FPGA IMPLEMENTATION OF DIGITAL FIR FILTER

Thesis submitted in the partial fulfilment of requirement for the award of degree of

Master of Technology

in

VLSI Design & CAD

Submitted By:

Harsh Kumar

Roll No.: 601061011

Under the Guidance of:

Mr. Sanjay Kumar

Assistant Professor

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)

PATIALA – 147004 (PUNJAB)

June-2012

CERTIFICATE

I hereby declare that the work which is being presented here in the thesis entitled, "**FPGA IMPLEMENTATION OF DIGITAL FIR FILTER**" in partial fulfilment of the requirement for the award of degree of M.Tech (VLSI Design and CAD) at Electronics and Communication Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Mr. Sanjay Kumar, Assistant Professor, ECED.

The matter presented in this thesis has not been submitted in any other University/Institute for the award of my degree.

Date: 13/7/12

Harsh

Harsh Kumar

Roll No. : 601061011

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

Mr. Sanjay Kumar

Assistant Professor

ECED, Thapar University

Date: 13/07/2012

Counter Signed By:

Dr. R. K. Khanna

Professor & Head

ECED, Thapar University

Patiala-147004

Dr. S. K. Mohapatra

Dean Academic Affairs

Thapar University

Patiala-147004

ACKNOWLEDGEMENT

I would like to take the opportunity to express my gratitude to some people who were involved in this thesis work. First, I owe my gratitude to my mentor Mr. Sanjay Kumar for doing everything from the inception of the project idea to giving invaluable suggestions at every step. I am also thankful to Dr. R. K. Khanna, Head of the Department and Dr. Kulbir Singh PG Coordinator as well as all the faculty members and staff of the Department of Electronics and Communication Engineering for being very supportive to me. I would also like to thank all my batchmates for motivating me all the time whenever I needed them and giving me useful tips. I thank all those who have contributed directly or indirectly to this work.

Harsh Kumar
601061007

ABSTRACT

The Finite Impulse Response (FIR) filter is a digital filter widely used in Digital Signal Processing applications in various fields like imaging, instrumentation, communications, etc. Programmable digital processors signal (PDSPs) can be used in implementing the FIR filter. However, in realizing a large-order filter many complex computations are needed which affects the performance of the common digital signal processors in terms of speed, cost, flexibility, etc.

Field-Programmable gate Array (FPGA) has become an extremely cost-effective means of off-loading computationally intensive digital signal processing algorithms to improve overall system performance. The FIR filter implementation in FPGA, utilizing the dedicated hardware resources can effectively achieve application-specific integrated circuit (ASIC)-like performance while reducing development time cost and risks.

In this thesis, a low-pass, band pass and high pass FIR filter is implemented on FPGA. Direct-form approach in realizing a digital filter is considered. This approach gives a better performance than the common filter structures in terms of speed of operation, cost, and power consumption in real-time. The FIR filter is implemented in Spartan-III-xc3s500c-4fg320 FPGA and simulated with the help of Xilinx ISE (Integrated Software Environment). Software WEBPACK project navigator 9.2i was used for synthesizing and simulation the code.

Codes for direct form fixed point FIR filter have been realized. Modules such as multiplier, adder, ram and two's compliment were used. For an N order filter the number of shift register and adders required is N and the number of multipliers required is N+1. These filters can work in real time.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ADC	Analog-To-Digital Converter
ASIC	Application Specific Integrated Circuits
BIBO	Bounded Input-Bounded Output
BPF	Band Pass Filter
CAD	Computer-Aided Design
CPLD	Complex Programmable Logic Device
DAC	Digital-To-Analog Converter
DCM	Digital Clock Management
DSP	Digital Signal Processor
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Arrays
HDL	Hardware Description Languages
HPF	High Pass Filter
IEEE	Institute of Electrical and Electronic Engineers
IIR	Infinite Impulse Response
ISE	Integrated Software Environment
LPF	Low Pass Filter
MAC	Multiply-Accumulate
MXE	ModelSim Xilinx Edition
PDSP	Programmable Digital Processors Signal
PLD	Programmable logic device
SPLD	Simple Programmable Logic Device
UCF	User Constraints File
VHDL	Very High Speed Integrated Circuit Hardware Description Language

CHAPTER 1
INTRODUCTION

1.1 GENERAL

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range[2].

Figure 1.1 A block diagram of a basic filter.

A filter is an electrical network that alters the amplitude and/or phase characteristics of a signal with respect to frequency. Ideally, a filter will not add new frequencies to the input signal, nor will it change the component frequencies of that signal, but it will change the relative amplitudes of the various frequency components and/or their phase relationships. Filters are often used in electronic systems to emphasize signals in certain frequency ranges and reject signals in other frequency ranges.
There are two types of filter: analog and digital. FIR Filter is the kind of digital filter, which can be used to perform all kinds of filtering.

1.2 TYPES OF FILTERS

There are two main kinds of filter, Analog and Digital.

1. Analog Filters
2. Digital Filters

1.2.1 ANALOG FILTER

An analog filter has an analog signal at both its input $x(t)$ and its output $y(t)$. Both $x(t)$ and $y(t)$ are functions of a continuous variable time (t) and can have an infinite number of values. An analog filter uses analog electronic circuits made up from components such as resistors, capacitors and op amps to produce the required filtering effect. Such filter circuits are widely used in such applications as noise reduction, video signal enhancement, graphic equalizers in hifi systems, and many other areas. At all stages, the signal being filtered is an electrical voltage or current which is the direct analogue of the physical quantity (e.g. a sound or video signal or transducer output) involved.

Advantages:

- Simple and consolidated methodologies of plan,
- Fast and simple realization.

Disadvantages:

- Little stable and sensitive to temperature variations,
- Expensive to realize in large amounts.

1.2.2 DIGITAL FILTER

A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general purpose computer such as a PC, or a specialised DSP (Digital Signal Processor) chip. Digital filters are used in a wide variety of signal processing applications, such as spectrum analysis, digital image processing, and pattern recognition. Digital filters eliminate a number of problems associated with their classical analog counterparts and thus are preferably used in place of analog filters. The analog input signal must first be sampled and digitised using an ADC (analog to digital converter). The resulting binary numbers, representing successive sampled values of the input signal, are transferred to the processor, which carries out numerical calculations on them. Fast DSP processors can handle complex combinations of filters in parallel or cascade (series), making the hardware requirements relatively simple and compact in comparison with the equivalent analog circuitry[1].

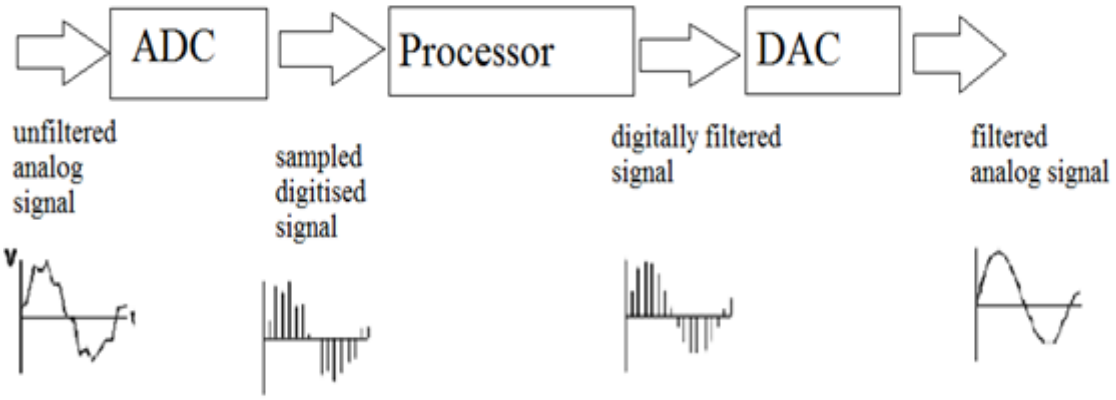


Figure 1.2 Block diagram of digital filter [1].

1.3 ADVANTAGES OF USING DIGITAL FILTER

The following list gives some of the main advantages of digital over analog filters[1].

1. A digital filter is programmable, i.e. its operation is determined by a program stored in the processor's memory. This means the digital filter can easily be changed without affecting the circuitry (hardware). An analog filter can only be changed by redesigning the filter circuit.
2. Digital filters are easily designed, tested and implemented on a general purpose computer or workstation.
3. The characteristics of analog filter circuits (particularly those containing active components) are subject to drift and are dependent on temperature. Digital filters do not suffer from these problems, and so are extremely stable with respect to both time and temperature.
4. Unlike their analog counterparts, digital filters can handle low frequency signals accurately. As the speed of DSP technology continues to increase, digital filters are being applied to high frequency signals in the RF (radio frequency) domain, which in the past was the exclusive preserve of analog technology.
5. Digital filters are very much more versatile in their ability to process signals in a variety of ways; this includes the ability of some types of digital filter to adapt to changes in the characteristics of the signal.
6. Fast DSP processors can handle complex combinations of filters in parallel or cascade (series), making the hardware requirements relatively simple and compact in comparison with the equivalent analog circuitry.

1.4 CHARACTERISTICS OF AN IDEAL FILTER

Ideal filters allow a specified frequency range of interest to pass through while attenuating a specified unwanted frequency range. The filters are classified according to their frequency range characteristics[2].

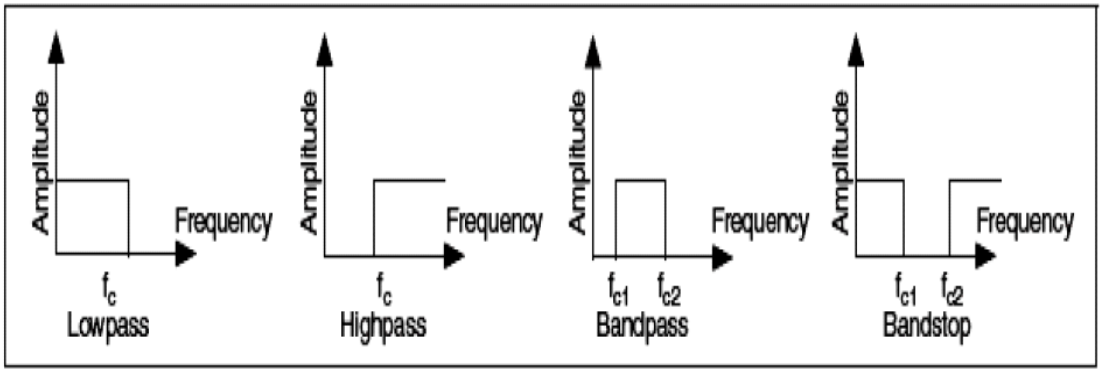


Figure 1.3 Ideal frequency characteristics [2].

In Figure 1.3, the filters exhibit the following behaviour:

• The low pass filter passes all frequencies below f_c.

• The high pass filter passes all frequencies above f_c.

• The band pass filter passes all frequencies between f_{c1} and f_{c2}.

• The band stop filter attenuates all frequencies between f_{c1} and f_{c2}.

The frequency points f_c, f_{c1}, and f_{c2} specify the cut off frequencies for the different filters. When designing filters, you must specify the cut off frequencies. The pass band of the filter is the frequency range that passes through the filter. An ideal filter has a gain of one (0 dB) in the pass band so the amplitude of the signal neither increases nor decreases.

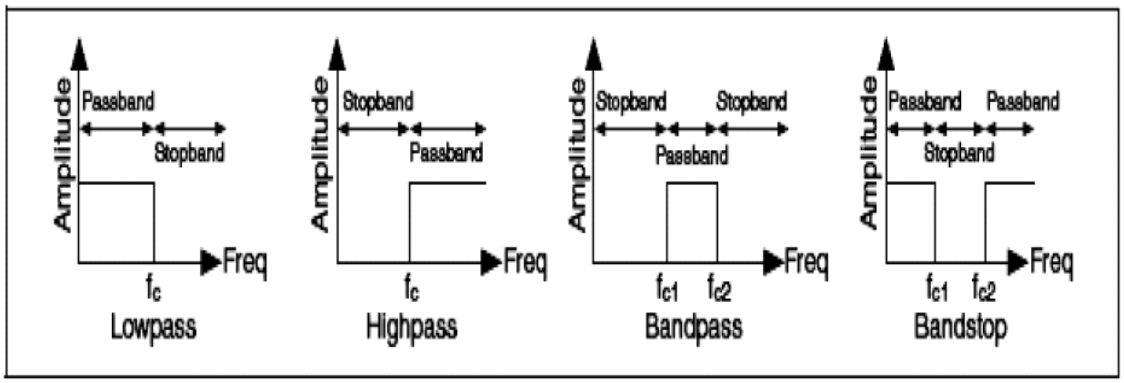


Figure 1.4 Pass band and Stop band [2].

1.5 PRACTICAL (NON IDEAL) FILTERS

In practical applications, ideal filters are not realizable. Ideally, a filter has a unit gain (0 dB) in the pass band and a gain of zero (–∞ dB) in the stop band. However, real filters cannot fulfill all the criteria of an ideal filter. In practice, a finite transition band always exists between the pass band and the stop band. In the transition band, the gain of the filter changes gradually from one (0 dB) in the pass band to zero (–∞ dB) in the stop band[2].

1.5.1 TRANSITION BAND

Figure 1.5 shows the pass band, the stop band, and the transition band for each type of practical filter. In each plot in Figure 1.5, the x axis represents frequency, and the y axis represents the magnitude of the filter in dB. The transition band is the region within which the gain of the filter varies from 0 dB to –3 dB. Here transition band is in between pass band and stop band. The filter of a large order has a narrow transition band. The shorter the transition band, the better the practical filter[2].

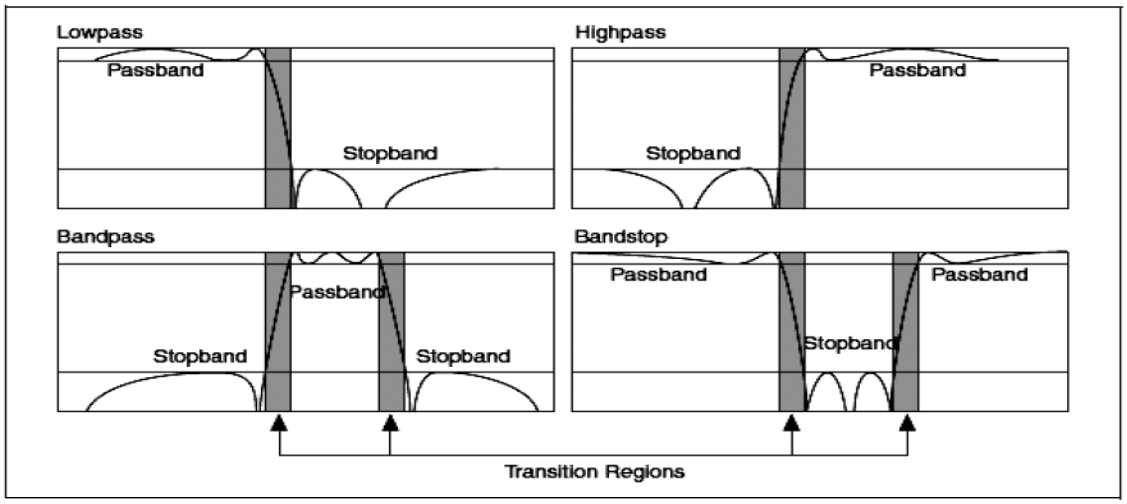


Figure 1.5 Response of Non-ideal filter [2].

1.5.2 PASSBAND RIPPLE AND STOPBAND ATTENUATION

In many applications, you can allow the gain in the pass band to vary slightly from unity. This variation in the pass band is the pass band ripple, or the difference between the actual gain and the desired gain of unity. In practice, the stop band attenuation cannot be infinite,

and you must specify a value with which you are satisfied. Measure both the pass band ripple and the stop band attenuation in decibels (dB)[2].

1.5.3 SAMPLING RATE

The sampling rate is important to the success of a filtering operation. The maximum frequency component of the signal of interest usually determines the sampling rate. In general, choose a sampling rate 10 times higher than the highest frequency component of the signal of interest[2] .

1.5.4 DIGITAL FILTER COEFFICENTS

All of the digital filter examples given above can be written in the following general forms[7] :

Zero order: $y_n = a_0 x_n$

First order: $y_n = a_0 x_n + a_1 x_{n-1}$

Second order: $y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2}$

Third order: $y_n=a_0x_n+a_1x_{n-1}+a_2x_{n-2}+a_3x_{n-3}$

Fourth order: $y_n=a_0x_n+ a_1x_{n-1}+a_2x_{n-2}+a_3x_{n-3}+a_4x_{n-4}$

Similar expressions can be developed for filters of any order.

The constants a_0, a_1, a_2, a_3, a_4 appearing in these expressions are called the filter coefficients. It is the values of these coefficients that determine the characteristics of a particular filter.

1.6 COMMON DIGITAL FILTERS

Traditional filter classification begins with classifying a filter according to its impulse response. These terms refer to the differing "impulse responses" of the two types of filter. Digital filter can be classified as one of the following types[7]:

• Finite impulse response(FIR) filter, also known as non recursive filters (in a non recursive filter the current output is calculated solely from the current and previous input values).

• Infinite impulse response(IIR) filter, also known as recursive filter (a recursive filter is one which in addition to input values also uses previous output values).

1.6.1 IMPULSE RESPONSE

An impulse is a short duration signal that goes from zero to a maximum value and back to zero again in a short time. The impulse response of a filter is the response of the filter to an impulse and depends on the values upon which the filter operates. The Fourier transform of the impulse response is the frequency response of the filter. The frequency response of a filter provides information about the output of the filter at different frequencies. In other words, the frequency response of a filter reflects the gain of the filter at different frequencies. For an ideal filter, the gain is one in the pass band and zero in the stop band. An ideal filter passes all frequencies in the pass band to the output unchanged but passes none of the frequencies in the stop band to the output[1].

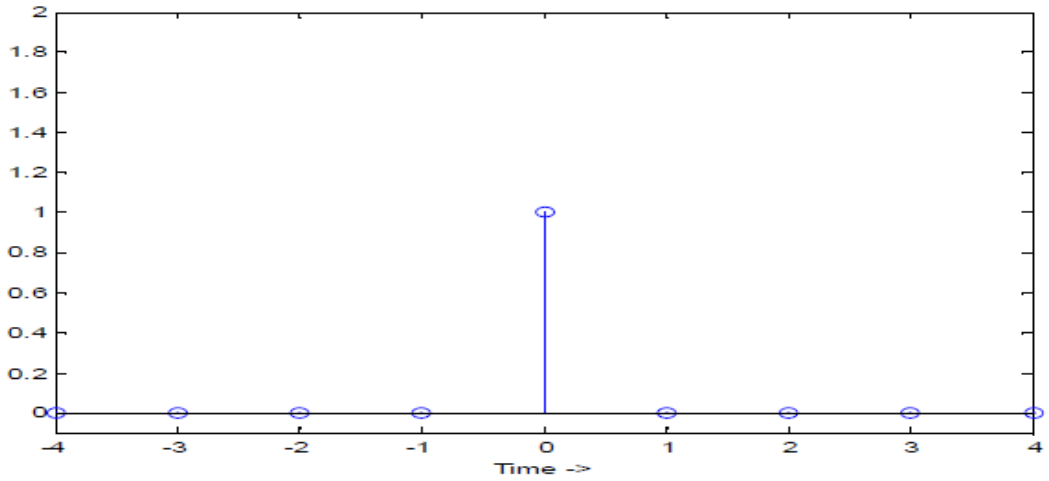


Figure 1.6 Impulse Response [1].

1.6.2 FIR FILTERS

Finite impulse response (FIR) filters are digital filters that have a finite impulse response. FIR filters operate only on current and past input values and are the simplest filters to design. FIR filters also are known as non recursive filters.

This can be stated mathematically as

$$h(n)= \begin{cases} 0, & n \geq n_1 \\ 0, & n \geq n_2 \end{cases}$$ where *n1* and *n2* lies between $-\infty$ and ∞.

where $h(n)$ denotes the impulse response of the digital filter, n is the discrete time index, and n_1 and n_2 are constants. A difference equation is the discrete time equivalent of a continuous time differential equation[7].

