"DESIGN OF DA BASED IP-CORE FOR HALF BAND FILTER AND POLYPHASE DECIMATOR"

Major Project Report

Submitted in Partial Fulfilment of the Requirements
For the Degree
Of

MASTER OF TECHNOLOGY

IN

ELECTRONICS & COMMUNICATION ENGINEERING
(VLSI Design)

By

Priyanka Gupta
(05MEC004)

Department of Electronics & Communication Engineering
Institute of Technology
Nirma University of Science & Technology
Ahmedabad-382481
May 2007

"DESIGN OF DA BASED IP-CORE FOR HALF BAND FILTER AND POLYPHASE DECIMATOR"

Major Project Report

Submitted in Partial Fulfilment of the Requirements
For the Degree
Of

MASTER OF TECHNOLOGY
IN
(VLSI Design)

By
Priyanka Gupta
(05MEC004)

Under the Guidance of
Dr. K.S.Dasgupta
Group Director, SAC (ISRO)

Department of Electronics & Communication Engineering
Institute of Technology
Nirma University of Science & Technology
Ahmedabad-382481
May 2007

CERTIFICATE

This is to certify that the Major Project Report entitled “**Design of DA based IP-CORE for Half Band Filter and Polyphase Decimator** " submitted by **Priyanka Gupta (**Roll No**. 05MEC004**) as the partial fulfilment of the requirements for award of the degree of M.Tech (EC-VLSI Design) awarded by Institute of Technology, Nirma University, Ahmedabad embodies work carried out by her under my supervision at Space Application Centre (ISRO), Ahmedabad, Gujarat during w.e. f. SEPTEMBER 2006-APRIL 2007.

Date:
Place:

Dr. K. S. Dasgupta
Group Director-ADCTG
SAC (ISRO)
Ahmedabad

CERTIFICATE

This is to certify that the Major Project Report entitled **“Design a DA Based IP-CORE of Half Band Filter and Polyphase Decimator**"submitted by **Priyanka Gupta (**Roll No. **05MEC004)** as the partial fulfilment of the requirements for the award of the degree of Master of Technology in Electronics & Communication **(VLSI Design)** of Institute of Technology **Nirma University** is the record of work carried out by her under my/our supervision and guidance. The work submitted has in our opinion reached a level required for being accepted for the examination. The results embodied in this major project work to the best of our knowledge have not been submitted to any other University or Institution for award of any degree or diploma.

Date:

Facilitator at Institute
Prof. N. P. Gajjar
Nirma University
Ahmedabad

Dr. N. M. Devashrayee
PG Coordinator-VLSI Design

Prof. A. S. Ranade
HOD
EC-Department
Nirma University
Ahmedabad

Prof. A. B. Patel
Director
Institute of Technology,
Nirma University
Ahmedabad

ACKNOWLEDGEMENT

It has been great pleasure for me in doing a major project work on DESIGN A DA BASED IP-CORE OF HALF BAND FILTER AND POLYPHASE DECIMATOR under Guidance of Dr. K. S. Dasgupta Group Director, SAC (ISRO). I am very grateful to him who assigned me a project under his expert guidance in the ACTD/ADCTG Department, without his invaluable guidance the work would have not been possible. His academic excellence continues to be a source of inspiration. He always inspires me to put best efforts to achieve the goal.

I would like to express my sincere thanks to Prof. N. P. Gajjar for his kind, ordinal and valuable guidance and support at every moment of the project His practical approach, needful help encouraged me to do better work. His dedication to work always inspired me to do task as a challenge.

I like to express thanks to Director, Dr. N. M. Devashrayee, PG co-ordinator-VLSI Design and Institute for providing me such nice opportunity to do my dissertation work at the prime organisation like SAC (ISRO) Ahmedabad.

I am very much thankful to Mr. Pinakin Thaker who has rendered their guidance and extended co-operation at all times. I would like to thank Mr. TVS Ram, Mr. V.K.Tank, Mrs. Anita, Mr. Deepak Mishra, Mr. Vipin Jain and Miss Priyanka kaushik for their support. I also express thanks to all ADCTG staff that have directly and indirectly helped me in the accomplishment of the project.

I express my gratitude to Mr. RJK Jain Head, HRD department (SAC) who accepted my candidature for dissertation work at SAC, ISRO Ahmedabad.
Thanks to my family members for their faith, colleagues for giving me support. Finally thanks to God for helping me out at difficult times.

Priyanka Gupta
05MEC004

ABSTRACT

The most common family of digital filters is made up of the finite impulse response (FIR), which is used in various applications. Digital FIR filtering is a very compute intensive operation, and since many applications require this filtering to be done in real-time, it calls for an efficient implementation of FIR filters.

Distributed Arithmetic(DA) algorithm provides us a platform to implement multiply-and-accumulate function more efficiently. The architecture based on DA algorithm makes extensive use of look-up tables. It makes them ideal for implementing digital signal processing functions on FPGAs. Distributed architectures are suitable for low power portable applications, because they replace the costly multipliers with shifts and look-up tables.

This work deals with DA based FIR implementation. Basic approach of distributed arithmetic algorithm is analyzed and simulated using Matlab. Timing analysis of control circuitry w. r. to input/output has been completed for Half band filter and Polyphase decimator. A common GUI is created for all programmable inputs in Matlab environment for the computation of filter coefficients, golden reference and required look up table for hardware implementation. The filter hardware implementation uses modular approach. The design is implemented using VHDL code styles and simulation is verified. The design details, its simulation, analytical evaluation and comparison with Matlab golden reference are also highlighted. Code is fully synthesizable. Chipscope testing is also carried out and results are matched with the golden references and Modelsim results.

LIST OF FIGURES

Figure no. continue

LIST OF TABLES

GLOSSARY OF TERMS

ASIC: Application Specific Integrated Circuit.

FPGA: Field Programmable Gate Array.

MAC: Multiply and Accumulate.

DA: Distributed Arithmetic.

PSC: Parallel to Serial Converter.

TSB: Time Skew Buffer.

BAAT: Bit at a Time.

SDA: Serial Distributed arithmetic FIR.

PDA: Parallel Distributed arithmetic FIR

CONTENTS

ABOUT ISRO

ABOUT ORGANIZATION:

The Indian Space Program was formally organized in 1972 when Government of India set up the development and application of space technology and space sciences for the socioeconomic benefit of the commission include the formulation if the policy of the Department of Space and implementation of the Government's policy in all matters concerning outer space.

The Indian Space Research Organization (ISRO) under DOS plays a key role, through its centers, in the planning and execution of National Space activities. It is also responsible for technical management in the area of Space Application and space technology. Basic natural resources survey, and meteorology, other R&D activities of ISRO in satellite achieving the basic application objectives

OBJECTIVES OF ISRO

- Long distance telecommunication, diffusion of TV signals using satellite. Remotes sensing of natural and renewable earth resources and meteorological parameters from satellites.
- Satellite based resources survey, management and environment monitoring.
- Development and operationalisation of indigenous satellite.
- To realize the above objectives of ISRO activities are oriented predominantly towards design and development of applications satellites for communications, Remote Sensing, TV broadcasting and meteorology.
- Design and Development of Rocket launching vehicles to place these application satellites into the required orbits.
- Establishment of ground stations/facilities for using these satellites and for launching.

The headquarters of both DOS and ISRO located at Bangalore provides overall direction to the technical, scientific and administrative functions of the ISRO centers/units. These are as follows

Satellite Launching Technology Development Centers:

- Vikram Sarabhai Space Centre (VSSC)-Trivandrum.
- SHAR Centre-Sriharikota.
- ISRO Telemetry, Tracking and Command Network (ISTRAC)-Bangalore.
- ISRO Satellite Centre (ISAC)-Bangalore.

SAC, Ahmedabad is ISRO’s application R& D centre. The primary responsibilities of the centre are to conceptualize plan and execute projects and research programmes leading to practical applications of space technology. The main area of activities are satellite based telecommunication, and Remote Sensing for natural resources survey and management, environment test facilities and reliability and quality assurance. SAC also manages the DELHI earth station.

The major tasks of this centre are to conceptualize and conduct research and development, and execute projects in the field of space application. To this end, SAC has two board areas of activity viz., satellite based communication, including television, and remote sensing for natural resources survey and management, meteorology and geodesy.

To carry out its programmes, the centre is organized functionally in to the communication area and the remote sensing area. The technical service group, test, evaluation, standards and calibration facility and Payload Fabrication Facility provide the support services.
Some of the major facilities at SAC include:

Experimental Satellite Communication Earth Station at Ahmedabad and Delhi.
Transportable Remote Area Communication Terminal (TRACT).
Emergency Communication Terminal (ETC).
Small Communication Terminal (SCOT).
Meteorological Data Reception Terminal.
Electronics and Mechanical Fabrication.
Environmental and Space Simulation.
Visual Photo Interpretation.
Digital Image processing computer.

Antenna Test Range.

INTRODUCTION TO ADCTG

It is responsible for the technology development related to on board signal processing required for future SATCOM services. ADCTG consists of the following two divisions.

On Board Signal Processing Division (OSPD)
OSPD is responsible for the development of on board signal processing technology for future SATCOM services.

Advanced Communication Technology Division (ACTD)
ACTD carry out advanced R&D for satellite based digital communication & secured communication. It is also responsible for technology development related to wide band networks.

ABOUT FACILITY

As ISRO is an R & D organization, it is very healthy in form of laboratory and in form of Equipment’s facility, we have separate Computer machine with required software. (e. g. Matlab, Xilinx. etc.). There is good facility of library, which contains almost all-technical literature in form of books, journals, research paper.

CHAPTER 1 INTRODUCTION

1.1 Motivation of Thesis

In the whole landscape of digital signal processing, digital filters stand in the very foreground. These are designed and implemented either through software or hardware. There are two types of digital filters: IIR (Infinite Impulse Response) and FIR (Finite Impulse Response) filters. FIR filters are very commonly used in digital signal processing applications and traditionally implemented using ASICs or DSP-processors. FIR filters have the advantage of linear phase response and stablityThe filter equation is given as

$$y[n] = \sum_{k=0}^{N-1} a[k]x[n-k]$$

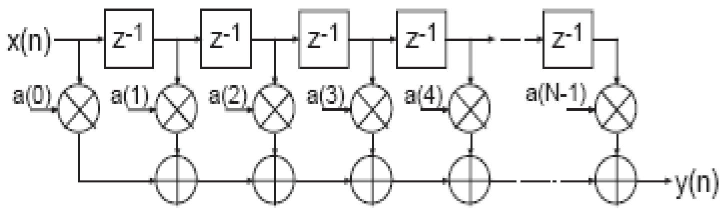


Figure 1.1*:* Conventional FIR Filter implementation

This is one of two approaches to implement a FIR filter with a dedicated hardware multiplier. If the filter coefficients are fixed, the flexibility offered by a multiplier is not required. Such filters can be implemented efficiently in terms of optimum area, delay and power.

Distributed Arithmetic provides an alternate approach for multiplier less implementation of FIR filter that performs multiplication using look up table. Look up table is a convenient way to implement the multiply intensive algorithms like FIR filters, provided that one of the multiplication operands is constant. As xilinx FPGA are LUT based and hence they are best suitable for this implementation.

1.2 Organization of Thesis

This report gives brief introduction of DA FIR, targeted algorithm and its impact on design, various timing requirements and related issues to meet the goal. Report is mainly divided into five chapters. Second chapter deals with basics of filter implementation algorithms. The second chapter highlights the mathematics associated the DA algorithm. Third chapter includes the types of DA FIR, characteristics of Half Band filter and polyphase decimator. Fourth chapter covers about the specification, interface and control signals those are needed for the design of Half Band filter and polyphase decimator. Next chapter includes design strategy for implementation of generic half band filter. Sixth chapters include the design simulation results in Matlab and Modelsim. It also includes the hardware implementation, design verification and comparison with simulation. Chapter 7 deals with concludes and projects future extension of the work. In Appendix-A, B Matlab and VHDL program listing are attached. The VHDL implementation is fully generic, synthesizable and tested with test bench for various input condition.

CHAPTER 2
FILTER IMPLEMENTATION ALGORITHMS

Most of the DSP algorithms require multiplication and addition in real-time. The unit carrying out this function is called MAC (multiply accumulate). Three choices of technology exist for the implementation of DSP algorithms. These are:

- Programmable DSP chips
- ASICs.
- FPGAs.

Programmable DSP chips typically have only one MAC unit that can perform one MAC in less than a clock cycle. DSP processors or programmable DSP chips are flexible, but they are not be fast enough. As DSP processor is general purpose and has architecture that constantly requires instructions to be fetched decoded and executed. ASICs can have multiple dedicated MACs that perform DSP functions in parallel. The FPGA architecture allows multiple MACs and pipelining. The only drawback is the speed, and this can easily be overridden by using effective computational algorithms suitable for FPGAs. But they have the advantage of re-configurability.

Here basics of two filter implementation algorithms are described.

Let x(n) be the input at time n, then the filter output is given by:

$$y(n) = \sum_{k=0}^{N-1} h(k) \cdot x(n-k) \qquad \text{----------------} \qquad (2.1)$$

$h(k)$ = filter coefficients for k = 0; 1; :::;N - 1 .

$x(n)$ = Input response at time n.

$x(n-1)x(n-2)----------x(n-N+1)$ Are the previous inputs.

$y(n)$ = Output response at time n.

2.1. MAC (Multiply-Accumulate) Algorithm:

In a DSP (Digital signal Processor), many “multiply-accumulate" operations are performed. Basically, it is a matter of building a sum of products, in slightly modified version

$$y = \sum_{k=0}^{N} a_k x_k = a_0 x_0 + a_1 x_1 + a_2 x_2 + --------------- + a_N x_N \qquad \text{----------} \qquad (2.2)$$

One uses to compute this expression gradually, by means of partial sum S_n

$$S_n = \sum_{k=0}^{n} a_k x_k = a_0 x_0 + a_1 x_1 + a_2 x_2 + \text{---------------} + a_n x_n) \qquad \text{-------} (2.3)$$

$$S_{n+1} = S_n + a_{n+1} x_{n+1} \qquad \text{-------} (2.4)$$

This is exactly required in MAC (multiply accumulate) whose schematic is shown below.

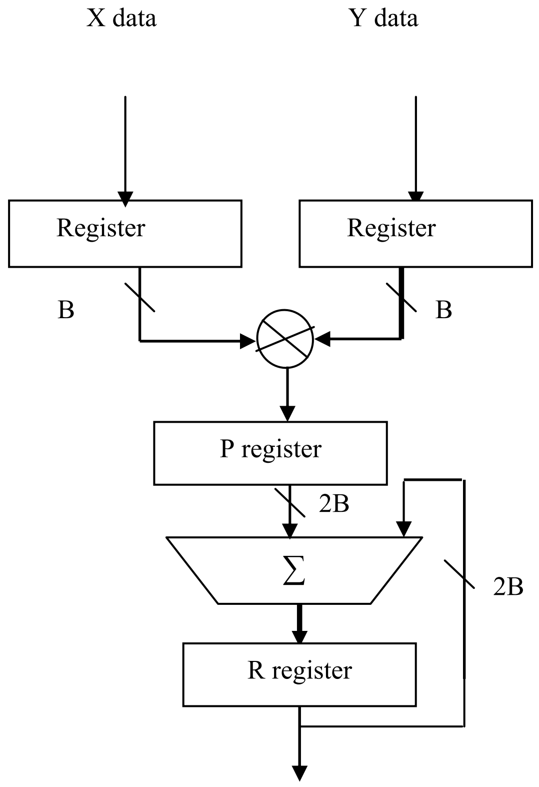


Figure 2.1: Multiply – accumulate structure

The main problem is that on any architecture, the multipliers are costly (they need a lot of logical gates to be built), and in many cases they do not exactly fit your needs, because their input data size is either too large or too small. However if the filter coefficients are fixed, the flexibility offered by a multiplier is not required. Such filters can then be implemented more efficiently in terms of power, area and delay.

