Accelerator Design And Hardware Implementation For Distributed Coherent Mesh Beamformer

by

Yang Li

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2024 by the
Graduate Supervisory Committee:

Daniel Bliss, Chair
Chaitali Chakrabart
Ahmed Alkhateeb
Antonia Papandreou

ARIZONA STATE UNIVERSITY

May 2024

ABSTRACT

This dissertation summarizes achievements and ongoing designs of Field-Programmable Gate Array (FPGA) accelerators for Distributed Coherent Mesh Beamforming (DCMB). The distributed coherent mesh beamforming program aims to create a distributed beam network. The radios that make up this network must be small in size, weight, power, and cost while overcoming long transmission distances and interference. Due to the limitations, researchers developed a solid communication link using high speed to increase signal strength and reduce interference significantly. There were two slots to calculate the beamformer for the target platforms. One route is purely FPGA-based. Another option is a hybrid approach using the FPGA for initial calculations and the rest on the Central Processing Unit (CPU). People can reduce overall latency significantly when performing FPGA calculations. DCMB has become a technology for improving wireless communication systems, providing adaptability and efficiency in dynamic environments. This dissertation presents an in-depth study of DCMB with specific innovations in accelerator design and overall controller architecture. I investigate the design and implementation of dedicated accelerators adapted for DCMB tasks, including Finite Impulse Response (FIR) filtering, matrix multiplication, QR decomposition, and compensation on FPGA platforms. I optimized these accelerators for real-time processing and better performance on DCMB systems. Compared to soft-core processors, my research shows that hardware accelerators provide significantly faster processing speeds, enabling fast execution and reduced latency in communication systems. In addition, I discuss the design and integration of a general controller that optimizes the operation of accelerators and coordinates the beamforming process between distributed nodes. Through experiments with analytical and simulation tools, my study highlights the superiority of hardware accelerators over soft-core processors for high-speed calculation tasks in DCMB systems.

ACKNOWLEDGMENTS

I thank my adviser, Dr. Daniel Bliss, for his constant support throughout my study and expert advice through this project. He was very patient and positive for every step in the path to the success of our research. I am thankful to Dr. Daniel Bliss, Dr. Chaitali Chakrabarti, Dr. Antonia Papandreou-Suppapplola and Dr. Ahmed Alkhateeb of Electrical Engineering for teaching important courses that built a strong foundation for conducting my research. I am thankful to all my colleagues and co-workers at WISCA center for their constant help and support in shaping this thesis. I learned a lot from those long technical discussions in the lab on several topics. I thank Owen Ma, Jacob Holtom, HanguangYu, Saquib Siddiqui, Narayanan Murugan, and Anish Nallamur for providing valuable materials for my report. I am grateful to my parents for always supporting me in every endeavor.

TABLE OF CONTENTS

Page

LIST OF TABLES v

LIST OF FIGURES vi

1 INTRODUCTION 1

1.1 Previous Work 6

1.2 Work and Contributions 11

2 BACKGROUND 13

2.1 Beamforming 14

2.1.1 Traditional Antenna Systems 14

2.1.2 Principles of Beamforming 15

2.1.3 Types of Beamforming 15

2.1.4 Applications of Beamforming 16

2.1.5 Minimum Mean Square Error (MMSE) Method 17

2.2 Field-Programmable Gate Arrays (FPGAs) 19

2.2.1 Xilinx FPGAs 24

3 DISTRIBUTED COHERENT MESH BEAMFORMING SYSTEM 30

3.1 System Overview 30

3.2 Beamforming Methods 32

3.3 Narrowband MMSE Beamformer 34

3.4 Wideband MMSE Beamformer 38

4 ALGORITHM DESIGN AND HARDWARE IMPLEMENTATION CHALLENGES 42

4.1 Synchronization Methods 42

4.2 Data Exchange and Limited Transmitting Rates 47

4.3 Modified Wideband MMSE Beamformer 51

CHAPTER Page

4.3.1 Cross-Correlation 53

4.3.2 Covariance Matrices 56

4.3.3 Channel Estimation 61

4.3.4 Building the beamforming filters 63

4.4 Data Exchange and Accelerator Scheduler 63

5 IMPLEMENTATION ONTO HARDWARE 66

5.1 Overview 66

5.2 FPGA Implementation 66

5.2.1 Overview 67

5.2.2 Convolution and Cross-Correlation 72

5.2.3 Downsampling 76

5.2.4 Matrix-Matrix Multiplication 78

5.2.5 Average and Taper 85

5.2.6 Diagonal Loading 85

5.2.7 QR Decomposition with Back-Substitution 87

5.2.8 Normalization Strategy 96

5.3 Hybrid FPGA-C++ Implementation 97

5.3.1 SNR Gain Beamformer Implementation 98

5.3.2 Modified MMSE Beamformer Implementation 99

6 SUMMARY AND FUTURE WORK 101

REFERENCES 103

APPENDIX

A MATLAB SIMULATIONS 106

LIST OF TABLES

Table Page

5.1 Variables and Description 69

5.2 Computation Block Mapping to Hardware Units. 71

5.3 Internal Precision of Each Block in Figure 5.1. 71

5.4 Execution Time for Computations. 72

5.5 Utilization Report for Entire HDL Design. 73

5.6 Utilization Report for Convolution and Cross-Correlation Blocks. 75

5.7 Utilization Report for Filtering and Downsampling Block. 79

5.8 State of the Shift Registers after Every Cycle. 83

5.9 Resource Utilization of the Matrix-matrix Multiplication Block. 84

5.10 Utilization Report for the QR Decomposition Plus Back-Substitution Block 98

LIST OF FIGURES

Figure Page

1.1 Trade-off Curves in Hardware Designs.[1] 4

1.2 Signal From Transmitter to Receiver. 6

1.3 Satellite Communication Application for Distributed Coherent Mesh Beamforming Technology.[2] 8

1.4 Ground Base Station Mobile Devices Communication Applications. 9

1.5 Zynq UltraScale+ RFSoC ZCU208 Evaluation Kit 12

2.1 Beamforming Example. 15

2.2 3D Beamforming Application Scenario [3]. 16

3.1 Overview of a Distributed Communications Topology That Includes Two Meshes and Constant Interference. 31

3.2 Single Network Topology: Send a Distributed Network Node to the Final Receiving Node in the Presence of an Interrupter. 33

4.1 Sources of Error. 44

4.2 Frequency Error vs. Update Rate for a two-element antenna array[4] . . 45

4.3 Block Diagram of Adapted Distributed Mmse Computation. 52

5.1 Block Diagram of the Beamformer Computation. 68

5.2 The Execution Timeline of the Blocks in Figure 5.1 Devoid of a Time Scale. 70

5.3 Block Diagram of the Digital Beamformer Hdl Implementation. 70

5.4 Block Diagram of Cross-correlation and Convolution Units. 74

5.5 4×1 Systolic Array Used in the Convolution and Cross-correlation Blocks. 74

5.6 Cross-correlation Output Comparison Between Matlab and FPGA. . . . 75

Figure Page

5.7 Convolution Output Comparison Between Matlab and Fpga for One Channel B and Downsampled Training Observation Vector. 76

5.8 Block Diagram of Filter and Decimate Block. 78

5.9 Data Flow of Downsampling Block. 79

5.10 Comparison of Downsampled Training Observations Resulting from the FPGA Simulation and Matlab. 80

5.11 Matrix Multiplication Block diagram. 81

5.12 4×4 Systolic Array Matrix Multiplier. 82

5.13 Matrix Multiplication Output Comparison Between Matlab and FPGA. 84

5.14 Block Diagram of the Average-Taper Block. 86

5.15 Data Flow of the Average Taper Block. 86

5.16 Block Diagram of the Scalable QR Accelerator [5]. 89

5.17 Data Flow in QR Using Systolic Array Architecture [5]............... 90

5.18 Boundary Cell Flow Diagram [5]. 91

5.19 Internal Cell Flow Diagram [5]. 91

5.20 Data Flow in Back-substitution [5]. 92

5.21 Block Diagram of Back-substitution Accelerator [5]. 93

5.22 Original Design of Internal Cells in QRD Process [5] 94

5.23 parallel Design of Internal Cells in the QRD Process. 95

5.24 Original Back-substitution Process Data Flow [5]. 95

5.25 Modified Back-substitution Process Data Flow. 96

5.26 Comparison of Beamformers Resulting from Computations Performed in Fpga Versus Matlab Given the Same Inputs. 97

Figure Page

5.27 QRD Timing Report from Vivado. 97

A.1 Example Gold and Blue Node Placement Where the Mosaic Is Arranged as a Uniform Linear Array(Previous tests report from Owen and Jacob). 112

Chapter 1

INTRODUCTION

Distributed Coherent Mesh Beamforming (DCMB) represents an advanced approach to multi-antenna wireless communication networks that offers the opportunity to improve performance and reliability significantly. The improvement is essential given the ineffectiveness of traditional communications networks in meeting the demands of today's applications. However, there are limitations in the adoption of DCMB by several critical challenges that need to be addressed, such as, for example, interference suppression, limited spectrum efficiency, and lower signal quality [6]. In addition, successful implementation of DCMB requires overcoming obstacles related to hardware design, algorithm optimization, and scaling in various operating environments. Therefore, an important goal is to develop efficient hardware and algorithm solutions to unlock the full potential of DCMB and provide more efficient wireless networks that meet the advanced needs of modern applications [7].

Beamforming empowers a multi-antenna system to take advantage of spatial degrees of freedom at the receiver. It can increase signal-to-noise ratio (SNR) and resistance to obstructions. Typically, designers built beamformers with multiple antennas with wired connections at the receiver; thus, transitioning to distributed coherent beamforming requires synchronization between the dispersed components. Other distributed transmit beamforming approaches need to consider the strength of impedances within the channel more straightforwardly. When building the beamformer, there will be some interference for the received signal. The interference can be generated by the channel effects, communication between the distributed elements, etc. So that SNR improvement and interference cancellation functions are both applied. Recent advances in distributed positioning and timing enable synchronization between distributed radio receivers[8]. Further advances in low-cost antenna design, high-performance domain-specific processor design, and efficient synchronization technology have also helped create a new class of distributed coherent RF systems previously considered too complex or expensive. Distributed beamforming networks between these systems enable huge beamforming apertures using randomly and irregularly spaced elements at the expense of significant synchronization and processing requirements. While this approach significantly overcomes the inherent limitations of traditional high-speed arrays, it also introduces new challenges and disadvantages. Previous research suggested practical distributed array processing systems are possible but difficult to build. A practical real-time distributed coherent system must be inexpensive and have independent RF processing circuits. Researchers accomplished independent elements without coherent noise at many stages of the processing chain but not the whole design. There are also limitations to the independent elements. For example, each system has differences in frequency and time perception. Further complicating matters is that these differences constantly change as clocks tick and systems move. The main task of real-time distributed coherent systems is to synchronize all the quantities sufficiently to realize coherence between nodes and perform coherent processing in real time. It requires timing, frequency, and phase alignment. No system naturally wants to operate so synchronously because realizable systems' natural chaos and noise work against that goal. Additionally, the components and models usually chosen prefer to avoid implementing such a system. It is also computationally and spectrally resource-intensive to achieve coherence.

In this system, the high update rate and operation frequency are essential for achievement, which refers to the high SNR gain at the receiver and successful interference mitigation. The algorithm team did tests with different sets of data and different setup parameters to identify parameter sets that provide the best system performance. The algorithm team helped me with the simulations. By providing me with the setup parameters I used for the hardware implementation. They also provided test data for verification and all the functions that I need in the hardware implementations. The faster a system operates, the better the SNR at the receiver[9]. The updated rate is related to the frequency offset and phase offset[9]. The algorithm team tested system performance for update rates of 100 ms, 50 ms, 25 ms, and 10 ms. The 10 ms test data has the best SNR performance, but the 10 ms updated rate is impossible to get because 10 ms is too short for all the computations to finish. As a result, the algorithm team and I decided to go with an update rate of 25 ms instead. I observed that the 10 ms rate is feasible but requires significantly more initialization resources than required in the design. The system also performs well enough at the selected update rate for the high SNR and interference mitigation goals. For the hardware design, I focused on baseband signal processing which means there are other parts on the same device for the whole processing chain, namely the transmitter and data transition processes. With the limited resources available for the DCMB design, I targeted a 25 ms update rate. As seen in Figure 1.1, there is always a trade-off between utilization and speed. In this scenario, the best way is to implement a folded architecture for all the accelerators and make it configurable. Folded architecture design can limit the resources used in the hardware but make the latency longer. It can help to save several times on the resources used on hardware. The configurable design can make the accelerator more useful for different functions. For example, I can use the same accelerator for FIR filter and convolution functions, saving half of the resources.

In this dissertation, I accomplished coherence in real-time with a relatively simple execution of hardware design. I display one set of procedures and approaches that

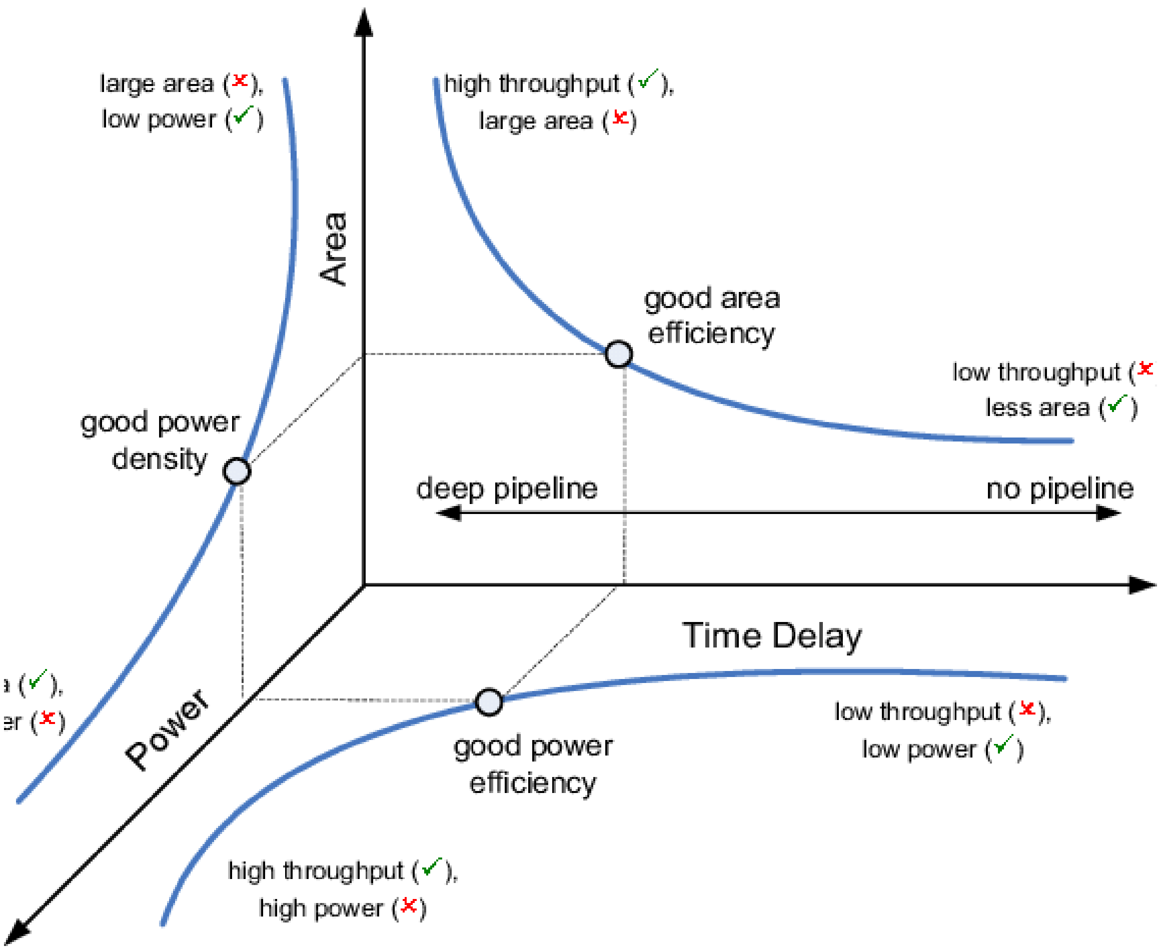


Figure 1.1: Trade-off Curves in Hardware Designs.[1]

address issues I met in the design stage. The accuracy required of the synchronization is a few orders of magnitude superior to what currently utilized communications frameworks work with. There are three central synchronization challenges. They are: achieving precise timing alignment across distributed nodes, maintaining coherence in signal reception and transmission, and minimizing synchronization overhead while ensuring accuracy and reliability[9]. Time synchronization has been scrutinized and appeared attainable. While conveyed time synchronization offers sufficient precision for many applications, researchers often overlook its potential due to certain limitations or challenges[10]. The algorithm team has considered recurrence synchronization as well. Adequate recurrence exactness is challenging and not widely done on hardware. Most existing frameworks don't require recurrence alignment within a little division of a hertz. In this scenario, conventional communication frameworks typically only measure the alignment with precision in the range of tens or even hundreds of hertz. And it does not have enough precision for our applications. Additionally, the system requires phase synchronization for coherent handling. In expansion, the proposed framework requires genuine time coherence between the disseminated frameworks. Not as it was in the framework to have the adequate exact time, stage, and recurrence arrangement; it must do so in real-time. The approach leverages transmit beamforming, so coherence must be realized over the signal processing procedure, not after.

I present a hardware design for a distributed mesh beamforming algorithm that uses many folded architectured and configurable accelerators to build three higher-performance data processing chains with increased speed and less latency. As part of a team, I developed and implemented a novel set of techniques that fully exploit the hardware implementation to maximize link range, data rate, and robustness. By performing transmit beamforming from the mesh, I can simultaneously receive an amplified signal observed by the intended receiver and eliminate the interference observed at the mesh; receiving nulling is performed through transmit beamforming in hardware.

Now, I briefly discuss the novel DCMB hardware implementation technique. A parallel process design is applied to extend and improve the performance of a typical wireless communications system. First, the signal was sent from the transmitter node as shown in Figure 1.2. The potentially shifted and compressed original radio signal in frequency has training signals incorporated. This signal is then distributed among the Mesh A nodes and is potentially transformed again through the air. An optimized set of distributed pre-distortion filter hardware implementations (one for

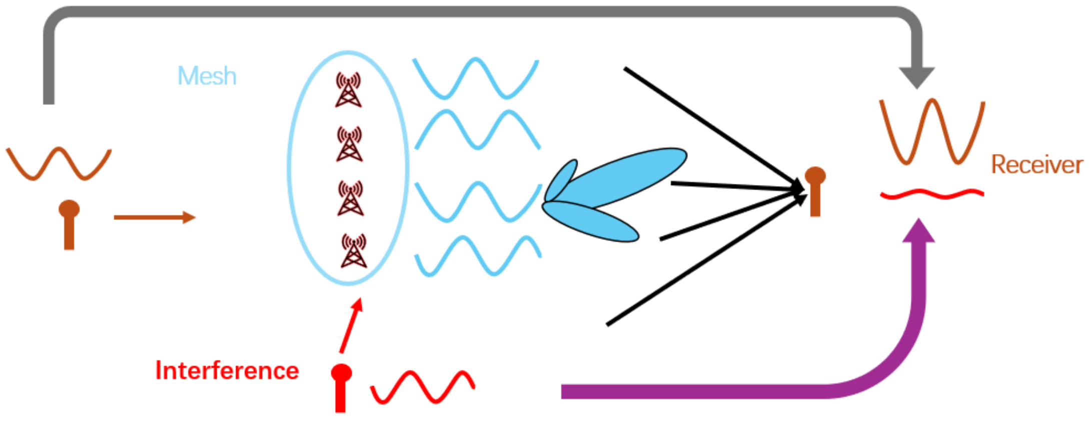


Figure 1.2: Signal From Transmitter to Receiver.

each mesh node) is applied to each signal, operating as a wideband beamformer. This beamformer maximizes the SINR at the receiving Mesh B [11].

1.1 Previous Work

In this section, I'll review earlier studies and attempts in distributed mesh beamforming and hardware implementations. While many ways exist to use many elements in an RF system, I will concentrate on current studies in the coherence and dispersed beamforming sector.

To boost performance, distributed coherent systems integrate distributed pieces through synchronization or other reconstruction methods. Using distributed components in a manner that is similar to a traditional antenna array is known as distributed mesh beamforming. Adaptive weightings are applied to every dispersed element to achieve this.

Previous work on distributed coherent network beamforming (DCMB) has mainly focused on exploring this innovative wireless communication system design's theoretical foundations, algorithmic developments, and experimental validations. Researchers have studied various aspects of DCMB: flash modeling techniques, distributed processing architectures, hardware implementations, and performance evaluations. Developing and optimizing flash modeling algorithms is a critical research area. These algorithms aim to maximize signal quality, reduce interference, and improve spectrum efficiency. It utilizes the spatial diversity provided by distributed antennas[6]. Additionally, researchers have tried to improve the robustness of DCMB systems to channel variations, Doppler effects, and environmental conditions[9]. In addition, researchers have proposed other new types of distributed processing architectures and applications, as shown in Figure 1.3 and Figure 1.4. All these new architectures can efficiently implement DCMB algorithms on hardware platforms. These architectures use parallel computing, distributed signal processing techniques, and optimized communication protocols to achieve real-time performance and scalability[7].

As shown in the previous paragraph, researchers utilize distributed coherence functions in multiple distinct regimes. For instance, post-processing is often used in radio astronomy systems with dispersed receivers to restore coherence and lessen the need for receiver synchronization. However, complete coherence phase and cycle synchronization are necessary for timing and positioning applications[12]. The distributed beamforming system is in the middle; it needs to be in the correct phase, but the overall synchronization of the system can withstand a few cycle slippages. Thanks to the approach in [12], phase-accurate distributed coherence can be achieved simultaneously with communications. Phase-accurate distributed coherence serves as the foundation for the provided distributed beamforming algorithm.

In [13], the viability of this class of systems is demonstrated, including instances with increased phase variance. There is a known trade-off between the highest possible benefit and inadequate synchronization. Researchers addressed subsequent problems, and advancements in laboratory demonstrations of these systems are presented in subsequent work [14].

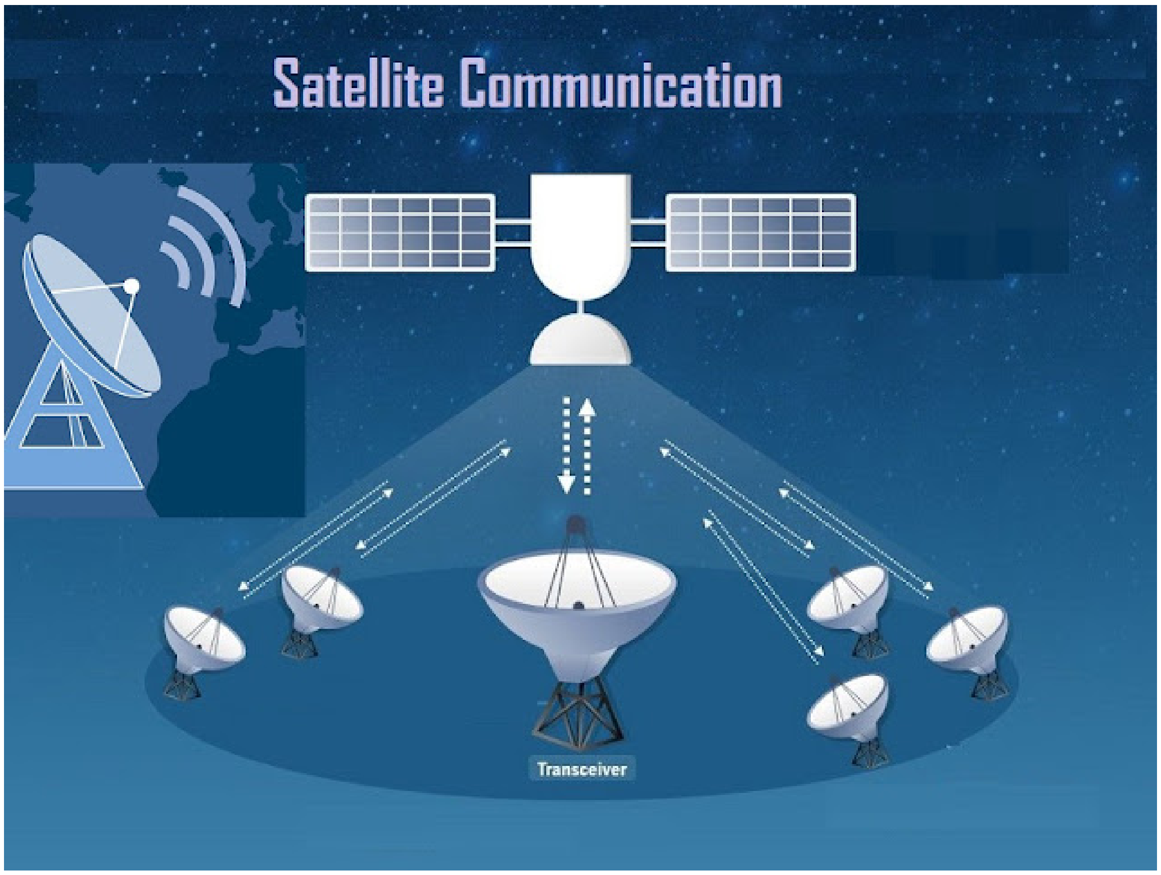


Figure 1.3: Satellite Communication Application for Distributed Coherent Mesh Beamforming Technology.[2]

Combining shared hard decisions and coding decisions yields the necessary SNR enhancement in another method described in [15]. Nonetheless, the designer did not consider robustness to interference in this technique. With a set of small size, weight, power, and cost radios conducting transmit beamforming to significantly extend the range and robust communications to a far receiver, the designer established the viability of the final transmit beamforming stage in [16]. For distributed transmit beamforming, the signal-to-interference-plus-noise ratio (SINR) loss related to carrying out the channel estimate concurrently with the data transmission is described in [17]. Combining shared hard decisions and coding decisions yields the necessary SNR enhancement in another method defined in [15].

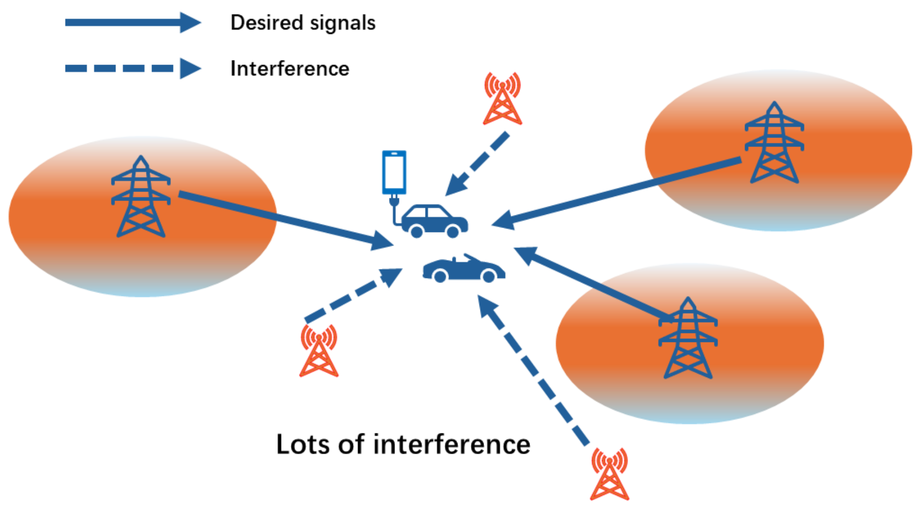


Figure 1.4: Ground Base Station Mobile Devices Communication Applications.

