IMPLEMENTATION OF FIXED POINT NOISE COMPENSATION IN IIR FILTERS USING ERROR FEEDBACK

By

HARIS MASOOD

Submitted to the Department of Computer Engineering

in Partial Fulfillment of the requirements for the Degree of

Master of Science

In

Computer Engineering

Thesis Advisor

Dr. Shoab Ahmed Khan

College of Electrical and Mechanical Engineering

National University of Sciences and Technology, Pakistan

2009

In the name of Allah, the most Merciful and the most Beneficent

Abstract

Infinite Impulse Response filters (IIR) with different approximations was first realized in 19th century. IIR filters have gained enormous popularity in the recent years due to its inherent ability to filter out the signals depending upon the approximation being used. A lot of work has been done on IIR filters since then, however concept of feedback compensation has been partially neglected. This thesis involves investigating a procedure to build an effective IIR filter with one of its trademark specifications. Different approximations have also been discussed to make IIR filter. Comparisons have also been made between different approximations with pros and cons of these approximations have also been stated.

The second part of thesis represents an implementation and tradeoff analysis of compensated IIR filter with Error feedback (EF). Hardware analysis is made so that a better comparison can be made between the two techniques i.e. IIR filter with error feedback and without feedback. A simple method is devised to reduce the quantization error of the biquadratic section implementation by means of EF approach. The compensation method including the EF application utilized the fact that filter with an EF is more resistant to noise then filter without EF. Then an iterative method can be used afterwards to reduce the silicon area. First a standard recursive IIR filter is developed with basic quantization properties. Then a more advanced EF-compensated biquadratic section is designed. The compensated portion of the filter involves the calculation of feedback coefficients. Different methods have been stated for calculation of feedback coefficients. The quantization error of both the solutions is compared numerically and graphically. Although EF implementation results in more usage of resources and more complexity but its results are less error prone at the output of the quantizer. The designs are implemented using Xilinx ACCELDSP 10.1 and graphs have been developed using Matlab.

Acknowledgement

I am really thankful to Allah for guiding me throughout my life and in this thesis as well. It was Allah's blessings that helped me produce this work. I'll always be grateful to my parents who believed in me and my abilities and it is their overwhelming support which enabled me to complete my MS thesis. I must pay my special thanks to my advisor, Dr. Shoab Ahmed Khan, for his kind guidance and constant motivation. I am particularly grateful that he reposed confidence in my abilities and stimulated my research in this area. The many thought provoking sessions I had with him have increased my interest and thirst to gain further understanding of my field of interest manifold.

Most importantly, I am thankful to my family for their unflinching support and encouragement they have always rendered. My friends, Aqeel Ahmad and Yasir Munir were also quite helpful throughout this process and I benefited a lot from friendly discussion with them about filtering and related concepts.

People at College of Electrical & Mechanical Engineering (EME) have also been very kind and friendly and they provided me an excellent environment to transform my ideas into reality. I will remember the exciting days I spent here with some of the most talented people around.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1.1 Introduction

This thesis analyzes the use of Infinite Impulse response filter to reduce the quantization error in biquadratic section implementation at the output of the quantizer. The goal of this work is to present the methodology and results of optimization of the biquadratic section implementation hardware cost with the help of the quantization error feedback (EF). The optimized structure has to keep all the properties of the biquadratic section without the error feedback, roundoff noise level including. The standard and the EF implemented biquadratic section implementations are both analyzed, their parameters and values are compared from various points of view (utilized logic, maximal clock frequency and quantization signal to noise ratio).

Error Feedback (EF) is a general method that can be used to reduce errors inherent in any quantization operation. It is better to reduce as much quantization error as possible at the output of the quantizer to get better results. To our knowledge, error feedback techniques are widely used in applications like predictive speech coding, predictive image coding, and sigma-delta analog-to-digital conversion. Error feedback can also be used to reduce quantization errors generated in finite word length implementations of recursive digital filters. Especially with fixed-point implementations of narrow-band low-pass filters its effect is much more appreciable than any other low noise structures.

To implement compensated structure i.e. a biquadratic structure with EF logic embedded with it we might have to add some extra resources and logic but the overall results obtained are much more emphatic and precise as compared to the filters with non EF logic. There are different types of IIR filters with each one having their own advantages and disadvantages but here the type used is an Elliptic IIR filter. An elliptic filter (also known as a Cauer filter)is an electronic filter with equalized ripple (equiripple) behavior in both the pass band and the stop band. The amount of ripple in each band is independently adjustable, and no other filter of equal order can have a faster transition in gain between the pass band and the stop band, for the given values of ripple (whether the ripple is equalized or not). Alternatively, one may give up the ability to

independently adjust the pass band and stop band ripple, and instead design a filter which is maximally insensitive to component variations. Elliptic filters are generally specified by requiring a particular value for the pass band ripple, stop band ripple and the sharpness of the cutoff. This will generally specify a minimum value of the filter order which must be used. Another design consideration is the sensitivity of the gain function to the values of the electronic components used to build the filter. The filter order and specifications used in this thesis work are chosen as to maximize the effect of EF quantization so that the resulting structure obtained is as less error prone as possible.

This thesis in general explores a method to generate an IIR filter structure with and without error feedback logic. There are many different methods by which the goal of minimization of error can be achieved. A complete biquadratic filter structure has been implemented here with poles and zeros are chosen to be close to the unit circle for better optimization of results. Different algorithms can also be added if required for the calculation of feedback coefficients involved in the calculation of compensated portion of the filter. After the calculation of feedback coefficients the compensated portion of the IIR filter is implemented. The results of both the sections i.e. with EF and without EF are analyzed and compared. Matlab graphs and Virtex tool are used for the analysis and comparison of both the techniques in terms of minimization of error, resources used i.e. Flip Flops (FF) and Look up Tables(LUT) and frequency. First the biquadratic sections with and without EF are implemented in Matlab and then to implement both the sections on hardware they are coded in Modelsim verilog tool. Synthesis results are obtained finally using the Virtex 2 FPGA tool. The results of both the techniques are compared and results obtained from FPGA synthesis report are finally plotted in Matlab for better optimization and comparison of results. Finally in the end we propose some methods and ideas for better improvement and optimization of results.

CHAPTER 2

2.1 Infinite Impulse Response (IIR) Filter

In this chapter we'll try to describe the designing, implementation and basic types of an Infinite Impulse Response (IIR) filter.IIR filters have usually infinite impulse responses, hence they can be, matched with analog filters all of which have infinite impulse responses. Three different types of an IIR filter are also explained with their respective properties and comparison between them. This digital filter has a gain curve that approximates the filter characteristics of a corresponding analog filter. IIR filters are digital filters which are used where analog filters are used to approximate the gain and phase response of analog filters. However, an area where using an IIR filter gives a definite edge is that when it is implemented in hardware it provide much more flexibility, reduces degradation and noise at the output of quantizer, provides better accuracy in terms of number of bits used and provides filter reproducibility. Some basic areas where they are used commonly are for sound and music enhancement, telecommunications, video image processing, biomedical instrumentation, and radar and sonar processing. There are many mathematical formulas that approximate an analog filter with an IIR filter, but the method we are using in this thesis is to approximate the IIR filter such that the poles and zeros are placed very close to unit circle [1].

Usually the specifications for a digital filter are given in terms of normalized frequencies. Also, in many applications, the specifications for an analog filter are realized by a digital filter in the combination of an Analog to digital converter (ADC) in the front end with a Digital to Analog converter (DAC) at the receiving end, and these specifications will be in the analog domain. The magnitude and frequency responses of the analog filters are shown in Figure 2-1.

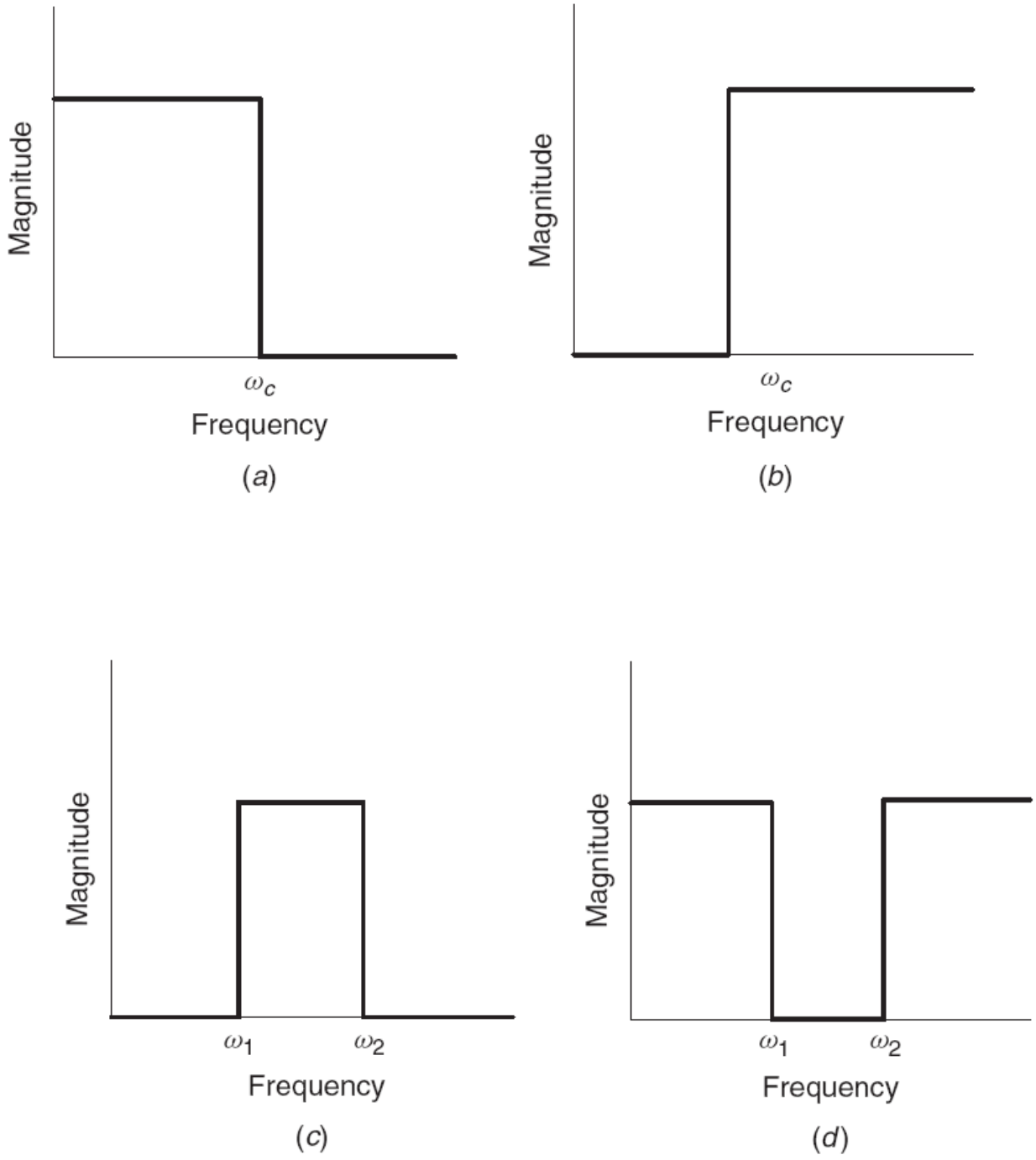

Figure 2-1: Magnitude responses of filter: (a) lowpass filter (b) highpass filter (c) bandpass filter (d) bandstop filter.

The primary advantage of IIR filters over FIR filters is that they typically meet a given set of specifications with a much lower filter order than a corresponding FIR filter. Although IIR filters have nonlinear phase, data processing within MATLAB is commonly performed "off-line," that is, the entire data sequence is available prior to filtering. This allows for a noncausal, zero-phase filtering approach which eliminates the nonlinear phase distortion of an IIR filter.

The thesis core portion contains a recursive portion and a feedback portion. T describe first portion of thesis which contains filter without EF portion contains only a recursive part. The recursive portion of the algorithm can be stated as:

$$y(n) = -\sum_{k=1}^{N} a(k)y(n-k) + \sum_{k=0}^{M} b(k)x(n-k) \qquad \textbf{(2.1)}$$

Where N specifies feedforward filter order, b specifies feedforward filter coefficients, a specifies feedforward filter coefficients, x (n) specifies the input signal whereas y(n) specifies the output signal. A more condensed form of the difference equation stated above can be written as:

$$y(n) = 1/a_0\left(\sum_{k=0}^{M} b(k)x(n-k) - \sum_{k=1}^{N} a(k)y(n-k)\right) \qquad \textbf{(2.2)}$$

Equivalent form of equation (2.2) can be written as:

$$\sum_{k=0}^{M} b(k)x(n-k) = \sum_{k=1}^{N} a(k)y(n-k) \quad \text{Where a(0)=1} \qquad \textbf{(2.3)}$$

The transfer function of IIR filter deduced from the above mentioned equation can be written as:

$$H(z) = \frac{\sum_{k=0}^{M} b(k)z^{-k}}{\sum_{k=0}^{N} a(k)z^{-k}} \quad \text{Where a(0)=1} \qquad \textbf{(2.4)}$$

Considering that in most IIR filter designs coefficient a(0)=1, the IIR filter transfer function takes the more traditional form:

$$H(z) = \frac{\sum_{k=0}^{M} b(k) z^{-k}}{1 + \sum_{k=1}^{N} a(k) z^{-k}} \quad \textbf{(2.5)}$$

The transfer function stated above helps us to identify whether the system is bounded-input bounded-output stable or not [1].

Before Implementing the implementation of IIR filter, the transfer function must be evaluated onto the unit circle i.e. $z=e^{j\omega}$ (where "ω" is normalized frequency in radians). Some properties of evaluation of the transfer function onto unit circle is given as:

$$H(e^{j\omega}) = \frac{\sum_{k=0}^{M} b(k)\cos(k\omega) - j\sum_{k=0}^{M} b(k)\sin(k\omega)}{\sum_{k=0}^{N} a(k)\sin(k\omega) - j\sum_{k=0}^{M} a(k)\sin(k\omega)} \quad \textbf{(2.6)}$$

In the above mentioned equation $H(e^{j\omega})$ is the frequency response of the filter or in other words it can be written as the discrete time Fourier transform (DTFT) of the filter. It can also be written as the product of frequency response and phase response i.e.

$$H(e^{j\omega}) = \left|H(e^{j\omega})\right| . e^{j\theta(\omega)}, \quad \textbf{(2.7)}$$

where $\left|H(e^{j\omega})\right|$ is magnitude response and $e^{j\theta(\omega)}$ is phase responses of the transfer function.
If x[n] is the input signal to the filter with a frequency response of $X(e^{jw})$ such that…

$$X(e^{j\omega}) = \left|X(e^{j\omega})\right| . e^{j\alpha(\omega)} \quad \textbf{(2.8)}$$

If $\left|X(e^{j\omega})\right|$ is the magnitude response and $e^{j\alpha(\omega)}$ is phase response of the input signal then

The frequency response $Y(e^{j\omega})$ can also be written as the product of frequency responses of the filter and the input signal i.e. $Y(e^{j\omega})=X(e^{j\omega}).H(e^{j\omega})$. The expression for the frequency response of the output signal can also be written as:

$$Y(e^{j\omega}) = \left|X(e^{j\omega})\right|\left|H(e^{j\omega})\right| e^{j\{a(\omega)+\theta(j\omega)\}} \qquad \textbf{(2.9)}$$

Eq (2.9) shows that the magnitude of the output signal is multiplied by $H(e^{j\omega})$ and consequently its phase will increase by $\theta(e^{jw})$ of the filter[1][2].

$$\left|H(e^{j\omega})\right| = \left\{ \frac{[\sum_{k=0}^{M} b(k)\cos(k\omega)]^2 + [\sum_{k=0}^{M} b(k)\sin(k\omega)]^2}{[\sum_{k=0}^{N} a(k)\cos(k\omega)]^2 + [\sum_{k=0}^{M} a(k)\sin(k\omega)]^2} \right\}^{1/2} \qquad \textbf{(2.10)}$$

The phase $\theta(e^{jw})$ can also be stated as:

$$\theta(j\omega) = -\tan^{-1}\frac{\sum_{k=0}^{M} b(k)\sin(k\omega)}{\sum_{k=0}^{M} b(k)\cos(k\omega)} + \tan^{-1}\frac{\sum_{k=0}^{M} a(k)\sin(k\omega)}{\sum_{k=0}^{N} a(k)\cos(k\omega)} \qquad \textbf{(2.11)}$$

Therefore magnitude squared function deduced from the above mentioned equations is

$$\left|H(e^{j\omega})\right|^2 = \left|H(e^{j\omega})H(e^{-j\omega})\right| \qquad \textbf{(2.12)}$$

$$= \left|H(e^{j\omega})H^*(e^{j\omega})\right|$$

where $H^*(e^{j\omega})= H(e^{-j\omega})$ is the complex conjugate of filter frequency response $H(e^{j\omega})$. It can be shown that the magnitude response is an even function of ω while the phase response is an odd

function of ω. Very often it is convenient to compute and plot the log magnitude of $\left|H(e^{j\omega})\right|$ as $10\log\left|H(e^{j\omega})\right|^2$ measured in decibels.

