VHDL Implementation of
MAD-based RFI Filtering Algorithm

Student Project
By
Sohankumar H. Patel

Electronics And Communication Engineering
Sardar Vallabhbhai National Institute of Technology,
Surat

Under the Guidance of
Mr. Kaushal Buch

NCRA • TIFR

GIANT METREWAVE RADIO TELESCOPE

NATIONAL CENTER FOR RADIO ASTROPHYSICS

TATA INSTITUTE OF FUNDAMENTAL RESEARCH

Narayangaon,Tal-Junnar,Dist-Pune

May 2022 - July 2022

ABSTRACT

In this era of Radio Communication technology, one finds much-unwanted radiation during the reception of the signal. For Radio Astronomy, the signal itself is feeble. Thus, any other radio frequency interference (RFI), man-made or natural causes, makes it challenging to analyze. This project explores the VHDL implementation of the Median Absolute Deviation (MAD) based RFI Filtering algorithm on the upgraded GMRT system. MAD computation requires recursive median calculation, which is a computationally challenging problem for real-time performance. The optimized approach is a histogram-based median computation method to meet real-time filtering.

This report describes the FPGA implementation of the MAD-based RFI filtering algorithm. Further, another variant of MAD called Median-of-MAD (MoM) is described. Both methods detect the data samples in the voltage domain only. Another way to detect RFI is to detect the data samples in the power domain. FPGA implementation of this algorithm is also described in this report.

All these implementations are done on the Xilinx Kintex Ultrascale FPGA device. The same algorithm is implemented on MATLAB as the golden model, and the functional verification of the design is carried out by using it.

Acknowledgements

I express my sincere gratitude and indebtedness to my guide Mr. Kaushal D. Buch, Engineer E, Digital Backend Group, GMRT, for their valuable guidance, encouragement, and suggestions throughout the work. Without him, I would not have been able to complete this report. His encouragement and valuable lessons kept me moving in the challenging parts. I heartily thank him for his constant support and guidance throughout.

I am grateful to the Group Coordinator, GMRT Backend Group, Mr. B. Ajith Kumar, for the cooperation and support and for allowing me to work in the Digital Back-end Group.

I also thank Prof. Yashwant Gupta, Center Director, Prof. Nissim Kanekar, Dean NCRA Faculty, and Prof. Jayaram Chengalur, Director TIFR, for allowing me to work at GMRT.

I sincerely thank the Digital Backend group members and the entire staff of GMRT. Everyone who met me taught me something and motivated me to work.

Sohankumar Patel

Contents

List of Figures

List of Tables

Chapter 01: Introduction

1.1 Overview

Giant Metrewave Radio Telescope is an observatory that explores the metre wavelength range of the radio spectrum. It is set up by the National centre for Radio Astrophysics (NCRA). GMRT consists of 30 fully steerable gigantic parabolic dishes of 45m in diameter, which are spread over distances of up to 25 km. The site was selected because it fulfills the following important criteria as

1) Low man-made radio noise.
2) Low wind speed.
3) It has a geographical latitude that is sufficiently north of the geomagnetic equator to have a reasonably quiet ionosphere and yet be able to observe a good part of the southern sky.

Figure 1.1 GMRT Antennas (Courtesy: NCRA Archives)

In the electromagnetic spectrum, the metre wavelength range of spectrum has been mainly chosen for study with GMRT because man-made radio interference is relatively less in this part of the spectrum in India, and many outstanding astrophysical phenomena are best studied at metre wavelengths [1].

1.2 Real-time RFI filtering for uGMRT

Radio Frequency Interference (RFI) is the 'unwanted signal' produced due to Natural and Man-made activities. Audio-Visual transmissions, RADAR, Satellite communication, Wi-Fi networking, power-line radiation, spark ignition noise, and radiation due to electronic devices are considered Man-made radio interference. Radio astronomical signals are very weak (typically -110 dBm at the input of the radio telescope receiver). Thus radio telescopes are very sensitive to capturing such weak signals here and are more prone to interference. This interference can severely damage the quality of the data, and excision of it can also cause loss of data.

There are two methods for handling RFI situation

1. Proactive methods - Prevention of RFI by establishing 'Radio Quiet Zones'
2. Reactive methods – Signal processing for RFI mitigation after reception of the astronomical data at different stages of the receiver chain.

Digital signal processing can help improve the data signal quality effectively by removing RFI without much loss of quality.

1.3 VHDL Implementation of RFI Filter

As shown in figure 1.2, the block takes signed 8-bit quantized data from the ADC of the ROACH board and performs the Histogram method to find the median of the window data. The calculated median is subtracted from each element of the window, and the element is then converted into an unsigned integer. Again the median of the updated window is calculated, which is the Median Absolute Deviation (MAD) value for the data window. Median and MAD values coming from the Median and MAD calculation block are used for the calculation of the threshold value of the signal. For that, the MAD value is multiplied by the constant value of 1.4826 to decide the window's variance. The Median value is added and subtracted from the output in deciding the positive and negative threshold, respectively. After obtaining the threshold values for one channel, data from all the channels are compared and flagged. The replacement in place of the flagged data is user-defined.

The other design in which the Median of MAD values has been carried out is called Median of MAD (MoM). This MoM value is used to compute threshold values.

The design is implemented using Xilinx Vivado software and on the Xilinx Kintex Ultrascale (XCKU series) FPGA board. MATLAB software is used to implement the same algorithm for functional verification.

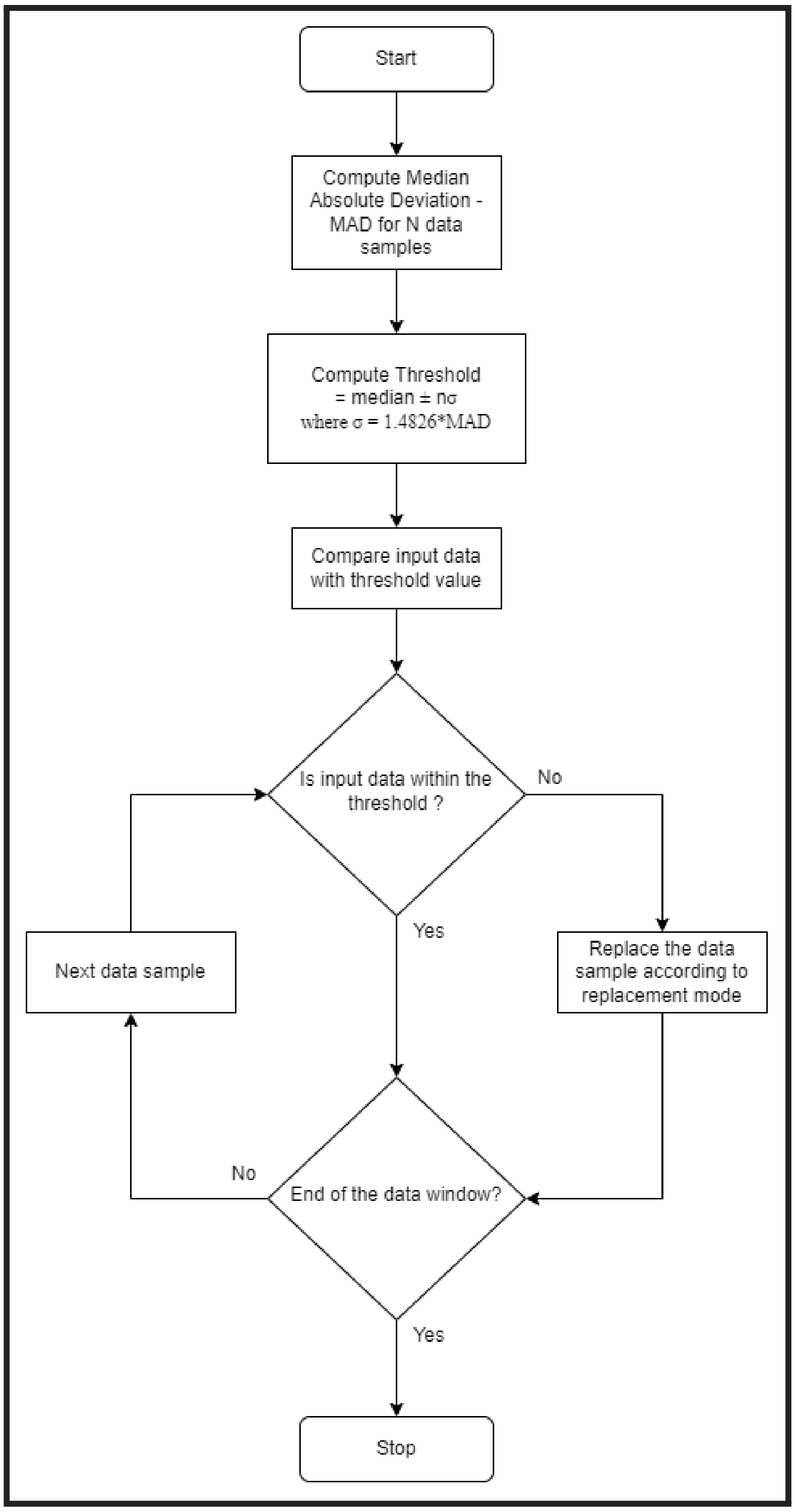

Start
Compute Median Absolute Deviation - MAD for N data samples
Compute Threshold = median ± nσ where σ = 1.4826*MAD
Compare input data with threshold value
Is input data within the threshold ?
No
Yes
Next data sample
Replace the data sample according to replacement mode
No
End of the data window?
Yes
Stop

Figure 1.2 Flowchart of MAD-based filtering algorithm

Chapter 02: Histogram-based Median & MAD computation

2.1 Median computation

There are many techniques to compute the median, like heap sort, bubble sort, Histogram, etc., but for meeting real-time processing requirements, Histogram-based Median computation is the optimal choice. The architecture of Histogram-based Median computation is shown in figure 2.1. The cumulative distribution is computed with the help of a comparator and accumulator block. First, the data are given to the comparator block and, based on compare value, generate the enable signal for the accumulator. At every high enable signal, the accumulator is increased by one value. For n bit sign numbers, there is a need for 2^n comparators, and comparison happens in the range of $-2^{(n-1)}$ to $2^{(n-1)}$ -1.

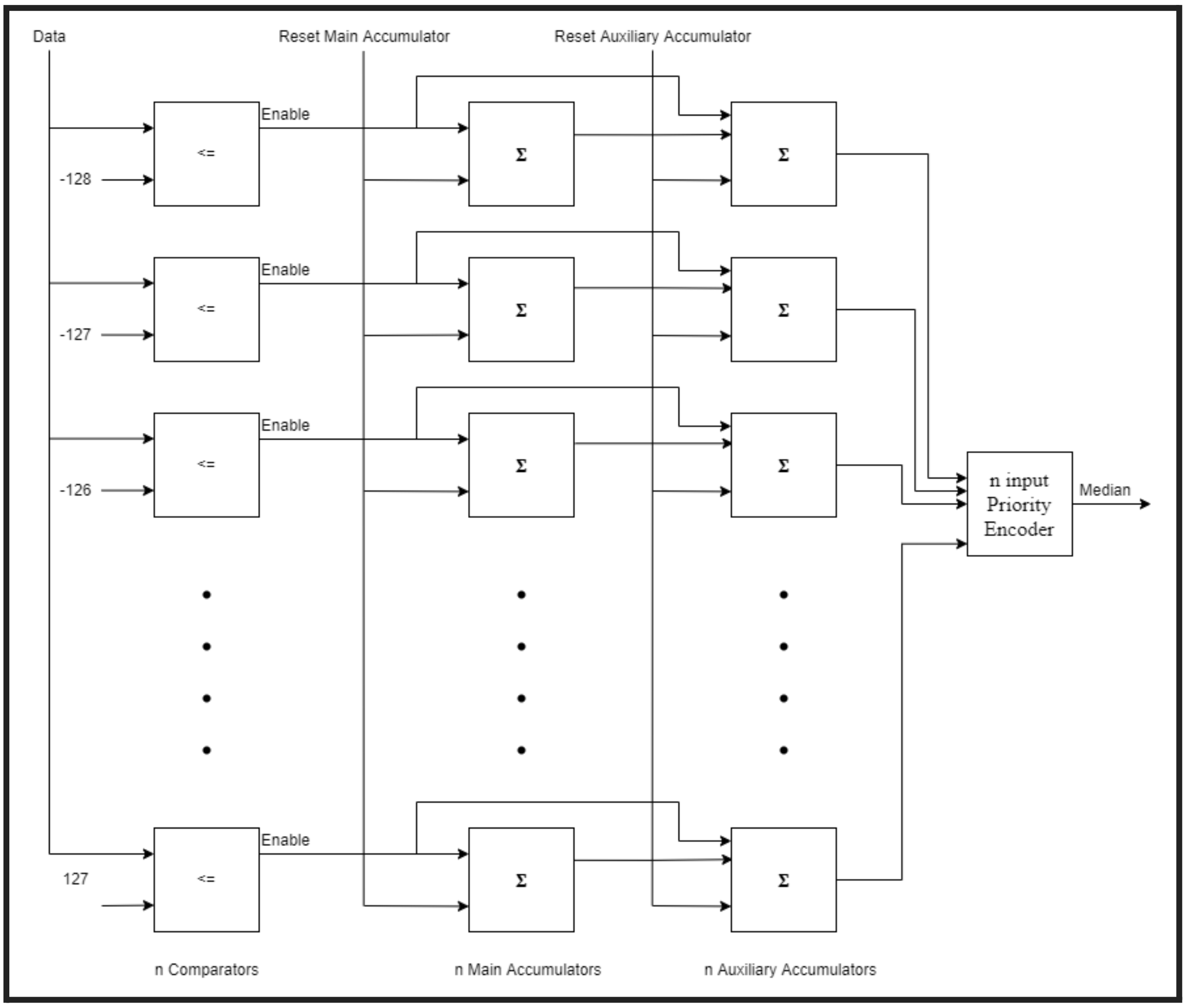


Figure 2.1 Block diagram of Histogram based Median computation method

For continuous operation on subsequent windows, the accumulator needs to be reset, which requires one clock cycle. To achieve real-time processing, an auxiliary accumulator is used to compute accumulation for the last window cycle, and the main accumulator gets reset on that cycle. So there is a need for a 2^n main accumulator and 2^n auxiliary accumulator. The output of the accumulator is given to the priority encoder, which finds the first value for which the accumulator output is greater than or equal to half of the window size, and the corresponding comparison value is the median of that particular window.

Along with the median, one control signal indicates a new Median value is computed. The current value gets registered till the new median value is calculated.

2.2 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 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).

$$\text{MAD} = \text{Median}(|\ X_i - \overline{X}\ |)$$

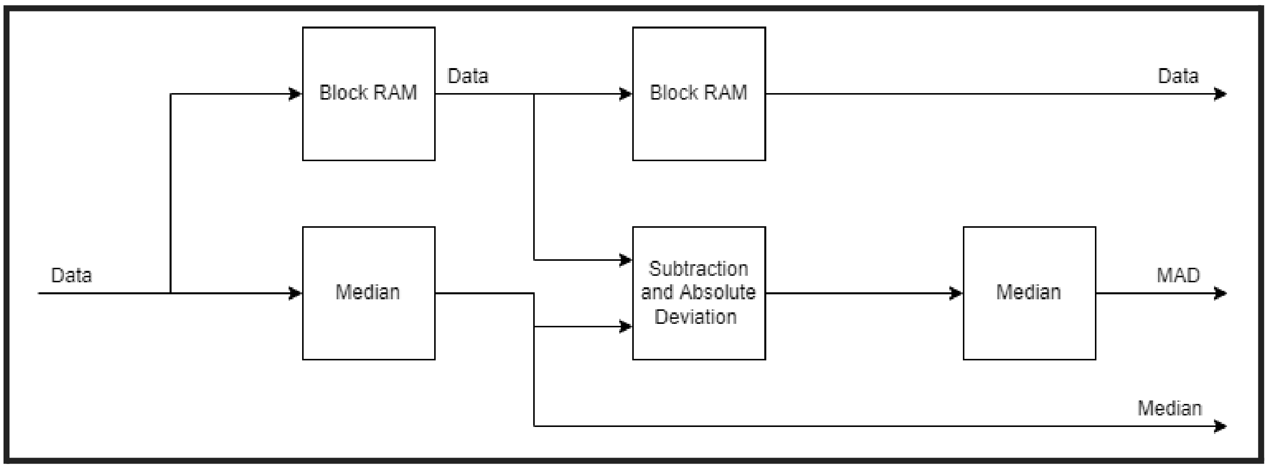


Figure 2.2 Block diagram of Median Absolute Deviation (MAD) computation

The median computation takes a W clock cycle, and here MAD requires two median computations, so there is a need to buffer the data for 2W clock cycles. For this purpose, two Block RAM IPs are used.

2.3 Implementation details

This section discusses the block level interface of Median, Subtraction Absolute deviation, and Block RAM IP. Each table is shown below consists of I/O ports and its description. The Block diagram view of Histogram based median block is shown in figure 2.3.

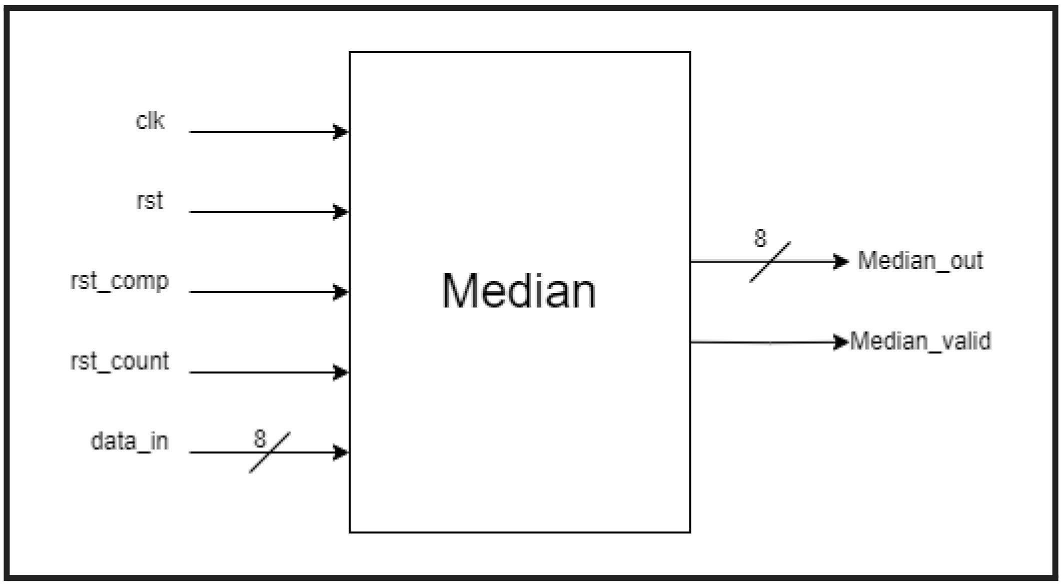


Figure 2.3 Block diagram of Median

Table 2.1 VHDL interface of Histogram based Median

Signal	Description
clk	Input clock
rst	Active high, Synchronous reset
rst_comp	Active high, Synchronous reset for comparator block
rst_count	Active high, Synchronous reset for counter block which generates reset signal for main accumulator and auxiliary accumulator
data_in [7 downto 0]	Input data
median_out [7 downto 0]	Output Median value
median_valid	Output Median valid control signal

Table 2.2 VHDL interface of Subtraction and Absolute Deviation

Signal	Description
clk	Input clock
rst	Active high, Synchronous reset
data_in [7 downto 0]	Input data
median [7 downto 0]	Input Median value

median_valid	Input Median valid control signal
data_out [7 downto 0]	Output = Absolute (data_in - median)

Table 2.3 VHDL interface of Block RAM

Signal	Description
BRAM_PORTA_0_addr [13 downto 0]	Writing port address
BRAM_PORTA_0_clk	Port A clock
BRAM_PORTA_0_din [7 downto 0]	PortA - input writing data
BRAM_PORTA_0_we	PortA - write enable signal
BRAM_PORTB_0_addr [13 downto 0]	Reading port address
BRAM_PORTB_0_clk	Port A clock
BRAM_PORTB_0_dout [7 downto 0]	PortB - output read data

Chapter 03: MoM computation

3.1 Algorithm

This technique is most preferable for longer bursts of RFI. In this technique, the Median of MAD values is carried out, called the Median of MAD - MoM.

$$\textbf{MoM} = \textbf{Median}(\textbf{MAD}_1,\textbf{MAD}_2,\textbf{MAD}_3,.....\textbf{MAD}_n)$$

where n = window size

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. The MAD value is stored in Block memory for the W window cycle. At the last MoM window cycle, the primary data path for the MAD computation block is switched to provide the initially stored MAD value, and the median of these values has been computed. This median value is called as Median of MAD - MoM. For the same window cycle, W samples are not considered for computing the MAD value and do not affect the overall performance. The current calculated MoM value is registered till the new value is calculated. Control signals for this switching are given for one clock cycle.