The general difference equation for a FIR digital filter is

$$y(n)= \sum_{k=0}^{n-1} b_k \, x(n-k) \qquad (1.1)$$

where $y(n)$ is the filter output at discrete time instance n, b_k is the k_{th} feed forward tap, or filter coefficient, and $x(nk)$ is the filter input delayed by k samples. The Σ denotes summation from $k = 0$ to $k = n_1$ where n is the number of feed forward taps in the FIR filter. FIR filters are the simplest filters to design. If a single impulse is present at the input of an FIR filter and all subsequent inputs are zero, the output of an FIR filter becomes zero after a finite time. Therefore, FIR filters are finite. The time required for the filter output to reach zero equals the number of filter coefficients. Equation describe the behaviour of the filter only in terms of current and past inputs. So FIR filter are also known as non recursive filters.

1.6.3 IIR FILTERS

Infinite impulse response (IIR) filters, also known as recursive filters operate on current and past input values and current and past output values. Theoretically, the impulse response of an IIR filter never reaches zero and is an infinite response. A recursive filter is one which in addition to input values also uses previous output values . The expression for a recursive filter therefore contains not only terms involving the input values (x_n, x_{n-1}, x_{n-2},...) but also terms involving the past output values y_n, y_{n-1},.....The following general difference equation characterizes IIR filters[7].

$$y_i = \frac{1}{a_0} \left(\sum_{j=0}^{N_b-1} b_j X_{i-j} \; - \; \sum_{k=1}^{N_a-1} a_k \, y_{i-k} \right) \qquad (1.2)$$

Where b is the set of forward coefficients, N_b is the number of forward coefficients, a_k is the set of reverse coefficients, and N_a is the number of reverse coefficients. Where x_i is the current input, x_{i-j} is the past inputs, and y_{i-k} is the past outputs.

From this explanation, recursive filters require more calculations to be performed, since there are previous output terms in the filter expression as well as input terms. In fact, the reverse is usually the case: to achieve a given frequency response characteristic using a recursive filter generally requires a much lower order filter (and therefore fewer terms to be evaluated by the processor) than the equivalent non recursive filter. IIR filters might have ripple in the pass band, the stop band, or both. IIR filters have a nonlinear phase response.

1.7 COMPARING FIR AND IIR FILTERS

Because designing digital filters involves making compromises to emphasize a desirable filter characteristic over a less desirable characteristic, comparing FIR and IIR filters can help in selecting the appropriate filter design for a particular application. IIR filters have the advantages of providing the higher selectivity for a particular order[7].

IIR filters can achieve the same level of attenuation as FIR filters but with far fewer coefficients. Therefore, an IIR filter can provide a significantly faster and more efficient filtering operation than an FIR filter. FIR filters provide a linear phase response. IIR filters provide a nonlinear phase response. FIR filters are used for applications that require linear phase responses like high quality audio systems. IIR filters are used for applications that do not require phase information, such as signal monitoring applications.

Compared to IIR filters, FIR filters sometimes have the disadvantage that they require more memory and/or calculation to achieve a given filter response characteristic. Also, certain responses are not practical to implement with FIR filters. FIR filters are always stable because they are implemented using an all zero transfer function. Since no poles can fall outside the unit circle, the filter will always be stable . But because of this, the order of FIR filter is much higher than the IIR filter which has the comparable magnitude response. The higher order of the FIR filters lead to longer processing times and larger memory requirements.

1.8 FPGA: AN OVERVIEW

FPGA arrived in 1984 as an alternative to programmable logic devices (PLDs) and ASICs. FPGA offers the significant benefits of being readily programmable. An FPGA is a completely reconfigurable computer logic chip. Like traditional hardwired gate arrays, the chip consists of a series of logic gates. In the traditional array, these gates are specified and hard interconnected at the manufacturing stage. The field programmable gate array differs in that it can be programmed, and re-programmed, This has the advantages of allowing fast prototyping for applications it is intended to be implement with hard-wired chips. FPGA can

be programmed again and again, giving designers multiple opportunities to tweak their circuits[20].

1.8.1 COMPUTER AIDED DESIGN FOR VLSI CIRCUITS

The design of digital systems with VLSI circuits containing millions of transistors is a formidable task and requires the assistance of computer-aided design (CAD) tools. CAD tools consist of software programs that support computer-based representation and aid in the development of digital hardware by automating the design process. The designer can choose between a full-custom IC, a programmable logic device (PLD), an application specific integrated circuit (ASIC), or a field-programmable gate array (FPGA)[20].

1.8.2 PROGRAMMABLE LOGIC

Programmable logic is loosely defined as a device with configurable logic and flip-flops linked together with programmable interconnect. Memory cells control and define the function that the logic performs and how the various logic functions are interconnected.
What kinds of programmable logic devices are available today? How are they different from one another?
There are a few major programmable logic architectures available today. Each architecture typically has vendor-specific sub-variants. The major types include: [21]
1. Simple Programmable Logic Devices (SPLDs)
2. Complex Programmable Logic Devices (CPLDs)
3. Field Programmable Gate Arrays (FPGAs)

1.9 OBJECTIVE OF THESIS

The thesis embodies following objectives:

(1) To study the different methods of calculating filter coefficients such as Windowing, frequency sampling and Optimal method for FIR filter design.
(2) To study various FIR filter structures used for implementing the filters.
(3) To study various synthesis and simulation tools used to implement FIR filter.
(4) To design and implement FIR low pass, band pass and high pass filters on FPGA.

1.9.1 ORGANIZATION OF THESIS

Chapter 3 discusses the design stage for digital filter, which includes specification of filter, calculation of filter coefficients, realization of filter structure, finite world length effect and hardware or software implementation of filter. Also discusses coefficient calculation method for FIR filter, such as window, frequency sampling and optimal method. And at last comparison between these methods are presented.

Chapter 4 discusses the analysis of linear, time-invariant FIR filter which is generally carried out by using the *Z*-transforms; a brief review of the *Z*-transform is presented. Also the filter structures characterizing the difference equations are represented using basic elements such as multipliers, time-delays, and adders. The characteristics of an ideal FIR filter and the design using windowing techniques are given in this chapter.

In Chapter 5, the Simulation and Synthesis tools which are used in implementation of FIR filter is discussed. This chapter also discusses the FPGAs architecture, FPGA Design Flow, Spartan-III FPGA kit specifications.

Chapter 6 discusses about how FIR filter can be realized and the process of implementation of FIR filter. It also discusses restriction and assumption of filter, choosing the filter structure, because it is often important to choose a particular filter structure for a given transfer function *H (z)*.This chapter also discusses Fixed Point Representation of data.

Chapter 7 contains results of simulation using Modelsim and XILINX ISE 9.2i of low pass, band pass and high pass filters of given specifications.

CHAPTER 2
Literature Review

Ruan,A.W., Liao,Y.B., Li,P. , Li,J.X., (2009)[4] proposed and presented An ALU based universal FIR filter where various FIR filters can be implemented just by programming instructions in the ROM with identical hardware architecture. They presented Arithmetic Logic Unit (ALU) based universal FIR filter suitable for implementation in Field Programmable Gate Arrays (FPGA) is proposed in this paper. Multiplier and accumulator based architecture used in conventional FIR, the proposed ALU architecture implements FIR functions by using accumulators and shift-registers controlled by the instructions of ROM.

Nekoei,F., Kavian,Y.S., Strobel,O., (2010)[5] presented realization of digital FIR filters on field programmable gate array devices was proposed. Two common architectures called direct and transposed architectures were employed for implementing FIR filters on a Xilinx SPARTAN2-XC2S50-5I-tq144 FPGA using Verilog hardware description language codes.
X.Jiang, Y.Bao(2010) proposed a structure characteristics and the basic principles of the finite impulse response (FIR) digital filter, and gives an efficient FIR filter design based on FPGA. Use MATLAB FDATool to determine filter coefficients, and designed a 16-order constant coefficient FIR filter by VHDL language, take use of Quartus-2 to simulate filters, the results meet performance requirements.

Li,J., Zhao,M., Wang,X., (2011)[6] They used distributed algorithm and its several structures, an implementation method of 60-order FIR filter based on FPGA is presented, which converts multiplication to look-up table structure, and implement multiplication operation. They used FPGA as the hardware platform. this filter system has a good performance, the filter speed is higher and the resource occupation is fewer.

Beyrouthy,T., Fesquet,L., (2011)[24] presented an asynchronous FIR Filter architecture was presented, along with an asynchronous Analog to digital converter(A-ADC). They designed FIR Filter architecture using the micro-pipeline asynchronous style. They successfully implemented for the first time on a commercial FPGA board (Altera-DE1). A specific library

has also been designed for this purpose. It allows the synthesis of asynchronous primitive blocks(the control path in this case) on the synchronous FPGA. Simulation results of the FIR Filter after place and route validate the implementation. This work is still going on, in order to optimize the implementation.

Jieshan,L., Shizhen,H.,(2009)[27] presented the basic structure and hardware characteristics of the FIR digital filter and design method of the FIR filter is discussed on the basis of the FIR filter structure. They proposed a structure which is based on FPGA, draws the coefficient by Matlab and adopts the pipeline to implete the FIR digital filter. They also presented the introduction of the overall framework of the FIR digital filter adopting the finite state machine as well as the principle of each module of the design. The design is impleted by use of the Verilog hardware description language and each module is verified and simulated by Quartus 8.0 and Modelsim-Altera.

Saab,S., Lu,W.-S., Antoniou,A.,(1999)[28] proposed method for the design of linear-phase IIR digital filters for low-power applications is proposed. In this method, the digital filter is implemented as a cascade arrangement of 2nd-order sections. Each section is designed through optimization techniques so that all sections in cascade satisfy as far as possible the overall required specifications. This process is repeated until a multi section filter is obtained which satisfies the required specifications under the most critical circumstances imposed by the application at hand. The minimum number of sections required to process a particular input signal can then by switched on through the use of a simple adaptation mechanism and, in this way, the power consumption can be minimized. This design structure is achieved by formulating the design of the k-1 sections as constraints. As an example, a low-power filter system is compared with a fixed-order linear-phase IIR filter of the same performance. It is shown that a power reduction of at least 10% can be achieved.

Yunlong,W., Shihu,W., Rendong,J., (2011)[29] They proposed a simple method for design an Fir filter as compare to other existed methods. This method is the simplest except rectangular window method. Filter transition bandwidth is smaller than 4.65/N for filter order N. For the same filter specifications filter order obtained by using the new method is much smaller than by using Kaiser window if minimum stopband attenuation is in the range of 39.5db to 48.5db and corresponding maximum passband ripple is from 0.35db to 0.18db.

Nian-qiang,L., Si-Yu,H., Shi-Yao,C. (2010)[30] presents principle of distributed algorithms and applied to the ECG signal FIR filter design methods, combined with Altera's Cyclone series chip made to achieve FIR digital filter design, distributed algorithms can greatly reduce the size of the hardware circuitry to improve the circuit speed of execution. Recently, as is widely used in recent FPGA digital signal processing, distributed algorithm in the FPGA to achieve the FIR digital filter to become very real needs. In the Quartus 2 and the Matlab environment to complete the simulation and synthesis, get good results.

Soni,V., Shukla,P., Kumar,M., (2011)[31] proposed that the Exponential window provides better side-lobe roll-off ratio than Kaiser window which is very useful for some applications such as beam forming, filter design, and speech processing. In this paper the second application i.e. design of digital non recursive Finite Impulse Response (FIR) filter by using Exponential window is proposed. The far-end stop band attenuation is most significant parameter when the signal to be filtered has great concentration of spectral energy. In a sub-band coding, the filter is intended to separate out various frequency bands for independent processing. In case of speech, e.g. the far-end rejection of the energy in the stop band should be more so that the energy leakage from one band to another is minimum. Therefore, the filter should be designed in such a way so that it can provide better far-end stop band attenuation (amplitude of last ripple in stop band).

Digital FIR filter designed by Kaiser window has a better far-end stop band attenuation than filter designed by the other previously well known adjustable windows such as Dolph-Chebyshev and Saramaki, which are special cases of Ultra spherical windows, but obtaining a digital filter which performs higher far-end stop band attenuation than Kaiser window will be useful. In this paper, the design of non recursive digital FIR filter has been proposed by using Exponential window. It provides better far-end stop band attenuation than filter designed by well known Kaiser window, which is the advantage of filter designed by Exponential window over filter designed by Kaiser window. The proposed schemes were simulated on commercially available software and the results show the close agreement with proposed theory.

CHAPTER 3
FIR FILTER DESIGN

3.1 INTRODUCTION

The design of a digital filter involves following five steps[3].

(a) Filter specification: This may include stating the type of filter, for example low pass filter, the desired amplitude and/or phase responses and the tolerances, the sampling frequency, the word length of the input data.

(b) Filter coefficient calculation: The coefficient of a transfer function $H(z)$ is determined in is this step, which will satisfy the given specification. The choice of coefficient calculation method will be influenced by several factors. The most important of which are the critical requirements i.e. specification. The window, optimal and frequency sampling method are the most commonly used.

(c) Realization: This involves converting the transfer function into a suitable filter network or structure.

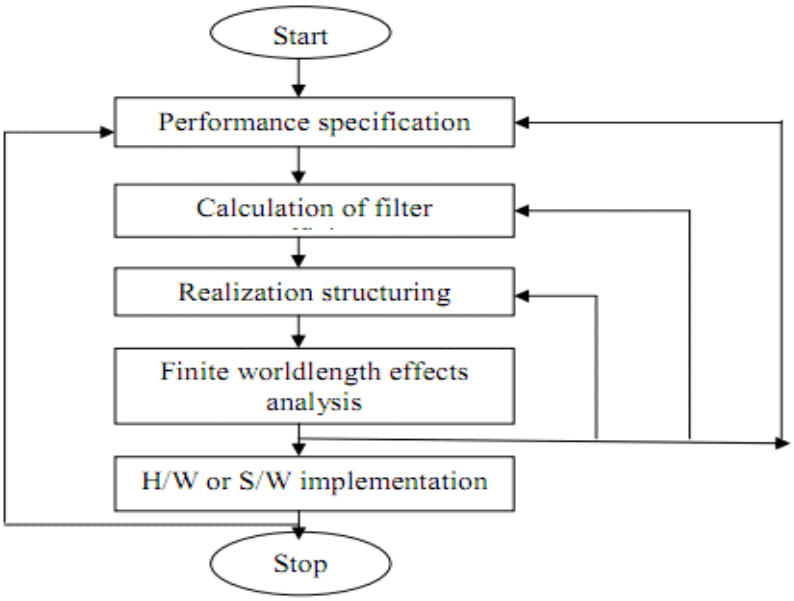


Figure 3.1 Summary of design stage for digital filter [3].

(d) Analysis of finite word length effects: The effect of quantizing the filter coefficients and input data as well as the effect of carrying out the filtering Start Performance specification Calculation of filter coefficients Realization structuring Finite world length effects analysis H/W or S/W implementation Stop operation using fixed word length on the filter performance is analyzed here.

(e) Implementation: This involves producing the software code and/or hardware and performing the actual filtering.

3.2 FIR FILTER SPECIFICATIONS

The specifications includes

(i) Signal characteristics.

(ii) The characteristics of the filter.

(iii) The manner of implementation.

(iv) Other design constraints (cost).

Although the above requirements are application dependent it will be helpful to devote some time on the characteristics of the filter. The characteristics of digital filters are often in specified in the frequency domain. For frequency selective filters, such as lowpass and bandpass filters, the specifications are often in the form of tolerance[4].

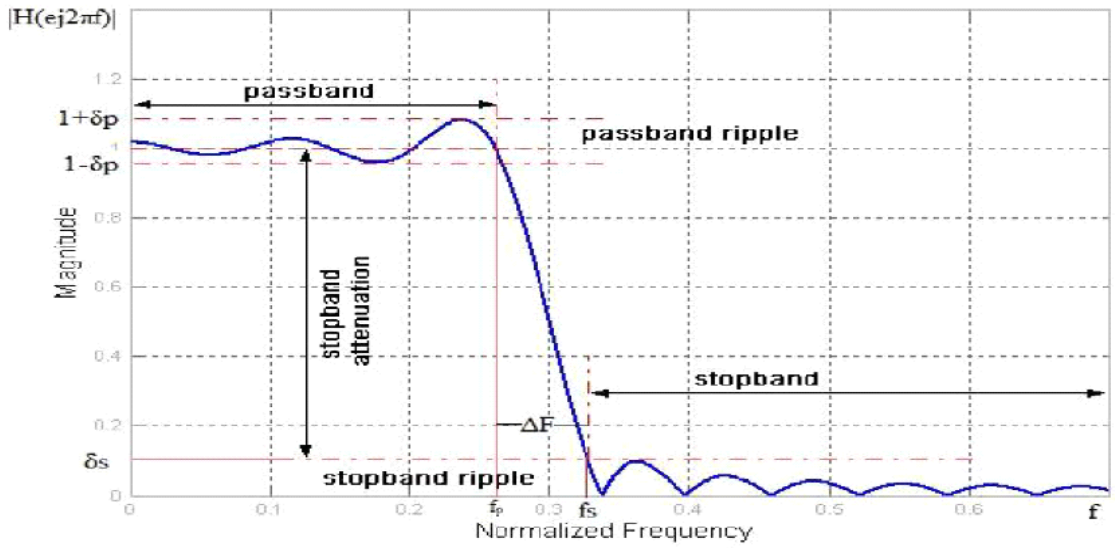


Figure 3.2 Magnitude frequency response specifications for a lowpass filter [4].

In the pass band, the magnitude response has a peak deviation of δ_p and in the stop band, it as a maximum deviation of δ_s. The width of transition band determines how sharp the filter is. The magnitude response decreases monotonically from the pass band to stop band in this region.The following are the key parameters of interest:

1. δ_p peak pass band deviation(or ripples).
2. δ_s stop band deviation.
3. f_s stop band edge frequency.
4. f_p pass band edge frequency.
5. F_s sampling frequency.

Thus the minimum stop band attenuation, As and the peak pass band ripple, Ap, in decibels are given as

A_s (stop band attenuation) = $-20\log_{10} \delta s$

A_p (pass band ripple) = $20 \log_{10} (1+\delta p)$

The difference between *fs* and *fp* gives the transition width of the filter. Another important parameter is the filter length, *N*, which defines the number of filter.

3.3 FIR COEFFICIENT CALCULATION METHODS

The objective of most FIR coefficient calculation methods is to obtain values of $h(n)$ such that the resulting filter meets the design specifications, such as amplitude frequency response and throughput requirements. Several methods are available for obtaining $h(n)$. The window, optimal and frequency sampling method are the most commonly used[7].

3.3.1 WINDOW METHOD

In this method, use is made of the fact that the frequency response of a filter, $H_D(\omega)$ the corresponding impulse, h_D are related by the Fourier transform[7] :

$$h_D(n)=\frac{1}{2\pi}\int_{-\pi}^{\pi} H_D(\omega) \quad e^{jwn} \quad d\omega \qquad (3.1)$$

Now start with the ideal low pass response shown in figure, where ω_c is cut off frequency and

the frequency scale is normalised: T=1. By letting the response go from -ω_c to ω_c we simplify the integration operation. Thus the impulse response is given by:

$$h_D(n)=\frac{1}{2\pi}\int_{-\pi}^{\pi} 1 * e^{j\omega n}\, d\omega \qquad (3.2)$$

$$=\frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{j\omega n}\, d\omega$$

$$=\frac{2f_c}{n\omega_c}\sin(n\omega_c)\ , \qquad \text{where } n\neq 0,\ -\infty \leq n \leq \infty$$

$$= 2f\ ,\ n = 0$$

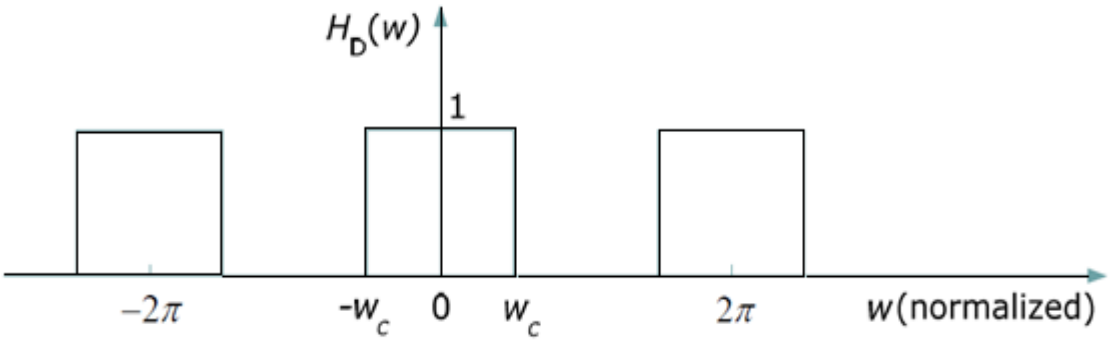


Figure 3.3 Ideal frequency response of a lowpass filter .

