2017-12-01

Advancements in Radio Astronomical Array Processing: Digital Back End Development and Interferometric Array Interference Mitigation

Mitchell Costus Burnett
Brigham Young University

Advancements in Radio Astronomical Array Processing: Digital Back End Development and Interferometric Array Interference Mitigation

Mitchell Costus Burnett

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

Master of Science

Department of Electrical and Computer Engineering

Brigham Young University

ABSTRACT

Advancements in Radio Astronomical Array Processing:
Digital Back End Development and Interferometric
Array Interference Mitigation

Mitchell Costus Burnett
Department of Electrical and Computer Engineering, BYU
Master of Science

The Brigham Young University (BYU) Radio Astronomy Systems group, in collaboration with the National Radio Astronomy Observatory (NRAO), the Center for Astrophysics at West Virginia University (WVU), and the Green Bank Observatory (GBO) have developed, and commissioned, a broadband real-time digital back end processing system for a 38-element phased array feed (PAF) with 150 MHz of instantaneous bandwidth. This system is capable of producing coarse and fine channel correlations, and implements a real-time beamformer that forms 7 simultaneous dual-polarized beams. This thesis outlines the hardware and software development for the digital back end and presents on-telescope commissioning results. This system has been measured to provide an unprecedented low T_{sys}/η noise level of 28 K and can perform maps of galactic hydrogen observations in a fraction of the time of a conventional single horn feed.

The National Radio Astronomy Observatory (NRAO) has recently announced the concept and development of the next generation Very Large Array (ngVLA), a large interferometric array consisting of 300 radio telescopes and longest baseline (distance between a pair of antennas) of 300 km. Large interferometric arrays have been shown to attenuate radio frequency interference (RFI) because it is decorrelated as it propagates across long baselines. This is not always sufficient, especially with dense core array geometries and with the ever-increasing amount of strong RFI sources. Conventional RFI projection-based mitigation techniques have performed poorly on large interferometers because of covariance matrix estimation error due to decorrelation when identifying interference subspace parameters. This thesis presents an algorithm that overcomes the challenge of decorrelation by applying subspace projection via subarray processing (SP-SAP). Each subarray is designed to have a set of elements with high mutual correlation in the interferer for better estimation of subspace parameters. In simulation, compared to the former approach of applying subspace projection on the full array, SP-SAP improves mitigation of the RFI on the order of 9 dB. A signal of interest is shown then to be observable through the RFI in a full synthetic image.

Keywords: radio astronomy, phased array feeds, digital back end, digital spectrometers, radio frequency interference, interferometers, Next Generation Very Large Array, digital signal processing

ACKNOWLEDGMENTS

There is no amount of praise I can give my wife, Shannon, for her love and support while accomplishing this work. I know she has sacrificed much to support me in pursuing and completing my graduate degree. I am grateful for her companionship and strength, especially in these last few months as we recently welcomed our son Oliver into our family. You both encourage me to be better.

I extend a very special gratitude to my academic advisor Professor Brain D. Jeffs. I would not be the student I am without his invaluable instruction. Constantly, Dr. Jeffs has allowed this work to be my own by steering me in the right direction and assisting me when my understanding was lacking. Thank you for your guidance and counsel.

I am grateful to my parents Rodger and Terresa for their love and support. You have always encouraged and believed in your children! Also, a special thanks to my brother, Parker. I cherish our friendship and the support you have given me through this time.

I owe a lot to my colleagues of the Radio Astronomy Systems group, specifically Richard Black and Mark Ruzindana. Thank you for your time, expertise, help, camaraderie and friendship. It was a pleasure to work along side you and see our mutual hard work lead us to many successful results.

Thank you to many of the great department faculty: Prof. Karl Warnick, Prof. Neal Bangerter, Prof. Michael Rice, and Prof. Brian Mazzeo. Your teaching and support has greatly impacted me and influenced this work.

A special mention to our collaborators Nick Pingel and Kaustubh Rajwade of West Virginia University and Richard Prestage, Luke Hawkins and staff at the Green Bank Observatory.

This work was supported and coordinated by the National Radio Astronomy Observatory, which is a facility of the National Science Foundation operated under cooperative agreement by Associated Universities, Inc. with federal award numbers AST-1519126 (ngVLA community studies), and AST-1309832 (FLAG).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

Astronomy has always been at the forefront of our science disciplines for understanding our place in the universe. Our scientific achievements throughout the evolution of human civilization are often marked by advances in astronomical understanding and tools. To early cultures the night sky was a playground for the Gods. The celestial cycles provided a calendar to measure time and season. The stars ensured safe navigation for the explorer. Understanding astronomy and the physics of celestial bodies led the scientific revolution. Even to this day, we are continuing to expand our knowledge by looking to the expanse of space. The naked eye can only paint a small portrait of a vast and complex universe. Optical telescopes are limited to wavelengths near the visible spectrum and so to get a better picture, astronomers use radio telescopes.

Radio Astronomy (RA) is a branch of astronomy that observes and studies the electromagnetic radiation of celestial bodies at radio frequencies. Many types of matter such as gaseous neutral Hydrogen cannot be seen by optical telescopes, but are easily detected and characterized by their radio emissions. These electromagnetic signals are observed using radio telescopes which often consist of large dish reflectors with an antenna feed at the focal point of the dish.

Discoveries in RA are motivating the development of systems that can achieve better sensitivity, resolution, and survey speed, and compared to receiver systems used for communication applications, instruments in RA are required to operate with signal-to-noise power ratios (SNRs) that are 30 dB or lower. These signal conditions make observations difficult in the presence of strong man-made radio frequency interference (RFI). To address these issues, many collaborative efforts have been formed resulting in developing new antenna systems as well as algorithms and methods for mitigating RFI [1]–[15].

1.1 Phased Array Feeds

Phased array feeds (PAFs) for RA applications are relatively new and replace the traditional single horn antenna feed with a closely spaced (order of 1/2 wavelength) 2-D planar array of small antennas at the focal plane of the reflector [16]. Linear combinations of the closely spaced element voltage time signals adjust the shape and direction of the illumination pattern on the dish. This is known as beamforming [17]. Through beamforming, the far-field beam pattern can be modified and steered to different locations on the sky. This allows for the formation of multiple simultaneous far-field beams, even with a single dish reflector, widening the instantaneous field of view (FOV) without a loss of sensitivity, and increasing survey speed [1].

1.2 Synthesis Imaging

Compared to single dish radio astronomy, synthesis imaging arrays have a higher spatial frequency cutoff (i.e., a much narrower beamwidth due to a very large, order 10 km or greater, sparsely sampled aperture) and are used to form high resolution images of galactic sources. Synthesis imaging arrays consist of separated antenna feeds and form images using the principle of interferometry [18]. The elements of the array provide a sparse spatial sampling of the incident wavefront as if it were sampled by a much larger synthetic aperture. Some examples of active synthesis imaging systems include: the Very Large Array (VLA) near Socorro, New Mexico [19], the Atacama Large Millimeter/Submillimeter Array (ALMA) in Chile [5], the Australian Square Kilometre Array Pathfinder (ASKAP) in western Australia [6], the Westerbork Synthesis Radio Telescope (WSRT) in Westerbork, Netherlands [20], the MeerKAT array as part of the South African Square Kilometre Array (SKA) [8], and the Giant Metrewave Radio Telescope (GMRT) near Pune, India [21].

In 2015, the National Radio Astronomy Observatory (NRAO) announced they would begin supporting an international effort for the initial concept, design and development of a large area radio instrument optimized for imaging thermal emissions down to milliarcsecond scales. The project is currently known as the next generation Very Large Array (ngVLA) [22]. The current VLA site in New Mexico, USA, is the leading proposed location for the new interferometer. The currently proposed array design consists of 300, 25 m single feed telescopes, with the longest baseline spanning up to 300 km. It will operate from 1 GHz to 115 GHz and includes a dense core for high surface brightness imaging.

1.3 Radio Frequency Interference Mitigation

All over the world, observatories and institutes are reporting damaging levels of RFI to the extent that data corruption occurs in up to 100% of their observations for a given frequency band [23]–[26]. RFI is a challenge that has burdened RA for a long time and the challenge will get worse as more services populate the electromagnetic spectrum. For example, the 21-cm (1420.4 MHz) region for moderately red-shifted neutral Hydrogen emissions (HI) is now almost entirely blocked by strong RFI due to downlinks from the GPS, GLONASS, Galileo, and COMPASS navigation satellites, and IRIDIUM communication satellites [27]–[30]. It is also not uncommon that because of contemporary science goals, a radio astronomer is compelled to observe outside frequency bands that are reserved for the exclusive use of passive astronomy observations. It is therefore becoming imperative to find ways to combat ubiquitous man-made interference both inside and outside these reserved bands.

There are many existing and proposed techniques for RFI mitigation in RA. By far, the most common approach is power detection followed by blanking at the correlator output, or in other words, to flag and discard the data [14]. With RFI channel occupancy increasing, flagging is becoming a less viable option for RFI mitigation. Adaptive projection-based methods have shown to be a better alternative that provide high levels of RFI excision without discarding data.

Each signal incident on an array, such as a cosmic signal of interest (SOI) or an interfering satellite, has an unique spatial response pattern on the array known as the "spatial signature." Adaptive projection-based spatial filtering involves estimating interfering array signatures, and then using this information to form deep cancellation nulls on the interferer by applying a projection into the subspace orthogonal to the interferer. This provides the capability to "see through" strong RFI conditions and unmask the SOI rather than discard corrupted data [12], [13].

1.4 Thesis Statement

Future discoveries in RA depend on advancements in signal processing capabilities to support new array systems such as PAFs and to continually improve RFI mitigation techniques that allow instruments to coexist in harsh RFI environments. This thesis addresses both of these issues with the development of a wideband digital back end processing system for a PAF and an improved subspace projection algorithm for large interferometric arrays.

Incorporating a PAF on a radio telescope requires a data acquisition system capable of processing the voltage signal from each element simultaneously. To be scientifically viable and useful to astronomers, such a back end would also be required to process large analog bandwidths with the capability to also provide fine frequency resolution. These requirements to provide a viable instrument results in a very high system data rate.

There has been significant work in the development of digital back ends for PAFs. An early system that was capable of processing a narrow instantaneous bandwidth for a 19 dual-polarized PAF was developed by former graduate students at Brigham Young University (BYU) [9], [31]. The digital back end used five PCs with analog-to-digital converter (ADC) cards and processed 450 kHz of bandwidth. Despite data rate limitations of the hardware and software along with the narrow processing bandwidth, it was an encouraging beginning and showed that cutting-edge hardware and software was required to scale back end systems to acquire more bandwidth.

To facilitate the need for systems to process multiple inputs at a high data rate, the Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) has developed open-source software for special purpose hardware that includes ADCs, Field Programmable Gate Arrays (FPGAs), and is capable of interfacing with a high data (10 GbE) network [32]. Using this hardware, and incorporating a new suite of software for a new digital back end, students at BYU were able to develop a 64 input system that processed 20 MHz of bandwidth and streamed data to disk [10]. The system was deployed at the Arecibo Observatory in Puerto Rico and was demonstrated successfully. With 512 frequency channels spanning 20 MHz this system is an example of a scientifically viable instrument.

Other observatories are currently in the development and testing phases for other back end systems. The second generation 188-element PAF back end of each ASKAP dish processes 384 MHz of bandwidth and forms 36 simultaneous beams with a real-time beamformer [7]. The

APERture Tile in Focus (APERTIF) PAFs used on the WSRT has a back end that process 121 elements and forms 37 simultaneous beams with 300 MHz of bandwidth [33].

This thesis presents the development and commissioning results of the digital back end processing system for the Focal L-Band Array on the Green Bank Telescope (FLAG). This system uses state-of-the-art processing boards from CASPER, High Performance Computers (HPCs), and graphical processing units (GPUs) capable of processing 150 MHz of bandwidth. This back end can provide either a coarse correlator output with 500 frequency channels across the full bandwidth or a fine channel correlator of 3200 frequency channels over 30 MHz of bandwidth. It is also designed to concurrently run a real-time beamformer that simultaneously forms 7 dual-polarized beams. This is the first permanent, science ready, cryogenically cooled, PAF instrument for a major single dish radio telescope, the 100 m diameter Green Bank Telescope (GBT).

Subspace projection techniques for large interferometric arrays were first proposed in [12], [13]. While spatial filtering methods are effective for smaller aperture arrays and PAFs, and appear promising in simple simulations, there is yet to be a commissioned interferometric instrument that has reported use of subspace projection methods in active science observations. This is because at the time, there were still many signal conditions for a practical observation scenario on an interferometer that were not considered, making proposed methods inadequate. When the interference-to-noise power ratio (INR) is low, RFI motion is fast relative to the maximum array aperture dimension and integration dump interval, or when correlation of the RFI is weak because of long distances between antenna pairs (baselines), accurate estimation of subspace parameters is difficult, and conventional adaptive cancelers perform poorly. RFI motion has been characterized and can be overcome with subspace time-evolution models and tracking techniques as in [34]–[37]. The use of auxiliary antennas that track and measure RFI to provide a high INR copy of the undesired signal has been shown to improve mitigation [11], [38]. However, to date, the literature has not yet presented a method to address decorrelation of the RFI. Characterization of the response to and decorrelation of the RFI was given in [39] where interferometric arrays were shown to be less susceptible to strong RFI over single dish telescopes because decorrelation attenuates the interferer. However, this is often inadequate because artifacts of interferers are still present.

Decorrelation of the RFI across the array has remained a persistent issue [40] preventing much of the substantial previous work from being applied to array systems for active science observations. This is because even if auxiliary antennas are available and subspace tracking techniques are implemented, decorrelation of the RFI supersedes all other challenges by corrupting the estimated signal covariance matrix that adaptive cancellation algorithms rely on. This thesis addresses the problem of decorrelation and presents an algorithm for applying subspace projection on subsets of array elements (or subarrays) where high mutual correlation of the RFI is expected to provide better estimates of subspace parameters and improve RFI excision. This subspace projection via subarray processing (SP-SAP) algorithm can readily be adopted and implemented on current interferometry arrays (e.g. VLA, ALMA) and should be considered for future systems (e.g. ngVLA, SKA). The proposed configuration for the ngVLA [41] is used as the platform for this thesis as part of the technical community studies program for developing the ngVLA.

1.5 Research Contributions

The research presented in this thesis results in the development of a new 150 MHz correlator and real-time beamformer digital back end system and SP-SAP algorithm which improves RFI mitigation for interferometric arrays. Specific contributions are:

- Development of a C compatible GPU library implementing a polyphase filter bank (PFB)
- Integration of the fine channel correlator mode into the digital back end
- Software development for the integration of the back end into the Green Bank Observatory (GBO) Python integrated control environment
- Performance testing, tuning, and improvements to the back end software libraries and pipeline codes
- Conducting two commissioning experiments on the GBT
- Simulation analysis for the response of an interferer on an interferometer
- Demonstrating how subspace projection on a large interferometer is ineffective due to estimation error in subspace parameters
- Developing a new algorithm that applies subspace projection on subsets of elements in an interferometer with high mutual correlation in the RFI

- Performing simulations which show improved RFI excision using SP-SAP on large interferometers

1.6 Thesis Outline

This thesis contains two major topics: a description of the hardware and software to develop a real-time processing back end for FLAG, and a new algorithm for applying subspace projection via subarray processing to improve RFI mitigation on large interferometric arrays. A brief description of each chapter follows.

Chapter 2 provides a high level, detailed description of FLAG, hardware, software, development process, and scientific commissioning results.

Chapter 3 presents a tutorial derivation for the polyphase filter bank (PFB) method that in general, is a good reference for the design of PFBs, but which are critical to providing efficient frequency channelization at many stages in the FLAG system.

Chapter 4 describes the implementation and design of the PFB in a graphical processing unit (GPU) for high data rate processing and is used as part of the fine channel correlator for FLAG.

Chapter 5 gives the background of synthetic imaging for an interferometric array to analyze the response of an interferer and decorrelation across large projected baselines. This chapter also shows the effectiveness of subspace projection across large baselines and presents a new algorithm for SP-SAP with simulation results.

Chapter 6 concludes the thesis and summarizes results. Potential paths for future work are also provided.

CHAPTER 2. FLAG BACK END DEVELOPMENT

FLAG is a professional research collaboration effort between the Radio Astronomy Systems group at BYU, the Center for Astrophysics at West Virginia University (WVU), the Green Bank Observatory (GBO), and the National Radio Astronomy Observatory (NRAO) to develop, test, and commission the first, cryogenically cooled PAF with a real-time beamformer and correlator back end capable of processing 150 MHz of instantaneous bandwidth. This system will be a permanent scientific instrument for the Green Bank Telescope (GBT), the worlds' largest 100 m fully steerable radio telescope, located in Green Bank, West Virginia USA.

With a variety of back end operational modes, high sensitivity, and increased survey speed of 3-5x that of a single-pixel feed, FLAG will be influential in accomplishing a number of contemporary science goals in Physics and Astronomy. Target goals for FLAG include, surveys and mapping of diffuse HI around galaxies, pulsar detection, and radio transient surveys. The purpose of this chapter is to give an overview of FLAG and its capabilities as well as to document the technical details and operation of the digital back end system.

2.1 FLAG Overview

A functional block diagram for FLAG is shown in Figure 2.1. The FLAG system is comprised of two principal parts: the front-end array receiver, and a digital signal processing back end.

The front-end array receiver can be further broken down into three components. First, is the PAF consisting of 19 dual-polarized, flared dipole antennas which have been previously designed and developed by BYU and NRAO [2]. The dipoles are then connected to 40 low-noise amplifiers (LNAs) which are cooled in a cryogenic dewar, leaving two disconnected. After the LNAs, each analog signal is then digitized by an integrated system which includes analog I/Q mixing from RF to baseband, sampling at 155.52 MHz, and serialization for transmission over

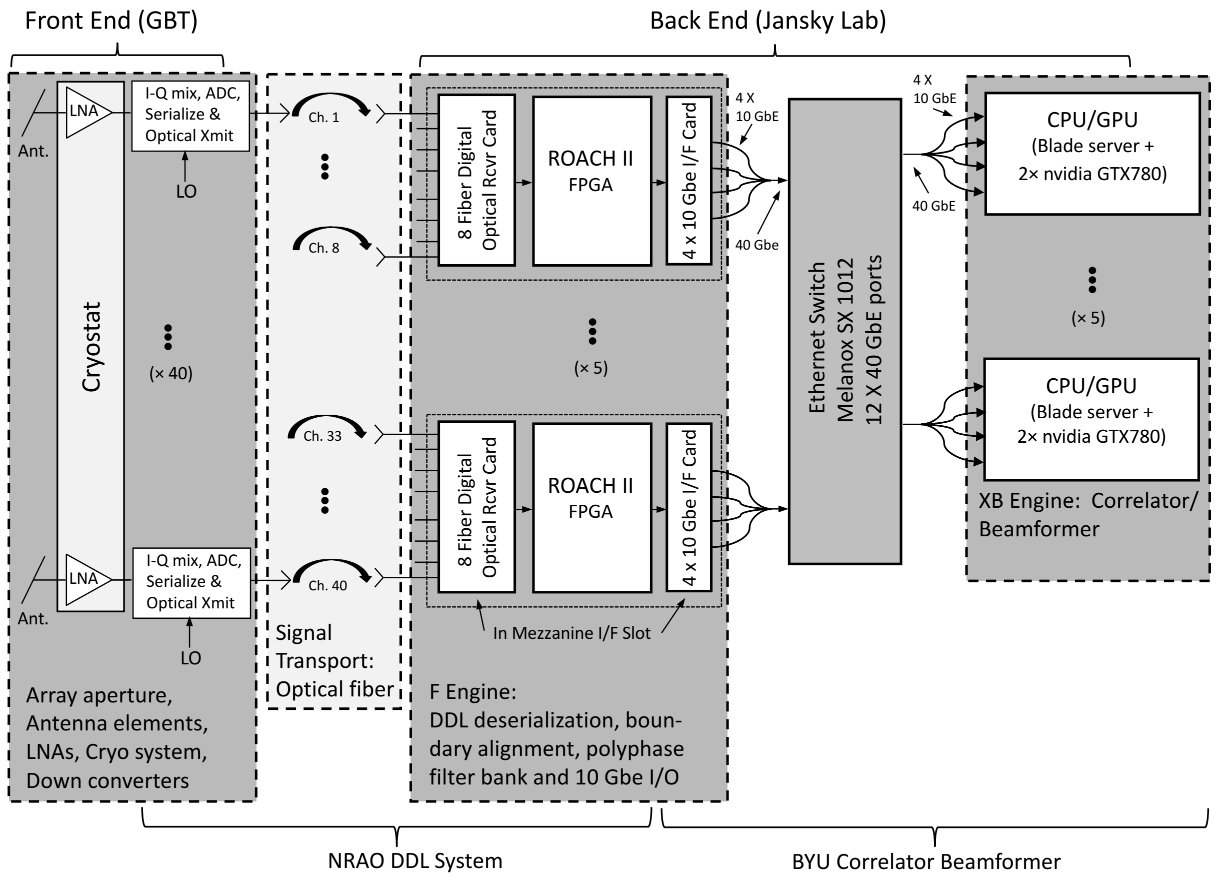

Front End (GBT)
Back End (Jansky Lab)
Ant.
LNA
I-Q mix, ADC, Serialize & Optical Xmit
LO
Cryostat
(× 40)
Ch. 1
Ch. 8
Ch. 33
Ch. 40
8 Fiber Digital Optical Rcvr Card
ROACH II FPGA
4 x 10 Gbe I/F Card
(× 5)
In Mezzanine I/F Slot
4 X 10 GbE
40 Gbe
Ethernet Switch Melanox SX 1012 12 X 40 GbE ports
40 GbE
CPU/GPU (Blade server + 2× nvidia GTX780)
XB Engine: Correlator/ Beamformer
Array aperture, Antenna elements, LNAs, Cryo system, Down converters
Signal Transport: Optical fiber
F Engine: DDL deserialization, boundary alignment, polyphase filter bank and 10 Gbe I/O
NRAO DDL System
BYU Correlator Beamformer

Figure 2.1: Block diagram for FLAG detailing a high-level system architecture and signal chain.

optical fiber (about 1.5 km) to the back end processing system located in the Jansky Lab telescope control building. This integrated system will be referred to as a blade and provides the first data downlink for the GBT that digitizes the analog signals at the front-end receiver [42]. A single blade processes eight analog inputs, requiring five blade systems to process all 40 inputs. The novelty of the blade architecture is that it provides an unformatted digital data sequence, requiring that the back end processing system perform boundary and sample alignment on the binary data in order discriminate between real and imaginary data across all elements. For the purpose of development, this process has been called "bit, byte, and word lock," and the algorithms and their implementation in the back end are presented in Appendix B.

After transmission from the GBT over optical fiber to the Jansky Lab, the signals are received by the back end processing system. The back end system consists of a network of five Re-

Table 2.1: FLAG HPC specifications

Chassis	Mercury AH-GPU408-SB14 4U
CPU	Intel Xeon E5-2630 v2 2.6 GHz
Memory	32 GB DDR3 ECC
GPU_*	NVIDIA GTX 980 Ti
NIC_1	2x Mellanox MCX 312A-XCBT Dual-Port 10 GbE
NIC_2	Mellanox MCX353A-QCBT InfiniBand Card
Hard drives	2x 500 GB SATA 7200 RPM

* One HPC has a 980 Ti GPU
^{1} Data offload from network
^{2} Data write to Lustre

configurable Open Architecture Computing Hardware version 2 (ROACH II or ROACH) boards, a Mellanox SX 1012 Ethernet Switch, and five HPCs. The ROACH boards were designed and manufactured by CASPER and are a CPU and FPGA platform for signal processing in RA [32].

The five HPCs, or processing nodes, are Super Micro model Mercury GPU408 4U GPU Servers. Each node has two Intel Xeon E5 2.6 GHz six core processors, with 32 GB of DDR3 ECC memory, and two CUDA capable NVIDIA GTX 980 Ti GPUs. For network connectivity, the nodes are each equipped with two dual-port 10-GbE Mellanox adapters to receive data over a switch. Processed data is written to a network Lustre file system in a FITS (Flexible Image Transport System) file format using a Mellanox InfiniBand adapter. A summary hardware description for an HPC is found in Table 2.1. Figure 2.2 shows both the ROACH chassis and HPCs as currently set up in the taperoom at the Jansky Lab.

Each ROACH board has been equipped with a custom mezzanine optical receiver card to process eight inputs from the blade fiber downlinks. A polyphase filter bank (PFB) is implemented in the FPGA and channelizes the 155.52 MHz signals into 512 frequency channels resulting in 303.75 kHz per channel. The PFB is also referred to as the F-Engine or coarse channel PFB to distinguish it from the fine channel PFB implemented on the HPCs for HI observations. Twelve of the 512 frequency channels are discarded resulting in 500 channels spanning 150 MHz of total instantaneous bandwidth. This is done to discard band edges corrupted by anti-aliasing filtering and to divide frequency channels evenly across the five processing nodes.

Figure 2.2: Back end HPCs (left) and Roach chassis assembly (right).

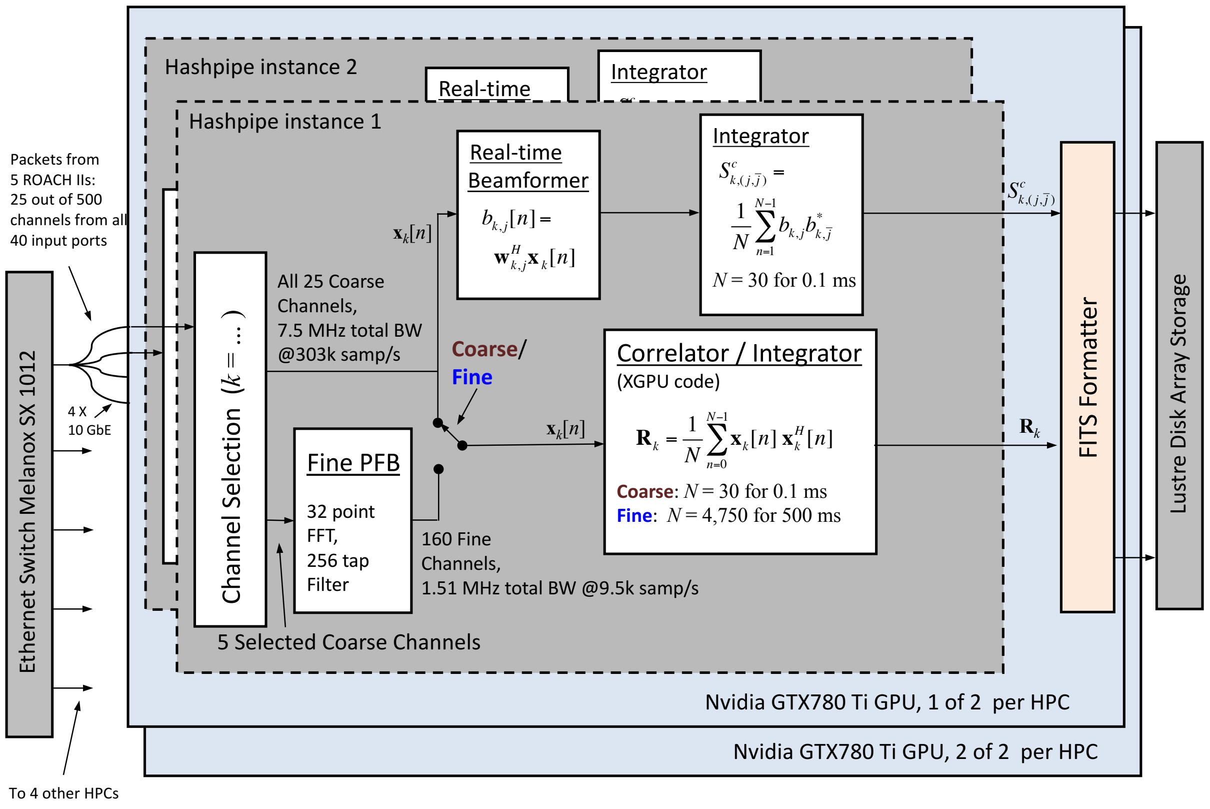
Each ROACH packetizes the 500 frequency channels into 20 User Datagram Protocol (UDP) packets and uses the four 10-GbE network interface ports to distribute them to all five nodes over a 40-GbE network. With each node having four 10-GbE ports, there are 20 unique destination IP addresses across the network. Each UDP packet is destined for one of the 20 IP addresses and the packet payload contains 20 decimated time samples of 25 frequency channels for eight element inputs. Therefore, each ROACH sends one packet to all 20 node ports, and each port receives five packets with 25 frequency channels from 8 different element inputs. This results in each node processing the same bandwidth for all 40 inputs. With four ports per node, and 25 channels per port, each node processes 100 frequency channels equal to 303.375 MHz of bandwidth.

The HPCs of the digital back end implement a suite of software programmed in Python, C, and CUDA for all critical data acquisition and processing operations. In order to simplify the current discussion and focus on the design of FLAG and its performance as a scientific instrument, only a brief explanation of the software is presented here as background. A more detailed discussion of the design, software contributions, and implementation are presented in Appendix A.

Data acquisition (setting observation parameters, recording data timestamps, activating the ROACHs) is managed by a Python wrapper developed to be compatible with the already established GBT telescope control systems. This wrapper manages and launches the real-time data processing pipeline known as Hashpipe [43]. Hashpipe is a framework for data processing which divides all processing operations (capturing packets from the network, correlation, beamforming, etc.) into different threads and handles system level functionality such as setting up shared memory segments and semaphore arrays for shared memory between processing threads.

In our implementation, the downstream threads which produce the correlation outputs and time-averaged beamformed spectra are implemented in CUDA, a parallel computing platform developed by NVIDIA [44]. CUDA enables high data throughput by performing parallel calculations using the large number of compute cores on a GPU. By implementing in CUDA the fine channelization, correlation, and beamforming, this allows for the processing nodes to accommodate the large data rate and keep up with I/O requirements. These operations are implemented as standalone GPU libraries, and each library has been programmed and compiled to be C compatible and interfaced by a single function call.

Each network interface port on a node is designated as the entry point for a single Hashpipe instance. There are therefore four Hashpipe instances per node, each assigned to process 25 coarse frequency channels from all 40 elements. Figure 2.3 shows a block diagram of the four Hashpipe instances and data processing chain on the GPUs. This figure does not show how Hashpipe manages scheduling or data transfer, rather just the operations performed (Appendix A.7). A Hashpipe instance is capable of producing spatial covariance matrices and time-averaged beamformed spectra for all of its 25 local frequency channels, and formats the output data into FITS files to be saved to a Lustre file system. A system that performs correlation or beamforming is often referred to as an X or B-Engine respectively. This processing pipeline can be thought of as an XB-Engine because it is capable of performing both operations. There are two Hashpipe instances, or XB-Engines, per GPU for high data throughput capability and to manage the large I/O requirements.



