2019-12-01

Development of a GPU-Based Real-Time Interference Mitigating Beamformer for Radio Astronomy

Jeffrey M. Nybo
Brigham Young University

Development of a GPU Based Real-Time Interference Mitigating Beamformer for Radio Astronomy

Jeffrey M. Nybo

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 Rice

Department of Electrical and Computer Engineering

Brigham Young University

ABSTRACT

Development of a GPU Based Real-Time Interference Mitigating Beamformer for Radio Astronomy

Jeffrey M. Nybo
Department of Electrical and Computer Engineering, BYU
Master of Science

Radio frequency interference (RFI) mitigation enables radio astronomical observation in frequency bands that are shared with many modern satellite and ground based devices by filtering out the interference in corrupted bands. The present work documents the development of a beamformer (spatial filter) equipped with RFI mitigation capabilities. The beamformer is intended for systems with antenna arrays designed for large bandwidths. Because array data post processing on large bandwidths would require massive memory space beyond feasible limits, there is a need for a RFI mitigation system capable of doing processing on the data as it arrives in real-time; storing only a data reduced result into long term memory. The real-time system is designed to be implemented on both the FLAG phased array feed (PAF) on the Green Bank telescope in West Virginia, as well as future radio astronomy projects. It will also serve as the anti-jamming component in communications applications developed for the United States office of naval research (ONR). Implemented on a graphical processing unit (GPU), this beamformer demonstrates a working single step filter using nVidia's CUDA technology, technology with high-speed parallelism that makes real-time RFI mitigation possible.

Keywords: RFI mitigation, spatial filtering, Green Bank telescope, beamforming, subspace projection

ACKNOWLEDGMENTS

Of course I would like to acknowledge Dr. Brian Jeffs, who opened the door to the Master's program for me. I appreciate him willing to advise me and review my work. I appreciate his spending extra time reviewing this thesis at various stages of the process. I also deeply appreciate him defending my work and helping me to be able to finish it by giving me support where he could.

Special thanks to Dr. Wilde who always encouraged me to succeed with a smile on his face. I owe a lot of my progress and success to him. We created some great things together.

I feel it most important to acknowledge my father, who's tireless efforts in the editing process deserve special attention. He spent a significant part of his vacation reading through this work twice, on a picnic table, using only his phone and a notebook to write out his edits. He is one true hero who really helped when it counted. His love and support, along with all my family's support, are truly unparalleled.

Finally, I must acknowledge my dear and loving Heavenly Father who has been my true support these past years. Only He has been there when others were not. He helped me through the darkest times when others chose to discourage. He was there when it was hard. He always knew I would complete this work. I know I owe everything to Him. He really is the one who provided for this all to happen. Without Him, this university wouldn't exist. BYU is literally built by the Church of Jesus Christ of Latter-Day Saints. I am pleased to be here and recognize that only the widow's mite, the generous donations of many and His grace really provide the opportunities here. I will forever be indebted to Him for the many blessings he has poured out upon me at this critical time of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

MATHEMATICAL NOTATION

$\mathbf{a}$	A vector
α, a	A scalar
A	A matrix or operator
a^*	Complex conjugate of a
$\mathbf{a}^T, \mathbf{a}^H$	Transpose and conjugate transpose of $\mathbf{a}$
$E\{\mathbf{x}\}$	Expected value of $\mathbf{x}$
$\Re(\mathrm{A})$	Range space (span of the columns) of matrix A
$x[n]$	A signal (discrete time series)
k	Frequency channel index
l	Antenna array element index
m	Short term integration widow index
K	Total number of frequency channels
L	Total number of antenna elements
M	Total number of short term integration windows

CHAPTER 1. INTRODUCTION

"Astronomy compels the soul to look upwards and leads us from this world to another."

-Plato

1.1 Introduction

Radio astronomy (RA) is the discipline of observing electromagnetic radiation in the cosmos. While astronomy is often thought of as observing what the eye can see, traditional optical telescopes reveal only a small portion of all that can be found in the universe. Celestial bodies emit light in all parts of the electromagnetic spectrum. In an effort to gather more insight into these celestial bodies, astronomers will observe the universe at many frequencies. Aside from visible light wavelengths, signals can be at gamma, ultraviolet, infrared, microwave, millimeter wave and all other radio frequency bands.

From pulsars to quasars, to active galactic nuclei, to spectral line emissions from thermally excited gas clouds, the night sky is replete with radiation found in radio bands which can not be observed optically. For example, hydrogen, the most abundant element found in the universe, emits a signal at 1.42 GHz. Radio astronomers call this the HI spectral line or HI radiation. The true extended shapes of galaxies can be seen by observing HI radiation. Other examples of sources detectable in the radio frequency bands include supernovae, black holes, fast radio bursts and the cosmic microwave background radiation to name a few. Some other sources of note include the sun, Sagittarius A (the center of the Milky Way), and Cassiopeia A which is the brightest radio source in the sky other than the sun and moon. Aside from the sun, moon and Cassiopeia A, however, most signals in RA are fairly faint.

The task of detecting signals for radio astronomical observation is non trivial. Signal to noise ratios (SNRs) for most radio sources are routinely 30 to 50 dB below the ambient noise floor in the night sky. To detect deep space sources, specially designed high gain large antennas and

sophisticated signal processing algorithms have been developed over the years, always trying to provide the clearest observations.

Among the many instruments developed for RA, phased antenna arrays have emerged as a particularly desirable solution. Coupled with signal processing algorithms, the antenna array can do many things such as interferometry, image forming, direction finding, and a kind of spatial filtering called beamforming. All these techniques (and others) are used in RA, but the subject of this thesis is beamforming.

A beamformer with a digital back end enables good signal detection for weak deep space signals. A beam can be formed and electronically steered (without antenna motion) in the direction of the signal of interest (SOI). All plane waves arriving in the direction of the beam will be amplified, while any signals arriving from other directions are attenuated. If the beam is pointed at a particularly weak SOI the signal can be observed over the noise floor by gathering signal energy over long time windows and integrating the result. When this is done, the signal is seen to "poke up" over the noise floor and estimation error fluctuations. The systems presented herein do all of this, but there is still a problem.

Were the night sky completely free from man made radio interference, the above solution would be sufficient for the science, however the ever growing satellite population in the world's modern information age makes it necessary for another level of sophistication. Radio Astronomy bands of interest lie in varying bandwidths between 13 MHz and 100 GHz. As an example of interference, modern GPS satellites use two carrier frequencies: the L1 frequency at 1575.42 MHz and the L2 at 1227.6 MHz which lie in Radio Astronomy bands. Specific bands such as the 37.5–38.25 MHz, the 322.1–328.6 MHz, the 406.1–410 MHz and three others are all allocated to be shared between RA, mobile and other services [1]. Nearly every other frequency band allocated for RA is shared with other services [1]. This motivates the need for radio frequency interference (RFI) mitigation in modern telescopes.

At the time of this publication, not many radio telescopes in the world do adaptive RFI mitigation with array signal processing. The current convention in RA is to write off the corrupted frequency channels as useless, with a practice called "flagging." Some say that RFI mitigation is unreliable and others simply don't appreciate its contribution.

Flagging does not solve the problem which will only get worse as time goes on. By mitigating the effect of RFI on the incoming data, many corrupted channels can be recovered and used again. Good algorithms have been developed by the signal processing community that not only demonstrate that RFI mitigation is possible but that it dramatically improves telescope performance. The following section is a survey of various publications that have shown RFI mitigation is possible and/or has been demonstrated to some extent on or for different telescopes around the world as well as a signal processing technique itself independent of RA.

1.2 A Survey of Literature on RFI Mitigation for Radio Astronomy

The work presented in this thesis uses a subspace projection method to do RFI mitigation (see Chapter 2). This was first introduced for array signal processing in radio astronomy by Leshem and van der Veen [2] in 2000. They explored spatial signal estimation with calibrated receivers in 2001 [3]. Projection techniques in RA were further explored in 2004 [4]. They also proved that many different spatial filtering techniques for RFI, in addition to subspace projection, could improve synthesis imaging techniques [5].

Brigham Young University has a rich history of RFI mitigation with antenna arrays for radio astronomical telescopes. Simulations and proposals for subspace projection using auxiliary antennas on the VLA in Socorro, New Mexico were explored in [6] and [7]. RFI mitigation with a phased array feed (PAF) using a single reflector was explored in [8]. Bias correction in subspace projection techniques for power spectral density estimation was presented in [9]. Experimental results from 2007 prove successful RFI mitigation is possible on a 19 element array feed [10]. This provides hard evidence that RFI mitigation actually works and is practical. Since 2003, BYU and NRAO have worked in collaboration to design the phased array feed and digital signal processing backend of the Greenbank telescope in West Virginia. In [11] RFI mitigation techniques including bias correction were proposed with the NRAO observatory as the target platform. Subspace projection was also introduced for the NRAO platform in [12]. Moving RFI, real-time updating of beamformer weights and "spectral scooping" with narrow band interference is explored in [13]. Subspace projection was explored as it would apply to the LOFAR (Low Frequency Array) in the Netherlands [14]. A survey of many RFI mitigation techniques, limitations and methods for deeper nulls and less main lobe distortion was presented in [15]. The interesting case of RFI mitigation in

low interference to noise ratio (INR) environments, and possible solutions, was presented in [16]. In 2014 [17] experimental results finally proved that the FLAG system on the NRAO telescope was functional making it a potential platform for RFI cancelling beamforming. In 2015 a collaboration was done on a multi-tier RFI mitigation system with Aaron Chippendale and Gregory Hellbourg [18].

Gregory Hellbourg has particularly explored subspace projection for radio astronomical observation. He proposed a unique subspace projection algorithm and technique called "Oblique Projection" [19] in 2012. He presents three new techniques on detecting the RFI spatial signature in [20]. He did some performance analysis on pre and post correlation data with subspace projectors [21], data from the LOFAR radio telescope and EMBRACE (Electronic MultiBeam Radio Astronomy ConcEpt) was used in the analysis. (EMBRACE is a 20,000 element, multi beam radio astronomical phased array demonstrator [22].) He, Aaron Chippendale and Brian D. Jeffs from BYU came together to develop a publication [23] on subspace tracking using a reference antenna, statistical performance was also presented [24]. To reduce subspace smearing and improve RFI mitigation Hellbourg presented solutions for estimating the subspace spanned by the RFI within the complexity of a phased array radio telescope [25]. He presents the corrupted array radio telescope model here [26]. Using the Cramer-Rao Bound to measuere the power estimation error variance on an SOI in the presence of RFI, the quality of the estimation of the spacial signature vector under different calibration quality levels from perfect to sub-perfect calibration was determined [27]. A non-linear technique that can work for RFI sources whose spatial signatures are very similar to that of the SOI was presented in [28]. He did some direct experiments on the ASKAP beta array [29]. on the Australian Square Kilometre Array Pathfinder (ASKAP) array located 800 km north east of Perth, Australia. Both Gregory Hellbourg and Aaron Chippendale presented the results of a first attempt to mitigate RFI in real-time on the ASKAP array in [30].

Other work worthy of note has been done to address RFI mitigation in radio astronomy. John M. Ford and Kaushal D. Buch present a survey of regulatory methods and technical methods for RFI mitigation techniques [31]. Also, a survey of all efforts to reduce RFI around the NRAO telescope is presented in [32].

As has been shown, significant effort has gone into RFI mitigation for radio astronomy because there is such a need for good observation. Clearly subspace projection is a widely studied

and successfully implemented method. The present work seeks to further the science by presenting the development of a subspace projection RFI mitigation array processing algorithm on a GPU for real-time adaptive filtering.

1.3 Problem Statement

On the BYU Focal L-band Array (FLAG) system housed at the NRAO observatory in Green Bank West Virginia there is a need for a real-time RFI mitigating beamformer system. As shown above, many subspace projection solutions have been explored and simulated. The problem with implementing RFI mitigation using subspace projection is that large volumes of time samples of data arrays need to be processed to get a good characterization of the subspace. Getting enough hard drive space to do the whole job, post observation, is not just expensive, it's not really feasible for large antenna arrays with large bandwidths.

The FLAG system does provide unique data storing features not found on other RA solutions. While it does store an averaged sum over a long term integration (LTI) window of the spectra observed coming out of the beamformer, it can also store short term integrated (STI) array covariance matrices in its HI fine spectral mode. By doing this the stored covarience matrices can be used for post-processing RFI nulling. However, in FLAG pulsar or FRB mode, and for all other known phased arrays in RA, this is not possible.

A solution for the FLAG pulsar and FRB modes, as well as all other arrays in the wider RA community, is to do the processing "on the fly" in real-time. That is, to take the data chunk by chunk as it flows through the system and do rapid subspace computation and RFI mitigation on the data as it arrives and then pass it along to the integrator having had the RFI component in the data filtered out. The reason why this has not yet been implemented is that there have not been strong enough parallel processing solutions developed that can keep up with the high data rate. This thesis presents a solution that will perform subspace computation at the required data rate.

1.4 Thesis Contribution

The present work documents the development of a real-time RFI mitigating beamformer doing subspace projection. It is realized on Graphical Processing Units (GPUs) which have power-

ful parallel processing capabilities that can handle large data rates. To the knowledge of the author, no real-time subspace projected RFI mitigation solution has been implemented on any radio astronomical telescope that can keep a 13ms per frequency channel data update cycle. This contribution represents a big performance achievement for RFI mitigation in the radio astronomy community.

1.5 Outline

The thesis is laid out as follows:

- Chapter 2: An introduction to the theory of array signal processing for beamforming and subspace projection for RFI mitigation.
- Chapter 3: The design of the system itself. First, the chapter will introduce the FLAG system and an overview of the system will be presented. The discussion will then lead into the tools used to make the system. Finally, the RFI mitigating beamformer will be introduced and described part by part.
- Chapter 4: Testing and verification of the full system will be presented. Here evidence that the system will be real-time, once finalized, is given. Successful RFI mitigation is then demonstrated with medium fidelity simulated data generated and passed through the filter showing the RFI component removed at the beamformer output.
- Chapter 5: The final steps needed to complete the filter will be presented here. Finally, the discussion ends with the future applications the beamformer will have in the radio astronomy and communications communities.

CHAPTER 2. THEORETICAL BACKGROUND

2.1 Introduction

To offer the reader perspective, this chapter will provide a brief introduction to antenna arrays and beamforming theory. First, the signal model for antenna arrays will be derived from the uniform line array approach. A discussion on beamforming with antenna arrays will then be explored. Finally, the RFI mitigation model will be introduced.

This should provide sufficient background for the system presented herein. If more detail is desired, Van Veen and Buckley [33] provide a useful overview tutorial to spatial filtering and beamforming. For a broader treatment see Hayes [34] and Van Trees [35], which provide great presentations on contemporary digital systems for stochastic signals, parametric modeling and adaptive filtering.

2.2 Signal Model for a Uniform Line Array

Consider a plane wave coming from a source at a specified angle Ω_s with respect to a uniform line array (ULA) of antennas as illustrated in Figure 2.1. (For simplicity of presentation, we use a ULA array as an example. The actual PAF arrays addressed in this work have a variety of regularly space array elements.) The line coming from the right represents the plane wave. Note that the antennas each receive the signal but at successive time delays (and thus frequency dependent phase shifts) from each other. Now we consider a single narrowband frequency channel and adopt the narrowband beamforming model [33]. Broadband signals are represented as a concatenation of a series of independent narrowband beamformers spanning a range of frequencies. If the signal arriving on this plane wave is unit amplitude being captured by the ULA, then let the recorded complex base banded voltage levels read at each antenna element due to the source be denoted as

$$\mathbf{a} = [a_0, a_1, ..., a_{L-1}]^T, \tag{2.1}$$

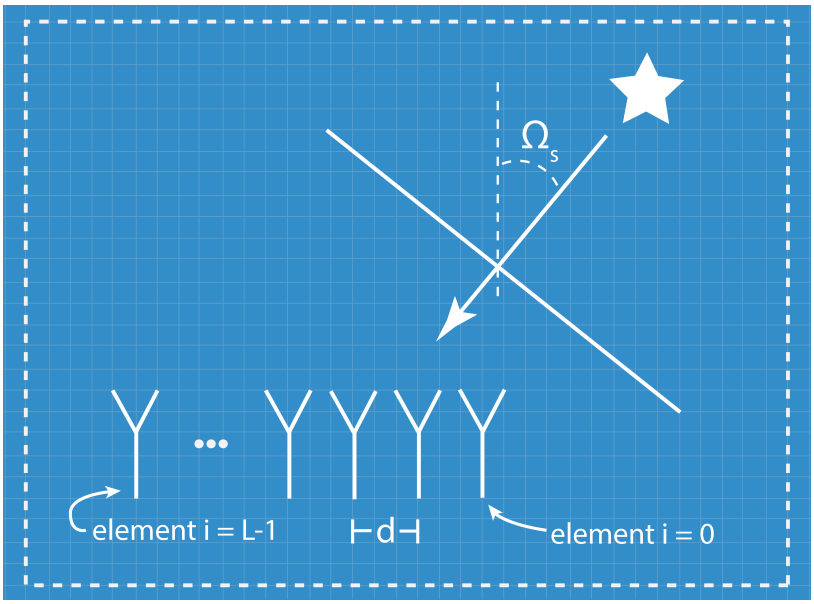

Figure 2.1: A plane wave arrives from a far-field source to a uniform line array (ULA)

where **a** denotes a vector of voltages and a_0 is the voltage recorded at antenna zero and a_1 antenna one and so on for L antenna elements. For a uniform line array with spatially isotropic element responses, the voltage seen from a unit amplitude plane wave can be modeled as

$$a_i = e^{j2\pi\frac{f}{c}id\sin(\Omega_s)}, \tag{2.2}$$

where d represents the distance between adjacent antenna elements. Let f be the center frequency of the narrow-band channel under consideration and c be the propagation speed of electromagnetic radiation. Equation (2.2) tells us that the complex base banded voltage seen at antenna element i has its unique phase offset from the $i = 0$ element. Call **a** the *array response vector* to a signal arriving from angle Ω_s.

