Simultaneous Detection and Estimation of DOA & PRI of Multiple Threat Radars using a Real-time Parallel FFT

Thesis submitted in partial fulfillment
of the requirements for the degree of

*(Master of Science in **Electronics and Communication Engineering** by Research*

by

Bheema Lakshmi Pradeep
2021702013
lakshmi.pradeep@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
October 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled ***"Simultaneous Detection and Estimation of DOA & PRI of Multiple Threat Radars using a Real-time Parallel FFT"*** by Bheema Lakshmi Pradeep, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Advisor: Dr. Syed Azeemuddin.

To the Past, Present and Future

Acknowledgments

First and foremost, I would like to thank my advisor,**Dr. Syed Azeemuddin** , for providing me with valuable guidance, support, and motivation throughout this project. I am grateful for your expertise and willingness to share your knowledge.

I would also like to thank **Aquibuddin Ahmed sir (Scientist D, RCI), Savitri mam (Scientist E, RCI), J.V Satyanarayana sir (Scientist F, RCI) & Abhiram sir** for their valuable insights and suggestions, which significantly contributed to the development of this research. A special thanks to **Pavan Vadakattu** for the time spent explaining me regarding the project, addressing my queries and helping me in successfully complete the project. I am grateful for the healthy discussions and constructive feedback given by **Rishu Anand, Rajesh, Viren, and Pragathi,**. This greatly influenced the progress of this research project.

I want to personally thank all my ***friends, colleagues, labmates*** for their constant support. Your encouragement and understanding have been instrumental in helping me stay strong and motivated, particularly during difficult times.

In conclusion, I could accomplish all of this thanks to the unwavering support of ***My Family***. I am fortunate to have an understanding and supportive family that constantly encourage my pursuits and honor my choices. Therefore, I would like to dedicate this research to their tireless efforts in helping me reach this point. ***Thank you, and I love you !!***

Abstract

The ability to accurately identify the direction-of-arrival (DOA) and other signal parameters of an incoming radio signal is crucial in many applications, such as military and commercial communication systems. To meet this need, this paper proposes a system for direction finding and pulse parameter estimation that has been designed and implemented using a novel real-time multi-channel parallel Fast Fourier Transform (FFT) on a Field Programmable Gate Array (FPGA).

The proposed system employs a unique approach that allows for the implementation of the FFT operation 5.5 times faster than the traditional IP-core-based implementation, with fewer arithmetic operations and a throughput of 1.35 Giga samples per second (Gsps). The parallel processing capabilities of the FPGA are utilized to enable the system to process multiple signals simultaneously in real-time.

To achieve high precision in DOA estimation, the system employs the sum-difference monopulse technique. This technique utilizes the difference in the amplitudes of two signals received by two antennas. By comparing the amplitude difference with the sum of the signals, the angle of arrival of the incoming radio signal can be estimated with high precision, up to 1 degree .

The proposed system's hardware implementation has been thoroughly studied and verified experimentally through the development of a real-time prototype of the DOA estimation system. The prototype is capable of processing multiple signals in real-time and accurately estimating the direction-of-arrival of the incoming radio signal.

Overall, the proposed system provides an efficient and reliable solution for direction finding and pulse parameter estimation in various commercial and military applications. The use of a novel real-time multi-channel parallel FFT on an FPGA enables fast and accurate signal processing, while the sum-difference monopulse technique ensures high precision in DOA estimation. This makes the system suitable for use in various fields, including communication systems, radar systems, and electronic warfare systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Radar technology has improved substantially over the years, resulting in the development of increasingly sophisticated and capable radar systems. Modern radar systems are complicated electronic devices that detect and track objects. They can operate in a variety of modes, including pulse, continuous wave, and frequency modulated, which allows for enhanced range, accuracy, and target discrimination. Furthermore, modern radar systems can operate in a variety of environments, including air, ground, and sea, and can be deployed on a variety of platforms, such as aircraft, ships, and ground vehicles. In addition to their traditional role in air traffic control and military applications, modern radar systems are also used in a wide range of civilian applications, including weather monitoring, maritime surveillance, and ground-based surveillance.

1.1 Problem Statement

The research was carried out at the behest of the Research Center Imarat (RCI) and will be addressed to either the user or client in the following paragraphs. The RCI provided the problem statement below as the framework within which the research will be constrained.

The project's primary objective is to develop a radar system capable of simultaneously detecting and tracking up to 6 threat radars. The radar system must operate within the frequency range of 1-18 GHz, which is a wide frequency range that includes both X-band and Ku-band frequencies commonly used by military and commercial radar systems. The project's objectives and requirements are detailed in the subsequent section. The project will involve the development of sophisticated signal processing algorithms that can accurately estimate the Direction of Arrival (DOA) of incoming radar signals and also involve estimating pulse parameters critical for identifying the type of radar system and its capabilities, such as its range and target tracking capabilities. This project requires a comprehensive understanding of radar systems and signal processing techniques. It will involve developing and testing advanced algorithms to achieve accurate DOA and pulse parameter estimations, critical capabilities for detecting and tracking multiple threat radars simultaneously.

1.2 Objectives & Requirements

1.2.1 Objectives:

- Detection of 6 targets simultaneously in the bandwidth 1-18 GHz frequency.
- Classification of Radar types: Pulsed, CW, FMCW.
- Estimation of Pulse Parameters of detected radar(s).

1.2.2 Scope:

- We intend to develop and implement an optimal detection of threat radar targets, direction of arrival and pulse parameters estimation algorithm for a passive wideband radar system.
- The system is designed using multiple receive antennas for simultaneous detection of 6 emitters (targets) in the Frequency Range 1-18 GHz.
- First, a MATLAB code for the specified estimator is developed to verify the functionality.
- Later, a Verilog code is developed to port the designed system into custom-made hardware devices (FPGA Board).

1.2.3 Requirements

A radar system consists of a set of antennas, a receiver, an ADC and a signal processor board. For our project, we have used TI's ADC10D1500CIUT, which has a max 1.5 Gsps throughput with 10-bit resolution. To obtain the frequency resolution $\leq$ 3 MHz & pulse-width resolution $\leq$ 500ns, the ADC is configured to 1350 MHz sampling frequency (i.e., it converts 1350 Mega Samples Per Second), and a 512-point FFT is performed.To classify a CW signal, we need 1350000 samples. Since the Virtex-7 FPGA board has a maximum operating frequency of 400 MHZ, the 80 bits of data are sent at 168.75 MHz frequency (i,e. 8 samples at 168.75 MHz), which constitutes the sampling frequency (1350 MHz). The radar signal processing requirements have been summarized in table 1.1.

1.3 Key Contributions

The key contributions in this thesis are as follows:

- We developed an algorithm for detecting and estimating the direction of the arrival of radars, verified the functionality using MATLAB software and implemented it on hardware.

Table 1.1: Requirements for a radar application

Parameter	**Requirement**
FFT size	512-point FFT
Input Data Rate	*80 bits/cycle*
Input Clock Rate	168 MHz
Min no. of samples required	*1350000 samples*
Frequency Resolution	fr<3 MHz
Pulse width Resolution	<500 ns
FPGA Technology	Virtex 7 FPGA

- Designed and Implemented a real-time parallel FFT to achieve high throughput and avoid data loss.
- Test the designed system with MATLAB-generated static and real-time data generated using a signal generator and horn antennas.

1.4 Thesis Organisation

Chapter 2 gives a briefing on different types of radar systems present, Direction of Arrival (DOA) techniques, and other information required to design a direction-finding system. Chapter 3 gives an overview of the architecture of the radar system and explains the functionality of various blocks present in the system. Chapter 4 focuses on the importance of the FFT operation in the radar system, various previously implemented architectures, problems faced during the real-time implementation of the system, and how we overcame the problems using our proposed 2-dimensional parallel FFT architecture. Chapter 5 explains the experimental setup we used to test the designed system. The various methods of testing we used to verify the functionality of the system are discussed. Chapter 6 discusses the results obtained from the above testing. Lastly, chapter 7 concludes the thesis and suggests a possible direction for future research.

Chapter 2

Literature Survey

2.1 Electronic Warfare

Electronic Warfare (EW) uses electromagnetic (EM) energy to gain a military advantage over an adversary. One of the critical components of EW is radar systems, which are used both for detection and as a source of EM energy to disrupt or deceive enemy systems. Electronic warfare (EW) can be classified into three distinct groups: electronic attack (EA), electronic protection (EP), and EW support measure (ESM), as shown in the Fig. 2.1. [2]

2.1.1 Electronic Attack (EA)

The electronic attack involves using electromagnetic energy, directed energy, or anti-radiation weapons to disable or destroy enemy facilities or equipment, intending to weaken their combat capabilities. Electronic attack comprises measures to disrupt or hinder an adversary from effectively employing the electromagnetic spectrum, such as jamming and electromagnetic deception.

2.1.2 Electronic Protection (EP)

Electronic Protection (EP) involves strategies and techniques to safeguard friendly electronic systems and personnel against an adversary's electronic attack or exploitation. EP measures include the use of encryption or frequency hopping to protect electronic signals from being intercepted or disrupted.

2.1.3 Electronic Support Measure (ESM)

Electronic support measures (ESM) involve utilizing electronic signals to detect, intercept, and identify an adversary's signals. A passive ESM attempts to provide information about an enemy's electronic systems, such as their geolocation, frequencies, and modulation, to enable the user to better understand their operations and objectives. All the information is stored in the form of pulse descriptive words (PDWs) and sent to EA systems for combat assistance or to provide situational awareness. ESM can

also be used to identify and track an enemy force's electronic emissions, such as radar or communications signals, and to warn of potential threats.

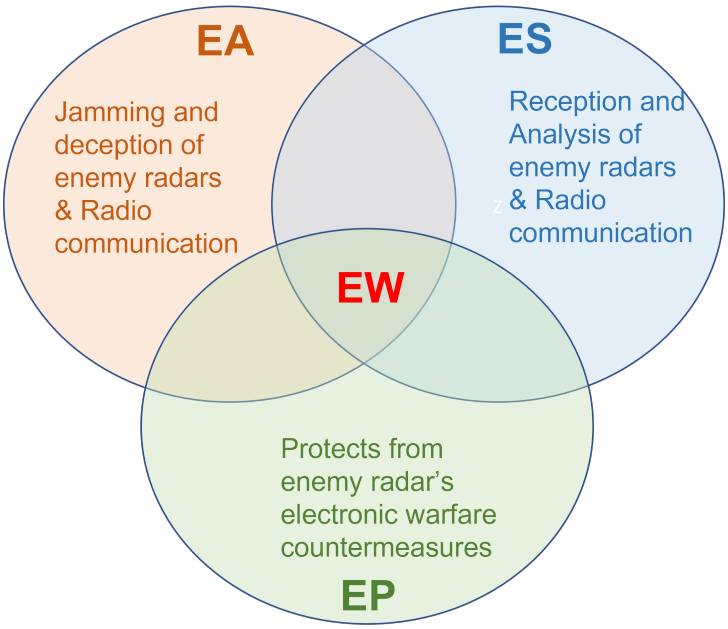


Figure 2.1: Classification of Electronic Warfare

Radar systems are particularly useful in EW due to their ability to detect and track targets at long ranges, even in adverse weather conditions. In addition, modern radar systems can use advanced signal processing techniques to improve their ability to detect and track targets in cluttered environments, making them an effective tool for offensive and defensive EW operations. Our current work mainly focuses on developing a radar system for ESM according to the user requirements as given in table 1.1.

2.2 Types of Radar Systems

There are several types of radar, each with unique capabilities and applications. Any radar can broadly be classified into two categories: continuous wave radar and pulsed radar.

2.2.1 Continous Wave (CW) Radars

A continuous wave radar system transmits an electromagnetic wave with a fixed frequency continuously through its antenna. The radar receiver continuously receives the reflected signal, which requires at least two antennas - one for transmission and one for reception. The antennas must be properly isolated to prevent energy from leaking into the receiving antenna, which can result in the transmitted signal interfering with the received signal. Since perfect isolation is impossible, CW radars typically

have lower transmit power compared to pulsed radars. This limits their range and makes them more suitable for short-range applications.

2.2.1.1 Frequency modulated continous radar (FMCW)

The primary limitation of continuous wave (CW) radar is that it cannot measure the radial range of a target due to the absence of time referencing in the signal. However, the use of frequency modulated continuous wave (FMCW) radar allows for the measurement of the radial range of stationary objects by periodically adjusting the frequency of the transmitted signal to create a time reference in the signal. FMCW radar uses the offset in frequency of the received signal, similar to a pulsed radar, to determine the delay offset. While it is possible to transmit signals with complex frequency patterns, basic ramp and triangular modulation are typically used in FMCW radar.

2.2.2 Pulsed Radars

In a pulsed radar system, electromagnetic waves are transmitted in pulses with periods of time in between to listen for echoes of the transmitted pulses. The range and directional information of a detected target is provided by simple pulsed radars. The directional information of the target is determined using the radiation direction of the rotating antenna when the echo is received. The transmit time of the pulse is referred to as the pulse width (τ) or pulse duration (PD). The time between the leading edge of one pulse to the leading edge of the next is the pulse repetition interval (PRI). The pulse repetition frequency (PRF) is the inverse of the PRI and indicates the number of times per second that a radar completes the transmit or receive cycle. The PRI of a radar plays a significant role in its performance.

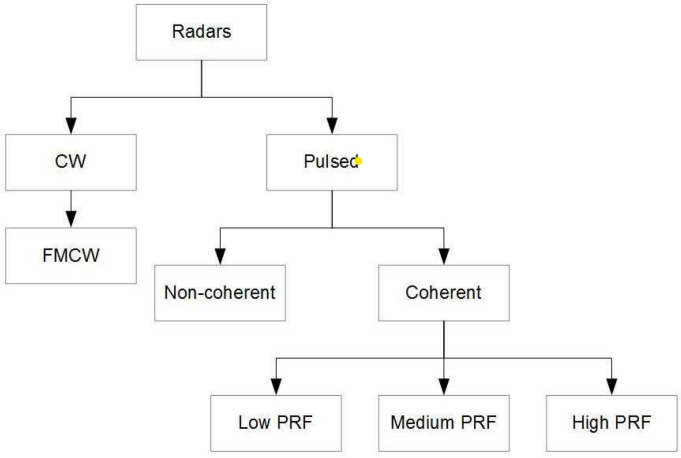


Figure 2.2: Classification of Electronic Warfare

Pulsed radar systems operate with different pulse repetition frequency (PRF) regimes: low, medium, and high. Low PRF radars receive echoes from all ranges of interest before transmitting the next pulse,

making range measurements unambiguous and suitable for use as search radars. High PRF radars receive echoes with Doppler shifts less than the PRF, allowing for unambiguous velocity measurements. Medium PRF radars receive target echoes after several PRIs elapse and have Doppler shifts several times greater than the PRF, resulting in ambiguous range and velocity measurements. Low PRF radars are typically used as moving target indicator radars, while high and medium PRF radars are usually pulsed Doppler radars.

2.3 Direction of Arrival (DOA)

The direction of arrival (DOA) refers to the direction from which an electromagnetic wave arrives at a receiving antenna. In the context of radar systems, DOA estimation is the process of determining the direction of a target relative to the radar antenna.

There are several methods for DOA estimation, including time-of-arrival (TOA) estimation, amplitude comparison monopulse, phase comparison monopulse, and Array-based estimation.

2.3.1 TOA Based Estimation

TOA estimation involves measuring the time difference of arrival between signals received by multiple antennas to determine the angle of arrival.

2.3.2 Amplitude comparision & Phase comparision Monopulse

Amplitude comparison monopulse compares the amplitude of signals received by two antennas to determine the direction of the target, whereas,Phase comparison monopulse compares the phase of signals received by two antennas to determine the direction of the target. Detailed implementation of the amplitude comparison monopulse algorithm is further discussed in chapter 3

2.3.3 Array Based DOA Estimation Techniques

There are four types of array-based DoA estimation techniques: conventional techniques, subspace-based techniques, maximum likelihood techniques, and integrated techniques.

2.3.3.1 Conventional Techniques

Conventional techniques are based on the principles of null steering & beamforming, and obtaining high resolution, typically requires large number of antenna elements. Conventional methods for estimating DoA involve electronically steering beams in every direction possible and search for output power peaks[37] . The delay-and-sum approach and Capon's minimum variance method are the most widely used conventional techniques.

2.3.3.2 Subspace Techniques

Although many classic beamforming-based techniques, such as Capon's minimum variance method, are widely used, they have some fundamental resolution limitations. Subspace techniques are high-resolution sub-optimal approaches that take advantage of the eigenstructure of the input data matrix. Schmidt [38] and Bienvenu & Kopp[32] were the first to take advantage of a more precise data model structure for a sensor array of any shape. In addition to Schmidt's Multiple Signal Classification (MUSIC) algorithm[5, 40], Roy et al.'s Estimation of Signal Parameters via Rotational Invariance Technique (ESPRIT) algorithm [34, 33] also made a significant contribution to the subspace-based techniques.

2.3.3.3 Maximum Likelihood Techniques

Maximum likelihood Techniques are computationally intensive techniques that work best in environments with a low signal-to-noise ratio. The computational complexity of ML approaches makes them less popular than suboptimal subspace techniques. However, in terms of performance, ML techniques outperform subspace-based techniques, particularly when the signal-to-noise ratio is low or the number of samples is limited.[46]

2.3.3.4 Integrated Techniques

The integrated approach separates various signals and estimates their spatial fingerprints, from which their arrival directions can be identified using subspace techniques.