For distributed transmit beamforming, the signal-to-interference-plus-noise ratio (SINR) loss related to carrying out the channel estimate concurrently with the data transmission is described in [17]. The researcher described a technique that allows timing synchronization to go close to fundamental limitations in [18]. The enablement of distributed coherent beamforming techniques is also covered. The most recent developments in this field have seen the implementation of workable systems using commercially available hardware (COTS) that accomplish some of the objectives and show viability.

A basic version employing three separated nodes achieves frequency synchronization [19] by feeding a small amount of receiver feedback into an Extended Kalman Filter (EKF). Phase is adjusted using a single bit of feedback to achieve beamforming. With N being the number of mesh antenna components, this technique produced an SNR gain of about 1.5 dB away from the N^2 bound on SNR improvement. It did, however, take a while to converge. The researcher obtained similar outcomes in [20] and [21], with the main distinction being that they utilized a reference LO signal for synchronization. Then, they sent it over the air to the virtual array's nodes.

These SNR gain-focused methods are expanded onto mobile platforms once more in [22] and [23]. A controlled guide transceiver bends the beam to support phase adjustment feedback. They used feedback from the destination receiver to synchronize frequencies. They examined beamformer phase estimation bounds and distributions.

In hardware implementation studies, researchers have investigated the design and optimization of dedicated hardware accelerators and platforms for DCMB applications. These hardware platforms aim to balance performance, power consumption, and resource utilization to meet the stringent requirements of DCMB systems [5]. [5] explains the QRD algorithm in matrix inversion and is one of the most complex blocks in the design. Researchers have achieved experimental validations of DCMB using simulation-based methods such as Matlab, but people have yet to implement them in real life. These experiments demonstrate the performance, scalability, and reliability of DCMB systems in various operating scenarios and network configurations [9].In addition, there are other previous studies [24], [25], [26] on DCMB hardware that focused on optimizing performance, efficiency, and scalability. However, current hardware implementations often need help with challenges such as limited processing capabilities, high power consumption, and significant form factors. Recent advances have explored new architectures, dedicated accelerators, and integrated circuits tailored for DCMB applications to address these limitations. These innovations [27] aim to improve performance, reduce latency, and improve energy efficiency, enabling DCMB deployments in resource-constrained environments. Overall, previous work on DCMB has contributed significantly to understanding distributed information, beamforming techniques, efficient processing architectures development, and DCMB feasibility validation in practical environments. These efforts will contribute to the widespread adoption of DCMB in real-world applications.

1.2 Work and Contributions

My work focuses on developing hardware design methods for Distributed Coherent Network Formation (DCMB). It is a critical technology for today's communication networks. My research mainly focuses on creating unique hardware accelerators for beamforming in and without DCMB systems. These accelerators are optimized to perform essential functions such as QR decomposition, decompensating, and matrix-to-matrix multiplication with expected efficiency and accuracy. By offloading these computationally intensive tasks to dedicated hardware, my approach significantly improves the performance and scalability of DCMB systems. These improvements allow for real-time processing and better signal quality. In addition, my work goes beyond hardware accelerator optimization. I also designed and implemented a new architecture to control distributed node-to-node beamformers. This scheduler achieved parallel execution of multiple timelines. Each of them is responsible for specific tasks in the speed update process. The scheduler allows it to meet the timing requirements of the DCMB network. I show the FPGA board I used to implement the design in Figure 1.5. I expressed my contributions to this work in the following bullets:

- Designed three timelines to shorten the latency for the whole process.
- Improved folded architecture for the cross-correlation and matrix multiplication accelerator to limit the utilization of hardware design with fewer hardware resources.
- Implemented two LUTs in the QRD accelerator to increase the precision of output data.
- Completed timing closure for the whole design with required timing of 2.5ms.

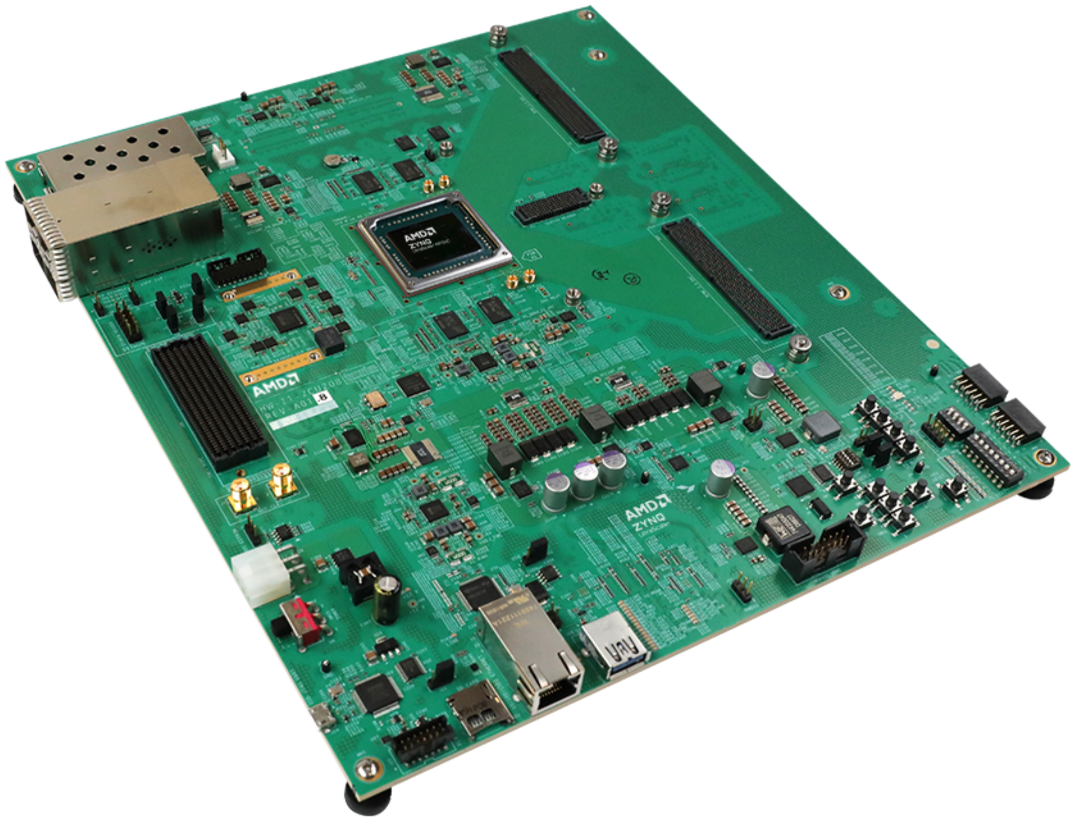


Figure 1.5: Zynq UltraScale+ RFSoC ZCU208 Evaluation Kit

- Developed two avenues for computing beamforming on the target platform; one is purely FPGA-based, and the other is a hybrid approach.

In addition, Dr. Daniel Bliss, Dr. Chaitali Chakrabart, Dr. Jacob Holtom, Dr. Owen Ma, Dr. Andrew Herschfelt, Dr. Hanguang Yu, Narayanan Murugan, Saquib Siddiqui, and Isabella Lenz were vital contributors to the distributed coherent beamforming research and demonstrations.

Chapter 2

BACKGROUND

Two key ideas in wireless communication systems are MIMO (Multiple Input, Multiple Output) and SISO (Single Input, Single Output). Each of them has a distinct history and set of properties. History of Single Input Single Output, or SISO, systems: Researchers utilize one transmission antenna and one reception in SISO systems. They are the conventional kind of wireless communication systems. Optimizing modulation methods, coding strategies, and signal processing algorithms to reach maximum spectral efficiency is the main focus of SISO systems. Regarding installation and signal processing, SISO systems are more accessible than MIMO systems. However, the capacity and robustness of SISO systems are constrained, particularly in fading and interference-filled situations. MIMO (multiple input, multiple output) systems' history: Designers utilized multiple antennas in MIMO systems for transmission and reception. It allows for the use of multiplexing gain and spatial diversity. MIMO technology has dramatically increased data speeds, spectrum efficiency, and dependability. The fundamental idea underlying MIMO is to use multiple antennas to generate numerous parallel communication channels between the transmitter and receiver. MIMO enhances the capacity and reliability of the system. MIMO systems may improve performance in data rates, coverage, and interference mitigation by utilizing various techniques such as beamforming, precoding, spatial diversity, and spatial multiplexing because MIMO technology may produce more significant data rates, better coverage, and increased speed. Researchers commonly use it in wireless communication protocols, including LTE, Wi-Fi, and 5G.

Researchers positioned several antenna systems to benefit from spatial diversity by utilizing various transmit and receive elements. A comparable single input single output (SISO) system cannot match spatial diversity created by various pathways connecting the transmitting and receiving systems. A system can become more resilient, handle higher data rates, or be less susceptible to interference by utilizing numerous antennas on both the broadcast and receiving ends [17]. Beamforming techniques, which treat the transmit antennas as an array, can combine desired signals coherently.

Several strategies and techniques exist to take advantage of various components and channel pathways. I will briefly address a few common ones in the following section, but I would like to address them quickly and immediately. Using space-time coding techniques offers a single way to use several antenna systems. A space-time trellis code divides a conventional trellis code over many antennas and periods to achieve diversity and coding gain. A data block is distributed among antennas and time slots using a space-time block coding, which only increases spatial diversity.

2.1 Beamforming

In wireless communication systems, beamforming is a signal-processing technique that focuses radio waves in a particular direction to improve the performance of the communication channel. The history of beamforming is as follows:

2.1.1 Traditional Antenna Systems

Signals are consistently broadcast and received in all directions in typical antenna systems. The communication system's range, data speeds, and general performance are all restricted by this omnidirectional transmission/reception pattern. It is particularly problematic in noisy and interference-filled situations.

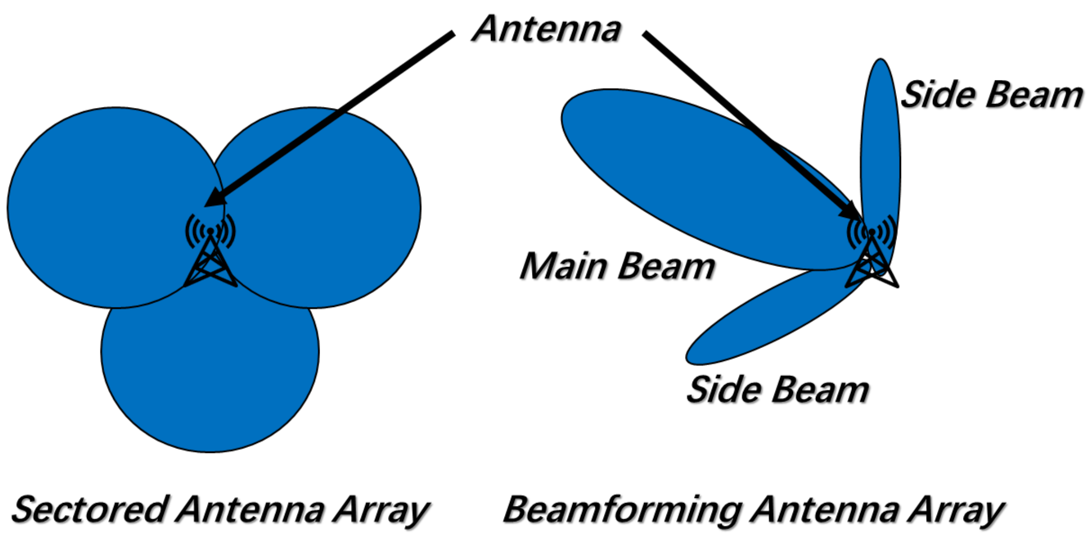


Figure 2.1: Beamforming Example.

2.1.2 Principles of Beamforming

Beamforming utilizes the idea of constructive and destructive interference. Beamforming maximizes the concentration of transmitted energy in one direction while reducing interference from other directions. Various antenna components produce a directed radiation pattern by varying the phase and amplitude of the signals sent and received[28]. Depending on the system's needs and available technology, people can perform beamforming using analog or digital signal processing techniques. In this dissertation, I will only discuss digital signal processing techniques.

2.1.3 Types of Beamforming

Digital beamforming as shown in Figure 2.1, is one of the primary categories of beamforming. Digital beamforming uses digital electronics to guide the beam in the appropriate direction. By processing the data digitally, digital beamforming offers greater flexibility and control over the beamforming procedure. Furthermore,

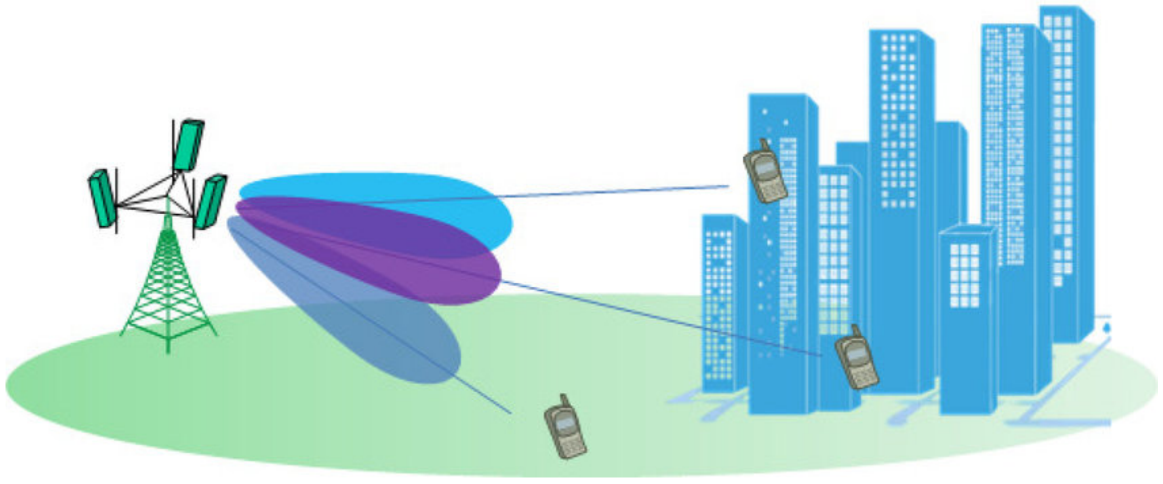


Figure 2.2: 3D Beamforming Application Scenario [3].

beamforming methods may include time domain-like phased array systems[29] and frequency domain-like frequency-domain beamforming applications[30].

2.1.4 Applications of Beamforming

Beamforming has extensive use in a range of wireless communication systems. For example, satellite communication, Wi-Fi, cellular networks as shown in Figure 2.2, and radar systems are all applications related to DCMB. It enhances system capacity and spectrum efficiency, expands coverage area, reduces interference, and improves signal reception quality. Beamforming is a crucial technique that enables enormous data speeds, better coverage, and an enhanced user experience in modern communication protocols, including LTE, 5G, and Wi-Fi 6/6E.

In conclusion, beamforming is an effective signal-processing method that allows signals to be transmitted and received in a specific direction in wireless communication systems. It improves wireless communication systems' efficiency, dependability, and performance.

The array factor represents the combined radiation pattern of several antenna elements in an array. I state them as follows:

$$AF(\theta) = \sum_{n=1}^{N} w_n e^{-j2\pi d \sin(\theta) \frac{n-1}{\lambda}} \tag{2.1}$$

Where $AF(\theta)$ is the array factor as a function of the angle of arrival (θ). w is the complex weight assigned to each antenna element. d is the spacing between adjacent antenna elements. λ is the wavelength of the transmitted signal.

The steering vector represents the beamforming array's directionality. I start it as follows:

$$\mathbf{a}(\theta) = [1, e^{-j2\pi d \sin(\theta)/\lambda}, \ldots, e^{-j2\pi d (N-1) \sin(\theta)/\lambda}]^T \tag{2.2}$$

where $\mathbf{a}(\theta)$ is the steering vector, θ is the angle of arrival, d is the spacing between adjacent antenna elements, N is the number of antenna elements, λ is the wavelength of the transmitted signal.

It is possible to compute the weights allocated to each antenna element using different beamforming techniques. The Maximum Signal-to-Interference-plus-Noise Ratio (Max-SINR) beamforming technique is one popular approach[31][9]. The weights for this method are provided by[9]:

$$\mathbf{w}_{\text{Max-SINR}} = \frac{\mathbf{R}^{-1}\mathbf{a}(\theta)}{\mathbf{a}^H(\theta)\mathbf{R}^{-1}\mathbf{a}(\theta)} \tag{2.3}$$

where $\mathbf{w}_{\text{Max-SINR}}$ is the vector of beamforming weights, $\mathbf{R}$ is the covariance matrix of the received signal.

2.1.5 Minimum Mean Square Error (MMSE) Method

In wireless communication systems, the Minimum Mean Square Error (MMSE) beamforming approach optimizes the transmit beamforming vectors to minimize the mean square error at the receiver while considering noise and channel conditions[32].

MMSE beamforming seeks to minimize the mean square error between the desired and received signals by accounting for noise and channel characteristics. It strikes a balance between reducing noise and interference and optimizing the strength of the received signal. MMSE beamforming is especially helpful when channel conditions are ambiguous, or channel state information (CSI) is imperfect.

Let w be the beamforming vector applied at the transmitter and h be the channel vector between the emitter and the receiver. The received signal at the receiver expressed as:

$$y = \mathbf{h}^H \mathbf{w} x + n \tag{2.4}$$

where the algorithm team denoted the received signal by y, the signal was sent x. The noise adds up to n. Then, the algorithm team shows the conjugate transpose by the symbol $()^H$.

The estimated transmitted signal x at the receiver is given by:

$$\hat{x} = \mathbf{w}^H \mathbf{h} y_{\text{eff}} \tag{2.5}$$

where: y_{eff} is the effective received signal after applying the beamforming at the transmitter.

Minimizing the mean square error (MSE) between the transmitted signal x and the estimated signal $\hat{x}$ is the aim of MMSE beamforming. The MSE is denoted as:

$$\text{MSE} = E\left[|x - \hat{x}|^2\right] \tag{2.6}$$

where E denotes the expectation.

The goal is to find the beamforming vector w that minimizes the MSE. To do so, to differentiate the MSE concerning the conjugate transpose of the beamforming vector x^* and set the derivative equal to zero. Setting the derivative equal to zero is because, at the minimum point, the derivative of the MSE concerning x^* should be zero. The MMSE beamforming vector $\mathbf{w}_{\text{MMSE}}$ can be derived by minimizing the MSE[28]:

$$\mathbf{w}_{\text{MMSE}} = \frac{\mathbf{R}_h^{-1}\mathbf{h}}{\mathbf{h}^H\mathbf{R}_h^{-1}\mathbf{h}} \tag{2.7}$$

where: $\mathbf{R}_h$ is the covariance matrix of the channel $\mathbf{h}$.

MMSE beamforming strikes a compromise between reducing noise and interference and optimizing the strength of the received signal. It functions effectively in situations when there is noise and faulty CSI. This formulation and solution provide the basis for comprehending and applying MMSE beamforming in wireless communication systems.

2.2 Field-Programmable Gate Arrays (FPGAs)

Integrated circuits known as Field-Programmable Gate Arrays (FPGAs) allow user configuration after manufacturing the device. Because of their great adaptability and customization, designers may use them in various sectors and for multiple purposes. A flexible routing network connects a variety of programmable logic blocks (PLBs) that make up FPGAs. PLBs comprise flip-flops and customizable logic elements (LEs), which may be programmed to perform certain logic operations. The routing network facilitates the development of intricate digital circuits by permitting flexible connections between PLBs. Designers can use High-level synthesis tools or Hardware Description Languages (HDLs) like Verilog or VHDL to program FPGAs. With HDLs, designers may specify the desired register-transfer functionality of the digital circuit. The configuration file, or bitstream, is created and loaded into the FPGA once the design has been synthesized and mapped onto it, setting it up to carry out the required functions. There are advantages for FPGA as follows:

- Flexibility: Fast prototyping and iteration of designs are made possible by the great flexibility and reconfigurability of FPGAs.
- Reconfigurability: Even after deployment, FPGAs are reprogrammable. With this functionality, designers can reconfigure hardware dynamically to accommodate upgrades or changing requirements without requiring new hardware.
- Rapid Development and Prototyping: designers utilized FPGAs extensively in developing and prototyping digital systems. Time-to-market is shortened for designers since they can test new algorithms, validate concepts, and iterate designs fast without waiting for specialized ASIC manufacturers.
- Customization: FPGAs make it possible to build hardware accelerators suited explicitly for a given application, boosting efficiency and performance. Parallelism: FPGAs are ideally suited for applications with high processing needs since they naturally allow parallelism.
- Parallelism: FPGAs inherently support parallelism, making them well-suited for applications with high computational demands.
- High Performance: FPGAs may attain high performance for particular activities by utilizing hardware acceleration and parallel processing. FPGA-based custom hardware accelerators can perform faster and more efficiently than software-based alternatives.
- Minimal Costs for Non-Recurring Engineering (NRE): FPGAs offer lower NRE costs than application-specific integrated circuits (ASICs) since they don't need costly masks and manufacturing processes. Because of this, FPGAs are more affordable for quick prototyping or low- to medium-volume production.

- Ease of Iteration and Debugging: FPGAs make it simple for designers to modify the hardware design, promoting iterative design processes. Additionally, FPGAs come with built-in tools for testing and debugging in real-time, which help find and fix problems quickly.

- High-degree Synthesis (HLS): HLS tools allow designers to use software-like structures to describe hardware operations at higher abstraction. HLS lowers the entry barrier for FPGA development by enabling software engineers to build and optimize hardware circuits using their programming expertise.

- Versatility: From digital signal processing, image and video processing, networking, and encryption to embedded systems, FPGAs serve various applications. Due to their adaptability, designers can use FPGAs in multiple sectors and applications.

- Low-Risk Investment: FPGAs provide a low-risk investment for creating new products because of their flexibility and reconfigurability. Designers can test several architectures and configurations before final hardware design.

- Customization and Differentiation: FPGAs allow businesses to set themselves apart from the competition with their goods by including unique hardware accelerators or customized processing units designed for specific applications. Customization may result in distinctive features and a competitive edge in the marketplace.

- Product Lifecycle Management and Length: FPGAs have longer product lifecycles than application-specific integrated circuits (ASICs). Longer lifecycles lower the risk of obsolescence by ensuring component availability for ongoing product maintenance and support.

- Time-to-Market: FPGAs can reduce time-to-market by enabling rapid prototyping and design validation without needing custom ASIC development.

FPGAs provide flexibility, performance, affordability, and ease of development compared to other options. The flexibility makes them a desirable choice for a variety of applications. People may find FPGA applications in many industries, including consumer electronics, data centers, automotive, aerospace, and telecommunications. Digital signal processing (DSP), image and video processing, machine learning inference, encryption and decryption, software-defined networking (SDN), and embedded systems are examples of everyday uses.

There are also challenges for FPGA as follows:

- Complexity of Design: Digital logic design and FPGA architectural knowledge are prerequisites for designing intricate circuits for FPGAs, which can be difficult. Timing restrictions, resource use, and route congestion become more challenging to manage as designs get more complex.

- Resource Restrictions: Logic cells, memory blocks, and input/output (I/O) pins are among the few resources available to FPGAs. To ensure the design works within the limitations of the FPGA chip, designers must carefully control resource use. Designers must compromise performance or functionality, which may occasionally happen due to resource constraints.

- Timing Closure: In FPGA design, achieving timing closureensuring all signals satisfy timing requirementscan be tricky. Designers carefully place and route high-speed designs to reduce signal propagation delays and fulfill timing limitations. Timing closure can take time and may require numerous iterations.

- Power Consumption: Using FPGAs to implement complicated designs can lead to excessive power consumption, mainly when those designs operate at high frequencies or with high logic. Applications that target battery-operated devices or those with tight power limits must control power usage.

- Verification and Validation: Ensuring the accuracy and operation of the implemented logic is a complicated procedure that goes into verifying and validating FPGA designs. While necessary, verification methods like hardware emulation, formal verification, and simulation can be laborious and demand a lot of processing power.

- Security Risks: FPGAs are vulnerable to security risks, including tampering, reverse engineering, and intellectual property theft (IP). Robust security mechanisms, such as encryption, authentication, and access control, must be implemented to guard FPGA designs from theft, piracy, and hardware assaults.

- Toolchain Complexity: FPGA development tools and toolchains can be challenging because of their complexity and high learning curve. The synthesis, place-and-route, and debugging tools offered by FPGA suppliers need designers to become adept in utilizing them; each tool may have peculiarities and restrictions.

- Vendor Lock-In: Selecting an FPGA vendor binds designers to that vendor's environment, which consists of their exclusive libraries, IP cores, and development tools. A significant redesign and modification of the design could be necessary to switch to a new supplier, raising questions about vendor lock-in and reducing flexibility.

- Testing and Debugging: It may be challenging to test and debug FPGA designs, particularly when they are complicated or include co-designing hardware and software. FPGA debugging tools and methods may not be as developed as software debugging, necessitating innovative problem-solving strategies.

- Cost: Although FPGAs are flexible and reconfigurable, in terms of unit cost, they might be more costly than software solutions or application-specific integrated circuits (ASICs), especially in high-volume production. Cost concerns may influence the decision between alternate solutions and FPGA-based solutions.

In conclusion, FPGAs provide a very adaptable and configurable framework for building digital circuits. It facilitates quick prototyping, personalization, and parallel processing for various business uses.

2.2.1 Xilinx FPGAs

In this dissertation, I describe the whole hardware implementation process. First of all, I am using FPGA to implement the algorithm. And then, I choose Xilinx FPGAs. There are a lot of choices in the market.

Xilinx is one of the top suppliers of FPGAs, and related software tools and IP cores are Xilinx, which is now a part of AMD. Xilinx provides various FPGA families, including the Virtex, Kintex, and Artix series, as well as specialist goods like RFSoCs and Zynq UltraScale+ MPSoCs. Numerous industries, including telecommunications, automotive, aerospace, data centers, industrial automation, and consumer electronics, employ their FPGAs. Intel has the FPGA part called Altera. With the acquisition of Altera, Intel has emerged as a significant participant in the FPGA industry. The companies designed the Stratix, Arria, and Cyclone series of Altera's FPGA families to meet the needs of distinct market groups and application areas. For example, networking, data center acceleration, automotive, and industrial automation are just a few of the many uses for Intel FPGAs. Currently a division of Microchip Technology, Microsemi provides a variety of FPGAs under the SmartFusion and PolarFire brands. Xilinx integrated ARM Cortex-M3 CPUs and FPGA fabric within SmartFusion FPGAs, which makes them appropriate for embedded applications. Applications requiring low power and security, such as industrial IoT, defense, and aerospace, are well-suited for PolarFire FPGAs. Semiconductor Lattice: Software tools, programmable mixed-signal devices, and low-power FPGAs are the areas of expertise for Lattice Semiconductor. Targeting applications including mobile devices, IoT, industrial automation, and consumer electronics, their FPGA families include the iCE40, ECP, and MachXO series. For low-volume applications, lattice FPGAs are renowned for their tiny form factor, low power consumption, and quick market time. QuickLogic Corporation specializes in embedded FPGA (eFPGA) IP solutions and ultra-low-power FPGAs for battery-powered and power-constrained applications. Their FPGA products include the ArcticPro and QuickLogic EOS S3 series, intended for edge AI, wearables, and Internet of Things applications. The semiconductor Achronix, High-performance FPGAs, and embedded FPGA (eFPGA) IP solutions for networking, data center acceleration, and automotive applications are Achronix Semiconductor's areas of expertise. The Speedster and Speedcore series are among their FPGA families; they include high-speed interfaces and low-latency processing, and designers can reconfigure fabric for customized acceleration.

These are only a handful of the businesses that manufacture FPGAs. Each has its own unique features, functions, and target customers. Designers have a wide range of FPGA families and suppliers to select from, depending on the particular requirements of an application.