Now let $s[n]$ denote a time sampled sequence corresponding to the signal waveform that was propagating as a plane wave and arrived at the antenna array. Also, let $\mathbf{n}[n]$ represent the additive noise seen by the array including the combined resulting noise response vector due to spillover noise, sky noise, thermal noise and receiver noise [11]. The signal model at the array is now

$$\mathbf{x}[n] = \mathbf{a}s[n] + \mathbf{n}[n]. \tag{2.3}$$

The above signal model is a random (stochastic) process since the signal arriving is noise-like for radio astronomy sources, and $\mathbf{x}[n]$ also includes a random noise component. The noise is zero mean with variance σ^2. The array correlation matrix is estimated over a set of M time samples as

$$\mathbf{R}_x = \frac{1}{M}\sum_{n=0}^{M-1}\mathbf{x}[n]\mathbf{x}^H[n] \approx E\{\mathbf{x}[n]\mathbf{x}^H[n]\}, \tag{2.4}$$

where the "H" denotes the conjugate vector transpose operator known as the Hermitian transpose and "$E\{\cdot\}$" is probabilistic expectation. It should be noted that equation (2.4) is valid only because $\mathbf{x}[n]$ is variance ergodic.

The final piece of the model is to include the contribution RFI adds to it. Assuming that there are Q interferers arriving at the array, the complete signal modal is

$$\mathbf{x}[n] = \mathbf{a}s[n] + \sum_{q=1}^{Q}\mathbf{v}_q b_q[n] + \mathbf{n}[n], \tag{2.5}$$

with $\mathbf{v}_q$ being the array response vector for the direction to the qth RFI source and $b_q[n]$ being the interferer's signal sequence.

2.3 Beamforming

One of the primary benefits of an antenna array is the ability to do beamforming. Simply put, a beamformer is a spatially selective filter. Just as frequency selective filters can "select" frequency channels by amplifying desired frequencies and attenuating unwanted frequencies, a beamformer can select incoming plane waves based on angle of arrival.

An antenna array with a digital back-end can leverage modern signal processing techniques to create specific beam patterns. Signal processing theory for beamformers builds directly upon well developed frequency selective theory, thus allowing the rich literature of digital filtering to be applied to antenna receivers. This connection is a signature innovation for both the antenna as well as the signal processing communities.

A beamformer would be called a "band pass" by traditional filtering verbage but operates in the spatial domain (i.e. direction of arrival) and not the frequency domain. The beamformer's "pass-band" (called it's main lobe) is pointed in the direction (Ω_s) of the signal of interest. The

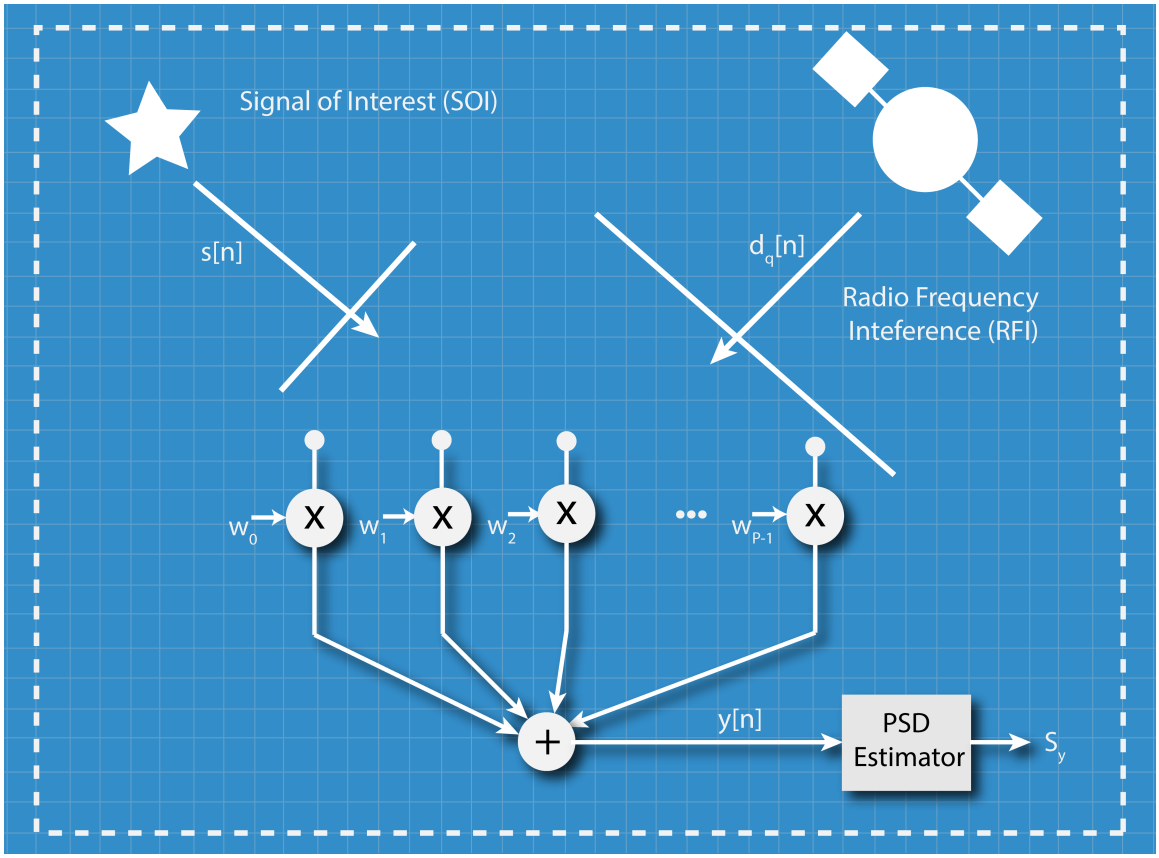

Figure 2.2: A basic diagram of a Beamformer. Plane waves arrive from SOI and RFI to the array. Each antenna element is weighted by the elements of the weight vector. The result is a summed and sent through a PSD estimator where the spectrum is then observed.

computed beamformer output is given by

$$y[n] = \mathbf{w}^H \mathbf{x}[n], \tag{2.6}$$

with $\mathbf{w}$ being an array of complex valued beamformer weights and $\mathbf{x}[n]$ being the received array signal vector, with RFI, as explained in equation (2.5). See Figure 2.2 for a visual representation of the beamformer filter. A typical application would then pass the beamformer output through a PSD estimator and the spectrum $\mathbf{S}_y$ is then sent to the user.

If the reader is familiar with traditional discrete-time signal processing DSP, the filter weights can be thought of as a set of FIR filter coefficients excepting that they apply to the spatial domain rather than the frequency domain.

Appropriate selection of these beamformer weights is a whole field unto itself. Statistically optimal beamforming, adaptive beamforming and other beamforming techniques are discussed in van Veen and Buckley [33]. Please refer to [33] and [35] for a more developed discussion. For purpose of this thesis, it is assumed that a maximum signal to noise (maxSNR) weight vector is used prior to application of an RFI mitigation algorithm. The maximum SNR weight vector is one statistically optimal choice described by van Veen and Buckley that will work well in an RA application. For a point source SOI,

$$\mathbf{w}_{\text{maxSNR}} = \hat{\mathbf{R}}_n^{-1}\mathbf{a}, \tag{2.7}$$

where $\hat{\mathbf{R}}_n$ is the sample estimate for the noise covariance matrix.

Though RFI mitigation can be done in a variety of ways, the system presented herein uses the subspace projection method. The next section will show how subspace projection can take the pre-computed maxSNR beamformer weights and project them into a space where the RFI is mitigated. The projected weights remove the RFI component from the signal. This manifests itself in the beam pattern with a null (a trough) placed in the direction of the RFI.

2.4 Subspace Projection Beamforming

The objective of the RFI filter is to remove the RFI component of the signal while maintaining as much integrity of the SOI as possible. Received signals are put through a correlator which takes M time samples to form the mth short term integration (STI) window resulting in a correlation matrix

$$\mathbf{R}_{x,m} = \frac{1}{M}\sum_{n=mM}^{(m+1)M-1}\mathbf{x}[n]\mathbf{x}[n]^H. \tag{2.8}$$

The span of the columns of $\mathbf{R}_x$ (its range or column space) constitutes an estimate of the signal space S of $\mathbf{x}$,

$$\Re(\mathbf{R}_x) = \text{span}([\mathbf{x}[mM],\ldots,\mathbf{x}[(m+1)M-1]) = S, \tag{2.9}$$

where $\Re(\cdot)$ denotes the range space. There are three, not necessarily orthogonal, subspaces that constitute the signal space:

$$\text{span}(\mathbf{a}) = O, \tag{2.10}$$

$$\text{span}([\mathbf{v}_1, \ldots, \mathbf{v}_Q]) = V, \tag{2.11}$$

$$\text{span}(\mathbf{n}) = N = \text{span}(\mathbf{x}[n]) = S. \tag{2.12}$$

So the task of the subspace projection filter is to accomplish the projection of the beamformer weights into a subspace W that is orthogonal (or as near to orthogonal as possible) to the RFI space V,

$$W \perp V, \tag{2.13}$$

$$W \subset S. \tag{2.14}$$

To produce a projection operator into the null space of the RFI, $P : S \rightarrow W$, an eigenvector decomposition is required:

$$\mathbf{R}_x = \mathrm{U}\Lambda\mathrm{U}^{\mathrm{H}}, \tag{2.15}$$

where the columns of U constitute the eigenvectors of $\mathbf{R}_x$ and $\Re(\mathrm{U})$ spans S. If $Q = 1$ and the RFI interference to noise ratio (INR) is sufficiently larger then the SNR then the eigenvector corresponding to the max eigenvalue $\mathbf{u}_{\max}$ represents the RFI and the projection operator is formed as follows:

$$\mathrm{P} = \mathrm{I} - \mathbf{u}_{\max}\mathbf{u}_{\max}^{\mathrm{H}}, \tag{2.16}$$

where I is the identity matrix whose column space spans the full $\Omega = \mathbb{R}^L$ vector space and $\mathbf{u}_{\max}\mathbf{u}_{\max}^{\mathrm{H}}$ spans V. This means that

$$\Re(P) = \Omega - V = W, \tag{2.17}$$

and the projection operator is $P : S \rightarrow W$. Which will project the previously computed beamformer weights into the null space of the RFI

$$\mathrm{P}\mathbf{w} = \mathbf{w}_{\text{sp}}. \tag{2.18}$$

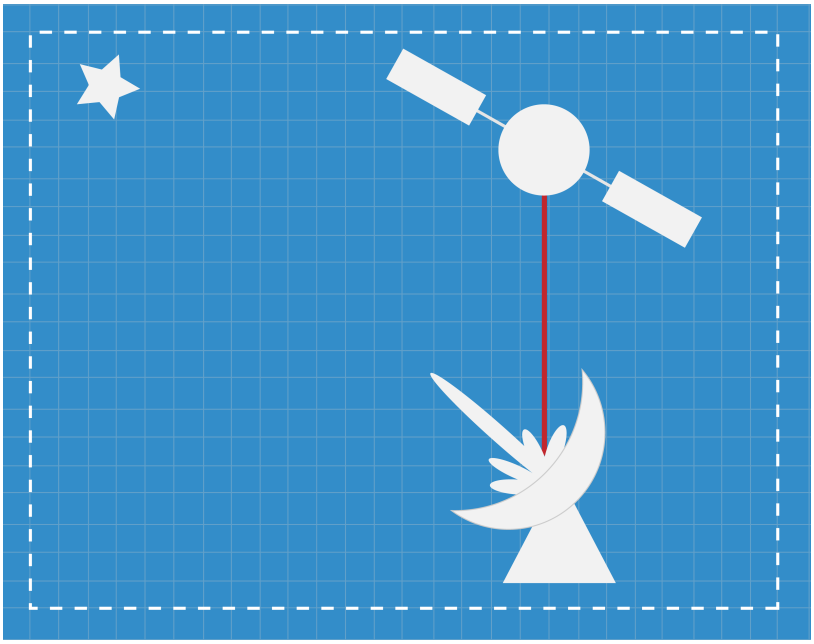

Figure 2.3: When the beamformer weights are projected into the RFI-mitigated subspace a null is placed in the direction of the RFI.

If the projection is successful, then the resulting beam-pattern produced by $\mathbf{w}_{sp}$ will place a null in the direction of the RFI source while maintaining the integrity of the main lobe in the beam pattern. By nulling out the RFI component of the signal the SOI can be observed far more clearly and the corruption is mitigated. The concept is depicted in Figure 2.3.

All of the parts of the real-time RFI mitigating beamformer will accomplish various parts of the above linear algebra in a GPU. The system design and layout is the subject of the next chapter.

CHAPTER 3. AN APPROACH FOR FILTERING RFI IN REAL-TIME ON THE GREENBANK TELESCOPE

3.1 Implementing a Real-Time RFI Mitigation System

As addressed in Chapter 1, RFI canceling has been explored and proven to be a viable solution in radio astronomical observation. Though it has not yet been widely adopted by astronomers, it has the capability of reducing the corrupting effect of RFI to a level which will allow the data to be used again. An implemented real-time prototype is a strong indicator that this technology can further the science.

This chapter presents the development of a real-time RFI cancelling beamformer implemented on a graphical processing unit (GPU). This is a prototype filter that will fulfill the real-time filtering constraints of a world class broadband beamforming phased-array-feed (PAF) equipped radio astronomical telescope in West Virginia at the Green Bank Radio Observatory. The prototype demonstrates critical functions and components of a real-time RFI mitigation system. The most closely related demonstration project was reported by Aaron Chippendale for an experiment with the ASCAP phased array feed mounted on the Parks telescope in Australia [30]. To the author's knowledge, however, no real-time system has met the tight real-time constraints that this system achieves.

3.2 Overview of the FLAG system on the Greenbank Telescope

The proposed RFI mitigation process is meant to add adaptive spatial filtering capabilities to the beamformer backend of the Focal L-band Array for the Greenbank Telescope (FLAG) system. It is a system designed to be on the dish of the NRAO Observatory in Green Bank, West Virginia. This state-of-the-art beamformer is fully equipped to observe HI radiation as well as radio transients [12] with its complete 150-MHz Bandwidth 38-element L band phased array feed (PAF) analog receiver and digital processor. FLAG consists of two parts: an analog receiver (front-

Figure 3.1: The Greenbank Telescope at the Greenbank Observatory (GBO), West Virginia.

end) and a digital signal processing (DSP) system (back-end). Figure 3.2 presents a general block diagram of the overall FLAG system. What follows is a brief description of the FLAG front-end followed by a more thorough investigation of the back-end where the beamforming and RFI mitigation will occur.

3.2.1 FLAG System Architecture

There is a phased array of 19 dual-polarized antenna elements connected to cryogenically cooled low noise amplifiers who's signals are then routed through and IQ down mixer and analog to digital converter (ADC). The signals are transported down from the telescope via a fiber optic digital down link (DDL) to the "F-engine" which uses a polyphase filter bank to "frequency channelize" the antenna data. The F-engine is implemented on an FPGA system from UC Berkeley called the ROACH II. The output from five ROACH II boards is a total of 500 frequency channels, each 303kHz wide, which are routed to five high powered computers (HPCs). Each HPC contains two nVidia GTX980 graphical processing units (GPUs), for a total of $5 \times 2 = 10$ GPUs in the whole system. These HPCs and GPUs house the heart of the DSP system where beamforming and array correlation processes are applied to the signals captured by the array.