Designing an IIR filter usually means that we find a transfer function H(z) in the form of equation (2.3) such that its magnitude response (or the phase response, the group delay, or both the magnitude and group delay) approximates the specified magnitude response in terms of a certain criterion. For example, we may want to amplify the input signal by a constant without any delay or with a constant amount of delay. But it is easy to see that the magnitude response of a filter or the delay is not a constant in general and that they can be approximated only by the transfer function of the filter.

In general IIR filters have infinite duration impulse responses hence they can be matched to analog filters all of which generally have infinite long impulse responses. Two analytical methods are commonly used for the design of IIR digital filters, and they depend significantly on the approximation theory for the design of continuous-time filters, which are also called analog filters. Therefore, it is essential to review the theory of magnitude approximation for analog filters before discussing the design of IIR digital filters.

2.2 Approximation of Analog Filters

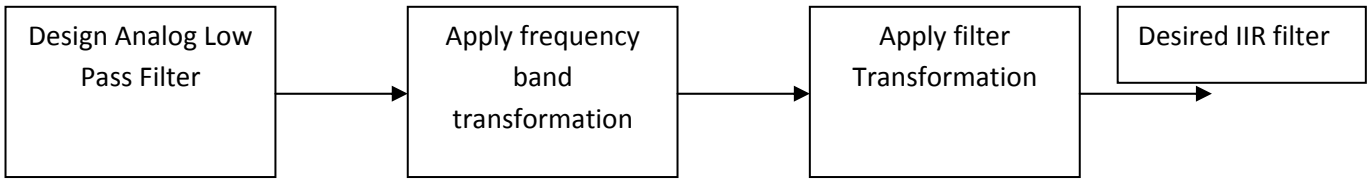

Figure 2-2: An approach to design IIR filter

The first approach is normally used in MATLAB to design IIR filters . A straight forward use of the built-in functions don't give the insight look into the design methodology. Hence in IIR filters designing technique following three steps are very important to follow.

- Design Analog lowpass filters.

- Study and apply filter transformations to obtain effective digital low pass filters.
- Study and apply frequency band transformations to obtain other digital filters from a simple digital low pass filters.

The transfer function of an analog filter H(s) is a rational function of the complex frequency variable "s", with real coefficients and is of the form [3]:

$$H(s) = \frac{c_0 + c_1 s + c_2 s^2 + \ldots c_t s^t}{d_0 + d_1 s + d_2 s^2 + \ldots d_k s^k}, t \le k \tag{2.13}$$

The frequency response or the Fourier transform of the filter is obtained as a function of the frequency ω, by evaluating H(s) as a function of jω,

$$H(j\omega) = \frac{c_0 + jc_1\omega - c_2\omega^2 - jc_3\omega^3 \ldots j^t c_t \omega^t}{d_0 + jd_1\omega - d_2\omega^2 - jd_3\omega^3 \ldots j^k d_k \omega^k}, t \le k \tag{2.14}$$

$$= \left|H(j\omega)\right| e^{j\phi(\omega)}$$

where H(jω) is the frequency response, |H(jω)| is the magnitude response, and θ(jω) is the phase response. We also find the magnitude squared and the phase response from the following:

$$\left|H(j\omega)\right|^2 = H(j\omega)H(-j\omega) \tag{2.15}$$

$$= H(j\omega)H^*(j\omega)$$

$$\frac{H(j\omega)}{H(-j\omega)} = e^{2\theta(\omega)} \tag{2.16}$$

The magnitude response of an analog filter is an even function of ω, whereas the phase response is an odd function. Although these properties of H(jω) are similar to those of $H(e^{j\omega})$, there are some differences. For example, the frequency variable ω in H(jω) is (are) in radians per second, whereas ω in $H(e^{j\omega})$ is the normalized frequency in radians. The magnitude response |H(jω)| (and the phase response) is (are) aperiodic in ω over the doubly infinite interval $-\infty < \omega < \infty$, whereas the magnitude response $\left|He^{j\omega}\right|$ (and the phase response) is (are) periodic with a period of 2π on the normalized frequency scale.

A comparison can be made between the FIR and IIR filters based on their properties. The main difference between an FIR and IIR filter can be stated as follows:

- Recursive (IIR) filters can be used to achieve the same processing as almost all non-recursive (FIR) filters but have the advantage of needing fewer multipliers/ coefficients and so are simpler in terms of hardware.

- As with any system that uses feedback, a recursive filter can be unstable – this is not the case with FIR filters - these are always stable. (In addition to that IIR filters also have a 'phase problem' compared to FIR filters)[2].

- IIR filters can achieve a given filtering characteristic using less memory and calculations than a similar FIR filter.

However, IIR filters are more susceptible to problems of finite-length arithmetic, such as noise generated by calculations, and limit cycles. (This is a direct consequence of feedback: when the output isn't computed perfectly and is fed back, the imperfection can compound). Therefore in implementing the IIR filters some assumptions have to make to make the results better and for the better optimization of the whole process.

CHAPTER 3

Basic Analog IIR filter Approximations

Many types of analog filters can be built. Anyone could be a lowpass, highpass, bandpass, or a stopband filter depending upon the user specifications. Although the type of IIR filter used in our research work is Elliptic low pass filter, but it's appropriate to discuss some analog IIR filter approximations so that a better comparison can be made between all the approximations. Comparison between different filter approximations also enables us to justify our selection of Elliptic lowpass filter in our system model.

3.1 Introduction to Basic IIR filters Approximations

Many types of analog filters can be built. Anyone could be a lowpass, highpass, bandpass, or a stopband filter as described earlier. However, because of the nature of electrical circuits used to build analog filters, any of these filter types can be divided into four basic analog approximations that meet the graphical specification. These approximations are based on where the gain curve has ripples or deviations from a smoothly varying curve. In the first approximation, called the Butterworth, there are no ripples in any passband or stopband. Thus the digital IIR filter has no ripples in it either. The general gain curve is given in Figure 3.1 for a lowpass filter specification. Similar graphical specifications could be drawn for highpass, bandpass, or bandstop filters.

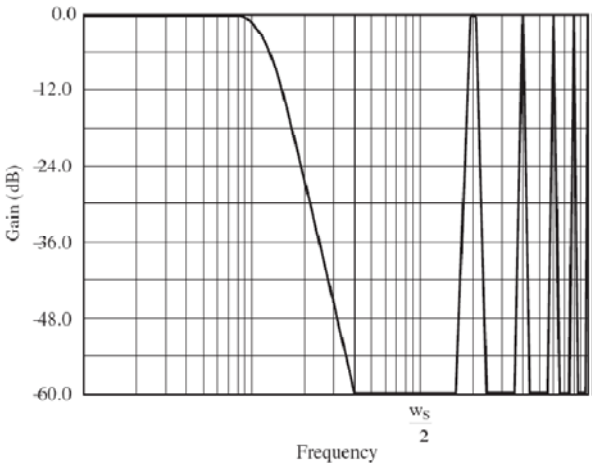

Figure 3-1: Butterworth lowpass IIR filter

Notice in Figure 3.1 that the important characteristic is that the gain curve smoothly varies in the passband and the stopband up to half the sampling frequency.

The second analog filter approximation to an ideal analog graphical filter specification is the Chebyshev, which has ripples in the passbands, but has a smoothly decreasing gain curve as compared to Butterworth filters in the stopband. In Figure 3.2 lowpass IIR filter is used to illustrate a Chebyshev approximation. Similar gain curves could be drawn for a highpass, bandpass, or bandstop filter. Notice in Figure 3.2 that the gain curve has ripple in the passband, because the gain increases before it decreases. For higher-order filters the ripple is more obvious, with several cycles of increasing and decreasing gain in the passband. This is an unwanted deviation from the ideal analog filter. However, as the Chebyshev filter will have a narrower transition band between the stop and passbands, it trades off ripple in the passband for a gain curve that more closely approximates the ideal graphical specification by having a narrower transition band than the Butterworth filter [4].

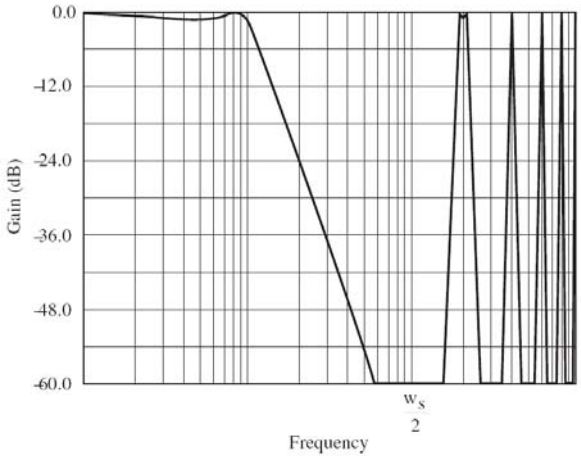

Figure 3-2: Chebyshev lowpass IIR filter

The third analog filter approximation to an ideal analog graphical filter specification is an elliptic filter. An elliptic filter (also known as a Cauer filter) is an electronic filter with equalized ripple (equiripple) behavior in both the passband and the stopband. The amount of ripple in each band is independently adjustable, and no other filter of equal order can have a faster transition in gain

between the passband and the stopband, for the given values of ripple (whether the ripple is equalized or not). Alternatively, one may give up the ability to independently adjust the passband and stopband ripple, and instead design a filter which is maximally insensitive to component variations. As the ripple in the stopband approaches zero, the filter becomes a type I Chebyshev filter. As the ripple in the passband approaches zero, the filter becomes a type II Chebyshev filter and finally, as both ripple values approach zero, the filter becomes a Butterworth filter.

Elliptic filter has ripples in the passbands, but has a smoothly decreasing gain curve as compared to Butterworth and Chebyshev filters in the stopband. In Figure 3.3 lowpass IIR filter is used to illustrate a Chebyshev approximation. Similar gain curves could be drawn for a highpass, bandpass, or bandstop filter. Notice in Figure 3.3 that the gain curve has ripple in the passband, because the gain increases before it decreases. For higher-order filters the ripple is more obvious, with several cycles of increasing and decreasing gain in the passband [4]. This is an unwanted deviation from the ideal analog filter. However, as the Elliptic filter will have a narrower transition band between the stop and passbands, it trades off ripple in the passband for a gain curve that more closely approximates the ideal graphical specification by having a narrower transition band than the Butterworth and Chebyshev filters.

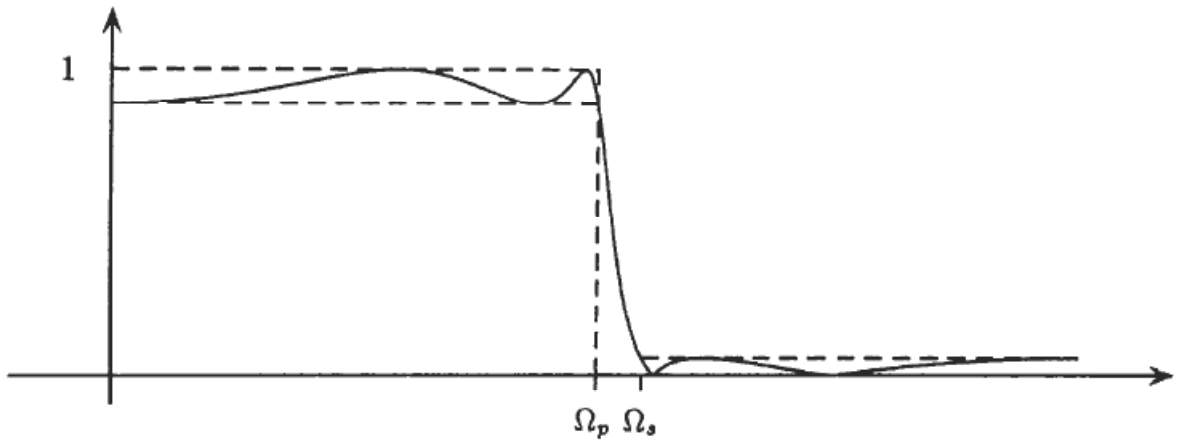

Figure 3-3: Elliptic IIR filter

The design of elliptic filters is more difficult than the Butterworth and Chebyshev filters, because their design relies on the use of tables or series expansions. Butterworth filter shown in Fig 3-1 shows that it is the smoothest one but it has no ripples. The last one is the Elliptic filter shown in

Fig 3-2 shows that it is the sharpest one but it shows ripples in both the pass-band and the stop-band. The Chebyshev filter in the middle have an average behavior, being quite sharp with ripples in part of the spectrum [6].

3.2 Butterworth Filter Approximations

The Butterworth filter is one type of electronic filter design. It is designed to have a frequency response which is as flat as mathematically possible in the passband. Another name for it is maximally flat magnitude filter. The frequency response of the Butterworth filter is maximally flat (has no ripples) in the passband, and rolls off towards zero in the stopband. When viewed on a logarithmic Bode plot, the response slopes off linearly towards negative infinity. For a first-order filter, the response rolls off at −6 dB per octave (−20 dB per decade) (all first-order lowpass filters have the same normalized frequency response). For a second-order lowpass filter, the response ultimately decreases at −12 dB per octave, a third-order at −18 dB, and so on. Butterworth filters have a monotonically changing magnitude function with ω, unlike other filter types that have non-monotonic ripple in the passband and/or the stopband[5][6].

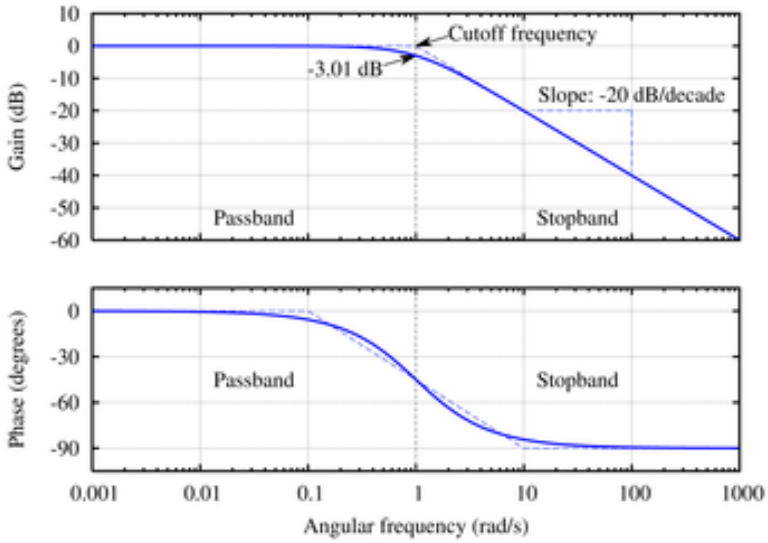

Figure 3-4: The Bode plot of a first-order Butterworth low-pass filter

Compared with a Chebyshev Type I/Type II filter or an elliptic filter, the Butterworth filter has a slower roll-off, and thus will require a higher order to implement a particular stopband

specification. However, Butterworth filter will have a more linear phase response in the passband than the Chebyshev Type I/Type II and elliptic filters.

