Muhammad Umair Arif

01-244222-014

RESOURCE OPTIMIZE FPGA IMPLEMENTATION OF SONAR SYSTEM

Master of science in Electrical Engineering

Supervisor: Dr Junaid Imtiaz

Co-Supervisor: Asim Altaf Shah

Department of Electrical Engineering

Bahria University Islamabad

MS - 13
Thesis Completion Certificate

Student Name: **Muhammad Umair Arif** Registration Number: **58575**

Program of Study: **Masters of Science in Electrical Engineering**

Thesis Title: **Implementation of Linear array signal processing of FPGA in signal domain**

It is to certify that the above student's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at **12%** that is within the permissible set by the HEC. For MS/MPhil/PhD. I have also found the thesis in a format recognized by the BU for MS/MPhil/PhD thesis.

Principle Supervisor's Signature:

Principle Supervisor's Name:

September 1, 2024

MS - 14 A
Author's Declaration

I, **Muhammad Umair Arif** hereby state that my MS thesis titled **"Implementation of Linear array signal processing of FPGA in signal domain"** is my own work and has not been submitted previously by me for taking any degree from **"Bahria University, Islamabad"** or anywhere else in the country / world.

At any time if my statement is found to be incorrect even after my Graduate the university has the right to withdraw cancel my MS degree.

Muhammad Umair Arif
01-244222-014
September 1, 2024

MS - 14B
Plagiarism Undertaking

I, **Muhammad Umair Arif** solemnly declare that research work presented in the thesis titled

Implementation of Linear array signal processing of FPGA in signal domain

is solely my research work with no significant contribution from any other person. Small contribution / help whenever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of Bahria University and the Higher Education Commission of Pakistan towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used is properly referred / cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS degree, the university reserves the right to withdraw / revoke my MS degree and HEC and the university has the right to publish my name on HEC / University Website on which name of students who submitted plagiarized thesis are placed.

Muhammad Umair Arif

01-244222-014

August 28, 2024

Acknowledgments

I must first and foremost express my deepest gratitude to my advisor, Mr Asim Altaf Shah for his unyielding support, guidance, and motivation throughout this process. Their invaluable insights and expertise has moulded this thesis into its current form.

I would like to thank my thesis committee Dr Junaid Imtiaz, Dr Atif Jafri, for their invaluable feedback, advice and time. We would like to acknowledge that their perspective significantly enriched this work.

I extend my sincere thanks to the lab caretaker and staff. This supportive environment was essential for the progression of my research.

Finally, I am grateful to my family, My parents for unconditional support and constant encouragement in everything I do.

This thesis is a reflection of the effort and support from all those amazing people Thanks, everyone you were a part of the journey.

Muhammad Umair Arif

Acronyms and Abbreviations

FPGA	Field Programmable gate arrays
Sonar	Sound Navigation and Ranging
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
SP	Signal Processing
IP	Intellectual properties

ABSTRACT

Field Programmable Gate Array (FPGA) is an efficient and compact tool for fast processing and computations of signals. In this project, we explore the capabilities of Sonar system implemented on FPGA, addressing the processing time to be minimize. Sonar technology is crucial for underwater target recognition, detection and also for other liabilities like, to check and measure the underwater crimes. Typically, Sonar system rely on hydrophones for reliable processing. In this project, we are implementing the 6-channel linear array beamforming on FPGA. From each channel, data of 2048 samples were fed into the model for processing. The process begins with the application of a Fast Fourier Transform of the input data. The resulting FFT data is then subjected to beamforming, where desired frequency bins of the FFT result are multiplied by pre-stored coefficients in a ROM. Finally, we take the Inverse Fast Fourier Transform to take the signal back into its original shape. We analyze the results by comparing it with the Matlab fix model benchmark. Our primary objective is the implementation of this linear beamforming on FPGA utilizing minimum Hardware resources. The system successfully meets the stringent requirement of processing the entire algorithm in under 2 milliseconds. The implementation was carried out using the Verilog hardware description language, ensuring an optimized and reliable solution for real-time sonar signal processing.

Table of Contents

List of Figures

List of Table

CHAPTER 1

INTRODUCTION

1.1: Project Background

The development of digital virtual systems is constantly replacing analog systems. Digital systems are the true strength of current-day corporations, products, techniques, and services, and their characteristics are increasingly enhanced by these technologies. Communication, traffic, control, weather forecasting systems, internet and so forth Programmable Gate Array (FPGA) devices are the applications of present-day digital technologies. These devices (FPGAs) operate at high clock frequencies and attain high execution in computations of digital and signal processing (DSP) algorithms. Sonar (also known as sound navigation and ranging) is a technique that uses sound propagation to navigate and measure distances for objects and targets. We required results in minimum time to quickly analyze and respond accordingly. FPGAs are best solutions to keep track of fast signal processing and having the functionalities capable to work efficiently in these conditions. Now modeling of these algorithms in a way to get the result with using minimum resources and in less time is the challenge and the target. The thesis focuses on the implementation of a 6-channel linear array beamforming system for SONAR, utilizing FPGA technology and

programmed in Verilog HDL. The project involves critical signal processing algorithms, including FFT, IFFT, and beamforming, implemented on FPGA to achieve real-time processing. Beamforming is optimized using nine azimuth angles for directing acoustic signals and enhancing detection accuracy. The system is designed to handle 444 coefficients per beam and employs ROM-based storage and multiplexing techniques to select beams for processing.

1.2: Project Description

In this project, data is received from six different channels, organized in a linear array format, enabling target detection and analysis in one dimension. The project is divided into three primary stages. The first stage involves applying a 2048-point FFT to the data from each of the six channels. This transformation converts the time-domain signals into the frequency domain for further processing. In the second stage, beamforming is performed. This involves multiplying specific frequency bins, corresponding to the 38 kHz - 42 kHz range, by pre-determined coefficients. The system generates nine distinct beams, each with its own set of coefficients. The final stage is the baseband processing. Here, the results from the beamforming stage are processed using an Inverse Fast Fourier Transform (IFFT) to convert the data back into the time domain. Before applying the IFFT, the sample length is extended from 74 to 2048 to match the original FFT length. The entire computation process takes 1.07 milliseconds. The total number of clock cycles required for the operation is determined by the input frequency, which is 112,000 samples per second. FPGA's parallel processing capabilities are utilized to handle large data sets in real time, ensuring efficient

processing within the given 10 ms time frame. The system's accuracy and performance are verified by comparing the FPGA results with MATLAB simulations, demonstrating the potential of FPGA for scalable and high-performance SONAR applications.

1.3: Project Objective

This project focuses on developing and evaluating efficient low-complexity FPGA implementation of the Sonar system. The project aims to achieve efficient signal processing by integrating key algorithms such as FFT, IFFT, and beamforming, optimized for multiple azimuth angles. The goal is to leverage FPGA's parallel processing capabilities to handle large data sets and real-time computations, ensuring high-speed and accurate detection. The performance will be validated by comparing the results with MATLAB simulations, demonstrating FPGA’s effectiveness in advanced SONAR systems. DSPs and GPP also provide fast results, but they take more hardware resources for operations like beam-forming. We are using VERILOG language for the implementation of the project. VIVADO is the software we use for project designing and to simulate the simulation. For the reference model, we use MATLAB to verify our Vivado design results.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

2.1.1 Scope of the review

We began by introducing planar array signal processing techniques, along with the operational environment necessary for their effective implementation. Signal processing in sonar systems can be broadly classified into two types: active and passive processing. Following this, we discussed the challenges encountered during sonar operation, such as the need for real-time processing, handling large volumes of data, and minimizing latency. These challenges necessitate a thorough investigation to optimize system performance. We then elaborated on why Field Programmable Gate Arrays (FPGAs) offer a superior solution compared to Digital Signal Processors (DSPs). FPGAs provide significant advantages, including reduced processing time and higher resource efficiency. This is due to their parallel processing capabilities and the ability

to customize hardware configurations for specific tasks, making them ideal for real-time signal processing in sonar systems. Moreover, we discussed various optimization techniques that can be leveraged through the use of FPGAs to enhance the efficiency of our solution. These techniques include pipelining, parallelism, and efficient memory management, all of which contribute to lower latency and more effective use of hardware resources. We also explored future trends and emerging techniques that could lead to even more compact and efficient solutions for sonar signal processing. These advancements may include the integration of advanced algorithms and the continued evolution of FPGA technology, which could further enhance performance and reduce system size. Finally, we reviewed the Fast Fourier Transform (FFT) algorithm, its functionality, and its role in signal processing. We also touched on other relevant algorithms that contribute to the overall effectiveness of the sonar system. These algorithms are essential for transforming data between the time and frequency domains, which is a critical aspect of signal analysis in both active and passive sonar processing.

2.2 Planar Array Signal Processing

2.2.1 Introduction

As the name indicates, “Planar Array Signal Processing” involves processing, data carrying signals collected from sensor arrays operating in the environment of interest (such as on the ground, above ground, or underwater). The relationship between the environment, sensor array,

and processor is illustrated in the system model in Fig 1. In passive array signal processing technology, the sensor array has the sole task of listening to the environment. In active sonar devices, emitters are used to illuminate the environment and sensor arrays listen for signals emitted from the environment and/or objects of interest [1]. Sensors can be used in many ways. Examples of sensors include:

(a) Antennas in radar, radio communications, and radio astronomy

(b) Hydrophones in sonar

(c) Geophones in seismology

(d) Ultrasonic probes and X- in medical imaging beam detectors.

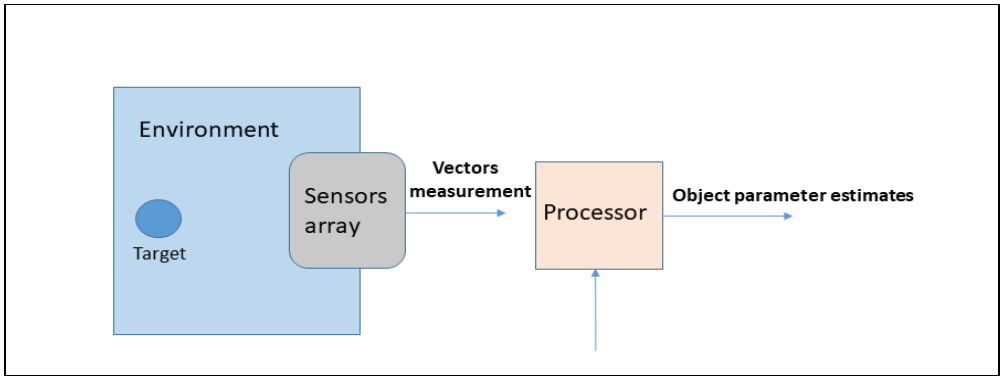

Fig. 2.1 Sensors Deployment

2.2.2 A review of sonar array limitations

In radar, radio communications, and radio astronomy, antennas function as electronic devices designed to capture electromagnetic waves. In contrast, sonar, seismology, and medical imaging rely on sound transducers such as hydrophones, geophones, and ultrasound probes—that are engineered to respond to acoustic energy waves. Measurement vectors, collected by sensor arrays in discrete time intervals (snapshots), provide critical information about the environment and the target of interest, which may include details about the target's location, orientation, movement, or the number of sources (emitters) interacting with the sensor array. [9]

The processor's role is to generate accurate estimates of these parameters by extracting relevant information from the measurement vectors. However, the performance of the processing algorithms is constrained by several key factors. Firstly, sensor noise, which may be isotropic or anisotropic, affects the accuracy of the measurements. Secondly, inaccuracies in the mathematical modeling can arise when the assumed model fails to accurately represent the physical phenomena responsible for the sensor output.

Finally, the sensor's capacity to collect data is inherently limited by its size and the time-bandwidth product of the event being measured. These limitations collectively impact the precision and reliability of signal processing in these applications.[1]

2.2.3 Areas of working

We have three areas in which we conceptualize the problems of array signal processing:

Object Space: The object space is characterized by a set of object parameters and environmental factors of interest, along with numerous other influencing elements. Specifically, the precise representation of the object's position is not directly known and must be inferred through the estimation of an unknown signal sequence function.

Measurement (observation) area: The measurement area is defined by two key factors: the specific types of light employed for environmental monitoring (relevant only to the operating system), and the data utilized for the specific design pattern structure of the retrieved collection. The data content within this space is consistently smaller than the corresponding physical space.

Estimated position: The third position in the conceptual model is defined by the estimated parameters that determine the position of the product. These parameter estimates, in essence, provide a solution to the array signal processing problem. [9]

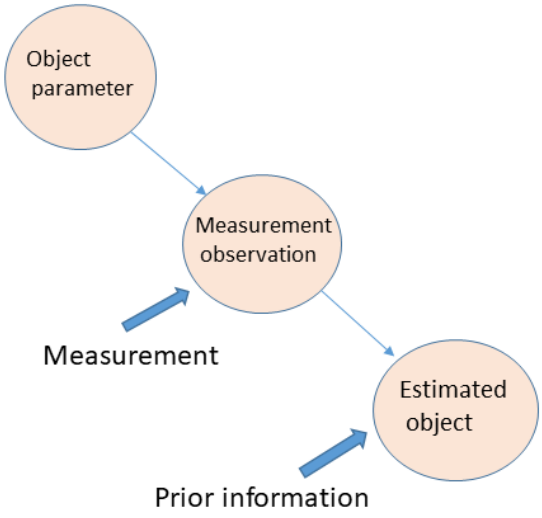

Fig. 2.1 Array signal field

The arrays contains single-element transducers mounted in a way that improves signal directivity, increase the audio signal power level, improve beamforming, and aids in beam steering and shielding. The most common types of array are linear arrays, such as discrete and continuous elements, and planar arrays, which include circular, square, and rectangular arrays, each containing multiple transducer elements in different configurations. Although individual array elements are usually omnidirectional, the array itself can have multiple directions depending on its extension or surface.

2.3 Overview of Sonar Technology

2.3.1 History of Sonar technology

Sonar technology has been developed over more than 100 years, with a long history of advancement and innovation in underwater acoustic technologies. The origins of sonar technology can be traced back to the nineteenth century, with significant progress made during the twentieth century, particularly during World War I and World War II where transducers finds an underwater applications, stands for Sound Navigation and ranging, encompasses both active and passive Sonar, driven by the need for underwater object detection and location.

The loss of the Titanic and submarine operation during World War I were important events that led to the advancement of sonar technology and highlighted the need for better underwater research [4]. The development of sonar and array technology continues with the increasing

demand for purposes such as underwater research, hydrology, swimming, and fisheries research, including the exploitation of oil, gas and minerals.

2.3.2 Types of Sonar System

Passive Sonar: The main purpose of Sonar is to navigate and locate objects underwater. In this echoes or sounds emitted by the objects and targets are listened and processed further. Passive systems can range from a single hydrophone to several hundred hydrophones arranged in various array geometries.

Active Sonar: We intentionally emits echoes and listened them once reflected back from the objects and then processed on the signals, extracting the information. [4]

2.3.3 Different Sonar principles:

Single beam echo sounders:

SBES operates on the principle of transmitting and receiving updated products in the realms of hardware, firmware, and software. Specifically, sinusoidal pulses, synchronized by the master clock, are amplified by a power amplifier, where the sound power level is adjusted according to the water depth and carrier frequency. At the receiver, inputs are typically time-gated to prevent direct capture of the transmitted signal and to mitigate the effects of transmitter ringing. The signal then passes through a band-pass filter to eliminate ship noise. Finally, the signal is

processed via time-varied gain (TVG), with amplification based on the amplitude of the signal received at varying depths. [4]

Multi beam echo sounders:

MBES operates as a single beam with a small width and a small distance between each beam. The best MBES systems today have beam widths of 0.5° to 1° and can run more than 800 lines.

Side scan Sonar:

Side-scan sonar is a tool used to visualize the seabed and objects located on or above the seafloor. It employs a sonar generator and the movement of a transducer through the water to produce high-resolution, two-dimensional images of sonar patterns. The transducer can be mounted on the underside of the vessel's hull. A side-scan sonar system may feature a single sensor that emits an acoustic signal to the side of the platform, or it may have two sensors that emit acoustic signals on both sides of the platform, allowing for the scanning of a wide area of the seabed as the system moves forward through the water.

2.3.4 Applications of Sonar Technology:

Seafloor mapping can also be used to search for wrecks of ships and airplanes. Sonar can also be used to locate lost items, such as cargo containers and other cargo on ships, or chemical weapons and war materiel, such as bombs and gas bombs.

Marine geology uses sonar systems to identify, study, and map the seafloor, as well as to record sub-bottom profiles. This is often achieved using low-frequency parametric sonar. Additionally, underwater research that focuses on ambient noise, coral reef ecosystems, pollution from oil spills and waste dumping, and marine archaeology such as the search for historical artifacts and the study of submerged cities.

Fisheries research estimating fish species and fish biomass. Sonar systems also increase efficiency and catching operations.

Physical oceanography used to study various ocean phenomena resulting from small local temperature changes in shallow waters.

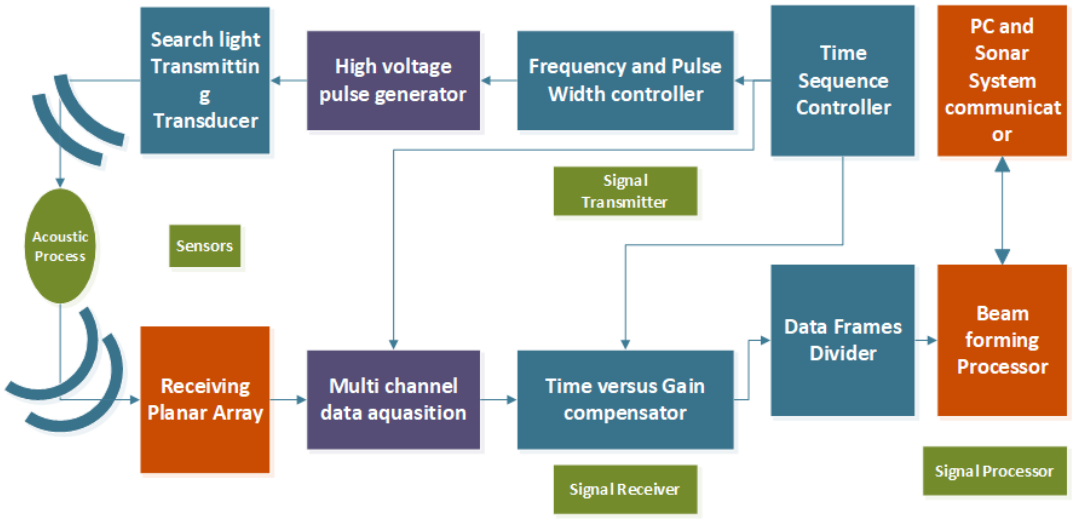

Fig. 2.2 Operating principle of Sonar

2.4 Fundamentals of FPGA Technology

2.4.1 Advantages of FPGA-Based Prototyping in VLSI Design

Many procedures are involved in manufacturing integrated circuits form VLSI design. FPGA prototyping (Field Programmable Gate Array) is one of the procedure of them. Because of the importance of Simplicity, DSP hardware limitations, and FPGA customization for application-specific I/O requirements, FPGA-based systems offer six to twelve times the performance of traditional equivalent DSP systems. FPGA-based beam formers can provide computational power equivalent to multiple DSPs by leveraging symmetries in sensor arrays, enabling efficient broadcast of data to processing elements and pipelined operations for faster calculations. [13] The advantages of FPGAs, such as strong parallelism and the ability to handle high data rate algorithms, make them a suitable platform for real-time signal processing in high-frequency active sonar systems.

2.4.2: Leveraging FPGA Technology for Advanced Signal Processing

FPGAs are instrumental in the heterogeneous integration of diverse "hard IP" blocks—such as SSDs and floating-point DSPs—enabling enhanced application-specific functionality. Recent advancements in FPGA technology, characterized by increased storage capacity, reduced power consumption, cost efficiency, and higher gate counts, have paved the way for high-performance

applications. These advancements allow FPGAs to parallelize algorithms efficiently, boosting power efficiency, flexibility, and accuracy, which are critical for SVM-based (Support Vector Machine) applications.[1]

In the context of digital beamforming, FPGA-based hardware implementations offer an efficient platform for deploying advanced algorithms and techniques such as Linearly Constrained Minimum Variance (LCMV) and Minimum Variance Distortion less Response (MVDR). When combined with a Quick-Support Vector Machine (QS-SVM) model for Direction of Arrival (DoA) estimation, these implementations ensure real-time execution with high throughput. The inherent parallelism and capability to handle high data rate algorithms make FPGAs an ideal choice for real-time signal processing in high-frequency active sonar systems. [13]

FPGAs facilitate the realization of complex algorithms in hardware, resulting in low latency and high data rates in practical applications. In signal processing, FPGA platforms decompose algorithms into various operations, such as multi-channel data demodulation, Fast Fourier Transform (FFT), beamforming, matched filtering, and Constant False Alarm Rate (CFAR) processing. These capabilities make FPGAs a powerful tool for executing sophisticated, real-time signal processing tasks in challenging environments.

