformer would not fit on the FPGA.

The most complicated subsystem is the mechanism to update the beamformer coefficients. To update the weights for one of the beams, the desired new coefficients are written to a shared BRAM on the FPGA. This BRAM is then copied to the B-Engine's coefficient BRAM. When the control signal is given for one of the beamformers to update its coefficients, it transitions into its coefficient update state and begins a counter that cycles through each of the beamformer coefficient addresses within its internal BRAM. Each address is sent to the shared BRAM, and the value stored at that address is written to the beamformer's

internal BRAM at that same address. This process forms a feedback loop which needs to be treated carefully if timing adjustments are made (such as adding or removing pipeline registers).

3.2 Real-Time Interference Cancellation System Design

The x64 Real-Time Interference Cancellation System (RTIC) is built on the xRTB and the 32-input real time correlator (xRTC), designed for use at the BEST-2 array in Medicina, Italy [65]. This system uses three ROACH boards, one for the xRTB and two for the real time correlator. In addition to this hardware, the host PC is needed to capture the output correlations, calculate new beamformer weights and update the beamformer weights on the xRTB. This extra work is done through a combination of MATLAB and Python scripts. With this system, the xRTB can accumulate for as little as 0.1 seconds, with a delay of 1 second between coefficient updates.

3.2.1 F/X Engine:

The real time correlator is built on the same FPGA platform as the xRTB. Two ROACH boards are required, along with an x64 ADC. The first ROACH houses an F-Engine similar to the xRTB (implementing a 2048-point FFT) and the second ROACH houses the correlator (X-Engine) itself. Channelized data is sent from the first ROACH to be correlated on the second ROACH through a 10GbE XAUI link. Correlated data is sent back to the host PC through a 1GbE link to be saved on disk. For a very in-depth explanation of the design and operation of this X-Engine, see [65].

3.2.2 Real-time Subspace Projection Software:

The F/X Engine control software handles saving the correlated data to the host PC. The RTIC control software handles the RFI cancellation from this point on. A combination of MATLAB and Python scripts are used to read the correlated data and perform subspace projection, calculate updated beamformer weights and upload them to the xRTB, and read beamformed data and plot the output spectra.

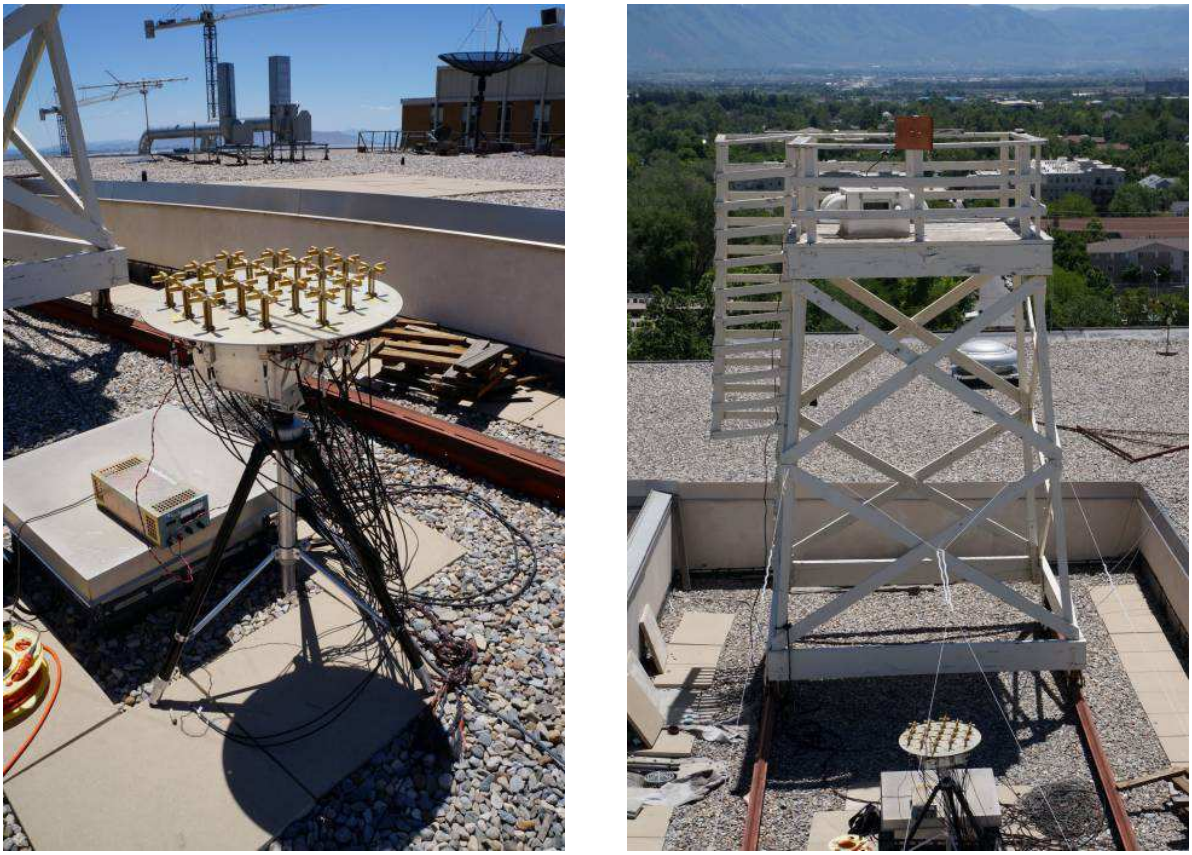
This software is still firmly in the "proof of concept" phase, and as such there is still some work to be done and a number of assumptions that have been made. The F/X Engine control software currently does not run automatically; new correlation matrices must be retrieved and saved to disk manually. There is no RFI detection performed, as it is assumed that there is always a single RFI source. Also, the RFI is always assumed to be stronger than the SOI; if this is not true, the subspace projection will try to cancel the SOI instead of the RFI.

3.3 Development and Testing of the xRTB

The development of the xRTB was primarily shared between two student. BYU student Megan Fuller designed, coded, and simulated the basic operation of the B-Engine including the complex multiply and add, reading beamformer weights from memory, and "partial beam" accumulation. This work was done as part of a senior team design project. When this author took over the development effort, the full xRTB design had been started but was in very early stages of development. In addition to designing the F-Engine, the B-Engine itself was expanded to allow the beamformer coefficients to be updated during operation and the summation and accumulation blocks were updated to fix overflow problems associated with bit-growth. The B-Engine was replicated to allow seven simultaneous beams to be formed and the requisite control logic for each beam was added. Because the FPGA resource utilization for this design is so high and timing closure was difficult to achieve due to routing, numerous minor changes (such as adding or removing pipeline registers) were made to the design before a bitstream could be produced.

Testing of the xRTB was done both in simulation and in the lab. Tests for each component of the beamformer were done in simulation (using a testbench written by Megan Fuller) before the beamformer was integrated into the full xRTB system. The xRTB was tested in hardware after the full design was completed and implemented on the ROACH boards. Using tools like Xilinx's ChipScope, the FPGA can be probed during operation to verify the integrity of the signals and processing. A number of table-top tests were performed to verify the operation of the xRTB using signal generators as a signal source. After the control software was complete, the xRTB system was testing using a 19 element dual-pol

PAF (the 'ear' dipole array described in [66]) mounted on the roof of the Clyde Engineering Building at BYU with another single transmitting antenna used as a signal source (see Figure 3.6). The xRTB was slated to be tested at the Arecibo Observatory during the experiment described in Chapter 4, but time constraints did not allow any substantial testing to be accomplished. Other testing platforms are being sought, including the GBT, but time and funding are currently limiting factors for a thorough real-data experiment using the xRTB.



(a) The 19 element PAF used to test the xRTB.

(b) The roof setup for the xRTB tests.

Figure 3.6: To test the xRTB, a 19 element PAF (a) was mounted on the roof of the Clyde Building. As a signal source, another antenna was mounted to the top of the large white tower. This can be seen in (b) as the small brown square on the railing of the tower. The transmitting antenna was moved to seven different locations to (mimicking the calibration procedure described in section 2.2.3) so that multiple beams could be formed.

Chapter 4

Experiment at the Arecibo Observatory

In July 2013, an experiment was conducted jointly by Cornell University, BYU, and scientists at the Arecibo observatory to test a 19-element dual-pol PAF prototype on the 305 meter Arecibo telescope; the worlds largest. The PAF was designed and built by Cornell University [26] and was coupled with the x64 Data Acquisition System (xDAQ) and xRTB designed at BYU. The experiment was conducted to characterize the capability of the Cornell PAF and as a first field test of the BYU digital back end.

4.1 Description of Experiment

During the experiment, a number of observations were conducted using the Cornell PAF and the xDAQ. These included calibration grids, an on-off weak-source observation, an extended source observation of the M87 radio galaxy and hot-cold observations using the moon. The calibration grids were used to calculate beamformer weights for many directions within the FOV, as described in section 2.2.3. The on-off weak-source observation consisted of two dish pointings, one looking directly at the weak source, and the other looking at blank sky, for 40 seconds each. With a good estimate of the noise, the weak source can be detected by subtracting the estimated noise from the on-source observation, as described in section 2.3. The M87 galaxy is too large to fit within a single beam of the PAF, so multiple dish pointing were used to be able to generate an image of the entire object. This extended source observation was done in much the same manner as the Cygnus X region observation described in [32]. A hot-cold observation provides a measurement of two sources of known temperature which can be used to determine the system noise temperature by utilizing the Y-factor method [67] [52]. The moon was used to provide the hot load and blank sky was used for the cold load.

The xDAQ recorded frequency channelized data to disk which was later processed to produce the final data products which included a sensitivity map of the PAF, far field beam patterns and a radio camera image of the elliptical galaxy M87 [26]. Unfortunately, due to time constraints, we were not able to test the xRTB in any meaningful way during this experiment.

4.1.1 Hardware Description

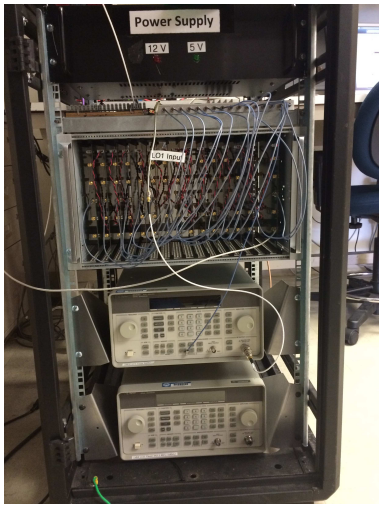
This experiment included three major hardware components: the Cornell PAF, the BYU analog receivers/downconverters and the BYU digital back-end. The Cornell PAF is described in detail in [26].

The BYU analog recerver/downconverter system (seen in Figure 4.1(a))was designed and built by graduate students Michael Elmer and Vikas Asthana [68] [34]. This system consists of a number of 4-channel receiver cards that prepare the PAF inputs for the digital back-end system. These cards take as input an L-Band signal, amplify and apply a bandpass filter to select out the 1200-1800 MHz band of interest. This signal is then lower-sideband down mixed, filtered, and amplified before being mixed again, amplified and bandpass filtered. The output signal lies within the 25-50 MHz IF band.

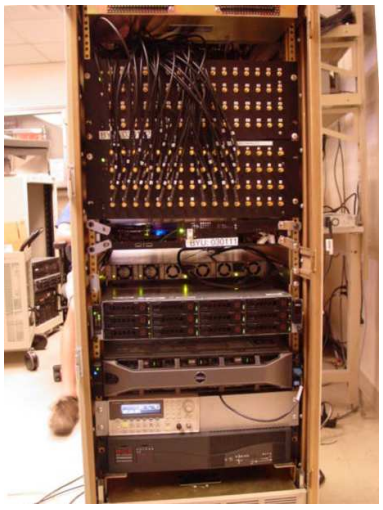
The BYU digital back-end was designed and built using CASPER ROACH boards with a 64-input 50 Msps ADC. As the IF band is from 25-50 MHz, the anti-aliasing filters on the ADC boards were removed to allow sampling within the second Nyquist zone. The BYU digital back-end includes both the xDAQ and the xRTB, although only one is generally used at a time. A brief description of the xDAQ will be provided here, however the reader is advised to read a detailed description in [41]. The xRTB is described in detail in Chapter 3 and will not be discussed further here.

x64 Data Acquisition System

The goal of the xDAQ is to sample the incoming IF band from the PAF, frequency channelize that data and then record it to a disk for post processing. The xDAQ is based on the same 64-input, 50 Msps ADC as the xRTB. Once the IF signal is sampled and digitized onto the ROACH board, it is passed into an F-Engine similar to that used in the xRTB. The



(a) BYU analog rack



(b) BYU digital rack

(c) Cornell PAF

Figure 4.1: (a) The analog rack contains 16 4-channel receiver cards, local oscillators (LOs) and the LO distribution network. (b) The digital rack contains three ROACH boards and two ROACH servers. Two of the ROACH boards are hidden behind the 64-channel ADC input panels (seen as four rows of 16 SMA RF connectors). (c) The Cornell cryo-PAF installed in the Arecibo feed dome. Both the LNAs and the antenna elements are contained inside the cryogenic dewar, and are not visible behind the orange vacuum window.

xDAQ F-Engine evolved from the xRTB F-Engine and includes the additional capability of allowing the user to choose between three different FFT lengths: 256, 512 and 1024. This frequency channelized data is sent to a data reduction subsystem that selects subsets of the full frequency spectrum and/or subsets of the 64 inputs. Next, the data is send to a UDP packetizer and 10GbE transceiver that transfers data to the ROACH control PC, through a 20 port 10 GbE network switch, to be written to disk. The data reduction frequency and input sub-selection is used to reduce the output data load so that 1) the 10 GbE over

which this data travels to the computer is not overloaded and 2) the RAID-0 configured hard drives can write all of the data without missing any packets from the ROACH. Once the UDP packets travel to the computer, a packet sniffing software ("Gulp") is used to read the data in each packet and place it into a large buffer (192 GB) in RAM and eventually write that data onto disk. Post-processing of the data was performed by a MATLAB correlator and post-correlation beamformer. An improved GPU based correlator was designed by this author and is described in Appendix B.

4.2 Experiment Results

After the experiment was completed, the next months were spent post-processing the data captured while in Arecibo. This post-processing included generating beamformer weights from the calibration grids and using these beamformer weights to generate the final data products.

4.2.1 Data Verification

While processing the data for this experiment, it became clear that there were some problems with the experiment. One of the problems caused some antenna power levels recorded by the xDAQ to be much higher than was thought possible. While the cause of this problem is still yet to be discovered, this "power bloom" resulted in bit overflow within the xDAQ.

During paketization, the xDAQ reduces the dynamic range of the recorded data by selecting an 8-bit window from the 18-bit F-Engine output. This data window is chosen by the user and is prone to user error causing overflow within this smaller window. There is no overflow detection logic within the xDAQ, so the recorded data needs to be checked to ensure there was no overflow during an observation.

Signals received from astronomical sources follow a Gaussian probability density function (pdf). We proposed that the Gaussian pdf of the recorded should 'fit' within the 8-bit window selected to ensure that the xDAQ did not overflow during packetization. For our case, we say a Gaussian pdf fits within the 8-bit window if the standard deviation σ is such that 4σ is less than the maximum value that can be represented in the 8-bit window (given

by $2^{n_{bits}-1}-1$ where n_{bits} is the number of bits used to represent values). The overflow test for the 8-bit window used by the xDAQ is

$$\begin{aligned} 4\sigma &< 2^{8-1}-1 \\ \sigma &< \frac{127}{4}. \end{aligned} \tag{4.1}$$

For such a pdf, there is the possibility that some of the data will overflow, but this small amount of overflow ($< .1\%$) was deemed tolerable.

Checking for this overflow is fairly easy if done visually, as shown in Figure 4.2. If the underlying Gaussian distribution of the recorded data is such that there is significant overflow, the pdf of the recorded data no longer looks Gaussian. As values overflow, the tails of the Gaussian pdf get folded back into the 8-bit window, which results in an increase in the standard deviation. Initially, overflow was detected by visually checking the histogram of recorded data. Because we recorded so much data, it was not feasible to verify all of the data visually.

To verify all of the recorded data, this author wrote a program that investigated the distribution of the complex data recorded during observation. To determine if a data set suffered from overflow, the standard deviation of that set was estimated. If equation 4.1 did not hold for the estimated standard deviation, the data were declared to have been corrupted by overflow. These data sets were flagged as corrupted by the data verification program and were unused in any analysis of the experiment.