This ideal lowpass filter as describe earlier passes all frequencies of the input continuous time signal in the interval $|\omega| \leq \omega_c$ with equal gain and completely filter out all the frequencies outside this interval. In the bandpass filter response the frequencies between $\omega 1$ and $\omega 2$ and between $-\omega 1$ and $-\omega 2$ only are transmitted and all other frequencies are completely filtered out.

For the ideal lowpass filter, the magnitude response in the interval $0 \leq \omega \leq \omega_c$ is shown as a constant value normalized to one and is zero over the interval $\omega_c \leq \omega < \infty$. Since the magnitude response is an even function, we know the magnitude response for the interval $-\infty < \omega < 0$.

For the lowpass filter, the frequency interval $0 \leq \omega \leq \omega c$ is called the passband, and the interval $\omega c \leq \omega < \infty$ is called the stopband. Since a transfer function H(s) of the form (2.13) cannot provide such an ideal magnitude characteristic, it is common practice to prescribe tolerances within which these specifications have to be met by $|H(j\omega)|$. For example, the tolerance of δ_p on the ideal magnitude of one in the passband and a tolerance of δ_s on the magnitude of zero in the stopband are shown in Figure 3-5. A tolerance between the passband and the stopband is also provided by a transition band shown in this figure. This is typical of the magnitude response specifications for an ideal filter.

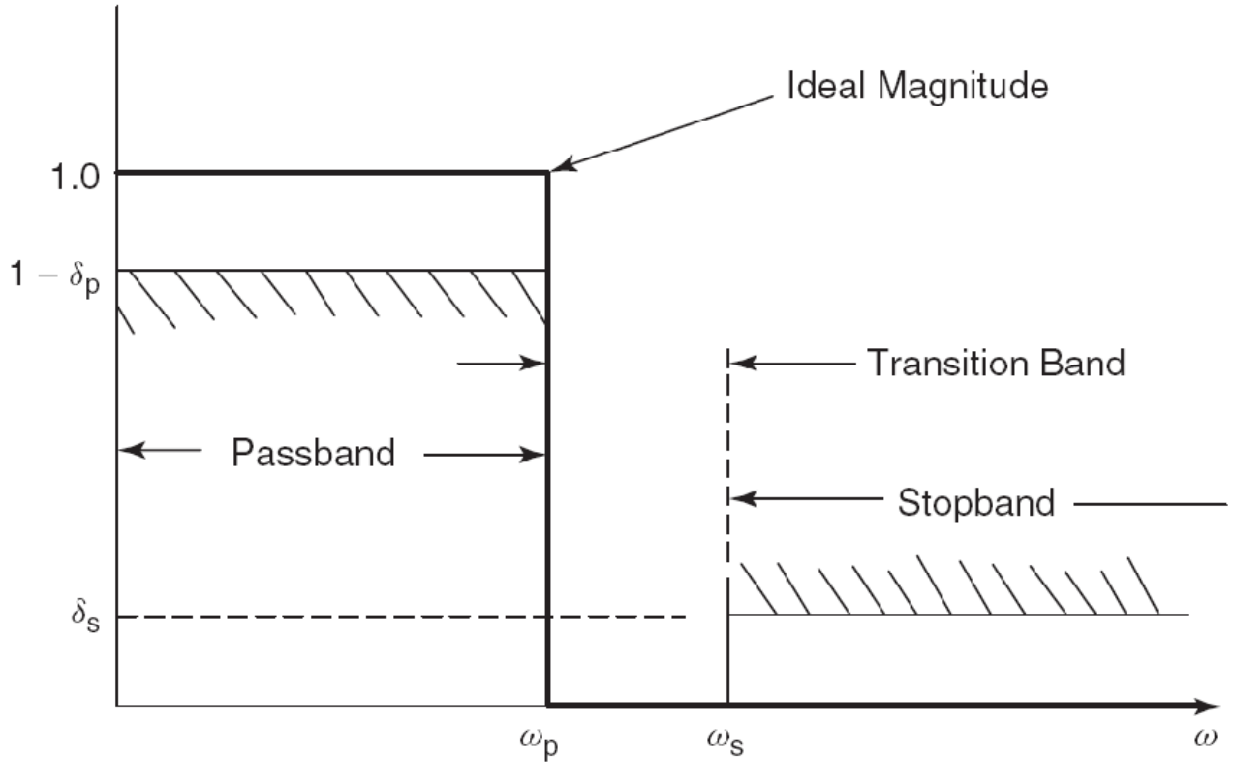

Figure 3-5: Magnitude response of an ideal lowpass analog filter showing the tolerances

In order that it approximates the magnitude of the ideal lowpass filter, let us impose the following conditions:

- The magnitude at ω = 0 is normalized to one.
- The magnitude monotonically decreases from this value to zero as ω→∞.
- The maximum number of its derivatives evaluated at ω = 0 are zero.

The magnitude response that satisfies conditions 2 and 3 is known as the Butterworth response, whereas the response that satisfies only condition 3 is known as the maximally flat magnitude response, which may not be monotonically decreasing. The magnitude squared function satisfying the three conditions is therefore of the form[1]:

$$\left|H(j\omega)\right|^2 = \frac{1}{1+D_{2n}\omega^{2n}} \qquad (3.1)$$

We scale the frequency ω by ω_p and define the normalized analog frequency $\Omega = \omega/\omega_p$ so that the passband of this filter is $\Omega_p = 1$. Now the magnitude of the lowpass filter satisfies the three conditions listed above and also the condition that its passband be normalized to $\Omega_p = 1$. Such a filter is called a prototype lowpass Butterworth filter having a transfer function H(p) = H(s/p), which has its magnitude squared function given by:

$$\left|H(j\Omega)\right|^2 = \frac{1}{1 + D_{2n}\Omega^{2n}} \tag{3.2}$$

Before we proceed with the analytical design procedure, we normalize the magnitude of the filter by H_0 for convenience and scale the frequencies ω_p and ω_s by ω_p so that the bandwidth of the prototype filter and its stopband frequency become $\Omega_p = 1$ and $\Omega_s = \omega_s/\omega_p$, respectively. The specifications about the magnitude at Ω_p and Ω_s are satisfied by the proper choice of D_{2n} and n in the function (3.2) as explained below. If, for example, the magnitude at the passband frequency is required to be $1/\sqrt{2}$, which means that the log magnitude required is −3 dB, then we choose $D_{2n} = 1$. If the magnitude at the passband frequency $\Omega = \Omega_p = 1$ is required to be $1 - \delta p$, then we choose D_{2n}, normally denoted by $\square^2$, such that:

$$\left|H(j1)\right|^2 = \frac{1}{1 + D_{2n}} = \frac{1}{1 + \varepsilon^2} = (1 - \delta_p)^2 \tag{3.3}$$

Let us consider the common case of a Butterworth filter with a log magnitude of −3 dB at the bandwidth of Ω_p to develop the design procedure for a Butterworth lowpass filter[6][1]. In this case, we use the function for the prototype filter, in the form

$$\left|H(j\Omega)\right|^2 = \frac{1}{1 + \Omega^{2n}} \tag{3.4}$$

This satisfies the following properties:

- The magnitude squared of the filter response at $\Omega = 0$ is one.
- The magnitude squared at $\Omega = 1$ is 1/2 for all integer values of n; so the log magnitude is −3 dB.

- The magnitude decreases monotonically to zero as $\Omega \to \infty$; the asymptotic rate is $-40n$ dB/decade.

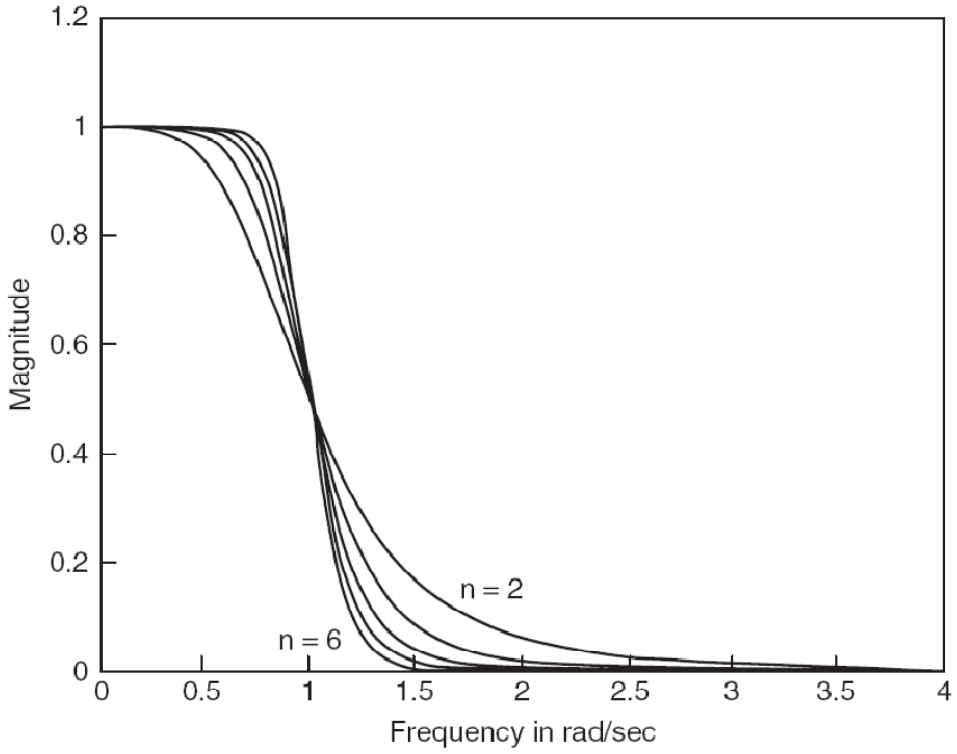

Figure 3-6: Magnitude responses of Butterworth lowpass filters.

The magnitude response of Butterworth lowpass filters is shown for n = 2, 3. . . 6 in Figure 3-6.

3.3 Chebyshev Filter Approximations

Chebyshev filters are analog or digital filters having a steeper roll-off and more passband ripple (type I) or stopband ripple (type II) than Butterworth filters. Chebyshev filters have the property that they minimize the error between the idealized filter characteristic and the actual over the range of the filter, but with ripples in the passband. Because of the passband ripple inherent in Chebyshev filters, filters which have a smoother response in the passband but a more irregular response in the stopband are preferred for some applications.

3.3.1 Chebyshev Type-I Approximation

These are the most common Chebyshev filters. The gain (or amplitude) response as a function of angular frequency ω of the nth order low pass filter is:

$$\left|H(j\Omega)\right|^2 = \frac{H_0{}^2}{1+\varepsilon^2 C_n{}^2(\Omega)} \qquad \textbf{(3.5)}$$

The Chebyshev I approximation for an ideal lowpass filter shows a magnitude that has the same values for the maxima and for the minima in the passband and decreases monotonically as the frequency increases above the cutoff frequency[1]. It has equal-valued ripples in the passband between the maximum and minimum values as shown in Figure 3-7.

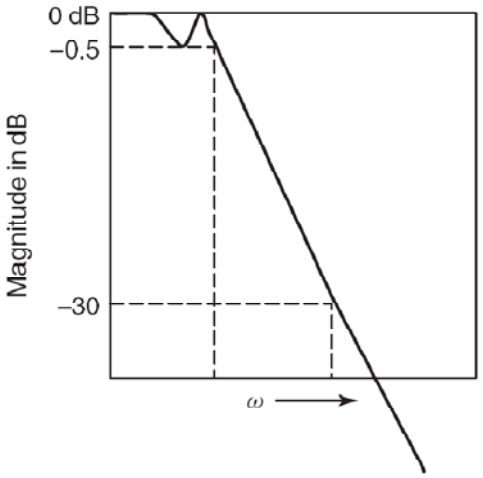

Figure 3-7: Magnitude response of Chebyshev (equiripple filter)

where $C_n(\Omega)$ is the Chebyshev polynomial of degree n. It is defined by:

$$C_n(\Omega) = \cos(n\cos^{-1}\Omega) \qquad \textbf{(3.6)}$$

Typically the specifications for a lowpass Chebyshev filter specify the maximum and minimum values of the magnitude in the passband; the cutoff frequency ω_p, which is the highest frequency of the passband; a frequency ω_s in the stopband; and the magnitude at the frequency ω_s. As in the case of the Butterworth filter, we normalize the magnitude and the frequency and reduce the

given specifications to those of the normalized prototype lowpass filter and follow similar steps to find the poles of H(p).

A property of Chebyshev I filters is that the total number of maxima and minima in the closed interval [−1 1] is n + 1. The square of the magnitude response of Chebyshev lowpass filters is shown in Figure 3-8 to indicate this particular property of the Chebyshev lowpass filters[6].

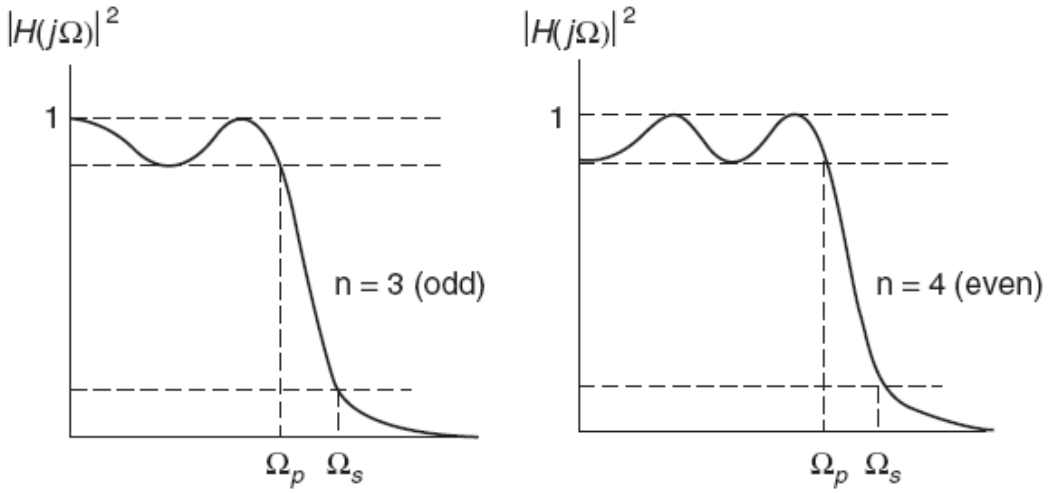

Figure 3-8: Magnitude response of Chebyshev type-I filters.

To elaborate the concepts of Chebyshev type-1 approximation equation (3-5) can be re-written as:

$$G_n(\omega) = |H_n(j\omega)| = \frac{1}{\sqrt{1+\varepsilon^2 T_n^{\,2}(\frac{\omega}{\omega_0})}} \qquad (3.7)$$

where □ is the ripple factor, ω_0 is the cutoff frequency and T_n is a Chebyshev polynomial of the nth order. The passband exhibits equiripple behavior, with the ripple determined by the ripple factor □. In the passband, the Chebyshev polynomial alternates between 0 and 1 so the filter gain will alternate between maxima at G = 1 and minima at $G = 1/\sqrt{1+\epsilon^2}$. At the cutoff frequency ω0 the gain again has the value $1/\sqrt{1+\epsilon^2}$but continues to drop into the stop band as the frequency increases. This behavior is shown in the diagram on the right. (note: the common definition of the cutoff frequency to −3 dB does not hold for Chebyshev filters.

3.3.2 Chebyshev Type-II Approximation

Also known as inverse Chebyshev, this type is less common because it does not roll off as fast as type I and requires more components. It has no ripple in the passband, but does have equiripple in the stopband. The Chebyshev II filters have a magnitude response that is maximally flat at ω = 0; it decreases monotonically as the frequency increases and has an equiripple response in the stopband. Typical magnitudes of Chebyshev II filters are shown in Figure 3-9.