Table 2.1: MATLAB Simulation-Based performance comparision of various DoA Algorithms

Algorithm	**Time Taken**	**No. Of Computations**
Beam Scan	0.00785 Sec	$9\times e^6$
Capon's MVDR	0.01422 Sec	$8\times e^7$
MUSIC	0.02215 Sec	$14\times e^8$
ESPRIT	0.0158 Sec	$5\times e^5$
MonoPulse	0.00942 Sec	$9\times e^4$

We have considered Beam-scan, MVDR, MUSIC, ESPIRIT, and Monopulse algorithms for our project and compared them based on MATLAB simulation-based performances. Conventional algorithms such as beam scan and MVDR have poor resolution compared to other algorithms such as MUSIC and ESPIRIT. In contrast, while providing much higher resolution, the MUSIC and ESPIRIT algorithms are computationally intensive and time-consuming. Thus, based on the MATLAB simulation results ta-

ble 2.1, we conclude that the monopulse algorithm estimates DoA with fewer computations and the least execution time. [20]

2.3.4 Recent Trends in DOA Estimation

In recent times, significant advancements have been observed in the direction of arrival (DOA) estimation field, showcasing several emerging trends. One prominent trend is the application of deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which have shown promising results in achieving accurate DOA estimation [24]. The utilization of compressed sensing techniques to attain accurate DOA estimation with fewer measurements is another area of research. Integrating data from multiple sensor modalities, multimodal DOA estimation, has emerged as a recent trend [6]. Finally, real-time and distributed DOA estimation algorithms have gained attention to cater to the demands of real-time applications and distributed sensor networks.

2.4 PRI

A pulsed radar emits EM waves in pulses, separated by a period which provides information about the direction and range of a detected target.[2] When the echo is received, the radiation direction of the revolving antenna estimates the target's direction. The interval between the leading edge of one pulse and the leading edge of the next is known as the pulse repetition interval (PRI), and the transit time of the pulse is known as the pulse width (t) or pulse duration (PD). The number of times a radar completes the transmit/receive cycle in one second is known as the pulse repetition frequency (PRF), which is the inverse of the PRI. Thus PRI, PRF plays an essential role in identifying the type of radar and its range.

Accidental variations in PRI are unavoidable due to the ambiguities present in the transmitter, amplifier, and oscillator. However, on some occasions, the PRI of radar is varied intentionally. By switching between PRIs, the radar system can operate at various unambiguous ranges and velocity values at any given time or even unfold the unambiguous range and velocity observations. Hence an ESM system needs to estimate the PRI of the radar to classify the radar.

2.4.1 Various PRI Schemes

The most commonly used PRI schemes in pulsed radar systems are: constant, jittered, dwell and switch, stagger, sliding, etc. Our system is designed to estimate constant, staggered, and jittered PRI schemes as shown in the Fig. 2.3

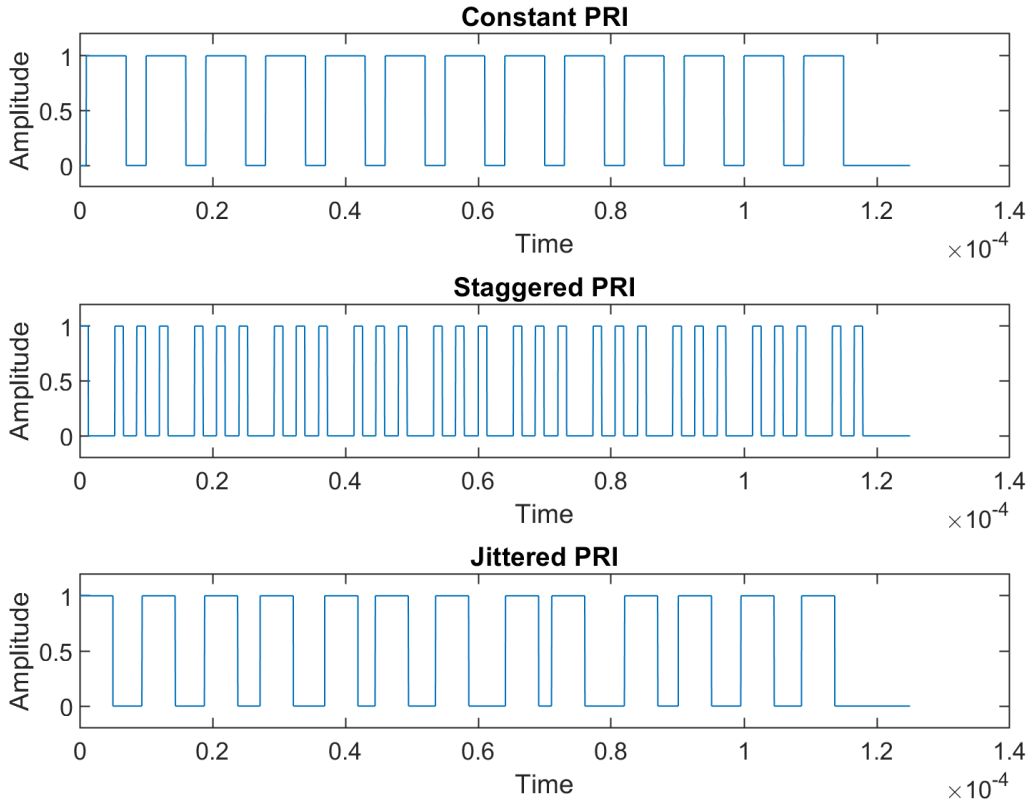


Figure 2.3: MATLAB simulation of the PRI Schemes used for in the project. a) Constant PRI scheme where there is no variation in PRI. b) Staggered PRI where the 2 or more PRIs repeat periodically. c) Jittered PRI where it follows the constant PRI scheme with 10% - 30% Random fluctuations.

2.4.1.1 Constant PRI

A constant PRI scheme is one with no variations in the PRI. If variations exist, they are unintentional and account for less than 1% Fig.2.3(a) depicts a continuous PRI pulse train.

2.4.1.2 Staggered PRI

A staggered PRI scheme is a periodic series of two or more PRIs. A stagger sequence may comprise multiple PRI instances. Staggered PRI schemes are grouped based on the number of 'positions' or 'levels'. Before it repeats itself, the number of PRIs in a sequence is expressed by positions, while levels give the number of different PRI elements present in the sequence. Fig.2.3(b) depicts a staggered PRI pulse train with varying intervals between pulses.

2.4.1.3 Jittered PRI

Radars that use a jittered PRI add random fluctuations to their PRI for each pulse. Since their PRI is constantly changing, this type of PRI scheme is utilized in radars as a form of electronic defense against some types of jamming. The parameters of interest for a jittered PRI radar are identical to those for a constant PRI radar; however, more emphasis should be placed on the jitter pulse train and the pulse train statistics as shown in Fig.2.3(c). The mean PRI of the pulse train, the jitter distribution curve, and the PRI range are a few critical parameters.

2.4.2 Deinterleaving

Nowadays, the environment consists of many emitters ranging from a very low PRF to a very high PRF acting simultaneously. Hence it is important to differentiate the various signals and map them to their respective emitters. This is known as signal sorting or signal de-interleaving. Even though there are many de-interleaving algorithms, the most relevant algorithms applicable for our project are sequence search, TOA difference histogram, CDIF, SDIF, CDIF with SS & SDIF with SS.

2.4.2.1 Sequence Search (SS)

One of the most straightforward TOA-based deinterleaving techniques is the sequence search (SS) algorithm. It offers more reliability and precision than histogramming-based deinterleaving techniques [26, 27] at the expense of processing efficiency. The algorithm compares estimated pulse trains to the received pulse train by estimating pulse trains with all possible PRI values, gradually rising from smallest to greatest. When the number of received pulses that match the estimated sequence reaches a predefined threshold, it is considered to have found an emitter with a constant PRI equal to the estimated pulse train's PRI. The identified sequence is then taken out of the received pulse train, and the procedure is repeated using fewer pulses. This algorithm is highly susceptible to interference and noise pulses. Hence, a more robust algorithm is required [26, 27].

2.4.2.2 TOA Difference Histogram

The TOA difference histogram, also called an all-difference histogram, is generated by grouping the differences in TOA between every combination of TOAs in a set of TOAs [26, 27]. This method is well-suited for fast processing since it involves subtractions and summations. When applied to an interleaved pulse train, the technique produces peaks in the histogram at the PRIs corresponding to the EW environment emitters. The peaks are detected using an exponentially decaying threshold level on the histogram, determined by the total time interval of the estimated TOA differences, the PRI being tested, and an experimentally determined parameter.

2.4.2.3 Cumulative Difference (CDIF) Histogramming

CDIF histogramming, called Cross-Difference of Arrival histogramming, is used to de-interleave pulse trains. The CDIF histogram algorithm is an improvement over the TOA difference histogram method. This algorithm creates a histogram using various levels, or depths, of TOA differences. The first level difference is the TOA difference between any pulse and the preceding pulse. The nth level difference refers to the TOA differences between any pulse and the nth pulse that arrived before it.[26, 39] This technique uses the cross-correlation function between pairs of received pulses to construct the CDIF histogram. The cross-correlation between pairs of received pulses makes it possible to estimate the time difference of arrival (TDOA) between the two pulses. These TDOA values can then be used to construct a histogram, where the peaks correspond to the PRIs associated with the emitters in the EW environment.

Initially, the algorithm constructs a histogram by binning the first-level difference values. If the frequency of a bin exceeds a predetermined threshold, an emitter with a constant PRI associated with that bin is assumed to be detected. The algorithm then removes the pulses corresponding to the detected PRI and begins again without those pulses.

In cases where none of the bins exceed the predetermined threshold, the algorithm creates a new histogram by binning all the TOA differences calculated so far and the next level of TOA differences. Creating histograms with increasing difference levels continues until an emitter is detected or the algorithm stops at a specified difference level.

The threshold is determined for each level of the CDIF histogram based on the number of bins, the current difference level, the number of differences binned, and two experimentally determined parameters [26, 39]. The first-level difference histogram has a predetermined number of bins and determines the bin size by dividing the range of first-level difference values by the number of bins. While drawing histograms for higher-level differences, the number of bins can vary, but the bin size remains the same. One advantage of CDIF histogramming is that it is less noise-sensitive than other de-interleaving techniques. However, it requires a higher computational overhead than other methods, which can disadvantage specific applications.

2.4.2.4 Sequential Difference (SDIF) Histogramming

The SDIF (Sequential Difference) histogram is an improved version of the CDIF (Cumulative Difference) histogram. This technique is more robust against missing or false pulses. The SDIF histogram algorithm is similar to the CDIF histogram algorithm but has a different histogram redrawing approach. In SDIF histogramming, when a histogram is constructed for a specific difference level, only the difference values for that level are considered while creating the new histogram. All previous difference-level values are discarded [25].

2.4.2.5 CDIF with SS

It is a two-pass weighted-search de-interleaving method that combines a histogramming-based algorithm with the sequence search (SS) algorithm[26]. This approach helps reduce the SS algorithm's processing time while maintaining reliability and accuracy. In the first pass of this algorithm, a histogramming based deinterleaving algorithm is employed to obtain the probable PRIs quickly. These PRIs are then passed to the SS algorithm for further processing in the second pass. The SS algorithm tests only these probable PRIs, which reduces the overall processing time and computational complexity of the algorithm [26]. The weighting factor used in this algorithm is experimentally determined to balance the trade-off between the processing time and the algorithm's accuracy. We have used the CDIF-SS algorithm to deinterleave and estimate the emitter's PRI.

2.4.3 Recent Trends in FPGA based PRI Estimation

FPGA-based PRI estimation implementations typically involve the design of custom hardware architectures and utilizing digital signal processing techniques. One approach is implementing PRI estimation algorithms using FPGA-based digital signal processing blocks, such as finite impulse response (FIR) filters, fast Fourier transforms (FFTs), and correlation modules, which can be interconnected and optimized for efficient processing [10]. Another approach is to leverage FPGA vendor-specific IP cores or libraries to accelerate PRI estimation by integrating pre-designed and optimized modules into the FPGA design. Furthermore, researchers have explored parallel processing techniques and hardware optimizations, such as pipelining and resource sharing, to enhance PRI estimation performance on FPGAs[43].

Chapter 3

Architecture Overview

In this chapter, we outline the architecture of the system, and various blocks present in the architecture and explain its functionality. An ESM system generates a Pulse Descriptive word (PDW) which contains all the information of the incoming signals from various emitters in a multi-emitter-dense environment. The generated PDW is later sent to the base station for further classification of the emitters. The various blocks involved in generating the PDW are explained in the following section.

An ESM system consists of antennas, a receiver, an ADC, and a signal processor board, as shown in Fig.3.1. The antennas constantly sweep across the bandwidth to detect enemy targets. Hence, the receiver continuously receives the high-frequency analog signal and down-converts it into an IF signal. The analog IF signal is then converted into digital data with the help of high-speed ADCs and sent to the signal processing unit. [18] The signal processing unit is the system's heart and is responsible for estimating and identifying the signal parameters. The SPU consists of various other blocks implemented on a Xilinx Vertex 7 FPGA, each having a unique functionality and generating a PDW. We will discuss the blocks in detail in the following sections.

3.1 Antenna Elements

The proposed system consists of 4 antenna array elements placed 90° apart, as shown in Fig.3.2a. In our system, we used Vivaldi antennas as antenna elements, where elements 1 and 3 are used for elevation angle estimation, and elements 2 and 4 are used for azimuth angle estimation. The antenna elements are placed with a squint angle w.r.t the axis perpendicular to the plane consisting of all four antenna elements, shown in Fig.3.2b.

3.2 Receiver

Recent improvements in digital processor speed and analog-to-digital converter (ADC) technology have allowed development of relatively wide-band digital channelized receivers. [36, 12], The antennas continuously sweep high-frequency bands (L band to Ku band) to detect enemy targets. The entire

bandwidth is divided into several frequency bands and searched for threat emitters.The receiver has two modes of operation: continuous search/tracking and scan.[22, 19]

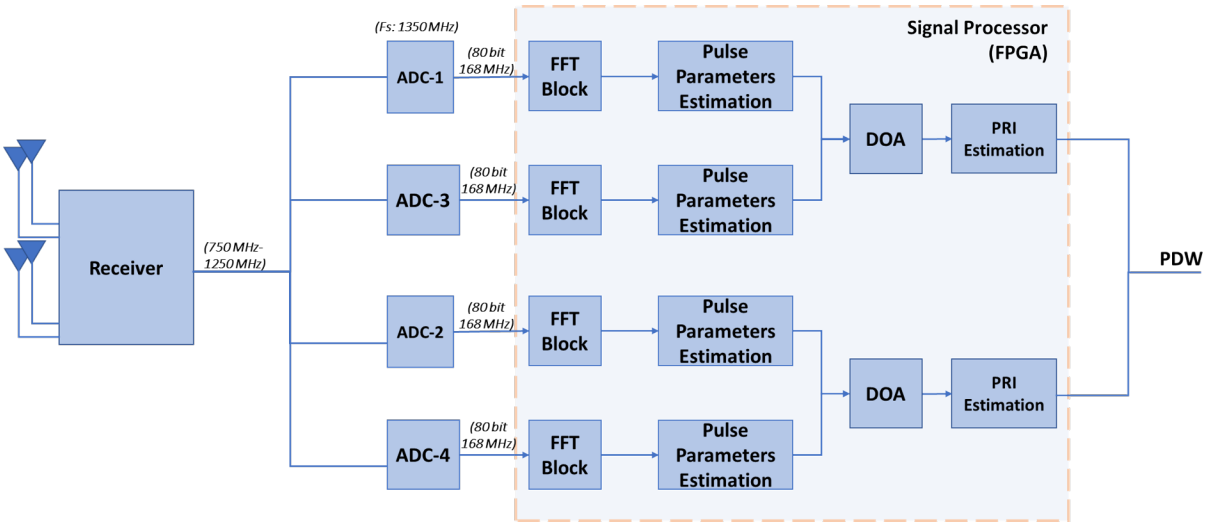
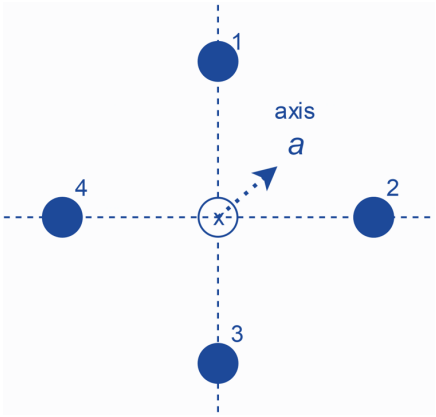
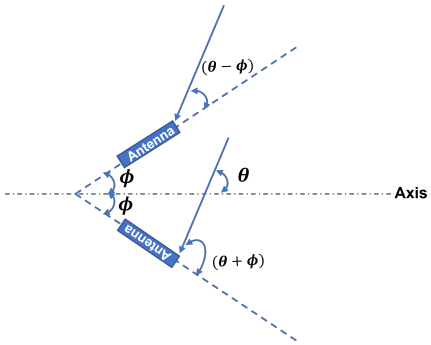


Figure 3.1: 4-channel ESM System Architecture



(a) The Front View of Placement of Vivaldi Antenna's. Here, Elements 1,3 gives azimuthal Angle whereas Elements 2,4 are responsible for the estimation of elevation angle



(b) The Side View of Placement of Vivaldi Antenna's. Here, the antennas are placed at a squint angle ϕ, which is later used in calculating the monopulse Ratio

Figure 3.2: Placement of Vivaldi Antennas

In tracking mode, the receiver is locked to a specific frequency band, and the antennas continuously detect radars only present in that frequency range. In scan mode, the receiver constantly sweeps all frequency bands for a fixed duration. As a result, the receiver continuously receives high-frequency

analog signals and converts them to IF signals. The converted IF is later sent to the analog to digital converter (ADC), where it is converted into digital data and sent to a signal processor board.