4.2.2 PAF Sensitivity

The sensitivity of a PAF can be calculated by observing a source with flux density F^s in units of Janksy (Jy) by

$$\frac{A_e}{T_{\text{sys}}} = \frac{2k_b}{10^{-26}F^s}SNR = \frac{2k_b}{10^{-26}F^s}\frac{\mathbf{w}^h\mathbf{R_s}\mathbf{w}}{\mathbf{w}^h\mathbf{R_n}\mathbf{w}}, \tag{4.2}$$

where A_e is the effective dish area using beamformer weights $\mathbf{w}$, T_{sys} is the beam equivalent noise temperature, and k_b is Boltzmann's constant [32]. In order to characterize the sensi-

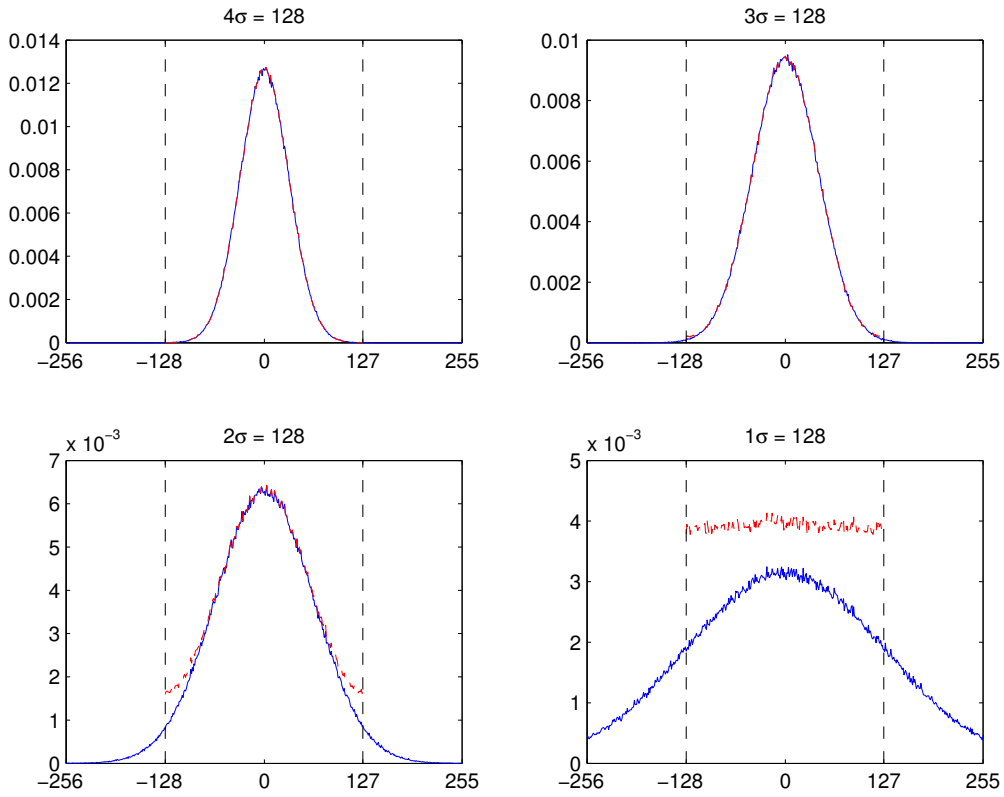


Figure 4.2: This figure shows the effect of bit overflow on Gaussian distributed data. The blue line shows the normalized histogram of Gaussian data that was generated with a standard deviation σ. This data was then subject to overflow by reducing each value to only 8 bits and the histogram of the result is shown in red. As fewer standard deviations fit within the 8-bit range of [-128,127) shown by the vertical black lines, there is more overflow. This overflow tends to raise the tails of the Gaussian so that the resulting distribution looks less Gaussian and more uniform.

tivity of the PAF, accurate estimates of $\mathbf{R}_{s,i}$ and $\mathbf{R}_n$ are required. These were generated after correlating the data from each of the uncorrupted calibration grids. Of the eight total uncorrupted calibration grids, there were six "fine" grids, which consisted of a 15 by 15 square grid of 221 pointings separated by 48 arcseconds in both azimuth and elevation, and two "superfine" grids, which consisted of a 31 by 31 square grid of 961 pointings separated by 25 arcseconds in both directions. One of the fine sensitivity maps is shown in Figure 4.3. Each pixel of the sensitivity map was calculated by using the beamformer weights for each pointing in the calibration grid in equation 4.2. These sensitivity maps show that there are a number of poor sensitivity elements. These poor sensitivity elements were primarily due LNAs that failed as the cryogenic dewer was depressurized. Ignoring these failed elements,

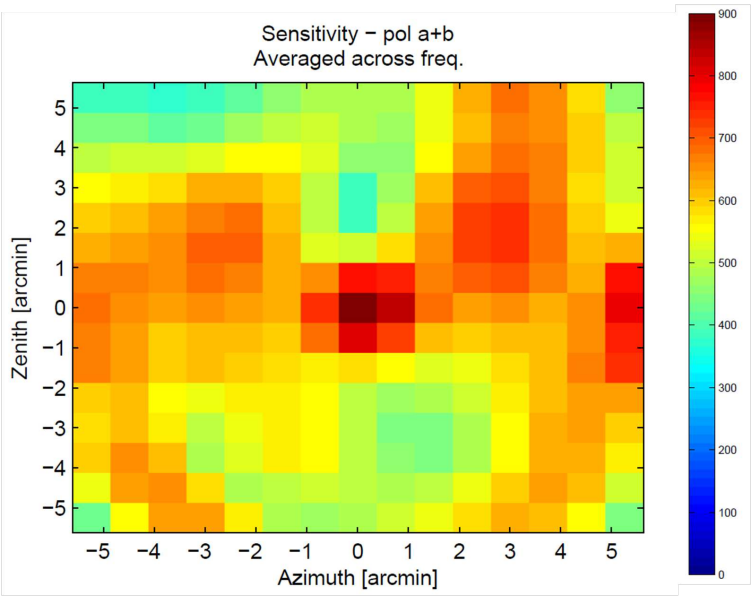


Figure 4.3: A 15x15 sensitivity grid generated using the 10.34 Jy source J2123+250. Ideally, the sensitivity should be constant across the FOV, but this image shows several regions (in green) with reduced sensitivity. This reduced sensitivity is due to LNAs that failed during the experiment.

the PAF has a sensitivity of 740 m^2/K, which corresponds to a T_{sys} of 53.6 K [41]. This sensitivity is between that of the ALFA cluster 7-horn feed used at Arecibo and the ASKAP 188 elemant PAF which have sensitivities of 30 K and 63 K respectivly [69] [20]. These results are encouraging and show that building large format cryo-PAF camera instruments for large radio telescopes is feasible [26].

4.2.3 M87 Mosaic Image

During the Arecibo experiment, a 5 by 5 grid around the galaxy known as M87 was acquired using the xDAQ. The spacing between each pointing was 7.5 arcmin. In order to form an image of the galaxy, the grid was subjected to a post-processing beamformer designed using one of the superfine calibration grid with a spacing of about 28 arcseconds. Applying these beamformers produced a 31 by 31 pixel radio camera image surrounding each of the 25 pointings within the M87 grid. As the spacing between the 5 by 5 grid pointings was less than the size of the 31 by 31 pixel image around each pointing, there was some overlap between adjacent pointings. For any pixels that overlapped others, the average pixel

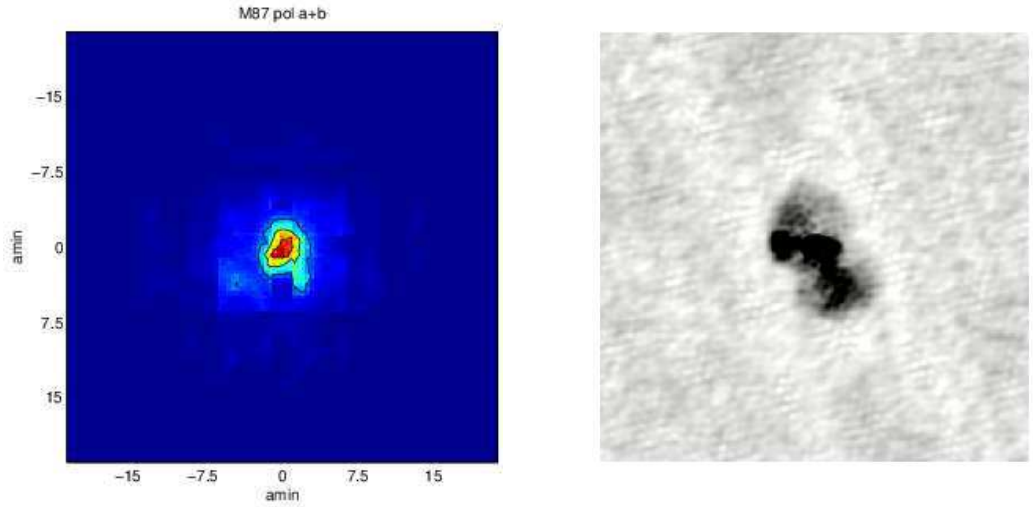
intensity was used in the final mosaic image. Some of the overlapping areas can be seen in the resulting image as four squares surrounding the central peak. The pixel intensities were then normalized across the full mosaic so that the maximum intensity pixel corresponds to the known flux density of the galaxy, 10.5 Jy [70].

The resulting radio camera mosaic image is shown in Figure 4.4(a). If we compare this image with an image produced using the VLA, we can see that the two images share some similarities. This is encouraging and shows that radio camera imaging using the Cornell PAF and BYU back-end can produce a reasonable image. The known dimensions of M87 are 8.3 by 6.6 arcmin. Using this mosaic image to calculate the major and minor diameters of M87 gives 7.5 arcmin by 6.5 arcmin respectively. The discrepancy between this estimate and the true dimensions of M87 is probably due to the relatively large beamwidth of the Conell PAF (about 3 arcmin) which results is poor spatial resolution. The failed elements of the PAF also had an effect on the mosaic image. This can be seen most clearly just below the central peak. There is a small cutout of the galaxy that corresponds with the location of one of the failed elements. The overlapping of pixels was able to reduce the effect of some of the failed elements since the overlapped pixels intensity was calculated using beams elements in another region of the PAF. The area directly below the central peak was not overlapped, so the pixel intensities there are based off of beams formed primarily using a failed element.

4.3 Conclusions

The Arecibo experiment gave both Cornell and BYU an opportunity to test their systems and evaluate the performance of these systems outside of the lab. While there were some problems with the Cornell PAF, it appears that the BYU receiver/downconverters and the xDAQ performed well. This is an encouraging outcome in that it showed that a scientifically viable PAF digital back-end system can be implemented on a ROACH-based hardware platform.

These preliminary results of the Cornell PAF have led to further collaboration between BYU and Cornell. We are currently engaged in a design effort for an 80 element dual-pol PAF (called AO40) for Arecibo. This new PAF will be accompanied by a 40 beam real-time beamformer.



(a) Mosaic image of M87 (b) Reference image of M87

Figure 4.4: The mosaic image of M87 generated from data gather during the Arecibo experiment is shown in (a). A comparison image of M87 at L-Band is shown in (b). Image courtesy of NASA/IPAC Extra-galactic Database.

Chapter 5

Study of Weak-Source Detection in the Presence of RFI

5.1 Introduction

Detecting a signal in radio astronomy is not always an easy thing to do. In terms of mean power, the signals we observe can be well below the noise floor of the telescope's electronics or the noise and interference filled environment surrounding the signal of interest. This results in a "Princess and the Pea" like scenario: we want to detect a signal that is buried underneath noise and interference which we often avoid. An approach to mitigating the RFI problem was discussed in Section 2.3.1.

This chapter explores what happens to the detection process in the presence of RFI, both with and without active spatial filtering mitigation methods. We will first discuss the two major motivations for this study. A description of the experiment will we laid out along with experimental results and analysis. Finally, we will draw some conclusions that will be beneficial to radio astronomers.

5.2 Motivation

Suppose one wishes to observe a astronomical source using a PAF radio telescope. We orient the dish so that the SOI is in the boresight direction for one of its beams (i.e. for the central beam, the dish points directly at the source). The telescope dish will focus the energy received from this source onto the PAF, however, each element will not receive the same amount of energy from the SOI. The energy will be distributed across the array in what is known as an Airy pattern due to diffraction. A maximum-SNR beamformr will weight the elements near the center of this pattern more strongly, while diminishing the contribution from the outer elements that do not receive the SOI as strongly. In contrast, the energy from RFI sources further away from boresight will not be focused and will spread

out across the array. If we apply a projection matrix to cancel this RFI, the weights for elements further away from the center of the SOI Airy pattern may be adjusted to have higher amplitude weights, which could increase their contribution to the total noise level seen in the beamformer output. Noise from elements which were previously de-weighted in the beamformer is now added in with higher weighting. This would lead to a reduction in SNR, which is undesirable. In fact, any projection matrix applied to maximum-SNR beamformer weights *must* result in a reduction of SNR (neglecting RFI and considering thermal noise only) since there is only one solution to the quadratic maximization of equation (2.9).

For weak source observations, it is important to be able to characterize the system noise temperature T_{sys} of radio telescope and receivers (see section 2.3.1). As shown in equation (2.18), the system temperature depends on the beamformer weights. If we apply a subspace projection matrix to the weights in order to reduce RFI, the system temperature will also be affected. Many RFI sources are not stationary, which means the projection matrix used to cancel that RFI must change with time to be effective. As the projection changes, the effect that that projection will have on the system temperature can also change. This may leads to a time-varying T_{sys} which can reduce the overall sensitivity of our telescope as discussed in section 2.3.1.

5.3 Description of the Experiment

This experiment was performed by first creating a PAF radio astronomy observation simulation. This simulation includes tracking a weak SOI through sidereal motion while a single strong moving RFI source passes through the beam's sidelobe pattern. The RFI sources used throughout the simulations were all members of the GPS constellation of satellites. Real-time beamforming and post-correlation beamforming along with oblique projection are used to form an RFI mitigating beam with main lobe focused on the SOI and spatial nulling on the satellite RFI. Different long-term integration (LTI) lengths were used, varying from 0.5 seconds up to 7200 seconds. Spectrograms (using Welch's method [71]) of the processed beamformer output data are visually inspected to determine what integration length allows the weak SOI to be detected. Other detection statistics were used to infer the behavior of RFI mitigation techniques in other situations.

5.3.1 Green Bank 20-Meter Telescope

The Green Bank 20-meter telescope was built in 1994 and was a part of geodetic VLBI experiments until its retirement in 2000. Since then, it has been used, among other things, for PAF and RFI mitigation testing by BYU [27] [72]. This experiment simulated radio astronomy observations using this telescope. Specifications for this telescope are given in Table 5.1.

Table 5.1: Green Bank 20-meter telescope specifications

Reflector Diameter	20 meters
F/D ratio	0.43
Slew Speed	2 degrees per second
Surface Accuracy	0.8 mm rms
Location	38 26'12.661" N 79 49'31.865" W

5.3.2 Simulation Model

In order to simulate a true-to-life observation, several factors were included in this simulation, such as the motion and flux density of both the SOI and RFI, the gain of the reflector and feed, and the details of the receiver system.

Signal and Interference

In order to track SOI and RFI sources, their on-sky locations must be known precisely as a function of time. This information was gathered from a software package (XEphem) that provides ephemeris for various stellar objects and artificial satellites [73]. XEphem uses two-line element (TLE) sets, which provide position and velocity of object orbiting the Earth, to determine the location of objects at any given time. The simulations presented here cover the two-hour window from Julian Date (JD) 2457134.88094 to 2457134.96428 (Coordinated Universal Time (UTC) 2015-04-22 9:09:33 to 2015-04-22 11:09:33).

The signal of interest for this study was modeled after the radio galaxy Cygnus A. The location of the SOI coincides with the star Sadr (Bayer designation γ Cyg) and the flux density of the SOI was arbitrarily set so that the SNR would be about -27 dB at the array. There were six RFI sources, although each observation only included one RFI source at a time. These satellites ate within the GPS constellation and include PRN 05, 13, 15, 21, 24, and 26. This simulation focused on a 10 kHz bandwidth centered at 1.57542 GHz (the center of the L1 civilian band for GPS). These satellites use a direct Pseudo Noise (PN) coded sequence spread spectrum modulated signal that appears temporarily white across the single beamformer channel bandwidth studied. The sidereal revolution period of the GPS satellites is roughly half a sidereal day, meaning that these satellites move about twice as far on the sky as the SOI during an observation. Figure 5.1 shows a screenshot of XEphem with the tracked GPS satellites and Sadr. Each tracked source position was updated every 0.1 seconds over the full 7200 second (2 hour) observation.