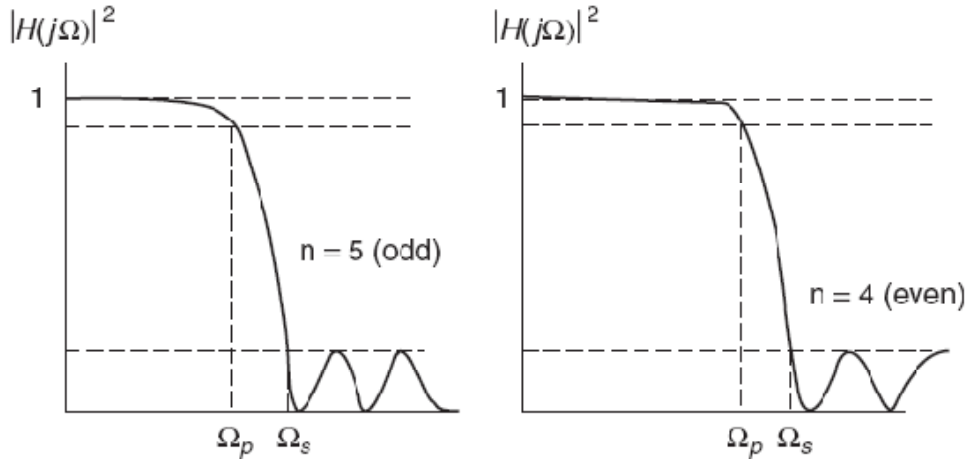

Figure 3-9: Magnitude response of Chebyshev type-II filters.

This class of filters is also called Inverse Chebyshev filters. The transfer functions of Chebyshev II filters are derived by applying the following two transformations:

- a frequency transformation Ω=1/ω in $|H(j\Omega)|^2$of the lowpass normalized prototype filter gives the magnitude squared function of the highpass filter $|H(1/j\Omega)|^2$,with an equiripple passband in |Ω|>1 and a monotonically decreasing response in the stopband 0<|Ω|<1.
- magnitude squared function of the inverse Chebyshev lowpass filter is:

$$\left|H\frac{1}{j\Omega}\right|^2 = \frac{1}{1+\varepsilon^2 C_n{}^2(1/\Omega)} \qquad \textbf{(3.8)}$$

$$1-\frac{1}{1+\varepsilon^2 C_n{}^2(1/\Omega)}=\frac{\varepsilon^2 C_n{}^2(1/\Omega)}{1+\varepsilon^2 C_n{}^2(1/\Omega)}=\left[\frac{1}{1+\dfrac{1}{\varepsilon^2 C_n{}^2(1/\Omega)}}\right] \quad \textbf{(3.9)}$$

The magnitude squared function $|H(j\Omega)|^2$ of a lowpass Chebshev-1 filter and $1-|H(1/j\Omega)|^2$ are shown in Figure 3-10

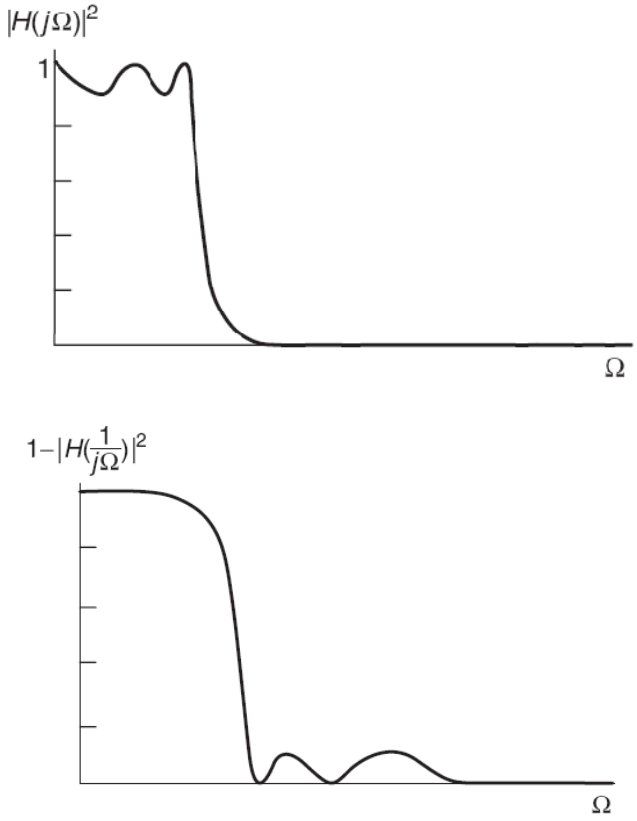

Figure 3-10: Transformation of Chebyshev I to Chebyshev II filter response

We make two important observations in Figure 3-10. The normalized cutoff frequency Ω= 1 becomes the lowest frequency in the stopband of the inverse Chebyshev filter at which the magnitude is $\epsilon^2/1+\epsilon 2$. Hence the frequencies ω_p and ω_s specified for the inverse Chebyshev filter must be scaled by ω_s and not by ω_p to obtain the prototype of the inverse Chebyshev filter. The magnitude response of Chebyshev II lowpass filter is shown in Figure 3-11.

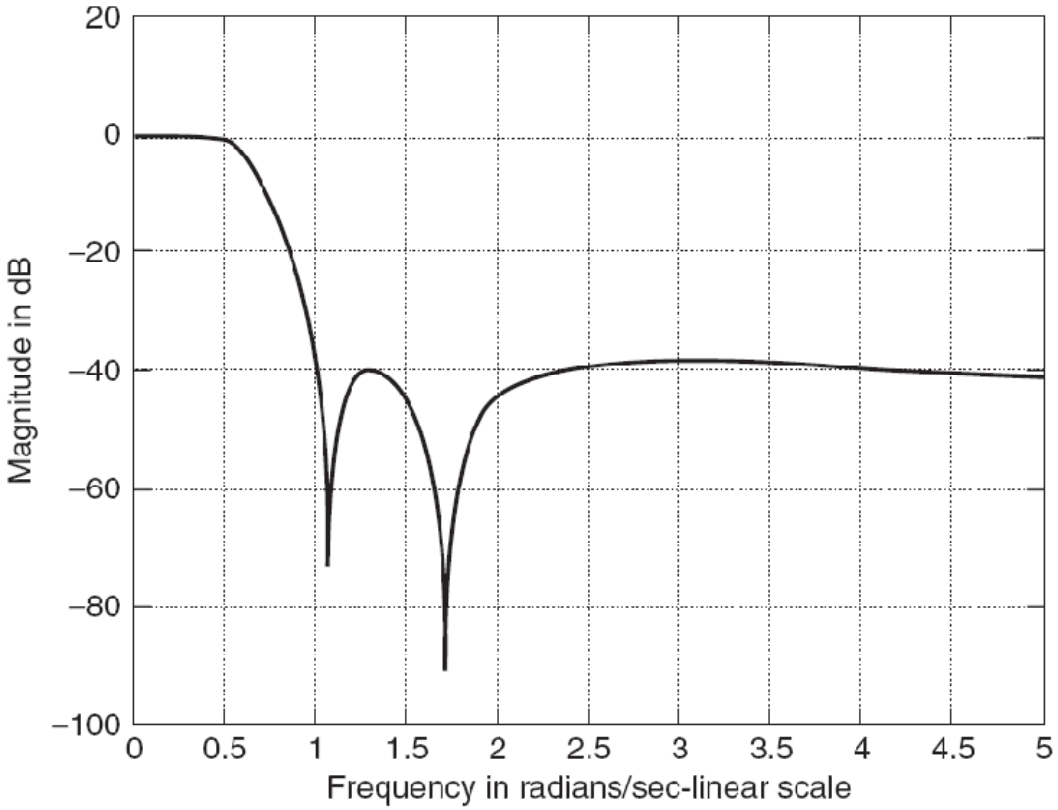

Figure 3-11: Magnitude response of Chebyshev II lowpass filter

3.4 Elliptic Filter Approximations

An elliptic filter (also known as a Cauer filter) is an electronic filter with equalized ripple (equiripple) behavior in both the passband and the stopband. The amount of ripple in each band is independently adjustable, and no other filter of equal order can have a faster transition in gain between the passband and the stopband, for the given values of ripple (whether the ripple is equalized or not). Alternatively, one may give up the ability to independently adjust the passband and stopband ripple, and instead design a filter which is maximally insensitive to component variations. They exhibit an equiripple response in the passband and also in the stopband. The order of the elliptic filter that is required to achieve the given specifications is lower than the order of the Chebyshev filter, and the order of the Chebyshev filter is lower than that of the Butterworth filter. As the ripple in the stopband approaches zero, the filter becomes a type I Chebyshev filter. As the ripple in the passband approaches zero, the filter becomes a type II

Chebyshev filter and finally, as both ripple values approach zero, the filter becomes a Butterworth filter.

The gain of a lowpass elliptic filter as a function of angular frequency ω is given by:

$$G_n(\omega) = \frac{1}{\sqrt{1+\varepsilon^2 R_n{}^2(\xi, \frac{\omega}{\omega_0})}} \tag{3.10}$$

where R_n is the nth-order elliptic rational function (sometimes known as a Chebyshev rational function) and ω_0 is the cutoff frequency :ε is the ripple factor :ξ is the selectivity factor.

An elliptic filter has a system function with both poles and zeros. The magnitude of its frequency response is[5]:

$$\left|H_a(\Omega)\right|^2 = \frac{1}{1+\varepsilon^2 U_N{}^2(\frac{\Omega}{\Omega_p})} \tag{3.11}$$

where $U_{N(\Omega/\Omega_p)}$is a Jacobian elliptic function. The Jacobian elliptic function $U_N(x)$ is a rational function of order N with the following property:

$$U_N(\frac{1}{\Omega}) = \frac{1}{U_N(\Omega)} \tag{3.12}$$

Elliptic filters have an equiripple passband and an equiripple stopband. Because the ripples are distributed uniformly across both bands (unlike the Butterworth and Chebyshev filters, which have a monotonically decreasing passband and/or stopband), these filters are optimum in the sense of having the smallest transition width for a given filter order, cutoff frequency Ωp, and passband and stopband ripples. The frequency response for a 4th-order elliptic filter is shown in Fig. 3-12. [6]

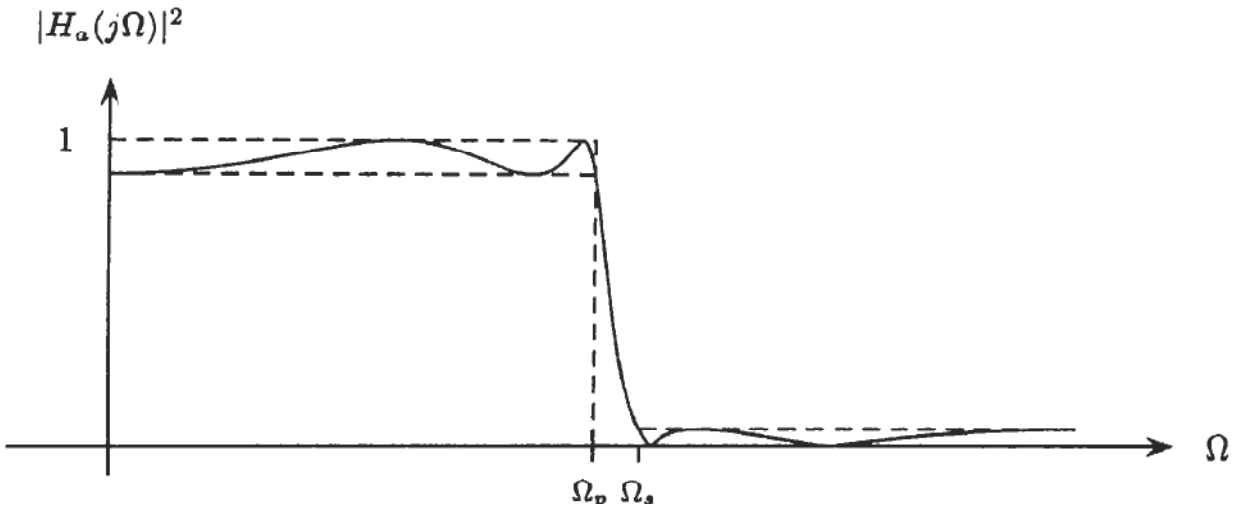

Figure 3-12: The magnitude of the frequency response of a sixth-order elliptic filter

The design of elliptic filters is more difficult than the Butterworth and Chebyshev filters, because their design relies on the use of tables or series expansions. However, the filter order necessary to meet a given set of specifications may be estimated using the formula:

$$N \geq \frac{\log(\frac{16}{d^2})}{\log\left(\frac{1}{q}\right)} \tag{3.13}$$

3.4.1 Advantages of Elliptic filter

The elliptical filter has an extremely sharp cutoff frequency, which makes it ideally suited for filter design cases where there must be severe attenuation in frequencies just entering the stop-band of the filter. Further, because the rippling effect is distributed across both the pass- and stop-bands in the elliptic filter, it makes it an excellent candidate for a low pass filter where the amount of error needs to be minimized on both sides of the cutoff frequency. The Chebyshev filter, with a slower roll-off and an imbalanced ripple, does not offer either of these advantages. Therefore, in cases where there are signals that are very close and must be cut off at exactly or very close to one particular frequency, the elliptic filter is recommended.

CHAPTER 4

System Model

Figure 4.1 gives the block diagram of the Single Carrier Communication system that we have simulated in MATLAB. This chapter will explain the whole model in detail.

Let **a**= [a_1, a_2…a_n] and **b**= [b_1, b_2…b_n] are IIR recursive filter coefficients and **β**= [β_1, β_2…β_n] be the feedback coefficients then the system model to be implemented is shown as follows:

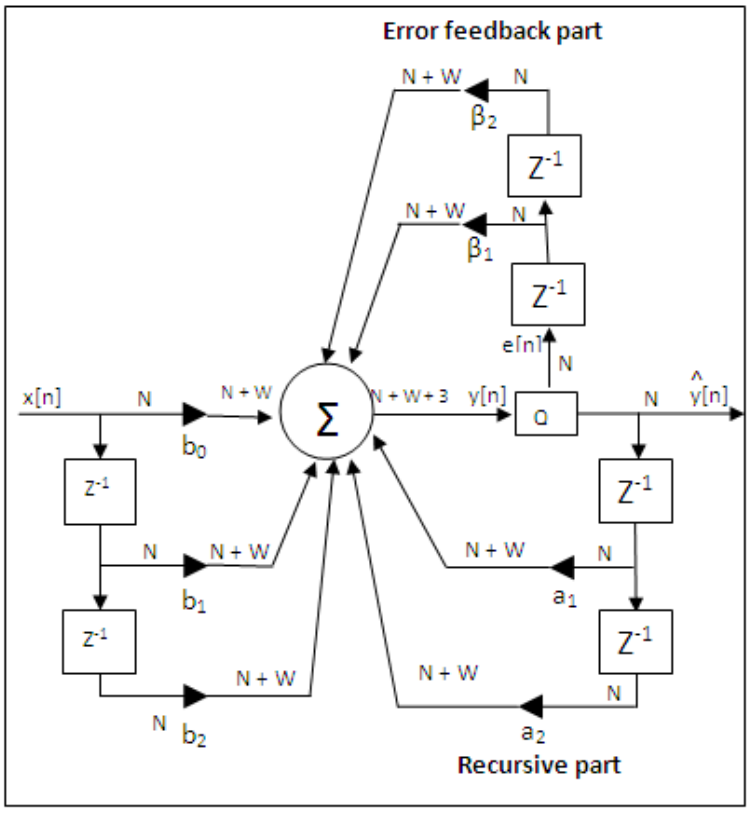

Figure 4-1: The biquadratic section is implemented here using Direct Form1 (DF-1) structure. Original signal coefficients are quantized to N-bits and filter coefficients are quantized to W-bits. Feedback coefficients are also quantized to N-bits. **β** represents feedback coefficients [Ref]

Feedback compensation is implemented by calculating the quantization error at the output of the quantizer "Q". The output of the first iteration and the error are separated. The product of feedback coefficients **β** and the error signal e[n] at the quantizer are fed back to be summed up with the next iteration of the input signal.