3.3 ADC Interface

Since there are 4 antenna elements, we have used 2 2-channel ADCs to convert the received IF signal into digital data. Each channel of dual channel ADC operates at 1/4th of the sampling clock at the dual data rate (DDR), providing 2 data interfaces for delayed and non-delayed samples.[19, 18, 12]Our system used TI's ADC10D1500 ADC board configured as DDR and Non-DES mode with SPI Interface. The ADC10D1500 is a recent addition to TI's Ultra-High-Speed ADC series. This low-power, high-performance CMOS analog-to-digital converter digitizes signals at 10-bit resolution for dual channels at sampling rates up to 1.5 GSPS (Non-DES Mode) or up to 3.0 GSPS for a single channel (DES Mode).

3.3.1 Design flow

The differential signals I and Q inputs of length 10 bits of each channel of an ADC are fed to the select-IO interface wizard with double data rate (DDR) selected. Hence the output of the select-IO wizard is doubled, i.e., it receives 40-bit output. We have configured the sampling frequency to 1350 MHz, and each ADC is operated at 337.5 MHz. The input clock for the select-IO wizard is 337.5 MHz applied from the clocking wizard, which is generated from the input Differential Converter Sampling Clock (337.5 Mhz). The flow from input to FFT (DI & DQ Values) is shown in the Fig.3.3. Thus a total of 80-bit data is sent to the signal processor board at 168.75 MHz frequency, which constitutes a 1350 MHz sampling frequency.

For our project, we have used Xilinx's virtex7 FPGA to implement the pulse parameter estimation & Direction finding systems for the proposed ESM system. To obtain the frequency resolution $\leq$ 2.67 MHz & pulse-width resolution $\leq 379ns$, the ADC is configured to 1350 MHz sampling frequency (i.e., it converts 1350 Mega Samples Per Second), and a 512-point FFT is performed. We need 1350000 samples (1 msec data) to classify a signal as CW. Since the Virtex-7 FPGA board has a maximum operating frequency of 400 MHz, the 80 bits of data are sent at 168.75 MHz frequency (i,e. 8 samples at 168.75 MHz simultaneously), which constitutes the sampling frequency (1350 MHz). The radar signal processing requirements have been summarized in table 1.1

3.4 FFT Block

Since the received signal is unknown, performing an FFT operation is the most reliable approach to estimating the signal characteristics. In the past decades, implementing FFT on hardware has been a key area of research. To meet the high performance and real-time requirements, hardware engineers have been trying to implement an efficient architecture for FFT calculation [45, 29, 28, 17, 7, 11, 35].

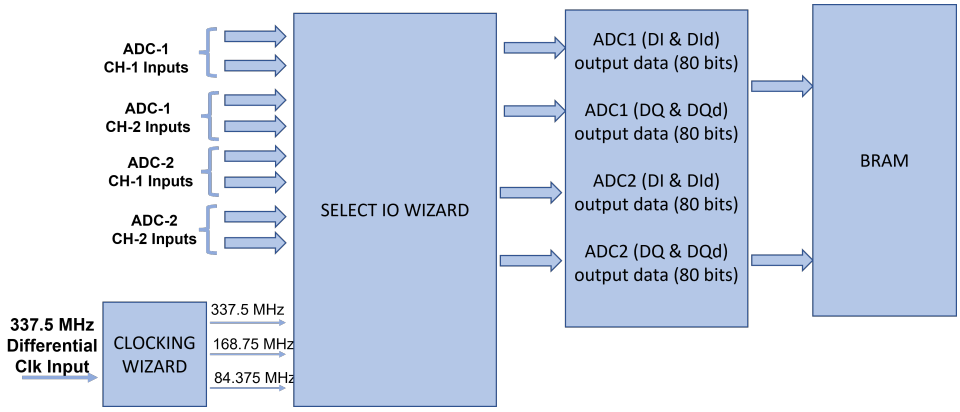


Figure 3.3: ADC design flow: IF Inputs to FFT Input

Efficient implementation primarily focuses on achieving high performance, low utilization, or low power consumption. Throughput is a key performance metric for streaming architectures. The ADC converts the received IF signal into 80-bit data and sends it to the signal processing board, i.e., an FPGA at 168.75 MHz. Since we are sampling the signal at 1350 MHz, to achieve frequency resolution $\leq$ 3MHz & pulse width resolution $\leq 0.5\mu s$, we performed a 512-point FFT operation on the incoming signal. The various ways of implementing FFT on hardware, challenges faced during implementing FFT for a real-time system, and how our proposed FFT architecture has resolved our issues are explained in detail in the following chapter 4. This chapter contributes to the majority of the research work

3.5 Pulse Parameter Estimation

To determine the nature of the incoming IF signal, we need to estimate the characteristics of the signal. The FFT module's output consists of the signal's frequency information. This output signal is passed through the pulse parameter estimation module to estimate the various pulse parameters such as Amplitude, Bandwidth, and Pulse Width. This module consists of various sub-modules, each having its functionality as shown in the fig.3.4.[30]

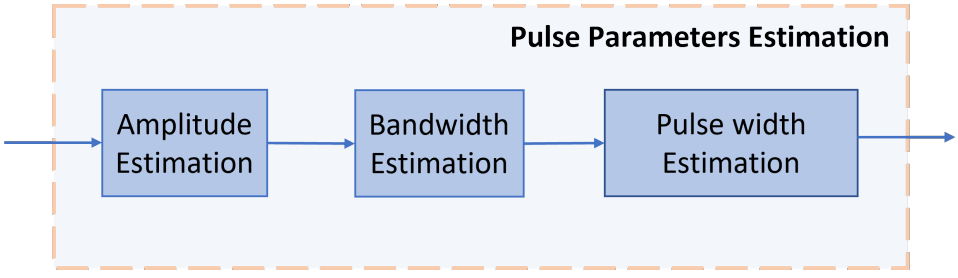


Figure 3.4: Pulse Parameter Estimation Block

3.5.1 Amplitude Estimation Module

Since we performed a 512-point FFT on a signal sampled at 1350 MHz, the frequency resolution of the signal is 2.67 MHz. Spectral leakage occurs if the input signal frequency is not a multiple of the frequency resolution. When spectral leakage occurs, the signal's energy spreads into multiple adjacent bins. Using proper windowing methods, we can reduce spectral leakage. The most commonly used windows are Rectangular, Hanning, Blackman, etc.[30] In our project, we are considering only 512 samples at a time. Thus, the window length is 512, and the same rectangular window is shifted with time. Each window is assigned a window number based on the "tlast" signal obtained from the FFT module. The FFT output and the window number are sent to the detector module to identify the number of valid signals present in it. The detector compares the obtained signal's power with a pre-calculated threshold value to determine whether or not the signal is valid. The threshold of the signal depends upon reciever sensitivity, noise floor, FFT processing gain, radiation losses, cable losses, IF gain etc. Considering all the above parameters, we have set a finite value as a system threshold value.

3.5.2 Bandwidth Estimation Module

Once the valid signals are detected from the FFT output, the bandwidth of the detected signals is estimated. We know that when FFT operation is performed on an FMCW signal, signal power spreads across the adjacent frequency bins due to change in frequencies. Bandwidth can be obtained by counting the number of frequency bins that are adjacent to each other.

$$Bandwidth = N_f \times (F_s/N)$$

N_f is the number of adjacent frequency bins, (F_s/N) is the frequency resolution of the FFT output. It means that they originate from a single radar source. If any two frequencies are far away, the signals correspond to different bands and originate from different radar sources. This method works for chirp, FMCW radar signals.

3.5.3 Pulse Width Estimation Module

For our project, we consider 512 FFT samples each time in a window making the window length 379.5 ns. The pulse width module counts the total no. of consecutive windows the signal is detected.[30] If there is any discontinuity, the counter stops and resets for that frequency, indicating the end of the pulse. According to the project requirements given in table 1.1, to classify any signal as a CW signal, the pulse width of the signal should be greater than 1 msec, i.e., the frequency should be continuously detected for at least 2365 window numbers. If a signal is CW, the check_cw flag in the system goes high. Fig.3.5 shows pulse width estimation using a state transition diagram.

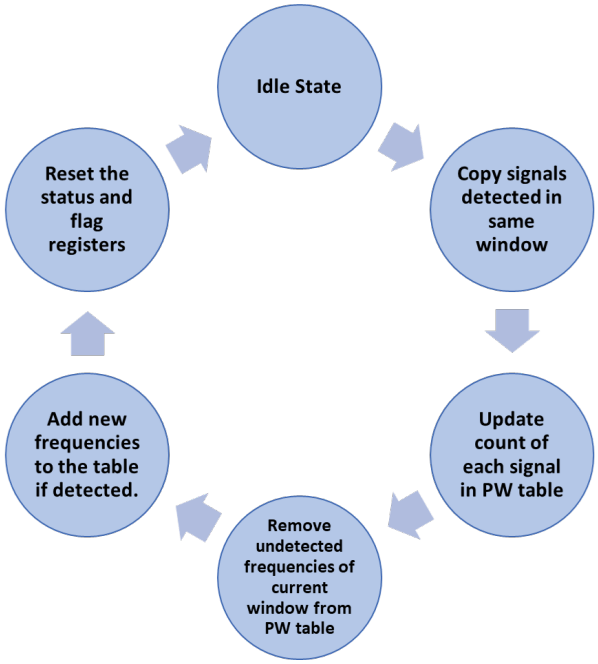


Figure 3.5: State flow of Pulse Width Estimation Block

3.6 Direction of Arrival Estimation Module

Earlier in chapter 2, we understood the importance of DOA and various techniques used to estimate the DOA. After comparing the metrics of various techniques, based on the requirement and resources available, we have chosen the amplitude comparison monopulse algorithm to estimate the DOA for our project.

3.6.1 Monopulse Algorithm

The most commonly used MUSIC algorithm has the best resolution and accuracy.[20] However, the drawback of the MUSIC algorithm is that the number of antenna array elements must be greater than the maximum number of targets the system can track or detect. Also, the MUSIC algorithm is computationally intense because it involves computing the Covariance matrix and Eigenvectors.[20] The alternative to the subspace methods is the Monopulse algorithm.

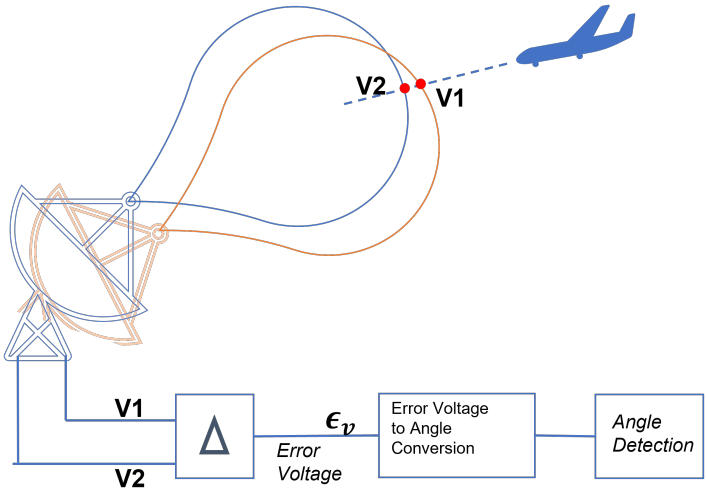


Figure 3.6: Simultaneous amplitude comparision

3.6.1.1 Amplitude-Comparision Monopulse Algorithm

By comparing the amplitudes of simultaneous echos (V_0, V_1) as they are received on two slightly offset beams, as shown in Fig.3.6, it is possible to determine an accurate angle of the source under the assumption that the amplitude of a target's echoes from repeated pulse transmission stays relatively constant. The constant amplitude assumption, however, is unrealistic as, in reality, echo strengths vary quickly from one pulse transmission to the next. Depending on its angle of arrival, the echo projects differently on the receive beams. As a result, the amplitude and phase components of V_0 and V_1 that rely on the angle of arrival are different. In contrast, amplitude and phase components produced by parameters outside the radar, such as target range, target cross-section, and medium losses, are the same for both V_0 and V_1. By taking a ratio of the two voltages, the components that occur identically for two voltages cancel out, but the angle-dependent components are retained.

$$\epsilon_v = f(V_0/V_1)$$

The value ϵ_v, known as the error voltage, carries purely directional information. Though it does not directly communicate the information in an angular form, it can be converted with the help of a mapping function called a monopulse response curve (MRC), given in terms of the error voltage.

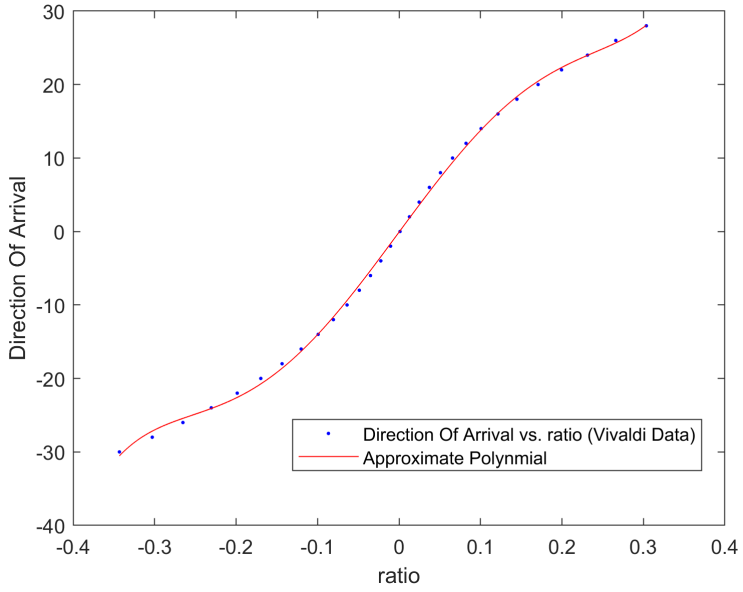


Figure 3.7: Approximated monopulse response curve for 8 GHz radar signal

From Fig.3.2b, we can depict that, for a particular angle of incidence θ w.r.t axis 'a' angle of incidence of the signal on to the antenna array element 1 is $(\theta - \phi)$ and antenna array element 3 is $(\theta + \phi)$. Assuming 'P' as the power of incident radar signal and G1 and G2 are the gains of the array elements 1 & 3, the total power received at the antenna elements 1 & 3 is PxG1 & PxG2, respectively. Thus the sum-difference monopulse ratio can be obtained as

$$m = \frac{P \times G_2 \; - \; P \times G_1}{P \times G_2 \; + \; P \times G_1} = \frac{G_2 - G_1}{G_2 + G_1} \tag{3.1}$$

3.6.1.2 Linear Regression Model

For our project, we have used a Machine Learning model to evaluate the direction of arrival of an emitter. Since the monopulse ratios obtained do not give direct angular information, we need to convert them into angular data by mapping them using an MRC modeled by a linear regression algorithm. Here the monopulse ratio Eq.(3.1) is calculated for a radar signal of 8 GHz frequency arriving at various angles and plotted in Fig.3.7 (represented by Dots). The obtained DoA vs. monopulse ratio plot is modeled with an n^{th} degree polynomial using mathematical linear regression techniques and is represented as a line in Fig.3.7. similarly, MRC is obtained for various other frequency bands,as shown in Fig.3.8 and coefficients are estimated for each frequency band. These coefficients are stored in a BRAM and are selected based on the receiver configuration. Using the coefficients of the regression model, we estimated the DoA of various real-time radar signals.

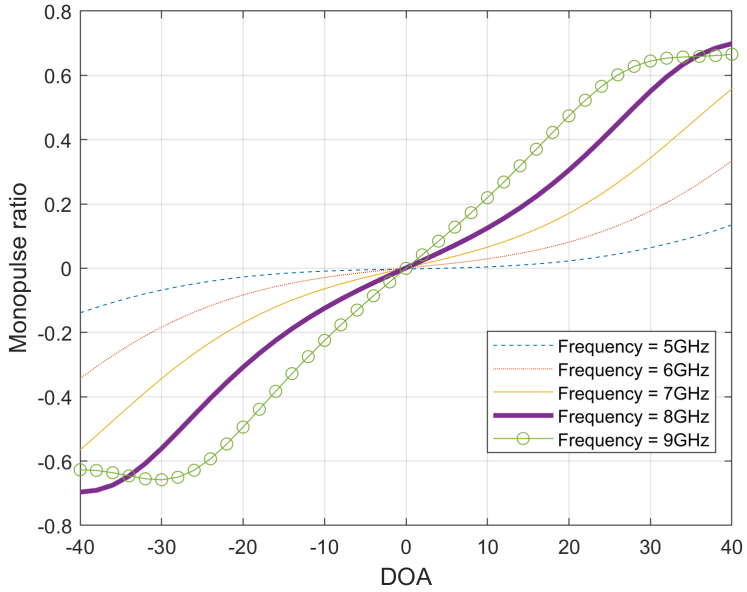


Figure 3.8: MATLAB simulation showing the variation in the approximated monopulse response curve for various frequency bands

$$DoA = P_n \times m^n + P_{n-1} \times m^{n-1} + P_{n-2} \times m^{n-2}$$

$$...P_2 \times m^2 + P_1 \times m^1 + P_0 \tag{3.2}$$

Where P_n, P_{n-1}, ... ,P_1, P_0 are the polynomial coefficients, n is the order of the polynomial, and m is the monopulse ratio. The polynomial degree n is selected such that the polynomial has a 95% confidence bound. Hence 5th degree polynomial or higher is used to achieve the curve-fitting constraints.