For this simulation, the interference is modeled as broadband white Gaussian noise, which is consistent with the modulation parameters and the narrowband beamformer architecture.. The power received from the interference source is modeled as a time-varying random process with instantaneous power level set by low-pass filtering white Gaussian noise. The signal of interest is modeled as a narrow-band Gaussian random process source, with a 3 dB pass-band of 2 kHz to 3 kHz. By focusing on only a narrow-band signal of interest, we can perform on-off subtraction with only a single dish pointing (as discussed in section 2.3). The signal-to-noise (SNR) ratio for all observations was set to be -27 dB. The interference-to-noise (INR) ratio varied from 0 dB to 30 dB.

Reflector and feed model

The telescope reflector is modeled as a 20-meter parabolic dish. The feed used is a 19-element single-polarization dipole PAF. A PAF model created by Karl Warnick generates the complex steering vectors ($\mathbf{v}_s, \mathbf{v}_i$ from equation (2.11)) needed to corretly model the recieved signal. The model includes details of the PAF, reflector, and receiver such as element spacing, mutual coupling, polarization, bandwidth, reflector size, focal length, and spillover, sky and LNA noise. See Figure 5.2 for a typical maximum-SNR beampattern for this dish and feed.

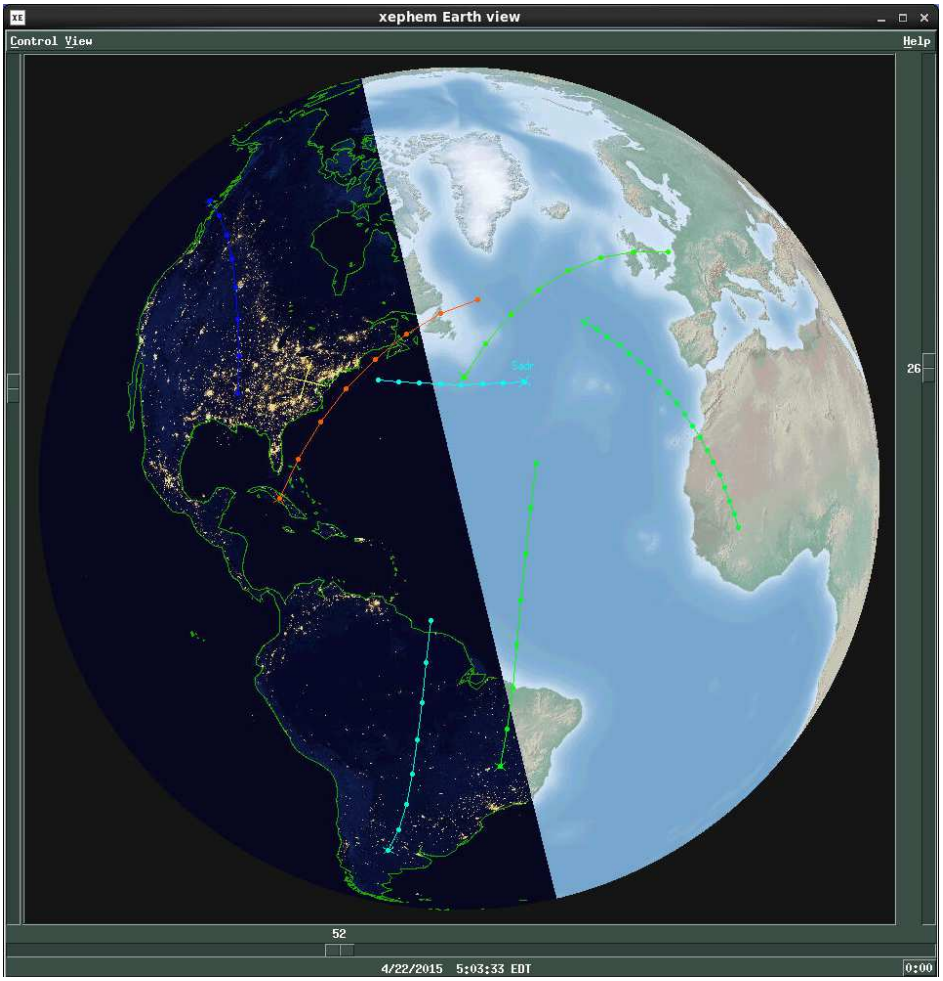


Figure 5.1: This screenshot of the XEphem program shows the tracked locations of the star Sadr (cyan, center) and several GPS satellites over a two-hour period starting at JD 2457134.88094. The yellow cross marks the locatino of the National Radio Astronomy Observatory in Green Bank, West Virginia.

Receiver system details

Using the array response vectors generated by the PAF model, a received signal vector $\mathbf{x}[n]$ can be produced. This complex vector is created by generating complex random voltages for the SOI and RFI, scaling those voltage by the appropriate array steering vector $\mathbf{v}_s$ or $\mathbf{v}_i$, and adding another complex random voltage that represents the noise seen by the array (including LNA, spillover, and sky noise). Since the received signals represent samples of a random process, 24 Monte Carlo trials were run and averaged for each simulated observation. The received signal is sampled at a rate of 10 kHz and integrated for 0.5 seconds. A 10 kHz

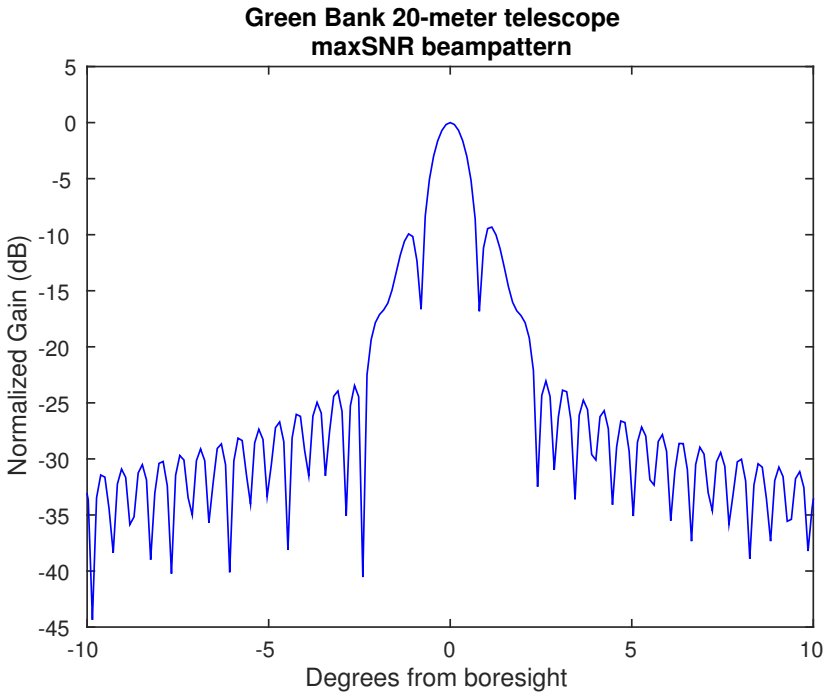


Figure 5.2: Using the Green Bank 20-meter telescope with a 19-element PAF, maximum-SNR beamforming produces this beampattern as a function of zenith angle

processing bandwidth is representative for a finely channelized narroband beamformer used in HI observations. This short-term integration (STI) length is the standard STI length of the Focal L-Band Array for the Green Bank Telescope (FLAG) correlator, currently being developed by NRAO/BYU/WVU. Recall that the SOI and RFI sources have 0.1 second position updates, which leads to some amount of "subspace smearing" as the sources move during each half-second STI. While the instantaneous interference subspace is rank-one, time averaging during motion produces a higher rank matrix, which requires more eigenvectors to span the interference subspace. This effect was dubbed subspace smearing in [38].

5.3.3 Signal Processing

During each observation, a number of signal processing steps take place. In order, these are:

- Generate the received signal $\mathbf{x}[n]$ for each 0.1 second position update (we will call these very short time integrations (VSTIs)).

- Perform an array covariance estimate for each VSTI and save both the time series $\mathbf{x}[n]$ and the correlation.
- Average VSTI correlations into a 0.5 second STI correlation and concatenate the VSTI time series into an STI time series.
- For each STI, apply an oblique projection to both the time series data and correlated data.
- Apply beamformer weights to the time series data and generate a power spectral density (PSD) estimate using Welch's method [71].
- Integrate the resulting averaged periodograms and the projected correlations for various LTI lengths.
- Apply a post-correlation beamformer to the integrated correlated data.
- Calculate figures of merit, including T_{sys}, $T_{\text{sys,rfi}}$, $T_{\text{sys,noise}}$ and the variance of the PSD estimate.

5.4 Experiment Results and Analysis

The results for this experiment will be discussed in three sections. First, the effect that subspace projection has on beamformer weights and T_{sys} will be considered. Next we will discuss the stability of the radio telescope while applying subspace projection to the received signals. Finally, detecting a narrowband signal masked by RFI will be discussed.

5.4.1 Effect of Subspace Projection on Beamformer Weights

Before the observation simulations were performed, we explored the question as to how much affect RFI mitigation would have on the beamformer weights and T_{sys}. Figure 5.3(a) compares beamformer weights for RFI from various arrival angles to the beamformer weights before subspace projection. Figure 5.3(b) shows the system temperature after subspace projection for RFI from various angles.

The effect that subspace projection has on the gross scale of beamformer weights seems to be fairly minimal. The thought that the outer elements could see a dramatic

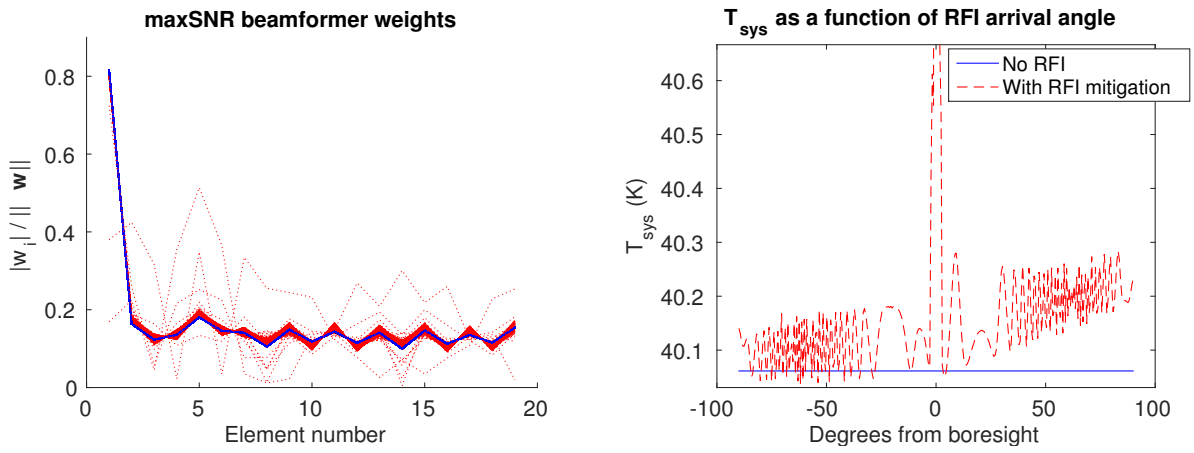


Figure 5.3: The effect of RFI mitigation on beamformer weights and system temperature for an interference source arriving from varying zenith angles is shown in these figures. In (a), the beamformer weights for many arrival angles are overlaid. This shows that there is some adjustment to the weights, with a dramatic change only happening for RFI sources within the FOV. In (b), the system temperature also shows some variation with arrival angle, but this increase is only substantial within the FOV.

increase in weight magnitude seems to be incorrect for most RFI arrival angles. The outer elements do see some adjustment to their weights, but that change is relatively small and spread across all of the PAF elements. As RFI typically is spread across the PAF, instead of being focused to a single point, this small effect is encouraging since there is not a major amplification of outer ring elements which could increase the noise substantially. However, as the RFI moves into the FOV, energy received by the central element is dominated by RFI. By placing a spatial null on the RFI, the weight for the central element is reduced. This reduction in the central element forces outer-ring element weights to be increased, seen as the widely varying red dashed lines of Figure 5.3(a). This leads to the noise amplification effect hypothesized, but only for RFI sources in the FOV.

Subspace projection also causes a change in the system temperature, and shown in Figure 5.3(b). For RFI arriving from directions away from boresight, this change is relatively small (this simulation shows a change of less than 1 K). However, as the interference gets into to the PAF FOV, the effect is dramatic (the central peak of Figure 5.3(b) extends upwards

to over 160 K). In addition to a very large increase in T_{sys}, any attempt to cancel an RFI source within the FOV could lead to cancellation of the SOI.

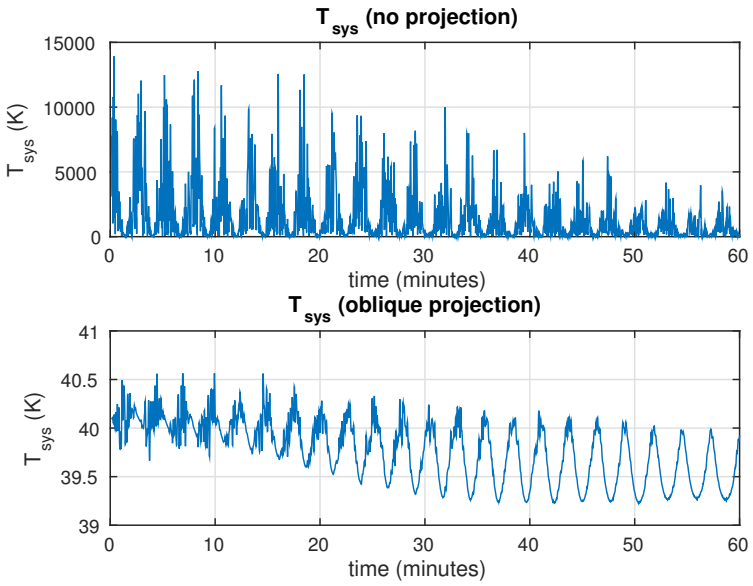
The performance metrics described in section 2.2.6 were calculated for each observation. The system temperature T_{sys}, including RFI contributions, before and after projection was estimated for each 0.5 second STI. This is shown in Figure 5.4(a). Figure 5.4(b) shows the contribution to T_{sys} from residual RFI and the perturbation of the noise field, $T_{\text{sys,rfi}}$ and $T_{\text{sys,noise}}$ respectively.

Figure 5.4(a) shows that without any attempt at RFI mitigation, the system temperature increased by 25-30 dB during this observation. In addition to this large increase, the system temperature varies dramatically with time. This time variation impedes our ability to reduce sample estimation error by integrating over longer periods of time. However, it may still be possible to get a detection without mitigation for some RFI power levels, as will be shown in section 5.4.3. If we apply subspace projection using an oblique projection during our observation, the resulting T_{sys} is much lower and is comparable to the no RFI T_{sys} of 40 K shown in Figure 5.3(b). However, there is some residual power level variation after subspace projection which will have an effect of the signals we can observe.

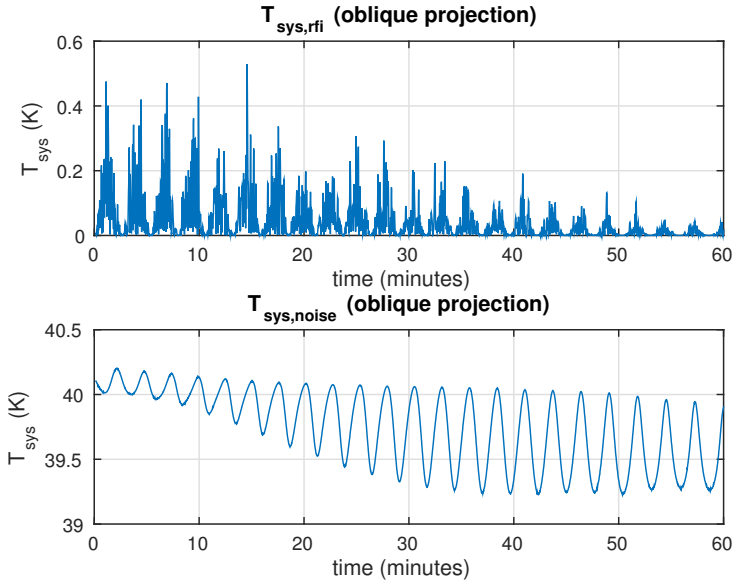
The increase in system temperature due to residual RFI was generally small (< 1K) for the simulated observations. This level was dependent on the physical geometry of the observation: observations in which the RFI source moved closer to the SOI resulted in higher levels of residual RFI when the two sources were closer together. This can be seen in Figure 5.5 which shows $T_{\text{sys,rfi}}$ for an observation that begins with the SOI and RFI separated by 2°. As the RFI moves away from the SOI, the residual RFI is reduced. However, $T_{\text{sys,noise}}$ does not seem to be as dependent on the relative positions of the SOI and RFI.