Hashpipe instance 2
Real-time
Integrator
Hashpipe instance 1
Real-time Beamformer
$b_{k,j}[n] = \mathbf{w}^H_{k,j}\mathbf{x}_k[n]$
Integrator
$S^c_{k,(j,\bar{j})} = \frac{1}{N}\sum_{n=1}^{N-1} b_{k,j} b^*_{k,\bar{j}}$
$N = 30$ for 0.1 ms
$S^c_{k,(j,\bar{j})}$
Packets from 5 ROACH IIs: 25 out of 500 channels from all 40 input ports
Ethernet Switch Melanox SX 1012
4 X 10 GbE
To 4 other HPCs
Channel Selection ($k = \ldots$)
All 25 Coarse Channels, 7.5 MHz total BW @303k samp/s
$\mathbf{x}_k[n]$
Coarse/ Fine
Fine PFB
32 point FFT, 256 tap Filter
160 Fine Channels, 1.51 MHz total BW @9.5k samp/s
5 Selected Coarse Channels
Correlator / Integrator
(XGPU code)
$\mathbf{R}_k = \frac{1}{N}\sum_{n=0}^{N-1}\mathbf{x}_k[n]\,\mathbf{x}^H_k[n]$
Coarse: $N = 30$ for 0.1 ms
Fine: $N = 4{,}750$ for 500 ms
$\mathbf{R}_k$
FITS Formatter
Lustre Disk Array Storage
Nvidia GTX780 Ti GPU, 1 of 2 per HPC
Nvidia GTX780 Ti GPU, 2 of 2 per HPC

Figure 2.3: Block diagram detailing the real-time processing in a single HPC.

After processing, the pipelines use a FITS file formatter code to configure the data, and are continuously writing to a Lustre file system for the duration of processing. The raw covariances and beamformed spectra can then be used offline in post-processing for image formation, spectral analysis, and post-correlation beamforming.

There are two principle modes of operation for observations: coarse channel and fine channel correlation. The real-time beamformer runs concurrently with either mode.

Coarse channel mode produces covariance matrices used for PAF calibration and the detection and analysis of transients, such as pulsars or fast radio bursts (FRBs). The correlator has the capability to set a flexible, variable integration length with a minimum dump time of 0.1 ms. A dump time on this short time scale is necessary for the detection of pulsars with a rotational period of 1-10 milliseconds, and for transient surveys. Integration lengths can also be arbitrarily long for weak source detection (non transient) or calibration and the calculation of weights for the real-time beamformer.

Prior to correlation, fine channel mode implements an additional PFB referred to as the fine PFB to set it apart from the coarse PFB which supplies the initial 500 frequency channels. Per pipeline, the fine PFB uses only five of the 25 coarse 303.75 kHz-wide input channels, and further channelizes them by implementing a 32-point PFB, resulting in 160 9.49 kHz-wide fine frequency channels. Using five coarse frequency channels in each pipeline across all nodes results in 30.375 MHz of total fine-channel bandwidth. Fine channel mode is often referred to as HI mode because its primary purpose is to perform HI surveys and maps. The minimum integration length for the fine channel correlator is 500 ms. The PFB method is discussed in more detail in Chapter 3 and the implementation of the fine PFB on GPU is covered in Chapter 4.

With either coarse or fine correlation modes running, the real-time beamformer can be concurrently running to provide time-average beamformed spectra. There are 7 dual-polarized beams formed for each of the 25 input coarse channels and power accumulated in time for a dump every 0.13 ms. This operation of the beamformer in parallel with a correlator is referred to as commensal operation.[1] The idea is that a target science HI observation could be in progress with the beamformer providing serendipitous broadband, short time integration, transient detection capability at the same time.

2.2 Performance Tuning, and Improvements

The FLAG digital back end system has a demanding I/O requirement, and the ability for the hardware and software to accommodate the high data rate is critical to its success. This section presents a framework and procedure for identifying and fixing potential bottlenecks, addresses the current performance of the FLAG HPC data pipeline, and how that framework has improved performance so far. Appendix A.9 should be considered a supplement to this discussion because this section characterizes performance generally, while the appendix identifies specific implementation details in the GPU and Hashpipe software which will improve overall system performance.

[1]Commensal operation is a specified operational requirement not yet demonstrated due to resource limitations in the current software implementation. Software modifications to better allocate and use resources are being made.

To assess performance issues, one must be able to identify possible bottlenecks in critical processing tasks and to allocate hardware resources efficiently. The critical functions performed on each node are:

1. Capture network data packets.

2. Transpose the data for processing.

3. Process data on a GPU.

4. Write the data from memory to disk.

Recognizing that each of these tasks is run as a different thread as a Hashpipe instance, and with four instances per node, this becomes a critical issue if resources are not managed properly to schedule tasks for all threads cooperatively. This requires understanding the CPU hardware architecture. Each FLAG node has two Intel Xeon 6 core processors with a motherboard which uses a non-uniform memory access (NUMA) architecture. The NUMA architecture divides resources such as memory regions, peripherals (including the PCIe bus), and interrupts to be associated more closely with a specific CPU socket than with another. These regions are called NUMA nodes and performance is best when data does not have to cross a bus to a different NUMA node.

To prevent this bottleneck, FLAG divides the four instances and two GPUs such that two Hashpipe instances are associated with a single GPU running on the same NUMA node. Hashpipe makes it really easy to specify core affinities of each thread, and various Linux system tools can be used to identify NUMA nodes. At this stage in the development process, properly assigning core affinities has made the greater contribution to improving system performance.

There are still minor issues however which prevent FLAG from being fully capable of the high data rate. Commensal mode is also not able to be supported with current implementations. At this point, with the four core processing tasks optimized to work together, it is a matter of iterating over each core task and identifying remaining bottlenecks.

Of the four processing tasks, we observed that the most time consuming operations were, first, the transpose operation which rearranges the data in the CPU into a specific data format prior to being copied onto the GPU, and second, the GPU libraries had too much latency. Each library must operate with a latency matching the requirements of that mode. We first optimized the GPU libraries using the kernel profiler tool in the CUDA integrated development environment (IDE) Nsight to meet latency requirements.

We then observed that with optimizations to the GPU libraries the fine channel correlator was operating successfully and able to manage all 20 Hashpipe instances for several observations. Intermittently, there will be an instance which hangs, suggesting there are still race conditions which have yet to be resolved. The real-time beamformer mode was still having many instances stall and this was due to the previously unresolved problem of the large amount of time taken to transpose the data in the CPU. We decided to implement the data transpose specific to the beamformer in the GPU, thus eliminating the transpose thread. With this change, we experienced a major improvement and only about two or three of the 20 Hashpipe instances will intermittently stall during observations.

Again, the system is not capable of the full data rate and commensal mode is still not operational. However, this does not preclude us from gathering sufficient data to characterize the system. Further improvements are required though to continue to identify other bottlenecks in each of the core processing tasks and determine how they can be implemented more efficiently. For example, although core affinities are being assigned efficiently, the current number of threads required to be shared by each core in a mode is not sustainable. Implementing the data transpose for the real-time beamformer in the GPU rather than the CPU eliminated a thread on the overburdened CPU and moved processing to the under utilized GPU. Improvements like this can be made to modify the other GPU libraries to incorporate similar behavior, better utilizing all computing resources. Other bottlenecks and specific implementation details are the subject of Appendix A.9.

2.3 Commissioning Results

The FLAG system, comprised of the PAF, front-end electronics, and digital back end has undergone three commissioning experiments. These experiments were split up over a year period with the first being July 2016, and the remaining two in May and August 2017. This thesis presents more details on the May and August 2017 experiments with an emphasis on results for overall system performance and the fine channel correlator.

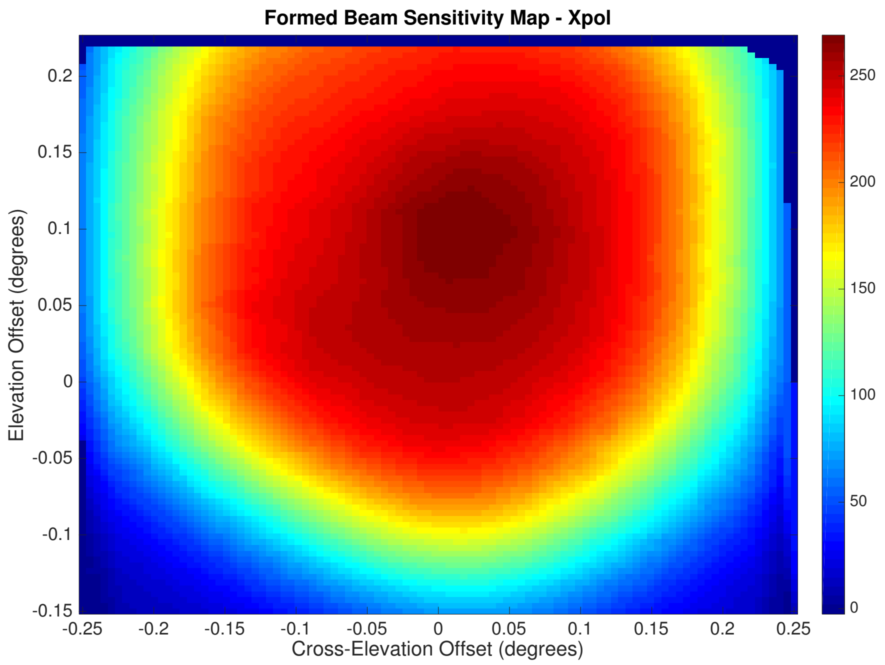


Figure 2.4: Sensitivity map for calibration grid on source 3C 295.

2.3.1 May 2017

In the first commissioning experiment, July 2016, only two of the five FLAG HPCs were installed. They provided approximately 45 of the 150 MHz total bandwidth and only the coarse channel correlator was used for significant testing. The PAF had also experienced some instabilities with the LNAs, and there were issues with back end correctly detecting digital data sample boundaries because of the unformatted data stream (Appendix B). This resulted in many lost elements.

After almost a year of development addressing these issues, the PAF had been stabilized with the remaining HPCs deployed and other back end operational modes ready for testing. This was the first commissioning with a stable PAF and the ability to observe over the full bandwidth, therefore many of the observations and tests were done to characterize the system.

One of the most important metrics used to characterize the performance of a PAF receiver system is the beamformed sensitivity. Sensitivity for a radio telescope is a measure related to the weakest source that can be detected. A measured sensitivity is important information for astronomers in order to determine the science observations which can be performed with FLAG.

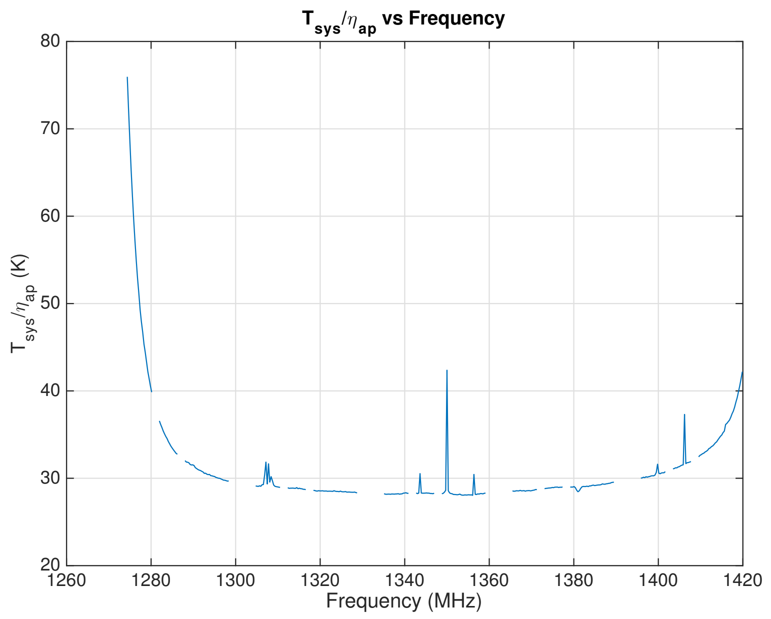


Figure 2.5: Beamformed spectrum for best measured T_{sys}/η during May 2017 commissioning.

Beamformed sensitivity is computed for channel k, and 2-D steering angle θ, according to

$$\begin{aligned} S_k(\theta) &= \frac{\eta A_p}{T_{\text{sys}}} \\ &= \frac{2k_B}{10^{-26}\Phi_k}\text{SNR}_k(\theta) \\ &= \frac{2k_B}{10^{-26}\Phi_k}\frac{\mathbf{w}_k^H(\theta)(\mathbf{R}_{\text{on},k}(\theta)-\mathbf{R}_{\text{off,k}})\mathbf{w}_k(\theta)}{\mathbf{w}_k^H(\theta)\mathbf{R}_{\text{off,k}}\mathbf{w}_k(\theta)}, \end{aligned} \tag{2.1}$$

where k_B is the Boltzmann constant, Φ_k is the flux density of the calibrator source in Jy (1 Jy $= 10^{-26}$ W/m^2) at the center frequency of channel k. The factor of two accounts for only being able to receive half of the total radiation due to polarization, $\mathbf{w}_k^H(\theta)$ is the maximum-SNR beamformer weight vector for beam pointing direction θ, $\mathbf{R}_{\text{on},k}(\theta)$ is the measured on-source correlation for the calibrator, and $\mathbf{R}_{\text{off},k}$ is a reference off pointing to obtain a noise-only correlation matrix. From sensitivity the quantity T_{sys}/η can be determined.

Figure 2.4 shows a plot of the X-polarized beamformed sensitivity grid from the third session's calibration grid on source 3C 295 at 1404.74 MHz. The best measured T_{sys}/η for this grid was 29.17 K, with a peak sensitivity of 269.2 K/m^2. The figure also shows that there was a pointing offset in the telescope position model for session three. This offset was a persistent issue during this commissioning experiment which we were not able to resolve prior to the last session. The issue however did not preclude us from obtaining other useful data and fully characterizing the system.

Over the course of the commissioning we performed several calibrations and on/off scans using the coarse channel correlator to measure sensitivity and T_{sys}/η. The best T_{sys}/η measured to be 28 K with the LO center frequency at 1350 MHz using calibrator 3C 48. A plot of the X-polarized beamformed spectrum is shown in Figure 2.5. There are breaks in frequency coverage because during this test the power distribution unit for one of the HPCs failed and we were left operating with only 120 MHz total bandwidth.

We were able to test the fine channel HI correlator but due to a bug in the transpose code that would index and write data to incorrect memory locations, all data gathered in this mode was corrupted. Prior to commissioning, we had performed several unit test cases to guarantee data integrity, however, we made a last minute software change to improve performance of the HI correlator without re-testing. Figure 2.6 shows how the data had been corrupted for an observation of the extended hydrogen source M51 using the HI correlator. This could have been corrected during commissioning, but we were unaware of the problem until post-processing shortly after the experiment ended.

The results of the real-time beamformer are not discussed in detail in this work, but to report on its performance as a functioning mode in the back end, it did experience several issues where, during many of the observations, up to half of the Hashpipe instances would stall. Despite the lost bandwidth, we were still able to detect a pulsar. The HI correlator also exhibited this behavior where several instances would stall because the process could not manage the I/O requirement. There were several problems discovered during this commissioning experiment, such as the stalling pipelines, indexing issue in the HI correlator and, digital sample word alignment across all 40 elements (word lock), but all of which would be resolved prior to the final commissioning of August 2017. Overall, the performance of the system during this commissioning experiment allowed for characterization of the system, with great results and a lot of success.

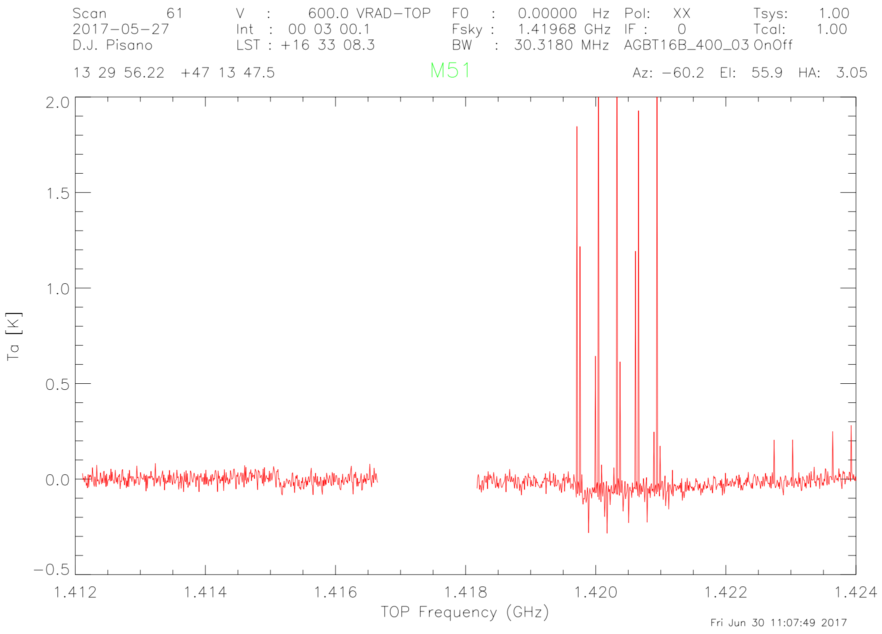


Figure 2.6: The plot shows that fine channel correlation was not functioning correctly in the May 2017 commissioning.

2.3.2 August 2017

During the months of June and July we were able to fix several issues with the Hashpipe codes and the indexing error introduced in the HI correlator. We also were able to perform tests with the HI correlator mode in the OTF and is discussed in Section 4.3. During this brief development time we begun to identify bottlenecks in the system and make adjustments as explained in Section 2.2 in order to improve the performance of Hashpipe, and prevent the loss of data. The major adjustments we made for this commissioning were: optimize the core affinity assignment for the Hashpipe threads in each operational mode, minimize GPU library latency, tune the Hashpipe data buffer block size, and implement the data transpose that is done prior to beamforming on the CPU as part of the beamformer GPU library.

With these changes, we saw major improvements in the performance of the system for each mode. The real-time beamformer still has, at most, three stalling Hashpipe instances, but both the coarse and fine channel correlator were able to operate without stalling under the demands of the

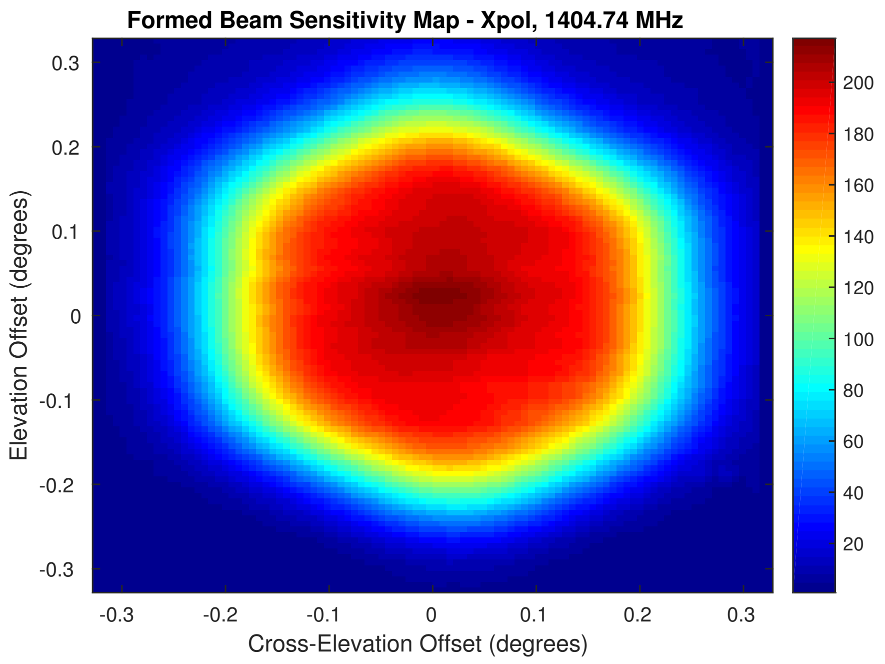


Figure 2.7: Full sensitivity map using an extended calibration grid. Calibration source is 3C 295.

data rates. This provided more productive sessions because we did not need to repeat scans because of lost data from stalled Hashpipe instances.

For system characterization we again acquired a few single on/off scans for T_{sys}/η and computed the sensitivity using the calibration grid from each session. Figure 2.7 shows the measured X-polarized beamformed sensitivity from an extended calibration grid intended to better sample the full map. The peak sensitivity for this commissioning was 234.2 W/m^2 with the lowest measured T_{sys}/η being 33 K. A plot for the X-polarized beamformed spectrum measuring this T_{sys}/η is shown in Figure 2.8. With a lower sensitivity and higher T_{sys}/η this has created discrepancies with the results from the previous commissioning. To this point, we have been unsuccessful in accounting for the increase in system temperature, but we believe it could be related to a phase or gain imbalance problem in the blades of the front-end electronics.

Having previously spent a lot of time characterizing the system, we wanted to begin testing the capabilities of the HI correlator once again. Figure 2.9 shows an X-polarized beamformed spectrum covering the full 30 MHz of the fine channel correlator mode during an observation on the extended galactic source NGC 6946. The previously encountered indexing problem had been

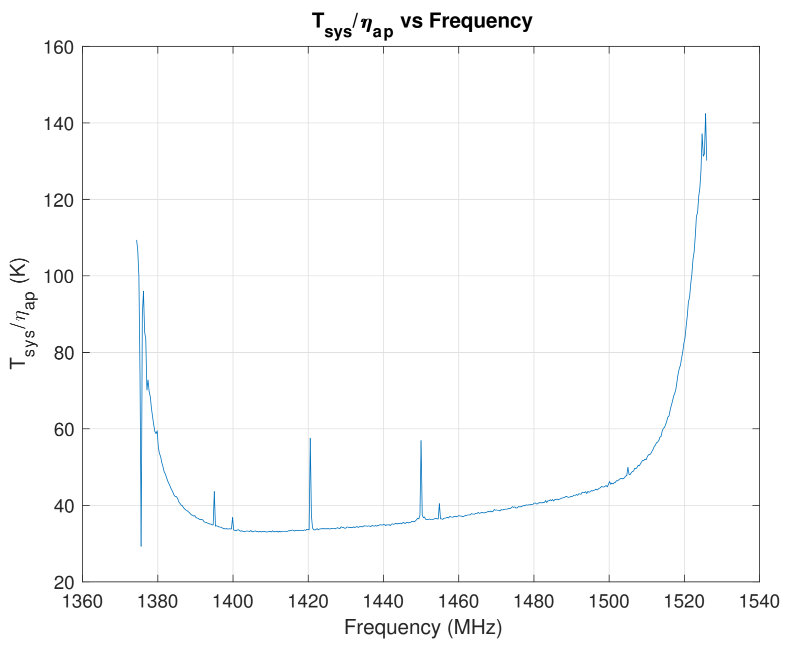


Figure 2.8: Beamformed spectrum for best measured T_{sys}/η during August 2017 commissioning.

resolved, making it now possible to begin performing science observations that can be used to compare FLAG with a traditional single horn receiver system.

One advantage that a PAF has over single horn receivers is the ability to form multiple beams and sample more locations in the sky, increasing the field of view (FOV) and thus decreasing the survey time needed. To demonstrate this we completed a 40-minute survey (which for a similar survey using the single horn feed would require about 2.5 hours) of the extended galactic source NGC 6946 and generated a HI map to compare with a similar map made using the prime focus L-band receiver on the GBT. The results of this survey, along with contours comparing FLAG to the prime focus receiver, are shown in the map of Figure 2.10. The red contours indicate detections made by FLAG and the white contours are for the prime focus receiver.

Differences between the contours can be attributed to the fact that the white contour overlaid on this map is using a data record of up to 80 hours of observation as compared to the 40 minutes of available data from the FLAG record. For example, a longer FLAG data record could reduce sample estimation error and smooth the contour approaching the prime focus receiver contour. Regardless, FLAG demonstrated superior spatial resolution and detection performance with a single pointing which is in good agreement with the historical data. Figure 2.11 shows another

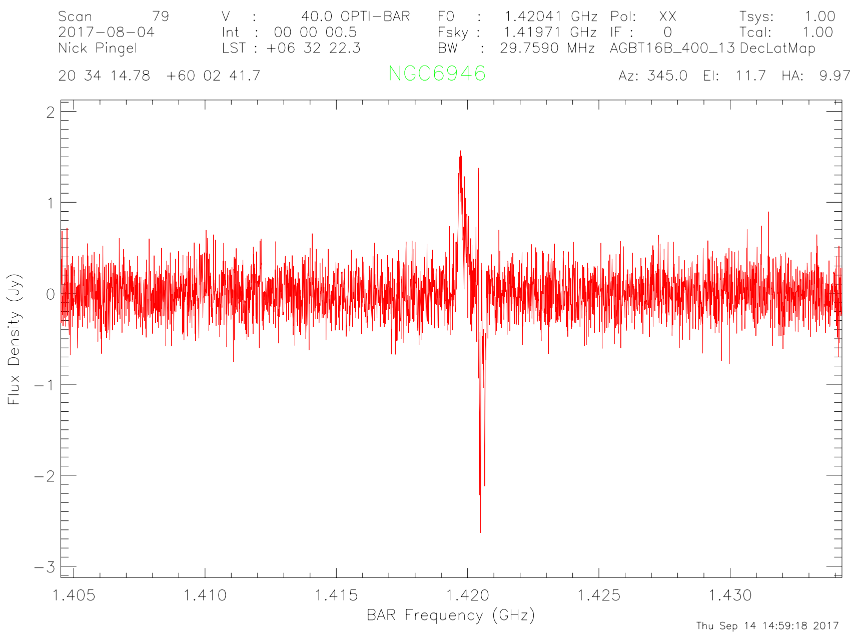


Figure 2.9: Beamformed spectrum during the observation of extended galactic source NGC 6946 using the fine channel correlator.

comparison of a FLAG beamformed spectrum using beam two for NGC 6946 and a similar survey performed in 2014 with the prime focus receiver [45]. Due to time constraints on telescope, our integration lengths were limited to half-second periods again accounting for the variations. There is again great agreement in the results and longer integrations would yield a better signal detection. Overall, further analysis of noise parameters and sensitivity need to be conducted to better characterize the comparison between FLAG and the prime focus receiver. However, with clear detections and spatial agreement in the FLAG data compared to the historical data, the conclusion can be made that FLAG is a viable scientific instrument capable of performing HI surveys.

2.4 Conclusion

This chapter presented the development and design of a real-time signal processing digital back end for a PAF capable of processing 40 antenna inputs with 150 MHz of instantaneous bandwidth. This back end computes 500 coarse channel correlations, or 3200 fine frequency channel

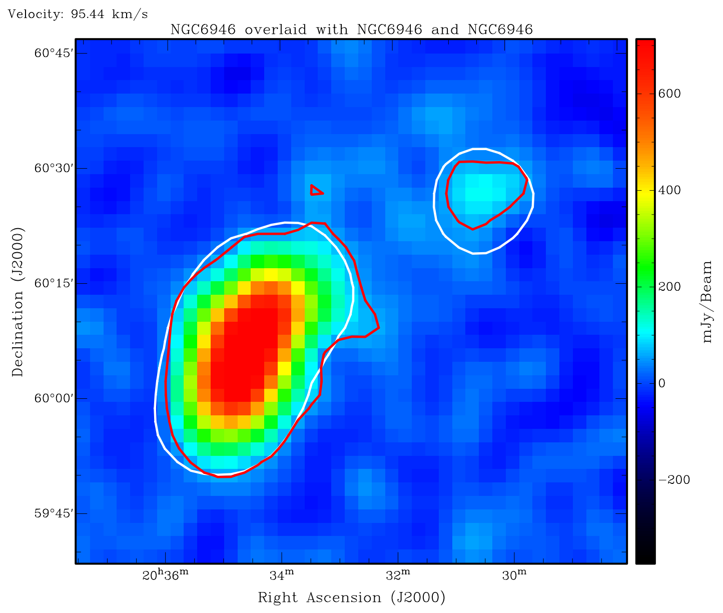


Figure 2.10: HI map for NGC 6946. The red contours identify detections made by FLAG and the white contours are for the prime focus L-band receiver for the GBT. The detection made with FLAG agrees quite well with that of the single pixel feed.

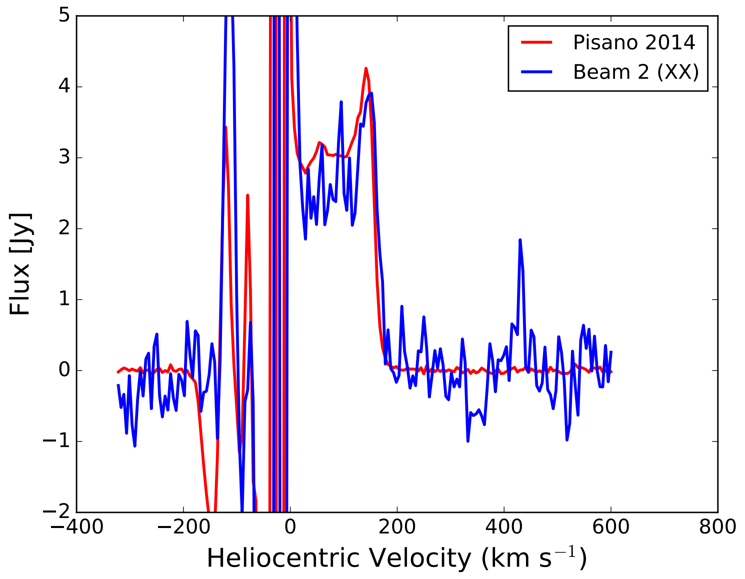


Figure 2.11: X-polarized beamformed spectrum using beam two for an observation of NGC 6946 compared with a previous survey of the same source in 2014 by the prime focus receiver.

correlations over a reduced bandwidth of 30 MHz. It is also capable of forming 7 real-time simultaneous dual-polarized beams with a dump rate of 0.13 ms for the 500 frequency channels. Commensal operation with the real-time beamformer and fine or coarse channel correlator currently has data rate limitations, but several bottlenecks have been identified with potential solutions presented. This system required integration of cutting-edge hardware and the development of a suite of software and algorithms to process data correctly and manage I/O requirements.

The FLAG system has undergone two successful commissioning experiments measuring an unprecedented T_{sys}/η of 28 K for a PAF. It has also demonstrated the capability of performing high fidelity, single-dish, HI surveys at a fraction of the time required for a single horn feed. In addition to the excellent sensitivity and promise that FLAG offers as a viable scientific instrument, it offers the ability for further PAF signal processing research such as real-time RFI mitigation for active observations using beamforming and spatial filtering techniques.

CHAPTER 3. THE POLYPHASE FILTER BANK METHOD

Many applications in digital signal processing depend on analyzing the frequency content of a signal. A simple example is that we might be interested in computing the power spectral density (PSD), $S_x(f)$, of a given time signal, $x(n)$, to examine the power present in a range of frequencies. This is an important application for RA because the spectral content of a source reveals important properties of radio signals that help astronomers determine the origin and nature of the signal. For example, a narrowband spectral line could be the result of HI (neutral hydrogen) emissions while a signal with continuum spectral content would be emissions from active galactic nuclei.