3.7 Pulse Repetitive Interval Estimation Module

The Pulse repetition interval module identifies the type of PRI and estimates the value of the PRI. The module uses the time of arrival information, which is obtained from the window number generated after the FFT module. The Basic idea behind the PRI module is to perform a histogram of the difference between the time of arrival of any two pulses. The pattern by which we take the difference is given by the flow chart diagram, as shown in the figure3.9.

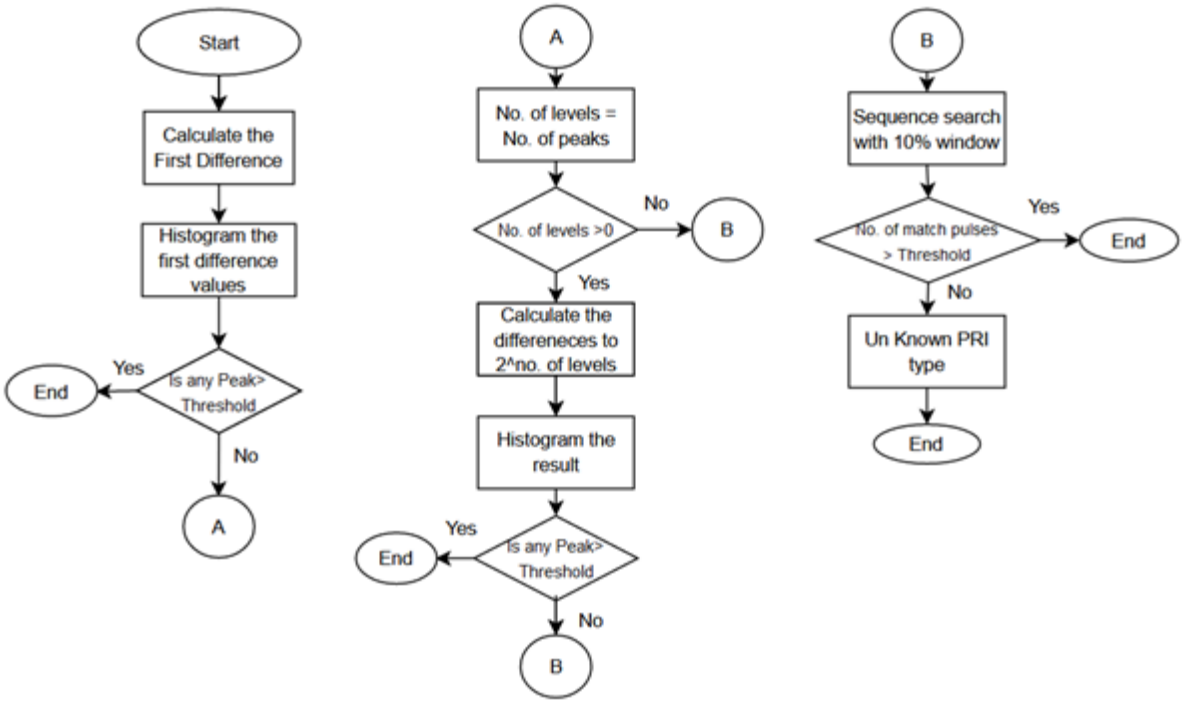


Figure 3.9: Flowchart for PRI estimation

In this module, the final PDW is generated containing the information of all the Pulse Parameters, Direction of arrival, PRI & PRI Type. Here, the module is divided into two parts. First, it de-interleaves the interleaved signal and then estimates the PRI and PRI type based on the difference in the TOA histogram method. The module is operated at 168 MHz frequency, and the output PDW is 116 bits.

Chapter 4

FFT Implementation

4.1 Introduction

The demand for high-speed signal processing systems in space and radar systems has risen dramatically in recent years. Unlike in the past, space/airborne systems plan to shift the data processing unit on board using FPGAs. Onboard data processing systems have their own set of problems. Since the data being processed is in real-time, the data's sampled rate is critical to the accuracy of the results. The FFT core is an essential part of onboard DSP. The FFT core's performance determines the system's overall performance.

The Discrete Fourier Transform (DFT) plays a crucial role in digital signal processing. Fast Fourier Transform is an efficient way of DFT which is used in sectors like signal processing [8], Image Processing [23, 3], Radar signal processing[13, 21, 29], radio astronomy [28, 44], and other fields. FFT algorithms can be broadly classified into two main categories: (i) composite/non-prime length FFT such as Cooley-Tukey[14] and (ii) prime length FFT such as winograd[41]. In the past decades, the implementation of FFT on hardware has been a key area of research. To meet the high performance and real-time requirements, hardware engineers have been trying to implement an efficient architecture for FFT calculation [45, 29, 28, 17, 7, 11, 35]. Efficient implementation primarily focuses on achieving high performance, low resource utilization, or low power consumption. Throughput is a key performance metric for streaming architectures. A high-throughput, energy-efficient parallel FFT architecture using a periodic memory activation scheme is proposed by Ren Chen in[11]. By optimizing the intermediate data processing and first stage pipelining, Yupu Zhao proposed a high-performance and resource-efficient FFT architecture in [45]. The CORDIC algorithm-based multiplication scheme has greatly reduced the overall resource utilization and system latency. Mario Garrido explains the concept of pipelining in hardware architecture and further classifies it into serial and parallel architectures based on the complexity of the input [17]. If the input data is a 2-dimensional data type, such as an image, we need to implement a 2-dimensional FFT. A parallel 2D FFT is performed on a real-time MRI data in [23]. We can also perform a 1D FFT by interpreting the 1D signal as 2D data. An N-point FFT is implemented by transforming the data into N_1 rows & N_2 columns or vice versa and performing FFT

in both dimensions [28, 41]. A similar method is used to develop a real-time multichannel FFT for a radio spectrometer in [44].

4.2 Background

4.2.1 Discrete Fourier Transform

Discrete Fourier transform is a method where discrete-time domain inputs are converted into discrete-frequency domain outputs. FFT is the fast algorithm of DFT. The most common method of implementing FFT is using the Cooley-Tukey algorithm. Mathematically, the DFT of a sequence x(n):x_0, x_1, ... x_{N-1} of length N can be expressed as :

$$X_k = \sum_{l=0}^{N-1} x_l W_N^{lk}, \qquad k = 0, 1, 2...N-1. \tag{4.1}$$

From Eq: 4.1, it is understood that each X_k requires $n-1$ complex additions and n complex multiplications. N complex multiplications can be achieved by performing $4N$ real multiplications and $2N$ real additions, whereas N complex additions can be achieved by performing $2(N-1)$ simple additions. Therefore, each X_k computation requires 4N real multiplications and 4N-2 real additions. Hence, to compute a DFT of length N, we need to perform $4N^2$ real multiplications and $4N^2-2N$ real additions. As we increase the length of the sequence N, the computational complexity increases and is difficult to realize on hardware. Thus, it is necessary to reduce the computational complexity of DFT. Using the symmetrical and periodic properties of twiddle factors, Cooley & Tukey proposed a new algorithm that greatly reduced arithmetic operations from $\theta(N^2)$ to $\Theta(Nlog_2N)$.

4.2.2 Cooley Tukey algorithm

FFT is the fast algorithm for performing DFT on signals. Among the various FFT algorithms available, the most widely used FFT algorithm was proposed by Cooley and Tukey. [14]. By definition, radix-2 DIT FFT divides the DFT of size N into size $N/2$ for every recursive stage. FFT algorithms are represented by their flow graphs; at each stage of the graph, butterflies and rotations are calculated. The number of stages in an FFT is given by $n = log_2N$. The signal flow graphs of FFT are either of DIT or DIF composition, where the difference is the rotations at each stage. In this paper, we are using radix-2 DIT butterflies and radix-4 DIT butterflies as basic structures, as shown in Fig: 4.1, Fig: 4.2 respectively.

4.2.2.1 Radix-2 DIT FFT

The radix-2 DIT equation can be obtained by rewriting Eq: 4.1 as

$$X_k = \sum_{l=0}^{N/2-1} x_{2l} W_N^{k(2l)} + W_N^k \sum_{l=0}^{N/2-1} x_{2l+1} W_N^{k(2l)} \qquad k = 0...N-1. \tag{4.2}$$

Using identity $W_{N/2} = W_N^2$,

$$X_k = \sum_{l=0}^{N/2-1} x_{2l} W_{N/2}^{kl} + W_N^k \sum_{l=0}^{N/2-1} x_{2l+1} W_{N/2}^{kl} \qquad k = 0...N-1. \tag{4.3}$$

Since $W_{N/2}$ can be denoted as W_N^2, we need to compute only the sums for k= x_0, x_1, ..., $x_{N/2-1}$,i.e each summation can be interpreted as a small DFT of size $N/2$. Defining $y_k = x_{2k}$, $z_k = x_{2k+1}$ and replacing N with $N/2$, we can rewrite Eq: 4.3 as

$$\begin{aligned} Y_k &= \sum_{l=0}^{N/2-1} y_l W_{N/2}^{kl} \\ Z_k &= \sum_{l=0}^{N/2-1} z_l W_{N/2}^{kl} \qquad k = 0...N/2-1. \\ X_k &= Y_k + W_N^k * Z_k, \\ X_{k+N/2} &= Y_k - W_N^k * Z_k \end{aligned} \tag{4.4}$$

The Eq: 4.4 can be graphically represented as the Cooley-Tukey butterfly model, as shown in the Fig: 4.1

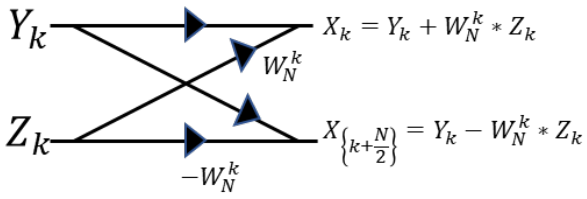


Figure 4.1: Radix-2 Butterfly diagram

4.2.2.2 Radix-4 DIT FFT

A sequence of the form $N = 2^{2n}$ can be represented as $N = 4^n$. Thus, implementing FFT in radix-4 reduces the total arithmetic cost. The radix-4 DIT equation can be obtained by rewriting Eq: 4.1 as

$$X_k = \sum_{l=0}^{N/4-1} x_{4l} W_N^{k(4l)} + \sum_{l=0}^{N/4-1} x_{4l+1} W_N^{k(4l)} + \sum_{l=0}^{N/4-1} x_{4l+1} W_N^{k(4l+2)} \sum_{l=0}^{N/4-1} x_{4l+1} W_N^{k(4l+3)} \qquad k = 0...N-1. \tag{4.5}$$

Since $W_{N/4}$ can be denoted as W_N^4, we need to compute only the sums for k= x_0, x_1, ..., $x_{N/4-1}$,i.e each summation can be interpreted as a small DFTs size $N/4$.Defining $y_k = x_{4k}$, $z_k = x_{4k+1}$,$g_k = x_{4k+2}$,$h_k = x_{4k+3}$ and replacing N with $N/2$, we can rewrite Eq: 4.5 as

$$\begin{aligned} Y_k &= \sum_{l=0}^{N/4-1} y_l W_{N/4}^{kl}, \\ Z_k &= \sum_{l=0}^{N/4-1} z_l W_{N/4}^{kl}, \\ G_k &= \sum_{l=0}^{N/4-1} g_l W_{N/4}^{kl}, \\ H_k &= \sum_{l=0}^{N/4-1} h_l W_{N/4}^{kl} \end{aligned} \qquad k = 0...N/2-1. \tag{4.6}$$

using the identities $W_N^{N/4} = e^- j(\pi/2) = -j$, $W_N^{N/4} = (-j)^2 = -1$, $W_N^{3N/4} = (-j)^3 = j$, we can simplify the equations as

$$\begin{aligned} X_k &= Y_k + W_N^k Z_k + W_N^{2k} G_k + W_N^{3k} H_k \\ X_{k_N/4} &= Y_k - j * W_N^k Z_k - W_N^{2k} G_k + j * W_N^{3k} H_k \\ X_{k_N/2} &= Y_k - W_N^k Z_k + W_N^{2k} G_k - W_N^{3k} H_k \\ X_{k_3 N/4} &= Y_k + j * W_N^k Z_k - W_N^{2k} G_k - j * W_N^{3k} H_k \end{aligned} \tag{4.7}$$

The Eq: 4.7 can be graphically represented as Cooley-Tukey butterfly model, as shown in the Fig: 4.2.

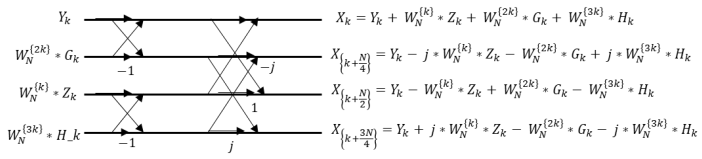


Figure 4.2: Radix-4 Butterfly diagram

4.2.3 1-D FFT

A 1-D FFT is a simple Fast Fourier transform performed on a one-dimensional signal. The FFT algorithm can be implemented in two architectures that optimize throughput or area. As we perform an FFT on real-time data, we need a pipelined architecture to process the data accurately. Since the incoming data is either row-wise or column-wise, we execute a 1D FFT row-wise or column-wise.

4.2.3.1 Streaming architecture

This architecture is used in high-throughput applications and supports scalar and vector inputs. Using vector inputs, we can achieve a throughput of Giga samples per second (GSPS).

4.2.3.2 Burst architecture

This architecture accepts only scalar inputs and is known for minimum resource implementation, especially with the large FFT sizes. The system must be able to handle burst data and high latency to implement burst architecture.

4.2.3.3 Pipelined architecture

A pipelined architecture can consist of n stages where the data flow happens from one stage to another. Here, each stage calculates all the computations of one stage of the FFT algorithm. As soon as the data moves out of one stage, a new data set is processed, making it a streamlined architecture. Fig: 4.3 shows the general structure of the pipelined architecture.

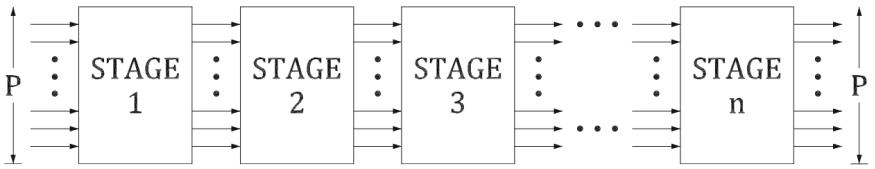


Figure 4.3: n-stage Parallel Pipelined Architecture

If the incoming signal is a two-dimensional matrix, such as an image, we must use a two-dimensional FFT rather than a one-dimensional FFT. Using several 1-D FFTs on multiple processors is a simple technique for implementing a 2-D FFT.

4.2.4 2-D FFT

The 2-D DFT of a two dimensional array or a matrix x can be defined by the Eq:4.8,where x can be denoted as x_{l_1,l_2}

$$X_{k1,k2} = \sum_{l_1=0}^{N_1-1} \sum_{l_2=0}^{N_2-1} x_{l_1,l_2} W_{N_1}^{k1l1} W_{N_2}^{k_2 l_2} \qquad k_1, k_2 = 0, 1, 2...N-1. \tag{4.8}$$

The arithmetic operations required to compute Eq:4.13 yield a cost $O(n_1^2 n_2^2)$ which can be significantly reduced by implementing 2D DFTs using a series of 1D DFFTs, implemented in a 1D FFT algorithm as shown in 4.4. The Cooley-Tukey algorithm proposed the following steps for performing FFT on a composite size $N = N_1 X N_2$ for a two-dimensional input:

- Perform N_1 DFTs of size N_2.
- Multiply by twiddle factors
- Transpose rows to columns
- Perform N_2 DFTs of size N_1.

$$\begin{aligned} X_{k1,k2} &= \sum_{l_1=0}^{N_1-1} \sum_{l_2=0}^{N_2-1} x_{l1,l2} W_{N_1}^{l_1 k_1} W_{N_2}^{l_2 k_2} \\ &= \sum_{l_1=0}^{N_1-1} W_{N_1}^{l_1 k_1} \left(\sum_{l_2=0}^{N_2-1} x_{l1,l2} W_{N_2}^{l_2 k_2} \right) \qquad k_1, k_2 = 0, 1, 2...N-1. \\ &= \sum_{l_1=0}^{N_1-1} \left(X_{l_1,K_1} \right) W_{N_1}^{l_1 k_1} \end{aligned} \tag{4.9}$$

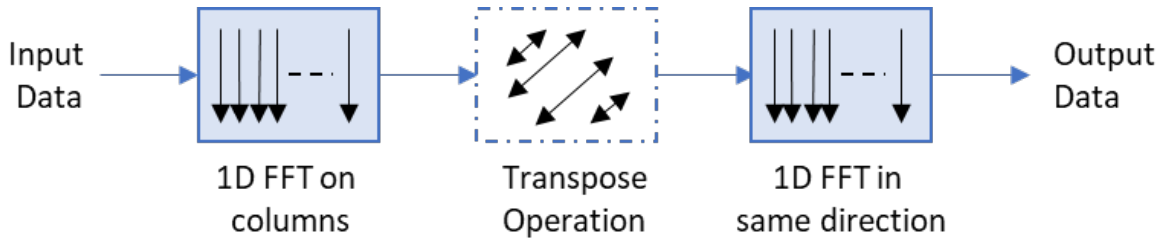


Figure 4.4: Implementation of 2D FFT using a set of 1D FFTs

4.2.4.1 Importance of transpose operation in a 2D FFT

The data received in a real-time radar application is mostly in one dimension. To implement a 2D FFT, we need to convert this one-dimensional array into a two-dimensional array. To achieve consecutive memory access through its addresses, the data are placed in either row-major or column-major

format, as shown in the Fig:4.5 The row-major scheme dictates that $mem[l] = A[p,q]$ where $l = L_0 + (p*Q+q)$, and the column-major scheme dictates that $mem[l] = A[p,q]$ where $l = L_0 + (q*P+p)$.