Filter Type	$h_D(n), n \neq 0$	$h_D(0)$
Low Pass	$\frac{2f_c\, sin(n\omega_c)}{n\omega_c}$	$2f_c$
High Pass	$-\frac{2f_c\, sin(n\omega_c)}{n\omega_c}$	$1-2f_c$
Band Pass	$\frac{2f_2\, sin(n\omega_2)}{n\omega_2}-\frac{2f_1\, sin(n\omega_1)}{n\omega_1}$	$2(f_2-f_1)$
Band Stop	$\frac{2f_1\, sin(n\omega_1)}{n\omega_1}-\frac{2f_2\, sin(n\omega_2)}{n\omega_2}$	$1-2(f_2-f_1)$

Table3.1 Summary of ideal impulse responses [3].

The ideal infinite impulse response is truncated by using various windows. Here we multiply the ideal frequency response with a window function. When this window is multiplied by the deal transfer function then all the coefficients within the window are retained and all that are outside the window are discarded.

Table3.2 Summary of important features of common window function

Window	**Transition width**	**Minimum stop band attenuation**
Rectangular	4π/N	-21dB
Bartlett	8π/N	-25dB
Hanning	8π/N	-44dB
Hamming	8π/N	-53dB
Blackmann	12π/N	-74dB
Kaiser	variable	Variable

(a) Rectangular

$$W_R(n)= \begin{cases} 1, & 0\le n\le N-1 \\ 0, & \text{otherwise} \end{cases}$$

(b) Bartlett (or triangle)

$$W_B(n)= \begin{cases} \frac{2n}{N-1}, & 0\le n\le (N-1)/2 \\ 2-\frac{2n}{N-1}, & \frac{N-1}{2}\le n\le N-1 \\ 0, & \text{otherwise} \end{cases}$$

(c) Hanning

$$W_{ham}(n)= \begin{cases} \frac{1-\frac{\cos 2\pi n}{N-1}}{2}, & 0\le n\le N-1 \\ 0, & \text{otherwise} \end{cases}$$

(d) Hamming

$$W_{har}(n)= \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), & 0\le n\le N-1 \\ 0, & \text{otherwise} \end{cases}$$

(e) Blackmann

$$W(n)=\begin{cases} 0.42\text{-}0.5\cos(2\pi n/M) + 0.08\cos\left(\frac{4\pi n}{N\text{-}1}\right), & 0\leq n \leq N\text{-}1 \\ 0, & \text{otherwise} \end{cases}$$

3.3.2 FREQUENCY SAMPLING METHOD

The frequency sampling method allows us to design non recursive FIR filter for both standard frequency filters (low pass, high pass & band pass filter) and filter with arbitrary frequency response. A unique attraction of the frequency sampling method is that it also allows recursive implementation of FIR filters[7].

3.3.2.1 NON RECURSIVE FREQUENCY SAMPLING

To obtain the FIR coefficients of the filter whose frequency response is given. By taking *N* samples of the frequency response at intervals of Kf_S/N, k = 0,1, ., N-1. The filter $h(n)$ coefficients can be obtained as inverse DFT of frequency samples[7].

$$h(n) = \frac{1}{N}\sum_{k=0}^{N-1} H(k)\, e^{j\left(\frac{2\pi}{N}\right)nk} \qquad (3.3)$$

where ,$H(k)$, k = 0,1,2,.....N-1, are the samples of the frequency response .The impulse response coefficients of linear phase FIR filter with positive symmetry, for *N* even, can be expressed as:

$$h(n) = \frac{1}{N}\ [\ \sum_{k=1}^{\frac{N}{2}-1} 2|H(k)|\cos[2\pi k(n\text{-}\alpha)/N\ ] + H(0)\ ] \qquad (3.4)$$

where α = (N-1)/2, and $H(k)$ are the samples of the frequency response of the filter taken at intervals of kF_s/N. For N odd, the upper limit in the summation is (N-1)/2.The resulting filter will have exactly the same frequency response as the original response at the sampling instants. To obtain a good approximation to the desired frequency response, a sufficient number of frequency samples must be taken. An alternative frequency sampling filter, know as type 2, results if frequency sample taken at intervals of f_k where f_k defined by

$$f_k = (k+1)/2F_s/N \quad , \quad k=0,1,2,.....N-1 \tag{3.5}$$

To improve the amplitude response of frequency samples in the wider transition, introducing frequency samples in the transition band. For a low pass filter the stop band attenuation increases, approximately, by 20 dB for each transition band frequency sample, with a corresponding increase in the transition width:

Approximate stop band attenuation = (25+20M) dB

Approximate transition width = (M+1) F_s/N

Where *M* is the number of transition band frequency samples and *N* is the filter length.

3.3.2.2 RECURSIVE FREQUENCY SAMPLING

Recursive forms of the frequency sampling offer significant computational advantages over the non recursive forms if a large number of frequency samples are zero valued. The transfer function of an FIR filter, *H*(*z*), can be expressed in a recursive form[7]:

$$H(Z) = \frac{1-Z^{-N}}{N} \sum_{k=0}^{N-1} \frac{H(K)}{1-Z^{-1}e^{j2\pi k/N}} = H_1(Z)H_2(Z) \tag{3.6}$$

Thus in recursive form, *H*(*z*) can be viewed as a cascade of two filters: in a comb filter, $H_1(z)$, which has *N* zeros uniformly distributed around the unit circle, and a sum of *N* single all-pole filters, $H_2(z)$. Thus the zero cancel the pole, making *H*(*z*) an FIR as it effectively has no poles. In practice, due to finite word length effects the poles of $H_2(z)$ not to be located exactly on unit circle so that they are not cancelled by the zeros, making *H*(*z*) an IIR and potentially unstable. Stability problems can be avoided by sampling *H*(*z*) at a radius, *r*, slightly less than unity. Thus the transfer function in this case becomes

$$H(Z) = \frac{1-r^N Z^{-N}}{N} \sum_{k=0}^{N-1} \frac{H(K)}{1-rZ^{-1}e^{j2\pi k/N}} \tag{3.7}$$

In general, the frequency samples, *H*(*k*), are complex. Thus direct implementation requires

complex arithmetic. To avoid this, the symmetry inherent use in frequency response of any FIR filters with real impulse $h(n)$. So above equation can expressed as

$$H(Z) = \frac{1-r^N Z^{-N}}{N}\left[\frac{|H(K)|\left\{2\cos\left(\frac{2\pi k\alpha}{N}\right)-2r\cos\left[\frac{2\pi k\,(1+\alpha)}{N}\right]Z^{-1}\right\}}{1-2r\cos\left(\frac{2\pi k}{N}\right)Z^{-1}+r^2 z^{-2}}+\frac{H(0)}{1-Z^{-1}}\right] \quad (3.8)$$

Where α= (N-1)/2 . For *N* odd M= (N-1)/2 and for *N* even M= N/2-1

3.3.3 THE OPTIMAL METHOD

The optimal method of calculating FIR filter coefficients is very powerful, very flexible and very easy to apply. The optimal method is based on the concept of equiripple pass band and stop band. Consider the low pass filter frequency response, in pass band the response oscillates between 1- δp and 1+ δp. In the stop band the filter response lies between 0 and δs. The difference between the ideal filter and the practical response can be viewed as an error function[7]:

$$E(\omega) = W(\omega)\,[H_D(\omega) - H(\omega)] \quad (3.9)$$

Where $H_D(\omega)$ is the ideal response and $W(\omega)$ is a weighting function that allows the relative error of approximation between different bands to be defined. In optimal method, the objective is to determine the filter coefficients, $h(n)$, such that the value of the weighted error, |E(ω)|, is minimized in the pass band and stop band. Mathematically, this may be expressed as: min[max|E(ω)|], over the pass bands and stop bands.

It has been established that when max|E(ω)| is minimized the resulting filter response will have equiripple pass band and stop band. The minima and maxima are known as extrema. For linear phase low pass filter, there are either r+1 or r+2 extrema, where r = (N+1)/2 (for type 1 filter) or r =N/2 (for type 2 filter). For a given set of filter specifications, the location of the extremal frequencies, apart from those at band edges (that is at f=fp and f= Fs/2), are not known a priori.

By knowing the locations of the extremal frequencies, it is a simple matter to work out the actual frequency response and the impulse response of filter.

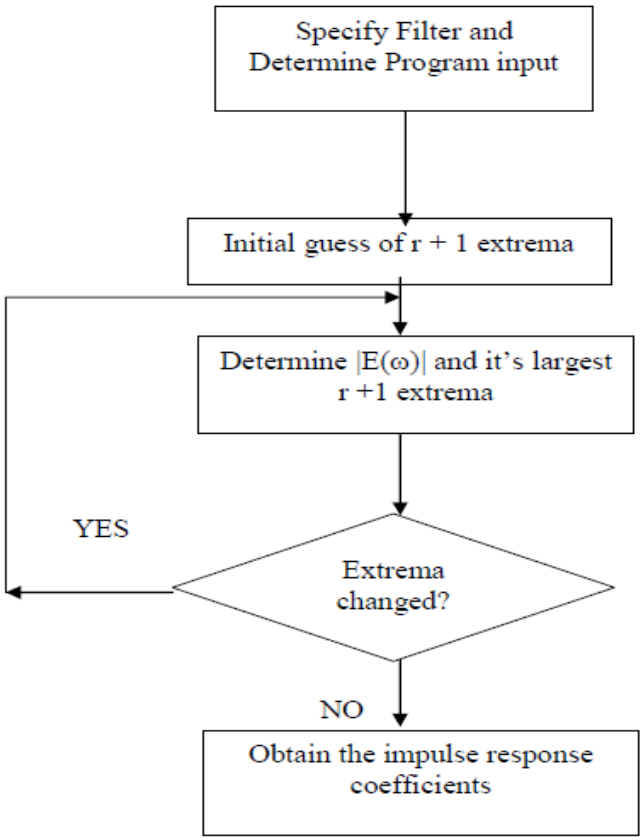


Figure 3.4 Simplified flowchart of the optimal method [7].

3.4 Comparison of The Window, Frequency Sampling and Optimal Method

The optimum method provides the easy and optimum way of computing FIR filter coefficients. Although the method provides total control of filter specifications, the availability of the optimal filter design software is mandatory. For most applications the optimal method will yield filters with good amplitude response characteristics for reasonable value of N. The method is particularly good for designing Hilbert transformers and differentiators. Other methods will yield larger approximation errors for differentiators and Hilbert transformers than the optimal method[7].

In the absence of the optimal software or when the pass band and stop band ripples are equal, the window method represents a good choice. It is a particularly simple method to apply and conceptually easy to understand. However, the optimal method will often give a more economic solution in terms of the numb of the filter coefficients. The window method does

not allow the designer a precise control of the cut off the cut-off frequencies or ripple in the pass band and stop band.

The frequency sampling approach is the only method that allows both non recursive and recursive implementations of FIR filters, and should be used when such implementations are envisaged as the recursive approach is computationally economical. The special form with integer coefficients should be considered only when primitive arithmetic and programming simplicity are vital, but a check should always be made to see whether its poor amplitude response is acceptable. Filters with arbitrary amplitude phase response can be readily designed by the frequency sampling method. The frequency sampling method lacks precise control of the location of the band edge frequencies or the pass band ripples and relies on the availability of the design.

CHAPTER 4

FIR FILTER STRUCTURES

4.1 INTRODUCTION

The analysis of linear, time-invariant FIR filter is generally carried out by using the *Z*-transforms. A brief review of the *Z*-transform is presented. The filter structures characterizing the difference equations are represented using basic elements such as multipliers, time-delays, and adders.

4.1.1 Z TRANSFORM

The *Z*-transform is very useful role in the analysis and characterization of the linear time-invariant systems. This is because the difference equations characterizing the discrete system are transformed into algebraic equations, which are much easier to manipulate[3].
The two sided *Z*-transform of discrete-time function *f(nT)* is given as

$$F(Z) = \sum_{n=-\infty}^{\infty} f(nT) z^{-n} \tag{4.1}$$

for all *z* for which *F*(*z*) converges. Here the argument *z* is a complex variable. Now, evaluating the Z-transform on digital filter Equation we obtain,

$$z\{y(nT)\} = z\left\{\sum_{i=0}^{N} a_i x(nT - iT)\right\}$$

By using the time translation property and the convolution property of *Z*-transform, Equation can be re-arranged as

$$\mathrm{Y(z)} = \mathrm{X(z)} \sum_{\mathrm{i=0}}^{\mathrm{N}} \mathrm{a_i z^{-i}} \tag{4.2}$$

$$\mathrm{Y(z)} = \mathrm{X(z)} * \mathrm{H(z)} \quad \text{where}$$

$$H(z) = \sum_{i=0}^{N} a_i z^{-i}$$

Where $H(z)$, $X(z)$, $Y(z)$ are the Z-transforms of Impulse Response, Input samples and Output samples [1]. $H(z)$ is called the transfer function of the filter and the time-domain samples of this transfer function, which are the filter coefficients are approximated according to the desired response. Basic elements, block representation and signal flow of FIR filter is shown in Figure 4.1. it is done using basic building blocks elements.

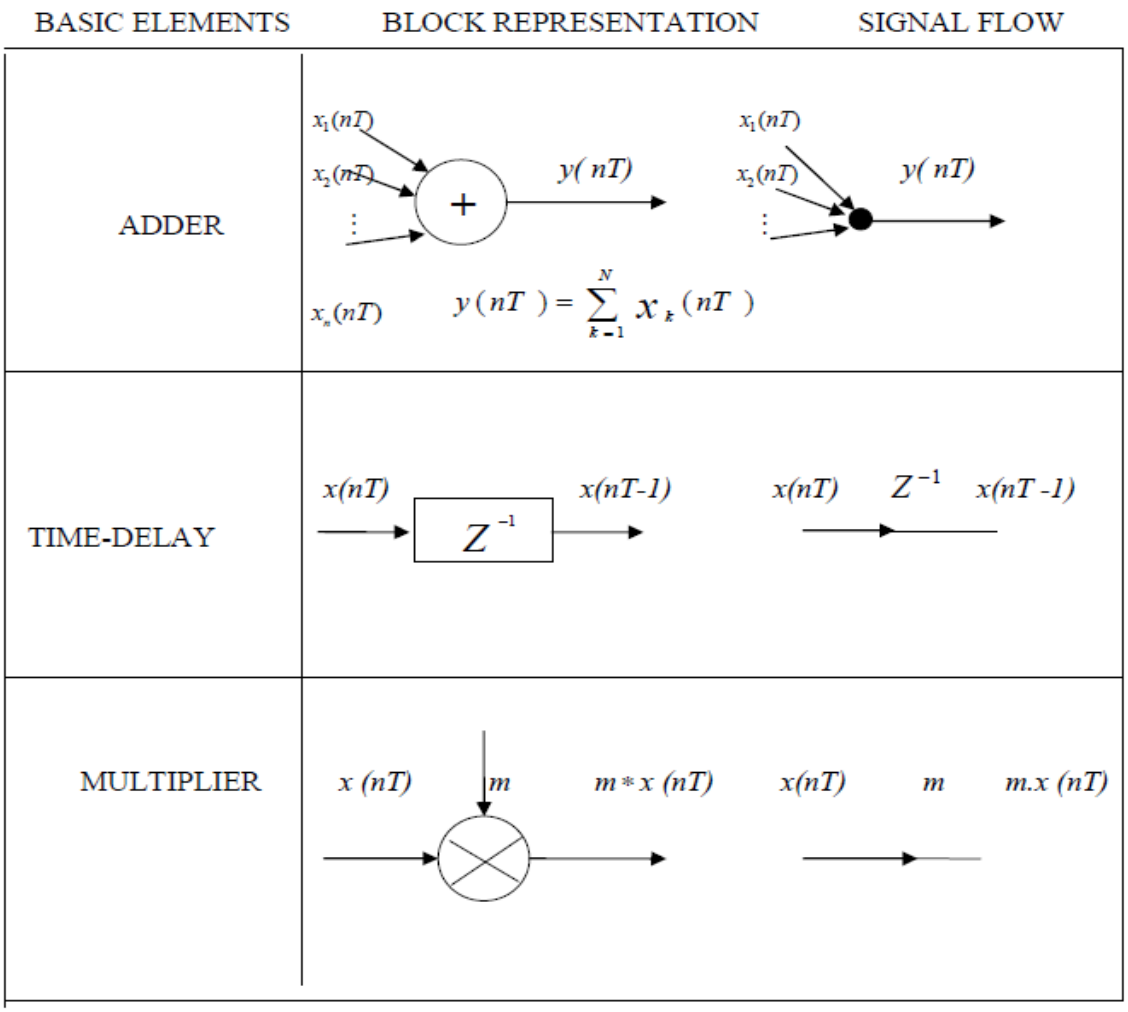


Figure 4.1 Block representation & Signal flow of basic elements [10].

4.2 FILTER STRUCTURES

The computational algorithm implementing Equation of an FIR filter can be conveniently represented in block diagram. It is done using the basic building blocks elements such as Multipliers, Adders, and Unit Delays. These basic block elements and their equivalent Signal Flow Diagrams are as shown in Figure 4.1[10].

This way of presenting the difference equations in the form of block diagram and signal flow diagram makes us easy to write an algorithm, which can be implemented in the digital computer. We will discuss here first about the direct form structure , transpose form structure Cascade structure and then lattice structure. A digital filter structure is said to be canonic if the number of delays in the block diagram representation is equal to the order of the transfer function Otherwise, it is a non canonic structure.

4.2.1 DIRECT-FORM STRUCTURE

Direct structures for the Digital filter are those in which the real filter coefficients appear as multipliers in the block diagram representation. If $X(z)$ is the filter input and $Y(z)$ is the filter output then the transfer function $H(z)$ is given as [10]

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{i=0}^{N} a_i z^{-i} \tag{4.3}$$

There are four Direct-form structures, which are different realizations of Equation (4.3).The first Direct structure only is presented here and is as shown in Figure 4.2.

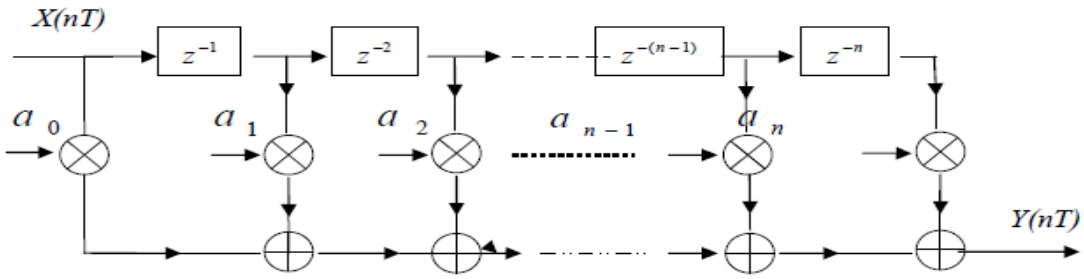


Figure 4.2 Direct-Form of FIR Filter [10].

The signal flow diagram of this structure is as shown below in Figure 4.3.

