VHDL Implementation of Optimized RFI Detection System Proposed for SKA

Student Project
By
Aditya Mathuriya

Electronics and Communication Engineering
SardarVallabhai National Institute of Technology,
Surat

Under the Guidance of

Mr. Kaushal D. Buch

NCRA • TIFR

GIANT METREWAVE RADIO TELESCOPE
NATIONAL CENTER FOR RADIO ASTROPHYSICS
TATA INSTITUTE OF FUNDAMENTAL RESEARCH
Narayangaon, Tal-Junnar, Dist-Pune
May 2023- July 2023

Acknowledgments

I would like to express my heartfelt appreciation and gratitude to my mentor, Mr. Kaushal D. Buch, Engineer F, Digital Backend Group, GMRT, for their invaluable guidance, encouragement, and insightful suggestions throughout this project. Their unwavering support and teachings were instrumental in the completion of this report. I am deeply grateful for their constant assistance and direction.

I extend my thanks to Mr. B. Ajith Kumar, Group Coordinator, GMRT Backend Group, for his cooperation, support, and permission to work within the Digital Back-end Group.

I would like to express my sincere appreciation to the members of the Digital Backend group and the entire staff of GMRT. Every interaction with them has been a valuable learning experience, and their motivation has been instrumental in keeping me inspired and focused on my work.

~Aditya Mathuriya

Contents

List Of Figures

List of Tables

Abstract

Radio Astronomy faces a formidable challenge in the form of Radio Frequency Interference (RFI), which severely affects the already weak radio signals under observation. To address this pressing issue, there is an urgent need to develop effective techniques for RFI mitigation. The objective of this project is to design an optimized architecture that enables the integration of previously developed Median Absolute Deviation (MAD) and Median-of-MAD (MoM) filters into the Tile Processing Module (TPM) beamformer design with the available hardware resources on the FPGA. Additionally, this project aims to analyze the TPM raw voltage data taken at the site and determine an optimal window size for the effective detection of RFI.

To enhance the architecture's resource utilization aspect, careful consideration was taken in the utilization of inbuilt units such as DSP slices. The selection of these units was specifically tailored to ensure optimal utilization of the TPM board's resources. By maximizing the utilization of available board resources, we aimed to achieve the highest possible efficiency and performance within the architecture.

The implementations of these methods were conducted on the Xilinx Kintex Ultrascale (part no xcku9p-ffve900-1-i) FPGA device. To validate the design, the same algorithm was also implemented in MATLAB as the golden reference model. Functional verification of the design was performed using MATLAB implementation, ensuring the accuracy and reliability of the proposed approach.

Chapter 01: Introduction to GMRT and RFI

1.1 Overview

The GMRT, which stands for Giant Metrewave Radio Telescope, is a powerful radio telescope located near Pune, India. It is operated by the National Centre for Radio Astrophysics (NCRA) of the Tata Institute of Fundamental Research (TIFR). The GMRT is one of the world's largest and most sensitive radio telescopes operating at meter wavelengths.[1]

The GMRT consists of an array of 30 antennas, each antenna is 45 meters in diameter. These antennas operate at different frequencies ranging from 150 MHz to 1450 MHz.The GMRT utilizes a novel antenna construction technique called SMART, with wire mesh panels and rope trusses for a lightweight and low wind-loading design, enabling economical construction of the entire array.

Figure 1.1 GMRT Antennas *(courtesy NCRA Archives)*

The GMRT Backend team focuses on developing the correlator and has also undertaken projects to address interference reduction, temperature monitoring, and other enhancements for the GMRT project. Following the establishment of the upgraded GMRT known as uGMRT, the primary processing takes place concurrently at 400 MHz and 32 MHz. This advanced system, referred to as the GMRT Wideband Backend (GWB), is responsible for handling the extensive data processing requirements at the telescope.

1.3 RFI Filtering at GMRT

RFI, or Radio Frequency Interference, refers to unwanted electromagnetic signals or noise that can disrupt or degrade the quality of radio frequency signals used in communication systems, including wireless devices, radios, and radar systems. RFI can originate from various sources, such as power lines, electrical equipment, electronic devices, and even natural phenomena like lightning. The presence of RFI can lead to signal distortion, reduced dynamic range, and increased error rates in communication systems.

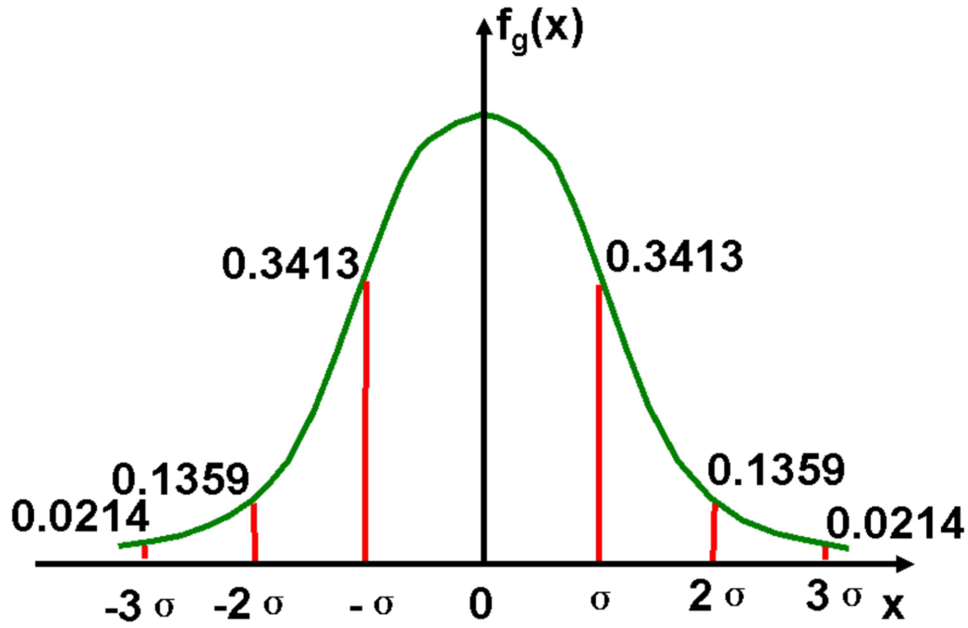

Figure 1.2 Distribution of Radio Signals Received From Space (*courtesy globalsino.com*)

The radio signals received from celestial objects follow a normal distribution, as depicted in the Figure 1.2. However, these signals are susceptible to interference from RFI, which is primarily a local phenomenon. The amplitude of RFI is typically stronger than that of the astronomical signal, resulting in the useful signal being concentrated primarily within the range of -3$\boldsymbol{\sigma}$ to +3$\boldsymbol{\sigma}$, at the center of the distribution curve. So the signals out of this range which have high amplitude are excised using different algorithms.

To mitigate RFI we at GMRT employ various algorithms like MAD (Median of Absolute Deviation) filtering and Median-of-MAD (MoM) filtering. These algorithms have been implemented on FPGA and are being used for real-time filtering of RFI [2].

1.3.1 Narrowband RFI

Narrowband radio frequency interference (RFI) typically manifests within a specific and limited portion of the electromagnetic spectrum. This particular type of signal tends to maintain its consistency over an extended period and generally does not result in any lasting harm to the system. Narrowband RFI often arises from the overlapping frequencies emitted by mobile towers and various communication devices. In Figure 1.3 presented below, the depicted narrow peaks illustrate the occurrence of narrowband RFI within the 325MHz observing band of GMRT.

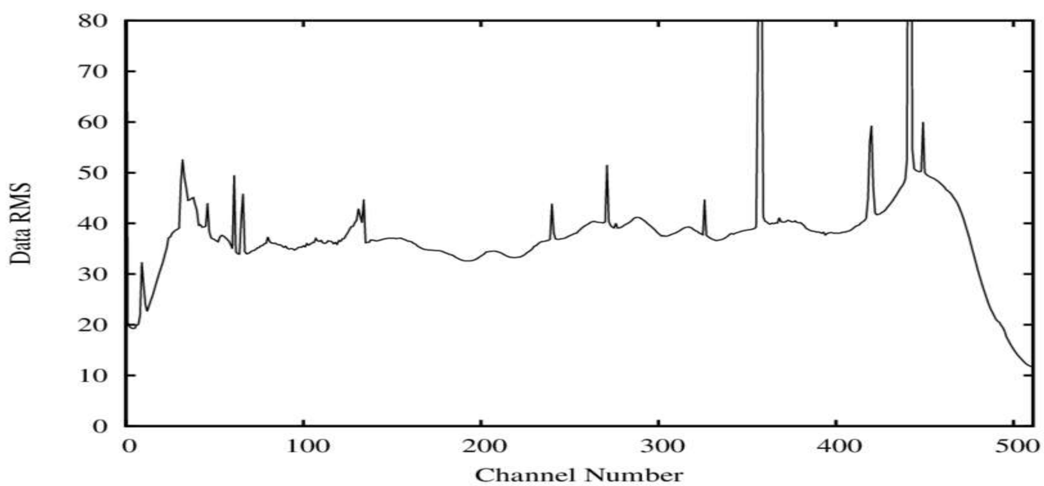

Figure 1.3 Narrowband RFI (Courtesy: Mayuresh Surnis)

1.3.2 Broadband RFI

Impulsive RFI refers to a type of interference characterized by sudden bursts of high-energy signals, which can overpower and obscure the desired astronomical signals. It occurs in a brief and sporadic manner, typically within a short time frame. Broadband RFI signals, on the other hand, possess a wide frequency range and have the potential to cause lasting harm to electronic receiver systems due to their intense nature. In Figure 1.4 displayed below, the depicted RFI exemplifies the presence of broadband interference observed at GMRT, primarily attributed to high voltage power lines, hence commonly referred to as powerline RFI.

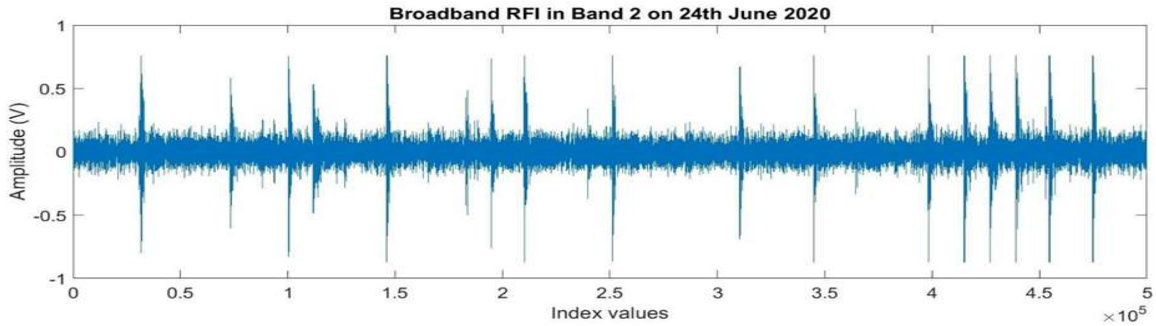

Figure 1.4 Broadband RFI

1.4 Square Kilometer Array (SKA)

The SKA (Square Kilometer Array) is an international project aimed at building the world's largest and most sensitive radio telescope. It will consist of thousands of radio antennas distributed across multiple continents, working together as a single integrated instrument. The SKA will be capable of observing the universe in unprecedented detail and sensitivity, enabling scientists to address fundamental questions about the nature of the universe, the formation of galaxies, the evolution of cosmic magnetism, and the search for extraterrestrial life. The project is a collaboration of over 20 countries and is expected to revolutionize our understanding of the universe through its cutting-edge technology and vast observation capabilities.

Indian astronomers with wide-ranging experience in low-frequency radio astronomy in a variety of astronomical phenomena and targets would be particularly well-placed to pursue time-critical transient objects with SKA and observatories at other bands[3].

1.5 Tile Processing Module

The signal processing in LFAA would be carried out using FPGA-based Tile Processing Module (TPM) developed by the Italian team for implementing the beamforming system. The Tile Processing Module is an essential component that performs multiple tasks, including data acquisition, channelization, and beamforming for the SKA Low-Frequency Aperture Array instrument. It is designed to handle these operations for a set of 16 antennas, enabling the efficient processing of signals received by the instrument. The data that is received by TPM is sampled at a frequency of 800 MHz [4] A similar setup has been made at GMRT for testing purposes, and we have developed a technique similar to that used for mitigating RFI in real-time in the Upgraded GMRT (uGMRT)backend.

Chapter 02: MAD/MOM Based RFI MITIGATION

The Median Absolute Deviation (MAD) is a way to measure how spread out or diverse a set of numbers is. To calculate MAD, we first find the middle value in the set, which is called the median. Then, we look at each number in the set and find the difference between that number and the median, always using positive values. We find the median of these differences to get the MAD which is helpful when we have numbers that are far away from the others, or when the numbers don't follow a typical pattern. For MoM computation, three Median computations are required. Further, both techniques have been explained in brief in sections 2.1 and 2.2

2.1 MAD

The process of calculating the MedianAbsolute Deviation (MAD) is depicted in Figure 2.1, which offers a comprehensive overview of the steps involved.