4.1 Recursive portion of IIR filter

First the biquadratic portion implementing the feedforward path i.e. the recursive part of the filter is implemented using Direct Form-1 structure. System model implementing the recursive portion of the filter is shown in Figure 4-2.

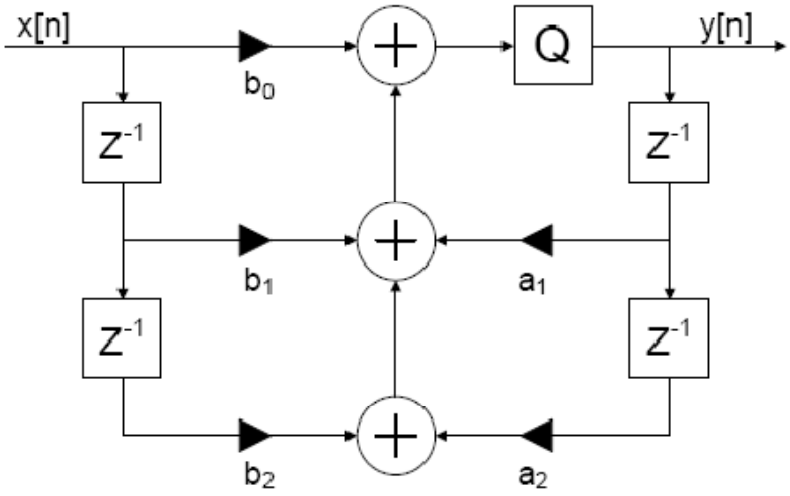

Figure 4-2: Biquadratic section is implemented as Direct Form-1

The macro block of filter implements the function:

$$y[n] = Q\left(\sum_{i=1}^{2} a_i y[n-i] + \sum_{i=0}^{2} b_i x[n-i] \right) \tag{4.1}$$

where Q is the operation of the quantization. A detailed description of the IIR filter structure has already been explained in the first chapter. Although the basic IIR structure implemented in this thesis is Direct Form-1 but it is important to have a look at the basic structures of IIR filter, because any of the proposed structure can be implemented depending upon the project and user requirements. Each structure differs from all other structures on the basis of resource usage like the number of delays and no. of adders and multipliers used to build the logic.

4.2 IIR filters structures

A linear time-invariant discrete-time system is in general represented by a linear constant-coefficient difference equation characterizing the input-output relation of the system. As a network structure, such a difference equation can be represented by a block diagram or a signal flow graph. A signal flow graph is a network of directed branches that connect at nodes. It is equivalent to block diagrams which we are already familiar with, except for a few notational differences. In a signal flow graph, the value carried by a specific branch is equal to the value of its originating node. Nodes in signal flow graphs represent variables. The value carried by a specific node is the sum of all branches coming into it. If there is only one entering branch, the node is a "branching note" rather than a "summing node." Two special types of nodes exist: source nodes have no entering branches, they present external signal sources; sink nodes have only entering branches, they extract output from a graph. From the difference equation representation, it can be seen that the realization of the causal IIR digital filters requires some form of feedback so by convention, normally the delay element has been represented by a branch gain of z^{-1}. The signal flow graph representation of an LTI system is not unique. In fact, for any given rational system function, equivalent sets of difference equations and network structures exist. In practical implementations, factors such as number of multipliers and adders, regularity of the structure, and finite-word-length effects are taken into account when deciding which network structure to use.

The difference equations representing IIR filters can be manipulated to represent the filter using various mathematical operations giving the same result. This gives different realizations of IIR filters when they are coded. Four different realizations are given in this section, the first being the direct form using the standard notation for a recursive difference equation. However, some simple mathematical manipulation gives the canonical form, which uses less storage of previous values. Finally, the denominator of the transfer function for IIR filters is usually factored into first- and second-order factors to reduce the effect of numerical precision on the filter. The IIR filter transfer function can then be written as a product of numerator factors over denominator factors, or the transfer function can be written as a partial fraction expansion as given in any

analog signal processing course or algebra course. Each of these product terms or sum terms can be in direct or canonical form. An *N*-th order IIR digital transfer function is characterized by 2*N*+1 unique coefficients, and in general, requires 2*N*+1 multipliers and2*N* two-input adders for implementation.

There are four different IIR filter structures; each of them is described briefly as follows:

- Direct Form I&II
- Transpose direct form
- Cascade form
- Parallel form

4.2.1 Direct form I & II

These filter structures are one in which the multiplier coefficients are precisely the coefficients of the transfer function and they are very easy to implement structures that's why we've chose this filter structuring technique in this thesis.

Consider the following general form of a difference equation and the corresponding system transfer function:

$$y[n] = \sum_{k=0}^{M} b_k x[n-k] + \sum_{k=1}^{N} a_k y[n-k] \Rightarrow H(z) = \frac{\sum_{k=0}^{M} b_k z^{-1}}{1 - \sum_{k=1}^{N} a_k z^{-1}} \quad \textbf{(4.2)}$$

As the name Direct Form-I suggests, the difference equation (4.2) is implemented as given using delays, multipliers, and adders. For the purpose of illustration, let M = N = 3. Then the difference equation (4.2) can be re-written as:

$$y(n) = b_0x(n)+b_1x(n-1)+ b_2x(n-2)+ b_3x(n-3)-a_1y(n-1) -a_2y(n-2) -a_3y(n-3)$$

which can be implemented in Figure 4-3. The direct form I structure implements each part of the rational function H(z) separately with a cascade connection between them. The numerator part is a tapped delay line followed by the denominator part, which is a feedback tapped delay line. Thus there are two separate delay lines in this structure, and hence it requires eight delay elements. We can reduce this delay element count or eliminate one delay line by interchanging the order in which the two parts are connected in the cascade.

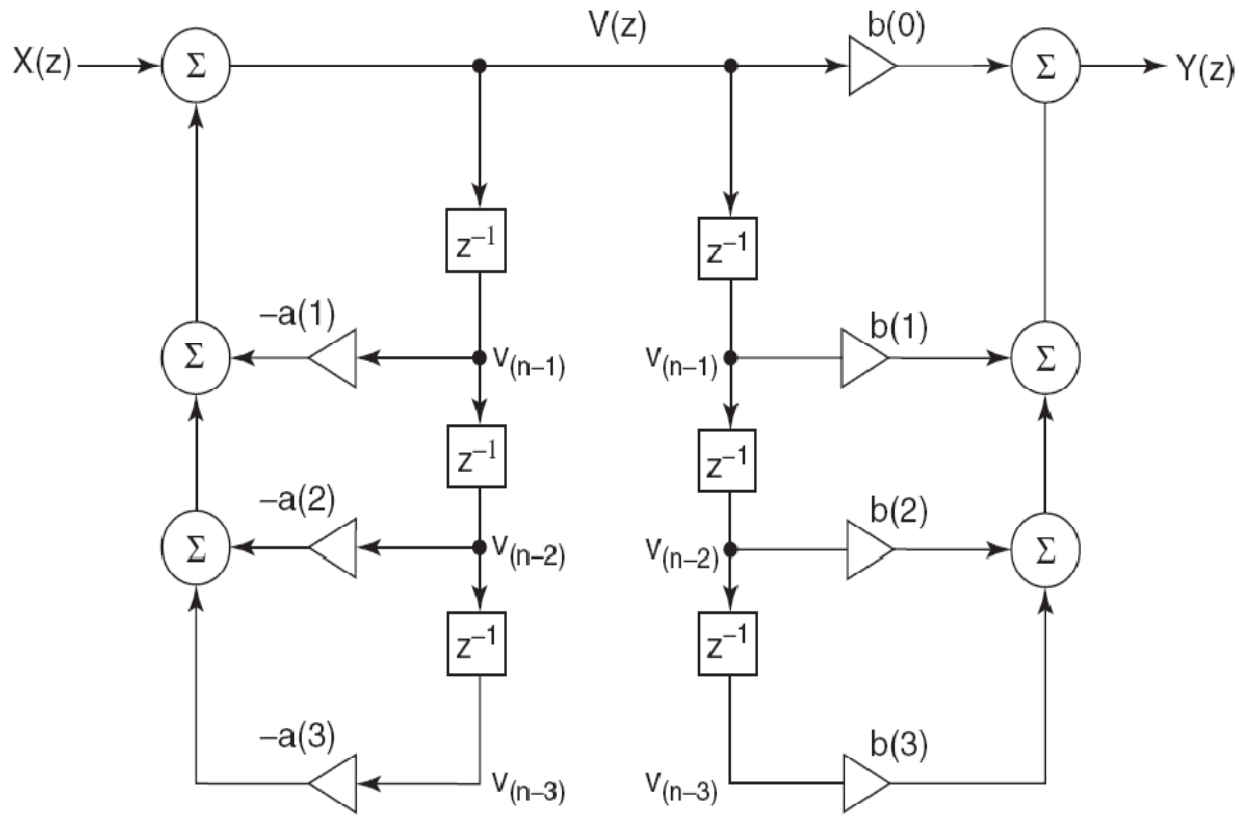

Figure 4-3: Direct form I of an IIR filter

To elaborate Direct Form-I, equation (4.2) can be re-written as:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-1}}{1 - \sum_{k=1}^{N} a_k z^{-1}}$$

$$= H_1(z)H_2(z) = \left[\frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}}\right]\left[\sum_{k=0}^{M} b_k z^{-k}\right] \qquad \textbf{(4.3)}$$

The realization of H1(z) in direct form I is shown in Figure 4-3 as the filter connected in cascade with the realization of the FIR filter H2(z) also in direct form I structure. The structure for the IIR filter is also called a direct form because the gain constants of the multipliers are directly available from the coefficients of the transfer function. We note that $H_1(z)$ = V (z)/X(z) and $H_2(z)$ = Y(z)/V(z). We also note that the signals at the output of the three delay elements of the filter for $H_1(z)$ are the same as those at the output of the three delay elements of filter $H_2(z)$. Hence we let the two circuits share one set of three delay elements, thereby reducing the number of delay elements. The result of merging the two circuits is shown in Figure 4-4 and is identified as the direct form II realization of the IIR filter[1].

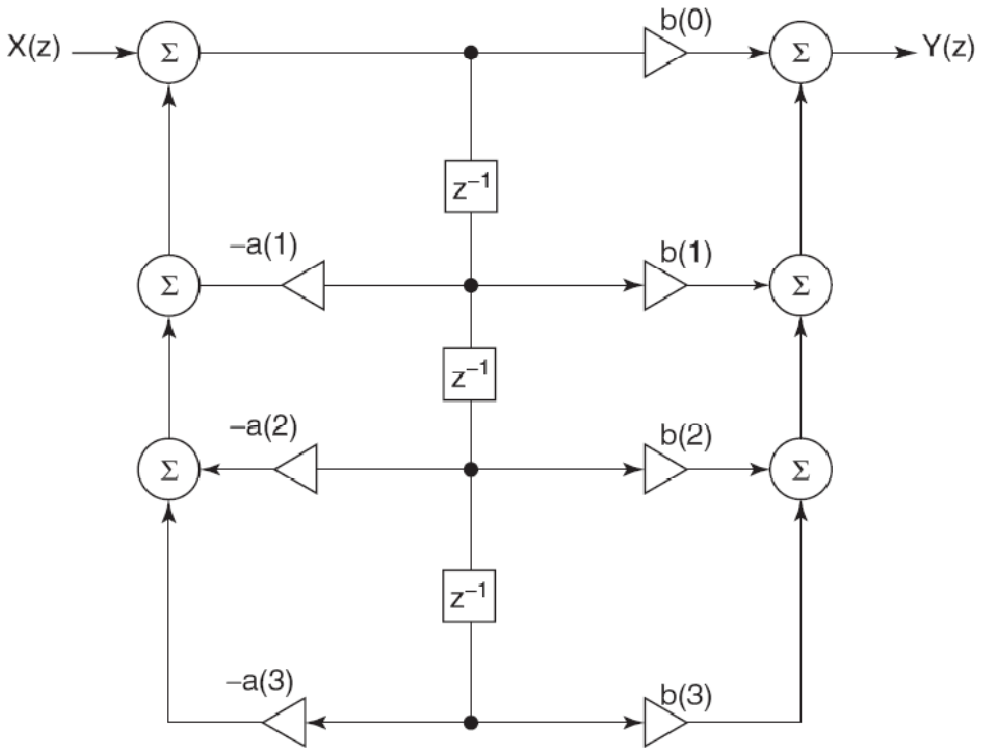

Figure 4-4: Direct form II of an IIR filter

4.2.2 Transpose Direct Form

Using signal flow graphs, we can transform a given system into a different network structure while maintaining the same system function. One such transformation is transposition.

4.2.2.1 Transposition Theorem

There are two basic rules of Transposition theorem which are as follows:

1. Reverse direction of all branches
2. Interchange input and output

For single-input single-out systems, interchanging the input and output nodes after reversing the flow graph gives the same transfer function as the original system. Although the transfer functions remain the same, different network structures represent different algorithms, which are equivalent under ideal infinite precision arithmetic. With finite precision arithmetic, the implementation structure determines internally generated noise which affects the overall system behavior[4][6].

The transposed form of the circuit shown in Figure 4-4 is shown in Figure given below:

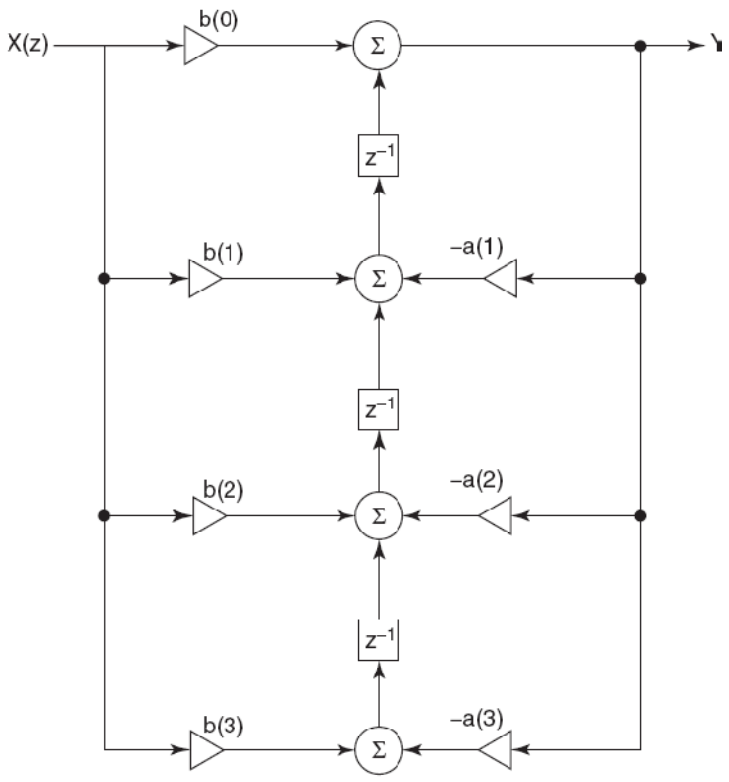

Figure 4-5: Direct form II transposed structure of an IIR filter

Both of them i.e. the Direct Forms and their Transpose use the minimum number of delay elements equal to the order of the IIR filter and hence are canonic realizations. The two filters realizing $H_1(z)$ and $H_2(z)$ can be cascaded in the reverse order [i.e., $H(z) = H_2(z)H_1(z)$], and when their transpose is obtained, we see that the three delay elements of $H_2(z)$ can be shared with $H_1(z)$, and thus another realization identified as direct form I as well as its transpose can be obtained.