2.5: Hardware Description Language (HDL) for FPGA Programming

2.5.1 VHDL (Very large scale Hardware Description Language)

VHDL was developed as part of VHSIC (Very High Speed Integrated Circuit) in the 1980s under the auspices of the US Department of Defense. Standard language is ideal for describing complex processes. It also includes design support that makes managing large projects easier, allow the characteristics of combinational and sequential logics. VHDL is widely used in industries where reliability and stability are important, such as aerospace, defense and telecommunications.

2.5.2: Verilog

Developed by Phil Moorby of Gateway Design Automation in the mid-1980s, Verilog has become one of the most popular HDLs due to its simplicity and ease of use. Verilog syntax is similar to the C programming language and can be used by engineers familiar with C. Verilog supports behavior and design, making it suitable for many levels of design abstraction. It includes design for modeling hardware components and connections. Verilog is widely used in the electronics, automotive and computers.

2.5.3: Comparison of VHDL and Verilog

Both VHDL and Verilog support multiple levels of abstraction, including behavior, register transfer Level (RTL), and gate level. VHDL's powerful types and granularity make it ideal for large, complex models that require reliability. Verilog's simplicity and flexibility make it ideal for rapid prototyping and re-engineering processes. Both languages are supported by a variety of integration and simulation tools from major EDA (electronic design automation) vendors, including Synopsys, Cadence, and Mentor Graphics.

2.6 Optimization Techniques for FPGA Implementation

2.6.1: Significance of Ping-Pong Buffers in FPGA Systems

Ping-pong buffers are important in FPGA-based systems due to their unique features and advantages:

Continuous Data Processing: Ping-pong provides a mechanism for synchronizing data reads and writes without causing the FPGA system to persist and make the data inconsistent.

Preventing Data Loss: By allowing the FPGA to switch between two buffers for data transfer, ping-pong buffers help prevent data loss during high-speed processing, enhancing the system's reliability.

Optimized Resource Utilization: These buffers increase the utilization of resources by supporting valuable data in the system, reducing downtime and improving overall performance.

Reduced Latency: Ping-pong buffers help reduce data transfer latencies in FPGA systems, instantly improving performance and responsiveness.

Enhanced Throughput: The use of ping-pong buffers increases data throughput by managing continuous data, which is important for applications requiring high speed and continuous operation. [12].

2.6.2: Optimization of Performance through Decomposition

Efficient Resource Utilization: By dividing the algorithm into stages such as multi-channel data demodulation, FFT, IFFT and matching filtering, the system effectively uses FPGA programs to integrate and enhance the overall system.

Enhanced Processing Speed: Breaking algorithms into smaller, manageable tasks can process larger data faster, thus improving the instantaneous performance of the system.

IP Core Integration: Basic signal processing such as FFT, IFFT, parallel and DDR controllers use FPGA IP cores to simplify algorithm implementation and improve performance.

This optimization strategy through algorithm decomposition on FPGA-based signal improves the performance of high-frequency active sonar systems.

2.6.3: Efficient Data Handling and Signal Processing in Underwater Using FPGA

In the context of underwater acoustic signal processing, Field Programmable Gate Arrays (FPGAs) play a crucial role from the initial capture of the input signal through to its processing stages. Upon receiving the input signal, the FPGA manages electronic control, orchestrating both data and signal processing within the system. The FPGA first evaluates the incoming data to ensure it meets specific criteria necessary for Fast Fourier Transform (FFT) processing. Once the FPGA verifies that the data satisfies the FFT requirements, it triggers an interrupt signal to the Digital Signal Processor (DSP), indicating that the data is prepared for further processing. This

detection mechanism ensures the FPGA operates efficiently, handling the input signal with precision and seamlessly preparing the data for communication with the DSP. By doing so, the system maintains optimal performance and minimizes the risk of errors, ensuring reliable operation in underwater acoustic applications

2.7: Future Trends and Research Directions

2.7.1: Enhancing Underwater Security: Challenges and Innovations in Sonar Technology

Now for advanced and more secure surveillance of underwater world, new research trends are seen on the topic. Nowadays underwater security is more vulnerable than ground security, and needs to improve its security level to the extent of adequate measures. Companies and underwater Security bodies are working on to develop high rated and more advanced sonar security systems for unseen underwater threats and risks. [2]

Due to no constant security check ins, unbridled measures, underwater crimes seems to be a easy way to pursue a crime. To make security level more secure as capabilities of ground securities, relevant bodies are keep working to make their underwater security system more advanced and up to date to cater and eliminate any type of crime comes under the vicinity of water. The common crimes observed underwater includes the smuggling of drugs and illegal weapons and substances, and also the disposal of harmful substances that causes serious threats also to the underwater livings.

The project was carried out by a Japanese company in order to improve underwater security and to protect the facilities near the water area, uses the Sonar system having three transducers operating in three different frequencies to cover different area ranges to detect objects and targets nearby and far. They also embedded a high resolution camera on different locations which can detect images and then analyzed by a software in real time, to eliminate the background images and noises. [2]

2.7.2: The Role of Fractional Fourier Transform in Sonar Systems

To make Sonar system more convenient and effective, we do use a Fractional Fourier Transform for major processing operations. It gives more advantages and processing capabilities than a conventional Fast Fourier Transform on Sonar signal processing. When using chirp signals (which frequency increases or decreases with time) for Sonar signal processing, we do use of Fractional Fourier Transform that has the better capability to intercept and process chirp signals rather than Simple Fast Fourier Transform. In the presence of reverberation often Sonar systems make use of chirp signals for processing for better detection. FrFT are also employed in radars to detect moving targets and objects.

2.8: Existing Hardware methods of Sonar Systems

2.8.1: ASIC (Application specific integrated circuit)

In the context of SONAR system hardware solutions, ASICs (Application-Specific Integrated Circuits) provide a highly specialized approach by offering dedicated circuits tailored specifically to the signal processing needs of the system. For SONAR applications, where high-speed, low-latency processing of acoustic signals is critical, ASICs excel by providing optimized performance for tasks such as beamforming, filtering, and target detection. Their architecture is designed to maximize efficiency by focusing exclusively on these tasks, resulting in significantly reduced power consumption and enhanced operational speed, which is crucial for large-scale, continuous operations in commercial or military SONAR systems.

However, due to their fixed-function nature, ASICs lack the flexibility to accommodate future updates or algorithmic changes once manufactured. This limitation makes them less ideal for applications that require frequent upgrades or new features, particularly in evolving fields like SONAR where processing requirements may change over time. The high design and production costs of ASICs also mean they are best suited for long-term deployments where the performance benefits outweigh the need for reconfigurability. In summary, ASICs offer a high-performance, energy-efficient solution for SONAR systems that prioritize stability and efficiency over adaptability.

2.8.2: ASIP (Application Specific Instruction set Processors)

In the context of SONAR system hardware solutions, ASIPs (Application-Specific Instruction-set Processors) offer a middle ground between the fixed-function efficiency of ASICs and the programmability of general-purpose processors. ASIPs are equipped with customized instruction sets that are specifically tailored to the signal processing tasks required by SONAR systems, such as beamforming, FFT, and target detection. This customization enables high performance while maintaining some programmability, which allows for limited adaptability and updates to the system after deployment, such as incorporating new algorithms or enhancing processing functions.

While ASIPs provide more flexibility than ASICs, they are not as adaptable as FPGAs, and their partial programmability can lead to higher power consumption compared to fully optimized ASICs. However, ASIPs are particularly useful in SONAR systems that require a balance between efficiency and adaptability, such as systems that may need occasional updates but do not require the full reconfigurability offered by FPGAs. ASIPs can handle specific SONAR processing tasks efficiently while still allowing for a moderate degree of future adjustments, making them suitable for systems where high performance and some level of flexibility are both necessary.

2.8.3 FPGAs (Field programmable gate arrays)

In SONAR system hardware solutions, FPGAs (Field-Programmable Gate Arrays) offer a highly flexible and reconfigurable platform, making them a preferred choice for complex signal processing tasks. FPGAs enable the implementation of real-time updates and reconfigurations, allowing algorithms such as FFT, IFFT, and beamforming to be optimized and adjusted even

after deployment. This adaptability is particularly beneficial in SONAR systems that require continuous evolution, such as those used in research and development, or systems deployed in dynamic environments where algorithmic updates are necessary over time.

One of the standout advantages of FPGAs is their ability to perform parallel processing, which makes them ideal for multi-channel SONAR systems that require high-speed, real-time processing of multiple input channels simultaneously. This capability ensures that FPGAs can efficiently handle the large volumes of data generated by SONAR systems and process them in a timely manner for accurate detection, tracking, and localization.

However, this flexibility comes with trade-offs, including higher power consumption compared to the more specialized ASICs, and the need for greater design effort to develop and optimize the hardware architecture and algorithms. Despite these challenges, the combination of programmability, reconfigurability, and high processing power makes FPGAs an optimal solution for SONAR systems that demand frequent updates, scalability, and efficient real-time signal processing capabilities.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The methodology for this research on "Planar Array Signal Processing and FPGA Applications in SONAR Systems" is designed to leverage the capabilities of FPGA technology for efficient and real-time processing of sonar signals. The project is structured around a six-channel system, where each channel processes data streams with a length of 2048 samples. To ensure the system meets real-time operational requirements, the total processing time is constrained to 19 milliseconds. This methodology focuses on the implementation of advanced signal processing algorithms on FPGA hardware, optimizing the performance of planar array system. The design process involves careful consideration of parallelism, data throughput, and computational efficiency to achieve the desired processing speed and accuracy. The approach also incorporates the integration of signal preprocessing steps, array signal processing techniques, and post-processing operations, all tailored to the specific requirements of sonar systems.

3.2 Conversion process of MATLAB code

3.2.1: Implementation of Multi beam Sonar using FPGA

As FPGA is a much better option to run and simulate digital circuits, so we do need to convert MATLAB codes usually into Verilog, to make them appropriate to modeled on Field Programmable Gate Arrays devices. Complex digital circuits and algorithms have an enormous amount of addition and multiplication operations that accelerate the processes needed to execute algorithms. Modern FPGAs have DSP blocks that can process a large number of addition and multiplication operations in no time. The process of conversion of MATLAB codes into Verilog is a little complicated. As MATLAB works in the floating-point format, but FPGA deals with fixed point values. We need to cater to this problem of conversion of floating-point values into fixed numbers so that FPGA configures the results accurately in fixed point. This project mainly has three parts,

I) The first one is to take FFT of desired bins,

II) The second is the beamforming process, and

III) The third and last is to take the IFFT of the previous step results.

Developing a Multi-Beam Sonar System based on Field Programmable Gate Arrays will provide real-time processing and parallel computations of multiplication and addition. We need to optimize our solution as much so that our system uses minimum resources for the implementation. An effective and reliable Sonar System can be implemented by careful consideration of memory

requirements and hardware selection.

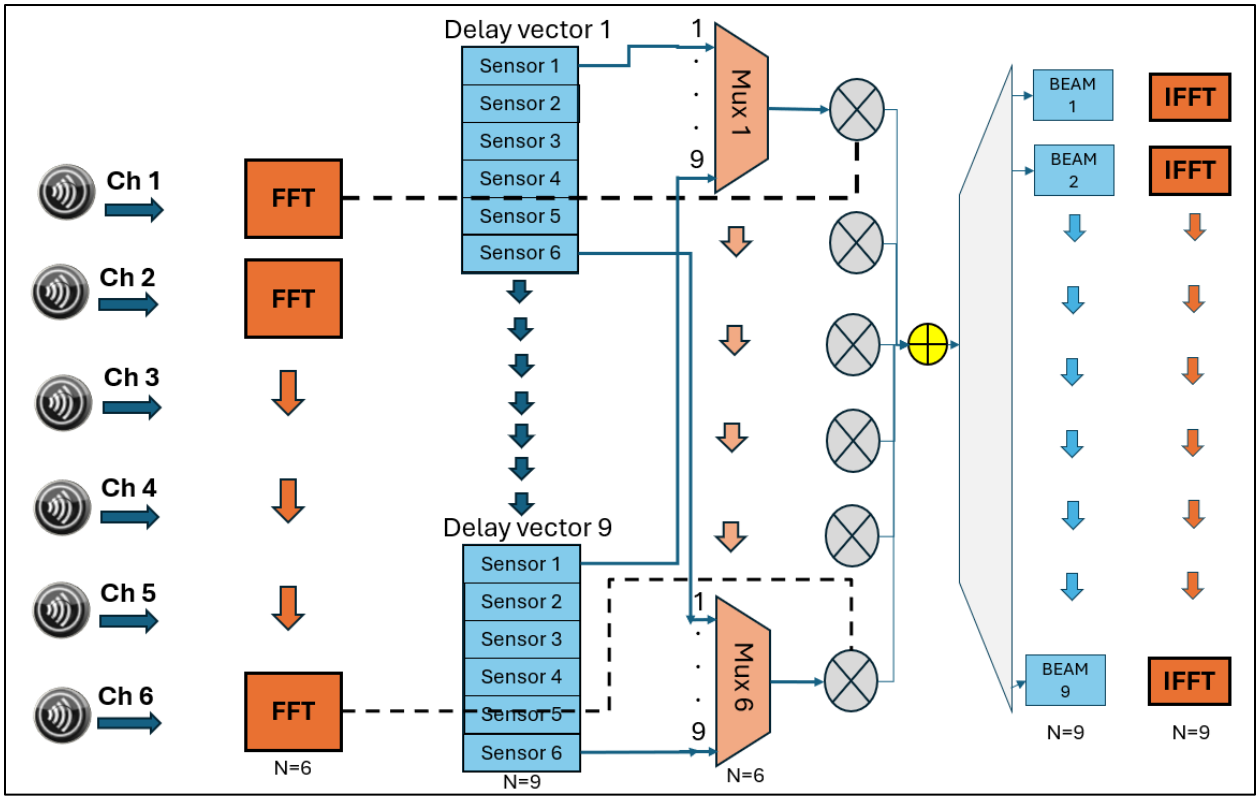

Fig. 3.1 MATLAB Model

3.2.2: Matlab to Verilog workflow

The figure provides an overview of the FPGA architecture designed for the SONAR system, detailing the various blocks integrated into the design. It includes both custom blocks developed in Verilog and IPs such as FFT, IFFT, and beamforming algorithms. Different controllers have been implemented to ensure the smooth and continuous flow of data processing.

In the beamforming section, we utilize delay vectors represented by beamforming coefficients, where each beam corresponds to a specific direction of focus for signal reception or transmission. A total of 9 beams are implemented, with each beam containing 444 coefficients.

These coefficients are stored across 6 ROMs per beam, with each ROM holding 74 coefficients. This results in a delay vector matrix of [9x6x74], where beamforming allows the system to enhance the detection and localization of underwater objects by amplifying signals from target directions and minimizing interference from others.

Multiplexers are used to select the desired beam for multiplication with FFT bins. For example, selecting beam 1 is done by setting the multiplexer control line to '0001,' wiring the beam for the multiplication process. Demultiplexers are employed to route the results from each beam to its corresponding storage location, with 9 RAMs dedicated to storing the results for each beam. This structured approach allows for precise control over beam selection and data flow in the system.

The given MATLAB code consists of 4 codes, in which the step-by-step process of the Sonar system has been accumulated. As discussed earlier there are three main steps in the whole process. Taking The Fast Fourier Transform of the signal, then operation of beamforming is done, and last, signals are converted back into their originated form. Our Sonar system consists of 6 sensors, which means, data is coming from six different sensors forming a single row or column. Usually writing MATLAB code for digital operations or for complex algorithms takes less time as MATLAB provides just simple commands for complex algorithms like for Fast Fourier Transform "FFT" command is used for incoming signals. The input signal is composed of 16 bits. The input length of each signal consists of 2048 samples.

3.2.3: FFT processing using Xilinx IP and Verilog for efficient utilization

In FPGA, FFT algorithm takes complex input in which the most significant half bits are imaginary and the least significant half bits are real parts. So, we consider an input as both a real and imaginary part. Also input to the FFT is of 32 bits, 16 for real, and 16 for imaginary. As MATLAB values are in floating point, so first, the input values we have for signals of 2048 samples will converted to fixed point number, to make them capable of going into the process of FFT. In Verilog, we don't have a simple command for FFT, but we do have a Xilinx IP for FFT in Vivado. The IP is customizable, we can use it to take FFT of 1024 length, 2048 length or length bigger than it or lesser than it. We have also an option to change the bit width of input signals. The Xilinx FFT IP of Vivado consists of many signals evaluating different phases capturing in our input signals. It works like a Master and Slave interface. It acts like a Slave when taking input, while signals act like a Master at that stage. Like While at the output, the core acts like a Master and we are receiving the operated form of signals, so we act like a Slave. We can use more than one FFT IP in our project, but to keep it resource-efficient, we reuse the FFT core, which takes very less resources.

We store our actual inputs of six signals into the Block RAMs. Inputs are kept inside RAMs in the. Coe format. So, we have 6 block Rams before FFT IP in our design to store inputs. Now as our signal original form is analogous, so we need to change it in binary format to make it eligible for the FPGA process. We do take the help of ADC modules in starting of our project design. We have use Verilog language syntax in our design, where needed. So, each input passes through ADC modules first, then they stored in their block Rams. Each input starts their way as with each positive edge of clock. So, we do need to keep them in different states before reaching into a single FFT core. We make six controllers, one to each block Ram. Counters in each controller summed their

rotation to a bigger value as we move from top to bottom in block Ram controllers. First signal input goes straight into FFT core, without goes into any wait state.

All the remaining inputs are under wait at that time. At the time we get the output from FFT of first signal, wait for the second signal input will get over. Now the second signal goes into the FFT core for the processing, and the remaining signal counters are still under the wait condition. Again after, we get the FFT results for the second input, the third input start to go into the FFT core and like this way, the whole process will go on till the last FFT output of channel 6 comes out from the core. To control these controllers, we do use another controller named "big controller" that smooth the process of signal processing.

3.3: Selective FFT bin extraction and beamforming using Verilog

As per the requirement of the project, we don't need the whole FFT results of signals for further processing. From 2048 output FFT length we just need to take out bins ranging from 697-769. So, we make another controller named "bins controller" to do the task of filtering out the desired bins. These bins we are required for beamforming process. So, what we do basically here in beamforming, is to multiply the desired bins of FFT result with the coefficients values we have stored in (ROMS). So, to do the whole process of taking FFT of six different channels by utilizing a single FFT core is much complicated than taking the FFT in MATLAB. It just requires a single command to do the task. Otherwise for filtering bins just require a four-line code to do it, in MATLAB. For beamforming process in MATLAB requires just some commands too for a multiplication process. Now in Verilog it requires a little work to do. Firstly, we store the desired

bins into separate six memories.

3.3.1: Sequential Beamforming with selective FFT bins using Verilog

Bins controller doing the work of bridge for signals from FFT IP core to these memories, where bins will get stored. As per the description of the project there are nine angles from which the data is coming. Practically, we use the term beams for angles. These desired bins will have to be multiplied with each one of beams. So, each beam should consist of six ROMS holding the coefficients, so that it multiplied with its corresponding RAMS of desired bins. We have total of 9 beams, and each beam comprised of 6 ROMS holding the coefficients. In that way bins will be multiplied with each beam one by one. To keep the design resource efficient, we used six complex multipliers for whole multiplication. Angle 1 coefficients first multiplied with desired bins memories, with the help of complex multipliers, then angle 2 coefficients will get under way for the multiplication with desired bins memories. So, angle coefficients multiplied serially, one after one. This way, it takes little time, but we can save much resources, which we can't with parallel computations. To control the multiplication process, we designed one more controller named "multiply-controller", and 6 Multiplexers, for alternately and timely selection for multiplications. Multiply controller vary from 9 different states, in each stage the desired coefficients of a beam get multiplied with the desired bins.

CHAPTER 4

IMPLEMENTATION DETAILS

4.1: Introduction

The implementation phase of the thesis focuses on translating the theoretical concepts and design methodologies discussed in the preceding chapters into a practical, functional system. This chapter details the step-by-step process of implementing a planar array signal processing system within a SONAR application, using Field Programmable Gate Arrays (FPGAs) as the primary platform. Given the inherent complexity and high computational demands of SONAR systems, particularly in real-time signal processing scenarios, FPGAs present a compelling solution. They offer parallel processing capabilities and the flexibility to customize digital circuits for specific tasks, making them an ideal choice for handling the rigorous requirements of multi-beam SONAR systems.