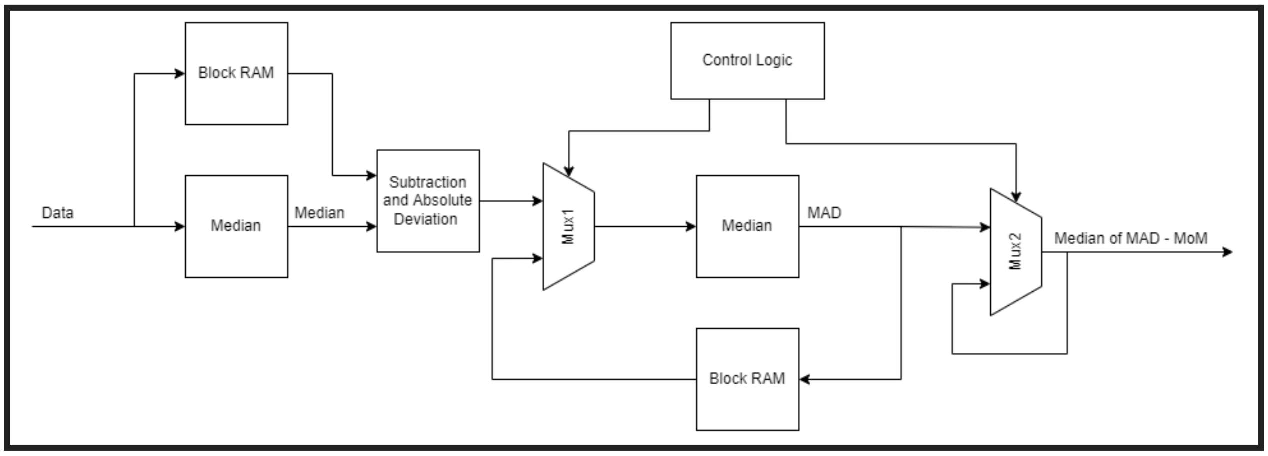


Figure 3.1 Block diagram of Median of MAD - MoM computation method

3.2 Implementation details

This section discusses the MoM computation block's block diagram interface with its I/O interface.

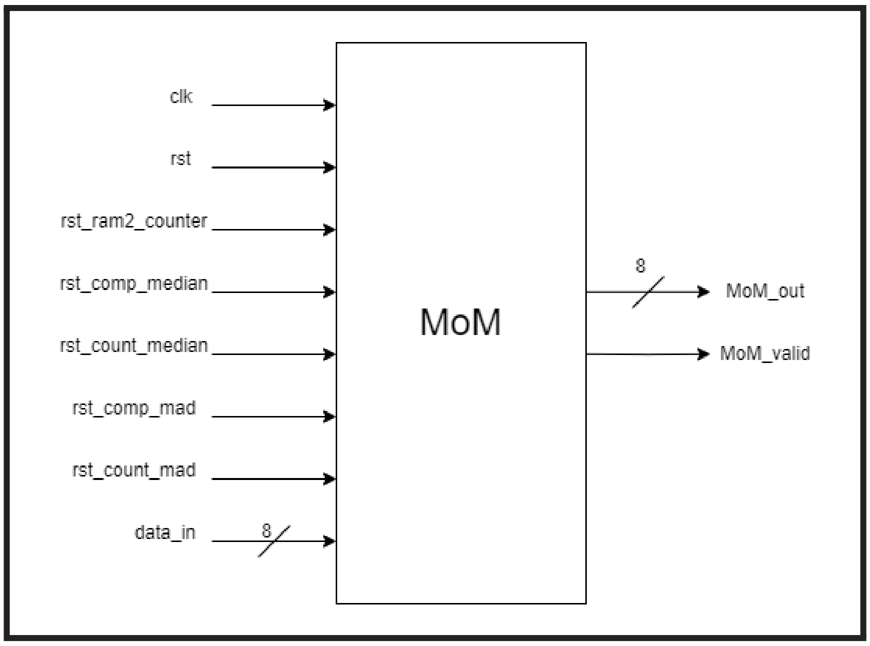


Figure 3.2 Block diagram view of MoM computation block

Table 3.1 VHDL interface of MoM computation block

Signal	Description
clk	Input clock
rst	Active high, Synchronous reset
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_count_median	Active high, Synchronous reset for counter block of median which generates reset signal for main accumulator and auxiliary accumulator
rst_comp_mad	Active high, Synchronous reset for comparator block of mad
rst_count_mad	Active high, Synchronous reset for counter block of mad which generates reset signal for main accumulator and auxiliary accumulator
data_in [7 downto 0]	Input data
mom_out [7 downto 0]	Output MoM value
mom_valid	Output MoM valid control signal

Chapter 04: Filtering

4.1 Algorithm

Filter block contains Threshold calculation, Detection, Filtering, and RFI and data counting algorithm. The threshold is computed using the Median and MAD values in the threshold calculate block. MAD value is scaled by 1.4826 to calculate the robust standard deviation (σ). The threshold factor(n) is multiplied by the standard deviation. The product is added to the median value to calculate the Upper Threshold and subtracted from the median value to calculate the Lower Threshold value.

$$\sigma = 1.4826 * MAD$$
$$\text{Lower Threshold} = \overline{X} - n*\sigma$$
$$\text{Upper Threshold} = \overline{X} + n*\sigma$$

The input data is compared with Lower and Upper Threshold values, and if data is beyond the threshold boundary, then it is detected as RFI, and the flag is asserted. This flag is used as a control signal for replacing multiplexers and counting the RFI. RFI samples are replaced by Constant values, the Threshold value, or Digital Noise which is decided by the Replacement mode control signal.

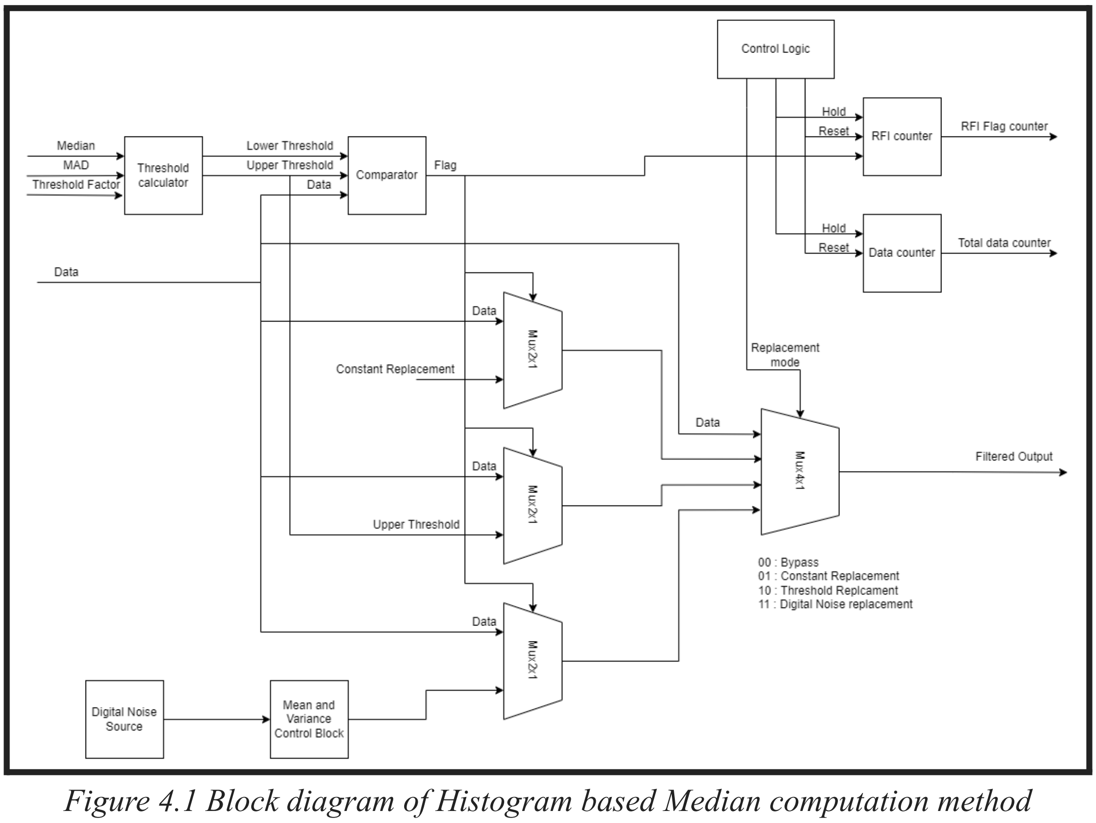


Figure 4.1 Block diagram of Histogram based Median computation method

Digital Noise is generated using Central Limit Theorem. Uniform Normal distribution is generated by Linear Feedback Shift Register (LFSR) of different polynomials, which are then added to generate Standard Normal distribution. This Standard Normal distribution is scaled according to the estimated Median and MAD of the input data.

Moreover, there are two counters for counting the data samples and RFI samples for the given time duration. Data count is incremented by one at every clock cycle, whereas the RFI count is only incremented when the RFI flag generated by the comparator block is high. There are two control signals: reset and hold. Reset is used to reset the counter and start counting from the zero value. The assertion of the Hold signal stops the counting, and the counting value is available at the output.

4.2 Implementation details

The block level diagram of Filter design is shown in Figure 4.2, and I/O interfaces of Threshold calculation, comparator, counters, and Filter blocks are described in table 4.2, 4.3, 4.4, and 4.5, respectively. There are four replacement modes, as discussed in section 4.1. Table 4.1 describes the control signals along with their description.

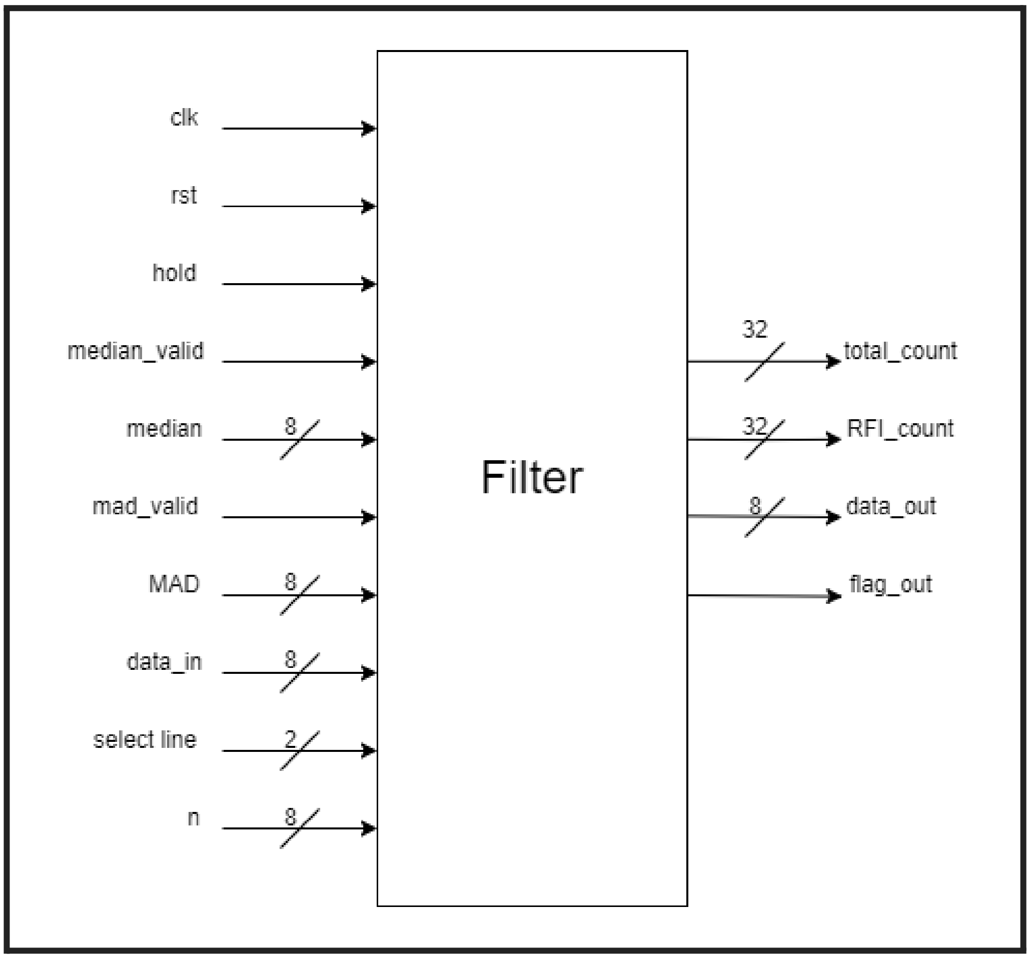


Figure 4.2 Block diagram of Filter design

Table 4.1 Filtering Replacement Modes

Select line controls	Filtering Modes	Description
00	Bypass mode	Only flag is generated, RFI data is not filtered out
01	Constant replacement mode	RFI data is replaced by constant values i.e, 0
10	Threshold replacement mode	RFI data is replaced by Upper threshold value.
11	Digital Noise replacement	RFI data is replaced by Digital Noise which is generated by Digital Noise source.

Table 4.2 VHDL interface of Threshold Calculation block

Signal	Description
clk	Input clock
rst	Active high, Synchronous reset
median [7 downto 0]	Input Median value
mad [7 downto 0]	Input MAD value
n [7 downto 0]	Input Threshold factor
Sigma [7 downto 0]	Output : Sigma = 1.4826*MAD
Lower_threshold [7 downto 0]	Output : LT = median - n*Sigma
Upper_threshold [7 downto 0]	Output : UT = median + n*Sigma

Table 4.3 VHDL interface of Comparator block

Signal	Description
data_in [7 downto 0]	Input data
Lower_threshold [7 downto 0]	Input Lower Threshold
Upper_threshold [7 downto 0]	Input Upper Threshold
data_out	Output RFI flag

Table 4.3 VHDL interface of RFI and Data counter block

Signal	total_data_count	total_RFI_count
clk	Input clock	Input clock
rst	Active high, Synchronous reset	Active high, Synchronous reset
hold	Control signal for enable output	Control signal for enable output
data_in	-	Input RFI flag
out_total_count [31 downto 0]	Total data count value	Total RFI count value

Table 4.4 VHDL interface of RFI Filter block

Signal	Description
clk	Input clock
rst	Active high, Synchronous reset
hold	Control signal for enable counter output
median_valid	Input Median valid signal
mad_valid	Input MAD valid signal
data_in [7 downto 0]	Input data
select_line [1 downto 0]	Replacement mode control signal
median [7 downto 0]	Input Median value
mad [7 downto 0]	Input MAD value
n [7 downto 0]	Input Threshold factor
total_count [31 downto 0]	Output data count value
RFI_count [31 downto 0]	Output RFI count value
data_out [7 downto 0]	Filtered output
flag_out	Output RFI flag

4.3 Synthesis and Implementation

In this section, Synthesis and Implementation details are mentioned. Section 4.3.1 contains the FPGA Implementation view of the RFI Filter design. Timing and Area-Utilizations details of the Implemented design are discussed in 4.3.2 and 4.3.3, respectively.