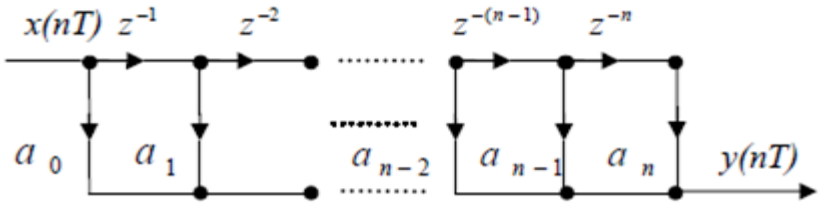


Figure 4.3 Signal flow diagram of Direct-Form [10].

The 1-D structure is also called canonical because it possesses n-time delay elements. As seen from the Signal Flow Diagram the above representation requires n Delay elements, n + 1 multipliers and n adders to implement in the digital computer. The above structure suffers extreme coefficient sensitivity as the value of grows large. That is a small change in a coefficient for large value of n causes large changes in the zeroes of $H(z)$.

4.2.2 TRANSPOSE-FORM STRUCTURE

The flow-graph-reversal theorem says that if one changes the directions of all the arrows, and inputs at the output and takes the output from the input of a reversed flow graph, the new system has an identical input-output relationship to the original flow graph [10].

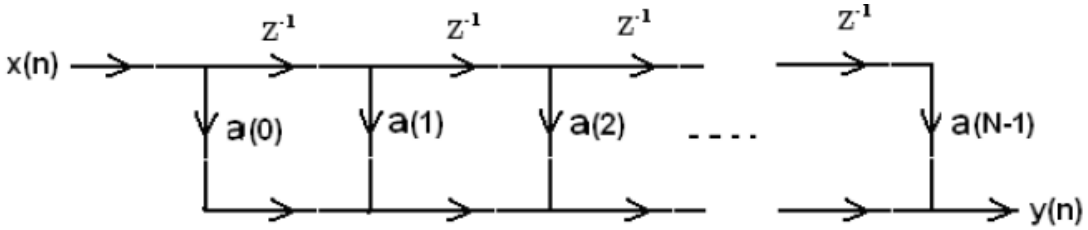


Figure 4.4 Direct form FIR Filter [10].

Now to get the transpose form of FIR filter we have to change the direction of all arrows.

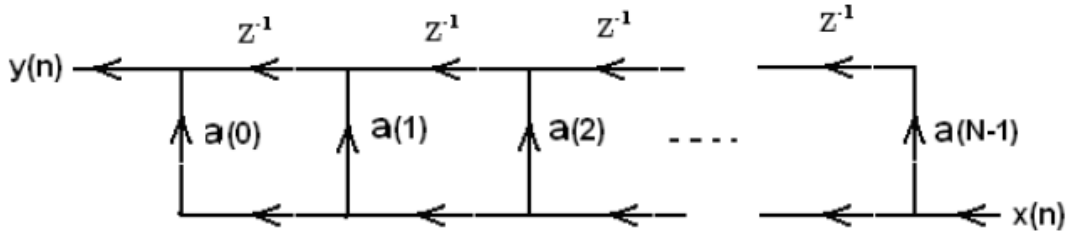


Figure 4.5 Transpose-form FIR filter structure [10].

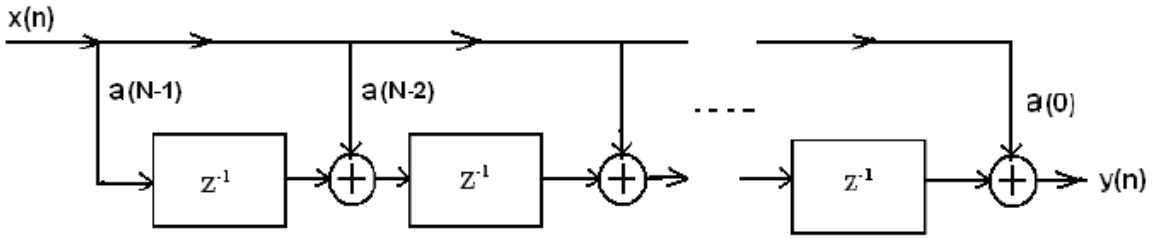


Figure 4.6 Transpose-form FIR filter structure[10].

4.2.3 CASCADE STRUCTURE

The z-transform of an FIR filter can be factored into a cascade of short-length filters[10].

$$b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots\ldots\ldots b_m z^{-m} = b_0(1 - z_1 z^{-1})(1 - z_2 z^{-2}) \ldots\ldots (1 - z_m z^{-1})$$

Where the z_i are the zeros of this polynomial[10].

Since the coefficients of the polynomial are usually real, the roots are usually complex-conjugate pairs, so we generally combine $(1 - z_j z^{-1})(1 - \bar{z}_j z^{-2})$ into one quadratic (length-2) section with real coefficients

The overall filter can then be implemented in a cascade structure.

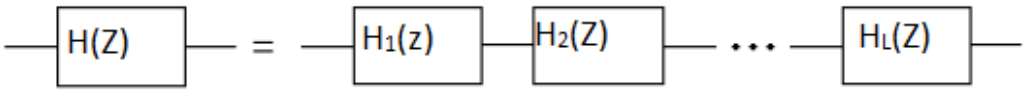


Figure 4.7 Cascaded Structures [10].

This is occasionally done in FIR filter implementation when one or more of the short length filters can be implemented efficiently.

4.2.4 LATTICE STRUCTURE

It is also possible to implement FIR filters in a lattice structure:

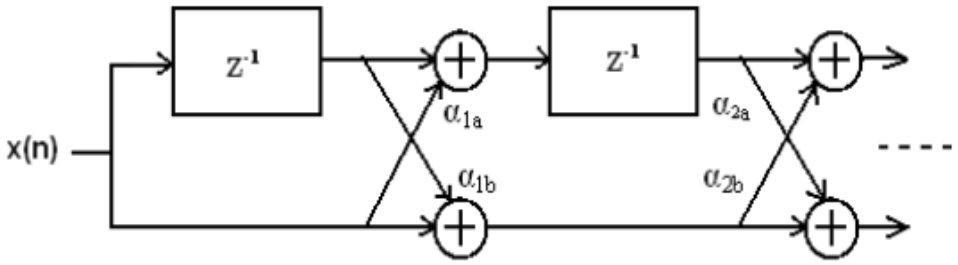


Figure 4.8 Lattice Structure [10].

This is sometimes used in adaptive filtering and digital speech processing. Lattice structure which exhibit robustness in finite word-length implementations[10].

4.3 COMPARISON OF VARIOUS STRUCTURE

The simplest of these structures, namely, the direct-form realizations. However, there are other more practical structures that offer some distinct advantages, especially when quantization effects are taken into consideration[10].

The cascade, parallel, and lattice structures, which exhibit robustness in finite word-length implementations. The frequency-sampling has the advantage of being computationally efficient when compared with alternative FIR realizations. Other filter structures are obtained by employing a state-space formulation for linear time-invariant system. Due to space limitations, state-space structures are not generally used.

CHAPTER 5

SIMULATION AND SYNTHESIS TOOLS

5.1 INTRODUCTION

The following tools for implementation of FIR filter on FPGA:

1. XILINX ISE web pack 9.2i for design, synthesis and implementation.
2. MODSIM 5.5c for simulation.

5.2 SIMULATION TOOLS

Very High Speed Integrated Circuit Hardware Description Language (VHDL) is used as the designing language. Hardware Description Languages (HDLs) are used to describe the behaviour and structure of system and circuit designs. [20]

5.2.1 ADVANTAGES OF USING HDLS TO DESIGN FPGAS

- Top Down Approach:- HDLs are used to create complex designs. The top-down approach to system design supported by HDLs is advantageous for large projects that require many designers working together[23].
- Functional Simulation Early in the Design Flow :- One can verify the functionality of your design early in the design flow by simulating the HDL description[23].
- Synthesis of HDL Code to Gates:- One can synthesize your hardware description to a design implemented with gates. This step decreases design time by eliminating the need to define every gate[23].
- Early Testing of Various Design Implementations:- HDLs allows one to test different implementations of your design early in the design flow. One can then use the synthesis tool to perform the logic synthesis and optimization into gates[23].
- Reuse of RTL Code:- One can retarget RTL code to new FPGA architectures with a minimum of recoding[23].

5.2.2 BASICS OF VHDL

VHDL stands for Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (HDL). It is a language for describing digital electronic systems. It was born out of united States Government's VHSIC program in 1980 and was adopted as a standard for describing the structure and function of Integrated Circuits (ICs). Soon after, it was developed and adopted as a standard by the Institute of Electrical and Electronic Engineers (IEEE) in the US (IEEE-1076-1987) and in other countries. VHDL continues to evolve. Although new standards have been prepared (VHDL-93) most commercial VHDL tool use 1076-1987 version of VHDL, thus making it most compatible when using different compilation tools [8]. VHDL enables the designer to[23]:

- Describe the design in its structure, to specify how it is decomposed into sub-designs, and how these sub-designs are interconnected.
- Specify the function of designs using a familiar, C-like programming language form.
- Simulate the design before sending it off for fabrication, so that the designer has a chance to rapidly compare alternative approach and test for correctness without the delay and expense of multiple prototyping.

VHDL is a C-like, general purpose programming language with extensions to model both concurrent and sequential flows of execution, and allowing delayed assignment of values. To a first approximation, VHDL can be considered to be a combination of two languages:
one describing the structure of the integrated circuit and its interconnections (structural description) and the other one describing its behaviour using algorithmic constructs (behavioural description).

VHDL allows three styles of programming:

- Structural
- Register Transfer Level (RTL)
- Behavioural

The first one, structural, is the most commonly used as it allows description of the structure of the IC very precisely by the user. Its behavioural style permits the designer to quickly test

concepts, where the designer can specify the high-level function of the design without taking much care how it will be done structurally. This can be very attractive for quick design of low and medium speed and low-volume applications,where the designer expertise is not available.

5.3 SYNTHESIS TOOLS

5.3.1 XILINX ISE 9.2I OVERVIEW

WebPACK ISE design software offers a complete design suite based on the Xilinx ISE series software. Individual WebPACK ISE modules give us the ability to the design environment to our chosen PLDs as well as the preferred design flow. In general, the design flow for FPGAs and CPLDs is identical[22].

WebPACK ISE software offers an easy-to-use GUI to visually create a test pattern. A test bench is then generated and compiled into MXE, along with the design under test. This XILINX has been used for synthesis and implementation of our design. The flow diagram below shows the similarities and differences between CPLD and FPGA software flows.

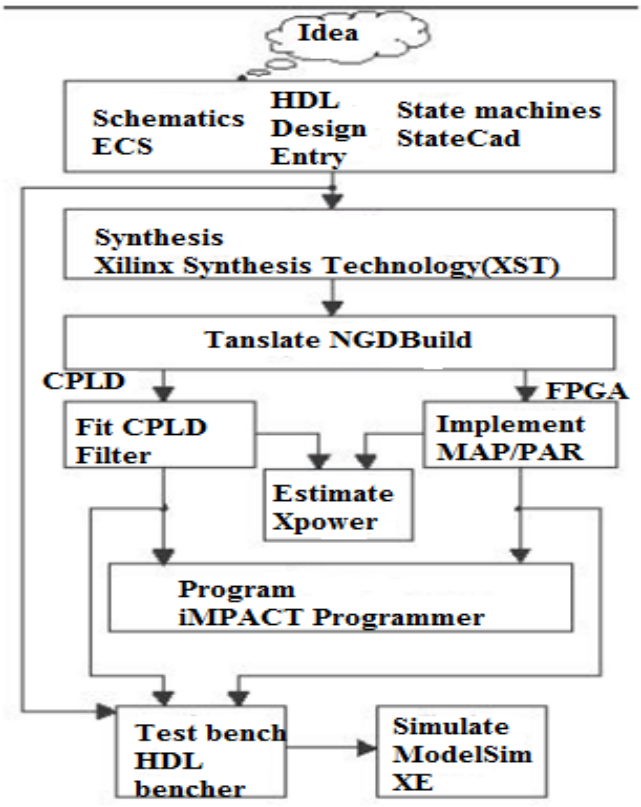


Figure 5.1: Webpack software design flow [20].

This powerful simulator is capable of simulating functional VHDL before synthesis, or simulating after the implementation process for timing verification.
WebPACK ISE software incorporates a Xilinx version of the ModelSim simulator from Model Technology (a Mentor Graphics company), referred to as MXE (ModelSim Xilinx Edition). In a diagrammatic form and let the software tools generate optimized code from a state diagram.

The various steps involved are as follows:

1. Synthesis: Synthesis is the general term that describes the process of transformation of the model of a design in HDL, from one level of Behavioural abstraction to a lower, more detailed level. With reference to VHDL, synthesis is an automatic method of converting a higher level of abstraction to a lower level of abstraction. The synthesis tools convert RTL descriptions to gate level net-lists. The preparation of a synthesizable model requires the knowledge about features. It is important here to note that not all features of VHDL can be synthesized; therefore, one must consult Xilinx Simulation and Synthesis Guide for a list of synthesizable features.

2. Implementation: It is divided into three major operations:
   (i) Translation**:** Merges all of the input net lists.
   (ii) Mapping**:** Map optimizes the gates and removes unused logic. This step also maps the designs logic resources.
   (iii) Place and Route**:** The Place and Route process places each macro from the synthesis net list into an available on the target silicon and connects the macros using routing resources available on the target silicon. The job of the place and route tool is to create the programming files that will be used to specify the logic function of the logic macros in the logic areas and the switch programming of the wires used to connect the macros together. Each switch adds capacitance and resistance to the routed signal. After a few connections, signals start to slow significantly because of capacitance and resistance of the line. The place and route tools can make tradeoffs if speed critical signals are known ahead of time and is implemented using the highest speed interconnecting signals. The placement algorithm also tries to place logical gates on the critical path close to each other so that local interconnect can used to connect the gates.

3. Generation of Programming File: This feature generates the bit file to be downloaded on to the target device (FPGA/PROM) using the downloading cable.
   Thereafter, the following tools are used to program the device:
   (i) iMPACT
   (ii) PROM File Formatter

The iMPACT programmer module allows you to program a device in-system for all devices available in the WebPACK software. (we must connect a JTAG cable to the PC.s parallel port.) For FPGAs, the programmer module allows you to configure a device via the JTAG. iMPACT, a command line and GUI based tool, allows one to:

a) Configure FPGA designs using Boundary-Scan, Master Serial
b) Download
c) Read-Back and Verify design configuration data
d) Perform functional tests on any device.

5.4 FPGAS: AN OVERVIEW

An FPGA is a completely reconfigurable computer logic chip. Like traditional hardwired gate arrays, the chip consists of a series of logic gates. In the traditional array, these gates are specified and hard interconnected at the manufacturing stage. The field programmable gate array differs in that it can be programmed, and re-programmed, insitu. This has the advantages of allowing fast prototyping for applications it is intended to be implement with hard-wired chips[20].

5.4.1 COMPUTER AIDED DESIGN FOR VLSI CIRCUITS

The design of digital systems with VLSI circuits containing millions of transistors is a formidable task and requires the assistance of computer-aided design (CAD) tools. CAD tools consist of software programs that support computer-based representation and aid in the development of digital hardware by automating the design process. The designer can choose between a full-custom IC, a programmable logic device (PLD), an application specific integrated circuit (ASIC), or a field-programmable gate array (FPGA) [20].

5.4.2 PROGRAMMABLE LOGIC

Programmable logic is loosely defined as a device with configurable logic and flip-flops linked together with programmable interconnect. Memory cells control and define the function that the logic performs and how the various logic functions are interconnected. What kinds of programmable logic devices are available today? How are they different from one another?

There are a few major programmable logic architectures available today. Each architecture typically has vendor-specific sub-variants. The major types include: [22]

- Simple Programmable Logic Devices (SPLDs)
- Complex Programmable Logic Devices (CPLDs)
- Field Programmable Gate Arrays (FPGAs)

5.4.3 FPGA - FIELD PROGRAMMABLE GATE ARRAY

The FPGA is advancing rapidly as highly important element of the future of computing. Already development have shown that it can massively reduce the price of specialized system development and it can compete on a variety of attributes with the top range commercially available microprocessors. Its initial role in rapid system prototyping is still important but in recent times it has grown in importance as a platform for implementing complete solutions. The ability to implement a fully functional system, microcontroller or even full blown computer using FPGAs and the recent advances in development software lead to highly exciting possibilities with regards to the development of complete re-configurable computing systems[20].

There are four main categories of FPGAs currently commercially available: symmetrical array, row-based, hierarchical PLD, and sea-of-gates (Figure 5.2). In all of these FPGAs the interconnections and how they are programmed vary.

The basic FPGA architecture consists of a two-dimensional array of logic blocks and flip-flops with means for the user to configure [20].

(i) the function of each logic blocks,

(ii) the inputs/outputs, and

(iii) the interconnection between blocks.

Families of FPGAs differ from each other by the physical means for implementing user programmability, arrangement of interconnection wires, and basic functionality of the logic blocks.

Currently there are four technologies in use. They are static RAM cells, anti-fuse, EPROM transistors, and EEPROM transistors. Depending upon the application, one FPGA technology may have features desirable for that application[21].

- Static RAM Technology**:** In the Static RAM FPGA programmable connections are made using pass transistors, transmission gates, or multiplexers that are controlled by SRAM cells. The advantage of this technology is that it allows fast in-circuit reconfiguration. The major disadvantage is the size of the chip required by the RAM technology.
- Anti-Fuse Technology**:** An anti-fuse resides in a high-impedance state and can be programmed into low impedance or "fused" state. A less expensive than the RAM technology, this device is a program-one device.
- EPROM / EEPROM Technology**:** This method is the same as used in the EPROM memories. One advantage of this technology is that it can be reprogrammed without external storage of configuration; though the EPROM transistors cannot be reprogrammed in-circuit [21].

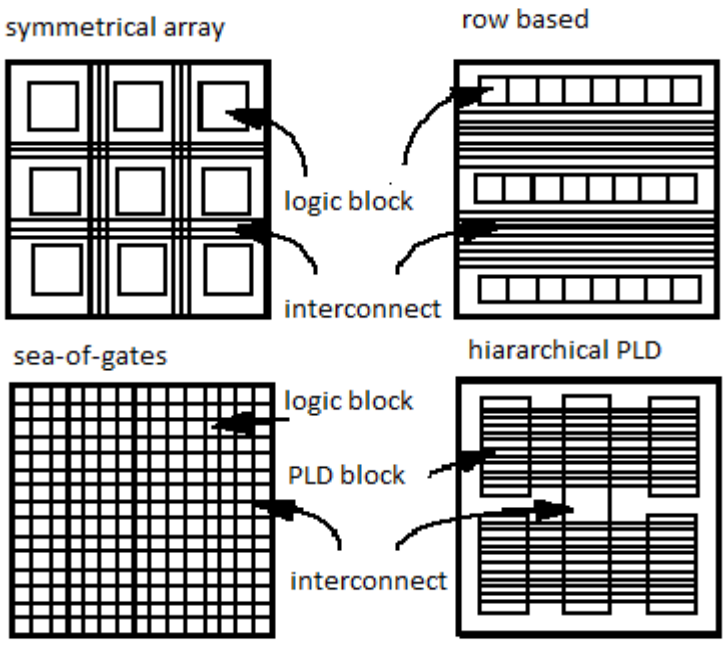


Figure 5.2 Classes of FPGAs [21].

The following table shows some of the commercially available FPGAs.

Table 5.1 Commercial FPGA Technology [21].