2.2 Distributed Arithmetic Algorithm

Distributed algorithm provides an approach for multiplier less implementation of FIR filters where the filter coefficients are programmable. In other words the same architecture can be used for a different set of coefficients.

Distributed Arithmetic is computation algorithm that performs multiplication with look-up table based schemes. DA specifically targets the sum of products (sometimes referred to as the vector dot product) computation that covers many of the important DSP filtering and frequency transforming functions. .

Distributed Arithmetic differs from conventional arithmetic only in the order in which it performs operations. The arithmetic sum of products that defines the response of linear, time-invariant networks (filters) can be expressed as

$$y(n) = \sum_{k=1}^{K} A_K \quad x_K(n) \qquad (2.5)$$

$y(n)$ = response of network at time n.
$x_K(n)$ = kth input variable at time n.
A_K = weighting factor of kth input variable that is constant for all n, and so it remains time-invariant.
In filtering applications the constants, A_K, are the filter coefficients and the variables $X_K(n)$ are the samples of a single data source (for example, an analog to digital converter).
The multiply-intensive nature of eqn (2.5) can be appreciated by observing that a single output response requires the accumulation of K product terms. In DA the task of summing product terms is replaced by table look-up procedures that are easily implemented in the Xilinx configurable logic block (CLB) look-up table architecture.

We start by defining the number format of the variable to be signed integers. The variable, $x_K(n)$, may be written in the integer format as shown in equation

$$x_K = -x_{K(B-1)} \cdot 2^{B-1} + \sum_{b=0}^{(B-2)} x_{Kb} \cdot 2^{-b} \qquad \text{----------------- } (2.6)$$

Distributed Arithmetic, along with Modulo Arithmetic, are computation algorithms that perform where x_{kb} a binary variable is and can assume only values of 0 and 1. A sign bit of value -1 is indicated by x_{k0} . Note that the time index, n, has been dropped since it is not needed to continue the derivation. The final result is obtained by first substituting equ.2 into eqn.(2.5)

$$\begin{aligned} y = & -[x_{10}\,.A_1 + x_{20}.A_2 + x_{30}.A_3 + -------- + x_{k0}.A_K] + \\ & [x_{11}.A_1 + x_{21}.A_2 + x_{31}.A_3 + ------------ + x_{k1}.A_K].2^{-1} + \\ & [x_{12}.A_1 + x_{22}.A_2 + x_{32}.A_3 + ------------ + x_{k2}.A_K].2^{-2} + \\ & ------------------------------ + \\ & [x_{1(B-1)}.A_1 + x_{2(B-1)}.A_2 + x_{3(B-1)}.A_3 + ----------- + x_{k(B-1)}.A_K].2^{-(B-1)} \end{aligned} \quad \text{-------------- (2.7)}$$

Each term within the brackets denotes a binary AND operation involving a bit of the input variable and all the bits of the constant. The plus signs denote arithmetic sum operations. The exponential factors denote the scaled contributions of the bracketed pairs to the total sum. We can now construct a look-up table that can be addressed by the same scaled bit of all the input variables and can access the sum of the terms within each pair of brackets. Such a table is shown in fig. 1 and will henceforth be referred to as a Distributed Arithmetic look-up table or DALUT. The same DALUT can be time-shared in a serially organized computation or can be replicated B times for a parallel Computation scheme, as described later. This ends our derivation of the DA algorithm.

Address X[k,k-1,k-2,----3,2,1,0]	**Lut content**
0000	0
0001	A_0
0010	A_1
0011	$A_1 + A_0$
0100	A_2
0101	$A_2 + A_0$
0110	$A_2 + A_1$
0111	$A_2 + A_1 + A_0$

Table 2.1: Look up table addresses and their contents

The arithmetic operations have now been reduced to addition, subtraction and binary scaling. With scaling by positive powers of 2, the actual implementation entails the shifting of binary coded data words toward the most significant bit and the use of sign extension bits to maintain the sign at its normal bit position.

2.3 Comparison of MAC algorithm and DA algorithm:

PAARAMETERS	MAC BASED FIR	DA BASED FIR
Multipliers	N	0
Adders	N-1	--
LUT	--	1
Scaling accumulator	--	1
Throughput	Depends on the length of Filter	Independent of the length, depends on I/P bit precision

Table 2.2: Comparison between MAC based FIR and DA based FIR for N tap

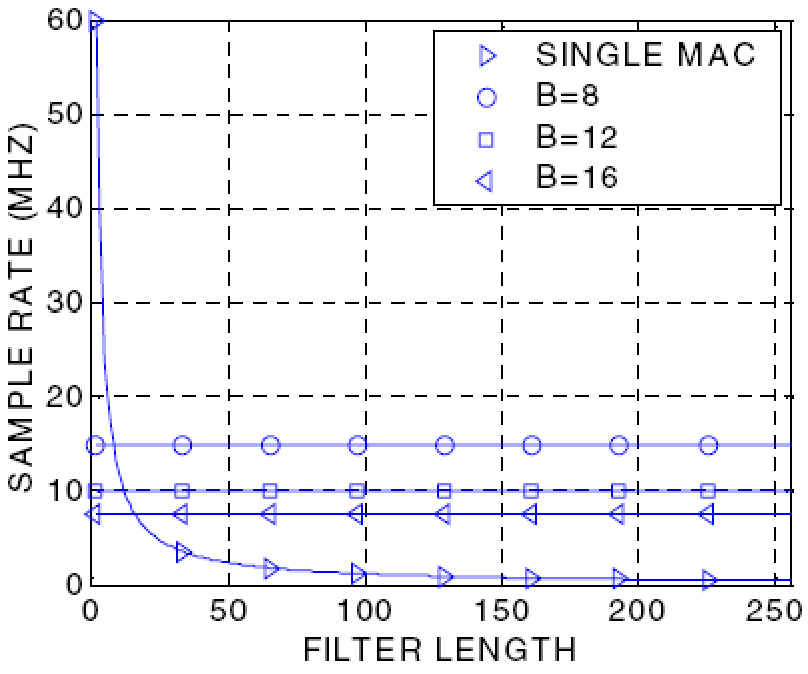


Figure 2.2: Throughput (Sample Rate) Comparison of Single-MAC-Based FIR and DA FIR.

As shown in the graph sample rate is inversely proportional the filter length in MAC. For small filter length sample rate is very high in MAC. As filter length is increased sample rate is decreased. It is also shown in the graph that sample rate is constant for all filter length in Distributed arithmetic It depends on input bit precision (B). for small values of B, sample rate is more compare to higher values of B in Distributed Arithmetic(DA).

CHAPTER 3
DISTRIBUTED ARITHMETIC FIR FILTER

3.1 Serial DA FIR:

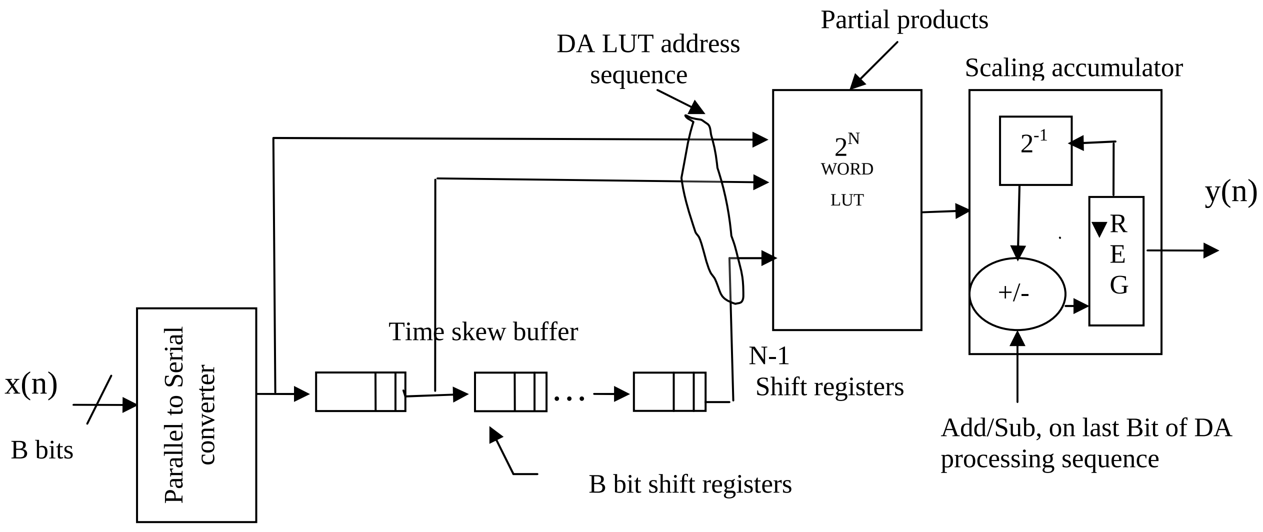


Figure 3.1: Serial Distributed Arithmetic FIR

Input samples are presented to the input parallel-to-serial shift register (PSC) at the input signal sample rate. As the new sample is serialized, the bit-wide output is presented to a bit-serial shift register or time-skew buffer (TSB). The TSB stores the input sample history in a bit-serial format and is used in forming the required inner-product computation. The TSB is itself constructed using a cascade of shorter bit–serial shift registers. The nodes in the cascade connection of TSBs are used as address inputs to a look-up table. This LUT stores all possible partial products over the filter coefficient space.

In a conventional multiply-accumulate (MAC)-based FIR realization, the sample throughput is coupled to the filter length. With a DA architecture, the system sample rate is related to the bit precision of the input data samples. Each bit of an input sample must be indexed and processed in turn before a new output sample is available. For *B*-bit precision input samples, *B* clock cycles are required to form a new output sample for a nonsymmetrical filter, and *B*+1 clock cycles are needed for a symmetrical filter. The rate at which data bits are indexed occurs at the *bit-clock* rate. The bit-clock frequency is greater than the filter sample rate (fs) and is equal to Bfs for a nonsymmetrical filter and (B+1) fs for a symmetrical filter. In a conventional instruction-set (processor) approach to the problem, the required number of multiply-accumulate operations are implemented using a time-shared or *scheduled* MAC unit. The filter sample throughput is inversely proportional to the number of filter taps. As the filter length is increased, the system sample rate is proportionately decreased. This is not the case with DA-based architectures. The filter sample rate is decoupled from the filter length. The trade off introduced here is one of silicon area (FPGA logic resources) for time. As the filter length is increased in a DA FIR filter, more logic resources are consumed, but throughput is maintained.

3.2 PARALLEL DA FIR:

In its most obvious and direct form, DA-based computations are bit-serial in nature; each bit of the samples must be indexed in turn before a new output sample becomes available (SDA FIR). When the input samples are represented with *B* bits of precision, *B* clock cycles are required to complete an inner-product calculation (for a nonsymmetrical impulse response). Additional speed can be obtained in several ways. One approach is to partition the input words into *M* sub words and process these sub words in parallel. This method requires *M*-times as many memory look-up tables and so comes at a cost of increased storage requirements. Maximum speed is achieved by factoring the input variables into single-bit sub words. The resulting structure is a fully parallel DA (PDA) FIR filter. With this factoring a new output sample is computed on each clock cycle. PDA FIR filters provide exceptionally high performance. The Xilinx filter Core provides support for parallel DA FIR

As for the fully parallel 8 bit PDA FIR filter implementation, the 8-bit input sample is partitioned into eight 1-bit sub-samples so as to achieve maximum speed. Figure shows the ultimate fully parallel PDA FIR filter, where all 8 input bits are computed in parallel and then summed by a binary-tree like adder network. The lower input to each adder is scaled down by a factor of 2. No scaling accumulator is needed in this case, since the output from the adder tree is the entire sum of products. If the data samples are processed 2 bits at a time (2-BAAT), a new output sample is ready every

12/2 = 6 clock cycles. With 3-,4-, 6- and 12-BAAT implementations, a new result is available every 4, 3, 2 and 1 clock cycles, respectively.

The higher the degree of filter parallelism (fewer number of clock cycles per output sample or smaller *L*), the greater the FPGA logic resources required to implement the design. Specifying the number of clock cycles per output sample is an extremely powerful mechanism that allows the designer to trade off silicon area with filter throughput.

3.3 Filter Throughput

The signal sample rate for a filter is a function of the core bit clock frequency, fclk Hz, the input data sample precision *B*, the number of channels, the number of clock cycles (*L)* per output sample, and the coefficient symmetry. For a single-channel nonsymmetrical FIR filter using *L*=B clock cycles per output sample, the filter sample frequency, or sample throughput, is fclk/B Hz. If the filter is symmetrical, the sample rate is fclk/(*B*+1) Hz. If the number of clock cycles per output sample is changed to *L*=1, the sample throughput is fclk Hz. For *L*=2, the throughput is fclk/2 Hz.

As a specific example, consider a filter with a core clock frequency equal to 100 MHz, 10-bit input samples, *L*=10 and a nonsymmetrical coefficient set. The filter sample rate is 100/10 = 10 MHz. Observe that this figure is independent of the number of filter taps. If a symmetrical realization had been generated, the sample throughput would be 100/11 = 9.0909 MHz. For *L*=1, the sample rate would be 100 MHz (nonsymmetrical FIR). If the input sample precision is changed to 8 bits, with *L*=8, the filter sample rate for a nonsymmetrical filter would be 100/8 = 12.5 MHz.

3.4 Half-Band FIR

The frequency response for a half-band filter is shown in figure

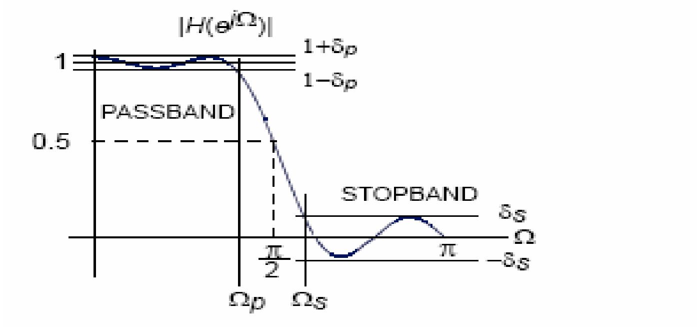


Figure 3.2: Half band filter frequency response

Filter characteristics:

1. The magnitude frequency response is symmetrical about quarter sample frequency Π/2 frequency..

2. The passband and stopband frequencies are positioned such that $\Omega_p = \Pi - \Omega s$

.3. The passband and stopband ripple are equal i.e.$\delta_p = \delta_s$

.4. Half of the filter coefficients are zero for an odd number of taps.

These properties are reflected in the filter impulse response. It is shown in figure for an 11-tap half-band filter.