High performance real-time spectral analysis is often performed by a set of narrowband bandpass filters, known as a filter bank, rather than a fast Fourier transform (FFT). This is because a filter bank can be designed with less spectral leakage and better frequency channel crossover characteristics. Generally there are two classes of filter banks that are used to examine signals: analysis and synthesis filter banks. Analysis filter banks are designed to decompose the input signal, $x(n)$, into various components while synthesis filter banks combine multiple components together. The polyphase filter bank (PFB) is a computationally efficient way to implement a filter bank for spectral analysis [46]. The purpose of this chapter is to take a detailed pedagogical approach to derive the PFB method for efficient spectral analysis.

3.1 The DFT as a Filter Bank

Presented with the problem of performing spectral analysis, this can readily be achieved with the computation of the discrete Fourier transform (DFT) using an M-point FFT. The DFT of the signal $x(n)$ is given by

$$X(k) = \frac{1}{M}\sum_{n=0}^{M-1} x(n)e^{-jk\frac{2\pi}{M}n}, \quad k = 0, 1, \ldots, M-1, \tag{3.1}$$

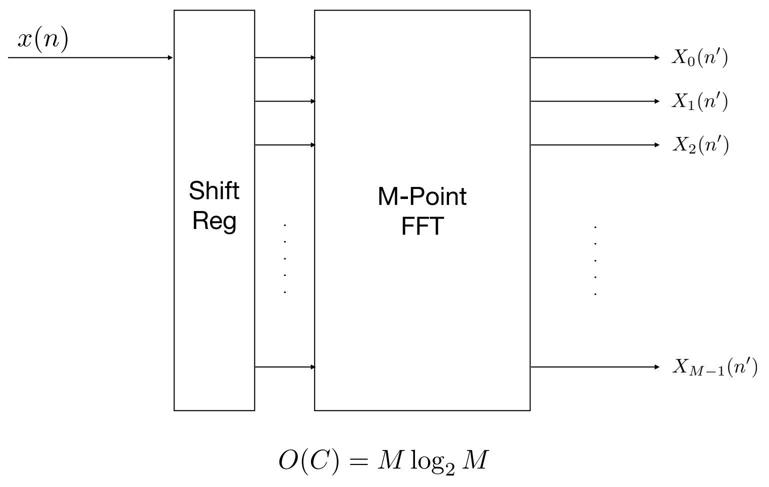


Figure 3.1: The FFT is an intuitive first approach to performing spectral analysis. A straightforward implementation would be to stream data and load a shift register that feeds an M-point FFT and a new sample for each bin occurs every Mth sample. The FFT is the most efficient algorithm for computing the DFT with computation complexity $O(C) = M\log_2 M$.

where M represents the size of the transform and is the number of frequency bins in the output, $X(k)$. The effective continuous-time frequency bandwidth of each bin is $\Delta f = f_s/M$ where f_s is the sample frequency. By repeating Equation (3.1) on successive M-sample long windows of $x(n)$, the DFT is therefore an analysis filter bank. By a similar argument, the inverse discrete Fourier transform (IDFT) of the channelized signal $X(k)$ will produce $x(n)$ and is considered a synthesis filter bank because it perfectly reconstructed $x(n)$ from its various components.

Implementation of the DFT for a spectral analysis application could be achieved as shown in Figure 3.1. The streaming samples of $x(n)$ are loaded into a shift register of length M, and when full, are the input into an M-point FFT. Compared to the original sample rate for $x(n)$, samples at the output of the FFT are decimated by M.

Computation complexity for a digital system or algorithm can be measured by the number of multiplies per operation. The direct DFT has complexity $O_{\text{DFT}}(C) = M^2$ while the FFT is the most efficient algorithm for computing the DFT with complexity $O(C) = M\log_2 M$. Using the FFT is a viable approach because spectral analysis is achieved as desired. However, despite being the most computationally efficient method for spectral analysis, closer examination shows that for many applications, there are issues that cannot be ignored.

Ideally, we want to compute the discrete-time Fourier transform (DTFT). This is not practical because digitally we are only able to compute a finite number of frequency samples of the DTFT using a finite number of time samples of $x(n)$. Consider then that the DFT can be written as

$$\begin{aligned} X(k) &= \sum_{n=0}^{M-1} x(n) e^{-jk\frac{2\pi}{M}n} \\ &= \sum_{n=-\infty}^{\infty} w_{\mathrm{R}}(n) x(n) e^{-jk\frac{2\pi}{M}n}, \end{aligned} \tag{3.2}$$

where $w_{\mathrm{R}}(n)$ is the rectangle or box function defined as

$$w_{\mathrm{R}}(n) = \begin{cases} 0 & n < 0, \\ 1 & 0 \le n \le M-1, \\ 0 & n > M-1. \end{cases} \tag{3.3}$$

This shows that the DFT is computed by applying a rectangular window to an infinite data sequence. Recognizing that the DFT is equivalent to a DTFT which is sampled M times over one period,

$$\begin{aligned} X(k) &= \mathscr{F}\left\{ w_{\mathrm{R}}(n) x(n) \right\} \\ &= \operatorname{sinc}(k) * \mathscr{F}\left\{ x(n) \right\}, \end{aligned} \tag{3.4}$$

and reveals the effect that the rectangular window has on the data sequence $x(n)$.

An intuitive explanation of Equation (3.4) is that in each frequency bin the DFT biases the perfect DTFT by convolving it with a sinc function. Because the sinc function has infinite support, this has the negative side effect that frequency content that would otherwise only be present at one frequency in the perfect DTFT of $x(n)$ is now spread, or leaked, across several bins. This is known as spectral leakage and an example of this can be seen in Figure 3.2 along with the normalized magnitude frequency response of the sinc function. In the magnitude frequency response the peaks that extend out from the center main lobe are called sidelobes.

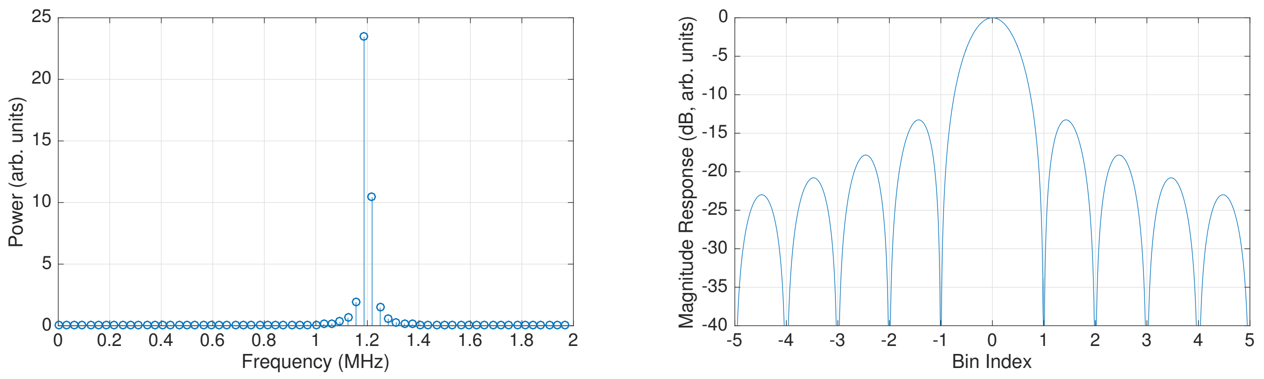


Figure 3.2: A 1.2 MHz complex tone is sampled at 2 MHz and a 64-point DFT is computed. The resulting power spectrum is shown (left). The DFT imposes a rectangular window on the data. The Fourier transform pair to the rectangular window is the sinc function. The single-bin frequency response of the sinc function (right) has a non-zero response outside the main lobe. This results in spectral leakage, energy of the complex tone being spread into adjacent bins.

Applying a window other than a rectangular window improves the response of the DFT by decreasing the sidelobe level, however, this comes at the cost of increasing the width of the main lobe at the bin center. At baseband, a window function is a finite impulse response (FIR) low-pass filter (LPF). Proper filter design is an important topic in digital signal processing. Specific details about window types, the design process, and trade-offs are not described here but can be found in more detail in [47], [48].

3.2 A Bank of Decimated Finite Impulse Response Low Pass Filters

It was shown in the last section that the DFT can be thought of as more than just a transformation from time to frequency. Instead, it can be thought of as a bank of M FIR filters downsampled by M. We can use this interpretation as a blueprint to design a filter bank that reduces spectral leakage and achieve a better frequency response.

A block diagram for this design is shown in Figure 3.3. The data sequence $x(n)$ is sent through a bank of processing chains consisting of multiplication by a complex exponential, followed by a LPF, and then downsampled by M. Each complex exponential has the general form, $e^{jk\frac{2\pi}{M}n}$, where $k = 0, 1, \ldots M-1$, and M is the number of processing chains and effective number of output bins similar to the length of a DFT transform. A multiplication in time is a shift or trans-

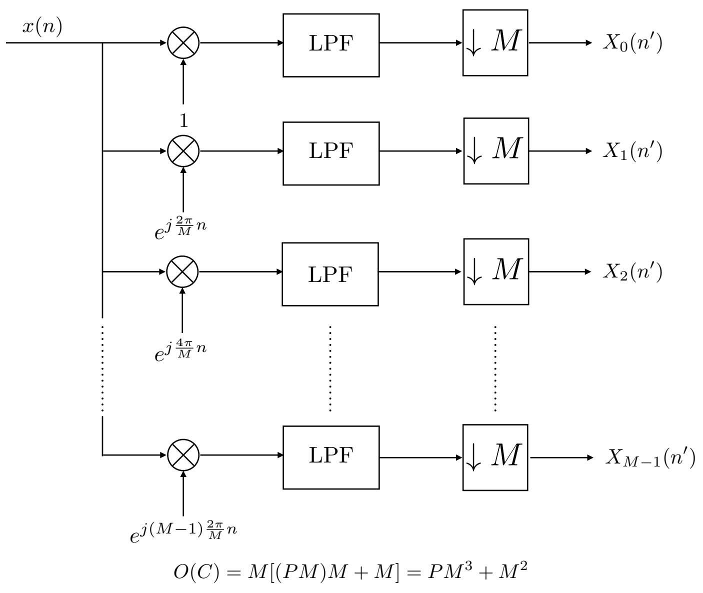


Figure 3.3: To improve the frequency response of the filter bank we design a prototype LPF such that the single-bin frequency response has lower sidelobe levels and a narrow transition band. The new design is a bank of FIR LPFs downsampled by M using complex multiplications to translate the LPF to evenly spaced bandpass locations over the frequency range of interest.

lation in the frequency domain, so the purpose of each multiplication is to translate the frequency response of the LPF to evenly spaced bin centers over the frequency range of interest.

In a spectral analysis application the general design requirements for the LPF are that the frequency response has low sidelobes, acceptable passband ripple level and that the transition band is relatively narrow (compared to the FFT filter bank) and overlap with adjacent bins at the -3 dB point. Overlap in the transition band guarantees a flat response across the full bandwidth of the filter bank. When the filter transition band is not designed properly this results in what is known as scalloping loss with an example shown in Figure 3.4. There are several window shapes and algorithms to choose from to achieve these requirements. However, in general to have low sidelobe levels the length of the filter will be larger than M. For now, we will assume that the LPF has L taps and is an arbitrary multiple of M such that, $L = PM$. We will show later that being an integer multiple of M is necessary for the PFB method.

In this new design, in order to get one set of outputs of the filter bank, there are LM multiplies at the output of each LPF, M multiplies from the complex exponential repeated on M

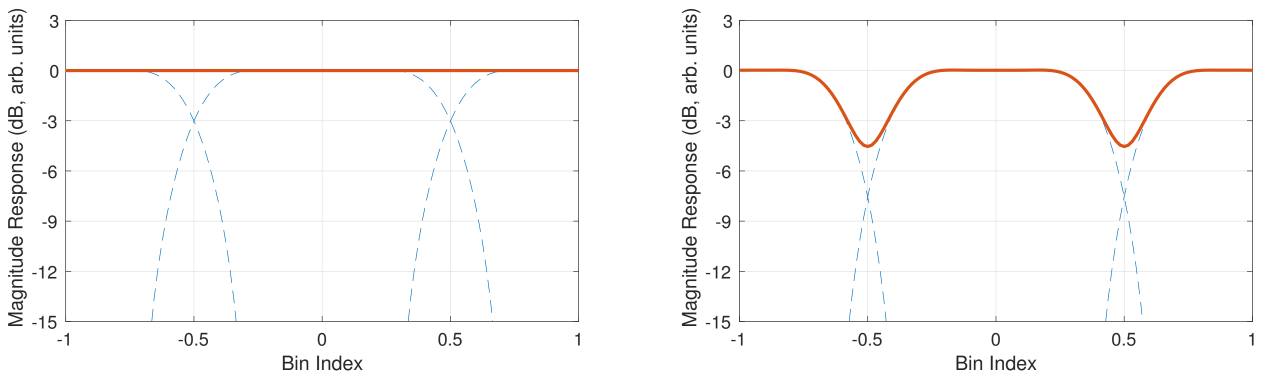


Figure 3.4: Filters used in filter banks for spectral analysis applications are designed such that the transition band will be relatively narrow but still overlap with adjacent bins at the -3 dB crossover point. This results in a uniform response across the passband of the filter bank (left). When the transition band is poorly designed this results in what has been called scalloping loss and results in attenuation to signals in the scalloping region (right).

branches. This results in complexity

$$\begin{aligned} O(C) &= M^2(L+1) \\ &= M^2(PM+1) \\ &= PM^3 + M^2. \end{aligned} \tag{3.5}$$

Computation complexity on the order of M^3 is unreasonable and impractical for most applications.

We can be more efficient with this architecture because after each LPF we are keeping only every Mth sample and discarding the rest. Computing only the multiplies that we need yields the decimated FIR filter. The output sequence of a length L FIR filter is

$$y(n) = \sum_{l=0}^{L-1} h(l)x(n-l), \tag{3.6}$$

and computing only every Mth sample,

$$y(n') = \sum_{l=0}^{L-1} h(l)x(n'M-l), \quad n' = \frac{n}{M}. \tag{3.7}$$

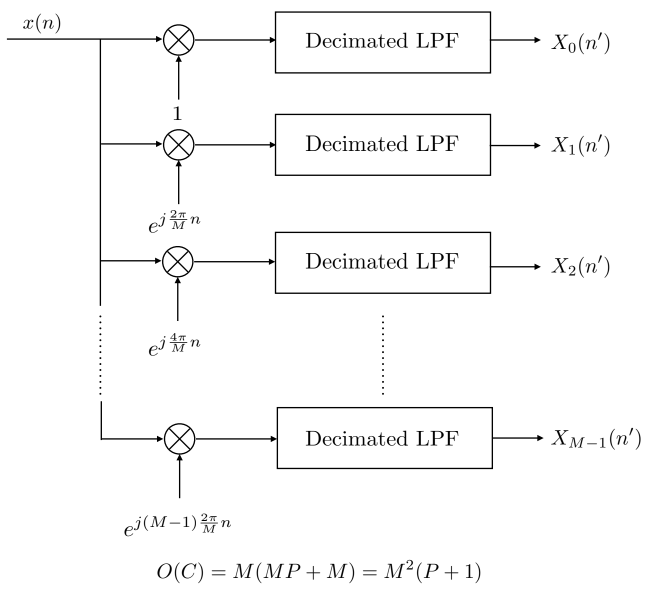


Figure 3.5: Using decimated FIR LPFs to only compute multiplications that are used at the output of the filter bank improves compute complexity.

Incorporating this change in the processing chain results in the block diagram shown in Figure 3.5. The number of multiplies at the output of each LPF has now decreased by a factor of M to L with $O(C) = M^2(P+1)$. This is a significant improvement and is similar to $O_{\text{DFT}}(C)$.

This implementation is a step in the right direction. A bank of decimated FIR LPFs achieves a better single-bin frequency response with complexity similar to the DFT. It would be easy to stop and have this be our architecture of choice. However, what we want is to have an algorithm that is just as efficient as the implementation of the FFT in Figure 3.1 but also has the advantages of windowed filter design. This is what the PFB method achieves. As the namesake suggests, to achieve better computational efficiency, the PFB takes advantage of polyphase decomposition, a topic of multirate signal processing.

3.3 Polyphase Decomposition

Given a length L filter impulse response $h(n)$, it can be decomposed into M subsequences or phases. Each subsequence, $h_i(n)$, contains every Mth sample from delayed versions of the

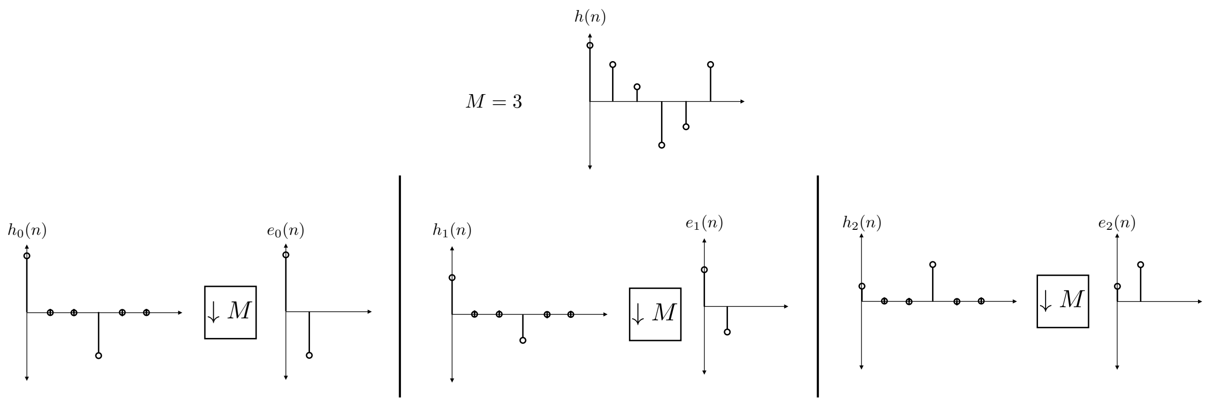


Figure 3.6: An example of polyphase decomposition. The impulse response $h(n)$ is decomposed into M subsequences or phases. The polyphase components $e_i(n)$ are downsampled sequences of $h_i(n)$.

sequence $h(n)$. The decomposition is expressed as

$$h_i(n) = \begin{cases} h(n+i), \quad n = iM, \quad i = 0, 1, \ldots, (P-1) \\ 0, \quad\quad\quad\quad\quad\quad \text{else,} \end{cases} \tag{3.8}$$

where $P = \frac{L}{M}$ and each subsequence has P filter coefficients [46], [48]. This implies that $L = PM$. When designing the LPF in the previous section we let L be an arbitrary multiple of the transform size. However, it turns out this condition was necessary to leverage polyphase decomposition.

Decimation of each subsequence $h_i(n)$ results in the polyphase components defined as

$$e_i(n) = h(nM + i) = h_i(nM), \tag{3.9}$$

and in the z-domain, the polyphase decomposition of $H(z)$ can be expressed in terms of its polyphase components $E_i(z)$,

$$H(z) = \sum_{i=0}^{M-1} E_i(z^M) z^{-i}. \tag{3.10}$$

An example of polyphase decomposition is shown in Figure 3.6.

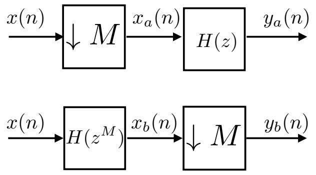


Figure 3.7: The interchange or Noble identity. Linear filtering and downsampling can be interchanged by making modifications to the filter.

3.4 The Noble Identity

Another characteristic of multirate systems that will prove useful in the derivation of the PFB is the interchange or Noble identity. A block diagram for the Noble identity is shown in Figure 3.7. Consider the signal $x(n)$ to be the input of both processing chains. It can be shown that

$$Y_a(e^{j2\pi f}) = Y_b(e^{j2\pi f}), \tag{3.11}$$

meaning the outputs of the two processing chains are equivalent [48]. The result of the Noble identity then is that the operations of linear filtering and downsampling (a similar identity can be proved for upsampling) can be interchanged with modifications to the filter.

3.5 The Polyphase Filter Bank

Applying a polyphase decomposition to each branch of the bank of FIR LPFs followed by decimation in Figure 3.3 results in the revised block diagram of Figure 3.8. Here we have used Equation (3.10) to express the LPF in the z-domain. This allows us to further manipulate the signal architecture that renders the PFB method.

Figure 3.9 details many of the steps taken in the derivation of the PFB. We start in subfigure (a) by considering the kth branch. The LPF is already represented as a polyphase decomposition where outputs of $H(z)$ are equal to $x(n)$ being delayed and filtered by the zero interpolated polyphase components $E_i(z)$, and then summed together. Next, in (b), downsampling is a linear operation and can be commuted inside the filter and across the sum. Doing so has placed the downsample block right after the filtering done by the polyphase components. Comparing each

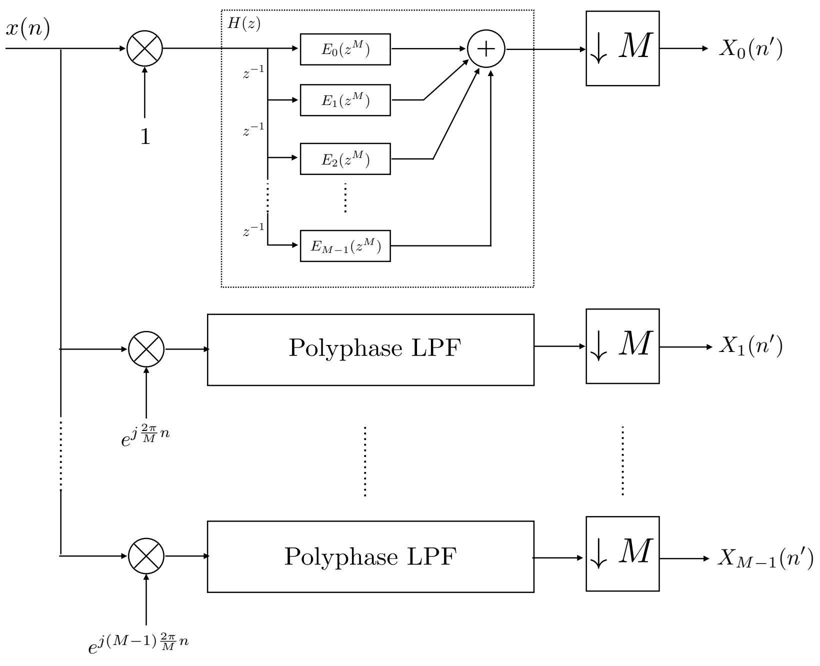


Figure 3.8: Applying a polyphase decomposition to each LPF in the bank of FIR LPFs followed by decimation is the starting point for deriving the PFB method. The impulse response, $h(n)$, of the LPF is expressed in the z-domain in terms of the polyphase components $E_i(z)$.

filter block followed by a downsample by M to the bottom processing chain of Figure 3.7 shows that the Noble identity can be used. In moving from (b) to (c), the Noble identity has been applied by modifying $E_i(z)$ to remove the zero interpolated samples and interchanging the filtering and downsample operation. Finally, in (d), we propagate the multiplication by $e^{jk\frac{2\pi}{M}n}$ through the processing chain. First, delays are inserted,

$$e^{jk\frac{2\pi}{M}n} * \delta(n-n_0) = e^{jk\frac{2\pi}{M}(n-n_0)},$$

where $n_0 = 0,1,\ldots,M-1$ and represents the M sample delays for each polyphase branch. Applying a downsample by M yields

$$e^{jk\frac{2\pi}{M}(n-n_0)}(\downarrow M) = e^{jk\frac{2\pi}{M}(nM-n_0)},$$

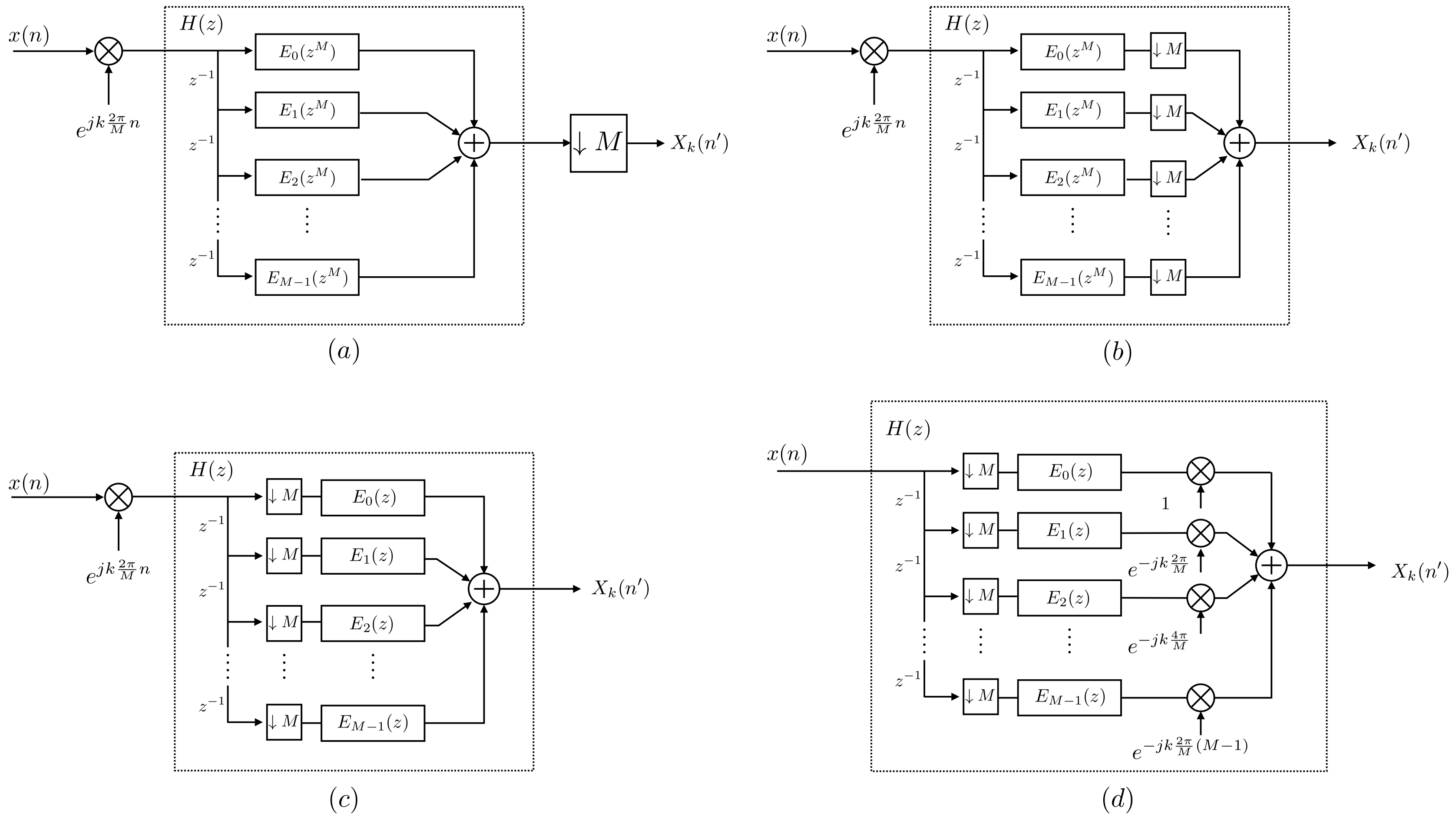


Figure 3.9: (a) Taking the *k*th branch in the bank of polyphase LPFs we can simplify the signal architecture. (b) Downsampling is a linear operation and can be commuted as part of $H(z)$. (c) Apply the Noble identity. (d) Propagate multiplication by a complex exponential through each processing chain.

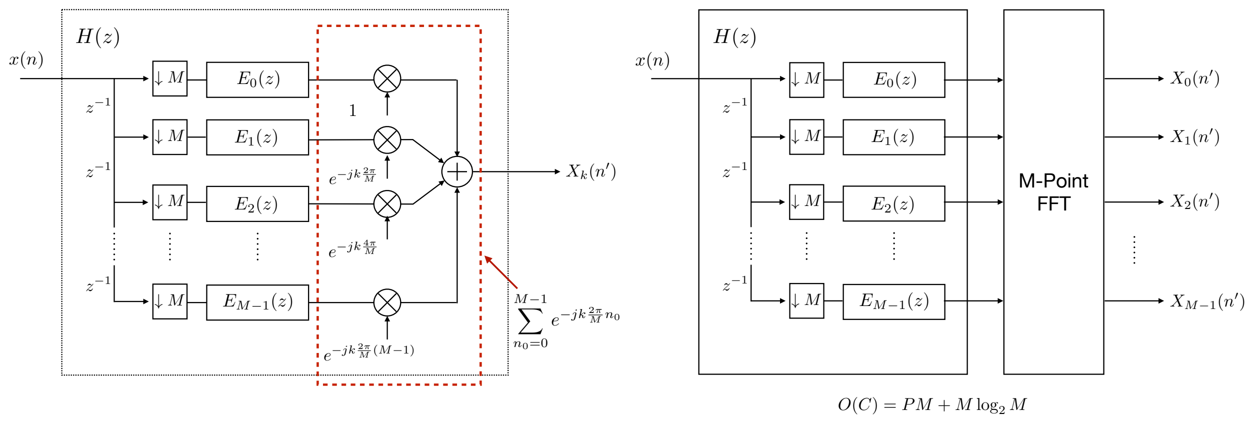


Figure 3.10: After simplifications to the signal architecture to the kth branch of a bank of polyphase filters, a DFT structure manifests itself (left). An M-point FFT is used with only one bank of polyphase filters as an efficient way to compute the multiplications across different bin index k. This is the PFB method with the resulting block diagram (right).

and can be simplified further by recognizing that the first term in the exponential is always a multiple of 2π,

$$\begin{aligned} e^{jk\frac{2\pi}{M}(nM-n_0)} &= e^{jk2\pi n} e^{-jk\frac{2\pi}{M}n_0} \\ &= e^{-jk\frac{2\pi}{M}n_0}. \end{aligned} \tag{3.12}$$

Multiplication commutes with the filtering operation and the multiplication ends up placed after the filter, and before the sum across each polyphase branch.

When examining the final form of the exponential as seen in Equation (3.12), the resulting processing chain for the kth branch of the filter bank has a sum over the M polyphase branches (highlighted in Figure 3.10). This is the same operation performed by the DFT, we can therefore use the FFT instead. We then only keep one polyphase decomposition of $h(n)$ from the bank of FIR LPFs in Figure 3.8, and the outputs from each of its polyphase filters become the input to an M-point FFT.

We have arrived at the final block diagram for the PFB and is shown in Figure 3.10. There are M polyphase filters from the polyphase decomposition of $h(n)$ each with P taps, followed by an M-point FFT, resulting in compute complexity $O(C) = PM + M\log_2 M$.

Table 3.1: Comparison of computational complexity for the various implementation of filter banks measured in multiples. M is the number frequency bins produced by the filter bank. The LPF is designed to have $L = PM$ taps where P is the number of polyphase components.

