## CHAPTER 1 INTRODUCTION

A rapid pace of development in communication technology makes the communication devices out of date soon after their engineering.To go with such pace communication system require insertion of latest technology.The upgraded modern device should be able to communicate with legacy of devices as well.The Software Defined Radio(SDR) helps one to add new functionality without much hardware changes,even after technological update. The ideal Software Radio, as defined by Wireless Innovation Forum [1] refers to the complete software control of the entire system. This means that analogue conversion should take place only at antenna, ensuring the support for a wide frequency range. These kinds of software radios will be obviously future proof as the whole radio system will be dependent on programmability, leading to the same hardware behaving differently at different instances.

SDR is fast becoming a crucial element of wireless technology.Many of the traditional methods of implementing transmitters and receivers are predicted to be replaced by SDR technology.The advantages offered by it including adaptability, reconfigurability, and multifunctionality encompassing modes of operation, radio frequency bands, air interfaces, and waveforms [1].A lot of research is going on for improving the architecture and the computational efficiency of SDR systems.

Computationally the most important part of an SDR receiver is the channelizer as it operates at high sampling rate [2].Channelization in SDR receivers involves the extraction of multiple narrowband channels from a signal using several bandpass filters called channel filters [3]. Low power and high-speed FIR filters are required in the channelizer [4]. The key functional units in a digital filter are delay, adder, and multiplier – of which the hardware complexity is dominated by multipliers.

Digital filters remove noise and interference from the original signal and are used for modification of various attributes of the signal. The most computationally intensive part of the wideband receiver of a software defined radio (SDR) is the intermediate frequency (IF) processing block. Digital filtering is the main task in IF processing. Digital filters are more accurate, more versatile and highly stable, thereby a preferred technique over its analogue counterpart [3].The principle objective of this exploration is to present a methodology for an upgraded framework of a FIR digital filter from software level to the hardware level that includes the selection of design method, structure and cost effective hardware utilization.

A digital filter is a system that performs mathematical operations on a sampled, digitized signal to reduce or enhance certain features of the processed signal. Digital

filter scheme consists of a prefilter or anti-aliasing filter to perform filtering of an input signal using a Band pass filter.We use an arithmetic scheme, known as floating-point representation to encode the filter coefficients.This thesis presents a method to implement FIR filters for SDR receivers using using various adders and multipliers.

antenna

RF

Tuner

ADC

Mixer

Filter

DSP

Oscillator

Figure:1 SDR Receiver

An interface is needed between the analog signal and the digital filter, this interface is known as analog-to-digital converter (ADC). After the process of sampling and converting, a digital signal is ready for further processing using an appropriate digital signal processor. The output signal that is digitized is usually changed back into analog form using digital-to-analog converter (DAC).

* 1. **MOTIVATION**

Due to the increased demand of implementation of sophisticated DSP algorithms,lowcost design ,I.e low area and low power cost,are needed to make hand held devices small with good performance.Generally,dedicated multipliers are expensive in terms of chip area and It has also been observed that efficient hardware implementation of a FIR filter requires a relatively large number of multipliers. In the realization of Fir filters it is evident that almost every structure utilizes Multipliers. which directly impacts the silicon area and power consumption.This report focuses on the development of a novel high-performance FIR Filter that requires an efficient multiplier unit to multiply coefficients with input samples, and hence,significantly reduces the computational complexity. In order to achieve this goal, our work is oriented towards the different multipliers unit which can be computationally efficient for the FIR filters.

###### PROBLEM STATEMENT

Generally Multipliers are slow and they also consume large area ,similarly Adders are a source of delay and consume more power.

The arithmetic cost is dictated by types of architecture of an FIR Filter such as

* Direct form
* Transpose form
* Lattice structure

This project aims at the following study and implementation

* Identifying the architecture which takes less arithmetic cost.
* Slices, Flip Flops

The main focus is on implementing an efficient FIR Filter with higher performance,consumes less area.

* 1. **OBJECTIVE**

The thesis embodies following objectives:

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

The Aim of this research was to design, implement, FIR Filter for various applications. In order to fulfill these goals a specific research approach was followed.The initial effort was taken to understand the mathematical theory and functionality of convolution theorem and the FIR, which is an efficient theoremfor computing the output samples. The filter design for different order,specification will be examined and implement using Xilinx tools. Multiplier module will investigated for more efficiency in the filter structure.Finally, new filter datapath unit techniques were investigated for portability requirements associated with the FPGA.Once a solid foundation was produced, the theoretical design was constructed using Verilog HDL in Xilinx and was targeted on FPGA device XC3S500E. With the design thoroughly tested and the behavioral simulations, complete synthesis and timing-based simulation was performed to verify the design would run as expected with the chosen parameters in the targeted FPGA

**INTRODUCTION**

## CHAPTER 2 LITERATURE SURVEY

In this chapter, literature survey on previous research papers related to the FIR filters, different structural architecture and hardware implementation of filters are discussed.

* 1. **LITERATURE SURVEY**

In Paper [8]. *Anubhuti Mittal , Ashutosh Nandi, Disha Yadav* presented this paper “*Comparative study of 16-order FIR filter design using different multiplication Techniques”* to study a comparation between various adders and multipliers such as to optimise filter area, delay and power.different multiplication techniques such as Vedic multiplier, add and shift method and Wallace tree(WT) multiplier are used for the multiplication of filter coefficient with filter input. Various adders such as ripple carry adder,Kogge Stone adder, Brent Kung adder, Ladner Fischer adder and Han Carlson adder are analysed for optimum performance.

To reduce the complexity of filter, coefficients were represented in canonical signed digit representation as it is more efficient than traditional binary representation. They designed finite impulse-response (FIR) filter in MATLAB using equiripple method and the same filter was synthesised on Xilinx Spartan 3E XC3S500E target field-programmable gate array device using Very High Speed Integrated Circuit Hardware Description Language (VHDL) subsequently the total on-chip power was calculated in Vivado2014.4. The comparison of simulation results of all the filters showed that FIR filter with WT multiplier is the best optimised filter.The FIR filter designed is of order 16,with density factor 20 using equirriple method.They compared various multipliers and adders combination to implement filter in xilinx ,which was designed in Matlab and concluded that FIR filter designed with WT multiplier has performance advantages among all the proposed filters.

The combination proposed and compared were FIR with vedic multiplier (VM) and ripple carry adder (RCA),FIR with VM and Brett Kung (BK),FIR with RCA and Barrel Shifter (BS),FIR with Kogge Stone (KS) and BS ,FIR with BK and BS, FIR with Han Carlson(HC) and BS, FIR with Ladner Fischer( LF) and BS, FIR with WT and RCA.

In paper [9] *Aneela Pathan,Tayab D Memon,Sharmeen Keerio,Imtiaz Hussain Kalwar* have presented comparative analysis of Booth and Wallace Tree multiplier architectures using Altera small commercial FPGA device ,in their paper “*FPGA Based performance analysis of multiplier policies for FIR filter”.*Comparison is done with respect[] to resources consumed and maximum frequency achieved for different multiplier bit width.In this work, two

algorithms are implemented on small commercial FPGA devices provided by Altera. They have chosen two architectures i.e., cyclone and Stratix, area-performance was evaluated for these two algorithms.

For varying bit widths i.e 6,8,10 sizes of multiplicand and mutipliers area performance were compared .They observed that maximum Performance (FMAX) of Cyclone and Stratix is consistent for Wallace tree multiplier, but when it is compared with Booth multiplier FMAX is decreasing by increasing the number of bits (i.e., multiplier bit-width) in both families.They concluded that performance of Wallace tree is much better than booth.