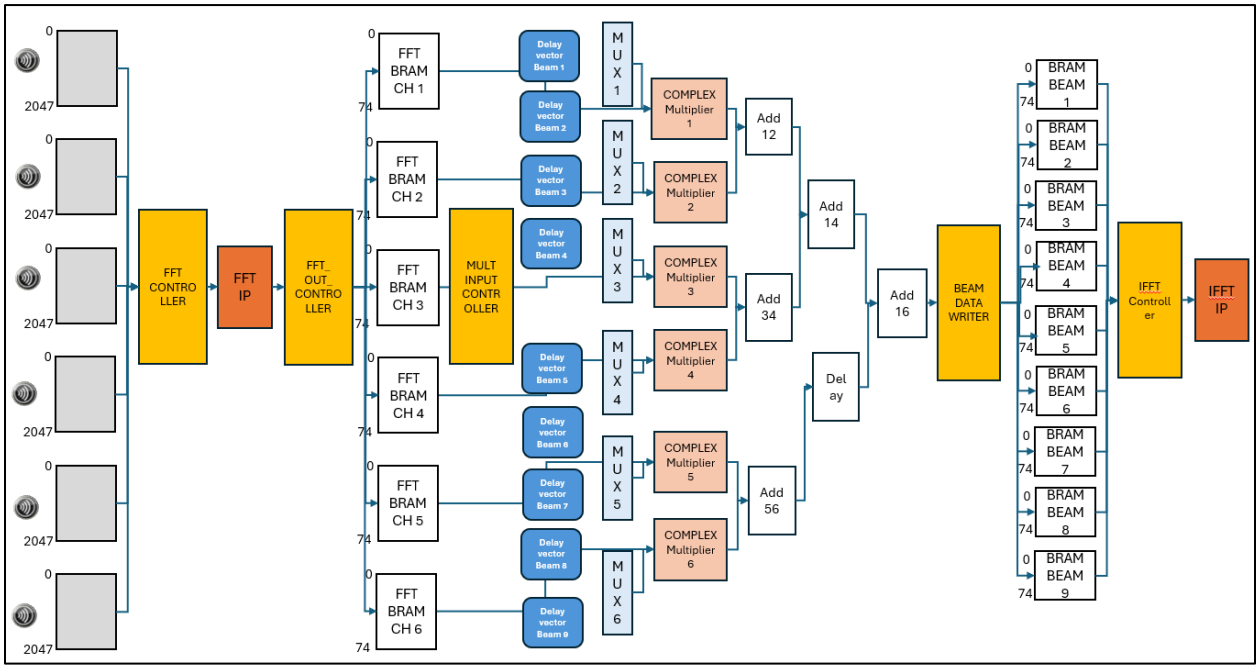

Fig. 4.1 Verilog Code Block diagram.

4.2: System Implementation

The architecture of multi-beam sonar implemented on FPGA may vary depending on the specific project requirements. In this project linear array multi-beam sonar is implemented, in which data is coming from six different channels.

4.2.1: Software Configuration

For each channel, the input data first passes through an Analog digital converter, which converts that signal to a 16-bit digital signal and the length of each signal is 2048 samples. After passing through ADC these 2048 samples are stored in the memories (Block RAM) placed after each channel.

For controlling the writing and reading of data from and into memories we make memory controllers to control the flow of data.

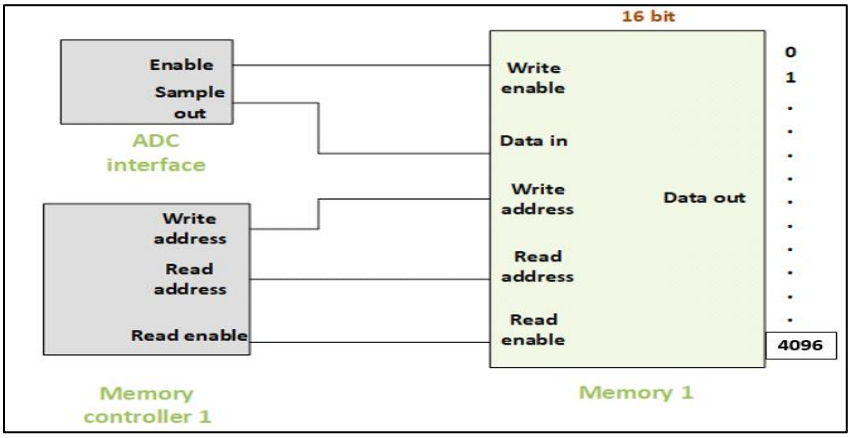

Fig. 4.2 Data Flow

To the continuous flow of data, the length of the memories should be double corresponding to the input sample length which is 4096. While focusing on the first target of the project, we take the FFTs of six channels by utilizing a single FFT IP from Xilinx. To control the input data while taking FFT, there is a controller named the FFT controller. This controller is responsible for the flow of data before reaching the FFT IP. All the memories are filled at the same time with 2048 samples, after this state machine takes the controller to the read state. Memories data out will go directly into their corresponding inputs of the FFT controller. As all the memories have their corresponding data out at the same time, the controller is designed to take the inputs one by one from the memories and go to the FFT IP. There are individual valid for all the outputs of the memories so that the data will come when their corresponding valid becomes high. At the output

of the FFT controller, serial data is coming out that goes directly to the input of the FFT IP. FFT IP takes data serially.

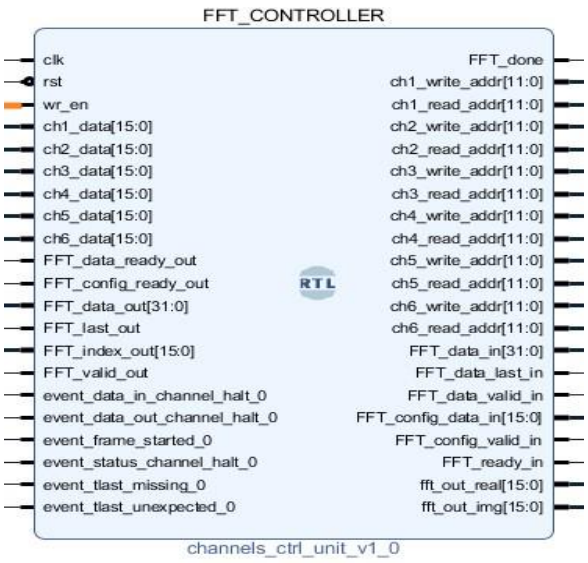

Fig. 4.3 FFT Controller

As our computation time is 19 milliseconds, the memory size should be 4096 locations. To establish the continuous flow of data, data will be out from memories after it is half-filled. The data in memory 1 will go to the read mode when the write addresses reach to 2048 location. It will continue to write data into memory and when the write addresses reach to 4096 location it will again start streaming out data to FFT IP. Each memory is of 16 widths as our input data is of 16 bits and 4096 locations are there in every memory.

4.2.2: FFT Integration

Xilinx provides a variety of IPs (intellectual properties). IPs make work much easier for the users and for the digital designers. They are easier to use as coding requires much time to analyze the design. For fast Fourier transform we set the s_axis_config_tvalid to '1'. Calculating the FFT of different channels requires careful consideration. Setting an appropriate scaling range is the important factor for efficient FFT results of six inputs. S_axis_config_tdata is used to set the range of scaling factors and the choice to take the FFT or IFFT. There are also different events available here in the IP. Events are used to test different scenarios of the data.

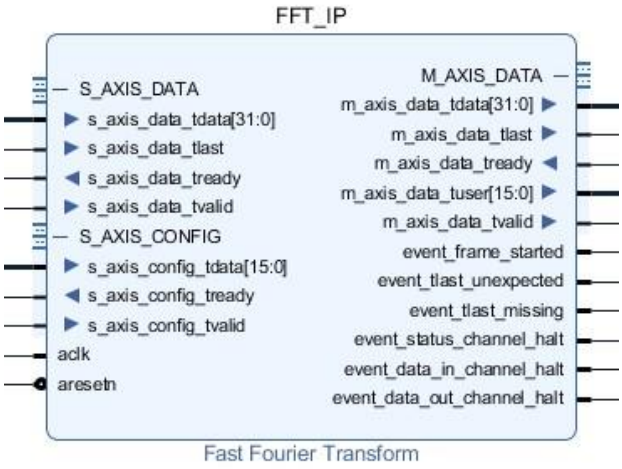

Fig. 4.4 FFT IP Core

The data of channels goes serially into the FFT core, so the memory controllers undergo wait conditions till the transfer of the first channel data and so on. The second channel input enters in FFT core, after we receive the output of the first input. To operational the FFT IP core there are a few settings inside the IP core that you need to address before using it in the project. According to the project requirements, we change the setting of the IP core.

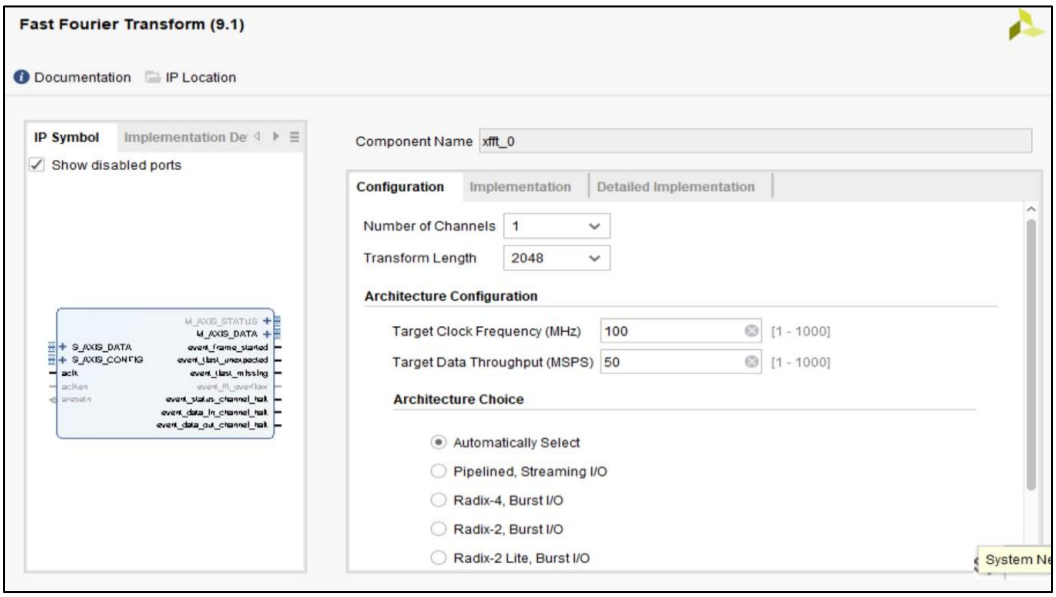

Fig. 4.5 FFT IP Core Settings

The clock frequency is 100 MHz, transform length is 2048 FFT point. Pipelined streaming, I/O is fast but utilizes more hardware resources as compared to Burst I/O which reuses the butterfly but saves resources.

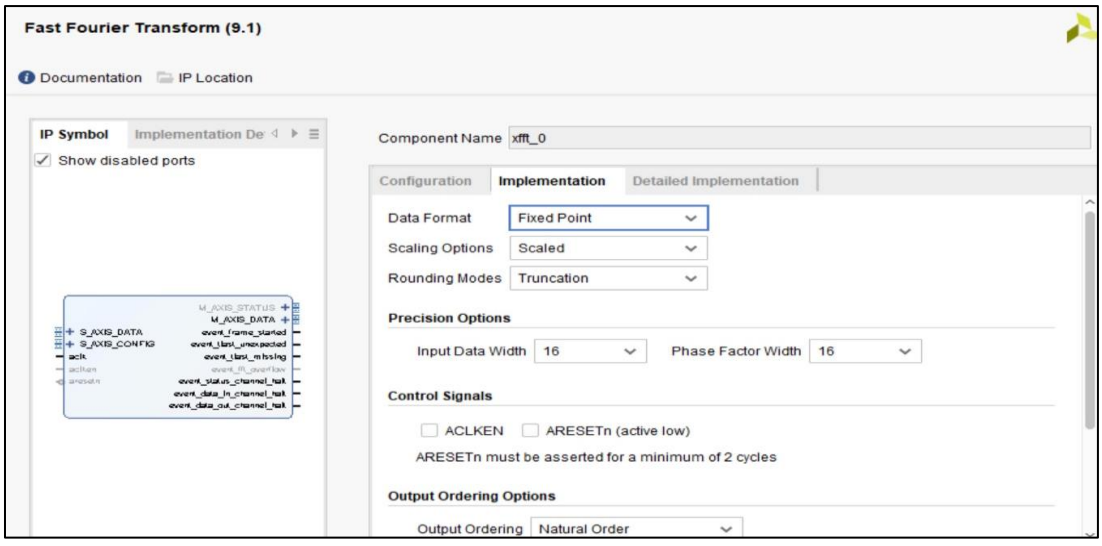

Fig. 4.6 FFT IP Core setting 2.

The input of FFT is a complex number. Before sending the data to the FFT core, we concatenate the input 16 bits with 16 zeros for imaginaries values. As the data is a 32-bit complex number, so input data width is 16, which means 16 bits is for the real part and 16 bits for the imaginary one., which makes 32 bits of total input. Phase factor width is also 16. For scaling options, we select scaled. A proper scaling factor also be chosen according to the input data samples length. The natural order will take more clock cycles than the bit reversed the order and it also takes more Block RAMs but it gives more clarity to the output. If we select the bit reversed order it will take 4 Block RAMs.

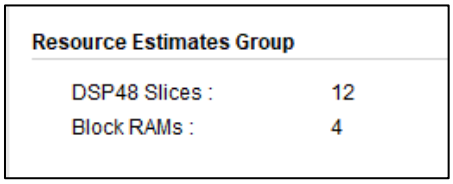

Fig. 4.7 Resources summary with bit reversed order.

Transform Length	Transform Cycles	Latency(µs)
2048	4221	42.210

Fig. 4.8 Implementation details.

For the 2048 transform length, the number of clock cycles it takes is 4221. So, the latency is 42.210 microseconds.

We take FFT as well as IFFT from IP core. If we want to take Fast Fourier Transform the s_axis_config_tvalid value is said to be 0. For Inverse Fast Fourier Transform the value of the s_axis_config_tvalid value should be 1. For IFFT computation the scaling schedule of the data should be chosen unscaled. It will create glitches in the result if it is scaled. The FFT result of the first input frame will take a little bit of time after the last input, it is because of the inner buffer circulating before delivering the result.

4.3: Beamforming Implementation

We need to take the desired bins of FFT results of 2048 samples for further operations. For this, a controller is made named a 'FFT_OUT_CONTROLLER' to take desired bins i.e. 679-769 (corresponds to 38 KHz to 42 KHz frequency) of the FFT result of each channel. These bins are then stored in Block RAMs before the process of beamforming. Six Block RAMs are utilized to store the result of FFTs bins. Now as the data coming from the FFT IP is also serially, we need to take the desired bins of FFT of all channels corresponding to their data. The FFT out controller is designed in such a way that it takes out the desired bins FFTs of the first channel and then waits till the next desired bins are available and so on. This process will carry on until the last possible bins are captured and sent to their corresponding memories.

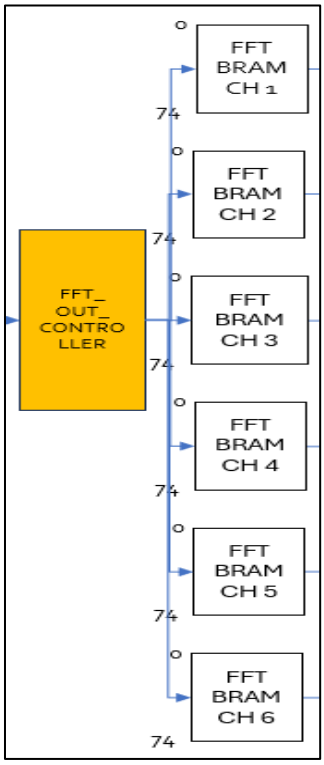

Fig. 4.9 FFT_OUT_CONTROLLER to BRAMs

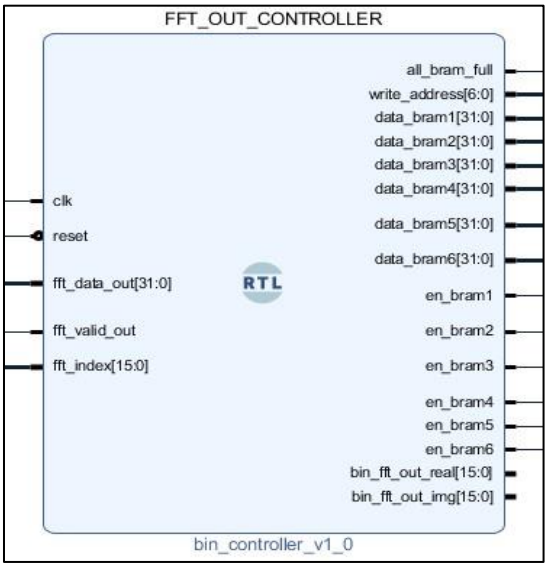

Fig. 4.10 FFT_OUT_CONTROLLER

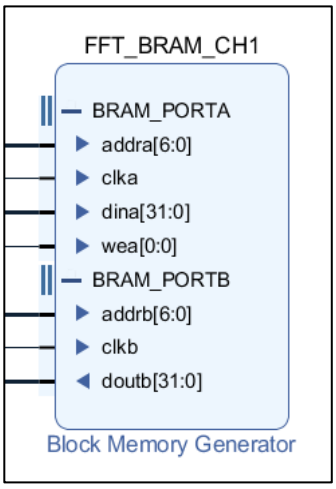

Fig. 4.11 FFT_BRAM IP

The FFT_OUT_CONTROLLER is instantiated with these block RAMs so the data is stored there. The counter in the FFT_OUT_CONTROLLER goes to the next state in every 74 counts. As how the block RAMs will fill down after 444 counts. Various settings of block RAM are there which you can change according to the process need. Standalone mode type is selected and the memory type we take is simple single Port ROM. Next, you can set the ports' width and depth according to your requirements. In our case, we set the width to 32 and the depth to 74 locations.