The whole team chose Xilinx FPGAs among all these worldwide manufacturers because they have advantages for our design. Xilinx FPGAs are well-known for their state-of-the-art features, exceptional performance, and broad ecosystem support. This paragraph is a more thorough overview of Xilinx FPGAs, encompassing several facets of their programming, architecture, benefits, uses, and unique characteristics. With programmable logic blocks (PLBs), customizable logic elements (CLEs), block RAMs, DSP slices, and high-speed transceivers, Xilinx FPGAs have a very flexible design. Designers can configure Flip-flops, CLEs, and other components found in the PLBs to perform unique logic tasks. While DSP slices offer specialized hardware for executing digital signal processing algorithms, block RAMs provide on-chip memory resources for data processing and storage. Xilinx offers a complete set of software tools for FPGA development, such as the ISE Design set (legacy) and the Vivado Design Suite. Vivado Design Suite provides a cutting-edge, integrated development environment for FPGA design, synthesis, place-and-route, and verification. The development tools provided by Xilinx provide many design entry techniques, including schematic capture, high-level synthesis (HLS) with languages like C, C++, or SystemC, and Hardware Description Languages (HDLs) like Verilog and VHDL. Advanced features like dynamic power management, on-chip security, and partial reconfiguration are all included in Xilinx FPGAs. Partial reconfiguration enables runtime flexibility to change requirements by allowing designers to program selected areas of the FPGA dynamically while the remainder of the device stays functional. In power-sensitive applications, dynamic power management techniques such as voltage scaling, power gating, and clock gating maximize battery life and minimize power usage. Xilinx FPGAs also have security capabilities to protect critical designs and intellectual property, including bitstream encryption, device authentication, and tamper detection. High-level synthesis (HLS) tools from Xilinx, including Vivado HLS, let designers use C, C++, or SystemC to define hardware functionality. High-level specifications are automatically converted into RTL (Register Transfer Level) code using

HLS tools, speeding up the FPGA design process and allowing software professionals to use their programming expertise to design hardware. Many sectors use Xilinx FPGAs, including telecommunications, automotive, aerospace, data centers, industrial automation, and consumer electronics. Digital signal processing (DSP), image and video processing, embedded systems, machine learning inference, encryption and decryption, and software-defined networking (SDN) are common uses. With a large selection of IP cores, development boards, and software solutions, Xilinx maintains a robust ecosystem of partners, developers, and outside IP vendors. The ecosystem of Xilinx consists of hardware suppliers, companies that offer design services, training partners, and online forums that give FPGA designers a wealth of information and assistance.

Xilinx FPGAs are a popular option in this field since they provide several benefits for creating wireless communication systems. The following are some of the main benefits. Excellent Throughput and Performance: High-performance logic resources, DSP slices, and hardware blocks specifically designed for digital signal processing are all features of Xilinx FPGAs. These resources match the rigorous criteria of contemporary wireless standards like 5G by enabling the development of intricate wireless communication algorithms with high throughput and low latency. Reconfigurability and Flexibility: Because Xilinx FPGAs are versatile and programmable, designers may easily modify and enhance wireless communication protocols and algorithms in response to changing standards or demands. Rapid development and testing of novel wireless technologies are made possible by reconfigurability, shortening time-to-market, and promoting innovation. Combining Acceleration and Processing: With the integration of FPGA fabric with multi-core ARM processors, Xilinx FPGAslike the Zynq UltraScale+ MPSoCsoffer a heterogeneous processing platform for wireless communication systems. The seamless implementation of baseband processing, protocol stack execution, and application-specific acceleration inside a single device is made possible by integrating processor cores with FPGA fabric. Personalization and Distinction: Xilinx FPGAs enable the implementation of customized hardware accelerators, protocol offload engines, and efficient processing pipelines. Hence, facilitating the customization and differentiation of wireless communication systems is essential in the design process. Designers can develop custom hardware accelerators to improve system speed and efficiency for tasks like channel coding, modulation/demodulation, beamforming, or interference mitigation. Minimal Power Usage: Xilinx FPGAs provide low-power design methodologies and extensive power management capabilities to reduce power usage for wireless communication systems. Clock gating, power gating, and voltage scaling are power optimization features that assist in lowering power consumption without sacrificing performance, making them appropriate for battery-operated and energy-efficient wireless devices. Interfaces and Integrated Connectivity: A vast array of fast transceivers and interfaces. They support widely used wireless communication protocols, including Ethernet, PCIe, JESD204, CPRI, and RF, which Xilinx FPGAs offer. System integration is made more accessible by integrated connections and interfaces, making it easy to interface with external parts like RF frontends, antennas, sensors, and network infrastructure. Features of Security: To guard against unwanted access, tampering, or intellectual property theft, Xilinx FPGAs include security capabilities such as bitstream encryption, secure boot, and device authentication in their critical wireless communication designs. Wireless communication systems' integrity, confidentiality, and authenticity are guaranteed by security measures, especially in applications that call for secure data processing and transmission.

Xilinx FPGAs provide a robust and adaptable foundation for speeding various applications and developing unique digital logic. With its sophisticated features, allinclusive development tools, and broad ecosystem support, Xilinx keeps pushing the boundaries of innovation in the FPGA sector and opening up new possibilities for next-generation solutions in various fields.

In conclusion, Xilinx FPGAs provide high performance, flexibility, customization, low power consumption, integrated processing, and security features to meet the changing needs of contemporary wireless standards and applications. All these feathers make them a powerful and adaptable platform for designing wireless communication systems.

Chapter 3

DISTRIBUTED COHERENT MESH BEAMFORMING SYSTEM

The WISCA center and the algorithm team develop the algorithm and most computations. The most detailed information can be found in[33][9][4], and some of the algorithms are from the reports that Owen and Jacob summarized for the research. In this chapter, I will talk only about the algorithms and computations implemented on the hardware. A distributed coherent network relay network consists of relays called initiating transmitters (transmitter and receiver at each end of the relay) or network nodes. Initiating transceiver nodes are the radios through which users can join the network. The system transmits the signal of interest (the payload) through the transmitter. The outgoing transmitter may amplify this signal to facilitate network propagation. The outgoing transmitter forwards this modified signal to a distributed, locally coherent collection of relay nodes. Relay nodes distort and amplify the received signal. The relay nodes then work together to forward the pre-distorted and augmented signal to another network or the incoming node. Repetition of this process will happen if the target set of receivers is another network. Otherwise, the receiving node effectively removes any remaining enhancements to isolate the signal of interest and forwards it for decoding or use. The radios and algorithms that make up this network need to be small in size, weight, power, and cost but able to overcome long transmission distances and sources of interference.

3.1 System Overview

Distributed Coherent Mesh Beamforming (DCMB) aims to create a network of distributed beams, shown in Figure 3.1, which act as signal relays. The network consists of beams marked as gold nodes and blue nodes. A radio user can ask for the help of a Distributed Coherent Mesh Beamforming (DCMB) system to transmit the signal by directing the output of the source beam to the input of the golden node.

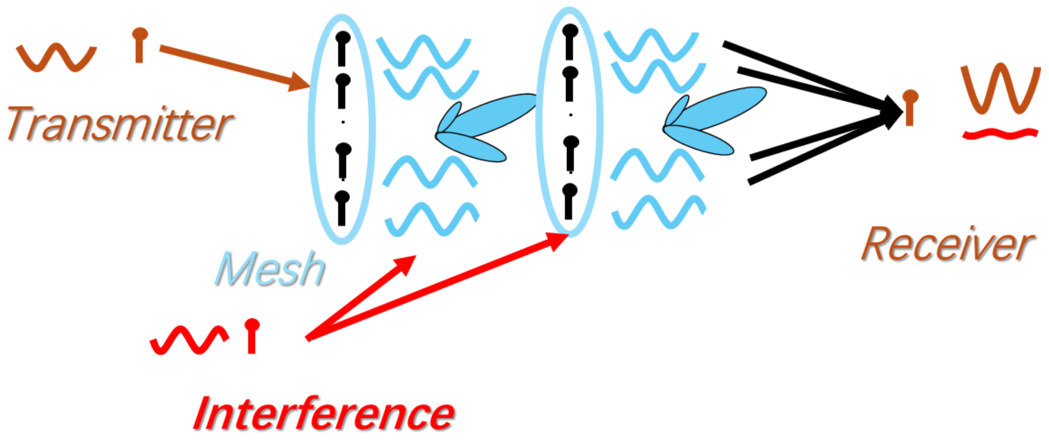


Figure 3.1: Overview of a Distributed Communications Topology That Includes Two Meshes and Constant Interference.

The system approaches each mosaic through an antenna array, with each mosaic tile representing one element, and the algorithm team builds the network to operate as a "bent-pipe." The pre-distortion filters applied to the relay signals, which coherently integrate the source waveform and reject incoming interference only when it reaches the receiving gold node antenna, make this technique distinctive. Put differently, transmit beamforming is used to achieve receiver beamforming. Therefore, the full benefits of the mosaic relay will not be available to any radio not co-located within the terminal gold node.

Frequency-division duplexing allows for bidirectional communications, with one antenna used for each direction. Both routes use the same processing. The waveform uses underlays to provide information feedback and training materials at every step. A transmitting gold node adds a low-power underlay and a center frequency translation as its only augmentations. Blue nodes filter out most of the underlay and apply the pre-distortion filter when processing incoming streaming data at the baseband to perform beamforming feathers. Subsequently, blue nodes add a new underlay and adjust the center frequency to prepare for the retransmission. Waveform agnosticism and a certain amount of computational simplicity are provided by this method, enabling flexible and attritable nodes. The computations used to create the beamformer are "generic," and no operations reveal protected information. A certain amount of infrastructure is required to enable such an activity. The algorithm team uses an inter-mosaic data exchange network to provide the blue nodes with the data they need to calculate the beamformer. Further, it depends on a background process of time-frequency synchronization to enable distributed beamforming by enabling the dispersed mosaic tiles to perform like a linked array.

3.2 Beamforming Methods

In this dissertation, I focus on the hardware design and implementation of a single mosaic scenario, depicted in Figure 3.2.

The filters essentially adjust the beam pattern of the array. The relay network differs from this traditional structure because every component of the receive array is not connected to the data fusion point. Mosaic tiles must broadcast over the air to a receiver to finish the signal-combining process. Through transmit beamforming, the system can obtain receiver beamforming. The following is an additional interpretation of this function. A gold n de sends a signal to another gold node equipped with a virtual antenna array over a channel that represents the two-stage propagation in analogy. At this point, the algorithm team can utilize standard processing techniques.

A particular beamformer building technique modifies the array beam pattern to

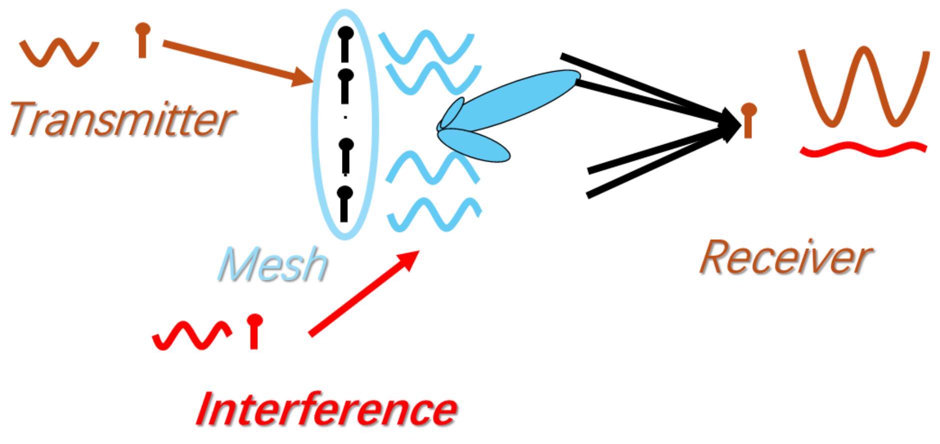


Figure 3.2: Single Network Topology: Send a Distributed Network Node to the Final Receiving Node in the Presence of an Interrupter.

align the main lobe concerning element placements in the best possible direction. Examples of such strategies include minimal power distortionless response (MPDR) and minimum variance distortionless response (MVDR). However, precise node location information is needed to attain the best performance. Introducing movable, unconnected parts exacerbates these approaches' sensitivity to positional precision. Their approach does not need the specific position of the transmitters and receivers. The adaptive wideband beamformer uses the minimal mean square error (MMSE) technique. Instead, they use established training waveforms and track the alterations they experience as they travel via communication channels. The observations provide information about the exact relative locations of each receive and transmit element, which is utilized implicitly throughout construction.

To mitigate the interference, researchers used the technique of a spatiotemporal data covariance matrix to guide the null location in the beam pattern. This amount includes the jammer's propagation characteristics. For these techniques to achieve interference rejection, data interchange is a prerequisite. The interference plus noise covariance, which may be challenging to achieve, is a prerequisite for methods that optimize the signal-to-interference plus noise ratio (SINR), such as the maximum voltage distortion ratio (MVDR). With the e approaches, free-running communications signals would not be allowed. The covariance matrices used by the MMSE and MPDR techniques may have a structure linked to the signal of interest, but there is a chance that they will self-null. The MMSE approach is applied to obtain interference rejection capabilities more effectively and to prevent sensitivity toward array element location knowledge. The algorithm team addresses a compound channel extension of the conventional narrowband and wideband MMSE beamforming problems in the remaining portion of this section. To briefly describe the adaptation, the first step is to review the narrowband model. The algorithm team can better comprehend the beamformer's design and performance limitations using narrowband models. The vast form, the most universal version of the answer, is shown next. However, to make this approach practically feasible, the algorithm team addresses modifications made to this generic form in the following section. In this part, I finally talk about performance measurement techniques.

3.3 Narrowband MMSE Beamformer

Our technique builds the beamform using training waveforms. Seeing how the surroundings impact the training data allows constructing a beamformer appropriate for the payload signal. This beamformer minimized the mean square error between the training data and the reception at the terminal gold node[4].

A gold node first transmits the training waveform $\underline{\mathbf{s}}$ to N mosaic nodes. For tile n, the source signal $\underline{\mathbf{s}}$ passes through the n^{th} path of channel A, $h_{A,n}$. Each node also receives an observation of a jamming signal $\underline{\mathbf{j}}$ that passes through a corresponding channel $h_{J,n}$. Lastly, the mosaic accumulates some complex-valued noise $\underline{\mathbf{q}}_n$. The narrowband signal model for the reception at mosaic node n is

$$\underline{\mathbf{z}}_n = h_{A,n}\underline{\mathbf{s}} + h_{J,n}\underline{\mathbf{j}} + \underline{\mathbf{q}}_n \,. \tag{3.1}$$

Matrix $\mathbf{Z}$, defined as

$$\mathbf{Z} = \begin{bmatrix} \underline{\mathbf{z}}_1 \\ \vdots \\ \underline{\mathbf{z}}_N \end{bmatrix} , \tag{3.2}$$

compiles all observations.

Before retransmission, the system applied a distortion transmit filter to the receipt by each mosaic tile. Since the filter is merely spatial, the beamformer modifies only the amplitude and phase.

Mosaic node n transmits

$$w_n^* \underline{\mathbf{z}}_n \,, \tag{3.3}$$

where w_n is one element of the spatial filter specific to node n.

The n^{th} relay signal passes through the n^{th} path of channel B $h_{B,n}$. The reception accumulates more complex-valued noise, $\underline{\mathbf{q}}_B$. They model the gold node reception as

$$\begin{aligned} \underline{\mathbf{y}} &= \sum_{n=1}^{N} h_{B,n} w_n^* \left(h_{A,n}\underline{\mathbf{s}} + h_{J,n}\underline{\mathbf{j}} + \underline{\mathbf{q}}_n \right) + \underline{\mathbf{q}}_B & (3.4) \\ &= \sum_{n=1}^{N} h_{B,n} w_n^* \underline{\mathbf{z}}_n + \underline{\mathbf{q}}_B & (3.5) \\ &= \mathbf{w}^{\mathrm{H}} \mathrm{diag}(\underline{\mathbf{h}}_B) \mathbf{Z} + \underline{\mathbf{q}}_B & (3.6) \\ &= \mathbf{w}^{\mathrm{H}} \mathbf{Y} + \underline{\mathbf{q}}_B \,, & (3.7) \end{aligned}$$

where

$$\mathbf{Y} = \begin{bmatrix} \underline{\mathbf{y}}_1 \\ \vdots \\ \underline{\mathbf{y}}_N \end{bmatrix} = \begin{bmatrix} h_{B,1}\underline{\mathbf{z}}_1 \\ \vdots \\ h_{B,N}\underline{\mathbf{z}}_N \end{bmatrix} . \tag{3.8}$$

Matrix $\mathbf{Y}$ contains in its rows each tile's re-transmitted signal having passed through the various paths of channel B.

The algorithm team now finds a beamformer that minimizes the error between the gold reception $\underline{\mathbf{y}}$ and $\underline{\mathbf{s}}$. Only $w_n^* \underline{\mathbf{z}}_n$is transmitted, so $\mathbf{w}$ must account for $\underline{\mathbf{h}}_B$ ahead of time. The compound channel MMSE problem is

$$\min_{\mathbf{w}} \quad \|\mathbf{w}^{\mathrm{H}}\mathbf{Y} - \underline{\mathbf{s}}\|^2 . \tag{3.9}$$

To consider a more typical MMSE adaptive beamforming problem, set $\underline{\mathbf{h}}_B = \underline{\mathbf{1}}$. This setup shows that the system connects the final channels to the same processing unit. The hardware design has a physical power restriction, but this minimization issue is unrestricted. That constraint is not known to this optimization problem.

$$\left(\mathbf{w}^{\mathrm{H}}\mathbf{Y} - \underline{\mathbf{s}}\right)\left(\mathbf{w}^{\mathrm{H}}\mathbf{Y} - \underline{\mathbf{s}}\right)^{\mathrm{H}} \tag{3.10}$$

$$= \mathbf{w}^{\mathrm{H}}\mathbf{w} - \underline{\mathbf{s}}\mathbf{Y}^{\mathrm{H}}\mathbf{w} - \mathbf{w}^{\mathrm{H}}\mathbf{Y}\underline{\mathbf{s}}^{\mathrm{H}} + \underline{\mathbf{s}}\underline{\mathbf{s}}^{\mathrm{H}} \tag{3.11}$$

To obtain the solution, expand the quadratic term in Equation 3.9, take the derivative while leveraging Wirtinger derivatives because $\mathbf{w}$ is complex, set the result to zero, and solve for $\mathbf{w}$. The derivation is as follows: A closed-form solution exists as

$$\mathbf{w} = \left(\mathbf{Y}\mathbf{Y}^{\mathrm{H}}\right)^{-1}\mathbf{Y}\underline{\mathbf{s}}^{\mathrm{H}} \tag{3.12}$$

$$\mathbf{w} = \mathbf{C}^{-1}\mathbf{r} . \tag{3.13}$$

Rearranging some terms within the MMSE beamformer expression results in the following equation, and it is interesting to note that

$$\mathbf{w} = \Big((\mathrm{diag}(\underline{\mathbf{h}}_B)\mathbf{Z})\,(\mathrm{diag}(\underline{\mathbf{h}}_B)\mathbf{Z})^{\mathrm{H}} \Big)^{-1} \mathrm{diag}(\underline{\mathbf{h}}_B)\mathbf{Z}\mathbf{s}^{\mathrm{H}} \tag{3.14}$$

$$= \mathrm{diag}(\underline{\mathbf{h}}_B^{*})^{-1}\,(\mathbf{Z}\mathbf{Z})^{-1}\,\mathbf{Z}\underline{\mathbf{s}}^{\mathrm{H}}\,. \tag{3.15}$$

It is only possible to simplify this in the narrowband model. This form demonstrates that, with a compensation term for the additional propagation stage, the transmit beamformer is the standard MMSE receive beamformer as if channel B were absent. Each element of the typical receive beamformer is pushed back off by $h_{B,n}^{*}$.

The form $\mathbf{C}^{-1}\mathbf{r}$ commonly shows amongst many other beamformer construction strategies. The cross-correlation vector and the inverse data covariance matrix multiply each other to create the beamformer. Coherent combining properties are provided by the cross-correlation $\mathbf{r}$, which embodies the combined effects of channel B and channel A. The term $\mathbf{Z}\mathbf{s}^{\mathrm{H}}$ differs from the least squares estimate of channel A by the normalization term, $\frac{1}{|\mathbf{s}|^2}$. Interference mitigation properties are provided by $\mathbf{C}^{-1}$. The system comprised the matrix $\mathbf{C}$ of an uncorrelated noise component and a sum of rank-1 components corresponding to each environment source. This mat ix is the solution's only term containing information about the jammer's propagation characteristics. The covariance matrix comprises a weighted sum of rank one components and noise, some of which are correlated. Information about the signal of interest also existed within $\mathbf{C}$.

The algorithm team eventually rescaled the signals to optimize their occupancy of the dynamic range of the digital-to-analog converter (DAC) without clipping, which washes out the original scaling. There is still a chance of self-nulling even with the same performance ceiling.

Similar work with the jamming component might be carried out to examine how covariance estimation mistakes, which distort the jammer's characteristics, affect nulling effectiveness. Similar to this, the exact impact of the beamformer on the jammer may be seen by applying the beamformer's solution to the jammer's channel and solving for it using the Woodbury matrix identity. The inverse data covariance matrix provides the interference-mitigating features. Examine the situation of noise alone.

In actuality, the system has to approximate this sum. Blue nodes increase the number of training sequences in the relay signal to sound the channel. The channel-sounding waveform does not get any beamforming or beamforming filter processing applied.

The terminal gold node receives the training waveform that the channel has distorted in addition to a beamformed payload signal. Subsequently, the gold node can estimate each channel and provide this information to the mosaic. The gold node approximated all channels simultaneously if the transmitter sent the sounding waveforms using a code division multiple access approach.

3.4 Wideband MMSE Beamformer

Transmit $\underline{\mathbf{s}}$ from gold to N mosaic nodes. The sign l $\underline{\mathbf{s}}$ passes through the n^{th} path of channel A, $\mathbf{h}_A$. Each node also receives an observation of a jamming signal $\underline{\mathbf{j}}$ that has passed through the n^{th} path of $\mathbf{h}_J[n]$. Lastly, the mosaic accumulates some complex noise $\underline{\mathbf{q}}_n$. The system has to contain beamforming filters to manage dispersion and be functional. The filters allow the system to support a more extensive range of bandwidths and mosaic node spacing. Here, the algorithm team expands the beamformer formulation and signal models into a wideband operational regime. The algorithm team designed spatiotemporal filters instead of just spatial ones.

The reception model at mosaic node n is

$$\underline{\mathbf{z}}_n = \underline{\mathbf{h}}_{A,n} * \underline{\mathbf{s}} + \mathbf{h}_{J,n} * \underline{\mathbf{j}} + \underline{\mathbf{q}}_n \,, \tag{3.16}$$

Where the * operation indicates a convolution, a finite impulse response (FIR) filter now represents each channel. Mosaic node n transmits after applying a pre-distortion FIR transmit filter on the receiver with T_w taps.

$$\mathbf{w}_n^{\mathrm{H}} * \underline{\mathbf{z}}_n \,. \tag{3.17}$$

The filter length is one design parameter that must be long enough to accommodate a maximum predicted delay spread. Through the n^{th} path of channel B, $\underline{\mathbf{h}}_{B,n}$, travels the n^{th} relay signal. The model for gold reception stated as

$$\begin{aligned}
\underline{\mathbf{y}} &= \sum_{n=1}^{N} \underline{\mathbf{h}}_{B,n} * \mathbf{w}_n^{\mathrm{H}} * \underline{\mathbf{z}}_n + \underline{\mathbf{q}}_B && (3.18)\\
&= \sum_{n=1}^{N} \mathbf{w}_n^{\mathrm{H}} * \underline{\mathbf{y}}_n + \underline{\mathbf{q}}_B && (3.19)\\
&= \sum_{n=1}^{N} \mathbf{w}_n^{\mathrm{H}} \tilde{\mathbf{Y}}_n + \underline{\mathbf{q}}_B && (3.20)\\
&= \mathbf{w}^{\mathrm{H}} \tilde{\mathbf{Y}} + \underline{\mathbf{q}}_B && (3.21)
\end{aligned}$$

where $\underline{\mathbf{y}}_n$ represents the n^{th} re-transmission through its corresponding path in channel B. Define the n^{th} re-transmission through its corresponding channel B as

$$\underline{\mathbf{y}}_n = \underline{\mathbf{h}}_{B,n} * \underline{\mathbf{z}}_n \,. \tag{3.22}$$

The expressions $\mathbf{w}_n^{\mathrm{H}} \tilde{\mathbf{Y}}_n$ implement vector-matrix form convolutions between the retransmission over channel B and the beamforming filter unique to each tile. $\mathbf{w}^{\mathrm{H}} \tilde{\mathbf{Y}}$ is the term that implements the summation and the convolutions of those signal streams. The data matrix $\tilde{\mathbf{Y}}$ is spatiotemporal. The algorithm team first creates component matrices to form this matrix.

$$\tilde{\mathbf{Y}}_n = \begin{bmatrix} \underline{\mathbf{y}}_n[0] & \underline{\mathbf{y}}_n[1] & \underline{\mathbf{y}}_n[2] & \cdots \\ 0 & \underline{\mathbf{y}}_n[0] & \underline{\mathbf{y}}_n[1] & \ddots \\ 0 & 0 & \underline{\mathbf{y}}_n[0] & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix}, \tag{3.23}$$

and then stack them according to the structure,

$$\tilde{\mathbf{Y}} = \begin{bmatrix} \tilde{\mathbf{Y}}_1 \\ \vdots \\ \tilde{\mathbf{Y}}_N \end{bmatrix}. \tag{3.24}$$

The unconstrained minimization problem is

$$\min_{\mathbf{w}} \quad \|\mathbf{w}^{\mathrm{H}}\tilde{\mathbf{Y}} - \underline{\mathbf{s}}\|^2 \,. \tag{3.25}$$

The procedures used in the narrowband formulation are also needed to achieve the closed-form solution. The solution for a wideband beamformer is

$$\mathbf{w} = \left(\tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^{\mathrm{H}}\right)^{-1} \tilde{\mathbf{Y}}\underline{\mathbf{s}}^{\mathrm{H}} \tag{3.26}$$

$$= \mathbf{C}^{-1}\mathbf{r}\,. \tag{3.27}$$

This formulation takes on a more conventional form when the receive antennas link to the same receiver, and each channel sets the B filter to a Dirac delta. The convention $\tilde{\mathbf{Y}}\underline{\mathbf{s}}$ is a backward cross-correlation, as stated in Equation 3.23. Beginning at a later time lag, the training waveform $\underline{\mathbf{s}}$ glides over $\underline{\mathbf{y}}_n$ backward. Through the compounded channels, the relative variations in flight time are adjusted for by this backward cross-correlation.

Lastly, $\mathbf{w}$, structured as

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_N \end{bmatrix}, \tag{3.28}$$

The system includes every N beamforming filter in it. The system can parse out individual filters by reading out continuous sequences coefficients of length T_w, given the $\tilde{\mathbf{Y}}$ structure. Before transferring them to be convolved with $\underline{\mathbf{z}}_n$, corresponding to the optimization problem statement, each $\mathbf{w}_n$ has to be complex conjugated.