	FFT Implementation	Bank of Decimated FIR LPFs	Polyphase Filter Bank
$O(C)$ (multiplies)	$M \log_2 M$	$M^2(P+1)$	$PM + M \log_2 M$

3.6 Conclusion

This chapter presented a detailed derivation of the PFB method. It started by showing that signal analysis can be performed by using a conventional FFT implementation of the DFT but suffers from spectral leakage because the DFT applies a rectangular window to the data. With the desire to build a filter bank with filters that provide a better single-bin frequency response a straightforward implementation was to have a bank of FIR LPFs followed by a downsample by M. When the LPF is designed properly, this architecture proved to give a better frequency response, however, at a compute complexity that was impractical for most applications. Noticing that many of the multiplications were being discarded we used a decimated FIR LPF instead. This improved the complexity to order M^2, a computational burden similar to the direct DFT. Being motivated by complexity $M \log_2 M$ of an FFT implementation we designed a LPF that leverages polyphase decomposition and simplified the processing chain to yield the PFB. Compared to the FFT filter bank, the PFB method mitigates the effects of spectral leakage by using a properly designed filter with a moderate increase to computational complexity. A comparison of the complexity for the methods discussed is summarized in Table 3.1.

CHAPTER 4. GPU IMPLEMENTATION OF A PFB FOR FLAG

The use of graphical processing units (GPUs) in science and engineering research started to become popular in the mid to late 2000s when GPUs were redesigned to be a highly threaded streaming processor due to new programming models that extended parallel constructs in C [49]. Also, in 2006 NVIDIA announced the development of Compute Unified Device Architecture (CUDA), which is a software and hardware architecture that enables GPUs to be programmed in C/C++, both being higher level languages. This new highly parallel computing environment improved the number of floating point operations per second (FLOPS) making it an appealing platform for applications with very intensive I/O requirements.

Signal processing applications in RA are examples of the popularity of this approach because of the need to process multi-sensor array data efficiently at high data rates. At the Green Bank Observatory (GBO) alone, there are two other digital back end systems, besides FLAG, which are using GPUs for processing. These instruments are the Breakthrough Listen Digital Recorder (BLDR) [50] and the Versatile GBT Astronomical Spectrometer (VEGAS) [51].

It was shown in Chapter 2 that all of the signal processing for correlation, fine channelization and time-averaged beamformed spectra in the FLAG back end is done using GPUs. This is done in order to meet the data rate and I/O requirements of the data streaming from all 40 inputs. The purpose of this Chapter is to detail how we applied parallel programming constructs to develop the FLAG GPU PFB library for the fine channel correlator and present measured filter performance using simulated and real-data experiments prior to commissioning.

4.1 The Parallel Programing Paradigm

The CUDA hardware and software architecture exploits the power of the large number of compute cores in a GPU by taking advantage of data parallelism which is intrinsic to many large data processing problems. This is the basis for the parallel programming paradigm. An example which illustrates the concept of data parallelism is an algorithm where the outputs at any given time step depends only on the current inputs. A serial implementation of such an algorithm would process a single input and subsequently provide a single output at each time step. However, because the problem is formulated in such a way that there is no dependence on other data, all input data could be buffered in memory to be considered at once, thus providing all outputs simultaneously. When a problem may be structured in such a way which groups data and associated processing into independent paths, it is possible to leverage parallel operations. This is considered data parallelism.

Approaching problems in this way is what leads to the parallel programing paradigm and is an abstract model for both representing data and the implementation of CUDA code. In the following discussion the host always refers to the CPU and the device is the GPU. The device architecture includes its own memory which is independent of the host. The host can allocate and manage device memory, however, the device cannot modify host memory. The device only operates by directive from the host.

In this model, data parallelism is abstracted into a multi-dimensional data cube where the computing tasks that are run in parallel are called threads. A kernel is a device function that is executed by each thread and is the fundamental parallel operation. Kernels contain the essence of how the problem can leverage data parallelism. A collection of threads, capable of being indexed in three dimensions, make up a single block, and a three dimensional collection of blocks forms a grid. This thread-block-grid arrangement together comprises the multi-dimensional data cube. Figure 4.1 illustrates the CUDA compute model. The host launches the kernel with specified grid and block dimensions. The threads within each block begin running in parallel, executing the task specified by the kernel. More information and specifics about how the device schedules thread operation can be found in [44], [52].

A simple image processing example can illustrate how this model is used. Given a 512 x 512 RGB pixel image, suppose the task at hand is to convert the RGB image to a gray-scale image. The image is transformed to gray-scale by averaging individual pixel RGB components together. Notice that transformation of one pixel does not affect the adjacent pixel, therefore, instead of a serial implementation which averages components, pixel by pixel, we can use data parallelism and perform all averaging at once.

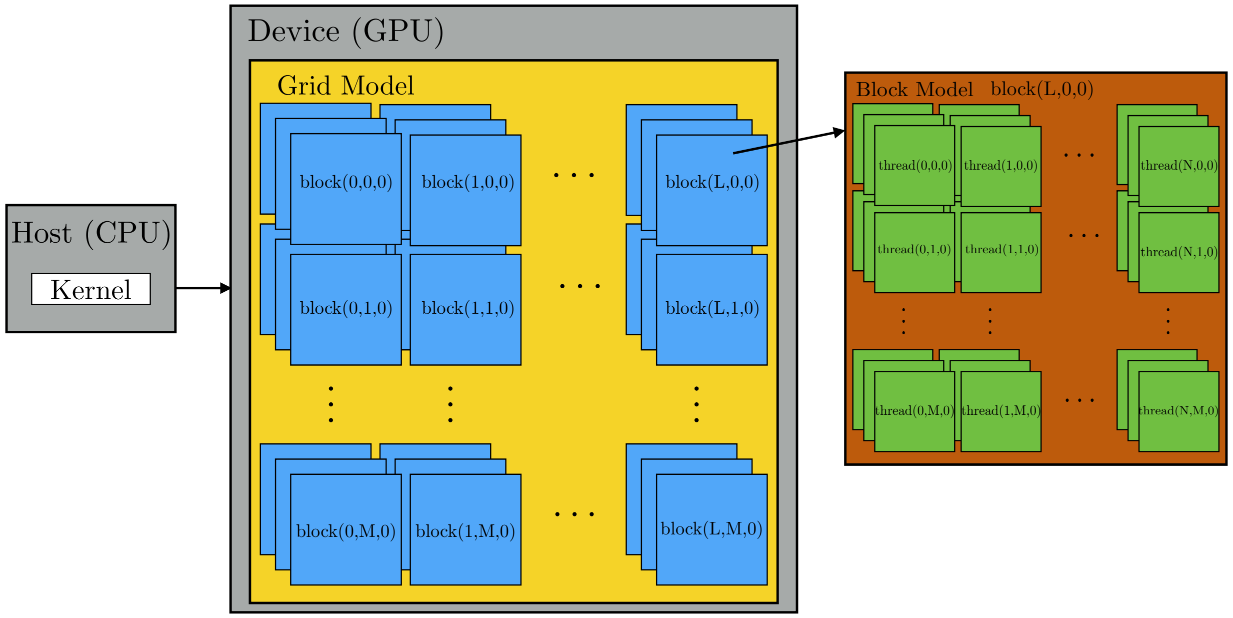
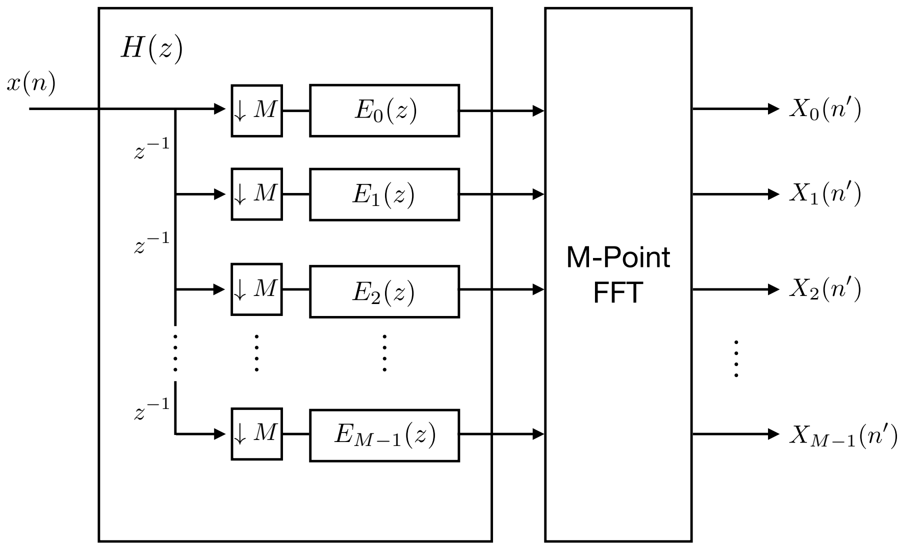


Figure 4.1: Illustration of data parallelism in CUDA.

The fundamental parallel operation is the RGB averaging, this computation is therefore specified as a kernel function. In the main function of the host, sufficient memory for the image is allocated on the device and subsequently loaded onto the device. The host code continues by configuring a 1 x 1 x 1 grid where that one block contains 512 x 512 x 1 threads, for a total of 262,144 threads. The host launches the kernel on the device with this grid and block configuration as a launch parameter and the kernel is then executed in all threads which begin operating in parallel. This was a simple example that did not require a large grid, but the concept abstracts as the image increases in size. Much of the work done in the parallel programming paradigm is to determine the kernel configuration which uses sufficient GPU resources while optimizing compute completion time or latency for the specified problem and data considerations.

4.2 The PFB GPU Library

Chapter 3 provided a derivation for the polyphase filter bank (PFB) method of frequency channelizing data, and noted that this approach reduces spectral leakage by applying windowed filter design to control the per-bin frequency response. A block diagram for the PFB is shown in Figure 4.2 for reference. The PFB consists of a bank of M polyphase filters, $e_m(n)$, which are the



$$O(C) = PM + M\log_2 M$$

Figure 4.2: Block diagram for the PFB

components of a polyphase decomposition of the low pass filter (LPF), $h(n)$. The outputs of those filters are fed into an M-point FFT. The LPF has L taps such that $L = PM$, where P is the number of taps in each polyphase filter, and M is the number of frequency channels that are produced at the output of the PFB.

A GPU implementation requires identifying and exploiting any data parallelism that presents itself. The most prominent parallel processing aspect that we leverage is the independent data sequences for each antenna element in the PAF. For FLAG, there are 40 antennas streaming time samples that are then channelized into 500 channels. In the fine channel correlator mode, only one-fifth of the 500 coarse frequency channels are processed for a zoomed spectrum. Each Hashpipe instance implementing a PFB across the five HPCs is therefore specified to process 5 coarse frequency channels for all 40 elements, using 4000 decimated time samples per data window as input. This results in 5 (coarse channels) x 40 (antenna elements), or 200 decimated time series that need to be independently filtered by a PFB. Data parallelism allows us to collapse the need for 200 iterations of a PFB into one call to a PFB kernel that runs all 200 iterations simultaneously. This is a powerful capability, because the number of decimated time series can scale and increase in dimension by either increasing the number of processed frequency bins, or antenna elements and all time series are processed simultaneously, substantially decreasing computation time. As long as there are sufficient GPU resources available to process all of the time series, the compute response time is as if only one PFB is being called.

To further exploit data parallelism we then analyze the block diagram to separate the tasks that need to be performed sequentially to produce the time decimated output of the PFB. First, all the data must be filtered by the polyphase filters, and is then passed to the FFT. Separating the problem this way gives a buffered implementation of the PFB rather than a streaming implementation. This is appropriate for a software implementation because the data already come buffered as a result of the Hashpipe implementation of the data pipeline.

The apparent parallelism in the polyphase filters for a single coarse frequency channel of a single antenna element time series is that all filtering is arranged in parallel and independent. With M parallel branches, the output of the mth polyphase filter is

$$y_m(n') = \sum_{p=0}^{P-1} e_m(p)x(n'M - pM - m), \quad m = 0, 1, \ldots, M-1. \tag{4.1}$$

The device kernel for the PFB implements this expression, identifying (4.1) as the task run by each thread. This is because Equation (4.1) is the fundamental parallel unit that needs to be computed for the mth branch of each decimated time series, providing the filtered data to the FFT block.

Computing the DFT using the FFT is a common operation in engineering applications and so NVIDIA has provided the cuFFT library as an implementation optimized for GPUs [53]. CUDA uses an object context model, meaning that similar to classes in C++ or structs in C, the context completely defines the object including input and output memory locations on the host and device. The cuFFT library is initialized with a handle to a context defining operational parameters. A full description of the cuFFT context and API is found in [54]. The context is robust and can be configured to efficiently perform multiple FFTs at once.

We therefore use cuFFT to compute the FFT across each of the 200, pre filtered, decimated time series. Applications using cuFFT operate in similar fashion to conventional FFT implementations in that all the time samples need to be processed in window sizes matching the transform length. Therefore in FLAG, with 4000 time samples in the frequency channel data, with a 32-Point FFT, there are $4000/32 = 125$ windows that cuFFT processes to produce all of the PFB outputs.

Table 4.1: Pseudo-code for host PFB implementation

```
void runPFB(char* dataIn_h, float2* dataOut_h, dim3 gridDim, dim3
   blockDim) {
    // initialize local variables and flags

    // copy data to device

    // run PFB filtering kernel
    PFB_kernel<<<gridDim, blockDim>>>(dataIn_d, FFTIn_d);
    // perform FFT
    while(!doneFFT) {
        doFFT();
        countFFT++;
        // update FFT data pointers

        // check exit condition
        if (countFFT > N_windows) {
            doneFFT = 1;
        }
    }
    // copy data from device to host

    return;
}

int main() {
// initialize main variables and flags

//setup grid and block dimensions
dim3 gridDim(N_time_series, N_windows, 1);
dim3 blockDim(NFFT, 1, 1);
// initialize host and device memory

// run PFB
runPFB(dataIn_h, dataOut_h, gridDim, blockDim);
return 0;
}
```

Each window requires a call to execute the FFT after advancing pointers in memory appropriately to the start of the next window.

Combining all of these aspects of data parallelism, we have been able to collapse all of the signal processing for this problem of applying a PFB that implements a 32-Point FFT, on 200 decimated time series of 4000 time samples, each with 125 windows of the FFT, to a single application of a PFB with an FFT over 125 windows. Translating these dimensions to configure the kernel call for the device results in a 200 (time series) x 125 (windows) grid of one dimensional blocks with 32 threads. There are 32 threads in each block because, Equation (4.1) represents the basic expression that needs to be executed in parallel for all time series, across all windows. Pseudo-code describing the code ran on the host is shown in Table 4.1.

In the above discussion, we have illustrated the GPU implementation of the PFB using specific details of the FLAG system. However, this library has been developed to abstract to different system and signal requirements. For example, for convenience in conforming to the structure of the GPU library that implements the correlator, we actually use 64-element time series. This matches the data sizes for which the FLAG GPU libraries are compiled to process, because the correlator has been optimized for matrix dimensions that are a power of two. For a GPU in general, grid and block dimensions are better optimized when they are set to a power of two.

4.3 Filter Performance Experiments

The PFB is used to provide fine frequency channels for post correlation analysis. As explained in chapter 3, it is important that the prototype LPF for a PFB implementation be designed to have low sidelobe levels as to minimize spectral leakage, and that it has a narrow transition band that overlaps with adjacent bins at the -3 dB crossover point. This section reports on measured performance characteristics of the PFB to verify accurate operation.

We first tested the PFB GPU library independent of the FLAG back end using simulated data to measure and verify the per-bin response as well as guarantee a uniform response across the passband of the filter bank. The PFB GPU library implements an $L = 256$ filter with a Hanning window and an $M = 32$ point FFT, the polyphase components are therefore 8-tap filters. The Hanning, Hamming and Kaiser-Bessel windows are among the most commonly used window choices for digital spectrometers in RA because of their extremely low sidelobe levels and sidelobe fall-off per octave [47].

To sample the frequency response, the input to the PFB is a series of complex exponential tones that are swept across frequency. Each tone is L samples in duration as to ensure there is a single instance (every L/M decimated time output) that a single tone would be seen by the entire filter bank. In normalized frequency (cycles/sample) each bin is M^{-1} cycles/sample wide. To have

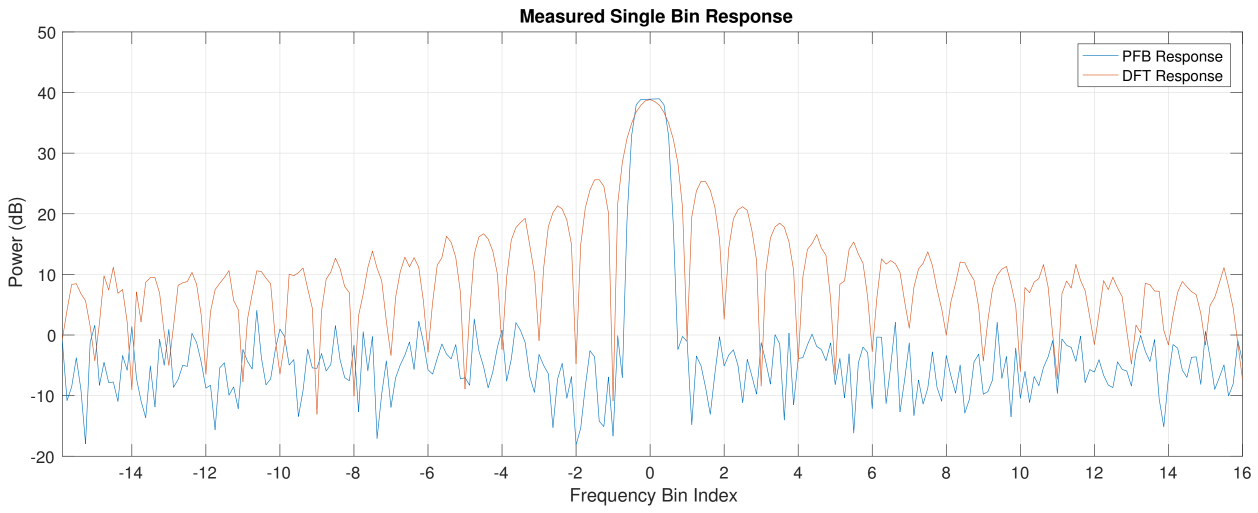


Figure 4.3: Measured single bin frequency response for an analysis filter bank bank implemented with an FFT and a PFB. The frequency response is measured using a series of complex tones in noise. The designed prototype LPF response of the PFB is narrower with lower sidelobes than the sinc function response of the FFT.

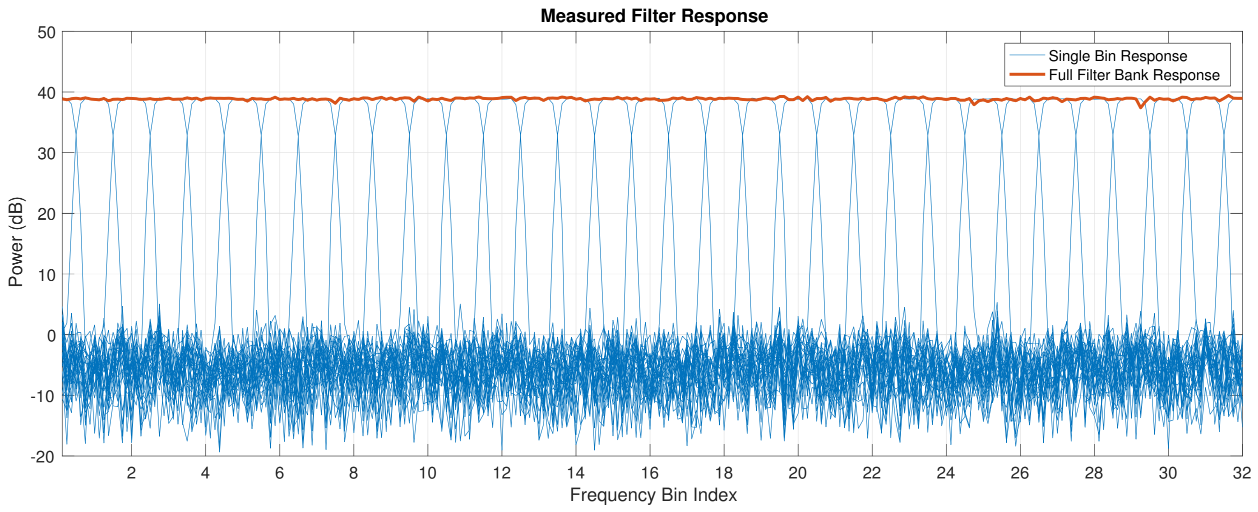


Figure 4.4: Measured PFB passband response using a series of complex tones in noise. The prototype LPF is shown to be designed properly such that there is no scalloping loss across the passband.

adequate frequency resolution, and sample at adjacent bin crossover points, this experiment uses tones that are swept in frequency by $\frac{1}{10}M^{-1}$ cycles/sample.

Figure 4.3 displays a single output from the PFB compared to that of a straightforward implementation of a filter bank using the FFT. In the experiments to measure the frequency response 100 realizations of a series of complex tones with power of 39 dB and additive uniform noise,

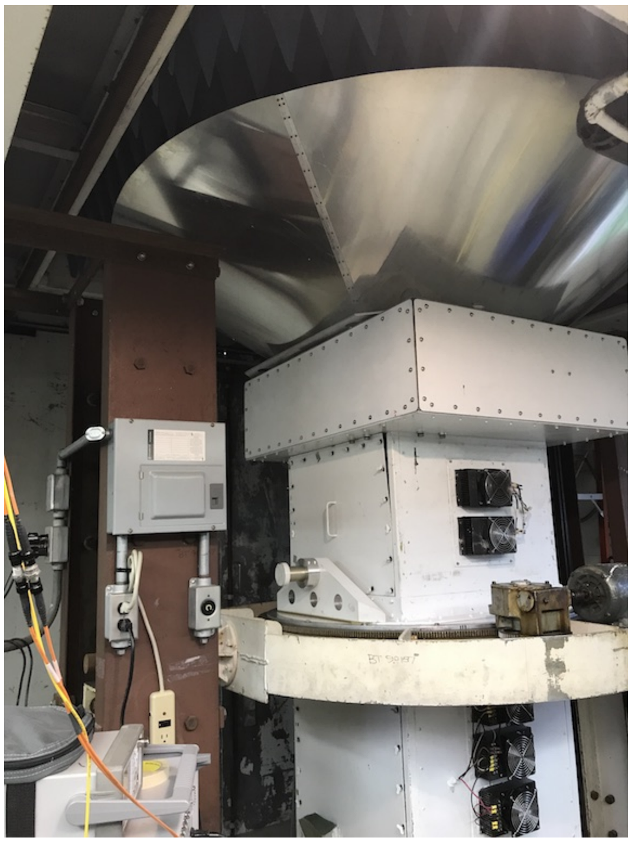


Figure 4.5: Front-end receiver system and array mounted in the OTF.

distributed $\mathscr{U}$(-31 dB, 31 dB), were filtered and averaged. The GPU PFB library uses the cuFFT library for the computation of the DFT following polyphase filtering. To generate the response for just the FFT, the input data bypassed the filtering step and was directly input into cuFFT. As desired, it is easily seen that the per-channel frequency response for the PFB has a narrower transition band and lower sidelobe levels than that of the FFT implementation. Figure 4.4 shows the per-bin response across the passband of the PFB. Again, as desired, we see that the PFB has a uniform response across the entire passband meaning the prototype LPF was designed correctly to avoid scalloping loss.

Following independent tests, the PFB GPU library was used to implement the fine channel correlator in the digital back end for real-data experiments to be conducted in the outdoor test facility (OTF) at GBO prior to commissioning. The OTF is equipped with a retractable roof, and inside, the front-end receiver system and array is mounted on a hydraulic lift which can be extended to expose the array to the sky. When the array is lowered and with the roof closed, there is a signal

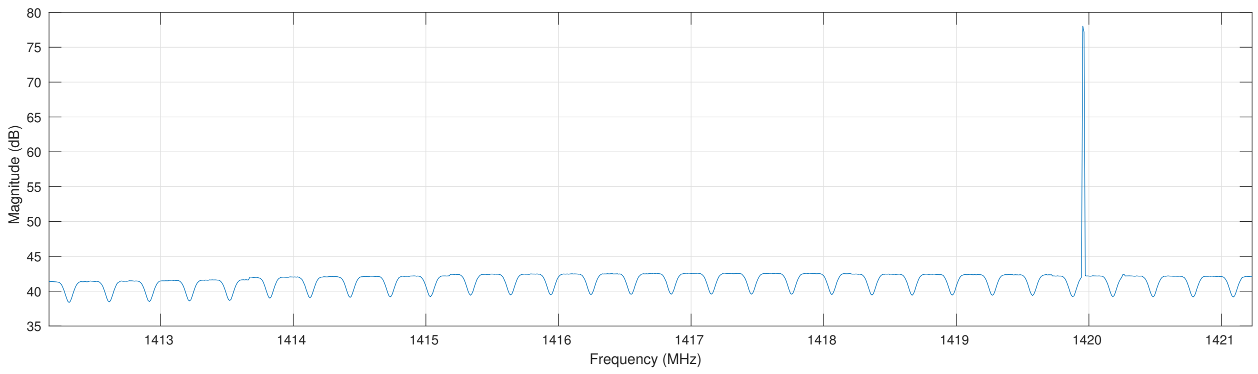


Figure 4.6: Testing the fine channel correlator in the OTF by injecting a tone at 1419.93 MHz. The tone is clearly present where expected. The scalloping loss present in the fine channel correlator is an inherent undesirable effect due to the spectrometer processing design which cascades across two PFBs.

generator that can be used to inject tones into the receiver system. Figure 4.5 shows the receiver system and array mounted in the OTF with the roof closed.

Before exposing the array to the sky, we injected several tones into the receiver system near 1420 MHz using a signal generator. Figure 4.6 shows the results for the Hashpipe instance of the fine channel correlator that processes the section of the passband including 1420 MHz with a single tone injected at 1419.93 MHz. The tone is clearly present where expected. This figure also shows that there is scalloping loss in the passband of the fine channel correlator.

This scalloping loss is attributed to the two stage PFB design for coarse and fine channelization in FLAG. The first stage coarse channel PFB in the ROACH is designed properly, but when the data are then further channelized by the fine channel PFB this results in the tracing out of the normal passband of the coarse channel bandpass filters at a finer frequency resolution. This design architecture also has the undesirable effect of aliasing because of the sample rate decimation in the coarse PFB stage to match the sample rate of the coarse channel bandwidth. Efforts to correct the scalloping and aliasing have been proposed with a fix to be implemented in future software updates of the digital back end.

Due to the abundance of galactic hydrogen, our goal in the OTF was to have a detection of neutral hydrogen at 1420.4 MHz (HI). This would give us the best confidence and indication that the fine channel correlator was operating correctly. Figure 4.7 presents the results of a brief 10 second observation with an integration time of 0.5 seconds over the full 30 MHz processing

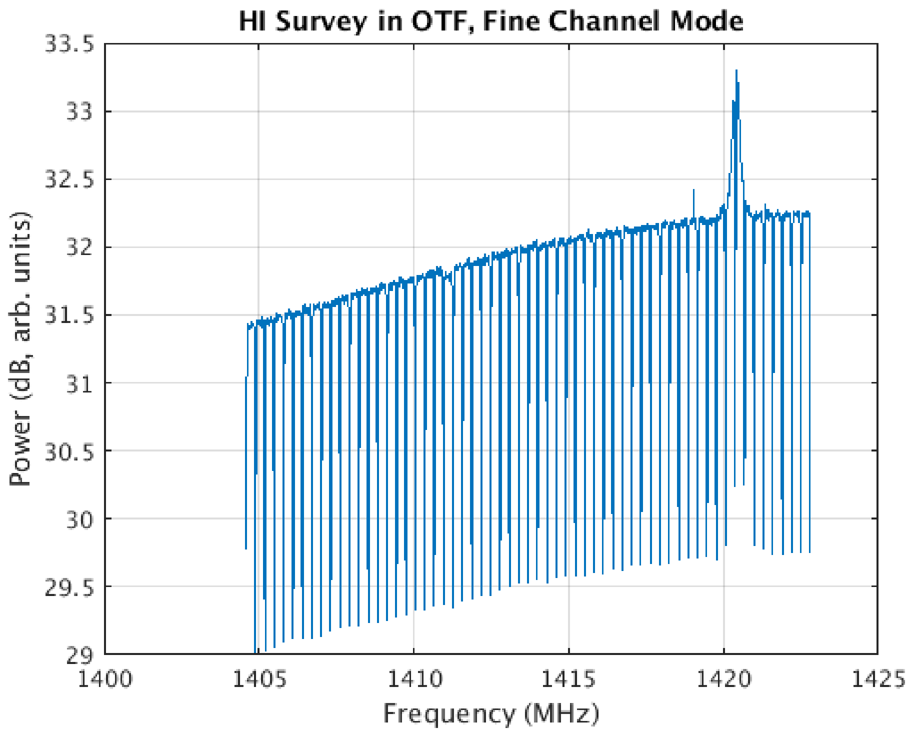


Figure 4.7: Detection of neutral hydrogen in the OTF.

bandwidth of the fine channel correlator mode. The half-second integrations are averaged together to produce the final figure. A clear detection of HI is seen spread around at 1420.4 MHz.

4.4 Conclusion

Advancements in GPU architecture, along with the introduction of highly parallel computing environments such as CUDA have allowed for prodigious improvements in computation power and resources across many scientific and engineering applications. For RA these improvements have allowed for processing large multi-sensor array systems in real-time with wide bandwidths capable of meaningful astronomical observations. In this chapter we presented a discussion on how the PFB, a conventional tool for digital spectrometers in RA, was adapted to implement the FLAG PFB GPU library.

We first presented the parallel programming paradigm model used in CUDA to represent data parallelism, or the highly parallel operations which are intrinsic to many large data processing problems. We then applied this model to identify parallel operations and sequences that are in the PFB architecture for application as part of the real-time fine channel correlator for the FLAG digital backend system. Following, we presented measured filter performance using simulated data verifying proper filter design and library implementation. This chapter also demonstrated pre-commissioning experiments in the OTF at GBO which verified successful integration of the PFB into the fine channel correlator with a clear HI detection.

CHAPTER 5. IMPROVED RFI MITIGATION FOR LARGE INTERFEROMETRIC ARRAYS

The high resolution images that an interferometer can create are motivating the construction and development of new array systems that are achieving improved resolution and sensitivity over their predecessors. For example, observations with baseline lengths of 14 km and millimeter-wave operating frequencies at the newly constructed ALMA observatory have helped to show the inaccuracies of current models in protoplanetary disk formation [55]. Further discoveries are expected as ALMA continues to operate. Likewise, other highly sensitive interferometer systems which will come on-line in the future, like the Square Kilometre Array (SKA) and the ngVLA, will add greatly to our understanding of the universe.

However, with many of the worlds observatories and institutes reporting damaging levels of RFI to the extent that data corruption occurs in up to 100% of their observations [23]–[26], the future of RA depends on these new systems and their ability to coexist in harsh RFI environments. In simulation, adaptive spatial filtering techniques, such as subspace projection, have shown to be effective in cancelling RFI [12], [13] for large interferometers. However, signal conditions for a practical observation scenario were not considered. This chapter presents a new spatial filtering algorithm for large interferometers taking into account these signal conditions improving RFI mitigation.