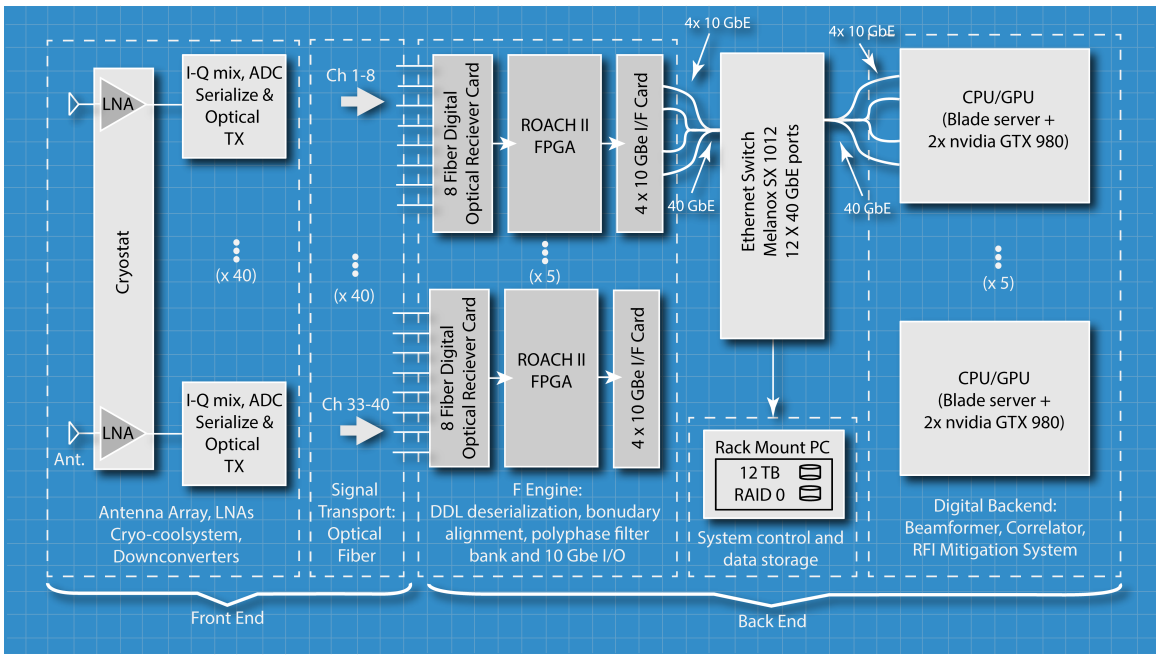

Figure 3.2: FLAG System Overview. The part on the left constitutes front end and the part on the right is considered the back end. Data flows from front to back.

First the CPU handles the Input/Output and data management which transfers the data, now frequency channeled from the front end, to one of the two GPUs in each HPC. Data is processed in blocks, with real-time handling through the GPUs managed using a pipe-lining scheduler called HASHPIPE (see section 3.2.4 for more detail regarding HASHPIPE). Two HASHPIPE instances are implemented in each GPU. There are two GPUs on each HPC so a total of 4 HASHPIPE instances reside on each HPC. Thus, $4 \times 5 = 20$ HASHPIPE instances work in parallel each on a separate selection of 25 frequency channels. In this way the task of signal processing across the whole bandwidth is subdivided.

The system is meant to accomplish different tasks at different times. Thus, it has been designed to run in various modes which can accomplish such jobs as array correlation, real-time beamforming, fine channelization through poly-phase filtering, computation of total received power per frequency channel (i.e. spectrometer), and so on, [36], [37], [17]. The new mode being proposed in this thesis, called "XRFI" Mode ("X" for "remove" or "cross out" and RFI for radio frequency interference), will accomplish the job of spatial filtering to remove RFI in real-time.

3.2.2 A Note on the Frequency Bin Ordering

In all modes, the F-engine's primary job is acquisition and frequency channelization. The beamformer/correlator (called the "BX-Engine") will process these channels, but the order in which the channels arrive there deserves some special note.

Each of the frequency channels are deliberately ordered in such a way that contiguous channels are spread across HPCs (See Figure 3.3). The frequency bins being generated at the ROACHES are routed in such a way that contiguous channels are spread across the HPCs. The task of processing contiguous frequency bins is thus divided evenly among the HPCs, even if the total bandwidth is reduced as in HI observing mode. Each GPU has two instances of the software running on it. Each instance is to receive 25 frequency channels to "divide and conquer" the task of processing the data.

$$\frac{500 \text{ frequency channels}}{5 \text{ HPCs} \times 2 \text{ GPUs} \times 2 \text{ software instances}} = 25 \frac{\text{frequency channels}}{\text{software instance}}$$

The first 5 frequency channels arriving at HPC 1, GPU 1, software instance 1 will be frequency channels 0-4. The next 5 frequency channels will be frequency channels 100-104 and so on. Thus if only the first 5 frequency channels are selected in each software instance then the contiguous channels 0-99 are selected in parallel across the whole array of HPCs. Also note that the 25 frequency channels arriving at a software instance are not contiguous. Putting the data in parallel across the GPUs maximizes throughput. The GPUs themselves will work on the 25 frequency bins handed to them in parallel as well. This parallel (or array) processing on the GPUs is made possible by CUDA.

3.2.3 CUDA

XRFI mode is designed for a CPU with a graphical processing unit (GPU) on it. Array signal processing code modules run on the GPU while the CPU focuses on pipe-lining the data using HASHPIPE. The subspace projection beamforming capabilities are implemented as code modules on the GPU using a programming suite developed by nVidia called CUDA.

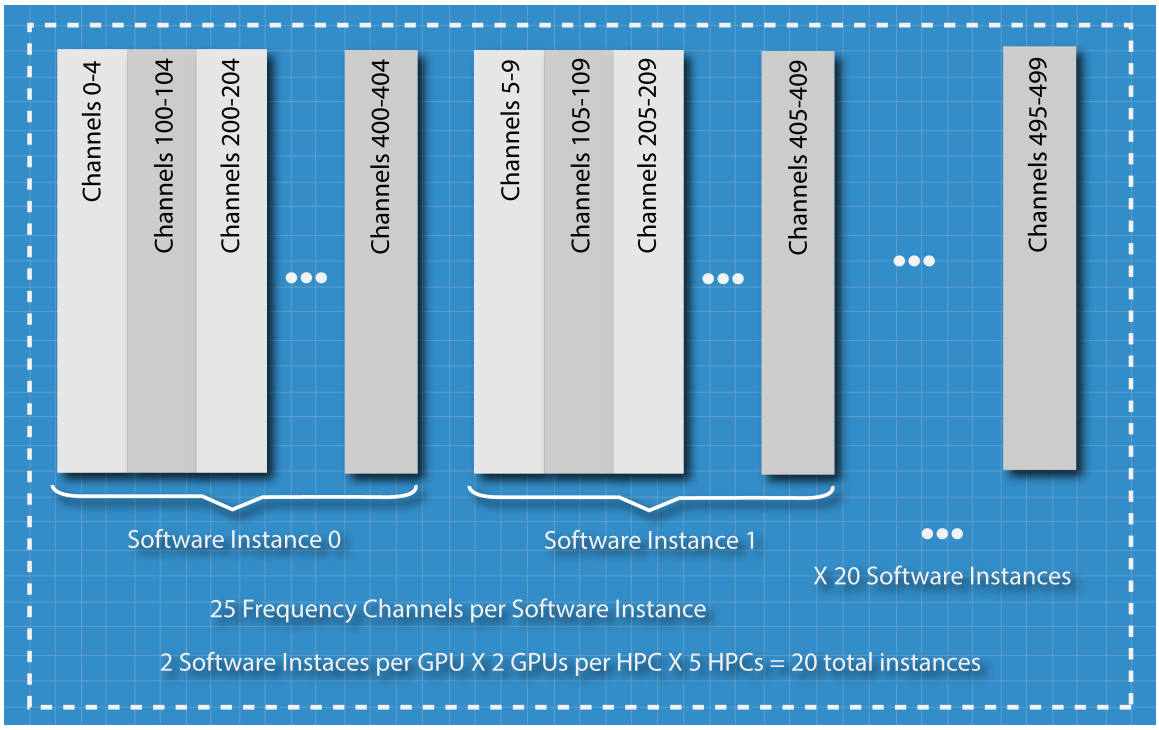

Figure 3.3: The order of the frequency bins as they arrive at the back end. A single software instance handles 25 non contiguous frequency bins. Contiguous bins are spread across software instances.

CUDA is a toolkit based on C/C++ that can be compiled on the host PC. It leverages the GPU's powerful parallel processing muscle for large data computation. Though it was originally developed for faster 2D and 3D graphical rendering, scientists and engineers have begun using it as a powerful solution to large data parallel processing problems. Since modern computer graphics and special effects require fast data rendering, such as the drawing of thousands of polygons in short time periods or advanced ray-tracing capabilities for dynamic real-time lighting, GPUs have been carefully designed and optimized for rapid parallel processing. Such a parallel muscle makes for rapid large array signal processing, which is why it was selected to satisfy the needs of a high bandwidth phased array radio telescope solution.

3.2.4 HASHPIPE

Real-time data throughput to the systems running on each GPU is accomplished on the CPU via an advanced pipe-lining scheduler. The system, developed by David MacMahon of the

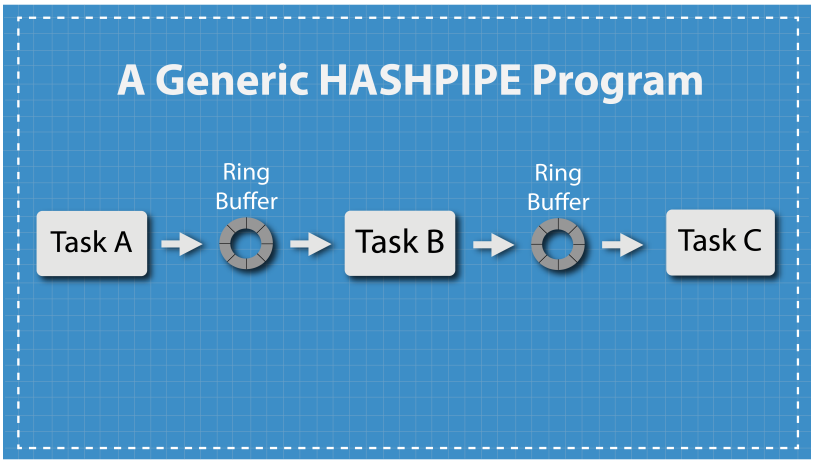

Figure 3.4: A generic HASHPIPE process pipeline

UC Berkley CASPER group specifically for radio astronomical observation applications, is called HASHPIPE.

HASHPIPE breaks tasks into threads that manage various steps in the signal process as a pipeline. Figure 3.2 shows a generic abstraction of a HASHPIPE process. A task is set up as a library (called a "plugin") that provides a series of threads that execute each process that operates on the signal. The tasks are then connected together in a pipeline by buffers that manage the data-flow.

Buffers run between each task allowing the data to flow in to one process and out another with semaphore controlled input and output to each stage. Block by block data is buffered and sent to the next task. This allows each process to run independently of the others and keeps data flowing down the pipeline. As the data flows into each plugin, it is the plugin's job to copy the data to the GPU for processing.

Within a thread a copy of the data is sent from CPU memory to GPU memory. The data is processed using the sophisticated parallel power of the GPU and then it is returned to the CPU when it's done. Thus the CPU spends most of its computational resources on moving the data around and the GPU is the powerful parallel computing system that does the signal processing itself.

Using HASHPIPE we can link up the various systems, processes, and mechanisms that make up our design. Different configurations of threads can accomplish different tasks. On FLAG, a particular HASHPIPE configuration is called a "mode".

Each mode on flag uses a certain HASHPIPE pipeline setup with its own plugins to accomplish a particular job. For example, the proposed "RTBF" mode accomplishes real-time beamforming using the net, transpose and beamforming threads. Another example is the "Fine PFB" mode, which does fine channelization using a poly-phase filter bank. RFI mitigation can also be done using HASHPIPE. A mode that will accomplish the real-time RFI mitigation, called the "XRFI" mode, will be added to HASHPIPE to be the RFI mitigation mode.

The HASHPIPE plugin to be developed for XRFI mode needs a payload that does the actual filtering work. Thus an XRFI "filter" needed to be developed first. In the first prototype implementation presented in this thesis, this filter works "stand alone" and receives a single data block as if executing one step on the HASHPIPE pipeline. If the filter can process one block successfully, tested multiple times one block at a time, then it can be integrated into a plugin that will automate the continuous input of blocks.

This thesis documents the XRFI filter in stand alone mode, prior to its integration into HASHPIPE. Due to time constraints, the XRFI filter has not yet been integrated into the HASHPIPE platform. It has been demonstrated, however, that the filter will accept a single block of data, as if from one of the data buffers on the HASHPIPE process line, and correctly and quickly processes the result. The following section will describe the system's design and functionality. Chapter four will demonstrate that the system works and will function in real-time, meaning it will be able to "keep up" with real-time processing time constraints, once it is integrated into the pipeline.

3.3 XRFI Stand Alone Filter Design

The XRFI filter consists of four parts: the correlator, the beamformer, the subspace computer, and the weight projector. Work is done on one data block at a time. The data block will be described first and then each part will be detailed.

3.3.1 A Typical Data Block in FLAG

The signal data arriving at the array follows Equation (2.5) with each vector being of length 40, for the 38 antenna elements plus two spares of the FLAG PAF. This signal is passed through

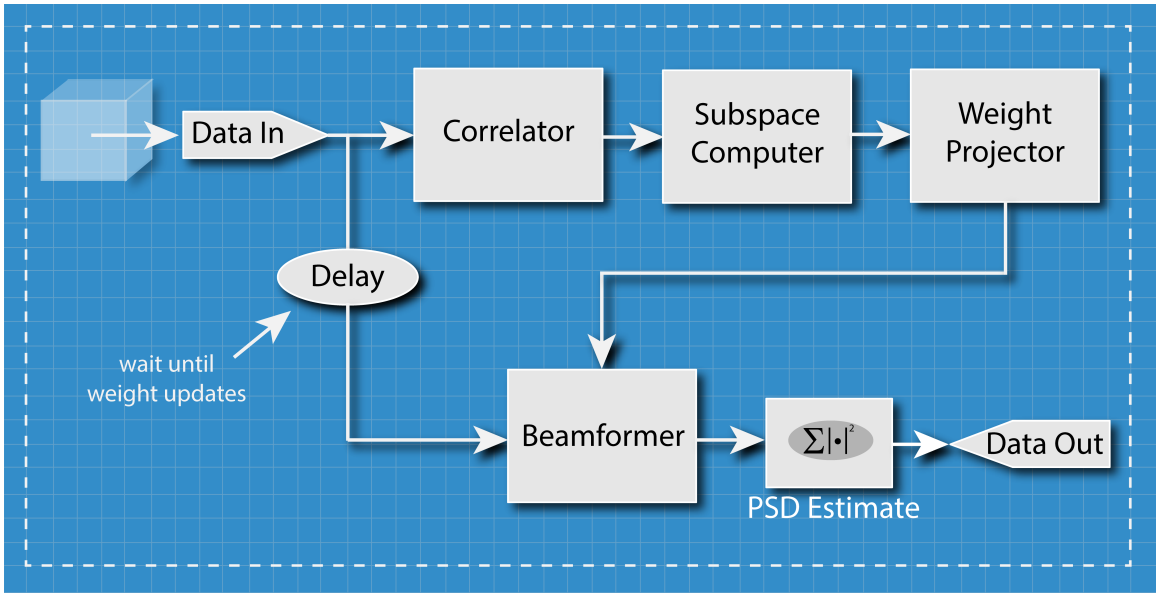

Figure 3.5: Block diagram of XRFI subspace projection beamformer in stand alone mode executing a single block of data

the FLAG system as described in the above sections and is split up into 500 frequency channel by the F-engine's FPGA polyphase filter bank (PFB). Each channel spans 303kHz. As described above, the data are shared among 5 HPCs. After all the frequency channels are routed, each HASHPIPE instance handles 25 frequency channels. As described in Equation (2.8), the signal must be captured in batches of time samples to form an STI. We choose for the RFI filter to have and STI of length 4000, which matches the HASHPIPE data block size used for FLAG.

All of this results in a data block that is 40x25x4000, 40 antenna elements, 25 frequency channels, and 4000 time samples. Since some operations, such as correlation, are optimized to perform most efficiently for data lengths which are powers of two, a typical data block is zero padded to create an extended block of size 64x25x4000 where the remaining unused antenna slots can be zeroed out (see Figure 3.6).

HASHPIPE pushes through the data one block at a time. The work is done in the GPU on one block and then it moves on to the next block. As mentioned before, the proposed XRFI system has been tested on one block and has resulted in the correct output. See the following chapter for results.