Company name	**Architecture**	**Logic block type**	**Programming technology**
Actel	Row-based	Multiplexer-Based	anti-fuse
Altera	Hierarchical-PLD	PLD Block	EPROM
Quick Logic	Symmetrical Array	Multiplexer-Based	anti-fuse
Xilinx	Symmetrical Array	Look-up Table	Static RAM

5.5 THE DESIGN FLOW

This Section examines the design flow for any device, whether it is an ASIC, an FPGA, or a CPLD. The design flow consists of the steps in[21]:

STEP 1:WRITING A SPECIFICATION

The importance of a specification cannot be overstated. This is an absolute must, especially a as guide for choosing the right technology and for making your needs known to the vendor. As specification allows each engineer to understand the entire design and his or her piece of it. It allows the engineer to design the correct interface to the rest of the pieces of the chip. It also saves time and misunderstanding. There is no excuse for not having a specification.
A specification should include the following information:

- An external block diagram showing how the chip fits into the system.
- An internal block diagram showing each major functional section.
- A description of the I/O pins including
- Output drive capability
- Input threshold level
- Timing estimates including
- Setup and hold times for input pins
- Propagation times for output pins

- Clock cycle time
- Estimated gate count
- Package type
- Target power consumption
- Target price
- Test procedures

STEP 2: CHOOSING A TECHNOLOGY

Once a specification has been written, it can be used to find the best vendor with a technology and price structure that best meets your requirements.

Step 3: CHOOSING A DESIGN ENTRY METHOD

You must decide at this point which design entry method you prefer. For smaller chips, schematic entry is often the method of choice, especially if the design engineer is already familiar with the tools. For larger designs, however, a hardware description language (HDL) such as Verilog or VHDL is used because of its portability, flexibility, and readability. When using a high level language, synthesis software will be required to synthesize the design. This means that the software creates low level gates from the high level description. This is the entire process for designing a device that guarantees that you will not overlook any steps and that you will have the best chance of getting backs a working prototype that functions correctly in your system.

STEP 4: CHOOSING A SYNTHESIS TOOL

You must decide at this point which synthesis software you will be using if you plan to design the FPGA with an HDL. This is important since each synthesis tool has recommended or mandatory method of designing hardware so that it can correctly perform synthesis. It will be necessary to know these methods up front so that sections of the chip will not need to be redesigned later on. At the end of this phase it is very important to have a design review and that the correct technology and design entry method have been chosen.

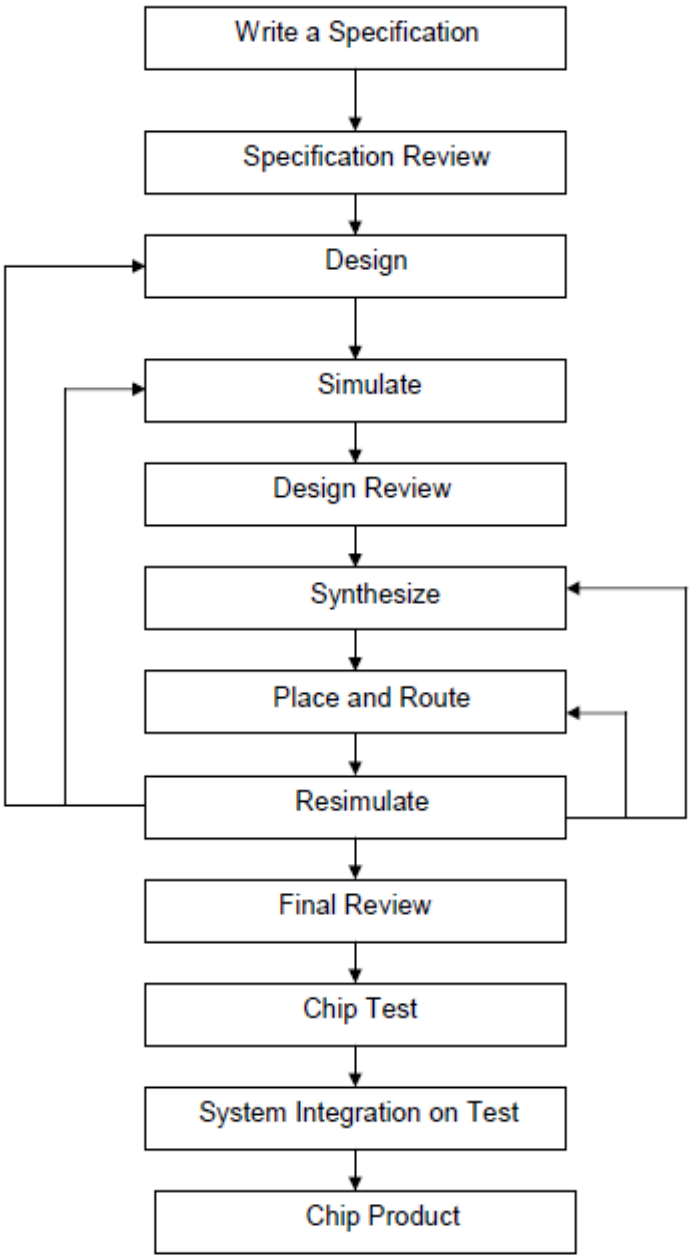


Figure 5.3 FPGA Design Flow [21].

STEP 5: DESIGNING THE CHIP

It is very important to follow good design practices. This means taking into account the following design issues that we discuss in detail later in this chapter.

- Top-down design
- Use logic that fits well with the architecture of the device you have chosen
- Macros

- Synchronous design
- Protect against meta stability
- Avoid floating nodes
- Avoid bus contention

STEP 6: SIMULATING - DESIGN REVIEW

Simulation is an ongoing process while the design is being done. Small sections of the design should be simulated separately before hooking them up to larger sections. There will be much iteration of design and simulation in order to get the correct functionality. Once design and simulation are finished, another design review must take place so that the design can be checked. It is important to get others to look over the simulations and make sure that nothing was missed and that no improper assumption was made. This is one of the most important reviews because it is only with correct and complete simulation that you will know that your chip will work correctly in your system.

STEP 7: SYNTHESIS

If the design was entered using an HDL, the next step is to synthesize the chip. This involves using synthesis software to optimally translate your register transfer level (RTL) design into a gate level design that can be mapped to logic blocks in the FPGA. This may involve specifying switches and optimization criteria in the HDL code, or playing with parameters of the synthesis software in order to insure good timing and utilization.

Step 8: PLACE AND ROUTE

The next step is to lay out the chip, resulting in a real physical design for a real chip. This involves using the vendor's software tools to optimize the programming of the chip to implement the design. As implied by the name, it is composed of two steps, placement and routing. The first step, placement, involves deciding where to place all electronic components, circuitry, and logic elements in a generally limited amount of space. This is followed by routing, which decides the exact design of all the wires needed to connect the placed components. Then the design is programmed into the chip.

STEP 9: RESIMULATING - FINAL REVIEW

After layout, the chip must be resimulated with the new timing numbers produced by the actual layout. If everything has gone well up to this point, the new simulation results will agree with the predicted results. Otherwise, there are three possible paths to go in the design flow. If the problems encountered here are significant, sections of the FPGA may need to be redesigned. If there are simply some marginal timing paths or the design is slightly larger than the FPGA, it may be necessary to perform another synthesis with better constraints or simply another place and route with better constraints. At this point, a final review is necessary to confirm that nothing has been overlooked.

STEP 10: TESTING

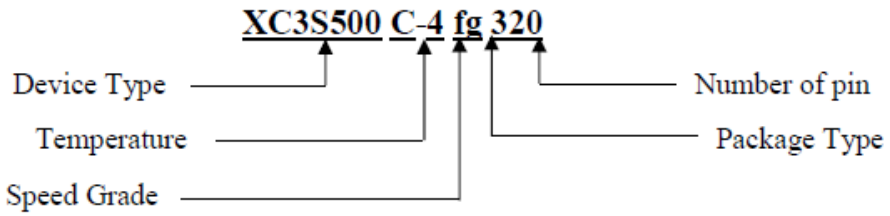
For a programmable device, you simply program the device and immediate have your prototypes. You then have the responsibility to place these prototypes in your system and determine that the entire system actually works correctly. If you have followed the procedure up to this point, chances are very good that your system will perform correctly with only minor problems. These problems can often be worked around by modifying the system or changing the system software. These problems need to be tested and documented so that they can be fixed on the next revision of the chip. System integration and system testing is necessary at this point to insure that all parts of the system work correctly together. When the chips are put into production, it is necessary to have some sort of burn-in test of your system that continually tests your system over some long amount of time. If a chip has been designed correctly, it will only fail because of electrical or mechanical problems that will usually show up with this kind of stress testing.

5.6 SPARTAN-III FPGA KIT

The Xilinx Spartan 3E starter board, made by Digilent Inc. uses a XC3S500E FPGA. It has a following feature, such as Flash Memory, DDR SDRAM, LCD display, ADCs, DACs, RS232, VGA, Ethernet Phy and much more. It has a number of 6 pin headers for adding small 4 bit modules, as well as a Hirose 100 pin FX2 connector, which can be used for such things as the VDEC-1 Video digitizer[22].

The limitation of the Spartan 3E start board is that there is no SRAM, which means we need a DDR-SDRAM controller core to use it unless we are using EDK. There may be a DDR SDRAM controller in ISE somewhere. Also, like the Spartan 3 starter board, the VGA connector only has 3 bits connected to it which means there are only 8 colours which limits it's use in displaying digitized images.

Xilinx Spartan FPGAs are ideal for low-cost, high volume applications. The Spartan-III family is based on IBM and UMC advanced 90 nm, eight layer metal process technology. Xilinx uses 90 nm technology to drive pricing down to under $20 for a onemillion-gate FPGA (approximately 17,000 logic cells), which represents a cost savings as high as 80 percent compared to competitive offerings. Our kit was XC3S500C-4fg320 Spartan-III device.The device, which we are using, has the following specifications:



CHAPTER 6

IMPLEMENTATION OF FIR FILTER ON FPGA

6.1 REALIZATION OF FIR FILTER

The realization of FIR filters can be accomplished by using the following design procedure[8]:

1. Choose filter structure
2. Choose between fixed-point and floating-point arithmetic.
3. Choose number representation, e.g. signed magnitude, two's compliment
4. Choose between serial and parallel processing
5. Implement software code, or hardware circuit, which will perform actual filtering.
6. Verify the simulation that the resulting design meets given performance specifications.

6.2 PROCESS OF IMPLEMENTING FIR FILTER

The analog input signal must be sampled first and digitized using an ADC (analog-to-digital converter). The resulting binary numbers, representing successive sampled values of the input signal, are transferred to the processor, which carries out numerical calculations on them. These calculations typically involve multiplying the input values by constants and adding the products together. If necessary, the results of these calculations, which now represent sampled values of the filtered signal, are output through a DAC (digital-to-analog converter) to convert the signal back to analog form[8].

6.2.1 RESTRICTION AND ASSUMPTION

There are certain assumptions and restrictions in this implementation which are as follows[8]:

1) The input and output are in the digital form.
2) For filter coefficient, used in this thesis are calculated using windowing technique.

3) Filter is considered as symmetric.

6.2.2 CHOOSING THE FILTER STRUCTURE

It is often important to choose a particular filter structure for a given transfer function H(z). In the design of fixed point, digital filters the choice is usually based on minimizing the effects of finite register lengths. These effects include round-off noise, coefficient sensitivity, overflow oscillations, and zero input limit cycles[19].

There are direct-form structures of FIR Filter These Direct structures are effected by coefficient sensitivity problems, which means, for large value of the order of filter the poles (in case of recursive filters) and zeroes locations could be changed. In our project, Direct form of FIR filter has been implemented whose new look is given in Figure 6.1.

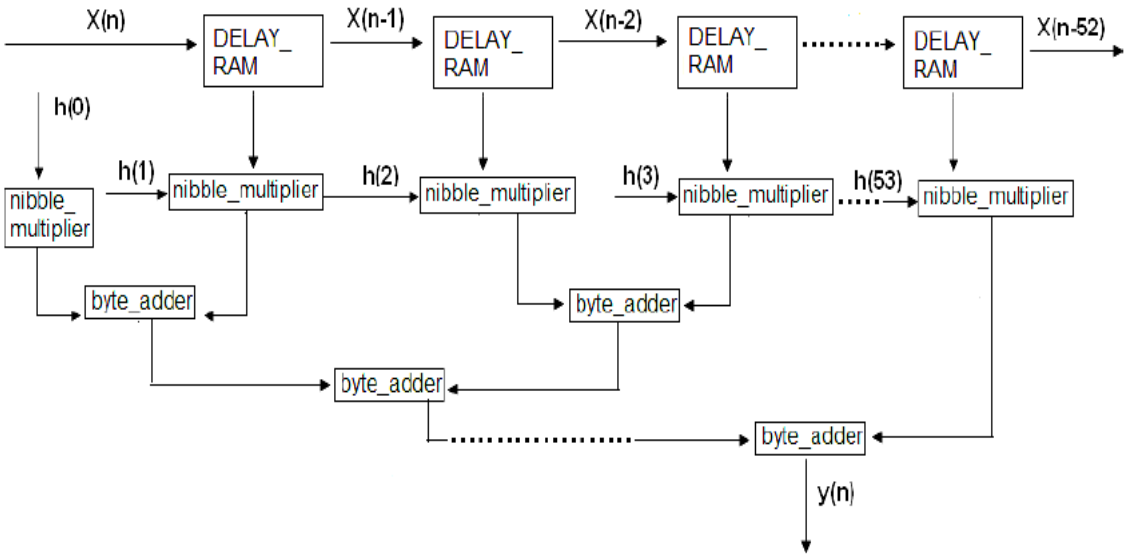


Figure 6.1 Direct-Form Structure [23].

6.2.3 DATA REPRESENTATION

In general, there are two kinds of Data representation, one is fixed-point representation, and the other is IEEE floating-point representation.

In this Thesis, the procedure of representing the filter coefficients and input samples is given as below:

A binary point is usually set between the first and second bit positions of the register as shown in Figure 6.2 is as given below [19].

The addition or subtraction of two fixed-point numbers falling in a given range may produce a result outside that range, though. Such a result, called overflow, it must be either avoided, or corrected during DSP calculations. We are avoiding here.

The data is represented in fixed-point notation. In the fixed-point format, the numbers are usually assumed proper fraction.

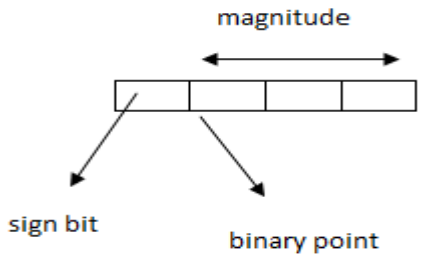


Figure 6.2 Fixed Point Representation

6.3 MODULE FOR IMPLEMENTING FIR FILTER

The modules used for implementing FIR filter are as follows: we need three module to implement the FIR filter multiplication module, addition module, delay and storing module[19].

6.3.1 ADDITION MODULE (BYTE ADDER)

It will add two numbers with taking care of negative number. If one of them or both are negative, then before addition, negative number will be converted into two's compliment format and then it will be added.

It will check whether result is overflowing or not if it is overflowing then it will take two's compliment of result[19].

6.3.2 DELAY AND STORING MODULE (DELAY RAM)

Input will be shifted to right and result will be storing into temporary register. Shifted input should have array size one more as compare to input but here array size remains constant[19].

The above modules are used to implement the filter structure and the results are discussed in next chapter.

6.3.3 MULTIPLICATION MODULE (NIBBLE MULTIPLIER)

It is known that the multiplication operation takes more cycles than an adder or shift register operation. If the number of multiplications in the structure is more , then more time is needed to perform the filtering operation. Thus, the speed of operation will be affected[23].

In our project adders and time-shifters, replace the multipliers, thereby increasing the speed of operation as compared to traditional filter structures in which multipliers were present.
The multiplication is the basic operation in computation of output y(k). Considering the multiplication of two n-bit numbers, we have the product of 2n bits. This way of multiplication is implemented in project while multiplying the coefficients and the input samples.

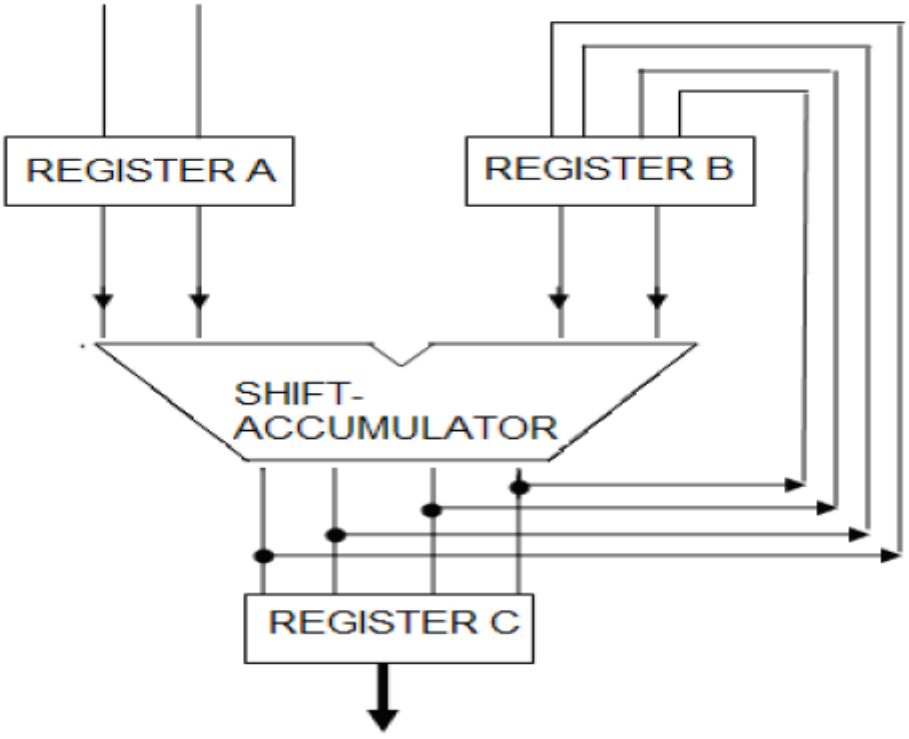


Figure 6.3 Multiplication Modules [23].

CHAPTER 7
Results and Discussion

In this chapter, the lowpass, bandpass and highpass filters are implemented on FPGA using real world examples. The filter specifications are real world and windowing method is used to design the filter coefficients. These coefficients are used to implement filter on Xilinx FPGA Spartan 3E kit using Xilinx ISE 9.2i.

Here we are using two methods to implement the filter on FPGA. We are using fixed package to enhance the speed of FIR filter.

(1) Fully parallel method using fixed package
(2) Fully serial method using fixed package

7.1 LOW PASS FILTER

Although any filter specifications can be taken but for the sake of implementation the following specifications are considered for lowpass filter:

Passband edge frequency =1.5 kHz
Transition width =0.5 kHz
Stopband attenuation > 50 dB
Sampling frequency =8 kHz

The filter order of given specification is calculated using Hamming window because here stopband attenuation > 50 dB.

$$\Delta f = 0.5/8 = 0.0625 \quad (7.1)$$

$$N = 3.3/\Delta f = 52.80 \approx 53 \quad (7.2)$$

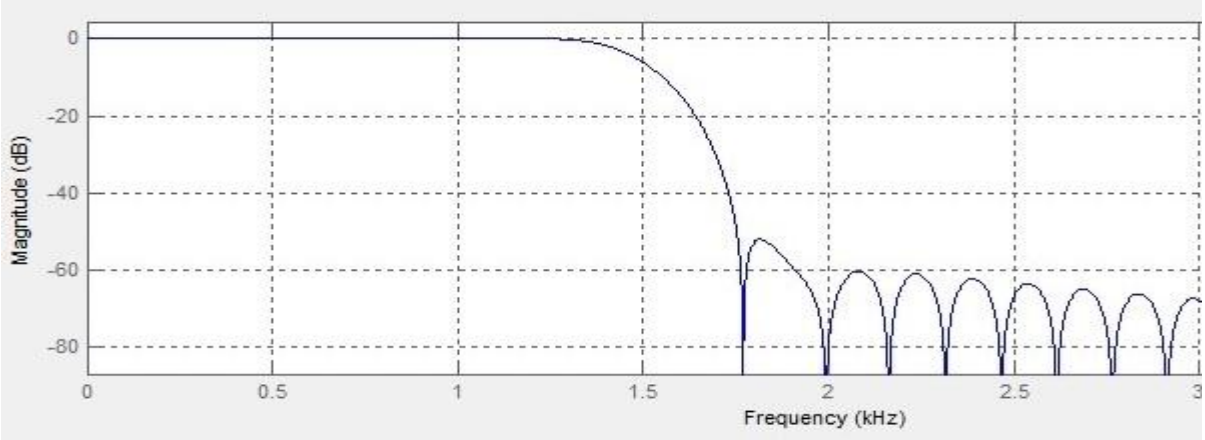


Figure 7.1 Magnitude Response of LPF in dB

The designed coefficients are given in Appendix-I. These coefficient are directly use in VHDL Code. Now this VHDL code is used to generate the circuit using Xilinx synthesis tool for low pass filter design. The main circuit block is shown in Figure 7.2.