5.1 Signal Model and Synthesis Imaging Equations

This section defines the relevant geometry, notation and signal models needed to provide a foundation for our discussion on spatial filtering. Further discussion and a more complete study of synthesis imaging can be found in [18] and [56]. Figure 5.1 depicts a simple imaging scenario for an interferometry array system as explained in the following discussion.

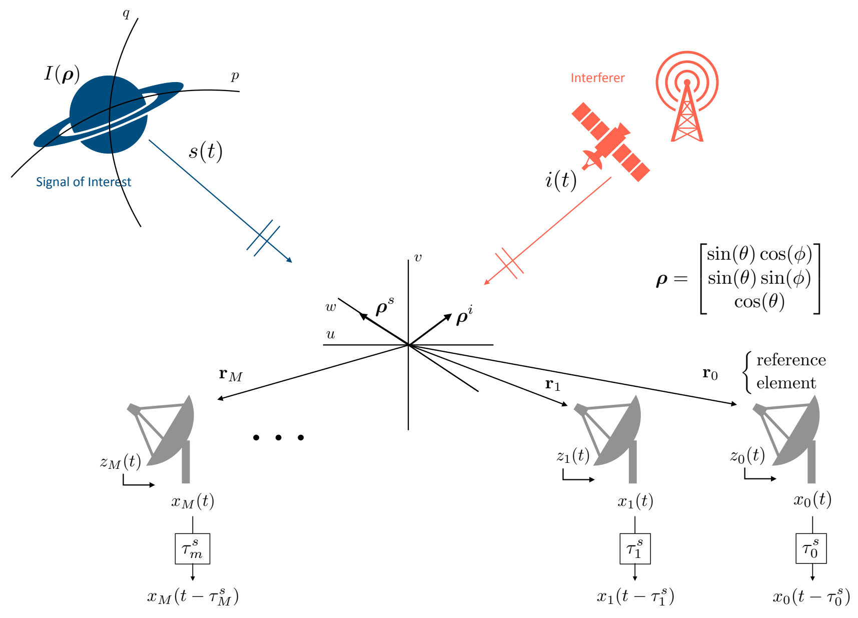


Figure 5.1: Simple imaging scenario for a synthesis imaging array. Given azimuth and elevation angles direction cosines can be used to calculate a propagation vector $\boldsymbol{\rho}$ for a SOI and interferer(s). The array has been co-phased with inserted time delays to image a celestial SOI. Interferers corrupt the visibilities for the SOI and introduce artifacts in the resulting radio image.

Consider imaging a celestial signal of interest (SOI) with an M-element radio interferometric array with antenna elements placed in arbitrary locations. The position of a particular element is described in the general coordinate system (u,v,w) in units of wavelength (λ) at a narrowband source observation frequency f_c. Broadband SOIs are observed using many such narrowband channels. Vector $\mathbf{r}_m$ describes the position of the mth element relative to an arbitrary origin. The SOI observation is projected onto a hypothetical celestial sphere which defines the imaging coordinate axis (p,q) where p is parallel to u and q is parallel to v. A radio image is formed by estimating the intensity distribution $I(\boldsymbol{\rho})$ of the electric field arriving from the direction described by unit-length propagation vectors $\boldsymbol{\rho} = (u_\rho, v_\rho, w_\rho)$ within the imaging field of view. Since the (u,v) and (p,q) coordinate axes are parallel, $\boldsymbol{\rho}$ may also be expressed in terms of (p,q); $\boldsymbol{\rho} = (\,p, q, \sqrt{1-p^2-q^2}\,)$. The range of (p,q) is limited (at most) to the narrow field of view defined by the beamwidth of the array antenna elements. Wavefronts from the celestial SOI travel perpendicular to the propagation vector. Here we have chosen $\boldsymbol{\rho}^s$ to represent the vector that points to the origin in the imaging reference plane, thus ρ^s= (0,0,1). This is the source reference position, and has at times been referred to as the phase center of the image.

For notational convenience the array element most distant from the SOI is arbitrarily designated as a reference element and is indexed as $m = 0$. Relative to this reference element, the time difference of arrival for the SOI wavefront at the mth element is denoted as τ_m^s. This geometric delay term is a function of the distance between the two elements, and may be expressed as

$$\tau_m^s = \frac{(\mathbf{r}_m - \mathbf{r}_0)^T \rho^s}{c}, \tag{5.1}$$

where c is the speed of light. For the reference element, $\tau_0 = 0$. In the interferometric image synthesis processing chain, time delays are first applied to receiver outputs to compensate for this geometric delay and co-phase the array to the phase center of the image.

Let the complex basebanded passband receiver signal output for the mth array element (antenna) be $x_m(t)$, and gather these to form the signal vector $\mathbf{x}(t)$. The system output vector time signal is modeled as a narrowband channel and is the linear combination of signal, interference, and noise given by

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_M(t) \end{bmatrix} = \mathbf{a}(\theta_s)s(t) + \mathbf{a}(\theta_i)i(t) + \mathbf{z}(t). \tag{5.2}$$

The vectors $\mathbf{a}(\theta_s)$ and $\mathbf{a}(\theta_i)$ represent the previously mentioned array spatial signatures in the direction of the SOI and interferer respectively. For an extended SOI, we consider θ_s to be the direction to a single pixel in the image for the SOI. All θ values are two dimensional spherical angles. The spatial signature is also referred to in the literature as a steering vector or the array response. The mth entry in $\mathbf{a}(\theta)$ is the narrowband complex magnitude and phase response of the array to a unit amplitude signal arriving from direction angle θ. The signals $s(t)$, $i(t)$ and $\mathbf{z}(t)$ are modeled as zero-mean Gaussian proper complex-valued random processes and represent the SOI, man-made interfering signal and noise respectively.

For the limited field of view in this imaging scenario, the fundamental quantity used to estimate the intensity image of the electric field distribution, $I(\rho)$, from receiver outputs is known as the visibility function [18]. In the absence of noise and interference the visibility function is approximately given by

$$\begin{aligned} V(\mathbf{r}_l, \mathbf{r}_m) &= E\left[x_l(t)x_m^*(t)\right] \\ &= \int\int |A(\rho)|^2 I(\rho) e^{-j2\pi(\rho-\rho^s)^T(\mathbf{r}_l-\mathbf{r}_m)}\, d\rho, \end{aligned} \tag{5.3}$$

where $E[\cdot]$ is the expectation operator and * denotes complex conjugate. The function $A(\rho)$ represents the antenna response pattern for each array element, and is assumed to be identical across elements. The difference vector $\mathbf{r}_l - \mathbf{r}_m$ is referred to as a baseline between elements l and m. Expressing these baselines in (u, v) coordinates and assuming a continuous range of all possible baseline vectors are available leads to the visibility function form:

$$V(u, v) = \int\int |A(p, q)|^2 I(p, q) e^{-j2\pi(up+vq)}\, dp\, dq. \tag{5.4}$$

An intuitive interpretation of this expression is that the imaging information is related to the spatial cross-correlation of antenna pairs (baselines) through a Fourier transform relationship as seen in (5.4).

However, since the available baselines are limited to the discrete set of antenna pairs (l, m), we observe only an irregularly sampled version, $V(u_{l,m}, v_{l,m})$, of the visibility function of (5.4). These pairs may be collected into a single matrix

$$\mathbf{R} = E\left[\mathbf{x}(t)\mathbf{x}^H(t)\right] = \begin{bmatrix} V(\mathbf{r}_0, \mathbf{r}_0) & \dots & V(\mathbf{r}_0, \mathbf{r}_{M-1}) \\ \vdots & \ddots & \vdots \\ V(\mathbf{r}_{M-1}, \mathbf{r}_0) & \dots & V(\mathbf{r}_{M-1}, \mathbf{r}_{M-1}) \end{bmatrix}, \tag{5.5}$$

where ^{H} is matrix conjugate transpose. This matrix is recognized as the array autocorrelation matrix (or covariance matrix since $\mathbf{x}(t)$ is zero mean) and its entries are known as visibilities. In general, RFI and noise are present and are statistically independent from each other and the SOI.

Therefore, the visibility matrix can be separated and expressed as

$$\begin{aligned}\mathbf{R} &= E\left[\mathbf{x}(t)\mathbf{x}^H(t)\right] \\ &= \mathbf{R}_s + \mathbf{R}_i + \mathbf{R}_z,\end{aligned} \tag{5.6}$$

where $\mathbf{R}_s$, $\mathbf{R}_i$, and $\mathbf{R}_z$ are the individual covariance matrices for SOI, interferer, and noise respectively. The true covariances are not known, therefore $\mathbf{R}_s$ can only be observed through a sample estimate, $\widehat{\mathbf{R}}$ (where $\widehat{\ }$ indicates an estimated quantity), which includes the undesired contributions from $\mathbf{R}_i$ and $\mathbf{R}_z$. These estimates are computed by the correlator of a digital back end processing system for a range of narrowband channels across the full processing bandwidth.

Most RFI cancelling array-based spatial filtering algorithms use the sample spatial covariance matrix $\widehat{\mathbf{R}}$ to estimate the array response of the interferer in order to subsequently remove interference [15]. These covariance estimates are in fact the noise and RFI corrupted visibility matrices and are the output of the central correlator:

$$\begin{aligned}\widehat{\mathbf{R}}_j &= \frac{1}{N}\sum_{n=jN}^{(j+1)N-1} \mathbf{x}[n]\mathbf{x}^H[n], \quad 0 \le j \le J-1 \\ &= \mathbf{R}_s + \mathbf{R}_{i,j} + \mathbf{R}_z + \mathbf{E}_j.\end{aligned} \tag{5.7}$$

Due to RFI motion the sample covariances must be recomputed frequently using a relatively small number, N, of samples over J periods, on ms time-scales. These are called short-term integrations (STIs). Over the period of one STI the spatial signature of the interferer is assumed to be stationary. The STIs are combined over a long-term integration (LTI) period of JN samples, on the order of 10 s, as the overall LTI estimate $\widehat{\mathbf{R}}$ used for image synthesis. The estimates of the signal and noise covariance matrices, $\mathbf{R}_s$ and $\mathbf{R}_z$, are also assumed to be stationary over both long and short-term integrations. The matrix $\mathbf{E}_j$ represents sample estimation error because of the finite number, N, of samples and the presence of $\mathbf{R}_i$ and $\mathbf{R}_z$ in the estimation process.

Ideally the desired uncorrupted image would be computed from unbiased estimates $\widehat{\mathbf{R}}_s$ for each channel. This chapter presents an algorithm to extend subspace projection RFI cancelling algorithms by using partitioned subarrays to better remove the bias introduced from $\mathbf{R}_i$ in large, long baseline imaging interferometers. The proposed method reduces estimation error while identifying the RFI parameters.

In practice, imaging arrays take advantage of the Earth's rotation by recomputing the visibilities on medium time scales. Each new computed set of visibilities has a unique rotated set of baselines, further filling in the (u,v) sample space in (5.4). An RFI cancelling projection operator would be computed and applied separately for each new set of baseline orientations. To simplify the discussion to follow, we will usually consider only snapshot imaging based on a single LTI window which produces a single correlator dump estimate of $\widehat{\mathbf{R}}$. The final imaging example however simulates a full 12-hour observation with many STIs and corresponding baseline rotations in (u,v). The algorithm is also readily applied to the full series of baseline sets by performing separate RFI projections for each STI.

5.2 RFI Decorrelation

Long baseline arrays like the proposed ngVLA and SKA are problematic for projection-based RFI cancelling applied to the visibilities. Indeed any adaptive cancelling algorithm based on covariance estimates (multiple sidelobe canceler, MMSE array filter, etc.) would suffer similar limitations. This is because the RFI, even if visible in all elements, is decorrelated over the longer baselines. This makes it difficult to estimate RFI spatial parameters needed for effective projection cancellation. This effect is also present in other existing interferometers such as the VLA. To illustrate the effects of RFI decorrelation we will examine a possible element configuration for the proposed ngVLA [41]. Figure 5.2 shows the full configuration of the array elements as well as a magnification of the core at 15 km, 5 km and 3 km radius views.

Large synthesis arrays such as the ngVLA are less sensitive to interference than single-dish telescopes [39]. Thompson presented threshold levels for which RFI is detrimental to observations on the VLA, as well as how similar levels could be computed to extend to other arrays. While it is true that threshold levels are higher due to RFI decorrelation in these long baseline arrays, interference is still clearly present in resulting maps and images.

There are two effects that Thompson investigated which reduce the response of the array to an interferer. The first is the averaging effect. The argument is that relative to the earth, a ground-based interferer is stationary, and as the array tracks the SOI there is a slight change in

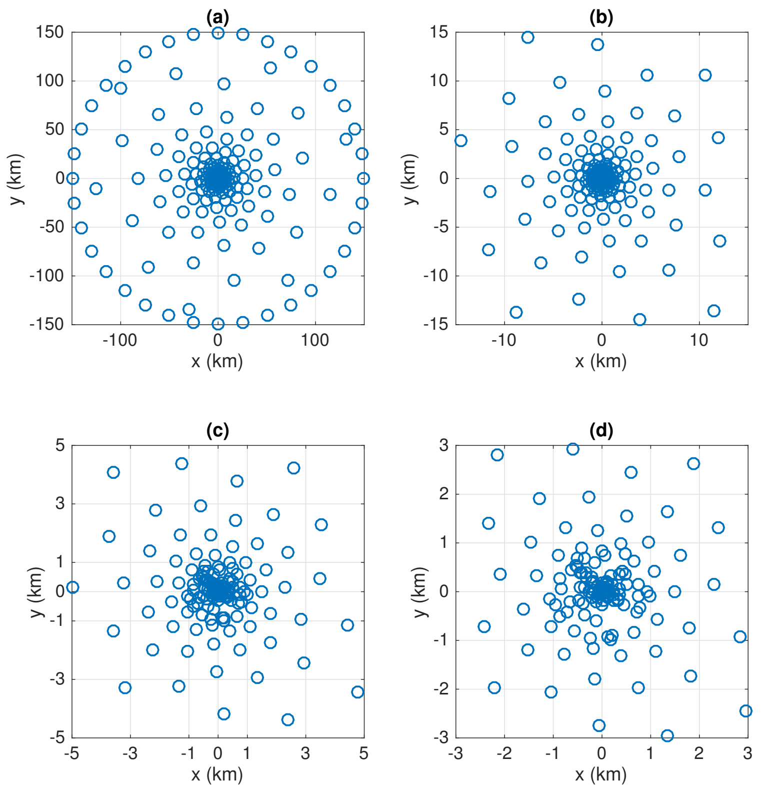


Figure 5.2: (a) One proposed ngVLA configuration consists of 300 antennas with the longest baselines extending to 300 km. This configuration also includes a dense core with 20% of the elements inside the radius of 0.6 km. (b)-(d) Magnified into the core at 15 km, 5 km and 3 km radius scales respectively.

relative phase. This relative phase difference is known as the fringe frequency [56]. At the time of image formation, due to the significantly large number of (u,v) samples that are interpolated onto a rectangular grid for an FFT-based Fourier inversion of (5.4), the averaging of these relative phase differences at individual points over the entire grid reduces interferer signal levels due to destructive cancelling of the RFI terms being summed out of phase.

The second effect is a result of the phase propagation that occurs due to the geometric delay of the interferer across the array. The geometric delay for the interferer is similar to the definition of (5.1) but is now dependent on the propagation vector ρ^i. Let this delay be specified as τ_m^i. Before applying the co-phasing time delays which compensate for the geometric delay due to the SOI, the output of the mth element in response to a wave originating from a point source at the phase reference location $(p=0, q=0)$ in the presence of single interfering source is

$$x_m(t) = a_m(\theta_s)s(t+\tau_m^s) + a_m(\theta_i)i(t+\tau_m^i) + z_m(t). \tag{5.8}$$

Subsequently, applying the co-phasing time delay yields,

$$x_m(t-\tau_m^s) = a_m(\theta_s)s(t) + a_m(\theta_i)i(t+\tau_m^i-\tau_m^s) + z_m'(t). \tag{5.9}$$

Because the noise process $z_m(t)$ is an independent and identically distributed wide sense stationary process, the co-phasing delay does not affect the statistics of the process and we can consider it as if no delay had been added to the noise. We see that for the phase reference location in the imaging plane, the output at time t across the entire array will not depend on the propagation path of the SOI. Or, in other words, when the correlator produces visibilities it will compare the signal between elements as if it had coherently sampled the entire time aligned wavefront for the SOI.

However, the same cannot be said of the interfering source. The time series for the interferer is affected by the geometric delay as it propagates across the array, as well as the co-phasing time delay resulting in an effective bulk geometric delay of $\tau_m^b = \tau_m^i - \tau_m^s$. The consequence is that there may be a significant time offset of the interferer across baselines. For a given processing bandwidth β the sample offset is $(\tau_m^i - \tau_m^s)\beta$ and the amount of phase that has propagated across the array over this delay is then given by

$$\begin{aligned} \psi_m &= 2\pi(\tau_m^i - \tau_m^s)\beta \\ &= 2\pi\tau_m^b\beta. \end{aligned} \tag{5.10}$$

This phase propagation results in incoherence of the interferer at the correlator and can be characterized by the correlation function

$$\gamma(\tau_m^b;\, \beta) = \frac{\sin(\pi\tau_m^b\beta)}{\pi\tau_m^b\beta} = \operatorname{sinc}(\tau_m^b\beta), \tag{5.11}$$

where the sinc function is defined as

$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x}.$$

Note that we can manipulate (5.10) to express (5.11) in terms of the phase propagation

$$\gamma(\psi_m) = \operatorname{sinc}\left(\frac{\psi_m}{2\pi}\right). \tag{5.12}$$

These correlation functions manifest a fundamental trade-off between baseline length, processing bandwidth, and the effects on correlation of the RFI in the visibilities. For narrow processing bandwidths on the order of a few Hz to a few kHz, which would be used for example in the processing of narrowband galactic emissions, the correlations vary slowly and significant decorrelation would only begin to occur at a few hundred kilometers. As the processing bandwidth increases for the observation of more continuum sources, the correlations decay more rapidly as the effective projected baseline length for the bulk geometry delay increases. Figure 5.3 depicts the decorrelation of an interferer for an antenna pair on the ngVLA for various processing bandwidths and baseline lengths. The interferer is arriving endfire to the baseline vector for a maximum, worst case, effective projected baseline length.

We can see that in general, correlations do decay, however, only in the limit are they zero. Also, it is important to recognize that for most practical processing bandwidths there will be significant correlations at several elements because of compact core configurations. This indicates that RFI is still a major problem for synthesis arrays and that RFI mitigation techniques will be required in order to preserve data and image integrity.

5.3 Subspace Projection

In the presence of Q interferers, subspace projection is a zero-forcing, null-forming algorithm that cancels interferers by applying projection operator $\mathbf{P}_j$ to the $M \times M$ estimated sample covariance matrix $\widehat{\mathbf{R}}_j$ for a given STI interval. In the analysis that follows we drop the subscript j, and it is implied that the calculation of a projection matrix is done for each STI. The projection is designed to be approximately orthogonal to the subspace that spans the spatial signature of the

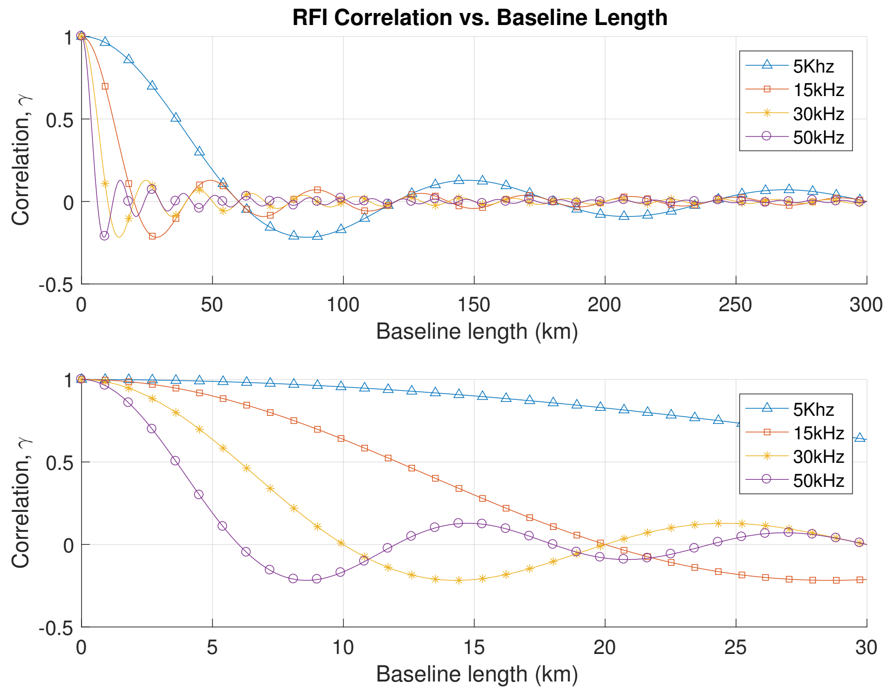


Figure 5.3: Extent of RFI decorrelation for an antenna pair at different processing bandwidths. The signal is arriving endfire for a maximum projected baseline length. In general, the correlation varies slowly for narrowband channel processing. As the bandwidth increases the correlations decreases more rapidly. With a dense core on the ngVLA, there will be significant correlation and without mitigation will result in corrupted images.

Q interferers. To design the projection matrix a dominant eigenvector analysis is performed on the sample covariances. The eigenvector decomposition of the sample covariance matrix is

$$\widehat{\mathbf{R}} = \mathbf{U}\Lambda\mathbf{U}^H, \tag{5.13}$$

where the columns of $\mathbf{U}$ are the eigenvectors of $\widehat{\mathbf{R}}$ and the diagonal entries of Λ are the corresponding eigenvalues. Assuming the interferers are much stronger than the SOI and system noise, the resulting eigenvector matrix $\mathbf{U}$ can be sorted and partitioned as $\mathbf{U} = [\mathbf{U}_i \,|\, \mathbf{U}_{s+z}]$, where $\mathbf{U}_i$ represents the eigenvectors corresponding to the Q largest eigenvalues. The conventional subspace projection matrix $\mathbf{P}_{\text{SP}}$ is then formed as

$$\mathbf{P}_{\text{SP}} = \mathbf{I} - \mathbf{U}_i(\mathbf{U}_i^H\mathbf{U}_i)^{-1}\mathbf{U}_i^H. \tag{5.14}$$

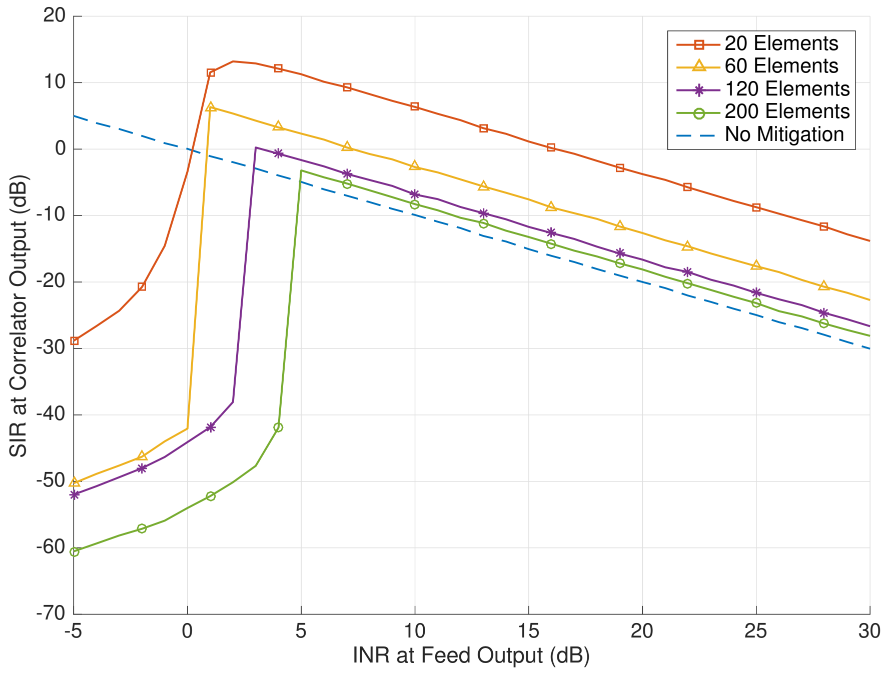


Figure 5.4: Illustration of the effectiveness of subspace projection on the ngVLA at a center frequency of 2.6 GHz and a processing bandwidth of 20 kHz. Starting from the dense core and increasing the number of elements that are included in the subspace estimation process this also increases the number of long effective projected baselines. With a large number of elements at longer baselines subspace projection becomes less effective at providing cancellation to the RFI.

The projection is applied to $\widehat{\mathbf{R}}$ by left and right multiplication,

$$\widetilde{\mathbf{R}} = \mathbf{P}_{\text{SP}}\widehat{\mathbf{R}}\mathbf{P}_{\text{SP}}^{H}, \tag{5.15}$$

resulting in the filtered visibility matrix $\widetilde{\mathbf{R}}$, where the Q interferers have been largely eliminated. Using the estimated subspace, the projections will only be approximately orthogonal to the RFI and therefore introduce a slight bias in $\mathbf{R}_s$. This can be corrected on average as show in [11], [13], [57].

A metric that is used to measure performance of a mitigation algorithm is the signal-to-interference power ratio (SIR) at the filter output, given by

$$\text{SIR} = \frac{\text{Tr}\left\{\mathbf{P}\mathbf{R}_s\mathbf{P}^H\right\}}{\text{Tr}\left\{\mathbf{P}\mathbf{R}_i\mathbf{P}^H\right\}}, \tag{5.16}$$

where $\mathrm{Tr}\{\cdot\}$ is the matrix trace operation. This metric compares the performance of the level to which $\mathbf{P}$ attenuates the interference relative to attenuation in the signal. Figure 5.4 illustrates how SIR is affected by baseline length for the ngVLA configuration. This detailed simulation computes visibilities and applies subspace projection across M elements starting from the central core and extending outward. This figure includes a curve labeled no-mitigation, which represents the resulting SIR had no projection been applied to the visibilities. The abscissa of this plot is the INR at the element antenna terminals. As previously mentioned, subspace projection techniques have been shown to be promising because of the level of excision that can be achieved due to the deep nulls (under low RFI decorrelation conditions) that are formed in the direction of the interferers [11], [13]. Figure 5.4 shows that the greatest attenuation to the interferer relative to the SOI was achieved for the compact array with small M. However, the effectiveness of subspace projection decreases with larger M. As M increases, the post-mitigation SIR is drawn closer and closer to the no-mitigation curve. Thus, the amount of RFI attenuation decreases due to long baseline decorrelation to the extent that it is as if no projection had been applied.

As was shown in Section 5.2, the level of correlation for the interferer is well-defined in terms of the phase propagation due to processing bandwidth and bulk geometric time delay. Small M results in a dense set of elements in the central core over baselines where there is a significant measure of coherence for the interferer. There are two factors which contribute to the decreased effectiveness of subspace projection across large baselines as seen in Figure 5.4.

The first factor is that the underlying assumption of RFI subspace estimation has been violated by using weak coherence levels in the sample estimate. The design of $\mathbf{P}_{\mathrm{SP}}$ had assumed interference is the strongest component present in the sample covariances. On the far baselines the dominant component in the covariance is now the SOI or noise, preventing accurate estimation of the interferer subspace. This issue would arise even if an exact $\mathbf{R}$ was available with no estimation error, and is related to the partitioning step to select $\mathbf{U}_i$ from $\mathbf{U}$. The second factor is that the level of sample estimation error due to $\mathbf{E}$ in (5.7) has contributions from not only $\mathbf{R}_i$ but $\mathbf{R}_s$ and $\mathbf{R}_z$ as well. The contributions to $\mathbf{E}$ are proportional to the scale of the entries in the estimated covariances and inversely proportional to the number of samples N [58], [59]. On long baselines where entries in $\mathbf{R}_i$ are small, these small values can be overwhelmed by the proportionally larger entries in $\mathbf{E}$ due to $\mathbf{R}_s$ and $\mathbf{R}_z$ [11].

5.4 Subspace Projection via Subarray Processing (SP-SAP)

Due to considerations discussed in Section 5.3, achieving an adequate estimate of the interferer subspace is not possible using the entire array covariance matrix $\mathbf{R}$ directly. In the proposed subarray processing method, modifications are made to the subspace projection algorithm to improve estimates of the RFI spatial signature. This allows spatial filtering to be a viable option on large interferometers in the growing presence of RFI.

The expression in (5.9) for bulk geometric time delay of the interferer can be expanded using definitions of delays for the SOI and interferer to become

$$\begin{aligned}\tau_m^b &= \tau_m^i - \tau_m^s \\ &= \frac{(\mathbf{r}_m - \mathbf{r}_0)^T \rho^i}{c} - \frac{(\mathbf{r}_m - \mathbf{r}_0)^T \rho^s}{c} \\ &= \frac{(\mathbf{r}_m - \mathbf{r}_0)^T (\rho^i - \rho^s)}{c}.\end{aligned} \tag{5.17}$$

Substitution into (5.10) yields the phase change for RFI across the array relative to the reference antenna $\mathbf{r}_0$:

$$\psi_m = 2\pi \frac{(\mathbf{r}_m - \mathbf{r}_0)^T (\rho^i - \rho^s)}{c} \beta. \tag{5.18}$$

The difference between the SOI and interference propagation vectors can be thought of as the effective propagation vector (after inserted time delays to co-phase the array to the SOI) for the interference. Planes perpendicular to this vector represent regions of constant RFI phase across the array.

SP-SAP uses ψ_m as a metric to partition the full array of M elements into sections of K subarrays. The kth subarray is denoted as L_k and is the set of elements satisfying

$$L_k = \{m : \zeta_{k-1} \leq \psi_m < \zeta_k\} \quad \forall \quad m, \tag{5.19}$$

where $\zeta_k = \zeta_{k-1} + \psi_{\text{thresh}}$ with $\zeta_0 = 0$, and ψ_{thresh} is a user defined parameter. Any two elements within a subarray have a phase propagation difference less than ψ_{thresh}. Grouping the elements in this way creates a set of smaller subarrays with elements that are aligned perpendicular to the effective propagation vector, thus placing elements in the planes of constant phase. Therefore, each

Table 5.1: Subarray formation

```
Algorithm:
Given $\psi_{\text{thresh}}$ and the set of $M$ array elements
Compute $\psi_m = 2\pi \frac{(\mathbf{r}_m - \mathbf{r}_0)^T(\rho^i - \rho^s)}{c}\beta$, for $m = 0, 1, \ldots, M-1$
Sort the $\psi_m$ values
$k = 1, \zeta_0 = 0$
while array elements remain unassigned to a subarray:
    $\zeta_k = \zeta_{k-1} + \psi_{\text{thresh}}$
    $L_k = \{m : \zeta_{k-1} \leq \psi_m < \zeta_k\} \quad \forall \quad m$
    $k = k + 1$
```

Subarrays are subsets of the entire set of elements. Any two elements within a subarray have a phase propagation difference less than ψ_{thresh}.

of the K subarrays is guaranteed to have high mutual RFI correlation among its own elements to better estimate the interference subspace and apply subspace projection.