In paper [10] *Evangelos Kyritsis ,Kiamal Pekmestzi* in their paper “*Hardware Efficient Fast FIR Filter Based on Karatsuba Algorithm”* proposed an efficient implementation of a programmable Finite Impulse Response (FIR) filter based on the use of the Karatsuba Multiplication Algorithm (KMA).Here an FIR filter circuit, a parallel, Modified Booth (MB) pre-encoded, Carry-Save (CS) Wallace tree multiplier have been used as building block.They have compared the proposed architecture with that of conventional architecture for speed, area, power.

The Karatsuba algorithm is applied to the FIR filter equation, a parallel architecture is obtained giving speed and area savings.In this paper, they proposed an architecture based on a modified formulation of KMA. In the proposed implementation carry-save (CS) arithmetic is used in order to decrease the critical path and to speed-up the calculations. For comparision between conventional and karatsuba FIR Filters ,both of them were designed for taps =16,32 and bits 2N.They used modelsimfor functional simulation, then the designs were implemented on Faraday 90nm (FSD0A\_A\_GENERIC\_CORE\_TT1V25C) technology using Synopsys tools.The synthesis has been made with Design Compiler.The mean power consumption has been obtained using PrimePower. They concluded that the proposed Karatsuba design showed an improved performance, a smaller circuit area and lower power consumption, compared with the Conventional transposed FIR filter.

In Paper [11] *Ila Sharma,Anil Kumar,Girish Kumar Singh, Heung-No Lee* proposed a hybrid method-based design of multiplierless two-channel filter bank has been proposed with a given stopband attenuation (As) and roll-off factor.Windowing technique has been used for filter design.The implemented filter is synthesised using target field programmable gate arrays XC3S500E-4-FG320 on Xilinx Spartan 3E starter board. The performances of designed prototype filter is compared with the earlier published works in terms of reconstruction error, amplitude distortion,slices, flip-flops, four-input lookup tables and adders.The synthesis results demonstrate that the significant reduction in hardware is achieved in term of adder gain. For

filter order, N = 32, and word length 12, the adder gain achieved in canonical sign digit (CSD).and factorised canonical sign digit (FCSD)is 41.77 and 43.07%, respectively, while for N = 30, it is 35.44% in CSD and 36.70% in FCSD.

In paper[12] *Arunadevi Jawahar,P.Pushpa Latha* in their paper “*Implementation of high-order FIR digital filtering for software defined radio receivers”*proposed an FIR filter for SDR receiver application.They utilized a transpose form structure.Wallace tree multiplier and kogge stone adder were used to implement the filter.A comparision was made between existing and proposed work.They achieved a minimum delay.

* 1. **SUMMARY**

In this chapter, literature survey of different IEEE papers has been discussed briefly. With this literature survey it is concluded that a FIR filters were realized by various structures, various authors utilized various adders and multipliers to design efficient filters. From the literature survey, I want to implement filter using various multipliers and adders.I’ll use different structures like,transpose,directform for comaparing among themselves such that it is application oriented.

**INTRODUCTION**

## CHAPTER 3 FILTERS

In general, filtering is the processing of a time-domain signal resulting in some change in that signal's original spectral content. The change is usually the reduction, or filtering out, of some unwanted input spectral components; that is, filters allow certain frequencies to pass while attenuating other frequencies. Figure below shows both analog and digital versions of a filtering process. Where an analog filter operates on a continuous signal, a digital filter processes a sequence of discrete sample values.

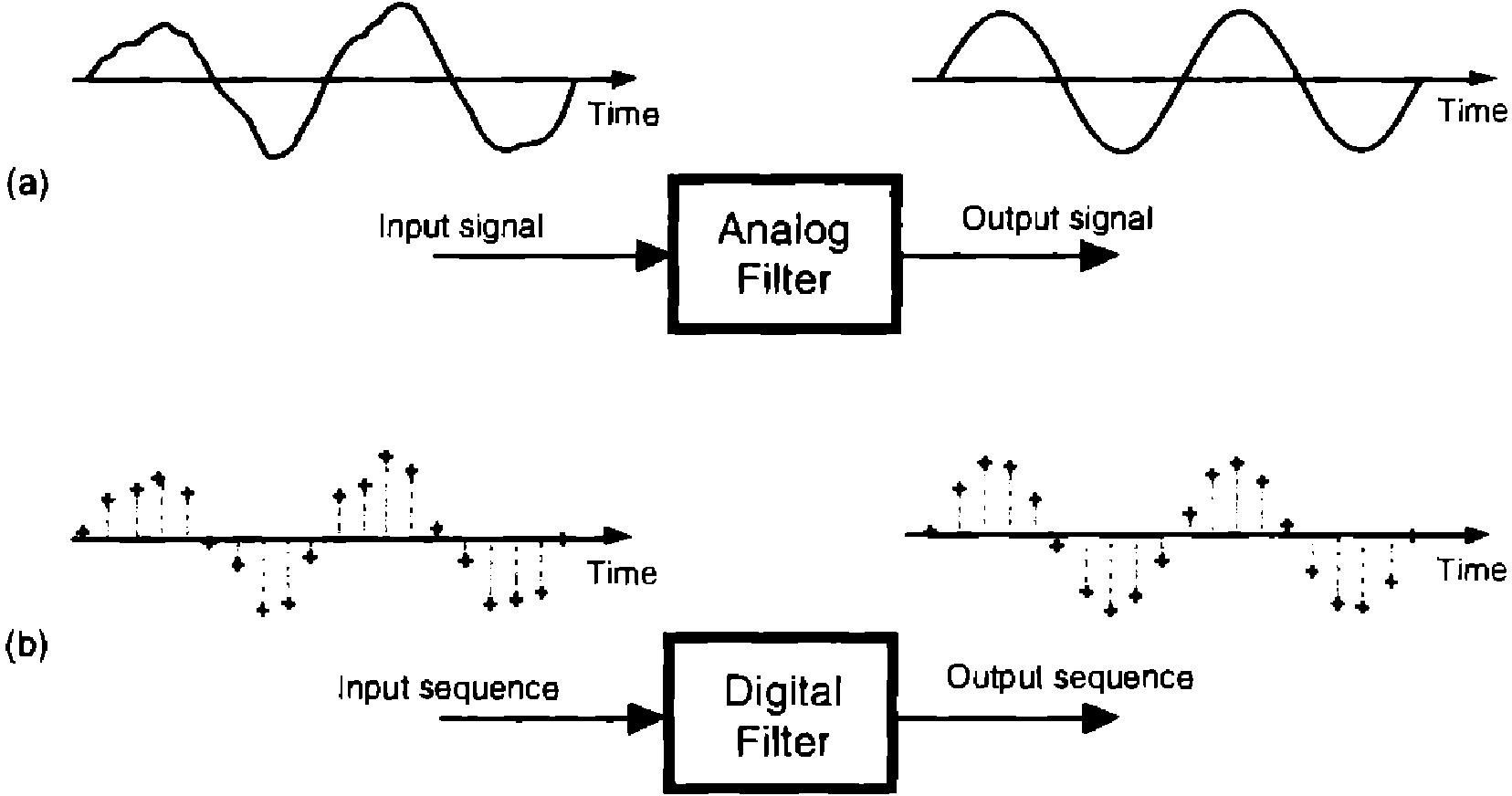


Figure 3.1 A block diagram of a basic filter.

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. **TYPES OF FILTERS**

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

1. Analog Filters
2. Digital Filters
   * 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. **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].