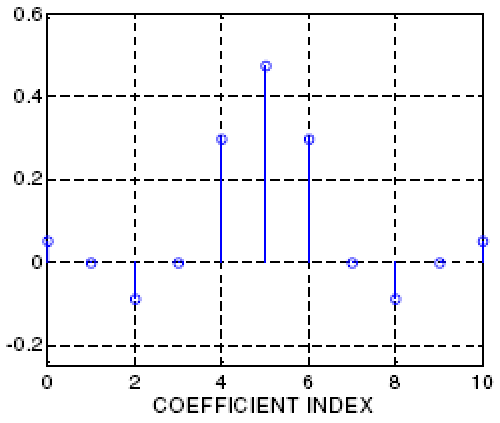


Figure 3.3: Impulse response of 11 tap half band filter

3.5 Basic operation in multirate system:

Decimator: Down sampler with a sampling factor M, develops an output sequence y(n)whose sampling rate is (1/M) that of the input sequence x(n) sampling rate.

$$y(n) = x(M.n) \quad \text{----------------------------} \quad (3 .1)$$

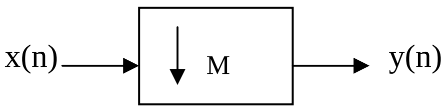


Figure 3.4: M fold Decimator

Where M is an integer. Only those samples of x(n) which occur at time equal to multiples of M are retained by the decimator. In the figure below for decimation factor M=2 output is shown.

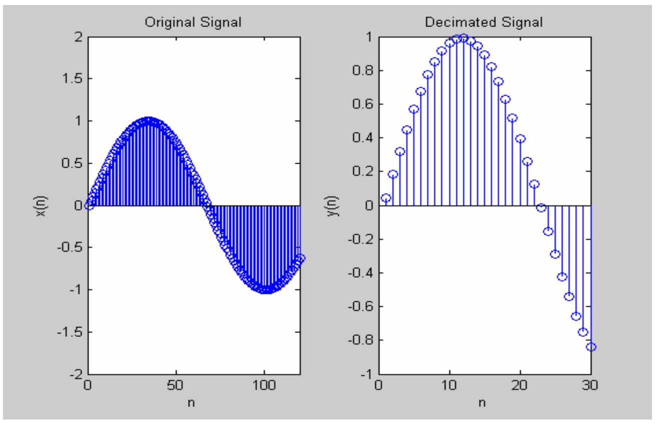


Figure 3.5: Decimator output M = 4

Interpolator: Up sampler with a sampling factor of L develops an output sequence whose sampling rate is L times that of original input signal.

$$y(n) = \begin{cases} x(n/L), & \text{if } n \text{ is integer multiple of } L \\ 0, & \text{otherwise} \end{cases} \quad \text{-------------------------------} \quad (3.2)$$

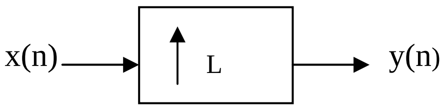


Figure 3.6: L fold up sampler

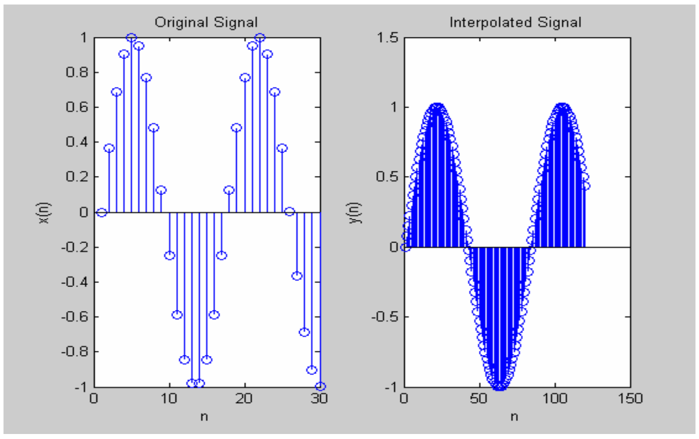


Figure 3.7: Interpolated signal L = 4

3.6 Polyphase decomposition:

Polyphase representation is very useful in multirate processing for the following reasons.

1. To perform all computations at the lowest rate permissible with in the given context.
2. It reduces the speed requirement on the processors.
3. It leads computationally efficient decimation \interpolation filter as well as filter banks

3.7 Polyphase decimator:

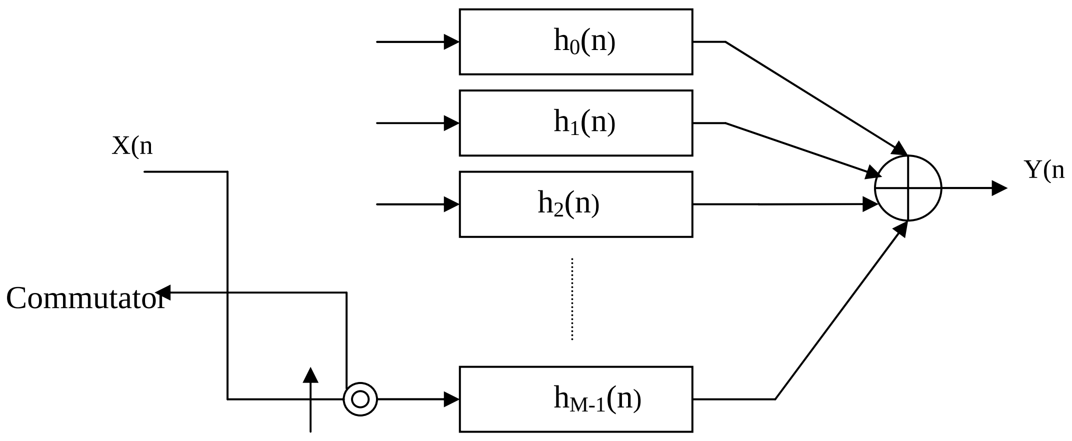


Figure 3.8: M to 1 polyphase decimator

A set of *N* prototype filter coefficients are mapped to the *M* polyphase sub-filters according to Equation 3.3

$$h_i(n) = a(i + M r) \quad \text{-------------------------------------} \quad (3.3)$$

Where i = 0,1,2,……… (M-1) and r = 0, 1, 2,……….. (N-M+i)

The polyphase segments are accessed by delivering the input samples *x*(*n*) to their inputs via an input commutator which starts at the segment index and decrements to index 0. After the commutator has executed one cycle and delivered *M* input samples to the filter, a single output is taken as the summation of the outputs from the polyphase segments. The output sample rate is fs'=fs/M where fs is sample rate of the input data stream

We observe that each of the polyphase segments is operating at the low output sample rate (compared to the high input sample rate) and a total of operations are performed per output point. . The polyphase decimator provides support for single-channel operation only.

The computational requirement is N multiplications and (N-1) additions per output sample. The arithmetic units are operative at all instants of the output sampling period, which is M times that of input sampling period.

CHAPTER 4
SYSTEM DESIGN

Main objective is to develop generic IP- core for DA based FIR filter for different filter configurations.

Filter configurations:

- ***Half band filter***
- ***Polyphase decimator***

Programmable parameters are:

- ***Input bit precision***
- ***Coefficient width***
- ***Number of tapes (order of filter)***
- *Decimation factor*

4.1 DA based FIR FILTER

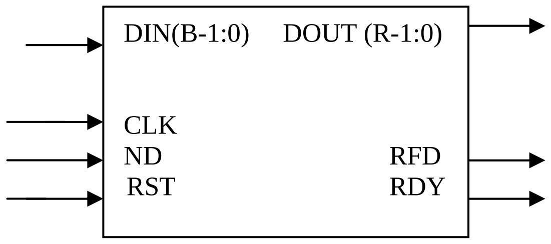


Figure 4.1: FIR filter symbol

The basic FIR filter core is a single-rate (input sample rate = output sample rate) finite impulse response filter. Figure 4.1 shows the schematic symbol for a single-channel instance of this module. This filter symbol is applicable to both the filter configurations. Filter input data is supplied on the **DIN** port and filter output samples are presented on the **DOUT** port. The **CLK** signal is the bit-rate clock for the core, and is recognized as being different (higher frequency) to the input signal sample frequency. The **ND, RDY,** and **RFD** signals are filter interface/control signals that permit a simple and efficient data-flow style interface for supplying input samples and reading output samples from the filter. The core interface signals are discussed.

Name	Direction	Description
DIN [B-1:0].	Input	FILTER INPUT DATA SAMPLE — B-bit wide filter input sample.
ND	Input	NEW DATA (Active High) — When this signal is asserted, the data sample presented on the *DIN* port is loaded into the PSC and an inner-product Computation is started.
RST	Input	Synchronous reset (Active High). Asserting *RST* synchronously with *CLK* resets the filter internal state machines.
RDY	Output	FILTER OUTPUT SAMPLE READY (Active High) Indicates that a new filter output sample is available on the *DOUT* port.
RFD	Output	READY FOR DATA — (Active High) Indicates when the final bit of the current data sample is about to be processed and new data may be supplied to the filter.
DOUT[R-1:0]	Output	FILTER OUTPUT SAMPLE R-bit-wide output sample bus for the FIR, half-band and interpolated filters. R depends on the filter parameters (data precision, coefficient precision, number of taps and coefficient optimization selection) .

Table 4.1: FIR filter port names and port functional definitions.

4.2 SPECIFICATION

Number of Taps: The number of filter taps. For a symmetric impulse response (either even or odd symmetric) the number of filters taps is between 2 and 1024 inclusive. For a nonsymmetrical coefficient set the range is 2 to 1024 inclusive.

Impulse Response: Indicates structure present in the coefficient set. The user may specify a symmetric, negative (odd)-symmetric or nonsymmetrical impulse response.

Coefficient Width: The bit precision of the filter coefficients. This is an integer value between 1 and 32 inclusive.

Coefficient Data Type: The coefficient data can be specified as either signed or unsigned. When the signed option is selected, conventional two's complement representation is assumed.

COE File Name: Coefficient file name. This is the file of filter coefficients. The file has a *.Coe* extension.

Coefficient Reload: When the Fixed radio button on the Coefficient Reload panel is selected, the filter Core is generated without a coefficient reload interface. When the Reload able button is selected, a coefficient reload interface is provided on the Core.

Optimize Coefficients: The look-up tables employed in the filter mechanization can be optimized to minimize the amount of FPGA logic fabric employed by the core. The optimization is data (filter coefficient set) dependent.

Load Coefficients: The filter coefficients are supplied in a coefficient or *coe* file. This is an ASCII file with a ".Coe" extension.

Show Coefficients: Selecting this tab displays the filter coefficient data.

Input Data Width: The precision (in bits) of the filter input data samples. The input sample precision is an integer value between 1 and 32 inclusive.

Input Data Type: The filter input data can be specified as either signed or unsigned. The signed option employs conventional two's complement arithmetic.

Output Options: The filter output bus can be registered or unregistered. When the registered output option is selected, the filter output bus *DOUT* is maintained at the core output between successive assertions of *RDY*. In the unregistered mode, the output sample is valid only when *RDY* is active. At other times, the port changes on successive clock cycles.

4.3 INTERFACE, CONTROL AND TIMING

All of the filter classes employ a data-flow style interface for supplying input samples to the core and for reading the filter output port. *ND (New Data*), *RFD (Read For Data)* and *RDY (Ready)* are used to co-ordinate *I/O* operations. The *ND* input signal is used for loading a new input sample into the filter. It is effectively used internally as a clock enable, and the actual sample load operation occurs on the rising of the clock (*CLK*). When the core is ready to accept a new input sample, the *RFD* signal is asserted. When a new output sample is available, *RDY* is asserted for a single clock period. When the registered output option is selected, the output sample remains valid between successive assertions of *RDY.*

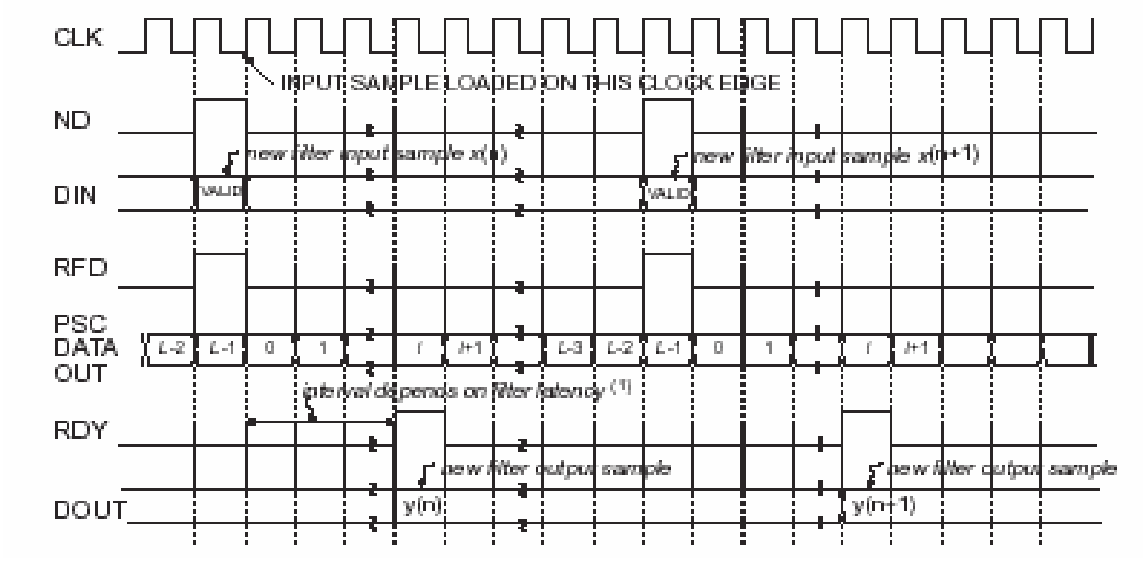

CLK
INPUT SAMPLE LOADED ON THIS CLOCK EDGE
ND
new filter input sample x(n)
new filter input sample x(n+1)
DIN
VALID
RFD
PSC DATA OUT
L-2
L-1
0
1
i
i+1
L-3
interval depends on filter latency (1)
RDY
new filter output sample
DOUT
y(n)
y(n+1)

Figure 4.2: Ideal Timing diagram for Single-Channel FIR filter

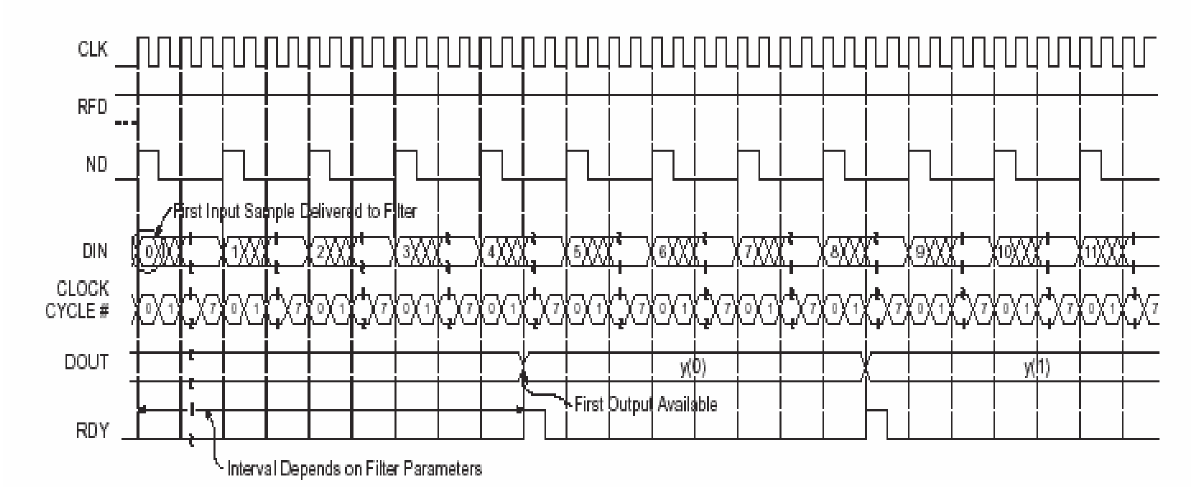

CLK
RFD
ND
First Input Sample Delivered to Filter
DIN
CLOCK CYCLE #
DOUT
y(0)
y(1)
First Output Available
RDY
Interval Depends on Filter Parameters

Figure 4.3: Ideal Timing diagram for polyphase decimator

CHAPTER 5
DESIGN OF DA FIR FILTER

5.1 GUI of DA Based FIR Filter:

Filter has number of programmable inputs like coefficients width (C), input bit precision (B), number of Taps (N). Filter design is catered to the design of half band filter, polyphase decimator. The multiple choices of filter type and filter inputs can be sophisticatedly selected by common GUI created in MATLAB 7.0. The mathematical modeling of filter design is created through MATLAB programming. The golden reference for filters under all input condition and LUT coefficients are computed through GUI.