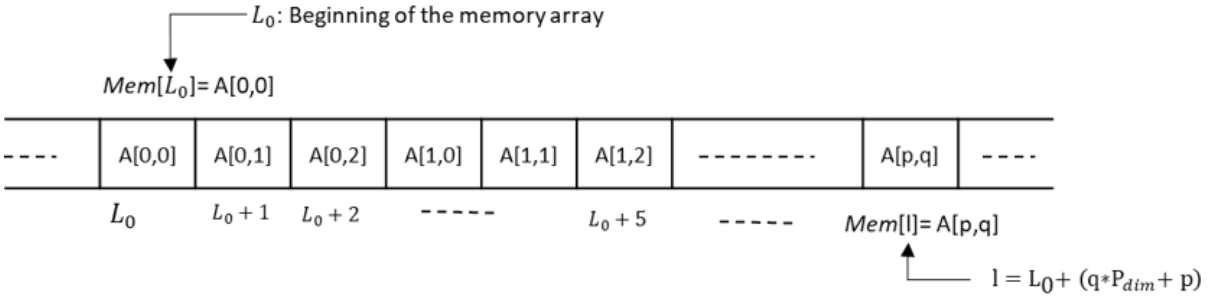


Figure 4.5: Storage of data in Row Major & Column Major Formats

If a vector of length $N = PXQ$, is stored in a 2D array $A[P,Q]$,in a column major format, then $x_l = A[p,q]$, where $l = qXP + p$. The DFT of the vector can be expressed as

$$\begin{aligned} X_k &= \sum_{l=0}^{N-1} x_l z_l, where z = W_N^k \\ &= \sum_{p=0}^{P-1}\sum_{q=0}^{Q-1} A[p,q] z^{(qXP+p)} \end{aligned} \qquad k = 0,1,2...N-1. \tag{4.10}$$

since we can represent $W_N^P = W_{N/P} = W_Q$, Z_P can be represented as $Z_P = \left(W_N^k\right)^P = w_Q^k$. Thus, Eq:4.10 can be modified as

$$\begin{aligned} X_k &= \sum_{p=0}^{P-1}\sum_{q=0}^{Q-1} A[p,q] z^{(qXP+p)} \\ &= \sum_{p=0}^{P-1}\left(\sum_{q=0}^{Q-1} A[p,q]\left(z^P\right)^q\right) z^p \\ &= \sum_{p=0}^{P-1}\left(\sum_{q=0}^{Q-1} A[p,q]\left(W_Q^k\right)^q\right)\left(w_N^k\right)^p \end{aligned} \qquad k = 0,1,2...N-1. \tag{4.11}$$

The above equation still requires N^2+NP number of arithmetic operations to compute all X_k. Since input vector x is stored column by column in A[P, Q], to reduce the number of arithmetic operations, we can store the output vector X_k of length $N = PXQ$ in the same manner in B[Q, P].

$$
\begin{aligned}
X_k &= B[\hat{p}, \hat{q}] \\
\text{where } k &= \hat{p}Q + \hat{q}\ , W_Q^Q = 1 \text{ and } W_Q^k = W_Q^{\hat{p}Q+\hat{q}} = W_Q^{\hat{q}} \\
X_k &= X_{\hat{p}Q+\hat{q}} \\
&= \sum_{p=0}^{P-1} \left(\sum_{q=0}^{Q-1} A[p,q] \left(W_Q^k\right)^q \right) \left(w_N^k\right)^p \\
&= \sum_{p=0}^{P-1} \left(\sum_{q=0}^{Q-1} A[p,q] \left(W_Q^{\hat{q}}\right)^q \right) \left(w_N^{\hat{p}Q+\hat{q}}\right)^p
\end{aligned}
\tag{4.12}
$$

It is observed that the inner sum in Eq:4.12 is independent of the index $\hat{p}$,hence can be pre-computed. The total arithmetic operations required to perform 2D FFT after transpose is $N(P+Q)$. Thus, storing input vector X in A[P, Q] column by column and storing the output vector X in B[Q, P] has significantly reduced the number of operations computed. Hence, the transpose operation plays an important role in computing a 2D FFT on a vector of size N=PxQ. According to [7, 23], BRAMs or any storage elements are used to perform transpose operations using BRAM or storage element, increasing resource utilization, latency, and design complexity.

4.3 Existing Methods

4.3.1 Data Processing using a sequential FFT IP Core

FFT IP core is a logical unit the FPGA manufacturers provide to perform FFT operations on the signals. There are certain limitations in using the FFT IP core for real-time streaming data in a pipelined structure. It can implement multiple N-point FFTs on multiple-channel data considering a single channel at a time. However, it cannot implement a single N-point FFT on multiple channels combined. In radar application, as mentioned in table 1.1, 80 bits of parallel data (i.e., 8 samples) is received at 168.75 MHz. To process the incoming data, we need to convert parallel data to serial and perform FFT calculations at a higher rate, as shown in the Fig:4.6. In our application, we are receiving 8 samples per clock cycle. After converting into serial data, the data rate of serial output should be 8 times the input data rate to avoid data loss. Therefore, the FFT IP core should operate at 8*168.75 MHz which is practically not possible due to the limitations on FPGA processing speed and resource availability[42].

If the FFT is to be computed at the same input data frequency, the excess data must be stored in a memory element such as BRAM/FIFO and operate the IP core in streaming mode, as shown in Fig:4.6. As per the requirements in table 1.1, we must process 1350000 samples without any loss to achieve the functionality, which is not possible due to the resource constraints.

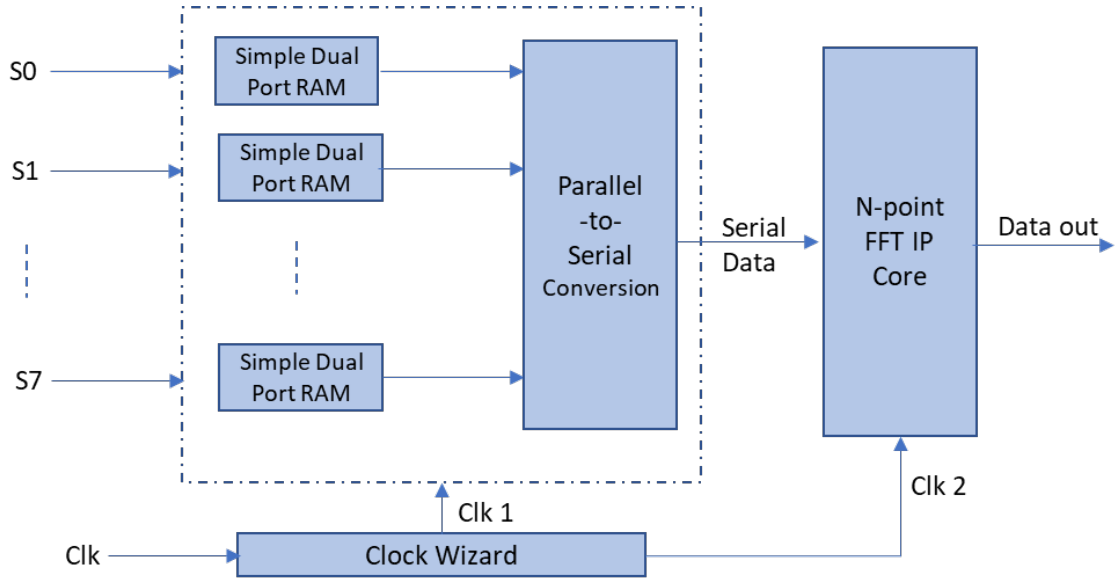


Figure 4.6: Implementation FFT using Conventional Xilinx FFT IP on parallel incoming data

4.3.2 Interpreting 1D as 2D signal

By storing 1D signal of N point data as 2D signal of N_1xN_2 point data, we can perform 2D FFT on it to get N point FFT output. The 2D FFT of a 2D array or a matrix can be defined by the Eq:4.13, where x_{l_1,l_2} is the data point stored on row l_1 and column l_2. This implies that 2D FFT can be realised as multiple 1D FFTs on rows & columns of the matrix with computing cost of $O(N_1N_2log_2(N_1N_2))$ or $O(N^2log_2(N))$ when N_1 = N_2 = N. Here twiddle factors are derivatives of $W^l_{N_1}$ and $W^l_{N_2}$ but for the calculation of N point 1D FFT using this equation, it should be the derivative of W^l_N[16].

$$X_{k1,k2} = \sum_{l_1=0}^{N_1-1} \left(\sum_{l_2=0}^{N_2-1} x_{l_1,l_2} W_{N_2}^{k_2 l_2} \right) W_{N_1}^{k1l1} \qquad k_1, k_2 = 0, 1, 2...N-1. \tag{4.13}$$

4.3.2.1 Transpose Split Method

The Transport Split method is based on the row-column 2D FFT algorithm, where multiple independent processors are assigned to each row or column, as shown in the Fig:4.7 [9]. Once the FFT operation is performed in one direction, the data is re-aligned, and another set of 1D FFT is performed on transposed data in the same direction. One significant drawback of this architecture is that it only applies to square matrices. Due to the use of multiple threads/processors, the architecture has more resource utilization, and latency increases due to the use of storage elements to perform transpose operations.

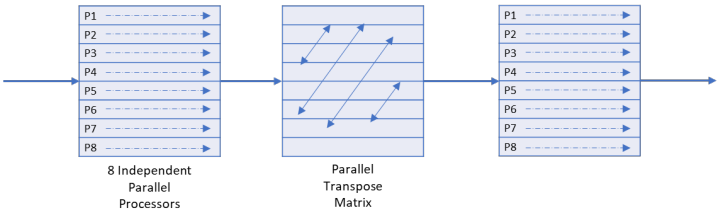


Figure 4.7: Architecture of Transpose split method with 8 parallel data channels implementing a 8x8 2D FFT

4.3.2.2 Six-Step Algorithm

This algorithm is best suited for distributed systems where the algorithm decomposes N-point FFT into N_1 point & N_2 point FFTs [4, 28]. Here, the input data is distributed into parallel channels and process a small portion of row-wise data simultaneously. Later, the data is transposed and perform a column-wise data. This data is again converted to a 1D signal. The six-step algorithm is designed to exploit parallelism in FFT to enhance the speed and efficiency of processing large data sets.The implementation of this algorithm depends upon various factors like hardware architecture, size of the data set, data rates etc. As depicted in the Fig:4.8,the six-step algorithm is implemented as follows:

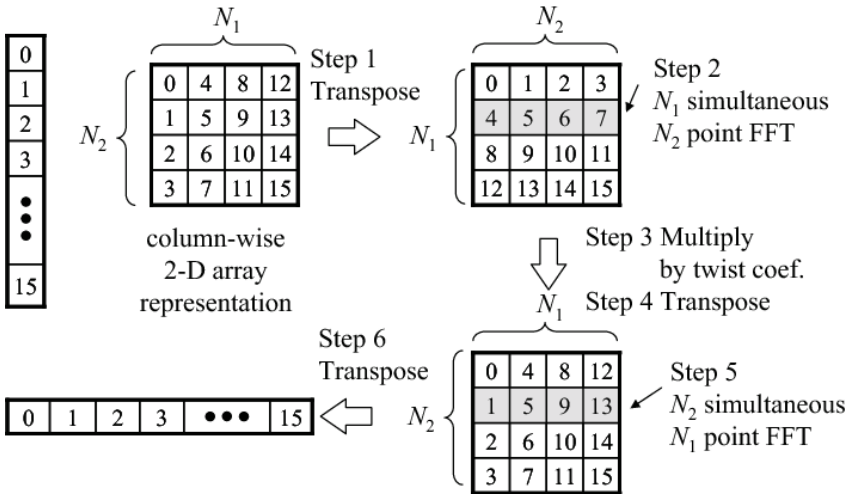


Figure 4.8: Architecture of Transpose split method with 8 parallel data channels implementing a 8x8 2D FFT

- Transpose the input data set ($N_1 \times N_2$) to ($N_2 \times N_1$).
- Perform N_1 individual N_2 point 1D FFT.
- Multiply the resultant $N_2 \times N_1$ complex matrix by W_n^k

- Transpose resultant $N_2 \times N_1$ matrix to $N_1 \times N_2$.
- Perform N_2 individual N_1 point 1D FFT.
- Transpose resultant $N_1 \times N_2$ matrix to $N_2 \times N_1$.

4.4 Proposed Method

In the proposed method, we have addressed all the issues that are faced in the existing methods for calculating real-time FFT on very high-speed ADC data for wideband radar systems on FPGA. We have implemented a parallel pipelined architecture that processes multiple inputs simultaneously to avoid the usage of FIFOs for parallel to serial data conversion and, hence, the data loss. To eliminate the use of transpose operations on the input and the output stage as in the six-step algorithm, we have arranged the input data in column-major format and accepted output data in row-major format to further reduce resource utilization and latency. We have used an external phase correction module to reuse Xilinx's FFT IP Cores, hence enabling the ease of re-configuring the FFT length. As the radar pulse parameter algorithm is independent of the order of the data in a window, we have replaced the transpose operation between both dimensions of FFT like in six step algorithm, with a simple parallel to serial data converter and moved it from the FFT algorithm to the peak detector modules.

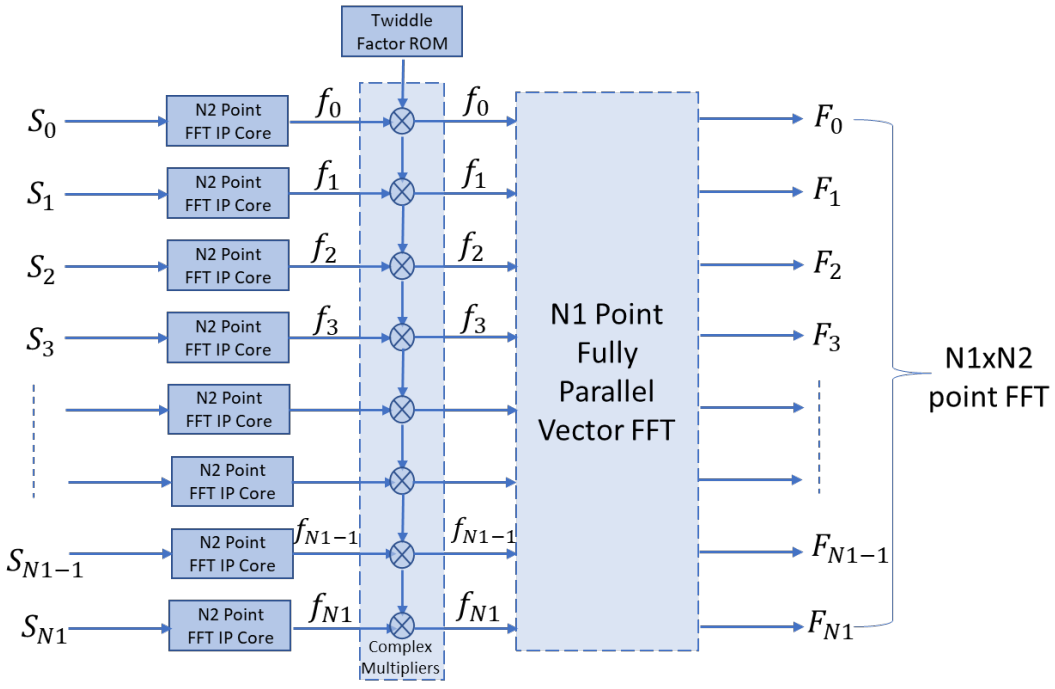


Figure 4.9: Implementation of a 512-point FFT by processing 64 data frames on 8 parallel channels using the proposed architecture.