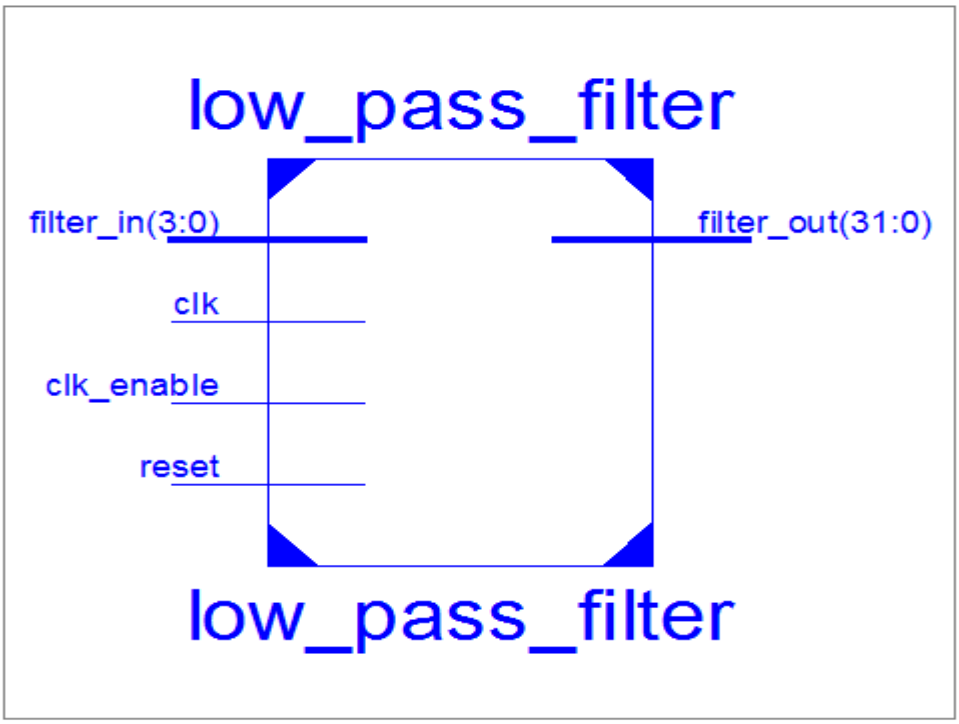


Figure 7.2 Block diagram of the Low Pass Filter (LPF).

Now the generated circuit is simulated using Xilinx simulator with certain inputs (8, 9 and 10) and for impulse input the corresponding input and output waveform are shown in Fig 7.3.

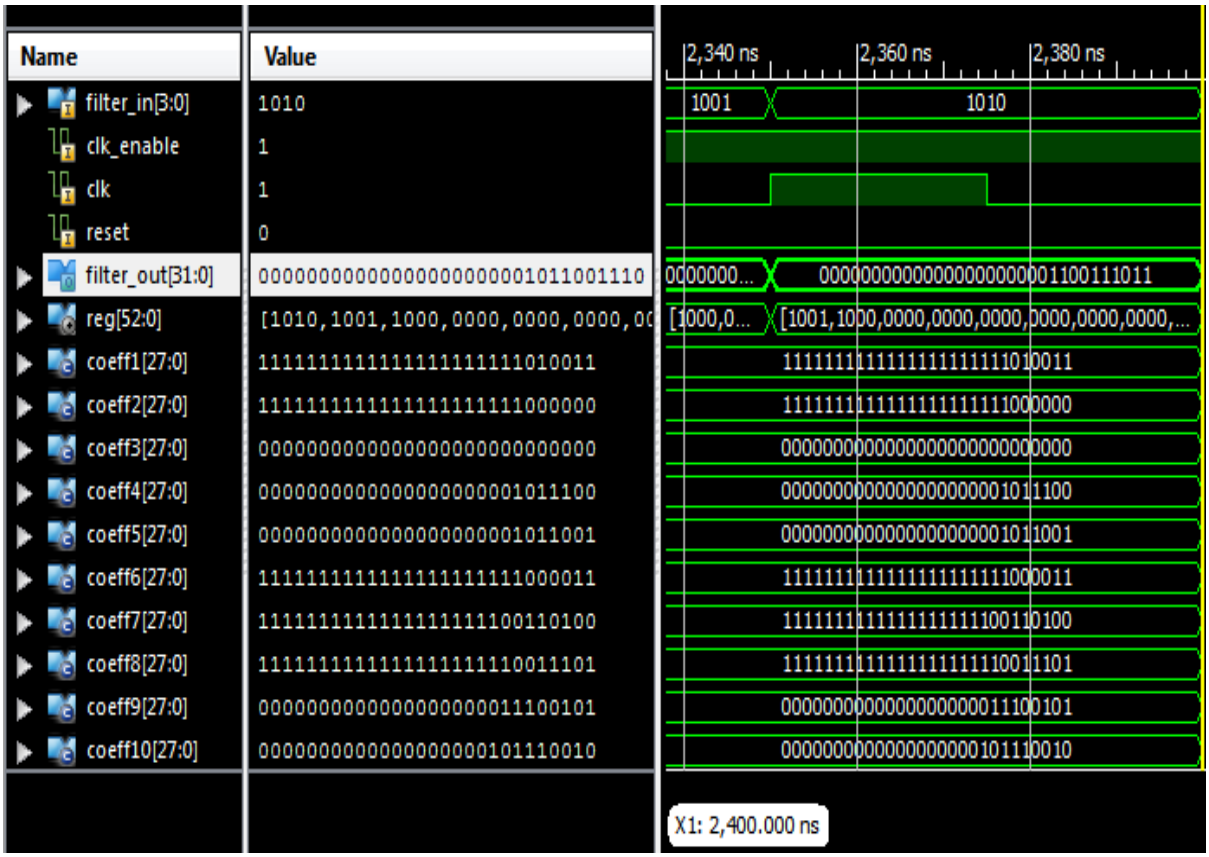


Figure 7.3 Input and Output waveform of Simulation

Table 7.1 Advanced HDL Synthesis Report of Parallel LPF

```
=========================================================================
*                      Advanced HDL Synthesis                           *
=========================================================================


=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Multipliers                                          : 45
 4x28-bit multiplier                                   : 45
# Adders/Subtractors                                   : 48
 32-bit adder                                          : 2
 33-bit adder                                          : 46
# Registers                                            : 212
 Flip-Flops                                            : 212
# Multiplexers                                         : 51
 32-bit 4-to-1 multiplexer                             : 51

=========================================================================
```

Table 7.2 Advanced HDL Synthesis Report of Serial LPF

```
=========================================================================
*                      Advanced HDL Synthesis                           *
=========================================================================


=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Multipliers                                          : 1
 28x4-bit multiplier                                   : 1
# Adders/Subtractors                                   : 1
 33-bit adder                                          : 1
# Counters                                             : 1
 32-bit up counter                                     : 1
# Registers                                            : 308
 Flip-Flops                                            : 308
# Multiplexers                                         : 1
 32-bit 4-to-1 multiplexer                             : 1

=========================================================================
```

Table 7.3 Timing Summary Report of Parallel LPF

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 1.984ns (Maximum Frequency: 504.032MHz)
   Minimum input arrival time before clock: 3.099ns
   Maximum output required time after clock: 261.684ns
   Maximum combinational path delay: No path found
```

Table 7.4 Timing Summary Report of Serial LPF

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 26.447ns (Maximum Frequency: 37.811MHz)
   Minimum input arrival time before clock: 5.969ns
   Maximum output required time after clock: 4.283ns
   Maximum combinational path delay: No path found
```

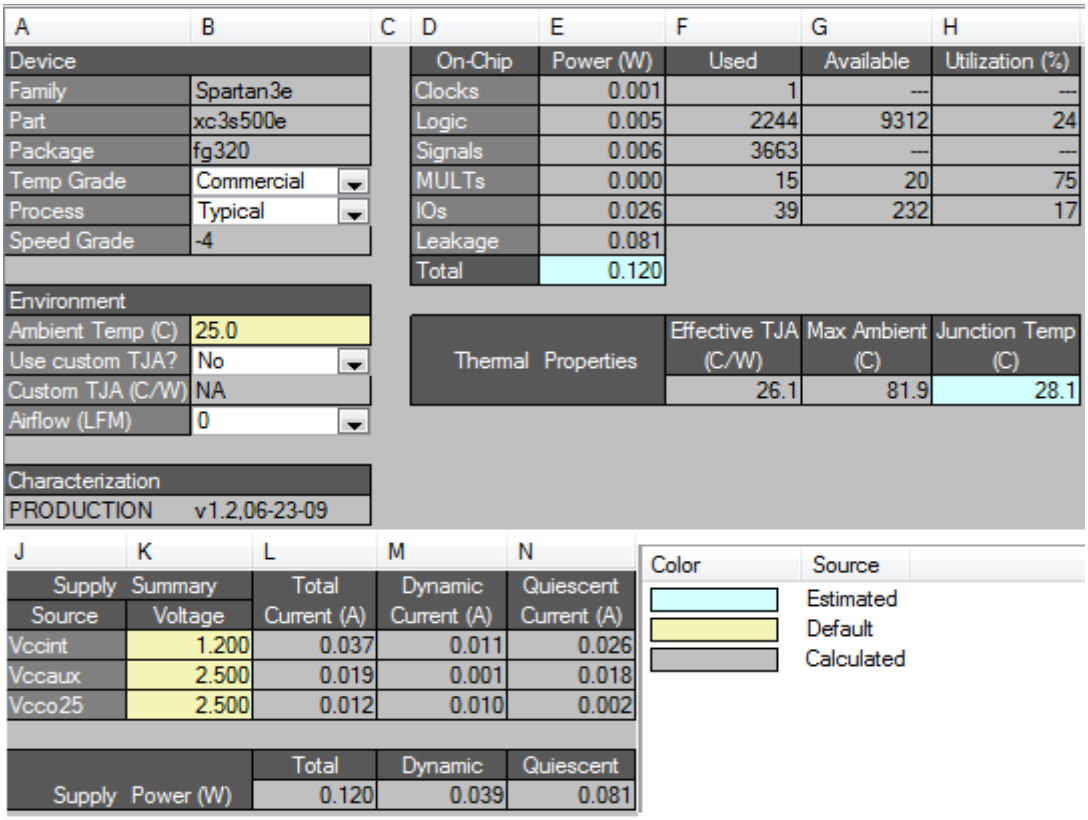
Table 7.5 Design Summary Report of Parallel LPF.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	233	9,312	2%
Number of 4 input LUTs	2,138	9,312	22%
Number of occupied Slices	1,287	4,656	27%
Number of Slices containing only related logic	1,287	1,287	100%
Number of Slices containing unrelated logic	0	1,287	0%
Total Number of 4 input LUTs	2,244	9,312	24%
Number used as logic	2,138		
Number used as a route-thru	106		
Number of bonded IOBs	39	232	16%
Number of BUFGMUXs	1	24	4%
Number of MULT18X18SIOs	15	20	75%
Average Fanout of Non-Clock Nets	2.60		

Table 7.6 Design Summary Report of Serial LPF.

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	343	9,312	3%
Number of 4 input LUTs	870	9,312	9%
Number of occupied Slices	603	4,656	12%
Number of Slices containing only related logic	603	603	100%
Number of Slices containing unrelated logic	0	603	0%
Total Number of 4 input LUTs	901	9,312	9%
Number used as logic	870		
Number used as a route-thru	31		
Number of bonded IOBs	39	232	16%
Number of BUFGMUXs	1	24	4%
Number of MULT18X18SIOs	2	20	10%
Average Fanout of Non-Clock Nets	3.59		

Table 7.7 Power Summary of Parallel Low Pass filter



Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-4

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.001	1	---	---
Logic	0.005	2244	9312	24
Signals	0.006	3663	---	---
MULTs	0.000	15	20	75
IOs	0.026	39	232	17
Leakage	0.081			
Total	0.120			

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	81.9	28.1

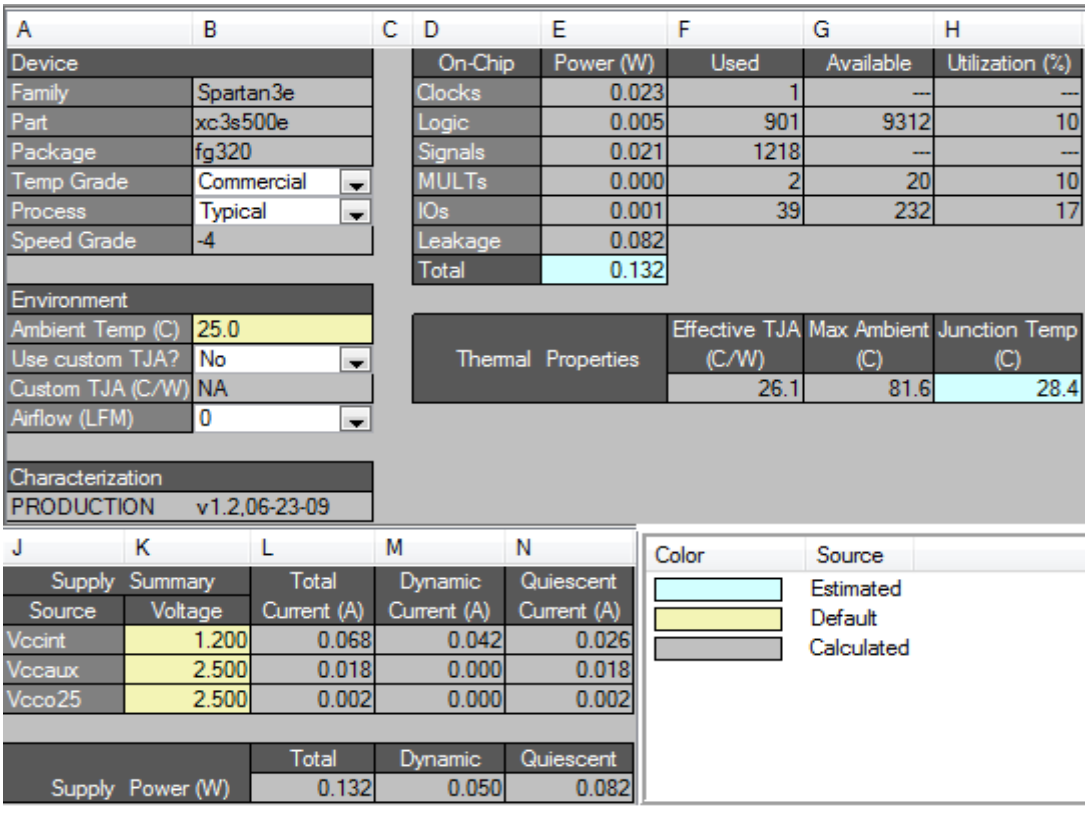
Characterization	
PRODUCTION	v1.2,06-23-09

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.037	0.011	0.026
Vccaux	2.500	0.019	0.001	0.018
Vcco25	2.500	0.012	0.010	0.002

Supply Power (W)	Total	Dynamic	Quiescent
	0.120	0.039	0.081

Color	Source
	Estimated
	Default
	Calculated

Table 7.8 Power Summary of Serial Low Pass filter



Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-4

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.023	1	---	---
Logic	0.005	901	9312	10
Signals	0.021	1218	---	---
MULTs	0.000	2	20	10
IOs	0.001	39	232	17
Leakage	0.082			
Total	0.132			

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	81.6	28.4

Characterization	
PRODUCTION	v1.2,06-23-09

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.068	0.042	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.002	0.000	0.002

Supply Power (W)	Total	Dynamic	Quiescent
	0.132	0.050	0.082

Color	Source
	Estimated
	Default
	Calculated

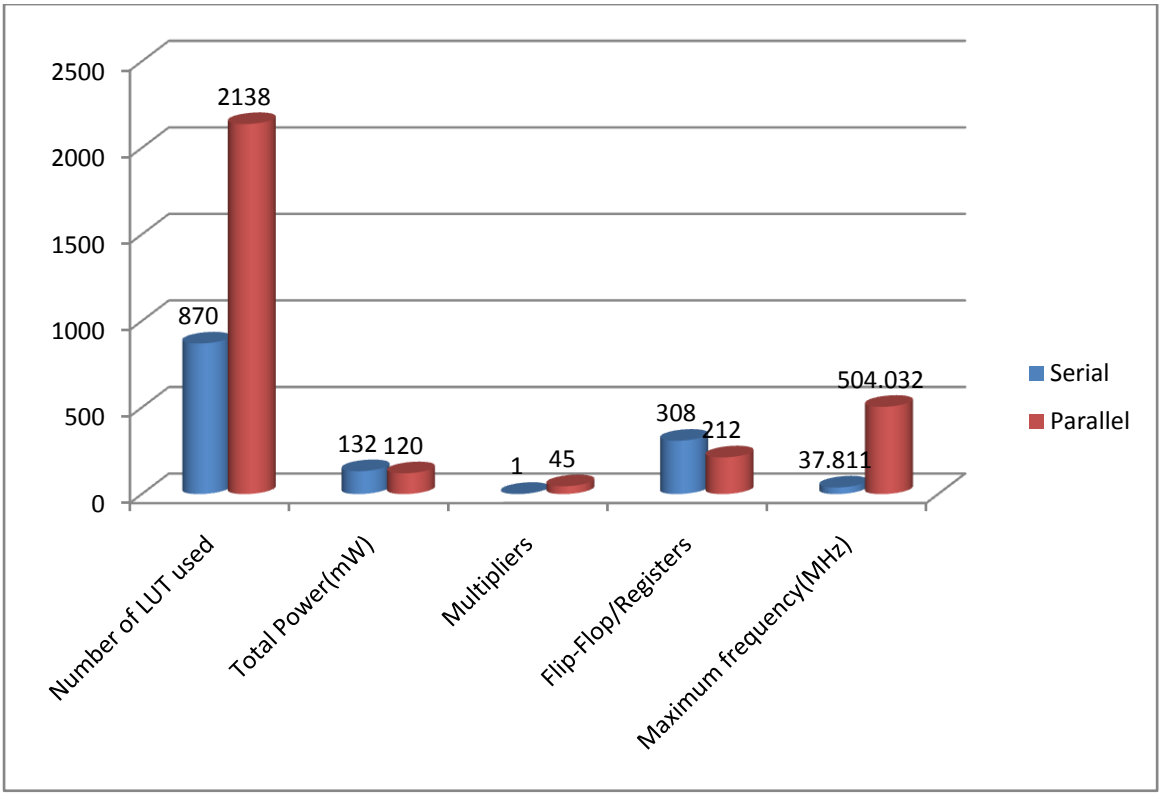


Figure 7.4 Performance chart of fully parallel and fully serial filter for LPF

7.2 BANDPASS FILTER

Although any filter specifications can be taken but for the sake of implementation the following specifications are considered for bandpass filter:-

Passband edge frequency =150 - 250 Hz
Transition width =50 Hz
Passband ripple = 0.1 dB
Stopband attenuation = 60 dB
Sampling frequency =1 kHz

From the specification, the passband and stopband ripples are

$$20\log(1+\delta_p) = 0.1dB, \quad \text{giving } \delta_p = 0.0115 \text{ and}$$
$$-20\log(\delta_s) = 60 \text{ dB}, \quad \text{giving } \delta_s = 0.001$$

Thus the attenuation requirements can be met by the Kaiser or the Blackman window.

For the Kaisar window, the number of filter coefficients is

$$N \geq A - \frac{7.95}{14.36\Delta F} = 60 - \frac{7.95}{14.36\left(\frac{50}{1000}\right)} = 72.49 \approx 73$$

Designed coefficients are given in Appendix II and these coefficients are directly used in VHDL code to generate the band pass filter.