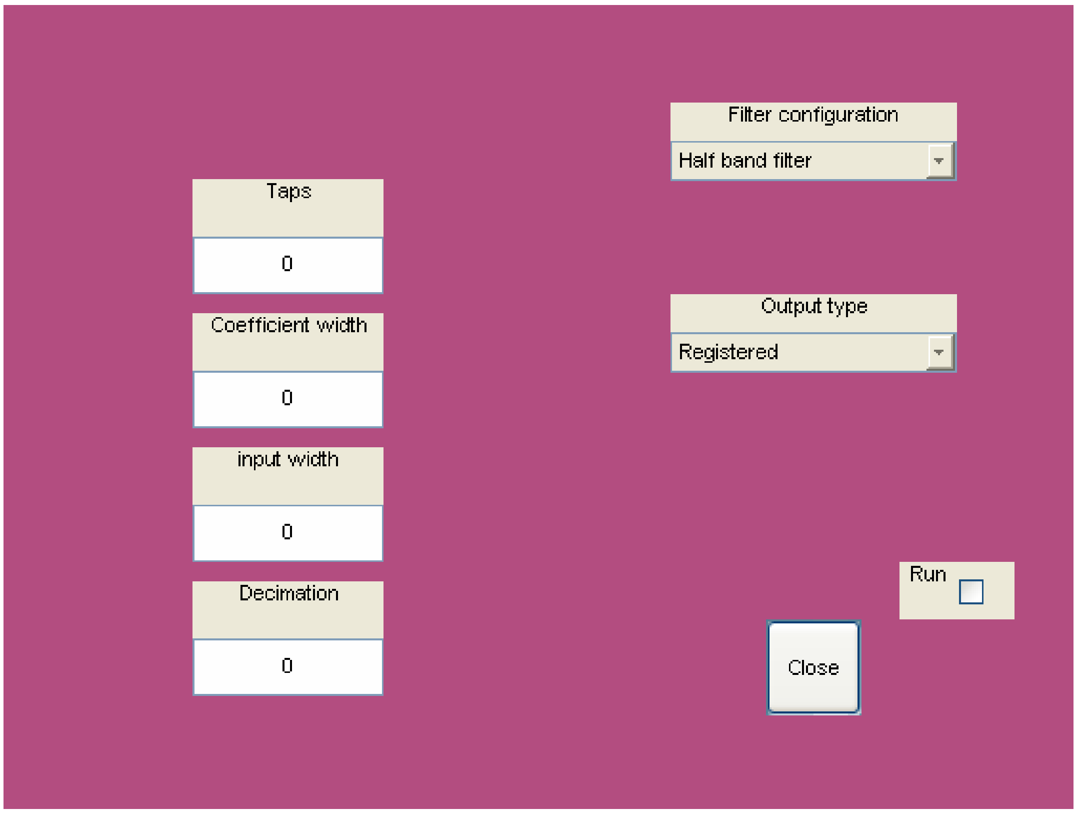

Taps: 0
Coefficient width: 0
input width: 0
Decimation: 0
Filter configuration: Half band filter
Output type: Registered

Figure 5.1: A Graphical user interface for DA FIR filter

This GUI also generates the required VHDL files for the DA based Half Band filter and Polyphase decimator. These files are stored into required project folder for Modelsim simulation.

5.2 DA based half band filter design:

Modular approach is followed for design implementation. The filter design have following main modules. Figure 5.2 shows block Level Schematic and signal flow diagram in design.

- **CONTROL SIGNAL**
- **PARALLEL TO SERIAL CONVERTER**
- **TIME SKEW BUFFER**
- **LOOK UP TABLE**
- **SCALING ACCUMULATOR**

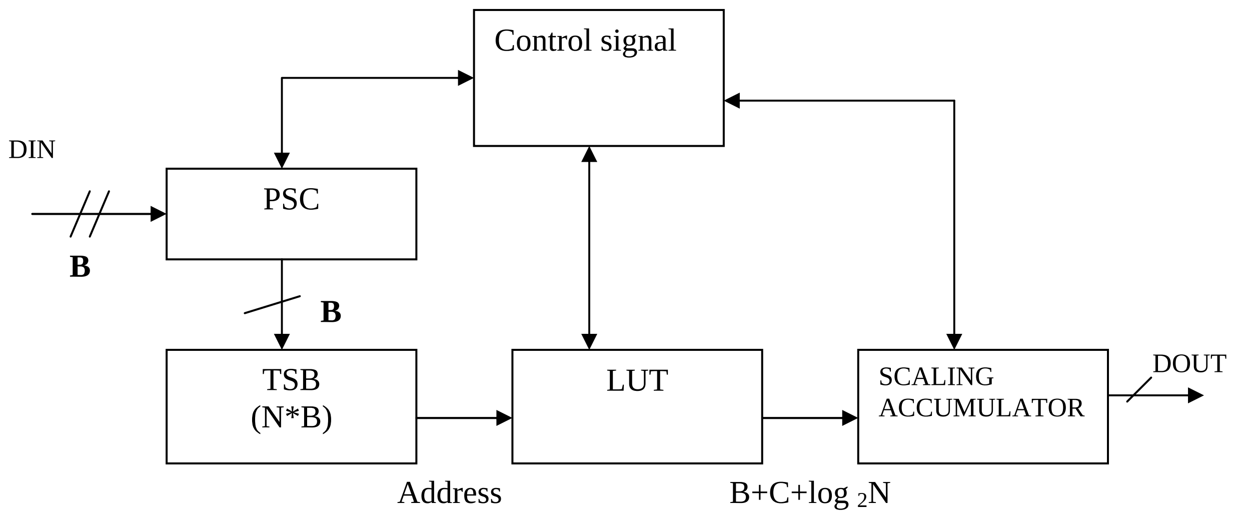


Figure 5.2: Block Level Schematic of design

B: Input bit precision

N: Number of Taps

C: Coefficient width

DOUT: Output bit precision

5.3 Brief description of basic modules:

Control signal:

This module generates the necessary control signals for driving different modules. Basically it generates divide by B counter that is required for the operation of different module.

Parallel to serial converter:

It accepts parallel data input and gives serial output.

Time skew buffer:

it is a shift register that stores all the input samples in bit serial format. The nodes in the cascade connection of TSBs are used as address inputs to a look-up table.

Look up table:

This LUT stores all possible partial products over the filter coefficient space.

Scaling accumulator:

It comprises of adder/sub tractor, register and a delay element. In this module inputs are given from the LUT and give the output after B clock cycles.

5.4 Design Verification Methodology:

Golden references for selective input condition are generated using mathematical modeling and used to compare with simulation results. Verification is completed in various phases. First individual modules are tested by applying different test vectors. Then integral functional verification is performed. The verification methodology is divided into following phases.

5.4.1 Mathematical Modeling:

Algorithm written in Matlab code (file: gen_hbf1.m) generates the golden references. Algorithms generate the LUT, address, LUT output, final output of filter. It takes numeric value of number of tap, input bit precision, coefficient width and data samples.

5.4.2 Preliminary Test Bench:

Preliminary test bench for simulation is used to generate the stimuli required for functional simulation of the design and compare all the output generated from MATLAB. This ensures the error free design before implementation on FPGA

5.4.3 Hardware testing:

After testing through simulation, design is downloaded on FPGA, and output is analyzed using chipscope analyzer. Results are again compared with Matlab and modelsim results.

CHAPTER 6
RESULTS AND ANALYSIS

6.1 FIR filter results

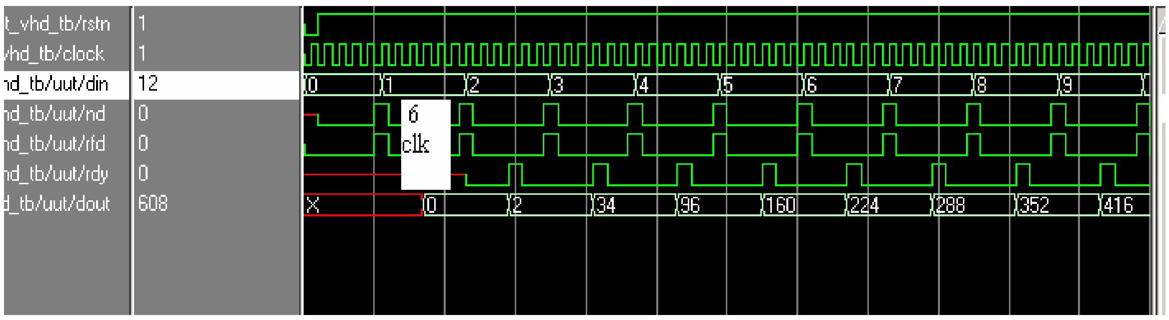


Figure 6.1.a: 4 tap - 6 bit ramp response

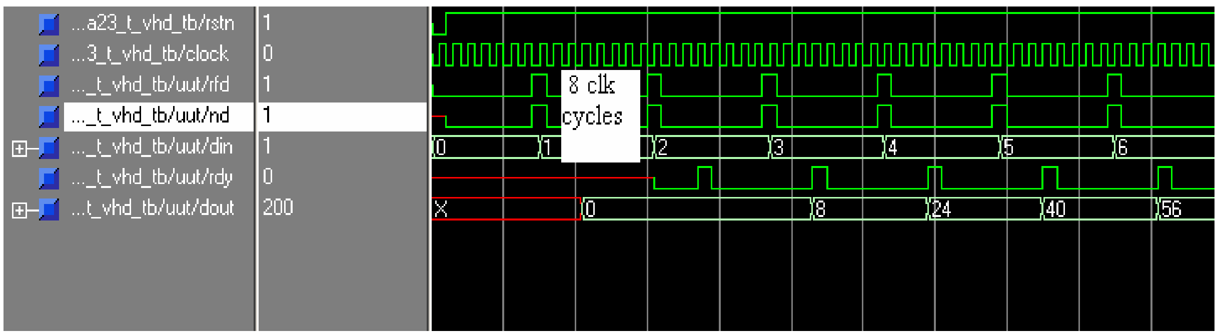


Figure 6.1.b:4 tap - 8 bit ramp response

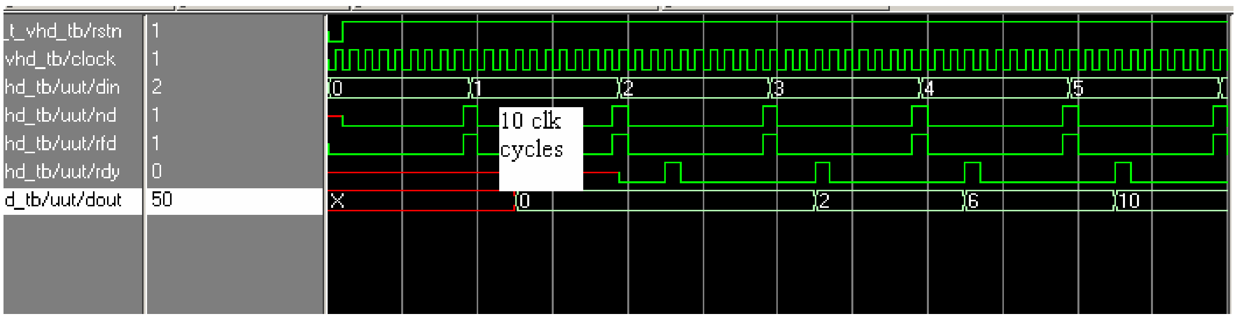


Figure 6.1.c: 4 tap -10 bit ramp response

The above figures show that the relationship between the three control signals. The no. of clock cycles between the successive pulses of RFD, ND and RDY is equal to the input bit precision.

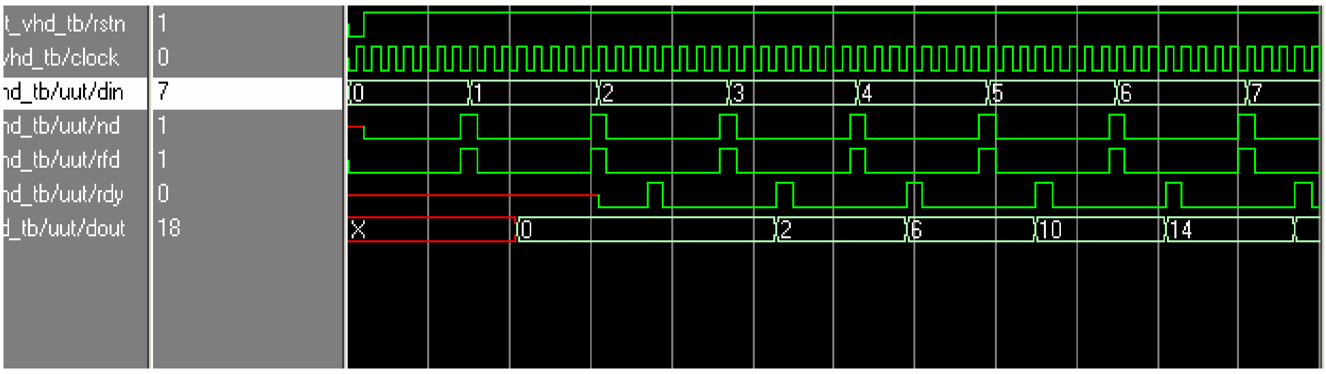


Figure 6.1.d: 4 tap -8 bit ramp response

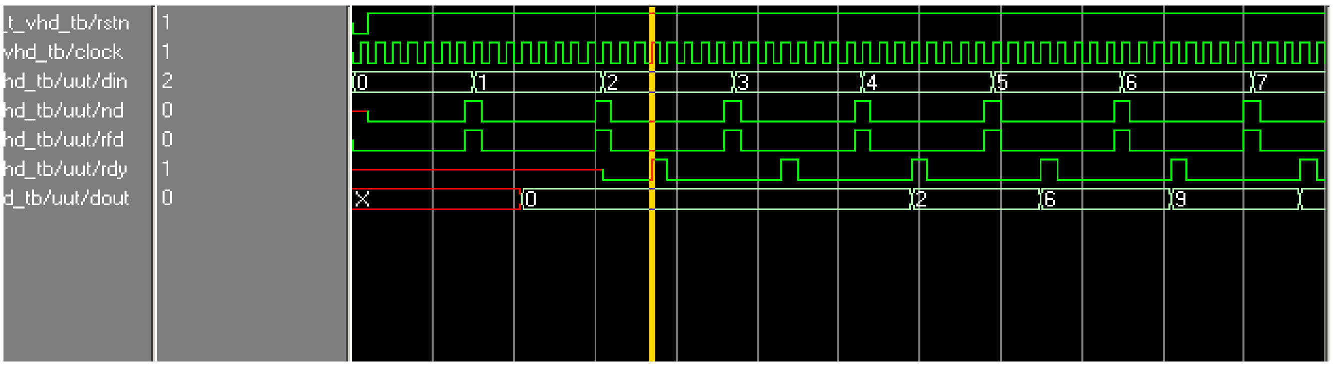


Figure 6.1.e: 6 tap- 8bit ramp response

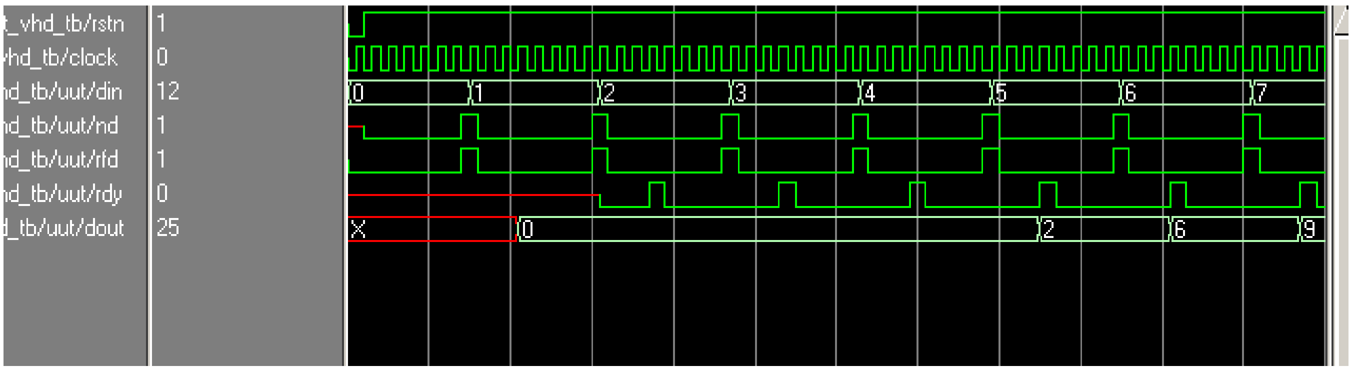


Figure 6.1.f: 8tap -8 bit ramp response

Here in the above figures input bit precision is kept same while the no. of tapes is varied. We can see that the throughput is constant for all the filter length.