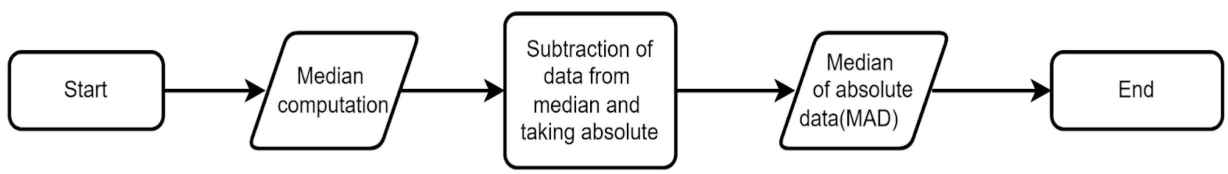

Figure 2.1 Flowchart MAD Computation

The first Median value is subtracted from the input data value. The absolute value of this difference is taken. If the difference is negative, then the two's complement is taken for conversing to absolute value. Then the second median of these absolute values is computed using the same algorithm, and it is called Median Absolute Deviation (MAD). For a dataset **X** having window size W, MAD can be computed as

$$\mathbf{MAD = median(|\ X_i - median(X)\ |)}$$
$$\mathbf{i = 1, 2, 3, \ldots, W}$$

2.2 MoM

For MoM computation, three Median computations are required. For real-time performance, it is challenging to buffer these data values. So, the calculated current MoM value is applied to the next cycle. Also, to optimize median computation, the third median computation is multiplexed with the second median computation see Figure 2.3. It is useful especially when the dataset is large or when traditional sorting methods may be inefficient. The MoM technique involves dividing the dataset into smaller subgroups and finding the MAD within each subgroup. These subgroup MADs are then treated as a new set of data points and their median is computed.

$$\mathbf{MoM = median\ (MAD_1,\ MAD_2,\ MAD_3,.....MAD_n)}$$

where n = MoM window size

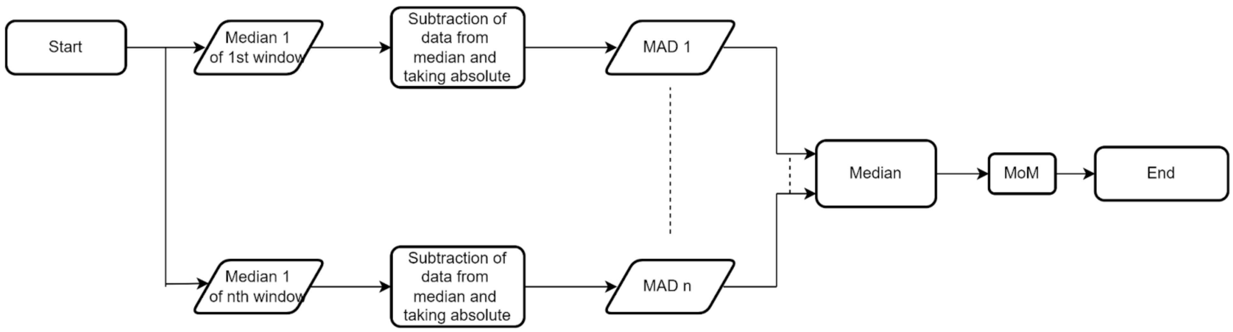

Figure 2.2 Flowchart of MoM Computation

2.3 VHDL Implementation of Filters

The present report focuses on the integration of previously developed Median Absolute Deviation (MAD) and Median of MAD (MoM) based RFI (Radio Frequency Interference) filters using VHDL (Very High-Speed Integrated Circuit Hardware Description Language). In the previous project, these filters were implemented individually for RFI mitigation. Both MAD and MoM filters were implemented for various window sizes i.e. 4k, 8k, and 16k.

The method utilized in this project for finding the median of the window data is the Histogram method. Initially, the median is determined for the window data, and this value is then subtracted from each element within the window(see Figure 2.2)[5]. The resulting elements are converted to unsigned integers. Subsequently, the median of the updated window is calculated, which serves as the MAD value for the data window. The calculated median and MAD values are utilized to derive the threshold value for the signal(see Figure 2.3). To determine the window's variance, the MAD value is multiplied by a constant factor of 1.4826. The positive and negative thresholds are established by adding and subtracting the median value, respectively. Once the threshold values are obtained for one channel, the data from all channels are compared and flagged. User-defined criteria determine the replacement of the flagged data.[5]

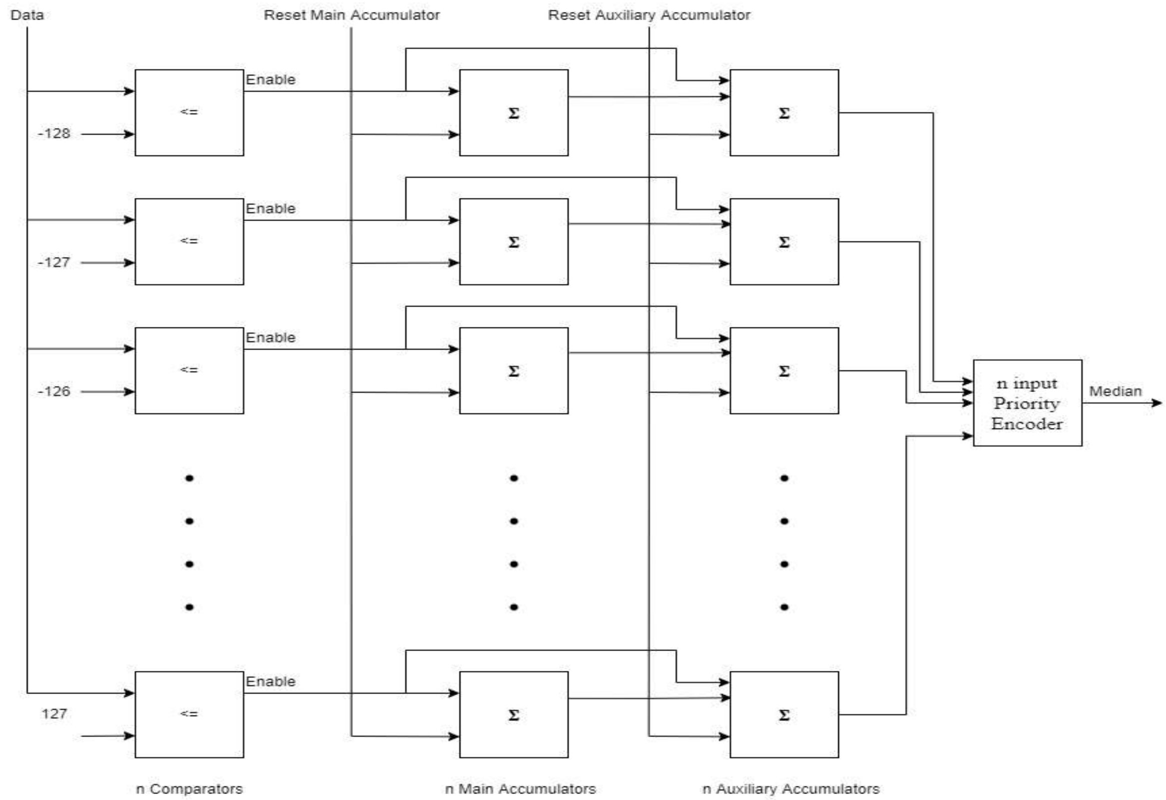

Figure 2.2 Histogram-based Median Calculation

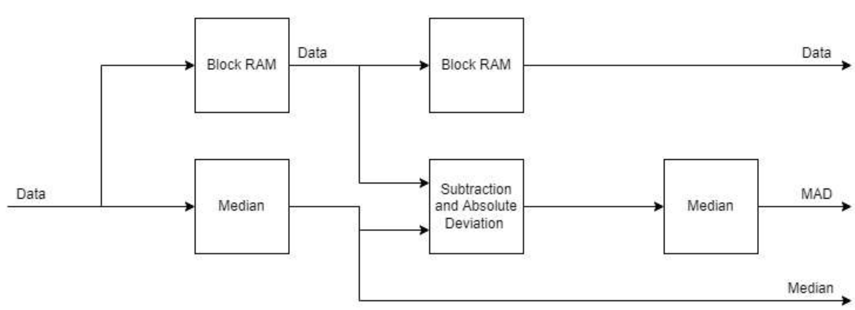

Figure 2.3 *Median Absolute Deviation (MAD) Computation*

Another approach employed in this project is the MoM method, which utilizes the MoM value to compute the threshold values.

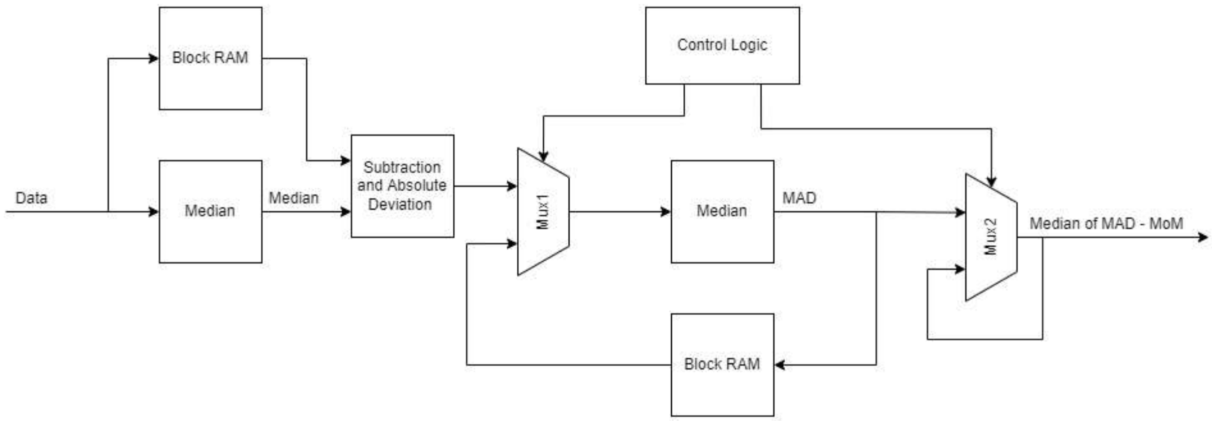

Figure 2.4 Block Diagram of MoM Computation

The design implementation utilizes Xilinx Vivado version 2019.1 and is targeted for deployment on the *Xilinx Kintex Ultrascale (xcku9p-ffve900-1-i) FPGA* board. MATLAB software is employed for functional verification of the algorithm.

Verification of these filters was done by comparing the outputs with the MATLAB output (golden reference).

2.4 Application of RFI Filters in TPM

The Tile Processing Module (TPM) comprises 16 antennas on a single tile. In our project, we aim to use RFI filters to identify windows that exhibit a high count of RFI. If the RFI count exceeds a specified reference value, the RFI flag is raised, indicating that the preceding data contains significant RFI and an appropriate action should be taken for subsequent processing. This approach allows us to effectively detect and flag data with substantial RFI content, enabling us to mitigate the impact of interference on further data analysis.

The TPM module receives inputs from 16 antennas, a total of 128 bits i.e. 8-bit for each antenna, as shown in Figure 2.4. The module processes this input and generates a 1-bit output for each antenna, resulting in a total of 16 bits of output.

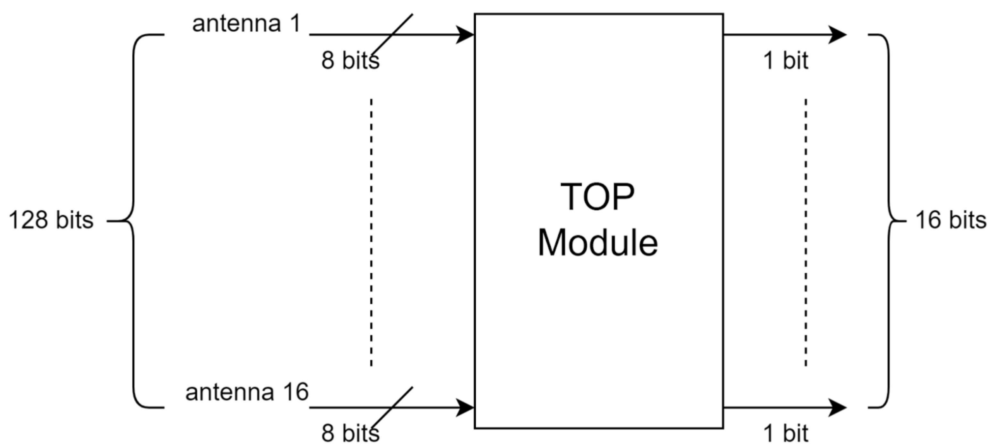

Figure 2.4 Input-Output View of Top Module

Based on the computed MAD or MoM values, the upper and lower thresholds are determined. If the incoming data falls outside these threshold boundaries, it is identified as RFI, resulting in an increment of the RFI counter. At the end of the window cycle, if the counter exceeds the reference value, the RFI flag is triggered, indicating the presence of highly corrupted data.

Due to limited FPGA resources, instead of having individual RFI filters for all 16 antennas simultaneously, a round-robin approach was implemented. In this approach, the thresholds are calculated sequentially, starting from the first antenna and moving to the next antenna cyclically. The updated thresholds for the first antenna are obtained once all 16 antennas have had their thresholds calculated. To facilitate this process, multiplexers are used to select the appropriate antenna for threshold calculation, and the data flow is directed to the threshold calculation block. The calculated thresholds are then passed through a demultiplexer and stored in latches, ensuring they are retained until the corresponding antenna's turn comes up again. Additionally, there is another demultiplexer that generates enable signals for these latches, ensuring proper synchronization of the threshold values, see Figure 2.5.

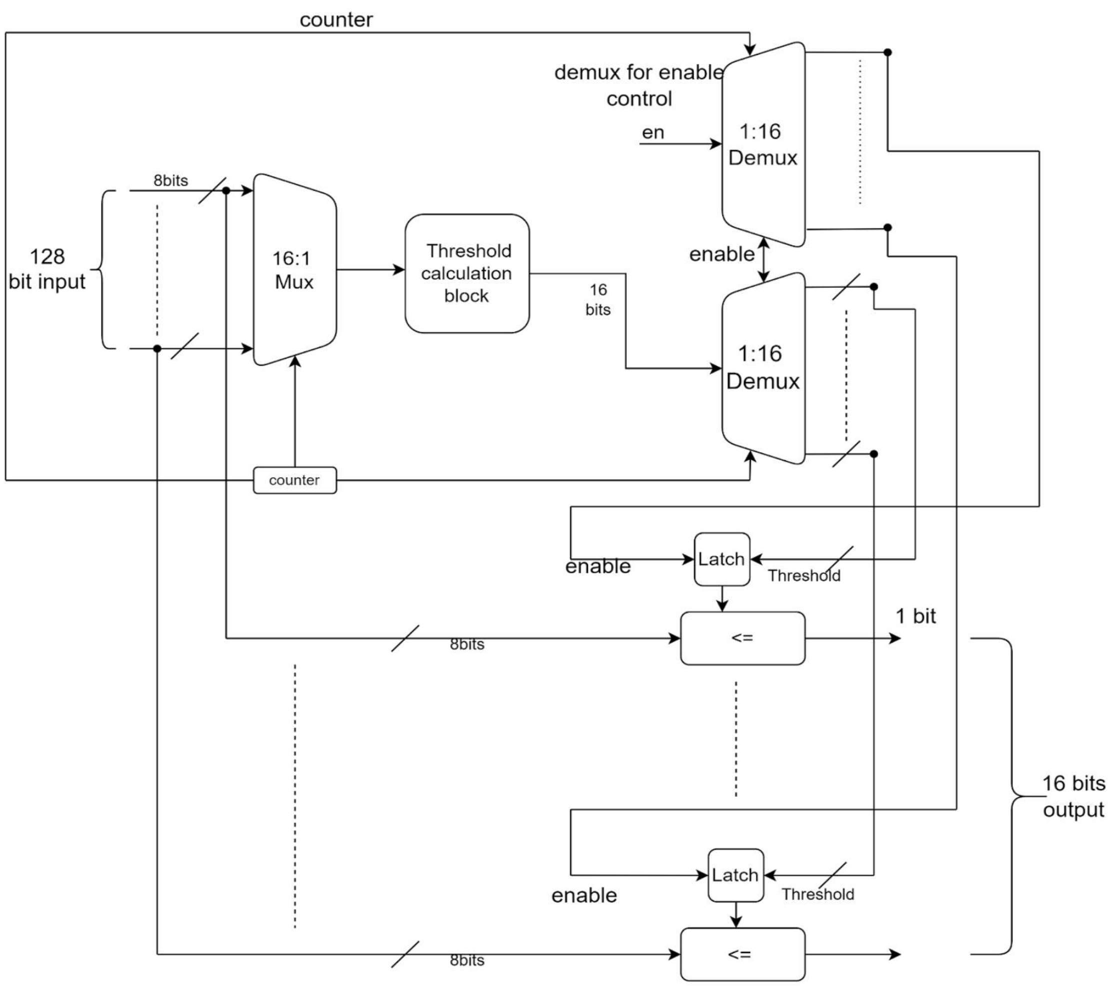

Figure 2.5 Block Diagram of MAD/MoM Multiplexed Architecture

We have also implemented an alternative method called the Single MAD/MoM Design, depicted in Figure 2.6, where we eliminate the need for rotating the threshold and instead calculate it directly based on a single antenna input.