Prior to running through the correlator, the data passes through a "transpose" or "corner turn" step. The primary purpose of the transpose thread is to align the data appropriately to be

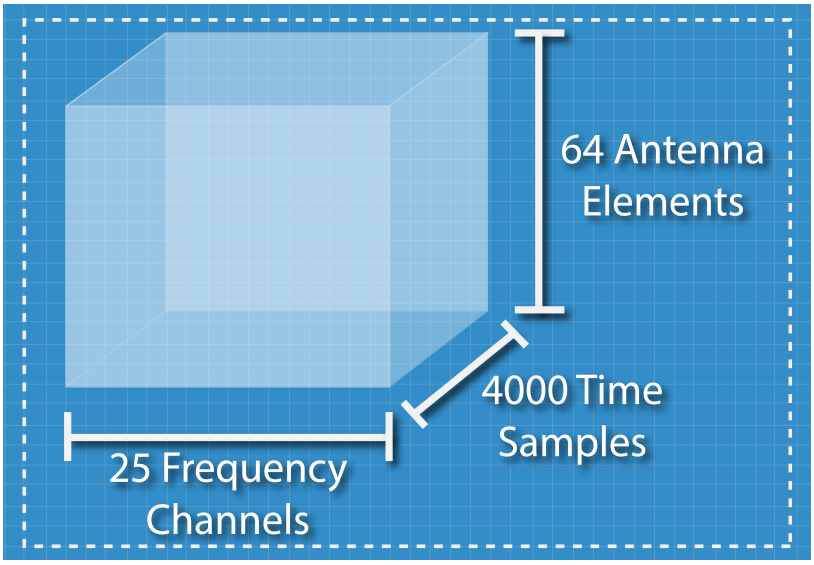

Figure 3.6: A Typical Data Block.

processed by the correlator thread in the GPU. A corner turn is done to ensure that the slowest moving index is the frequency index. After this is done data moves on to the correlator.

3.3.2 Correlator

The job of the correlator is to do as was shown in equation (2.8) and produce the $\mathbf{R}_x$ matrix

$$\mathbf{R}_{x,k,m} = \frac{1}{M} \sum_{n=mM}^{(m+1)M-1} \mathbf{x}_k[n]\mathbf{x}_k[n]^H = \frac{1}{M}\mathbf{X}\mathbf{X}^{\mathrm{H}}, \tag{3.1}$$

for the kth frequency bin, where $\mathbf{X} = \left[\mathbf{x}[1], \mathbf{x}[2], \ldots, \mathbf{x}[M]\right]$ represents a data matrix of 64 elements by $M = 4000$ time samples. One correlation matrix is computed for each of the 25 frequency channels. The current filter computes one block ($m = 1$. Once implemented in HASHPIPE, this 4000 time sample block will be computed once every system tick (13 millisecond windows).

The correlation is performed for 64 antenna elements (24 of them zero padded) across 25 frequency bins for 4000 time samples. Were this done in C, it would take three "`for`" loops for a total of 6,400,000 sequential mathematical operations. In pseudo-code the operations can be described as:

```
for (int freq=0; freq<NUM_FREQ_BINS; freq++) {
    for (int time=0; time<NUM_TIME_SAMPLES; time++) {
        for(int element=0: element<NUM_ELEMENTS; element++){
            ...
            x[freq][time][element]*x_her[freq][time][element];
            ...
        }
    }
}
```

But in CUDA, on a GPU all 6,400,000 operations are done in parallel. One step. Very efficient. In pseudo-code:

```
...
idx = get_index(freq, time, element);
x[idx]*x_her[idx];
...
```

and the GPU knows where each element is and causes each part to multiply itself by the proper neighbor and sums it using a reduction algorithm. The actual implementation of this correlator uses a linear algebra operator to do the $\mathbf{X}\mathbf{X}^{\mathrm{H}}$ operation and under the hood it leverages the parallelism described above.

3.3.3 Beamformer

The job of the beamformer is to compute the output,

$$y_k[n] = \mathbf{w}_k^{\mathrm{H}} \mathbf{x}_k[n], \tag{3.2}$$

for the kth frequency bin. This output represents the spatially filtered voltage time series result coming out of the beamformer. See Figure 2.2 for a visualization. The output $y_k[n]$ is then put through a power spectral density (PSD) estimator which is just a simple summing reduction to provide the power seen coming out of the beamformer for a given beamforming angle. The beamformer behaves exactly as described in Chapter 2 with data flowing in and an inner product between the weight vector and the data results in an output. The key to the RFI filtering is to project the

beamformer weights into a subspace where there is little or no RFI. When this is done the beamformer output will contain predominantly the signal and noise components with little if any of the RFI component.

3.3.4 The Subspace Computer

An autocorrelation matrix from the correlator of Equation (3.1) is fed into the subspace computer. The subspace computer computes the vector space that spans the RFI, and forms a projection matrix to map vectors onto the null space of the RFI. As described in Chapter 2, the space spanned by the RFI is found by doing an eigenvector decomposition on the full rank matrix **R** (due to the noise space) as in equation (2.15):

$$\mathbf{R}_{x,k,m} = \mathrm{U}\Lambda\mathrm{U}^{\mathrm{H}}. \tag{3.3}$$

This is done using a CUDA "cuSolver" function called `cusolverDnCheevd()` which can compute an eigenvector decomposition for dense complex-valued matrices. The subspace computer resides on the GPU.

Outputs from the correlator are saved in the GPU's memory, not the CPU memory, and are used in computing the eigenvectors. No part of the matrix is ever removed from the GPU and sent back to the CPU. Keeping all data on the GPU and doing a CUDA eigenvector decomposition makes the process significantly more efficient than moving the data back to the CPU and computing it there only to have to transport it back to the GPU. The triumph of accomplishing such a complex linear algebra operation on the GPU serves as the "special sauce" of the XRFI filter. This is a significant performance improvement and contribution to RFI mitigation in radio astronomy.

After an eigenvector decomposition is done, the projection matrix P_k for the kth frequency channel is computed by finding the subspace as explained in Chapter 2:

$$\mathrm{P}_k = \mathrm{I} - \mathbf{u}_k\mathbf{u}_k^H, \tag{3.4}$$

where I is the identity matrix and $\mathbf{u}_k$ is the eigenvector corresponding to the largest eigenvalue for the kth frequency bin.

To meet timing constraints, the filter is designed to compute the subspace projection matrix for only one frequency bin per data block. The filter is set up such that k will increment by one, from 1 to 25, each time a new data block arrives. For each new block from the HASHPIPE data buffer, a new subspace is computed once for a single frequency channel. Details of this time multiplexed frequency channel solution will be more thoroughly explained in Chapter 5. The subspace computer finds the subspace and computes a projection matrix into the subspace. Once computations are complete it sends the newly computed projection matrix off to the weight projector where the weights are projected into the null space of the RFI.

3.3.5 The Weight Projector

The job of the weight projector of Figure 3.5 is to project the weights into the null space of the RFI

$$\mathbf{w}_{sp} = \mathrm{P}_k \mathbf{w}_{\mathrm{maxSNR}},$$

where $\mathbf{w}_{\mathrm{maxSNR}}$ are the pre-computed max SNR weights already loaded onto the beamformer during initialization and $\mathbf{w}_{\mathrm{sp}}$ are the weights projected into the subspace. The projection matrix operator that spans the subspace, P_k comes from the subspace computer. Once the weights are projected into the subspace they are ready to be placed back into the beamformer where it will compute the spatially filtered output removing the RFI component from the incoming signal.

3.3.6 Closing the Loop

At the end of block processing by the weight projector, the newly updated weights must be fed back into the beamformer and are then applied to the same raw sample vectors $\mathbf{x}_k[n]$ which went through the correlator. In Figure 3.5 it can be seen that, in the stand alone case, the very same data that is being used to compute the subspace is also being pushed through to the beamformer. A delay needs to be put in place so that the computed subspace corresponds to the correct data set. This design solution is sufficient to run the beamformer in stand alone mode but some revisions will need to be made for it to be integrated into HASHPIPE. See Chapter 5 for more information and a proposal on how this will be addressed.

3.4 Summary

This chapter has focused primarily on the design and development of the real-time RFI mitigation "XRFI" filter on the FLAG system of the Greenbank Telescope in Greenbank, West Virginia. The system as a whole was examined in brief whilst the specific XRFI filter has been explained in detail. While a complete description of the system functionality and its parts demonstrates the research process, it is essential to prove that the system works. The primary focus of the next chapter will be to document the verification procedures and resulting output of the system as a proof of concept for a functional system.

CHAPTER 4. REAL-TIME RFI MITIGATION: MEETING THE MARK

To prove that a system can perform subspace projected beamforming operations, complete with all the constituant functions of beamforming, array correlation and weight calculations running in real time, a few things need to be shown. First, it is necessary to demonstrate that the minimum timing constraints for the beamforming update rate are met by all parts of the system. This includes worst case timing metrics of more computationally intensive routines such as the correlator, beamformer, and subspace projection computer blocks. Secondly, valid test data must be processed through the system to verify RFI cancellation, including evaluating the resulting beam response pattern and constituent signal levels (i.g, SNR and INR) in the beamformed output time series.

This chapter will document the results of various tests that expose different parts of the system showing timing measurements as well as data processing. The objective is to prove concept where proof is required and demonstrate functionality.

4.1 Real-Time Processing Constraints

As with any timing sensitive embedded application, as simple as a small state machine to as complex as an operating system, all implementations must satisfy a maximum timing constraint. If the whole process can meet the timing constraint, dictated by the refresh rate of the system, then the process runs seamlessly and continuously for its intended job. Such processes are called "real-time" processes.

In terms of the RFI filter, how often do the beamformer weights need to be updated? The answer has already been specified [38]. The real-time beamformer has been specified to meet a 13ms time window because the 4000 time samples specified in chapter three represent 13 ms of time. This means that if the RFI filter, which includes the beamformer in its critical path, is to be real-time it must also meet the 13ms timing constraint.

The timing constraint specifies the maximum amount of processing time the four parts of the XRFI filter (correlator, beamformer, subspace computer and weight projector) can take to run and complete their jobs. The 13ms window is a worst case maximum time spec for a single data block (as specified in chapter three) to run through the filter and have a result rendered out. It includes the computation time of the PSD estimate (in Table 4.1 it's included as part of the Beamformer since it is a component of the Beamformer code). It also includes memory copy time between the CPU to the GPU. It does not include initialization needed to set up data buffers, allocate memory and instantiate components of the system. The initialization will all be complete before the data begins to flow. The following section will discuss the resulting timing measurements taken on the function calls and the memory copy time that comprise the data throughput of the XRFI filter.

4.2 Profiling the CUDA Implementation

As part of the CUDA toolkit, nVidia has provided a handy profiling application that can analyze how long each function call and memory copy take. Table 4.1 shows the measured timing analysis as summed up for each implemented part of the proposed RFI cancelling system in GPU clocked time. The table presents an abbreviation of the profiler output for the reader's convenience (For the complete profiler report refer to Appendix A). Note that the analysis shows the beamformer and correlator operating at full bandwidth (i.e. for all 25 frequency channels of a software instance). This is how it has been tested. The numbers included in this analysis for the subspace computer are working on only one frequency channel, NOT all 25. By working on only one frequency channel per GPU data block instance (i.e. each `mcnt` increment using FLAG nomenclature) per time window the time specification can be met.

The reader may ask if only working on one frequency channel per 13ms time window is a practical result. Remember that the 25 frequency channel filter is replicated across all the GPUs on all of the HPCs. Recall from the previous chapter that a total of 20 instances of the filter will be running at the same time: 5 HPCs $\times$ 2 GPUs per HPC $\times$ 2 instances per GPU = 20 instances of the XRFI filter. If each frequency channel is 303 kHz then 20 $\times$ 303 kHz = 6.06 MHz Also recall from the previous chapter that the frequency channels are spread in a stripe like pattern across the 5 HPCs and thus if only frequency channel 1 is running on all 20 instances then 6.06 MHz

Table 4.1: Abbreviated nVidia Profiler Functional Timing Report

Function Name	Time	Freq. Channels	Time % of 13ms
Beamformer + Weight Projector	1.14 ms	25	8.8%
Correlator	1.128 ms	25	8.7%
Subspace Computer	2.573 ms	1	19.8%
Pinned Memory Copy (CPU Host to GPU)	6.26 ms	25	48.2%
Pinned Memory Copy (GPU to CPU Host)	23 μs	25	0.2%
Total (One Frequency Channel):	11.128 ms		85.7%

of contiguous bandwidth is successfully being filtered in real-time as long as this one frequency channel meets the timing window.

Does the filter meet the 13ms timing constraint? The total time for beamforming, correlating, subspace computation and weight update sums to 4.845ms (see Table 4.1). The memory copy takes a grand total of 6.283 ms using nVidia's pinned memory data paradigm. Therefore the whole timing profile for a single block of data to pass from input to output of the filter is 11.128ms per frequency channel. This fits within the 13ms time window with a 1.88ms margin and therefore one instance of the the filter does meet the real-time specification, according to the nVidia profiler.

It is important to note that Table 4.1 shows GPU clocked time, not clock time, for all of the functions. "Clock time" refers to time as reckoned by the rotation of the earth. GPU time refers to the number of clock cycles attached to the task, but does not represent the actual clock time to complete the task. Also, the timing experiments evaluate computational load for a single instance of the beamformer on a single GPU. The FLAG GPU software architecture runs two concurrent instances of the beamformer in each GPU, so we expect that we will effectively need to complete the RFI canceling processing in half of the available 13 ms data block update cycle time. Other unavoidable operations such as context switching, data movement, and other overhead processes would not get counted in this GPU time profile report but they do add to the total time needed to meet the timing constraints of the system.

To address the concern of wall clock and GPU clocked time, another timing experiment was conceived that illustrates more of the story. All of the functions in the XRFI system were run 1000 times and averaged. Table 4.2 shows the results of this experiment. In this particular test, the whole XRFI filter was run with an event clock that reflects more what the wall clock time would be. Notice that the result averages to about 17 ms. This is above the 13 millisecond result. It is

Table 4.2: Wall Time Report

Total Elapsed Time (all functions over 1000 iterations)	16874.28 ms
Number of Iterations:	1000
Average Elapsed Time:	16.87 ms

also true that this will have to work on two instances running on the GPU and they both have to run under the 13 ms time constraint. The above experiments address only one instance being timed. Can XRFI be a real-time system?

One of the reasons the second experiment doesn't meet the 13 ms constraint is that this timing experiment was done with standard CUDA "MemCopy" functions and not nVidia's pinned memory functionality. When the filter is finally implemented with pinned memory, the timing will go down significantly.

Another thing to consider is the 13 ms timing constraint itself. The only reason there is a 13 ms time constraint is because 13 ms worth of samples are gathered per data block. If the block size is increased, then there is more time to process the block. Doubling or tripling the data block size would allow for a window that could be 26 or 39 ms. Furthermore, the MemCopy does not double or triple linearly as the block size increases, it remains relatively fixed. What's the cost? Memory, in the GPU.

The increased memory in the GPU is not a concern because the current implementation only uses a tiny fraction of the memory available. Current and future GPUs will have more than enough memory to allow for doubling or tripling of the data block.

This doubling or tripling of the block solution is also attractive because, with two instances of HASHPIPE running for each GPU, the larger time margin will allow both instances to run cleanly with plenty of extra room for error. Further tradeoff analyses will have to go into studying block sizing as the filter is implemented but the researchers who are developing the final XRFI system are exploring this solution.

The filter is still actively being incorporated into HASHPIPE for continuous real-time operation. By showing that the profiler finds that the system will work in 11 ms, acknowledging the wall clock time and the need to double the work load for two HASHPIPE instances and address-

ing solutions to the MemCopy cost, this thesis proposes that the implemented XRFI filter can be real-time, once complete. See Chapter 5 for further discussion on the final implementation.

4.3 Initial Unit Tests

To verify that the system operates correctly a test model was generated that represented a simulation of a uniform line array (ULA). Once the test data was generated, each step in the system was pre-computed in MATLAB: correlation, eigenvector decomposition, projection matrix computation and weight projection. The data was then quantized and run through the filter and the result compared with the MATLAB simulation. Unit tests were considered a "PASS" when the output data was equal to within a scale factor and quantization range of the simulated result.