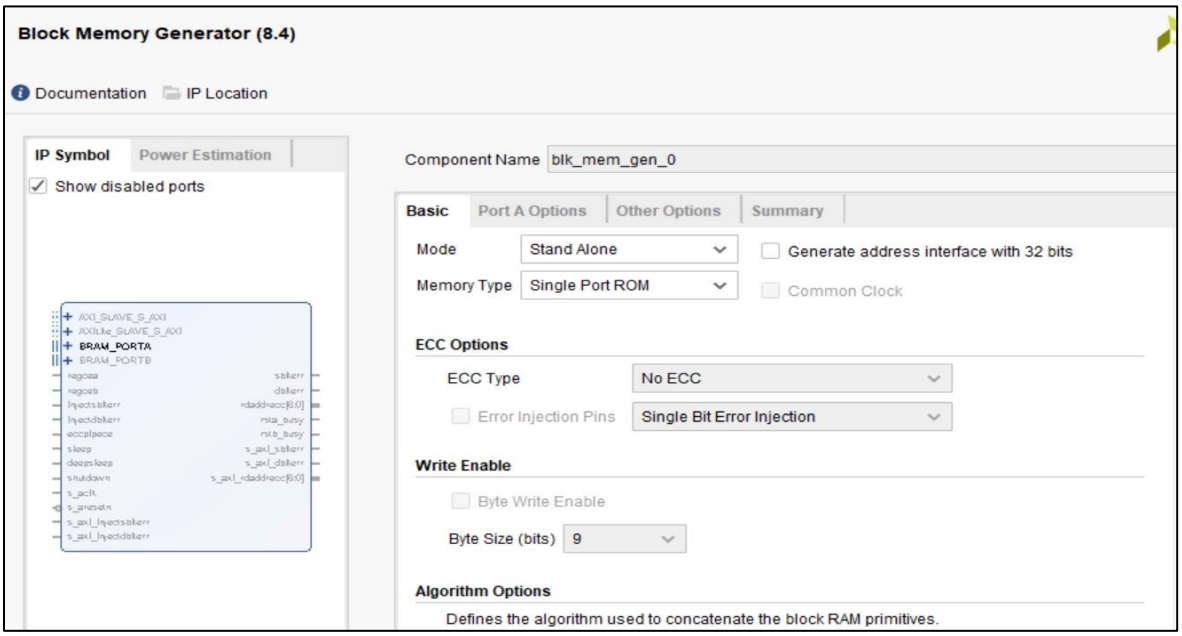

Fig. 4.12 Block RAM IP Detail 1

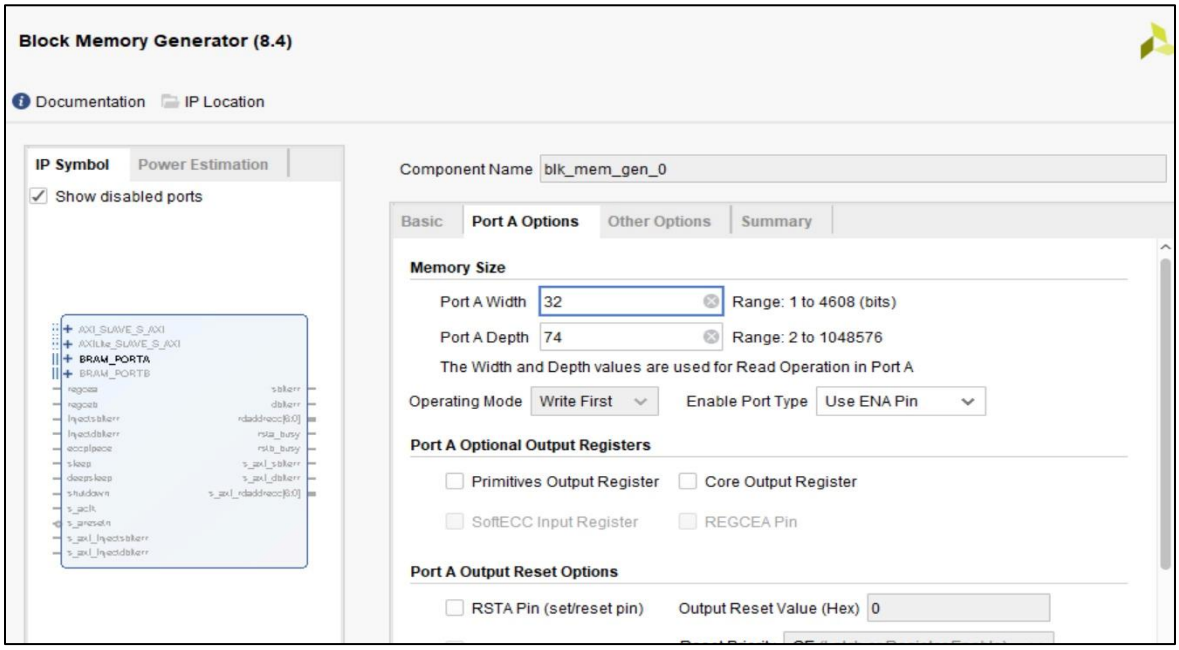

Fig. 4.13 Block RAM IP Detail 2

4.3.1: Implementation of Beamforming Algorithm

For beamforming operation, there are nine beams required in the project. These beams carried coefficients that would be multiplied with the desired FFT bins that is stored in Block RAMs. Each beam is composed of six memories that will be multiplied by the six-channel data at a time and so on. To store the coefficients of the beams we again used simple single Port Block RAM. Each beam utilized six Block RAMs IPs, so a total of 54 block RAMs IPs is used to store coefficients in the beamforming process. A multiplier controller (MULT_INPUT_CONTROLLER) is there to control the whole multiplication process. When the last FFT bin memory is filled fully by the input data, it will generate a tick signal that goes to the input of the multiplier controller. This enables the read valid of both memories and the counter in the controller starts counting the addresses that go to the read addresses of the memories. We take the help of 6 multiplexers to select the coefficient values for multiplication. There are nine total states in a multiplier controller that cover the multiplication of all the beams with the FFT bins. Each MUX is composed of 9 inputs. While in the first state of a multiplier controller, select all the first inputs of the 6 muxes. In this way, it selects the first whole beam for further process to be multiplied with the FFT bins. Same way the second state, selects all the second inputs of the 6 muxes, the same way the whole second beam will be selected for the multiplication, and so on.

For multiplication, we used a complex multiplier IP from Vivado. A total of six multipliers are used for the multiplication process. The output of the FFT bins memories goes into one input of the multiplier, and the output of the muxes will go into the second input of the multiplier. Muxes are used to select the values of the beams.

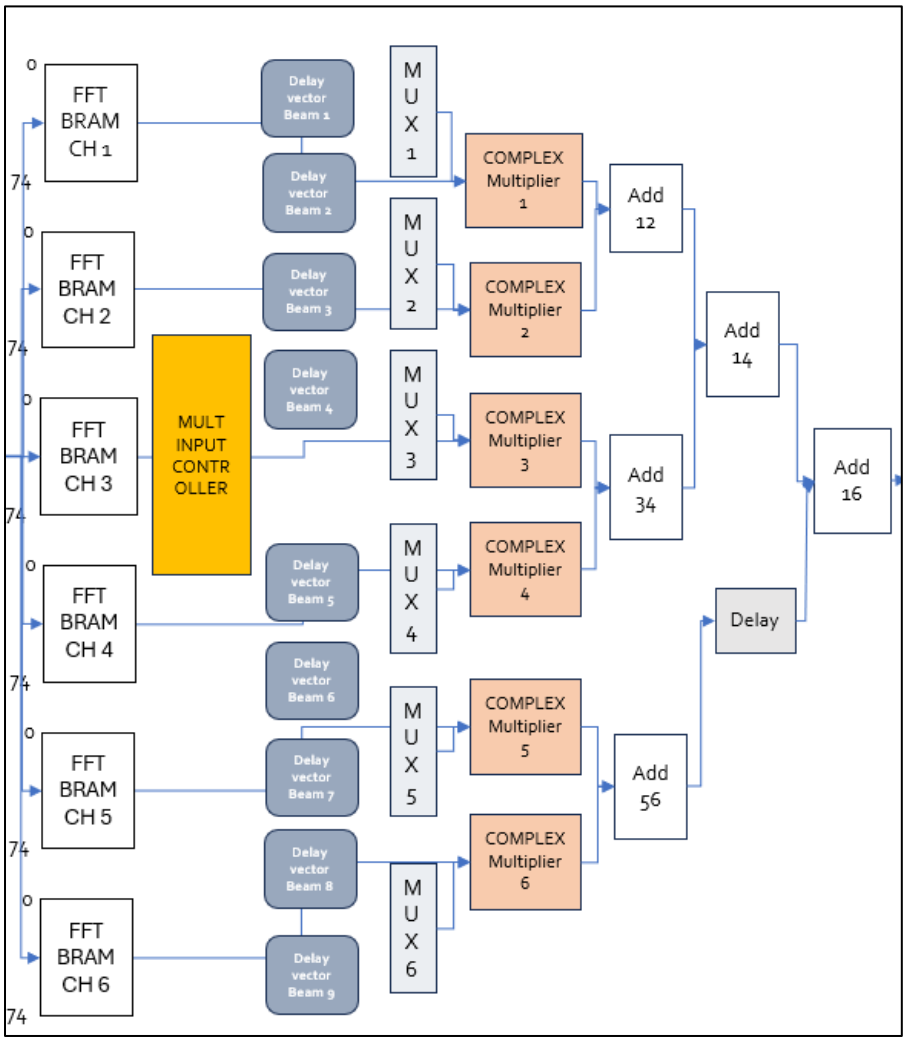

Fig. 4.14 Beam formation.

FFT bins and beams coefficients (stored in delay vector beam) should come to multipliers input at the same time. A common enable is used for the bins output and coefficients output to ensure the right flow of data to the multiplication. So, to control the flow of reading from both memories the right event should be necessary. In this way, the multipliers get the input data in a sequence.

The first six memories of beam 1 will be multiplied by the six memories of the channels FFT data, and the results will be added up together forming an array of [n ....…… 74]. Likewise, 9 different arrays will be generated. The multiplication results are added up together by using an adder IP from Xilinx. We are having 9 added arrays at the end. After the addition, the data will be stored again in Block RAMs (BRAM BEAM). To control the flow of data after the addition process there is another controller (BEAM DATA WRITER) to place the data to the 9 Block RAMs (BRAM BEAM). Each beam multiplication result will be stored in its corresponding BRAM. The multiplier valid signal goes into the controller input. The controller starts operating the flow after it receives a valid signal. The state of the controller goes to the next state after each memory is filled up with the data.

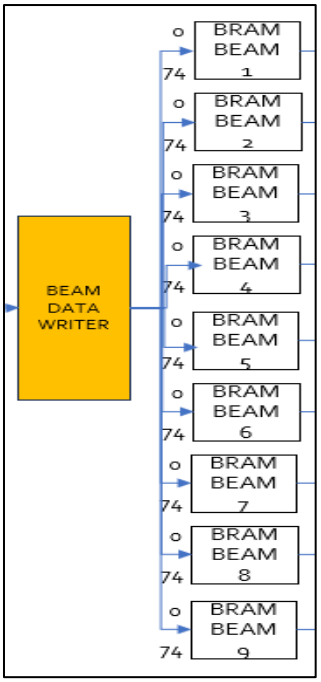

Fig. 4.15 Beam data writer

4.4: Inverse Fast Fourier Transform Implementation

Now in the third section of the design, we are required to do the baseband process. In the baseband, we take the IFFT of the data. But the problem is we have only desired bins of 74 lengths by which we cannot take the inverse Fast Fourier transform. For IFFT, the Data will be of proper length as it is before the FFT operation. So, to take the IFFT, we have to convert the data length to 2048 again. To do this we made another controller named as “IFFT CONTROLLER”. At every state, the data out of the controller is set to 0 till the counter reaches the 695 value. From 696 to 769 values it takes data from block RAMs and then again it puts zeros to the data till the counter again reaches 2048 length. As how it goes to the IP to take Inverse Fast Fourier Transform.

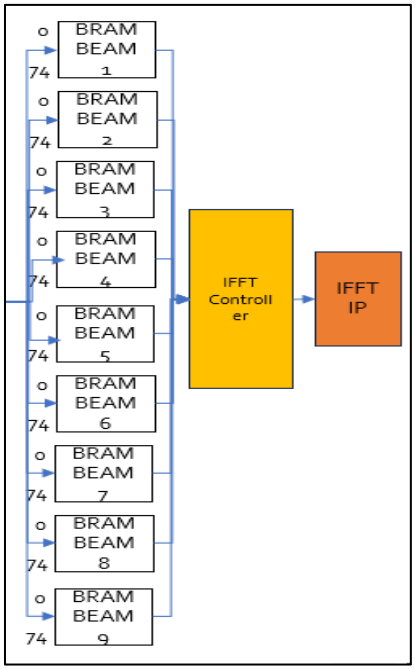

Fig. 4.16 IFFT Controller

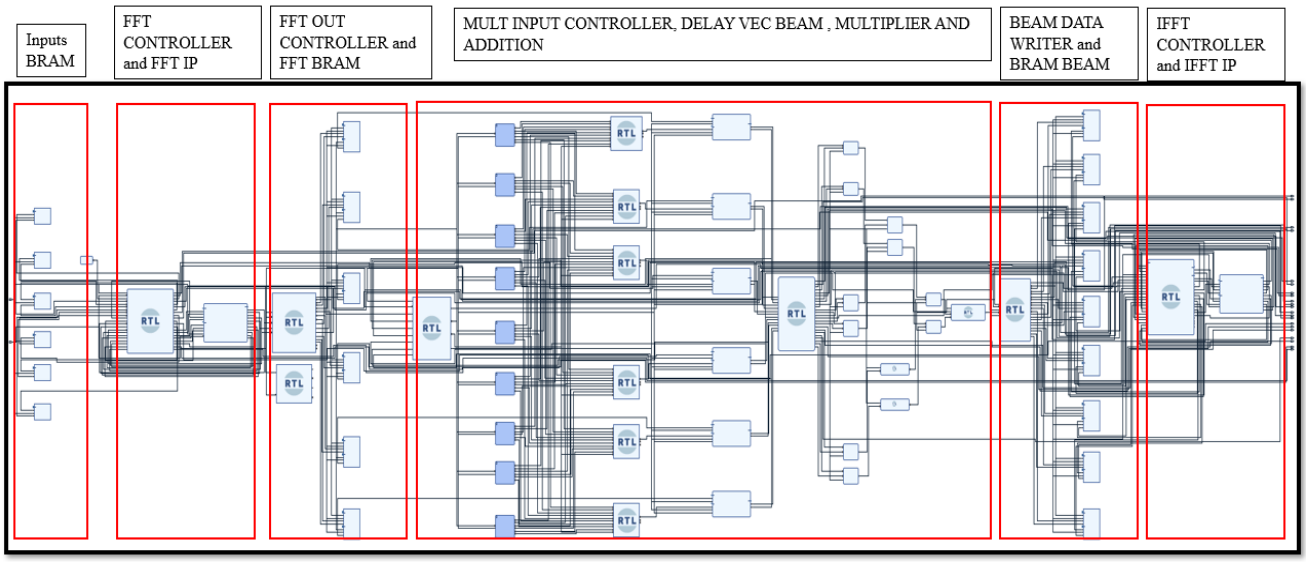

Fig. 4.17 Block Design Model

That is the whole block design of the project by using Vivado. All the IPs and the RTL blocks are added to the design to make a full design as per requirement.

CHAPTER 5

EVALUATION RESULTS

5.1 Results

Result is analyzed and compare with MATLAB reference model at three different stages.

1) After taking FFT
2) After Multiplication
3) After taking IFFT

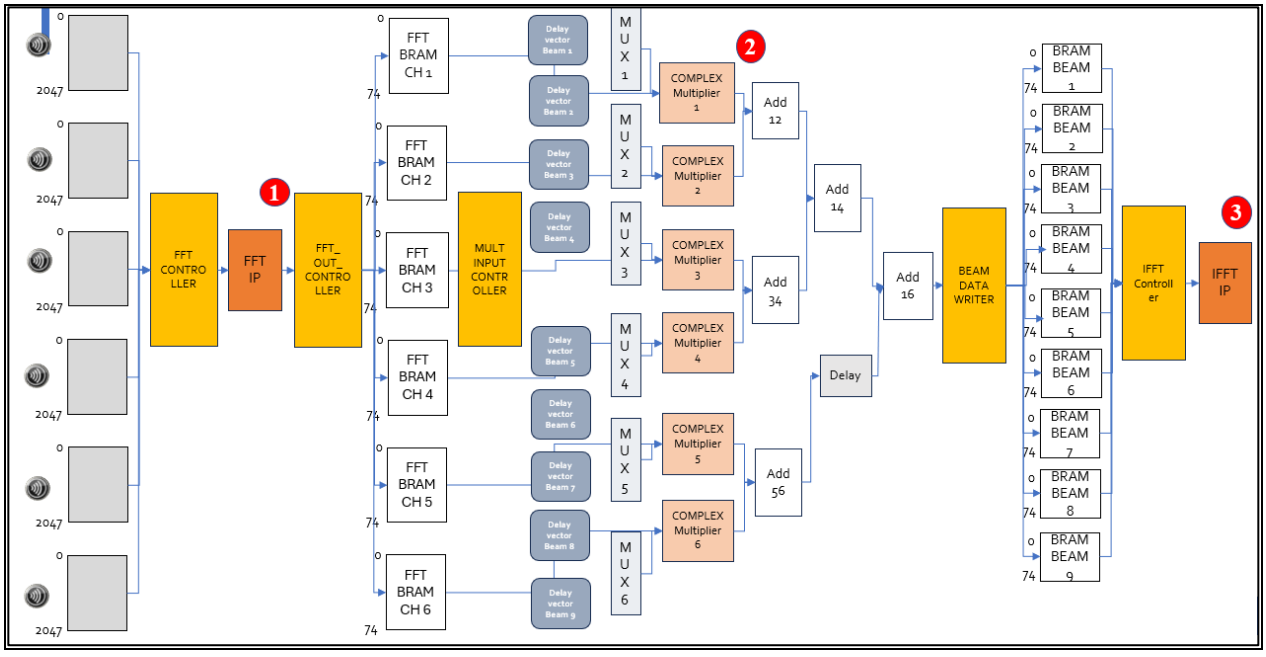

Fig. 5.1 Result comparison stages

Values in Vivado are stored in a text file, which is subsequently imported into MATLAB. These values are initially in fixed-point format. Therefore, they undergo conversion to floating-point format before being compared with the MATLAB values.

In the FFT output, both MATLAB and FPGA values match, as depicted in figures 27 and 28. However, after multiplication, there is a variance between the MATLAB and FPGA values, illustrated in figures 45. This discrepancy, amounting to 10^{-5}, falls within an acceptable range. The output after the IFFT matches with max error of 0.03658 as shown in figure 46.

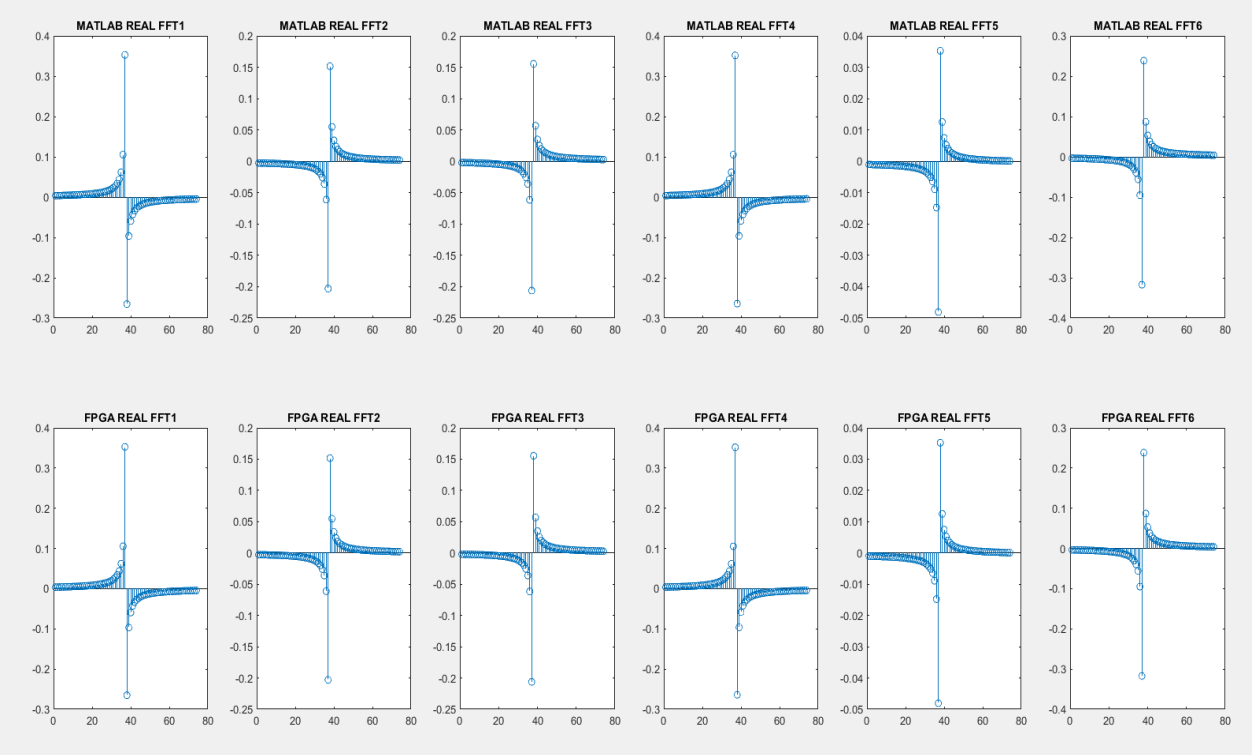

Fig. 5.2 FFT output result comparison (Real values)

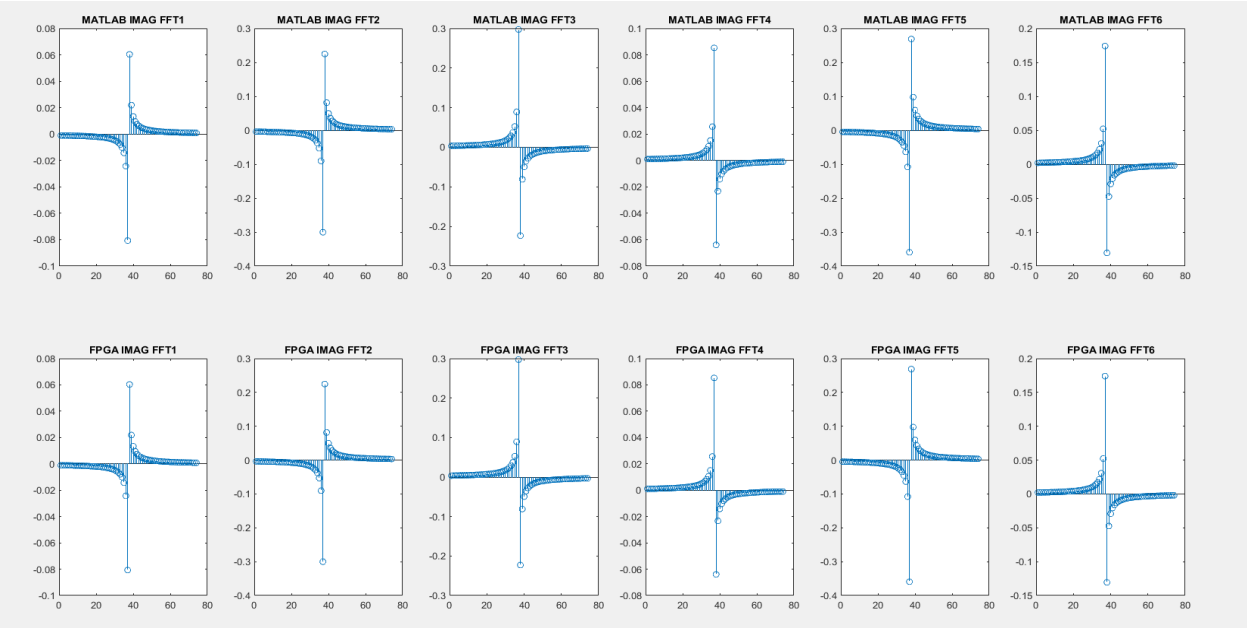

Fig. 5.3 FFT output result comparison (Imaginary values)

To validate the accuracy of the results obtained from the FPGA, it is essential to compare the outcomes from both MATLAB and the FPGA. The primary challenge in this comparison lies in the difference in numerical formats used by the two systems: MATLAB operates with floating-point numbers, while the FPGA provides results in a fixed-point number system.