If an input data of length $N = N_1 \times N_2$, is stored in a 2D array $A[N_1, N_2]$ in a column major format such that $a_l = A[n_1, n_2]$ where $l = n_1 + n_2 \times N_1$, the N-point DFT of the array A can be given by Eq:4.1 as,

$$b_r = \sum_{l=0}^{N-1} a_l (W_N^r)^l \tag{4.14}$$

which can be further written as,

$$b_r = \sum_{n_1=0}^{N_1-1} \left(\sum_{n_2=0}^{N_2-1} a_{n_1+n_2N_1} (W_{N_2}^r)^{n_2} \right) (W_N^r)^{n_1} \tag{4.15}$$

which is still an $\Theta(N^2)$ algorithm as it requires N_2 multiply-adds to compute each inner sum element and N_1 multiply-adds to compute each b_r. Thus, requiring total of $N(N_1 \times N_2 + N_1)$ multiply-adds operations. But if we accept the DFT output in a row major format in a 2D array $B[N_1, N_2]$ such that $b_r = B[k_1, k_2]$ where $r = k_1 \times N_2 + k_2$[15], the N-point DFT of the array A now can be given by rewriting Eq:4.15 as,

$$\begin{aligned} b_{k_1N_2+k_2} &= \sum_{n_1=0}^{N_1-1} \left(\sum_{n_2=0}^{N_2-1} a_{n_1+n_2N_1} (W_{N_2}^{k_1N_2+k_2})^{n_2} \right) (W_N^{k_1N_2+k_2})^{n_1} \\ &= \sum_{n_1=0}^{N_1-1} \left(\sum_{n_2=0}^{N_2-1} a_{n_1+n_2N_1} (W_{N_2}^{k_2})^{n_2} \right) (W_N^{k_1N_2+k_2})^{n_1} \end{aligned} \tag{4.16}$$

in which inner sum is independent of the index k_1 and hence it can be pre-computed for all n_1 and k_2 and requires only total of $N(N_1 + N_2)$ multiply-adds operations. On further simplifying Eq:4.16, we can write

$$\begin{aligned} b_{k_1N_2+k_2} &= \sum_{n_1=0}^{N_1-1} (W_N^{k_2})^{n_1} \left(\sum_{n_2=0}^{N_2-1} a_{n_1+n_2N_1} (W_{N_2}^{k_2})^{n_2} \right) (W_{N_1}^{k_1})^{n_1} \\ &= DFT_{N_1}(TF_N \times DFT_{N_2}(a_{n_1+n_2N_1})) \end{aligned} \tag{4.17}$$

which can be used to explain the three main steps of the proposed architecture. The first step is to compute N_1 N_2-point DFTs, multiplying the results with Twiddle Factors (TF) of derivative N and then computing N_2 N_1-point DFTs on the results. In order to reduce the arithmetic complexity, we used FFTs over DFTs. For first step, we used sequential 1D DIF-FFT using fully pipelined and radix-2 based algorithm as given in Eq:4.2. The rotation vectors/twiddle factors were pre-calculated and stored in ROM for multiplication with the intermediate FFT results. For final step, we used parallel 1D DIT-FFT using fully pipelined and radix-2^2 based algorithm as given in Eq:4.5 to generate the final $N_1 \times N_2$ point FFT, as shown in the Fig:4.9. Here, the N_1 dimension constitutes the parallelism factor of the architecture, and the N_2 dimension is chosen based on the number of frames received in one window. To meet the requirements of the given radar signal processing application, as mentioned in the table 1.1, we implemented a 512-point FFT using the proposed architecture.

4.4.1 Significance of N_1 & N_2 dimensions:

Using the row-column decomposition technique, N is divided into $N_1 \times N_2$ where N_1 constitutes to the parallelism factor and N_2 determines the frame count in one window. According to Fig4.10, the optimal frame size to perform 512-point FFT of 1D signal using 2D arrangement consuming minimal resources is 16x32 or 8x64. Increasing N_1, which is the parallelism factor, increases overall resource utilization of the system and complexity of parallel 1D FFT used to calculate N_2 N_1-point FFTs, but decreases overall latency of the system and vice-versa for increasing N_2, i.e., the number of frames in one window. Since we receive 8 samples of data per clock cycle, we have 8 parallel channels (N_1) and perform a 64-point FFT on each channel. Since the ADC is configured to 1350 MHz sample frequency to compute 8 parallel channels, the maximum operable frequency for an 8×64 point FFT is 168.75 MHZ.

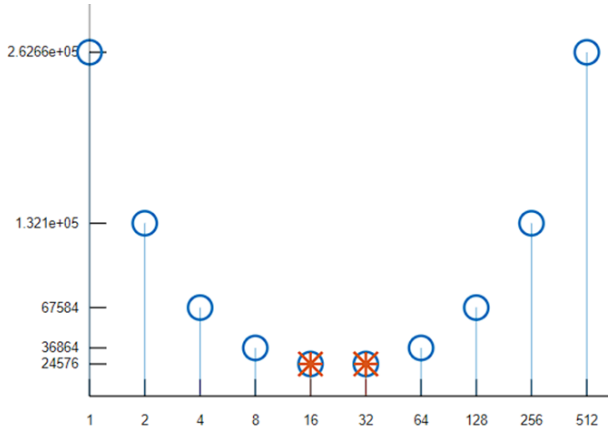


Figure 4.10: Comparison of total number of arithmetic operations of DFT required for various values of N_1 & N_2

Chapter 5

Experimental Setup

The functionality of the proposed system is tested by first implementing in MATLAB and later implementing it on hardware.

5.1 MATLAB Implementation

We have developed the direction-finding algorithm in MATLAB, as shown in Fig.5.1 We have tested the code by simulating the dense radar environment and feeding it as an input to the system. The algorithms were fed time of arrival (TOA) values of pulses from the simulated EW environments in sequential order. The parameters of the EW environments were modified to see how they affected the algorithm's performance. An EW receiver cannot completely intercept all pulses in congested EW surroundings, referred to as missing pulses[1]. Spurious pulses are pulses received by the EW receiver but are not of interest. The percentage of random pulses associated with each emitter not detected by the EW receiver (missing pulses) varied from zero to 10 percent in the simulated EW environments. Missing pulses owing to an antenna beam rotating are not considered here.

As a part of our project, we have constructed a complex virtual radar environment consisting up to 6 radar emitters with different pulse widths, DoA, PRI, and PRI-type, interleaved together through a single channel. The interleaved signal is passed through the algorithm, and we determine the DoA of each emitter simultaneously. When testing the deinterleaving performance of algorithms, the number of emitters in the environment was varied between 1 and 6. Each emitter transmitted 10 times the maximum time period of all emitters.

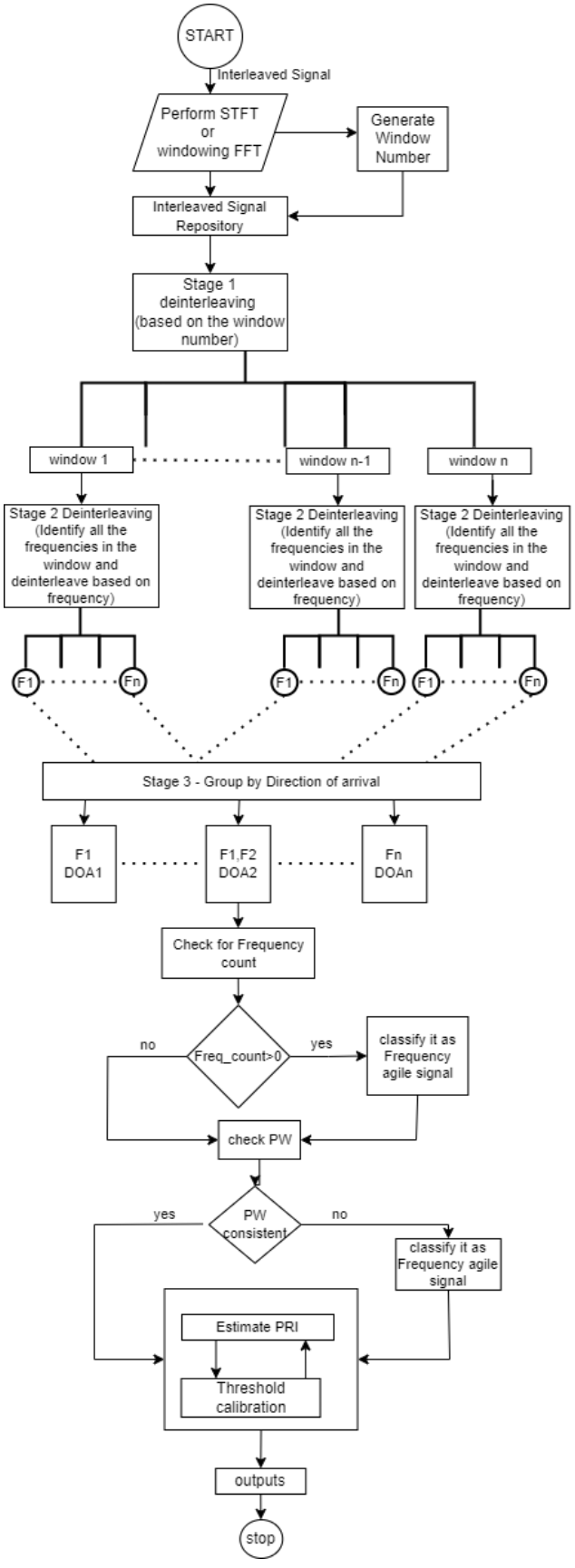


Figure 5.1: Workflow of the architecture in MATLAB

5.2 Hardware Implementation

Once we have verified the algorithm's functionality in MATLAB, we have replicated the system on hardware using verilog HDL. Each block of the system was implemented and verified individually. Later combined all the blocks as an integrated system and tested by feeding the inputs in 3 different methods.

5.2.1 Static Data Testing

In this mode of testing, we implemented the system on Xilinx Virtex 7 FPGA VC707 evaluation platform and tested its functionality by feeding the Matlab simulated static data. Here, we have converted the MATLAB input into a 32 bit data (16-bit real & 16-bit imaginary) and stored into a BRAM to feed it to other modules at 100 MHz clock frequency. The data set consists of 6 emitters ranging from frequencies 750 MHz to 1250 MHz, interleaved with different pulse widths and different PRI Schemes. To satisfy the requirements given in the table 1.1, we have taken up-to 2-lakh samples to verify the functionality. The results obtained from the static input data is discussed further in section.

5.2.1.1 Challenges faced during Static Data testing

Data of width 32 bits is given as input to FFT IP using a single port BRAM at 200 MHz clock frequency. Since we are trying to read a huge number of samples $(around - 2lakh)$ at a very high speed, the data is getting corrupted while reading from BRAM. We have resolved this issue by reducing the clock frequency and improving synchronization.

5.2.2 Radiation Mode Testing

In radiation mode testing, the input is fed to a horn antenna setup in a radiation chamber using a signal generator. As the horn antennae radiate the signals, the set of Vivaldi antennas of the system receives the incoming RF signals, down-convert it into the IF signal, and sends it to the ADC module. Here the digitized input is further fed to the signal processing board to test its functionality. The Injection mode testing was carried out for up to 2 emitters in-band as well as out-of-band -500 MHz. A continuous test signal and pulsed test signal with different PRI values are fed, and the results obtained are discussed further in the section.

5.2.2.1 Challenges faced during Real-time Integration

Here, in our system, we are sampling the incoming signal with a 168.75 MHz sampling frequency and generating 80 bits of data per second. Since the sampling rate is very high and 80 bits of data is sent at an instance, we cannot store such a large amount of data in a FIFO or BRAM due to FPGA resource constraints.

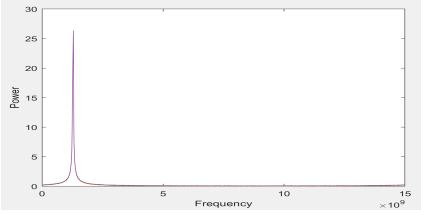
Chapter 6

Implementation Results

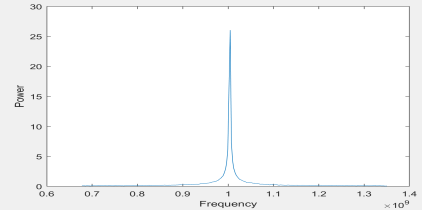
This section presents some MATLAB simulation results to demonstrate the FFT architecture functionality and the system's functionality. Later, we also presented the hardware implementation results of the FFT architecture & ESM, on two boards: 1) Xilinx Virtex 7 FPGA evaluation board using static MATLAB inputs, 2) custom board by providing input using a signal generator (Injection Mode Testing) and using Horn Antenna (Radiation Mode Testing).

6.1 Parallel FFT Implementation Results

In our tests, according to table1.1 requirements, the incoming data is sampled at 1.350 Gsps, and data is sent in 8 parallel channels to the FPGA at a 168.75 MHz frequency. Each parallel data channel is processed into a single channel, and an FFT operation is performed for this single channel in MATLAB (shown in Fig:6.1a) using the FFT IP. We also realized the 8-channel proposed architecture in MATLAB (shown in Fig:6.1b). We compared the MATLAB results of both the conventional FFT method and the proposed architecture (shown in Fig:6.1) and later implemented it on an FPGA. Before working with real-time data, we simulated a radar environment in MATLAB and generated a set of static data, which we used to test the system functionality.



(a) A direct 512-point FFT result in MATLAB



(b) MATLAB simulation result of 512-point FFT using parallel architecture

Figure 6.1: MATLAB simulation results of implementing a 512-point FFT. The peak is obtained at the 1 GHz frequency

Fig:6.2(a,b,c) show the implementation of 512-point FFT on FPGA using conventional Xilinx IP-core method using MATLAB generated static data as well as the proposed architecture using both static data and real-time data. Here, we can see that the peaks are obtained at the respective frequency index of 1 GHz input signal. Since the incoming data rate of the static data is very low, we were able to achieve system functionality by implementing a 512-point FFT using Xilinx logic-core IP. However, it was not applicable for real-time data, as the inputs are sampled at a much faster rate, and storing the samples has become a challenging task. Fig:6.2 shows that the proposed architecture is able to implement the FFT and has results similar to the conventional FFT.

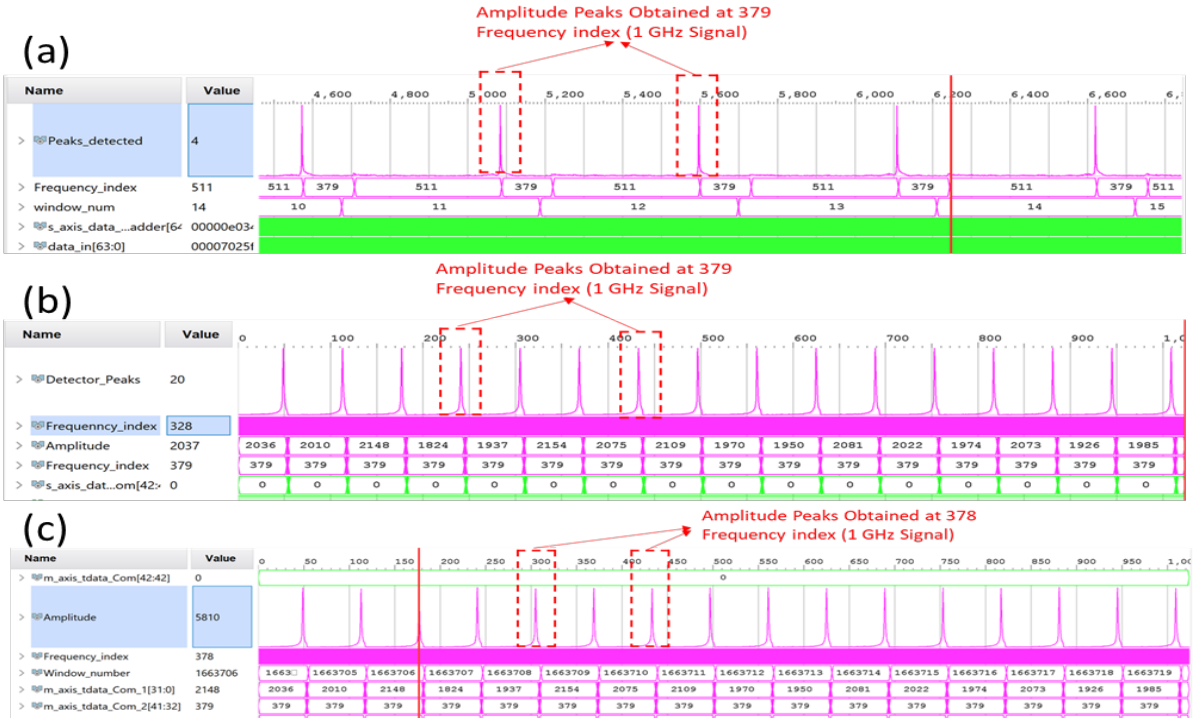


Figure 6.2: Comparison of the FPGA Implementation results implementing 512-point FFT using (a) Xilinx IP-core with Static data (b) Proposed architecture with Static data (c) Proposed architecture with Real-time data. Here The peaks are obtained at frequency index of 1 GHz signal

In Fig:6.3(a), $FREQ\_sfifo$ signal shows that for a few window numbers (given in Win_num_sfifo), a few data packets are missing due to high input writing speed, which causes the FIFO to overflow. The proposed architecture successfully solves the problem, as seen in Fig: 6.3(b). Here, $FREQ\_sfifo$ is continuous for all the window numbers (given in Win_num_sfifo). We may, therefore, assert that the proposed architecture can handle real-time data processing.