5.4.2 Radiometer Stability in the Presense of RFI

To determine the stability of the instrument over long observations during RFI canceling, the spectral noise power sample estimate of standard deviation was calculated for varying LTI lengths. Figure 5.6 shows this standard deviation as a function of integration time for observations without RFI, with RFI but without any RFI mitigation, and with RFI and an oblique subspace projection. Recall from 2.3.1 that regions of these curves with a



(a)



(b)

Figure 5.4: For an observation with GPS satellite PRN5 as an interference source, the effect of RFI mitigation is substantial. 5.4(a) shows the system temperature being reduced from levels exceeding 10000 K down to about 40 K. 5.4(b) shows the figures of merit $T_{\mathrm{sys,rfi}}$ and $T_{\mathrm{sys,noise}}$. The residual RFI is responsible for an increse in T_{sys} of less than 0.6 K while the noise floor varies by about 1 K.

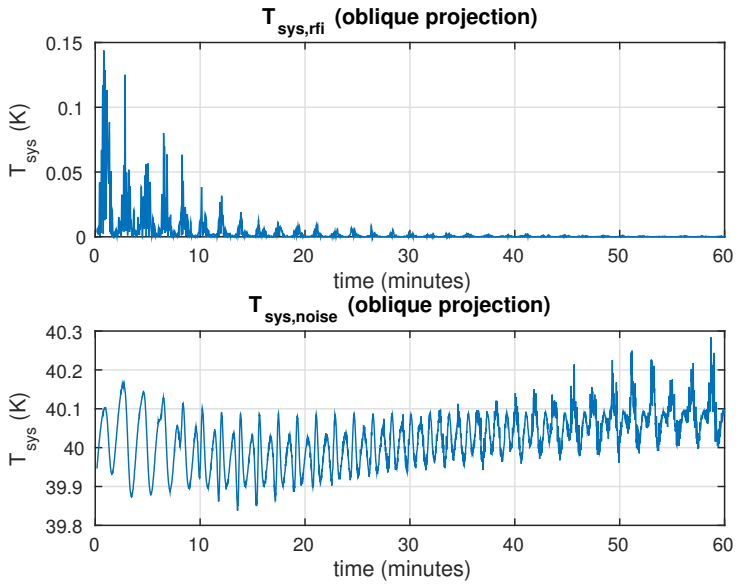


Figure 5.5: For the GPS satellite PRN13 as an interference source, the residual RFI is less than 0.15 K and decreases as the satellite travels further away from the SOI.

slope of -0.5 are dominated by Gaussian thermal noise [59] [60], whose variance can be reduced by integration (this corresponds to the first term in 2.24). As the slope of the standard deviation curve becomes less negative, our ability to reduce estimation error by integration is reduced as the second term of 2.24) begins to dominate.

While Figure 5.6 gives a comparison between all of the observation scenarios, it is difficult to visually determine the slope of the standard deviation from this figure. To aid in determining how the estimated standard deviation evolves with time, Figure 5.7 shows three of the standard deviation curves from Figure 5.6. These curves have had a piecewise linear curve fit applied to them to estimate the local slopes. Each curve was split by hand into segments, and a line was fit to each segment. The segments boundaries (shown by the green vertical lines) were chosen manually to "optimize" the linear fit within that segment, which means the segments are not always the same length or in the same location. The linear fit equation for each segment is given in the bottom left corner of each plot. Notice that the slope for most of these piecewise segments is less than zero, which suggest that we will continue to see improvement with integration for the no RFI and oblique projection cases. However, for the observation with no RFI mitigation, integration beyond about 2

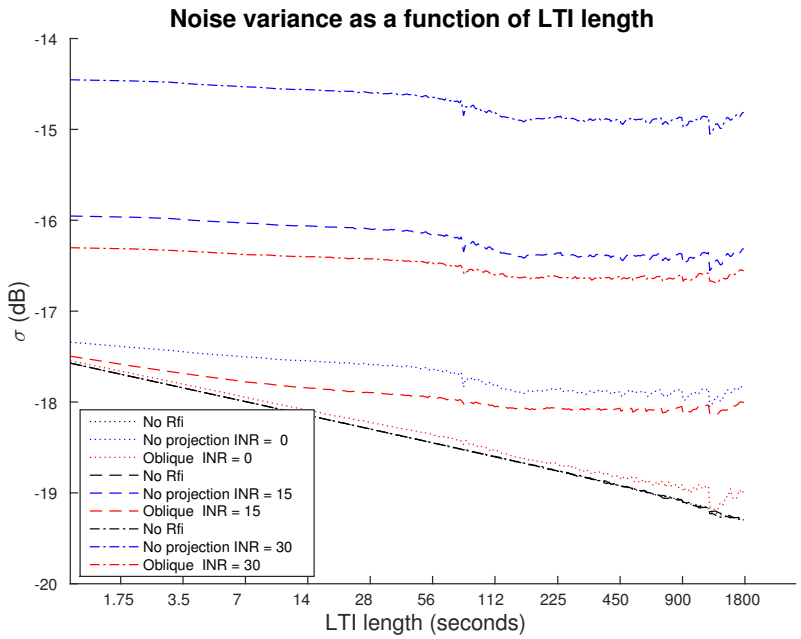


Figure 5.6: The sample estimated standard deviation of the noise power provides a measure of the radiometer's stability. For longer integration times with no RFI, this variation continues to decrease. If RFI is present, the slope is very shallow suggesting an increased difficulty in detecting weak sources. RFI mitigation brings the variation closer to the no RFI case which will reduce the integration time needed to detect a weak source.

minutes will not reduce the standard deviation as the slope is no longer less than zero. This figure also shows that there are two very clear inflection points in both the no mitigation and oblique projection curves. These coincide with the end of the first peak and the beginning of the second peak in T_{sys} shown in Figure 5.4(a). Once we begin to integrate the time samples with a lower system temperature between these two peaks, the standard deviation reduces more quickly, producing a more negative slope. When we begin to integrate the second peak, the standard deviation begins to level off and does not reduce as quickly.

5.4.3 Narrow-Band Weak Source Detection

Finally, to determine if we can detect the SOI, power spectral estimates are given in Figures 5.8 - 5.10. We declare a detection if the variance of the noise only frequency band is less than the power within the SOI frequency band. The dashed lines show the 5σ

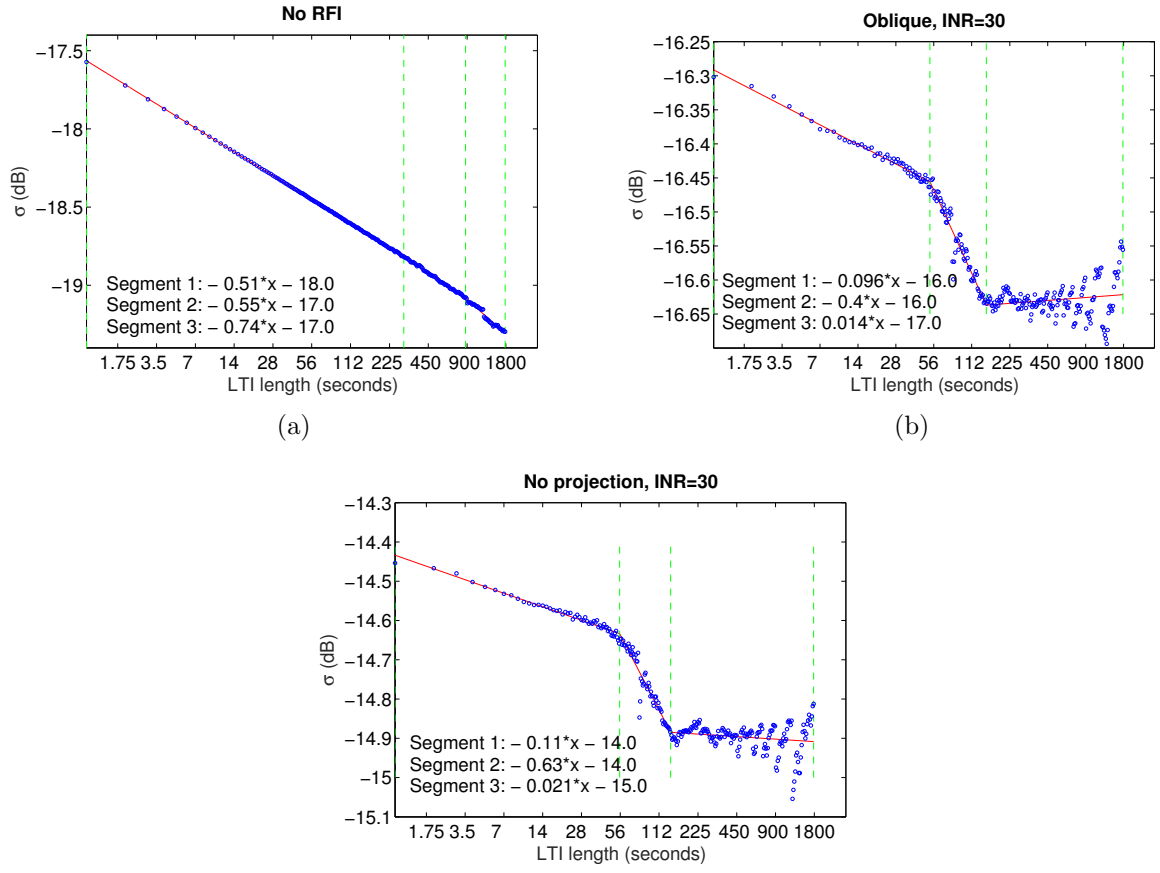


Figure 5.7: In (c), the standard deviation continues to decrese with a slope of -0.5 for the full observation as expected. In (b) and (a), the slope flattens near the end of the observation when gain variation begin to dominate the radiometer equation.

detection threshold. To declare a detection of the weak source, the signal power within the SOI passband of 2-3 kHz must be greater than this detection threshold.

For the observation periodograms shown in Figures 5.8 - 5.10, the no RFI cases (black lines) provide a detection in about 300 seconds. For the oblique projection cases (red lines), we get a detection after some extra integration (between 300 and 1800 seconds depending on interference strength). With no mitigation (blue lines), we do see a detection in the weaker RFI cases, but not in the strong RFI case.

In section 5.4.2, we saw that the standard deviation reduced most quickly for the case with no RFI, and most slowly for the case with RFI, but without any RFI mitigation. Figure 5.10 shows that a detection is made with the least integration for the case without any RFI.

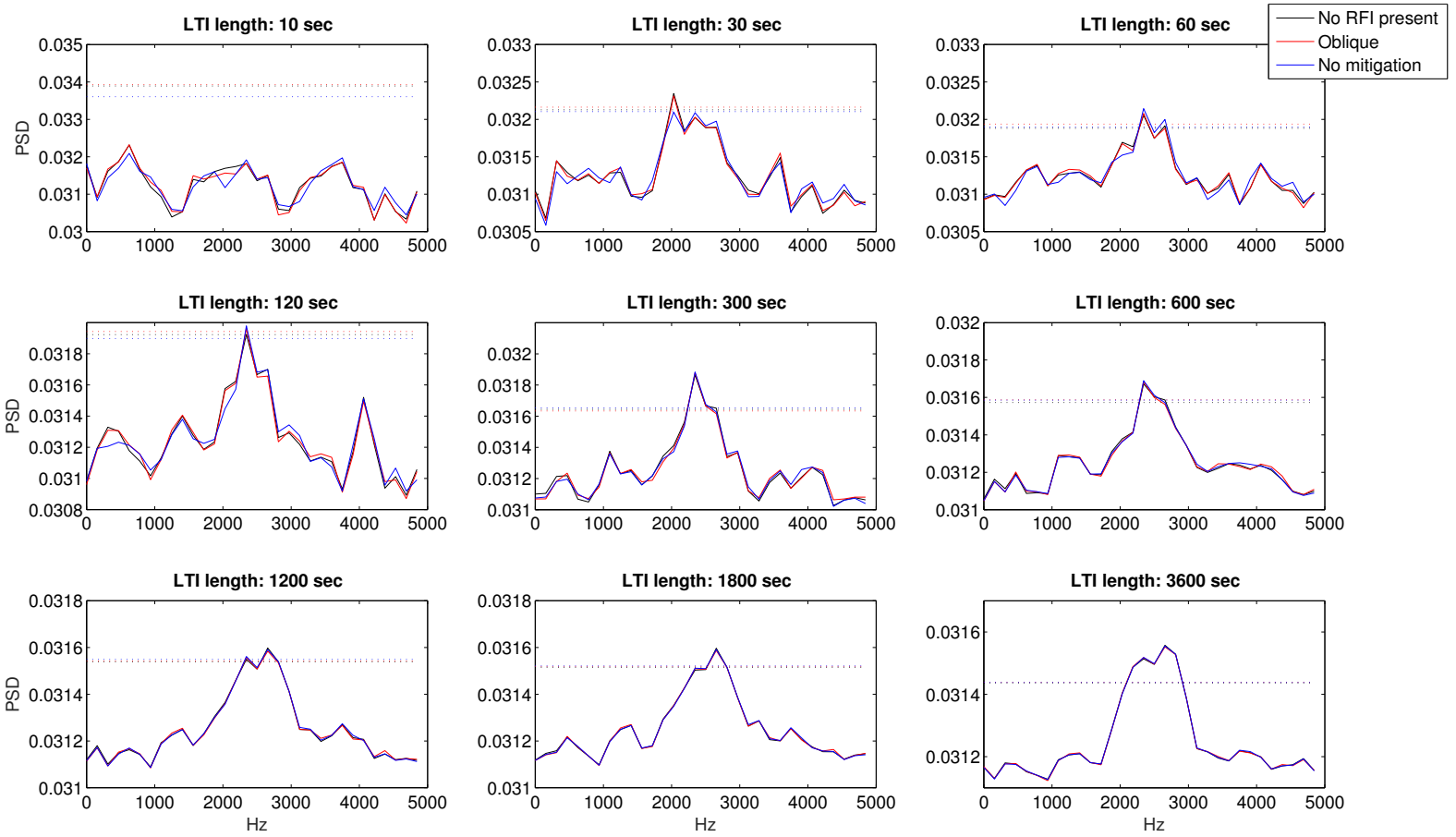


Figure 5.8: Averaged periodograms for varying LTI lengths and an INR of 0 dB. These show that a weak source detection can be made in the presence of a weak source of interference even with no RFI mitigation.