6.2 Half band filter results

6.2.1 Matlab results

Specification of half band filter:

Input bit precision- 4bit

***Coefficient width* -** 4 bit

***Number of tapes* -**31 tap

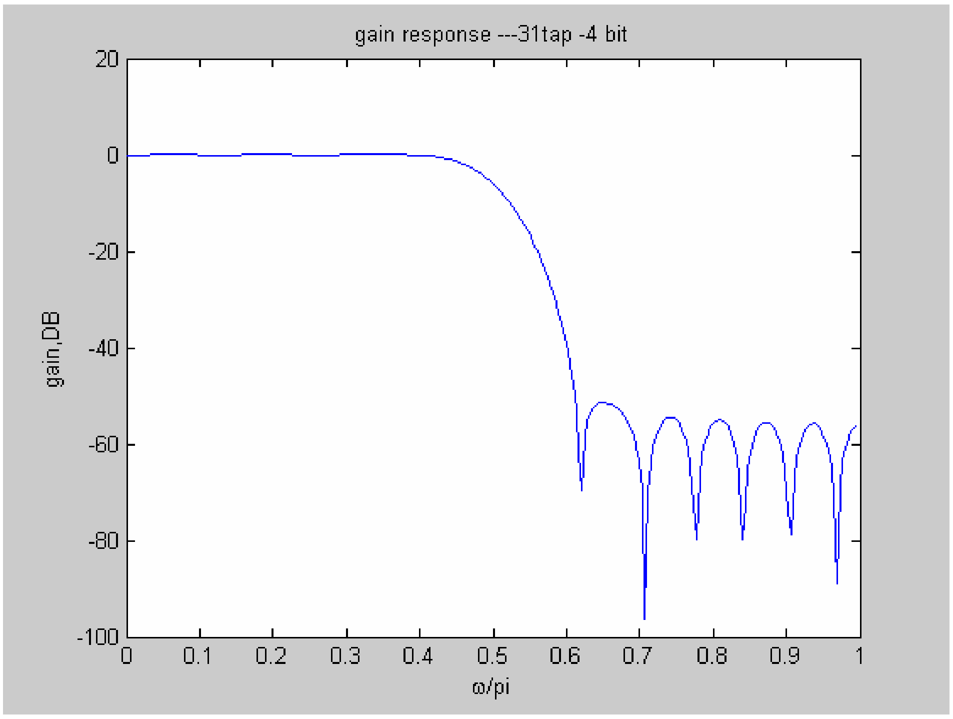


Figure-6.2.1.1: Gain response of 31 tap-4bit Half Band Filter

INPUT (DIN)	MATLAB OUTPUT(DOUT)	VHDL OUTPUT (DOUT)
1	0	0
0	0	0
0	3	3
0	8	8
0	3	3
0	0	0
0	0	0

Table 6.2.1.a: 7 tap-4bit Impulse response

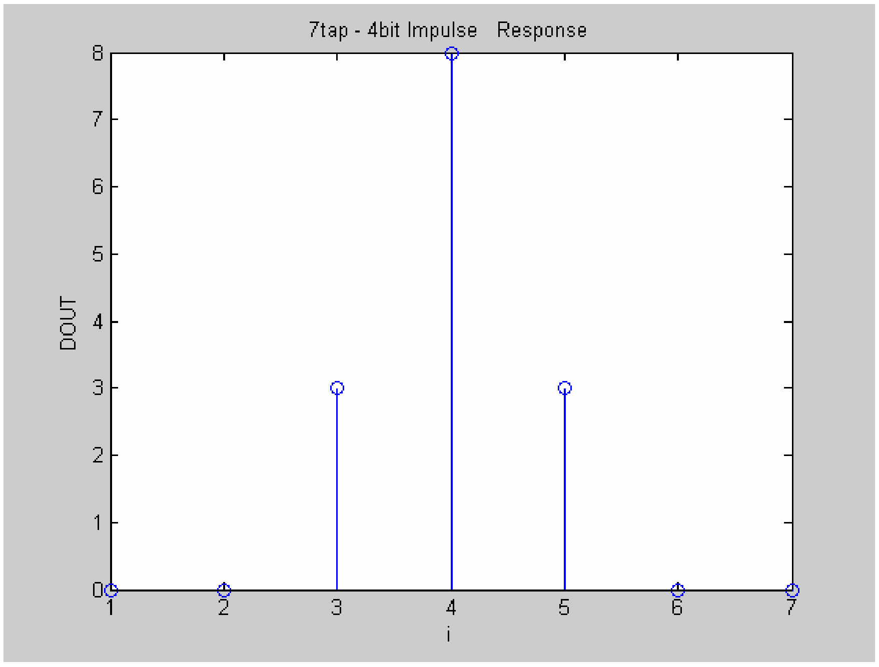


Figure 6.2.1a: 7 tap-4bit Impulse response in MATLAB

INPUT (DIN)	MATLAB OUTPUT(DOUT)	VHDL OUTPUT (DOUT)
1	0	0
1	0	0
1	3	3
1	11	11
1	15	15
1	15	15
1	15	15
1	15	15
1	15	15
1	15	15
1	15	15
1	15	15
1	15	15
1	15	15
1	15	15

Table 6.2.1.b: 7tap –4bit step response

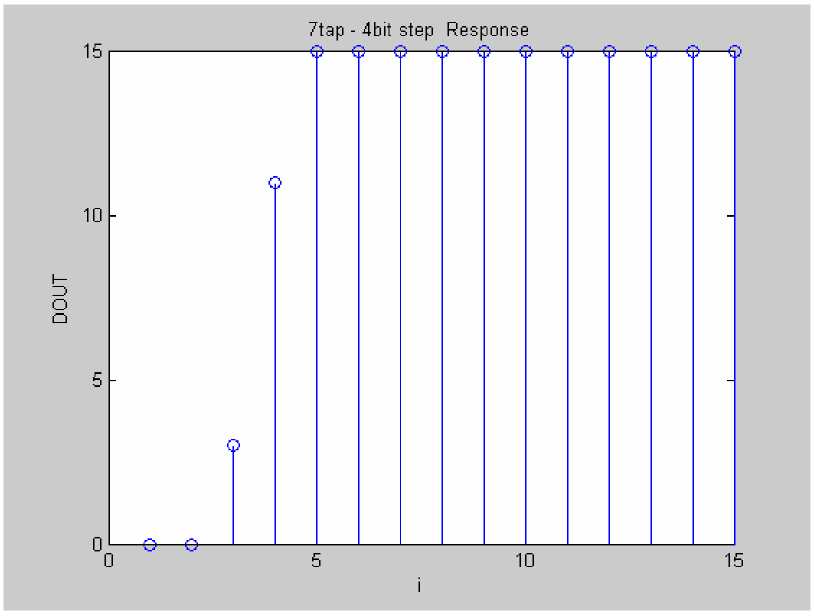


Figure 6.2.1.b: 7tap –4bit step response in MATLAB

INPUT (DIN)	MATLAB OUTPUT(DOUT)	VHDL OUTPUT (DOUT)
1	0	0
2	0	0
3	4	4
4	16	16
5	31	31
6	46	46
7	62	62
8	79	79
9	93	93
10	108	108
11	124	124
12	141	141
13	157	157
14	170	170
15	186	186

Table 6.2.1.c: 7 tap -4bit ramp response

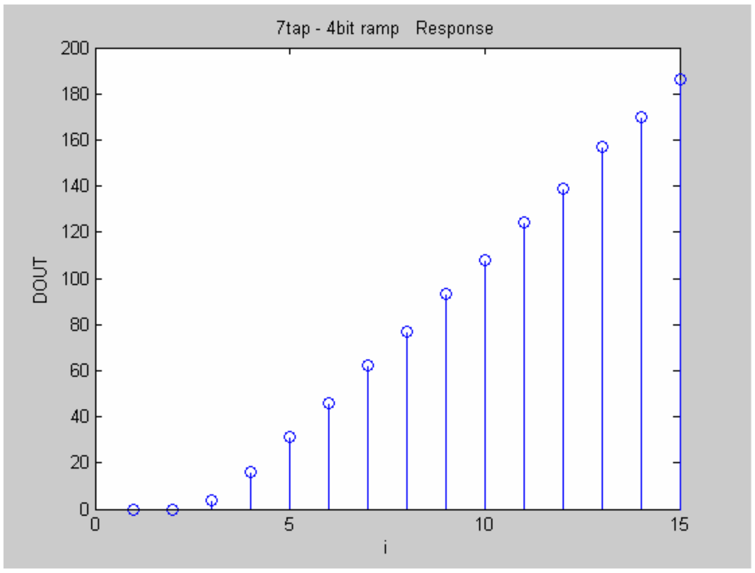
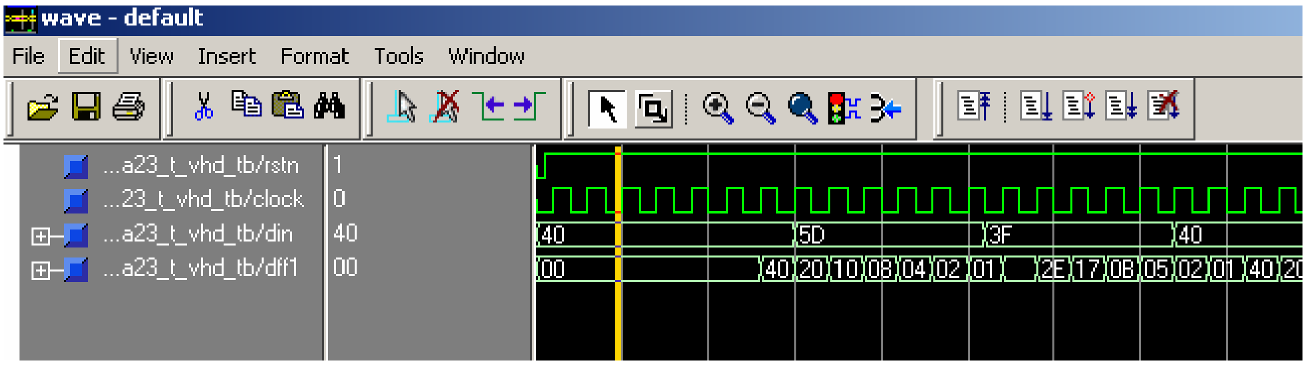


Figure 6.2.1.c: 7 tap -4bit ramp response in MATLAB

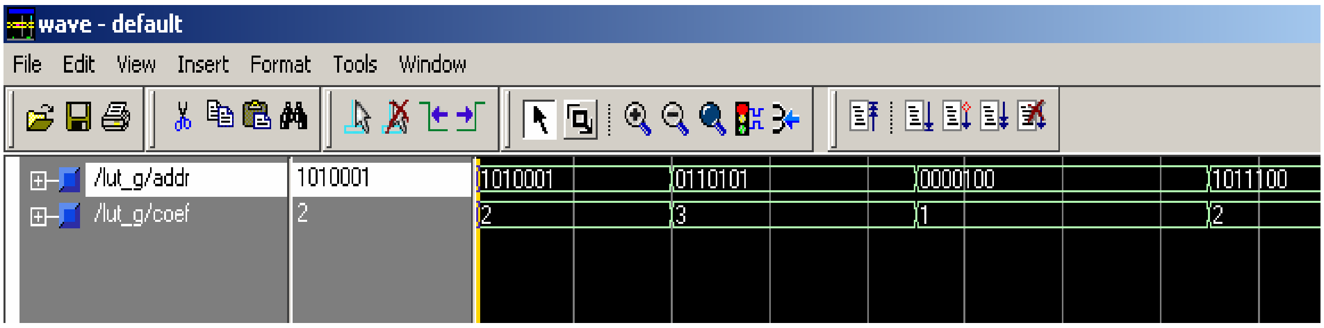
6.2.2 Half Band filter Results in Modelsim

MODELSIM simulation results for individual blocks


wave - default
File Edit View Insert Format Tools Window
40 5D 3F 40
00 40 20 10 08 04 02 01 2E 17 0B 05 02 01 40 20

```
...a23_t_vhd_tb/rstn    1
...23_t_vhd_tb/clock    0
...a23_t_vhd_tb/din     40
...a23_t_vhd_tb/dff1    00
```

Figure 6.2.2.a: parallel to serial converter



```
/lut_g/addr    1010001
/lut_g/coef    2
```