Since we are dealing with complex numbers, our output consists of both real and imaginary components. To ensure a thorough comparison, we examine the real and imaginary parts separately. Specifically, we compare the real components of the FPGA output with those from MATLAB, and then do the same for the imaginary components. In the accompanying figure, the results from both MATLAB and the FPGA are displayed, showing a close agreement between the two. The observed discrepancies are minor and fall below the acceptable error threshold.

To facilitate this comparison, the FPGA results are stored in files generated using Vivado. These files are then imported into MATLAB for further analysis. In MATLAB, we plot the results, which provides a clearer visual representation and aids in analyzing the outcomes.

For the MATLAB results, we first extract the real components from the complex output and convert them into a fixed-point format. Subsequently, we visualize the data using graphical representations, enhancing our ability to analyze and understand the results effectively.

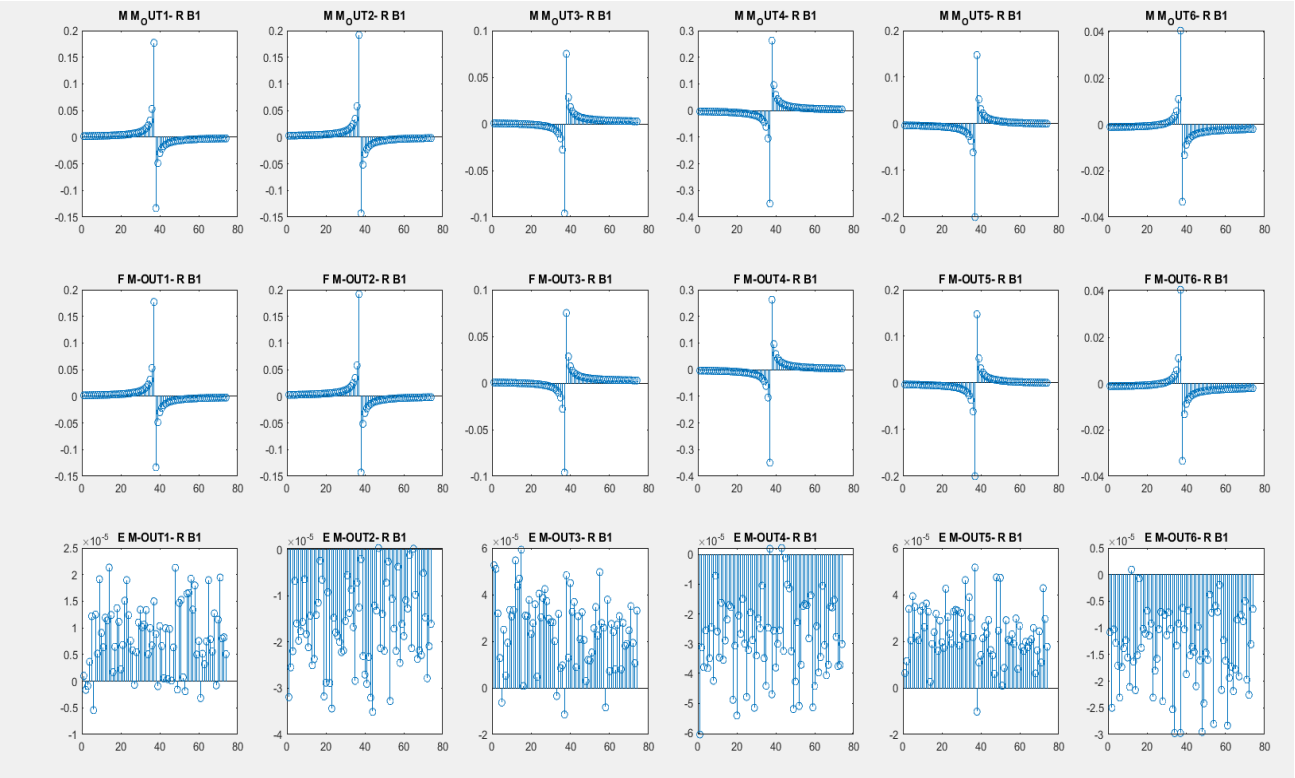

Fig. 5.4 Multiplication results comparison Real of BEAM1

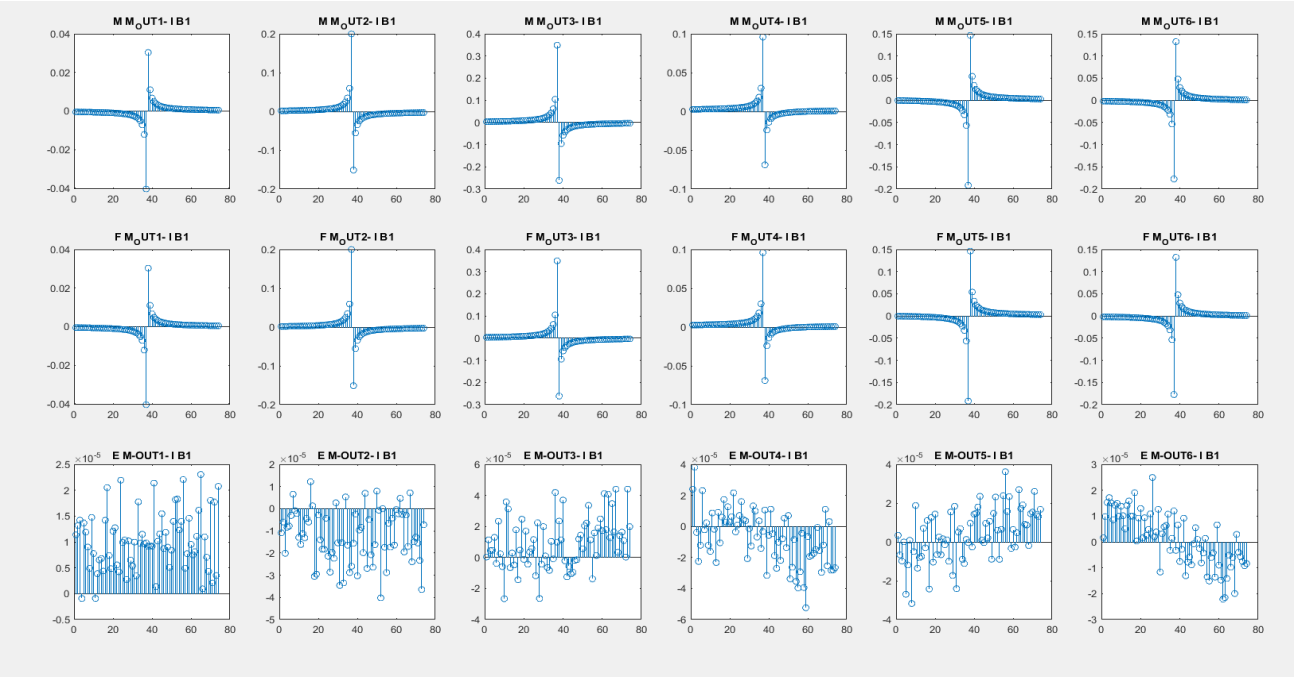

Fig. 5.5 Multiplications results comparison imaginary of BEAM1

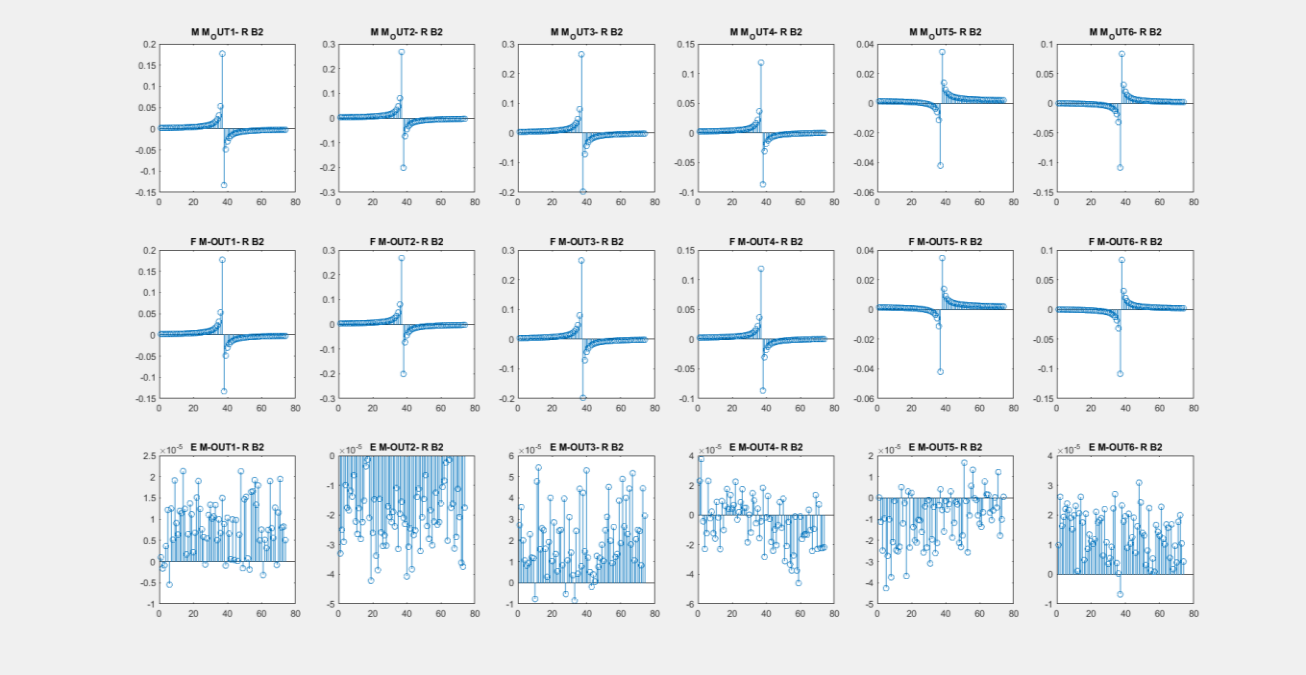

Fig. 5.6 Multiplications result comparison real of BEAM2

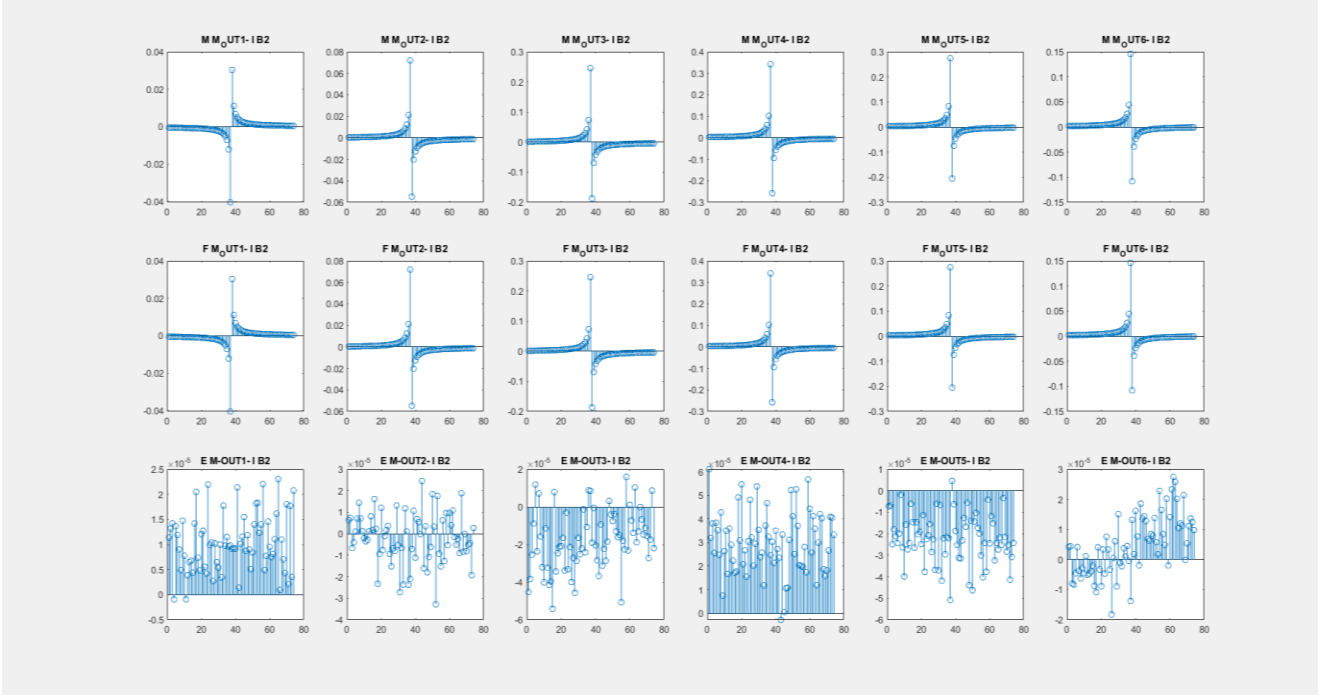

Fig. 5.7 Multiplications result comparison Imaginary of BEAM2

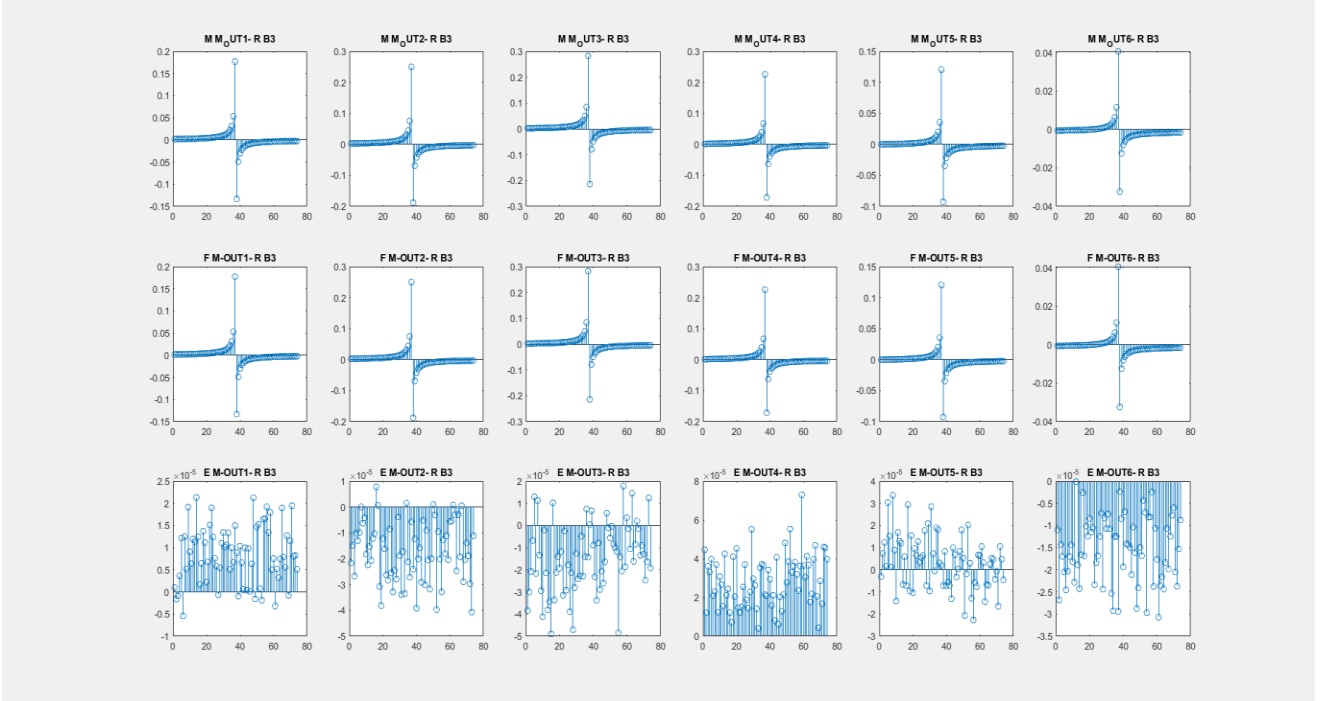

Fig. 5.8 Multiplications result comparison Real of BEAM 3

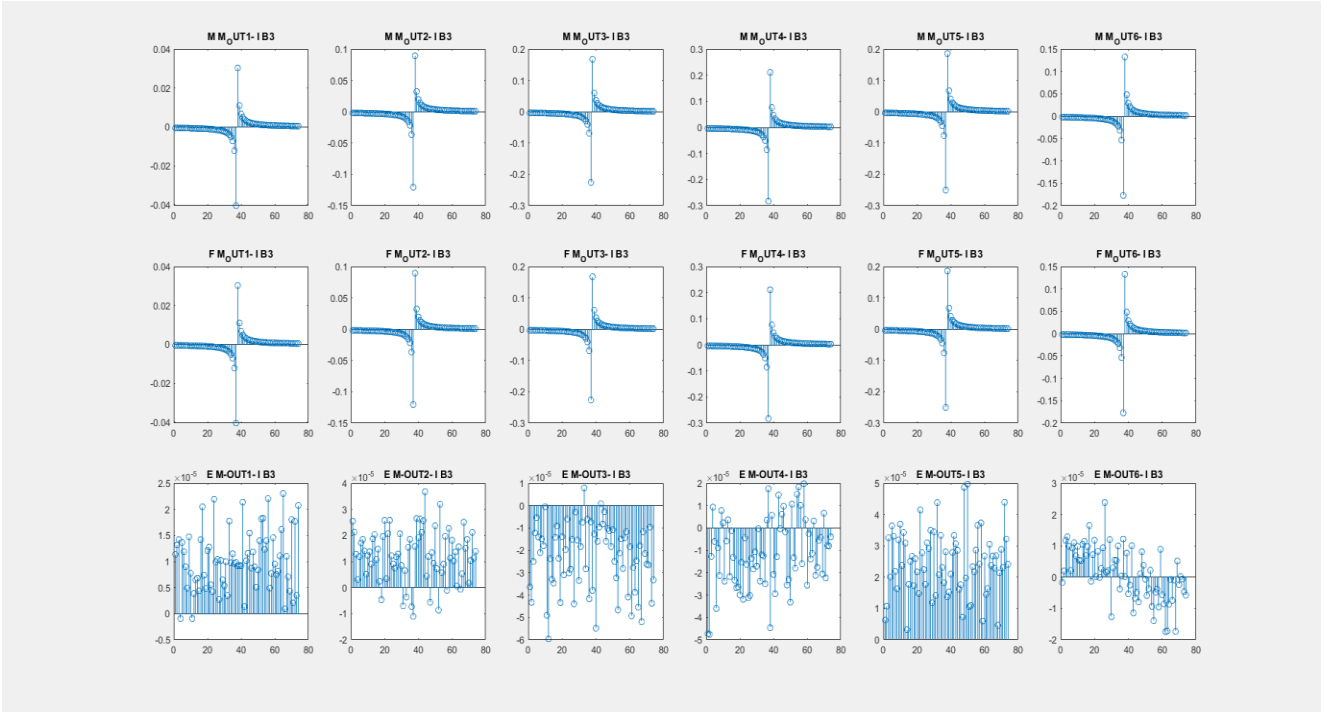