#### Advantages Of Using Digital Filters

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

ADC

Processor

DAC

Unfilterd analog signal

Sampled digitised signal

Unfilterd analog signal

filterd analog signal

Figure 3.2 Block diagram of digital filter

* Digital filters are easily designed, tested and implemented on a general purpose computer or workstation.
* 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.
* 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.
* 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.
* 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. **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].

the filters exhibit the following behaviour:

* The low pass filter passes all frequencies below *fc*.

stopband

passband

stopband



stopband

stopband

passband

passband passband

Amplitude

passband

Low pass High pass Band pass Bandstop

Amplitude

Amplitude

Figure 3.4 Pass band and Stop band

* The high pass filter passes all frequencies above *fc*.
* The band pass filter passes all frequencies between *fc1* and *fc2*.
* The band stop filter attenuates all frequencies between *fc1* and *fc2*.

The frequency points *fc*, *fc1*, and *fc2* 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.

* 1. **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. **TRANSITION BAND**

Figure below shows the pass band, the stop band, and the transition band for each type of practical filter. In each plot in Figure below 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].

* + 1. **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

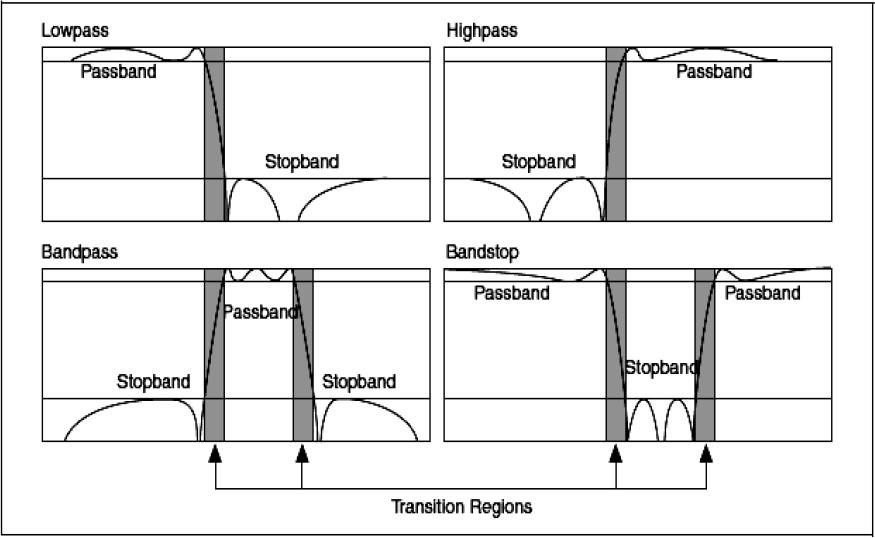


Figure 3.5 Response of Non-ideal filter

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

* + 1. **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. **DIGITAL FILTER COEFFICENTS**

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

Zero order: yn = a0 x n

First order: yn = a0 x n + a1 xn-1

Second order: yn = a0 x n + a1 xn-1 + a2 xn-2 Thirdorder:yn=a0xn+a1xn-1+a2xn-2+a3xn-3

Fourth order: yn=a0xn+ a1xn-1+a2xn-2+a3xn-3+a4xn-4

Similar expressions can be developed for filters of any order.The constants *a0 ,a1,a2,a3,a4* 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. **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. **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.

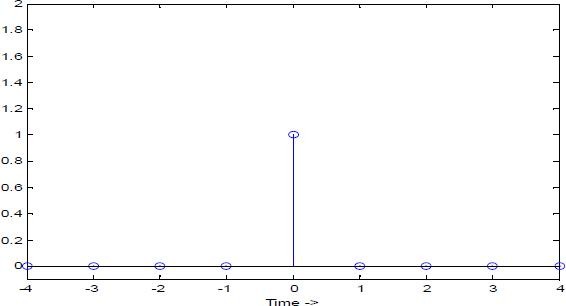


Figure 3.6 Impulse Response

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. **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* ) 

#####  0 ,

 0 ,



n≥n1, n≥n2 where n1 and n2 lies between -  and  .

where *h*(*n*) denotes the impulse response of the digital filter, *n* is the discrete time index, and *n1* and *n2* 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

*N* 1

*y* (*k* ) 

 *b* (*n* ) *x* (*n*  *k* )

*n*  0

(1)

where *y*(*n*) is the filter output at discrete time instance *n, bk* is the *kth* 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* = *n1* 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. **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 (*xn*, *xn-1*, *xn-2*,...) but also terms involving the past output values *yn*, *yn-1*,......The following general difference equation characterizes IIR filters[7].

*yi*  1 

*Nb*1 *bjXi*  *j* 

*Na*1 *akyi*  *k* 

*a*0  *j*  0



*k*  1



 (2)

Where *b* is the set of forward coefficients, *Nb* is the number of forward coefficients, *ak* is the set of reverse coefficients, and *Na* is the number of reverse coefficients. Where *xi* is the current input, *xi-j* is the past inputs, and *yi-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. **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. **SUMMARY**

From this chapter we learnt the basics of filters .There are two types of filters,how digital filters are preferred over analog filters.Ideal filters are impractical in real world.we can even conclude the more narrower the transition band the more efficient the filter would be.we also learnt the specifications of filters .we compared FIR with IIR filters to specify that where ever stability is the concern FIR filter should be the first selection for designer.

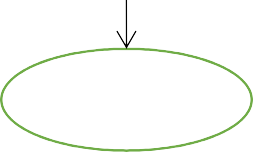
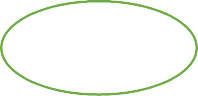
**INTRODUCTION**

## CHAPTER 4

**FIR FILTER DESIGN**

Digital Filter design Process can be summarized into the following steps. The design of a digital filter involves following five steps[3].

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



**start**

Performance Specification

Caluculation of Filter

Realization Structuring

Finite worldlength Effects analysis

H/W or S/w Implementation

stop

Figure 4.1 Summary of design stage for digital filter .

* 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.

* Implementation: This involves producing the software code and/or hardware and performing the actual filtering.
  1. **FIR FILTER SPECIFICATIONS**

The specifications includes

1. Signal characteristics.
2. The characteristics of the filter.
3. The manner of implementation.
4. 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]

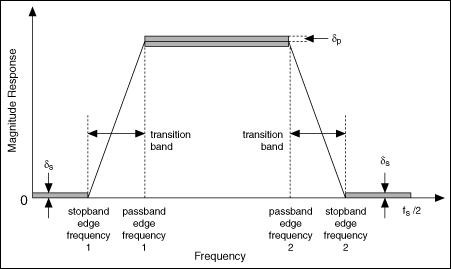
.

Figure 4.2 Magnitude frequency response specifications for a band pass filter.

In the pass band, the magnitude response has a peak deviation of *δp* and in the stop band, it as a maximum deviation of *δs*. 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:

*δp* peak pass band deviation(or ripples).

*δs* stop band deviation.

*fs1* stop band edge frequency.

*fs2* stop band edge frequency.

*fp1* pass band edge frequency. *fp2* pass band edge frequency. *fs* sampling frequency

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

As (stop band attenuation) = -20log10 δs Ap (pass band ripple)=20 log10 (1+δp)

* 1. **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].