Figure 6.2.2.b: Look up table

Half Band filter Results in Modelsim

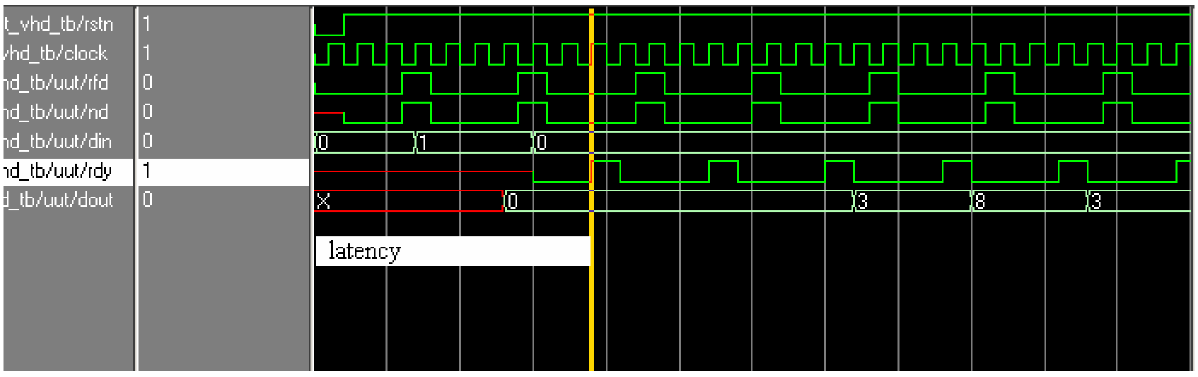


Figure 6.2.2.c: 7 taps- 4bit Impulse Response

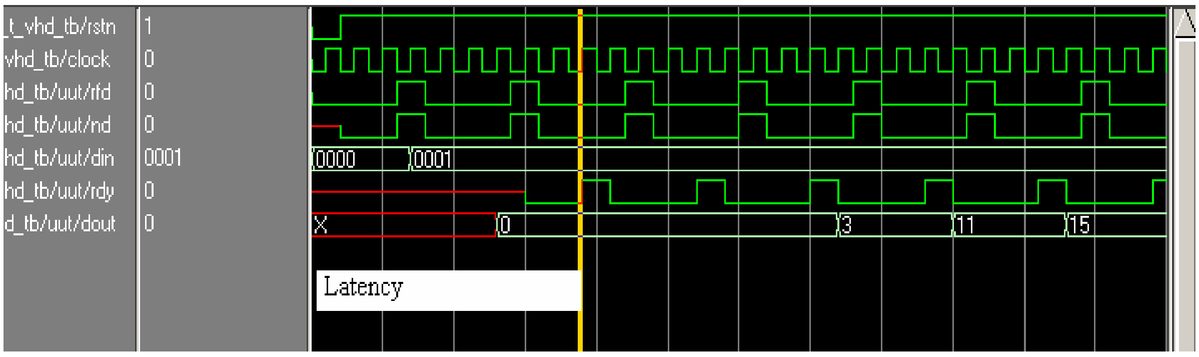


Figure 6.2.2.d: 7 tap-4 bit Step Response

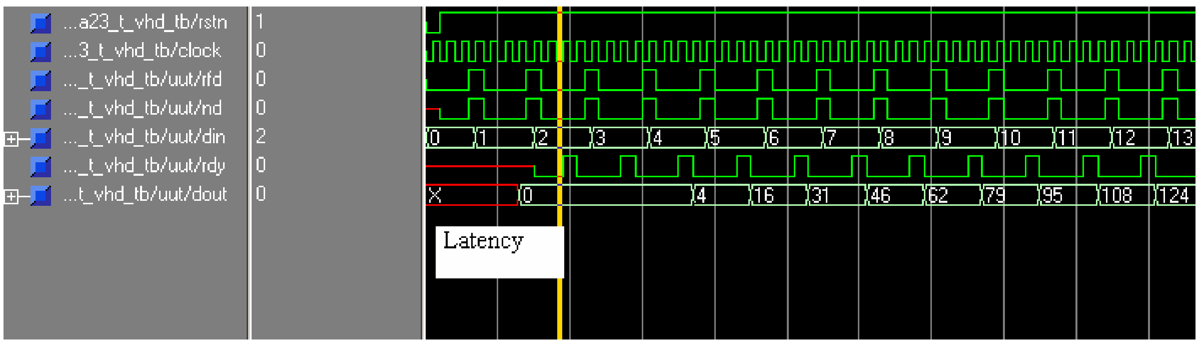


Figure 6.2.2.e: 7 tap- 4bit Ramp response

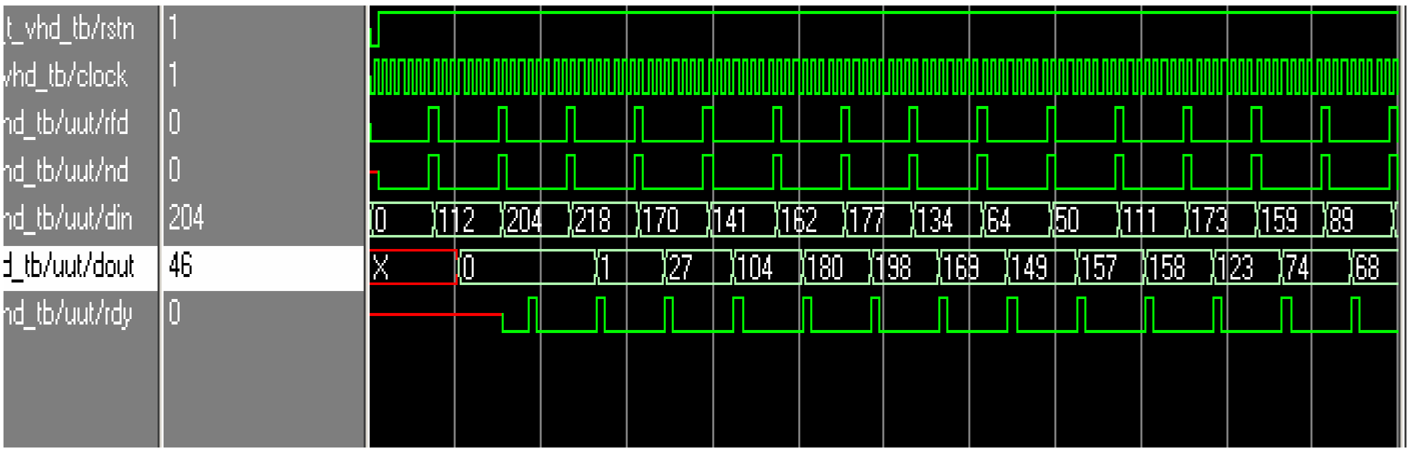


Figure 6.2.2.f: 7 tap- 8 bit Sine response

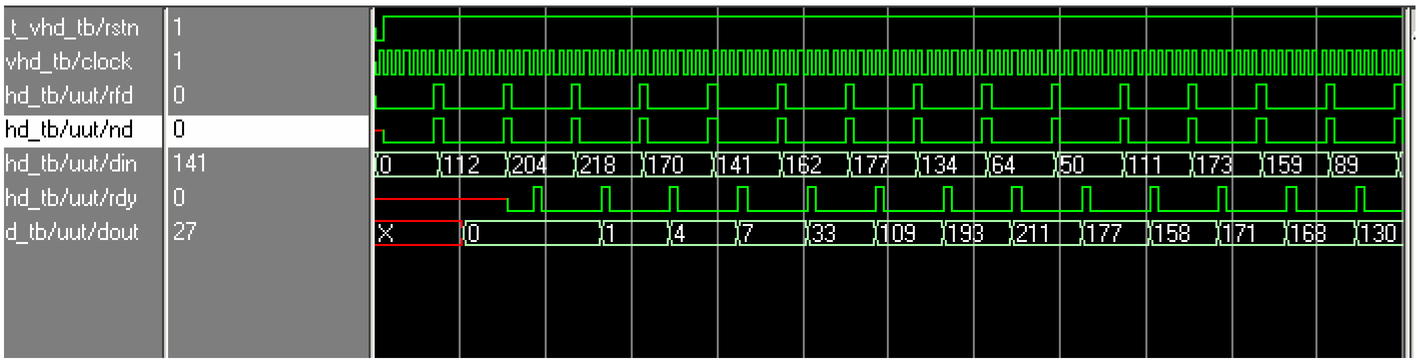


Figure 6.2.2.g: 11 tap- 8 bit Sine response

6.2.3 Chipscope Results

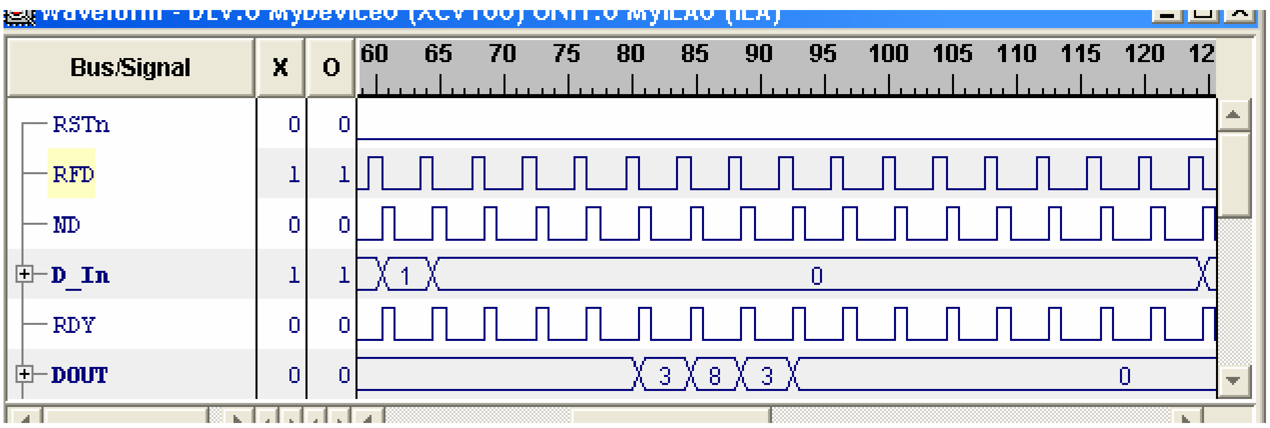


Figure 6.2.3.a: 7 tap-4 bit Impulse Response

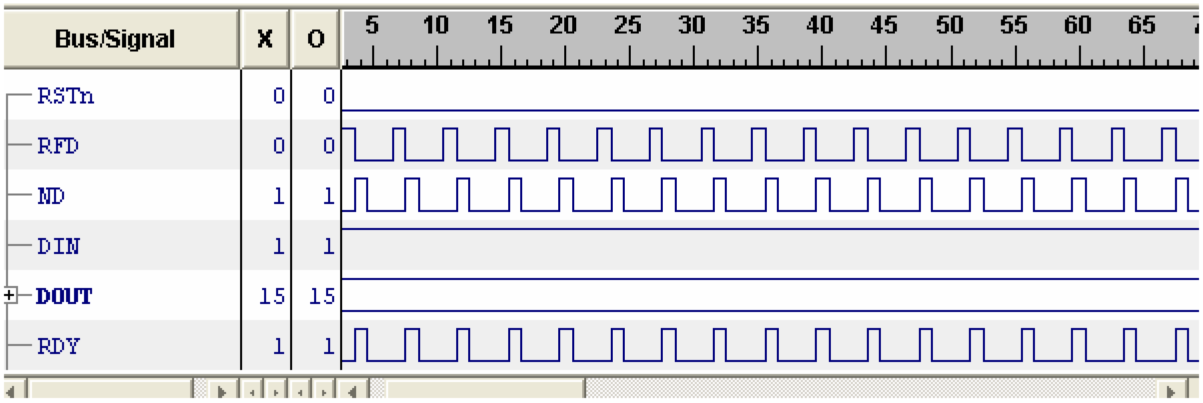


Figure 6.2.3.b: 7 tap -4 bit Step Response

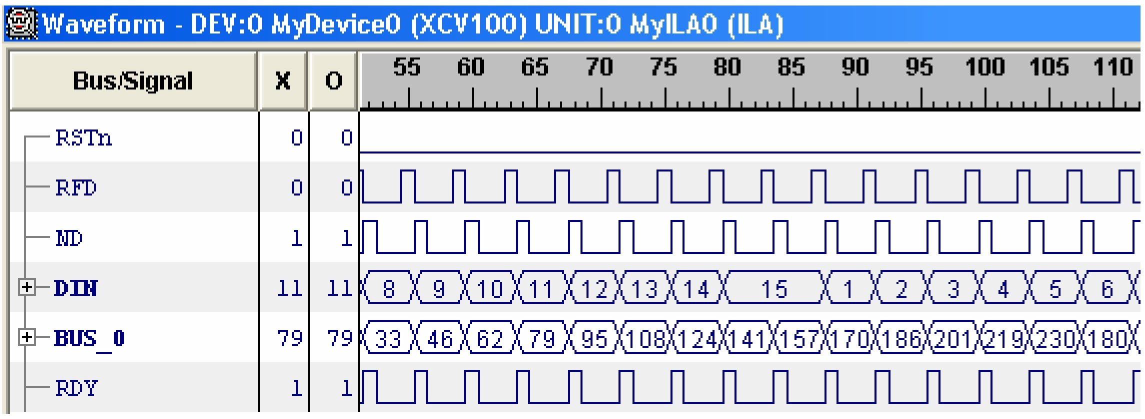


Figure 6.2.3.c: 7tap-4bit Ramp Response

6.3 polyphase decimator results

6.3.1. Matlab results:

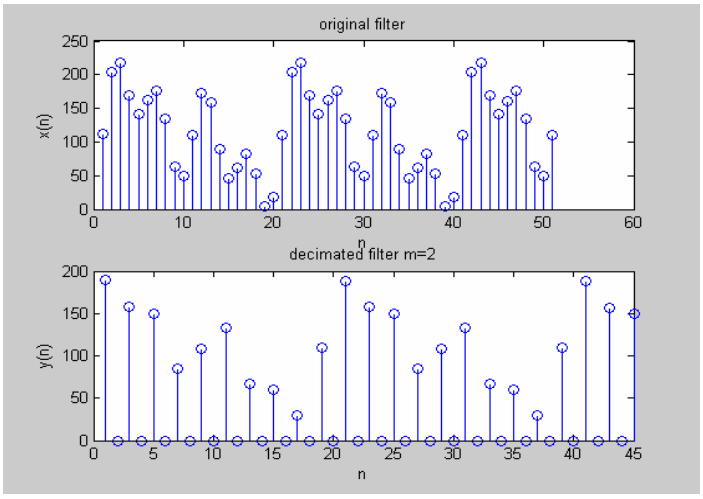


Figure 6.3.1.a: 4 tap-8 bit, decimation factor M=2

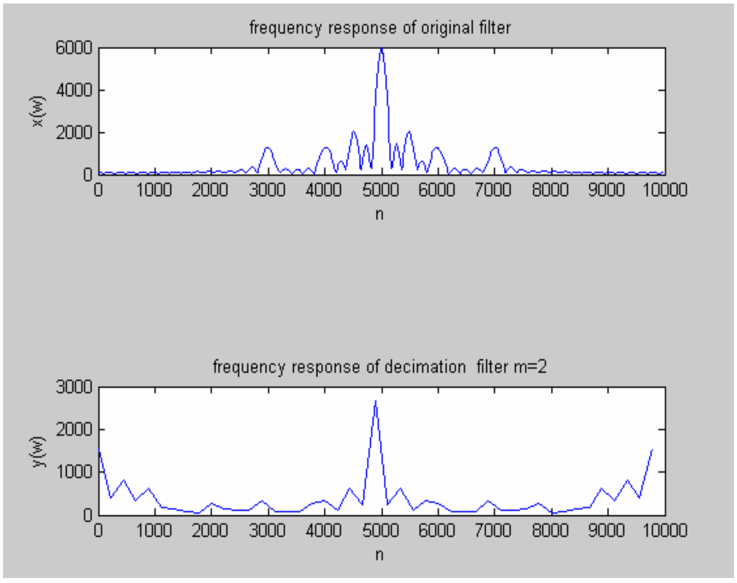


Figure 6.3.1 .b: 4 tap-8 bit, decimation factor M=2

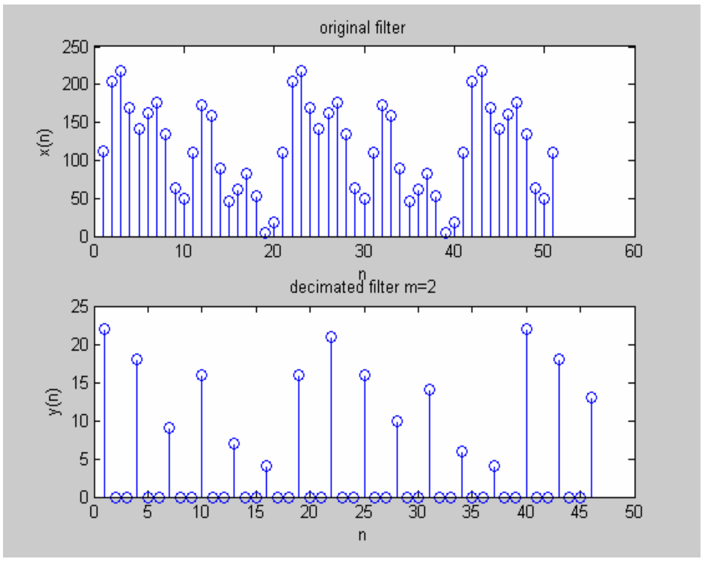


Figure 6.3.1.c: 4 tap-8 bit decimation factor M=3

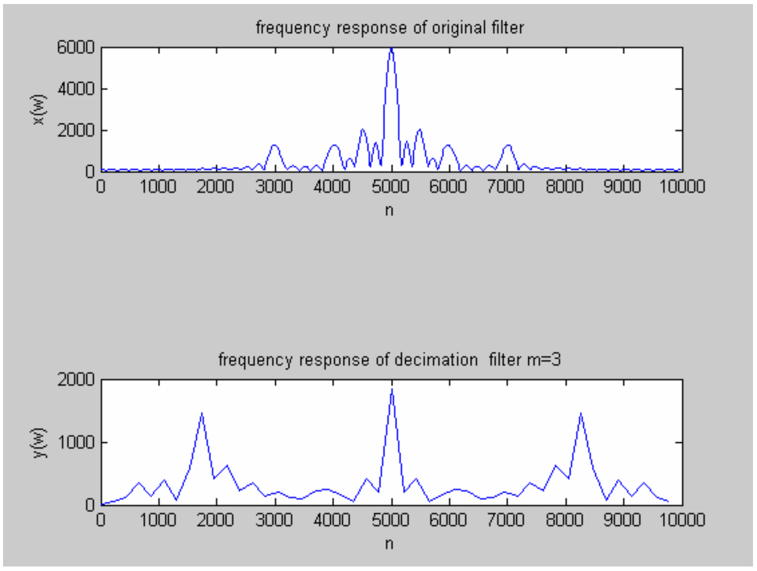


Figure 6.3.1.d: 4 tap-8 bit decimation factor M=3

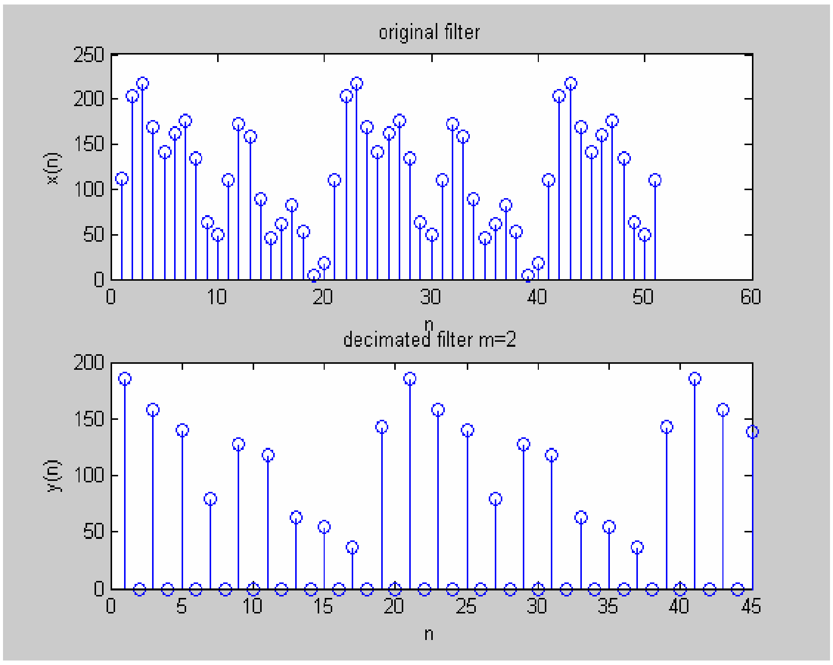


Figure 6.3.1.e: 5 tap-8 bit, decimation factor M=2

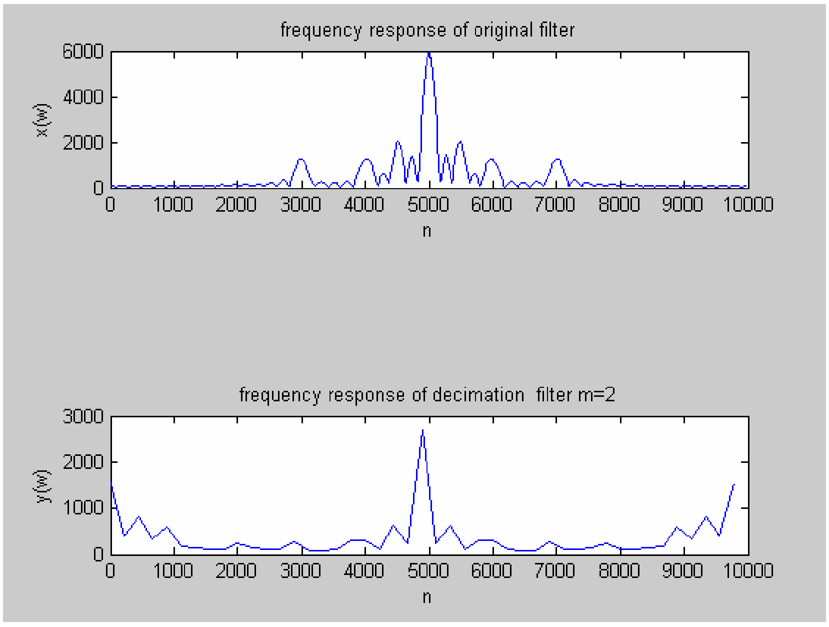


Figure 6.3.1.f: 5 tap-8 bit, decimation factor M=2

6.3.2. Modelsim results:

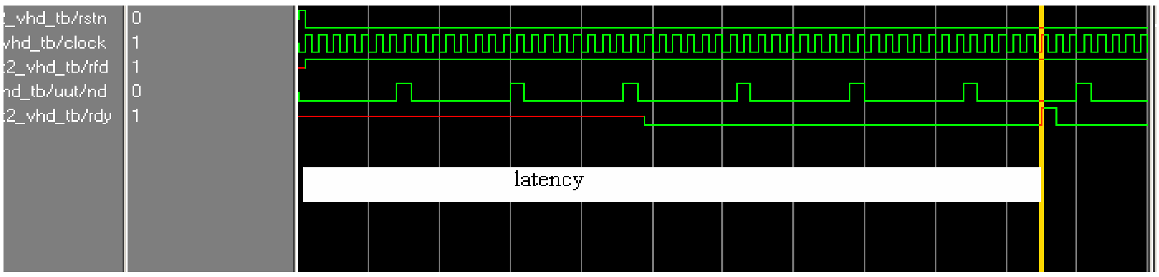


Figure 6.3.2,a: Timing Diagram of Polyphase Decimator M = 4,B = 8,N = 4

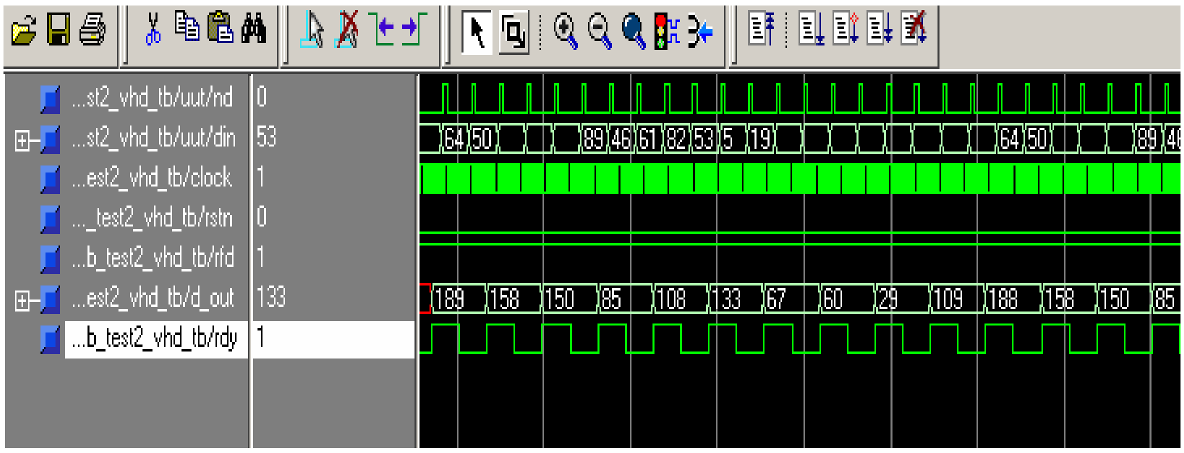


Figure 6.3.2 .b: 4 tap-8 bit decimation factor M=2

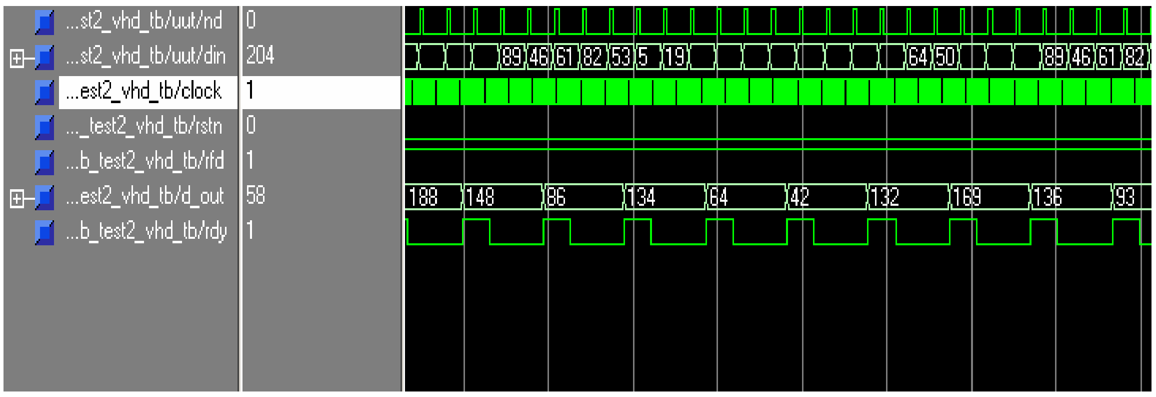


Figure 6.3.2.c:4 tap-8 bit decimation factor M=3

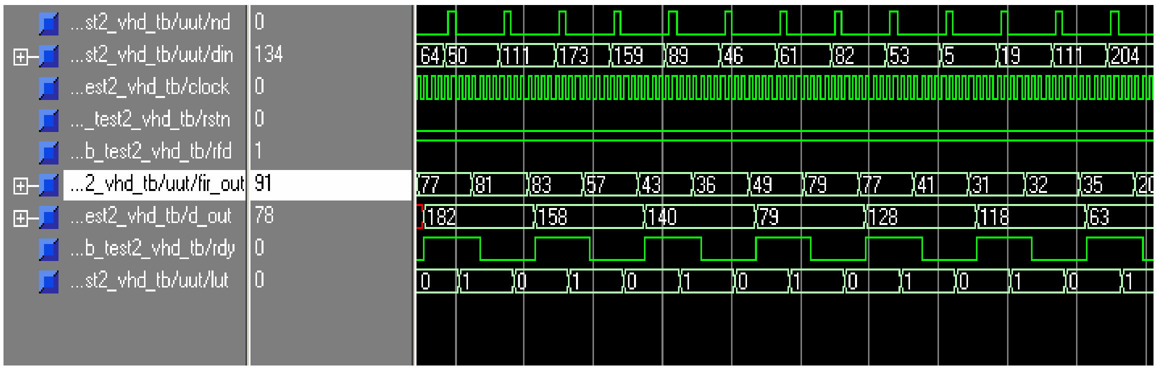


Figure 6 .3.2.d: 5 tap -8 bit decimation factor M = 2

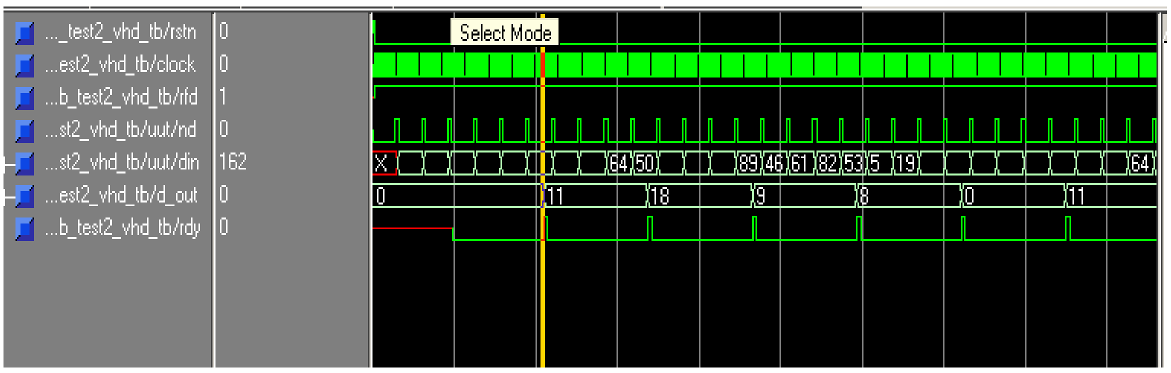


Figure 6 .3.2.e: 4tap -8 bit decimation factor M = 4

6.3.3. Chipscope Results:

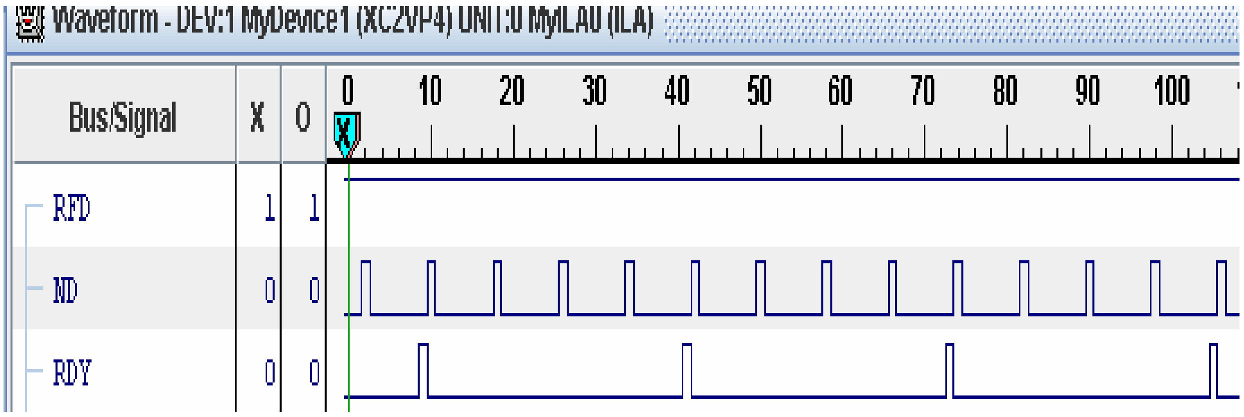


Figure 6 .3.3.a: Timing Diagram of 4tap -8 bit decimation factor M = 4

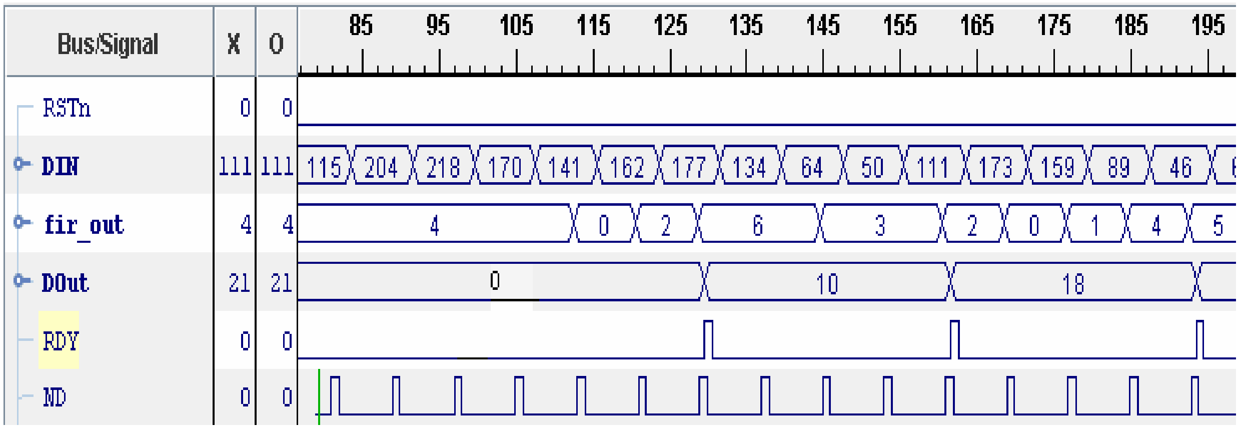


Figure 6 .3.3.b: 4tap -8 bit decimation factor M = 4

6.4 Performance analysis

6.4.1 Comparison of half band filter for different configuration:

Hardware used	**Filter size(in taps)-I/P width(in bits**			
	7tap - 8bit	**11tap-8 bit**	**7 tap -4bit**	**11 tap-4 bit**
Number of Slices	49	57	30	38
Number of Slice Flip Flops	72	83	52	79
Number of 4 input LUTs	51	75	44	64
Number of bonded IOBs	25	25	22	22
Number of Block RAMs	1	1	1	1
Number of GCLKs	1	1	1	1
Speed(MHz)	117.12	109.325	111.166	99.325

Table 6.4.1 Comparison table of half band filter

6.4.2 Comparison of Half band filter design with XILINX DA-core

Hardware used	**Filter size(in taps)-I/P width(in bits**			
	Half band filter		**XILINX DA-core**	
	7tap - 8bit	**11tap-8 bit**	**7 tap -8 bit**	**11 tap-8 bit**
Number of Slices	49	57	34	36
Number of Slice Flip Flops	72	83	61	67
Number of 4 input LUTs	51	75	48	51
Number of bonded IOBs	25	25	30	21
Number of Block RAMs	1	1	NIL	NIL
Number of GCLKs	1	1	1	1
Speed(MHz)	117.12	109.325	135.566	132.566

Table 6.4.2 Comparison of Half band filter design with XILINX DA-core

6.4.3 Comparison of Polyphase Decimator

Hardware used	**Decimation factor(M) -taps(N)-i/p bit precision(B)**		
	M=2 N=4 B=8	M=3 N=4 B=8	M=4 N=4 B=8
Number of Slices	64	69	72
Number of Slice Flip Flops	96	98	102
Number of 4 input LUTs	83	105	108
Number of bonded IOBs	29	29	29
Number of GCLKs	1	1	1
Speed	114.916 MHz	112.916 MHz	111.773 MHz

Table 6.4.3 Comparison of polyphase decimator

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 Conclusion:

The basics of distributed Arithmetic Algorithm and DA based FIR filter has been studied. A generic IP-core has been developed for Half Band filter and Polyphase Decimator. This IP –core has been implemented on Xilinx FPGA XCV100 and results are captured through chipscope for various parameters, and compared with golden references and found matched.

It is shown that DA based FIR filter is independent of filter length. It depends on input bit precision. For small number of filter length this algorithm provides better and speed efficient solution. As LUT size grows (2^N) with filter length, there is tradeoff in speed at higher number of filter length. As the number of tapes is increased ,more logic resources are consumed.