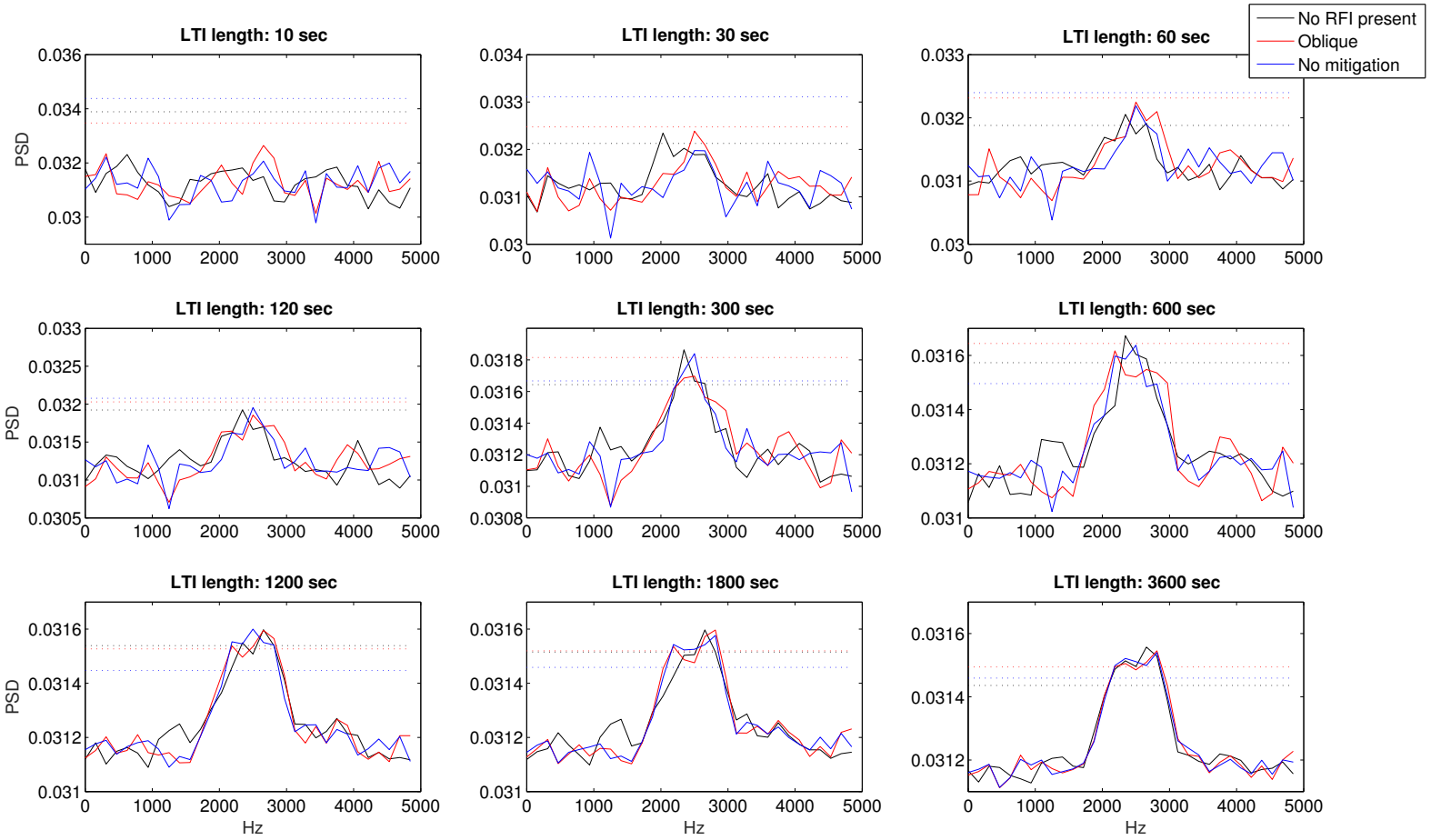


Figure 5.9: Averaged periodograms for varying LTI lengths and an INR of 15 dB. Like the case with an INR of 0 dB, a detection is made for all three observation scenarios, although it takes some extra integration to achieve a detection.

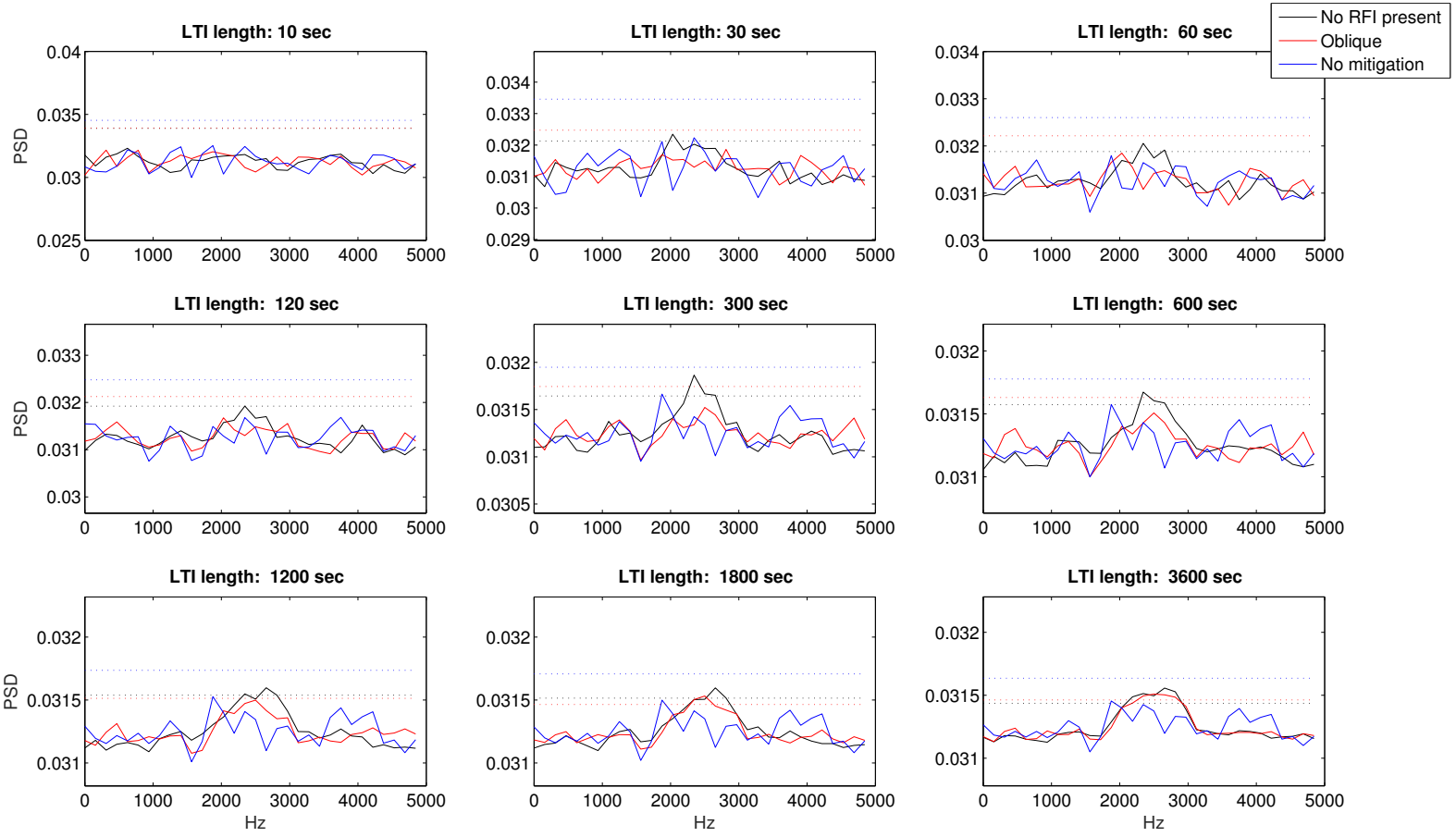


Figure 5.10: Averaged periodograms for varying LTI lengths and an INR of 30 dB. In this case, a detection is made with oblique proejction after 1800 seconds of integration. Without mitigation, there is no detection.

With RFI present, and using an oblique projection to cancel that RFI, a detection is made after a some amount of extra integration. Without using any RFI mitigation techniques, the estimated noise standard deviation does not reduce sufficiently to allow a detection for the strongest source of interference.

5.5 Recommendations

We will now provide some recommendations for astronomers attempting to make weak-source detections in the presence of RFI.

First, it is critical to know the strength of the interference. As discussed in [40], if the INR is small it is possible to inadvertently apply a projection which cancels the SOI. A test for detection of RFI of sufficient strength to provide a subspace estimate is given in [40].

Another concern involves the location of the RFI. As previously discussed, if the RFI is very near to the SOI, a subspace projection can place a spatial null on or near the SOI. To detect this situation, the angle between the two subspaces (that of the SOI and the RFI) can be evaluated. This angle is given by

$$\Theta = \cos^{-1}\left(\frac{\langle \mathbf{u}_i, \mathbf{w}\rangle}{||\mathbf{u}_i||||\mathbf{w}||}\right), \tag{5.1}$$

where $\mathbf{u}_i$ is the eigenvector of $\hat{\mathbf{R}}$ corresponding to the estimated interference subspace and $\mathbf{w}$ is the vector of beamformer coefficients. This can be calculated at the same time the oblique projection correction factor is being calculated. This angle should be close to 90°. For the simulations discussed in this chapter, this angle was between 75°and 90°. As this angle gets further away from 90°, the astronomer should consider excising the corresponding time series data, so that portion of the observation is ignored during which the signal is in danger of being projected out along with the RFI.

Secondly, the stability of the observation should be monitored in real-time. To do this, a 'strip-chart" that records the noise variance as a function of time should be used (see [74] for an algorithm to estimate a running variance without arithmetical error). This will allow the astronomer to understand the effect of integration on the observation. If the variance does not continue to decrease with integration time, a weak-source detection will be more

difficult. If this strip chart shows an increasing variance, it is very possible a weak-source detection will not be achievable. In this case, the astronomer should consider stopping the observation and attempting again when the RFI environment is more well behaved.

5.6 Conclusions

In order to test the effect of RFI mitigation using subspace projection on weak source observations, we designed a simulation that would mimic a weak source with RFI scenario using a 19-element PAF mounted on the Green Bank 20-meter telescope. There are three principal conclusions that can be drawn from these simulations. First, while subspace projection does effect the beamformer weights, this does not generally cause a dramatic increase in the noise floor. Second, the change in T_{sys} due to subspace projection is primarily due to residual RFI that is not fully canceled. Finally, T_{sys} becomes time varying because the subspace projection matrix varies with time. This leads to a reduction in long-term stability of the instrument. In some cases, this pattern rumble is not substantial enough to negate the benefits of integration. The results of these simulations have led to two recommendations for astronomers. First, the angle between the interference and signal subspaces should be watched during an observation. If this angle is small, subspace projection can place a null on the SOI. Using data gathered in this situation should be avoided. Second, the stability of the observation should be tracked as well. If the stability is decreasing substantially, the astronomer should know that a detection will become much more difficult, and may not be possible in the current RFI environment.

Chapter 6

Conclusions and Future Work

In this chapter we will summarize the results of the work presented in this thesis and discuss some areas that should be researched further.

6.1 Conclusions

Phased-array feed radio astronomy shows a great deal of promise, but comes with the price of increased signal processing complexity. A real-time digital beamformer is presented in this thesis that utilizes FPGA hardware to perform much of this required signal processing. This beamformer, and the hardware platform used, reduce both the time necessary to form multiple beams and the disk space necessary to house that data, when compared to other post-correlation beamforming systems (such as that used in the Arecibo experiment described in Chapter 4). Additionally, the ROACH based system provides a much smaller physical size when compared with previous BYU PAF back-ends.

A real-data experiment was conducted at the Arecibo Observatory. During this test, another ROACH-based back-end was used to record the data received by a PAF designed by Cornell. This data was later processed to first check its validity and to ensure that the data did not suffer from overflow during recording. Using a histogram-based detection scheme, it was found that nearly half of the data recorded during this experiment was unusable. Using the data that remained, a mosaic radio-camera image was formed for the M87 galaxy.

Radio-frequency interference mitigation techniques have been shown to be very effective, but some side-effects have not been explored. A simulated observation in the presence of RFI has shown that while there is some reduction in the radiometric stability using these techniques, the benefits greatly outweigh the reduced stability. Signals that would otherwise be masked completely by interference, even with substantial integration, are detectable

using oblique projection, albeit with some extra integration when compared with a similar observation with no RFI.

Subspace projection does add some time-variation in the system temperature, which can lead to some strange effects, such as a signal that can be detected with some LTI length can "disappear" and no longer be visible with a longer integration. Two techniques have been proposed that address this issue. First, the real-time tracking of the vector angle between the signal and interference subspace should be utilized. Second, the integration stability measure (ISM) should also be monitored. If the stability shows signs of significant deterioration, the observation should be stopped. While this does not ensure a detection, it may salvage a detection that would otherwise be lost.

6.2 Future Work

The reserach presented in this thesis can be extended in a few different directions. The real-time beamformer could be expanded to include more simultaneous beams. The software control system used by the beamformer is also in need to optimization, which could dramatically reduce the time needed to update beamformer weights. This optimization would allow the real-time interference cancellation system to work on much finer timescales than other similar systems currently in use. This is essential for RFI cancellation of a moving source. The Arecibo experiment showed that there is still some development needed for the x64 Data Acquisition System. Overflow is a major problem that should be addressed before the system is used for other observations. The weak-source observation simulation only covers a narrow range of the types of observations done by astronomers. Other situations that need to be studied include broadband signals and RFI that is not spectrally white. Other mitigation techniques also need to be studied, such as cross-subspace projection. Automated controls that decide when to use RFI mitigation, when to continue or stop an integration, and when to discard data should also be studied. Also, the results of the simulations presented in this thesis should be verified by real-data experiments.

Bibliography

[1] R. Hulse and J. Taylor, "Discovery of a pulsar in a binary system," *Neutron stars, black holes, and binary X-ray sources*, vol. 48, p. 433, 1975. 1

[2] D. Lorimer, M. Bailes, M. McLaughlin, D. Narkevic, and F. Crawford, "A bright millisecond radio burst of extragalactic origin," *Science*, vol. 318, no. 5851, pp. 777–780, 2007. 1

[3] A. Bosma, "21-cm line studies of spiral galaxies. ii. the distribution and kinematics of neutral hydrogen in spiral galaxies of various morphological types." *The Astronomical Journal*, vol. 86, pp. 1825–1846, 1981. 1

[4] R. Ekers and J. Bell, "Radio frequency interference," *arXiv preprint astro-ph/0002515*, 2000. 1

[5] K. Jansky, "Electrical disturbances apparently of extraterrestrial origin," *Radio Engineers, Proceedings of the Institute of*, vol. 21, no. 10, pp. 1387–1398, Oct 1933. 2

[6] W. Baade and R. Minkowski, "Identification of the Radio Sources in Cassiopeia, Cygnus A, and Puppis A." *Astrophysical Journal*, vol. 119, p. 206, Jan. 1954. 2

[7] P. Patel, D. J. Bacon, R. J. Beswick, T. W. B. Muxlow, and B. Hoyle, "Radio weak gravitational lensing with VLA and MERLIN," *Monthly Notices of the Royal Astronomical Society*, vol. 401, pp. 2572–2586, Feb. 2010. 2

[8] G. G. Pavlov, V. E. Zavlin, B. Aschenbach, J. Trümper, and D. Sanwal, "The Compact Central Object in Cassiopeia A: A Neutron Star with Hot Polar Caps or a Black Hole?" *The Astrophysical Journal*, vol. 531, pp. L53–L56, Mar. 2000. 3

[9] R. Hall and L. J. King, "The green bank telescope," in *Antennas and Propagation Society International Symposium, 1992. AP-S. 1992 Digest. Held in Conjuction with: URSI Radio Science Meeting and Nuclear EMP Meeting., IEEE*, June 1992, pp. 862–865 vol.2. 2

[10] L. M. LaLonde, "The upgraded arecibo observatory," *Science*, vol. 186, no. 4160, pp. pp. 213–218, 1974. [Online]. Available: http://www.jstor.org/stable/1739624 2

[11] O. Hachenberg, B. Grahl, and R. Wielebinski, "The 100-meter radio telescope at effelsberg," *Proceedings of the IEEE*, vol. 61, no. 9, pp. 1288–1295, Sept 1973. 2

[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] A. Wootten and A. Thompson, "The atacama large millimeter/submillimeter array," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1463–1471, Aug 2009. 2

[14] M. de Vos, A. Gunst, and R. Nijboer, "The lofar telescope: System architecture and signal processing," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1431–1437, Aug 2009. 2

[15] C. Lonsdale, R. Cappallo, M. Morales, F. Briggs, L. Benkevitch, J. Bowman, J. Bunton, S. Burns, B. Corey, L. deSouza, S. Doeleman, M. Derome, A. Deshpande, M. Gopala, L. Greenhill, D. Herne, J. Hewitt, P. Kamini, J. Kasper, B. Kincaid, J. Kocz, E. Kowald, E. Kratzenberg, D. Kumar, M. Lynch, S. Madhavi, M. Matejek, D. Mitchell, E. Morgan, D. Oberoi, S. Ord, J. Pathikulangara, T. Prabu, A. Rogers, A. Roshi, J. Salah, R. Sault, N. Shankar, K. Srivani, J. Stevens, S. Tingay, A. Vaccarella, M. Waterson, R. Wayth, R. Webster, A. Whitney, A. Williams, and C. Williams, "The murchison widefield array: Design overview," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1497–1506, Aug 2009. 2

[16] E. Petroff, E. F. Keane, E. D. Barr, J. E. Reynolds, J. Sarkissian, P. G. Edwards, J. Stevens, C. Brem, A. Jameson, S. Burke-Spolaor, S. Johnston, N. D. R. Bhat, P. Chandra, S. Kudale, and S. Bhandari, "Identifying the source of perytons at the Parkes radio telescope," *ArXiv e-prints*, Apr. 2015. 4