The algorithm for partitioning subarrays is shown in Table 5.1 and described in an example as follows:

1. Select a phase threshold, ψ_{thresh}.
2. Compute ψ_m using (5.18) for each element of the full array and sort in ascending order. Zero relative phase is assigned arbitrarily to the reference element $m = 0$.
3. Initialize $k = 1$ and the range of allowed phase in (5.19) with $\zeta_0 = 0$ and $\zeta_1 = \psi_{\text{thresh}}$.
4. Using (5.19) for subarray L_1, compare entries in the sorted array of ψ_m to the range of allowed phase and group elements into L_1 until no other element satisfies the current range condition.
5. Increment k and update ζ_k.
6. Continue to use (5.19) to compare remaining entries of the sorted array of ψ_m and group elements into subarray L_k.

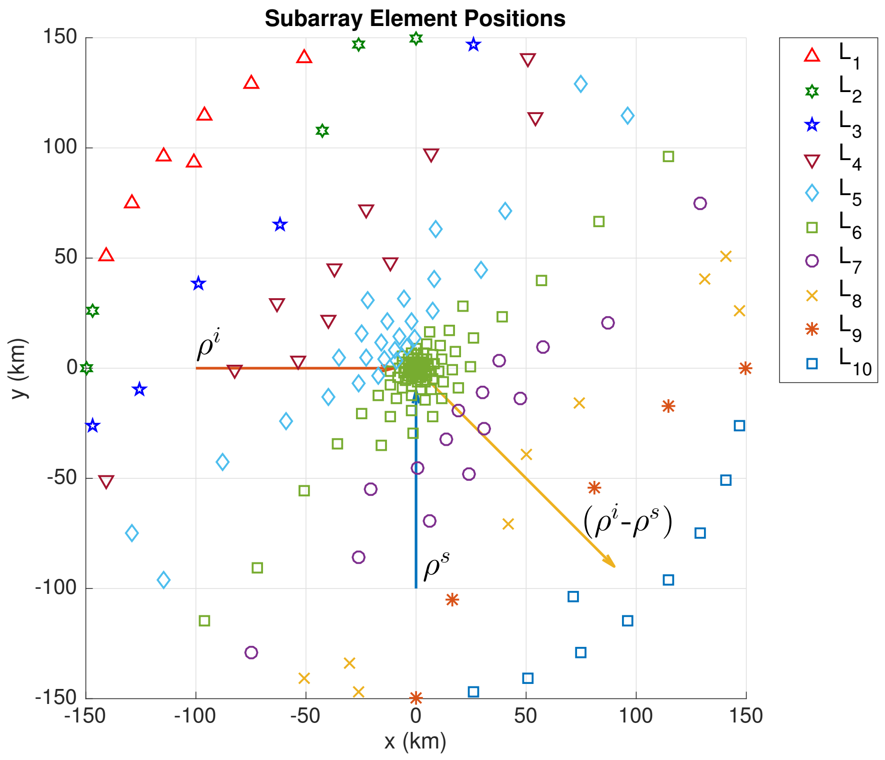


Figure 5.5: Resulting subarrays after being organized based on the phase propagation metric. This metric orients the subarrays perpendicular to the effective propagation vector $\rho^i - \rho^s$. To form the subarrays shown, the phase threshold was chosen to be high to exaggerate the illustration of the subarrays and their orientation. A small phase threshold will result in more subarrays with less elements per subarray. Conversely, a high threshold results in less subarrays with more elements per subarray.

7. Repeat steps 5 and 6 until all elements in the sorted array of ψ_m have been assigned to a subarray.

Figure 5.5 shows an example of the algorithm to determine subarrays. As expected, the arrays are arranged in bands aligned perpendicular to the effective propagation vector, indicating that the subarrays lie in regions of similar phase.

Each subarray has been designed to contain a set of elements with high mutual correlation of the RFI. This improves the estimate of the array response by decreasing residuals from sample estimation error. The resulting projection matrix for each subarray will then better project out RFI in the local grouped elements. To apply subspace projection to the entire visibility set, a projection matrix $\mathbf{P}_k$ is computed for each subarray L_k and included in the block diagonal matrix $\mathbf{P}_{\text{SAP}}$,

$$\mathbf{P}_{\text{SAP}} = \begin{bmatrix} \mathbf{P}_1 & 0 & 0 & 0 \\ 0 & \mathbf{P}_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \mathbf{P}_K \end{bmatrix}. \tag{5.20}$$

Any projection matrix must satisfy the condition that it is symmetric and idempotent (i.e. $\mathbf{PP} = \mathbf{P}$). Note that $\mathbf{P}_{\text{SAP}}$ satisfies both of these properties.

In the process of assigning subarrays, the array elements have gone from an arbitrarily indexed set to an ordered set by the phase metric ψ_m. Elements of vector $\mathbf{x}(t)$ must then be reordered to be compatible with new projection matrix $\mathbf{P}_{\text{SAP}}$. This is quickly achieved using a selection matrix $\mathbf{S}$. Sparse matrix $\mathbf{S}$ has a single non-zero entry of one in each row and column. For example, if array element n in $\mathbf{x}(t)$ is to be moved to the mth element location in the reordered vector $\mathbf{x}'(t)$, then $\mathbf{S}$ has a one in row m and column n. The transformation is reversed with left multiplication by $\mathbf{S}^T$:

$$\begin{aligned} \mathbf{x}'(t) &= \mathbf{S}\mathbf{x}(t) \\ \mathbf{x}(t) &= \mathbf{S}^T\mathbf{x}'(t). \end{aligned} \tag{5.21}$$

The reordered covariance matrix is computed as

$$\begin{aligned} \mathbf{R}' &= E\left[\mathbf{x}'(t)\mathbf{x}'^H(t)\right] \\ &= E\left[\mathbf{S}\mathbf{x}(t)\mathbf{x}^H(t)\mathbf{S}^T\right] \\ &= \mathbf{S}\mathbf{R}\mathbf{S}^T, \end{aligned} \tag{5.22}$$

and projecting out RFI is now straightforward by applying $\mathbf{P}_{\text{SAP}}$ to the reordered sample covariance matrix,

$$\widetilde{\mathbf{R}}' = \mathbf{P}_{\text{SAP}}\mathbf{S}\widehat{\mathbf{R}}\mathbf{S}^T\mathbf{P}_{\text{SAP}}^H. \tag{5.23}$$

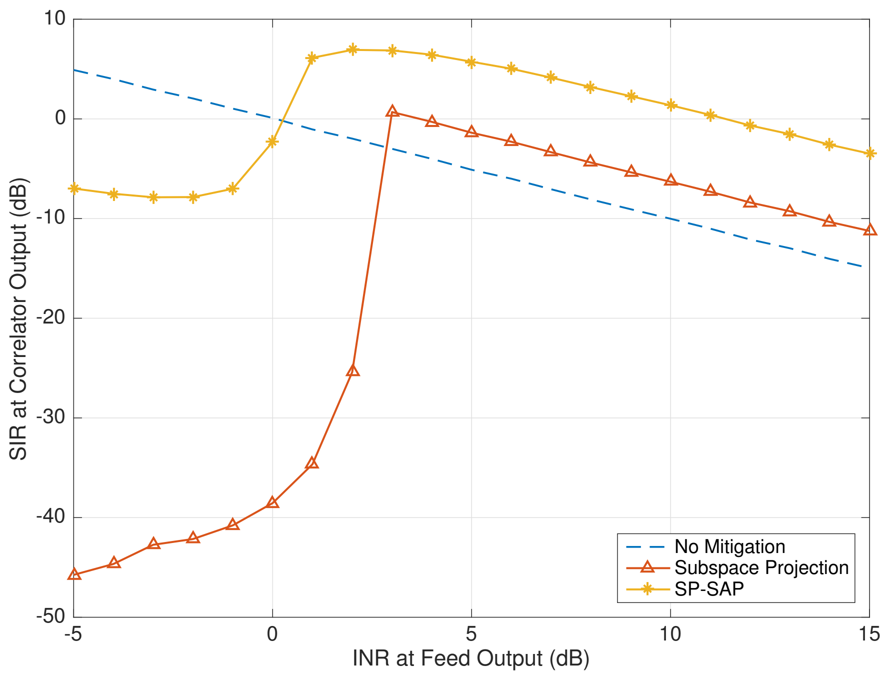


Figure 5.6: Comparison of SIR post-mitigation at the correlator output. SP-SAP is able to mitigate the RFI at lower INRs and consistently improves the SIR as compared to subspace projection over the entire array.

This matrix is then transformed back to the original order,

$$\widetilde{\mathbf{R}} = \mathbf{S}^T \mathbf{P}_{\text{SAP}} \mathbf{S} \widehat{\mathbf{R}} \mathbf{S}^T \mathbf{P}_{\text{SAP}}^H \mathbf{S}, \tag{5.24}$$

resulting in the final filtered visibility matrix. Defining $\mathbf{P}'_{\text{SAP}} = \mathbf{S}^T \mathbf{P}_{\text{SAP}} \mathbf{S}$, it is easy to see that SP-SAP takes on the familiar form of subspace projection,

$$\widetilde{\mathbf{R}} = \mathbf{P}'_{\text{SAP}} \widehat{\mathbf{R}} \mathbf{P}'^H_{\text{SAP}}. \tag{5.25}$$

5.5 Simulation Results

In this section we compare interference cancelling simulation results for SP-SAP and the conventional approach of applying subspace projection directly over the entire array. In both simulations, the full array for the ngVLA configuration of Figure 5.2 is used. The SOI is a hypothetical galactic signal at 2800 MHz, with a single interferer. Processing bandwidth is 15 kHz to satisfy the narrowband assumption with respect to the interferer. In these simulations we assume short-integration periods are used therefore the interferer can be approximated as stationary relative to the array.

Earlier the performance measure of SIR was introduced. Another measure of performance for an interference mitigation algorithm is the normalized mean square error in $\widetilde{\mathbf{R}}_s$, given by

$$\varepsilon_s^2 = \frac{\left\|\widetilde{\mathbf{R}}_s - \mathbf{R}_s\right\|_F^2}{\|\mathbf{R}_s\|_F^2}, \tag{5.26}$$

where $\widetilde{\mathbf{R}}_s = \mathbf{P}\mathbf{R}_s\mathbf{P}^H$ and $||\cdot||_F$ is the Frobenius norm. This metric measures post-mitigation bias introduced in $\mathbf{R}_s$ as a result of the applied projection $\mathbf{P}$. Also, we define the attenuation levels for interference and SOI respectively as

$$\alpha_i = \frac{\mathrm{Tr}\{\mathbf{R}_i\}}{\mathrm{Tr}\{\mathbf{P}\mathbf{R}_i\mathbf{P}^H\}},$$

$$\alpha_s = \frac{\mathrm{Tr}\{\mathbf{R}_s\}}{\mathrm{Tr}\{\mathbf{P}\mathbf{R}_s\mathbf{P}^H\}}. \tag{5.27}$$

Metrics α_i and α_s show the overall effectiveness of removing power from the interferer as well as undesired SOI attenuation.

Figure 5.6 compares output SIR for subspace projection and SP-SAP across the whole array. Consistent with Figure 5.4, due to sample estimation error, subspace projection using the full array does not a achieve a satisfactory level of interference cancellation. However, SP-SAP is able to begin cancelling RFI at weaker INR levels and achieves better overall cancellation. Examining attenuation factors α_s and α_i provides further insight as to the effects of SP-SAP. Figure 5.7 depicts these values in dB for both the SOI and interferer. Compared to SP-SAP, subspace projection does better at preventing bias to the SOI. However, examination of the interferer attenuation shows that the 1 dB SOI attenuation results in more than a 9 dB improvement in interference cancellation. No bias correction has been made and so any losses in the SOI could still be recovered while achieving the same level of RFI attenuation [11]–[13], [57].

Figure 5.8 presents the results of the mean squared error determined by (5.26). This metric is relevant for synthetic imaging because in the absence of RFI and noise, $\mathbf{R}_s$ represents the true

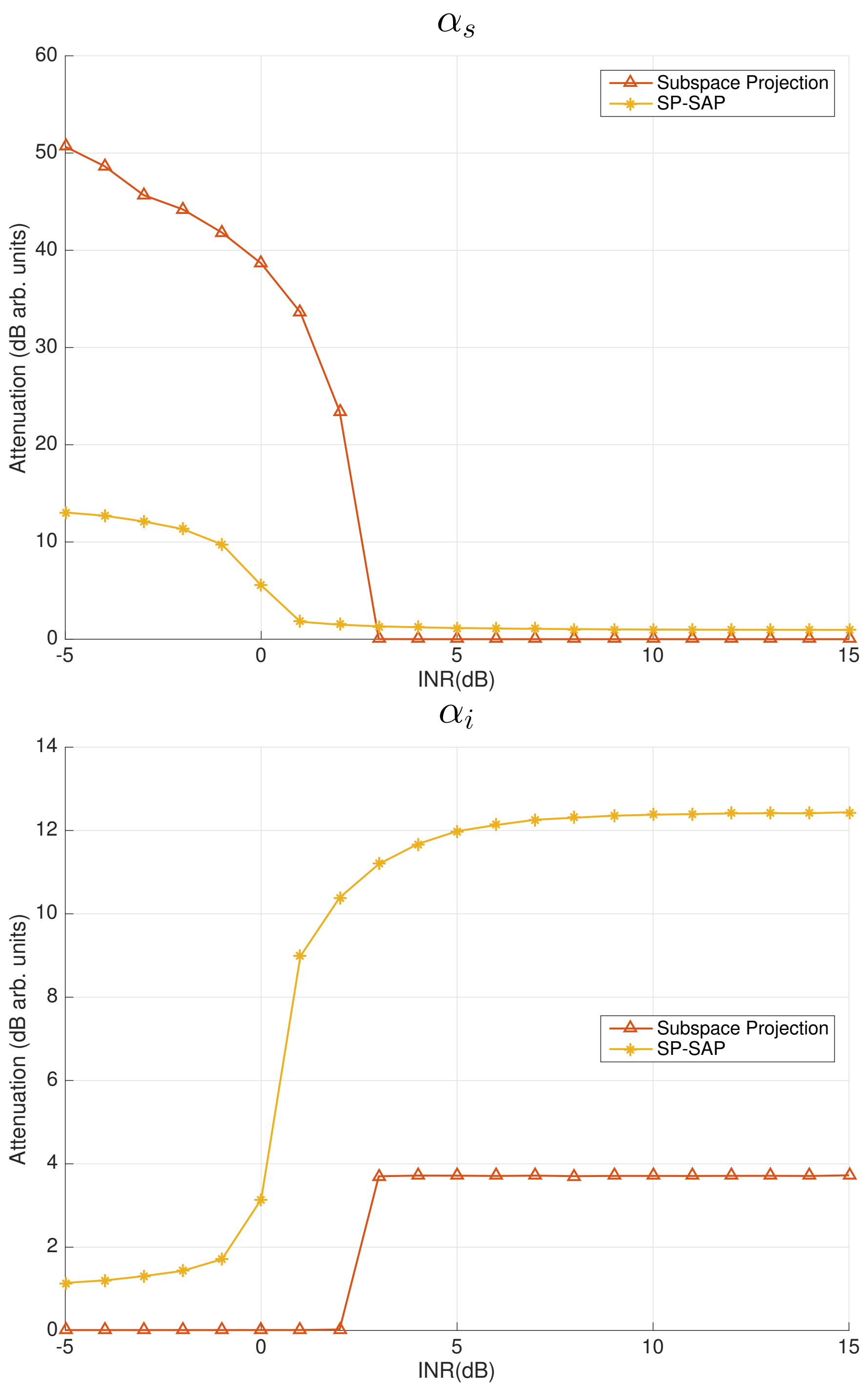


Figure 5.7: Post-mitigation attenuation to the SOI (top) and RFI (bottom). Subspace projection achieves a modest level of attenuation to the RFI while SP-SAP increases the overall mitigation by 9 dB. There is a slight 1 dB of attenuation to the SOI. However, bias corrections can be applied on a per subarray basis to restore the SOI.

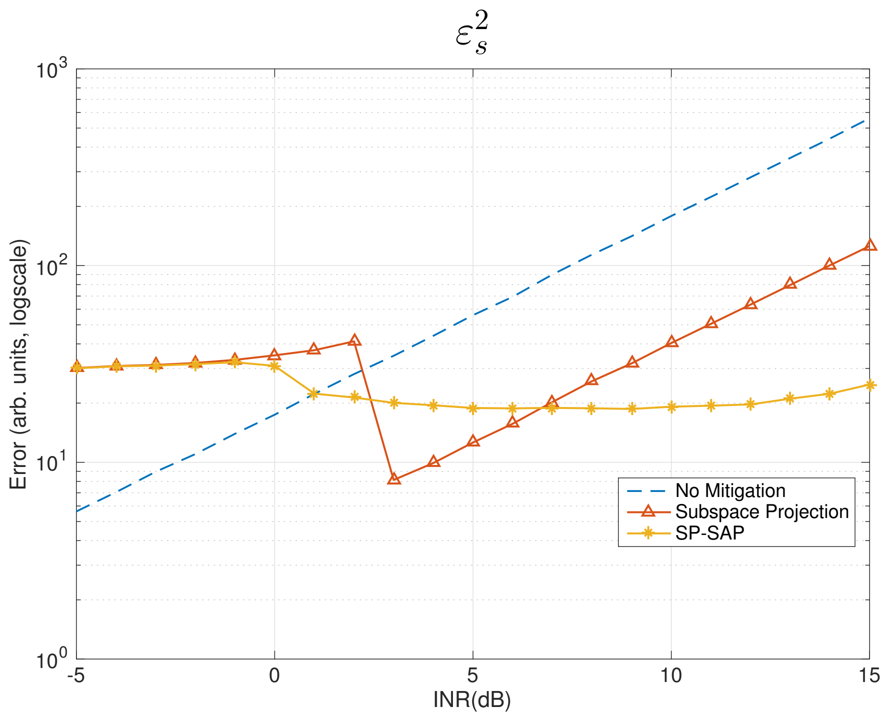


Figure 5.8: Post-mitigation mean square error in the visibilities. Both subspace projection and SP-SAP perform well for low INRs. However, the error in subspace projection begins to diverge at a low INR while SP-SAP achieves lower, stable residuals for a larger range of INRs.

visibility matrix. Therefore, the mean square error measures how close the filtered signal covariance matrix is to the desired visibility matrix. Over the simulated range of INRs, SP-SAP achieves approximately the same residual level. Subspace projection applied directly across the entire array does slightly out-perform SP-SAP for a small range of INR values. However, there quickly comes a point where as INR increases, so do the residuals, implying that subspace projection is no longer effective. The point at which the residuals for SP-SAP begin to increase is beyond practical INR levels. At that stage, other issues such as non-linearity of the LNAs or saturation in the ADCs become more relevant. Thus we may claim that SP-SAP delivers effective RFI cancellation over a wider input INR range than conventional subspace projection.

Figure 5.9 shows a comprehensive simulated imaging example comparing the two algorithms, and illustrates the success of SP-SAP at better recovering the desired image. The simulation shows a 12-hour observation with 30-minute updates and is the same imaging scenario and array geometry as previously described with an INR of 10 dB. The image formation process is a

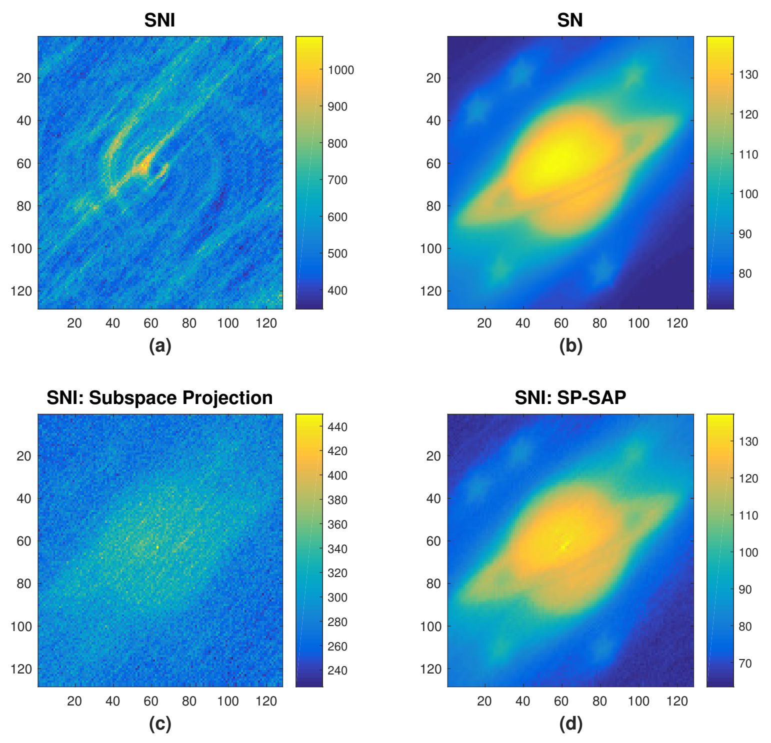


Figure 5.9: Synthetic imaging simulation comparing subspace projection and SP-SAP. (a) Shows the resulting image in the presence of noise and a strong interferer with INR of 10 dB. (b) The image in the absence of RFI. (c) Filtered image after applying conventional subspace projection to the entire array. (d) Filtered image after applying SP-SAP. Interference is still very prominent in (c) as only faint characteristics of the SOI are visible. SP-SAP better recovers the image in the presence of RFI.

brute-force approach using a direct computation of the inverse 2-D Fourier transform to recover the intensity values $I(\rho)$ in (5.4). There is therefore no u-v cell averaging, re-binning, or interpolation onto a rectangular grid or deconvolution to remove the effects of the dirty beam. Figure 5.9 (a) shows the resulting image in the presence of interference and noise, and (b) shows the resulting image in the absence of the interferer. Using projection-based RFI mitigation the goal is to remove interferer artifacts as seen in (a) to produce an image similar to (b), as if only signal and noise were present. Applying $\mathbf{P}_{\text{SP}}$ and $\mathbf{P}_{\text{SAP}}$ to the covariance matrix image of (a) results in Figures 5.9 (c) and (d) respectively. Figure (c) shows that conventional subspace projection removes some interference since faint characteristics of the desired image are beginning to appear. However, SP-SAP results in (d) and more fully recovers the image as there is no apparent evidence of RFI artifacts or corruption to the desired signal.

5.6 Conclusion

In the growing presence of RFI, sensitive synthesis array instruments will need to rely on methods other than flagging. In this chapter we presented the SP-SAP algorithm for large interferometric arrays, which has shown to improve RFI mitigation on the order of 9 dB as compared to conventional subspace projection across the entire array.

Motivated by the increasing amount of strong RFI corrupting important observation frequency bands, we desired to apply array mitigation techniques. Therefore, we started by introducing the signal model and synthesis imaging equations necessary to apply such methods. Following, we analyzed that inherent to large interferometers, the RFI is attenuated due to decorrelation across long projected baselines. This phenomena is dependent upon the processing channel bandwidth where wide channel bandwidths results in more attenuation to the RFI. This introduces an inherit trade-off between the processing bandwidth and attenuation of the RFI across the array. However, for array geometries which use many elements in a dense central core, such as the ngVLA, there will still be significant correlation in the RFI subsequently corrupting observations.

We then provided an introduction to subspace projection. It was shown that subspace projection was ineffective using all of the elements in a large interferometer because as more elements are included where significant decorrelation of the RFI has occurred, sample estimation error becomes the dominant factor degrading subspace parameter estimates. We therefore formulated SP-SAP which places elements in smaller subarrays in planes of constant phase aligned perpendicular to the effective RFI propagation vector. This guarantees high mutual RFI correlation among elements of a subarray which better estimate the interference subspace for subspace projection. The application of SP-SAP in a hypothetical imaging simulation showed to outperform subspace projection, recovering the original image with no apparent evidence of RFI artifacts.

Assumptions made in this chapter, such as known arrival angles for interferers and correlator dump abilities for STIs, are not unrealistic. The interference arrival angle θ_i is only used to determine the propagation vector which determines the effective phase propagation ψ_m for subarray formation. For any satellite or fixed ground-based signal, the arrival angle θ_i is known to high accuracy. Even modest errors in θ_i still yield effective cancellation. Unless the angle of arrival for the interferer is geostationary θ_i will need to be updated regularly for each STI along with the calculation of subarray partitions. In the past decade, hardware for correlator designs have advanced such that many observatories can now support dump intervals on ms time scales. This is valuable to mitigate the effect of interferer motion. A correlator which can support rapid integration dumps is already operational in the VLA receiver. We recommend that a comparable system be considered in the design of the ngVLA so that spatial interference mitigation techniques such as SP-SAP can be utilized. Projection-based RFI mitigation algorithms rely on an eigenvalue decomposition, therefore, in the design of the correlator, it is also important that the antenna self-power terms, or the diagonal entries of $\mathbf{R}_j$ be saved out by the correlator.

Spatial filtering techniques such as SP-SAP and subspace projection can be performed offline as part of post-correlation image forming processes. As has been shown, subspace projection techniques can introduce a bias as part of the estimation process, however, that can be corrected for with known bias correction methods [11]–[13], [57]. There is no risk of data corruption with post-correlation processes because projections can be applied after raw visibilities have been saved.

CHAPTER 6. CONCLUSION

Expanding our knowledge and understanding of the universe, and the physics which govern it, will depend on continuing advancements in hardware and signal processing capabilities for multi-sensor array systems. Also, with ever popular civil and military services such as global navigation, transportation, radar and communication systems, which all use satellites or ground-based transmitters and interfere with the observations of sensitive radio telescopes, it is important to consider coexisting in harsh RFI environments.

This thesis presented the development of FLAG, with an emphasis on the development of the 150 MHz correlator and real-time beamformer digital back end system processing 19 dual-polarized antennas and forming 7 simultaneous dual-polarized beams. FLAG is the first permanent, cryogenically cooled, PAF instrument for a major single dish radio telescope. The commissioning results in Section 2.3 characterize system performance with an unprecedented low T_{sys}/η noise level of 28 K. The survey speed, with the main boresight beam and six other simultaneous beams, will increase by approximately a factor of 3-5x compared to a single beam system. FLAG will therefore be a key instrument to enable significant new science, including the detection of pulsars and a census of diffuse HI around galaxies. Proposals for maps and surveys have been approved and may even start as soon as the end of 2017.

There is yet more work to continue improvements to the digital back end. Section 2.2 and Appendix A.9 outlines current system limitations and suggestions to implement commensal observation mode, which is the concurrent operation of both the fine channel correlator and the real-time beamformer modes. The implementation of an integrated user interface to provide support to the astronomers whom will be operating the system is also important for FLAG to be considered in future proposals.

Future work done with FLAG is not just limited to the science performed, or maintenance to the digital system, but to further advances in signal processing for PAFs. The spatial information in arriving signals, because of spatially located elements, provides the ability to use adaptive array-based projection algorithms for RFI mitigation, such as subspace projection (Section 5.3) [1]. These algorithms can be tested and implemented using current back end hardware configurations and provide, either in post-processing or real-time, interference cancellation. As mentioned previously, RFI plagues many observatories and development of such capabilities for current and future systems using PAFs will greatly benefit their work.

This thesis also presented the SP-SAP algorithm for interference mitigation on large interferometric arrays, which was shown to improve RFI mitigation on the order of 9 dB as compared to conventional subspace projection algorithms.

Future work could show that other array-based spatial filtering algorithms such as oblique projection [60], [61] and cross subspace projection (CSP) [11] can benefit from subarray processing. For example, in CSP, auxiliary antennas improve INR levels and help achieve better RFI mitigation because there is a more accurate representation of the interference parameters [11], [15]. Just as presented here, on larger interferometers, the computed projection operators from these methods when applied to the full array are not as effective because of the decorrelation of RFI across the array and the sample estimation error introduced. CSP can then benefit by using subarray processing and designating one of the array elements in a subarray as the auxiliary to improve RFI cancellation.

In addition, the presented algorithm for designing subarrays, while effective, requires a user defined threshold and there may be a better approach to subarray design. Further investigation into subarray design using statistical methods such as *k*-means clustering [62], or other clustering algorithms, could provide more natural partitions suiting the present observation scenario. Other design considerations could allow subarrays to overlap or use an algorithm which applies SP-SAP for a large threshold, resulting in large subarrays, and then descend iteratively on the phase threshold with smaller and smaller subarrays.

Interest in the use of adaptive RFI mitigation is growing, it therefore may become possible in the near future to apply these techniques on current interferometry systems such as the VLA. This would allow for the validation and further investigation of adaptive spatial filtering techniques on large interferometers. Similar to the FLAG architecture, parallel processing hardware such as GPUs and FPGAs can be adopted into future correlator designs for interferometric arrays. In this case, algorithms such as SP-SAP can easily be adapted to these systems. For example, computing projection matrices for the different subarrays lends itself to a problem that can be solved in parallel. Interferometric arrays with this design would then be able to provide, either in post-processing or real-time, effective cancellation to RFI.