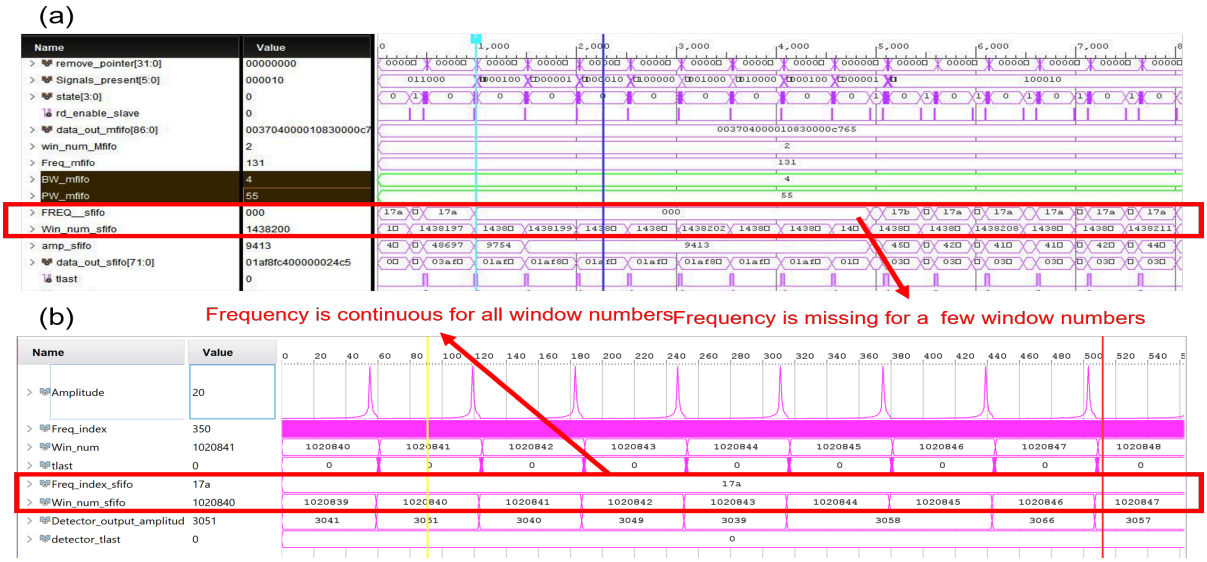


Figure 6.3: Implementation results of both Xilinx IP-core based method (a) explaining the data loss due to the FIFO overflow and proposed architecture (b) resolving the issue.

Table 6.1: Comparison of latency & the no. of clock cycles required to compute 1 512-point FFT

Architecture	**No. Of Clock Cycles**	**Latency**	**real-time**
Conventional Xilinx IP-based FFT	1133 cycles	6.744 μs	real-time
Proposed parallel FFT	220 cycles	1.309 μs	real-time

On a Virtex-7 FPGA board, we implemented a 512-point FFT using both the traditional IP core and the proposed architecture methods. Table6.1 presents a comparison of the processing speeds of two designs. The processing speed (time from the first raw data entering to the last transformation data output leaving) is expressed by the number of clock cycles, and the total time is calculated by a working clock of 168.75 MHz. Table 6.1 shows that the processing speed of the proposed parallel architecture is 913 clock cycles faster than that of the IP-based FFT [31].

Using the formulas used in Eq:4.2, Eq:4.5, we calculated the total number of complex multiplications & additions required to implement the proposed architecture. Table6.2 compares the number of arithmetic operations, multipliers, and adders/subtractors required to perform the FFT by the two architectures. As mentioned in the table, the proposed parallel architecture requires fewer arithmetic operations than the conventional Xilinx IP-based FFT implementation. Compared to the IP-based implementation, the proposed parallel architecture requires 240 fewer complex multiplications and 1412 less complex additions.

Architecture	Xilinx IP-based FFT	Proposed parallel FFT	% Improvement
Complex Multiplications	2304	2064	10.42%
Complex Additions	4608	3196	30.64%
Multipliers	9216	8256	10.42%
Adders	13824	10520	23.90%

Table 6.2: Arithmetic operations table

6.2 ESM Architecture Implementation Results

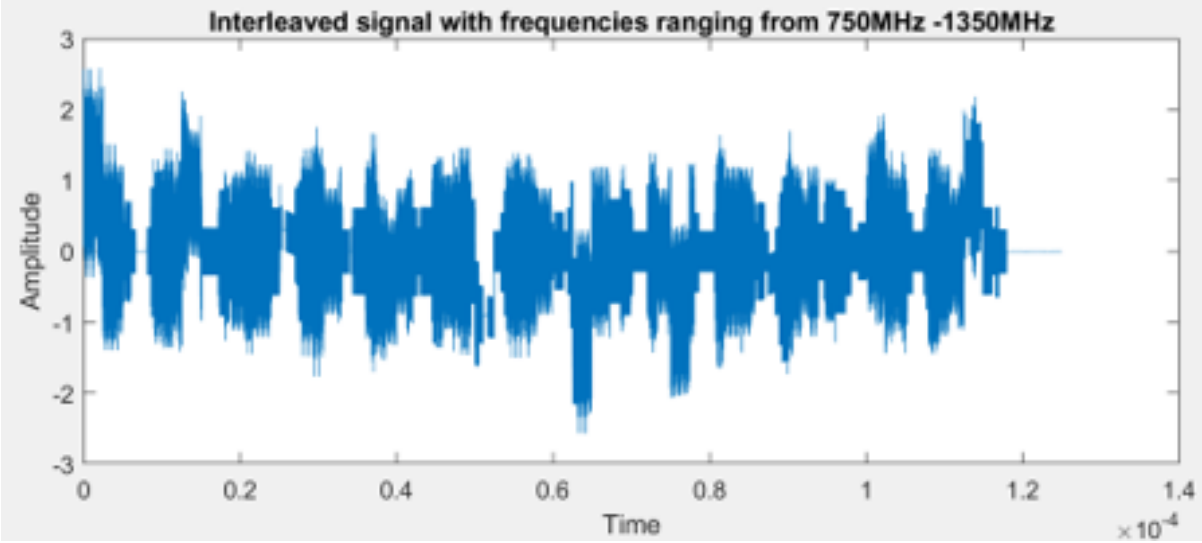
6.2.1 MATLAB Implementation Results:

We tested the system using an interleaved signal comprised of six emitters with different pulse parameters. The table 6.3 lists the characteristics of each emitter.

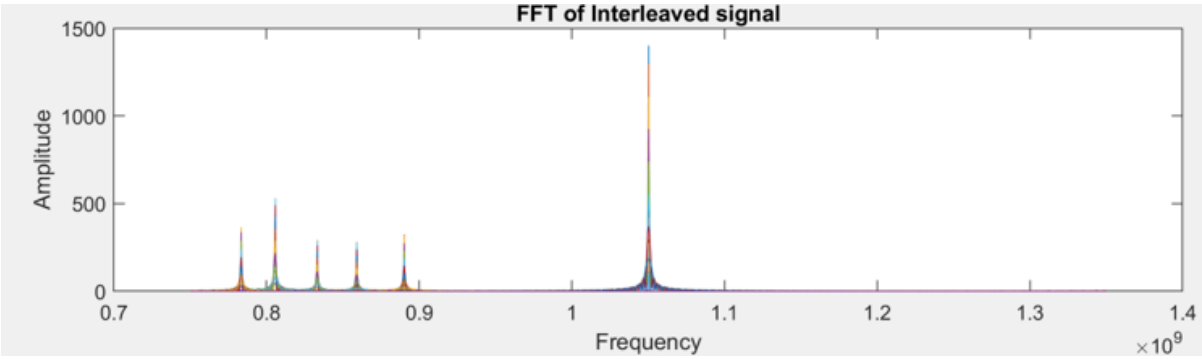
Table 6.3: Static Data Inputs given to the system

Frequency (GHz)	**Pulse Width (μs)**	**PRI (μs)**	**PRI Scheme**
862	6	9	Constant(1)
750	1.3	12	Staggered(2)
920	5	9	Jittered(3)
1100	2.4	12.5	Staggered (2)
800	4.3	8.6	Constant(1)
990	3.3	6.6	Jittered(3)

In the first step, we performed a 512-point windowing FFT on the interleaved signal to get the frequency components. Fig.6.4a gives the time domain representation of an LFM signal in MATLAB. The output of the FFT consists of multiple peaks, and each peak in the FFT output corresponds to an emitter existing at that frequency. Here in Fig.6.4b, we obtained 6 peaks at different frequencies ranging from 750 MHz - 1250 MHz, indicating the presence of 6 emitters in the interleaved signal.



(a) Interleaved LFM Signal



(b) Frequency Response of an Interleaved signal

Figure 6.4: Interleaved LFM signal MATLAB plot and its respective Frequency Response

The amplitudes obtained are compared to the threshold value to filter out valid signals from the FFT result. The valid signals are later sent to the pulse width estimation block and DoA block to obtain the pulse parameters. According to the design flow given in Fig.5.1, the interleaved signal is de-interleaved based on window number, frequency, amplitude, and direction of arrival. Fig.6.5 shows the 6 de-interleaved emitters. The de-interleaved signals are sent simultaneously to the PRI estimation block to estimate the PRI & PRI type of the signal. Here, we evaluated the system for several test cases, including PRI, pulse-on-pulse scenarios, two emitters with the same pulse width, etc., and obtained promising results. Table 6.4 summarizes the outputs obtained for the inputs given in the table 6.3.

Since we are de-interleaving the system based on window number, DoA & Frequency, we can also identify frequency-agile radars. The total simulation time was further reduced by performing PRI estimation on each de-interleaved signal simultaneously.

Table 6.4: Outputs obtained from the MATLAB Implementation of the Architecture

Window Number	Frequency (MHz)	Pulse Width (μs)	Amplitude (DBm)	DoA (deg)	TOA (μs)	PRI (μs)	PRF (GHz)	PRI TYPE
1	751.46	1	49.1	4.97	0	12	0.083	2
1	801.56	4.2	49.2	20.45	0	8.6	0.116	1
1	863.5	5.5	49.1	-12.64	0	9	0.111	1
1	921.5	4.5	48.8	-18.35	0	7.8	0.128	3
18	991.4	3	49.4	-2.81	1.7	7.3	0.136	3
1	1100.8	2.2	49.4	9.42	0	12.5	0.08	2

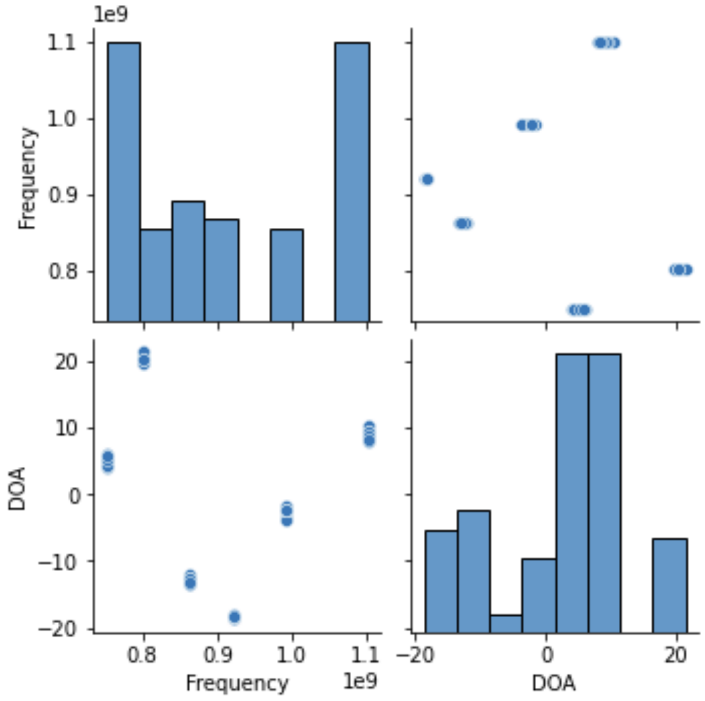


Figure 6.5: De-interleaving of emitters based on the frequency and DoA. Here, the cluster of dots represent the presence of each emitter

6.2.2 Hardware Implementation Results:

6.2.2.1 Static Data Test Results

We have implemented the system on FPGA and tested the system with the same set of inputs as given in table 6.3. We have performed a 512-point FFT having a window length of 512 samples which constitutes 379.5 ns. The frequency of an emitter based on the frequency index can be achieved using the eq:6.1.

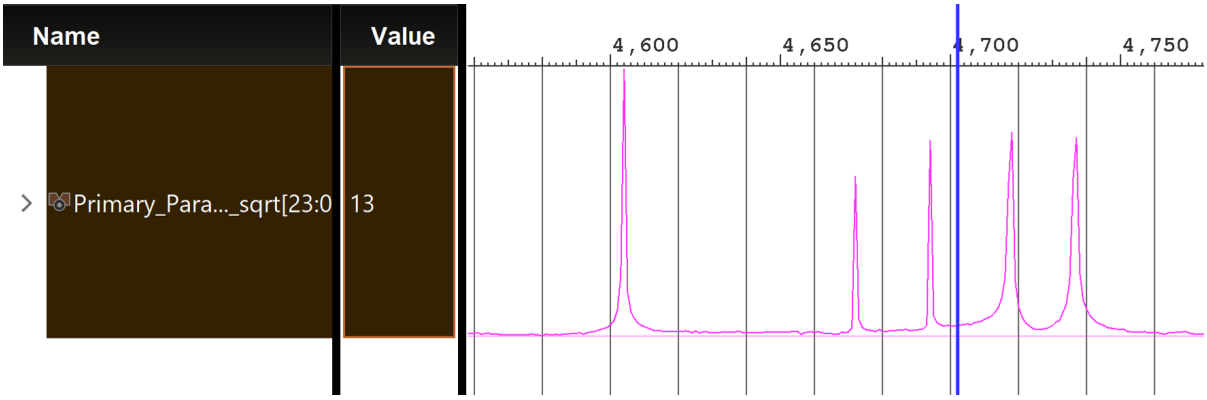
$$f_{index} = 256 \pm \frac{(f - 675MHz)}{2.67MHz}$$

$$f = 675MHz \mp (f_{index} - 256) * 2.67MHz \tag{6.1}$$

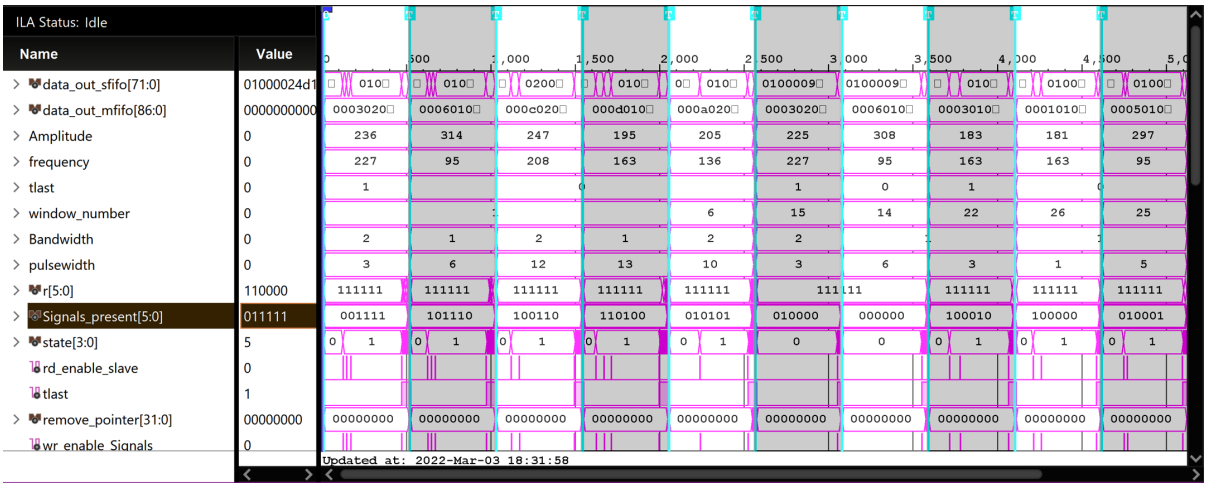
Fig.6.6a, shows the peaks obtained after performing the FFT operation, indicating the presence of emitters. Here, Outputs are obtained at frequency indexes 95,163,185,209,228,136 whose corresponding frequencies are 1104 MHz, 923 MHz, 864 MHz, 800 MHz,749 MHz, 995 MHz respectfully. Fig.6.6b gives the estimated pulse width of the input signal. The pulse widths obtained for the frequencies at frequency indexes 95,163,185,209,228 is 6, 13, 15, 12, 3 respectively. The output of the PRI module is shown in Fig.6.6c, where the PRI of the signals at indexes 95,163,185,209,228,136 is 32, 13, 26, 36, 32, 22 respectively. The overall inputs and outputs are summarized in the table 6.5. Thus, we concluded the designed system is working for a given set of test inputs and verified with the MATLAB results.