Fig. 5.9 Multiplications result comparison Imaginary of BEAM3

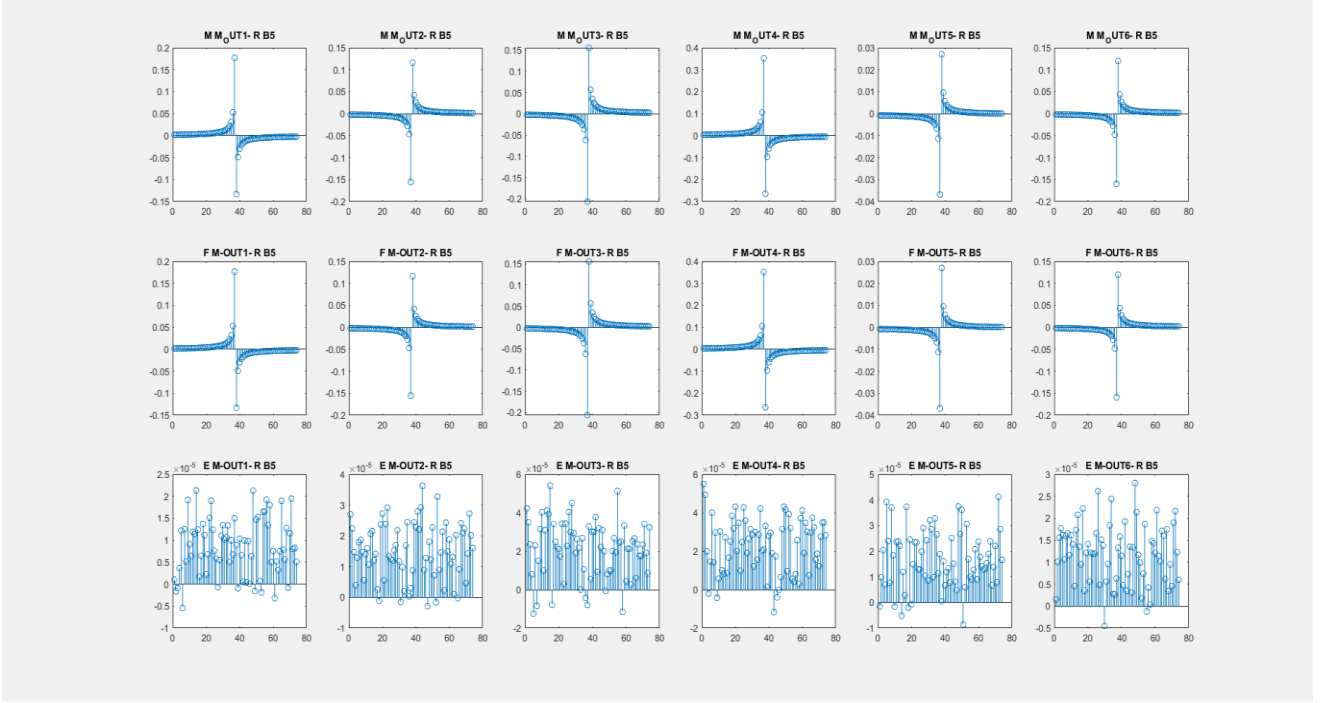

Fig. 5.10 Multiplications result comparison real of BEAM 5

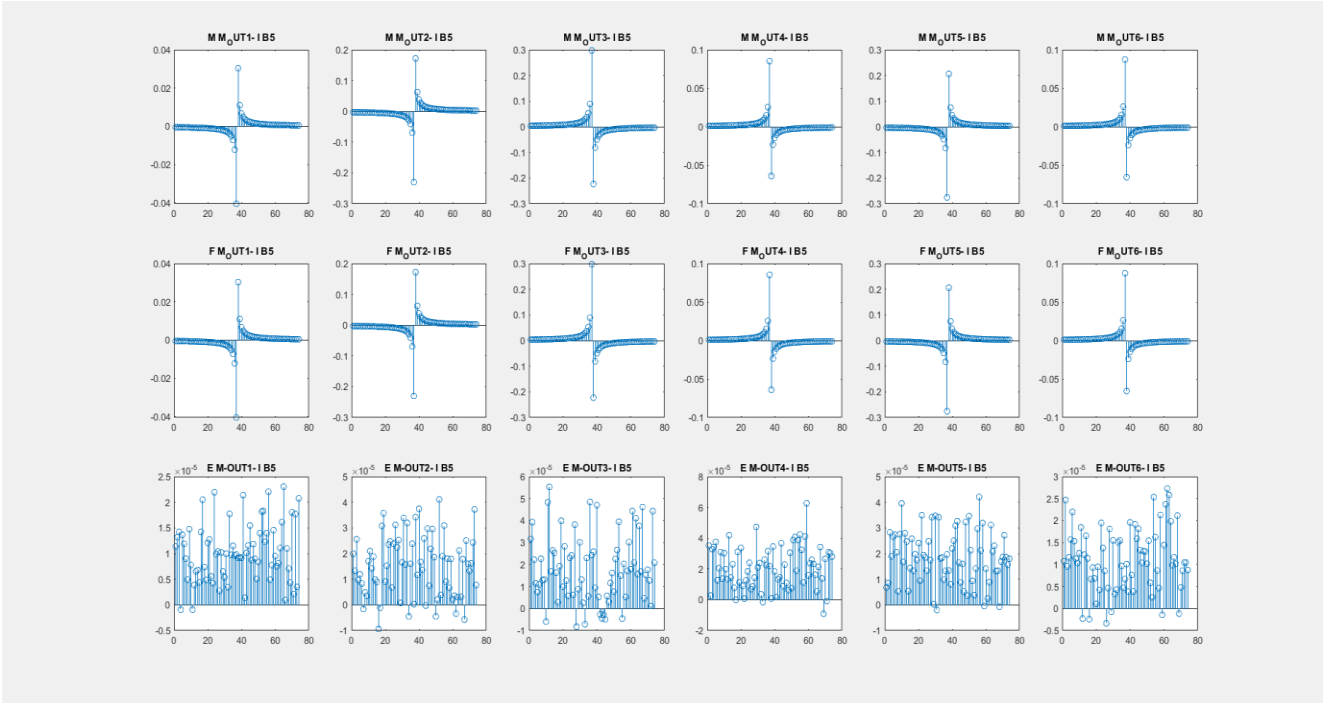

Fig. 5.11 Multiplications result comparison Imaginary of BEAM 5

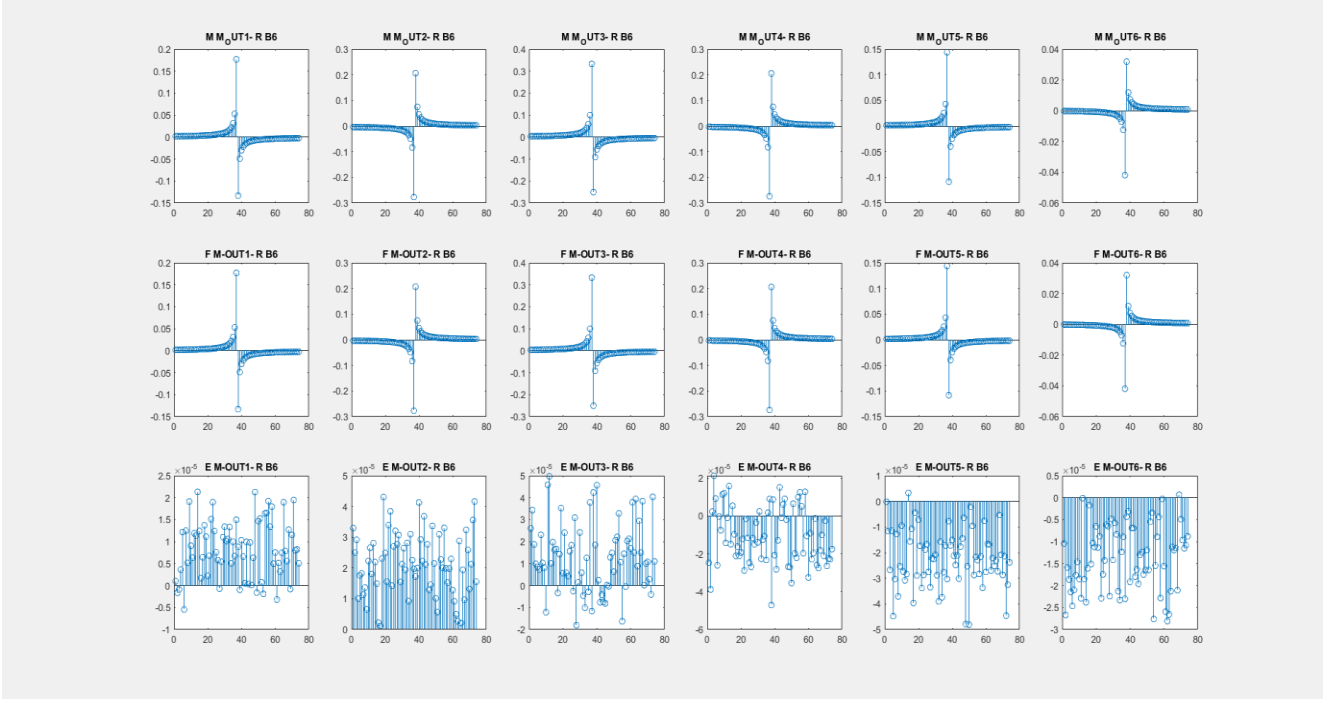

Fig. 5.12 Multiplications result comparison real of BEAM 6

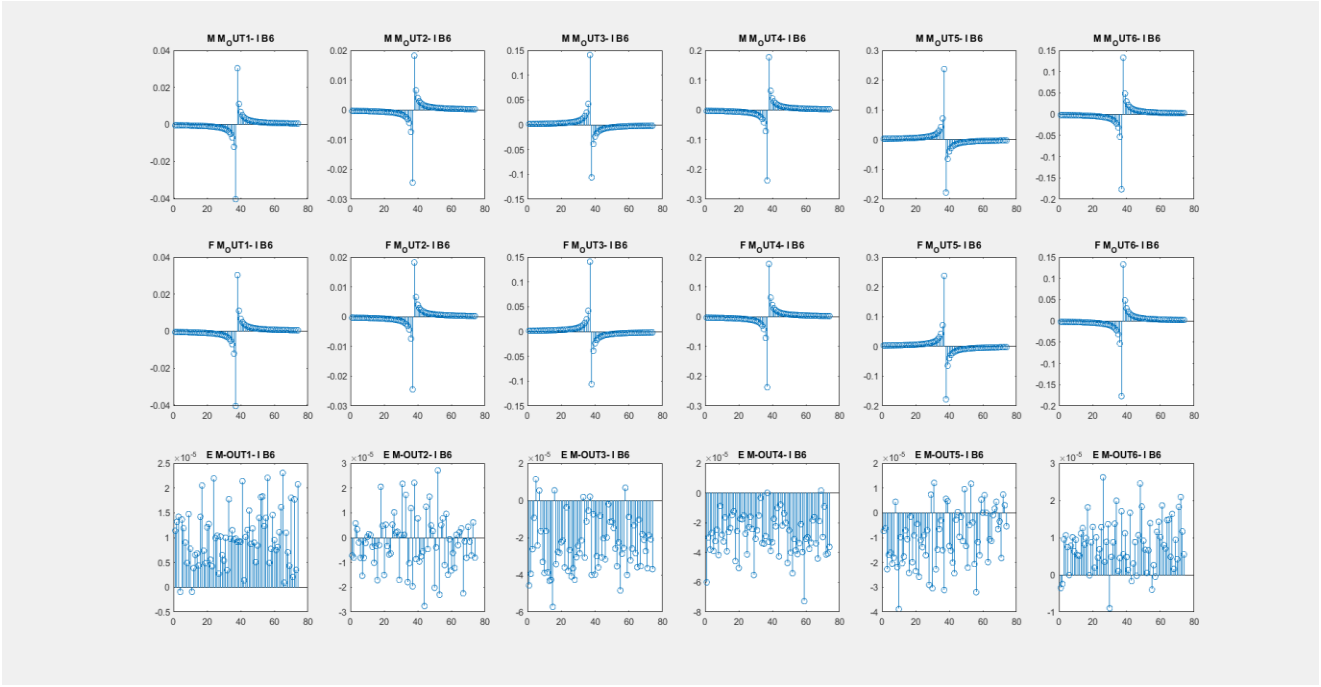

Fig. 5.13 Multiplications result comparison Imaginary of BEAM 6

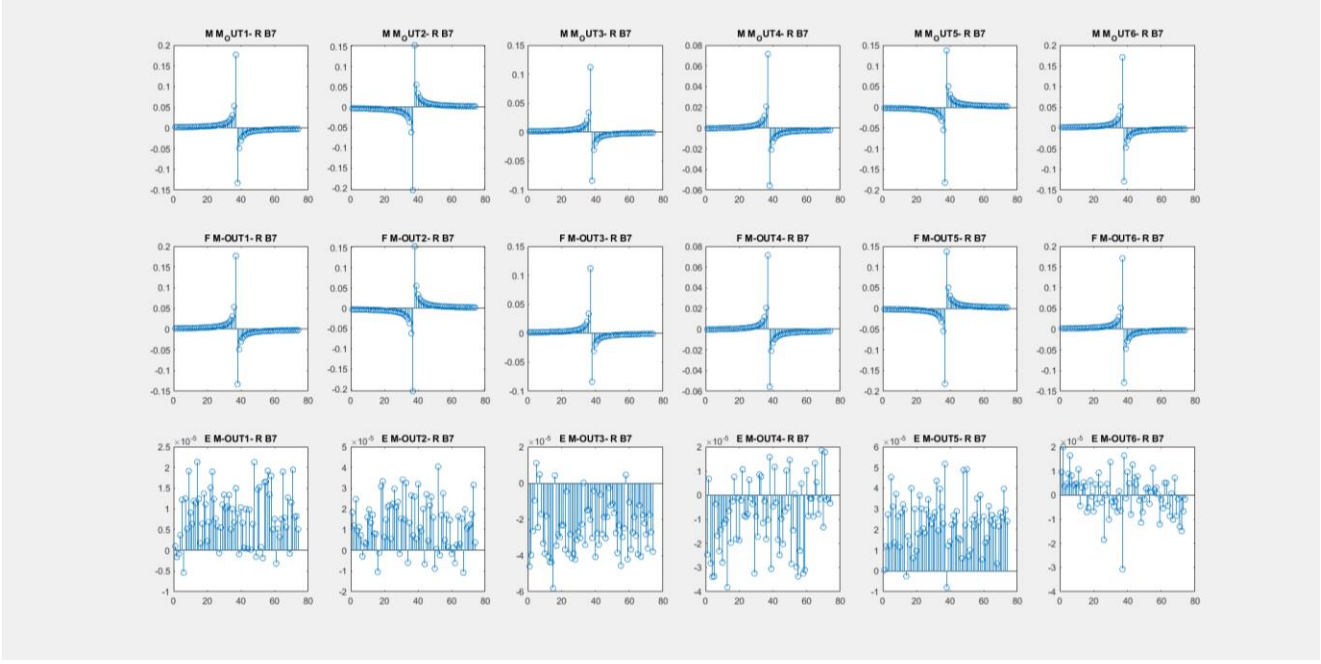

Fig. 5.14 Multiplications result comparison real of BEAM 7

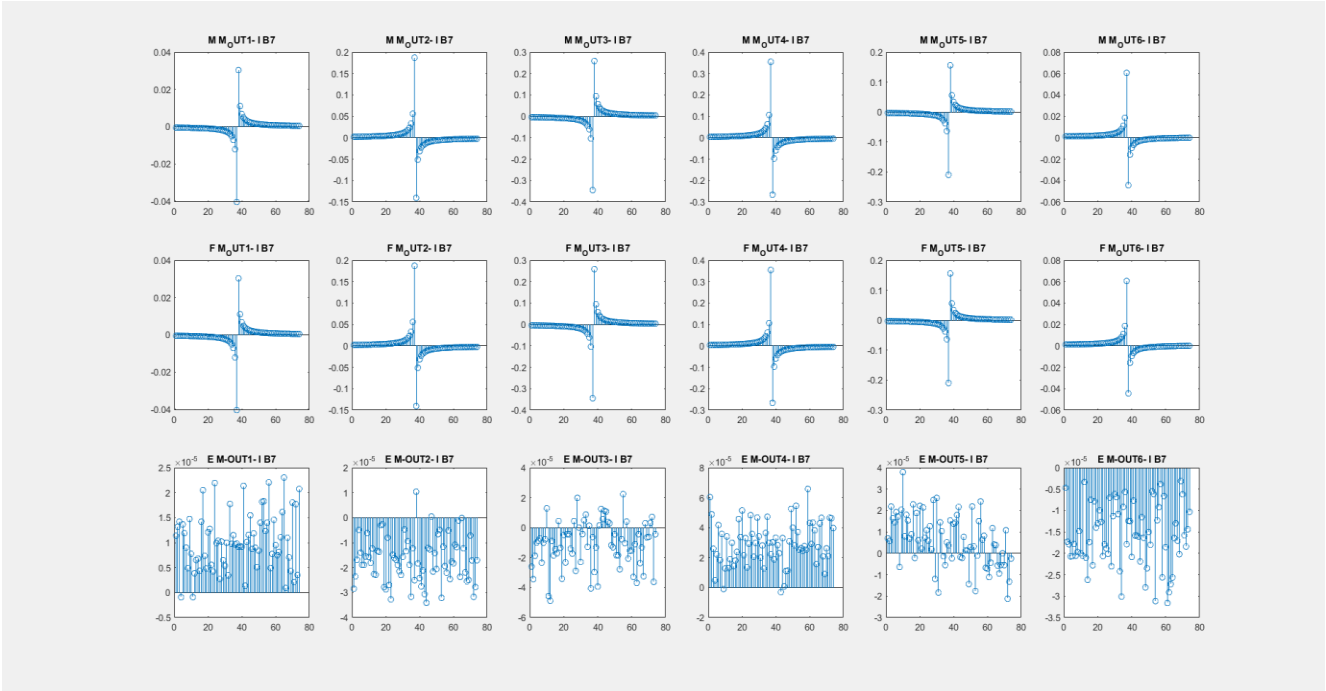

Fig. 5.15 Multiplications result comparison Imaginary of BEAM 7

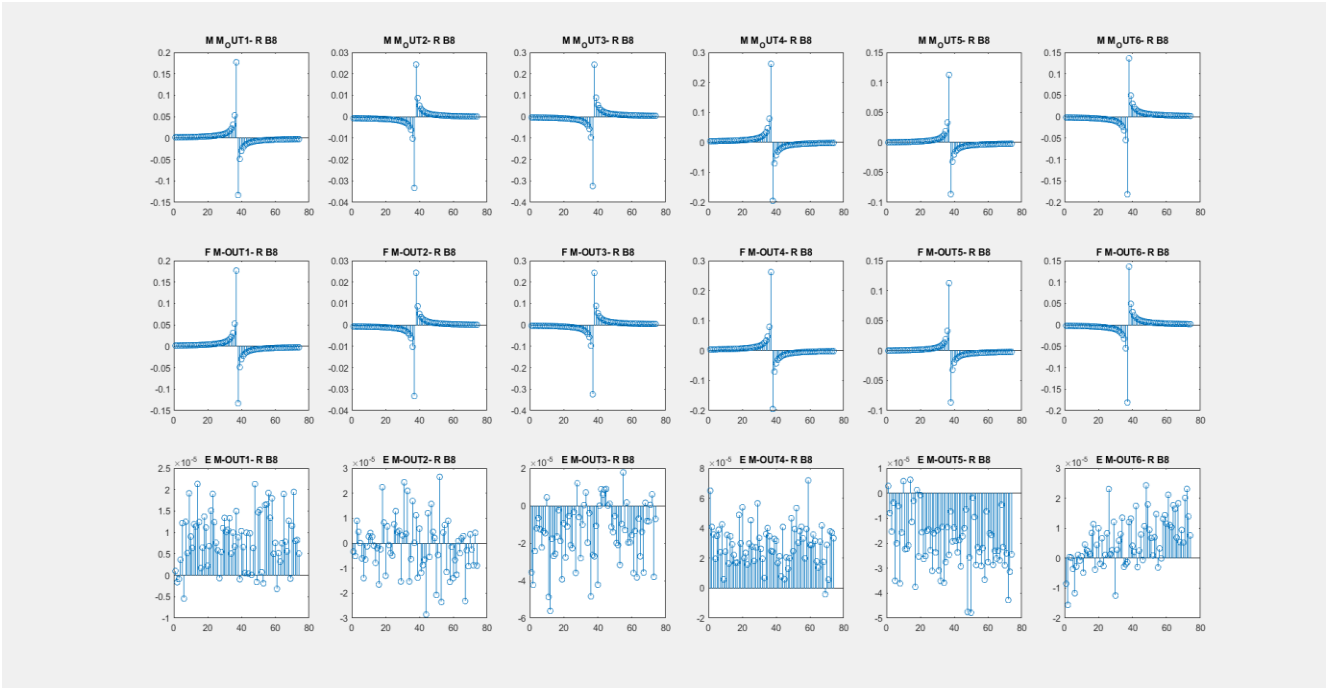

Fig. 5.16 Multiplications result comparison real of BEAM 8

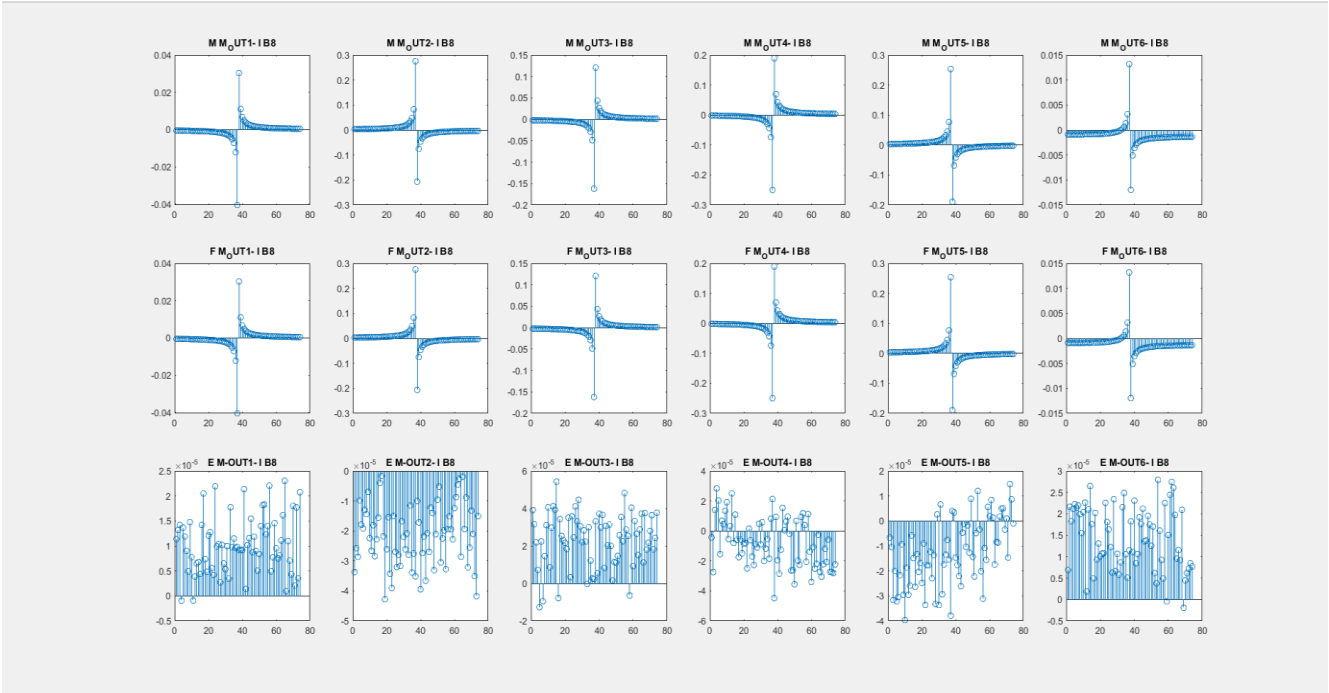

Fig. 5.17 Multiplications result comparison imaginary of BEAM 8

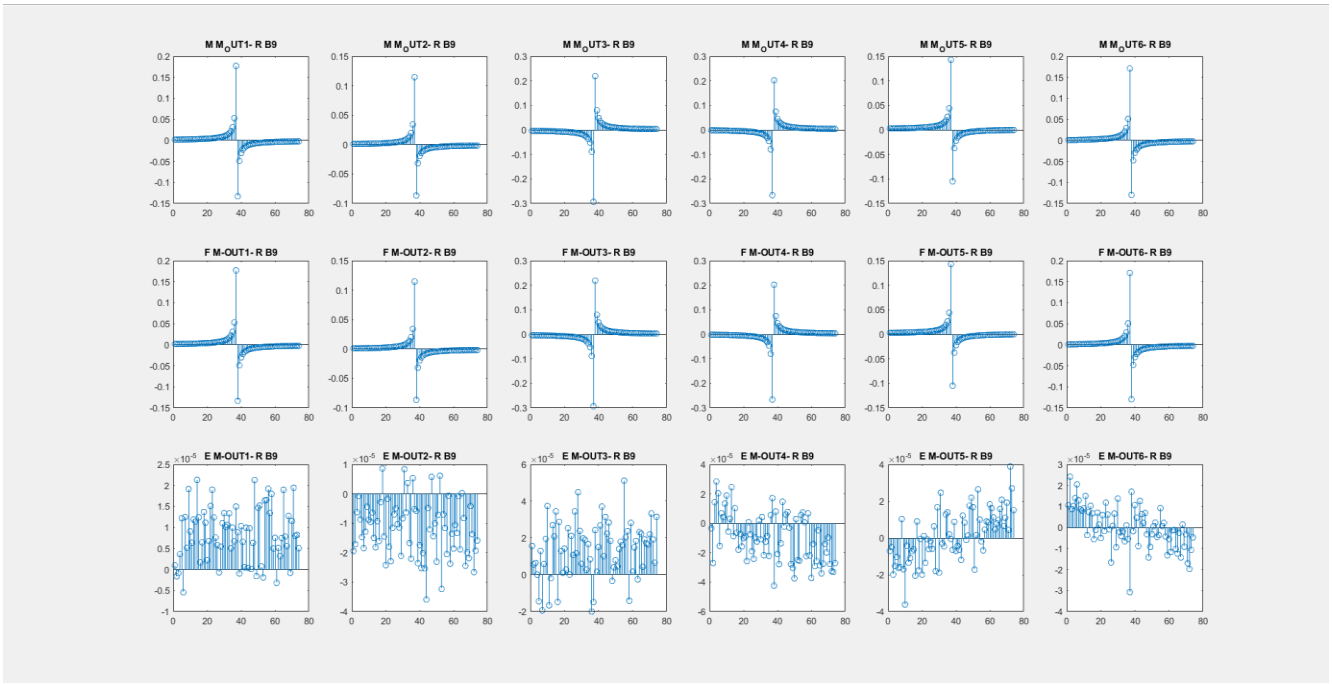

Fig. 5.18 Multiplications result comparison real of BEAM 9

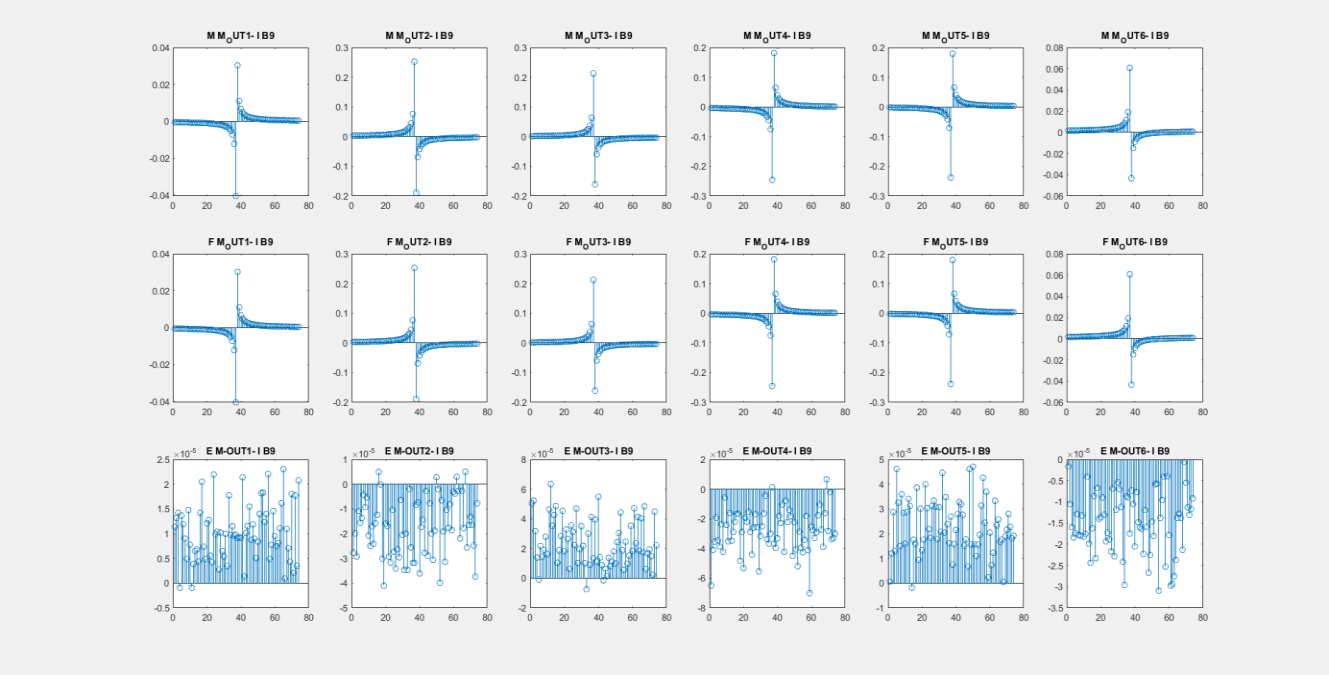

Fig. 5.19 Multiplications result comparison Imaginary of BEAM 9

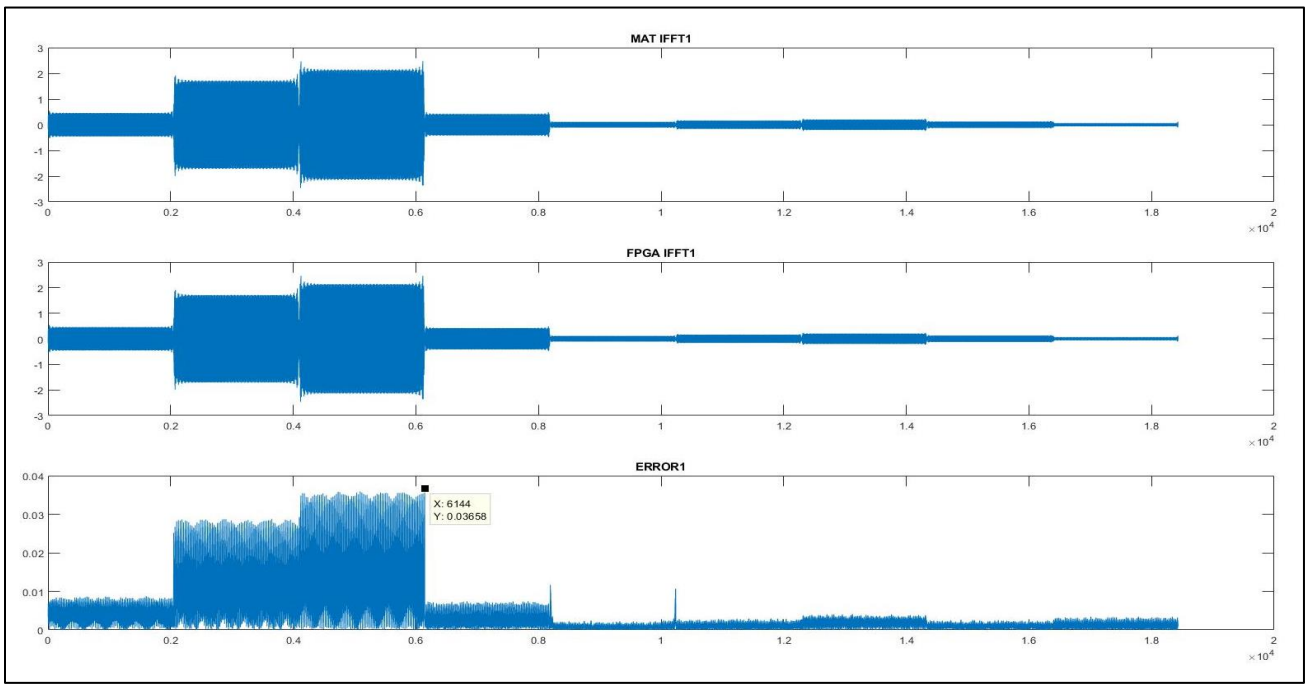

Fig. 5.20 IFFT Results comparison IFFT: MATLAB vs FPGA

The graph above illustrates the final results of the Inverse Fast Fourier Transform (IFFT) as computed by MATLAB and the FPGA. A minimal error is observed, which falls well within the acceptable range, indicating that the outputs from both systems are in close agreement.