The design of half band filter and polyphase decimator is done in VHDL.The design approach is modular and it has fully programmable parameters. This design is platform independent, so it can be used for any environment. This design is tested with different number of configuration. Testing methodology include the golden references generated through MATLAB GUI, comparison of MODELSIM results with golden references and finally through chipscope. Comparison of design is done with Xilinx DA core. The hardware utilization is almost similar in design however more optimization is required.

7.2 Future Work:

Here mainly three parameters are kept programmable.other programmable parameters can be included to make the design more flexible.

Implementation of other filter configurations that are supported by Xilinx core.

Optimizing the code to make it speed and Hardware efficient.

REFERENCES

[1] S.A.White, "Application of Distributed Arithmetic to digital signal Processing,"IEEE ASSP Magazine, vol.6 (3), pp.4-19, July 1989.

[2] A.Peled and B.Liu , "A New Hardware Realization of Digital Filters," IEEE Trans On Acoustic, speech, signal processing vol - 22 pp 456-462 Dec. 1974.

[3] Uwe Mayer-Base, "Digital signal processing with field programmable gate Arrays" Springer international edition 2003.

[4] Infector and Jervis, "Digital signal processing –a practical approach", Addison-Wesley 1999.

[5] P.P Vaidyanathan,"Multirate Systems and Filter Banks", Pearson education 2004.

[6] Xilinx/logic core/distributed arithmetic FIR filter v 9.0.

APPENDIX -A

MATLAB CODING FOR GENERIC HALF BAND FILTER

```
% program for generic half band filter

Clc;
Clear all;

% programmable parameter


%       No. of tapes                  -----     N
%       Input  bit  precision          ------    B
%       coefficient bit precision     ------    c
N= input('type the filter length');
l=input('type  the value of l=');
c=input('type  the value of coefficient width');
B=input('type  the value of input bit width');


%design of  half band filter
k=(N-1)/2;
n=-k:k;
b=sinc(n/l)/l;
display(b);


win=hamming(N);
 c1=b.*win';
 %impulse response
x=1:length(c1);
figure(1)
stem(x,c1);
title('impulse response');
```

```
%gain response  (unquantized   coefficient)

[h,w]=freqz(c1,1,256);
g=20*log10(abs(h));
 %c=quantize(cq,c1);
 qc=round(c1*(2^(c-1)-1));
[h,w]=freqz(qc,1,256);
gg=20*log10(abs(h));

% gain  response (quantized   coefficient)
figure(2)
plot(w/pi,g,'b',w/pi,gg,'r');
xlabel('\omega/pi');
ylabel('gain,DB');
title('gain response ---11tap -8 bit quantization');
legend('unquantized', 'quantized',2);

%  formation of  look up table of size(2^N).

n=0:(2^N-1);
 d= dec2bin(n)
 total=0;
for j=1:2^N
    total=0;
  for i=1:1:N
     if  qc(1,i)~=0   %for zero  valued coefficients  in half band filter
   R=str2num (d(j,i));
    temp=R*qc(1,i)
total= total+temp;
```

```
 end;
end;
 li(1,j)=total;
 end;
 display(li)

sum=0;
 x=input('enter input sigal in decimal'); %input equal to no. of tapes
 xx=dec2bin(x,B);
  q=xx';
 ee=fliplr(q);
% formation of  matrix to produce address .
ef=flipud(ee);

   %   scaling accumulator
 for i=1:B  %i=no of bits in input
      a=   ef(i,:);
      cc=bin2dec(a);



lut_in=li(1,cc+1);
%  y1=bitor(sum,lut_in);

% u2=bitshift( lut_in,-1);
% sum=bitor(sum,u2);
sum=lut_in+y1;
y1=bitshift(sum,-1);
end;
```

APPENDIX -B

Matlab code for polyphase Decimator

```
clc;
Clear all;
Close all;

M=input ('ENTER THE VALUE OF DECIMATION FACTOR');

N=input ('ENTER THE VALUE OF FILTER LENGTH');

B=input ('ENTER THE VALUE OF INPUT BIT PRECISION');
C=input ('ENTER THE VALUE OF COEFFICIENT WIDTH');


%%%%% CALCULATING COEFFICIENT %%%%%%%%
win=hamming(N)
coeff=fir1((N-1),0.25,win);
display('coefficients');
display(coeff);
coeff=fliplr(coeff);
cq=round(coeff*(2^(C-1)-1));
h=cq;


Ts= 0.0001; n=0:1:50; Fs=1/Ts;
x=2.24+(sin(1000*pi*n*Ts)+sin(2000*pi*n*Ts)+sin(4000*pi*n*Ts));
x1=50*x;
q=quantizer('Mode','ufixed','Overflowmode','saturate','Format',[8 0]);
 x2=quantize(q,x1);


%%%%%PLOTING OF NORMAL FILTER%%%
```

```
z=filter(h,1,x2);
figure(1)
subplot(2,1,1)
nn=length(x2);
i=1:1:nn
stem(i,x2);
title('original filter')
xlabel('n');
 ylabel('x(n)');