4.3.1 The Test Model

Using the signal model as discussed in previous chapters, a simulated incoming data block was generated with a SOI, one RFI source, and noise:

$$\mathbf{x} = \mathbf{C}(\mathbf{a}s[n] + \mathbf{v}d[n] + \mathbf{n}[n]), \tag{4.1}$$

with $\mathbf{n}[n]$ being the noise vector of unit variance, and C will be explained below. The SOI was modeled as a zero mean circularly symmetric complex white Gaussian random process with a signal-to-noise ratio (SNR) of -30dB

$$\mathbf{S} = \mathbf{Y} + j\mathbf{Z} \tag{4.2}$$

$$\mathbf{Y} \sim \mathcal{N}(0, \sigma^2) \tag{4.3}$$

$$\mathbf{Z} \sim \mathcal{N}(0, \sigma^2) \tag{4.4}$$

$$\sigma = 10^{-30\text{dB}/20} \tag{4.5}$$

$$\rho_{yz} = 0 \tag{4.6}$$

and $s[n]$ is a realization of the random process S. RFI sequence $d[n]$ is also a realization of a zero mean circularly symmetric complex white Gaussian random process with and interferer-to-noise

ratio (INR) of +20 dB. Note that this is designed around the assumption that deep space signals -30 dB below the noise floor are common and strong interferers, such as the downlink for a GPS satellite moving overhead, could be as strong as 20 dB over noise levels.

A test data block was generated as described in Section 3.3.1 with 4000 time samples across 25 frequency channels arriving at a simulated uniform line array having 64 elements spaced $\frac{\lambda}{2}$ apart. The data were generated independently across all 25 frequency bins as a broadband signal and broadband interference.

To add additional fidelity to the simulation a mutual coupling model with element-wise complex gain variations was included in C, a Cholesky factorization of the following matrix (for L=64 elements):

$$\mathbf{A}_L = \begin{pmatrix} 1 & \frac{1}{2}^1 & \cdots & \frac{1}{2}^{L-1} \\ \frac{1}{2}^1 & 1 & \cdots & \frac{1}{2}^{L-2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2}^{L-1} & \cdots & \cdots & 1 \end{pmatrix}, \tag{4.7}$$

where $\mathrm{A}_L = CC^H$. When a plane wave arrives at the ULA, the current induced in one antenna element causes the antenna to re-radiate some of the energy to the other elements in the array. More is induced between antennas closer together while less is induced between antennas farther apart. The matrix A models the pairwise mutual coupling between all antennas in the array. The Cholesky factor C can then be multiplied by the data and the result is a relatively high fidelity model of a uniform line array (ULA) receiving a Gaussian signal with interferers overhead.

Using MATLAB to run the model explained above, a 64 element $\times$ 25 frequency bin $\times$ 4000 time sample data block was generated for the GPU. This data block was to be fed directly into the XRFI filter housed on the GPU, but first it needed to be reformatted.

4.3.2 Quantizing the Data

While it is true that the data were fed directly from the MATLAB simulation into the GPU, they need to be modified to match the FLAG data format.

FLAG will only accept the data in (8-bit real, 8-bit imaginary) structures. This means that the complex double precision floating point numbers coming out of MATLAB needs to first be restructured into an interleaved pattern: real, imaginary, real, imaginary. It also means that each data entry needs to be recast from double precision floating point representations to 8 bit integers. This is called the "quantization" step for the simulated data.

In a fashion similar to an ADC, the higher resolution double precision data points were quantized and mapped to a number in the range $[-128 : 127]$ reflecting the range of a signed 8 bit integer. The number -128 corresponding to a lower limit set on the double precision data and the number +127 corresponding to an upper limit placed on the double precision data.

Also, a data scaling value prior to quantization had to be chosen carefully to ensure that the data were preserving at least two significant digits after quantization. If the scale was set too low then the signal could be lost in quantization error. If the scale was set too high then the relative signal to interferer ratio would be distorted, the subspace would be difficult to find and the data output would not show good RFI mitigation even though the system is functioning properly. After picking a good quantization scale the data could be processed and the output from the correlator, subspace computer, and beamformer could be observed.

4.3.3 Data Output Results

Data was first read in and copied over to the GPU. It was then processed through all four component sub-systems of the interference nulling beamformer: correlator, subspace computer, weight projector, and beamformer. Resulting data output was compared with a MATLAB-simulated expected result for each part. If the data matched, the sub-system was considered functional, at least at an initial level. The following sections present the resulting data output that verified correct operation of the XRFI filter.

Correlator Output

To test the correlator the data modeler was set with the following parameters. The simulated SNR was set to -30dB below the noise floor. The simulated INR was set to -100 dB below the noise floor to simulate relatively 0 interference. The signal is broadband; evenly distributed across all

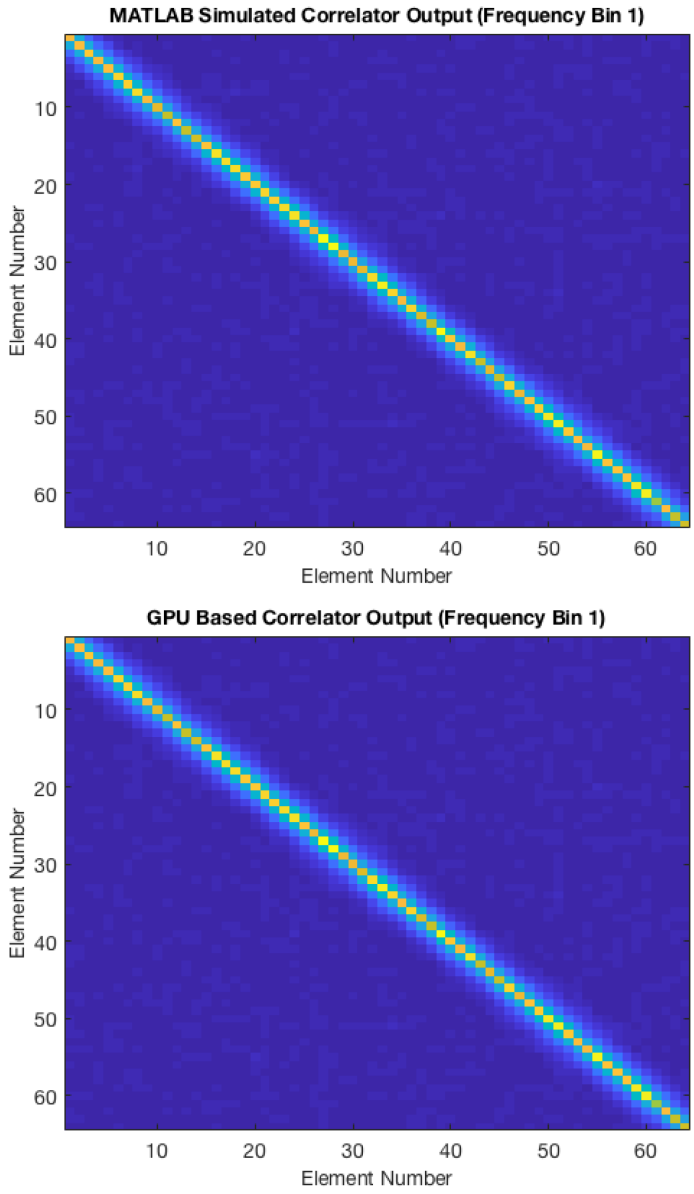

Figure 4.1: Correlator Unit test. Top: Simulated Result. Bottom: Output from GPU

frequency channels. For this test, there is little signal component, just as in real radio astronomy applications, no interference component, and noise is simulated to be the strongest component. With the mutual coupling model, the expected resulting correlation matrix, is Toeplitz for all 25 frequency channels, and the output shows that it was. Figure 4.1 shows a colored rendering of the MATLAB simulated correlation matrix and the corresponding matrix gathered at the output of the CUDA correlator.

Projection Matrix Output

To ensure proper operation of the subspace computer, the projection matrix derived from the eigenvector decomposition must be verified. Thus an eigenvector decomposition is done in MATLAB and a projection matrix is computed from it. The data are then placed through the real-time GPU correlator and the resulting correlation matrix is sent to the subspace computer where it produces a projection matrix which is read out for comparison.

Since this test requires an RFI source, the INR was set to be at the noise floor while the SNR was maintained at -30 dB below the noise floor. This provision ensures that the dominant eigenvector can be attributed to the RFI source and a good subspace projection can be constructed.

The two matricies were compared by taking the Frobenius norm of the difference between the two matrices (P_{MAT} for the MATLAB simulated projection matrix and P_{GPU} for the projection matrix computed from on the GPU):

$$\varepsilon = \|P_{MAT} - P_{GPU}\|_F. \tag{4.8}$$

This test yielded $\varepsilon = 0.0018$. Since $\|P_{MAT}\|_F$ = 7.9373 and $\varepsilon << \|P_{MAT}\|_F$, the matrices were a close match.

Beamformer Output

The Beamformer output is a binned version of the sample power spectrum estimate. To finalize the unit testing of the XRFI filter, data are read into the input of the filter. It then moves through all four stages of the filter and the beamformer's weights are projected into the RFI mitigated subspace. The expected power spectrum, with subspace projected beamformer weights, is computed in MATLAB and the actual GPU output is compared with the simulation. Figure 4.2 shows the comparison with a few data points called out to show decimal accuracy. The signal, in this case, has an SNR of -30 dB with an INR of 10 dB with respect to the noise floor to simulate dominant RFI. Also recall from Chapter 3 that the output from a single GPU has 25 non-contiguous channels, the results from this test are also non-contiguous.

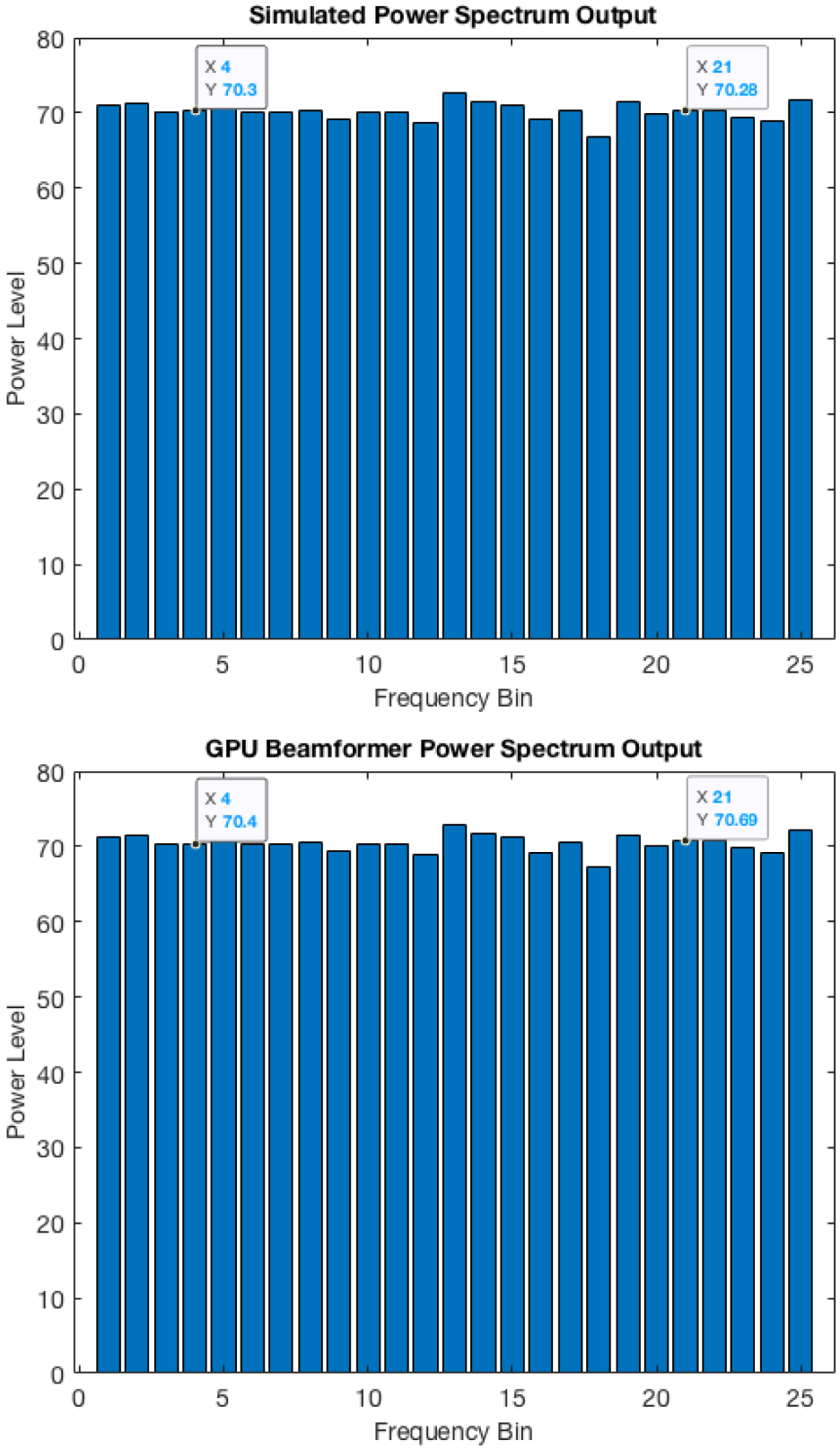

Figure 4.2: Beamformer Initial Test. Top: Simulated Result. Bottom: Output from GPU

4.4 Verification of the Adaptive RFI Mitigating Beamformer Output

The ultimate goal of the Adaptive RFI Mitigating Beamformer is to mitigate RFI in real-time. An argument that real-time operation can be achieved with XRFI was presented in Section 4.2. In this section the beampattern is examined to verify a null was in fact placed in the direction of the RFI as well as a series of six power spectrum tests that should prove XRFI is a fully functional 64 element array adaptive beamformer implemented in a GPU that is capable of detecting a signal of interest in the presence of RFI.

4.4.1 The Beampattern Result

The purpose of rendering the beampattern is to provide insight into the spatial behavior of the beamformer-antenna system. With this visualization a full 180 degree scan of the beampattern response is presented for the ULA. Adjacent element spacing is set to $\frac{\lambda}{2}$ between the 64 elements. Max SNR weights are pre-computed for the array and are then projected into the RFI mitigated subspace using P. A signal of interest is to be located at $\Omega_s = +10$ degrees and an interference source is at a direction $\Omega_i = -25$ degrees. The SNR is set to -30 dB and the INR is set to +10 dB to insure that the strongest eigenvector corresponds to the RFI source.

The beampattern is rendered in MATLAB by taking the GPU computed weights and multiplying them by the simulated signal to get the power output response as a unit test source is moved through all angles -90 to 90.This is shown in Figure 4.3. Note first that the beam main lobe is centered on Ω_s, at 10 degrees. The dotted line in the beampattern plot represents the direction of the incoming RFI source Ω_i. Notice that the null placed in the direction of the RFI, so we conclude that the RFI mitigating beamformer works!

4.4.2 Power Spectrum Estimation Tests

The beamformer must provide clear SOI detection at signal levels well below the noise floor and also mitigate RFI while causing minimal corruption to the SOI. To verify that the beamformer meets this expectation, six power spectrum tests have been devised. All of these involve modifications to the incoming signal to expose the functionality of the system.

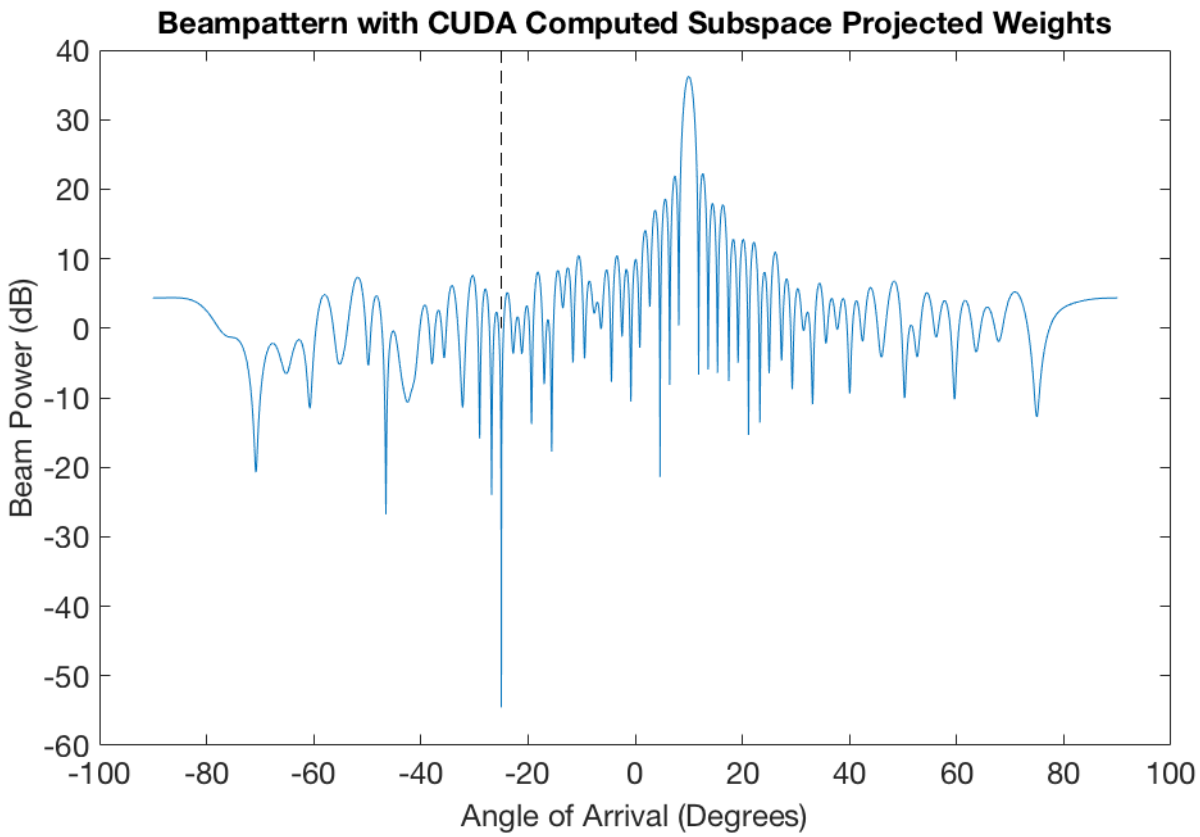

Figure 4.3: Visual MATLAB rendering of a single beampattern produced from GPU computed beamformer weights. MATLAB computed the power at all angles of arrival, GPU computed the weights. The beamformer has a main lobe detecting a signal of interest at 10 degrees. The dotted line shows the direction the RFI at -25 degrees. A null is placed in the direction of the interference.