4.3.1 Implementation View

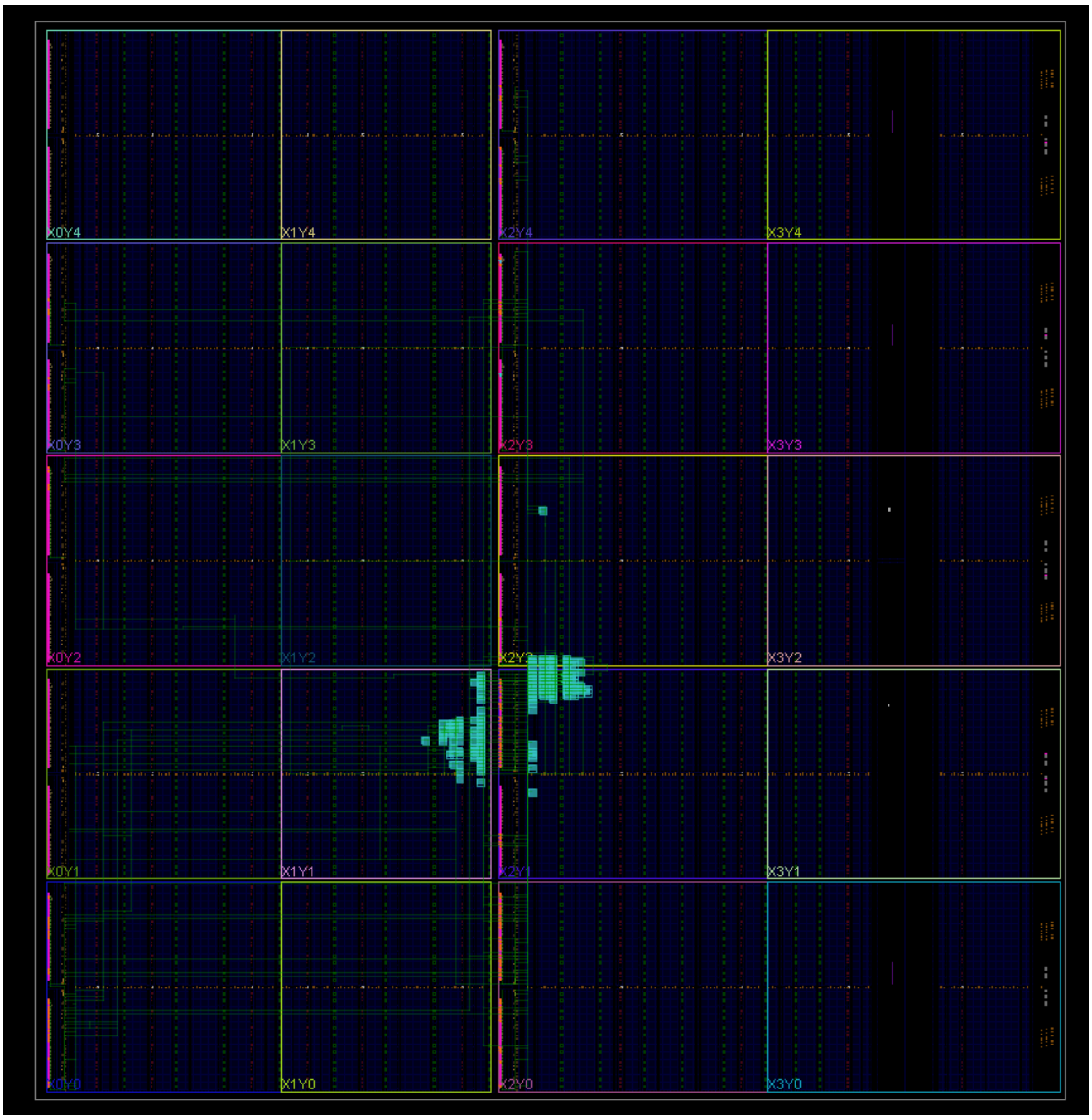
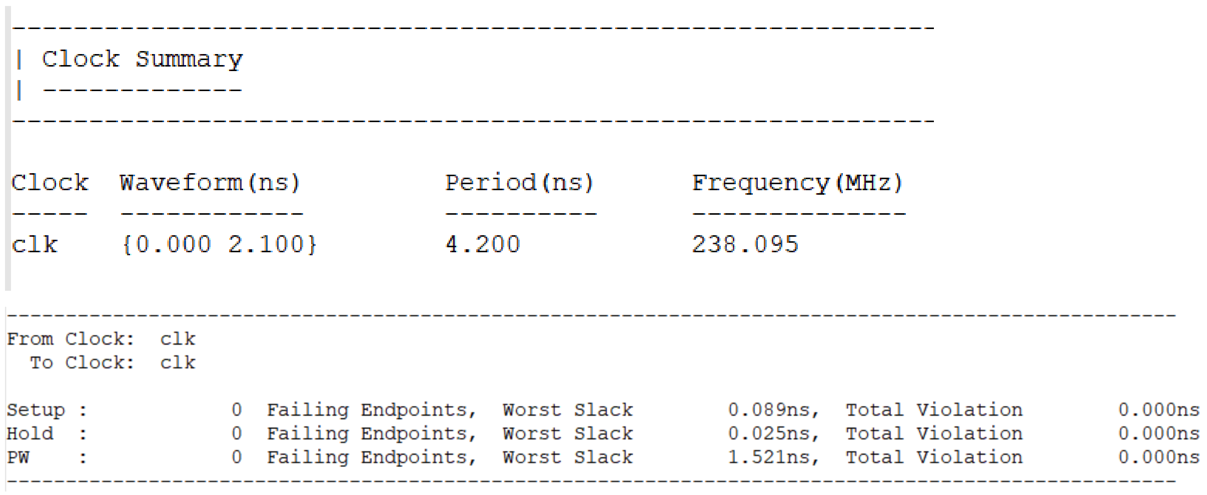


Figure 4.3 FPGA Implementation view of RFI Filter

4.3.2 Timing details

As shown in Figure 4.4, the Filter design works on 238.095 MHz frequency without any STA violation.



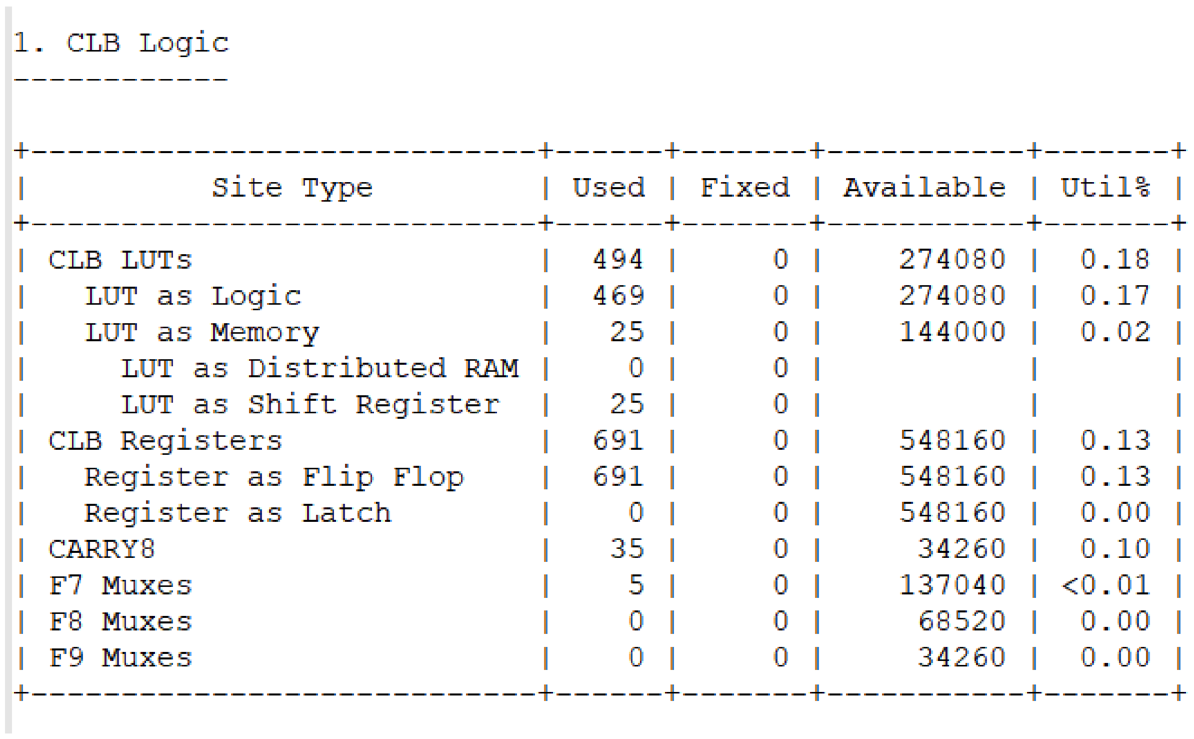
```
------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------

Clock  Waveform(ns)         Period(ns)      Frequency(MHz)
-----  ------------         ----------      --------------
clk    {0.000 2.100}        4.200           238.095
```

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

Setup :            0  Failing Endpoints,  Worst Slack        0.089ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.025ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack        1.521ns,  Total Violation        0.000ns
----------------------------------------------------------------------------------------------------
```

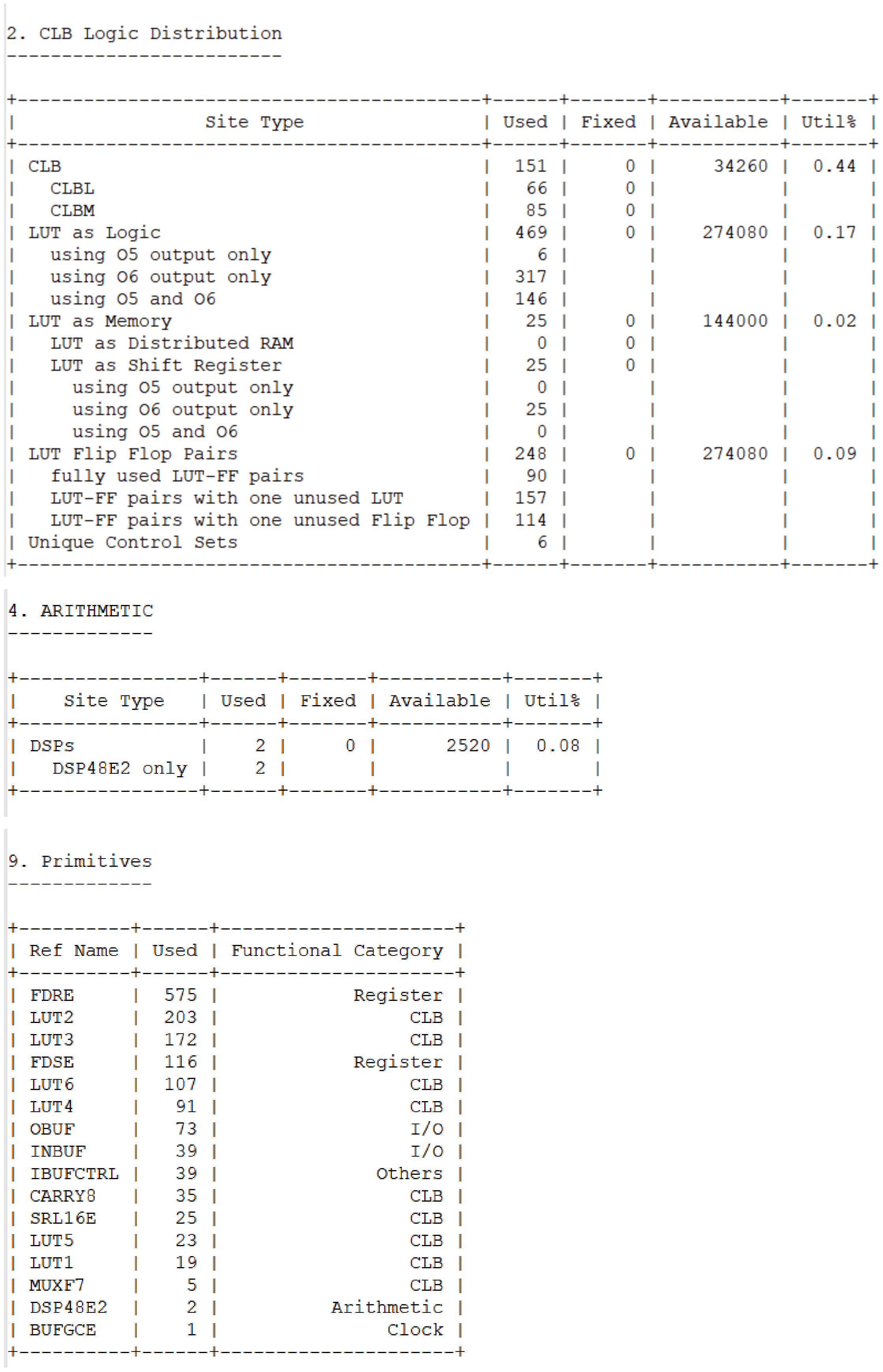
Figure 4.4 Timing report of FPGA implementation of Filter

4.3.3 Utilization details

Figure 4.5 gives an overall idea about resource utilization. Also, we can say that arithmetic inside the threshold calculation block infers the DSP block of FPGA, and there is a no register as a latch.



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

+----------------------------+------+-------+-----------+-------+
|          Site Type         | Used | Fixed | Available | Util% |
+----------------------------+------+-------+-----------+-------+
| CLB LUTs                   |  494 |     0 |    274080 |  0.18 |
|   LUT as Logic             |  469 |     0 |    274080 |  0.17 |
|   LUT as Memory            |   25 |     0 |    144000 |  0.02 |
|     LUT as Distributed RAM |    0 |     0 |           |       |
|     LUT as Shift Register  |   25 |     0 |           |       |
| CLB Registers              |  691 |     0 |    548160 |  0.13 |
|   Register as Flip Flop    |  691 |     0 |    548160 |  0.13 |
|   Register as Latch        |    0 |     0 |    548160 |  0.00 |
| CARRY8                     |   35 |     0 |     34260 |  0.10 |
| F7 Muxes                   |    5 |     0 |    137040 | <0.01 |
| F8 Muxes                   |    0 |     0 |     68520 |  0.00 |
| F9 Muxes                   |    0 |     0 |     34260 |  0.00 |
+----------------------------+------+-------+-----------+-------+
```



```
2. CLB Logic Distribution
-------------------------

+-------------------------------------------+------+-------+-----------+-------+
|                 Site Type                 | Used | Fixed | Available | Util% |
+-------------------------------------------+------+-------+-----------+-------+
| CLB                                       |  151 |     0 |     34260 |  0.44 |
|   CLBL                                    |   66 |     0 |           |       |
|   CLBM                                    |   85 |     0 |           |       |
| LUT as Logic                              |  469 |     0 |    274080 |  0.17 |
|   using O5 output only                    |    6 |       |           |       |
|   using O6 output only                    |  317 |       |           |       |
|   using O5 and O6                         |  146 |       |           |       |
| LUT as Memory                             |   25 |     0 |    144000 |  0.02 |
|   LUT as Distributed RAM                  |    0 |     0 |           |       |
|   LUT as Shift Register                   |   25 |     0 |           |       |
|     using O5 output only                  |    0 |       |           |       |
|     using O6 output only                  |   25 |       |           |       |
|     using O5 and O6                       |    0 |       |           |       |
| LUT Flip Flop Pairs                       |  248 |     0 |    274080 |  0.09 |
|   fully used LUT-FF pairs                 |   90 |       |           |       |
|   LUT-FF pairs with one unused LUT        |  157 |       |           |       |
|   LUT-FF pairs with one unused Flip Flop  |  114 |       |           |       |
| Unique Control Sets                       |    6 |       |           |       |
+-------------------------------------------+------+-------+-----------+-------+

4. ARITHMETIC
-------------

+----------------+------+-------+-----------+-------+
|    Site Type   | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
| DSPs           |    2 |     0 |      2520 |  0.08 |
|   DSP48E2 only |    2 |       |           |       |
+----------------+------+-------+-----------+-------+


9. Primitives
-------------

+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+
| FDRE     |  575 |            Register |
| LUT2     |  203 |                 CLB |
| LUT3     |  172 |                 CLB |
| FDSE     |  116 |            Register |
| LUT6     |  107 |                 CLB |
| LUT4     |   91 |                 CLB |
| OBUF     |   73 |                 I/O |
| INBUF    |   39 |                 I/O |
| IBUFCTRL |   39 |              Others |
| CARRY8   |   35 |                 CLB |
| SRL16E   |   25 |                 CLB |
| LUT5     |   23 |                 CLB |
| LUT1     |   19 |                 CLB |
| MUXF7    |    5 |                 CLB |
| DSP48E2  |    2 |          Arithmetic |
| BUFGCE   |    1 |               Clock |
+----------+------+---------------------+
```

Figure 4.5 Utilization report of FPGA implementation of RFI Filter

Chapter 05: Integration (MAD and MoM)

5.1 Algorithm

Figures 5.1 and 5.2 show that Median, MAD, or MoM computation blocks are integrated with the Filter block. Further, for storing the data block, RAM is being used. Up counter is used for providing the address for block RAM. In MAD-based filtering, data samples are buffered for two window cycles using two BRAMs, whereas, in MoM-based filtering, one BRAM is used for storing the data and another for MAD values.

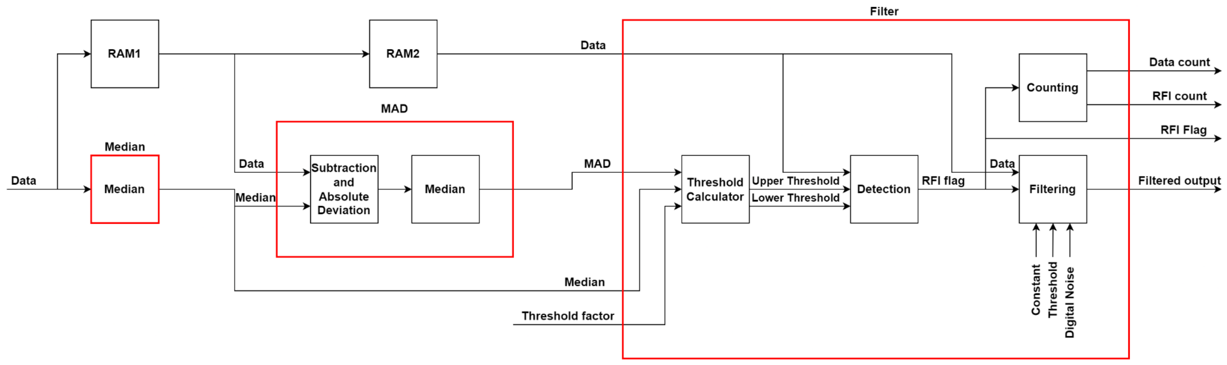


Figure 5.1 Block diagram of MAD-based Filtering implementation

For MoM-based filtering, switching is required in the last MoM cycle to calculate the median of the MAD values. As discussed in 3.1, Control logic for switching is required. The multiplexer is used for selecting the proper data path, and select-lines for that are provided from control logic. The counter is used for counting the window cycles, and a high signal is asserted to select lines of two multiplexers.

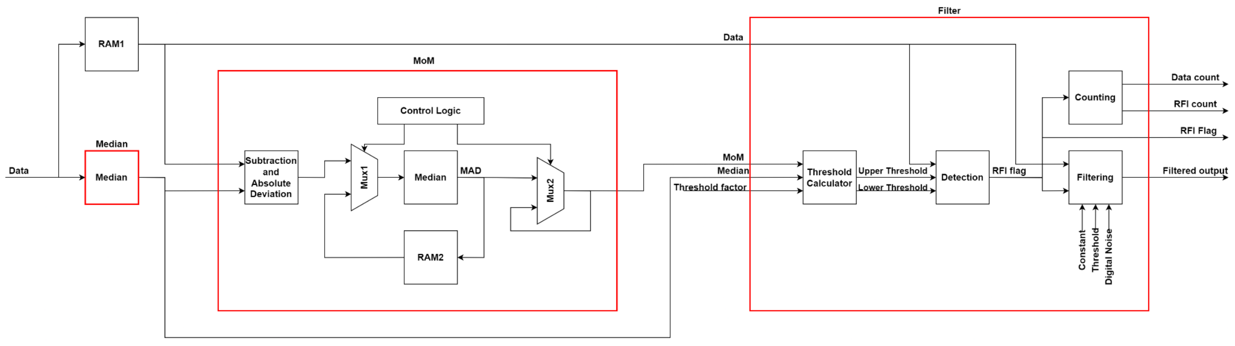


Figure 5.2 Block diagram of MoM-based Filtering implementation

5.2 Implementation details

In table 5.1, the I/O interface of the top entity is described for both MAD and MoM-based Filter. As two median is computed inside the MAD computation, there is 2W window cycle initial latency. Also to buffer the data for 2W window cycle, buffer is used and is implemented using BRAM blocks inside the FPGA.

Table 5.1 VHDL interface of MAD and MoM based RFI Filter

Signal	Description
clk	Input clock
rst	Active high, Synchronous reset
hold	Control signal for enable 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 counter block of median which generates reset signal for 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 counter block of mad which generates reset signal for main accumulator and auxiliary accumulator
rst_filter	Active high, Synchronous reset for filter block
select_line [1 downto 0]	Replacement mode control signal
n [7 downto 0]	Input Threshold factor
data_in [7 downto 0]	Input data
data_out [7 downto 0]	Filtered output
total_count [31 downto 0]	Output data count value
RFI_count [31 downto 0]	Output RFI count value
flag_out	Output RFI flag

5.3 Synthesis and Implementation

In this section, Synthesis and Implementation details are mentioned. Section 5.3.1 contains the FPGA Implementation view of the MAD-based and MoM-based Filter. Moreover, the Timing and Area-Utilizations details of Implemented designs are discussed in 5.3.2 and 5.3.3, respectively.