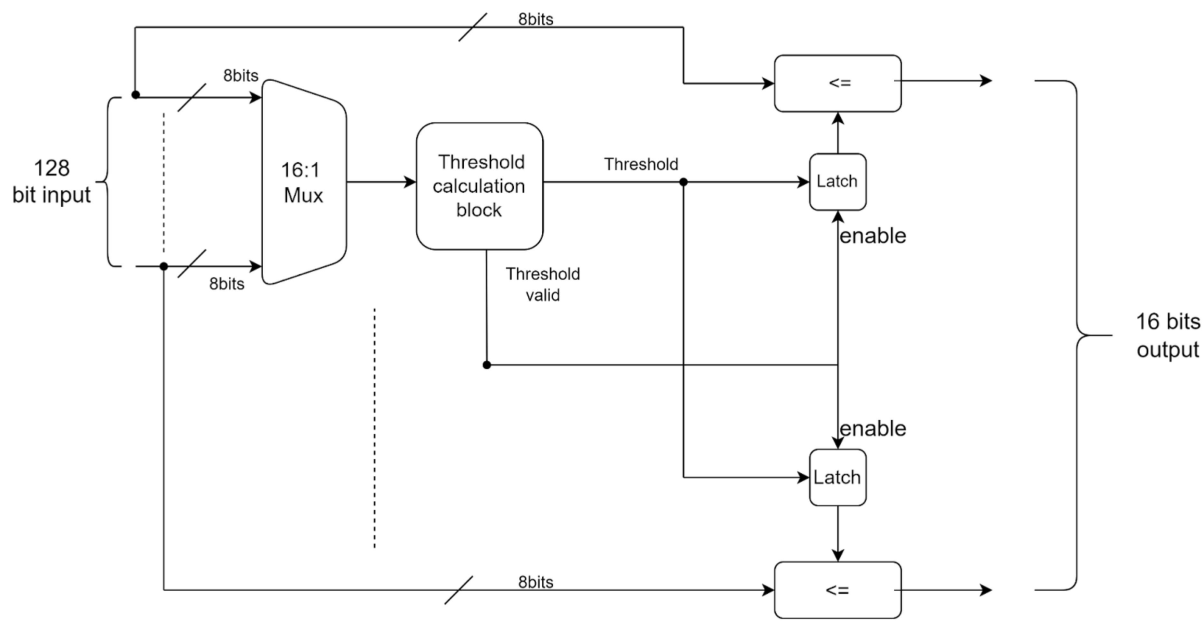

Figure 2.6 Single MAD / MOM Multiple Detection

2.5 Need for Optimization

During the design process of the proposed architecture, careful consideration is given to the utilization of resources not only within the architecture itself but also in the larger context of the TPM hardware and overall design. It is crucial to optimize resource utilization to ensure efficient operation and seamless integration of the architecture into the larger design framework. This approach ensures that the proposed architecture aligns effectively with the TPM hardware and contributes to the overall success of the project.

Resources	TPM (%)	MAD without filter (%)	Multiplexed MAD scheme (V1) (%)
LUTS	59.9	2.28	2.9
SLICES	88.4	10.2	11
BRAM	42.2	0.9	0.44
DSPs	66.7	0	0.08

Table 2.1 Resource Utilization Table (16k window)

The utilization of SLICES in the FPGA has reached a significant level of 99.4%. This indicates that the current resource utilization is approaching the maximum threshold and careful consideration should be given to resource management and optimization to ensure the proper functioning of the FPGA.

In this project, two optimized versions have been proposed to address the high resource utilization issue. These versions involve reducing the utilization of SLICES by leveraging the utilization of DSP slices. By strategically utilizing DSP slices, we aim to optimize resource allocation and achieve a more efficient design that minimizes overall resource utilization while maintaining the desired functionality.

Chapter 03: Analysis of TPM Data

3.1 Data Format

We received the TPM data for 5 days from March 2022. The received data is from the TPM site with 50-350 MHz RF and sampled at 1.25 ns (800 MHz) sampling period. The files were in zipped format. Following are the details of the files.

Format of the file name: **YYYY-MM-DD_Terra15_Tile-16.tar**
After extracting this file a folder with the same name is created, each folder varies in size from 6.5 Gbs to 8.5 GB approximately, depending on the number of files it contains.

Each folder contains around 2.8K to 5.5K files, all these files are in **.hdf5** format, and are further extracted using a Python script, which converts them into **.out** format.
Format of the file name: **raw_burst_15_YYYYMMDD_XXXX_0.hdf5**

Among the four Python scripts provided with the data, one particular script holds significant importance. This script generates a crucial output file that encompasses data for 16 antenna dual poles. The resulting file comprises 32 columns, each representing data for 100 or more bursts. It is noteworthy that each burst consists of 32768 voltage samples. Leveraging this vital file, we have developed a Python tool specifically designed for the analysis of TPM data.

3.2 Python Tool

We have developed a Python tool for analyzing long TPM data that allows manual analysis and identification of significant bursts of RFI. The tool is designed as a graphical user interface (GUI) using Tkinter, making it user-friendly and intuitive.

Using this tool, one can easily extract data from HDF5 files and generate output files in the .out format. The user has the flexibility to choose the desired name for the output file. The output file contains 32 columns, representing the data of 16 antennas with dual polarization. The tool also provides the ability to plot 16 figures simultaneously, allowing for a clear understanding of the data behavior across antennas on the same tile.

One of the key features of the tool is the simplified selection of antennas of interest with which the user can conveniently specify their preferences using comma-separated values or a range. The tool validates the selected antennas and generates a plot that showcases the behavior of the chosen antennas on a single tile. The 16 subplots in the figure enable users to visualize and analyze the antenna data effectively.

By streamlining the process of data extraction and visualization, the tool enhances the manual analysis of TPM data. Users can identify significant RFI bursts and gain insights into the behavior of different antennas. The tool is quite useful in quick time domain analysis of the TPM Data.

The flow is shown in section 3.3.

3.3 Flowchart of Tool

A brief flow of the tool is shown in Figure 3.1, the user has to follow these steps to extract and plot the TPM data.

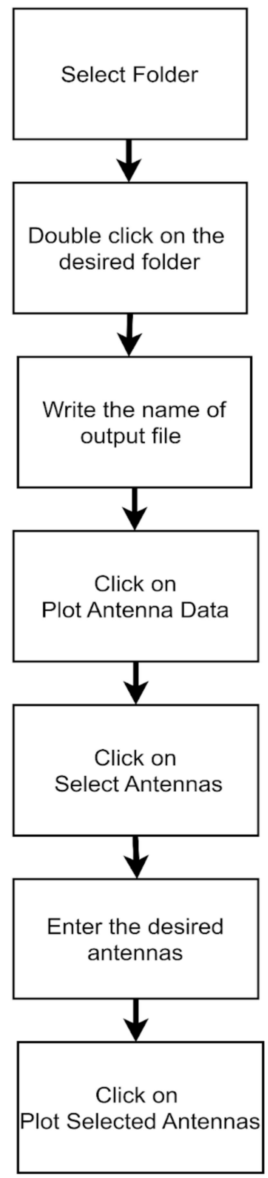

Figure 3.1 Flowchart of Tool

3.4 Time Domain Analysis

Time domain analysis is done on TPM data to find the similarity of the data between the antennas on the same tile. The data plotted here is from 03/03/2022.

Here is a sample plot:

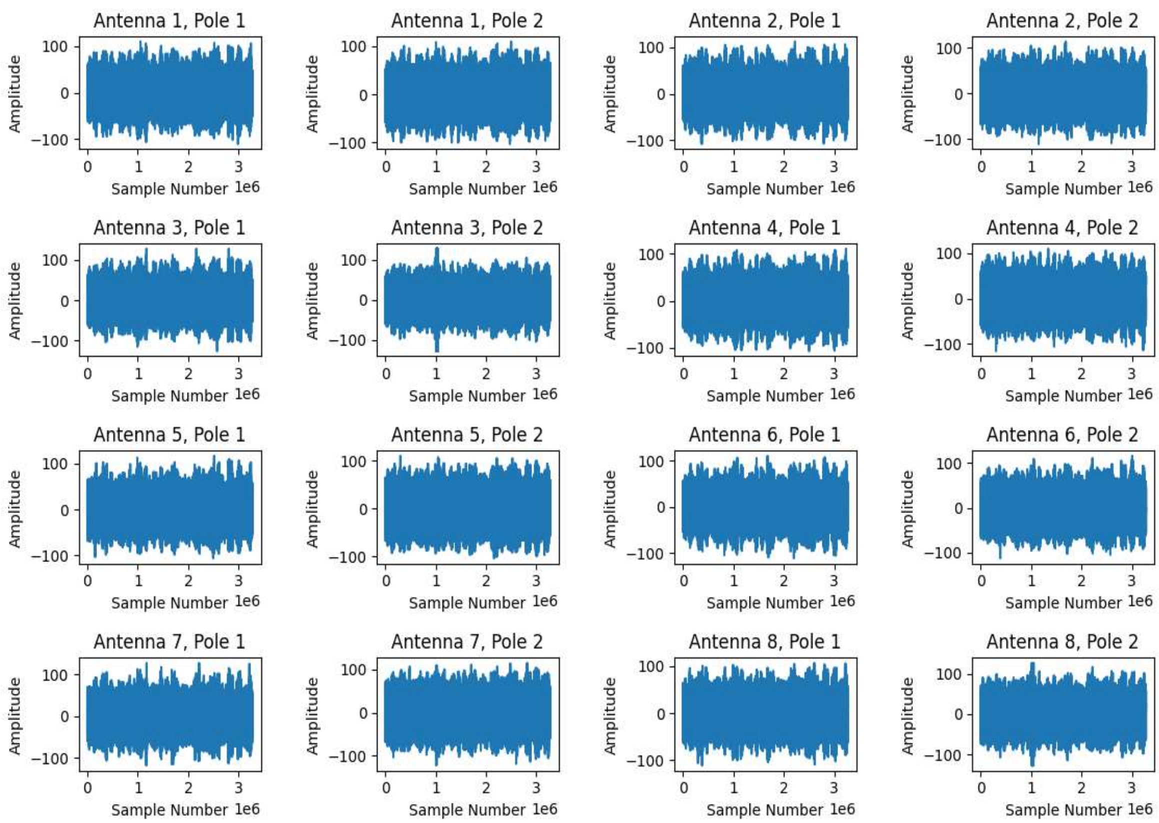

Figure 3.2 Time domain analysis

As it is clear from Figure 3.2 that the data pattern is the same for the antennas on the same tile. So we can also propose a single threshold scheme for all the antennas on one tile. As the separation between them is quite less.

Analyzing the data with MATLAB to determine an appropriate window size for the detection based on the methods of MoM and MAD.