[17] D. Roshi, M. Bloss, P. Brandt, S. Bussa, H. Chen, P. Demorest, G. Desvignes, T. Filiba, R. Fisher, J. Ford, D. Frayer, R. Garwood, S. Gowda, G. Jones, B. Mallard, J. Masters, R. McCullough, G. Molera, K. O'Neil, J. Ray, S. Scott, A. Shelton, A. P. Siemion, M. Wagner, G. Watts, D. Werthimer, and M. Whitehead, "Advanced multi-beam spectrometer for the green bank telescope," in *General Assembly and Scientific Symposium, 2011 XXXth URSI*, Aug 2011, pp. 1–4. 4

[18] A. P. Siemion, G. C. Bower, G. Foster, P. L. McMahon, M. I. Wagner, D. Werthimer, D. Backer, J. Cordes, and J. van Leeuwen, "The allen telescope array fly's eye survey for fast radio transients," *The Astrophysical Journal*, vol. 744, no. 2, p. 109, 2012. [Online]. Available: http://stacks.iop.org/0004-637X/744/i=2/a=109 4

[19] R. Nan, D. Li, C. Jin, Q. Wandg, L. Zhu, W. Zhu, H. Zhang, Y. Yue, and L. Qian, "The five-hundred-meter aperture spherical radio telescope (FAST) project," *International Journal of Modern Physics D*, vol. 20, no. 06, pp. 989–1024, 2011. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218271811019335 4

[20] A. Chippendale and A. Schinckel, "ASKAP: Progress towards 36 parabolic reflectors with phased array feeds," in *General Assembly and Scientific Symposium, 2011 XXXth URSI*, Aug 2011, pp. 1–4. 4, 36

[21] G. Hampson, A. Brown, J. Bunton, S. Neuhold, R. Chekkala, T. Bateman, and J. Tuthill, "ASKAP redback-3 – an agile digital signal processing platform," in *General Assembly and Scientific Symposium (URSI GASS), 2014 XXXIth URSI*, Aug 2014, pp. 1–4. 4

[22] A. Szomoru, "The uniboard: A multi-purpose scalable high-performance computing platform for radio-astronomical applications," in *General Assembly and Scientific Symposium, 2011 XXXth URSI*, Aug 2011, pp. 1–4. 4

[23] J. Kocz, L. Greenhill, B. Barsdell, G. Bernardi, A. Jameson, M. Clark, J. Craig, D. Price, G. Taylor, F. Schinzel *et al.*, "A scalable hybrid FPGA/GPU FX correlator," *Journal of Astronomical Instrumentation*, vol. 3, no. 01, 2014. 4

[24] W. van Cappellen, L. Bakker, and T. Oosterloo, "Experimental results of a 112 element phased array feed for the westerbork synthesis radio telescope," in *Antennas and Propagation Society International Symposium, 2009. APSURSI '09. IEEE*, June 2009, pp. 1–4. 4

[25] G. Hampson, A. Macleod, R. Beresford, M. Brothers, A. Brown, J. Bunton, C. Cantrall, R. Chekkala, W. Cheng, R. Forsyth, R. Gough, S. Hay, J. Kanapathippillai, D. Kiraly, M. Leach, N. Morison, S. Neuhold, P. Roberts, R. Shaw, A. Schinckel, M. Shields, and J. Tuthill, "ASKAP PAF ADE – advancing an l-band PAF design towards SKA," in *Electromagnetics in Advanced Applications (ICEAA), 2012 International Conference on*, Sept 2012, pp. 807–809. 4

[26] G. Cortes-Medellin, A. Vishwas, S. Parshley, D. B. Campbell, P. Perillat, R. Black, J. Brady, K. F. Warnick, and B. D. Jeffs, "A fully cryogenic phased array camera for radio astronomy," *Antennas and Propagation, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015. 4, 30, 31, 36

[27] R. D. Norrod, J. R. Fisher, B. D. Jeffs, and K. F. Warnick, "Development of cryogenic phased array feeds for radio astronomy antennas," in *Phased Array Systems and Technology (ARRAY), 2010 IEEE International Symposium on*, Oct 2010, pp. 629–631. 4, 41

[28] Y. Wu, K. Warnick, and C. Jin, "Design study of an l-band phased array feed for wide-field surveys and vibration compensation on fast," *Antennas and Propagation, IEEE Transactions on*, vol. 61, no. 6, pp. 3026–3033, June 2013. 4

[29] H. L. V. Trees, *Optimum Array Processing*. Wiley-Interscience, 2002. 4, 13

[30] A. R. Thompson, J. M. Moran, and G. W. Swenson Jr., *Interferometry and Synthesis in Radio Astronomy*. Wiley-VCH, 2001. 4

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

[32] J. Landon, M. Elmer, J. Waldron, D. Jones, A. Stemmons, B. D. Jeffs, K. F. Warnick, J. R. Fisher, and R. D. Norrod, "Phased array feed calibration, beamforming, and imaging," *The Astronomical Journal*, vol. 139, no. 3, p. 1154, 2010. 5, 11, 17, 30, 34

[33] M. Elmer, B. D. Jeffs, K. F. Warnick, J. R. Fisher, and R. D. Norrod, "Beamformer design methods for radio astronomical phased array feeds," *Antennas and Propagation, IEEE Transactions on*, vol. 60, no. 2, pp. 903–914, Feb 2012. 5, 11

[34] M. J. Elmer, "Improved methods for phased array feed beamforming in single dish radio astronomy," Ph.D. dissertation, Brigham Young University, 2012. 5, 31

[35] D. Hayman, T. Bird, K. Esselle, and P. Hall, "Experimental demonstration of focal plane array beamforming in a prototype radiotelescope," *Antennas and Propagation, IEEE Transactions on*, vol. 58, no. 6, pp. 1922–1934, June 2010. 5

[36] K. Zarb-Adami, A. Faulkner, J. bij de Vaate, G. Kant, and P. Picard, "Beamforming techniques for large-n aperture arrays," in *Phased Array Systems and Technology (ARRAY), 2010 IEEE International Symposium on*, Oct 2010, pp. 883–890. 5

[37] K. F. Warnick and B. D. Jeffs, "Gain and aperture efficiency for a reflector antenna with an array feed," *IEEE Antennas and Wireless Propagation Letters*, vol. 5, p. 499, 2006. 5

[38] J. Landon, B. D. 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. 5, 13, 14, 44

[39] B. D. 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. 5, 14

[40] S. Van Der Tol and A. van der Veen, "Performance analysis of spatial filtering of rf interference in radio astronomy," *Signal Processing, IEEE Transactions on*, vol. 53, no. 3, pp. 896–910, 2005. 5, 14, 55

[41] R. Black, "Digital back end development and interference mitigation methods for radio telescopes with phased-array feeds," Ph.D. dissertation, Brigham Young University, 2014. 5, 31, 36, 81

[42] 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. 5

[43] G. Hellbourg, A. Chippendale, M. Kesteven, and B. D. Jeffs, "Reference antenna-based subspace tracking for rfi mitigation in radio astronomy," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, Dec 2014, pp. 1286–1290. 5

[44] G. Hellbourg, "Radio frequency interference spatial processing for modern radio telescopes," Ph.D. dissertation, Université d'Orléans, 2014. 5, 14

[45] R. T. Behrens and L. L. Scharf, "Signal processing applications of oblique projection operators," *Signal Processing, IEEE Transactions on*, vol. 42, no. 6, pp. 1413–1424, 1994. 5

[46] 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. P. 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*, Oct 2006, pp. 2031–2035. 7

[47] J. DAmbrosia, S. Rogers, and J. Quilici, "Xaui: an overview," *white paper*, 2002. [Online]. Available: https://www.10gea.org/whitepapers/xaui-interface/ 8

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

[49] 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. 10

[50] R. A. Monzingo and T. W. Miller, *Introduction to Adaptive Arrays*. SciTech Publishing, 2003. 10

[51] B. D. Steinberg, *Microwave Imaging with Large Antenna Arrays*. John Wiley & Sons Inc, 1983. 11

[52] K. F. Warnick, B. D. Jeffs, J. Landon, J. Waldron, D. Jones, J. R. Fisher, and R. D. 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. 12, 30

[53] M. Elmer, B. D. Jeffs, and K. F. Warnick, "Reducing relative gain and noise response variations for phased-array feed imaging of radio astronomical sources," *Antennas and Propagation, IEEE Transactions on*, vol. 62, no. 12, pp. 6067–6080, Dec 2014. 12

[54] B. D. Jeffs, K. F. Warnick, J. Landon, J. Waldron, D. Jones, J. R. Fisher, and R. D. 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, Oct 2008. 14

[55] K. F. Warnick and B. D. Jeffs, "Efficiencies and system temperature for a beamforming array," *Antennas and Wireless Propagation Letters, IEEE*, vol. 7, pp. 565–568, 2008. 15, 17

[56] F. Ulaby, *Microwave Radar and Radiometric Remote Sensing*. University of Michigan Press, 2013. 17

[57] K. Rohlfs, T. L. Wilson, and K. Rolfs, *Tools of Radio Astronomy (Astronomy and Astrophysics Library,)*. Springer, 1999. 17

[58] D. B. Campbell, "Measurement in Radio Astronomy," in *Single-Dish Radio Astronomy: Techniques and Applications*, ser. Astronomical Society of the Pacific Conference Series, S. Stanimirovic, D. Altschuler, P. Goldsmith, and C. Salter, Eds., vol. 278, Dec. 2002, pp. 81–90. 18

[59] D. W. Allan, "Statistics of atomic frequency standards," *Proceedings of the IEEE*, vol. 54, no. 2, pp. 221–230, 1966. 18, 49

[60] D. Land, A. Levick, and J. Hand, "The use of the allan deviation for the measurement of the noise and drift performance of microwave radiometers," *Measurement Science and Technology*, vol. 18, no. 7, p. 1917, 2007. 18, 49

[61] J. Chennamangalam, "The polyphase filter bank technique," online. [Online]. Available: https://casper.berkeley.edu/wiki/The_Polyphase_Filter_Bank_Technique 22

[62] R. G. Lyons, *Understanding Digital Signal Processing (3rd Edition)*. Prentice Hall, 2010. 22

[63] M. G. Bellanger and J. Daguet, "Tdm-fdm transmultiplexer: Digital polyphase and fft," *Communications, IEEE Transactions on*, vol. 22, no. 9, pp. 1199–1205, Sep 1974. 22

[64] W. T. Padgett and D. V. Anderson, *Fixed-Point Signal Processing*, ser. Synthesis Lectures on Signal Processing. Morgan & Claypool Publishers, 2009. [Online]. Available: http://dx.doi.org/10.2200/S00220ED1V01Y200909SPR009 26

[65] A. Mattana, M. Bartolini, and M. Naldi, "A digital backend architecture for fourier imaging," IRA 449/11. [Online]. Available: https://www.ira.inaf.it/Library/rapp-int/449-11.pdf 27

[66] T. D. Webb, *Design and Polarimetric Calibration of Dual-Polarized Phased Array Feeds for Radio Astronomy*. Brigham Young University, 2012. 29

[67] D. M. Pozar, *Microwave Engineering*. Wiley, 2011. 30

[68] V. Asthana, "Development of l-band down converter boards and real-time digital backend for phased array feeds," Ph.D. dissertation, Brigham Young University, 2012. 31

[69] J. Deneva, J. Cordes, M. McLaughlin, D. Nice, D. Lorimer, F. Crawford, N. Bhat, F. Camilo, D. Champion, P. Freire *et al.*, "Arecibo pulsar survey using alfa: probing radio pulsar intermittency and transients," *The Astrophysical Journal*, vol. 703, no. 2, p. 2259, 2009. 36

[70] H. Kuehr, A. Witzel, I. I. K. Pauliny-Toth, and U. Nauber, "A catalogue of extragalactic radio sources having flux densities greater than 1 Jy at 5 GHz," *Astronomy and Astrophysics Supplement Series*, vol. 45, pp. 367–430, Sep. 1981. 37

[71] P. Welch, "The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on audio and electroacoustics*, pp. 70–73, 1967. 40, 45

[72] J. C. Landon, "Development of an experimental phased array feed system and algorithms for radio astronomy," Ph.D. dissertation, Brigham Young University, 2011. 41

[73] C. S. Institute, "Xephem: The serious interactive astronomical software ephemeris." [Online]. Available: http://www.clearskyinstitute.com/xephem/ 41

[74] D. E. Knuth, *Art of Computer Programming, Volume 2: Seminumerical Algorithms (3rd Edition)*. Addison-Wesley Professional, 1997, p. 232. 55

Appendix A

xRTB Operation

A.1 Summary

The Beamformer captures beamformed data for up to 7 simultaneous beams. Using the x64 ADC, the Beamformer can operate using up to 64 channels sampled at 50 MHz. The beamformed data is saved to the host PC in a binary .mat file, and Matlab can be used to view the data in real time or to review the data at a later time. The Beamformer can also be used as part of a Real Time Interference Cancellation (RTIC) system. This system requires multiple ROACH boards running simultaneously. This system uses the Beamformer (one ROACH board) and the Real Time Correlator (two ROACH boards) as well as a couple of Matlab functions that perform the subspace projection and display the data. The RTIC system currently only functions with a single beam.

The control code for the Beamformer was written to minimize the number of function calls to run the system. This minimalist operation is explained here. For more discussion of the functions controlling the Beamformer, see Section A.9.

Note: In this document some code statements span multiple lines. For Python, the line extend character is the backslash ('\'). For Matlab, the line extend character is an ellipsis ('...').

A.2 Beamformer Weight Calculations

Because there are a few different ways to collect correlation data, there are a number of functions used to calculate beamformer weights. The correlation data will generally be coming from a calibration grid generated by the DAQ system, so generating weights using this system will be discussed first. A correlation matrix can also be generated by the F/X Engine Although the F/X Engine is better suited for the RTIC system, a set of weights can also be generated using its output. This will be discussed second.

Generating weights using the DAQ system

There are three Matlab functions that can be used to generate a set of weights using a correlation matrix from the DAQ system: `createBfCoeffFile_fromDAQ.m`, `calcSteeringVectors_DAQ.m` and `createBfCoeffs_DAQ_fromGrid.m`. The first is possibly the simplest to use, but the next two were written to speed up the process when generating weights from a calibration grid.

Generating weights using the F/X Engine

To calculate weights from the F/X Engine, there are four functions: `createBfCoeffFile_fromh5c.m`, `createBfCoeffs_XENG_fromGrid.m`, `calcSteeringVectors_XENG.m`, and `readhdf5c.m`. Again, the first is the simplest and the next two are for generating weights from a grid. The last file is a simple helper function needed to parse the `.h5c` files generated from the F/X Engine output.

A.2.1 createBfCoeffFile_fromDAQ.m

```
[w, w_vec] = createBfCoeffFile_fromDAQ( ...
                    R_o, R_n, R_nn, elemEnd, fileName)
```

w	A 64x64 matrix containing the beamformer weights.
w_vec	A vector of length 4096 containing the reorganized beamformer weights (see Section A.10.1).
R_o	Filename for the "on pointing" correlation matrix.
R_n	Filename for the "off pointing" correlation matrix.
R_nn	Filename for the local noise correlation matrix.
elemEnd	Total number of elements.
fileName	Filename for the output coefficient file. A .coe extension will be added.

R_o should be a string giving the filename for the "on pointing" correlation matrix. This matrix should be 64x64x256 - 64 elements x 64 elements x 256 frequency channels.
If less than 64 elements were used, then this matrix should be organized so that all non-zero elements are contiguous starting with index 1. For example, consider the 3x3 matrices below. These correspond to some arbitrary 3 element correlation matrices, with only 2 non-zero elements (i.e. only two elements are connected to the DAQ). The first matrix is organized correctly, with all non-zero elements being contiguous and starting at index 1. The other two are organized incorrectly.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & 0 \\ a_{2,1} & a_{2,2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_{1,1} & 0 & a_{1,3} \\ 0 & 0 & 0 \\ a_{3,1} & 0 & a_{3,3} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{2,2} & a_{2,3} \\ 0 & a_{3,2} & a_{3,3} \end{bmatrix}$$

In addition to the organization of the correlation matrix, the `.mat` file containing the matrix has some restrictions. When this function was written, the name of the correlation variable was unknown; therefore, it was assumed that the `.mat` file would only contain a single variable.