5.3.1 Implementation View

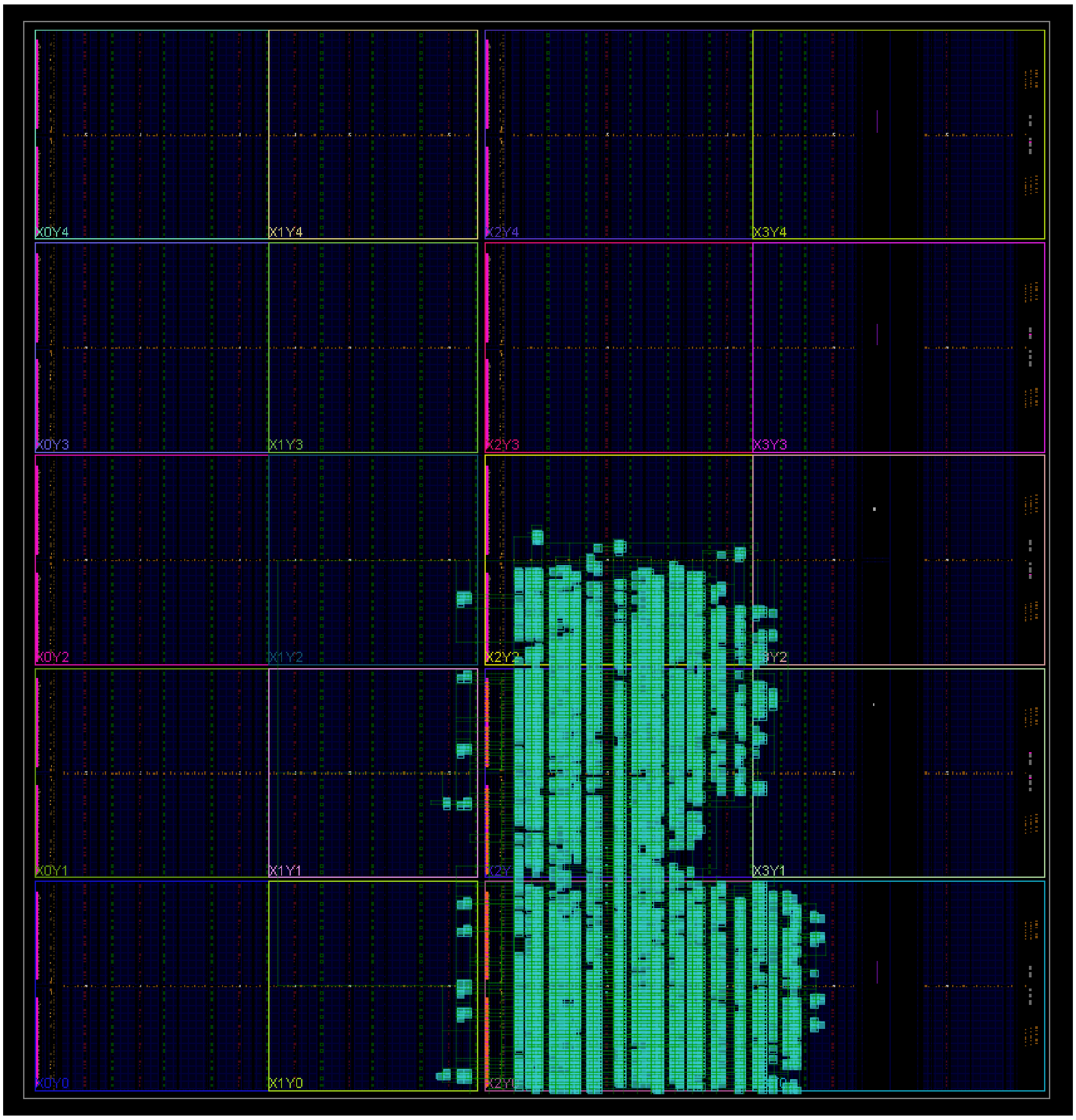


Figure 5.3 FPGA Implementation view of MAD-based RFI Filter

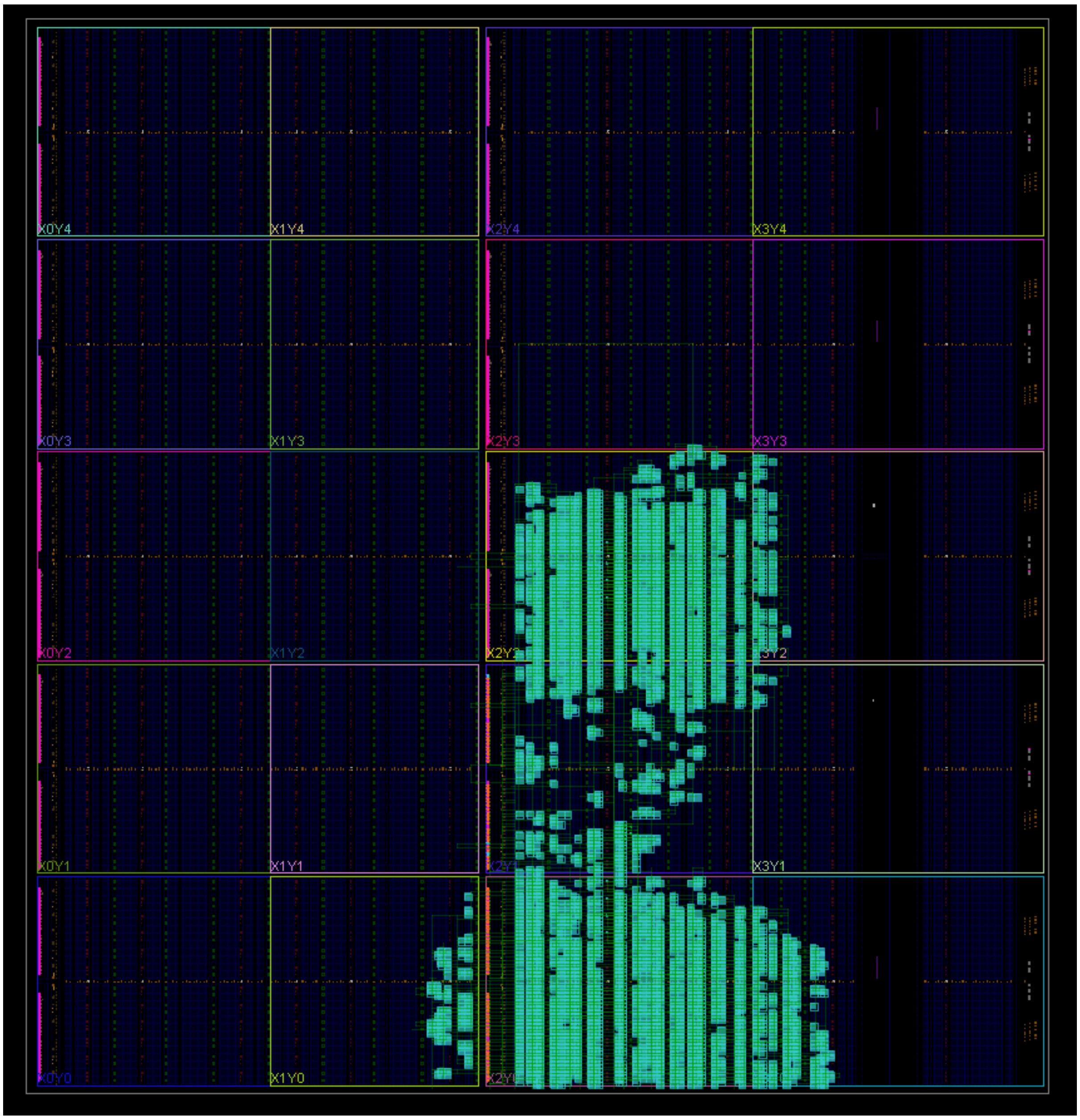
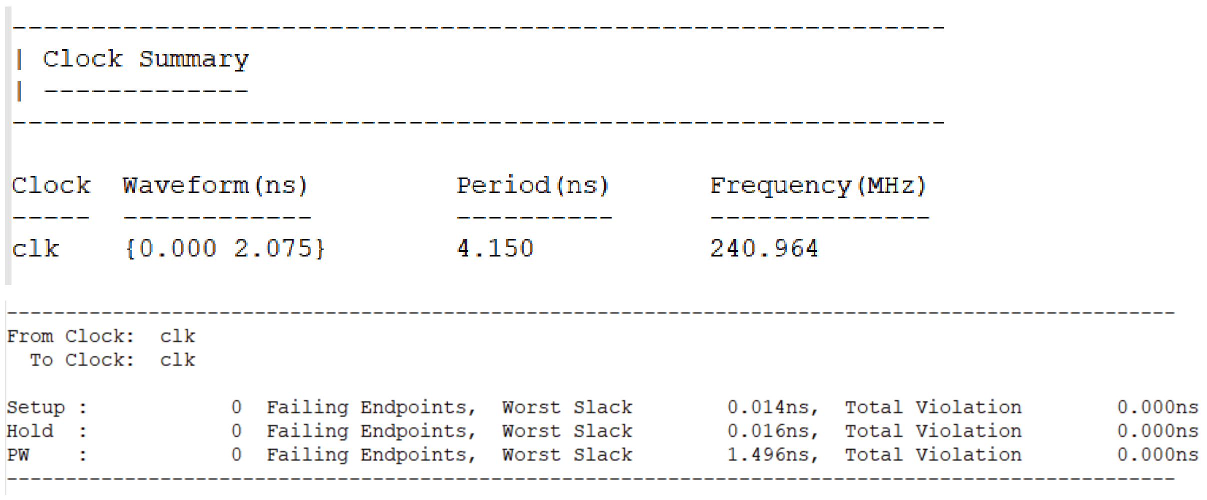


Figure 5.4 FPGA Implementation view of MoM-based RFI Filter

5.3.2 Timing details

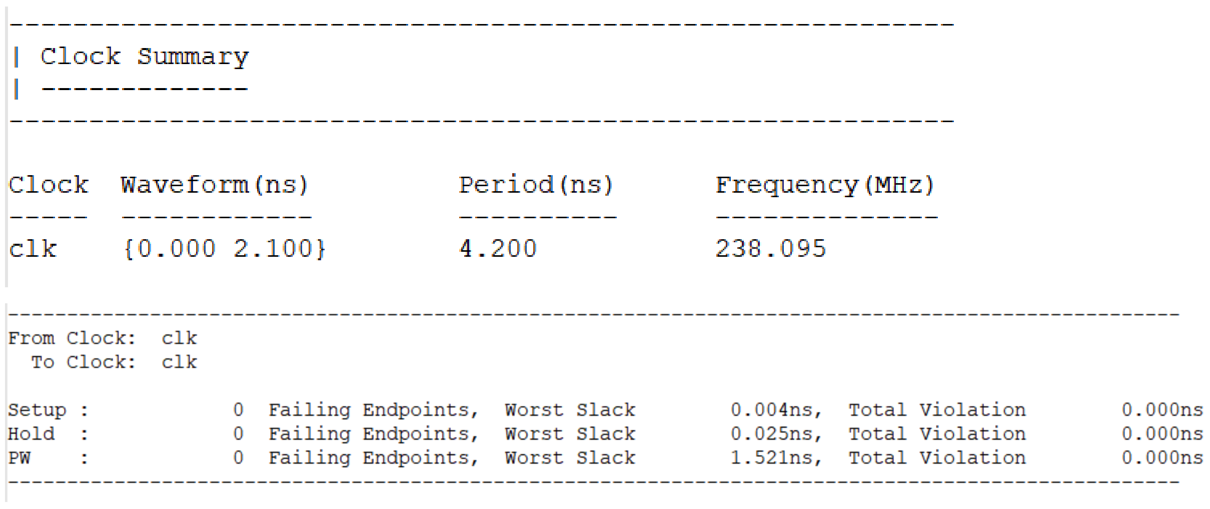
Figures 5.5 and 5.6 show that the MAD-based Filter design works on 240.964 MHz frequency and the MoM-based Filter design works on 238.095 MHz frequency without any STA violation on 16384 window size. Moreover, a comparison between both designs on different window sizes is shown in Table 5.2. From the table, we can see that delay is increasing by increasing the window size.



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

Clock  Waveform(ns)         Period(ns)      Frequency(MHz)
-----  ------------         ----------      --------------
clk    {0.000 2.075}        4.150           240.964
```

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

Setup :            0  Failing Endpoints,  Worst Slack        0.014ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.016ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack        1.496ns,  Total Violation        0.000ns
------------------------------------------------------------------------------------------------------
```

Figure 5.5 Timing report of FPGA implementation of MAD-based RFI Filter of 16k window



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

Clock  Waveform(ns)         Period(ns)      Frequency(MHz)
-----  ------------         ----------      --------------
clk    {0.000 2.100}        4.200           238.095
```

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

Setup :            0  Failing Endpoints,  Worst Slack        0.004ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.025ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack        1.521ns,  Total Violation        0.000ns
------------------------------------------------------------------------------------------------------
```

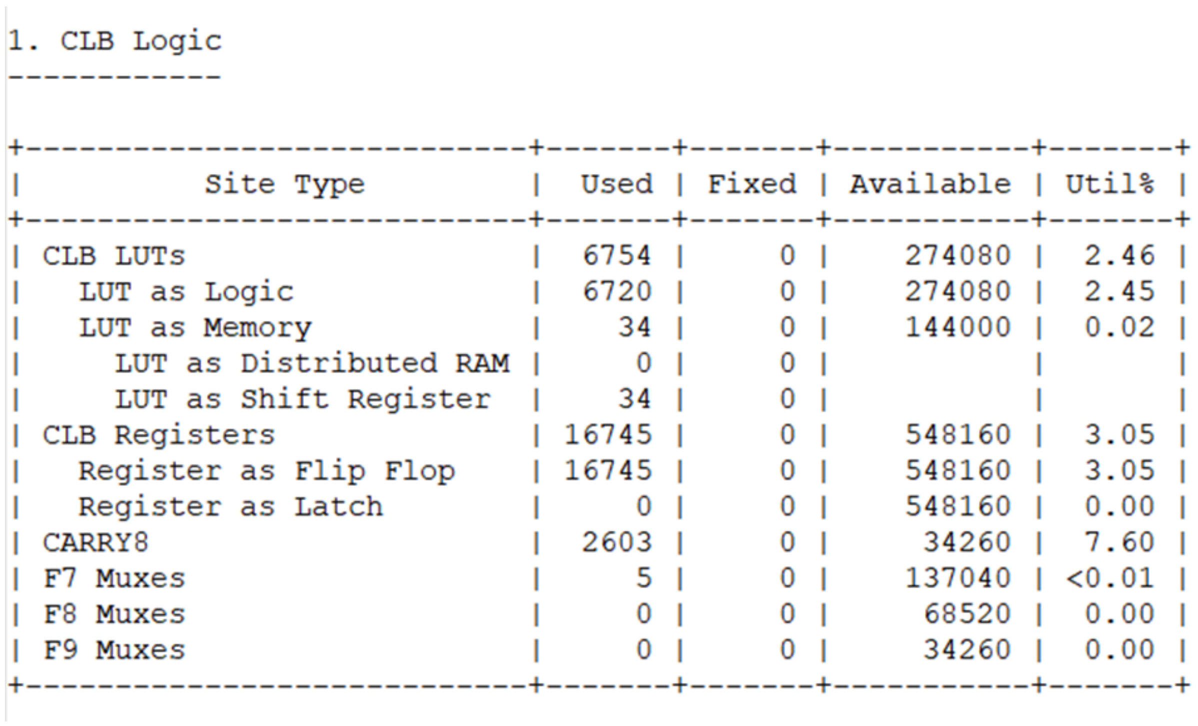
Figure 5.6 Timing report of FPGA implementation of MoM-based RFI Filter of 16k window

Table 5.2 Frequency comparison of MAD and MoM based Filter on different window size

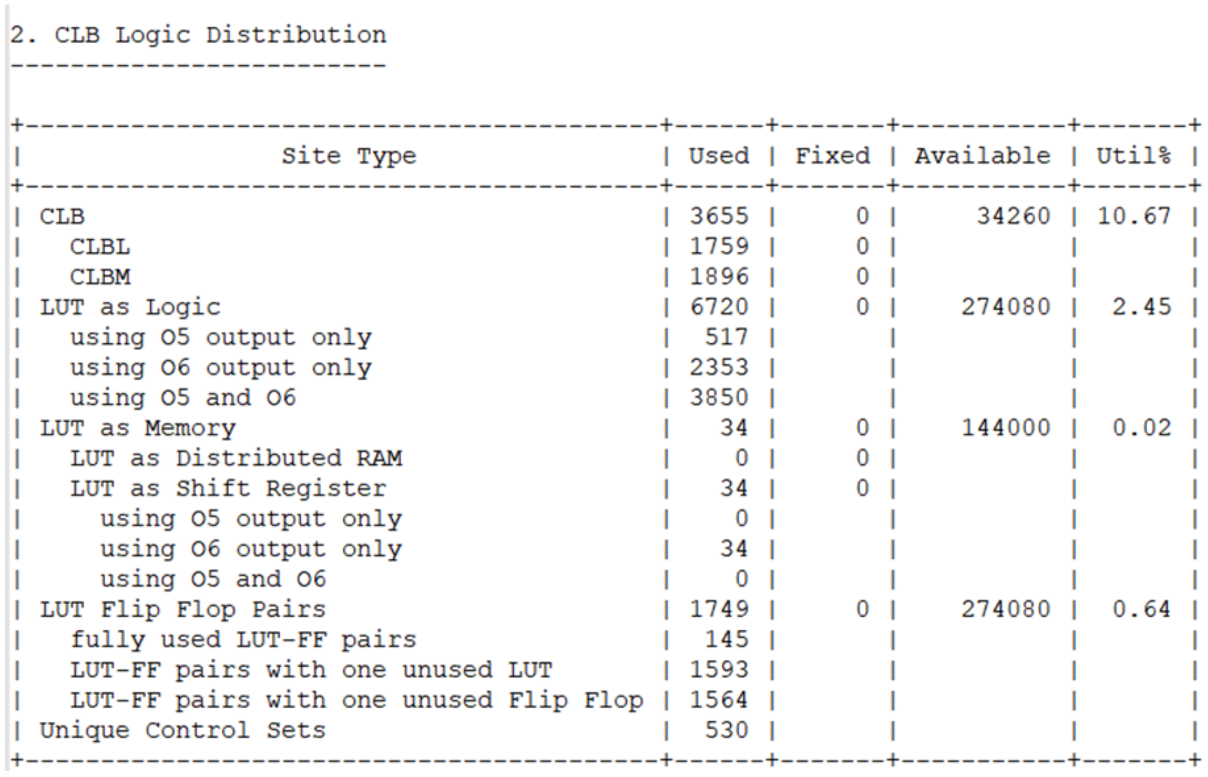
Window Size	Maximum Operating Frequency (MHz)	
	MAD Filter	MoM Filter
1024	246.914	250
2048	246.914	240.964
4096	246.914	238.095
8192	243.902	239.234
16384	240.964	238.095

5.3.3 Utilization details

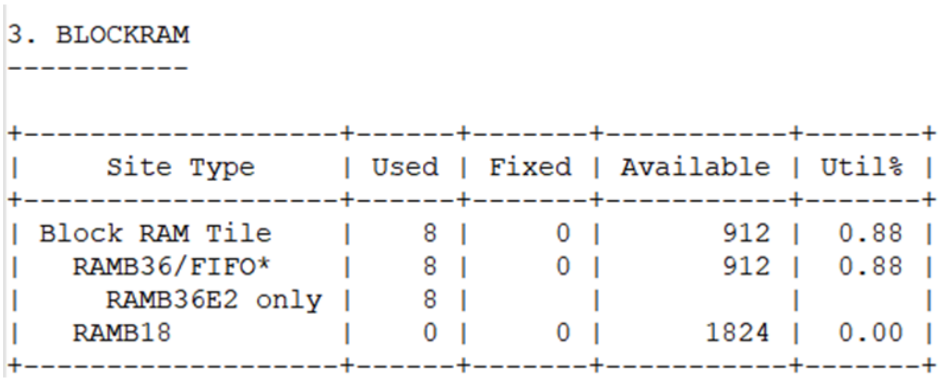
Figure 4.4 gives an overall idea about resource utilization. Also, we can say that arithmetic operations inside the threshold calculation block infer the DSP block of the FPGA, and there is no register as a latch.