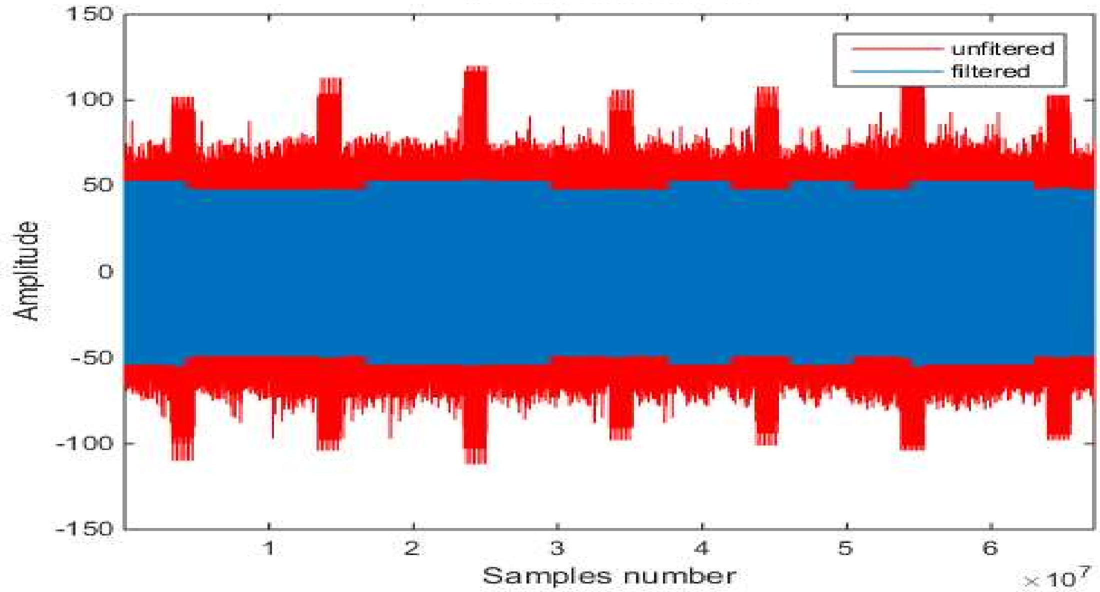

Figure 3.3 MoM with 2k window size

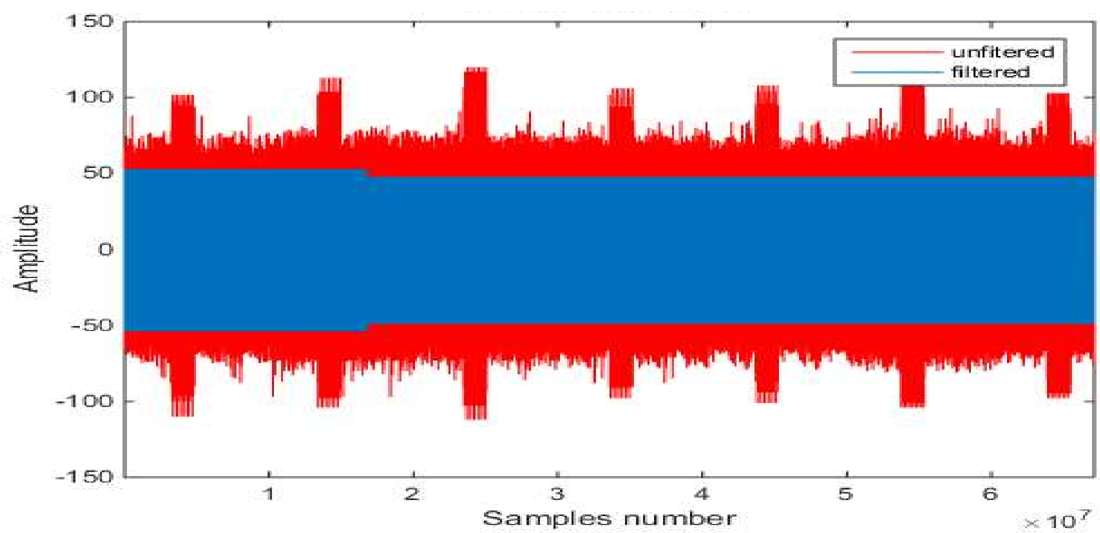

Figure 3.4 MoM with 4k window size

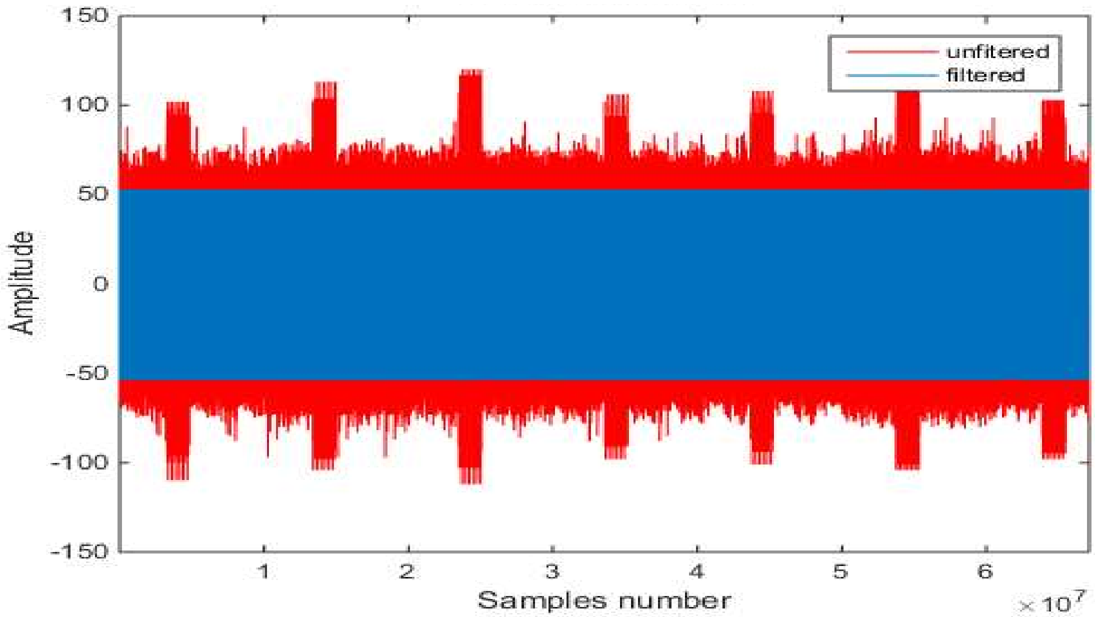

Figure 3.5 MoM with 8k window size

As it is quite evident from the Figures 3.3, 3.4, 3.5, 4k and 8k are the best window sizes for MOM filtering, 4k and 8k are quite effective in filtering the data. But considering the constraints for resources we concluded that 4k is best for our use.

3.5 Frequency Domain Analysis

Frequency domain analysis is a valuable technique that involves analyzing signals by decomposing them into their constituent frequency components. It allows for the identification of specific frequencies or frequency ranges of interest and provides insights into signal characteristics.

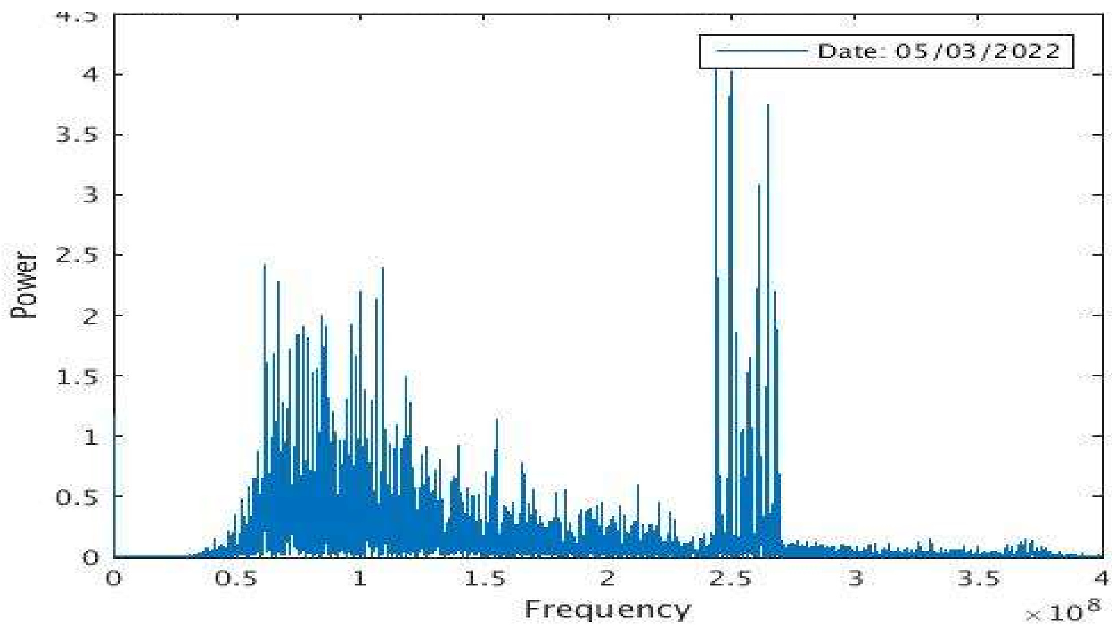

Figure 3.6 Cross-Correlation Spectrum

Auto-correlation and Cross-correlation are two important statistical methods used in Frequency domain analysis. On one hand, auto-correlation helps us find whether a signal distorted by noise is periodic or not, on the other hand, cross-correlation helps us find the copy of one signal into another noisy signal.

We did the cross-correlation of the filtered and unfiltered data, and as we can clearly see from the graph the output is highly correlated, which suggests that the signal information is not lost during filtering of data. The spike that we see in Figure 3.5 is RFI.

In Figure 3.6, we presented a power spectrum plot(auto-correlation) comparing the filtered and unfiltered data. The graph clearly illustrates that the filtered data exhibits fewer/lower RFI compared to the unfiltered data. This observation signifies the effectiveness of the MOM (Median of MAD) filtering technique in successfully removing RFI. The reduction in power spikes indicates that the filtering process significantly mitigates the impact of RFI on the signal, leading to a cleaner and more reliable data representation.

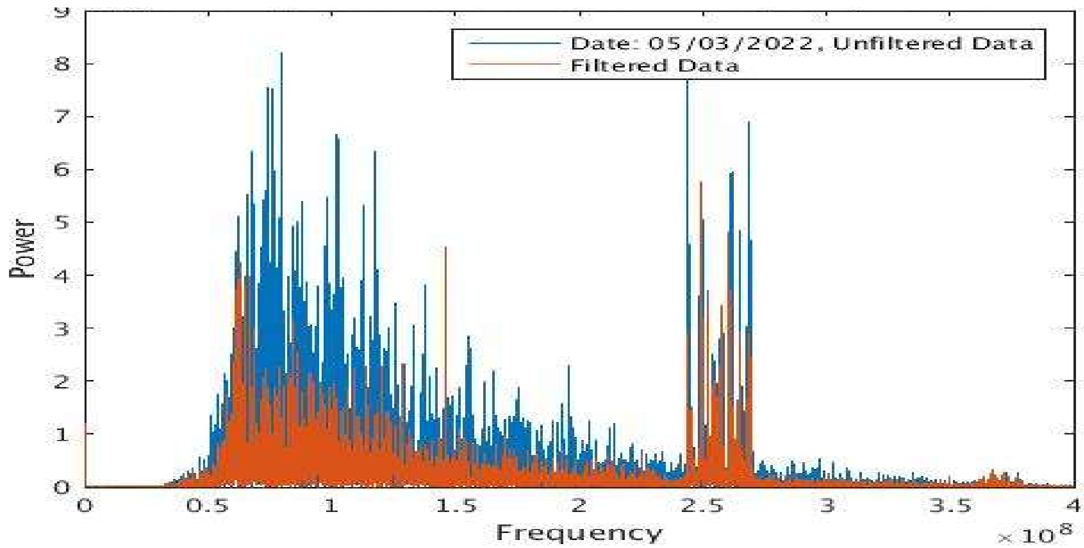

Figure 3.7 Power Spectrum Overlay for Unfiltered and Filtered Data

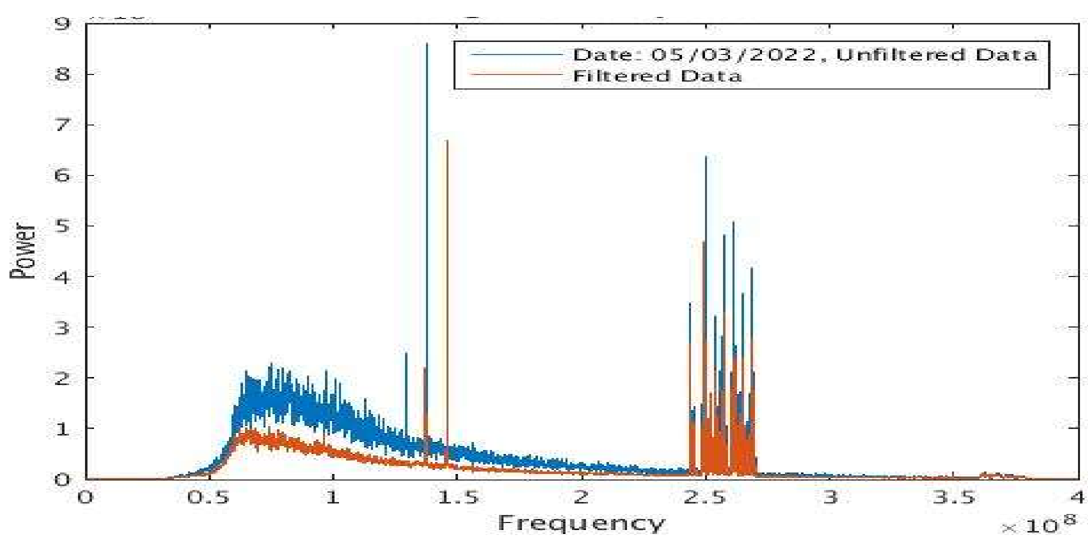

Figure 3.8 Average Power Spectrum for unfiltered and filtered data

To remove the high-frequency noise we did the power averaging in Figure 3.7 which acts as a low pass Filter. We plotted Average Power Spectrum for better visualization in Figure 3.8.

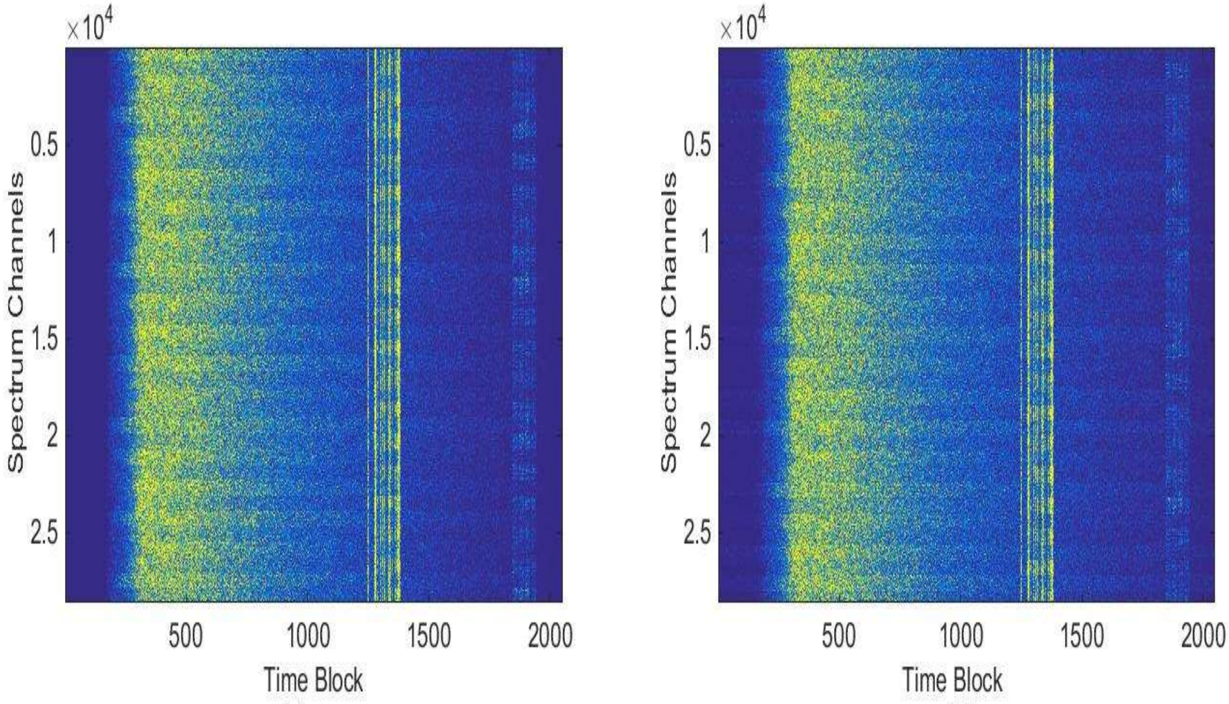

Figure 3.9 Average Power Spectrum

Chapter 04: Optimization

As mentioned in Chapter 02, due to limited resources in the TPM module, an optimized approach was necessary. In this regard, we found that the DSP slices were underutilized, leaving room for more efficient utilization to reduce the over-utilization of CLB slices.

To achieve this, we strategically relocated the accumulators of our design to the inbuilt DSP accumulators, thereby making better use of the available resources and improving overall efficiency.