Once again, the beamformer constructor has to get a channel estimate. Instead, the system has to estimate the collection of filters. An estimator of least squares utilizing a tapped delay line model expressed as

$$\min_{\underline{\mathbf{h}}_{B,n}} \quad (\underline{\mathbf{y}} - \underline{\mathbf{h}}_{B,n}\tilde{\mathbf{S}})(\underline{\mathbf{y}} - \underline{\mathbf{h}}_{B,n}\tilde{\mathbf{S}})^{\mathrm{H}} \tag{3.29}$$

$$\tag{3.30}$$

can be used here, where

$$\tilde{\mathbf{S}} = \begin{bmatrix} \underline{\mathbf{s}}[0] & \underline{\mathbf{s}}[1] & \underline{\mathbf{s}}[2] & \dots \\ 0 & \underline{\mathbf{s}}[0] & \underline{\mathbf{s}}[1] & \ddots \\ 0 & 0 & \underline{\mathbf{s}}[0] & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix} \tag{3.31}$$

The algorithm team can approximate all channels concurrently if the transmitter sends sounding waveforms using a code division multiple access approach. They can obtain the channel estimation by taking the complex derivative about $\underline{\mathbf{h}}_{B,n}^{\mathrm{H}}$ while utilizing Wirtinger derivatives, setting the result to 0, and rearranging the formula as the estimator is

$$\underline{\mathbf{h}}_{B,n} = \underline{\mathbf{y}}\mathbf{S}^H(\mathbf{S}\mathbf{S}^H)^{-1} \tag{3.32}$$

Chapter 4

ALGORITHM DESIGN AND HARDWARE IMPLEMENTATION CHALLENGES

Adapting the previously provided solutions for the specific hardware implementation and real-world applications is necessary. In my research, I implemented the design on the FPGA platform. Numerous modifications are required from the typical MMSE problem.

The beamformer construction formulae covered in the previous section are better suited for systems that use preload training sent via the same spectral support as the tactical signal. The use of underlays causes issues with self-interference. It is also essential to think about data exchange rate concerns. Providing a shorter cycle route becomes necessary when this worry is transferred to the hardware.

The challenges instantly break two key assumptions: data availability and time-frequency synchronization. I also must put this into practice on hardware. Many decisions regarding algorithm adaptation and system design are ultimately driven by the need to satisfy these two needs.

4.1 Synchronization Methods

The wideband MMSE beamformer structure described above presupposes the usage of observation data, wherein each stream is recorded beginning at a specific global time, and the transmission and reception center frequencies are consistently synchronized throughout all components. Because the nodes are dispersed, each has a separate clock source, which might have oscillation frequencies that are out of sync, leading to offsets in the clock time and carrier frequency. In center frequency misalignments, there are discrepancies in local oscillator frequencies. Furthermore, there's a chance that every source and receiver in the environment will be mobile, resulting in additional random frequency offsets via Doppler shifts.

A depiction of the network labeled with all the possible frequency offsets is shown in Figure 4.1. If frequency alignment is poor enough, then poor correlations may be observed. More importantly, the payload signal will not be able to maintain coherence. This statement means the system has stringent timing requirements, and the software design can hardly satisfy the criteria. If timing alignment is poor, then the geometry of the network is inaccurately represented, which will produce a poorly optimized beamformer that points the main lobe and nulls in incorrect directions.

The network has to adjust the time and carrier frequency offsets to achieve the required beamforming gain and interference rejection at the receiver. It is necessary to have synchronization regardless of the relay technology employed.

Additionally, the algorithm team will suppose a Doppler shift exists between the Gold a and Gold b mosaics. The Doppler shift results from the relative velocity of Gold B and the mosaic pieces.

If the frequencies are not aligned, the interference protection performance falls off according to some equation. In Figure 4.2, the behavior for a two-element array frequency offset increases.

The provided MMSE beamformer construction explicitly corrects for time and phase offsets. However, mosaic tiles must be sufficiently synchronized to accurately represent the time difference of arrival (TDOA) information. The mosaic must also be punctual while signaling toward the gold receiver so that the channel estimator can also accurately represent TDOA information. A shared sense of time should be established and can be accomplished by declaring a master mosaic node for all others to reference. Beamformer construction is somewhat robust in the case of misalignments between each mosaic's sense of global time and a genuine global timeline if it

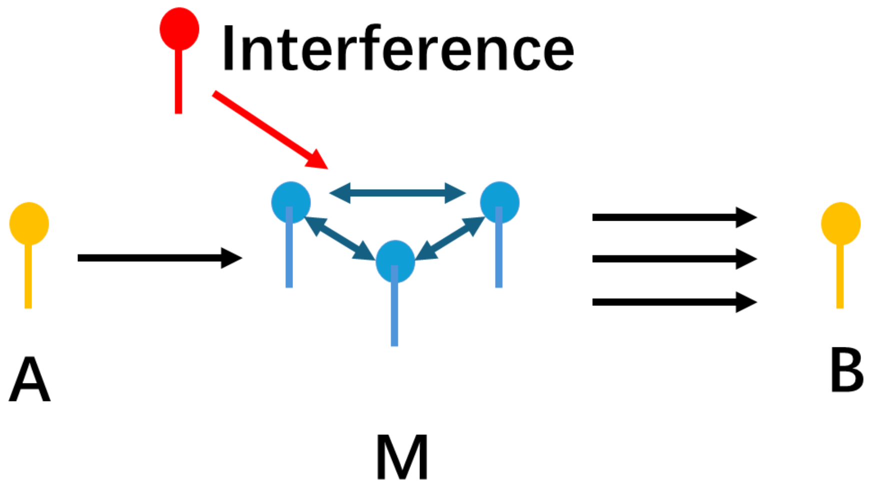


Figure 4.1: Sources of Error.

meets certain conditions. One of these conditions is that mosaic data captures, and the subsequent transmission of the channel-sounding waveform obey the same offset. The transmission causes the misalignment embedded in the channel B estimate and will be corrected when the beamformer is applied. For example, suppose a mosaic node captures information early. In that case, it will also transmit early conditions that the misalignments do not cause signal arrivals to miss the analysis window, which is related to the maximum delay spread the mosaic can support. Alignment should be accurate within one chip of the tactical signal. Misalignment is more tolerable when achieving only SNR gain is sufficient than when effective interference rejection is desired. Lastly, any misalignments must be stable and jitter minimally. A mechanism to coordinate the timing between the mosaic nodes is required.

The provided MMSE beamformer architecture does not directly account for frequency offsets. It can implicitly make little adjustments, but only for a limited range of offsets. One can create augmentations that explicitly rectify carrier frequency

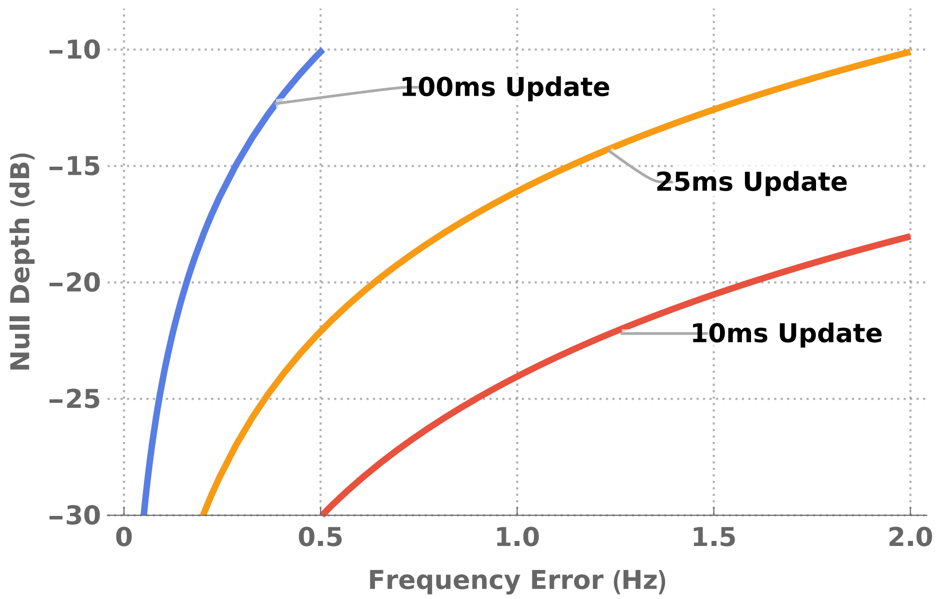


Figure 4.2: Frequency Error vs. Update Rate for a two-element antenna array[4]

alignments, albeit at a much more considerable computational expense. An extra device is needed to synchronize the broadcasts and receptions. Every node in the DCMB network needs to compensate for its offset from a reference frequency. In a relay network, the reference node is typically identified. Upon receipt, each mosaic tile needs to have its offset from the gold transmitter adjusted. To ensure that signals are correctly centered at the gold receiver after down-conversion, each mosaic node must have its own pre-correction applied during transmission.

This technique requires that the system performs beamformer computation as though the mosaic to gold offset does not exist. The hardware computations will take this into account. The estimate may require a phase ramp correction before feedback, contingent on the channel B estimating approach. Estimates for Channel B need to be adjusted so that the gold node can use the training data obtained at the nominal center frequency to create the initial estimate. Compound adjustments will result in inaccurate beamformers. The algorithm team pinpointed every area that needs correction and every area that is cause for concern. Every element in the mosaic will have a frequency correction applied, consistent from Gold A to the mosaic. Every mosaic tile will then undergo separate frequency corrections to Gold B. It will be calculated at Gold B and returned.

They apply estimators at the mosaic and gold nodes to calculate the offsets and Doppler shifts that must be adjusted for oscillator mismatch. It is not the frequency offset size but the frequency offset estimate error, which is the performance-affecting measure that is the cause for concern. Assume, for example, that nominally operating clocks powered all radios. Even so, a frequency offset is still possible if the nodes move. It is easier to correct offsets resulting from smooth, predictable motion than from chaotic motion, which is harder to detect and has a higher variance in the estimation error. The problems arise from discontinuous disciplining in the context of faulty clocks. To adjust to changes in the network and surroundings, the mosaic needs to update the beamformer regularly. The primary factor determining how frequently updates are required, though, is the accuracy of the frequency offset estimator. Any residual error will contribute to excess phase rotation due to the lag between estimating the offset measurement and applying the correction, potentially leading to the signals losing coherence. Update frequency is reduced if the estimator's accuracy is exact. While updating often increases the estimator's margin for error, other system components, such as information sharing and beamformer computation, also need to operate more quickly. Decreasing the building time and increasing the data rates is not an easy process.

For an N-element antenna array, SNR gain[28] as a function of the update interval τ and estimation error variance σ^2 is given by

$$(N^2 - N)(e^{-4(\pi\tau)^2\sigma^2}) + N \, . \tag{4.1}$$

The INR reduction[9] parameterized in the same way is given by

$$\frac{N-1}{N}(1 - e^{-4(\pi\tau)^2\sigma^2})\,. \tag{4.2}$$

People can project estimation performance by constructing the cramer-rao lower bound (CRLB) for frequency offset estimation. The CRLB can be parameterized according to the observation's integrated SNR (ISNR). Whether or not an estimator hits the lower bound depends on its construction. A bound that allows sufficient SNR gain and INR reduction[28] should be targeted. The ISNR associated with that bound can be used to inform waveform design and the signaling strategy. A bound should be targeted to provide some tolerance for achieving the required performance when the bound cannot be met precisely. This bound maps to some ISNR, which can inform waveform and signaling strategy aspects.

4.2 Data Exchange and Limited Transmitting Rates

Program limitations are in place to reduce data exchange. Constrained by practicality is another issue. This may have an impact on interface and hardware design. The interface manages the data rate in addition to accelerator designs. In terms of data throughput, accelerator requirements have grown. The beamformer's effectiveness depends on our capacity to estimate the cross-correlation and the spatiotemporal data covariance matrix precisely within a time window when the statistics are deemed stationary, provided that the time-frequency alignment is sufficiently precise. In the end, estimation performance depends on how long training and observation data are available.

The beamformer computation requires the observations of all mosaic nodes to be compiled onto some processing agent. This information must be exchanged through an intramosaic data exchange network. Because all mosaic nodes are distributed, a hefty restriction is placed on the amount of information that can be practically exchanged. Due to the beamformer update rates needed, the amount of information that can be practically exchanged is heavily restricted. Furthermore, one program goal is to minimize the information exchange between nodes. Data demands can quickly become expensive as well. These goals oppose each other, leading to operational constraints and further algorithm adaptations. Several methods are employed to reduce the traffic needed to build the MMSE beamformer. All these methods can help with the hardware design, which means a reduced data rate and lower latency in the computation time.

The dimensionality of the covariance matrix determines the minimal number of observation samples required to obtain a certain level of estimation accuracy. The more dimensions there are, the more samples are needed. The quantity of beamforming filter taps and tiles determines how many dimensions are required. From an alternative viewpoint, the size of the covariance matrix determines how effective those samples are, as the update rate restricts the amount of information that can be shared. How many samples of each $\underline{\mathbf{z}}_n$ that can be shared throughout the network determines the estimation accuracy; however, the dimensionality determines how useful they are.

Estimating the cross-correlation depends on the number of samples available and the worst-case underlay to interference plus noise ratio (UINR) supported at the mosaic. The number of samples required to estimate the cross-correlation reasonably depends on the worst-case underlay to interference plus noise ratio (UINR) supported at the mosaic. The underlay is generally significantly jammed. Transmitting the number of raw samples to overcome this jamming is infeasible, especially given the required update rates. Exchanging a locally computed cross-correlation result is much cheaper, in which case the amount of data needed to be exchanged depends only on the number of beamformer taps used.

Therefore, each node must transmit the maximum number of correlation and covariance estimation samples, which can support information exchange strictly between mosaic nodes. Mosaic tiles need to send extra data in addition to these bits to support time-frequency synchronization between mosaic nodes and uniquely identify the mosaic tile. It is necessary to take into account elements like forward error-correcting codes. The capacity to reasonably estimate the mosaic to gold channel is inherent to estimating the covariance matrix and the cross-correlation vector. The estimation quality depends on the number of samples required to overcome a substantial UINR at the gold node, just like the cross-correlation.

But all that needs to be sent back to the mosaic are the estimations themselves. The number of tiles that are supported and the number of taps that are used determine how much data needs to be delivered. The transmitter used a portion of the underlying transmission cycle; this estimate is transmitted back across a gold-to-mosaic channel. Because of the propagation length, this program targets some applications where at least one gold-mosaic transmission stage experiences considerable attenuation. The supported data rates are often modest in these situations. Given a particular amount of mosaic tiles, the designer must be careful about how many taps it can help.

The observation samples needed for covariance estimation are the most costly data set. In certain situations, estimating the mosaic to be sent to the gold channel can be expensive. The necessary data rates are determined by the number of taps designated for the beamforming filters.

The number of beamforming filter taps is a design choice tied to the maximum expected delay spread of the end-to-end transmission when the payload signal passes through the compounded channels and the sampling rate. The worst-case delay spread in samples for a sampling rate f_s is

$$\tau_{max} = 2\frac{D_{max}}{c} f_s \,, \tag{4.3}$$

$\frac{D_{max}}{c} f_s$ is the delay spread for a single stage. The system's operating constraint is the maximum pairwise separation D_{max} between two mosaic nodes. The worst-case situation could be the consequence of a corner case. Lowering the number of filter taps needed may be possible if the designer wants to improve the mosaic nodes' placement can be improved. Our design assumes that could have to deal with the worst scenario. Practically speaking, it is advisable to add additional taps to account for sinc forms that result from fractional delays, which are crucial to maintain. After considering extra buffer samples and the maximum delay spread in samples, then gets the tap length T_w.

Using the wideband MMSE formulation directly means building a beamformer that affects at least the chipping rate of the underlay depending on the oversampling rate. For a practical set of mosaic geometries, an impractically large covariance matrix tends to result in an infeasible number of training observation samples being exchanged. The covariance matrix's construction and inversion would also place an immense computational burden on the system.

To reduce demands on the data exchange links, people have elected to build a beamformer that only affects the spectral support of the payload signal. Achieving gain and interference rejection within that space is sufficient. Any interference received beyond the spectral support of the payload waveforms can be filtered out using a low-pass filter.

Our system builds the beamformer at a sampling rate that oversamples the tactical waveform by a small multiple more significant than 1. For the same geometry, fewer taps are needed in this strategy. Although a higher bandwidth training waveform is transmitted, then estimate covariance matrices and correlations such that they characterize the spatiotemporal relationships strictly within the band of interest. Augmentation of the covariance matrix's dimensionality is not as severe, making its estimation much more manageable. Given a fixed number of samples that the intramosaic network can support exchanging, convergence toward an acceptable estimate is more easily attained.

This beamformer can be applied to the relay signals in one of two ways. The first option is to use the filter to bent-pipe data that has been first downsampled to the same processing rate at which the beamformer was constructed. The result is resampled to the system sampling rate and prepared for retransmission. The second option is to keep the bent pipe data sampled at the system rate, use a low-pass filter closely around the tactical spectrum, upsample the beamforming filter, and apply the filter to the pre-processed bent pipe data.

Despite this modified construction, the required network traffic to support it remains relatively high. Trading off algorithmic simplicity for demands on the information exchange network, they relegate reducing traffic demand toward future research. Prospective techniques for reducing the required data rates include beamformer tracking, subspace tracking, and source compression.

4.3 Modified Wideband MMSE Beamformer

To make data exchange and computation more feasible, the algorithm team built a beamforming filter sampled at a lower rate, affecting a much smaller portion of the spectrum than the underlay. Our strategy is still to solve an MMSE problem, but there is a mismatch between the spectral support of the training data and the signal, whose error to minimize. The beamformer is still built based on the form $\mathbf{w} = \mathbf{C}^{-1}\mathbf{r}$. However, extra care must be taken because the computation is distributed, and components are computed at different sampling rates. A block diagram describing the flow of operations within each mosaic node in Figure 4.3.

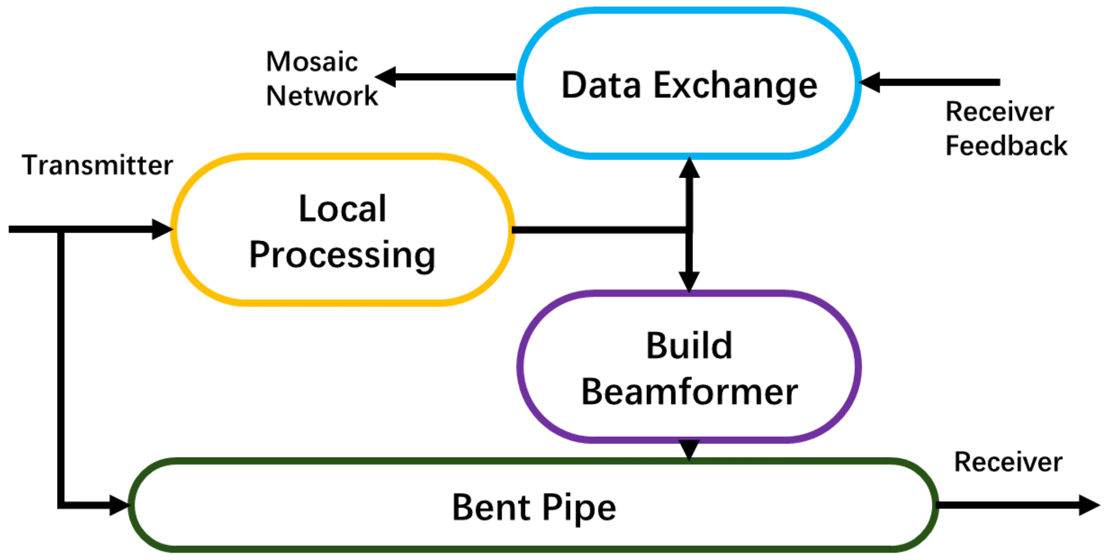


Figure 4.3: Block Diagram of Adapted Distributed Mmse Computation.

Firstly, the receiver must capture training data carefully. The radios operate with independent clocks and synthesizers, so each has its sense of time. Due to differences in propagation distances, the signal emitted from the gold transmitter arrives at each of the blue nodes at slightly different times according to some common sense of time amongst the nodes. The beamformer corrects for differences in propagation distances, so capturing and fusing the data in a way that preserves this information is essential. Ideally, the mosaic should collectively capture data like an antenna array wired to the same clock source. In other words, the first sample in each capture should refer to the exact timestamp of some reference timeline. Additionally, data must be captured so that all detections appear within a limited window. Some reference is needed to cue captures to ensure this requirement is satisfied. One method is scheduling captures based on the earliest training signal arrival at any mosaic node. Mosaic nodes need to establish some sense of global time to be successful, which requires additional coordination. Assuming such a mechanism exists and is integrated with the system.

Pieces of the beamformer computation are completed at several different sampling rates. All data begins sampled at the system rate f_S. Data passing through the bent pipe remains sampled f_S. However, the beamformer computation consumes data first downsampled to f_U. This rate f_U is 1.5 or 2 times the underlay chipping rate B_U, depending on the implementation version. Eventually, computation products are converted to characterize dynamics sampled at $f_T = f_U/8$, which oversamples the maximum supported tactical bandwidth by a factor of 1.5. Mosaic tiles compute the beamformer sampled at f_T and then upsample them to f_S to apply the filters to the bent pipe data stream directly.

Granted a proper data capture, each tile performs pre-processing on local observations. They then provide intermediate computation products to each other. These products are fused and post-processed. Then, each mosaic tile computes the beamformer filters.

Hypothetically, if compile all the raw captures together, this would be given by This is given by

$$\mathbf{Z} = \begin{bmatrix} \underline{\mathbf{z}}_1 \\ \vdots \\ \underline{\mathbf{z}}_N \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{h}}_{A,1} * \underline{\mathbf{s}}_{U1} \\ \vdots \\ \underline{\mathbf{h}}_{A,N} * \underline{\mathbf{s}}_{U1} \end{bmatrix} \tag{4.4}$$

The channels $\underline{\mathbf{h}}_{A,n}$ modeled here have embedded time difference of arrival (TDOA) information. The first entry in each row refers to the same global timestamp.

4.3.1 Cross-Correlation

Cross-correlations must be computed locally at the higher sampling rate f_U to maximize the available integration gain. Then, the result must be downsampled to be compatible with the spatiotemporal covariance matrix. The cross-correlation should not be directly estimated at f_T because doing so would require first downsampling the data vector, significantly attenuating the training waveform.

Notice the term $\mathbf{r} = \tilde{\mathbf{Y}}\underline{\mathbf{s}}^{\mathrm{H}}$ in Equation 3.27, the wideband MMSE formulation presented earlier. The training data is cross-correlated against each $\underline{\mathbf{y}}_n$, the result of convolving the channel estimate with the corresponding training observations. In this case, the same computation should not be performed directly because the channel estimate must be returned from the gold node downsampled to rate, f_T. Instead, the cross-correlation between $\underline{\mathbf{z}}_n$ and $\underline{\mathbf{s}}$ should be performed first as follows:

$$\mathbf{r}'_n = \tilde{\mathbf{Z}}_n \underline{\mathbf{s}}_{U1}^{\mathrm{H}} \,, \tag{4.5}$$

where

$$\tilde{\mathbf{Z}}_n = \begin{bmatrix} \underline{\mathbf{z}}_n[0] & \underline{\mathbf{z}}_n[1] & \underline{\mathbf{z}}_n[2] & \cdots \\ 0 & \underline{\mathbf{z}}_n[0] & \underline{\mathbf{z}}_n[1] & \ddots \\ 0 & 0 & \underline{\mathbf{z}}_n[0] & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix} . \tag{4.6}$$

The reference signal $\underline{\mathbf{s}}_{U1}$ should contain the transmit filter and the receive filter that processes the data $\underline{\mathbf{z}}_n$ beforehand, if applicable, to estimate the over-air-channel. The transmit and receive filters are a part of the underlay's channel but not the tactical signal. Only the standard pieces of the channel should be left over in the cross-correlation. This intermediate cross-correlation $\mathbf{r}'_n$ result is downsampled and stored as $\hat{\mathbf{r}'}_n$.

Poor UINR is expected, approximately -30 dB in the worst case. With approximately 40 dB worth of integration gain, hope to build cross-correlations where peaks of energy associated with the signal arrivals can be reliably detected. However, the training sequence length cannot effectively suppress the off-peak correlations. If just a few samples of a more extended sequence (on the order of 100) make up the detection peak, convolving an anti-alias filter with this sequence will likely bury the channel information. Most of the information within the sequence is unrelated to the underlay itself. Here, leveraing windowing to help preserve this information during downsampling.

Assume that the system operates within a line-of-sight dominant channel. The algorithm team further leans on this assumption and applies a rectangular window centered on the detection peak. The window is short, with just a few samples on either side of the energy peak, so any sinc structure present can be preserved. Observing a tail corresponding to some scattering behavior is possible in higher UINR cases. If the line of sight component is strong enough and the tail short enough, these components can be negligible. The tail is likely poorly estimated. Applying the window results in a cross-correlation that appears as if a smaller analysis window was used, and the result was zero-padded. Applying the window should not destroy the TDOA information. They filter and decimate the cross-correlations. The goal is to ensure the downsampled cross-correlations maintain the correct phase offset, relative amplitude, and timing offsets. Having extraneous information in the source material only aids in corrupting this information. This rectangular window enforces a line-of-sight dominant structure. Performance will break down if the narrower-band channel is not line-of-sight dominant. The resulting cross-correlation should resemble a fractional delay filter. Each tile provides its own $\hat{\mathbf{r}'}_n$ to every other tile.

Finally, leveraging the commutative property of convolutions, the channel estimate $\hat{h}_{B,n}[\tau]$ can be convolved with the intermediate cross-correlation $\hat{r'}_n[\tau]$ to produce the required cross-correlation $\hat{r}_n[\tau]$. The convolution between $\hat{h}_{B,n}[\tau]$ and the $\hat{r'}_n[\tau]$ must be performed in the correct direction. Recall that the beamformer formulation uses a cross-correlation stored in the backward direction,

$$\hat{\mathbf{r}'}_n = \begin{bmatrix} \hat{r'}_n[T_w] \\ \vdots \\ \hat{r'}_n[0] \end{bmatrix}, \tag{4.7}$$

which corrects timing offsets. The full-length convolution is

$$\hat{r}_n[\tau] = \left(\hat{h}_{B,n} * \hat{r'}_n\right)[\tau]\,. \tag{4.8}$$

After compensating for filter delay and truncation, the result is

$$\hat{\mathbf{r}}_n = \begin{bmatrix} \hat{r}_n[T_w] \\ \vdots \\ \hat{r}_n[0] \end{bmatrix}. \tag{4.9}$$

where n denotes the specific node where this particular cross-correlation is computed. Care must be taken when truncating the full-length convolution, where the same window of time is selected from all $\hat{r}_n[\tau]$. The compounded channel response from $\underline{\mathbf{h}}_{A,n}$ and $\underline{\mathbf{h}}_{B,n}$ must remain intact within the window. The cross-correlation computation is complete after compiling all vectors into the order given by

$$\hat{\mathbf{r}} = \begin{bmatrix} \hat{\mathbf{r}}_1 \\ \vdots \\ \hat{\mathbf{r}}_N \end{bmatrix}. \tag{4.10}$$

4.3.2 Covariance Matrices

Building a lower sampling rate spatiotemporal covariance matrix is straightforward. Each tile must capture an excerpt of the f_U sampling rate data, $\underline{\mathbf{z}}_n$. The capture window must refer to the same block of global time across all tiles. This data excerpt is downsampled to form $\hat{\underline{\mathbf{z}}}_n$. These vectors retain TDOA information of the gold and jammer transmissions but are sampled at a lower rate, f_T. Again, downsampling removes all out-of-band information, so the beamformer only rejects interference within the band of interest. Only $\underline{\hat{\mathbf{z}}}_n$ needs to be exchanged between tiles.