R_n should be a string giving the filename for the "off pointing" correlation matrix. This matrix should be 64x64x256 - 64 elements x 64 elements x 256 frequency channels. This file has the same restrictions and assumptions as R_o

R_nn should be a string giving the filename for the local noise correlation matrix. This matrix should be 64x64x256 - 64 elements x 64 elements x 256 frequency channels. This file

has the same restrictions and assumptions as R_o

elemEnd should be an integer value 1-64.

fileName should be a string giving the desired output coefficient filename. This will have a `.coe` extension added to it.

A.2.2 calcSteeringVectors_DAQ.m

```
[steer_vecs] = calcSteeringVectors_DAQ(file, nElem)
```

steer_vecs	An array of structs containing the steering vectors for each grid pointing.
files	Cell array containing the correlation files to use.
nElem	Number of elements in the correlation files.

files should be a <arbitrary number of files>x2 cell array of strings. Each row in the array contains the correlation filenames for the on and off pointing for each grid pointing. These correlation files follow the same restrictions as above, with some additions. The correlation matrix variable needs to be named `corr` in both the on and the off pointings. The "on pointing" `.mat` file also needs to contain `row` and `col` variables giving the row and column within the calibration grid.

nElem should be an integer value 1-64.

A.2.3 createBfCoeffs_DAQ_fromGrid.m

```
[w, w_vec, w_notScaled] = createBfCoeffs_DAQ_fromGrid( ...
                steeringVecs, pointing, R_nn, nElem, fileName)
```

w	A 64x64 matrix containing the beamformer weights.
w_vec	A vector of length 4096 containing the reorganized beamformer weights (see Section A.10.1).
w_notScaled	A 64x64 matrix containing the beamformer weights. These weights have not been scaled down to 8 bits yet.
steeringVecs	Cell array containing the correlation files to use.
pointing	A row vector [x,y] containing the x and y coordinates within the grid for this pointing.
R_nn	Filename for the local noise correlation matrix (`.mat` file).
nElem	Number of elements in the correlation files.
fileName	Filename for the output coefficient file. A .coe extension will be added.

steeringVecs should be an array of structs as returned from `calcSteeringVectors_DAQ.m`.

pointing should be a row vector containing the x and y coordinates within the grid that corresponds to the desired grid pointing. The steering vector for this pointing will be used to generate a set of beamformer weights.

R_nn should be a string containing the filename of the local noise correlation matrix. This file should be a `.mat` file, following the same rules as the other no signal correlation matrices.

nElem should be an integer 1-64.

fileName should be a string giving the desired output coefficient filename. This will have a `.coe` extension added to it.

A.2.4 createBfCoeffFile_fromh5c.m

```
[w, w_vec] = createBfCoeffFile_fromh5c( ...
                R_o, R_n, R_nn, elemEnd, fileName)
```

w	A 64x64 matrix containing the beamformer weights.
w_vec	A vector of length 4096 containing the reorganized beamformer weights (see Section A.10.1).
R_o	Filename for the "on pointing" `.h5c` file.
R_n	Filename for the "off pointing" `.h5c` file.
R_nn	Filename for the local noise `.h5c` file.
elemEnd	Total number of elements.
fileName	Filename for the output coefficient file. A .coe extension will be added.

R_o should be a string containing the filename of the "on pointing" correlation `.h5c` file. The F/X Engine will output a `.h5` file. This file must be modified by the `fix_med_corr.py` python function to parse and reorganize the data. From the `.h5` file a `.h5c` file is created that has all the data in the correct format so that the correlation matrix can be extracted.

R_n hould be a string containing the filename of "off pointing" correlation `.h5c` file.

R_nn hould be a string containing the filename of local noise correlation `.h5c` file.

elemEnd should be an integer 1-64

fileName should be a string giving the desired output coefficient filename. This will have a `.coe` extension added to it.

A.2.5 createBfCoeffs_XENG_fromGrid.m

```
[w, w_vec] = createBfCoeffs_XENG_fromGrid( ...
             steeringVecs, pointing, R_nn, nElem, fileName)
```

w	A 64x64 matrix containing the beamformer weights.
w_vec	A vector of length 4096 containing the reorganized beamformer weights (see Section A.10.1).
steeringVecs	Cell array containing the correlation files to use.
pointing	A row vector [x,y] containing the x and y coordinates within the grid for this pointing.
R_nn	Filename for the local noise correlation matrix (`.h5c` file).
nElem	Number of elements in the correlation files.
fileName	Filename for the output coefficient file. A .coe extension will be added.

steeringVecs should be an array of structs as returned from `calcSteeringVectors_XENG.m`.

pointing should be a row vector containing the x and y coordinates within the grid that corresponds to the desired grid pointing. The steering vector for this pointing will be used to generate a set of beamformer weights.

R_nn should be a string containing the filename of the local noise correlation matrix. This file should be a `.h5c` file.

nElem should be an integer 1-64.

fileName should be a string giving the desired output coefficient filename. This will have a `.coe` extension added to it.

A.2.6 calcSteeringVectors_XENG.m

```
[steer_vecs] = calcSteeringVectors_XENG(files)
```

files	Cell array containing the correlation filenames.

file should be a <number of files>x4 cell array. (x,1) is the on pointing file, (x,2) is the off pointing file, (x,3) and (x,4) are the row and column to which this on pointing corresponds.

A.2.7 readhdf5c.m

```
[corr] = readhdf5c(filename, chOut)
```

corr	A 64x64xN array containing correlation matrices across N frequency bins.
filename	Filename of the `.h5c` file to parse.
chOut	An integer used to determine how many frequency bins to output.

filename should be a string containing the filename of the `.h5c` file to parse.

chOut should be an integer. This is used to determine how many frequency bins will be included in the output correlation matrix. If this value is 64, every 16th bin will be output, starting with bin 9. If this value is anything else, all 1024 bins will be output.

A.3 ROACH Control via Python

The ROACH is controlled by the python file `BeamformerCtrl.py`. The main function which is used to run the beamformer is `runBeamformer(...)`. This function will load the correct weights into the beamformers, create a directory to store data, write a log file, and collect data for the specified amount of time.

Usage Example:

```
>> import BeamformerCtrl
>> roach = BeamformerCtrl.x64('roach3')
>> allWeightFiles = ['~/Files/weightFile1', \
                     '~/Files/weightFile2', \
                     '~/Files/weightFile3']
>> roach.runBeamformer(allWeightFiles, './newDirectory', \
                        1, 0.5, 7)
```

This will program the roach with the current boffile, initialize a list of coefficient files, then run the beamformers using the coefficients specified in the list of coefficient files. Data will be gathered for 1 second using a 0.5 second integration time. Bits 7-10 of the FFT output will be used as the input for the beamformers. A date stamp will be appended to the end of `./newDirectory` along with an integer indicating which run this is for that day (i.e. `./newDirectory_2013_July_11_1` would be the first run on July 11 2013. Next would be `./newDirectory_2013_July_11_2` and so on).

A.3.1 x64

BeamformerCtrl.x64 is the wrapper class for controlling x64 based ROACH builds. The roach variable must be initialized before any functions can be called.

```
__init__(roach)
```

roach	IP address for the ROACH board to be used

roach should be the IP address for the roach to be used. (CasperMain has had its `/etc/hosts` file modified so that 'roach3' points to the IP address for one of our ROACH boards).

A.3.2 runBeamformer

```
runBeamformer(allWeightFiles, dirName, duration, period, \
              sliceIdx, noVerify=True)
```

allWeightFiles	A list of coefficient filenames.
dirName	Path for the directory in which to save data.
duration	Total length of time to run beamformer (seconds).
period	Integration time (seconds).
sliceIdx	LSB of FFT output to be used to the beamformer.
noVerify	For future use.

allWeightFiles should be a list of strings containing the path for each coefficient file to use. This list must contain 1-7 filenames. These paths should NOT contain the `.coe` extension. The weight files themselves must have the `.coe` extension, but this is not to be included in this list. The coefficient files used should have an associated `.log` file with the same basename as the `.coe` file. This log file is created by the same Matlab function that creates the `.coe` file.

dirName should be a string containing the path for the directory in which to save data. This directory path will have a date stamp added along with an index corresponding to which run this is for that date. For example if dirName was `TEST`, and this was the first time this function was run today, a directory named `TEST_2013_July_11_1` would be created. If run again with the same dirName, a directory named `TEST_2013_July_11_2` would be created.

duration should be a value greater than 0. The function will run for the specified amount of time. If you wish to stop the function prior, a simple ctrl+c will stop the function.

period should be a value greater than 0.01 OR -1. If -1, only one FFT will be accumulated (this is mostly for testing purposes). This will be the integration time for the beamformers (in seconds). If the integration length is too small, the data will not be able to be read quickly enough, and this function will quit if an accumulation is missed.

sliceIdx must be an integer 0-7. This corresponds to the LSB of the FFT ouput to use for the beamformer input. A 4 bit window is sliced off of the FFT output.

noVerify If True, the weights will not be read after being written (to verify that the proper weights were written correctly). The current beamformer model does not support this verification, so noVerify will default to True.

A.4 Real Time Display via Matlab

The function `BeamformerCtrl.x64.runBeamformer(..)` continually saves data while running. There are two Matlab functions that can be used to view this data. This can be done while `BeamformerCtrl.x64.runBeamformer(..)` is running, or using previously recorded data.

A.4.1 plotBfData_noAvg_numBfs.m

```
[ ] = plotBfData_noAvg_numBfs(dirPath, fileSkip, accLen, ...
                              numBfs, removeBins)
```

dirPath	Path for the directory containing beamformed data.
fileSkip	Number of files to skip from beamformed output data.
accLen	Integration time (seconds).
numBfs	Number of beamformers to plot.
removeBins	Number of bins to ignore from the ends of the spectrum.

dirPath should be a string containing the directory in which the beamformed data was saved. This path will be given by `runBeamformer(...)` after it has finished initialization. This path can also be set to a directory containing previously recorded data.

fileSkip should be a positive integer. This function has a refresh rate of about 0.25 seconds (depending on the machine). If `runBeamformer(...)` has an accumulation length less then 0.25, then this function will not be able to display the beamformer output in real time. If this is the case, fileSkip can be set so that a number of files are skipped between each refresh. For example: if an accumulation length of 0.05 is set, using a fileSkip of 4 will plot every 5th file which will keep up with real time better. This function will output to the console the elapsed time between each refresh. If the next data file has not been written yet, then a `waiting...` message will be written. If that waiting message is being written often, you can be sure that this plotting function is keeping up with the real time data coming out of the beamformer.

accLen should be the same value that was used in `runBeamformer(...)`

numBfs should be an integer 1-7.

removeBins should be an integer 0-128. Because the bandwidth of the downconverter boards is less than 25MHz, some of the beamformer weights near the edges of the spectrum are artificially larger than they should be. These incorrect bins can be ignored in the display by setting removeBins greater then zero. This number of bins will be zeroed out on both ends of the spectrum. Default is 28; bins 1-28 and 228-256 will be set to zero when displayed.

A.4.2 plotBfData.m

```
[ ] = plotBFData(dirPath, numFiles, accLen, removeBins)
```

dirPath	Path for the directory containing beamformed data.
numFiles	Number of files to average from beamformed output data.
accLen	Integration time (seconds).
removeBins	Number of bins to ignore from the ends of the spectrum.

dirPath should be a string containing the directory in which the beamformed data was saved. This path will be given by `runBeamformer(...)` after it has finished initialization. This path can also be set to a directory containing previously recorded data.

numFiles should be an integer greater than 0. This variable controls how many files are averaged together when plotted. A smaller number updates more quickly, but can lead to some lagging if the beamformer is saving data too quickly. It is difficult to give a good idea of what this value should be, but with an accumulation length of 0.25 seconds, a numFiles value of 2 works well. This function will output to the console the elapsed time between each refresh. If the next data file has not been written yet, then a `waiting...` message will be written. If that waiting message is being written often, you can be sure that this plotting function is keeping up with the real time data coming out of the beamformer.

accLen should be the same value that was used in `runBeamformer(...)`

removeBins should be an integer 0-128. Because the bandwidth of the downconverter boards is less than 25MHz, some of the beamformer weights near the edges of the spectrum are artificially larger than they should be. These incorrect bins can be ignored in the display by setting removeBins greater then zero. This number of bins will be zeroed out on both ends of the spectrum. Default is 28; bins 1-28 and 228-256 will be set to zero when displayed.

A.5 Real Time Interference Cancellation Control Code

The Real Time Interference Cancellation (RTIC) system is a bit more complex then the simple beamformer. The RTIC system requires 3 ROACH boards, 2 of which have the x64 adc attached. One ROACH runs the beamformer, while the other two run the F/X Engine. The operation of the F/X Engine is not covered here, but Zhu Kai, a visiting engineer from NAOC, left detailed notes on its operation.

There are three major pieces of code for the RTIC system: the RTIC calculations handle all all of the subspace projection that cancels out interference; the python ROACH control function handles all of the ROACH control signals and other I/O; and the Matlab display functions handle the display of the beamformed data.

A.6 RTIC Calculations

The RTIC system starts with an initial set of beamformer weights that are constantly modified to cancel out a major interference source. To do this, the F/X Engine needs to be running and saving new `.h5` files which contain the new correlation data. These `.h5` files are parsed and converted to a `.h5c` file. Once the `.h5c` file is created, it is read and used to create a new set of beamformer weights. These weights are then be loaded into the beamformer.

A.6.1 createBfCoeffs_RTIC.m

```
[ ] = createBfCoeffs_RTIC(w_in, h5cDir, coeDir, ...
                          coeBaseName, nElem)
```

w_in	A 64x64 matrix containing the initial beamformer weights.
h5cDir	The directory where the `.h5c` files are being created.
coeDir	The directory in which the new `.coe` files should be saved.
coeBaseName	The basename for the new `.coe` files.
nElem	Number of elements to be used. Should be 32 for RTIC system.

w_in should be a 64x64 matrix which would be returned from most of the other `createBfCoeff` functions.

h5cDir should be a string containing the path for the directory in which the new `.h5c` files are being saved. This function will periodically check for a new `.h5c` file in this directory. As soon as a new file is available, it will be used to make a new set of beamformer weights.

coeDir should be a string containing the path for the directory in which to save the new coefficient files.

coeBaseName should be a string containing the basename for the new coefficient files. If the desired filename is `test.coe`, the basename would be `test`. When a new coeffecient file is created, an integer index is appended to the end of the basename which then has a `.coe` extension added to the end. The integer index is used by the python RTIC control function.

nElem should be an integer 1-64.

A.7 RTIC ROACH Control via Python

The python portion of the RTIC system functions similarly to the standard beamformer. The major difference is that the RTIC system periodically checks for a new set of beamformer weights and uploads them to the ROACH as soon as thy are available.

A.7.1 runRTICBeamformer

```
runRTICBeamformer(allWeightFiles,dirName,coeDirName, \
                  coeBaseName, period, sliceIdx)
```

allWeightFiles	A list of coefficient filenames.
dirName	The path for the directory in which to save data files.
coeDirName	The path for the directory in which the new `.coe` files are being written.
coeBaseName	Basename of the new `.coe` files which are being written.
period	Integration time (seconds).
sliceIdx	LSB of FFT output to be used to the beamformer.

allWeightFiles should be a list of strings containing the path for each coefficient file to use. For the RTIC system, this list should only contain 1 filename. This path should NOT contain the `.coe` extension. The weight files themselves must have the `.coe` extension, but this is not to be included in this list. The coefficient files used should have an associated `.log` file with the same basename as the `.coe` file. This log file is created by the same Matlab function that created the `.coe` file.

dirName should be a string containing the path for the directory in which to save data. This directory path will have a date stamp added along with an index corresponding to which run this is for that date. For example if dirName was `TEST`, and this was the first time this function was run today, a directory named `TEST_2013_July_11_1` would be created. If run again with the same dirName, a directory named `TEST_2013_July_11_2` would be created.

coeDirName should be a string containing the path for the directory in which to check for new `.coe` files.

coeBaseName should be a string containing the basename of the new `.coe` files which are being written. If the new files are names `Weights_x.coe` then the basename would be `Weights`.

period should be a value greater than 0.01 OR -1. If -1, only one FFT will be accumulated (this is mostly for testing purposes). This will be the integration time for the beamformers (in seconds). If the integration length is too small, the data will not be able to be read quickly enough, and this function will quit if an accumulation is missed.

sliceIdx must be an integer 0-7. This corresponds to the LSB of the FFT ouput to use for the beamformer input. A 4 bit window is sliced off of the FFT output.

A.8 RTIC Real Time Display via Matlab

The function `BeamformerCtrl.x64.runRTICBeamformer(..)` saves the data from the beamformers continually while it is running. The Matlab function to view this data is similar to the other Matlab display functions, but they are not interchangeable. Like the other display functions, this function can be used while the RTIC system is running, or it can be used on previously recorded data.