4.2.3 Cascade Form

As previously shown, the IIR filter can be realized or coded in direct form or canonical form. This is taken even further by factoring the numerator and denominator of the transfer function of the IIR filter into first and second-order factors. Then it could be written as a product of terms with first- and second-order numerators and denominators. This is usually done, since it contributes to numerical stability. However if several poles are multiplied together to form the

denominator, numerical truncation or rounding of the coefficients of the resulting polynomial in z could produce a pole magnitude greater than one which can introduce instability in a circuit. The terms are usually then written in the canonical first- or second-order form. These terms are then each converted to difference equations using the shifting property, and each difference equation is then coded.

The filter function (4.2) can be decomposed as the product of transfer functions in the form:

$$H(z) = \frac{N_1(z)N_2(z)...N_K(z)}{D_1(z)D_2(z)D_3(z)...D_K(z)} \tag{4.4}$$

$$= \left[\frac{N_1(z)}{D_1(z)}\right]\left[\frac{N_2(z)}{D_2(z)}\right]\left[\frac{N_3(z)}{D_3(z)}\right]...\left[\frac{N_K(z)}{D_K(z)}\right] \tag{4.5}$$

$$= H_1(z)H_2(z)H_3(z)...H_K(z) \tag{4.6}$$

where K = N/2 when N is even and the polynomials $D_1(z)$, $D_2(z)$, $D_3(z)$, and so on are second order polynomials, with complex zeros appearing in conjugate pairs in any such polynomial. When N is odd, K = (N − 1)/2, and one of the denominator polynomials is a first-order polynomial. The numerator polynomials $N_1(z)$,$N_2(z)$, . . . may be first-order or second-order polynomials or a constant:

$$H(z) = H_0\left[\frac{1+\beta_{11}z^{-1}}{1+\alpha_{11}z^{-1}}\right]\prod_k\left[\frac{1+\beta_{1k}z^{-1}+\beta_{2k}z^{-2}}{1+\alpha_{1k}z^{-1}+\alpha_{2k}z^{-2}}\right] \tag{4.7}$$

Each of the transfer functions $H_1(z)$,$H_2(z)$, . . . , $H_K(z)$ is realized by the direct form I or direct form II or their transpose structures and then connected in cascade. They can also be cascaded in many other sequential order, for example, H(z) = $H_1(z)H_3(z)H_5(z)$ or H(z) = $H_2(z)H_1(z)H_{4(}z)H_3(z)$ . . . .

There are more choices in the realization of H(z) in the cascade connection in addition to those indicated above. We can pair the numerators $N_1(z)$,$N_2(z)$, . . . and denominators

$D_1(z), D_2(z), D_3(z)$, . . . in many different combinations; in other words, we can pair the poles and zeros of the polynomials in different ways. So the number of realizations that can be obtained from a nominal IIR transfer function is very large, in general[1]. Besides the difference in the algorithms for each of these realizations and the consequent effects of finite wordlength when the coefficients of the filter and the signal samples are quantized to a finite number, we have to consider the effect on the overall magnitude of the output sequence and the need for scaling the magnitude of the output sequence at each stage of the cascade connection and so on.

4.2.4 Parallel Form of an IIR filter

The IIR transfer function can also be expanded as the sum of second-order structures. It is decomposed into its partial fraction form, combining the terms with complex conjugate poles together such that we have an expansion with real coefficients only[6].

There are many different ways to develop a parallel IIR structure. An alternative way to factoring H(z) is to expand the system function using a partial fraction expansion. For example, with

$$H(z) = \frac{\sum_{k=0}^{q} b(k) z^{-k}}{1 + \sum_{k=1}^{p} a(k) z^{-k}} = A \frac{\prod_{k=1}^{q} 1 - \beta_k z^{-1}}{\prod_{k=1}^{p} 1 - \alpha_k z^{-1}} \tag{4.8}$$

if $p > q$ and $a_i < a_k$ (the roots of the denominator polynomial are distinct), H (z) may be expanded as a sum of p first-order factors as follows:

$$H(z) = \sum_{k=1}^{p} \frac{A_k}{1 - \alpha_k z^{-1}} \tag{4.9}$$

where the coefficients A_k and α_k are, in general, complex. This expansion corresponds to a sum of p first-order system functions and may be realized by connecting these systems in parallel. If h(n) is real, the poles of H(z) will occur in complex conjugate pairs, and these complex roots in the partial fraction expansion may be combined to form second-order systems with real coefficients:

$$H(z) = \sum_{k=1}^{N} \frac{\gamma_{0k} + \gamma_{1k} z^{-1}}{1 + \alpha_{1k} z^{-1} + \alpha_{2k} z^{-2}} \qquad \textbf{(4.10)}$$

The resultant sixth order parallel structure is shown in Fig 4-6

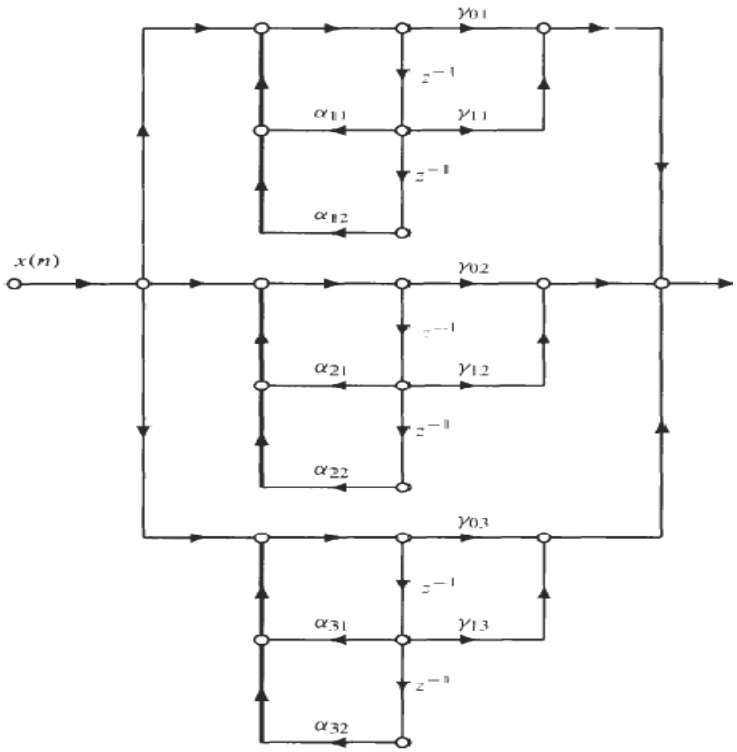

Figure 4-6: A sixth-order IIR filter implemented as a parallel connection of three second-order direct form II structures.

4.2.5: IIR Filter Structure Comparison

All the structures that have been defined above have their own pros& cons. We have chosen Direct form- I form to build a feedforward path as well as compensated part i.e. feedback path because of two main reasons.

1. Direct Form-I structure is very easy to build than most of the other IIR structures.
2. It is very easy to build any other IIR filter structure from Direct Form-I structure by merging of different resources like delays, multiplies and adders etc. In other words Direct Form-I is highly compatible with all of the other IIR structural techniques.

However, a comparison can be made between all of the above mentioned IIR filter structuring techniques by using a fourth order IIR filter. All the techniques are used one by one to build the IIR structural logic. In addition to the above mentioned techniques some more techniques have also been introduced to build IIR structure [7]. The results showing the number of resources are shown in table given below:

Structures	Number of multiplications	Number of additions and subtractions	Total number of operations	Required bandwidth
Direct Form	16	16	40	21
Cascade	13	16	34	12
Parallel	18	16	39	11
Transpose	18	16	35	23
Ladder	17	32	50	14
Wave Digital	11	30	47	12

Table 4-1: Comparison of complexity of different IIR filters

CHAPTER 5

Implementation

After the calculations of feedforward path i.e. the recursive portion of the filter, next step is that the output of the filter y[n]is then fed to the input of the quantizer for effective quantization of the filter coefficients.

5.1 Quantization of filter coefficients

Numbers are represented as either fixed-point or floating-point data types. A fixed-point data type is characterized by the word size in bits, the binary point, and whether it is signed or unsigned. The binary point position defines a scaling of the fixed-point values. Floating-point data types are characterized by a sign bit, a fraction (or mantissa) field, and an exponent field.
We choose the (signed) fixed-point representation since it often offers advantages in terms of power consumption, size, memory usage, speed, and cost of the final product, compared to the floating-point representation.

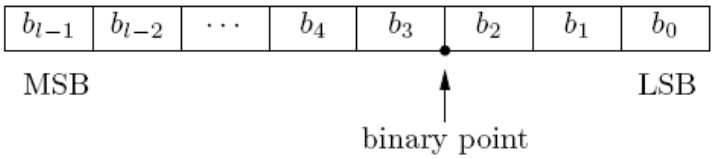

Figure 5-1: Binary fixed point number representation

A common representation of a binary fixed-point number (either signed or unsigned) is shown in Figure 5-1, which illustrates the following points.

- b_l are the binary digits (bits: 0 or 1)
- The size of the word in bits is given by "l"
- The binary point position defines a scaling for the fixed-point numbers. In our example, the binary point position is chosen the position left of b_2.

A real world value V can then be approximated by a quantized number $\tilde{V}$ as follows:

$$V \approx \tilde{V} = S.Q \qquad \textbf{(5.1)}$$

which shows following points in detail

- $S = 2^E$ the slope, and E an (integer) fixed power-of-two exponent;
- $Q = -b_{l-1}2^{l-1} + \sum_{i=0}^{l-2} b_i .2^i$ the quantization

The range of a number gives the limits of the representation while the precision gives the distance between successive numbers in the representation. The range and precision of a fixed-point number depends on the length of the word and the scaling. For a fixed word size l the range of the filter coefficients will determine the slope S (in order to avoid underflow or overflow), and as a consequence the exponent E. The approximation ~v of the real-world value V can then be found by rounding, truncation or magnitude truncation as shown in figure 5-2.

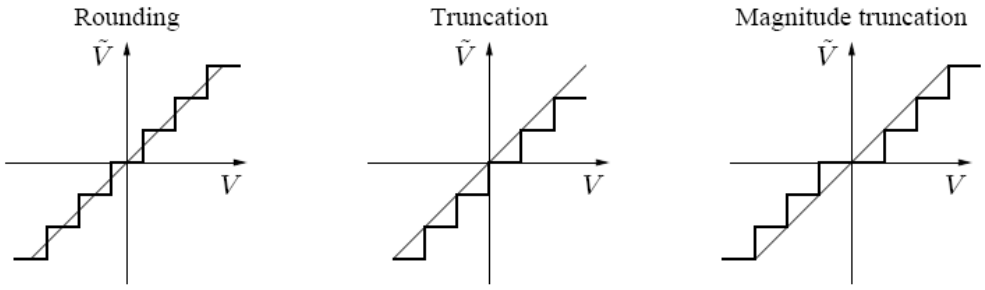

Figure 5-2: Rounding and Truncation

Feedback compensation is implemented by calculating the quantization error at the output of the quantizer "Q". The output of the first iteration and the error are separated. The product of feedback coefficients β and the error signal e[n] at the quantizer are fed back to be summed up with the next iteration of the input signal [8].

Let us consider the 2nd order error feedback quantizer mentioned in Figure 4-1. The main relation that we can deduct between full precision output y[n] i.e. non-quantized output and filter output y[n] can be defined by the quantizer operation Q that is:

$$\hat{y}[n] = Q\{y[n]\} \tag{5.2}$$

$$\hat{y}[n] + e[n] = y[n] \tag{5.3}$$

Equation (5.3) specifies that when full precision output is quantized, error signal is generated which is denoted by e[n]. The error part is amplified or rectified by the values of coefficients in the recursive portion of the filter. The products of filter coefficients a and input signal x[n] are add up with the product of filter output y[n] and filter coefficient b to produce full precision output y[n].

The error feedback is implemented by modifying the quantizer in the filter structure. In a fixed-point implementation, the quantization is usually performed by discarding the lower bits of the double-precision accumulator (two's complement truncation), and thus the quantization error equals this residue left in the lower part. The error is fed back through a simple FIR filter, as shown in Figure 4-1.

5.2 Implementation of Compensation Logic

After the quantization of filter coefficients, error signal e[n] is generated at the output of the quantizer which is then multiplied with the feedback coefficients to obtain the compensation logic of our proposed system model, in other words error feedback loop is thus generated. The difference equation with the EF path is given below:

$$\hat{y}[n] - \left\{ Q\sum_{i=0}^{2} b_i . x[n-i] + \sum_{i=1}^{2} a_i . y[n-i] \right\} + \sum_{i=1}^{2} \beta_i . e[n-i] \tag{5.4}$$

Some basically different ways to formulate and apply the EF can be distinguished in the existing literature. We divide the proposed EF schemes into four categories according to how the parameters of the EF quantizer, the structure, the coefficients, and the order, are determined:

5.2.1 At Unit Circle

One technique is to calculate the feedback coefficients like placing the zeros at unit circle i.e. by taking value of coefficients at points z=1 and z= -1. By analyzing this property we can say that this technique is useful for narrowband low pass and high pass filters. Their hardware implementation was studied in earlier works [9], [10].

5.2.2 Predetermined Values

In this technique all the values i.e. the order and structure of the error feedback quantizer as well its type is determined by means of recursive portion of the proposed IIR filter. The techniques simply emphasizes on obtaining the feedback coefficients by placing the zeros of the compensated path over the denominator poles. The structure and formation of the compensated (EF) portion of the filter is true replica of the recursive portion of the filter. In some cases even non-recursive portion of the filter is also considered while calculating the error feedback path which is also called error feed forward [11].

By feeding the values of the coefficients from recursive portion of the filter, feedback coefficients can be obtained. An easy and more comprehensive way to obtain filter feedback coefficient values is by setting them equal to the values of denominator poles which are already determined during recursive portion of the filter i.e. **β=a** means by setting the values of the feedback coefficients exactly equal to denominator poles.

5.2.3 Modified Quantization

This particular technique emphasizes the use of general and isolated error feedback quantizer from the rest of recursive feedback portion. Limitation of this method is that it is only useful while handling fewer quantization bits. As soon as quantization bits starts to grow up in number the complexity in calculation and excessive use of resources makes it very hard to handle.

Minimization of quantization noise power is basic issue in modified quantization that needs to be taken care of. Optimization of coefficients is done in this method by quantization of noise power at filter output using least mean square criterion [11], [12], [13] and [14].

Lot of Algorithms have been devised so far proposing formulas to calculate optimal error feedback coefficients using modified quantization e.g. method proposed by Higgins & Munson [9] use numerical integration to calculate optimal error feedback coefficients.

5.2.4 Symmetric Method

As described earlier the lower order optimal error feedback method is easier to handle i.e. when quantization bits are lesser in number.

A simple method to reduce cost while using higher order filter i.e. using more quantization bits is by constraining the error feedback coefficients to have symmetric or asymmetric values [15]. This method is mostly proposed and implemented in structures where coefficient symmetry is devised and thus results in noise reduction with minimum loses as compared to optimal solution.

The quantization error always behaves like width extension which leads to lower noise levels and better area utilization. The closer the poles are to the unit circle, lesser will be the noise at the filter output and better filter properties will be attained. To minimize quantization error, coefficients β are chosen precisely so that zeros of error feedback path approximately compensate the poles of the transfer function denominator. Placement of zeros of EF path can be calculated using any one of above mentioned techniques.

5.3 Comparison of Compensation Logic with Non Compensation Logic

Implementation of compensation first requires choosing the length of feedback coefficients. The technique becomes much more effective if the coefficient width in bits is chosen the same as the width of filter coefficients i.e. β1 and β2 have same bit widths as that of a1 and a2. Increasing or decreasing the bits of feedback coefficients will result in a less optimal solution. If the coefficient bit width of feedback coefficient β is greater than filter coefficient a then the occupied logic will increase requiring a larger multiplier to do the calculations which is not an optimal solution.