Assuming that the data vectors have been exchanged, the computation can proceed according to the earlier stated wideband MMSE formulation. First, each data vector is convolved with the channel estimate $\underline{\hat{\mathbf{h}}}_{B,n}$ and stored as

$$\hat{\mathbf{Y}} = \begin{bmatrix} \underline{\hat{\mathbf{y}}}_1 \\ \vdots \\ \underline{\hat{\mathbf{y}}}_N \end{bmatrix} = \begin{bmatrix} \underline{\hat{\mathbf{h}}}_{B,1} * \underline{\hat{\mathbf{z}}}_1 \\ \vdots \\ \underline{\hat{\mathbf{h}}}_{B,N} * \underline{\hat{\mathbf{z}}}_N \end{bmatrix} . \tag{4.11}$$

Spatiotemporal data matrix $\tilde{\mathbf{Y}}$ is built from $\hat{\mathbf{Y}}$ in the following way:

$$\tilde{\mathbf{Y}}_n = \begin{bmatrix} \underline{\hat{\mathbf{y}}}_n[0] & \underline{\hat{\mathbf{y}}}_n[1] & \underline{\hat{\mathbf{y}}}_n[2] & \cdots \\ 0 & \underline{\hat{\mathbf{y}}}_n[0] & \underline{\hat{\mathbf{y}}}_n[1] & \ddots \\ 0 & 0 & \underline{\hat{\mathbf{y}}}_n[0] & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix} \tag{4.12}$$

$$\tilde{\mathbf{Y}} = \begin{bmatrix} \tilde{\mathbf{Y}}_1 \\ \vdots \\ \tilde{\mathbf{Y}}_N \end{bmatrix} . \tag{4.13}$$

The organization of $\tilde{\mathbf{Y}}$ must correspond to that of $\hat{\mathbf{r}}$. Practically, the spatiotemporal data matrix should not be explicitly formed.

The spatiotemporal covariance matrix is constructed by evaluating

$$\mathbf{C} = \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^{\mathrm{H}} . \tag{4.14}$$

Our system is explicitly sample starved because of not enough bandwidth between the mosaic tiles to share sufficient samples, even when computing a lower sample rate beamformer. Fewer samples of $\underline{\hat{\mathbf{z}}}_n$ than there are dimensions in $\mathbf{C}$ are exchanged, which means the covariance matrix is poorly estimated. Furthermore, $\mathbf{C}$ will almost certainly be poorly conditioned. Employ averaging to refine the covariance matrix estimate to resolve this issue.

Specifically, we employ an exponential moving average. This procedure is expressed as

$$\mathbf{C}_t = (1 - b)\mathbf{C}_{t-1} + b\mathbf{C}\,, \tag{4.15}$$

where b is a forgetting factor, $\mathbf{C}$ is the most recent covariance computation, $\mathbf{C}_{t-1}$ is the average from the previous construction epoch, and $\mathbf{C}_t$ is the current average. The forgetting factor b weights the history of all $\mathbf{C}$ such that the average depends most on more recent estimates. Over time, the influence a previous estimate has on the average decays. We employ this averaging instead of a moving average due to its computational simplicity and ability to adapt to a dynamic environment. The system requires a few iterations of training from a startup to produce a helpful covariance matrix. When network nodes have high mobility, then the pseudo-stationary assumption is violated over a shorter period. In these situations, the averaging mechanism must rely less on the past. Consequently, the covariance matrix is at risk of being poorly estimated again.

Additional tools are needed to stabilize the computation further and make the algorithm robust. In sample-starved scenarios, the covariance matrix tends to be poorly conditioned. Spatiotemporal data matrices tend to contain plenty of redundancy, exacerbating poor covariance matrix conditioning. Inverses of poorly conditioned covariance matrices will introduce heavy distortions into the beamformer. One common technique that permits the construction of a sensible beamformer despite these scenarios is L2 regularization[9], otherwise known as diagonal loading. L2 regularization introduces a penalty term into the MMSE objective function. The new optimization problem is given by

$$\min_{\mathbf{w}} \quad \mathbf{w}^{\mathrm{H}}\mathbf{C}_t\mathbf{w} - \mathbf{w}^{\mathrm{H}}\mathbf{r} - \mathbf{r}^{\mathrm{H}}\mathbf{w} + \|\underline{\mathbf{s}}\|^2 + d\|\mathbf{w}\|^2 \,, \tag{4.16}$$

Where d is a parameter set by the user. The result of taking the derivative, setting it equal to zero, and solving for $\mathbf{w}$ yields The minimization problem has a closed-form solution expressed as

$$\mathbf{w} = \left(\mathbf{C}_t + d\mathbf{I}\right)^{-1}\hat{\mathbf{r}} \,. \tag{4.17}$$

The parameter d is the loading level related to the penalty weight for increasing the norm of $\mathbf{w}$. Diagonal loading constrains the white noise gain, often the most poorly estimated component of the covariance matrix.

The loading level d must be judiciously set. Increasing d too much causes the covariance matrix to function as a scaled identity matrix, in which case the beamformer will be proportional to $\mathbf{r}$. Consequently, excessive loading destroys any interference mitigation properties. Setting the d loading level too low may fail to stabilize the computation, producing nonsensical filters. The appropriate diagonal loading depends on the scenario but is often referenced from either the maximum eigenvalue or the system noise floor. The former method is expressed by

$$\alpha\lambda_1 \,, \tag{4.18}$$

where α scales down the maximum eigenvalue λ_1. A more practical approach is to use a proxy measure of the maximum eigenvalue instead. The simplest alternative is

$$d_1 = \alpha\frac{\mathrm{tr}(\mathbf{C})}{T_w} \,, \tag{4.19}$$

the trace is divided by T_w to account for the redundancy introduced by the spatiotemporal variant. The latter is expressed by

$$d_2 = \beta \max_n |\hat{\underline{\mathbf{h}}}_{B,n}|^2\sigma_n^2 \,, \tag{4.20}$$

β is a parameter that scales up the strongest noise floor. The noise floors considered are the noise floor of $\hat{\underline{\mathbf{z}}}_n$, σ_n^2, with an adjustment for the channel estimate applied to the data. The noise floor is obtained by performing calibration measurements on the system. Given some fixed automatic gain control (AGC) setting, $\hat{\underline{\mathbf{z}}}_n$ must be recorded with the front end port terminated. The signal power of this recorded data is a reference point. This calibration value is adjusted according to the AGC state and internal normalization to produce σ_n^2 before use with (4.20).

The signal-to-interference ratio (SIR) measured at the mosaic is the primary determinant of the diagonal loading level. Assessing this connection in real-time on the system is prohibitive due to processing capacity limitations. The trace-based approach is suitable for many high-SIR circumstances when α is set appropriately. The noise-based approach is ideal for a variety of low SIR scenarios that the other approach might not be able to handle, provided that β is set appropriately.

Both loading methods are considered, and the system switches between them to robustly and automatically adapt to the environment. Mosaic nodes switch between the two methods according to the following rule:

$$d = \max\{d_1, d_2\} \,. \tag{4.21}$$

As the SIR changes, so will d_1, but d_2 will remain constant. The switching rule sets a lower limit on the loading level d.

Covariance matrix tapering is another technique that can be used to add robustness. Covariance matrix tapering applies a phase dither in time and space to increase the spatial extent of every emitter represented. The main lobe and nulls of the beamformer are widened as a result. When the covariance matrix is poorly estimated, the interferers are inaccurately represented in space. Mismatches between the estimate and truth will yield ineffective interference mitigation. Widening the nulls provides a better opportunity to mitigate the interference. Tapering also suppresses sidelobes of the beamforming pattern, and any error in the beamforming target's statistics is compensated by the wider main lobe. Taper the covariance matrix is implemented by first constructing a tapering matrix. The construction of the tapering matrix $\mathbf{T}_{\mathrm{DLMZ}}$ (including diagonal loading) is given by

$$\mathbf{T}_{\mathrm{DLMZ}} = (\mathbf{T}_{\mathrm{MZ}_n} \otimes 1_{n_{taps} \times n_{taps}}) \odot (1_{N \times N} \otimes \mathbf{T}_{\mathrm{MZ}_\tau}) + d\mathbf{I} \tag{4.22}$$

$$\mathbf{T}_{\mathrm{MZ}_n} = [\mathrm{sinc}((m-n)\Delta_n)] \tag{4.23}$$

$$\mathbf{T}_{\mathrm{MZ}_\tau} = [\mathrm{sinc}((m-n)\Delta_\tau)] \, , \tag{4.24}$$

where $\mathbf{T}_{\mathrm{MZ}_n}$ tapers spatially and $\mathbf{T}_{\mathrm{MZ}_\tau}$ tapers temporally. We apply the taper through a Hadamard product with the covariance matrix. To solely rely upon diagonal loading, an all 1s matrix can be used for the taper. All these computations are implemented on hardware. I designed every accelerator to compute these functions and tested the accelerator with test data. This way, I can ensure the accelerator performs well enough to operate the required tasks. I set the standard as 40dB SNR by comparing the MATLAB and FPGA output results data. With this precision, the implementation can perform interference mitigation and N^2 gain at the receiver.

4.3.3 Channel Estimation

The channel estimation process is similar to cross-correlation computation, where the estimate must be computed at the higher f_U sampling rate first and downsampled afterward. The channel estimation is where I utilize the convolution accelerators to achieve channel estimation functions. The difference is that the estimation carries the usual way, according to Equation 3.32, without any intermediate downsampling. The only actual expense of the computation is the cross-correlation between the data $\underline{\mathbf{y}}$ and the $\underline{\mathbf{s}}_{U2}$. Like $\underline{\mathbf{s}}_{U1}$, $\underline{\mathbf{s}}_{U2}$ should contain the transmit and receive filters if applicable when performing the cross-correlation. The covariance matrix is built purely on the training data. Furthermore, the inverse covariance matrix can be computed offline and stored. This covariance matrix contains plenty of redundancy and tends to have a poor condition number. We are leveraging diagonal loading again to remedy this issue.

The sounding underlay will inevitably be strongly jammed. This estimator has far more integration gain available than the mosaic tile cross-correlators because the sounding underlay is not required to carry data from the mosaic to the gold node. Regardless, we enforce the strong line of sight assumption on the mosaic to gold b channel estimates and use a short number of taps in the tapped delay line model. The analysis window is centered around the detection peak. This idea is similar to the windowing used on $\mathbf{r}'_n$. Estimates are computed as training sequences arrive. Time stamps of the analysis windows are documented.

The sounding underlay is transmitted from each mosaic tile in a TDMA fashion. The gold node will observe arrivals spaced out by TDMA slot size with deviations corresponding to differences in propagation distances. Consider that a channel estimate for each tile to gold node path, $\underline{\mathbf{h}}_{B,n}$, has been estimated but not yet downsampled. Implicit in these channel estimates, despite being short in length and calculated separately from each other, is the TDOA information. The TDOA includes timing related to the schedule and the differences in propagation time. The channels need to be realigned such that the TDOA information only includes differences in propagation distance. To realign the responses, the gold node assumes the exact scheduling and determines offsets in arrival times. Based on this offset information, the channel estimates sampled at f_U can be placed into a sampling lattice according to the corrected timing. The realigned collection of estimates characterizes a group of responses as if the training was emitted simultaneously from all tiles. The channel estimates are now filtered and decimated, producing a set of estimates $\hat{\underline{\mathbf{h}}}_{B,n}$ that are now sampled at f_T. This estimate is especially sensitive to residual frequency offset errors because the channels are sounded in a TDMA fashion. The gold node must account for and compensate for a much longer phase ramp.

4.3.4 Building the beamforming filters

Let us denote the postprocessed covariance, which includes diagonal loading and any desired tapering, as $\mathbf{C}_{REG}$. There is a matrix inverse computation in the equation below. I also applied a QR decomposition and back substitution algorithm to handle the inversion. Tiles then solve the expression

$$\hat{\mathbf{w}} = \mathbf{C}_{REG}^{-1}\hat{\mathbf{r}}\,. \tag{4.25}$$

Each tile reads out its own $\hat{\mathbf{w}}_n$. We choose to resample the beamforming filters to the system sampling rate f_S and apply them to the full rate bent-pipe signal. Denote the resampled filters as $\mathbf{w}_n$. Each tile only needs to resample its filter. The parsed-out filters are already time-aligned appropriately. The resampling process must preserve this structure at a higher sampling rate. The set of filters should appear to be a collection of sinc shapes. The payload must have been filtered low-pass beforehand to isolate the spectrum of interest. The pre-distorted payload signal is

$$\mathbf{w}^{\mathrm{H}} * \underline{\mathbf{h}}_{\mathrm{LPF}} * \underline{\mathbf{z}}_n\,. \tag{4.26}$$

4.4 Data Exchange and Accelerator Scheduler

The requirement for the beamformer is to update every 25 ms. Completing the computation requires waiting for several pieces to be exchanged across the network. Only when all intermediate computation products have been exchanged can a beamformer be computed. The overall process has been split into three parallel schedules for pipeline beamformer construction, making adhering to the update rate possible. Each scheduler in hardware hardware controls the components in one timeline, and then I composed them together into one comprehensive scheduler. State machines are implemented multiple times in the scheduler.

Timeline 1 consists of operations that preprocess the data. Training information is transmitted from gold A on a strict and consistent 25 ms schedule. The mosaic nodes receive the training data and immediately begin processing. Each node produces both $\hat{\underline{\mathbf{z}}}_n$ and $\hat{\mathbf{r}'}_n$. These operations should be completed well before receiving the following training observation. Timeline 1 repeats on a strict 25 ms cycle due to gold A's transmission schedule. The mosaic nodes stage their results for exchange, which triggers the following timeline.

Timeline 2 encompasses the data exchange and prepares the data for computing the beamformer. Each mosaic node takes turns broadcasting its information within a 25 ms window using a TDMA structure. The average time per slot is 2.5 ms, given ten nodes. Upon reception, nodes convolve $\hat{\underline{\mathbf{z}}}_n$ and $\hat{\mathbf{r}'}_n$ with corresponding $\hat{\underline{\mathbf{h}}}_{B,n}$. Timeline 2 repeats its 25 ms cycle due to the TDMA structure. This cycle of events is offset from timeline one due to the computation execution time and propagation delay. Completing these computations triggers a third timeline.

Timeline three encompasses the MMSE computation. Once a cycle of timeline two has been completed, each node has all the necessary $\hat{\underline{\mathbf{y}}}_n$ and $\hat{\mathbf{r}}_n$ vectors, so the nodes finally complete the beamformer computation. The computations across the network are expected to finish approximately simultaneously. Timeline 3 is similarly offset from timeline two and should repeat consistently on a 25 ms loop consequent of events in the latter timeline. Each timeline can be executed in parallel with the others. How often can timeline three repeats assess the effective update rate of the beamformer? The offset from Timeline 1 and 3 is the processing latency and propagation delay, which is expected to remain stable. It is shown in Figure 5.1

Chapter 5

IMPLEMENTATION ONTO HARDWARE

5.1 Overview

Two paths for beamformer computation on the target platforms were created and designed. Only FPGA is used in one route. The alternative pathway is a hybrid strategy that does part of the initial computations on an FPGA and the remaining calculations on a CPU. The implementation of the calculations on a real-time system presents several difficulties. To provide the necessary update time, beamformer creation must be fast enough. Because the FPGA employs fixed point math, maintaining precision can be problematic, mainly when hardware resources are scarce and time is intricate. Beamformers that accomplish both effective SNR gain and INR reduction are less susceptible to calculation mistakes than those that accomplish SNR gain.

I must be aware of memory and FPGA consumption since some calculations, such as the cross-correlator, linear solver, and covariance estimator, might be costly for time and resources because the accelerator's input data is extensive. It is over 10000 numbers. Although the scenarios our system is designed to handle are manageable, the system may be under stress due to the necessary algorithm settings because it strives to maintain computation speed and accuracy.

5.2 FPGA Implementation

When exclusively using the FPGA to construct the beamformer, all computations occur within the FPGA, except for capture scheduling and receiving the intramosaic and gold information exchange. It means only capture scheduling, receiving shared data, and beamforming upsampling occur outside of the beamformer computing block when the beamformer is built entirely on the FPGA. Only information exchange and a small amount of the capture schedule are processed on the CPU during this procedure.

Data is resampled to a sampling rate of $1.5B_U$ outside the beamformer constructor at the capture time. Upon loading all required auxiliary variables, the current building cycle computations commence. The beamformer calculation block delivers the filter suitable for the tile, computed on an upsampler after the building process. The bent pipe is then given as the upsampled beamforming filter.

- Modem detects a signal
- Software schedules the FPGA to make a capture (based on the FPGA clock)
- When the capture is made, that data is resampled to $1.5B_U$ and provided to the FPGA block. Assuming all necessary auxiliary variables have been loaded, the computation for the current cycle begins.
- At the end of the computation, the block provides the filter appropriate for the tile it was computed on.
- That beamformer is upsampled and provided to the bent pipe

This section explains in detail the architecture, specifications, and performance of the Digital Beamformer FPGA implementation and the individual computation blocks of the design.

5.2.1 Overview

Figure 5.1 displays a block diagram of all internal calculations used to construct the beamforming filters servicing a single relay direction. This processing block must be instantiated twice to maximize communication in both ways. The computing block diagram comprises three components: the QR decomposition (QRD) based linear solver, data fusion, and local preprocessing. I also include the block's relationship with the architecture surrounding it, which consists of the bent pipe and the front end.

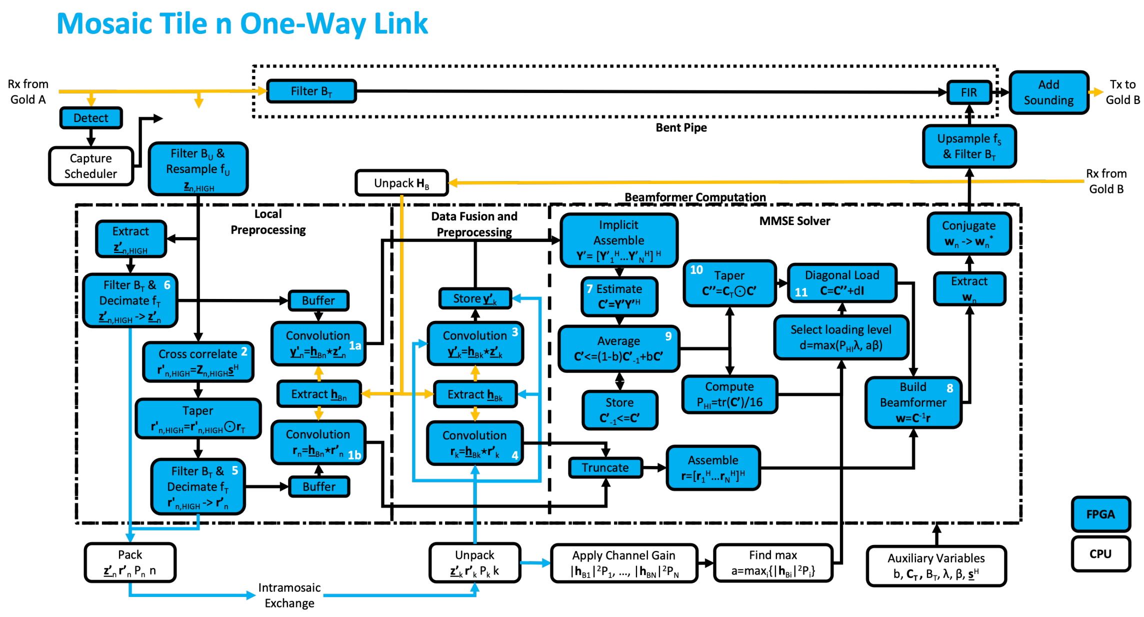

Mosaic Tile n One-Way Link
Rx from Gold A
Filter B_T
Bent Pipe
FIR
Add Sounding
Tx to Gold B
Detect
Capture Scheduler
Filter B_U & Resample f_U $\underline{z}_{n,HIGH}$
Unpack H_B
Upsample f_S & Filter B_T
Rx from Gold B
Beamformer Computation
Local Preprocessing
Data Fusion and Preprocessing
MMSE Solver
Extract $\underline{z}'_{n,HIGH}$
Filter B_T & Decimate f_T $\underline{z}'_{n,HIGH} \to \underline{z}'_n$ 6
Buffer
Convolution $\underline{y}'_n=\underline{h}_{Bn}\star\underline{z}'_n$ 1a
Cross correlate $r'_{n,HIGH}=\underline{z}_{n,HIGH}\underline{s}^H$ 2
Extract $\underline{h}_{Bn}$
Taper $r'_{n,HIGH}=r'_{n,HIGH}\odot r_T$
Convolution $r_n=\underline{h}_{Bn}\star r'_n$ 1b
Filter B_T & Decimate f_T $r'_{n,HIGH} \to r'_n$ 5
Buffer
Store $\underline{y}'_k$
Convolution $\underline{y}'_k=\underline{h}_{Bk}\star\underline{z}'_k$ 3
Extract $\underline{h}_{Bk}$
Convolution $r_k=\underline{h}_{Bk}\star r'_k$ 4
Implicit Assemble $Y'=[Y'^H_1 \dots Y'^H_N]^H$
7 Estimate $C'=Y'Y'^H$
Average $C'<=(1-b)C'_{-1}+bC'$ 9
Store $C'_{-1}<=C'$
Truncate
10 Taper $C''=C_T\odot C'$
Compute $P_{HI}=tr(C')/16$
Assemble $r=[r^H_1 \dots r^H_N]^H$
Diagonal Load 11 $C=C''+dI$
Select loading level $d=max(P_{HI}\lambda, a\beta)$
Build Beamformer $w=C^{-1}r$ 8
Conjugate $w_n \to w_n^*$
Extract w_n
Pack $\underline{z}'_n\, r'_n\, P_n\, n$
Intramosaic Exchange
Unpack $\underline{z}'_k\, r'_k\, P_k\, k$
Apply Channel Gain $|h_{B1}|^2P_1, \dots, |h_{BN}|^2P_N$
Find max $a=max_i\{|h_{Bi}|^2P_i\}$
Auxiliary Variables $b, C_T, B_T, \lambda, \beta, \underline{s}^H$
FPGA
CPU

Figure 5.1: Block Diagram of the Beamformer Computation.

Three-state machines divide and oversee the computation sequence. The calculation blocks displayed in Figure 5.1 are executed in a sequence described by a timeline in Figure 5.2. The three distinct state machines in hardware represent the three more minor sequences that make up the overall chronology.

State machine one sequentially executes the filter-decimation of an excerpt of $\underline{\mathbf{z}}_{n,\mathrm{HIGH}}$, cross-correlation of $\underline{\mathbf{z}}_{n,\mathrm{HIGH}}$ with the training sequence $\underline{\mathbf{s}}$ (complex-conjugating the latter), and then filter-decimation of the cross-correlation $\mathbf{r}'_{n,\mathrm{HIGH}}$ which produces $\mathbf{r}'_n$. Once this sequence is complete, the tile has prepared the information that needs to be exchanged. This state machine also performs the convolutions with channel B, $\underline{\mathbf{h}}_{B,n}$, for the locally generated $\underline{\mathbf{z}}'_n$ and $\mathbf{r}'_n$ vectors.

Variable	Description
`Znhigh`	Local training observation capture
`rnhigh`	Local cross-correlation
`zn`	Local downsampled training observation
`rn`	Local downsampled cross-correlation
`zk`	Distributed downsampled training observation
`rk`	Distributed downsampled cross-correlation
`Y'`	Spatiotemporal data matrix

Table 5.1: Variables and Description

State machine two fuses the data and performs the final data preprocessing steps. The subscript n denotes local measurements, and data from distributed nodes are subscripted by k in Figure 5.1. The remaining preprocessing steps involve convolving $\underline{\mathbf{z}}'_k$ with $\underline{\mathbf{h}}_{B,k}$ and $\mathbf{r}'_k$ with $\underline{\mathbf{h}}_{B,k}$. This state machine progresses when any $\underline{\mathbf{z}}'_k$ and $\mathbf{r}'_k$ are received. State machine two depends on the distributed nodes successfully progressing through state machine one.

State machine three encompasses the covariance matrix computation and the linear solver that produces the beamformers. This state machine sequentially executes matrix-matrix multiplication, covariance matrix averaging, covariance matrix tapering, diagonal loading, and QRD with back-substitution. The result of the QRD with back-substitution is the set of beamforming filters for all tiles. The computation block complex conjures the filter specific to the tile, performing the computation before feeding it to the upsampler. It adapts the result to be used with the bent pipe architecture. This state machine depends on completing state machine one and state machine two.

Motivated by several standard computations performed in the chain, the hard-

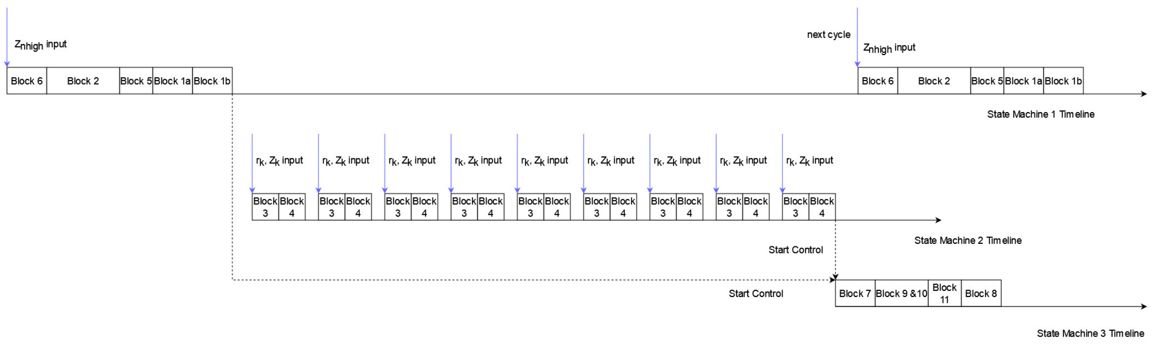


Figure 5.2: The Execution Timeline of the Blocks in Figure 5.1 Devoid of a Time Scale.

ware description language (HDL) is implemented to use standard hardware for both the anti-alias filter and convolution blocks. Inputs and computations are scheduled according to the timeline shown in Figure 5.2 to achieve the desired products. An HDL implementation block diagram showing all computational blocks, their interconnections, their inputs, and their outputs are shown in Figure 5.3.

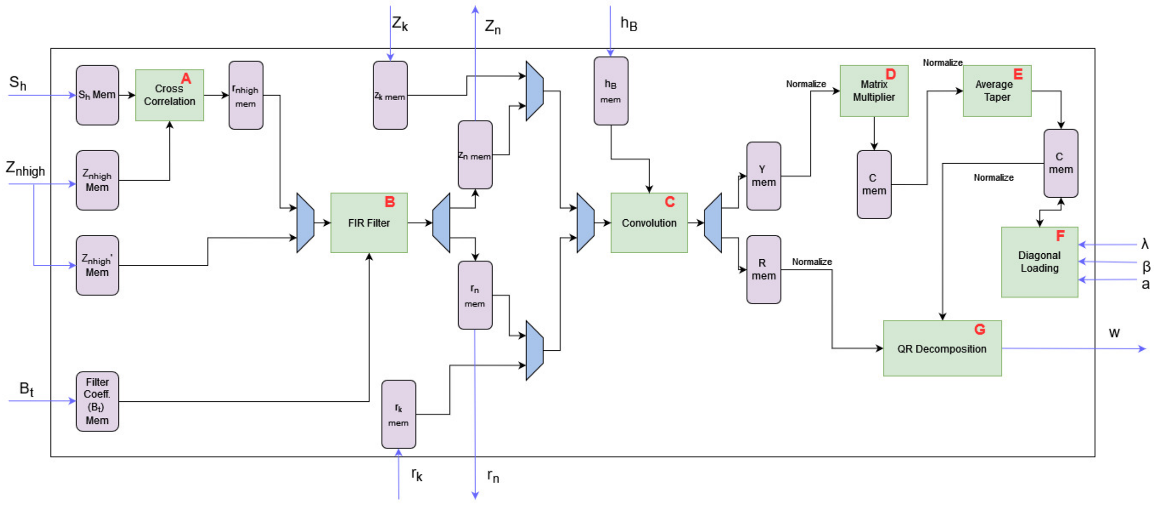


Figure 5.3: Block Diagram of the Digital Beamformer Hdl Implementation.

The mapping of computation blocks from Figure 5.1 to the FPGA hardware blocks shown in Figure 5.3 are listed in Table 5.2.

The input and output precision for all blocks shown in Figure 5.1 are listed in Table 5.3. The number of bits listed is the depth used separately for genuine and

Table 5.2: Computation Block Mapping to Hardware Units.