* + 1. **WINDOW METHOD**

In this method, use is made of the fact that the frequency response of a filter, *HD*(*ω*) the corresponding impulse, *hD* are related by the Fourier transform[7] :

1 ** *jn*

*hD*(*n*)   *HD*(**) *e d*

2**

**

(3)

Now start with the ideal low pass response shown in figure, where *ωc* is cut off frequency and the frequency scale is normalised: T=1. By letting the response go from *-ωc* to *ωc* we simplify the integration operation. Thus the impulse response is given by:

*h D* ( *n* ) 

1  1 \**e jn d*

2** **

**

(4)

*c*

 1  *e jnd*

 2 *fc* sin(*nc*),

*nc*

2***c*

where n≠0,-  ≤n≤  (5)

=2f , n=0

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.

* + 1. **FREQUEN****CY 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

HD(ω)

-ωc2

-ωc1

ωc1

ωc2

ω(normalized)

Figure 4.3 Ideal frequency response of a Bandpass filter Table 4.1 Summary of ideal impulse responses

Table 4.2 Summary of important features of common window function

|  |  |  |
| --- | --- | --- |
| **Filter Type** | **hD(n), n≠ 0** | **hD(0)** |
| Low Pass | 2 *fc* sin(*nc*) *nc* | 2fc |
| High Pass | * 2 *fc* sin(*nc*)   *nc* | 1-2fc |
| Band pass | 2 *f* 2 sin(*n*2)  2 *f* 1 sin(*n*1)  *n*2 *n*1 | 2(f2-f1) |
| Band stop | 2 *f* 1 sin(*n*1)  2 *f* 2 sin(*n*2)  *n*1 *n*2 | 1-2(f2-f1) |

|  |  |  |
| --- | --- | --- |
| **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 |

response. A unique attraction of the frequency sampling method is that it also allows recursive implementation of FIR filters[7].

### 1,

1. Rectangular WR(n)= 

### 0,

0  *n*  *N*  1 ,

*otherwise*

 2 *n* ,

0  *n*  (*N* 1) / 2

  1

*N*



2 *n*



*N* 1

1. Bartlett wB(n)=  2 



*N*  1 ,

 *n*  *N* 1

##### 2

 0 ,



*otherwise*

  2*n* 

1. Hanning W

(n)= 0.5  0.5 cos *N* 

0  *n*  (*N*  1) / 2

ham



0



 

 2*n* 

*otherwise*

1. Hamming W

(n)= 0.54  0.46 cos *N* 

0  *n*  *N*  1

hamm



0

  *otherwise*

  2*n*   4*n* 

**(e)** Blackman W(n)= 0.42  0.5 cos *N*  1  0.08 cos *N*  1

0  *n*  *N*  1



0

  

 *otherwise*

* + - 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 *KfS/N*, *k* = 0,1, ., N-1. The filter *h*(*n*) coefficients can be obtained as inverse DFT of frequency samples[7].

*h* ( *n* ) 

1 *N* 1

*N k*  0



*H* ( *k* ]*e*

*j* ( 2** ) *nk N*

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* ) 

*N* 1

1 [

2



*N k* 1

2 | *H* ( *k* ) | cos[ 2*k* ( *n*  **) / *N* ]  *H* ( 0 )]

where *α* = (N-1)/2, and *H*(*k*) are the samples of the frequency response of the filter taken at intervals of *kFs/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 *fk* where *fk* defined by fk =(k+1)/2Fs/N , k=0,1,2, N-1

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) Fs/N

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

* + - 1. **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]:

1  *Z* *N N*1 *H* (*K* )

*k* 0

*H* (*Z* ) 

*N* 1  *Z* 1*e j* 2*k* / *N*

 *H* 1(*Z* )*H* 2(*Z* )

Thus in recursive form, *H*(*z*) can be viewed as a cascade of two filters: in a comb filter, *H1*(*z*), which has *N* zeros uniformly distributed around the unit circle, and a sum of *N* single all-pole filters, *H2*(*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 *H2*(*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

1  *r N Z* *N N*1 *H* (*k* )

*k* 0

*H* (*Z* ) 

*N* 1  *rZ* 1*e j* 2*k* / *N*

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* (*k* ){2 cos 2*k*  2*r* cos 2*k* (1 **) *Z* 1} 

*H* (*Z* ) 

1  *r N Z* *N* 

 

 *N* 





 *N* 

 *H* (0) 

*N*   2*k*  1



2 2

1  *Z* 1 

1  2*r* cos *Z*

  *N* 

* *r Z* 

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

* + 1. **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(ω) = W(ω) [HD(ω) - H(ω)]

Where *HD*(*ω*) is the ideal response and *W*(*ω*) 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.

* 1. **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.



Specify Filter and Determine program input

Initial guess of r+1 extrema

Determine |E(W)| and it’s largest r+1 Extrema

YES

Extrema Changed

NO

Obtain the impulse response coefficients

Figure 4.4 Simplified flowchart of the optimal method

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.

#### SUMMARY

Here from this chapter we learnt filter design procedure,the various stepps to be followed in desiging.we also saw how to to caluculate filter coefficients.we have seen the three most popular methods of calculating filter coefficients.we have even compared these methods to conclude optimal method is easy to compute filter coefficients.and benefits of other methods in their respective fields.

**INTRODUCTION**

## CHAPTER 5

**FIR FILTER STRUCTURES**

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.

* 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* ) 

 *f* (*nT* )*z* *n*

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* *aix*(*nT*  *iT* )

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

*Y*(*z*)  *X* (*z*)*N*

*i*0

*i*0

*aiz**i*

*Y* (*z*)  *X* (*z*)\* *H*(*z*)

where

*H* (*z*)  *N*

*aiz* *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 below. it is done using basic building blocks elements.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **BASIC ELEMENTS** | **BLOCK REPRESENTATION** | | **SIGNAL FLOW** | | |
| ADDER | x1(nT)  x2(nT)  xn(nT) | **Y(nT)**  *N*  *y*(*nT* )   *x* 2(*nT* )  *k* 1 | x1(nT)  x2(nT) |  | **Y(nT)** |
| TIME-DELAY | x(nT) | x(nT-1)  Z-1 | x(nT) | Z-1 | x(nT-1) |
| MULTIPLIER | x(nT) | m m\*x(nT) | x(nT) | m | mx(nT) |

Figure 5.1 Block representation & Signal flow of basic elements.

* 1. **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 5.1.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.

* + 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* )  *Y* (*Z* )

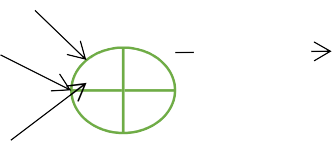
*X* (*Z* )

*N*

*i*0

 

*aiZ* *i*

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 below



*x* ( *n* )



*x* ( *n*  1 )

*x* ( *n*  2 )

*x* ( *n* 

*L*  1 )

*z*  1

*b* 0 *b*1

*z*  1

*b* 2

*b L*  1

*z*  1



*y* ( *n* )

  

Figure 5.2 Direct-Form of FIR Filter

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

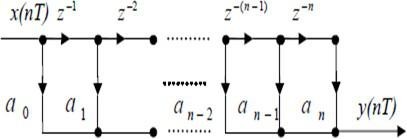


Figure 5.3 Signal flow diagram of Direct-Form

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*).

* + 1. **TRANSPOSE-FORM STRUCTURE**

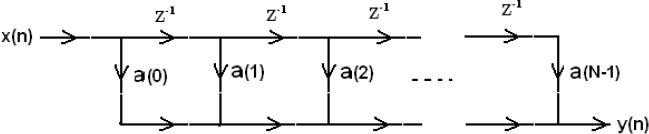
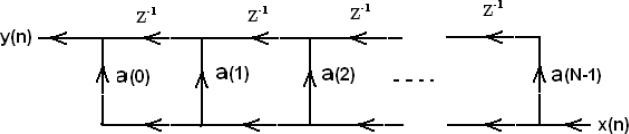
The flow-graph-reversal theorem says that if one changes the directions of all the arrows, and inputs at the output and takes the output from the input of a reversed flow graph, the new system has an identical input-output relationship to the original flow graph [10].