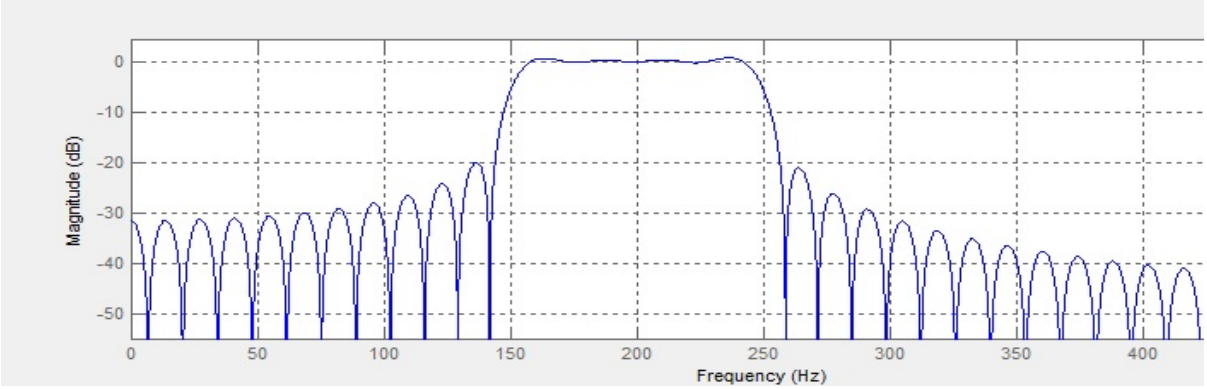


Figure 7.5 Magnitude Response of BPF in dB

Now this VHDL code is used to generate the circuit using Xilinx synthesis tool for bandpass filter design and main circuit block is shown in Figure 7.6.

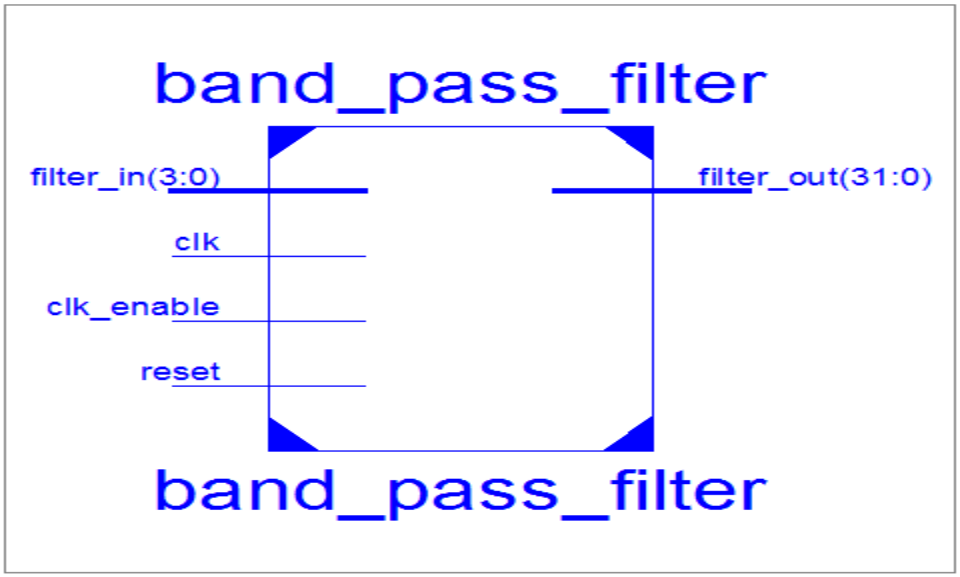


Figure 7.6 Block diagram of the Band Pass Filter (BPF).

Now the generated circuit is simulated using Xilinx simulator with inputs(1,3 and 4) and the corresponding input and output waveform showing in Fig7.7.

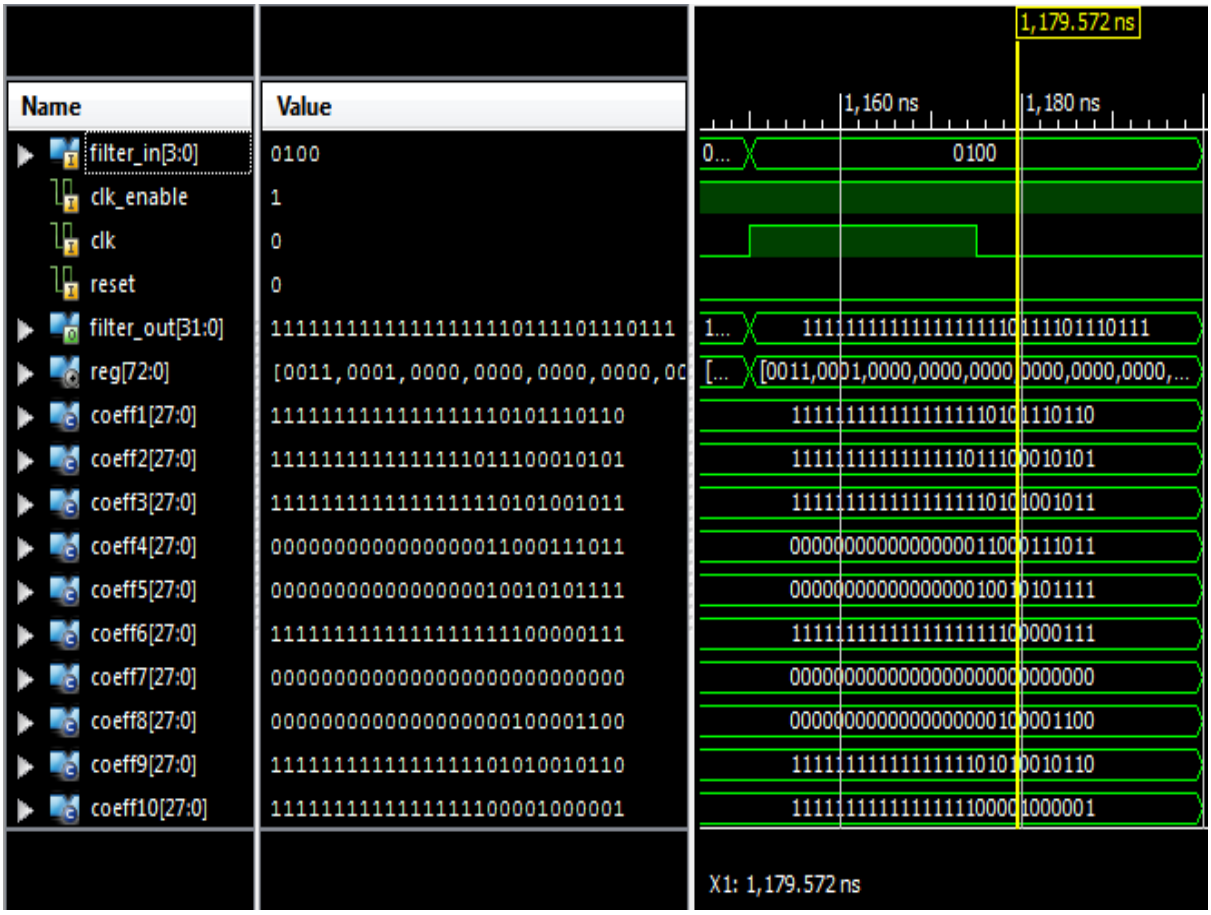


Figure 7.7 Input and Output waveform of simulation

Table 7.9 Advanced HDL Synthesis Report of Parallel BPF

```
=========================================================================
*                      Advanced HDL Synthesis                           *
=========================================================================


=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Multipliers                                        : 66
 4x28-bit multiplier                                 : 66
# Adders/Subtractors                                 : 65
 33-bit adder                                        : 65
# Registers                                          : 292
 Flip-Flops                                          : 292
# Multiplexers                                       : 71
 32-bit 4-to-1 multiplexer                           : 71

=========================================================================
```

Table 7.10 Advanced HDL Synthesis Report of Serial BPF

```
=========================================================================
*                      Advanced HDL Synthesis                           *
=========================================================================


=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Multipliers                                        : 1
 28x4-bit multiplier                                 : 1
# Adders/Subtractors                                 : 1
 33-bit adder                                        : 1
# Counters                                           : 1
 32-bit up counter                                   : 1
# Registers                                          : 388
 Flip-Flops                                          : 388
# Multiplexers                                       : 1
 32-bit 4-to-1 multiplexer                           : 1

=========================================================================
```

Table 7.11 Timing Summary of Parallel BPF

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 1.984ns (Maximum Frequency: 504.032MHz)
   Minimum input arrival time before clock: 3.133ns
   Maximum output required time after clock: 364.968ns
   Maximum combinational path delay: No path found
```

Table 7.12 Timing Summary of Serial BPF

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 28.740ns (Maximum Frequency: 34.795MHz)
   Minimum input arrival time before clock: 5.969ns
   Maximum output required time after clock: 4.283ns
   Maximum combinational path delay: No path found
```

Table 7.13 Design Summary of Parallel Band Pass Filter

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	341	9,312	3%
Number of 4 input LUTs	3,153	9,312	33%
Number of occupied Slices	1,893	4,656	40%
Number of Slices containing only related logic	1,893	1,893	100%
Number of Slices containing unrelated logic	0	1,893	0%
Total Number of 4 input LUTs	3,286	9,312	35%
Number used as logic	3,153		
Number used as a route-thru	133		
Number of bonded IOBs	39	232	16%
Number of BUFGMUXs	1	24	4%
Number of MULT18X18SIOs	14	20	70%
Average Fanout of Non-Clock Nets	2.69		

Table 7.14 Design Summary of Serial Band Pass Filter

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	422	9,312	4%
Number of 4 input LUTs	1,309	9,312	14%
Number of occupied Slices	883	4,656	18%
Number of Slices containing only related logic	883	883	100%
Number of Slices containing unrelated logic	0	883	0%
Total Number of 4 input LUTs	1,344	9,312	14%
Number used as logic	1,309		
Number used as a route-thru	35		
Number of bonded IOBs	39	232	16%
Number of BUFGMUXs	1	24	4%
Number of MULT18X18SIOs	2	20	10%
Average Fanout of Non-Clock Nets	3.66		

Table 7.15 Power Summary of Parallel BPF

Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-4

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.2,06-23-09

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.002	1	---	---
Logic	0.007	3286	9312	35
Signals	0.009	5260	---	---
MULTs	0.000	14	20	70
IOs	0.026	39	232	17
Leakage	0.082			
Total	0.125			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	81.7	28.3

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.041	0.015	0.026
Vccaux	2.500	0.019	0.001	0.018
Vcco25	2.500	0.012	0.010	0.002

Supply Power (W)	Total	Dynamic	Quiescent
	0.125	0.043	0.082

Color	Source
	Estimated
	Default
	Calculated

Table 7.16 Power Summary of Serial BPF

Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-4

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.2,06-23-09

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.033	1	---	---
Logic	0.003	1329	9312	14
Signals	0.014	1710	---	---
MULTs	0.000	2	20	10
IOs	0.001	39	232	17
Leakage	0.082			
Total	0.132			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	81.5	28.5

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.069	0.042	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.002	0.000	0.002

Supply Power (W)	Total	Dynamic	Quiescent
	0.132	0.051	0.082

Color	Source
	Estimated
	Default
	Calculated

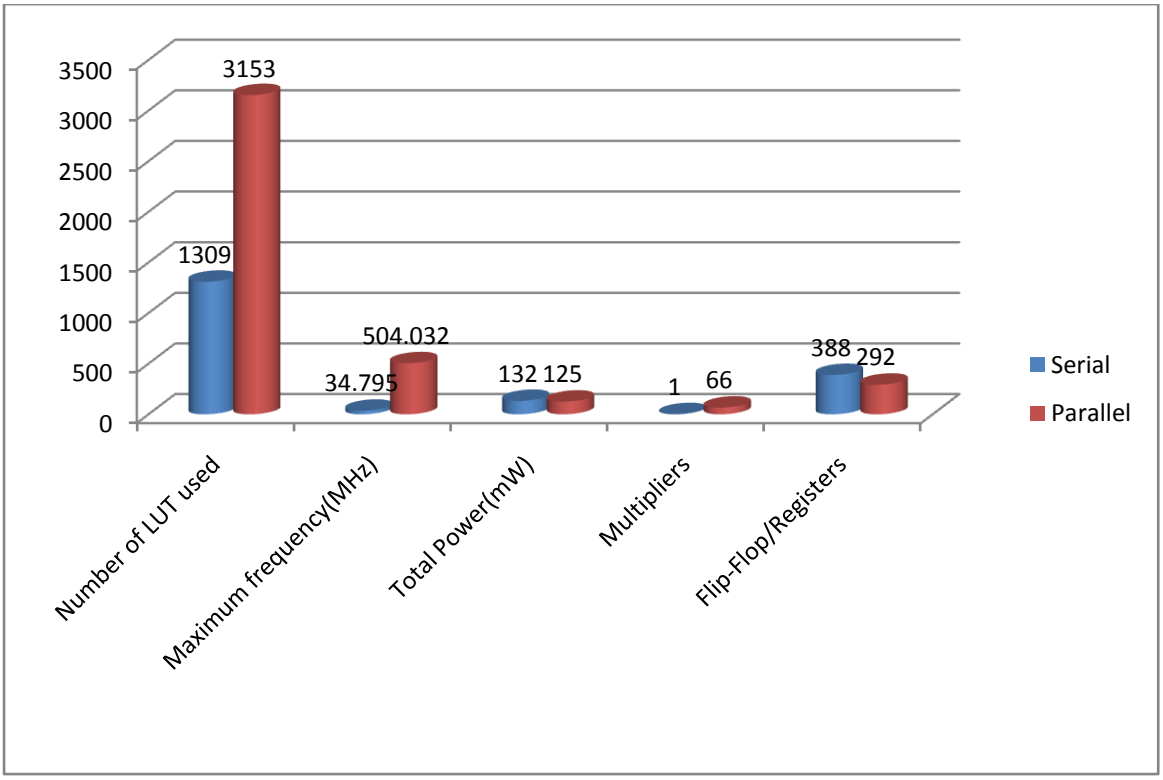


Figure 7.8 Performance chart of fully parallel and fully serial filter for BPF

7.3 HIGH PASS FILTER

Although any filter specifications can be taken but for the sake of implementation the following specifications are considered for low pass filter :-

Pass band edge frequency = 10 kHz
Transition width =0.5 kHz
Pass band ripple = 0.1 dB
Stop band attenuation = 60 dB and
Sampling frequency =48 kHz

The filter order of given specification is calculated using Hamming window technique as ;

$$\Delta f = \frac{0.5}{8} = 0.0625 \quad (7.3)$$

$$N = \frac{3.3}{\Delta f} = 52.80 \approx 53 \quad (7.4)$$

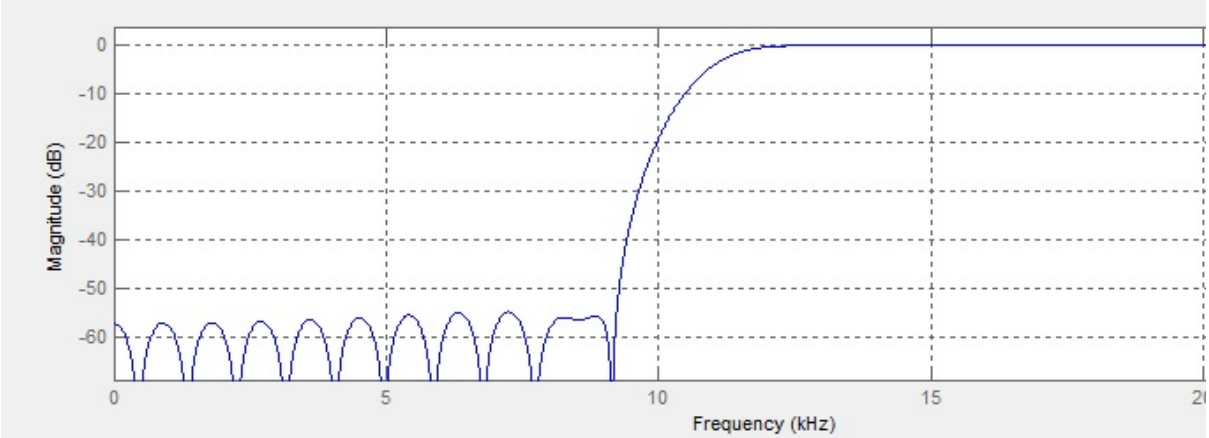


Figure 7.9 Magnitude Response of HPF in dB

The designed coefficients are given in Appendix III. Now this VHDL code is used to generate the circuit using Xilinx synthesis tool for high pass filter design and main circuit block is shown in figure 7.10.

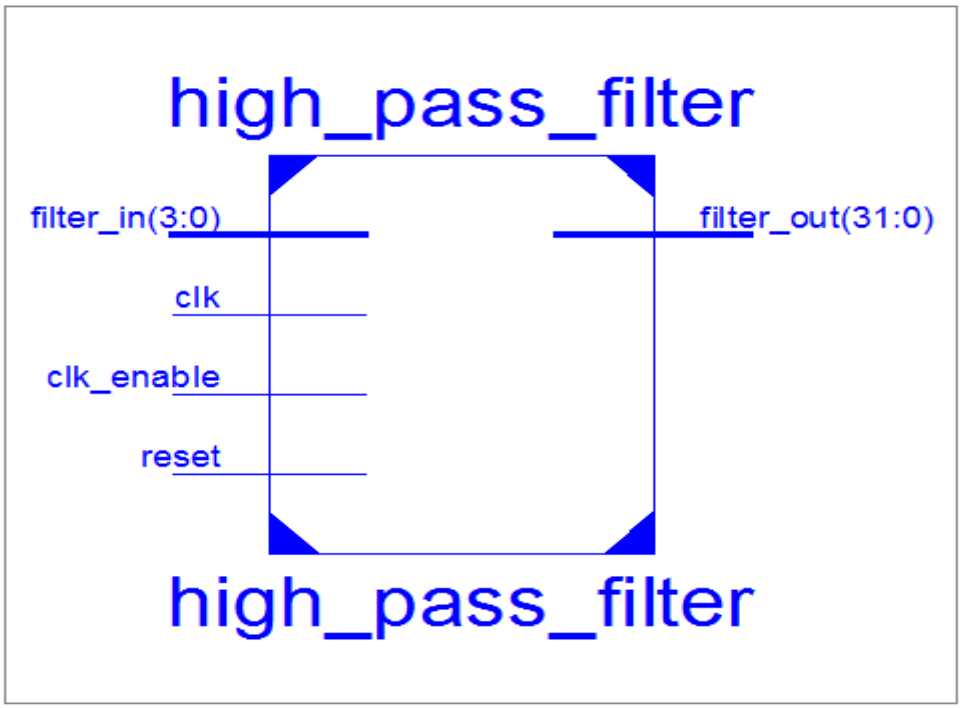


Figure 7.10 Block diagram of the High Pass Filter (HPF).

Now the generated circuit is simulated using Xilinx simulator with step inputs and the corresponding input and output waveform are shown in Fig 7.11.