Table 6.5: Comparision of the Static Data Inputs and Outputs captured using ILA

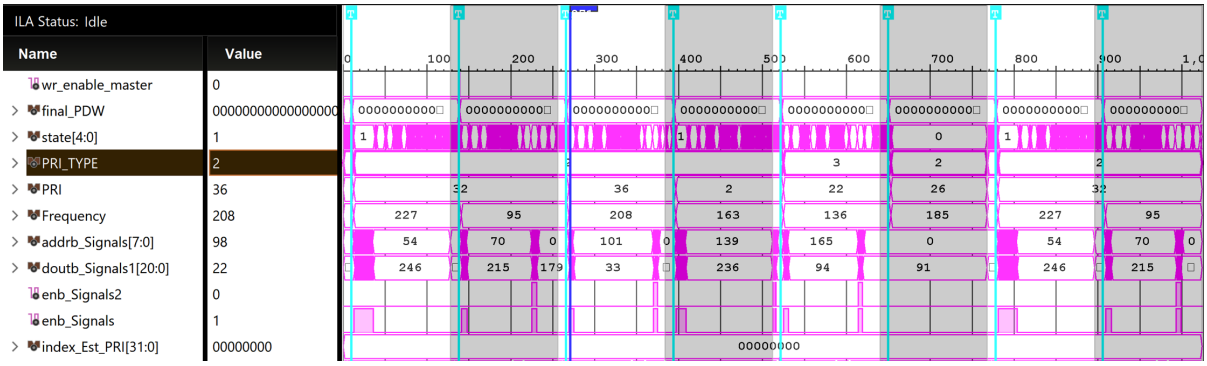
	Inputs			Outputs		
Frequency (MHz)	Pulse Width (μs)	PRI (μs)	PRI TYPE	Pulse Width (μs)	PRI (μs)	PRI TYPE
751.46	1.3	12	2	1	12	2
801.56	4.3	8.6	1	4.2	8.6	1
863.5	6	9	1	5.5	9	1
921.5	5	9	3	4.5	7.8	3
991.4	3.3	6.6	3	3	7.3	3
1100.8	2.4	12.5	2	2.2	12.5	2



(a) Output of FFT IP-core indicating the presence of Various emitters. Each emitter is represented by a peak.



(b) ILA capturing the Pulse width of the emitters present in the LFM. Here, pulse width is represented as multiples of window length (379.5 ns)



(c) ILA capturing the PRI of the emitters present in the LFM. Here, PRI is represented as multiples of window length (379.5 ns)

Figure 6.6: Hardware Implementation of the ESM System using Static Data

6.2.2.2 Real-time Data Test Results

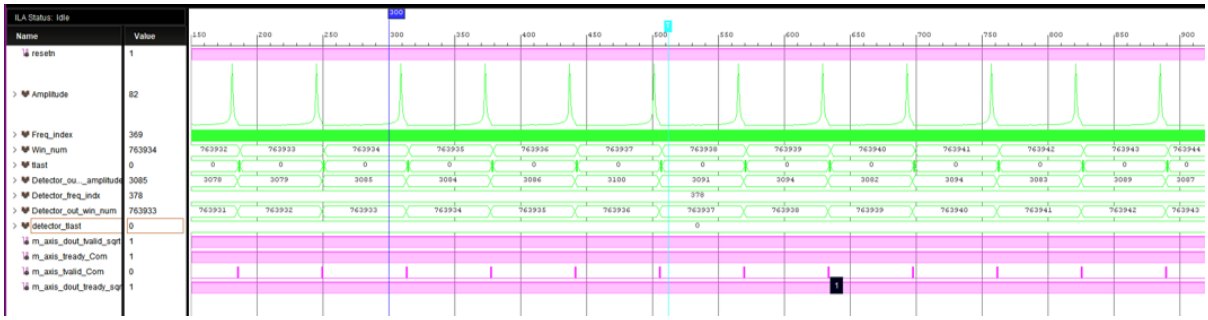
Once the functionality of the system is proved using the static data, we tested the system by passing a stream of real-time data using a signal generator (injection mode testing) and using a horn antenna (radiation mode testing). Due to the unavailability of a multi-emitters simulator, we have tested the system with only single emitter frequency. Using our new proposed parallel-FFT architecture, we were able to implement 512-point FFT on the incoming real-time data stream.

Table 6.6: Comparison of inputs and outputs of a signal by varying Pulse width and PRI

	INPUTS			OUTPUTS		
	Pulse Width (μs)	PRI (μs)	PRI Type	Pulse Width ($\times$ 379ns)	PRI ($\times$ 379ns)	PRI TYPE (1-constant)
LOW PRI	0.5	1	Constant	1	2	1
	1	2	Constant	3	4	1
	5	10	Constant	13	26	1
MEDIUM PRI	10	20	Constant	26	52	1
	15	30	Constant	40	78	1
	15	45	Constant	40	118	1
HIGH PRI	60	120	Constant	158	316	1
	45	120	Constant	119	316	1
	200	400	Constant	527	1054	1

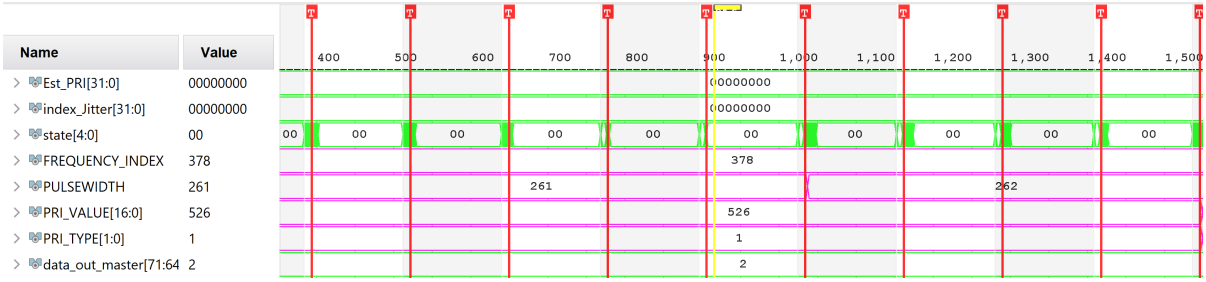
The peaks obtained at frequency index 378 in fig. 6.7a indicate the presence of an emitter with 1 GHz frequency. These IF frequencies are converted into RF frequencies by using LO frequency & band information. Initially, a continuous sinusoidal signal is fed as an input to the system. The check_cw flag in fig. 6.7b indicating the incoming signal is from a continuous radar. Later, a 1 GHz pulsed signal with pulse width 100 μs and PRI 200 μs is sent to the receiver. The pulse width and PRI estimated by the system are captured by an ILA and is given in fig. 6.7c. The ILA probes out the pulse width as 262 and PRI as 526, corresponding to 99.46 μs and 199.61 μs, respectively as shown in fig. 6.7c. The system is also tested by varying the signal pulse width & PRI values and are tabulated in table 6.6

6.2.3 FFT architecture Results



(a) Output of FFT IP-core indicating the presence of an emitters, represented by a peak.

(b) ILA capturing the Pulse width of the continuous signal. Here, Check_cw flag is high indicating the emitter as continuous radar



(c) ILA capturing the Pulse width and PRI of the emitters present in the LFM. Here, pulse width & PRI is represented as multiples of window length (379.5 ns)

Figure 6.7: Hardware Implementation of the ESM System using real-time Data

Chapter 7

Conclusions and Future Work

The ESM system presented in the paper is designed to process real-time data and estimate the direction of arrival of multiple emitters simultaneously with an absolute error of less than 1 degree, a frequency resolution of 2.67 MHz, and a pulse width resolution of 379 nanoseconds. The system achieves this by implementing an N-point FFT on multiple parallel channels, allowing for efficient incoming signal processing. One key innovation in the system is its FFT architecture, which can implement a 512-point FFT five times faster than a conventional IP-based approach while requiring fewer arithmetic operations. The new FFT architecture makes the system more efficient and faster than other existing approaches. Another critical aspect of the system is its use of a linear regression algorithm to obtain the monopulse ratio curve (MRC), which is used to calculate the direction of arrival. This approach helps to reduce the absolute error to less than 1 degree, making the system highly accurate. The system has been implemented on a Virtex-7 FPGA, and its functionality has been verified using both static and real-time data. Overall, the ESM system presented in the paper is an innovative and efficient approach to processing real-time data and estimating the direction of arrival of multiple emitters simultaneously.

In conclusion, the ESM system presented in this paper provides a significant improvement in the real-time processing of data and estimation of the direction of arrival of multiple emitters. However, there is still scope for further research and development in this field. One potential area for future exploration is the incorporation of machine learning algorithms to improve the accuracy and efficiency of the system further. Additionally, the system's performance could be evaluated under more challenging conditions, such as in the presence of interference or in a dynamic environment. Overall, the ESM system has the potential to benefit a wide range of applications, including military surveillance, radar systems, and telecommunications, and future research in this field can lead to even more significant advancements.

Related Publications

B. L. Pradeep, R. Anand, P. Vadakattu, S. Azeemudin and A. Ahmed, ”Design and Implementation of a Real-time Parallel FFT for a Direction-Finding System on an FPGA,” 2022 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 2022, pp. 1-7,
doi: 10.1109/HPEC55821.2022.9926310.

Bibliography

[1] K. Abdul Ghani, K. Dimyati, A. Sha'ameri, and N. G. Nik Daud. Statistical modeling for missing and spurious pulses in pulse repetition interval (pri) analysis. volume 9, pages 18–27, 01 2016.

[2] D. Adamy. *EW 102: A Second Course in Electronic Warfare*. 2003.

[3] T. G. Anitha and S. Ramachandran. Novel algorithms for 2-D FFT and its inverse for image compression. In *International Conference on Signal Processing, Image Processing and Pattern Recognition 2013, ICSIPR 2013*, volume 1, 2013.

[4] D. H. Bailey. Ffts in external or hierarchical memory. In *Supercomputing '89:Proceedings of the 1989 ACM/IEEE Conference on Supercomputing*, pages 234–242, 1989.

[5] A. J. Barabell, V. Jw, C. P. F. Delong, and K. D. Senne. Performance comparison of super resolution array processing algorithms. page 193, 1998.

[6] E. Ben-Ari and J. Remez. Performance verification of a multimodal interferometric doa-estimation antenna. *IEEE Antennas and Wireless Propagation Letters*, 10:1076–1080, 2011.

[7] N. Bhagat, D. Valencia, A. Alimohammad, and F. Harris. High-throughput and compact FFT architectures using the Good-Thomas and Winograd algorithms. *IET Communications*, 12(8):1011–1018, 2018.

[8] J. S. Bruno, V. Almenar, and J. Valls. FPGA implementation of a 10 GS/s variable-length FFT for OFDM-based optical communication systems. *Microprocessors and Microsystems*, 64:195–204, feb 2019.

[9] C. Calvin. Implementation of parallel FFT algorithms on distributed memory machines with a minimum overhead of communication. *Parallel Computing*, 22(9):1255–1279, 1996.

[10] S. camlica. Pri tracking using adaptive fir filter. In *2014 22nd Signal Processing and Communications Applications Conference (SIU)*, pages 886–889, 2014.

[11] R. Chen, N. Park, and V. K. Prasanna. High throughput energy efficient parallel FFT architecture on FPGAs. *2013 IEEE High Performance Extreme Computing Conference, HPEC 2013*, 2013.

[12] C. H. Cheng. Wideband receiver with I/Q channels for multiple Nyquist zone coverage. *IET Radar, Sonar and Navigation*, 9(4):421–428, apr 2015.

[13] C. H. Cheng, D. M. Lin, L. L. Liou, and J. B. Tsui. Electronic warfare receiver with multiple FFT frame sizes. *IEEE Transactions on Aerospace and Electronic Systems*, 48(4):3318–3330, 2012.

[14] J. W. Cooley and J. W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation*, 19(90):297, 1965.

[15] C. de Boor. Fft as nested multiplication, with a twist. *Siam Journal on Scientific and Statistical Computing*, 1:173–178, 1980.

[16] G. Fabbretti, A. Farina, D. Laforenza, and F. Vinelli. Mapping the synthetic aperture radar signal processor on a distributed-memory MIMD architecture. *Parallel Computing*, 22(5):761–784, 1996.

[17] M. Garrido. A Survey on Pipelined FFT Hardware Architectures. *Journal of Signal Processing Systems*, (March), 2021.

[18] K. George and C.-I. H. Chen. Multiple signal detection digital wideband receiver using hardware accelerators. *IEEE Transactions on Aerospace and Electronic Systems*, 49(2):706–715, 2013.

[19] M. Groden and L. Raffaelli. High speed digital receivers for electronic warfare applications. *16th International Conference on Microwaves, Radar and Wireless Communications, MIKON 2006*, pages 142–144, 2006.

[20] P. I. A. Hansson and P. Hansson. Analysis of some methods for deinterleaving of pulse trains Analysis of some methods for deinterleaving of pulse trains. 2007.

[21] M. Jamali, J. Downey, N. Wilikins, C. R. Rehm, and J. Tipping. Development of a FPGA-based high speed FFT processor for wideband direction of arrival applications. In *IEEE National Radar Conference - Proceedings*, 2009.

[22] P. LACOMME, J.-P. HARDANGE, J.-C. MARCHAIS, and E. NORMANT. Design overview. *Air and Spaceborne Radar Systems*, pages 371–402, 1 2001.

[23] L. Li and A. M. Wyrwicz. Parallel 2D FFT implementation on FPGA suitable for real-time MR image processing. *Review of Scientific Instruments*, 89(9), 2018.

[24] Q. Li, X. Zhang, and H. Li. Online direction of arrival estimation based on deep learning. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2616–2620, 2018.

[25] S. Lin, M. W. Thompson, S. Davezac, and J. C. Sciortino. Comparison of time of arrival vs. multiple parameter based radar pulse train deinterleavers. In *SPIE Defense + Commercial Sensing*, 2006.

[26] K. Manickchand, J. Strydom, and A. Mishra. Comparative study of toa based emitter deinterleaving and tracking algorithms. In *2017 IEEE AFRICON*, pages 221–226, 2017.

[27] H. K. Mardia. New techniques for the deinterleaving of repetitive sequences. *IEE proceedings. Part F. Communications, radar and signal processing*, 136:149–154, 1989.

[28] H. Nakahara, K. Iwai, and H. Nakanishi. A high-speed FFT based on a six-step algorithm: Applied to a radio telescope for a solar radio burst. *FPT 2013 - Proceedings of the 2013 International Conference on Field Programmable Technology*, pages 430–433, 2013.

[29] K. Nguyen, J. Zheng, Y. He, and B. Shah. A high-throughput, adaptive FFT architecture for FPGA-based space-borne data processors. In *2010 NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2010*, pages 121–126, 2010.

[30] J. R. Pennington. Radar Signal Characteristic Extraction with FFT-based Techniques. pages 1–58, 2011.

[31] B. L. Pradeep, R. Anand, P. Vadakattu, S. Azemuddin, and A. Ahmed. Design and implementation of a real-time parallel fft for a direction-finding system on an fpga. In *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7, 2022.

[32] B. D. Rao and K. V. Hari. Analysis of subspace-based direction of arrival estimation methods. *Sadhana*, 16:183–194, 11 1991.

[33] R. Roy and T. Kailath. Esprit-estimation of signal parameters via rotational invariance techniques. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(7):984–995, 1989.

[34] R. Roy, A. Paulraj, and T. Kailath. Estimation of signal parameters via rotational invariance techniques - esprit. In *MILCOM 1986 - IEEE Military Communications Conference: Communications-Computers: Teamed for the 90's*, volume 3, pages 41.6.1–41.6.5, 1986.

[35] H. K. Samudrala, S. Qadeer, S. Azeemuddin, and Z. Khan. Parallel and pipelined VLSI implementation of the new radix-2 DIT FFT algorithm. *Proceedings - 2018 IEEE 4th International Symposium on Smart Electronic Systems, iSES 2018*, pages 21–26, 2018.

[36] M. Sanchez, M. Garrido, M. Lopez-Vallejo, J. Grajal, and C. Lopez-Barrio. Digital channelised receivers on fpgas platforms. In *IEEE International Radar Conference*, pages 816–821, 2005.

[37] S. V. Schell and W. A. Gardner. 18 high-resolution direction finding. *Handbook of Statistics*, 10:755–817, 1 1993.

[38] R. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986.

[39] S. Sridharan, D. N. Prasad, R. C. George, and M. Brindha. Improved pulse repetition interval (pri) deinterleaving for electronic support measure (esm) receiver. 2015.

[40] P. Stoica and A. Nehorai. Music, maximum likelihood, and cramer-rao bound. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(5):720–741, 1989.

[41] B. D. Tseng. On computing the discrete Fourier transform. *Mathematics of Computation*, 32(141):175–199, 1978.

[42] D. G. Wilms. X. *European University Institute*, (2):2–5, 2012.

[43] Y. Xi, X. Wu, Y. Wu, and L. Deng. A fast and real-time pri transform algorithm for deinterleaving large pri jitter signals. In *2018 37th Chinese Control Conference (CCC)*, pages 4465–4469, 2018.

[44] Y. Y. Zhang, L. Zhang, Z. Q. Shang, Y. R. Su, Z. Wu, and F. B. Yan. A New Multichannel Parallel Real-time FFT Algorithm for a Solar Radio Observation System Based on FPGA. *Publications of the Astronomical Society of the Pacific*, 134(1033):0, 2022.

[45] Y. Zhao, H. Lv, J. Li, and L. Zhu. High performance and resource efficient FFT processor based on CORDIC algorithm. *Eurasip Journal on Advances in Signal Processing*, 2022(1), 2022.

[46] I. Ziskind and M. Wax. Maximum likelihood localization of multiple sources by alternating projection. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(10):1553–1560, 1988.