Figure 5.4 Direct form FIR Filter

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



5.5 Transpose-form FIR filter structure

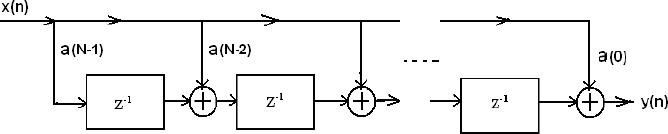


Figure 5.6 Transpose-form FIR filter structure

* + 1. **CASCADE STRUCTURE**

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

b0+ b1z−1 + b2z−2 + … … … . bt z−t = b01 − z1z−1 1 − z2z−2 … . . (1 − zt z−1)

Where the *zi* 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 − zt z−1 1 − zt z−2 into one quadratic (length-2) section with real coefficients.The overall filter can then be implemented in a cascade structure.

Figure 5.7 Cascaded Structures

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

* + 1. **LATTICE STRUCTURE**

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

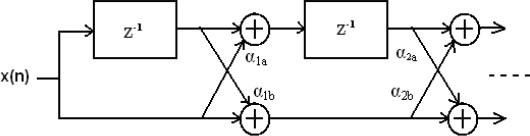


Figure 5.8 Lattice Structure

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

* 1. **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.

###### SUMMARY

In this chapter we have studied various structures available for realization of FIR filters.These structures contain multipliers,adders and delay element as their building blocks.The basic structure for implementation of FIR Filter is direct form.Lattice structure suitable for speech processing.Polyphase structure are also a kind of structure where filter coefficients are divided into groups.

## CHAPTER 6 MULTIPLIERS

**INTRODUCTION**

Multiplication is a very important arithmetic operation for many signal processing applications such as convolution, correlation, frequency analysis, image processing etc. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation.

The speed of multiplication operation is an important aspect in the processors of these applications. The processor requires considerable amount of time in performing the multiplication operation and performance of the multiplier determines system performance. Hence, the speed and power efficient multipliers are of greater importance in signal and image processing applications.

This chapter discusses the performance of different types of multipliers particularly on Vedic multipliers and constant multipliers and compares the efficiency of the multipliers in order to utilize in the processors to enhance the system performance.

* 1. **ARRAY MULTIPLIER**

Array multiplier is an efficient layout of a combinational multiplier. Multiplication of two binary number can be obtained with one micro-operation by using a combinational circuit that forms the product bit all at once thus making it a fast way of multiplying two numbers since only delay is the time for the signals to propagate through the gates that forms the multiplication array.

In array multiplier, consider two binary numbers A and B, of m and n bits. There are summands that are produced in parallel by a set of an AND gates. n x n multiplier requires n (n-2) full adders, n half-adders and n2 AND gates. Also, in array multiplier worst case delay would be (2n+1) Designing multipliers that are of high-speed, low power, and regular in layout are of substantial research interest. Speed of the multiplier can be increased by reducing the generated partial products. Many attempts have been made to reduce the number of partial products generated in a multiplication process one of them is array multiplier.

In Array multiplier, half adder has been used to sum the carry products in minimum time. Achieving high speed integrated circuits with low power consumption is a major concern for the VLSI circuit designers.

* + 1. **CONVENTIONAL ARRAY MULTIPLIER**

An existing array multiplier is very much regular in its structure when compared to the conventional array multiplier and uses an only short wire that goes from one full adder cell to adjacent full adder cell. It has very simple and efficient layout in VLSI and can be easily and efficiently pipelined. The conventional array multiplier is synthesized. This low power full adder cell has a drawback of giving large delay when compared to other adders. This multiplier consumes significant less amount of power and have more delay compared to other adders. The product of these two bits can be written as P7P6P5P4P3P2P1P0.Where P0 is the LSB AND P7 is the MSB as shown in figure 6.1.

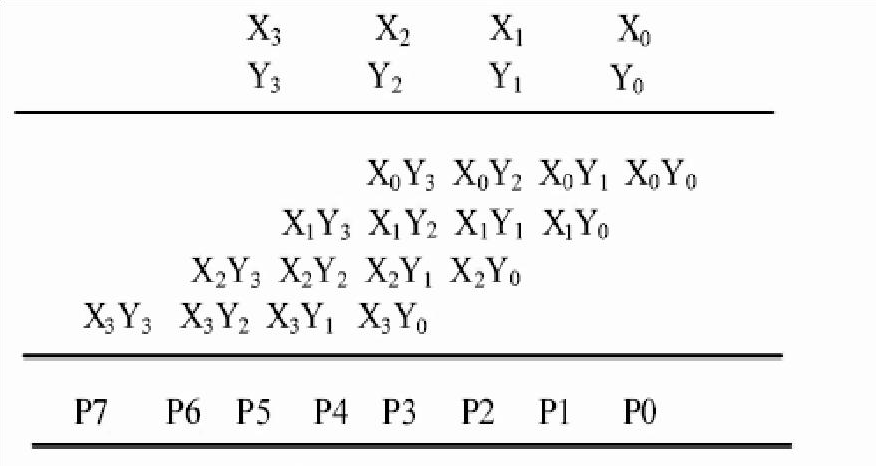


Figure 6.1: 4x4 Array Multiplication

The conventional array multiplier uses carry save addition to add the products. In the carry save addition method, the first row will be either half adders or full adders. If the first row of the partial products is implemented with full adders, Cin will be considered as 0. Then the carries of each full adder can be diagonally [15]forwarded to the next row of the adder.The resulting multiplier is said to be carry save array multiplier as the carry bits are not immediately added but rather saved for the next stage of addition. Hence the name carry save multiplier. In the design if the full adders have two input data the third input is considered as zero.

The final adder which is used to add carries and sums of the multiplier is removed. Then the carries of the multiplier at the final stage is carefully added to the inputs of the multiplier as shown in Figure 6.2. The carry of the fourth column of the multiplier is given to the input of the fifth column instead of zero. Then the carry of the fifth column is forwarded to the input of the

sixth column so on. And in this carry of the seventh column of the adder is considered as the most significant bit.

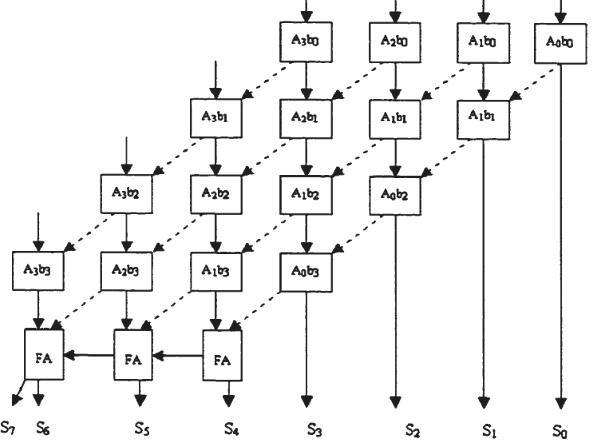


Figure 6.2: Array Multiplier

* 1. **MODIFIED BOOTH ALGORITHM**

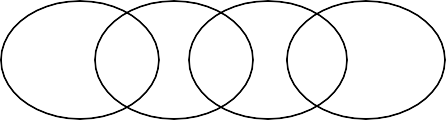
Mac Soorley [12] proposed a modification to the Booth Algorithm. The advantage of this algorithm is minimizing the partial products by half regardless of the input bits. The partial products generated in the second stage of multiplication are reduced, hence it[13] is known as one of the fastest multiplication algorithm.