 %DECOMPOSITION OF FILTER COEFFICIENTS%%
 th=length(h);
lp=floor((th-1)/M)+1;
p=reshape([reshape(h,1,th),zeros(1,lp*M-th)],M,lp);
len_p=length(p(1,:));
 n1=0:(2^len_p-1)
 d= dec2bin(n1,length(p(1,:)))

 %%%lut formation%%%%
  for k=1:M
     qc=p(k,:);
     qc1=fliplr(qc);
      a=length(p(1,:))
 for j=1:2^a
    total=0;
  for i=1:1:a

   R=str2num(d(j,i));
    temp=R*qc1(1,i)
```

```
total= total+temp;
 end;
 li(k,j)=total;

%  lut(1,k)=total;
end;
end;

   %%%% DECOMPOSITION OF INPUT%%%

i=1:51
u(1,i)=x2(i);

 i=1:49
% u(1,i)=x2(2:1:51);
u(2,i)=[204,218,170,141,162,177,134,64,50,111,173,159,89,46,61,82,53,5,19,111,204,21
8,170,141,162,177,134,64,50,111,173,159,89,46,61,82,53,5,19,111,204,218,170,141,161,
177,134,64, 50];
 i=1:48
u(3,i)=[218,170,141,162,177,134,64,50,111,173,159,89,46,61,82,53,5,19,111,204,218,17
0,141,162,177,134,64,50,111,173,159,89,46,61,82,53,5,19,111,204,218,170,141,161,177,
134,64, 50];
i=1:47
u(4,i)=[170,141,162,177,134,64,50,111,173,159,89,46,61,82,53,5,19,111,204,218,170,14
1,162,177,134 64,50,111,173,159,89,61,82,53,5,19,111,204,218,170,141,161,177,134,64,
50];
% % SCALING ACCUMULATOR

 out=0; total1=0;
 for k =1:M
 l_in=li(k,:);
 y_d=u(k,:);
%  y_d(27)=0;y_d(28)=0;y_d(29)=0;
   for c1=1:(N-1)
    y_d(1,(49+c1))=0;
end
%
 for c=1:4:46

   k1=[y_d(1,c),y_d(1,c+1),y_d(1,c+2),y_d(1,c+3)];

    total1=0;
 xx=dec2bin (k1,B);
  q1=xx';
  ee=fliplr(q1);
ef=flipud((ee));


 y1=0;sum=0;
 for i=1:B
% i=no of bits in input
%
      a =  ef(i,:);
       cc=bin2dec(a);
       lut_in=l_in(1,cc+1);
     sum=lut_in+y1;
 y1=bitshift(sum,-1);
      %  total1 = total1+((2^(-(i-1)))*lut_in);
%        dd(i)=total1;


   end;
   out(k,c)=sum;
 end;
 end;
 for c=1:4:51
       outans=out(1,:)+out(2,:)+out(3,:)+out(4,:);
     end;
 subplot(2,1,2)
 stem(outans);
 title('decimated filter m=4')
xlabel('n');
 ylabel('y(n)');

%
ff=fftshift(abs(fft(x2,512)));

v=((0:length(ff)-1)/length(ff))* Fs;
figure(3)
subplot(3,1,1)
plot(v,ff);
title('frequency response of original filter')
xlabel('n');
 ylabel('x(w)');

of=fftshift(abs(fft(outans)));
 mm=((0:length(of)-1)/length(of))*Fs;
subplot(3,1,3)
 plot(mm,of);
 title('frequency response of decimation  filter m=4')
xlabel('n');
 ylabel('y(w)');
```

APPENDIX -C

VHDL Test bench

```
LIBRARY IEEE;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std. ALL;
Use IEEE.STD_LOGIC_UNSIGNED.ALL;



ENTITY gen_da23_gen_da23_t_vhd_tb IS
generic (B: POSITIVE:=4;      ---B=Input bit precision--
  N: POSITIVE: =11;         ---N = Number of tapes--
);
END gen_da23_gen_da23_t_vhd_tb;

ARCHITECTURE behavior OF gen_da23_gen_da23_t_vhd_tb IS

      COMPONENT gen_da23
      generic(B:positive);
      PORT (
            RSTn: IN std_logic;
            CLOCK: IN std_logic;
             DATA: IN std_logic_vector ((B-1) downto 0);
            DFF1: OUT std_logic_vector ((B-1) downto 0);
            dout:  out std_logic_vector (11 downto 0);
            RFD, RDY: out std_logic
            );
      END COMPONENT;

      SIGNAL RSTn: std_logic;
      SIGNAL CLOCK: std_logic;
      SIGNAL DFF1: std_logic_vector ((B-1) downto 0);
      SIGNAL DATA: std_logic_vector ((B-1) downto 0);
       Signal dout: std_logic_vector (11 downto 0);
```

```
BEGIN

      uut: gen_da23
      generic map(B)
      PORT MAP (
            RSTn => RSTn,
            CLOCK => CLOCK,
            DFF1 => DFF1,
            DATA=> DATA

      );



        ■  *** Test Bench - User Defined Section ***
        ■  --------Clock define---------
   tb : PROCESS---
   BEGIN

   CLOCK <= '0';
   wait for 1 ns;

   CLOCK <= '1';
   wait for 1 ns;
   END PROCESS;

-----------Reset define-------------
 tb1 : PROCESS
BEGIN
  RSTn <= '0';
   wait for 2 ns;

       RSTn <= '1';
   wait for 1 ms;
END  PROCESS;
```

```
-----------input stimuli to the design--------------
tb2 : PROCESS
BEGIN
  DATA<="0001";
wait for 8 ns;
 DATA<="0010";
wait for 8 ns;
 DATA<="0011";
wait for 8 ns;
 DATA<="0100";
wait for 8 ns;
 DATA<="0101";
wait for 8 ns;
 DATA<="0110";
wait for 8 ns;
 DATA<="0111";
wait for 8 ns;
 DATA<="1000";
wait for 8 ns;
 DATA<="1001";
wait for 8 ns;
 DATA<="1010";
wait for 8 ns;
 DATA<="1011";
wait for 8 ns;
 DATA<="1100";
wait for 8 ns;
 DATA<="1101";
wait for 8 ns;
 DATA<="1110";
wait for 8 ns;
 DATA<="1111";
wait for 8 ns;
END PROCESS;
END;
```