This consistency between MATLAB and FPGA results confirms the reliability of the FPGA's performance in handling IFFT computations. By comparing the outputs from these two different systems, we have demonstrated that the differences are negligible, underscoring the accuracy of the FPGA's fixed-point calculations compared to MATLAB's floating-point operations. The slight discrepancies observed are attributed to the inherent differences in numerical precision and representation between floating-point and fixed-point systems, yet they remain minimal and do not impact the overall validity of the results.

This level of accuracy is critical for applications that rely on precise signal processing, as even minor errors can significantly affect performance. The close match between MATLAB and FPGA outputs suggests that the FPGA implementation is robust and capable of delivering high-precision results, making it a suitable choice for complex signal processing tasks where efficiency and accuracy are paramount.

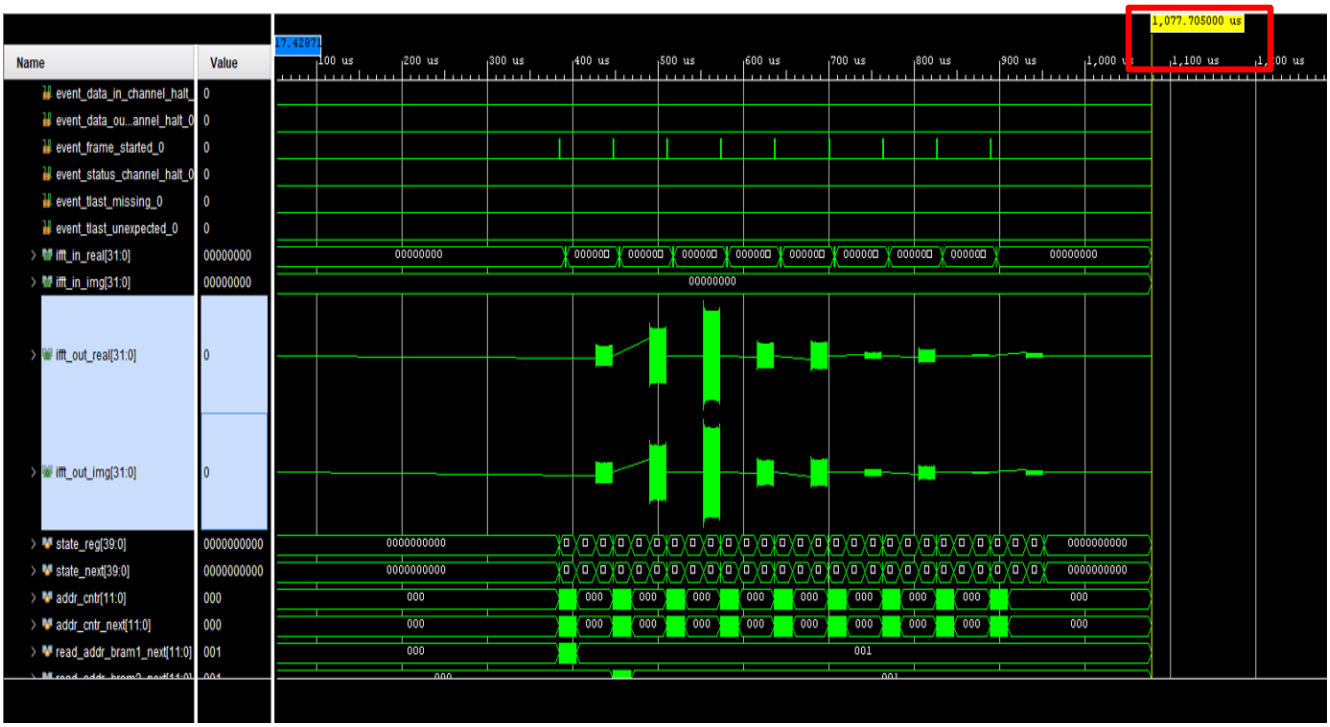

Fig. 5.21 Timing of Process showing IFFT result of last beam received.

The simulation results indicate that the FPGA design model takes only 1.077 milliseconds to compute the results, which is significantly less than our benchmark of 10 milliseconds. This demonstrates the efficiency of our FPGA implementation in achieving fast computation times.

Given that the input sample length is 2048, the total input time is 19 milliseconds. Remarkably, the FPGA completes the processing in under 2 milliseconds, highlighting its capability to handle large datasets with minimal latency. This efficiency is crucial for real-time signal processing applications, where rapid data processing is essential. The FPGA's ability to deliver results in such a short timeframe confirms its suitability for high-speed computational tasks