4.1 Optimized MAD Multiplexed Design

Resources	**TPM (%)**	**MAD without filter (%)**	**Multiplexed MAD scheme (V1) (%)**	**Multiplexed MAD scheme (Accum in dsp)(V2)**
LUTS	59.9	2.28	2.9	2.9
SLICES	88.4	10.2	11	10.79
BRAM	42.2	0.9	0.44	0.44
DSPs	66.7	0	0.08	20.40

Table 4.1 Resource Utilization MAD Multiplexed

The utilization of DSP slices significantly increased from a mere 0.08% to a substantial 20.40%, indicating a more efficient use of these resources. On the other hand, the utilization of Slices decreased slightly from 11% to 10.79%, further highlighting the successful optimization of the design.

Signal	*Description*
clk	Input clock
rst	Active high, synchronous reset
hold	The control signal for enabling counter output
rst_ram2_counter	Active high, Synchronous reset for address generating counter of Block RAM2 which stores the MAD values
rst_comp_median	Active high, Synchronous reset for comparator block of median
rst_counter_median	Active high, Synchronous reset for a counter block of the median which generates a reset signal for the main accumulator and auxiliary accumulator
rst_comp_mad	Active high, Synchronous reset for comparator block of MAD
rst_counter_mad	Active high, Synchronous reset for a counter block of MAD which generates a reset signal for the main accumulator and auxiliary accumulator
rst_filter	Active high, Synchronous reset for filter block
rst_count	Active high synchronous reset for counter that controls mux and demux
count	The control signal for the counter that controls mux and demux
8-bit inputs	Data inputs for 16 antennas.
rfi_flag_1 to rif_flag_16	Flag output of the detection block.

Table 4.2 VHDL interface of Multiplexed MAD design

4.1.1 Synthesis and Implementation

In this section, Synthesis and Implementation details are mentioned. This section contains the FPGA Implementation view of the MAD-Multiplexed design. Moreover, the Timing and Area-Utilizations details of Implemented designs are discussed in this section.

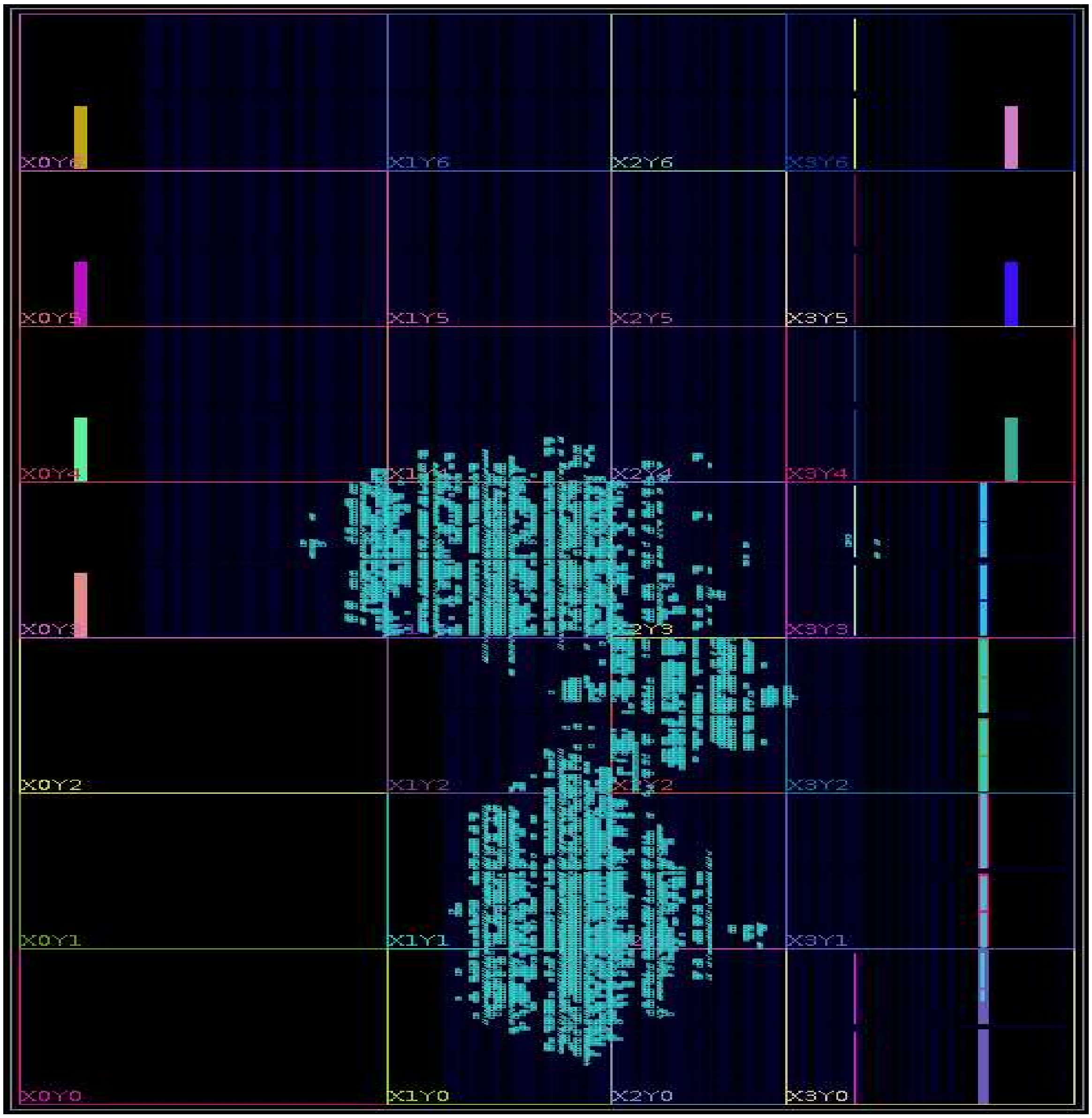

Figure 4.2 Implementation View of MAD Multiplexed Design

4.1.2 Timing Details

As shown in Figure 4.2, the Filter design works on 312.5 MHz frequency without any STA violation.

```
------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------

Clock  Waveform(ns)         Period(ns)      Frequency(MHz)
-----  ------------         ----------      --------------
clk    {0.000 1.600}        3.200           312.500

--------------------------------------------------------------------------------------
From Clock:  clk
  To Clock:  clk

Setup :            0  Failing Endpoints,  Worst Slack       0.071ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack       0.020ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack       1.057ns,  Total Violation        0.000ns
--------------------------------------------------------------------------------------
```

Figure 4.2 Timing Report MAD Multiplexed

4.1.3 Utilization details

Figure 4.3 gives an overall idea of resource utilization. There is no register as a latch.

1. CLB Logic

Site Type	Used	Fixed	Available	Util%
CLB LUTs	7955	0	274080	2.90
LUT as Logic	7955	0	274080	2.90
LUT as Memory	0	0	144000	0.00
CLB Registers	11587	0	548160	2.11
Register as Flip Flop	11587	0	548160	2.11
Register as Latch	0	0	548160	0.00
CARRY8	2713	0	34260	7.92
F7 Muxes	21	0	137040	0.02
F8 Muxes	8	0	68520	0.01
F9 Muxes	0	0	34260	0.00

Figure 4.3 CLB Utilization MAD Multiplexed

4. ARITHMETIC

Site Type	Used	Fixed	Available	Util%
DSPs	514	0	2520	20.40
DSP48E2 only	514			

2. CLB Logic Distribution

Site Type	Used	Fixed	Available	Util%
CLB	3695	0	34260	10.79
CLBL	1537	0		
CLBM	2158	0		
LUT as Logic	7958	0	274080	2.90
using O5 output only	19			
using O6 output only	3341			
using O5 and O6	4598			
LUT as Memory	0	0	144000	0.00
LUT as Distributed RAM	0	0		
LUT as Shift Register	0	0		
CLB Registers	11572	0	548160	2.11
Register driven from within the CLB	9618			
Register driven from outside the CLB	1954			
LUT in front of the register is unused	1743			
LUT in front of the register is used	211			
Unique Control Sets	51		68520	0.07

9. Primitives

Ref Name	Used	Functional Category
FDRE	11587	Register
LUT2	8369	CLB
CARRY8	2713	CLB
LUT3	1638	CLB
LUT6	922	CLB
LUT5	725	CLB
LUT1	534	CLB
DSP48E2	514	Arithmetic
LUT4	368	CLB
INBUF	147	I/O
IBUFCTRL	147	Others
MUXF7	21	CLB
OBUF	16	I/O
MUXF8	8	CLB
RAMB36E2	4	Block Ram
BUFGCE	3	Clock

Figure 4.4 Utilization Report of MAD Multiplexed

4.2 Optimized MOM Multiplexed Design

The architecture for MoM multiplexed is the same as MAD multiplexed, but we have made it using the first median 0. The only change is now the threshold is calculated based on MoM.

Resources	**MAD without filter** **(%)**	**Multiplexed MAD scheme (V1)** **(%)**	**Multiplexed MAD scheme (accum in dsp)(V3)**	**MAD multiplexed Scheme (first median =0) (V4)**	**Multiplexed MoM (first median=0) (V5)**
LUTS	2.28	2.9	2.9	1.75	1.68
SLICES	10.2	11	10.79	6.20	6.02
BRAM	0.9	0.44	0.44	0.44	0.22
DSPs	0	0.08	20.40	10.24	10.24

Table 4.3 Resource Utilization Multiplexed MoM Design

Signal	*Description*
clk	Input clock
rst	Active high, synchronous reset
hold	The control signal for enabling counter output
rst_ram2_counter	Active high, Synchronous reset for address generating counter of Block RAM2 which stores the MAD values
rst_comp_mad	Active high, Synchronous reset for comparator block of MAD
rst_counter_mad	Active high, Synchronous reset for a counter block of MAD which generates a reset signal for the main accumulator and auxiliary accumulator
rst_filter	Active high, Synchronous reset for filter block
enable demux	Active high, Synchronous enable for both demultiplexers.
count	Control signal for the counter, that controls multiplexer
sel_filter(3 downto 0)	User has given input for selecting the input, which will select the data on which threshold will be calculated
8-bit inputs	Data inputs for 16 antennas.
rfi_flag_1 to rif_flag_16	Flag output of the detection block.

Table 4.4 VHDL interface of MOM multiplexed design

4.2.1 Synthesis and Implementation

In this section, Synthesis and Implementation details are mentioned. This section contains the FPGA Implementation view of the MoM-Multiplexed design. Moreover, the Timing and Area-Utilizations details of Implemented designs are discussed in this section.

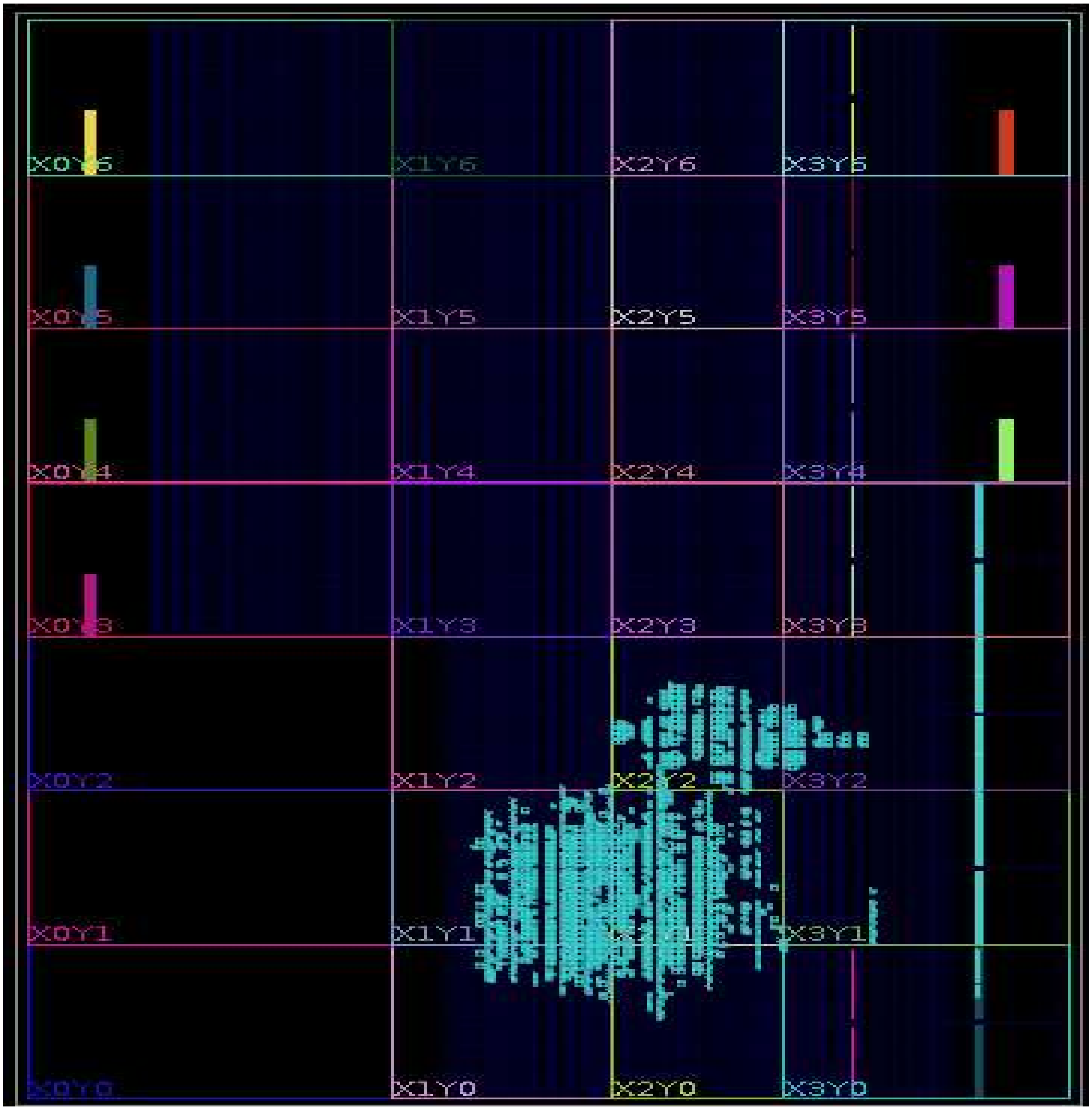

Figure 4.5 Implementation view of MoM Multiplexed Design

4.2.2 Timing Details

As shown in Figure 4.2, the Filter design works on 312.5 MHz frequency without any STA violation.

```
----------------------------------------------------------------------
| Clock Summary
| -------------
----------------------------------------------------------------------

Clock  Waveform(ns)         Period(ns)      Frequency(MHz)
-----  ------------         ----------      --------------
clk    {0.000 1.600}        3.200           312.500

------------------------------------------------------------------------------------------------
From Clock:  clk
  To Clock:  clk

Setup :            0  Failing Endpoints,  Worst Slack        0.012ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.010ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack        1.057ns,  Total Violation        0.000ns
------------------------------------------------------------------------------------------------
```