The signal model was set to an SNR of -20 dB for all six tests with an INR set to +20. Also note that, though FLAG is equipped with the ability to run seven beams simultaneously, each analysis was done for a single beam only.

Recall that the output of the beamformer for the kth frequency channel is

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

The output observed in each test is always a power spectrum. The measured power spectrum output from the beamformer S is defined as

$$S = [s_{k=0},\ s_{k=1},\ \ldots\ s_{k=K-1}]^T \tag{4.9}$$

for K frequency channels and

$$s_k = \frac{1}{N}\sum_{n=0}^{N-1} |y_k[n]|^2 = \frac{1}{N}\sum_{n=0}^{N-1} (\mathbf{w}_k^H \mathbf{x}_k[n])^H (\mathbf{w}_k^H \mathbf{x}_k[n]) = \mathbf{w}_k^H \mathbf{R}_{x,k}[n]\mathbf{w}_k. \tag{4.10}$$

Each test is described as follows:

Test One The first test is a signal only test. The signal model is modified to

$$\mathbf{x}_{\text{signal}}[n] = \mathbf{a}s[n], \tag{4.11}$$

with a resulting power spectrum which we will call S_{signal}. By doing this, one observes the power output as arriving from the source alone without the presence of noise. This represents the desired detected spectrum the observer is looking for when viewing a radio emitting signal out in the universe.

Figure 4.4 shows the source only spectrum as viewed from the beamformer output with no RFI mitigation operations running (i.e. turn off the subspace computer and weight projector and view the signal through the beamformer using max SNR weights)

$$y[n] = \mathbf{w}_{\text{maxSNR}}^H \mathbf{x}_{\text{signal}}[n]. \tag{4.12}$$

The idea is to use this as a point of reference for the other tests. The RFI mitigation is turned off here and will be turned on in later tests to verify that it does not corrupt (or minimally corrupts) the spectrum of the signal of interest.

It's important to note that, though the signal arrives at the beamformer at -20 dB below the noise floor, the arriving signal is amplified by a factor of 35 dB or so through the beamformer. As was discussed above, this amplification is seen in Figure 4.3 recognizing that the beamformer is designed to amplify all signals arriving at 10 degrees.

Test Two The second test involves observing the noise alone:

$$\mathbf{x}_{\text{noise}}[n] = \mathbf{n}[n]. \tag{4.13}$$

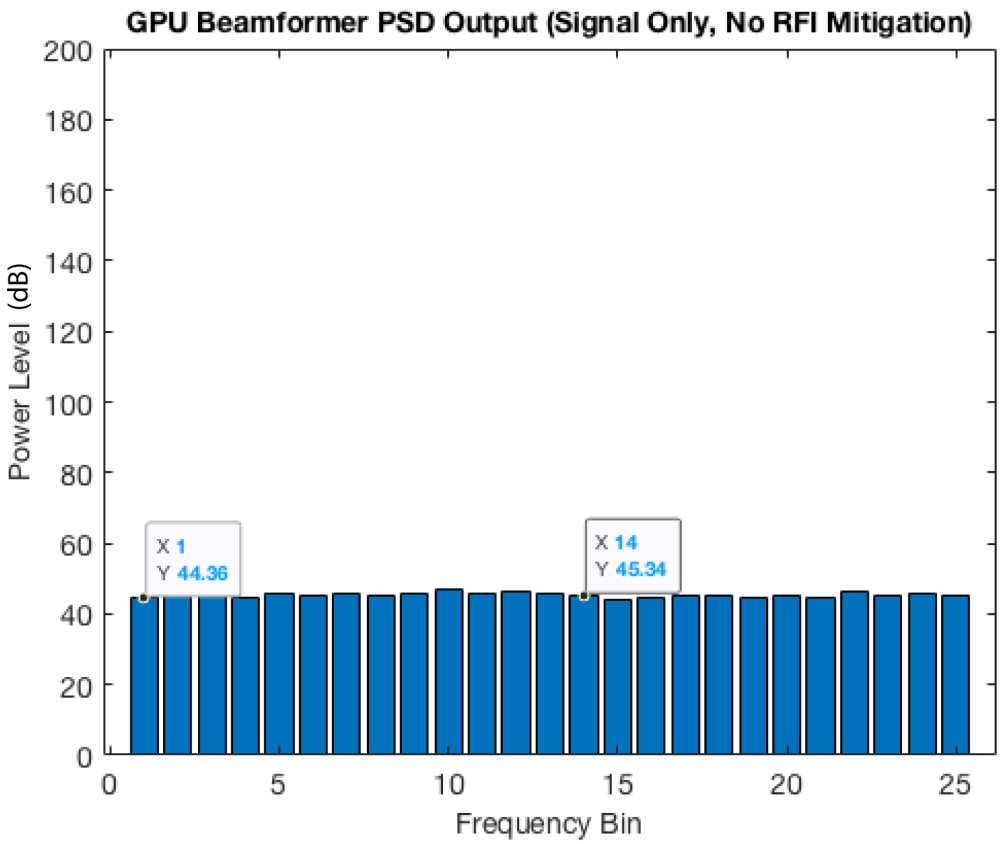

Figure 4.4: Test One: The computed power spectrum estimate of the broadband signal alone, no noise, no RFI, coming out of the beamformer using no RFI mitigating subspace projection. Note: the data tabs show the power levels on two frequency bins.

This power spectrum output (S_{noise}) should reflect only that of noise power arriving at the antenna array. RFI mitigation is turned off for this test, thus only the max SNR weights are used on the beamformer. See Figure 4.5 for the noise power spectrum output.

Test Three The third test involves the SOI and noise together:

$$\mathbf{x}_{sn}[n] = \mathbf{a}s[n] + \mathbf{n}[n], \tag{4.14}$$

with power spectrum S_{sn}. Again, RFI mitigation is off for this test. The beamformer is working correctly if

$$S_{\text{sn}} = S_{\text{source}} + S_{\text{noise}}. \tag{4.15}$$

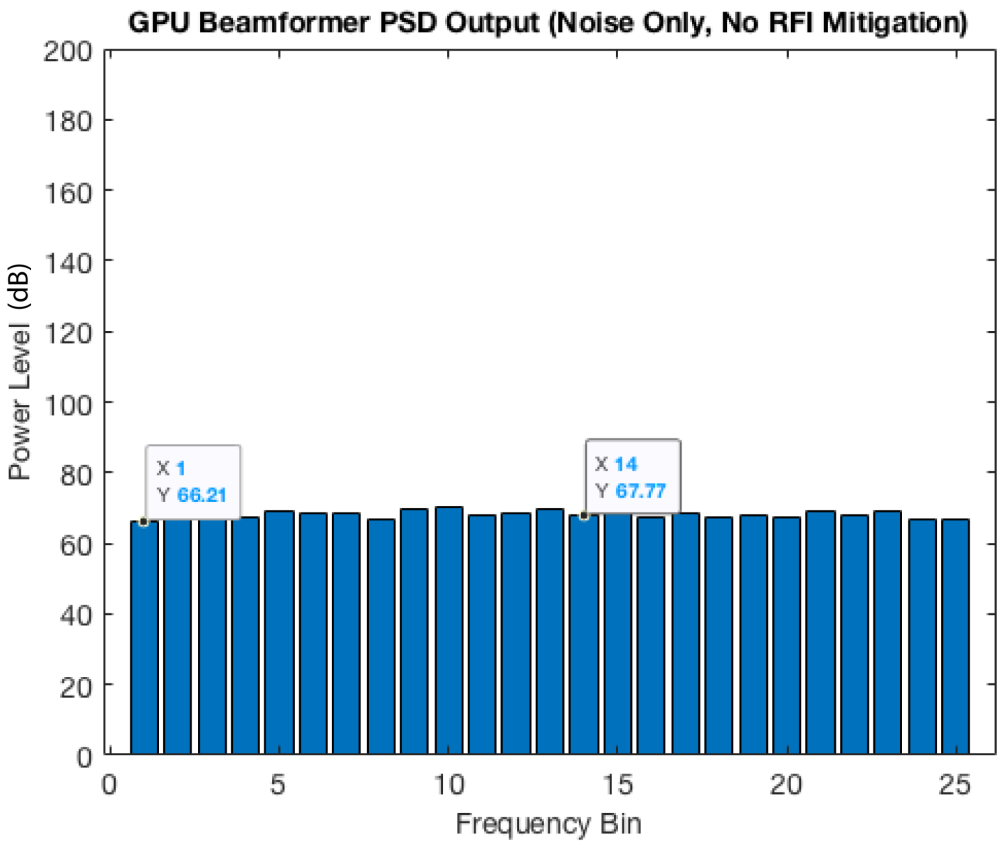

Figure 4.5: Test Two: The computed PSD coming out of the beamformer detecting noise only. No RFI mitigating subspace projection.

Figure 4.6 shows the signal and noise spectrum. The spectrum is clearly the sum of the signal and noise spectra.

Test Four The fourth test is to observe the resulting output of the beamformer with the signal model including signal, interferer and noise:

$$\mathbf{x}[n] = \mathbf{a}s[n] + \mathbf{v}d[n] + \mathbf{n}[n]. \tag{4.16}$$

This power spectrum we label S_{maxSNR} since it is the output of the beamformer with all three signal components using only max SNR weights loaded onto the beamformer. No RFI mitigation is enabled. With RFI mitigation turned off for this test, one can infer the contribution the RFI adds

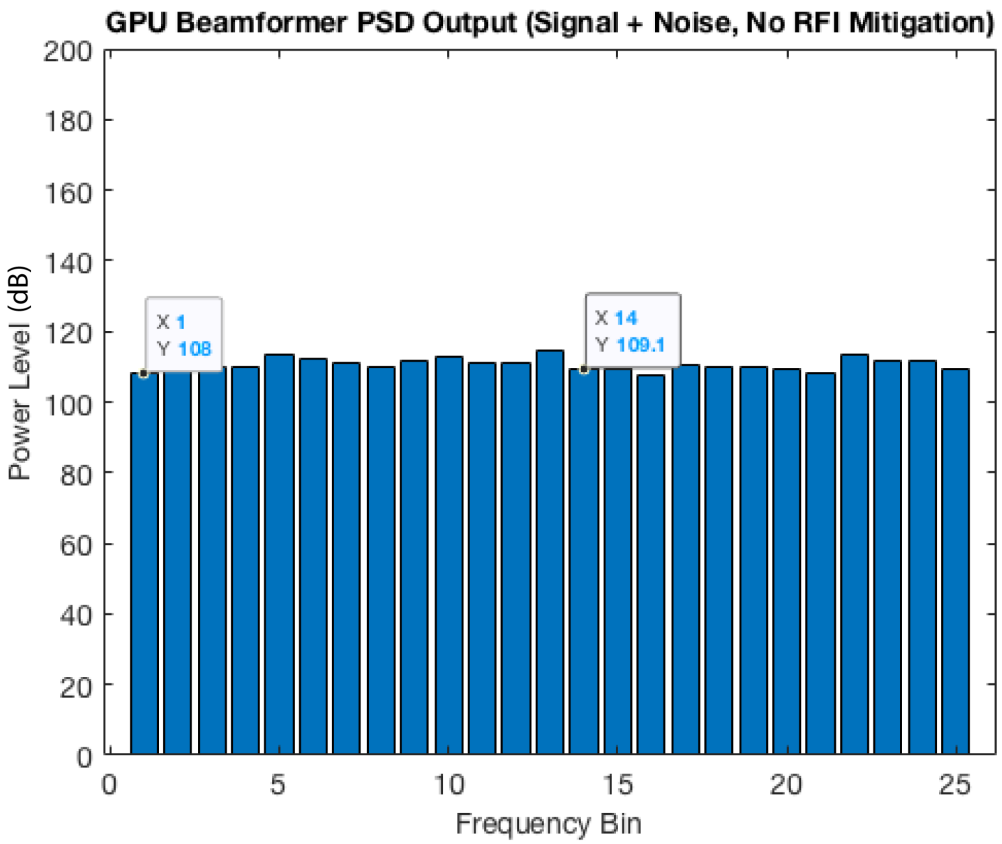

Figure 4.6: Test Three: The computed PSD coming out of the beamformer detecting both signal and noise, no RFI. The subspace projection is turned off for this test. Note that the spectrum is the sum of the two previous spectra.

to the signal:

$$S_{\text{RFI}} = S_{\text{maxSNR}} - S_{\text{sn}}. \tag{4.17}$$

Figure 4.7 shows the power spectrum for test four. This spectrum will be compared with the spectrum in the next test where RFI mitigation is turned on. The beamformer is working correctly if the RFI component is missing.

Test Five The fifth test is the test that sends the full signal model through the beamformer with RFI mitigation enabled. This reflects the fully filtered response and the resulting spectrum S_{sp} is the desired RFI mitigated spectrum. Figure 4.8 shows the output. The beamformer is correctly

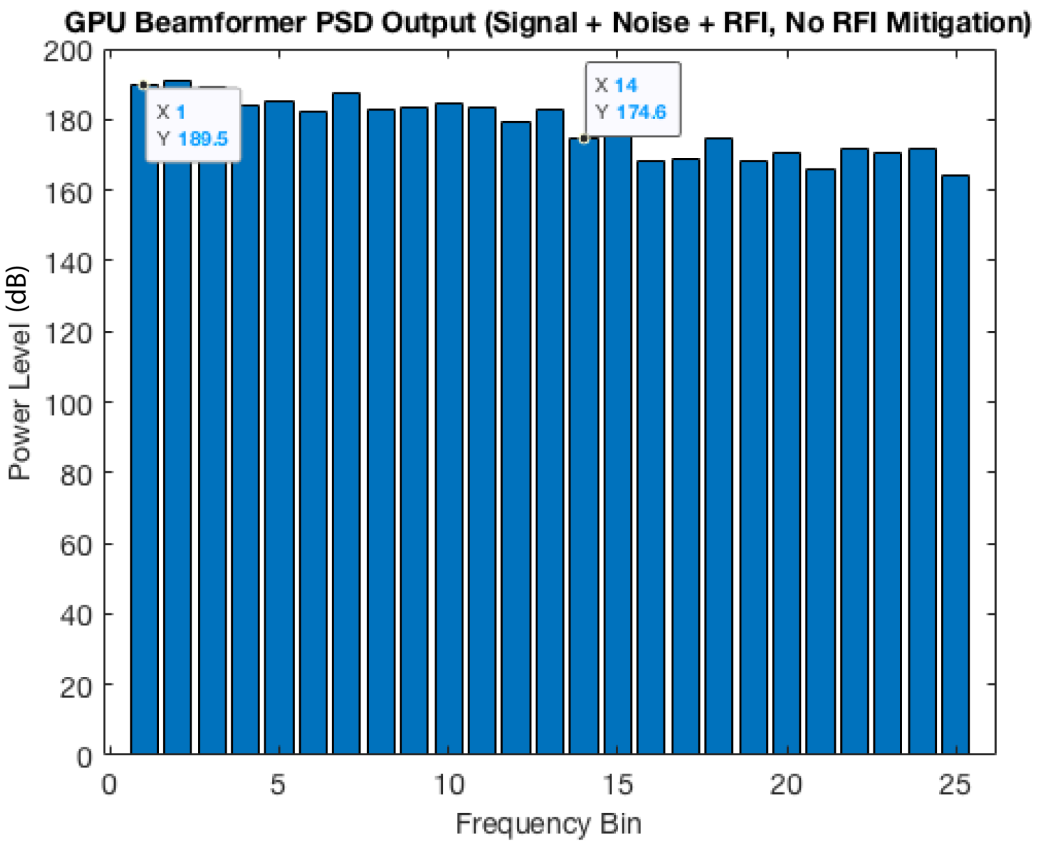

Figure 4.7: Test Four: The computed PSD coming out of the beamformer detecting signal, noise and RFI. The subspace projection is turned off for this test. Note that this spectrum shows an added RFI component on top of the signal + noise spectrum observed previously.

removing (or more correctly "mitigating") the RFI if

$$S_{\text{sp}} \approx S_{\text{sn}}. \tag{4.18}$$

The above equation is an approximate equality and not an equality simply because the signal space and the RFI space are not necessarily orthogonal to one another and thus the RFI is more likely "mitigated" rather than "removed" (see Chapter 2).