Table 6.1: Booth bits

|  |  |  |
| --- | --- | --- |
| Qi | Qi-1 | No. of partial products |
| 0 | 0 | Zero partial products |
| 0 | 1 | One partial product |
| 1 | 0 | One partial product |
| 1 | 1 | Two partial products |

In Table 6.1 we can infer that in the last operation (1 1), two partial products are generated. In order to reduce the partial products, the last operation of (1 1) should be reduced or replaced according to the booth recoded table. Three bits are taken at a time and are encoded into one of

{-2, -1,0,1,2}. Grouping is started from the LSB, the first block uses only two bits of the multiplier and assumes a zero for the third bit as shown in figure 6.3.



1 1 0 1 0 1 0 1

Figure 6.3: Grouping of numbers by overlapping technique

Table 6.2: Booth Recoded Table

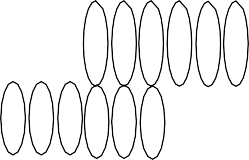
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Multiplier Bits | | | Encoded Multiplier value ∑ | Partial products | Operation |
| Qi+1 | Qi | Qi- 1 |
| 0 | 0 | 0 | 0 | 0M | No action |
| 0 | 0 | 1 | 1 | 1M | Add |
| 0 | 1 | 0 | 1 | 1M | Add |
| 0 | 1 | 1 | 2 | 2M | Shift left and add |
| 1 | 0 | 0 | -2 | -2M | Shift left and subtract |
| 1 | 0 | 1 | -1 | -1M | Subtract |
| 1 | 1 | 0 | -1 | -1M | Subtract |
| 1 | 1 | 1 | 0 | 0M | No action |

Table 6.2 describes the Booth Recoding table. In the above table, 3 bits of the Multiplier Q are taken for inspection to find the encoded value. ∑ is the encoded value which is to be multiplied with the Multiplicand ‘M’ and is given by ∑ = Qi-1 + Qi -2 \* Qi+1.

* 1. **WALLACE-TREE MULTIPLIER**

The Wallace tree multiplier [16] is a parallel multiplier and offers faster performance for larger operands because its height is logarithmic in word size, not linear. It uses the carry save addition algorithm to reduce the propagation delay.

The multiplication operation performed in the Wallace tree, every possible bit in every column is covered by the 3:2 (*full adder*) or 2:2 (half adder) compressors repetitively until the final partial product is left with a depth of only 2. Thus, to compress the partial products a Wallace tree multiplier uses more hardware is utilized to get final product as quickly as possible. Figure 6.4(a) shows the logic used for 8x8 bits Wallace tree multiplication and the tree structure organised according to the addition performed for the partial products shown in figure 6.4(b).



# (b)

~~(a)~~

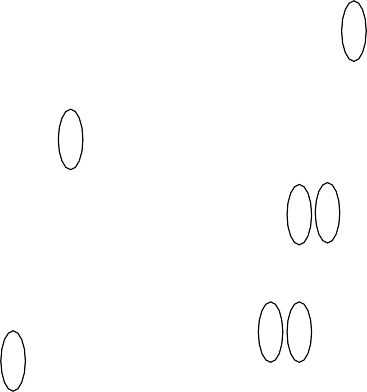
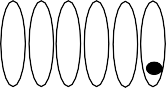
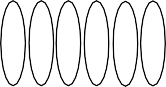
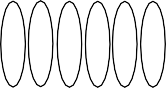
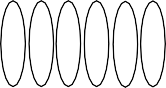


Figure 6.4 (a)Wallace tree structure for 8x8 multiplication. (b) Reorganized Matrix of Wallace tree matrix.

Wallace tree has three steps: -

1. Multiply each bit of multiplier with same bit position of multiplicand. Depending on the position of the multiplier bits generated partial products have different weights.
2. Reduce the number of partial products to two by using layers of full and half adders.
3. After second step we get two rows of sum and carry, add these rows with conventional adders.

Explanation of second step: -

As long as there are three or more rows with the same weight add a following layer:

* 1. Take any three rows with the same weights and input them into a full adder. The result will be an output row of the same weight i.e. sum and an output row with a higher weight for each three input wires i.e. carry.
  2. If there are two rows of the same weight left, input them into a half adder.
  3. If there is just one row left, connect it to the next layer.
  4. The advantage of the Wallace tree is that there are only O (log n) reduction layers (levels), and each layer has O (1) propagation delay. As making the partial products is O (1) and the final addition is O (log n), the multiplication is only O (log n), not much slower than addition (however, much more expensive in the gate count). For adding partial products with regular adders would require O (log n2) time.
  5. **VEDIC MULTIPLIER**

Vedic multiplier or Urdhva Tiryakbhyam Sutra is a multiplication algorithm [13-14] which is one of the easiest and fastest approach to perform mathematical operation. Due to coherence and symmetry, these algorithms when deployed in Digital processors consume less area with lower power consumption. If the number of bits are increased, then the gate delay and area increase very slowly as compared to other multipliers.

Figure 6.5 shows multiplication of two binary numbers, the vertical and crosswise combination of the binary bits generates the partial products concurrently. The partial products of 4x4 multiplier (P7, P6, P5, P4, P3, P2, P1, P0) are given by the following equations:

Steps for performing multiplication using sutra

The above figure illustrates the steps involved in Vedic multiplication using Urdhva Tiryagbhyam Sutra which are explained below:

* + Step1: Vertical multiplication of first digits of the numbers is done i.e. (a0b0)
  + Step2: Crosswise Multiplication Addition is done of 1st 2 digits i.e. (a1b0+a0b1)

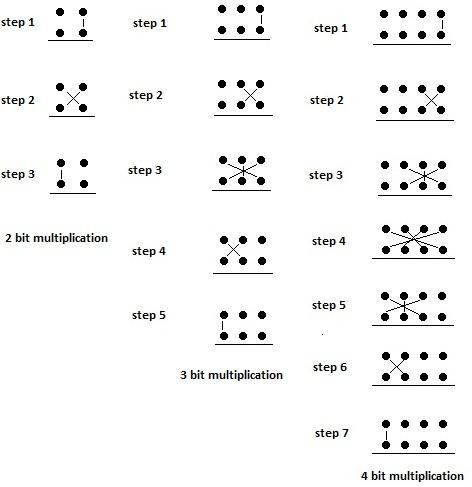


Figure 6.5: Illustration of Urdhva Tiryakbhyam Sutra for 4 bits.

* + Step3: Crosswise Multiplication addition of 1st 3 digits of the inputs is being done. (a2b0+a0b2+a1b1)
  + Step4: Crosswise multiplication addition of all the digits of both the inputs is performed (a0b3+a3b0+a2b1+a1b2)
  + Step5:Crosswise multiplication addition of last 3digits isperformed i.e.(a1b3+a3b1+a2b2)
  + Step6: Crosswise multiplication addition of last 2 digits is done i.e. (a2b3+a3b2)
  + Step7: Finally, Vertical multiplication of last two digits is performed i.e. (a3b3)

For all above steps except the first step, the carry is initialized and used for further multiplication process.

In 4\*4 Vedic multiplier, multiplication is done in a single line using Urdhva Tiryakbhyam Sutra, where in 4x4 multiplier partial products are generated to obtain the resultant product. Initially carry is taken as zero, if we encounter more than one line in any step, then all the results are added to the previous carry. In each step, LSB acts as the result bit and all other bits act as a carry for further step. Hence the number of steps required for the multiplication process gets reduced and speed of the multiplier increases.