Our main system model has already been described in Figure 4-1. To implement the system logic we have to first design an Elliptic Low Pass filter using Filter Designing Toolbox (FDA tool) in Matlab. We can give any specifications to the toolbox based on the needs of the user. After the

implementation of the filter next the quantization is applied on the coefficient values of the filter. At the output of the quantizer an error signal e[n] is generated along with the output. This error signal is then multiplied with the feedback coefficient values and fed back again to be summed up eventually with the next incoming input. The values of feedback coefficients can be obtained using any one of the above mentioned techniques. Both the structures i.e. with error feedback (WEF) and without error feedback (WOEF) are tested using MATLAB based on different levels of quantization to elaborate the results more clearly and for better comparison. In this thesis we have used five different levels of Quantization and then compared the results of both the standards to check their efficiency in terms of reduction of error. The graphical and numerical results obtained will be shown later in this report.

5.4 Hardware Implementation

After the implementation of system model in Matlab, the logic is then implemented on Hardware using Xilinx AccelDSP 10.1 tool so that a better comparison can be made between the two standards based on usage of resources like Flip Flops (FF's), Slice Registers, Lookup tables (LUT's) etc. Before going any further into the hardware implementation of the system model first we briefly have a look at the AccelDSP tool.

5.4.1 AccelDSP 10.1

The AccelDSP™ Synthesis Tool is a product that allows you to transform a MATLAB floating point design into a hardware module that can be implemented in a Xilinx FPGA .This section describes briefly about the AccelDSP coding style guidelines for the MATLAB floating point model. In order to synthesize the basic structure of a MATLAB design using AccelDSP™ Synthesis Tool, the design must be represented by a minimum of two M-files, a script M-file and a function M-file. These two basic files may also reference other related function files.

The two main elements of the script file are (1) the streaming loop and (2) the top-level design function call. Other constructs may be intermixed with these basic elements, but are not recognized by the AccelDSP Synthesis Tool for hardware synthesis. These secondary constructs are typically used for design verification.

5.4.1.1 Streaming Loop

For hardware synthesis, the infinite streams of data entering and leaving the design must be partitioned into manageable groups or "slices" in order to properly process the data. A streaming loop (a for loop or while loop) is the construct that performs this task. A MATLAB script file must have a streaming loop.

5.4.1.2 The top-level design function call

The top-level design function call represents the hardware to be synthesized. This function must reside inside the streaming loop. During the pre-analyze phase, the AccelDSP Synthesis Tool attempts to automatically identify the top-level design function. If it cannot, the tool prompts you to manually identify the design function call in the AccelDSP Project Explorer window.

AccelDSP generally follows some steps before synthesizing the results .The results are shown as follows:

- Reads and analyzes a MATLAB floating-point design
- Automatically creates an equivalent MATLAB fixed-point design
- Invokes a MATLAB simulation to verify the fixed-point design
- Provides you with the power to quickly explore design trade-offs of algorithms that are optimized for the target FPGA architectures
- Creates a synthesizable RTL HDL model and a Testbench to ensure bit-true, cycle accurate design verification
- Provides scripts that invoke and control down-stream tools such as HDL simulators, RTL logic synthesizers, and Xilinx ISE implementation tools.

The AccelDSP Synthesis Tool provides a direct link to MATLAB so you don't have to leave the tool to run a MATLAB simulation. If you have already verified the floating point model, you may skip this step. After the verification and analyzing of floating point model the next step is to generate fixed point process. During the Generate Fixed Point process, the streaming loop and the top-level design function are identified. Default assumptions are made about design objects in your design like variables and loops. If need be, you can change the properties before

generating the RTL model. Also note that you can view and change the quantization parameters on variables from the Generate Fixed Point.

Next step is to verify the Fixed point Model. Click on the Verify Fixed Point icon to start the MATLAB verification process. MATLAB is automatically invoked on the fixed-point model. It may take a few moments to invoke MATLAB and run the simulation. The fixed point plot is displayed when the simulation is finished.

The fixed point of a simple IIR filter is shown below on the right.

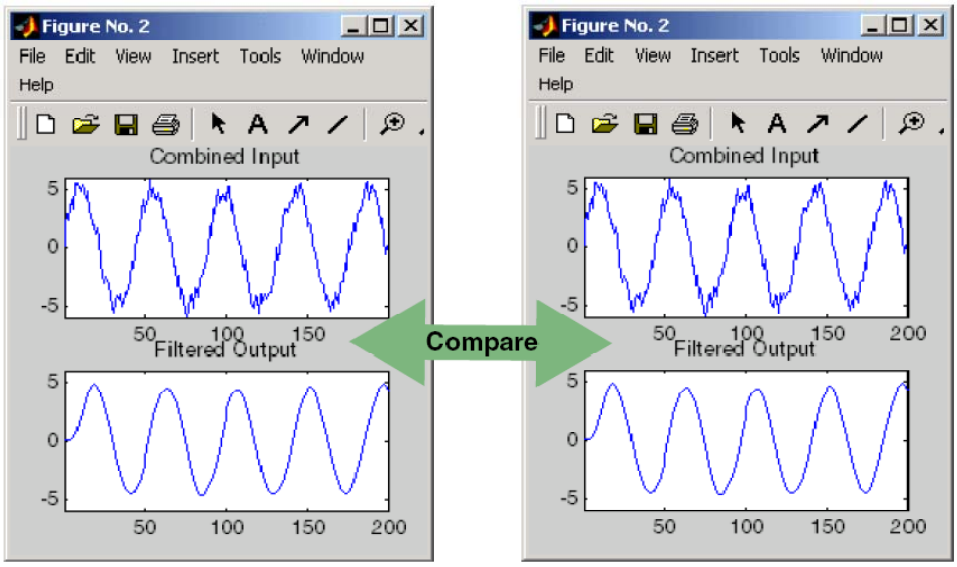

Figure 5-3: Verification of fixed point model

Visually compare the fixed-point plot with the floating-point plot. If you are satisfied with the results, you are ready to proceed to synthesis. If the fixed-point plot deviates too much from the floating-point plot, you may need to adjust the quantizer properties on some design variables, and then repeat the verification process. When you are satisfied with the fixed-point verification results, you are ready to generate the RTL design.

Next step is to generate an RTL design for your proposed system model. The results are summarized in the generated report. The HDL module that is generated (not synthesized) by the AccelDSP Synthesis Tool when the MATLAB design function is synthesized to an RTL model.

When you execute the Verify RTL step in the AccelDSP flow, an HDL simulator runs a simulation on the Testbench which, in turn, applies stimulus to the inputs of the RTL model and compares the result to a known golden data. VHDL or Verilog folder can also be expanded to view the generated HDL files and the associated Testbench files. After the successful generation of the RTL design, next step is to verify that particular RTL design. The AccelDSP Synthesis Tool causes the simulation tool to compile the Testbench and simulate the RTL model. When the simulation is finished, a 'PASSED' message is shown in the transcript window.

Last step is to synthesize the RTL model. Once you verify the RTL Model, you are ready to transform the model into a gate-level netlist. This is normally done by invoking your RTL Synthesis tool on the RTL model and synthesized the design to a gate-level EDIF netlist.

5.5 Results

After the implementation of system model mentioned in Figure 4-1 in MATLAB, we then implemented it in hardware using ACCELDSP tool. Main purpose of doing this is to compare the results of both the standards i.e. IIR filter with error feedback and IIR filter without error feedback. The results obtained from MATLAB and from a synthesizable report generated via ACCELDSP are tabulated in table 5-1:

Quantization bits	**16-bits**		**18-bits**		**20-bits**		**24-bits**	
WEF/WOEF	WEF	WOEF	WEF	WOEF	WEF	WOEF	WEF	WOEF
ERROR	61	79	47	54	39	50	36	54
Quantized Error	$9*10^{-4}$	$1.2*10^{-3}$	$2.0*10^{-4}$	$1.8*10^{-4}$	$2.99*10^{-6}$	$2.37*10^{-6}$	$1.64*10^{-7}$	$1.35*10^{-7}$
Slice Registers	390	273	398	285	410	299	428	314
LUT FF's	170	57	176	65	181	71	187	76
LUT FF's with unused LUT	231	125	238	131	247	139	256	153
LUT FF's pairs	159	148	159	149	160	149	162	152
Clock Frequency [MHZ]	63.6	76.6	63.6	76.8	64.7	76.8	65.8	77.1

Table 5-1: Comparison between WOEF and WEF techniques based on different parameters

The results are obtained using different levels of quantization so that better comparison can be concluded from the result. A number of values have been calculated via Xilinx ACCELDSP 10.1 tool using 16, 18, 20 and 24 bits of quantization. In first case i.e. in 16-bits case both the signal sample width "N" and the filter coefficient width "W" is 16-bits. FF and LUT are used to build the logic. Clock frequency specifies the maximum system frequency determined by Xilinx timing Analyzer.

Comparing the error values of IIR-with error feedback(WEF) macros with the IIR-without error feedback(WOEF) macros from the table mentioned above we can conclude that by adding an extra compensation logic with an already built recursive IIR logic we can reduce the quantization error at the output after few number of iterations. Thus in comparison with the simple IIR-WOEF the compensated logic in addition with recursive logic i.e. IIR-WEF provides a more optimum and less error prone solution.

We can also compare the results of both the techniques with the help of graphs made in MATLAB as shown below.

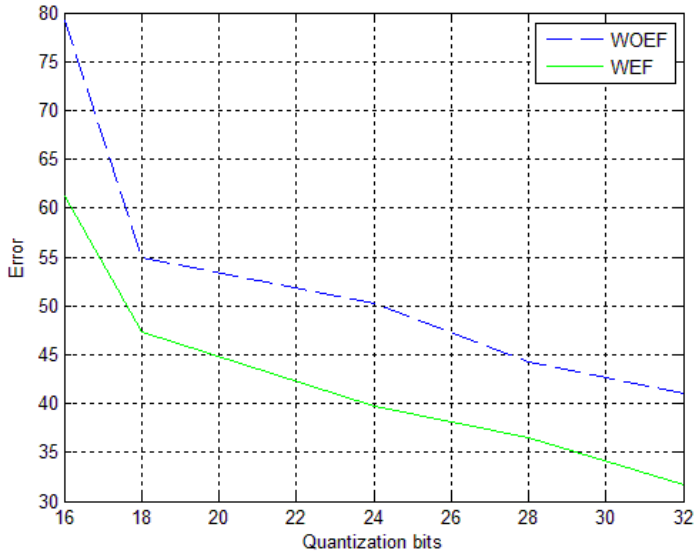

Figure 5-4: Error values using both techniques i.e. WEF and WOEF using different quantization bits. Blue& dotted specifies WOEF, Green& simple specifies WEF.

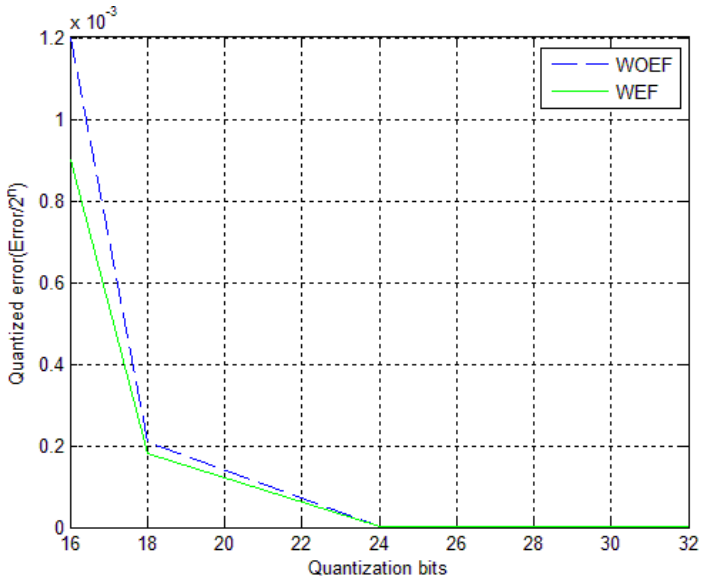

Figure 5-5: Quantized Error values using both techniques i.e. WEF and WOEF using different quantization bits. Blue& dotted specifies WOEF, Green& simple specifies WEF.

Graphically we can see that the IIR filters with embedded compensation logic with them are much less error prone than the IIR filters without compensation logic. Quantized error is normalized error/2^n where "n" specifies number of quantization bits.

Now we can also deduce a well defined comparison from the results obtained from ACCELDSP tool graphically using Matlab graphs. The graphs are shown below:

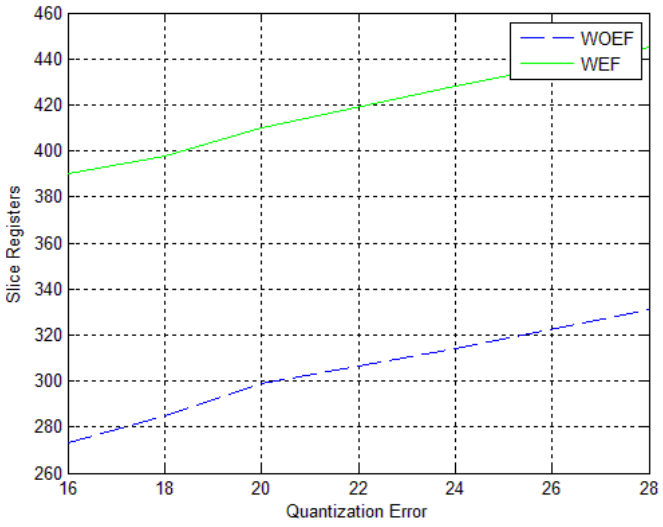

Figure 5-6: Slice registers usage in both techniques i.e. WEF and WOEF using different quantization bits. Blue& dotted specifies WOEF, Green& simple specifies WEF.

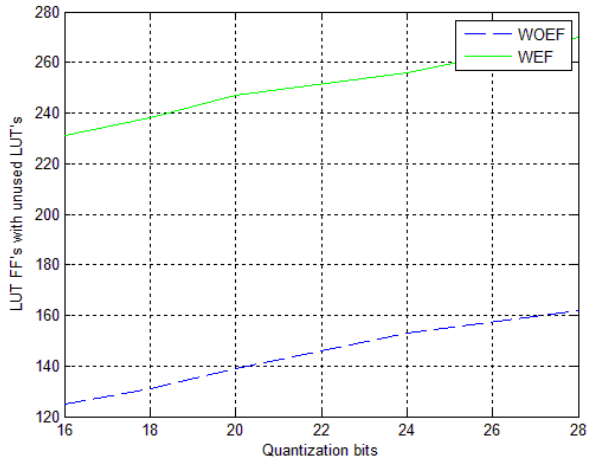

Figure 5-7: LUT FF's with unused LUT's used in both techniques i.e. WEF and WOEF using different quantization bits. Blue& dotted specifies WOEF, Green& simple specifies WEF.

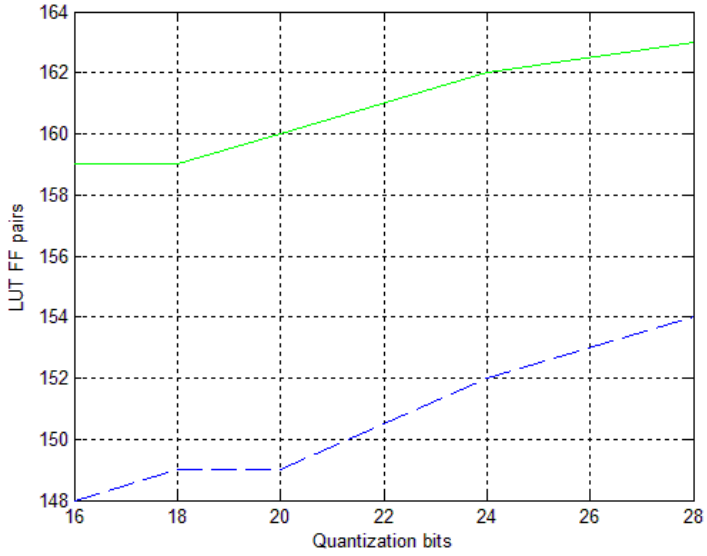

Figure 5-8: LUT FF pairs used in both techniques i.e. WEF and WOEF using different quantization bits. Blue& dotted specifies WOEF, Green& simple specifies WEF.