Figure 4.6 Timing Report MoM Multiplexed

4.2.3 Utilization Details

1. CLB Logic

Site Type	Used	Fixed	Available	Util%
CLB LUTs	4580	0	274080	1.67
LUT as Logic	4580	0	274080	1.67
LUT as Memory	0	0	144000	0.00
CLB Registers	6930	0	548160	1.26
Register as Flip Flop	6930	0	548160	1.26
Register as Latch	0	0	548160	0.00
CARRY8	1433	0	34260	4.18
F7 Muxes	21	0	137040	0.02
F8 Muxes	8	0	68520	0.01
F9 Muxes	0	0	34260	0.00

Figure 4.7 CLB Logic MoM Multiplexed

4. ARITHMETIC

Site Type	Used	Fixed	Available	Util%
DSPs	258	0	2520	10.24
DSP48E2 only	258			

2. CLB Logic Distribution

Site Type	Used	Fixed	Available	Util%
CLB	2062	0	34260	6.02
CLBL	940	0		
CLBM	1122	0		
LUT as Logic	4580	0	274080	1.67
using O5 output only	21			
using O6 output only	2042			
using O5 and O6	2517			
LUT as Memory	0	0	144000	0.00
LUT as Distributed RAM	0	0		
LUT as Shift Register	0	0		
CLB Registers	6930	0	548160	1.26
Register driven from within the CLB	5031			
Register driven from outside the CLB	1899			
LUT in front of the register is unused	1651			
LUT in front of the register is used	248			
Unique Control Sets	49		68520	0.07

9. Primitives

Ref Name	Used	Functional Category
FDRE	6930	Register
LUT2	4208	CLB
CARRY8	1433	CLB
LUT3	1088	CLB
LUT5	653	CLB
LUT6	509	CLB
LUT4	327	CLB
LUT1	312	CLB
DSP48E2	258	Arithmetic
INBUF	146	I/O
IBUFCTRL	146	Others
MUXF7	21	CLB
OBUF	16	I/O
MUXF8	8	CLB
RAMB36E2	2	Block Ram
BUFGCE	2	Clock

Figure 4.8 Utilization Report of MoM Multiplexed

4.3 Single MAD Design

As shown in Table 4.1, it is clear that the resources have not undergone significant reduction. Therefore, we decided to explore an alternative approach. Considering that the antennas on one tile are in close proximity, which was also confirmed by the time domain analysis in Chapter 03, we observed that the data received from all antennas exhibited considerable similarity. Based on this observation, we put forth the proposal to use a single MAD computation for all the antennas on the same tile.

Single MAD means that all the comparators will be provided with the same threshold instead of a Round Robin method. The thresholds will be calculated on the data of one antenna selected by the user.

Here is the proposed architecture.

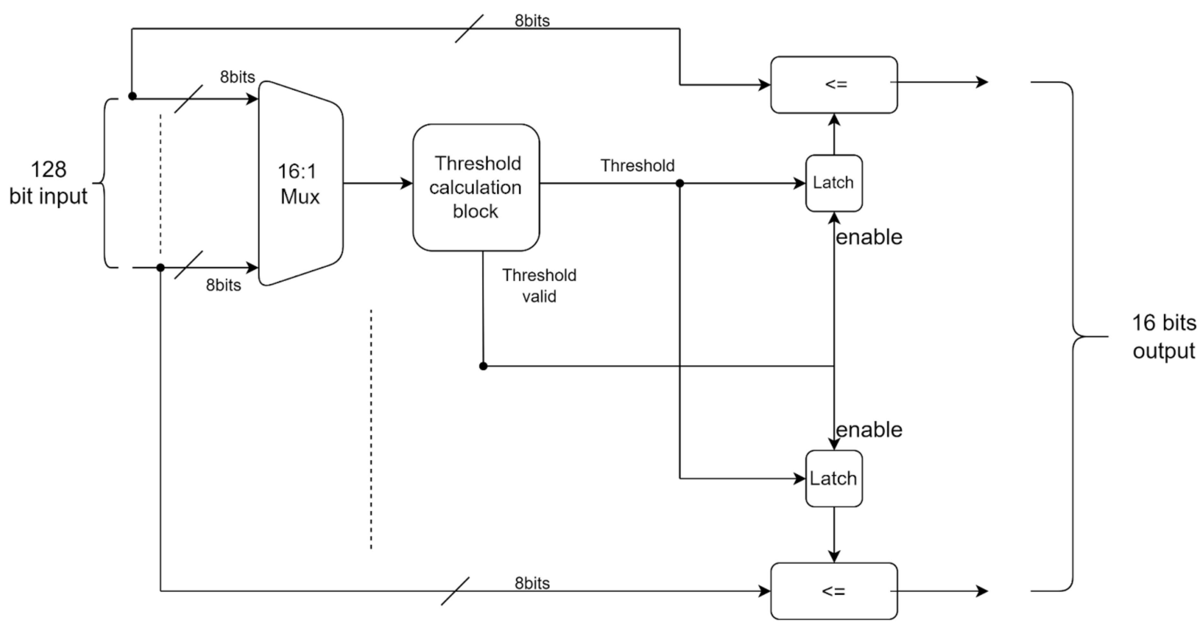

Figure 4.9 Single MAD / MOM Multiple Detection

The architecture depicted in Figure 4.9 is shared between both MAD and MoM designs, with the only distinction arising in the threshold calculation block.

Resources	TPM (%)	MAD without filter (%)	Multiplexed MAD scheme (V1) (%)	Multiplexed MAD scheme (accum in dsp)(V2) (%)	Single MAD multiple detection scheme (V3)(%)
LUTS	59.9	2.28	2.9	2.9	2.88
SLICES	88.4	10.2	11	10.79	10.56
BRAM	42.2	0.9	0.44	0.44	0.44
DSPs	66.7	0	0.08	20.40	20.40

Table 4.5 Resource Utilization for Single MAD design

We can see in Table 4.5 that the utilization of DSP slices significantly increased from a mere 0.08% to a substantial 20.40%, indicating a more efficient use of these resources. On the other hand, the utilization of Slices decreased slightly from 10.79% to 10.56%, further highlighting the successful optimization of the design.

But all this reduction is not quite enough, we have proposed a final more optimized version in the subsequent chapters in this report.

Signal	*Description*
clk	Input clock
rst	Active high, synchronous reset
hold	The control signal for enabling counter output
rst_ram2_counter	Active high, Synchronous reset for address generating counter of Block RAM2 which stores the MAD values
rst_comp_median	Active high, Synchronous reset for comparator block of median
rst_counter_median	Active high, Synchronous reset for a counter block of the median which generates a reset signal for the main accumulator and auxiliary accumulator
rst_comp_mad	Active high, Synchronous reset for comparator block of MAD
rst_counter_mad	Active high, Synchronous reset for a counter block of MAD which generates a reset signal for the main accumulator and auxiliary accumulator
rst_filter	Active high, Synchronous reset for filter block
sel_filter(3 downto 0)	User given input for selecting the input, which will select the data on which threshold will be calculated
8-bit inputs	Data inputs for 16 antennas.
rfi_flag_1 to rif_flag_16	Flag output of the detection block.

Table 4.6 VHDL interface of Single MAD design

4.3.1 Synthesis and Implementation

In this section, Synthesis, and Implementation details are mentioned. This section contains the FPGA Implementation view of the Single-MAD design. Moreover, the Timing and Area-Utilizations details of Implemented designs are discussed in this section.

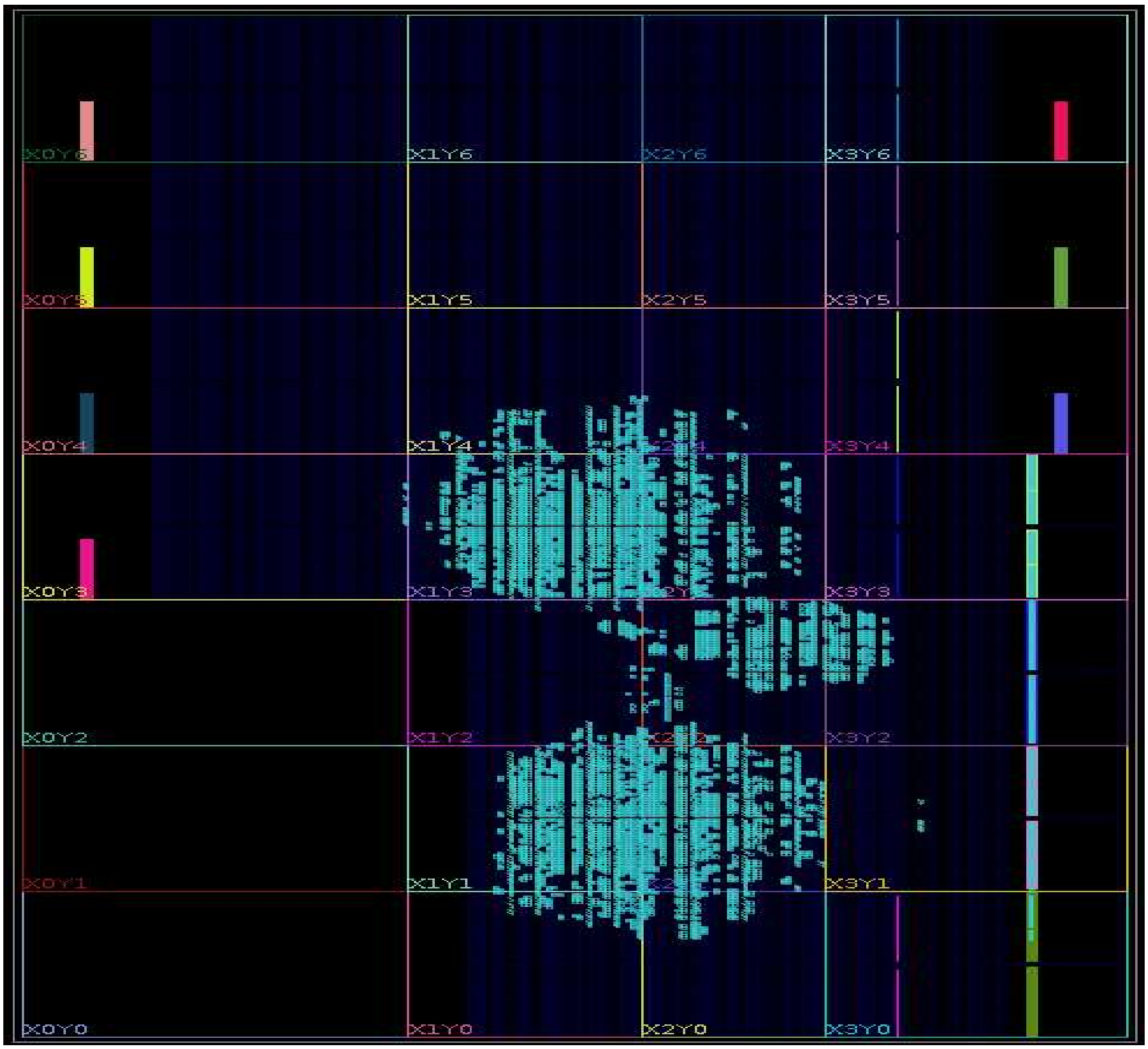

Figure 4.10 Implementation View of Single MAD Design

4.3.2 Timing Details

As shown in Figure 4.7, the Filter design works on 320.5 MHz frequency without any STA violation.

```
-------------------------------------------------------------
| Clock Summary
| -------------
-------------------------------------------------------------

Clock  Waveform(ns)        Period(ns)      Frequency(MHz)
-----  ------------        ----------      --------------
clk    {0.000 1.560}       3.120           320.513

------------------------------------------------------------------------------------------
From Clock:  clk
  To Clock:  clk

Setup :          0  Failing Endpoints,  Worst Slack       0.004ns,  Total Violation       0.000ns
Hold  :          0  Failing Endpoints,  Worst Slack       0.020ns,  Total Violation       0.000ns
PW    :          0  Failing Endpoints,  Worst Slack       1.017ns,  Total Violation       0.000ns
------------------------------------------------------------------------------------------
```

Figure 4.11 Timing Report Single MAD

4.3.3 Utilization Details

Figure 4.8 gives an overall idea of resource utilization. There is no register as a latch.

1. CLB Logic

Site Type	Used	Fixed	Available	Util%
CLB LUTs	7898	0	274080	2.88
LUT as Logic	7898	0	274080	2.88
LUT as Memory	0	0	144000	0.00
CLB Registers	11043	0	548160	2.01
Register as Flip Flop	11043	0	548160	2.01
Register as Latch	0	0	548160	0.00
CARRY8	2711	0	34260	7.91
F7 Muxes	21	0	137040	0.02
F8 Muxes	8	0	68520	0.01
F9 Muxes	0	0	34260	0.00

Figure 4.12 CLB Logic Single MAD

4. ARITHMETIC

Site Type	Used	Fixed	Available	Util%
DSPs	513	0	2520	20.36
DSP48E2 only	513			

2. CLB Logic Distribution

Site Type	Used	Fixed	Available	Util%
CLB	3618	0	34260	10.56
CLBL	1576	0		
CLBM	2042	0		
LUT as Logic	7898	0	274080	2.88
using O5 output only	13			
using O6 output only	3302			
using O5 and O6	4583			
LUT as Memory	0	0	144000	0.00
LUT as Distributed RAM	0	0		
LUT as Shift Register	0	0		
CLB Registers	11043	0	548160	2.01
Register driven from within the CLB	9642			
Register driven from outside the CLB	1401			
LUT in front of the register is unused	1294			
LUT in front of the register is used	107			
Unique Control Sets	34		68520	0.05

9. Primitives

Ref Name	Used	Functional Category
FDRE	11010	Register
LUT2	8275	CLB
CARRY8	2711	CLB
LUT3	1634	CLB
LUT6	908	CLB
LUT5	758	CLB
LUT1	533	CLB
DSP48E2	513	Arithmetic
LUT4	373	CLB
INBUF	148	I/O
IBUFCTRL	148	Others
FDCE	33	Register
MUXF7	21	CLB
OBUF	16	I/O
MUXF8	8	CLB
RAMB36E2	4	Block Ram
BUFGCE	3	Clock

Figure 4.13 Utilization Report Single MAD

4.4 Single MoM Design

The approach that we have taken for MAD we can do the same for MoM-based design, the only difference is that we have taken the first median zero in this design. The architecture is the same as Single MAD see Figure 4.5.

Here is the resource utilization of this design.