Compare Figure 4.7 with Figure 4.8. Note that the extra RFI component has been removed when the full signal is placed through the RFI mitigating filer. Also compare Figure 4.6 with Figure 4.8. Note that they are nearly identical. This also suggests that, with subspace projection enabled, the only part of the signal coming through the filter is the signal + noise component.

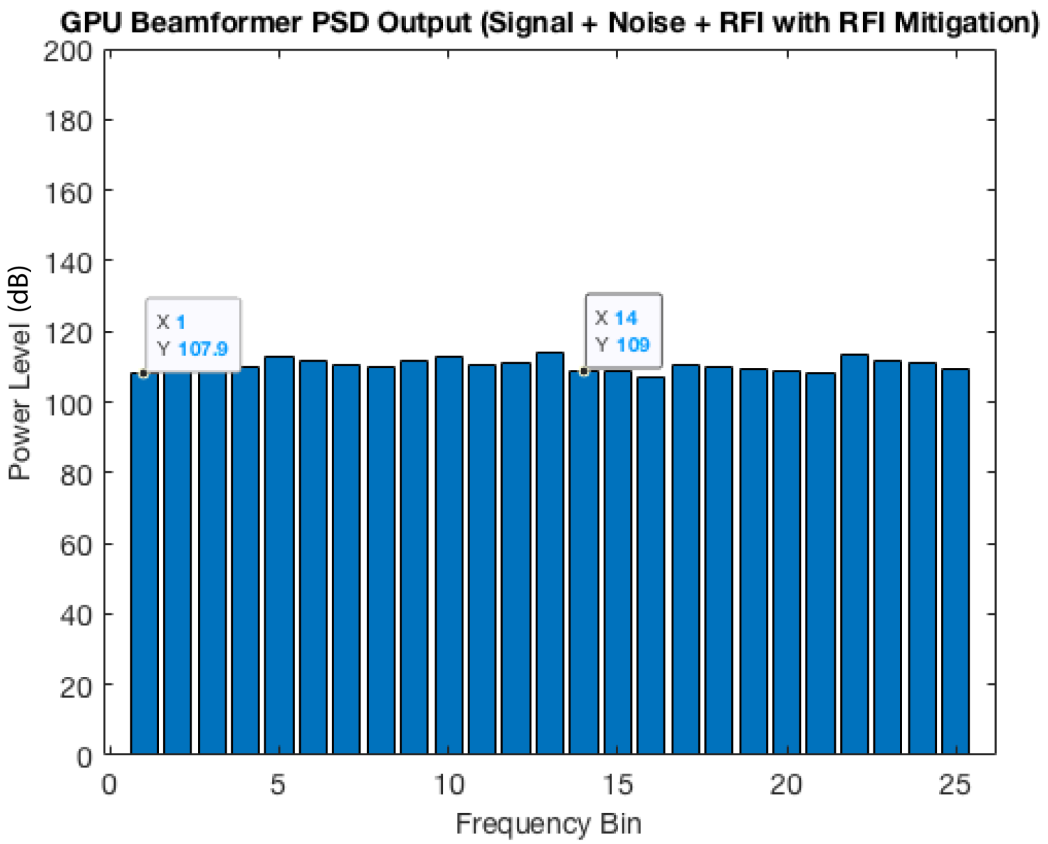

Figure 4.8: Test Five: The computed PSD coming out of the beamformer detecting signal, noise and RFI with RFI mitigation enabled. Note that this spectrum approximately matches the signal + noise spectrum of Figure 4.6 showing that it successfully removed the RFI leaving the signal + noise untouched, and that it shows the RFI component missing from the signal + noise + RFI spectrum of Figure 4.7.

Test Six The final test is to check if the original signal remains uncorrupted after passing through the beamformer with subspace projected weights. For this test

$$\mathbf{x}_{\text{signal}}[n] = \mathbf{a}s[n]$$

again and the weights used at the beamformer are the projected weights:

$$y[n] = \mathbf{w}_{\text{sp}}^{H}\mathbf{x}_{\text{signal}}[n]. \tag{4.19}$$

The resultant spectrum $S_{\text{sig,sp}}$ should be approximately equal to the spectrum found in test one

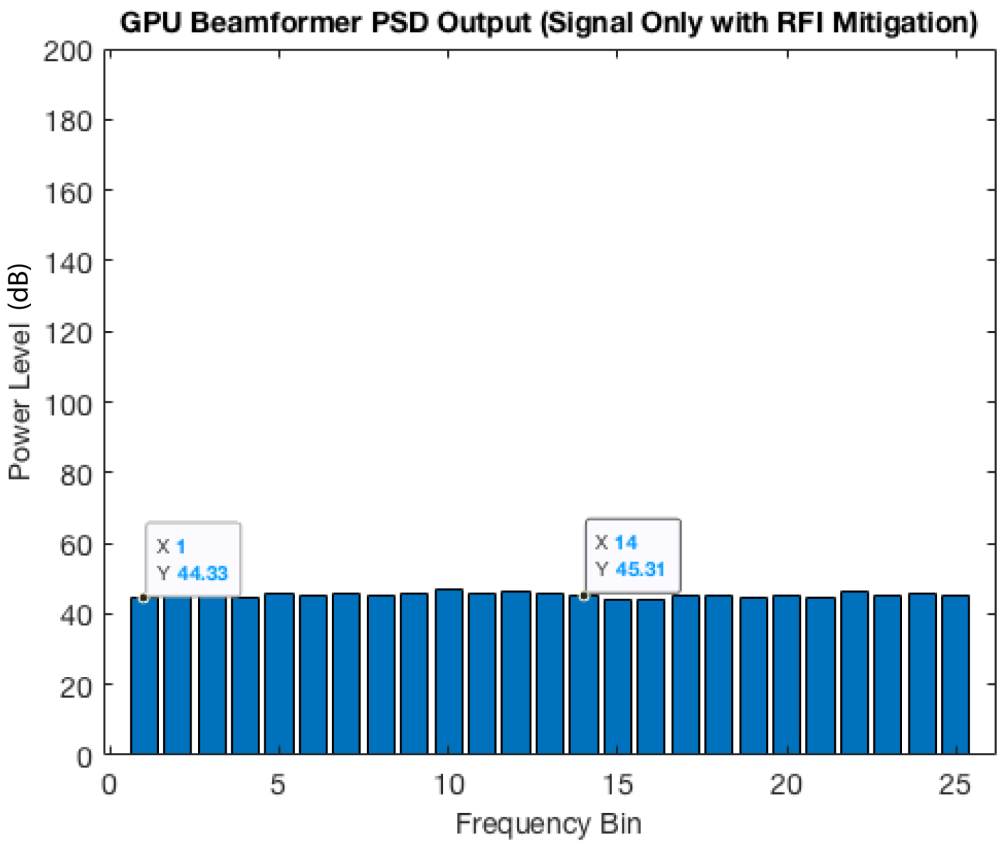

Figure 4.9: Test Six: The computed PSD coming out of the beamformer detecting signal only while also beamforming using the subspace projected weights. Note that the signal is uncorrupted by comparing this Figure with Figure 4.4.

(S_{signal}), thus the beamformer is working correctly if

$$S_{\text{sig,sp}} \approx S_{\text{signal}}. \tag{4.20}$$

Observe Figure 4.9 to see the results of the sixth test. Note that Figure 4.9 nearly exactly matches Figure 4.4.

Summary of the Six Tests After examining the results of each test, the conclusion is that the beamformer has proven both that it can correctly mitigate the RFI while also maintaining minimal corruption of the SOI. This implies that the user will be able to reduce the corruption on channels with RFI and be able to recover at least some if not all needed signal data in previously unreliable channels!

4.5 Conclusion

It has been the goal of the present discussion to demonstrate that the spatial filter does in fact meet the requirements for it to be considered a real-time RFI mitigating beamformer capable of radio astronomy grade applications. First a timing analysis of the GPU performance was explored which proved that the computation time of the XRFI filter meets the timing budget. The beamformer then needed to show correctly filtered output. The initial data comparison tests proved that the GPU implemented system rendered data that matched the MATLAB simulated expected results. Next, the MATLAB rendered beampattern showed a null in the direction of the RFI. This visually confirms that the filter spatially filters the RFI. Finally, the six power spectrum tests show that, at least for this test signal scenario, the SOI can successfully be recovered from an RFI infected channel if passed through the XRFI real-time beamformer.

The XRFI filter is ready to be implemented on the FLAG system by integration into HASHPIPE. Its robust timing and filtering capabilities also make it the perfect solution for other applications that need interference cancellation in real-time. The future steps and solutions that XRFI will offer are the subject of the following chapter.

CHAPTER 5. CONCLUSIONS: THE FUTURE OF XRFI

5.1 Where to from Here?

Now that the filter has been demonstrated to function as a stand alone beamformer, and it has been shown that it will meet the real-time requirements, there is more that this new system can offer when fully implemented. First, the system needs to be integrated into HASHPIPE as explained in previous chapters. This will unlock the full potential of the XRFI system for FLAG. After it is integrated into HASHPIPE, this RFI mitigation solution becomes applicable to fields other than radio astronomy! The XRFI beamformer is also planned to be directly inserted into a communications application where it will act as the key component to an anti-jamming system for the office of naval research (ONR). It is the hope of the author that the system will be of use to both the radio astronomy as well as the communications communities.

This chapter will describe what the future of the XRFI beamformer should be. It will begin with a discussion of a proposed round-robin scheme for real-time broadband RFI mitigation in the HASHPIPE environment. We will end with a discussion on other applications outside of FLAG.

5.2 Integration into HASHPIPE

Currently, the XRFI filter is in a state where it can only operate on one data block at a time. Since stand-alone mode is a proof of concept, it was documented. However, in order to enable the FLAG system to cancel RFI, the XRFI stand-alone beamformer must be integrated into the whole. This means that it must be made into a plugin for HASHPIPE.

Creation of the HASHPIPE plugin is straightforward. It involves moving all of the function calls in the stand alone filter into the HASHPIPE plugin environment. Wrapper functions in the HASHPIPE state machine are well defined and have been done for the real-time beamformer [38].

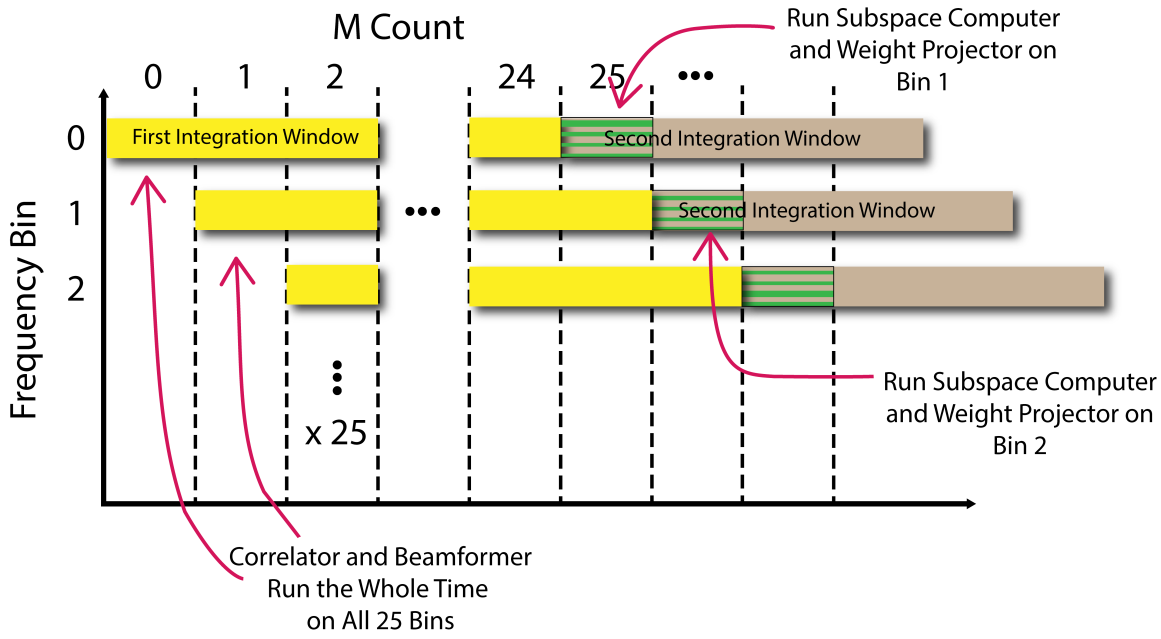

Figure 5.1: Round Robin Timing Diagram

XRFI will be integrated in the same manner as the real-time beamformer. The format of the programming files should be easy enough to follow.

5.3 Extending the Bandwidth: the Round-Robin Proposition

After integration into HASHPIPE occurs, the system will mitigate RFI across a significant band but it doesn't operate on the whole bandwidth yet. Recall from chapters three and four that, due to computational resource limitations, the subspace computer can only operate on one frequency channel per block of data (here we assume the FLAG data block size has been doubled as suggested in Ch. 4 to accommodate two instances per GPU, thus producing a 26 ms block time window). Also recall that at any given moment a total of 20 instances of the HASHPIPE are running (5 HPCs $\times$ 2 GPUs per HPC $\times$ 2 HASHPIPE instances per GPU). This means that, if the XRFI beamformer is running on all instances, 20 bins are being mitigated on a 13ms time window. Since the bin width is 303kHz, running the filter on 20 frequency bins accounts for 20 $\times$ 303kHz = 6.06 MHz. While this is a good start it can be improved upon.

The proposed solution involves cycling through the frequency bins, moving from bin to bin on each 13ms time window in a round-robin scheme, as shown in Figure 5.1. The figure is

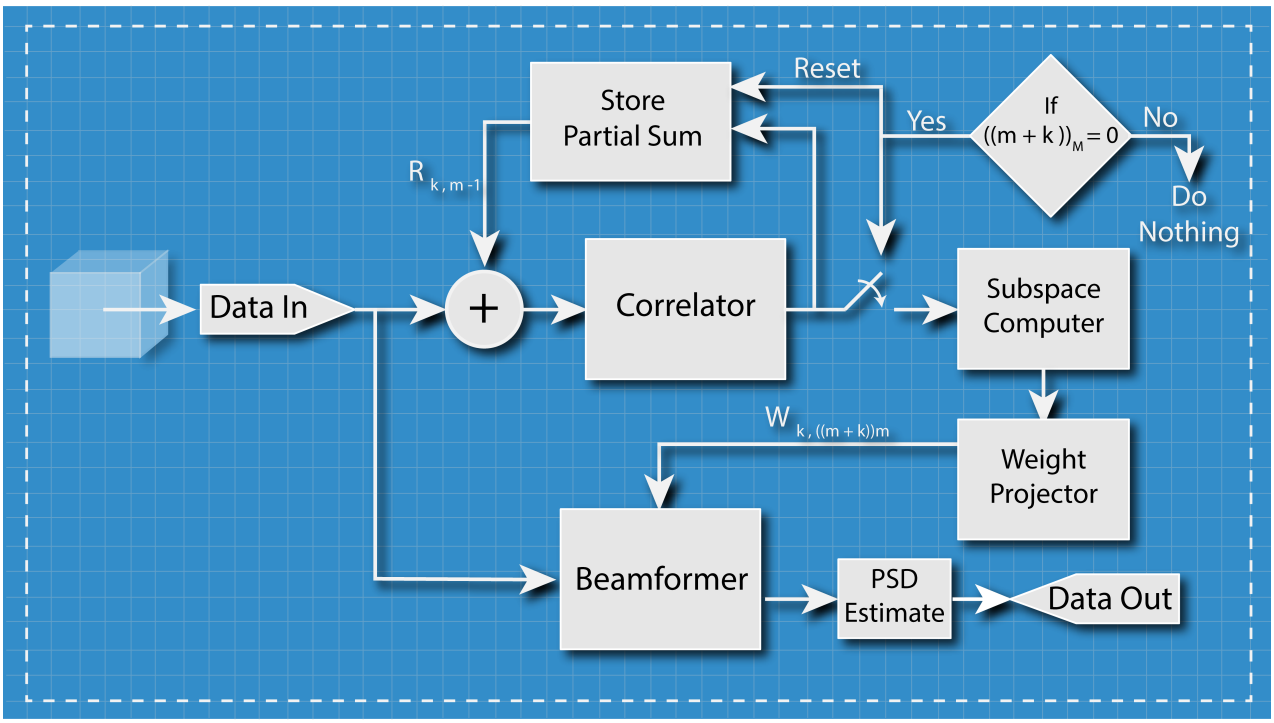

Figure 5.2: Modified Signal Flow Diagram for Round Robin for the *k*th frequency channel.

organized into "M counts." A single M count corresponds to a 4000 time sample block. Data flows in HASHPIPE on a block-by-block basis, so one may think of the block as a single system tick. Both the correlator and the beamformer execute at the same time across all 25 frequency channels at every system tick. The correlator output is saved into a partial sum every system tick and data is dumped when the subspace computer asks for an autocorrelation matrix once every 25 time windows. Work is done on each frequency channel in a staggered fashion so that after the first 25 blocks have passed by, the weights are updated once for each frequency channel on each time window. Figure 5.2 shows a signal flow diagram for the *k*th frequency channel.