REFERENCES

[1] B. D. Jeffs, K. F. Warnick, J. Landon, J. Waldron, D. Jones, J. R. Fisher, and R. D. Norrod, "Signal processing for phased array feeds in radio astronomical telescopes," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 5, pp. 635–646, Oct 2008. 1, 2, 75

[2] T. D. Webb, "Design of polarimetric calibration of dual-polarized phased array feeds for radio astronomy," Master's thesis, Brigham Young University, August 2012. 1, 8

[3] D. A. Roshi, K. F. Warnick, J. Brandt, J. R. Fisher, P. Ford, B. D. Jeffs, P. Marganian, M. McLeod, M. Mello, M. Morgan, R. Norrod, W. Shillue, R. Simon, and S. White, "Towards the development of a high-sensitivity cryogenic phased array feed," in *2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS)*, Aug 2014, pp. 1–4. 1

[4] ——, "A 19 element cryogenic phased array feed for the Green Bank Telescope," in *2015 IEEE International Symposium on Antennas and Propagation USNC/URSI National Radio Science Meeting*, July 2015, pp. 1376–1377. 1

[5] T. L. Wilson, "The Atacama large millimeter array," in *2007 Joint 32nd International Conference on Infrared and Millimeter Waves and the 15th International Conference on Terahertz Electronics*, Sept 2007, pp. 591–593. 1, 2

[6] D. R. DeBoer, R. G. Gough, J. D. Bunton, T. J. Cornwell, R. J. Beresford, S. Johnston, I. J. Feain, A. E. Schinckel, C. A. Jackson, M. J. Kesteven, A. Chippendale, G. A. Hampson, J. D. O'Sullivan, S. G. Hay, C. E. Jacka, T. W. Sweetnam, M. C. Storey, L. Ball, and B. J. Boyle, "Australian ska pathfinder: A high-dynamic range wide-field of view survey telescope," *Proceedings of the IEEE*, vol. 97, no. 8, pp. 1507–1521, Aug 2009. 1, 2

[7] A. J. Brown, G. A. Hampson, P. Roberts, R. Beresford, J. D. Bunton, W. Cheng, R. Chekkala, D. Kiraly, S. Neuhold, and K. Jeganathan, "Design and implementation of the 2nd generation ASKAP digital receiver system," in *2014 International Conference on Electromagnetics in Advanced Applications (ICEAA)*, Aug 2014, pp. 268–271. 1, 4

[8] D. B. Davidson, "MeerKAT and SKA phase 1," in *ISAPE2012*, Oct 2012, pp. 1279–1282. 1, 2

[9] V. Asthana, "Development of L-band down converter boards and real-time digital back end for phased array feeds," Master's thesis, Brigham Young University, April 2012. 1, 4

[10] R. A. Black, "Digital back end development and interference mitigation methods for phased-array feeds," Master's thesis, Brigham Young University, August 2014. 1, 4

[11] B. D. Jeffs, L. Li, and K. F. Warnick, “Auxiliary antenna-assisted interference mitigation for radio astronomy arrays,” *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 439–451, Feb 2005. 1, 5, 61, 62, 68, 73, 75

[12] A. Leshem, A. van der Veen, and A. Boonstra, “Multichannel interference mitigation techniques in radio astronomy,” *Astrophysical Journal Supplements*, vol. 131, no. 1, pp. 355–374, 2000. 1, 3, 5, 51, 68, 73

[13] A. Leshem and A. van der Veen, “Radio-astronomical imaging in the presence of strong radio interference,” *IEEE Transactions on Information Theory*, vol. 46, no. 5, pp. 1730–1747, Aug. 2000. 1, 3, 5, 51, 61, 62, 68, 73

[14] J. M. Ford and K. D. Buch, “RFI mitigation techniques in radio astronomy,” in *2014 IEEE Geoscience and Remote Sensing Symposium*, July 2014, pp. 231–234. 1, 3

[15] P. A. Fridman and W. A. Baan, “RFI mitigation methods in radio astronomy,” *Astronomy and Astrophysics*, vol. 378, no. 1, pp. 327–344, 2001. [Online]. Available: https://doi.org/10.1051/0004-6361:20011166 1, 55, 75

[16] J. R. Fisher and R. F. Bradley, “Full-sampling array feeds for radio telescopes,” vol. 4015, 2000, pp. 4015 – 4015 – 11. [Online]. Available: http://dx.doi.org/10.1117/12.390425 2

[17] B. D. V. Veen and K. M. Buckley, “Beamforming: a versatile approach to spatial filtering,” *IEEE ASSP Magazine*, vol. 5, no. 2, pp. 4–24, April 1988. 2

[18] A. Thompson, J. Moran, and G. S. Jr., *Interferometry and Synthesis in Radio Astronomy*, 2nd ed. Wiley, New York, 2001. [Online]. Available: https://search.lib.byu.edu/byu/record/lee.2832137 2, 51, 54

[19] P. J. Napier, A. R. Thompson, and R. D. Ekers, “The Very Large Array: Design and performance of a modern synthesis radio telescope,” *Proceedings of the IEEE*, vol. 71, no. 11, pp. 1295–1320, Nov 1983. 2

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

[21] Y. Gupta, “Observatory report for the GMRT,” in *2011 XXXth URSI General Assembly and Scientific Symposium*, Aug 2011, pp. 1–1. 2

[22] C. L. Carilli, “Science working groups project overview,” *Next Generation Very Large Array Memo Series*, vol. 1, no. 5, Oct 2015. 2

[23] L. Hoppmann, L. Staveley-Smith, W. Freudling, M. A. Zwaan, R. F. Minchin, and M. R. Calabretta, “A blind HI mass function from the arecibo ultra-deep survey (AUDS),” *Monthly Notices of the Royal Astronomical Society*, vol. 452, no. 4, pp. 3726–3741, 2015. [Online]. Available: +http://dx.doi.org/10.1093/mnras/stv1084 3, 51

[24] A. J. Otto, R. P. Millenaar, and P. S. van der Merwe, “Characterising RFI for SKA phase 1,” in *2016 Radio Frequency Interference (RFI)*, Oct 2016, pp. 81–84. 3, 51

[25] M. Sokolowski, R. B. Wayth, and M. Lewis, "The statistics of low frequency radio interference at the Murchison radio-astronomy observatory," in *2015 IEEE Global Electromagnetic Compatibility Conference (GEMCCON)*, Nov 2015, pp. 1–6. 3, 51

[26] Offringa, A. R., de Bruyn, A. G., Zaroubi, S. *et al.*, "The LOFAR radio environment," *Astronomy and Astrophysics*, vol. 549, p. A11, October 2012. [Online]. Available: https://doi.org/10.1051/0004-6361/201220293 3, 51

[27] S. W. Ellingson, J. D. Bunton, and J. F. Bell, "Removal of the GLONASS C/A signal from OH spectral line observations using a parametric modeling technique," *The Astrophysical Journal Supplement Series*, vol. 135, no. 1, p. 87, 2001. 3

[28] ——, "Cancellation of GLONASS signals from radio astronomy data," vol. 4015, 2000, pp. 400–407. 3

[29] D. M. L. Vine, J. T. Johnson, and J. Piepmeier, "RFI and remote sensing of the Earth from space," in *2016 Radio Frequency Interference (RFI)*, Oct 2016, pp. 49–54. 3

[30] B. T. Indermuehle, L. Harvey-Smith, C. Wilson, and K. Chow, "The ASKAP RFI environment as seen through BETA," in *2016 Radio Frequency Interference (RFI)*, Oct 2016, pp. 43–48. 3

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

[32] A. Parsons, D. Backer, A. Siemion, H. Chen, D. Werthimer, P. Droz, T. Filiba, J. Manley, P. McMahon, A. Parsa, D. MacMahon, and M. Wright, "A scalable correlator architecture based on modular FPGA hardware, reuseable gateware, and data packetization," *Publications of the Astronomy Society of the Pacific*, vol. 120, p. 1207, nov 2008. 4, 10

[33] W. A. van Cappellen and J. G. B. de Vaate, "Status update on APERTIF, phased array feeds for the Westerbork radio telescope," in *2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS)*, Aug 2014, pp. 1–4. 5

[34] J. Landon, B. D. Jeffs, and K. F. Warnick, "Model-based subspace projection beamforming for deep interference nulling," *IEEE Transactions on Signal Processing*, vol. 60, no. 3, pp. 1215–1228, March 2012. 5

[35] R. A. Black, "Enhanced phased array feed instrumentation and signal processing for astronomical transient parameter estimation and interference mitigation," Ph.D. dissertation, Brigham Young University, December 2017. 5

[36] S. W. Ellingson and G. A. Hampson, "A subspace-tracking approach to interference nulling for phased array-based radio telescopes," *IEEE Transactions on Antennas and Propagation*, vol. 50, no. 1, pp. 25–30, Jan 2002. 5

[37] G. Hellbourg, "Subspace smearing and interference mitigation with array radio telescopes," in *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*, Aug 2015, pp. 278–282. 5

[38] B. D. Jeffs, K. F. Warnick, and L. Li, “Improved interference cancellation in synthesis array radio astronomy using auxiliary antennas,” in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP ’03). 2003 IEEE International Conference on*, vol. 5, April 2003, pp. V–77–80 vol.5. 5

[39] A. Thompson, “The response of a radio-astronomy synthesis array to interfering signals,” *IEEE Transactions on Antennas and Propagation*, vol. 30, no. 3, pp. 450–456, May 1982. 5, 56

[40] R. A. Black, B. D. Jeffs, K. F. Warnick, G. Hellbourg, and A. Chippendale, “Multi-tier interference-cancelling array processing for the ASKAP radio telescope,” in *2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE)*, Aug 2015, pp. 261–266. 5

[41] C. L. Carilli, “The strength of the core,” *Next Generation Very Large Array Memo Series*, vol. 1, no. 12, Oct 2016. 6, 56

[42] M. Morgan, J. Richard Fisher, and J. J. Castro, “Unformatted digital fiber-optic data transmission for radio astronomy front-ends,” *Publications of the Astronomical Society of the Pacific*, vol. 125, 05 2013. 9, 94, 95, 96

[43] “Hashpipe repository,” https://github.com/david-macmahon/hashpipe. 12, 88

[44] “CUDA toolkit documentation,” http://docs.nvidia.com/cuda/, accessed: 2017-08-31. 12, 40, 90, 91

[45] D. J. Pisano, “Green Bank Telescope observations of low column density HI around NGC 2997 and NGC 6946,” *The Astronomical Journal*, vol. 147, p. 48, Mar. 2014. 23

[46] P. P. Vaidyanathan, *Multirate Systems and Filterbanks*. Prentice-Hall, Inc., 1993. 26, 33

[47] F. J. Harris, “On the use of windows for harmonic analysis with the discrete Fourier transform,” *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, Jan 1978. 29, 45

[48] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Pearson, 2010. 29, 33, 34

[49] V. Boyer and D. E. Baz, “Recent advances on GPU computing in operations research,” in *2013 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum*, May 2013, pp. 1778–1787. 39

[50] D. H. E. MacMahon, D. C. Price, M. Lebofsky, A. P. V. Siemion, S. Croft, D. DeBoer, J. E. Enriquez, V. Gajjar, G. Hellbourg, H. Isaacson, D. Werthimer, Z. Abdurashidova, M. Bloss, R. Creager, J. Ford, R. S. Lynch, R. J. Maddalena, R. McCullough, J. Ray, M. Whitehead, and D. Woody, “The breakthrough listen search for intelligent life: A wideband data recorder system for the Robert C. Byrd Green Bank Telescope,” *ArXiv e-prints*, Jul. 2017. 39, 88

[51] R. M. Prestage, M. Bloss, J. Brandt, H. Chen, R. Creager, P. Demorest, J. Ford, G. Jones, A. A. Kepley, A. Kobelski, P. Marganian, M. Mello, D. McMahon, R. McCullough, J. Ray,

D. A. Roshi, D. Werthimer, and M. Whitehead, “The versatile GBT astronomical spectrometer (VEGAS): Current status and future plans,” in *2015 USNC-URSI Radio Science Meeting (Joint with AP-S Symposium)*, July 2015, pp. 294–294. 39

[52] D. Storti and M. Yurtoglu, *CUDA For Engineers*. Pearson, 2016. 40

[53] J. L. Jodra, I. Gurrutxaga, and J. Muguerza, “A study of memory consumption and execution performance of the cuFFT library,” in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, Nov 2015, pp. 323–327. 43

[54] “CUDA toolkit documentation, cuFFT,” http://docs.nvidia.com/cuda/cufft/index.html, accessed: 2017-09-01. 43

[55] S. M. Andrews, D. J. Wilner, Z. Zhu, T. Birnstiel, J. M. Carpenter, L. M. Pérez, X.-N. Bai, K. I. berg, A. M. Hughes, A. Isella, and L. Ricci, “Ringed substructure and a gap at 1 au in the nearest protoplanetary disk,” *The Astrophysical Journal Letters*, vol. 820, no. 2, p. L40, 2016. [Online]. Available: http://stacks.iop.org/2041-8205/820/i=2/a=L40 51

[56] G. B. Taylor, C. L. Carilli, and R. A. Perley, Eds., *Synthesis Imaging in Radio Astronomy II*. Astronomical Society of the Pacific, San Francisco, Calif., 1999. [Online]. Available: https://search.lib.byu.edu/byu/record/lee.2636600 51, 57

[57] J. Raza, A. J. Boonstra, and A. J. van der Veen, “Spatial filtering of RF interference in radio astronomy,” *IEEE Signal Processing Letters*, vol. 9, no. 2, pp. 64–67, Feb 2002. 61, 68, 73

[58] M. Kaveh and A. Barabell, “The statistical performance of the music and the minimum-norm algorithms in resolving plane waves in noise,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 2, pp. 331–341, Apr 1986. 62

[59] A. C. Kot, Y. D. Lee, and H. Babri, “Bias and variance calculation of a principal component based frequency estimator,” *IEE Proceedings - Vision, Image and Signal Processing*, vol. 142, no. 4, pp. 247–251, Aug 1995. 62

[60] R. T. Behrens and L. L. Scharf, “Signal processing applications of oblique projection operators,” *IEEE Transactions on Signal Processing*, vol. 42, no. 6, pp. 1413–1424, Jun 1994. 75

[61] G. Hellbourg, R. Weber, C. Capdessus, and A. J. Boonstra, “Oblique projection beamforming for RFI mitigation in radio astronomy,” in *2012 IEEE Statistical Signal Processing Workshop (SSP)*, Aug 2012, pp. 93–96. 75

[62] S. Lloyd, “Least squares quantization in PCM,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, March 1982. 75

[63] M. H. Clark, “Control software architecture for the Green Bank Telescope,” in *Proc. SPIE*, vol. 3351, 1998, pp. 3351 – 3351 – 10. [Online]. Available: http://dx.doi.org/10.1117/12.308829 83

[64] K. O’Neil, A. L. Shelton, N. M. Radziwill, and R. M. Prestage, “The astronomer’s integrated desktop: A unified suite of applications for scheduling-block based observing with the GBT,”

in *Astronomical Data Analysis Software and Systems XV*, ser. Astronomical Society of the Pacific Conference Series, C. Gabriel, C. Arviset, D. Ponz, and S. Enrique, Eds., vol. 351, jul 2006, p. 719. 84

[65] "DIBAS documentation," http://www.gb.nrao.edu/ mwhitehe/dibas/html/index.html, accessed: 2017-08-25. 84

[66] "Zeromq documentation," http://api.zeromq.org, accessed: 2017-08-25. 84

[67] G. E. Krasner and S. T. Pope, "A cookbook for using the model-view controller user interface paradigm in smalltalk-80," *Journal of Object Oriented Program*, vol. 1, no. 3, pp. 26–49, Aug. 1988. [Online]. Available: http://dl.acm.org/citation.cfm?id=50757.50759 87

[68] R. DuPlain, S. Ransom, P. B. Paul Demorest, J. Ford, and A. L. Shelton, "Launching GUPPI: the Green Bank ultimate pulsar processing instrument," in *Proc.SPIE*, vol. 7019, 2008, pp. 7019 – 7019 – 10. [Online]. Available: http://dx.doi.org/10.1117/12.790003 88

[69] "xGPU repository," https://github.com/david-macmahon/xgpu. 90

[70] J. G. Proakis and M. Salehi, *Digital Communications*. McGraw-Hill, 2008. 96

APPENDIX A. FLAG DEVELOPMENT REFERENCE

A.1 Introduction

Chapter 2 provided a high-level overview of the FLAG system architecture and its capabilities from signal acquisition at the PAF, to raw correlations and time-averaged beamformed power spectra saved on disk. However, much of the implementation details for the Hashpipe, GPU and python back end codes were not demonstrated earlier for a clearer presentation of the FLAG system capabilities and performance. Also, In order to acquire useful astronomical data and become a commissioned instrument as part of the arsenal of receivers and back ends available for target science at GBO, it is required that FLAG be able to interface with established system-wide control tools. Because different processing back ends are not expected to have similar behavior, there is no default implementation and this requires unique software tools in each back end. The purpose of this appendix is to provide relevant background of the GBO telescope control system architecture, implementation details to interface with that control system, and document background for FLAG data processing software tools and frameworks.

A.2 Scan Coordinator and GBT Telescope Control System

The integrated telescope control system that interfaces with and operates the GBT consists of three major components: "Scan Coordinator" (SC), ASTRID, and an SQL database "GBT Status."

The SC sits at the top of a modular hardware and software hierarchy called the "GBT Monitor and Control System" [63]. The Monitor and Control System incorporates different managers for the control of critical hardware on the telescope such as the rest frequency for the Local Oscillators (LOs), dish steering, receiver selection, the dish's active surface, and the PAF itself. The SC is responsible for configuring and commanding these various managers to setup the systems to the desired astronomical configuration for observation.

ASTRID (Astronomers Integrated Desktop) [64] is an interactive python-based scripting environment capable of configuring the SC. The astronomer writes an ASTRID script detailing the astronomical configuration and the movement of the telescope in a subsequent scan. The script is submitted as a “scheduling block” to the SC. When the scheduling block gets to the top of the queue, the SC executes the script, issuing configuration commands and observation directives, such as the start and stop of a scan to the telescope. A “scan” is one fundamental period of coherent, related data collection, which may involve commanded dish motion or tracking. The various managers of the GBT Monitor and Control system log FITS files during a scan detailing metadata for post-processing such as telescope position encoder values. Meanwhile, the scan coordinator is also pushing other configuration metadata, such as scan start and stop timestamps to the GBT status database for the back end. It is therefore the job of the back end system to query the SC and GBT status for updates on telescope operation.

A.3 The Distributed Back End System (DIBAS) Architecture and Dealer/Player

To accommodate integration for different back end systems at GBO, they have provided a Python software framework known as DIBAS (Distributed Back End System) [65]. Each processing pipeline for a node connects to the GBO network as a ZeroMQ [66] server and is designated as a “player.” Another program, which may run on any machine on the GBO network, becomes a lightweight client that connects to each player. This client is capable of interacting, configuring, and coordinating the players and is designated as the “dealer.” This creates the dealer/player relationship.[1] A player is responsible for running a “bank,” which is one instance of an available back end type, or class, in DIBAS. Each back end type is equipped with different operational modes and represents one of the modes for a Hashpipe instance as described in Section 2.1.

The file `dibas.conf`, located in the `etc/conf` directory of the root DIBAS directory, contains the configuration details for the target DIBAS back end: node IP address, bank destination

[1]The nomenclature is a legacy feature from when DIBAS was first upgraded to incorporate many back ends. At the time, VEGAS was under development inspiring the Las Vegas theme.

IP addresses, ROACH bitstream file name, and available operational modes, etc. As the back end operates, it will query the configuration file for information as needed.

The following is an example of the typical work flow in the DIBAS framework: A ZeroMQ proxy server and a bank are created, defining a player. The player waits for a directive from the dealer as to which back end mode its bank should run. The player propagates this directive to the bank and it creates a back end based on the selected type. The back end is then returned to the player and notifies the dealer that the nodes data processing pipeline is running, waiting for commands. The dealer can then configure mode parameters such as integration length and beamformer weight directory. After all parameters have been set, the dealer could then command the player to start a scan. Commands continue to be issued by the dealer until either the dealer disconnects from the player, the back end object for the player is destroyed, or the program running the player terminates.

A.4 Beamformer Back End

The addition of FLAG to DIBAS required creation of a new back end type and is called Beamformer Back End (BFBE). The operational modes for BFBE are defined in DIBAS and the complete description of the modes, their implementation, and interaction as a player are defined in `Beamformerbackend.py`. This includes all parameters that are to be set by the dealer, such as correlator integration length, beamformer weight directory and channel selection for the fine PFB mode.

The DIBAS framework works well for FLAG by offering many “out-of-the-box” capabilities. The dealer/player relationship, for example, is critical for interfacing with SC. However, FLAG deviates in its design from many underlying assumptions DIBAS was built on, preventing it from providing much of the base back end class features to BFBE.

For example, the new unformatted data sequence coming from the blades and the bit/byte/-word lock solution (Appendix B) requires coherent and frequent interaction with the ROACH to complete correctly. DIBAS was also built under the assumption of a one-HPC-to-one-ROACH relationship. This is not the case for FLAG since each ROACH streams packets to all nodes, where it is again preferred that a predictable access pattern to the ROACH is guaranteed. The solution has been to have one bank manage all configuration and interaction with the ROACH boards.

Integrated control and configuration for each ROACH is provided in the new object class RoachDoctor defined in `RoachDoctor.py`. The DIBAS configuration file specifies the bank designated to create a RoachDoctor instance to program, set parameters, configure the network, etc. for each ROACH.

To read the DIBAS configuration file, the default behavior is that anytime a back end needs information from the configuration file it would reopen and query the file. To improve this, BFBE uses the new object class DibasParser defined in `DibasParser.py`. At runtime, a new DibasParser instance is created as a member object of each banks' back end object. All at one time, DibasParser opens and extracts the information from the configuration file and is stored as a python dictionary accessible throughout the lifetime of the back end. A python dictionary as a member object is preferable because it is then accessible across all BFBE functions that work together to initialize the target operational mode.

A.5 Scan Overlord

Integration of the digital back end with SC is crucial because the metadata of a scan and instructions to the telescope to start a scan is essential to collecting useful data. The SC was introduced in Section A.2 and is the interface broadcasting controls to the telescope over the GBO network. The dealer is the connection between the banks of the digital back end and the commands issued by SC. Because each back end type is not expected to have the same behavior from one observation to the next, there is no default behavior or implementation of a dealer using status updates from SC in DIBAS.

For FLAG, Scan Overlord is the implementation of a dealer that listens to commands issued to the telescope by SC. Scan Overlord listens for messages indicating a status change like the start of a next scan, or that the current scan has just finished. Other metadata such as the project file path for the data, project name, etc. are pushed by SC to the GBT status SQL database.

This information is also relevant to the operation of the digital back end. Scan Overlord implements another new object class, FlagCommander, defined in `FlagCommander.py`. This class is a simple interface which manages the connection to the SQL database and frequently queries the database for new metadata information. Scan Overlord uses the dealer to update relevant metadata in shared memory for the players to use.

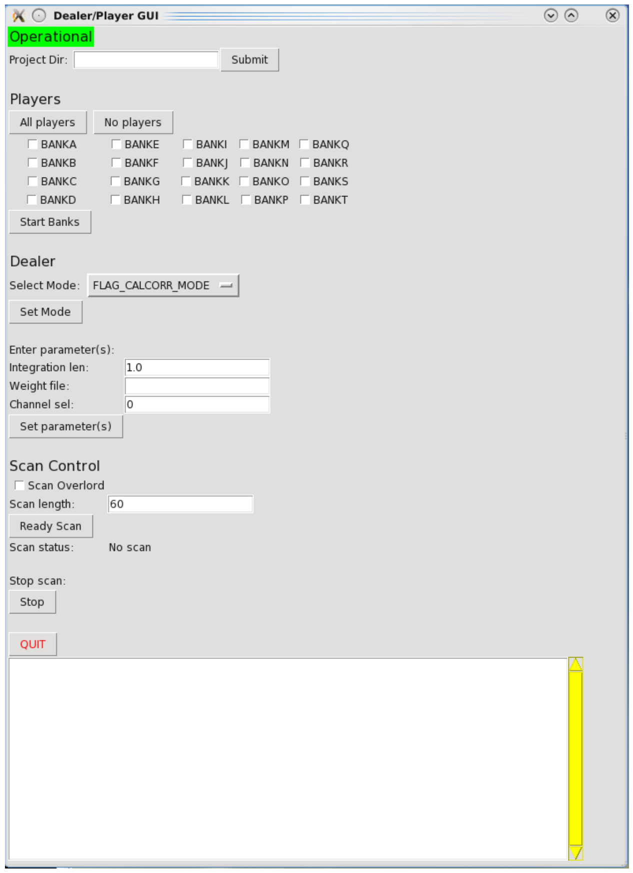
A.6 FLAG Back End Graphical User Interface

Starting up and operating FLAG on the command line, as an "expert user mode," has been the primary procedure for testing all the software and functionality of FLAG during commissioning experiments. A graphical user interface (GUI) was created in order to facilitate the use of FLAG. The GUI can be thought of as an "interactive dealer" which replaces command line interaction with the system for a more user friendly experience. It provides the capability to start any or all players on the FLAG nodes, set operational modes, and run scans.

The GUI uses a standard Model-View-Controller (MVC) design [67]. The model is the data object that the user wants to manipulate in the interface. For FLAG, this would be the different players that the user desires to start up and issue commands to. The view is what is actually presented to the user to interact with. Then, the controller is made up of both the dealer and Scan Overlord (with added functionality to interact with the view) to pass directives from the user to the model (the players) and vice versa.

For the commissioning experiments in expert mode, the back end system is running 20 player as active blocking processes(i.e 20 different terminals running a player). The players alone are therefore not a sufficient model object because they are not able to operate concurrently under one controller. The object class FlagColorguard, defined in `FlagColorguard.py`, is a container for the different players. As a container, this class operates as the root thread managing the initialization of each player as an independent thread.

This GUI was tested and used during the May commissioning experiments. During these experiments we experienced problems with GUI such as hanging and not being able to close all player instances correctly. At the time however, it was inconclusive if these problems were related to the GUI because this was before the application of fixes which resolved race conditions in the Hashpipe processes causing similar issues when running from the command line. The GUI was also not revisited prior or during the August commissioning experiment. Further work is needed to test and identify improvements to this capability in order to promote FLAG as a system astronomers can easily operate.



Project Dir:

Select Mode: FLAG_CALCORR_MODE

Integration len: 1.0

Weight file:

Channel sel: 0

Scan length: 60

Scan status: No scan

Figure A.1: Graphical user interface for operating FLAG.

A.7 Hashpipe

The software package Hashpipe is a real-time processing framework specialized for pipeline processing [43]. Hashpipe was first developed for RA applications with a primitive version first being implemented as part of the GUPPI (Green bank Ultimate Pulsar Processing Instrument) data acquisition system [68]. It has since evolved to a more generic software package but is primarily used in RA back end applications. The package is implemented in the VEGAS data acquisition system as well as the Breakthrough Listen Digital Recorder [50].

Hashpipe is a framework which divides consecutive pipeline processing into different threads, and handles system level functionality such as setting up shared memory segments and semaphore arrays for shared memory. The shared memory segments are implemented as a ring-buffer and are shared between two consecutive threads. Threads then have the shared buffer relationship that its input data buffer is the output data buffer of the preceding thread. The operation of the threads

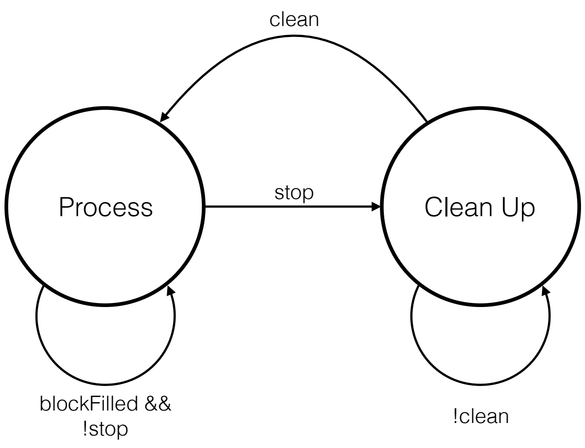


Figure A.2: Tasking the operation of threads in Hashpipe is modeled by a state machine and is driven by waiting for the output buffer to have a filled block. When a block is filled, the thread takes that block as input, performs its task and fills its output buffer for next thread to process.

is then tasked by running a state machine waiting for either data to become available at the input buffer, or an external "stop" command from the fifo. Figure A.2 shows a generalized example of the state machine implementation for a thread. The thread processes filled blocks from the input buffer until a stop is issued. It then performs any clean up necessary such as freeing all filled blocks in the data buffer.

An application of Hashpipe combines two or more threads together in the processing chain and represents one of the operational modes of the back end system. These applications are created as shared library "plugins." Figure A.3 shows a generalized Hashpipe plugin. When a player gets the directive to set the mode, BFBE initializes the Hashpipe plugin associated with the desired mode.

In the FLAG implementation of Hashpipe a plugin starts when a player gets the set mode directive. That players bank initializes BFBE and starts the plugin corresponding to the desired mode. The first thread in each FLAG plugin is the "net thread" and is responsible for capturing the UDP packets from the network. This thread parses packet headers and determines the destination location in the data buffer. When a contiguous block of data is filled the net thread marks the block as filled and is now eligible to be passed to subsequent threads to produce the covariance matrices and time-averaged beamformed spectra.

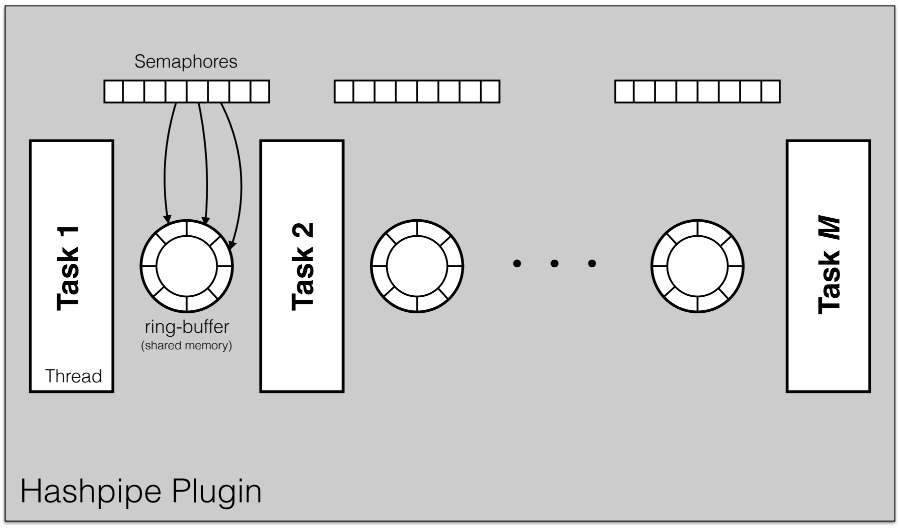


Figure A.3: Generalized plugin. A Hashpipe application has are a series of two or more threads with a shared memory segment implemented as a ring buffer shared between the two threads.

A.8 The FLAG GPU Library

The downstream threads run the processing codes which produce the correlation outputs and time-averaged beamformed spectra. These codes are implemented in CUDA to run on GPUs in order to support the large data rate and keep up with I/O requirements. CUDA is a hardware and software architecture developed by NVIDIA [44] which enables GPUs to be programmed in C/C++, both being higher level languages. This parallel computing architecture and how to apply the model framework to specific problems was presented in Chapter 4.

There are three GPU libraries which have been developed for signal processing and collectively are known as the FLAG GPU library. They are the PFB, Beamformer, and xGPU libraries ("x" for correlator). The xGPU library, capable of producing correlation matrices, is adapted from the implementation by David McMahon [69]. For coarse and fine channel correlations different versions of xGPU are compiled to be compatible with the data formats. Each library has been programmed and compiled to be C compatible and are callable by a single function, that at minimum, requires as arguments the input pointer to the current block of data to be processed and the output pointer to the destination block location in the threads output buffer. The Hashpipe threads that perform these operations are responsible for any initialization that is required for the libraries.

A.9 Performance Tuning, and Improvements

Chapter 2 reported on the current performance of FLAG and generally identified processing bottlenecks. This section identifies more specifically current bottlenecks with suggested approaches to improve system performance.

A.9.1 GPU Library Improvements

The individual operations that make up the GPU library are written to interface directly with the thread in Hashpipe that calls it. This requires that each thread copy data to the GPU, the GPU then processes the data, and when finished, is copied back to the CPU. This works well for modes where there is only one thread that calls a GPU library, such as the calibration correlator. However, when multiple threads are chained together, each implementing a different library, (e.g. the fine channel correlator or commensal operation) there are now multiple instances of copying data on and off the GPU. It is well known that the time required to make memory transfers from the CPU to GPU and vice versa are often many times greater than the time to actually process the data on the GPU.