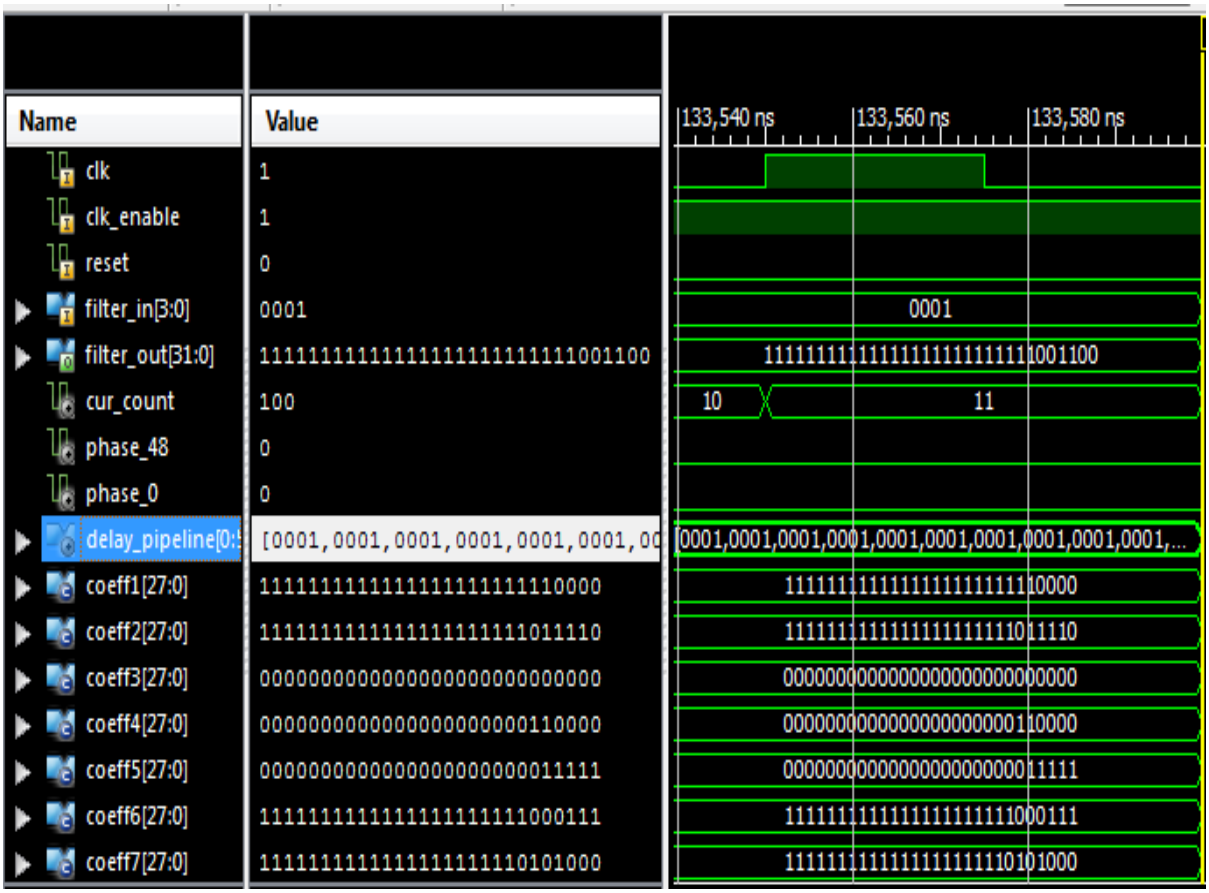


Figure 7.11 Input and Output waveform of HPF

Table 7.17 Advanced HDL Synthesis Report of parallel HPF

```
=========================================================================
*                      Advanced HDL Synthesis                           *
=========================================================================


=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Multipliers                                          : 46
 4x28-bit multiplier                                   : 46
# Adders/Subtractors                                   : 49
 32-bit adder                                          : 2
 33-bit adder                                          : 47
# Registers                                            : 212
 Flip-Flops                                            : 212
# Multiplexers                                         : 51
 32-bit 4-to-1 multiplexer                             : 51

=========================================================================
```

Table 7.18 Advanced HDL Synthesis Report of serial HPF

```
=========================================================================
*                      Advanced HDL Synthesis                           *
=========================================================================


=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# Multipliers                                          : 1
 28x4-bit multiplier                                   : 1
# Adders/Subtractors                                   : 1
 33-bit adder                                          : 1
# Counters                                             : 1
 32-bit up counter                                     : 1
# Registers                                            : 308
 Flip-Flops                                            : 308
# Multiplexers                                         : 1
 32-bit 4-to-1 multiplexer                             : 1

=========================================================================
```

Table 7.19 Timing Summary of Parallel HPF

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 1.984ns (Maximum Frequency: 504.032MHz)
   Minimum input arrival time before clock: 3.098ns
   Maximum output required time after clock: 266.771ns
   Maximum combinational path delay: No path found
```

Table 7.20 Timing Summary of Serial HPF

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 27.201ns (Maximum Frequency: 36.763MHz)
   Minimum input arrival time before clock: 5.969ns
   Maximum output required time after clock: 4.283ns
   Maximum combinational path delay: No path found
```

Table 7.21 Design Summary of Parallel High Pass Filter

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	231	9,312	2%
Number of 4 input LUTs	2,135	9,312	22%
Number of occupied Slices	1,288	4,656	27%
Number of Slices containing only related logic	1,288	1,288	100%
Number of Slices containing unrelated logic	0	1,288	0%
Total Number of 4 input LUTs	2,243	9,312	24%
Number used as logic	2,135		
Number used as a route-thru	108		
Number of bonded IOBs	39	232	16%
Number of BUFGMUXs	1	24	4%
Number of MULT18X18SIOs	16	20	80%
Average Fanout of Non-Clock Nets	2.61		

Table 7.22 Design Summary of Serial High Pass Filter

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	342	9,312	3%
Number of 4 input LUTs	951	9,312	10%
Number of occupied Slices	665	4,656	14%
Number of Slices containing only related logic	665	665	100%
Number of Slices containing unrelated logic	0	665	0%
Total Number of 4 input LUTs	983	9,312	10%
Number used as logic	951		
Number used as a route-thru	32		
Number of bonded IOBs	39	232	16%
Number of BUFGMUXs	1	24	4%
Number of MULT18X18SIOs	2	20	10%
Average Fanout of Non-Clock Nets	3.56		

Table 7.23 Power Summary of Parallel High Pass Filter

Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-4

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.2,06-23-09

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.002	1	---	---
Logic	0.005	2243	9312	24
Signals	0.007	3652	---	---
MULTs	0.000	16	20	80
IOs	0.023	39	232	17
Leakage	0.081			
Total	0.118			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	81.9	28.1

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.037	0.011	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.011	0.009	0.002

Supply Power (W)	Total	Dynamic	Quiescent
	0.118	0.036	0.081

Color	Source
	Estimated
	Default
	Calculated

Table 7.24 Power Summary of Serial High Pass Filter

Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Temp Grade	Commercial
Process	Typical
Speed Grade	-4

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Characterization	
PRODUCTION	v1.2,06-23-09

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.022	1	---	---
Logic	0.006	970	9312	10
Signals	0.021	1290	---	---
MULTs	0.000	2	20	10
IOs	0.001	39	232	17
Leakage	0.082			
Total	0.131			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	81.6	28.4

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.200	0.068	0.041	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.002	0.000	0.002

Supply Power (W)	Total	Dynamic	Quiescent
	0.131	0.050	0.082

Color	Source
	Estimated
	Default
	Calculated

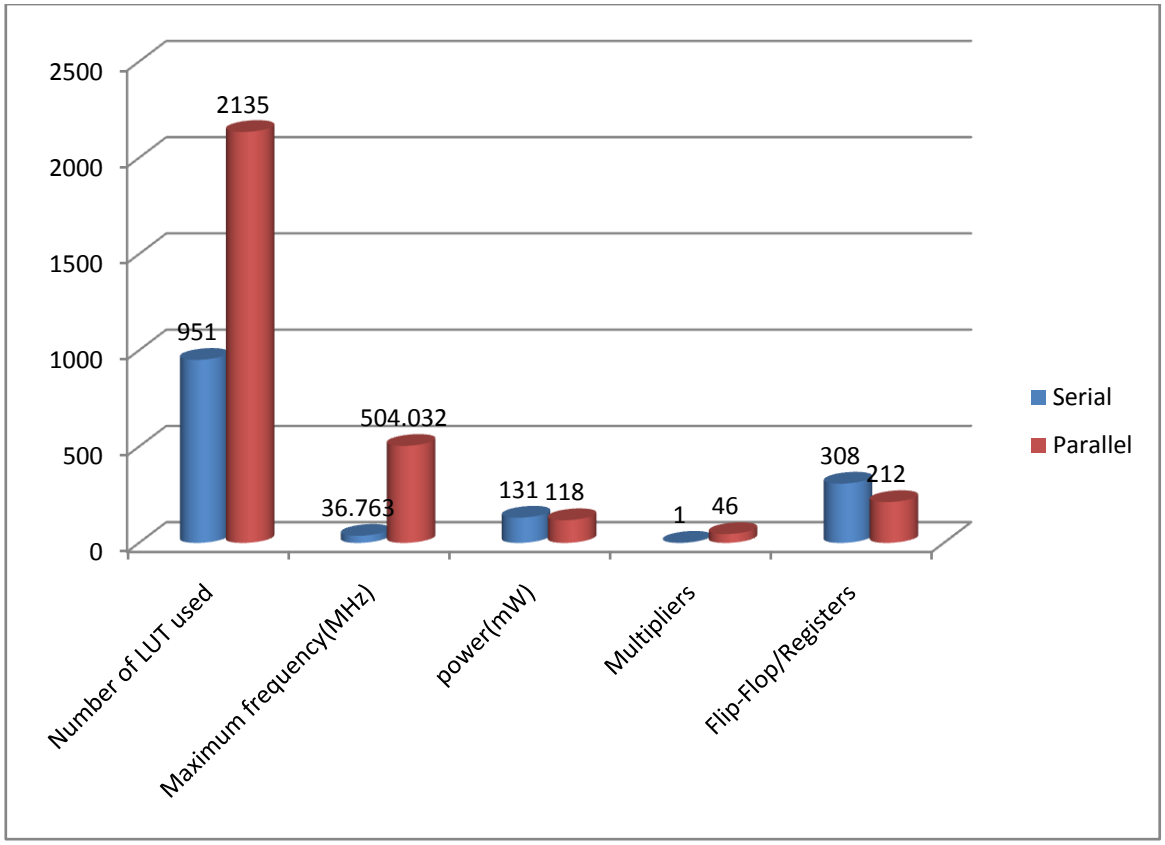


Figure 7.12 Performance chart of fully parallel and fully serial filter for HPF

7.4 DISCUSSIONS

A fully parallel architecture uses a dedicated multiplier and adder for each filter tap. All taps execute in parallel. A fully parallel architecture is optimal for speed. However, it requires more multipliers and adders than a serial architecture, and therefore consumes more chip area.

Serial architectures reuse hardware resources in time, saving chip area. A fully serial architecture conserves area by reusing multiplier and adder resources sequentially. For example, a four-tap filter design would use a single multiplier and adder, executing a multiply/accumulate operation once for each tap. The multiply/accumulate section of the design runs at four times the filter's input/output sample rate. This saves area at the cost of some speed loss and higher power consumption.

In a fully serial architecture, the system clock runs at a much higher rate than the sample rate of the filter. Thus, for a given filter design, the maximum speed achievable by a fully serial architecture will be less than that of a parallel architecture.

If we consider the HPF, in a parallel architecture it using 2135 LUTs and total power consume is 118 mW while serial architecture using 951 LUTs and total power consuming is 131mW. Maximum frequency of parallel architecture is 504.032 MHz and for serial architecture is 36.763 MHz.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE OF WORK

8.1 CONCLUSION

The FIR filters are widely used in digital signal processing and can be implemented using programmable digital processors. But in the realization of large order filters the speed, cost, and flexibility is affected because of complex computations. So, the implementation of FIR filters on FPGAs is the need of the day because FPGAs can give enhanced speed. This is due to the fact that the hardware implementation of a lot of multipliers can be done on FPGA which are limited in case of programmable digital processors.

In this thesis, a fifty three-order low-pass and high pass and seventy-three order band pass FIR filter is implemented in Spartan-III-xc3s500c-4fg320 FPGA. The Direct form structure of these filters are implemented. This approach gives a better performance than the common filter structures in terms of speed of operation, cost, and power consumption. In the Direct-form structure, N Shift Registers, N Adder and N+1 multipliers are used to realize the N order low pass, high pass and band pass filter. The designed filters can work for real time processing of any digital signal.

8.2 FUTURE SCOPE OF WORK

The future scope of this work includes the following:

- The A/D and D/A converter can be interfaced within the FPGA.
- The optimization of the design can be done in terms of area occupied on chip.

REFERENCES

[1] Tan,Li "Digital Signal Processing : Fundamentals and Applications" Edition 2007.

[2] Mitra,S. K., "Digital Signal Processing" 3rd Edition, Tata Mc. Graw Hill Publications.

[3] Ifeachor,E.C., Jervis,B.W., "Digital Signal Processing",2nd Edition, Low Price Edition 2007.

[4] Ruan,A.W., Liao,Y.B., Li,P. , Li,J.X., "An ALU-Based Universal Architecture for FIR Filters" in proc. *International Conference on Communications, Circuits and Systems, IEEE,* pp. 1070 – 1073, 2009.

[5] Nekoei,F., Kavian,Y.S., Strobel,O., "Some Schemes Of Realization Digital FIR Filter On FPGA For Communication Application" in proc. *20th International Crimean Conference on Microwave and Telecommunication Technology,* pp. 616 – 619, 2010.

[6] Li,J., Zhao,M., Wang,X.,"Design and Simulation of 60-order Filter Based on FPGA" in proc. *International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), IEEE,* pp. 113 - 115, Oct 2011.

[7] Proakis,J.G., Manolakis,D.G., "Digital Signal Processing" 3rd Edition, PHI publication 2004.

[8] Antoniou, A., " Digital Filter", 3rd Edition, Tata Mc. Graw Hill publications, 2001.

[9] Wenjing,H., Guoyun,Z., Waiyun,L.,"Self Programmable Multipurpose Digital Filter Design Based On FPGA" in proc. *International Conference on Internet Technology and Applications (iTAP), IEEE,* pp. 1-5, 2011.

[10] Parhi K K., "A Systematic Approach for Design of Digit-serial Signal Processing Architectures", Circuits and Systems, 1991.

[11] "An Introduction to Digital Filters", by INTERSIL, Application Note, January 1999.

[12] Mirzaei,S., Hosangadi,A., Kastner,R., "FPGA Implementation of High Speed FIR Filters Using Add and Shift Method",in proc. *International Conference on Computer Design (ICCD), IEEE*, pp 308-313, 2006.

[13] Takahashi Y. and Yokoyama M., "New cost-effective VLSI implementation of multiplierless FIR filter using common subexpression elimination", in Proc. *IEEE Int. Symp. on Circuits and Systems*, pp. 845-848, 2005.

[14] Rocha Ed., "Implementation trade-offs of digital FIR filters," Military Embedded System, open system publishing, 2007.

[15] Choi, Seak C. and Lee H., "A Partial Self-Reconfigurable Adaptive FIR Filter System", *IEEE Workshop on signal processing systems*, pp. 204-209, 2007.

[16] Takalashi,Y., Sekine,T. and Yokoyama,M., "A 70MHz Multiplierless FIR Hilbert Transformer in 0.35 μm standard CMOS Library," *IEICE Trans.*, pp. 1376-1383, 2007.

[17] Kumar,B. and Kumar,A., "Design of Efficient FIR Filters for the Amplitude Response", *IEEE Trans. on signal processing*, vol. 16, pp. 122-125, 1999 .

[18] McClellan,J. H., Parks, T.W., and Rabiner,L.R., "A computer program for designing optimum FIR linear phase digital filters", *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp.506 -526, 1973 .

[19] Jones, D. L., "FIR Filter Structures", Version 1.2: Oct 10, 2004.

[20] "FPGA Architect - XilinxXC4000/Spartan" by ELANIX Inc.

[21] Chen,W. K., "Logic Design",CRC Press, 2000.

[22] http://www.Xilinx.com/bvdocs/whitepapers/wp116.pdf.

[23] http://www.dsptutor.freeuk.com/dfilt1.html.

[24] Beyrouthy,T., Fesquet,L., "An event-driven FIR filter: Design and implementation", in proc. *22nd IEEE International Symposium on Rapid System Prototyping (RSP)*, pp. 59 - 65, 2011.

[25] Wakerly, J. F., "Digital Design & Practice", Pearson Education Asia 3rd edition.

[26] Chapman, S. J., "Matlab Programming for Engineers", 3rd Edition, Thomson learning 2005.

[27] Jieshan,L., Shizhen,H., “An Design of the 16-order FIR Digital Filter Based on FPGA” in proc. *International Conference on Information Science and Engineering (ICISE), IEEE,* pp. 489-492, 2009.

[28] Saab,S., Lu,W.-S., Antoniou,A., “Design and implementation of low power IIR digital filter system” in proc. *International Symposium on Circuits and Systems, IEEE*, vol. 3, pp. 391 - 394, 1999.

[29] Yunlong,W., Shihu,W., Rendong,J., “An Extreme Simple Method for Digital FIR Filter Design” in proc. *Third International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), IEEE*, vol. 1, pp. 410-413, 2011.

[30] Nian-qiang,L., Si-Yu,H., Shi-Yao,C., “Application of Distributed FIR filter based on FPGA in the analyzing of ECG signal” in proc. *International Conference on Intelligent System Design and Engineering Application, IEEE,* pp. 335-338, 2010.

[31] Soni,V., Shukla,P., Kumar,M., “Application of Exponential Window to Design a Digital Nonrecursive FIR Filter” in proc. *International Conference on Advanced Communication Technology (ICACT), IEEE*, pp. 1015-1019, 2011.

APPENDIX-I

FIR coefficients for lowpass filter:

h(0)	-45	h(52)
h(1)	-64	h(51)
h(2)	0	h(50)
h(3)	92	h(49)
h(4)	89	h(48)
h(5)	-61	h(47)
h(6)	-204	h(46)
h(7)	-99	h(45)
h(8)	229	h(44)
h(9)	370	h(43)
h(10)	0	h(42)
h(11)	-553	h(41)
h(12)	-511	h(40)
h(13)	332	h(39)
h(14)	1037	h(38)
h(15)	473	h(37)
h(16)	-1039	h(36)
h(17)	-1617	h(35)
h(18)	0	h(34)
h(19)	2330	h(33)
h(20)	2178	h(32)
h(21)	-1468	h(31)
h(22)	-4948	h(30)
h(23)	-2586	h(29)
h(24)	7289	h(28)
h(25)	19239	h(27)
h(26)	24616	h(26)

APPENDIX-II

FIR coefficients for bandpass filter:

h(0)	-650	h(72)
h(1)	-2283	h(71)
h(2)	-693	h(70)
h(3)	1595	h(69)
h(4)	1199	h(68)
h(5)	-249	h(67)
h(6)	0	h(66)
h(7)	268	h(65)
h(8)	-1386	h(64)
h(9)	-1983	h(63)
h(10)	927	h(62)
h(11)	3289	h(61)
h(12)	1009	h(60)
h(13)	-2351	h(59)
h(14)	-1789	h(58)
h(15)	377	h(57)
h(16)	0	h(56)
h(17)	-418	h(55)
h(18)	2203	h(54)
h(19)	3216	h(53)
h(20)	-1537	h(52)
h(21)	-5586	h(51)
h(22)	-1761	h(50)
h(23)	4230	h(49)
h(24)	3333	h(48)
h(25)	-731	h(47)
h(26)	0	h(46)
h(27)	895	h(45)
h(28)	-5019	h(44)
h(29)	-7900	h(43)
h(30)	4141	h(42)
h(31)	16917	h(41)
h(32)	6217	h(40)
h(33)	-18468	h(39)
h(34)	-20131	h(38)
h(35)	8086	h(37)
h(36)	26604	h(36)

APPENDIX-III

FIR coefficients for highpass filter:

h(0)	-16	h(52)
h(1)	-34	h(51)
h(2)	0	h(50)
h(3)	48	h(49)
h(4)	31	h(48)
h(5)	-57	h(47)
h(6)	-88	h(46)
h(7)	33	h(45)
h(8)	161	h(44)
h(9)	52	h(43)
h(10)	-213	h(42)
h(11)	-211	h(41)
h(12)	180	h(40)
h(13)	418	h(39)
h(14)	0	h(38)
h(15)	-595	h(37)
h(16)	-367	h(36)
h(17)	618	h(35)
h(18)	905	h(34)
h(19)	-326	h(33)
h(20)	-1537	h(32)
h(21)	-496	h(31)
h(22)	-2139	h(30)
h(23)	2385	h(29)
h(24)	-2572	h(28)
h(25)	-10040	h(27)
h(26)	19112	h(26)