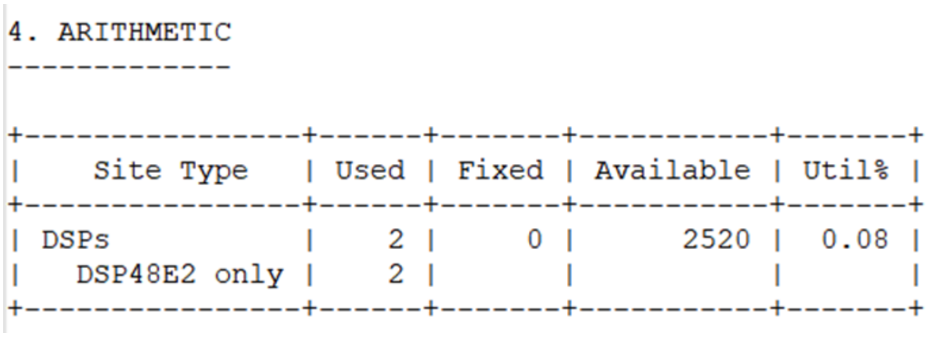
```
1. CLB Logic
------------

+----------------------------+-------+-------+-----------+-------+
|          Site Type         |  Used | Fixed | Available | Util% |
+----------------------------+-------+-------+-----------+-------+
| CLB LUTs                   |  6754 |     0 |    274080 |  2.46 |
|   LUT as Logic             |  6720 |     0 |    274080 |  2.45 |
|   LUT as Memory            |    34 |     0 |    144000 |  0.02 |
|     LUT as Distributed RAM |     0 |     0 |           |       |
|     LUT as Shift Register  |    34 |     0 |           |       |
| CLB Registers              | 16745 |     0 |    548160 |  3.05 |
|   Register as Flip Flop    | 16745 |     0 |    548160 |  3.05 |
|   Register as Latch        |     0 |     0 |    548160 |  0.00 |
| CARRY8                     |  2603 |     0 |     34260 |  7.60 |
| F7 Muxes                   |     5 |     0 |    137040 | <0.01 |
| F8 Muxes                   |     0 |     0 |     68520 |  0.00 |
| F9 Muxes                   |     0 |     0 |     34260 |  0.00 |
+----------------------------+-------+-------+-----------+-------+
```



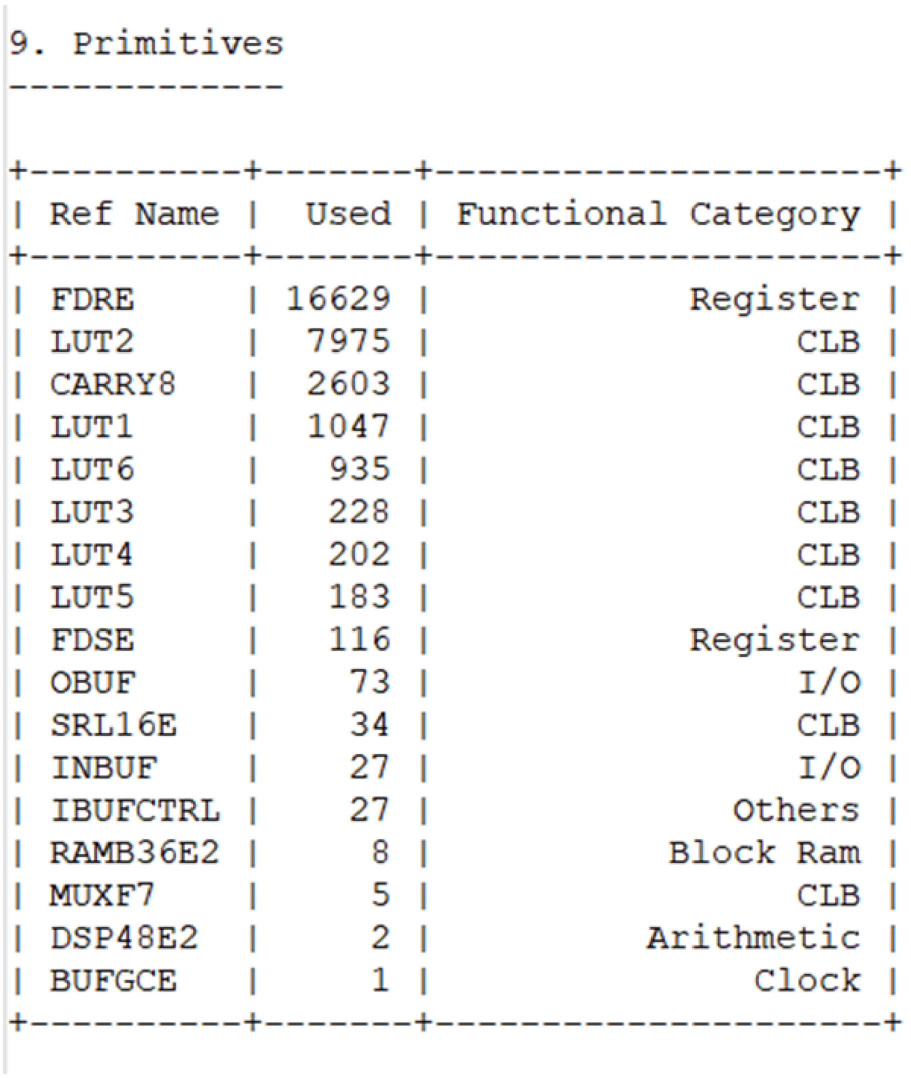
```
2. CLB Logic Distribution
-------------------------

+--------------------------------------------+------+-------+-----------+-------+
|                  Site Type                 | Used | Fixed | Available | Util% |
+--------------------------------------------+------+-------+-----------+-------+
| CLB                                        | 3655 |     0 |     34260 | 10.67 |
|   CLBL                                     | 1759 |     0 |           |       |
|   CLBM                                     | 1896 |     0 |           |       |
| LUT as Logic                               | 6720 |     0 |    274080 |  2.45 |
|   using O5 output only                     |  517 |       |           |       |
|   using O6 output only                     | 2353 |       |           |       |
|   using O5 and O6                          | 3850 |       |           |       |
| LUT as Memory                              |   34 |     0 |    144000 |  0.02 |
|   LUT as Distributed RAM                   |    0 |     0 |           |       |
|   LUT as Shift Register                    |   34 |     0 |           |       |
|     using O5 output only                   |    0 |       |           |       |
|     using O6 output only                   |   34 |       |           |       |
|     using O5 and O6                        |    0 |       |           |       |
| LUT Flip Flop Pairs                        | 1749 |     0 |    274080 |  0.64 |
|   fully used LUT-FF pairs                  |  145 |       |           |       |
|   LUT-FF pairs with one unused LUT         | 1593 |       |           |       |
|   LUT-FF pairs with one unused Flip Flop   | 1564 |       |           |       |
| Unique Control Sets                        |  530 |       |           |       |
+--------------------------------------------+------+-------+-----------+-------+
```



```
3. BLOCKRAM
-----------

+-------------------+------+-------+-----------+-------+
|     Site Type     | Used | Fixed | Available | Util% |
+-------------------+------+-------+-----------+-------+
| Block RAM Tile    |    8 |     0 |       912 |  0.88 |
|   RAMB36/FIFO*    |    8 |     0 |       912 |  0.88 |
|     RAMB36E2 only |    8 |       |           |       |
|   RAMB18          |    0 |     0 |      1824 |  0.00 |
+-------------------+------+-------+-----------+-------+
```

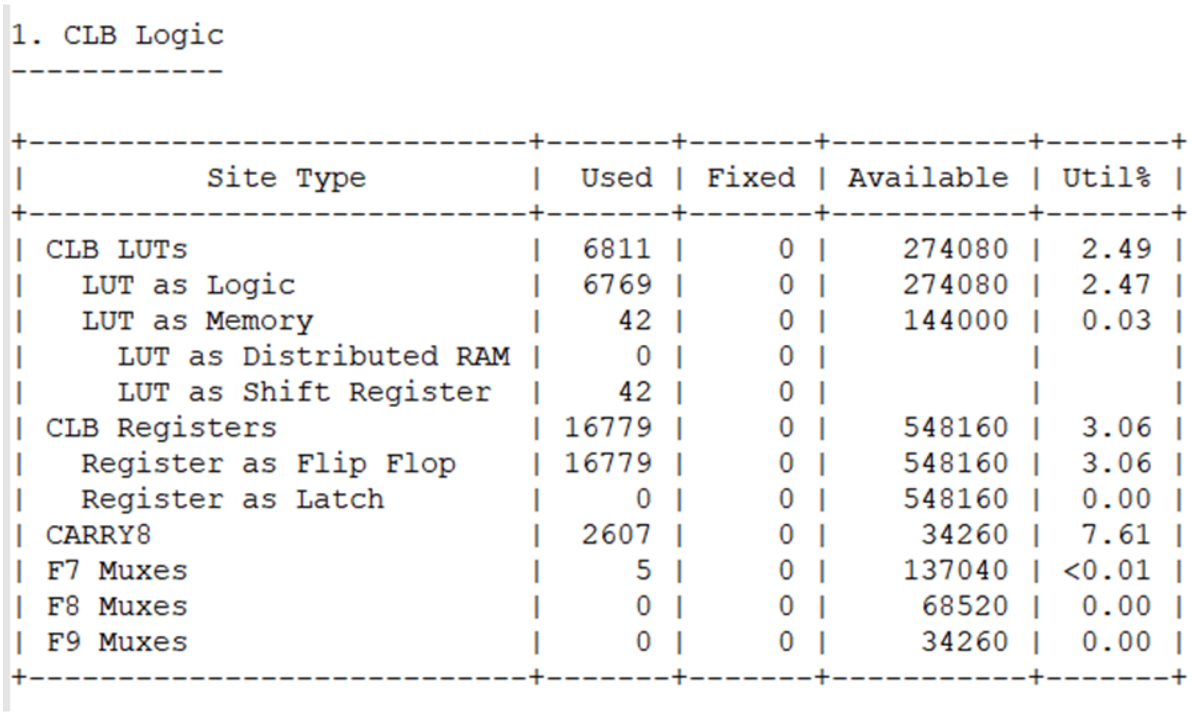


```
4. ARITHMETIC
-------------

+----------------+------+-------+-----------+-------+
|    Site Type   | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
| DSPs           |    2 |     0 |      2520 |  0.08 |
|   DSP48E2 only |    2 |       |           |       |
+----------------+------+-------+-----------+-------+
```

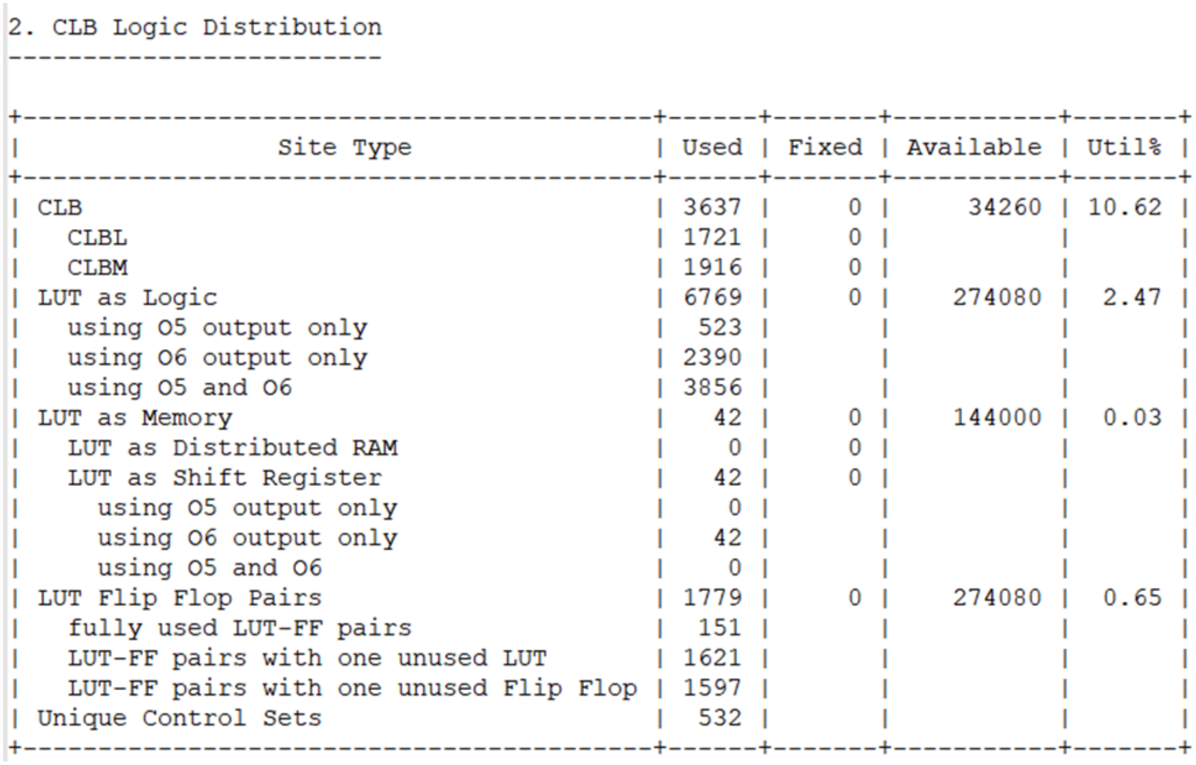


```
9. Primitives
-------------

+----------+-------+---------------------+
| Ref Name |  Used | Functional Category |
+----------+-------+---------------------+
| FDRE     | 16629 |            Register |
| LUT2     |  7975 |                 CLB |
| CARRY8   |  2603 |                 CLB |
| LUT1     |  1047 |                 CLB |
| LUT6     |   935 |                 CLB |
| LUT3     |   228 |                 CLB |
| LUT4     |   202 |                 CLB |
| LUT5     |   183 |                 CLB |
| FDSE     |   116 |            Register |
| OBUF     |    73 |                 I/O |
| SRL16E   |    34 |                 CLB |
| INBUF    |    27 |                 I/O |
| IBUFCTRL |    27 |              Others |
| RAMB36E2 |     8 |           Block Ram |
| MUXF7    |     5 |                 CLB |
| DSP48E2  |     2 |          Arithmetic |
| BUFGCE   |     1 |               Clock |
+----------+-------+---------------------+
```

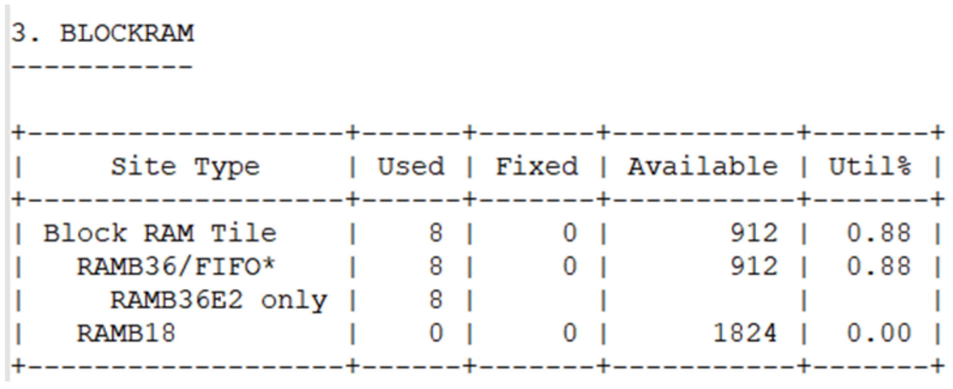
Figure 5.7 Utilization report of FPGA implementation of MAD-based RFI Filter



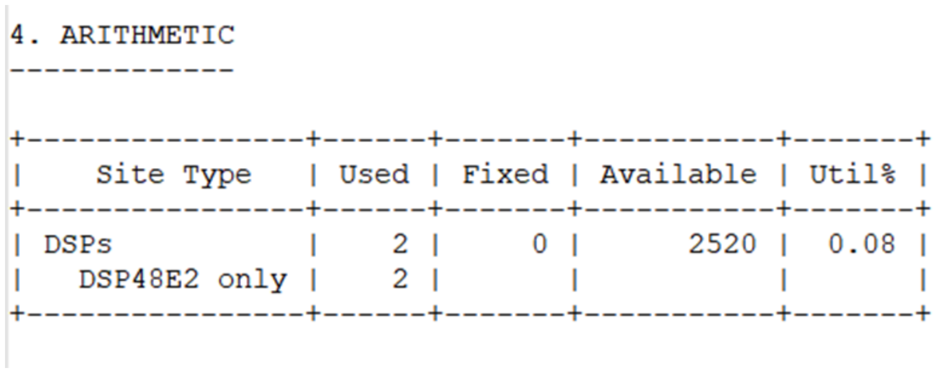
```
1. CLB Logic
------------

+----------------------------+-------+-------+-----------+-------+
|          Site Type         |  Used | Fixed | Available | Util% |
+----------------------------+-------+-------+-----------+-------+
| CLB LUTs                   |  6811 |     0 |    274080 |  2.49 |
|   LUT as Logic             |  6769 |     0 |    274080 |  2.47 |
|   LUT as Memory            |    42 |     0 |    144000 |  0.03 |
|     LUT as Distributed RAM |     0 |     0 |           |       |
|     LUT as Shift Register  |    42 |     0 |           |       |
| CLB Registers              | 16779 |     0 |    548160 |  3.06 |
|   Register as Flip Flop    | 16779 |     0 |    548160 |  3.06 |
|   Register as Latch        |     0 |     0 |    548160 |  0.00 |
| CARRY8                     |  2607 |     0 |     34260 |  7.61 |
| F7 Muxes                   |     5 |     0 |    137040 | <0.01 |
| F8 Muxes                   |     0 |     0 |     68520 |  0.00 |
| F9 Muxes                   |     0 |     0 |     34260 |  0.00 |
+----------------------------+-------+-------+-----------+-------+
```