Hardware Block (Figure 5.3)	**Computation Blocks (Figure 5.1)**
A	2
B	5, 6
C	1a, 1b, 3, 4
D	7
E	9, 10
F	11
G	8

complex parts. For example, block A represents the fundamental part of a sample using 32 bits and the imaginary part using 32 bits. The bit length of the complex number is 64 bits in total. Block G, however, is a case where its input precision is 20 bits for the natural part and 20 bits for the imaginary. Still, the output precision of the natural and imaginary parts is 12 bits each. Different bit depths are used for each block based on the accuracy requirements of each output.

Table 5.3: Internal Precision of Each Block in Figure 5.1.

Hardware Block	**Internal Precision (bits)**
A	32
B	24
C	32
D	24
E	24
F	24
G*	20

The execution times of the computations displayed in Figure 5.1 differ because different computation blocks are implemented using the same hardware unit. Table 5.4 contains a list of these computations' execution times. The output SNRs of each block are also listed. The accuracy is calculated by comparing the FPGA output with the same calculation performed using double precision in MatLab.

Table 5.4: Execution Time for Computations.

Computation Blocks	**Execution Time (μs)**	**SNR (dB)**
2	3400	42.65
6	90	35.63
5	7	45.98
1a, 3	3.3	34
1b, 4	1	37.74
7	2100	40
9 & 10	52	42
11	50	44.21
8	1500	27.23

Resource utilization of the entire design is summarized in Table 5.5. The overall utilization attributed to beamformer construction is double the values in Table 5.5 when accounting for the required second instantiation.

5.2.2 Convolution and Cross-Correlation

Each tile must compute its local cross-correlation estimate $\mathbf{r}'_{n,\mathrm{HIGH}}$ and a series of convolutions between $\underline{\mathbf{h}}_{B,k}$ and $\mathbf{r}'_k$ as well as between $\underline{\mathbf{h}}_{B,k}$ and $\mathbf{z}'_k$, according to Figure 5.1 notation. A time reversal and a complex conjugation relate cross-correlations and convolutions. Consequently, the same architecture is used for both tasks. Two

Table 5.5: Utilization Report for Entire HDL Design.

Resource	**Utilization**	**Available**	**Utilization (%)**
LUT	83748	425280	19.69
FF	43943	850560	5.17
BRAM	175.5	1080	16.25
URAM	12	80	15
DSP	918	4272	21.49

instantiations are used to enable parallel execution of both computations. The cross-correlator invokes many operations, making this additional utilization necessary for minimizing scheduling conflicts. One instance implements the cross-correlation, and the other implements the convolution. The cross-correlation computation is triggered within state machine 1, whereas most convolutions occur within state machine two while tiles exchange information.

Figure 5.4 shows a high-level diagram describing the architecture. The convolution controller coordinates inputs for a 4×1 vector-vector multiplication along with corresponding partial outputs to iteratively obtain the results of cross-correlating or convolving sequences much longer than four-element vectors. The architecture of the 4×1 vector-vector multiplication systolic array is shown in Figure 5.5.

Table 5.6 lists the resource utilization of the convolution or cross-correlation blocks. The resource usage is the same because the computation procedure is the same for both despite consuming different length inputs and producing different length outputs.

Narayana demonstrates in Figure 5.6 the accuracy of the HDL implementation of the cross-correlation computation by comparing its result with that of MATLAB. The real part is compared in the top panel, and the imaginary part is shown in the bottom

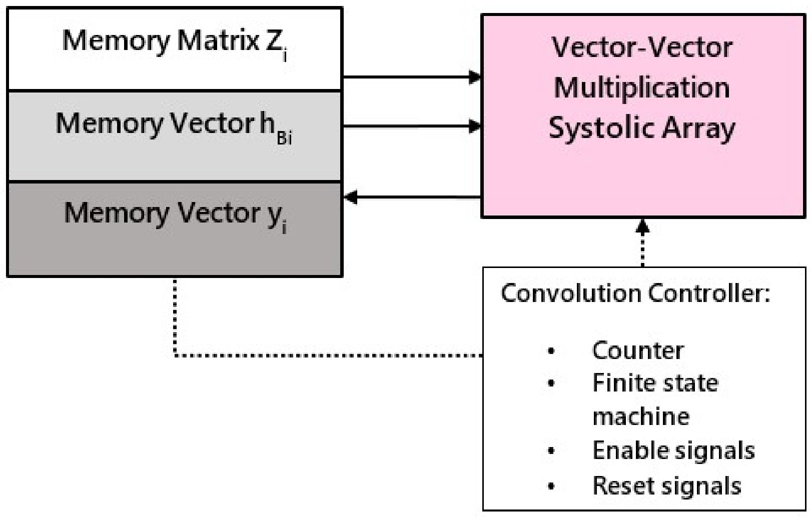


Figure 5.4: Block Diagram of Cross-correlation and Convolution Units.

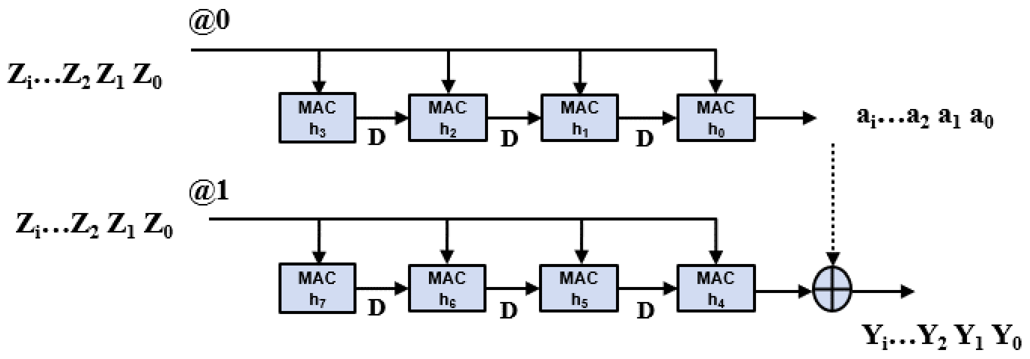


Figure 5.5: 4×1 Systolic Array Used in the Convolution and Cross-correlation Blocks.

panel. The overall accuracy in the example shown is 42.6546 dB when considering MATLAB's result as the gold standard. The results are virtually indistinguishable.

Similarly, I demonstrate in Figure 5.7 the accuracy of the HDL implementation of the convolution computation relative to MATLAB's product. In this example, it produces a $\underline{\mathbf{y}}'_n$ by convolving $\underline{\mathbf{z}}'_n$ with $\underline{\mathbf{h}}_{B,n}$. This data vector is partly used to estimate the covariance matrix used in the MMSE form. The overall accuracy in this case is 42.5776 dB compared to MATLAB's result. Note that this accuracy measurement is

Table 5.6: Utilization Report for Convolution and Cross-Correlation Blocks.

Resource	Utilization	Available	Utilization (%)
LUT	1026	425280	0.24
FF	1099	850560	0.13
DSP	18	4272	0.42

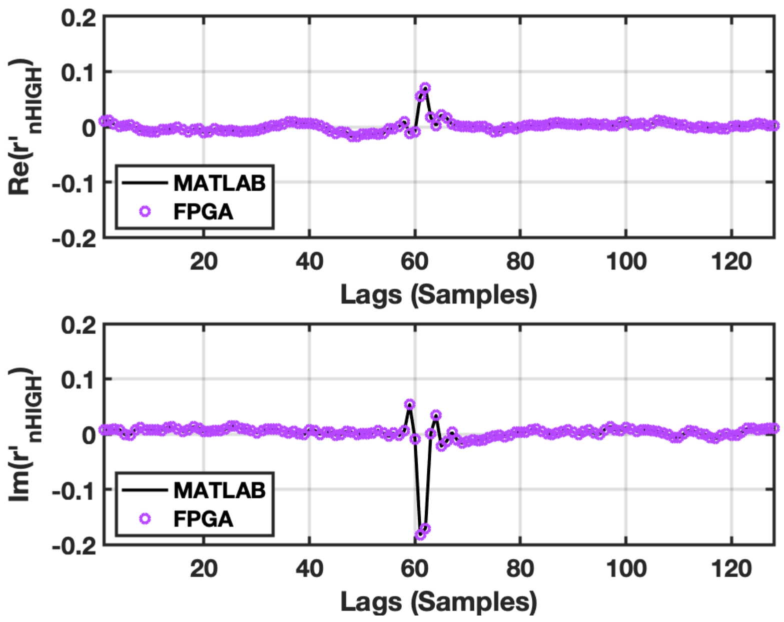


Figure 5.6: Cross-correlation Output Comparison Between Matlab and FPGA.

not related to the accuracy of the channel B estimate.

The result is the full-length convolution with $\underline{\mathbf{h}}_{B,n}$. This method of computing the result results in a cross-correlation longer than the 16 taps specified. To correctly center the channel replies into the appropriate time window, I need to consider the filter delay in the cross-correlation.

The 4th peak is where the initial arrivals in channel B and the downsampled cross-correlation appear by design. For this reason, the delay that channel B adds to the cross-correlation is predictable. To ensure that the correct filter length is maintained and all detections show inside the relevant analysis window, I read 16

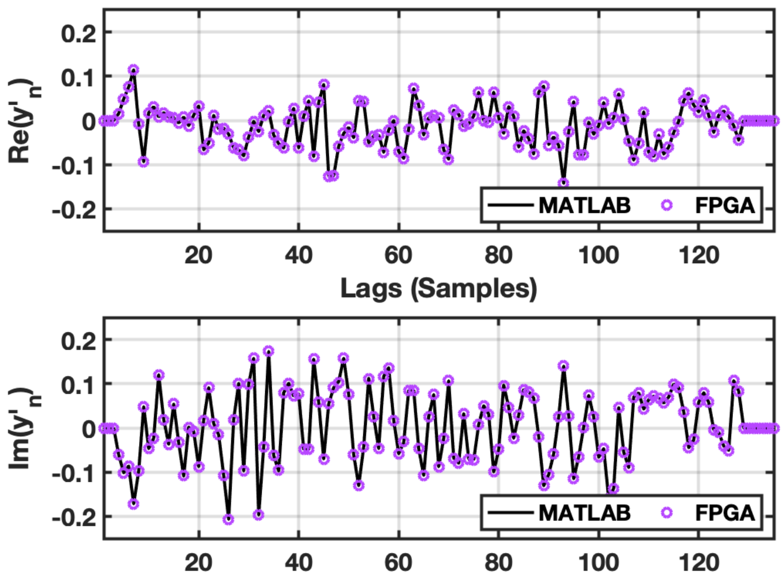


Figure 5.7: Convolution Output Comparison Between Matlab and Fpga for One Channel B and Downsampled Training Observation Vector.

samples, beginning with the third. Each of the $\underline{\mathbf{y}}'_k$ vectors produced is a 1×135 sample. Since $\underline{\mathbf{y}}'_k$ vectors are solely utilized to estimate the covariance matrix, which describes relationships as a function of relative elements and time differences, truncation is not required.

The downsampled cross-correlation vectors are 16 samples a piece. doing a full convolution typically results in a longer sequence. In this case, it's 26 sample vectors. it is more crucial to truncate in this case to maintain the correct filter lengths. By design, the first peak in channel B is designed to occur at the 4th peak due to the resampler's sinc structure, so delay compensation is included to remove the delay. The result is a 16-sample vector. The result must still be 16 samples.

5.2.3 Downsampling

The system uses two downsampling blocks to change the cross-correlation vectors and train observation sampling rates. $\underline{\mathbf{z}}'_n$ is produced by this block when $\underline{\mathbf{z}}'_{n,\mathrm{HIGH}}$ is given. In contrast, $\mathbf{r}'_{n,\mathrm{HIGH}}$ yields $\mathbf{r}'_n$. This block simultaneously carries out the two tasks of FIR filtering and decimation, avoiding calculations that result in samples that will be eliminated during the decimation process.

A systolic array structure is used in these blocks to reduce FPGA utilization and to adapt to various input data lengths. Both downsampling blocks implement the exact sampling rate conversion: f_U to $f_U/8$.

Figure 5.8 illustrates the computational architecture for this block. An example of downsampling the cross-correlation computed at $1.5B_U$, $\mathbf{r}'_{n,\mathrm{HIGH}}$, is shown in this image. A comparable high-level diagram summarizing the downsampling block's data flow and control flow is displayed in Figure 5.9. The block takes in four consecutive inputs at a time, each of which feeds a multiplier with an anti-alias filter coefficient. The outcomes are totaled. The block then cycles over the whole filter and data vector, adding the sum of products with $\mathbf{r}'_n[t]$, a matching previously calculated intermediate result. Through the $\mathbf{r}'_n[t]$ port, the relevant intermediate results that have been saved are read out and supplied. The controller controls data transmission into and out of the block to guarantee that the folded computation yields accurate outputs. Every cycle, the data kept in the memory banks is refreshed. Each memory bank receives the outputs via a demultiplexer (DMUX).

The final output is the accumulation of all $\mathbf{r}'_i$ from each cycle. It contains four multipliers, eight adders, and one demultiplexer (DMUX). There are four memory banks to store the input data. So, there are four input ports in the filter array as well. The same is true for the output data. There is only one output port in the filtering and downsampling array, so a MUX is applied to distribute the output into four memory banks.

Table 5.7 summarizes the downsampling block use report as calculated by Vivado. This block performs one of the less expensive calculations compared to other com-

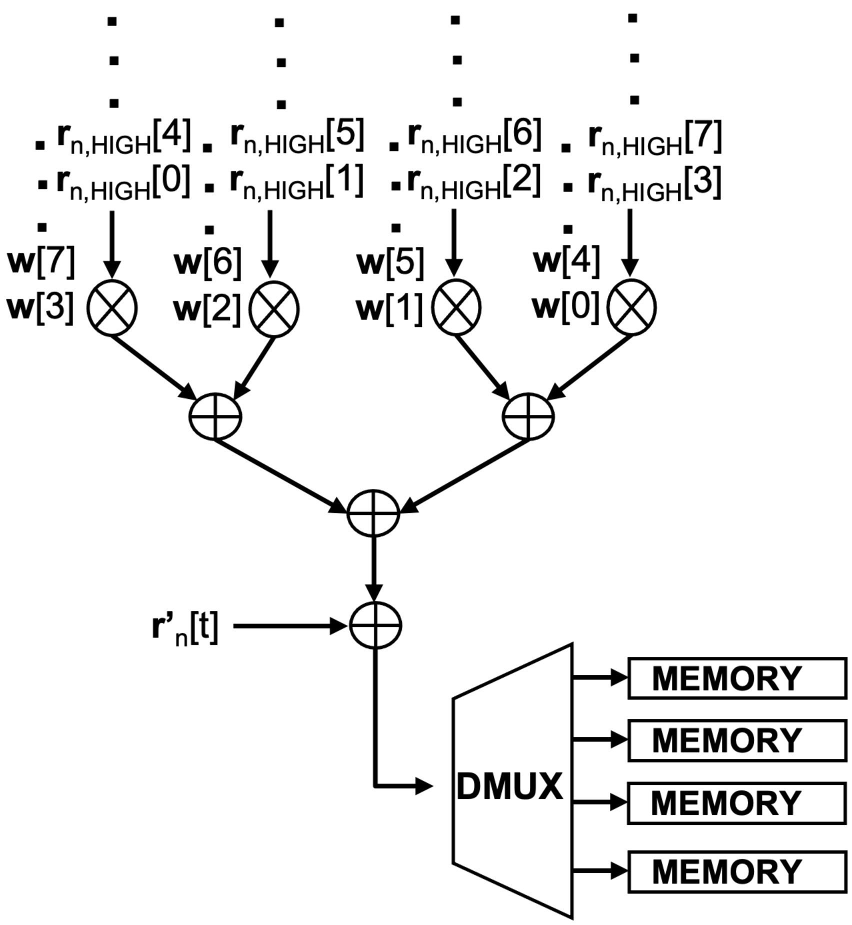


Figure 5.8: Block Diagram of Filter and Decimate Block.

putations. Because fewer operations are required than for different blocks, usage is often negligible, and the calculation usually finishes rather rapidly.

I demonstrate that the HDL implementation of the downsampling block yields acceptable accuracy in Figure 5.10. Compared to MATLAB's result, the FPGA outputs a result with 42.0105 dB SNR.

5.2.4 Matrix-Matrix Multiplication

All ten tiles' channel B convolution outputs are aggregated to form a matrix Y_i of size 10×135. State machine two produces 10 $\underline{\mathbf{y}}'_n$ vectors, which then triggers the start of state machine three. From this collection of vectors, the computation block can

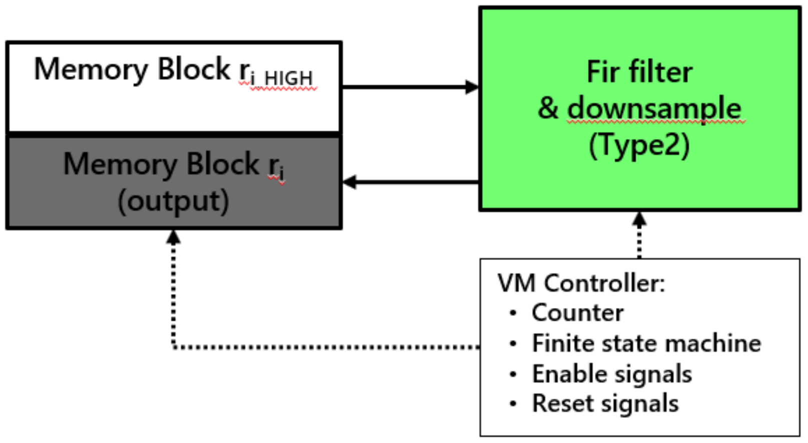


Figure 5.9: Data Flow of Downsampling Block.

Table 5.7: Utilization Report for Filtering and Downsampling Block.

Resource	**Utilization**
LUT	846
FF	828
CARRY	45
DSP	32

now estimate the spatiotemporal covariance matrix required to mitigate interference. This operation is the first in the third state machine. They together form the data matrix Y.

Narayanan notes the two properties regarding the inputs of this particular matrix-matrix multiplication. First, the covariance matrix is constructed by multiplying the spatiotemporal data matrix $\mathbf{Y}$ with its hermitian transpose. Second, the spatiotemporal data matrix $\mathbf{Y}$ ($\tilde{\mathbf{Y}}$ in Equation 4.13) of size 160×150 is generated by creating 16 sample shifted versions of each 1×135 vector $\underline{\mathbf{y}}'_n$. Narayanan uses these two key features and optimizes the matrix-matrix multiplier by adopting two strategies. These two fundamental relations are used to design efficiently. Matrix Multiplier by

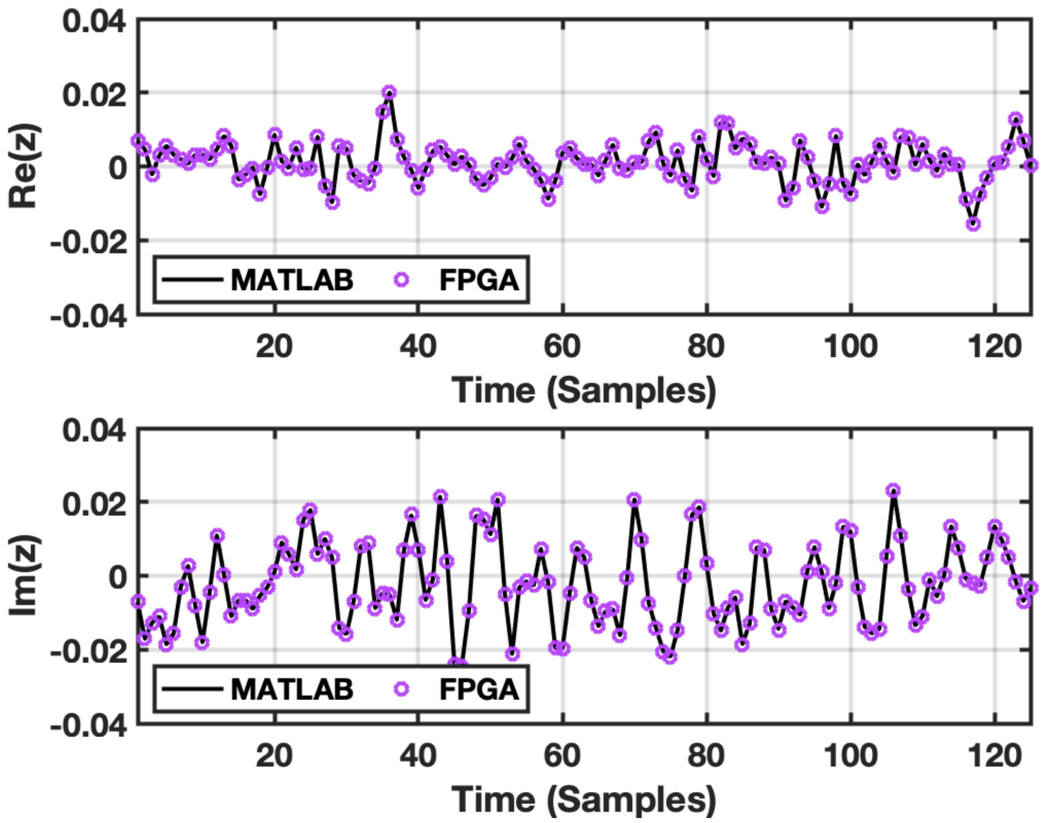


Figure 5.10: Comparison of Downsampled Training Observations Resulting from the FPGA Simulation and Matlab.

adopting the following two strategies. Spatiotemporal data matrix $\mathbf{Y}$ is only implicitly generated from the 10 $\underline{\mathbf{y}}'_n$ vectors on the fly during the computation, thereby reducing memory requirements to store the input matrix by 94.37%. The spatiotemporal data matrix is constructed from sample shifts, so the exact computation can be achieved by simply readdressing the inputs. Memory requirements are further reduced by also calculating the hermitian conjugate $\mathbf{Y}^{\mathrm{H}}$ on the fly without physically storing it. Forming the conjugate only requires a sign flip for the imaginary part. Only a 10×135 matrix needs to be physically stored. Furthermore, these data savings do not incur a significant computational cost.

Figure 5.11 displays the matrix multiplication unit's block diagram. A folded

4×4-systolic multiply-accumulate (MAC) array divides the large matrix-matrix multiplication into smaller 4x4 matrix-matrix multiplications. All partial outcomes from the past are collected with the partial products that are kept in memory. The controller schedules the input and partial outputs from the previous cycle to be sent to the systolic array.

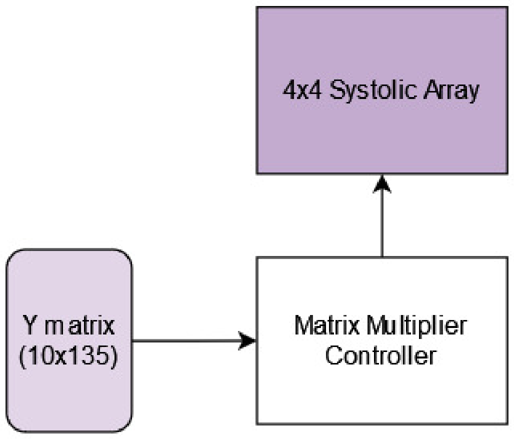


Figure 5.11: Matrix Multiplication Block diagram.

In Figure 5.12, the systolic array utilized in the matrix-matrix multiplication unit is displayed. Intermediate product totals drop in a staggered pattern from the top. The cells receive data. Spatial-temporal covariance matrices are obtained via staggered input of complex-conjugated data from the left.

The matrix multiplication accelerator works correctly in this stage, but this design takes a considerable memory block size to store the input matrix data. Narayanan Murugan worked on optimizing the matrix multiplication accelerator to decrease resource utilization. I also included this part to complete the whole accelerator design. The following paragraphs explain how he did the optimization.

Since the space-delay data matrix $\mathbf{Y}$ is generated by appending together time-shifted versions of the 10×135 data matrix, I create 4×4 sub-matrices of $\mathbf{Y}$ by

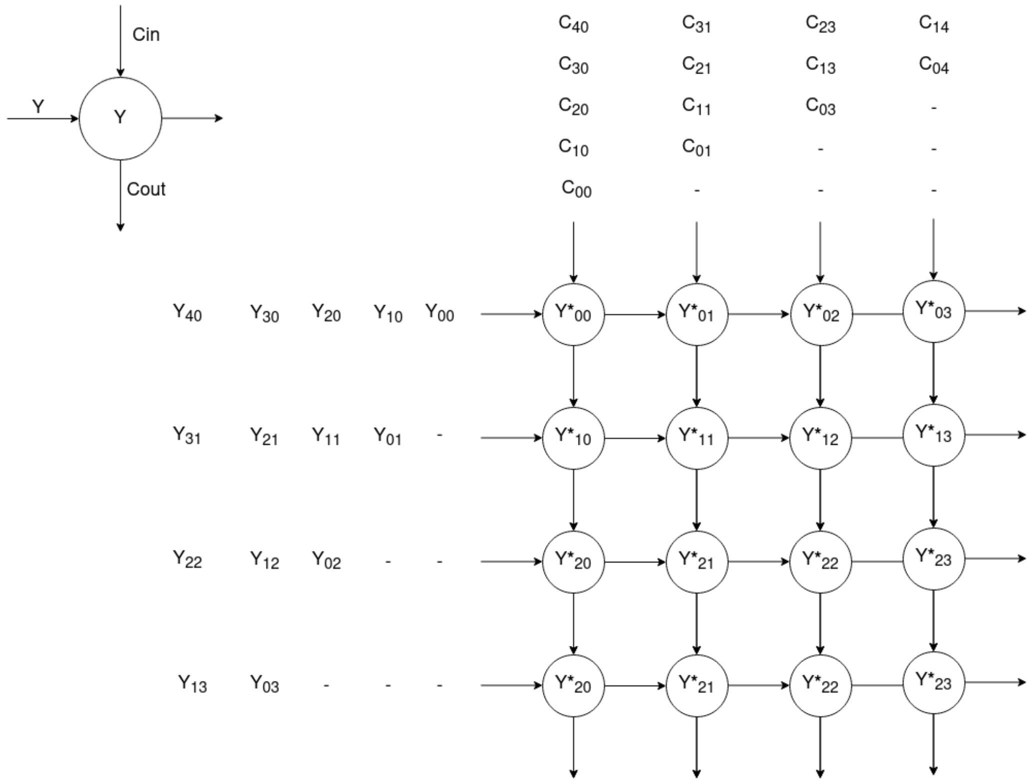


Figure 5.12: 4×4 Systolic Array Matrix Multiplier.

passing each $\underline{\mathbf{y}}'_n$ vector through a shift register. Every four shifts produces a new 4×4 sub-matrix. I demonstrate this process with the following example.

Consider an initial 1×4 vector $\underline{\mathbf{y}}'$,

$$\underline{\mathbf{y}}' = \begin{bmatrix} y_0 & y_1 & y_2 & y_3 \end{bmatrix}, \tag{5.1}$$

which is to be converted to a 4×7 matrix,

$$\mathbf{Y} = \begin{bmatrix} y_0 & y_1 & y_2 & y_3 & 0 & 0 & 0 \\ 0 & y_0 & y_1 & y_2 & y_3 & 0 & 0 \\ 0 & 0 & y_0 & y_1 & y_2 & y_3 & 0 \\ 0 & 0 & 0 & y_0 & y_1 & y_2 & y_3 \end{bmatrix}, \tag{5.2}$$

through 3 sequential shifts from the 0^{th} delay. Narayanan does this conversion in hardware by maintaining a shift register of depth 7. The state of the shift register over seven cycles is shown in Table 5.8.

Table 5.8: State of the Shift Registers after Every Cycle.

Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**
R0	y_0	y_1	y_2	y_3	0	0	0
R1	0	y_0	y_1	y_2	y_3	0	0
R2	0	0	y_0	y_1	y_2	y_3	0
R3	0	0	0	y_0	y_1	y_2	y_3
R4	0	0	0	0	y_0	y_1	y_2
R5	0	0	0	0	0	y_0	y_1
R6	0	0	0	0	0	0	y_0

For a particular cycle, parsing out subsets of the register's contents into 4×4 blocks, structured as

$$\begin{bmatrix} R3 & R2 & R1 & R0 \\ R4 & R3 & R2 & R1 \\ R5 & R4 & R3 & R2 \\ R6 & R5 & R4 & R3 \end{bmatrix}, \tag{5.3}$$

yields the appropriate delay structure in the input to produce the spatiotemporal covariance matrix. The example here, shown for the smaller 1×4 vector, can be scaled to the longer $\underline{\mathbf{y}}'_n$ vectors used in the actual implementation.

To ensure that the block can manage the 16 delays, a shift register with 20 slots is employed explicitly for our system. The shift register is where the longer 1×135 $\underline{\mathbf{y}}'_n$ vectors travel through, producing a new 4×4 block with each shift. The controller arranges for the appropriate sub-blocks to multiply in tandem.

Resource utilization of the matrix-matrix multiplication block is listed in Table 5.9. Despite the number of operations involved in computing a 160×160 covariance matrix from a spatiotemporal data matrix with dimensions 160×150, the utilization is relatively small because of the systolic array architecture.

Table 5.9: Resource Utilization of the Matrix-matrix Multiplication Block.

Resource	Utilization	Available	Utilization (%)
LUT	8644	425280	2.03
FF	7746	850560	0.91
DSP	65	4272	1.52

After Narayanan's optimization, I compared the matrix-matrix multiplication outputs from MATLAB and the FPGA implementation in Figure 5.13. In this example, the FPGA design achieves an overall SNR of 41.7601 dB relative to the MATLAB result.

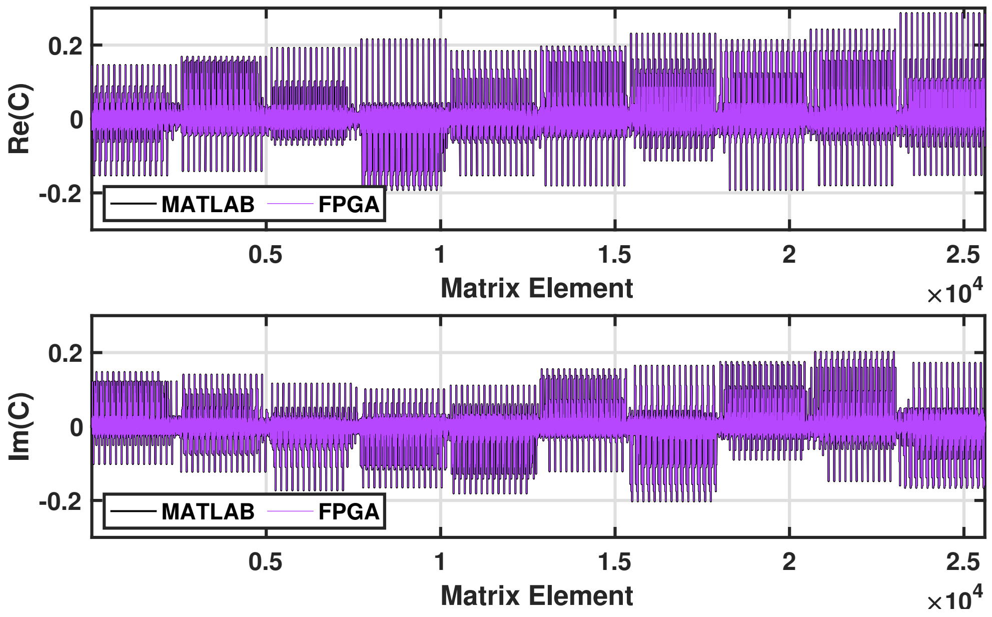


Figure 5.13: Matrix Multiplication Output Comparison Between Matlab and FPGA.

More research has been done on the optimization of the matrix multiplication. The output matrix is a hermitian matrix. I can take advantage of this property and further decrease the total latency and memory usage. Only the upper triangle of the output matrix is needed for the final output, so I only need to calculate and store the upper triangle of the matrix in the memory. The rest part is conjugate of the upper triangle part. I finished the simulation of this algorithm and have not integrated this design yet. It will be the future work.

5.2.5 Average and Taper

The first step in the covariance matrix postprocessing is averaging and tapering. The two operations are in the same group and conveniently fit into one hardware block.

Figure 5.14 illustrates the construction of the average-taper block. Since averaging and tapering are element-wise processes, they may be completed in parallel efficiently. Four subsets of the matrices are processed in parallel by four identical computer units. As seen in Figure 5.14, each computing unit initially averages as a weighted sum between the prior average and the present estimate ($\mathbf{C}'_{-1}$ and $\mathbf{C}'$, respectively). The averaged result is supplied to tapering and returned to memory storage. As seen in Figure 5.14, tapering is achieved by multiplying the updated average by a pre-computed coefficient in the matrix $\mathbf{C}_T$.

The data flow of the averaging-tapering block is outlined in Figure 5.15. Several memory banks are initialized to store the previous averaged covariance matrix $\mathbf{C}'_{-1}$, the current epoch's estimate $\mathbf{C}'$, and the tapering matrix $\mathbf{C}_T$. The controller transfers data into and out of the memory banks accordingly.

5.2.6 Diagonal Loading

I implement in hardware the diagonal loading strategy as explained described in Equation 4.19, Equation4.20, and Equation 4.21.

Using the obtained covariance matrix, the first loading level option, d_1 in Equation 4.19, is directly determined. However, the second loading level choice d_2 is calculated

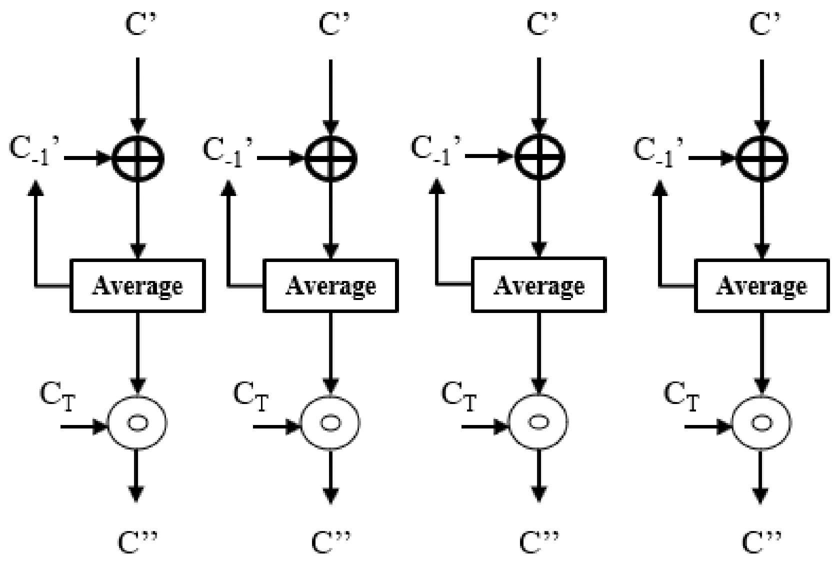


Figure 5.14: Block Diagram of the Average-Taper Block.

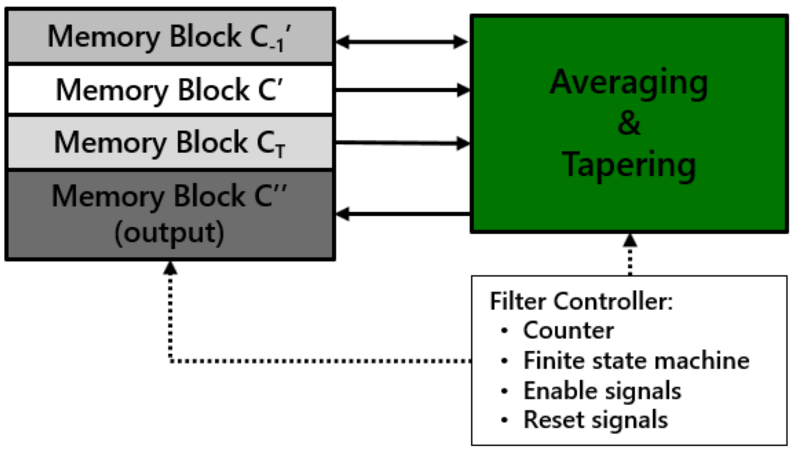


Figure 5.15: Data Flow of the Average Taper Block.

using a noise floor measurement that has been calibrated. Since the covariance matrix is dynamically scaled, d_2 needs to be normalized for it to roughly represent the noise power in the covariance matrix as d_2/β (or a in Figure 5.3). d_2 is transformed into the same binary format as the covariance matrix elements in the calculation block. The covariance matrix's diagonal is increased by the amount obtained by computing the maximum of the two possibilities.

5.2.7 QR Decomposition with Back-Substitution

The last major step in the beamformer computation is implementing a linear solver that produces the beamformer. The linear system that must be solved is

$$\mathbf{C}\mathbf{w} = \mathbf{r}\,, \tag{5.4}$$

where $\mathbf{C}$ is a fully post-processed covariance matrix, $\mathbf{r}$ is a downsampled cross-correlation convolved with channel B, and $\mathbf{w}$ contains all 10 beamforming filters. My approach to obtaining the solution involves a QR decomposition (QRD) with a back-substitution algorithm. Srimayee designed a folded, scalable QRD with a back-substitution block to implement this operation on a resource-constrained system.

The hardware implementation architecture of the QRD and back-substitution accelerator was borrowed from Srimayee's thesis[5]. I implemented the folded architecture from [5] on FPGA. The following paragraph explains the algorithm roughly and its structure. However, there are limitations to this design. One of them is that the setup frequency is only 50 MHz, and I changed the design and ran it with a 140 MHz clock frequency. Another change is about the reciprocal operation. The rotation method was employed in [5], and I used lookup tables instead. Because the lookup tables can provide higher precision and produce fewer clock cycles, all the changes are also described in this section.

A QRD is used to facilitate the matrix inversion. This procedure decomposes a complex-valued matrix $\mathbf{A}$ into

$$\mathbf{A} = \mathbf{QR} = \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \dots & \mathbf{q}_N \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & . & . & . & r_{1N} \\ 0 & r_{22} & . & . & . & r_{2N} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & . & . & . & r_{NN} \end{bmatrix}, \tag{5.5}$$

where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is an upper triangular matrix. When leveraging this decomposition, the system is solved in the following way:

$$\mathbf{Ax} = \mathbf{b} \tag{5.6}$$

$$(\mathbf{QR})\,\mathbf{x} = \mathbf{b} \tag{5.7}$$

$$\mathbf{Rx} = \mathbf{Q}^{-1}\mathbf{b} \tag{5.8}$$

$$\mathbf{Rx} = \mathbf{b}'\,. \tag{5.9}$$

Inverting $\mathbf{Q}$ is simple because $\mathbf{Q}^{-1} = \mathbf{Q}^{\mathrm{H}}$. In this application, Equation5.4 corresponds with Equation 5.6. When analyzing the structure of Equation 5.9,

$$\begin{bmatrix} r_{11} & r_{12} & . & . & . & r_{1N} \\ 0 & r_{22} & . & . & . & r_{2N} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & . & . & . & r_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b'_N \end{bmatrix}, \tag{5.10}$$

she noticed that the solution of the resultant linear system can be easily obtained by solving for individual elements of $\mathbf{x}$ in a bottom-up approach given by the following sequence:

$$\begin{aligned} r_{n,n}x_n &= b'_n \\ r_{n-1,n-1}x_{n-1} + r_{n-1,n}x_n &= b'_{n-1} \end{aligned}\,. \tag{5.11}$$

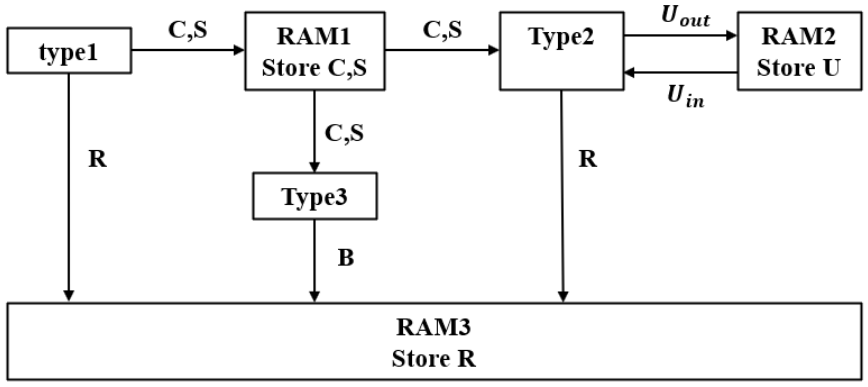


Figure 5.16: Block Diagram of the Scalable QR Accelerator [5].

The QRD block directly computes $\mathbf{Rx} = \mathbf{Q}^{-1}\mathbf{b}$ by applying Givens rotations to either side of the equation. The orthogonal matrix is never explicitly formed nor stored. She uses a folded architecture to compute the QRD of a large-sized matrix.

Supporting the folded architecture are three blocks. These blocks are linked to three memory blocks and each other, as Figure 5.16 illustrates. Each block is made up of two different kinds of programming elements: internal cells and border cells. Block type I is triangular, with four border cells and six interior cells, a structure akin to matrix $\mathbf{R}$. Type II has sixteen internal cells and is rectangular. Type III, which has four internal cells, is a column vector.

Figure 5.17 illustrates how the systolic array is constructed utilizing boundary and interior cells. To illustrate how the cells interact, Srimayee specifically shows an example of decomposing a 4×4 matrix $\mathbf{A}$ while changing the vector $\mathbf{b}$. A type I block and a type III block are adequate. Squares stand in for interior cells and circles for border cells. Figure 5.17 also shows input scheduling. The covariance matrix's columns cascade across one another in an uneven pattern. Type II blocks are utilized especially when breaking down more significant matrices. These blocks effectively represent the remaining off-diagonal, higher triangular components by being put between type I and type III blocks.

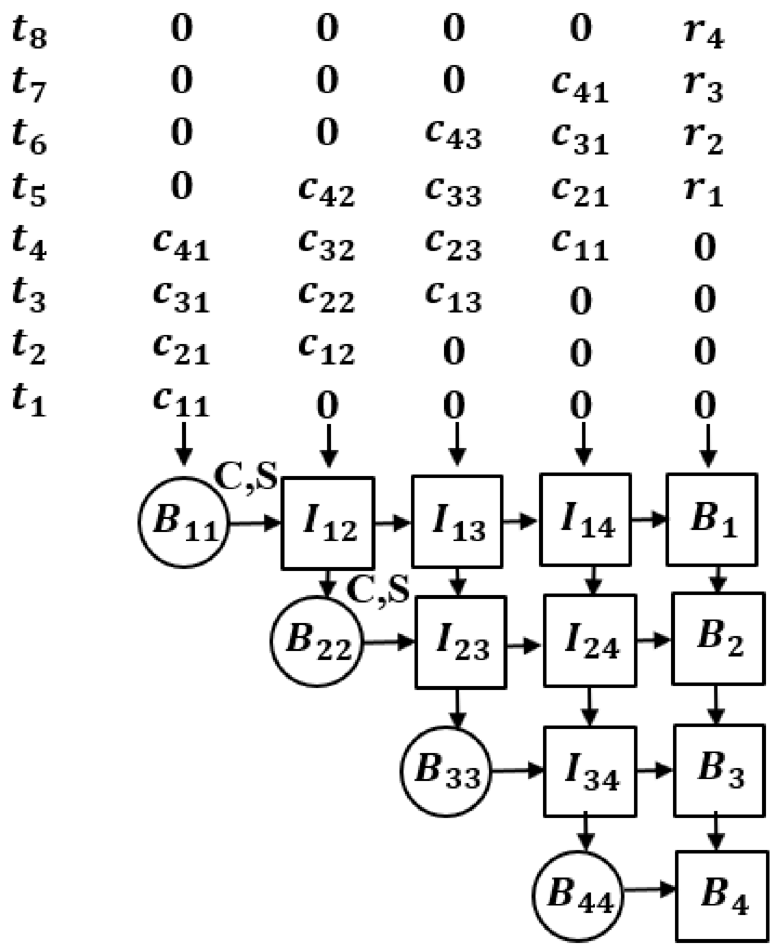

t_8 0 0 0 0 r_4
t_7 0 0 0 c_{41} r_3
t_6 0 0 c_{43} c_{31} r_2
t_5 0 c_{42} c_{33} c_{21} r_1
t_4 c_{41} c_{32} c_{23} c_{11} 0
t_3 c_{31} c_{22} c_{13} 0 0
t_2 c_{21} c_{12} 0 0 0
t_1 c_{11} 0 0 0 0
C,S
B_{11}
I_{12}
I_{13}
I_{14}
B_1
C,S
B_{22}
I_{23}
I_{24}
B_2
B_{33}
I_{34}
B_3
B_{44}
B_4

Figure 5.17: Data Flow in QR Using Systolic Array Architecture [5].

She summarizes the data flow of boundary cells in Figure 5.18. Algorithm 1 executes within these cells. Boundary cells receive a complex number U_{in} as input. The cell computes a Givens rotation parameterized by C, an actual number, and S, a complex number. Lastly, the cell computes the result of the rotation and stores it in memory for the next computation iteration.

Internal cells represent the off-diagonal elements of the upper triangular matrix **R** in the triangular systolic array. She summarizes the data flow in and out of

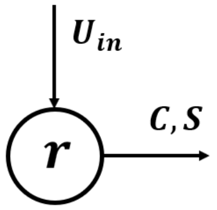


Figure 5.18: Boundary Cell Flow Diagram [5].

Algorithm 1 Boundary Cell Computation.

```
if $U_{in} \leftarrow 0$ then
    $C \leftarrow 1;\ S \leftarrow 0;\ R \leftarrow R'$
else
    $R' \leftarrow \sqrt{R^2 + |U_{in}|^2}$
    $C \leftarrow R/R'$
    $S \leftarrow U_{in}/R'$
    $R \leftarrow R'$
end if
```

internal cells Figure 5.19. Algorithm 2 executes within internal cells, which applies the Givens rotation computed in the corresponding boundary cell. Similarly, an updated upper triangular matrix element value is stored for future computations.

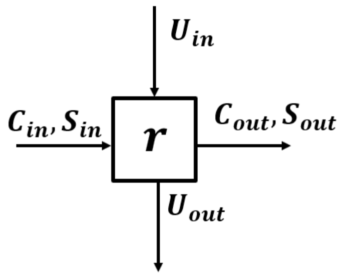


Figure 5.19: Internal Cell Flow Diagram [5].

Once the QRD operation is completed, the back-substitution procedure begins. The back-substitution step reuses the QRD structure consisting of boundary and internal

Algorithm 2 Inner Cell Computation.

```
$U_{out} \leftarrow CU_{in} - SR$
$R' \leftarrow S * U_{in} + CR$
```

cells to produce the solution **x** of Equation 5.6. In other words, the back-substitution sequence produces the beamformer. A control signal is used to switch between the forward and backward computations in the cell. Data flow for back-substitution, depicted in Figure 5.20, flows in the opposite direction of the QRD step.

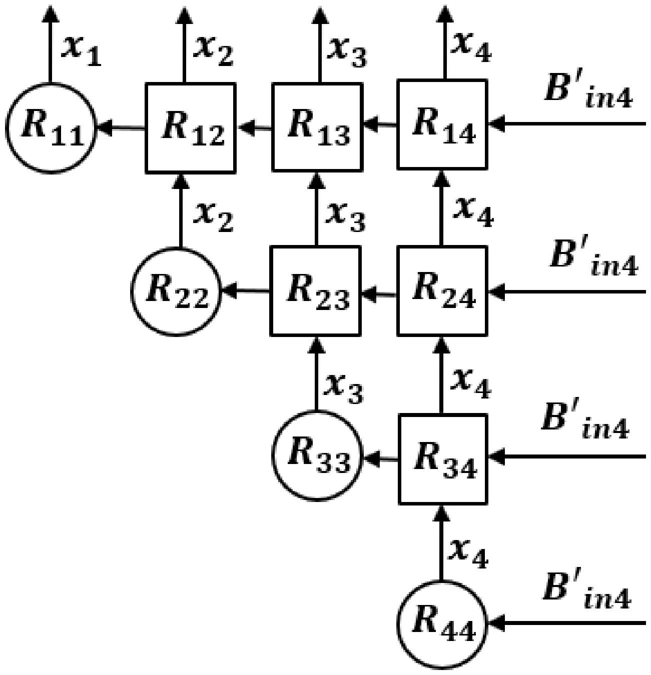


Figure 5.20: Data Flow in Back-substitution [5].

To simplify the folding calculation, the block types are merged as seen in Figure 5.21. An enable signal signals the start of the backwards calculation once the forward path provides B'_N. From random access memory (RAM) III, the relevant $R_{N,N}$ value is read and supplied as input to the type I block's last boundary cell. All other internal cells in the same column receive the X_N value calculated in the border cell. The R values from RAM III and the B' values from the type III block are also supplied concurrently to calculate intermediate values. The type I block's border cell to calculate X_{n-1}. The computed X values are kept in RAM IV to be transferred to the type II block, where the input matrix size is $N \times N$, to calculate the intermediate values for the remaining $N - 4$ rows.

So, the basic structure of the QRD and back-substitution block is described above. As I mentioned at the beginning of this paragraph, there are other changes I made to this design to make sure it can work under 140MHz clock frequency and have the required number of clock cycles to finish. After implementing this folded architecture and synthesis of the design, the 140 MHz frequency requirement is from other parts of the system. There was a setup time violation and lower precision than expected.

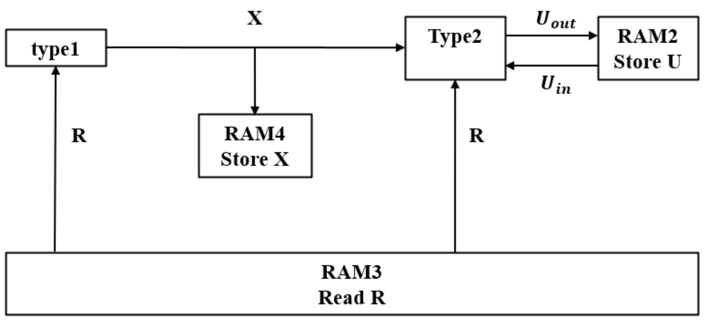


Figure 5.21: Block Diagram of Back-substitution Accelerator [5].

To solve the setup time violation, I used the Vivado program to find the critical path that caused the problem and extracted the circuit to do analysis. The critical path is inside the internal cell in the QRD process. The original design is shown in Figure 5.22. And it results in 8 ns on the critical path. This critical path caused the setup time violation. This means that this design does not fit the 140 MHz clock frequency. Then, I designed it in parallel and made it two cycles. The circuit is shown in Figure 5.23. After this parallel design, the critical path changed to 6.93 ns, which meets the 140 MHz clock frequency requirement. The setup time problems are now solved, but not yet since the number of clock cycles for the internal cell. The block scheduler must also be changed to ensure the data flow is still correct. Both the QRD and back-substitution design need to be modified. Here is an example of the back-substitution data flow. The original data flow design is shown in Figure 5.24. And the new design is demonstrated in Figure 5.25. Since the data flow has changed, the overall latency has also changed. After calculating the total latency, it is 100 μs longer than the original design, but it is acceptable and does not affect other accelerators. Narayanan helped with the rescheduling design as well.

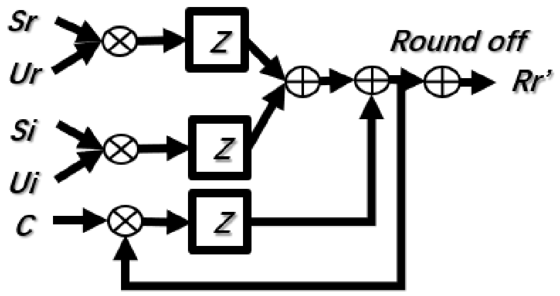


Figure 5.22: Original Design of Internal Cells in QRD Process [5]

Another problem is about the square root reciprocal operation. I implemented the lookup table, but it took a considerable memory block. It is over the requirement of resource utilization. Narayanan helped to design a new strategy that uses two small lookup tables to complete the calculation. This strategy results in similar precision and much less resource utilization. In this way, the new lookup table design satisfies the timing requirement and precision requirements at the same time.

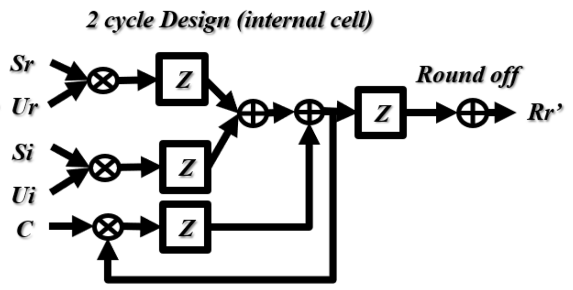


Figure 5.23: parallel Design of Internal Cells in the QRD Process.

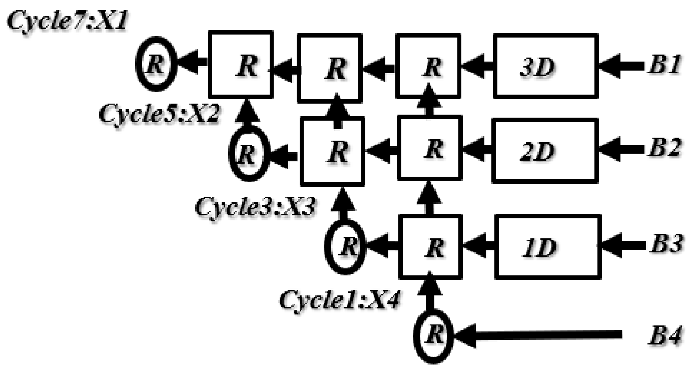


Figure 5.24: Original Back-substitution Process Data Flow [5].

By comparing the generated beamformers in Figure 5.26, I show the precision of the FPGA calculation in comparison to a floating point computation. In this case, the final output's SNR is 35.4394 dB. Errors will likely worsen since mosaic noise-dominated scenarios typically produce poorly conditioned covariance matrices. Cases where gold noise predominates typically produce higher conditioned covariance matrices and computational accuracy closer to 30 dB.

The timing report from Vivado, Figure 5.27, shows that the critical path length is 5.848 ns. I document the FPGA utilization in Table 5.10. The output of the entire QRD with the back-substitution process is sensitive to internal precision. Consequently, extensive look-up tables (LUTs) are used to achieve the required accuracy. The reports are based on the XCZU43DR-FFVE1156-1LV-i part from Xilinx.

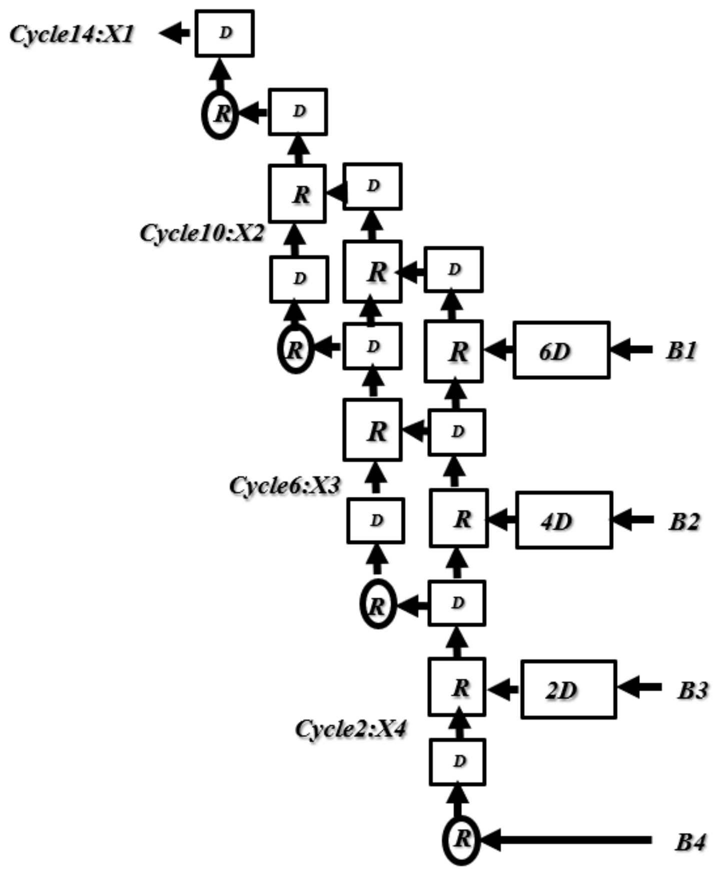


Figure 5.25: Modified Back-substitution Process Data Flow.