Resources	**MAD without filter (%)**	**Multiplexed MAD scheme (V1) (%)**	**Single MAD multiple detection scheme (V2)(%)**	**Multiplexed MAD scheme (accum in dsp)(V3)**	**MAD multiplexed Scheme (first median =0) (V4)**	**Multiplexed MoM (first median=0) (V5)**	**Single MoM First (median=0) (V6)**
LUTS	2.28	2.9	2.88	2.9	1.75	1.68	1.68
SLICES	10.2	11	10.56	10.79	6.20	6.02	6.02
BRAM	0.9	0.44	0.44	0.44	0.44	0.22	0.22
DSPs	0	0.08	0.08	20.40	10.24	10.24	10.24

Table 4.7 Resource utilization Single MoM

We can see in Table 4.4 that the utilization of Slices decreased slightly from 6.20% to 6.02%, further highlighting the successful optimization of the design.

Signal	*Description*
clk	Input clock
rst	Active high, synchronous reset
hold	The control signal for enabling counter output
rst_ram2_counter	Active high, Synchronous reset for address generating counter of Block RAM2 which stores the MAD values
rst_comp_mad	Active high, Synchronous reset for comparator block of MAD
rst_counter_mad	Active high, Synchronous reset for a counter block of MAD which generates a reset signal for the main accumulator and auxiliary accumulator
rst_filter	Active high, Synchronous reset for filter block
sel_filter(3 downto 0)	User given input for selecting the input, which will select the data on which threshold will be calculated
8-bit inputs	Data inputs for 16 antennas.
rfi_flag_1 to rif_flag_16	Flag output of the detection block.

Table 4.8 VHDL interface of Single MOM design

4.4.1 Synthesis and Implementation

In this section, Synthesis, and Implementation details are mentioned. This section contains the FPGA Implementation view of the MAD-Multiplexed design. Moreover, the Timing and Area-Utilizations details of Implemented designs are discussed in this section.

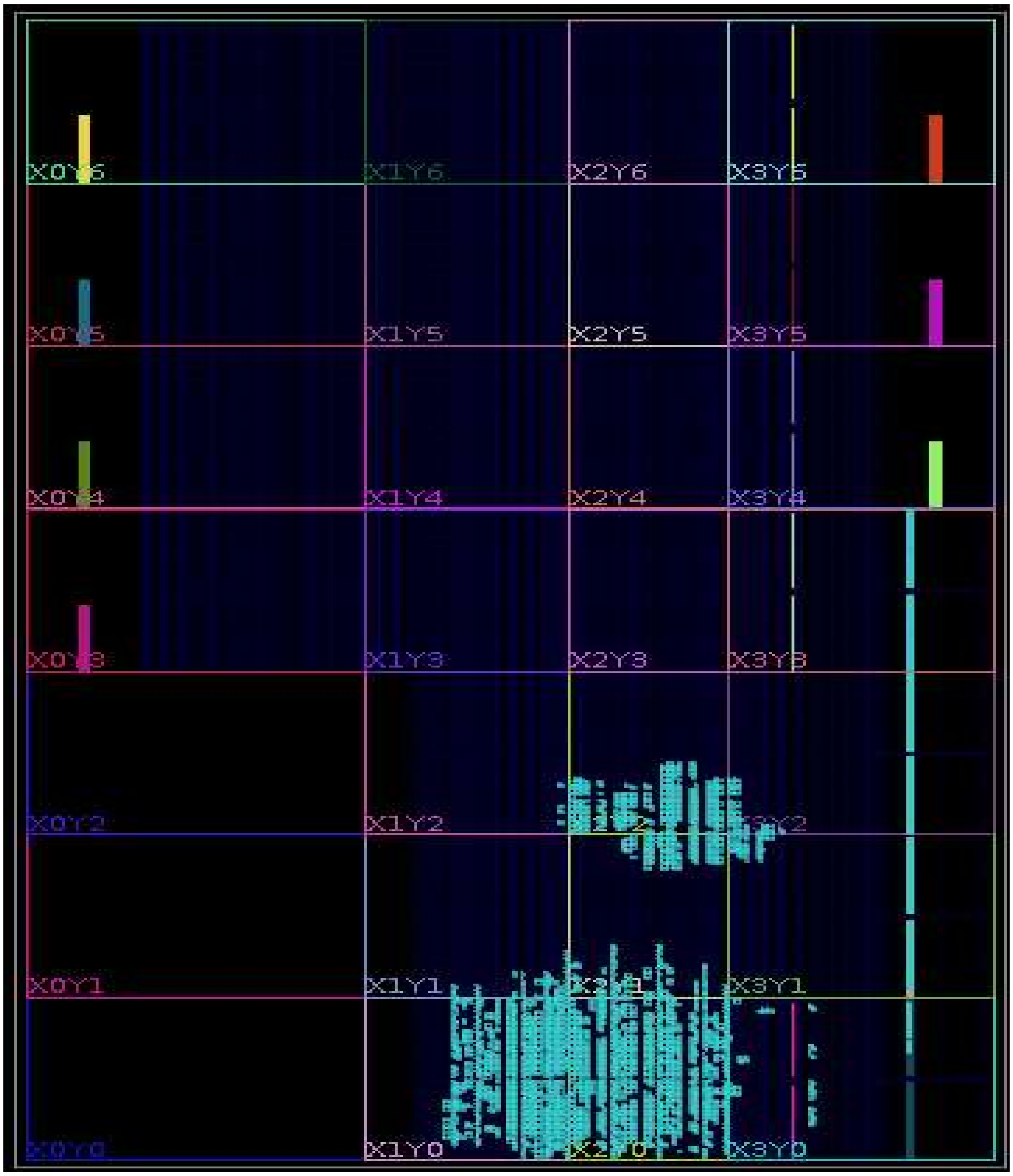

Figure 4.14 Implementation View of Single MoM Design

4.4.2 Timing Details

As shown in Figure 4.11, the Filter design works on 316.456 MHz frequency without any STA violation.

```
------------------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------------------

Clock  Waveform(ns)       Period(ns)     Frequency(MHz)
-----  ------------       ----------     --------------
clk    {0.000 1.580}      3.160          316.456

-----------------------------------------------------------------------------------------------
From Clock:  clk
  To Clock:  clk

Setup :            0  Failing Endpoints,  Worst Slack       0.039ns,  Total Violation       0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack       0.014ns,  Total Violation       0.000ns
PW    :            0  Failing Endpoints,  Worst Slack       1.037ns,  Total Violation       0.000ns
-----------------------------------------------------------------------------------------------
```

Figure 4.15 Timing Report Single MoM

4.4.3 Utilization Details

1. CLB Logic

Site Type	Used	Fixed	Available	Util%
CLB LUTs	4612	0	274080	1.68
LUT as Logic	4612	0	274080	1.68
LUT as Memory	0	0	144000	0.00
CLB Registers	6480	0	548160	1.18
Register as Flip Flop	6480	0	548160	1.18
Register as Latch	0	0	548160	0.00
CARRY8	1437	0	34260	4.19
F7 Muxes	21	0	137040	0.02
F8 Muxes	8	0	68520	0.01
F9 Muxes	0	0	34260	0.00

Figure 4.16 CLB Logic Single MoM

4. ARITHMETIC

Site Type	Used	Fixed	Available	Util%
DSPs	258	0	2520	10.24
DSP48E2 only	258			

2. CLB Logic Distribution

Site Type	Used	Fixed	Available	Util%
CLB	2062	0	34260	6.02
CLBL	949	0		
CLBM	1113	0		
LUT as Logic	4612	0	274080	1.68
using O5 output only	20			
using O6 output only	2109			
using O5 and O6	2483			
LUT as Memory	0	0	144000	0.00
LUT as Distributed RAM	0	0		
LUT as Shift Register	0	0		
CLB Registers	6480	0	548160	1.18
Register driven from within the CLB	4990			
Register driven from outside the CLB	1490			
LUT in front of the register is unused	1286			
LUT in front of the register is used	204			
Unique Control Sets	33		68520	0.05

9. Primitives

Ref Name	Used	Functional Category
FDRE	6447	Register
LUT2	4193	CLB
CARRY8	1437	CLB
LUT3	1087	CLB
LUT5	666	CLB
LUT6	512	CLB
LUT4	325	CLB
LUT1	312	CLB
DSP48E2	258	Arithmetic
INBUF	147	I/O
IBUFCTRL	147	Others
FDCE	33	Register
MUXF7	21	CLB
OBUF	16	I/O
MUXF8	8	CLB
RAMB36E2	2	Block Ram
BUFGCE	2	Clock

Figure 4.17 Utilization Report of Single MoM

Chapter 05: Further Optimization

Design with the first median zero

As we can see in Table 4.1 the utilization of CLB slices has not reduced significantly so we had to optimize the design further, so we came up with an approach and after analyzing data, we suggested that we can make the first median zero.

Resources	***TPM (%)***	***MAD without filter (%)***	***Multiplexed MAD scheme (V1) (%)***	***Multiplexed MAD scheme (accum in dsp)(V2) (%)***	***Single MAD multiple detection scheme (V3)(%)***	***MAD multiplexed Scheme (first median =0) (V4)***
LUTS	*59.9*	*2.28*	*2.9*	*2.9*	*2.88*	*1.75*
SLICES	*88.4*	*10.2*	*11*	*10.79*	*10.56*	*6.20*
BRAM	*42.2*	*0.9*	*0.44*	*0.44*	*0.44*	*0.44*
DSPs	*66.7*	*0*	*0.08*	*20.40*	*20.40*	*10.24*

Table 5.1 Resource Utilization MAD Multipltiplexed with first median 0

As it is evident from Table 5.1, there has been a noticeable decline in slice utilization, dropping from 11% to 6.20%, while the DSP utilization has also decreased from 20.40% to 10.24%, considering that the initial median is zero.

Signal	*Description*
clk	Input clock
rst	Active high, synchronous reset
hold	The control signal for enabling counter output
rst_ram2_counter	Active high, Synchronous reset for address generating counter of Block RAM2 which stores the MAD values
rst_comp_median	Active high, Synchronous reset for comparator block of median
rst_counter_median	Active high, Synchronous reset for a counter block of the median which generates a reset signal for the main accumulator and auxiliary accumulator
rst_comp_mad	Active high, Synchronous reset for comparator block of MAD
rst_counter_mad	Active high, Synchronous reset for a counter block of MAD which generates a reset signal for the main accumulator and auxiliary accumulator
rst_filter	Active high, Synchronous reset for filter block
rst_count	Active high synchronous reset for counter that controls mux and demux
count	The control signal for the counter that controls mux and demux
8bit inputs	Data inputs for 16 antennas.
rfi_flag_1 to rif_flag_16	Flag output of the detection block.

Table 5.2 VHDL interface of MAD multiplexed design with zero median

5.1 Synthesis and Implementation

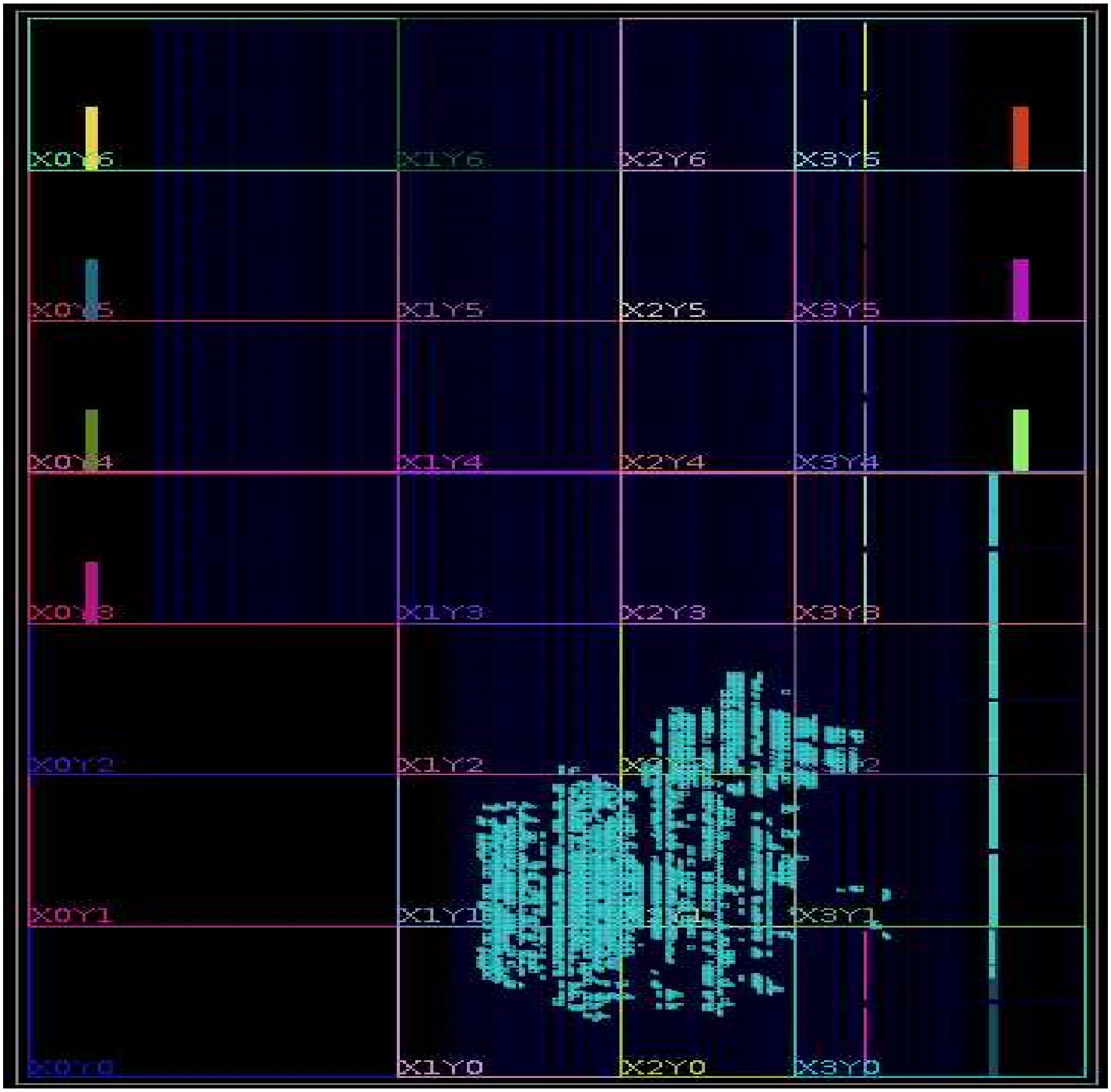

Figure 5.1 Implementation View of MAD Multiplexed first Median zero

5.2 Timing Details

As shown in Figure 5.2, the Filter design works on 320.5 MHz frequency without any STA violation.

```
---------------------------------------------------------------------
| Clock Summary
| -------------
---------------------------------------------------------------------

Clock  Waveform(ns)        Period(ns)      Frequency(MHz)
-----  ------------        ----------      --------------
clk    {0.000 1.580}       3.160           316.456

---------------------------------------------------------------------------------------
From Clock:  clk
  To Clock:  clk

Setup :           0  Failing Endpoints,  Worst Slack       0.027ns,  Total Violation       0.000ns
Hold  :           0  Failing Endpoints,  Worst Slack       0.019ns,  Total Violation       0.000ns
PW    :           0  Failing Endpoints,  Worst Slack       1.037ns,  Total Violation       0.000ns
---------------------------------------------------------------------------------------
```