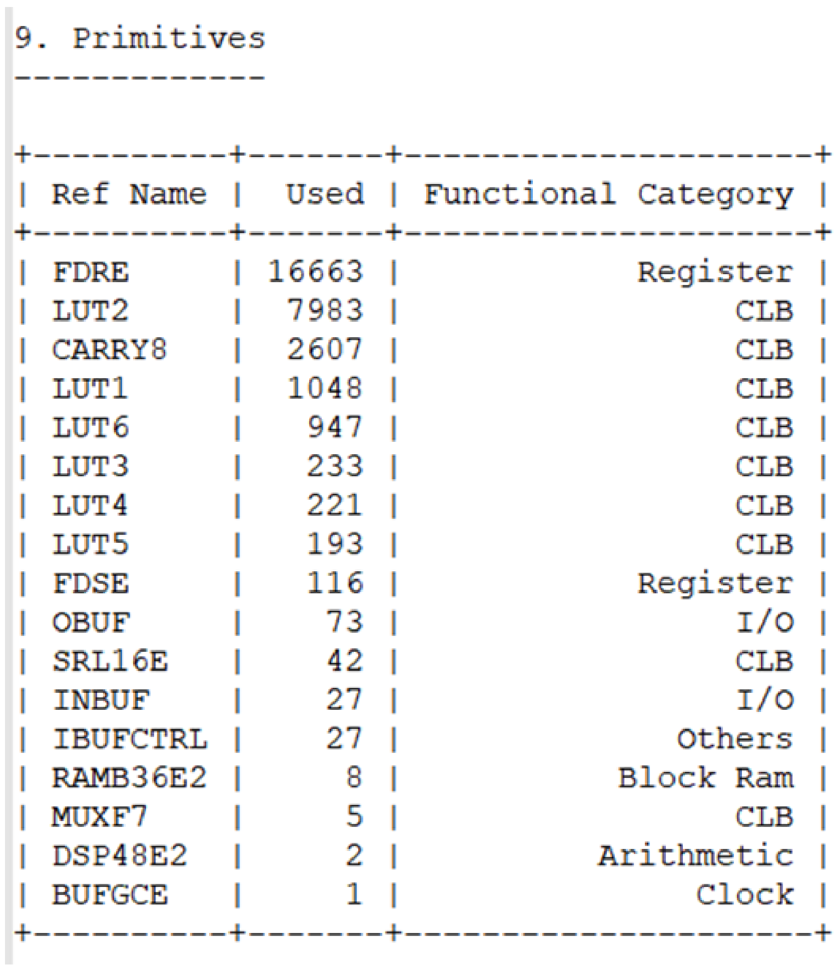
```
2. CLB Logic Distribution
-------------------------

+-------------------------------------------+------+-------+-----------+-------+
|                 Site Type                 | Used | Fixed | Available | Util% |
+-------------------------------------------+------+-------+-----------+-------+
| CLB                                       | 3637 |     0 |     34260 | 10.62 |
|   CLBL                                    | 1721 |     0 |           |       |
|   CLBM                                    | 1916 |     0 |           |       |
| LUT as Logic                              | 6769 |     0 |    274080 |  2.47 |
|   using O5 output only                    |  523 |       |           |       |
|   using O6 output only                    | 2390 |       |           |       |
|   using O5 and O6                         | 3856 |       |           |       |
| LUT as Memory                             |   42 |     0 |    144000 |  0.03 |
|   LUT as Distributed RAM                  |    0 |     0 |           |       |
|   LUT as Shift Register                   |   42 |     0 |           |       |
|     using O5 output only                  |    0 |       |           |       |
|     using O6 output only                  |   42 |       |           |       |
|     using O5 and O6                       |    0 |       |           |       |
| LUT Flip Flop Pairs                       | 1779 |     0 |    274080 |  0.65 |
|   fully used LUT-FF pairs                 |  151 |       |           |       |
|   LUT-FF pairs with one unused LUT        | 1621 |       |           |       |
|   LUT-FF pairs with one unused Flip Flop  | 1597 |       |           |       |
| Unique Control Sets                       |  532 |       |           |       |
+-------------------------------------------+------+-------+-----------+-------+
```



```
3. BLOCKRAM
-----------

+-------------------+------+-------+-----------+-------+
|     Site Type     | Used | Fixed | Available | Util% |
+-------------------+------+-------+-----------+-------+
| Block RAM Tile    |    8 |     0 |       912 |  0.88 |
|   RAMB36/FIFO*    |    8 |     0 |       912 |  0.88 |
|     RAMB36E2 only |    8 |       |           |       |
|   RAMB18          |    0 |     0 |      1824 |  0.00 |
+-------------------+------+-------+-----------+-------+
```



```
4. ARITHMETIC
-------------

+----------------+------+-------+-----------+-------+
|    Site Type   | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
| DSPs           |    2 |     0 |      2520 |  0.08 |
|   DSP48E2 only |    2 |       |           |       |
+----------------+------+-------+-----------+-------+
```



```
9. Primitives
-------------

+----------+-------+---------------------+
| Ref Name |  Used | Functional Category |
+----------+-------+---------------------+
| FDRE     | 16663 |            Register |
| LUT2     |  7983 |                 CLB |
| CARRY8   |  2607 |                 CLB |
| LUT1     |  1048 |                 CLB |
| LUT6     |   947 |                 CLB |
| LUT3     |   233 |                 CLB |
| LUT4     |   221 |                 CLB |
| LUT5     |   193 |                 CLB |
| FDSE     |   116 |            Register |
| OBUF     |    73 |                 I/O |
| SRL16E   |    42 |                 CLB |
| INBUF    |    27 |                 I/O |
| IBUFCTRL |    27 |              Others |
| RAMB36E2 |     8 |           Block Ram |
| MUXF7    |     5 |                 CLB |
| DSP48E2  |     2 |          Arithmetic |
| BUFGCE   |     1 |               Clock |
+----------+-------+---------------------+
```

Figure 5.8 Utilization report of FPGA implementation of MoM-based RFI Filter

5.4 Verification Environment

For all the designs, functional verification has been carried out. The Golden model for functional verification has been implemented in MATLAB.

For VHDL design, all the parameters, like threshold factor, replacement mode, window size, etc., are declared in the VHDL testbench. The testbench is capable of reading and writing text files. The same file generates stimulus using the input text file, i.e., raw voltage data for the DUT - Design Under Test block. The DUT block consists of the main design file. Output is generated by the DUT block using behavioral simulation, which is faded to testbench for converting the text file. The block diagram of the Verification Environment is shown in figure 5.9.

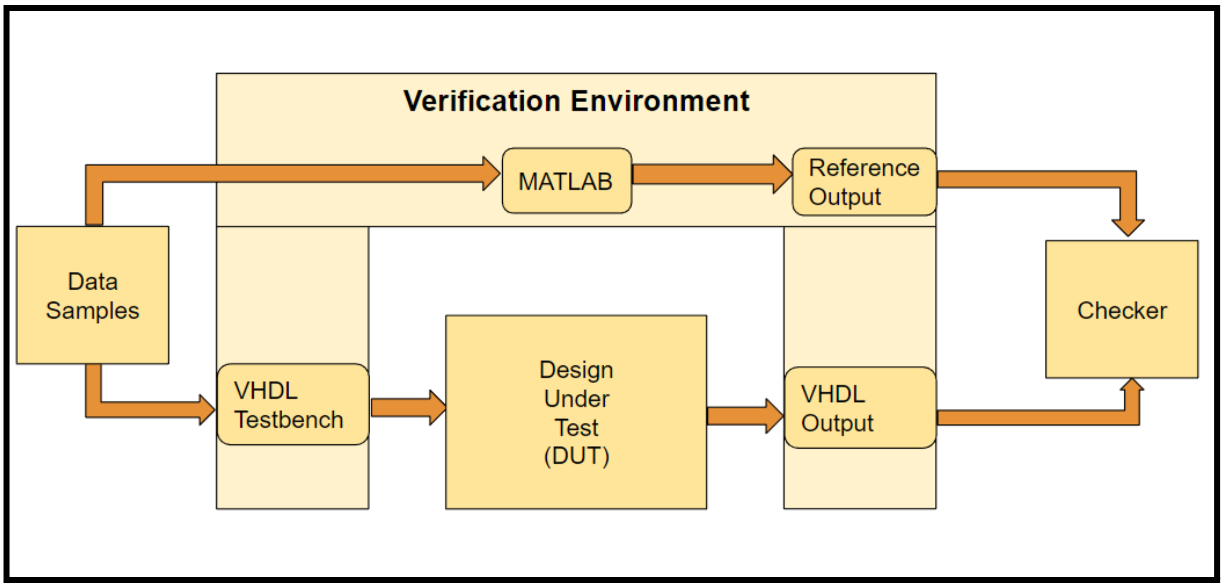
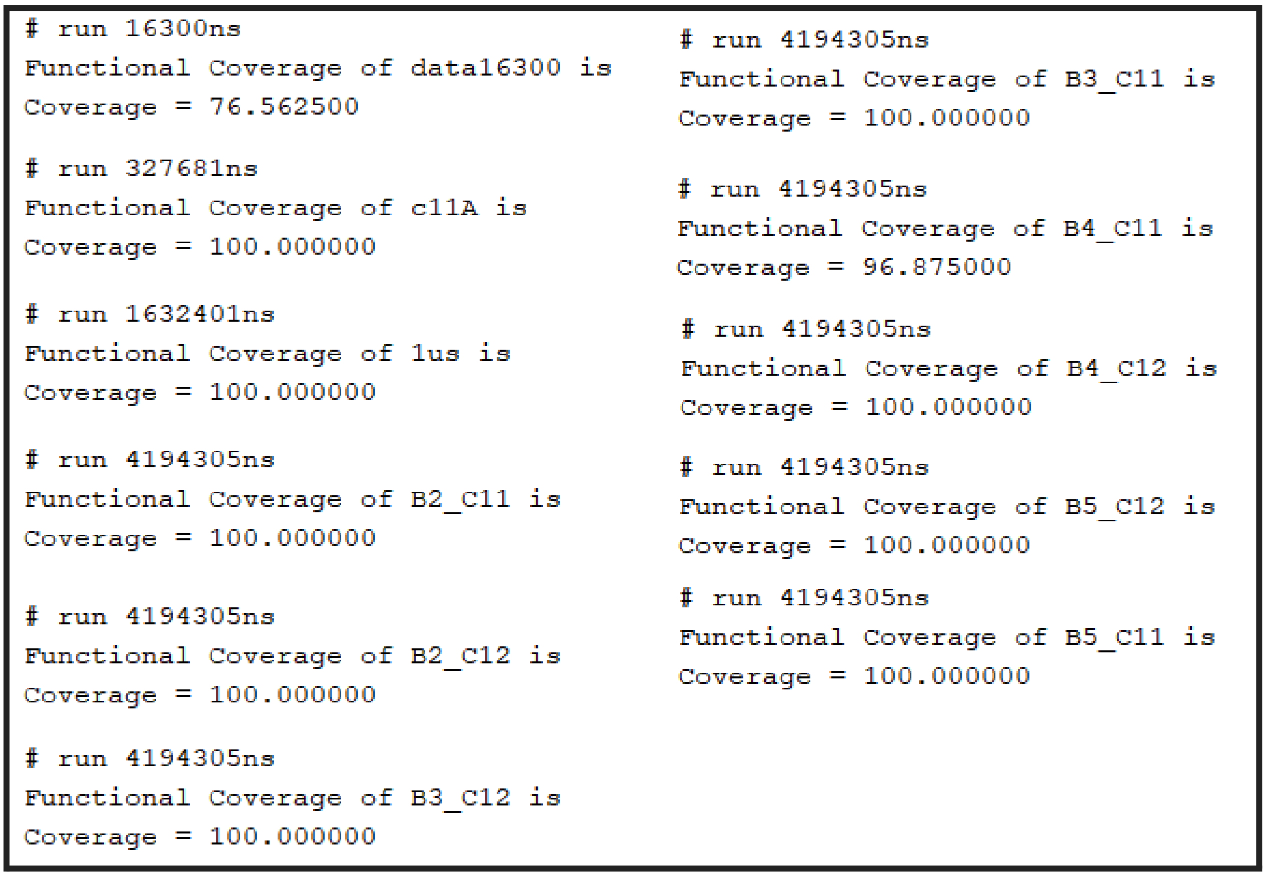


Figure 5.9 Verification Environment used for Functional Verification

The- diff command compares both files generated by the Golden reference model and DUT. There are some mismatches in comparison because MATLAB works on full precision values, whereas VHDL works only on integer values.

Further, Functional Coverage is carried out of all the RAW Voltage data using the System Verilog script. The result of Functional Coverage is shown in Figure 5.10.



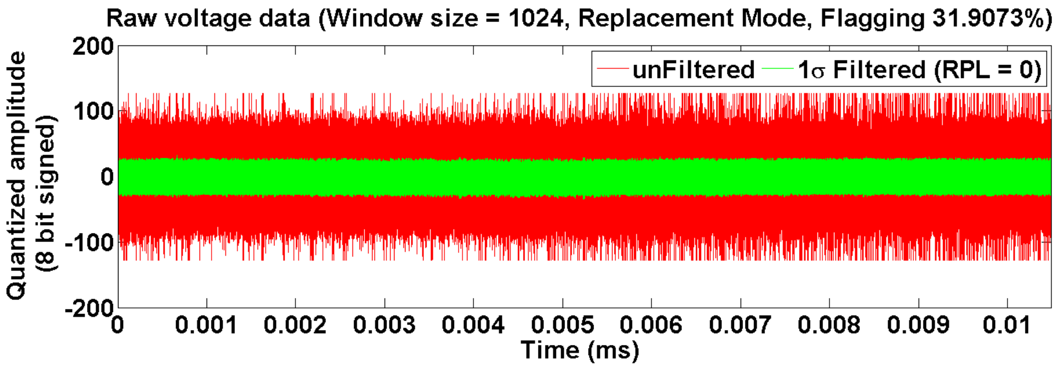
```
# run 16300ns
Functional Coverage of data16300 is
Coverage = 76.562500

# run 327681ns
Functional Coverage of c11A is
Coverage = 100.000000

# run 1632401ns
Functional Coverage of 1us is
Coverage = 100.000000

# run 4194305ns
Functional Coverage of B2_C11 is
Coverage = 100.000000

# run 4194305ns
Functional Coverage of B2_C12 is
Coverage = 100.000000

# run 4194305ns
Functional Coverage of B3_C12 is
Coverage = 100.000000

# run 4194305ns
Functional Coverage of B3_C11 is
Coverage = 100.000000

# run 4194305ns
Functional Coverage of B4_C11 is
Coverage = 96.875000

# run 4194305ns
Functional Coverage of B4_C12 is
Coverage = 100.000000

# run 4194305ns
Functional Coverage of B5_C12 is
Coverage = 100.000000

# run 4194305ns
Functional Coverage of B5_C11 is
Coverage = 100.000000
```

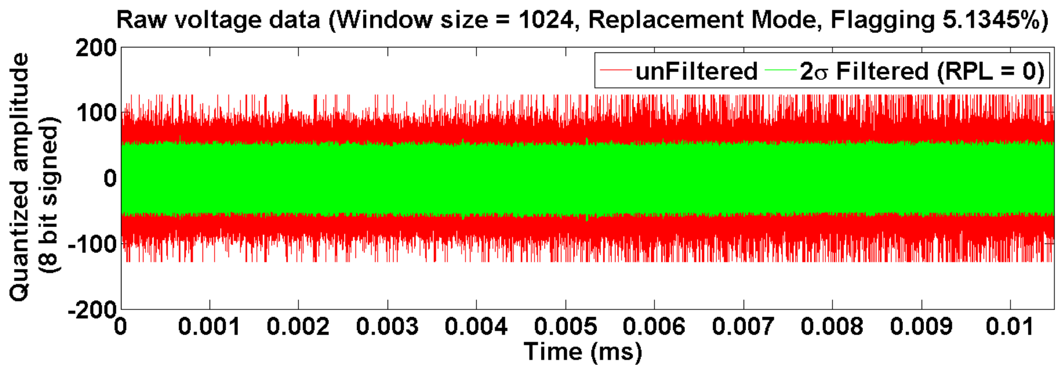
Figure 5.10 Functional Coverage result of GMRT RAW Voltage data samples

5.5 Results

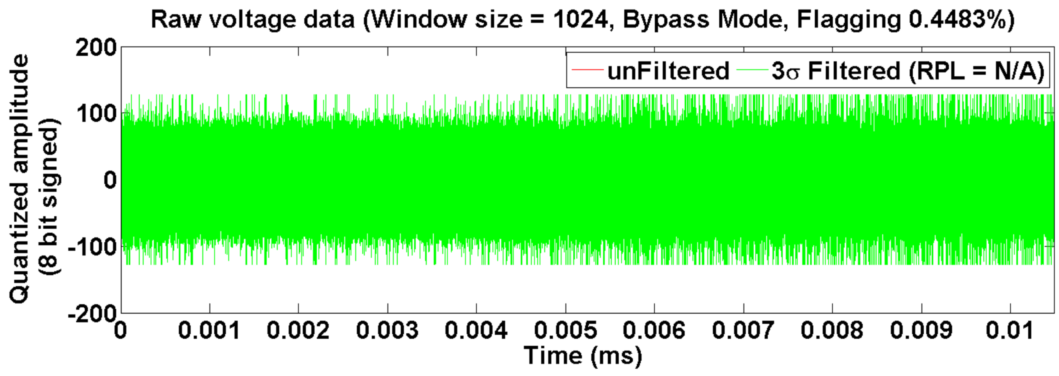
In the following section, the result of MAD-based filtering and MoM-based filtering is shown. A comparison between original data and output of VHDL designs has been carried out, and MATLAB overlay plots have been plotted. The unfiltered signal is shown in red, and its filtered output is shown in green. The result of 1k window size is shown with different threshold factors, i.e.,1,2,3 and 3σ filtering with all the replacement modes.