* 1. **CONSTANT MULTIPLIER**

Constant coefficient multiplier has one fixed input and one variable input. By shifting, adding and subtracting the two variables are multiplied. The proposed multiplier design uses two techniques CSD and CSE to reduce the hardware and fixed point representation to improve the speed.

* + 1. **CANONICAL SIGNED DIGIT (CSD)**

In computing, CSD is a method that can be used to encode a value in a signed digit representation and using it one number can be represented in many ways. Probability of digit being zero is 60% and 50% in two‟s complement encoding. CSD leads to efficient implementation of adders and subtractors (one such example is multiplication by a constant) in a hardwired DSP[14].

* Important Characteristics of CSD

1. CSD representation of a number consists of numbers 0, 1, -1
2. It is a unique method of representation.
3. Number of nonzero digits is minimal.
4. There cannot be two consecutive nonzero digits.

* Representation

The representation is done using a sequence of one or more of the symbols -1, 0, 1 or (-, 0,+) i.e., each position with symbol -, + or -1 or +1 represents subtraction or addition of a power of 2.

In CSD, the representation of numbers is considered as a succession of bits, where each bit is in the set {0, 1, and -1}.

* Implementation

CSD is obtained by transforming every sequence of zero followed by ones (011…1) into +1 followed by zeros and the least significant bit by -1.Starting from the right (LSB), if there are more than one non-zero elements (1 or - 1) in a row, take all of them, plus the next zero. (if there is not zero at the left side of the MSB, create one there).

Table 6.3 Represent of binary and CSD

|  |  |  |
| --- | --- | --- |
| **Coefficient Values** | **Binary** | **CSD** |
| 0.008956 | 00111100\_00010010\_10111100\_00110000 | 01000100\_00010101\_01000100\_01010000 |
| 0.005974 | 00111011\_11000011\_11000001\_10001011 | 01000100\_01000100\_01000010\_10010101 |
| -0.00800 | 10111100\_00000011\_00010010\_01101111 | 101000100\_00000101\_00010010\_10010001 |
| -0.00132762 | 10111100\_10010101\_10000111\_01010001 | 101000101\_01010010\_00010001\_01000101 |

CSD is used to realize the multiplication when the one factor is constant. The CSD unit is used to construct number that contains the minimum possible number of non-zero bits.

**6.7 SUMMARY**

This chapter summarizes a comparative study on Vedic multipliers, conventional array multiplier, Wallace tree multiplier, modified booth multiplier and constant multiplier. Multipliers are mainly classified into two types viz. serial multiplier and parallel multiplier. Serial multiplier which requires more processing time and the Parallel multiplier such as Modified Booth multiplier, Wallace tree multiplier, Array multiplier, floating point arithmetic, fixed point arithmetic etc. Array multiplier reduces the delay, but it requires a large number of gates, which increases area and power consumption. Booth multipliers are used for multiplication of signed binary numbers but, it does not work when there are alternate zeros and ones. Wallace tree multiplier works at high speed, but it exhibits structural irregularity which is more complex for hardware implementation.

**INTRODUCTION**

## CHAPTER 7 ARITHMETIC BUILDING BLOCKS

In electronics, adder circuit performs addition of the binary numbers in various computers and other types of processors. Adder circuits are not only used in ALUs, but also used in various processors to calculate increment or decrement operations, table indices, addresses, etc. A typical adder circuit generates sum and carry as the output. When the one’s or two’s complement are being used to specify negative numbers, it is small to alter adder to subtractor. A more complex adder is used to represent other signed numbers. The applications of adder circuits are not only used to add binary numbers, but also used in [digital](http://www.edgefx.in/led-digital-voltmeter-circuit-and-working/) applications such as address, table index, decoding and calculation etc.The adders used in this proposed work are Ripple Carry Adder,Carry Save Adder,Carry Look Ahead Adder,Kogge stone Adder.

* 1. **CARRY LOOK AHEAD ADDER**

A Carry Look ahead (Look Ahead) Adder is made of a number of [full-adders](https://www.nandland.com/vhdl/modules/module-full-adder.html) cascaded together. It is used to add together two binary numbers using only simple [logic gates](https://www.nandland.com/vhdl/examples/example-logical-operators.html). Four full-adders connected together to produce a 4-bit carry look ahead adder. Carry look ahead adders are similar to [Ripple Carry Adders](https://www.nandland.com/vhdl/modules/module-ripple-carry-adder.html). The difference is that carry look ahead adders are able to calculate the Carry bit before the Full Adder is done with its operation. This gives it an advantage over the Ripple Carry Adder because it is able to add two numbers together faster. The drawback is that it takes more logic.

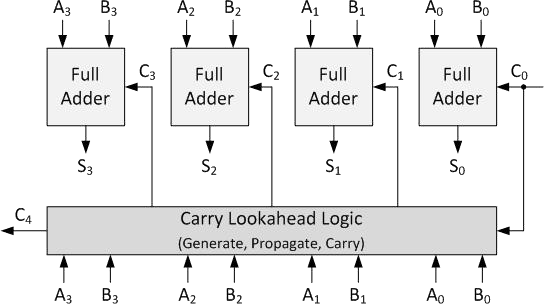


Figure 7.1 block diagram of carry look ahead adder

A full adder is a combinational circuit that performs the arithmetic sum of three input bits: augends Ai, addend Bi and carry in in C from the previous adder. Its results contain the sum Si and the carry out, out C to the next stage.

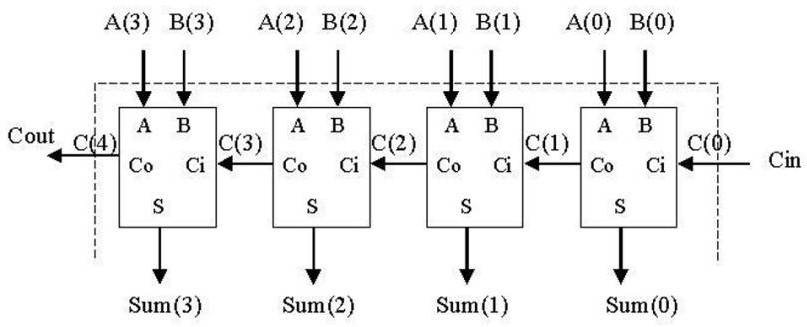


Figure 7.2 circuit diagram of carry look ahead adder

So to design a 4-bit adder circuit we start by designing the 1 –bit full adder then connecting the four 1-bit full adders to get the 4-bit adder as shown in the diagram above. For the 1-bit full adder, the design begins by drawing the Truth Table for the three inputs and the corresponding output SUM and CARRY. The Boolean Expression describing the binary adder circuit is then deduced. The equations of carry look ahead adder are

Pi= Ai xor Bi

Gi = Ai Bi Ci+1 = Gi + PiCi

Si = Pi xthCi

#### CARRY-SAVE ADDER

A carry-save adder is a type of [digital adder](https://en.wikipedia.org/wiki/Adder_(electronics)), used in computer microarchitecture to compute the sum of three or more *n*-bit numbers in [binary](https://en.wikipedia.org/wiki/Binary_numeral_system). It differs from other digital adders in that it outputs two numbers of the same dimensions as the inputs, one which is a sequence of partial sum bits and another which is a sequence of [carry](https://en.wikipedia.org/wiki/Carry_(arithmetic)) bits.