This cycling makes it such that the subspace computer and weight projectors only have to work on one frequency channel at a time but all 25 frequency bins in a HASHPIPE instance can be serviced. The total time window is 25 $\times$ 26ms = 650ms. This means that after 650ms the whole bandwidth is fully subspace projected with a null being placed in the direction of the moving RFI for 500 frequency channels. In other words, 500 $\times$ 303kHz = 151.5 MHz of bandwidth are updated at a refresh rate of $\frac{1}{650ms} = 1.54$ times per second. If the flying satellite or RFI source is relatively stationary over a 650ms time window then no subspace smearing will occur.

Prior work has shown that for a phased array feed even larger than FLAG, GPS satellites move slowly enough that this 650ms correlation integration window would not produce "subspace smearing" to an extent which would reduce cancelling null depth [18]. We conclude that 650ms integration and weight update time windows would be more than adequately frequent for many real-world RFI moving sources.

This is an exciting proposition considering that, at present, no system can mitigate across such a bandwidth at such a rate. With this fully implemented it will become the backbone for future projects that need interference mitigation.

5.4 The Communications Application for the Office of Naval Research (ONR)

RFI mitigation in a radio astronomy paradigm is only one application of the XRFI filter. Interference mitigation is not just for deep space observation. If the idea of an interferer is recast into the idea of a jammer in a communications application then the XRFI filter becomes an anti-jamming beamformer.

Consider an incoming communications signal arriving at an antenna array. The signal could be modulated using a digital modulation scheme (i.e. QPSK). The plane wave model would be exactly the same as was pictured in Figure 2.1. A jammer interferes in a similar manner to an RFI source in radio astronomy. Thus the signal model could be essentially the same:

$$\mathbf{x}[n] = \mathbf{a}s[n] + \mathbf{v}d[n] + \mathbf{n}[n], \tag{5.1}$$

excepting that the second component would refer to the jamming source intent on blocking communication on all frequency channels by spamming a strong signal across a wide band.

For a communication link, the XRFI filter will be useful in the scenario where the jamming signal is significantly stronger than the transmission source. First, the transmitter and beamforming receiver are designed to optimize the SNR between a maximum transmission power and a minimum acceptable bit error rate (BER) without the presence of a jammer. The link is then assumed to be implemented in an environment with a jammer INR large enough that the strongest eigenvector of the antenna array correlation matrix would correspond to the jamming source. Under this

condition, a matrix that projects into the null space of the jamming signal can be constructed using the exact same principles explained in Chapter 2.

The XRFI filter need only be modified in a few ways to accomplish the task of jamming mitigation in an array communications environment. First, the vector lengths need to match the number of antenna elements in the communications application. Next, the number of frequency channels and bin widths need to match the bandwidth of the link. Finally, The time windows would need to be adjusted to meet the relevant integration times appropriate to expected jammer motion. Other than these changes the filter would work the same as it will in the FLAG system as it is integrated into HASHPIPE.

The office of naval research (ONR) is currently sponsoring a prototype anti-jamming communication link that will use the XRFI system as its primary anti-jamming payload. The system will use a 16-element patch antenna connected to F-engines housed on a newer board from UC Berkely called the SNAP board. These feed into a set of four HPCs that house nVidia GTX 2080 GPUs. The design paradigm is very similar to FLAG and will use HASHPIPE (or something similar) to do the data throughput. Successful final implementation the full XRFI yields both an RFI mitigation system for RA as well as a communication interference cancelling solution for ONR.

5.5 Conclusion

The future of the XRFI filter is to provide the RFI mitigation solutions to various applications. It will need to be integrated into HASHPIPE and placed in the FLAG ecosystem in order to run on real world data. This should be a feasible task as discussed. Then it will be integrated into the communication link being developed for the office of naval research. The system is capable of mitigating interference in many applications.

As an integral part of the radio astronomy systems group at Brigham Young University the XRFI filter should be a launch point for future RFI mitigation subsystems on other projects involving GPU solutions to communications and radio astronomy. It may even be a starting place for RFI mitigation in BYU's new ALPACHA phased array feed project for the radio astronomy observatory down in Puerto Rico. It is the author's best hope that the XRFI filter will serve the community well and help promote the use of RFI mitigation solutions in radio telescopes as the science advances in the coming years.

Radio frequency interference (RFI) mitigation enables radio astronomical observation in-frequency bands that are shared with many modern satellite and ground based devices by filteringout the interference in corrupted bands. The present work documents the development of a beam-former (spatial filter) equipped with RFI mitigation capabilities. The beamformer is intended forsystems with antenna arrays designed for large bandwidths. Because array data post processing onlarge bandwidths would require massive memory space beyond feasible limits, there is a need fora RFI mitigation system capable of doing processing on the data as it arrives in real-time; storingonly a data reduced result into long term memory. The real-time system is designed to be imple-mented on both the FLAG phased array feed (PAF) on the Green Bank telescope in West Virginia,as well as future radio astronomy projects. It will also serve as the anti-jamming component incommunications applications developed for the United States office of naval research (ONR). Im-plemented on a graphical processing unit (GPU), this beamformer demonstrates a working singlestep filter using nVidias CUDA technology, technology with high-speed parallelism that makesreal-time RFI mitigation possible.

REFERENCES

[1] *ITU Radio Regulations, CHAPTER II Frequencies, ARTICLE 5 Frequency allocations, Section IV Table of Frequency Allocations*. 2

[2] Leshem, A., and van der Veen, A., 2000. "Radio-astronomical imaging in the presence of strong radio interference." *IEEE Transactions on Information Theory,* **46**(5), Aug, pp. 1730–1747. 3

[3] Leshem, A., and van der Veen, A., 2001. "Multichannel detection and spatial signature estimation with uncalibrated receivers." In *Proceedings of the 11th IEEE Signal Processing Workshop on Statistical Signal Processing (Cat. No.01TH8563)*, pp. 190–193. 3

[4] van der Veen, A., Amir Leshem, and Boonstra, A., 2004. "Signal processing for radio astronomical arrays." In *Processing Workshop Proceedings, 2004 Sensor Array and Multichannel Signal*, pp. 1–10. 3

[5] Leshem, A., and van der Veen, A., 2001. "Adaptive suppression of rfi and its effect on radio-astronomical image formation." In *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, Vol. 3, pp. 616–619 vol.3. 3

[6] Jeffs, B. D., Warnick, K. F., and Li, L., 2003. "Improved interference cancellation in synthesis array radio astronomy using auxiliary antennas." In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, Vol. 5, pp. V–77. 3

[7] Jeffs, B. D., Li, L., and Warnick, K. F., 2005. "Auxiliary antenna-assisted interference mitigation for radio astronomy arrays." *IEEE Transactions on Signal Processing,* **53**(2), Feb, pp. 439–451. 3

[8] Hansen, C., Warnick, K. F., and Jeffs, B. D., 2004. "Interference cancellation using an array feed design for radio telescopes." In *IEEE Antennas and Propagation Society Symposium, 2004.*, Vol. 1, pp. 539–542 Vol.1. 3

[9] Jeffs, B. D., and Warnick, K. F., 2007. "Bias corrected psd estimation with an interference canceling array." In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, Vol. 2, pp. II–1145–II–1148. 3

[10] Warnick, K. F., Waldron, J., Landon, J., Lilrose, M., Jeffs, B. D., Fisher, J. R., and Bradley, R., 2007. "Experimental results on interference mitigation with a 19 element array feed." In *The Second European Conference on Antennas and Propagation, EuCAP 2007*, pp. 1–5. 3

[11] Jeffs, B. D., Warnick, K. F., Landon, J., Waldron, J., Jones, D., Fisher, J. R., and Norrod, R. D., 2008. "Signal processing for phased array feeds in radio astronomical telescopes." *IEEE Journal of Selected Topics in Signal Processing,* **2**(5), Oct, pp. 635–646. 3, 8

[12] Warnick, K. F., Jeffs, B. D., Carter, D., Webb, T., Landon, J., Elmer, M., Norrod, R. D., and Fisher, J. R., 2010. "Active impedance matching, calibration, and interference mitigation for the byu/nrao l-band phased array feed." In *2010 IEEE International Symposium on Phased Array Systems and Technology*, pp. 624–628. 3, 14

[13] Jeffs, B. D., and Warnick, K. F., 2009. "Spectral bias in adaptive beamforming with narrowband interference." *IEEE Transactions on Signal Processing,* **57**(4), April, pp. 1373–1382. 3

[14] Boonstra, A. J., Wijnholds, S. J., van der Tol, S., and Jeffs, B. D., 2005. "Calibration, sensitivity and rfi mitigation requirements for lofar." In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, Vol. 5, pp. v/869–v/872 Vol. 5. 3

[15] Jeffs, B. D., and Warnick, K. F., 2013. "Spatial array processing methods for radio astronomical rfi mitigation." In *2013 US National Committee of URSI National Radio Science Meeting (USNC-URSI NRSM)*, pp. 1–1. 3

[16] Landon, J., Jeffs, B. D., and Warnick, K. F., 2012. "Model-based subspace projection beamforming for deep interference nulling." *IEEE Transactions on Signal Processing,* **60**(3), March, pp. 1215–1228. 4

[17] Warnick, K. F., Jeffs, B. D., Diao, J., Black, R. A., Brady, J., Roshi, A., Shillue, B., White, S., Simon, B., and Fisher, R., 2014. "Experimental tests and signal processing for a cryogenic l-band phased array feed on the green bank telescope." In *2014 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, pp. 339–340. 4, 16

[18] Black, R. A., Jeffs, B. D., Warnick, K. F., Hellbourg, G., and Chippendale, A., 2015. "Multitier interference-cancelling array processing for the askap radio telescope." In *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*, pp. 261–266. 4

[19] Hellbourg, G., Weber, R., Capdessus, C., and Boonstra, A., 2012. "Oblique projection beamforming for rfi mitigation in radio astronomy." In *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 93–96. 4

[20] Hellbourg, G., Trainini, T., Weber, R., Moreau, E., Capdessus, C., and Boonstra, A. J., 2012. "Rfi subspace estimation techniques for new generation radio telescopes." In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pp. 200–204. 4

[21] Hellbourg, G., Weber, R., Abed-Meraim, K., and Boonstra, A. J., 2014. "Rfi spatial processing at nancay observatory : Approaches and experiments." In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5387–5391. 4

[22] Kant, G. W., Patel, P. D., Wijnholds, S. J., Ruiter, M., and van der Wal, E., 2011. "Embrace: A multi-beam 20,000-element radio astronomical phased array antenna demonstrator." *IEEE Transactions on Antennas and Propagation,* **59**(6), June, pp. 1990–2003. 4

[23] Hellbourg, G., Chippendale, A. P., Kesteven, M. J., and Jeffs, B. D., 2014. "Reference antenna-based subspace tracking for rfi mitigation in radio astronomy." In *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1286–1290. 4

[24] Hellbourg, G., Chippendale, A. P., Tuthill, J., and Jeffs, B. D., 2015. "Statistical performance of reference antenna based spatial rfi mitigation for radio astronomy." In *2015 IEEE International Symposium on Antennas and Propagation USNC/URSI National Radio Science Meeting*, pp. 1518–1519. 4

[25] Hellbourg, G., 2015. "Subspace smearing and interference mitigation with array radio telescopes." In *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*, pp. 278–282. 4

[26] Hellbourg, G., 2016. "Interference detection and estimation for spatial filtering: Application to radio interferometry." In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 306–308. 4

[27] Hellbourg, G., Abed-Meraim, K., and Weber, R., 2016. "Impact of array calibration on rfi mitigation." In *2016 24th European Signal Processing Conference (EUSIPCO)*, pp. 285–289. 4

[28] Wyckoff, P. S., and Hellbourg, G., 2016. "Polar excision for radio frequency interference mitigation in radio astronomy." In *2016 Radio Frequency Interference (RFI)*, pp. 132–135. 4

[29] Hellbourg, G., Bannister, K., and HotarP, A., 2016. "Spatial filtering experiment with the askap beta array." In *2016 Radio Frequency Interference (RFI)*, pp. 37–42. 4

[30] Chippendale, A. P., and Hellbourg, G., 2017. "Interference mitigation with a modified askap phased array feed on the 64m parkes radio telescope." In *2017 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, pp. 948–951. 4

[31] Ford, J. M., and Buch, K. D., 2014. "Rfi mitigation techniques in radio astronomy." In *2014 IEEE Geoscience and Remote Sensing Symposium*, pp. 231–234. 4

[32] Beaudet, C., Ford, J., Minter, T., McCarty, M., O'Neil, K., and Prestage, R., 2013. "Radio frequency interference management efforts at the national radio astronomy observatory green bank site." In *2013 US National Committee of URSI National Radio Science Meeting (USNC-URSI NRSM)*, pp. 1–1. 4

[33] Van Veen, B. D., and Buckley, K. M., 1988. "Beamforming: a versatile approach to spatial filtering." *IEEE ASSP Magazine,* **5**(2), April, pp. 4–24. 7, 11

[34] Hayes, M. H., 2014. *Statistical digital signal processing and modeling*. Wiley. 7

[35] Van Trees, H. L., 2001. *Detection, estimation, and modulation theory*. John Wiley. 7, 11

[36] Black, R. A., 2017. "Phased-array feed instrumentation and processing for astronomical detection, interference mitigation, and transient parameter estimation." PhD thesis, Brigham Young University. 16

[37] Burnett, M. C., 2017. "Advancements in radio astronomical array processing: Digital back end development and interferometric array interference mitigation." Master's thesis, Brigham Young University. 16

[38] Ruzindana, M. W., 2017. "Real-time beamforming algorithms for the focal l-band array on the green bank telescope." Master's thesis, Brigham Young University. 27, 47

APPENDIX A. FULL NVIDIA TIMING REPORT

Table A.1: Full nVidia Profiler Functional Timing Report

% of 13ms	Time (μs)	Calls	Avg (μs)	Function Name
Beamformer + Wgt. Projector:				
5.60%	728.31	1	728.31	cgemmBatched_64x32
0.22%	27.904	1	27.904	cgemmBatched_64x32
2.22%	287.81	1	287.81	data_restructure
0.78%	100.45	1	100.45	sti_reduction
Total 8.8 %	1.14 ms			
Correlator:				
6.42%	834.01	1	834.01	cgemmBatched_32x32
2.26%	293.34	1	293.34	correlator_data_restructure
Total: 8.7%	1.128 ms			
Subspace Computer:				
3.68%	478.46	63	7.594	syhemv_kernel
2.99%	388.21	62	6.2610	nrm2_kernel
2.94%	382.21	63	6.0660	her2_kernel
6.61%	339.01	62	5.4670	nrm2_kernel
1.36%	176.74	63	2.8050	dot_kernel
1.35%	174.94	64	2.7330	ger_kernel
1.32%	171.49	63	2.7220	gemv2T_kernel_val
1.07%	138.88	63	2.2040	reduce_1Block_kernel
0.92%	119.70	63	1.9000	axpy_kernel_val
0.73%	94.016	62	1.5160	scal_kernel_val
0.45%	57.536	26	2.2120	initIdentityGPU
0.38%	48.401	62	0.780	[CUDA memset]
0.02%	1.7600	1	1.7600	lacpy_kernel
0.02%	1.5360	1	1.5360	reset_diagonal_real
Total: 19.8%	2.573 ms			
MemCpy:				
48.16%	6.26 ms			Host to Device
0.18%	23 μs			Device to Host
Total: 48.27%				
Grand Total: 85.7%	11.128 ms	/	freq. bin	