A.8.1 plotRTICData_noAvg.m

```
function [ ] = plotRTICData_noAvg(dirPath, fileSkip, ...
                                  accLen, removeBins)
```

dirPath	Path for the directory containing beamformed data.
fileSkip	Number of files to skip from beamformed output data.
accLen	Integration time (seconds).
removeBins	Number of bins to ignore from the ends of the spectrum.

dirPath should be a string containing the directory in which the beamformed data was saved. This path will be given by `runRTICBeamformer(...)` after it has finished initialization. This path can also be set to a directory containing previously recorded data.

fileSkip should be a positive integer. This function has a refresh rate of about 0.25 seconds (depending on the machine). If `runRTICBeamformer(...)` has an accumulation length less then ~0.25, then this function will not be able to display the beamformer output in real time. If this is the case, fileSkip can be set so that a number of files are skipped between each refresh. For example: if an accumulation length of 0.05 is set, using a fileSkip of 4 will plot every 5th file which will keep up with real time better. This function will output to the console the elapsed time between each refresh. If the next data file has not been written yet, then a `waiting...` message will be written. If that waiting message is being written often, you can be sure that this plotting function is keeping up with the real time data coming out of the beamformer.

accLen should be the same value that was used in `runRTICBeamformer(...)`

removeBins should be an integer 0-128. Because the bandwidth of the downconverter boards is less than 25MHz, some of the beamformer weights near the edges of the spectrum are artificially larger than they should be. These incorrect bins can be ignored in the display by setting removeBins greater then zero. This number of bins will be zeroed out on both ends of the spectrum. Default is 28; bins 1-28 and 228-256 will be set to zero when displayed.

A.9 Other Beamformer Operations

There are a number of functions that can be used to control many different beamformer actions. Some of these are buried in the wrapper functions `runBeamformer` and `runRTICBeamformer`, while others were written for testing/debugging purposes. This section contains brief explanations for many of the functions contained in `BeamformerCtrl.x64` class.

`set_bitstream(new_bitstream)`

A class variable (`bitsream`) contains the current boffile filename. This function updates this variable without reprogramming the ROACH board.

`set_new_bitsream()`

A class variable dict (`bitstreams`) contains a number of beamformer boffiles. Many of these are still in the testing stages. This function will ask you to choose which bitstream to use and reprograms the ROACH.

`program_roach()`

Program the ROACH using the boffile currently contained in the class variable `bitstream`.

`calibrate_adc()`

Run the ADC calibration script (previously `adc_softcal.py`).

`get_data(bram_num='', log = '')`

Plot the data from beamformer number <bram_num>. To plot on a log scale, use <log>= 'log'. <bram_num> defaults to an empty string (for older beamformer models with only a single beamformer). Data is returned.

`get_all_data(bram_num, log = '')`

Plot the data from multiple beamformers (controlled by <bram_num>) on the same plot. To plot on a log scale, use <log> = 'log'. Data is not returned, only displayed on screen.

```
updateBFCoeffs(fileName, bf_idx, bramName='coeff_bram', \
               bramSize=2048, triggerReg='ctrl')
```

Update the coefficients for a single beamformer. <fileName> is the path for the coefficient file, and <bf_idx> is the integer index corresponding to the beamformer to updata (1-7).

```
updateBFCoeffs_no_print(fileName, bf_idx, bramName='coeff_bram',\
                        bramSize=2048, triggerReg='ctrl')
```

Update the coefficients for a single beamformer with out printing anything to the console. Uses the same definition as `updateBFCoeffs`.

`set_slice(idx)`

Set the slice window for the FFT output. <idx> must be an integer 0-7.

`set_acc_len(seconds)`

Change the amount of time for each accumulation. Must be greater than 0.00002. Can be set to -1 to turn off accumulation (i.e. beamformer output is not accumulated - each output is saved to the final bram).

`toggle_bf_input(value='00000000')`

Toggle the beamformer input between the F-Engine and a specified value. <value> must be a 32 bit hex string (with no leading '0x\'

`test_suite(numFileRuns, acclen, numBfs, getOutFiles=False)`

Run a test to check that the outputs of the beamformers are correct. This function reads the output of the beamformers and checks them against a known correct data set. A correct data set is aquired by setting <getOutFiles> True. <numFileRuns> is an integer that controll how many times each coefficient file is checked. <acclen> must match the acclen for the known data set. <numBfs> is an integer 1-7.

`read_snaps(filename, bf_idx,start, end)`

Read the snap blocks in `bf7_read_weights_2013_Jun_03_1331.bof`. To trigger a read, a coefficient file <filename> is written to a beamformer <bf_idx> until it is known than the weights were written incorrectly. <start> and <end> are the bounds on how many snap addresses to display. Data is not returned, only printed to the terminal.

`get_bram_contents(bram_name, data_width, address_width)`

Read a yellow block shared bram <bram_name> of size <data_width> and <address_width>. Data is returned.

`save_bram_contents(bram_name, data_width, address_width, filename)`

Similar to `get_bram_data(..)` except data is saved to a file <filename> with a `.brm` extension.

```
check_bf_weights(bf_idx, coe_filename, \
                 save_file=False, brm_filename='bramFile')
```

Verify the weights of the specified beamformer. Currently loaded beamformer weights can be saved to a file: set <save_file> to True and specify a <brm_filename>.

`read_coeffs(bf_idx)`

Read the currently loaded coefficients in beamformer number <bf_idx>. Data is returned.

`check_sync()`

Check if the ADC chips are correctly synced. Pulse the sync signal if the chips are ready.

`rerun_cal()`

Re-program and re-calibrate ROACH.

`adc_reset(self)`

Send the ADC reset signal.

`save_output(outfile, outval)`

Pickle a variable <outval> into a file <outfile>.

`compare_files(file1, file2, shallow=False)`

Compare the contents of two files <file1> <file2> using `Lib.filecmp.py`. See `filecmp.__doc__` for more info.

A.10 Beamformer Coeffecient Files

The coefficients used in the beamformers are read out of a coeffecient file with a `.coe` extension. These files have a very specific organization and format. The correct way to organize and format these files is explained here.

A.10.1 Coefficient File Organization

Like the ADC input ordering, the coefficient file has a confusing organization. The organization comes from the need to present coefficients in the order that the ADC samples data. Because not all inputs are sampled simultaneously, they are also not presented to the beamformers simultaneously. The beamformers see samples from inputs 0, 1, 16, 17, 32, 33, 48, 49 at once, followed by 8, 9, 24, 25, 40, 41, 56, 57 and so on. The coeffecient file must be organized so that all frequency bins for each set of eight inputs are presented before moving on to the next set of eight inputs.

In addition to the input ordering, the beamformers have been designed to use a single weight for four consecutive frequency bins. (This was done to meet timing constraints.) So, for the 64 input, 512 point FFT design, the coefficients can be organized into a 64 x 64 matrix.

For example, consider the matrix W

$$W_{m,n} = \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{1,n-1} \\ w_{1,0} & w_{1,1} & \cdots & w_{2,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m-1,1} & w_{m-1,2} & \cdots & w_{m-1,n-1} \end{bmatrix}$$

where the m corresponds to each set of four frequency bins, and n corresponds to each input port.

The coefficients should be reorganized into a single vector before being written to the coefficient file. The correct organization of the coefficient vector $\overline{w}$ for the example matrix W is as follows:

$$\overline{w} = \begin{bmatrix} w_{0,0} & w_{0,1} & w_{0,16} & w_{1,17} & w_{0,32} & w_{0,33} & w_{0,48} & w_{0,49} \\ w_{0,8} & w_{0,9} & w_{0,24} & w_{0,25} & w_{0,40} & w_{0,41} & w_{0,56} & w_{0,57} \\ w_{0,2} & w_{0,3} & w_{0,18} & w_{1,19} & w_{0,34} & w_{0,35} & w_{0,50} & w_{0,51} \\ w_{0,11} & w_{0,12} & w_{0,26} & w_{0,27} & w_{0,42} & w_{0,43} & w_{0,58} & w_{0,59} \\ w_{0,4} & w_{0,5} & w_{0,20} & w_{1,21} & w_{0,36} & w_{0,37} & w_{0,52} & w_{0,53} \\ w_{0,12} & w_{0,13} & w_{0,28} & w_{1,29} & w_{0,44} & w_{0,45} & w_{0,60} & w_{0,61} \\ w_{0,6} & w_{0,7} & w_{0,22} & w_{1,23} & w_{0,38} & w_{0,39} & w_{0,54} & w_{1,55} \\ w_{0,14} & w_{0,15} & w_{0,30} & w_{1,31} & w_{0,46} & w_{1,47} & w_{0,62} & w_{1,63} \\ w_{1,0} & w_{1,1} & w_{1,16} & w_{2,17} & w_{2,32} & w_{2,33} & w_{2,48} & w_{2,49} \cdots \\ w_{63,14} & w_{63,15} & w_{63,30} & w_{63,31} & w_{63,46} & w_{63,47} & w_{63,62} & w_{63,63} \end{bmatrix}$$

This vector follows the input port organization explained in [41]

A.10.2 Coefficient File Format

Each line of the coefficient file contains two complex, 16 bit (8 real, 8 imaginary), coefficients written as a two's compliment hexadecimal number. Any hex designation (\x, #, 0h, etc.) is left off. As stated, each line contains two coefficients. Referring back to the coefficient vector $\overline{w}$ these will be consecutive values, i.e. $w_{0,0}$ $w_{0,1}$. The file is a simple text file with a `.coe` extension.

As an example, consider the coefficients $c_1 = 0$, $c_2 = 1$, $c_3 = j$ and $c_4 = 1 + j$.

Using these coefficients, the file would be as follows:

```
00000100
00010101
<empty line>
```

A.11 Beamformer Output

The beamformer output is fairly straightforward. The accumulated data is stored in a yellow block shared bram as a 32 bit unsigned value. This is read and saved to a `.mat` file (using `scipy.io.savemat`).Because of the way the bram read function works, the `.mat` file must still be parsed by the Matlab display functions. The data also must be corrected for the 8 bit scale value applied when the beamformer weights are generated. These scale values are saved in the `.log` file for the beamformer weights.

Appendix B

x64 GPU Correlator

B.1 Summary

The x64 GPU Correlator was written in an effort to speed up the time it takes to correlate the data received using Richard Black's x64 Data Acquisition System (DAQ). A MATLAB correlator was written and used initially, but a single calibration grid could take 20+ hours to correlate. To reduce the compute time required to correlate the data gathered from the DAQ, a new correlator was designed to take advantage of the parallelized nature of a graphics processing unit (GPU).

B.2 Theory

Let $\mathbf{x}[n]$ be a discrete random process. Estimators for the mean, autocorrelation, and autocovaraiance of $\mathbf{x}[n]$ are respectively defined as

$$\begin{aligned}
\hat{\boldsymbol{\mu}}_x &\equiv \frac{1}{N}\sum_{n=0}^{N-1}\mathbf{x}[n] \\
\hat{\mathbf{R}}_x &\equiv \frac{1}{N}\sum_{n=0}^{N-1}\mathbf{x}[n]\mathbf{x}^H[n] \\
\hat{\mathbf{C}}_x &\equiv \frac{N}{N-1}\left[\hat{\mathbf{R}}_x - \ \hat{\boldsymbol{\mu}}_x\hat{\boldsymbol{\mu}}_x^H\right].
\end{aligned}$$

It can be shown that the following is an equivalent expression for $\hat{\mathbf{C}}_x$

$$\hat{\mathbf{C}}_x \equiv \left(\frac{1}{N-1}\sum_{n=1}^{N-1}\mathbf{x}[n]\mathbf{x}^H[n]\right) - \ \left(\frac{1}{N-1}\sum_{n=1}^{N-1}\mathbf{x}[n]\right)\left(\frac{1}{N-1}\sum_{n=0}^{N-1}\ \mathbf{x}[n]\right)^H. \tag{B.1}$$

Both the MATLAB and GPU correlator implement Equation B.1 to provide an estimate for the covariance of the received signal.

B.3 Code

The code for the GPU correlator is straightforward and is contained in a single file, `x64correlator.cu`, with 365 lines of code. Much of the code handles file and packet parsing, while the meat of the code is in the five GPU kernals that perform the correlation calculations.

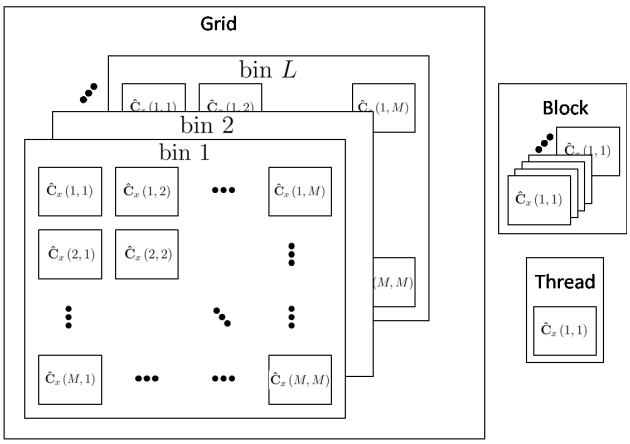


Figure B.1: This figure shows how the correlation kernel is divided into a grid of blocks and threads. The grid contains $M \times M$ blocks, each of which contains L threads. Each block computes one entry for all frequency bins with each thread computing one entry for one frequency bin.

These kernals implement the vector outer product, summation, subtraction, and division operations required in Equation B.1.

Each file that comes from the DAQ is a collection of UDP packets crated by the DAQ. To create this file, UDP packets are captured and written to disk using a program called Gulp. Gulp will write it's own small header to the top of the file, followed by the UDP packets containing the frequency channelized data. When parsing the file for correlation, the Gulp header and the UDP headers are ignored. The 8 byte DAQ header contains the FFT length and the frequency bins and ADC inputs that are included in that packet. These headers are all identical, so it is only necessary to read one. Once these parameters are gathered, memory requirements can be calculated and data storage arrays can be initialized.

The packets are processed one at a time and the data is stored into a single array. This array is copied over to the GPU device for correlation. This array is also added into a running sum which will later be used to subtract the correlation of the mean from the mean correlation. The GPU memory consists of a temporary correlation array, a running sum correlation array, a temporary data array, and a running sum data array. Each of these arrays is a two or three dimension matrix which has been flattened down to a single dimension array. The data matrix is stored in row-major order indexed first by frequency bin then by element. The correlation matrices are stored in row-major order indexed first by frequency bin then by row and column.

The performance improvement comes from the parallalization of the computations on the GPU. The correlation operation is fairly easy to parallelize as each entry in the correlation matrix can be calculated independently. The grid size of the kernal launch follows the dimensions of the output correlation matrix: MxMxL where M is the number of ADC inputs used and L is the number of frequency bins (see Figure B.1). The kernal grid is broken into MxM blocks, which are further broken down into L threads. Each thread calculates a single entry in the correlation matrix for a single frequency bin; this calculation

consists of only a complex multiply and add. Each packet's correlation matrix is added to a running sum, for later calculating the mean of the correlation.

Once all of the packets have been processed, each of the running sums is divided by $N - 1$ to find the mean. The mean data is then correlated, and subtracted from the mean correlation. The result is then saved to disk. A simple MATLAB script can then be used to read in the data and rearrange it into a format that is more convenient for the post-processing codes which had been used with the older MATLAB correlator.

B.4 Usage

To use the correlator, you need to first compile it using the Nvidia C compiler (`vncc`). There are two required arguments to run the correlator: the input raw data filename and the output correlation filename. The output filename does not need to be an existing file.

The correlator will create two files: one is a binary file with the filename you specify, and the other is a text version of the binary file. The text file will take the filename you specify and append a `.txt` extension. Both of these files have the same data order which matches the ordering of the correlation array discussed previously.

There are a number of MATLAB scripts and functions to do the post processing for the correlated data (including beamforming, sensitivity grids and imaging). To use these, you may need to format the data from the GPU correlator to match the format of the MATLAB correlator.

B.5 Performance

Although the MATLAB correlator can processes up to 12 files simultaneously, it is still very slow. It will usually take 1000 or more seconds to process each file. The GPU correlator will process each file in 60 seconds or less. This was tested by comparing the time it took to correlate a 15 by 15 beamformer calibration grid. The GPU correlator was able to correlate all 50 frequency channels of the full grid 4.5 hours. The MATLAB correlator was able to correlate only a single frequency channel of the full grid in 19.5 hours.