There are many cases where it is desired to add more than two numbers together. The straightforward way of adding together m numbers (all n bits wide) is to add the ﬁrst two, then add that sum to the next, and so on. This requires a total of m−1 additions, for a total gate delay of O(mlgn) (assuming look ahead carry adders). Instead, a tree of adders can be formed, taking only O(lgm·lgn) gate delays. Using carry save addition, the delay can be reduced further still. The idea is to take 3 numbers that we want to add together, x + y + z, and convert it into 2 numbers c + s such that x + y + z = c + s, and do this in O(1) time. The reason why addition cannot be performed in O(1) time is because the carry information must be propagated. In carry save addition, we refrain from directly passing on the carry information until the very last step. To add three numbers by hand, we typically align the three operands, and then proceed column

by column in the same fashion that we perform addition with two numbers. The three digits in a row are added, and any overﬂow goes into the next column. Observe that when there is some non-zero carry, we are really adding four digits (the digits of x, y and z, plus the carry).

X3 Y3

X2 Y2

X1 Y1

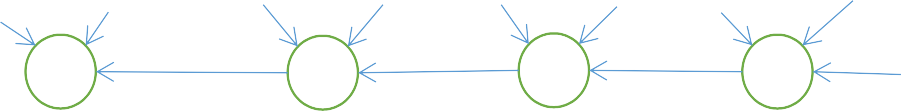
X0 Y0

CSA

CSA

CSA

C3 s3 C2 s2 C1 s1 C0 s0



+

+

+

+

CSA

Figure 7.3 carry save adder

* 1. **RIPPLE CARRY ADDER**

Multiple full adder circuits can be cascaded in parallel to add an N-bit number. For an N- bit parallel adder, there must be N number of full adder circuits. A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. It is called a1ripple carry adder because each carry bit gets rippled into the next stage. In a ripple carry adder the sum and carry out bits of any half adder stage is not valid until the carry in of that stage occurs. Propagation delay inside the logic circuitry is the reason to make the circuit slow. Propagation delay is time elapsed between the application of an

input and occurrence of the corresponding output. Consider a NOT gate, When the input is “0” the output will be “1” and vice versa. The time taken for the NOT gate’s output to become “0” after the application of logic “1” to the NOT gate’s input is the propagation delay here. Similarly the carry propagation delay is the time elapsed between the application of the carry in signal and the occurrence of the carry out (Cout) signal. Circuit diagram of a 4-bit ripple carry adder is shown below.

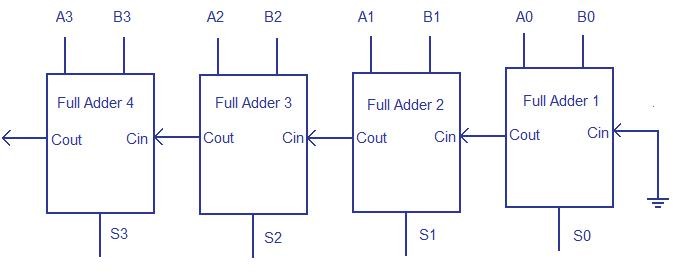


Figure 7.4 Ripple carry adder

Sum out S0 and carry out Cout of the Full Adder 1 is valid only after the propagation delay of Full Adder 1. In the same way, Sum out S3 of the Full Adder 4 is valid only after the joint propagation delays of Full Adder 1 to Full Adder 4. In simple words, the final result of the ripple carry adder is valid only after the joint propagation delay of all full adder circuits inside it.

**7.2 SUMMARY**

In this chapter, we have discussed about the different adders. These adders are very essential during the design of the processor.The adders learnt were ripple carry adder,carry save adder,carry look ahead adder,koggee stone adder.From table 8.3 in chapter 8 RCA is delay efficient and area efficient,later carry look ahead adder is present.

## CHAPTER 8 RESULTS AND DISCUSSIONS

#### CONCLUSION

**CHAPTER 9 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 thirty one order band-pass FIR filter is implemented in Virtex6 XC6VLX760 FPGA. The Direct form and Transpose structure of these filters are implemented. In both the forms of structures, N Adder and N+1 multipliers are used to realize the N order Band pass filter. The designed filters can work for real time Receivers or processing of any digital signal.

Here in this work number representation was discussed.Efficient floating point arithmetic unit with high precision was developed using Verilog code on Xilinx ISE 14.7 and comparison were made.In this thesis Five different multipliers Array multiplier, Wallace tree multiplier, Vedic multiplier, Modified Booth multiplier and constant multiplier were implemented in floating point format.

This presented work has successfully developed an efficient data path unit for an FIR filter processor using different types of multiplier unit in floating point representation and implemented it for 31-order using Direct form and Transpose form structures in Xilinx Virtex-6 FPGA. The analysis shows that using a vedic multiplier in the filter structure will be efficient way to reduce area compared to other multipliers.

* 1. **FUTURE WORK**

Optimizing a hardware structure is a trade-off between various design constraints, such as performance, resource utilization, power consumption, and precision. This depends of course on the application too. Here in this thesis two structures (direct form and transposed form) were used and a comparison was drawn between the window techniques for the receiver. Their are other structures like poly phase ,lattice structures which can also be studied in this context.Another future work could be a re-configurable filter implementation which are high in demand.

## 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]Akanksha Kant,Shobha Sharma, “APPLICATIONS OF VEDIC MULTIPLIER DESIGNS”IEEE 2015

1. Joseph Cavanagh,”Computer Arithmetic and Verilog HDL Fundamentals”2010 by Taylor & Francis Group, LLC
2. Mi Lu,”Arithmetic and Logic in Computer Systems”2004 by John Wiley & Sons, Inc [7]Proakis,J.G., Manolakis,D.G., “Digital Signal Processing” 3rd Edition, PHI publication 2004.
3. Anubhuti Mittal , Ashutosh Nandi, Disha Yadav presented this paper “Comparative study of 16-order FIR filter design using different multiplication Techniques”The Institution of Engineering and Technology 2017
4. Aneela Pathan,Tayab D Memon,Sharmeen Keerio,Imtiaz Hussain Kalwar,“FPGA Based performance analysis of multiplier policies for FIR filter”The Institution of Engineering and Technology 2016
5. Evangelos Kyritsis ,Kiamal Pekmestzi in their paper “Hardware Efficient Fast FIR Filter Based on Karatsuba Algorithm”2016 IEEE
6. Ila Sharma,Anil Kumar,Girish Kumar Singh, Heung-No Lee,”Design of multiplierless prototype filter for two-channel filter bank using hybrid method in FCSD space”.The Institution of Engineering and Technology 2016
7. Sandesh S. Saokar, R. M. Banakar, Saroja Siddamal, “High Speed Signed Multiplier for Digital Signal Processing Applications” source: 978-1-4673-1318-6/12/$31.00 ©2012 IEEE. [13]David Villeger, Vojin G Oklobdzija, “Evaluation of Booth Encoding Techniques for Parallel Multiplier Implementation” source: Volume 29, Issue 23, 11 November 1993, ISSN 0013-5194. [14]R. Hartley, “Subexpression sharing in filters using canonic signed digit multipliers,” IEEE Trans. Circuits & Syst. II, vol. 43, Oct. 1996.

[15]S. D. Pezaris, "A 40-ns 17-Bit by 17-Bit Array Multiplier", IEEE Trans. on Computers, pp. 442-447, Abr. 1971.

[16]Ron S. Waters, Earl E. Swartz lander, “A Reduced Complexity Wallace Multiplier Reduction,” IEEE TRANSACTIONS ON COMPUTERS, VOL. 59, NO. 8, pp. 1134 – 1137,

AUGUST 2010.