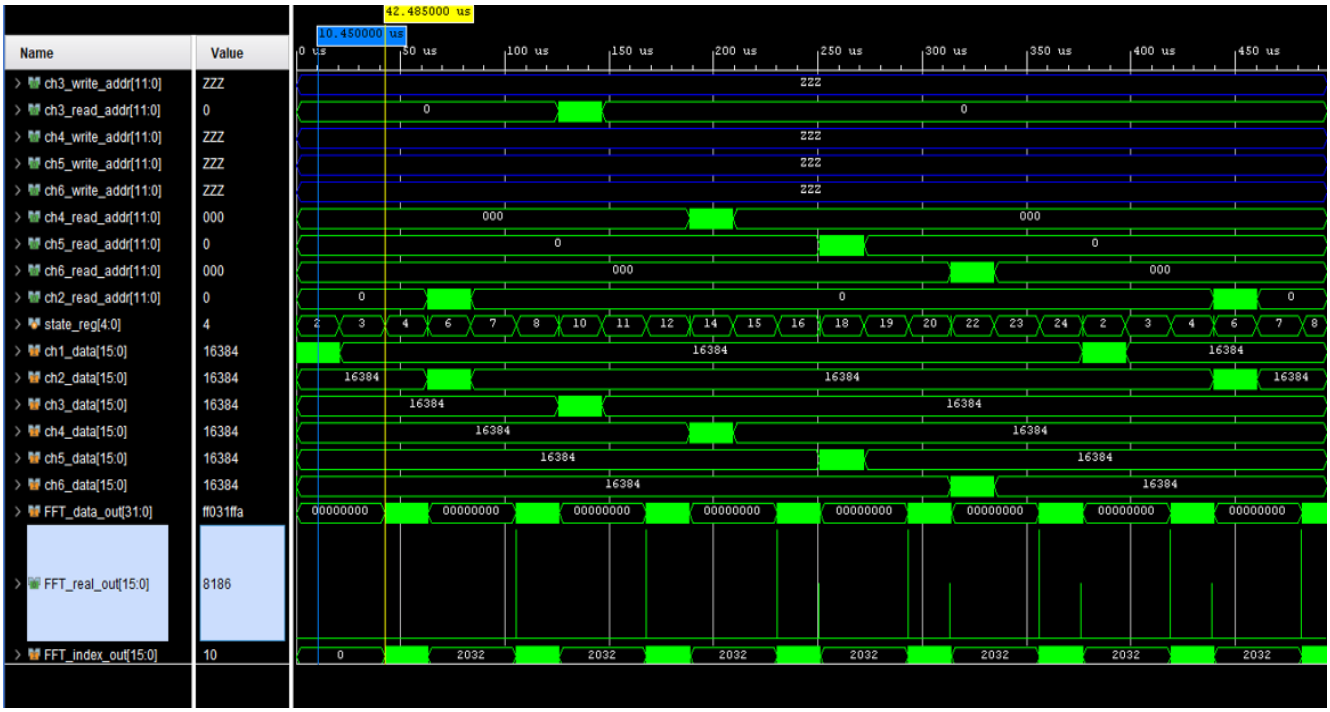

Fig. 5.22 FFT results of all 6 channels

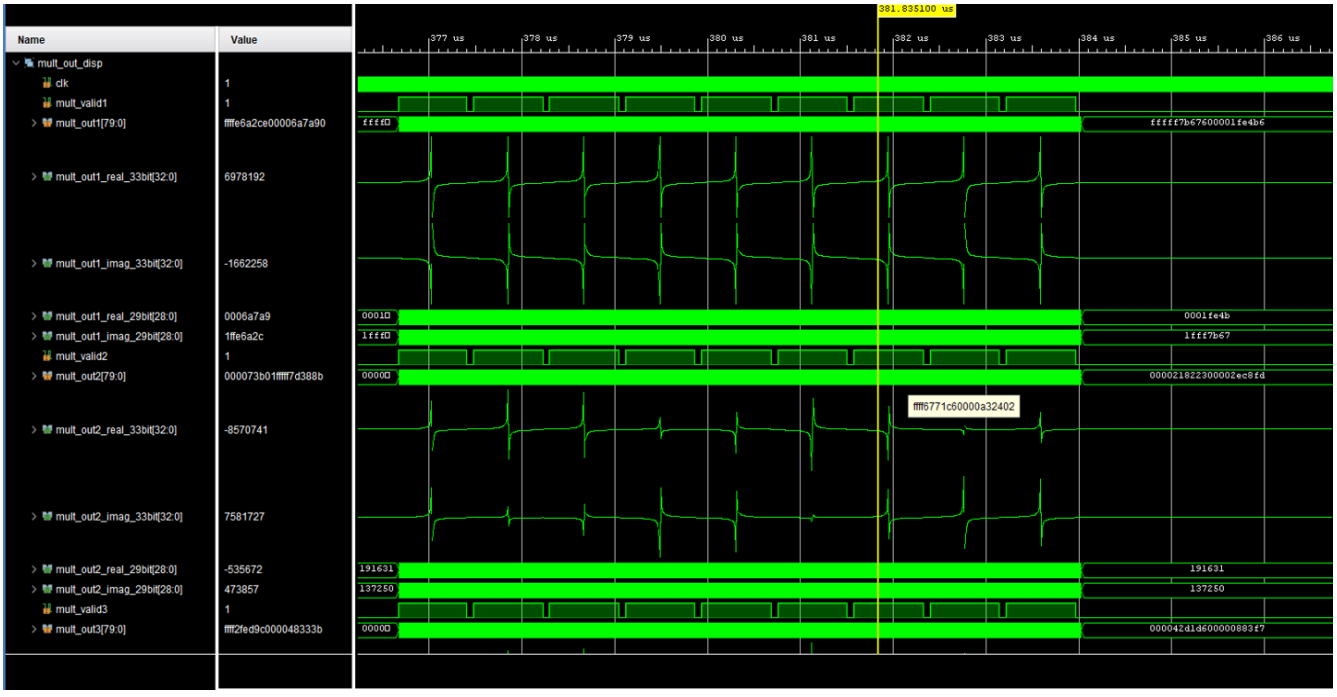

Fig. 5.23 multiplication results mult1 and mult2

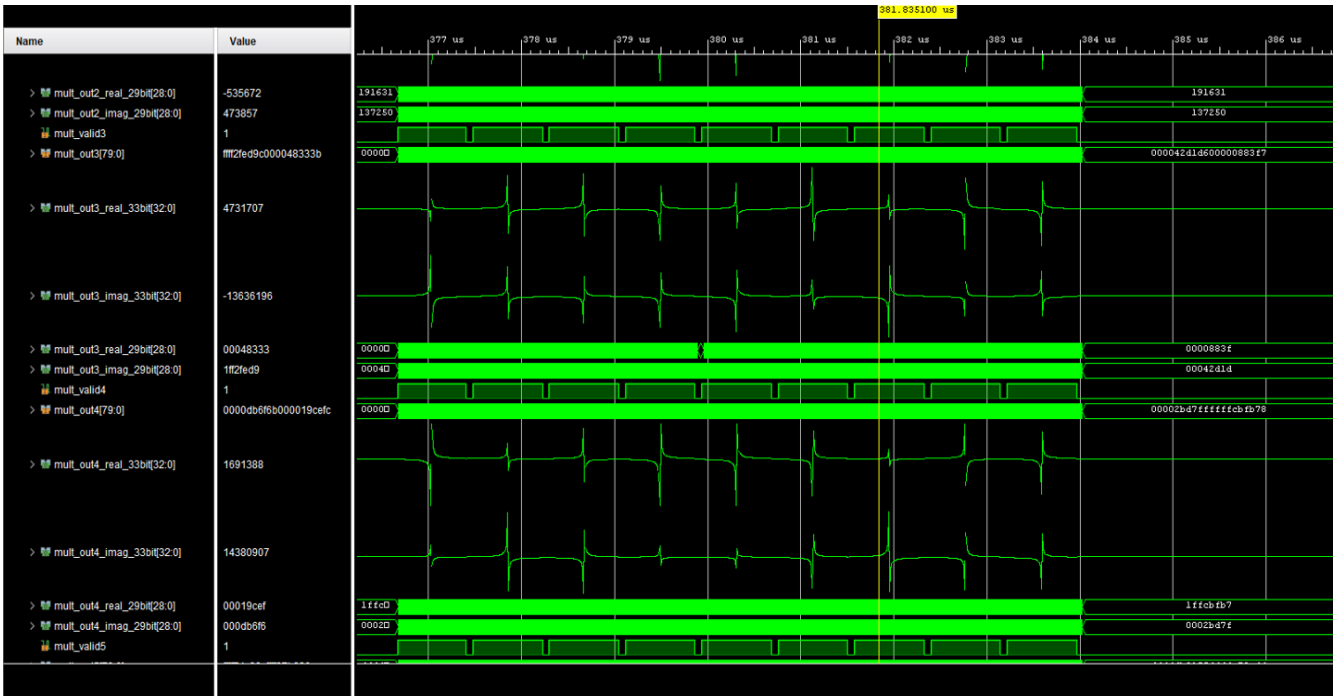

Fig. 5.24multiplication results mult3 and mult4

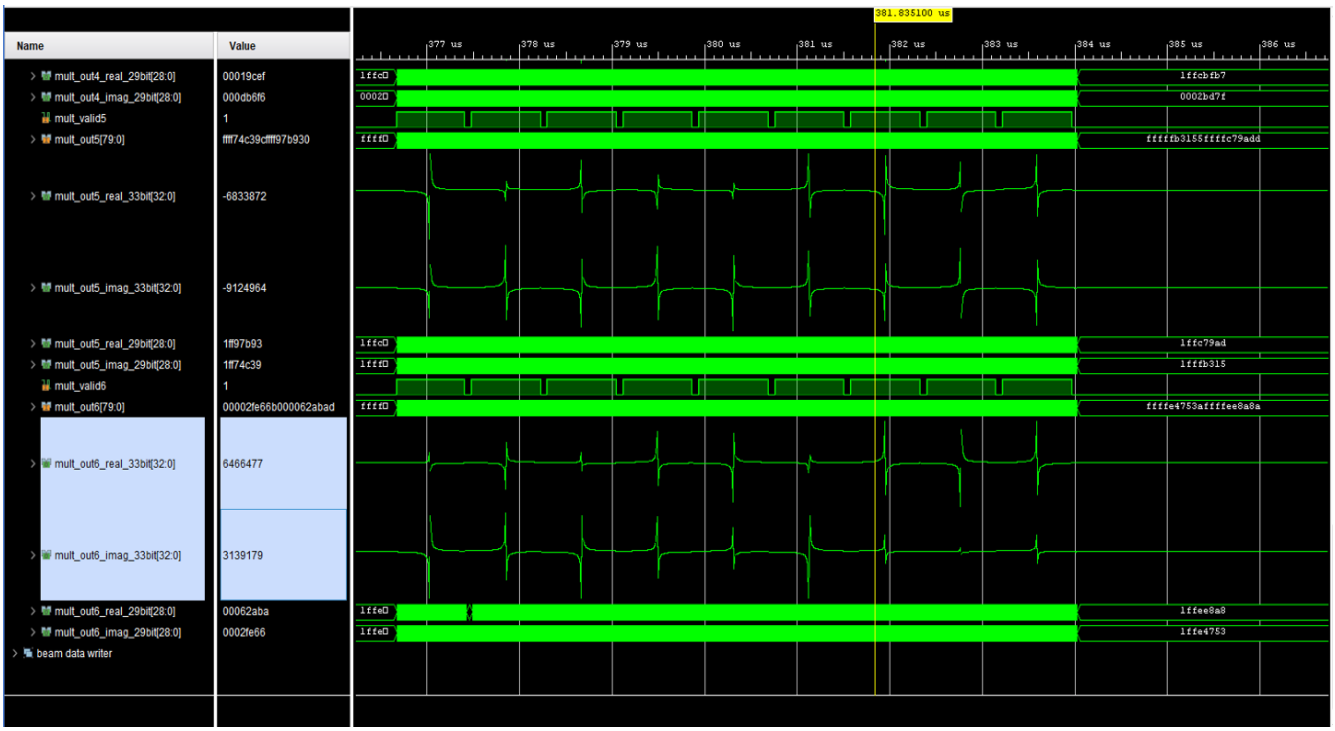

Fig. 5.25 multiplication result mult5 and mult6

5.2 Resource utilization

Table 1 *Utilized Resources in Research*

Resource	Utilization	Available	Utilization %
LUT	3479	53200	6.54
LUTRAM	490	17400	2.82
FF	4462	106400	4.19
BRAM	61	140	43.57
DSP	15	220	6.82
IO	27	200	28.50
BUFG	2	32	6.25
MMCM	1	4	25.00

The table above provides a detailed breakdown of the FPGA resources utilized in our design, including LUTs, BRAM, DSP blocks, and other critical components. Compared to previous work, our design demonstrates superior resource efficiency, making it more optimized for the intended application.

The optimization of our design is evident in the reduced usage of FPGA resources. This efficiency is achieved through careful architectural planning and implementation, ensuring that each component is utilized to its fullest potential without unnecessary overhead. By adhering strictly to the architectural design, we minimized resource consumption, which not only reduces costs but also enhances the performance and scalability of the FPGA implementation.

5.2.1 Comparison

After a thorough review of numerous research papers, I have identified two highly relevant studies that provide a meaningful basis for comparison with my research results. First paper is "FPGA based Real time Sonar beamforming using high level Synthesis" [13]

The study by Chen and Wang (2018) presented an FPGA-based real-time sonar beamforming solution using high-level synthesis. While their approach successfully implemented real-time processing capabilities, it did so with relatively high resource utilization. Our research achieves a more resource-efficient implementation, as detailed in the table below:

Table 2 *Resources utilized in prior work*

Resource	**Utilization**
LUTs	11500
BRAM	190
DSP blocks	75
FF	22000

Despite differences in the number of parameters and specifications of both works, our research demonstrates significant improvements. To ensure a fair comparison, we consider a hypothetical scenario where all implementations use the same number of beams. Even under these conditions, our design remains more resource efficient. This efficiency can be attributed to our optimized architecture and efficient resource allocation strategies. Our implementation achieves lower usage of Look-Up Tables (LUTs), Block RAM (BRAM), and DSP blocks, indicating that even if all systems were configured to process the same number of beams, our design would still outperform in terms of resource utilization.

In contrast, our approach leverages advanced optimization techniques, such as pipeline structuring and resource sharing, to minimize the use of LUTs, BRAM, and DSP blocks. This not only reduces the overall footprint of the design but also allows for greater flexibility in deploying the FPGA for other tasks, making it a more versatile and powerful solution.

In summary, the results shown in the table reflect a carefully crafted FPGA design that outperforms previous implementations in terms of resource efficiency, without compromising on performance or accuracy. This level of optimization makes our design particularly well-suited for complex, resource-constrained environments.

CHAPTER 6

CONCLUSIONS

Our end target in this project is to take the IFFT. We used Vivado software for design and Verilog language to code different modules and controllers for the system. There are three main parts in designing the system model. First, take the Fast Fourier Transform of the input data, and then the crucial step of beamforming architecture comes. After the designing of beamforming, we take the IFFT of the data. The data is coming from six different input channels. Input data is 16 bits and composed of 2048 samples per channel. For beamforming, we have nine different angles, by which each beam will multiply with desired bins of input channels forming a matrix of [6x9x74].

For the FFT and IFFT operation, we used the available IP of fast Fourier transform in Vivado software. We used a number of block RAMs for storing the data after occurring different operations.

There are numerous Controllers that we design by Verilog language for controlling different types of operations. Every step needs to be controlled by proper instructions or logic according to the

requirements of the project. The processing time of computation is 10.285 milliseconds. The entire duration from input to the IFFT completion is 1077 μs, equivalent to 1.077 milliseconds, comfortably meeting the specified time constraint of 10.285 milliseconds.

CHAPTER 7

REFERENCES

[1] Tian, Haowen & Shixu, Guo & Zhao, Peng & Gong, Minyu & Shen, Chao. (2021). Design and Implementation of a Real-Time Multi-Beam Sonar System Based on FPGA and DSP. Sensors. 21. 1425. 10.3390/s21041425.

[2] A. Asada, F. Maeda, K. Kuramoto, Y. Kawashim a, M. Nanri and K. Hantani, "Advanced Surveillance Technology for Underwater Security Sonar Systems," *OCEANS 2007 - Europe*, Aberdeen, UK, 2007, pp. 1-5, doi: 10.1109/OCEANSE.2007.4302220.

[3] Jacob, R., Thomas, T., & Unnikrishnan, A. (2009). Applications of Fractional Fourier Transform in Sonar Signal Processing. IETE Journal of Research, 55(1), 16–27. https://doi.org/10.4103/0377-2063.51320

[4] L. Bjørnø, "Developments in sonar and array technologies," 2011 IEEE Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, Tokyo, Japan, 2011, pp. 1-11, doi: 10.1109/UT.2011.5774169.

[5] Abraham, Douglas. (2019). Introduction to Underwater Acoustic Signal Processing. 10.1007/978-3-319-92983-5_1

[6] A. Elfes, "Sonar-based real-world mapping and navigation," in IEEE Journal on Robotics and Automation, vol. 3, no. 3, pp. 249-265, June 1987, doi: 10.1109/JRA.1987.1087096

[7] H. Al-Khatib *et al*., "The widely scalable Mobile Underwater Sonar Technology (WiMUST) project: An overview," *OCEANS 2015 - Genova*, Genova, Italy, 2015, pp. 1-5, doi: 10.1109/OCEANS-Genova.2015.7271688.

[8] Jie, Zhu & Wei, Chen & Tao, Guo & Gui, Tang. (2011). Design of multi-channel data acquisition system based on FPGA. 247 - 249. 10.1109/ICCSN.2011.6014433.

[9] S. Haykin, J.P. Reilly, V. Kezys, E. Vertatschitsch Some aspects of array signal processing Source: IEE Proceedings F (Radar and Signal Processing), Volume 139, Issue 1, p. 1 –26 DOI: 10.1049/ip-f-2.1992.0001

[10] Tian, H.; Guo, S.; Zhao, P.; Gong, M.; Shen, C. Design and Implementation of a Real-Time Multi-Beam Sonar System Based on FPGA and DSP. *Sensors* 2021, *21*, 1425. https://doi.org/10.3390/s21041425

[11] Paul Graham and Brent Nelson. 1998. FPGA-based sonar processing. In Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays (FPGA '98). Association for Computing Machinery, New York, NY, USA, 201–208. https://doi.org/10.1145/275107.275140.

[12] P. Chen, X. Tian and Y. Chen, "Frequency-Domain Sonar Processing in FPGAs," 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, Wuhan, China, 2008, pp. 756-760, doi: 10.1109/PACIIA.2008.232.

[13] J. Wang and K. Liu, "High-frequency Active Sonar Real-time Signal Processing System Based on FPGA," 2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), Qingdao, China, 2018, pp. 1-4, doi: 10.1109/ICSPCC.2018.8567799.

[14] Chen, L., & Wang, J. (2018). FPGA-Based Real-Time Sonar Beamforming Using High-Level Synthesis. IEEE Transactions on Instrumentation and Measurement, 67(5), 1208-1217.

Index

CCS,

Electrical Engineering

Sonar thesis

ORIGINALITY REPORT

14%	9%	6%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.accc.gov.au Internet Source	2%
2	Bjorno, L.. "Developments in sonar and array technologies", 2011 IEEE Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, 2011. Publication	2%
3	Submitted to University of Limerick Student Paper	2%
4	S. Haykin, J.P. Reilly, V. Kezys, E. Vertatschitsch. "Some aspects of array signal processing", IEE Proceedings F Radar and Signal Processing, 1992 Publication	1%
5	usermanual.wiki Internet Source	1%
6	Submitted to Higher Education Commission Pakistan Student Paper	1%

7	Saeed V. Vaseghi. "Multimedia Signal Processing", Wiley, 2007 Publication	<1 %
8	umpir.ump.edu.my Internet Source	<1 %
9	Submitted to University of Bolton Student Paper	<1 %
10	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
11	www.mdpi.com Internet Source	<1 %
12	Submitted to University of Hertfordshire Student Paper	<1 %
13	ijisrt.com Internet Source	<1 %
14	Submitted to Glasgow Caledonian University Student Paper	<1 %
15	arizona.openrepository.com Internet Source	<1 %
16	Bjorno, L.. "Developments in sonar technologies and their applications", 2013 IEEE International Underwater Technology Symposium (UT), 2013. Publication	<1 %

17	www.coursehero.com Internet Source	<1 %
18	www.scribd.com Internet Source	<1 %
19	dokumen.tips Internet Source	<1 %
20	www.jasonmars.org Internet Source	<1 %
21	Submitted to 41850 Student Paper	<1 %
22	Submitted to Derby College Student Paper	<1 %
23	Submitted to Florida Institute of Technology Student Paper	<1 %
24	Jing Zhang, Lei Huang, Long Zhang, Bo Zhang, Zhongfu Ye. "Robust widely linear beamformer based on a projection constraint", 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015 Publication	<1 %
25	Lecture Notes in Computer Science, 1999. Publication	<1 %
26	repository.tudelft.nl Internet Source	<1 %

27	scholarworks.wm.edu Internet Source	<1%
28	www.encyclopedias.biz Internet Source	<1%
29	dl.acm.org Internet Source	<1%
30	www.rspo.org Internet Source	<1%
31	prr.hec.gov.pk Internet Source	<1%
32	Submitted to University College London Student Paper	<1%
33	ruor.uottawa.ca Internet Source	<1%
34	theses.ncl.ac.uk Internet Source	<1%
35	wiredspace.wits.ac.za Internet Source	<1%
36	digital.library.adelaide.edu.au Internet Source	<1%
37	www.geocities.ws Internet Source	<1%
38	"Cryptographic Hardware and Embedded Systems – CHES 2017", Springer Science and	<1%

Business Media LLC, 2017
Publication

39	Artde D.K.T. Lam, Stephen D. Prior, Siu-Tsen Shen, Sheng-Joue Young, Liang-Wen Ji. "Engineering Innovation and Design", CRC Press, 2019 Publication	<1 %
40	Jun Wang, Kun Liu. "High-frequency Active Sonar Real-time Signal Processing System Based on FPGA", 2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC), 2018 Publication	<1 %
41	Lei Guan. "FPGA-based Digital Convolution for Wireless Applications", Springer Science and Business Media LLC, 2017 Publication	<1 %
42	SHISHIR B SAHAY, T MEGHASYAM, RAHUL K ROY, GAURAV POONIWALA, SASANK CHILAMKURTHY, VIKRAM GADRE. "Parameter estimation of linear and quadratic chirps by employing the fractional fourier transform and a generalized time frequency transform", Sadhana, 2015 Publication	<1 %
43	Wu-Sheng Lu, Andreas Antoniou. "Two-Dimensional Digital Filters", CRC Press, 2020	<1 %

	Publication	
44	digitalcommons.uri.edu Internet Source	<1 %
45	mafiadoc.com Internet Source	<1 %
46	opus.lib.uts.edu.au Internet Source	<1 %
47	orca.cf.ac.uk Internet Source	<1 %
48	repositorio.comillas.edu Internet Source	<1 %
49	audentia-gestion.fr Internet Source	<1 %
50	dokumen.pub Internet Source	<1 %
51	Morteza Babaee Altman, Wenbin Wan, Amineh Sadat Hosseini, Saber Arabi Nowdeh, Masoumeh Alizadeh. "Machine learning algorithms for FPGA Implementation in biomedical engineering applications: A review", Heliyon, 2024 Publication	<1 %
52	Stergios Stergiopoulos. "Advanced Signal Processing Handbook - Theory and	<1 %

Implementation for Radar, Sonar, and Medical Imaging Real Time Systems", CRC Press, 2019

Publication

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		