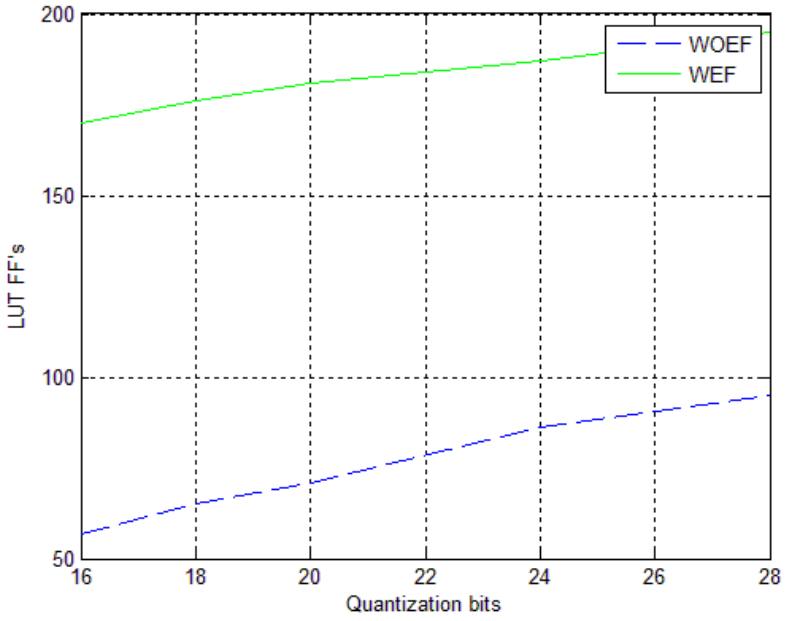

Figure 5-9: LUT FF's used in both techniques i.e. WEF and WOEF using different quantization bits. Blue& dotted specifies WOEF, Green& simple specifies WEF.

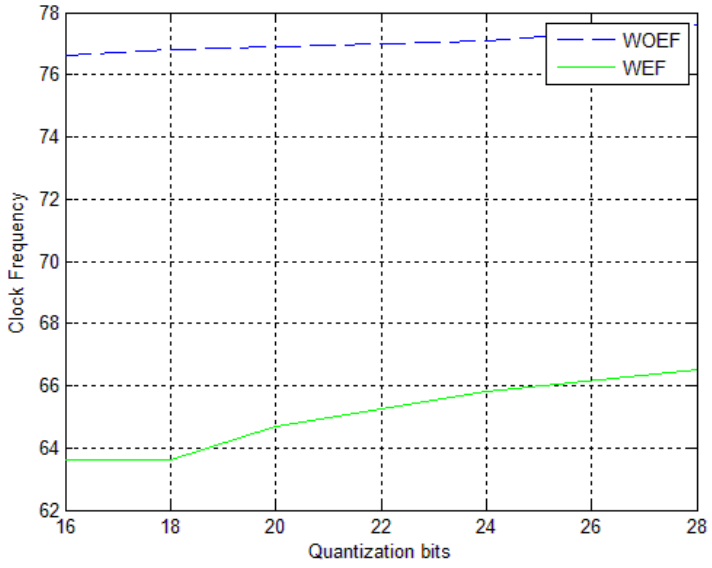

Figure 5-10: Clock Frequency in both techniques i.e. WEF and WOEF using different quantization bits. Blue& dotted specifies WOEF, Green& simple specifies WEF.

A comparison has been made between IIR-WEF macros and IIR-WEF macros based on the usage of Slice registers, LUT flip flops pairs with unused flip flops and unused LUT. Using the numerical and graphical results obtained in this paper we can easily deduced that IIR-WEF occupies more silicon area because it uses more resources to build a less error prone logic. It takes more resources in the form of Slice Registers, LUT and FF used to build noise compensation logic as can be seen from the graphs mentioned above.

From the graphical and numerical results it can be deduced that while IIR with EF technique minimizes error at the output of the quantizer when compare to IIR- WOEF technique, the amount of silicon area occupied by IIR macros with EF is more than the area occupied by IIR macros WOEF. However, in IIR-WEF it is possible to reach lower roundoff noise level to compensate the above mentioned drawback [11]. The idea is to minimize the logic area by reducing the IIR-WEF filter coefficient width until the same roundoff noise level is achieved as WOEF.

CHAPTER 6

Design Improvements

As discussed earlier that IIR-WEF technique is much less error prone as compared to IIR-WOEF technique, but it costs more area (LUT's and FF's) to build the desired logic. However, in IIR-WEF it is possible to reach lower roundoff noise level to compensate the above mentioned drawback. If the area of concern is the number of resources used, then in IIR-WEF technique the area can be reduced by simply reaching lower roundoff noise level i.e. in other words reduce the number of quantization bits. However, it can be seen in Table 5-1 that the difference of quantization bits in order to occupy same area in terms of LUT's and FF's is quite large. For example IIR-WEF occupies same area to build our proposed model using 18 quantization bits that IIR-WOEF technique takes 32 quantization bits to build. Quantization bits as explained earlier are a measure of precision and accuracy. If we take our proposed IIR-WEF model to lower roundoff noise level by reducing the number of quantization bits, the effect will be somehow negative in terms of precision and accuracy. To compensate for this, architecture can be proposed which is used to build IIR-WEF model using same number of quantization bits and area as compared to IIR-WOEF model, yet it is much less error prone than the IIR-WOEF model.

6.1 Timeshared Architecture

The ration of the sampling frequency to circuit clock frequency dictates major decisions in the design of a digital system. In most of signal processing applications, the circuit clock frequency is greater than twice the sample accusation frequency. If such an application is mapped on a dedicated fully parallel architecture, it will obviously not utilize the HW in every clock cycle. As a consequence of the circuit clock running at higher rate than the sampling clock, the HW components will only execute computation at sampling rate. A logical design decision should be to use only required number of HW computational resources and share them for multiple computations of the algorithm in different clock cycles. HW functional blocks shared among

various computation increases HW utilization. The architecture where one functional block is reuse in time to execute different computations of the algorithm is termed as time-shared or folded architecture.

Any synchronous digital design sharing the HW building blocks for different computation in different cycles will require a controller. The controller directs different computations of the algorithm to use the resource in different clock cycles. This type of a controller is also called a scheduler. There are several options of designing the controller or scheduler. In this chapter we cover the controller design for time-shared architectures.

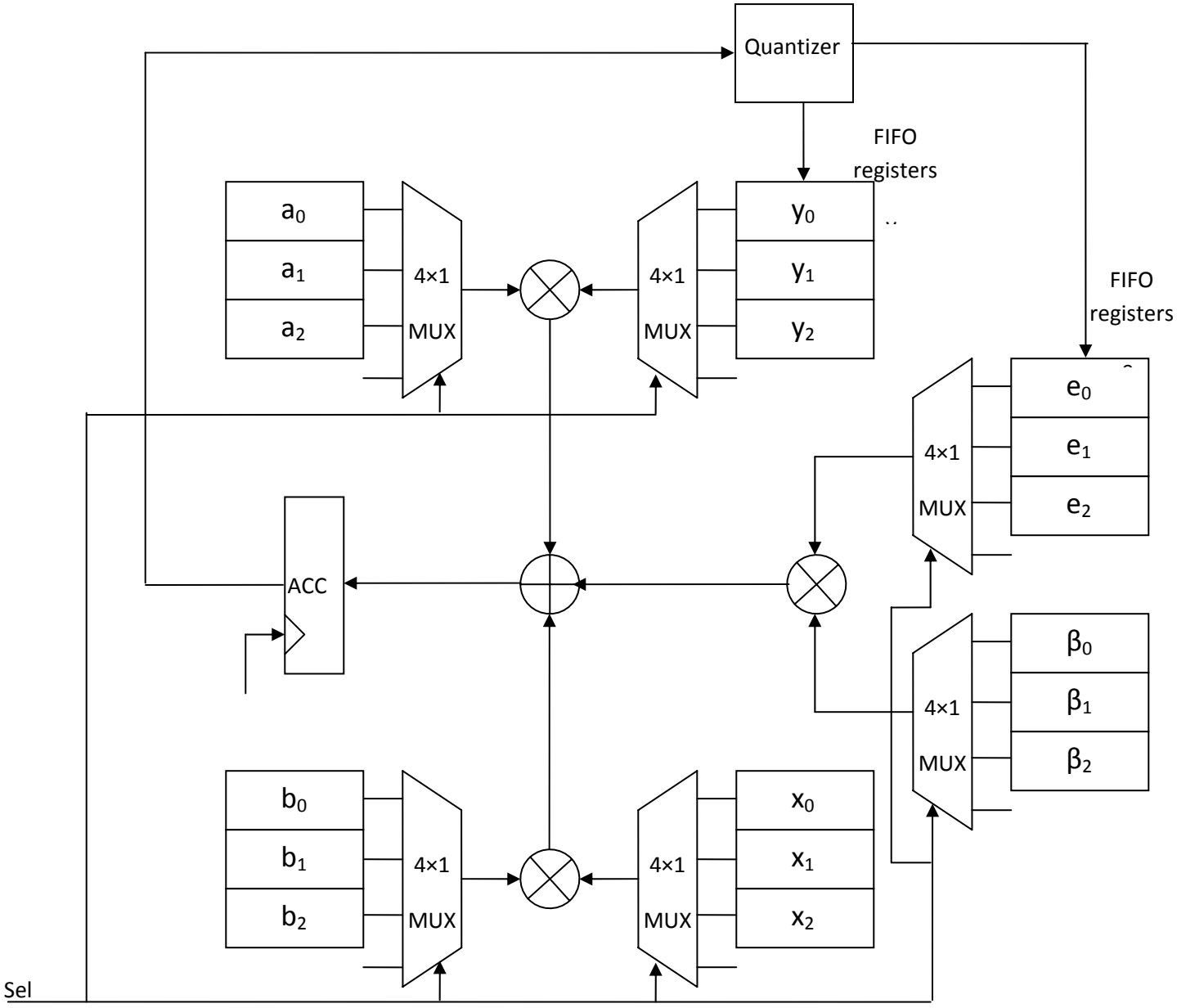

Figure 6-1: Time Shared Architecture of Proposed system model shown in Figure 4-1.

Figure 6-1 shows the time architecture model of our proposed system model and it can be used to address the solution of our problem i.e. to reduce the area in IIR-WEF technique without reducing quantization bits. The main motive behind using Time Architecture technique is to reduce the silicon area (i.e. LUT's and FF's) occupied by IIR-WEF technique by introducing reusability of resources. The model takes following steps to generate the compensated filter output $y\hat{}n$

1. Instead of using separate input signals for input x[n], y[n], a[n], b[n] and β[n], FIFO registers are used to store each of the signals as shown in Figure 6-1.
2. Multiplexers (MUX) must be used in addition to select the input we want to use during a specific iteration. The select line of MUX will be used to select any one of the signal from a FIFO register e.g. a select line will be used to select a_0 and b_0 from their respective FIFO registers during first iteration so that b_0 can be multiplied with x_0 (i.e. the input signal) which is selected from its respective FIFO register. Same is the case with filter error feedback coefficients β.
3. Accumulator (ACC) is used to store temporarily the product of input signal with filter feedforward and feedback coefficients. The output of the ACC is the fed to the quantizer where output y[n] and error signal e[n] are generated.
4. The select lines for the multiplexers and Multipliers/adders are common for input signals and FIFO registers which makes this time architecture design for IIR-WEF model much more economical in terms of occupied silicon area compared with our proposed system model.
5. The architecture shown in Figure 6-1 will reduce the error at the output of the quantizer compared with IIR-WOEF design yet using same number of resources and occupied area (LUT's and FF's).
6. The only disadvantage is that the time architecture will require more number of cycles to compute the output i.e. the frequency will increase in IIR-WEF design.

CHAPTER 7

7.1 Conclusions

The structure that has been implemented here with an Error feedback (EF) logic decreases the error at the output of the quantizer which allows us to implement a filter with coefficients quantized to lower bits that provides better efficiency. An approach to estimate signal samples width N and IIR filter coefficients W with logic area estimation, error and usage of resources is presented here in this paper. The suitable characteristics for better design optimization were chosen. The filter with an error feedback and without error feedback logic was implemented using Virtex2 FPGA and the results were compared using numerical and graphical means. The amount of occupied logic and Quantization error can be reduced depending on position of filter poles.

The efficiency of the EF schemes was examined by test implementations of some standard filters. It was found that the error feedback is a very powerful and versatile method to cut down the quantization noise in any recursive filter implemented as a cascade of second-order direct form I sections. Second-order error feedback seems to be sufficient for standard cascade implementations, whereas the new high-order schemes are attractive for use with high order direct form sections.

The tools used to build our proposed model are MATLAB and ACCELDSP 10.1. At the end it is concluded that compensation logic provides a very efficient and less error prone solution with the expense of more resources and logic area usage compared to the model with no compensation logic embedded with it. A method has also been discussed as how we can reduce the logic area of a model with compensation logic embedded with it to make it much more efficient.

7.2 Further Optimizations

Some further optimizations to improve logic area consumption can be made by setting the value of poles exactly equal to the value of zeros. By setting the width of poles coefficients a equal to zeros coefficients b results in minimization of resources in terms of FF and Registers usage.

Different implementation platforms can also be used in addition to Virtex2 platform for better comparison of results.

Above mentioned system model can also be made using a different IIR filter structure like cascade or parallel IIR filter structures and the results of compensated logic implemented on different structures can be compared so that the best and most optimal model with a better structure can be deduced.

REFERENCES

[1] B.A.Shenoi, Introduction to Digital Signal Processing and Filter design, John Wiley & Sons,Inc., 2002.

[2] Bob Meddins, Introduction to Digital Signal Processing, Essential Electronics Series.

[3] V.K.Ingle, J.G.Proakis, Digital Signal Processing using Matlab V4, Bookware Companion Series, 2000.

[4] S.White, Digital Signal Processing-filtering approach, copyright 2000.

[5] http://en.wikipedia.org/wiki/Infinite_impulse_response.

[6] M.H.Hayes, Digital Signal Processing, Schaum's Outline Series, McGraw Hill 1999.

[7] www.stanford.edu/class/ee375/lectures/EE375_Chapter_15.ppt.

[8] V.L.Nir, P.K.Pandey, Marc Moonen, IIR filter designing exercise,Digital Signal Processing II, 2001.

[9] Tran-Thone and B. Liu., "A recursive digital filter usine DPCM." IEEE Trans. Commun., vol COM-24, ppr2-11, Jan. 1976.

[10] I. Abu-El-Haija, "An approach to eliminate roundoff error in digital filters," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-27, pp. 195-198, Apr. 1979.

[11] W. E. Higgins and D. C. Munson, "Noise reduction strategies for digital filters: Error spectrum shaping versus the optimal linear statespace formulation, "IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-30, pp. 963-973, Dec. 1982.

[12] P. S. R. Diniz and A. Antoniou, "Low sensitivity digital filter Structures that are amenable to error-spectrum shaping:" IEEE Trans. Circuits Syst., vol. CAS-32, pp. 1000-1007, Oct. 1985.

[13] W. E. Higgins and D. C. Munson, "Optimal and suboptimal error spectrum shaping for cascade-form digital filters," IEEE Trans. Circuits Syst., vol. CAS-31, pp. 429-437, May 1984.

[14] P. S. R. Diniz and A. Antoniou, "Digital filter structures based on the concept of the voltage conversion generalized-immittance converter," Can. J. Elec. Comp. Eng., vol. 13, no. 3-4, pp. 90-98, July 1988.

[15] T.I. Laakso and I. 0. Hartimo, "Noise Reduction in Recursive Digital Filters Using High-Order Error Feedback," IEEE Trans. Signal Processing, vol. 40, pp. 1096-1107, May 1992.