Figure 5.2 Clock Summary MAD Multiplexed first Median zero

5.3 Utilization Details

```
1. CLB Logic
------------

+--------------------------+------+-------+-----------+-------+
|        Site Type         | Used | Fixed | Available | Util% |
+--------------------------+------+-------+-----------+-------+
| CLB LUTs                 | 4793 |     0 |    274080 |  1.75 |
|   LUT as Logic           | 4793 |     0 |    274080 |  1.75 |
|   LUT as Memory          |    0 |     0 |    144000 |  0.00 |
| CLB Registers            | 7384 |     0 |    548160 |  1.35 |
|   Register as Flip Flop  | 7384 |     0 |    548160 |  1.35 |
|   Register as Latch      |    0 |     0 |    548160 |  0.00 |
| CARRY8                   | 1427 |     0 |     34260 |  4.17 |
| F7 Muxes                 |   21 |     0 |    137040 |  0.02 |
| F8 Muxes                 |    8 |     0 |     68520 |  0.01 |
| F9 Muxes                 |    0 |     0 |     34260 |  0.00 |
+--------------------------+------+-------+-----------+-------+
```

Figure 5.3 CLB LogicMAD Multiplexed first Median zero

2. CLB Logic Distribution

Site Type	Used	Fixed	Available	Util%
CLB	2125	0	34260	6.20
CLBL	919	0		
CLBM	1206	0		
LUT as Logic	4793	0	274080	1.75
using O5 output only	13			
using O6 output only	2032			
using O5 and O6	2748			
LUT as Memory	0	0	144000	0.00
LUT as Distributed RAM	0	0		
LUT as Shift Register	0	0		
CLB Registers	7384	0	548160	1.35
Register driven from within the CLB	5535			
Register driven from outside the CLB	1849			
LUT in front of the register is unused	1692			
LUT in front of the register is used	157			
Unique Control Sets	44		68520	0.06

4. ARITHMETIC

Site Type	Used	Fixed	Available	Util%
DSPs	258	0	2520	10.24
DSP48E2 only	258			

9. Primitives

Ref Name	Used	Functional Category
FDRE	7384	Register
LUT2	4452	CLB
CARRY8	1427	CLB
LUT3	1341	CLB
LUT5	624	CLB
LUT6	500	CLB
LUT4	316	CLB
LUT1	308	CLB
DSP48E2	258	Arithmetic
INBUF	145	I/O
IBUFCTRL	145	Others
MUXF7	21	CLB
OBUF	16	I/O
MUXF8	8	CLB
RAMB36E2	4	Block Ram
BUFGCE	2	Clock

Figure 5.4 Utilization Report MAD Multiplexed first Median zero

Chapter 06: Resource Comparison and Verification

Here is the complete comparison of all the designs that we have tested, Figure 6.1 shows the Graphical comparison. Table 6.1 provides a comparison between the clock frequency of all designs, while Figure 6.2 gives a graphical representation.

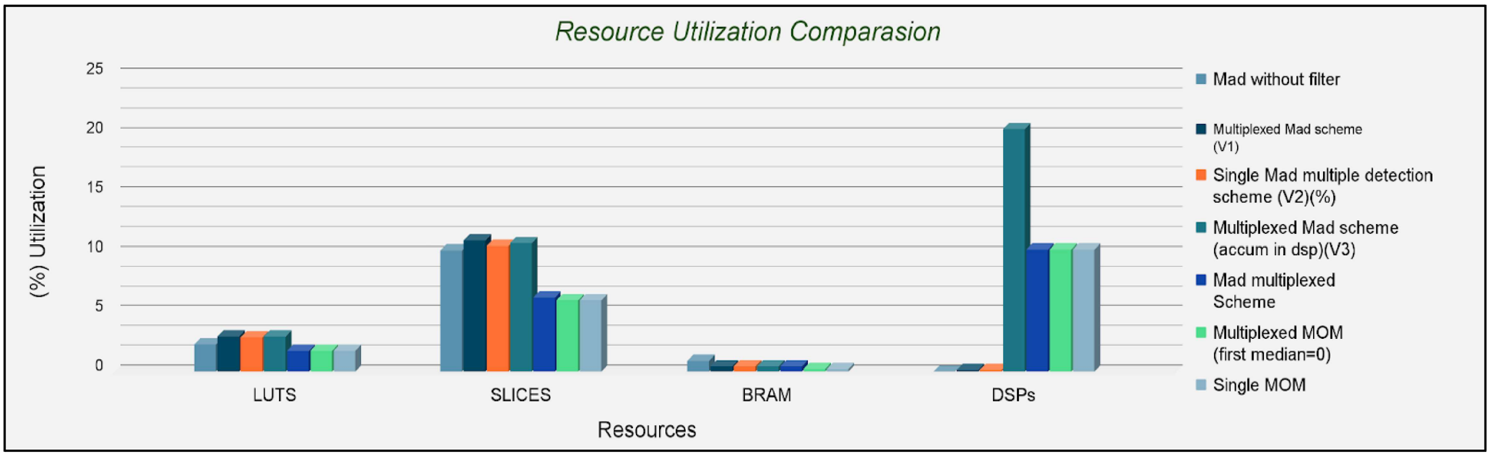

Figure 6.1 Graphical Comparison of Resource Utilization across different methods

Resources	**MAD without filter (%)**	**Multiplexed MAD scheme (V1) (%)**	**Single MAD multiple detection scheme (V2)(%)**	**Multiplexed MAD scheme (accum in dsp)(V3)**	**MAD multiplexed Scheme (first median =0) (V4)**	**Multiplexed MoM (first median=0)**	**Single MoM First (median=0)**
Clk freq(Mhz)	238.095	312.5	320.5	316.5	320.5	312.5	316.5

Table 6.1 Clock Frequency comparison

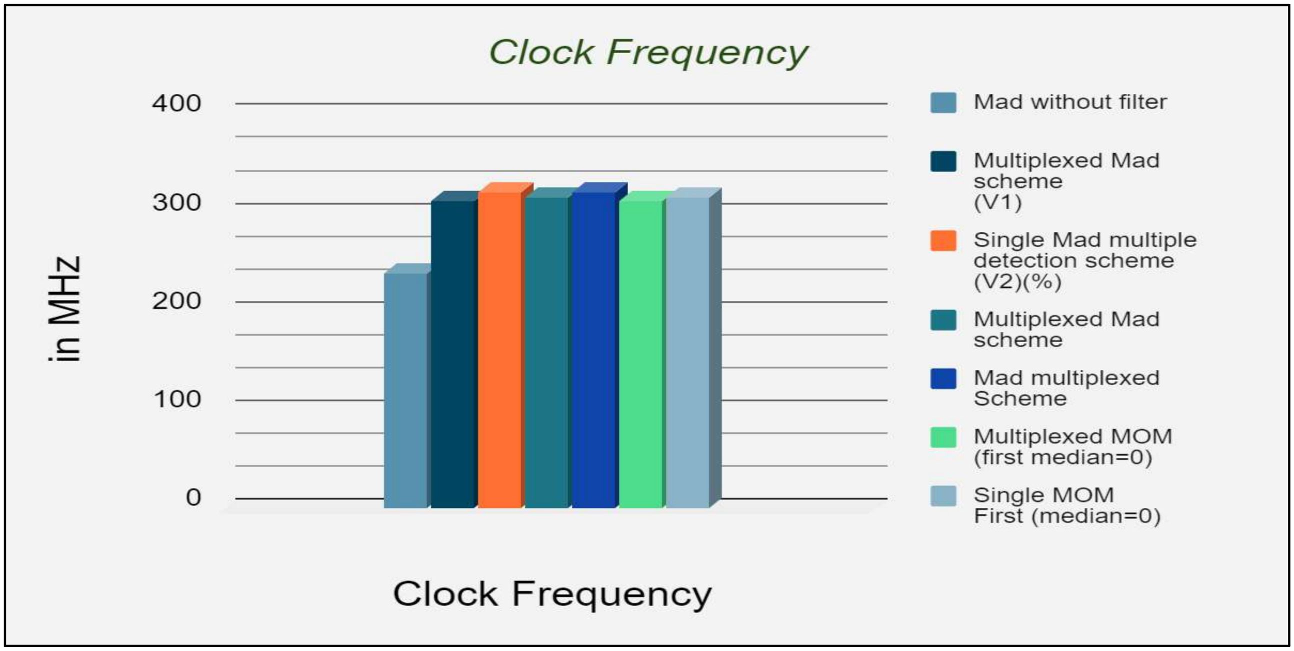

Figure 6.2 Clock Frequency for different schemes

Verification with Matlab

Verification of these designs have been done using MATLAB, and code with similar functionality has been developed in MATLAB. A C++ program was used to compare the output of VHDL and MATLAB. Figure 6.3 shows the flow of verification.

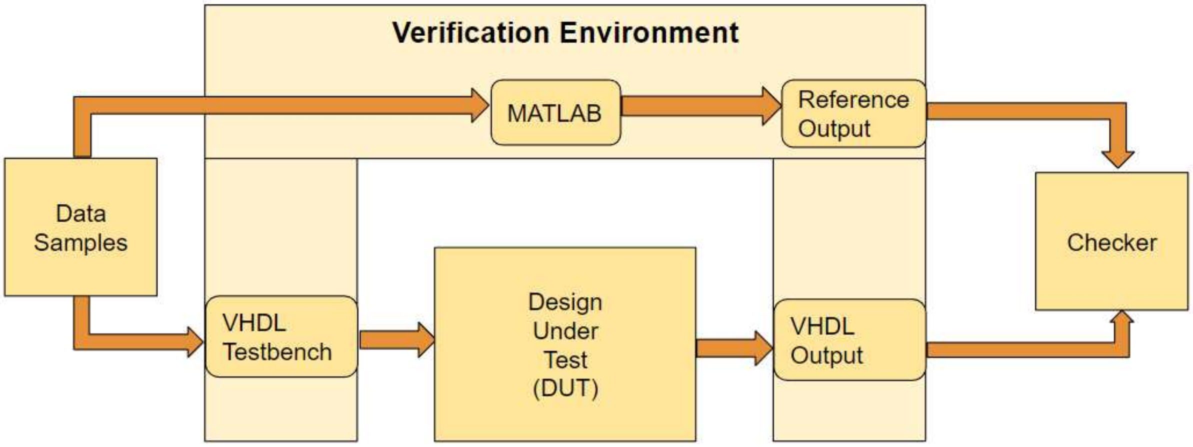

Figure 6.3 Verification Environment

VHDL Testbench

To test the designs we have written testbench, in testbench TPM extracted data is given input to the design, and the output is stored in text files which are further used for verification with MATLAB.

The VHDL testbench has 16 inputs for 16 antennas in total 128 bits and for output, we have 16 bits 1 bit flag output for each antenna. For testing purposes, we have also added assert statements to the testbench. The messages/warnings of the assert statement could be seen either on the **TCL console** or in the **simulate.log** file.

Conclusion and Future Scope

The implementation of MAD-based RFI detection using VHDL was performed on the Xilinx Kintex Ultrascale FPGA device. Multiple architectures were created and tested. Additionally, a Python-based graphical user interface (GUI) was developed for analyzing TPM data. To optimize the design, the accumulators were efficiently utilized by incorporating slight modifications and leveraging the built-in accumulators in the DSP.[6] As a result, the design achieved the desired resource utilization and clock frequency targets.

Additional data can be employed for further testing of these designs, and adjustments can be made to facilitate integration within the TPM module. Additionally, it is recommended to conduct an in-depth analysis of the TPM data to extract valuable insights.

References

[1] Low-Frequency Radio Astronomy
http://www.ncra.tifr.res.in/ncra/gmrt/gmrt-users/low-frequency-radio-astronomy

[2]Kaushal D. Buch, Kishor Naik, Swapnil Nalawade, Shruti Bhatporia, Yashwant Gupta and B. Ajithkumar, "Real-Time Implementation of MAD-Based RFI Excision on FPGA" 2019 Radio Frequency Interference (RFI), 2016, pp. 11-15, doi: 10.1109/RFINT.2016.7833523.

[3] Square Kilometer Array (SKA) India http://www.ncra.tifr.res.in/ncra/skaindia/ska-india

[4]G. Comoretto1, C. Belli1, SKA Project Series, LFAA tile processing module programming manual.

[5]Patel, Sohan, "VHDL Implementation of MAD-based RFI Filtering", GMRT STP Project Report, July 2022

[6]https://docs.xilinx.com/r/e-US/ug953-vivado-7series-libraries/CARRY4

Appendix

Generic value to be defined in top entity

Generic name	Value	Description
data_width	8	8 bit signed integer data
reg_width	14 and 12	16384 for MAD-based designs and 4096 for MoM-based designs
count_width	32	Counter width
mode_select	2	4 different filtering options
reference_value	user defined	For comparison with rfi count of each window

Table A.1 Generic Description

Versions of Designs

Design Name	**Description**
MAD Multiplexed Design V1	Takes 16 inputs and calculates threshold in round-robin fashion.
Single MAD Design V2	Takes 16 inputs and calculates the threshold on only single input.
MoM Multiplexed Design V3 (first median zero)	Takes 16 inputs and calculates the threshold in Round Robin fashion. Taking the first median zero.
Single MoM Design V4 (first median zero)	Takes 16 inputs and calculates the threshold on only single input. Taking the first median zero.

MAD Multiplexed V5 (first median zero)	Takes 16 inputs and calculates the threshold in a round-robin fashion but with first median zero.

Table A.2 Versions of Design

Output Comparison Table

The output comparison of MATLAB and VHDL is given for MAD-Multiplexed Design, see Table 6.3.

MAD Multiplexed Design V1 (16k)		
File name		
raw_burst_15_1.out	**Same values**	**Different values**
flag1	676	1
flag2	666	2
flag3	665	1
flag4	664	0
flag5	661	1
flag6	658	2
flag7	656	2
flag8	655	1
flag9	638	16
flag10	639	13
flag11	640	10
flag12	632	15
flag13	638	7
flag14	639	4
flag15	638	3
flag16	632	8

Table A.3 Output Comparison