5.2.8 Normalization Strategy

Even for near-zero magnitudes, clipping is risky when using a restricted bit depth in calculations. Preserving computational correctness requires normalizing intermediate outcomes. I applied the normalization by the beamformer calculation at the points where the block diagram in Figure 5.3 is designated. I employ a normalization technique that exclusively uses binary shifts to avoid requiring division operations, adding delay, and increasing the amount of FPGA utilized. The relative scaling between the data vectors must not be disturbed by mosaic tiles during normalization. Before the covariance averaging, QR decomposition blocks, and matrix-matrix

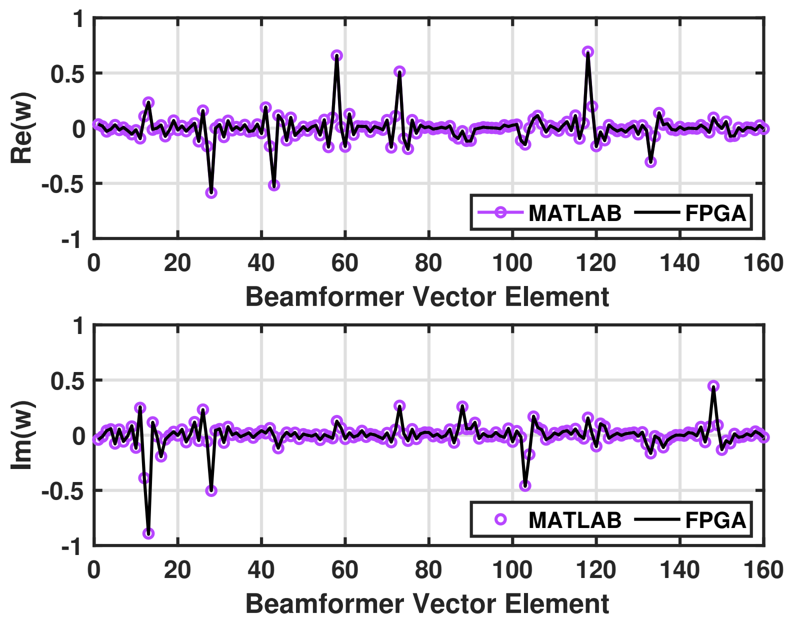
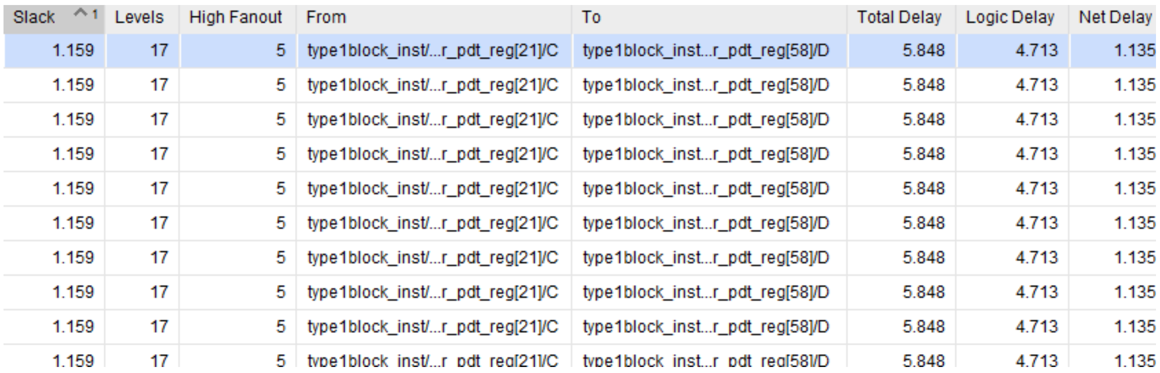


Figure 5.26: Comparison of Beamformers Resulting from Computations Performed in Fpga Versus Matlab Given the Same Inputs.



Slack ^ 1	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135
1.159	17	5	type1block_inst/...r_pdt_reg[21]/C	type1block_inst...r_pdt_reg[58]/D	5.848	4.713	1.135

Figure 5.27: QRD Timing Report from Vivado.

multiplier, normalizing inputs is the safest and maybe most important thing to do.

5.3 Hybrid FPGA-C++ Implementation

ASU created an effective beamformer construction algorithm solution that runs on an FPGA and a CPU to ease system integration. The processor does the majority of the work in this hybrid method. The FPGA starts the calculation, which migrates early to the CPU to finish. The final filters are resampled at the sampling rate of the

Table 5.10: Utilization Report for the QR Decomposition Plus Back-Substitution Block

Resource	Utilization	Available	Utilization (%)
LUT	57864	425280	13.61
LUTRAM	480	213600	0.22
FF	25080	850560	2.95
URAM	26	80	32.50
DSP	561	4272	13.13

bent pipe processing. The implementation's software is made to operate on a single A53 core under AArch64 Petalinux.

One advantage of this approach is the use of floating-point computations. Complexity arises during integration, though. The construction of the beamformer risks disrupting the timing of other system routines.

5.3.1 SNR Gain Beamformer Implementation

I created an implementation that merely achieves SNR Gain to construct and verify a fully integrated system. The rejection of interference, if any, is accidental. When operating in this mode, the system adjusts the phase and timing offsets caused by the compounded channels when computing beamforming filters. A beamformer of this kind is only appropriate when interference is minimal or nonexistent. This beamformer design has a considerably lower computational complexity than an MMSE beamformer construction.

This beamformer is computed directly from the cross-correlation vector in Equation 4.9. The specific filter for node n is

$$\mathbf{w}_n = \hat{\mathbf{r}}_n \, . \tag{5.12}$$

Note that $\mathbf{w}_n$ must undergo complex conjugation before providing it to the filter resampler.

Local cross-correlations computed within the FGPA and used for detection in the modem are provided to the software. Again, the cross-correlations must be constructed so that TDOA information is preserved. From then on, the rest of the computation progresses on a processor core instead of the FPGA. This cross-correlation has lag spacing determined by a sampling rate of $2B_U$. The beamformer is still constructed at a sampling rate of $3B_U/16$, so the initial cross-correlation must be downsampled. The downsampled cross-correlation is convolved with a corresponding channel B estimate to yield $\hat{\mathbf{r}}_n$.

This construction does not require the exchange of observation vectors rvz_n or cross-correlations $\mathbf{r}_n$. The computation is complete once the local cross-correlation estimate affected by the appropriate channel response is estimated. This implementation forms beamformers quickly. Consequently, there is little concern regarding its integration and effect on the system's timing. Additionally, the system can normalize the signal to maximize its digital power without clipping because achieving strictly SNR gain is insensitive toward relative power differences.

5.3.2 Modified MMSE Beamformer Implementation

As mentioned in the preceding sections, the whole method is likewise implemented. Although necessary, the FPGA effectively does the timeline one data pre-processing. A simplified FPGA implementation is used when utilizing the CPU to finish the calculation, eliminating all timeline two and three activities. Of course, information needs to be shared. All convolutions using channel B, covariance estimation, covariance post-processing, and linear system solution are essentially executed in the CPU during timelines two and three. The cross-correlation generated by the modem can be utilized directly if needed. Similarly, the processor must downsample the cross-correlation sequence to the proper sampling rate of $3B_U/16$.

Chapter 6

SUMMARY AND FUTURE WORK

In this dissertation, I set out to address the research topic of hardware design of the distributed coherence mesh beamforming system. Through a comprehensive investigation, I aimed to implement the DCMB on hardware and make sure the hardware design satisfies the timing and resource utilization requirements. I employed a standard FPGA design flow and several optimization technologies in hardware implementation to achieve my research objectives. This methodology allowed me to achieve the goal and test the implementation to ensure it can be used in the real world. My research yielded several key findings that shed light on how the DCMB can be implemented in the real world and achieve SNR increase at the receiver. The QRD folded design needs some modification based on the frequency requirements. The basic design of the QRD folded architecture was inspired by[5]. I changed the schedule for type one, two, and three blocks to meet the timing requirements. Narayanan helped with the implementation of two LUT technology in the QRD accelerator. I designed the matrix multiplication accelerator and Narayanan helped with the optimization of the design to increase the precision of the output. Notably, I discovered that the whole DCMB system can be implemented on the hardware, and optimization methods are helpful in the design. The findings of this dissertation make several significant contributions to the field of wireless communication system design and efficient hardware design of wireless communication systems. First and foremost, my research advances existing knowledge by implementing DCMB on the Xilinx FPGA board and demonstrating the system works as expected on hardware. Additionally, my work provides valuable insights into optimizing different accelerators and the design of the overall scheduler. While this study has provided useful insights, it is not without limitations. One primary limitation is that this design is sensitive to the input data. The other limitation is that the results could get lower SNR than simulation data when I used some actual data for the verification. Also, precision needs to be improved. These limitations may have influenced the interpretation of my findings to some extent.

Building upon the findings of this dissertation, several avenues for future research warrant exploration. For example, future studies could be related to the further optimization of individual accelerators and memory reuse strategies. Dynamic range analysis of the input data is also needed in the future. The findings of this dissertation have practical implications for improving wireless communication devices. By demonstrating the DCMB system on FPGA, our work has the potential to be used in autonomous cars, satellite communication, and cell phone communications.

In summary, this dissertation has provided a comprehensive investigation into the hardware implementation of DCMB. Through design, development, simulation, and verification of the design, I have an advanced understanding of wireless communication system design and FPGA implementation. Moving forward, I hope that my findings will help develop wireless communication system design and hardware implementation area.

REFERENCES

[1] R. Hourani, R. Jenkal, W. Davis, and W. Alexander, “Automated design space exploration for dsp applications,” *Signal Processing Systems*, vol. 56, pp. 199–216, 09 2009.

[2] N. C. A. News, “India to be first country to auction satcom spectrum,” *Indias Current Affairs 2023: National Current Affairs News*, 2022.

[3] J. Wang, W. Deng, X. Li, H. Zhu, M. Nair, T. Chen, N. Yi, and N. Gomes, “3d beamforming technologies and field trials in 5g massive mimo systems,” *IEEE Open Journal of Vehicular Technology*, vol. 1, pp. 362–371, 01 2020.

[4] J. Holtom, O. Ma, A. Herschfelt, D. G. Landon, and D. W. Bliss, “Distributed beamforming techniques for flexible communications networks,” in *2021 55th Asilomar Conference on Signals, Systems, and Computers*, 2021, pp. 400–404.

[5] S. Kanagala, “Accelerator for flexible qr decomposition and back substitution,” 2020.

[6] R. Mudumbai, G. Barriac, and U. Madhow, “On the feasibility of distributed beamforming in wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 6, no. 5, pp. 1754–1763, 2007.

[7] S. Jayaprakasam, S. K. A. Rahim, and C. Y. Leow, “Distributed and collaborative beamforming in wireless sensor networks: Classifications, trends, and research directions,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2092–2116, 2017.

[8] A. Herschfelt, A. Chiriyath, D. W. Bliss, C. D. Richmond, U. Mitra, and S. D. Blunt, “Vehicular rf convergence: Simultaneous radar, communications, and pnt for urban air mobility and automotive applications,” in *2020 IEEE Radar Conference (RadarConf20)*, 2020, pp. 1–6.

[9] J. Holtom, “Distributed coherent mesh beamforming: Algorithms and implementation,” *ProQuest Dissertations and Theses*, p. 129, 2023, copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-05-19.

[10] S. M. Lasassmeh and J. M. Conrad, “Time synchronization in wireless sensor networks: A survey,” in *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, 2010, pp. 242–245.

[11] D. W. Bliss and S. Govindasamy, “Adaptive wireless communications,” 2013.

[12] A. Herschfelt, H. Yu, S. Wu, H. LEE, and D. Bliss, “Joint positioning-communications system design: Leveraging phase-accurate time-of-flight estimation and distributed coherence,” in *Conference Record of the 52nd Asilomar Conference on Signals, Systems and Computers, ACSSC 2018*, ser. Conference

Record - Asilomar Conference on Signals, Systems and Computers, M. Matthews, Ed. IEEE Computer Society, Feb. 2019, pp. 433–437, 52nd Asilomar Conference on Signals, Systems and Computers, ACSSC 2018 ; Conference date: 28-10-2018 Through 31-10-2018.

[13] R. Mudumbai, G. Barriac, and U. Madhow, "On the feasibility of distributed beamforming in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 5, pp. 1754–1763, 2007.

[14] R. Mudumbai, D. R. Brown Iii, U. Madhow, and H. V. Poor, "Distributed transmit beamforming: challenges and recent progress," *IEEE Communications Magazine*, vol. 47, no. 2, pp. 102–110, 2009.

[15] D. R. Brown III, U. Madhow, M. Ni, M. Rebholz, and P. Bidigare, "Distributed reception with hard decision exchanges," *IEEE Transactions on Wireless Communications*, vol. 13, no. 6, pp. 3406–3418, 2014.

[16] D. Scherber, P. Bidigare, R. ODonnell, M. Rebholz, M. Oyarzun, C. Obranovich, W. Kulp, D. Chang, and D. R. B. III, "Coherent distributed techniques for tactical radio networks: Enabling long range communications with reduced size, weight, power and cost," in *MILCOM 2013 - 2013 IEEE Military Communications Conference*, 2013, pp. 655–660.

[17] D. W. Bliss, S. Kraut, and A. Agaskar, "Transmit and receive space-time-frequency adaptive processing for cooperative distributed mimo communications," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5221–5224.

[18] P. Bidigare, U. Madhow, R. Mudumbai, and D. Scherber, "Attaining fundamental bounds on timing synchronization," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5229–5232.

[19] F. Quitin, U. Madhow, M. M. U. Rahman, and R. Mudumbai, "Demonstrating distributed transmit beamforming with software-defined radios," in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2012, pp. 1–3.

[20] M. M. Rahman, H. E. Baidoo-Williams, R. Mudumbai, and S. Dasgupta, "Fully wireless implementation of distributed beamforming on a software-defined radio platform," in *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, 2012, pp. 305–316.

[21] R. Mudumbai, U. Madhow, R. Brown, and P. Bidigare, "Dsp-centric algorithms for distributed transmit beamforming," in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*. IEEE, 2011, pp. 93–98.

[22] S. Hanna and D. Cabric, "Distributed transmit beamforming: Design and demonstration from the lab to uavs," *IEEE Transactions on Wireless Communications*, vol. 22, no. 2, pp. 778–792, 2022.

[23] S. Hanna, E. Krijestorac, and D. Cabric, "Destination-feedback free distributed transmit beamforming using guided directionality," *IEEE Transactions on Mobile Computing*, 2022.

[24] R. Mudumbai, D. R. Brown Iii, U. Madhow, and H. V. Poor, "Distributed transmit beamforming: challenges and recent progress," *IEEE Communications Magazine*, vol. 47, no. 2, pp. 102–110, 2009.

[25] J. A. Nanzer, S. R. Mghabghab, S. M. Ellison, and A. Schlegel, "Distributed phased arrays: Challenges and recent advances," *IEEE Transactions on Microwave Theory and Techniques*, vol. 69, no. 11, pp. 4893–4907, 2021.

[26] J. Mack, S. E. Arda, U. Y. Ogras, and A. Akoglu, "Performant, multi-objective scheduling of highly interleaved task graphs on heterogeneous system on chip devices," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 9, pp. 2148–2162, 2022.

[27] S. Jayaprakasam, S. K. A. Rahim, and C. Y. Leow, "Distributed and collaborative beamforming in wireless sensor networks: Classifications, trends, and research directions," *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2092–2116, 2017.

[28] D. W. Bliss, *Modern communications: a systematic introduction.* Cambridge University Press, 2021.

[29] H. Zhang, J. Luo, X. Chen, Q. Liu, and T. Zeng, "Fresnel based frequency domain adaptive beamforming for large aperture distributed array radar," in *2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, 2016, pp. 1–5.

[30] G. He, X. Gao, L. Sun, and R. Zhang, "A review of multibeam phased array antennas as leo satellite constellation ground station," *IEEE Access*, vol. 9, pp. 147 142–147 154, 2021.

[31] Z. Lin, M. Lin, J. Ouyang, W.-P. Zhu, A. D. Panagopoulos, and M.-S. Alouini, "Robust secure beamforming for multibeam satellite communication systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6202–6206, 2019.

[32] N. Benvenuto, E. Conte, S. Tomasin, and M. Trivellato, "Predictive channel quantization and beamformer design for mimo-bc with limited feedback," in *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, 2007, pp. 3607–3611.

[33] J. Holtom, G. Gubash, A. Herschfelt, O. Ma, W. Standage-Beier, H. Yu, and D. W. Bliss, "Rapid implementation and demonstration of radio applications using wiscanet," in *2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC).* IEEE, 2021, pp. 1500–1505.

APPENDIX A

MATLAB SIMULATIONS

A fully parameterized MATLAB simulation is also built to quickly iterate and project our algorithm's performance in various circumstances. This Matlab simulation program was mainly created by Owen and Jacob. I added this section to explain how to get the reference data to compare with FPGA results. There are several parameters in the algorithm itself that need to be adjusted. The simulation makes efficient investigation of these parameters possible. Regarding the intended use cases and surroundings The MATLAB code may be a C++ version of the beamformer computation customized for the algorithm team's platform and the environments in which they anticipate it will operate. Furthermore, this simulation may save test vectors for use in testing by the developers of FPGA implementations.

The simulation runs in a loop, where the beginning is marked by the transmission of a training waveform from gold A. Each cycle tracks the propagation of an epoch of gold A's data stream through all relay stages. The cycle ends once Gold B has received and processed this stream. Cycles are split into a gold A to mosaic transmission stage and a mosaic to the gold B stage. In between stages, the beamformer is computed, although, in actuality, the filters are constructed in parallel with signal relaying. The beamformer calculated using the current epoch is deployed on the next cycle.

The algorithm team employs surrogate waveforms in our simulation. BPSK signals make up the tactical and underlay waveforms. The decision to choose BPSK was not entirely random. The tactical waveform and the underlay need good auto-correlation characteristics appropriate for low SNR situations. The foundation of both signals is a random binary sequence. Dead time between transmissions is absent from the simulated underlays, but including it in real life is still beneficial.

Gold A's underlay structure repeats every 25 ms, so a new beamformer is constructed every 25 ms. Additional processing latency should be included to assess the beamformer's performance more accurately.

To simplify processing, the analysis window's duration is much shorter than 25 ms but long enough to preserve the actual duration of the deployed training waveform. Channel sounding sequences are sent on a 2.5 ms interval. Similarly, in simulation, the TDMA slots have been shortened only to include training information in the underlay. Despite the shorter transmissions, all estimators in the processing chain use the actual length of the training sequences. The analysis window the mosaic receives is long enough to transmit ten-channel sounding sequences from the mosaic to gold B. As a result, the bent pipe transmission is long enough to jam the channel-sounding signal realistically. The duration of the tactical and bent pipe signals is as long as the analysis window. One assumption they implement that allows them to simplify this is using genie channels. Gold A's underlay carries back channel estimates for constructing the reverse direction's beamformer. However, the genie channel allows however much information necessary to be exchanged instantaneously when demanded. Consequently, they can shorten the transmission while disregarding the waveform needed to communicate the feedback. The channel estimate produced at gold B is fed back and used in the next epoch's beamformer computation.

The jammer is a Gaussian signal that spans the entire frame's duration. Our team has only tested our system with constant streaming jamming. In future studies, we plan to include support for intermittent interference. People usually investigate with the jammer occupying the same spectral support of the tactical waveform to evaluate the beamformer's exclusive ability to reject interference. The signal is filtered and upsampled to limit energy inside the tactical waveform's spectral support. Of course, we do not influence the jammer waveform. Except for a tiny variation in the system sample rate, the system faithfully replicates the data processing. To ease the process of resampling to and from all beamformer construction data sampling rates and signal chipping rates, the simulation system's sampling rate of 122.88 MHz is marginally lower than the hardware's real rate of 125 MHz.

By employing 122.88 MHz, the actual underlay bandwidth of 40.96 MHz is precisely three times oversampled. As long as there are no appreciable distortions brought about by resampling, the difference should have no impact. The data is resampled to the same processing speeds utilized on the hardware implementation immediately following receive filtering. Both in simulation and hardware, the beamformer is built at 7.68 MHz, oversampling the highest tactical bandwidth supported by a factor of 1.5. Under 5.12 MHz, tactical bandwidths can be utilized if the proper low-pass and anti-aliasing filters are applied. The actual underlay bandwidth of 40.96 MHz is precisely three times oversampled by using 122.88 MHz. As long as resampling doesn't result in any noticeable distortions, the difference shouldn't matter. After receive filtering, the data is resampled to the same processing rates used on the hardware implementation. The beamformer is designed at 7.68 MHz in both simulation and hardware, oversampling the maximum tactical bandwidth supported by a factor of 1.5. Tactical bandwidths below 5.12 MHz are usable if the appropriate low-pass and anti-aliasing filters are used.

After ensuring a standard reference power in all component signals, each can be weighted directly according to the prescribed USR before summing. The proper USR could not be guaranteed without these intermediate normalization steps. This normalization scheme is best suited for determining the nominal performance of the algorithm in simulation. However, the MMSE optimization problem does not limit the maximum power. The algorithm only implicitly imposes an average power constraint. This scheme, in practice, may result in clipping if insufficient headroom is provided while attempting to meet some target transmit power. One way to resolve this is to provide such headroom and calibrate the power to meet the target transmit power. The system may only be able to rescale signals using bit shifts. If the power amp can compensate for less digital power, there is likely a tiny performance gap. To properly assess performance based on SISO power measurements, the baseline measurement should be made using the same normalization scheme.

The algorithm is tested using several mosaic tile arrangements, including a linear array with a max distance of 200 m and random positions within a bounded box generated according to a uniform distribution. The simulation supports positioning in 3 dimensions, but the network is typically in the XY plane. The mosaic is centered on the origin. We place jammers at some distance from the origin but at uniformly random angles of arrival. To validate the methodology and set a performance baseline, the static simulation runs with all nodes. Even with static nodes, frequency offsets can still occur. Specifically, we model residuals from frequency estimation and correction algorithms applied by L3 Harris. To establish nominal performance, we simulate the system under time-frequency aligned conditions.

Wideband channels are generated based on a Rician distribution that also includes the effects of relative time offsets, differences in attenuation, and phase delays dictated by the generated geometry. Impulse responses are directly generated at the system sampling rate. First, each channel for a particular transmission stage is created time aligned with each other and the sampling lattice. This structure is given by

$$\underline{\mathbf{h}}_m = \frac{D_m^2}{\max\limits_n D_n^2} \left(\sqrt{\frac{K}{1+K}} \begin{bmatrix} e^{i2\pi f_c/cD_m} & 0 & \dots & 0 \end{bmatrix} \right. \\ \left. + \sqrt{\frac{1}{1+K}} \underline{\mathbf{h}}_{scatter} \right) , \tag{A.1}$$

where the fraction $D_m^2 / \max\limits_n D_n^2$ represents a relative attenuation based on the squared distances between a gold node and the m^{th} tile, D_m. The scattering component $\underline{\mathbf{h}}_{scatter}$ is a white, complex-valued Gaussian process, where each sample is distributed by $\mathcal{CN}(0, \mathbf{I})$, weighted by a taper,

$$e^{\frac{\log(0.1)}{T_{decay}}\left[0 \quad \dots \quad T_{decay}\right]}, \tag{A.2}$$

That lowers the signal's reception strength while it travels along longer distances. Compared to the first, the longest scattering route is 20 dB less intense. The line of sight power about the scattering response power is modulated by the parameter K. The relative temporal offsets of the line of sight component are then incorporated by zero-padding the impulse responses to the closest total delay.

Subsequently, every impulse response undergoes a fractional delay filtering process to finalize the time shift and precisely include TDOA data according to the network architecture in the channels. The channel impulse responses are static between loops for baseline testing.

We take for granted that data will be sent instantly and reliably via a genie channel upon request. This presumption simplifies reviewing the necessary quantity of samples sent across the network. The beamformer calculation is always sample-hungry during performance evaluation. More specifically, the covariance matrix estimator may use only 125 samples from each tile.

The algorithm team showcases simulated outcomes for the most severe scenarios of domination of gold and mosaic noise that the system is intended to manage. System and signal settings that try to mimic the deployed system as closely as feasible were used to get these findings. The technique employs 125 samples, sampled at $\frac{3}{16}B_U$ from each tile to estimate the covariance matrix. For cross-correlation and channel estimation, 30000 samples taken at $2B_U$ are used for each tile. The USR is -10 dB for short-link communications and -6 dB for long-link communications. A 20 dB INR interferer in the same spectrum as the mosaic observes the tactical waveform. During the long link transmission stage, the received SNR is 0 dB. Last but not least, the short link transmission stage's received SNR is 20 dB.

In the discussed examples, the mosaic is arranged in a linear array centered around the origin of the XY plane. The most considerable pairwise distance is 200 m. The gold nodes are placed in positions such that communicating in one direction yields the gold noise dominant operating regime, and relaying in the opposite direction yields the mosaic noise dominant operating regime. The distant gold node is placed 50 Km from the mosaic center at a random angle. The closer gold node is also placed 200 m from the mosaic center at a random angle. An example set of the radio positions is shown in Fig. A.1. The top panel of Fig. A.1 shows all nodes from a bird's eye view. The bottom panel shows a restricted view containing only the close gold node and the mosaic.

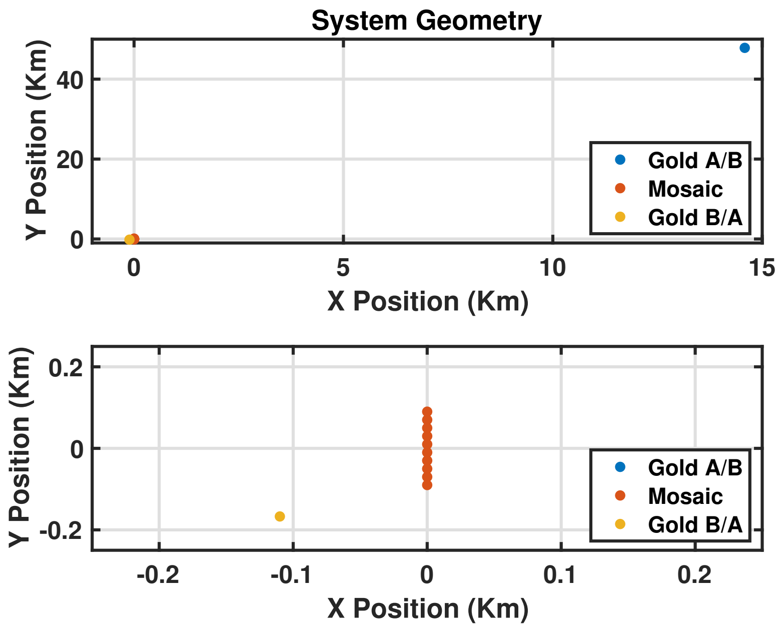


Figure A.1: Example Gold and Blue Node Placement Where the Mosaic Is Arranged as a Uniform Linear Array(Previous tests report from Owen and Jacob).