5.5.1 MAD based filtering plots

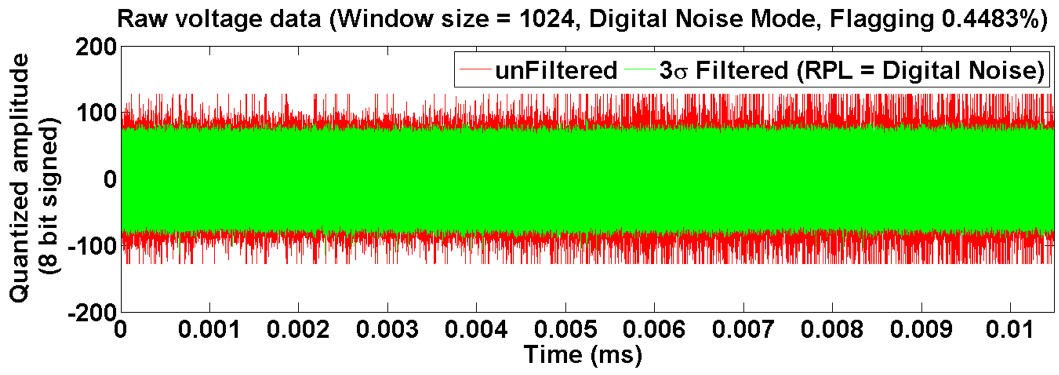




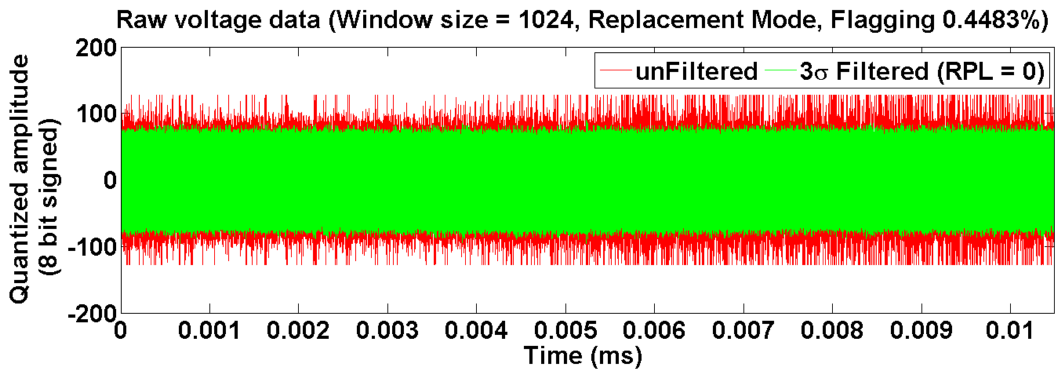
Raw voltage data (Window size = 1024, Replacement Mode, Flagging 5.1345%)
unFiltered
2σ Filtered (RPL = 0)
Quantized amplitude (8 bit signed)
200
100
0
-100
-200
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
Time (ms)



Raw voltage data (Window size = 1024, Bypass Mode, Flagging 0.4483%)
unFiltered
3σ Filtered (RPL = N/A)
Quantized amplitude (8 bit signed)
200
100
0
-100
-200
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
Time (ms)



Raw voltage data (Window size = 1024, Digital Noise Mode, Flagging 0.4483%)
unFiltered
3σ Filtered (RPL = Digital Noise)
Quantized amplitude (8 bit signed)
200
100
0
-100
-200
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
Time (ms)



Raw voltage data (Window size = 1024, Replacement Mode, Flagging 0.4483%)
unFiltered
3σ Filtered (RPL = 0)
Quantized amplitude (8 bit signed)
200
100
0
-100
-200
0
0.001
0.002
0.003
0.004
0.005
0.006
0.007
0.008
0.009
0.01
Time (ms)

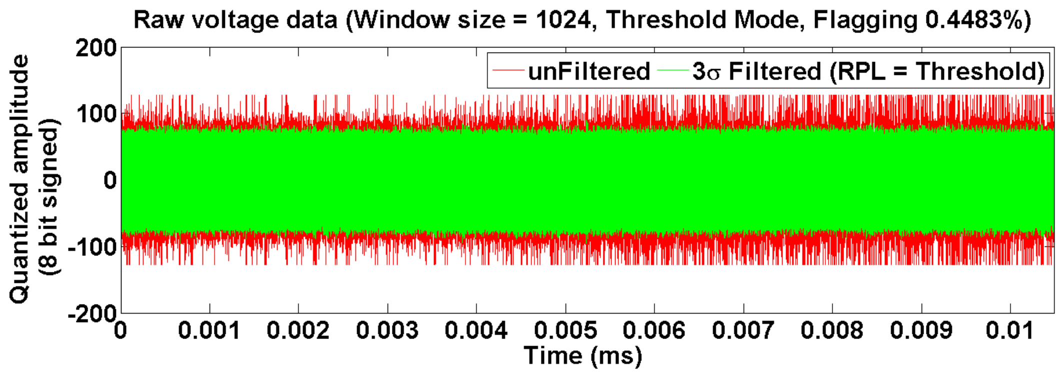
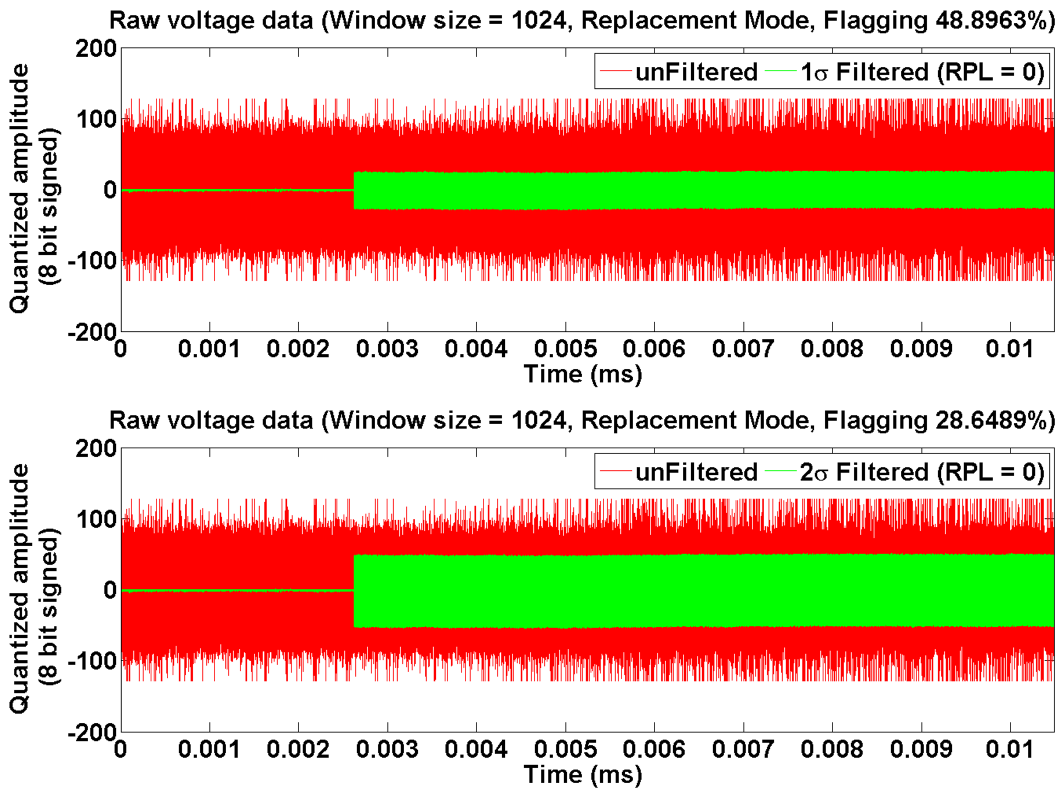


Figure 5.11 Filter output of MAD-based RFI Filter

5.5.2 MoM based filtering plots

The result of MoM-based RFI filter design is shown in figure 5.12. As discussed in section 3.1, the value of corresponding MoM value is applied in the following MoM cycle. So, initial latency has been found during the first MoM cycle. This fact is clearly seen in the below figure.



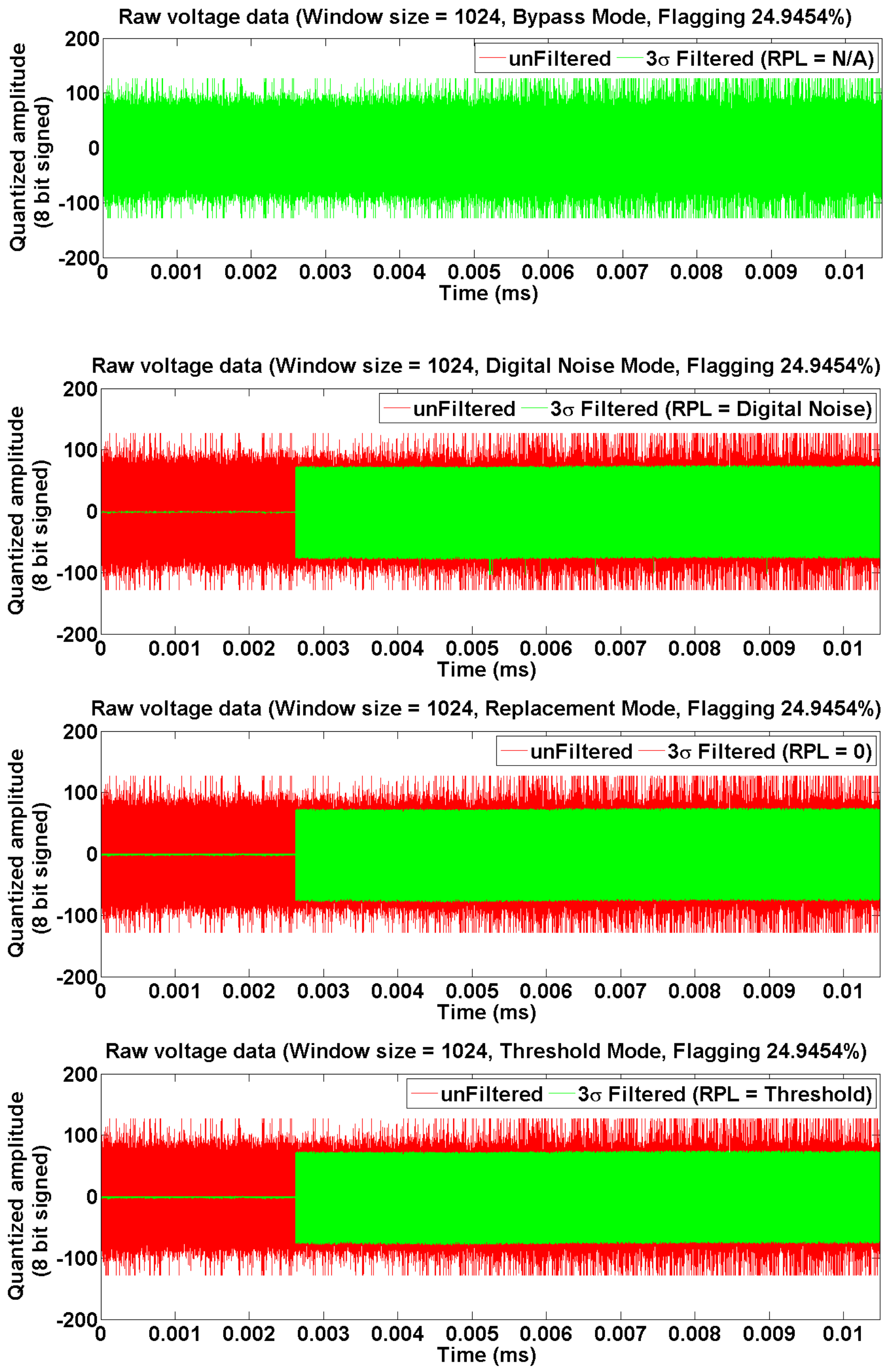


Figure 5.12 Filter output of MoM-based RFI Filter

5.5.3 Functional Verification

As discussed in section 5.4, Functional verification is carried out by comparing the result of the VHDL design and the golden reference-MATLAB. The overlay plot is shown in figure 5.13. The plots look almost similar. There are some mismatches in comparison because MATLAB works on full precision values, whereas VHDL works only on integer values.

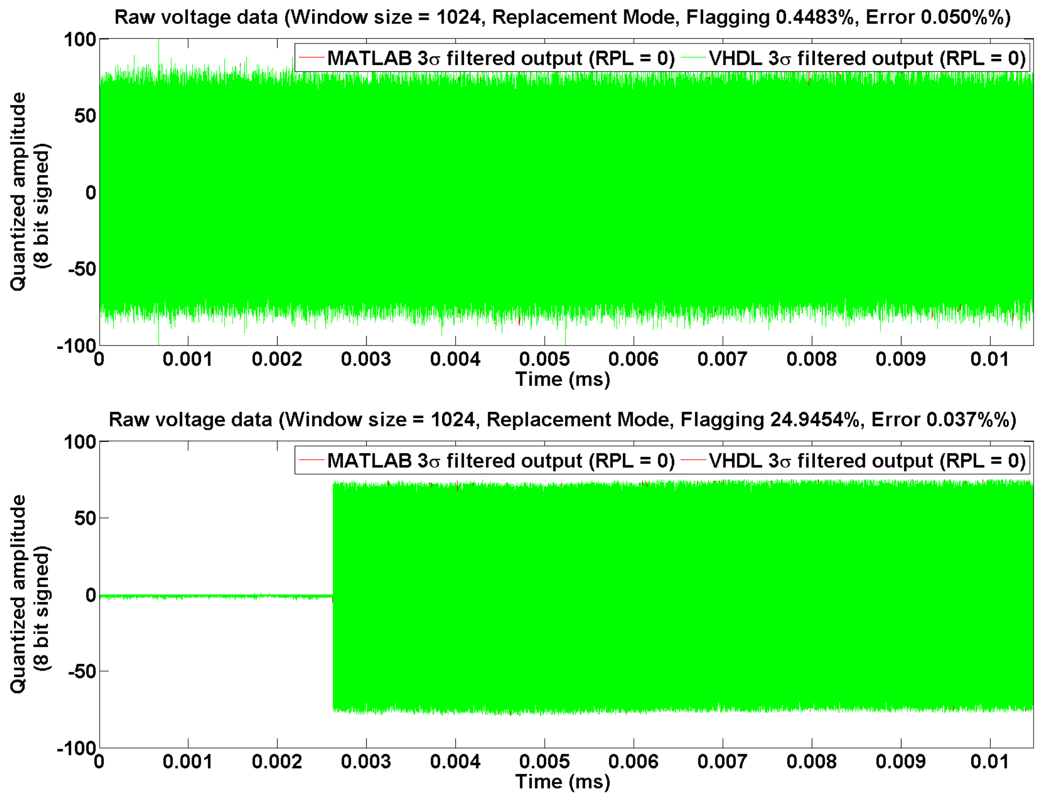


Figure 5.13 Functional Verification of VHDL implementation with golden model-MATLAB

Chapter 06: Power detection

6.1 Algorithm

The power detection technique uses power domain signals for detecting RFI samples. The sample values in the voltage domain are squared and accumulated. Accumulating samples improves the detection performance, reduces noise variance, and makes the RFI samples distinct.

The detector operates on chi-squared distribution (sum of the square of Gaussian distributed samples). The squaring and accumulation operation on the data results in a central chi-square distribution whose mean and variance depend on the voltage domain data and the number of samples accumulated.

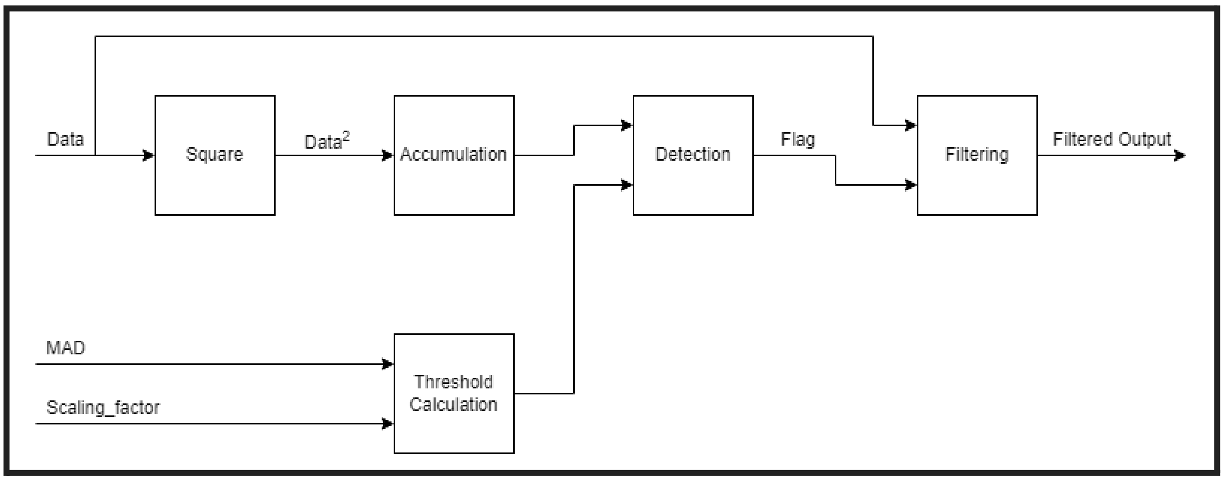


Figure 6.1 Block diagram of Filtering in Power domain

The relationship between the mean and standard deviation in the chi-square domain as computed from the respective mean and standard deviation in the time-domain is given by

$$\mu_p = n\sigma^2$$

$$\sigma_p = \sqrt{(2n)}\sigma^2$$

Threshold in terms of μ_p and σ_p

$$\text{Threshold} = \lambda * \sigma^2\ (n+\sqrt{(2n)})$$

If the accumulation of square values for subwindow size is greater than the threshold value, then it is detected as RFI, and the entire block of subwindow size samples is flagged as RFI.

6.2 Implementation details

Table 6.1 describes the I/O interface of the top entity for RFI Filter design in the Power domain. Moreover, squares of data samples are getting accumulated and then being replaced based on threshold comparison, there is a need of buffering the data till sub-window cycles and that is being done by BRAM inside the FPGA.

Table 6.1 VHDL interface of RFI Filter in Power domain

Signal	Description
clk	Input clock
rst	Active high, Synchronous reset
hold	Control signal for enable counter output
rst_sub_win_count	Active high, Synchronous reset for counter block which generates reset signal for main accumulator and auxiliary accumulator
rst_addr_count	Active high, Synchronous reset for RAM address generating counter
rst_counter	Active high, Synchronous reset for counter
median_valid	Input Median valid signal
mad_valid	Input MAD valid signal
data_in [7 downto 0]	Input data
select_line [1 downto 0]	Replacement mode control signal
median [7 downto 0]	Input Median value
mad [7 downto 0]	Input MAD value
scaling_factor [7 downto 0]	Input Scaling factor
total_count [31 downto 0]	Output data count value
RFI_count [31 downto 0]	Output RFI count value
data_out [7 downto 0]	Filtered output
flag_out	Output RFI flag

6.3 Synthesis and Implementation

In this section, Synthesis and Implementation details are mentioned. Section 6.3.1 contains the FPGA Implementation view of the RFI Filter design in the power domain. Timing and Area-Utilizations details of the Implemented design are discussed in 6.3.2 and 6.3.3, respectively.