To overcome this, the GPU libraries should be refactored to be compatible with consecutive function calls without moving data off the GPU as well as leverage the optimization tools available in CUDA. The language is mature and offers the built-in ability to pipeline data using “streams” [44]. Streams are included as a configuration call to individual GPU functions that specify where the kernel operates in the pipeline. When no stream is specified in the kernel call the default “stream 0” is used.

An execution time line for fine channel correlator and the improvements that can be made by using streams and grouping library function calls is shown in Figure A.4. First, the current implementation of the fine channel correlator, which uses both the PFB and correlator GPU libraries, is shown following a block of data through the processing chain using the default stream with time advancing to the right.

By first refactoring the libraries to be compatible and process data in a single function call without the extraneous copy off, then back onto the GPU, one-third of the operations to produce a single output are eliminated. Although the time taken to complete each task is shown here to be

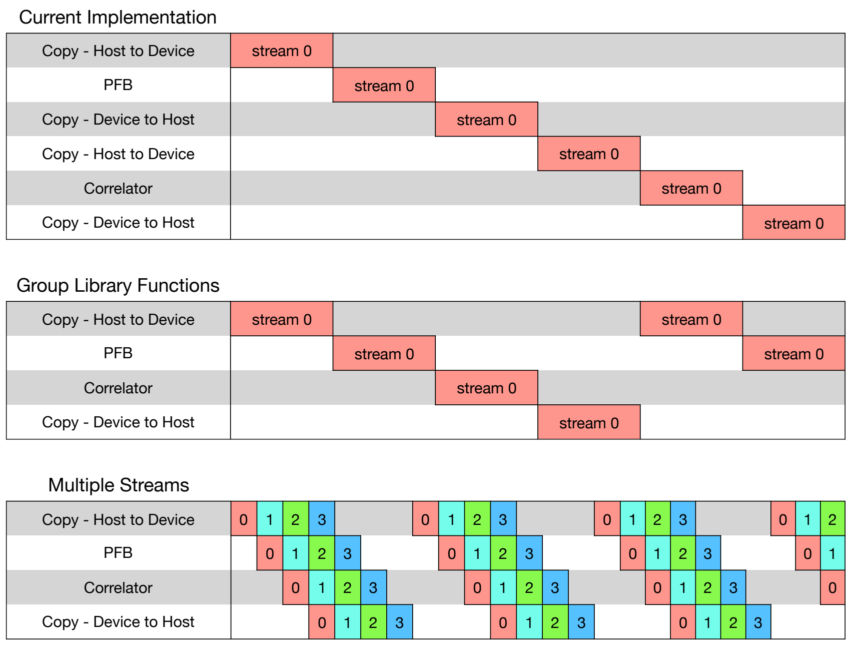


Figure A.4: Improvements can be made in data throughput by grouping calls to GPU library functions and implementing CUDA streams.

equal, as mentioned previously, copies on and off the device are among the most costly operations in GPU processing. This improvement alone will result in a significant improvement.

Further improvements can then be made by using multiple streams. In this example, four streams are used to distribute processing a single data block in smaller portions across parallel kernel calls. Each of the smaller portions is assigned one of the four streams. The GPU tasks and manages the execution of the streams and attempts to run them in parallel as often as possible. As a task in a stream completes the next task is executed. The correlator library currently applies streams to efficiently implement correlation operations but also would need to be refactored to be compatible with other processes to be streamed with it.

We are confident that the data pipeline can be improved by better utilizing the GPU. For example, in preparation for the August 2017 commissioning we noticed that the latency for the thread that transposes the data prior to GPU processing was operating at the edge of the requirement for real-time beamforming. We then removed the transpose thread and implemented this operation directly in the beamformer library. This reduced the number of stalling Hashpipe instances in this mode.

A.9.2 Hashpipe Improvements

Hashpipe plays a critical role by capturing packets from the network and buffering the data between processing threads and writing to disk. We have noticed that stalling Hashpipe instances are caused between the net thread, which captures packets, and downstream processing threads. This happens either because the downstream processes are lagging behind the memory blocks that the net thread fills up, or the net thread is not capturing packets fast enough. We have observed both cases.

There are a few potential ways to diagnose this problem. First, Linux offers the ability to provide "memory mapped sockets." When using memory mapped sockets, memory buffers are created in kernel space and mapped into the current user process's address space without the need to call the standard system function `recv()` to capture packets. Another feature that the network interface (NIC) in Linux offers is the ability to modify how the NIC interrupts the CPU. For example, because of the high data rate, the NIC is frequently interrupting the CPU to give priority to the net thread. Other processes cannot be shared on the same core as the net thread. This is known as "interrupt coalescing" and can be configured using the Linux `ethtool` utility.

Another possible bottleneck is that the block sizes in the ring buffer could be inappropriately configured. The FLAG implementation of Hashpipe inherits the default data type for the data buffer data that is used as the ringbuffer and redefines the size, and number of blocks used. More verification and tuning of this parameter would result in better data hand off between threads.

APPENDIX B. BIT, BYTE AND WORD LOCK

B.1 Introduction

As mentioned in Section 2.1 the analog signals from the PAF are digitized at the telescope by five integrated electronic receiver systems, each known as a blade. This front-end system is comprised of all the non-cryogenic components (i.e. mixers, ADCs). It produces complex base-banded sample voltages of array antenna terminal outputs for transmission over optical fiber to the back end system located in Jansky Lab. The samples are serialized into 8-bit real and 8-bit imaginary components (one byte each) with an offset binary encoding and are combined to form a 16-bit word per time sample (2-byte complex sample frame). The stream of serialized data are transmitted over optical fiber to the back end processing system without any encoding, pilot symbols, start or stop bits, or metadata. The result is a stream of unformatted data requiring that the back end ROACH boards determine proper bit, byte, and 16-bit word boundaries for arriving samples prior to processing.

In RA, the received astronomical SOIs are modeled as circularly symmetric complex Gaussian random processes and the techniques for correctly identifying these bit, byte, and word boundaries leverage the well-known mathematical and statistical properties of this signal model. This appendix presents the algorithms and procedure for identifying bit, byte, and word boundaries, and discusses the tools available to achieve this, and their implementation in the FLAG back end to accommodate this requirement. Figure B.1 abstracts the discussion to follow into a graphical illustration for reference. More detail about the initial design, development, and testing of this front-end receiver and transmission system can be found in [42].

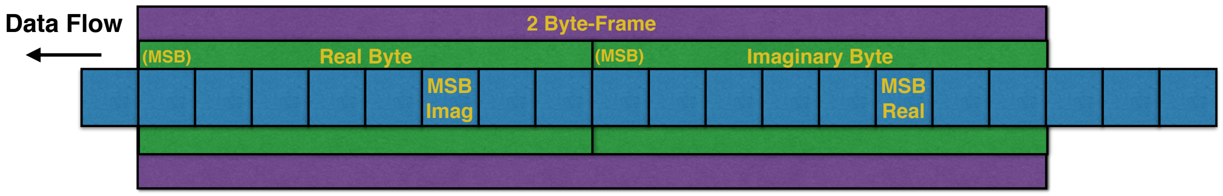


Figure B.1: Data being transmitted from the front-end receiver to the digital back end is unformatted. The back end therefore has no information about how to align the 16-bit words (samples), or the MSB for the real and imaginary byte that make up the sample. Using statistical and mathematical markers of the Gaussian signals in RA the back end must detect and determine bit, byte, and word boundaries. In this example, the data needs to be bit and byte locked. The symbols need to be shifted five times until the MSB is aligned in the correct position, i.e. at the far left of each byte. After bit lock, the MSB for the real and imaginary bit are incorrectly identified. The bits are then shifted eight more times to shift the incorrect byte out of the frame, correctly bringing a real and imaginary byte into alignment.

B.2 Bit Lock

The continuous-time BPSK encoded optical communications link which carries serialized sample data of the signal arrives at the ROACH and first encounters clock recovery circuitry which detects symbol (bit) transitions and produces a stream of OOK (On-Off Keying) symbols. Assuming the received signal is well-conditioned (i.e. not saturating the ADCs at the front-end receiver) and is a zero-mean Gaussian process, the probability that a stream of symbols from raw samples contains a long sequence of identical symbols is small. For the 8-bit real and 8-bit imaginary offset binary encoded samples, it is shown in [42] that this small probability of a long sequence of repeated symbols results in a sufficient number of symbol transitions such that no further adjustments are required to achieve clock recovery.

After clock recovery, the incoming stream of symbols are then deserialized into 16-bit words for processing. Each word represents a complex 2-byte sample frame with a designated real and imaginary byte. However, the deserializer in the back end receiver has no information as to which symbols correspond to the intended most-significant bits (MSBs) with respect to the sample frame sent by the transmitter. The process of aligning symbols such that an MSB from the transmitter is correctly identified as an MSB at the receiver is called "bit lock."

Without any prior knowledge, the best the receiver can do is begin to arbitrarily collect sequences of eight consecutive symbols to form bytes. But since there is no distinction between these bits, this arbitrary partitioning will likely align the transmitted MSB somewhere other than the intended (MSB) location. An example of this is shown in Figure B.1. Here the back end receiver arbitrarily groups eight consecutive bits from the data stream to form bytes, but MSBs that were sent by the transmitter are misaligned by five bit locations. Correctly detecting and aligning the MSB into the correct position depends on a statistical analysis of the incoming samples.

Recall that the received signal is a circularly symmetric complex random process. Thus real and imaginary components are statistically independent and independently distributed (iid) with zero-mean [70]. Therefore, the real and imaginary bytes of a word (sample) arriving at the back end receiver are uncorrelated and jointly Gaussian. If the bytes have been grouped from arriving symbols with the MSB correctly aligned, then a histogram of sample byte values would match the canonical shape of the probability density function (pdf) for a Gaussian distribution.

Further, the MSB can be detected by the correlation between adjacent symbols. Consider the histogram for simulated sample values shown in Figure B.2 for a 4-bit offset binary encoded scheme. Drawn from a jointly Gaussian random process distributed $\mathcal{N}(0,2)$ a 4-bit rather than 8-bit scheme is shown to simplify the illustration. Recall that the integral of a pdf over a specific range of values returns the probability that realizations of a random process fall within that range. Notice that for the middle half of the offset binary encoding table the MSB and second most-significant bit are not the same symbol. These codes are assigned to the most probable range of histogram bins, so these codes will be more prevalent in the data stream. The less probable codes have repeated bits (11XX or 00XX) in the two most-significant locations.

The exact probabilities that any given bit will not equal an adjacent bit is calculated in [42], and it is shown that for typical power values (σ^2) in radio astronomical applications the probability that the MSB and second most-significant bit will not be the same is close to unity. The probability that any other bit is not equal with its adjacent bit is either 0.5 or less.

The algorithm to test for bit lock compares the MSB and second most-significant bit for several bytes, and if they are almost always not equal, then the MSB is correctly aligned. If the MSB is misaligned then the MSB will not equal the second most-significant bit about half the time. In this case, symbols are shifted one symbol at a time (slipped) and re-tested until the MSB

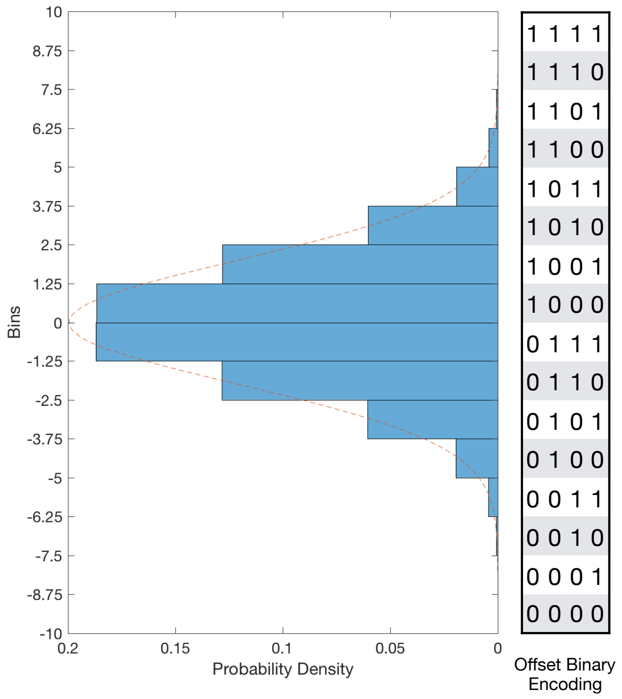


Figure B.2: Histogram for samples from a jointly Gaussian random process with $\mu = 0$ and $\sigma^2 = 2$. An offset binary encoding scheme for 4-bit samples is also shown. Notice that the range of bins where samples are most likely to occur correspond with sample codings where the MSB and second most-significant bit are not equal.

is correctly aligned. For the example in Figure B.1, the symbols would be slipped five times, until MSBs are aligned correctly at the beginning of a byte.

B.3 Byte Lock

After achieving bit lock, the receiver groups two consecutive bytes together and designates them respectively as alternating real and imaginary bytes for the complex 2-byte sample frames used in processing. However, there remains an ambiguity because we do not know which bytes were intended to be real (or imaginary) at the transmitter. The receiver may unknowingly designate imaginary bytes as real bytes, and real bytes as imaginary bytes.

Again, this is shown in the example of Figure B.1. After slipping the symbols five times to correctly align the MSBs, the imaginary bytes are incorrectly identified by the receiver as real bytes, and vice versa. The process of correctly detecting the real and imaginary bytes is known as "byte lock."

Consider the complex baseband signal $x(t)$ and its complex valued Fourier transform $X(f)$ where $x(t)$ can be represented as an analytic signal with in-phase and quadrature components $i(t)$ and $q(t)$

$$x(t) = i(t) + jq(t), \tag{B.1}$$

where $j = \sqrt{-1}$. If the in-phase (real) and quadrature (imaginary) components were swapped, as in the case when the receiver has incorrectly identified real and imaginary bytes, this forms the new signal

$$y(t) = q(t) + ji(t). \tag{B.2}$$

Signals $x(t)$ and $y(t)$ have the relationship

$$y(t) = jx^*(t). \tag{B.3}$$

Taking the Fourier transform of Equation (B.3),

$$\mathscr{F}\left\{y(t) = jx^*(t)\right\}$$
$$Y(f) = jX^*(-f), \tag{B.4}$$

and then taking the magnitude of the resulting spectrum yields

$$|Y(f)| = |X(-f)|. \tag{B.5}$$

Equation (B.5) can be interpreted to mean that in a magnitude spectrum, if we are expecting $|X(f)|$, but instead observe $|X(-f)|$, then we are in fact analyzing $y(t)$ where the real and imaginary parts of $x(t)$ have been incorrectly identified, i.e. we are not byte locked. Symbols need to be slipped eight more times to advance the incorrect byte out of the frame and correctly align real and imaginary bytes. For the example of Figure B.1, after bit lock, the bytes are incorrectly identified, slipping the symbols eight more times correctly aligns the real and imaginary bytes.

The algorithm for byte identification is to inject a relatively strong tone at a known frequency relative to the LO frequency, for example, with a LO setting of 1450 MHz the tone may be above the LO at 1500 MHz. If an image of the tone is symmetrically present below the LO at 1400 MHz, then the bytes are incorrectly identified. The symbols are then slipped eight bit locations to correctly align the bytes.

B.4 Word Lock

After bit and byte lock, word boundaries have been correctly identified. However, due to the variation in clock recovery across antenna inputs to all the ROACHs, different numbers of bit slips are required to bit and byte lock each antenna. This induces a variable sample offset between data sequences which can change with each new round of bit/byte lock calculations. For proper beamformer operation, the array samples must be synchronous and any differential sample delays need to be repeatable between beamformer calibration and subsequent observations. If there are different sample delays in antenna data sequences between calibration and observations, the previously calculated beamformer calibration weights are invalid due to the different phase response in the new observation data. “Word lock” is the process of inserting single sample delays (shifts of 16-bits) as needed in each data stream to achieve a repeatable state of sample delays across the array. Properties of the Fourier transform are utilized to detect word lock and determine the sample offsets relative to a selected single reference element.

Consider the noise source at the front-end receiver array which may be injected into each element. Let samples of this Gaussian random process, denoted as $z[n]$, be distributed with mean $\mu = 0$, and variance $\sigma^2 = \sigma_z^2$. In the absence of other sources and sample offsets, data vector $\mathbf{x}[n]$, representing received complex basebanded sample voltages at the back end system from the M antenna output terminals, is

$$\mathbf{x}[n] = \begin{bmatrix} x_0[n] \\ x_1[n] \\ \vdots \\ x_M[n] \end{bmatrix} = \begin{bmatrix} z[n] \\ z[n] \\ \vdots \\ z[n] \end{bmatrix}. \tag{B.6}$$

Every element ideally sees the same sequence at the same time sample. However, this is not the case. There is an unknown sample offset, τ_m, in each element sequence that has been introduced because of the bit/byte lock solutions and fixed propagation delays in the noise source distribution system. The resulting vector is then,

$$\mathbf{x}[n] = \begin{bmatrix} z[n-\tau_0] \\ z[n-\tau_1] \\ \vdots \\ z[n-\tau_M] \end{bmatrix}, \tag{B.7}$$

where τ_m indicates the integer sample delay for the mth element index.

The back end takes vector $\mathbf{x}[n]$ and produces samples of channelized baseband data at a decimated sample rate. By the time shift property of the Fourier transform, the decimated time samples of channelized data can be represented as,

$$\mathrm{DFT}\Big\{\mathbf{x}[n]\Big\} = \mathbf{X}_k[n'] = \begin{bmatrix} Z_k[n']e^{-j\Omega\tau_0} \\ Z_k[n']e^{-j\Omega\tau_1} \\ \vdots \\ Z_k[n']e^{-j\Omega\tau_M} \end{bmatrix}, \tag{B.8}$$

where $n' = \frac{n}{K_c}$, K_c is the total number of narrowband frequency channel bins, $\Omega = \frac{2\pi f_k}{f_s}$, f_k is the absolute sampled frequency value, $k = 0, 1, \ldots, K_c$ are the channel bin indexes, and f_s is the sample rate.

The channelized data are processed by the correlator producing the covariance matrix $\mathbf{R}_k$ for the kth frequency bin,

$$\mathbf{R}_k = E\Big[\mathbf{X}_k[n']\mathbf{X}_k^H[n']\Big] = \begin{bmatrix} S_{z,k}[0] & \ldots & S_{z,k}[0]e^{-j\Omega(\tau_0-\tau_M)} \\ \vdots & \ddots & \vdots \\ S_{z,k}[0]e^{-j\Omega(\tau_M-\tau_0)} & \ldots & S_{z,k}[0] \end{bmatrix}, \tag{B.9}$$

where $S_{z,k}[0] = \sigma_z^2$, and is the power spectral density for the noise source evaluated at time sample lag zero. Equation (B.9) is simplified to be,

$$\mathbf{R}_k = \sigma_z^2 \begin{bmatrix} 1 & e^{-j\Omega(\tau_0-\tau_1)} & \dots & e^{-j\Omega(\tau_0-\tau_M)} \\ e^{-j\Omega(\tau_1-\tau_0)} & 1 & & \vdots \\ \vdots & & \ddots & \vdots \\ e^{-j\Omega(\tau_M-\tau_0)} & \dots & \dots & 1 \end{bmatrix}, \tag{B.10}$$

and each correlation term in this matrix has the form,

$$\begin{aligned} \mathbf{R}_k(m,l) &= \sigma_z^2\, e^{-j\Omega(\tau_m-\tau_l)} \\ &= \sigma_z^2\, e^{-j\Omega\tau'_{m,l}}, \end{aligned} \tag{B.11}$$

where $\tau'_{m,l} = \tau_m - \tau_l$, is the relative integer sample delay between element pair (m,l). The phase of each covariance entry, measured across all frequency bins is,

$$\begin{aligned} \phi_{m,l}(k) = \mathrm{Arg}\left\{\mathbf{R}_k(m,l)\right\} &= -\Omega\tau'_{m,l} \quad 0 \le k \le K_c - 1, \\ &= -\frac{2\pi f_k}{f_s}\tau'_{m,l}. \end{aligned} \tag{B.12}$$

Inspection of Equation (B.12) reveals that the different sample delays in (B.7) result in linear phase across frequency with a slope of $-\frac{2\pi}{f_s}\tau'_{m,l}$. Phase which varies linearly across frequency is often referred to as a "phase ramp." When the sample offset applied to each data sequence is the same, $\tau'_{m,l} = 0$, and the slope of the ramp is also zero. Word lock is achieved when there is no variation in measured phase across frequency relative to a single element for the entire array. The goal then is to estimate the slope of the phase ramp and compensate by inserting added delays (i.e. shifting full 16-bit samples) that drive the slope to zero.

The true covariance and phase are not known, therefore $\mathbf{R}_k$ and $\phi_{m,l}(k)$ can only be observed by computing sample estimates of $\widehat{\mathbf{R}}_k$ and $\widehat{\phi}_{m,l}(k)$ (where $\widehat{\ }$ indicates an estimated quantity). The estimated phase is modeled as,

$$\mathrm{Arg}\left\{\widehat{\mathbf{R}}_k(m,l)\right\} = \widehat{\phi}_{m,l}(k) = a_{m,l} f_k + b_{m,l}, \tag{B.13}$$

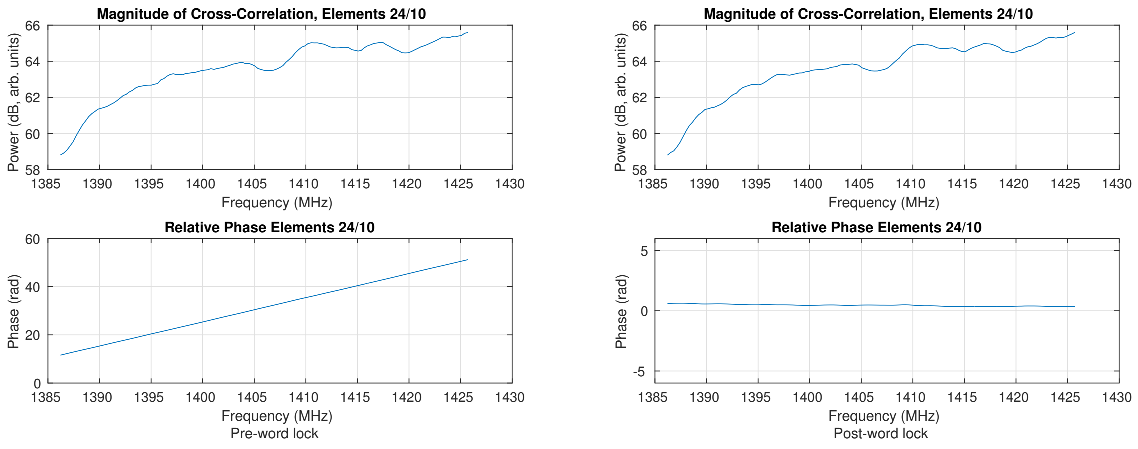


Figure B.3: Example word lock solution. Before word lock (left) a phase ramp is seen in the measured phase of the correlation between reference element 24 and element 10. After word lock (right) there is no longer a ramp by applying the necessary sample delay.

where a is the slope of the phase ramp and b is the phase-axis intercept, which allows for any constant vertical phase offset that may be introduced extraneously by the system. Phase samples across frequency channels are grouped together to form the linear system of equations,

$$\widehat{\Phi}_{m,l} = \begin{bmatrix} \widehat{\phi}_{m,l}(0) \\ \widehat{\phi}_{m,l}(1) \\ \vdots \\ \widehat{\phi}_{m,l}(K_c) \end{bmatrix} = \begin{bmatrix} f_0 & 1 \\ f_1 & 1 \\ \vdots & \vdots \\ f_{K_c} & 1 \end{bmatrix} \begin{bmatrix} a_{m,l} \\ b_{m,l} \end{bmatrix} = \mathbf{FA}, \tag{B.14}$$

with solution,

$$\mathbf{A} = \mathbf{F}^{-1}\widehat{\Phi}_{m,l}. \tag{B.15}$$

The phase ramp slope, $a_{m,l}$ is related to the respective time delay as

$$\begin{aligned} a_{m,l} &= \frac{2\pi}{f_s}\tau'_{m,l} \\ \tau'_{m,l} &= a_{m,l}\frac{f_s}{2\pi}. \end{aligned} \tag{B.16}$$

Having chosen reference element m, and then determining $\tau'_{m,l}$ for all l, this indicates the number of full 16-bit samples the ROACH must delay in the lth sequence to achieve word lock.

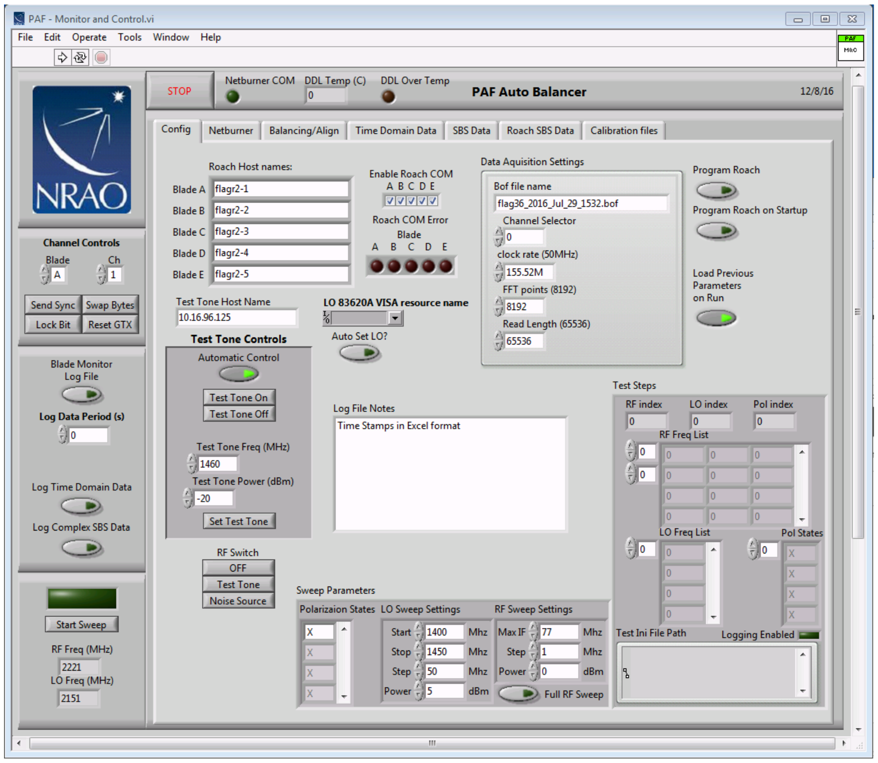
Figure B.3 shows the results of a word lock solution for a real-data experiment using the FLAG back end. Only causal delays can be implemented, so the word lock algorithm first does a search for the element with the most negative delay and designates that element as the reference. In this experiment element 24 was determined to be the reference. Phase is then measured in the correlation for each element relative to the reference element and it is shown in Figure B.3 that there is a significant phase ramp relative to element 10. After solving for the slope, the necessary sample offset needed to compensate for the ramp is calculated and applied. A measure of the phase after applying the sample offset shows that the slope has almost gone to zero, indicating we have achieved word lock. We found that best estimation of sample offsets happened when measuring phase near, but several frequency bins away from the roll off of the 150 MHz passband edges, and away from the passband center frequency (the LO frequency).

B.5 Implementation in FLAG

Much of the work to accommodate bit and byte lock in the ROACH has been done by NRAO at their Central Development Lab (CDL). They provided a GUI named "PAF Monitor and Control" which interfaces with both the blades at the front-end, and the ROACHs at the back end. Among its many features, the GUI provides capability to access the RF switch on the telescope and allow selection of the injection of a single tone or noise source into all the elements. There is also real-time feedback, plotting histograms and spectra, for the data arriving at the ROACH. This aids the operator to manually perform bit/byte lock. Figure B.4 shows a screen capture of the PAF Monitor and Control GUI.

PAF Monitor and Control provides a great interface to perform bit/byte lock manually, and up to this point the bit/byte lock done for FLAG commissioning has all been done manually. However, to be an instrument astronomers can use, the system needs to move to an automated implementation of these procedures. We were aware of the need for bit/byte lock before the May 2017 commissioning. However, it was during testing of that commissioning experiment we discovered word lock was an issue and that an algorithm to determine the number of sample offsets needed to be developed.

The back end ROACH configuration tool, Roach Doctor, has methods implemented to bit and byte lock, but they have not yet been fully tested. They were partially tested during the May



PAF Auto Balancer 12/8/16

DDL Temp (C): 0

Blade: A Ch: 1

Log Data Period (s): 0

RF Freq (MHz): 2221

LO Freq (MHz): 2151

Roach Host names:
Blade A: flagr2-1
Blade B: flagr2-2
Blade C: flagr2-3
Blade D: flagr2-4
Blade E: flagr2-5

Enable Roach COM A B C D E: ✓ ✓ ✓ ✓ ✓

Roach COM Error Blade A B C D E: ● ● ● ● ●

Test Tone Host Name: 10.16.96.125

LO 83620A VISA resource name

Test Tone Freq (MHz): 1460

Test Tone Power (dBm): -20

Data Aquisition Settings
Bof file name: flag36_2016_Jul_29_1532.bof
Channel Selector: 0
clock rate (50MHz): 155.52M
FFT points (8192): 8192
Read Length (65536): 65536

Log File Notes: Time Stamps in Excel format

Test Steps
RF index: 0 LO index: 0 Pol index: 0

RF Freq List: 0 0
0	0	0
0	0	0
0	0	0
0	0	0

LO Freq List: 0 — 0, 0, 0, 0

Pol States: 0 — X, X, X, X

Test Ini File Path

Sweep Parameters
Polarizaion States: X, X, X, X

LO Sweep Settings
Start 1400 Mhz
Stop 1450 Mhz
Step 50 Mhz
Power 5 dBm

RF Sweep Settings
Max IF 77 Mhz
Step 1 Mhz
Power 0 dBm

Figure B.4: PAF Monitor and Control GUI

commissioning, but due to time constraints in the outdoor test facility (OTF) and on telescope, were not fully tested. Updates and modifications to BFBE and the Roach Doctor tool were made in the interim before the August 2017 commissioning to accommodate word lock. However, this also has not yet been fully tested. More verification of all these capabilities need to be conducted in the OTF.

B.6 Summary

To summarize the foregoing discussion of this appendix: The stream of unformatted data from the blades requires that the back end ROACH boards determine proper bit, byte, and 16-bit word boundaries. To first achieve bit lock, a noise source generates white noise across all elements in the array. Symbols are slipped until the MSB and second most-significant bit for a sequence of bytes are almost always uncorrelated, or until the correct PDF for a histogram of incoming bytes is achieved. Next is byte lock, where the noise source is interchanged for a single tone seen by all elements. The real and imaginary bytes of the data are detected by rejecting images of the tone in channelized data. Then to perform word lock the injected tone signal is once again a noise source. The full back end system acquires data and saves covariance matrices to file. These files are read and processed to determine sample offsets relative to a single element. The back end updates the delay block in the ROACH with the correct sample delay.