6.3.1 Implementation View

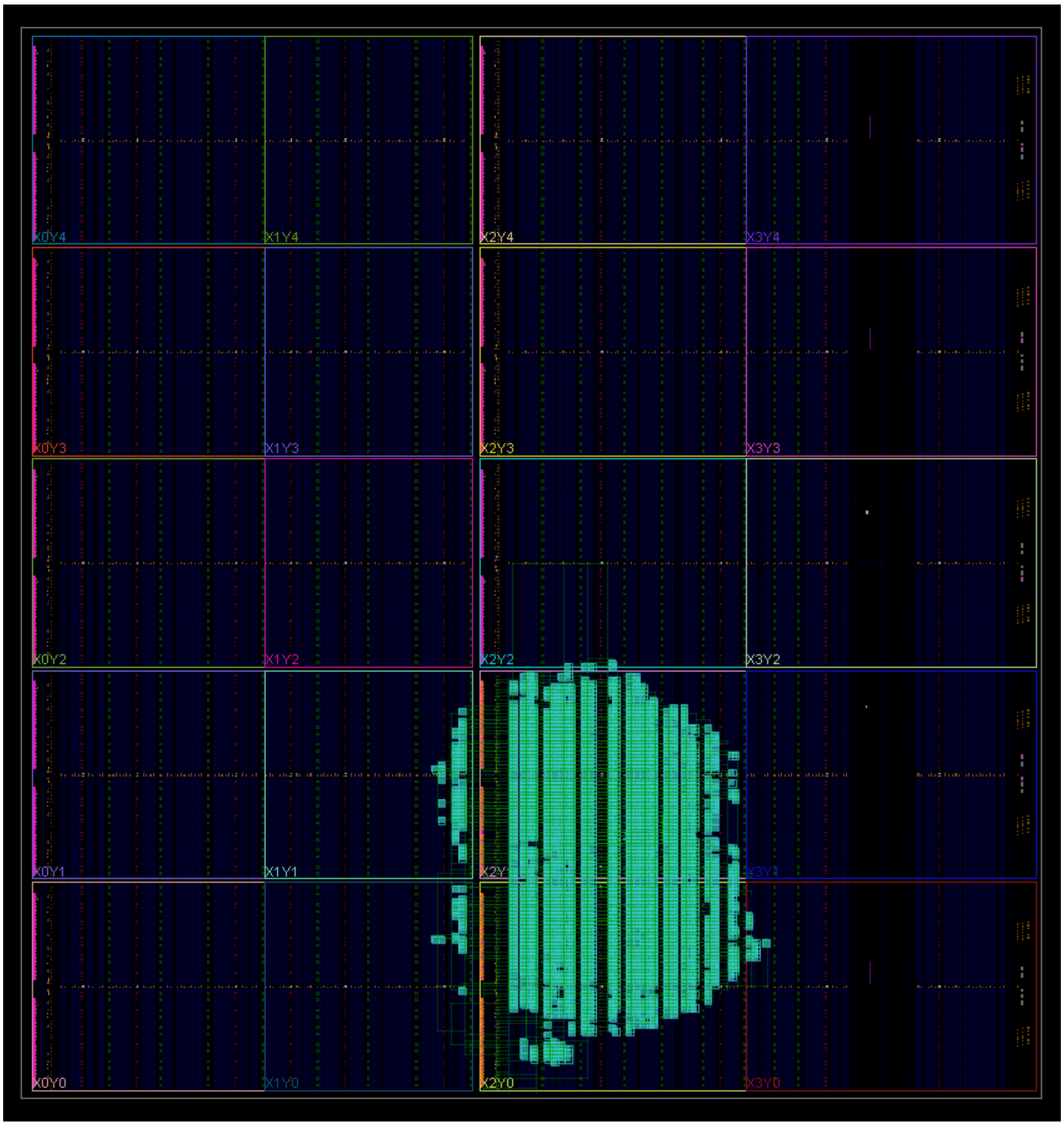
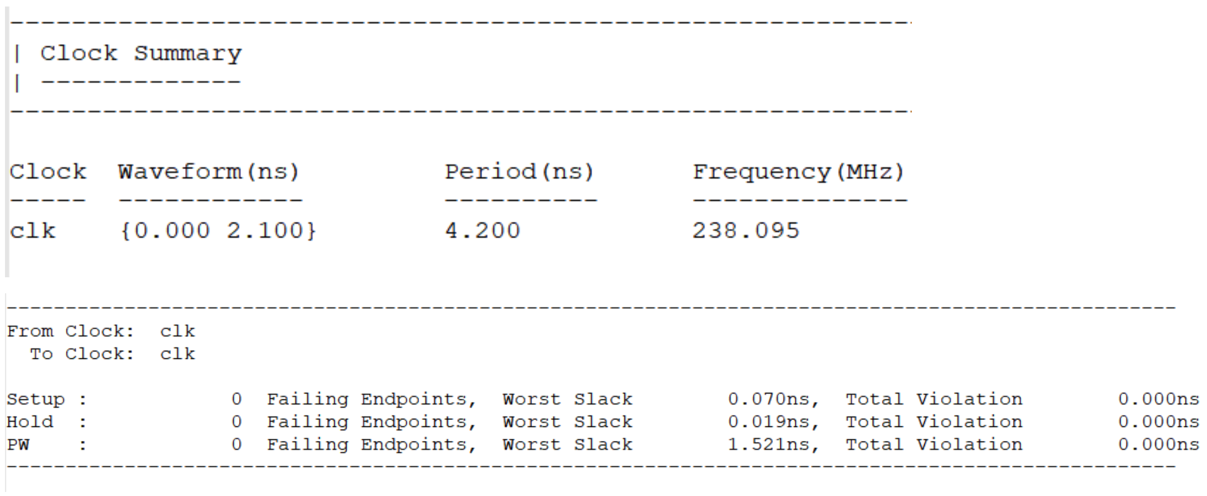


Figure 6.2 FPGA Implementation view of MAD based RFI Filter in Power domain

6.3.2 Timing details

Figure 6.3 shows that the Filter design works on 264.550 MHz frequency without any STA violation.

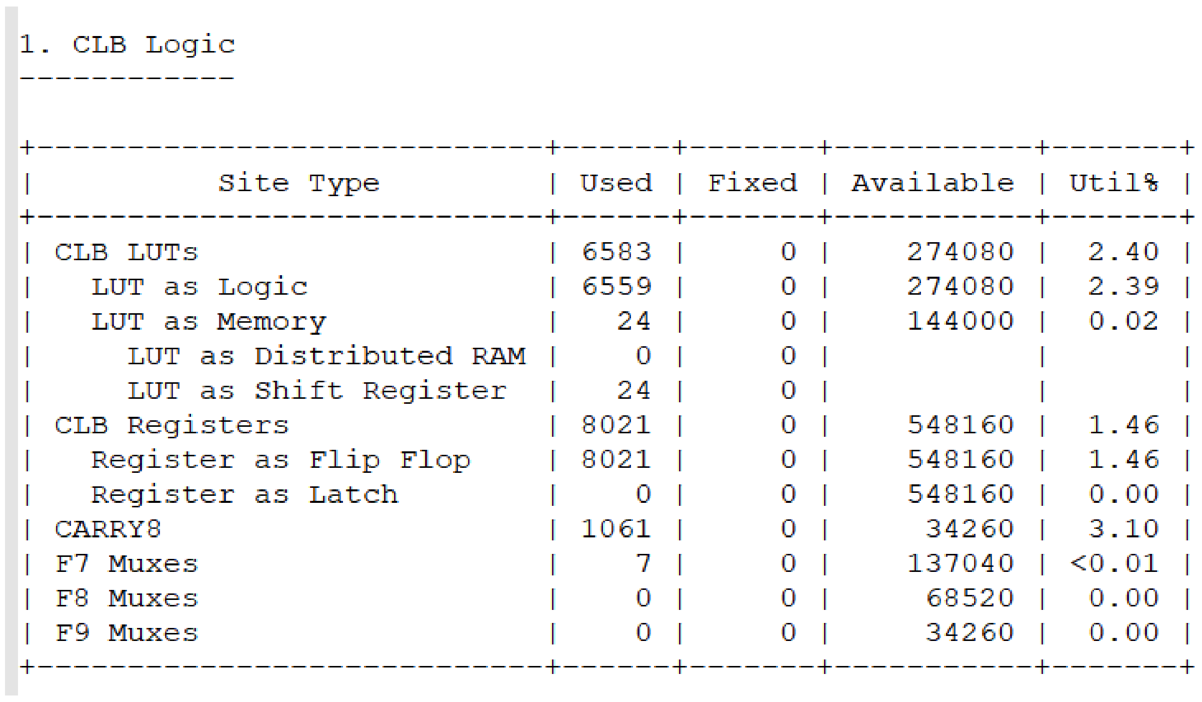


```
------------------------------------------------------------
| Clock Summary
| -------------
------------------------------------------------------------

Clock  Waveform(ns)         Period(ns)      Frequency(MHz)
-----  ------------         ----------      --------------
clk    {0.000 2.100}        4.200           238.095

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

Setup :            0  Failing Endpoints,  Worst Slack        0.070ns,  Total Violation        0.000ns
Hold  :            0  Failing Endpoints,  Worst Slack        0.019ns,  Total Violation        0.000ns
PW    :            0  Failing Endpoints,  Worst Slack        1.521ns,  Total Violation        0.000ns
----------------------------------------------------------------------------------------------------
```

Figure 6.3 Timing report of FPGA implementation of MAD-based Power domain Filter

6.3.3 Utilization details

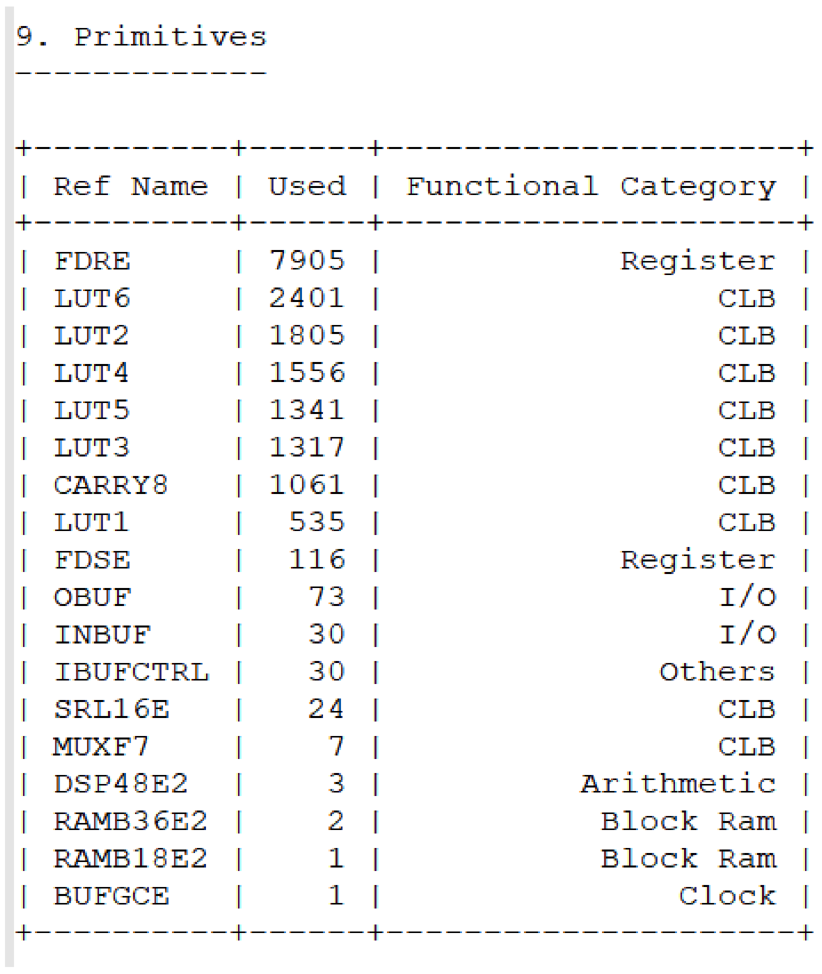
Figure 6.4 gives an overall idea about resource utilization. Also, we can say that arithmetic operations inside the threshold calculation block infer the DSP block of the FPGA, and there is no register as a latch. Moreover, subwindow size data buffers infer RAMB18.



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

+-----------------------------+------+-------+-----------+-------+
|          Site Type          | Used | Fixed | Available | Util% |
+-----------------------------+------+-------+-----------+-------+
| CLB LUTs                    | 6583 |     0 |    274080 |  2.40 |
|   LUT as Logic              | 6559 |     0 |    274080 |  2.39 |
|   LUT as Memory             |   24 |     0 |    144000 |  0.02 |
|     LUT as Distributed RAM  |    0 |     0 |           |       |
|     LUT as Shift Register   |   24 |     0 |           |       |
| CLB Registers               | 8021 |     0 |    548160 |  1.46 |
|   Register as Flip Flop     | 8021 |     0 |    548160 |  1.46 |
|   Register as Latch         |    0 |     0 |    548160 |  0.00 |
| CARRY8                      | 1061 |     0 |     34260 |  3.10 |
| F7 Muxes                    |    7 |     0 |    137040 | <0.01 |
| F8 Muxes                    |    0 |     0 |     68520 |  0.00 |
| F9 Muxes                    |    0 |     0 |     34260 |  0.00 |
+-----------------------------+------+-------+-----------+-------+
2. CLB Logic Distribution
-------------------------

+------------------------------------------+------+-------+-----------+-------+
|                Site Type                 | Used | Fixed | Available | Util% |
+------------------------------------------+------+-------+-----------+-------+
| CLB                                      | 2218 |     0 |     34260 |  6.47 |
|   CLBL                                   | 1069 |     0 |           |       |
|   CLBM                                   | 1149 |     0 |           |       |
| LUT as Logic                             | 6559 |     0 |    274080 |  2.39 |
|   using O5 output only                   |  531 |       |           |       |
|   using O6 output only                   | 3632 |       |           |       |
|   using O5 and O6                        | 2396 |       |           |       |
| LUT as Memory                            |   24 |     0 |    144000 |  0.02 |
|   LUT as Distributed RAM                 |    0 |     0 |           |       |
|   LUT as Shift Register                  |   24 |     0 |           |       |
|     using O5 output only                 |    0 |       |           |       |
|     using O6 output only                 |   24 |       |           |       |
|     using O5 and O6                      |    0 |       |           |       |
| LUT Flip Flop Pairs                      | 4279 |     0 |    274080 |  1.56 |
|   fully used LUT-FF pairs                | 2225 |       |           |       |
|   LUT-FF pairs with one unused LUT       | 2036 |       |           |       |
|   LUT-FF pairs with one unused Flip Flop | 2015 |       |           |       |
| Unique Control Sets                      |  534 |       |           |       |
+------------------------------------------+------+-------+-----------+-------+
```

```
3. BLOCKRAM
-----------

+-------------------+------+-------+-----------+-------+
|     Site Type     | Used | Fixed | Available | Util% |
+-------------------+------+-------+-----------+-------+
| Block RAM Tile    |  2.5 |     0 |       912 |  0.27 |
|   RAMB36/FIFO*    |    2 |     0 |       912 |  0.22 |
|     RAMB36E2 only |    2 |       |           |       |
|   RAMB18          |    1 |     0 |      1824 |  0.05 |
|     RAMB18E2 only |    1 |       |           |       |
+-------------------+------+-------+-----------+-------+
```

```
4. ARITHMETIC
-------------

+----------------+------+-------+-----------+-------+
|    Site Type   | Used | Fixed | Available | Util% |
+----------------+------+-------+-----------+-------+
| DSPs           |    3 |     0 |      2520 |  0.12 |
|   DSP48E2 only |    3 |       |           |       |
+----------------+------+-------+-----------+-------+
```



```
9. Primitives
-------------

+-----------+------+---------------------+
| Ref Name  | Used | Functional Category |
+-----------+------+---------------------+
| FDRE      | 7905 |            Register |
| LUT6      | 2401 |                 CLB |
| LUT2      | 1805 |                 CLB |
| LUT4      | 1556 |                 CLB |
| LUT5      | 1341 |                 CLB |
| LUT3      | 1317 |                 CLB |
| CARRY8    | 1061 |                 CLB |
| LUT1      |  535 |                 CLB |
| FDSE      |  116 |            Register |
| OBUF      |   73 |                 I/O |
| INBUF     |   30 |                 I/O |
| IBUFCTRL  |   30 |              Others |
| SRL16E    |   24 |                 CLB |
| MUXF7     |    7 |                 CLB |
| DSP48E2   |    3 |          Arithmetic |
| RAMB36E2  |    2 |           Block Ram |
| RAMB18E2  |    1 |           Block Ram |
| BUFGCE    |    1 |               Clock |
+-----------+------+---------------------+
```

Figure 6.4 Utilization report of FPGA implementation of MAD-based Power domain Filter

Conclusion and Future Scope

MAD-based RFI filtering algorithm is implemented using VHDL on the Xilinx Kintex Ultrascale FPGA device. Different variants like MoM-based filtering and Power domain filtering are also implemented. By comparing the results of VHDL designs with MATLAB golden reference, we can see minor differences in the output values because MATLAB works on full precision, whereas VHDL works on integer values. All the designs are working on 238 to 300 MHz synthesis frequency.

References

[1] https://en.wikipedia.org/wiki/Giant_Metrewave_Radio_Telescope

[2] https://public.nrao.edu/telescopes/radio-frequency-interference/

[3] https://github.com/casper-astro/hdl_devel/wiki/Verilog-coding-guidelines

[4] K. D. Buch, Y. Gupta, S. Bhatporia, S. Nalawade, K. Naik and B. Ajithkumar, "Real-time RFI excision for the GMRT wideband correlator," 2016 Radio Frequency Interference (RFI), 2016, pp. 11-15, doi: 10.1109/RFINT.2016.7833523.

[5] Buch, Kaushal & Naik, Kishor & Nalawade, Swapnil & Bhatporia, Shruti & Gupta, Yashwant & Ajithkumar, B.. (2019). Real-time Implementation of MAD-based RFI Excision on FPGA. Journal of Astronomical Instrumentation. 08. 10.1142/S2251171719400063.

[6] Buch, Kaushal & Gupta, Yashwant & Kumar, B.. (2014). Variable Correlation Digital Noise Source on FPGA — A Versatile Tool for Debugging Radio Telescope Backends. Journal of Astronomical Instrumentation. 03. 1450007. 10.1142/S225117171450007X.

[7] Chu, P. & Jones, R. [1999] Design techniques of FPGA based random number generator, Military and Programmable Logic Devices (MAPLD) Conf., pp. 1–6.

[8] https://casper.astro.berkeley.edu/wiki/Tutorial_-_Noise_Source

[9] https://web.mit.edu/6.111/volume2/www/f2019/handouts/labs/lab3_19/rom_vivado.html

[10] https://vlsiverify.com/system-verilog/functional-coverage/functional-coverage

Appendix

A.1 Generics value to be define in top entity

Table A.1 Generic Description

Generic name	Value	Description
data_width	8	8 bit signed integer data
reg_width	10 to 14	Window size varies from 1024 to 16384
count_width	32	Counter width
mode_select	2	4 different filtering options

A.2 Datasets used for testing

Table A.2 Details of Datasets used for testing

Category	Name	Data length	Data length (condensed)
Small Sample data	data16300.txt	16300	16 KB
	c11A.txt	327680	320 KB
Simulator data	1us.txt	1632256	1.5 MB
Band 2 Antenna Data	B2_C11_1.txt	4194304	4 MB
	B2_C12_1.txt	4194304	5 MB
Band 3 Antenna Data	B3_C11_1.txt	4194304	6 MB
	B3_C12_1.txt	4194304	7 MB
Band 4 Antenna Data	B4_C11_1.txt	4194304	8 MB
	B4_C12_1.txt	4194304	9 MB
Band 5 Antenna Data	B5_C11_1.txt	4194304	10 MB
	B5_C12_1.txt	4194304	11 MB
4 Bit Data	4bit_C01_short.txt	327680	320 KB
	4bit_C01.txt	16384000	16 MB
	4bit_C11.txt	16384002	16 MB
Long Data for MOM	1g.txt	449998848	430 MB
	2g.txt	999997440	953 MB