A Decimated Electronic Cochlea on a Reconfigurable Platform

Wong Chun Kit

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

Abstract

Electronic cochlea models are used for physiological modeling as well as in many signal processing tasks such as pitch detection and speech recognition. Hardware based electronic cochlea systems offer improved performance, lower power consumption and smaller footprint over software based systems. Field Programmable Gate Arrays (FPGA) provide a means to reconfigure cochlea systems so that they can serve as the front-end signal processing for different models of the auditory nervous system.

This study is concerned with the efficient implementation of cochlea filters in digital hardware. It is demonstrated that computations in the low frequency sections of the cochlea cascade can be reduced by employing decimation. High performance FPGA-based implementation of cochlea systems with different levels of decimation were developed. This study also introduces a sequential processing architecture and evaluates the accuracy, performance and resource utilization of different implementations using fixed-point and dual fixed-point arithmetic.

A baseline cochlea employing a sequential processing unit was developed and could achieve a maximum processing rate of 933 kHz. A variable decimation cochlea was developed to speed up the computations of the cochlea systems and provided an efficient method to evaluate the aliasing and performance tradeoffs of a cochlea system by varying decimation used. To demonstrate the efficiency of dual fixed-point arithmetic, cochlea systems using dual fixed-point arithmetic were implemented and compared with fixed-point systems.

The resulting decimated implementations achieved 173% and 49% speedups with cutoff values of -60 dB and -80 dB respectively. The DFX designs had a 20-30 dB higher signal to noise ratio than the fixed-point designs. Measurements show that these DFX implementations used 24.9% more logic resources but the number and size of the multipliers used in both fixed-point and DFX implementations were the same.

摘要

電子耳蝸模型被使用在生理學上的模型及很多訊號處理任務裡，它們包括音高偵測和語音識別等。硬件築成的電子耳蝸系統比軟件築成的會提供更高的性能，更低的耗電量及更少的所佔空間。由於現場可編程門陣能提供一個方法來重新配置電子耳蝸系統，因此它能作為不同聽覺神經系統模型的前端訊號處理。

本研究涉及電子耳蝸在數位硬體內的有效實施，和證明了能透過取樣法來減少耳蝸串聯中低頻部分的計算次數。本研究已開發了在現場可編程門陣上具不同取樣水平的高性能電子耳蝸及介紹了一個順序處理式的架構，而且在準確性、性能和資源利用各方面評估了使用固定點運算法和雙固定點運算法的不同電子耳蝸。

本研究已實現了一個具有順序處理式裝置的電子耳蝸基礎，它並且能實現 933 千赫茲的最高處理速度，也發開了一個具可變取樣法的電子耳蝸來加速耳蝸系統內的計算，及通過改變取樣法來提供一種有效率的方法去評估怎樣平衡取樣法在電子耳蝸中所引致的失真效果及性能。為了證明雙固定點運算法的效用，在本研究中實現的電子耳蝸系統使用雙固定點運算法，並且與使用固定點運算法的系統作出比較。

在運用了負 60 分貝和負 80 分貝截頻點的電子耳蝸中，取樣法已分別取得了 173% 和 49% 的加速效果，而用雙固定點運算法的電子耳蝸設計比用固定點運算法的已擁有多 20 至 30 分貝的訊號雜訊比。本研究雖然也測量出用雙固定點運算法的電子耳蝸比用固定點運算法的多用了 24.9% 的邏輯資源，但是它們使用了相同大小及數量一樣的乘算器。

Acknowledgments

It would be impossible to complete this dissertation without help of many people. I would like to take this opportunity to thank them.

Firstly, I want to thank my final year project and Master Degree supervisor, Prof. Leong Heng Wai Philip for his help and guidance in the past two years. He shows his generosity and gave me encouragement and numerous ideas for my research.

I would like to thank Prof. Lee Kin Hong for his suggestions and comments for improving this work.

I would like to thank my colleagues, Mr. Y.M. Lam, Mr. K.H. Tsoi Brittle, Mr. M.H. Li Brian, Mr. Y.H. Cheung Ocean and Mr. W.S. Lau for their help in my research.

I would like to thank my family for their support. This dissertation is dedicated to my family.

Contents

1 Introduction **1**

1.1 Background and Motivation . . . 1

1.2 Objectives . . . 4

1.3 Contributions . . . 4

1.4 Thesis Outline . . . 5

2 Digital Signal Processing **6**

2.1 Introduction . . . 6

2.2 Discrete-time Signals and Systems . . . 7

2.2.1 Discrete-time Signals . . . 7

2.2.2 Discrete-time Signal Processing Systems . . . 9

2.2.3 Linear Time-Invariant (LTI) Systems . . . 10

2.3 Finite Impulse Response (FIR) Filters . . . 13

2.3.1 Introduction . . . 13

2.3.2 Windowing FIR Filter Design Method . . . 15

2.4 Infinite Impulse Response (IIR) Filters . . . 17

2.4.1 Introduction . . . 17

2.4.2 Bilinear Transform IIR Filter Design Method . . . 18

2.4.3 Spectral Transformations of IIR Filters . . . 22

2.5 Comparison on FIR and IIR Filters . . . 25

2.6 Digital Signal Resampling . . . 26

2.6.1 Introduction 26
2.6.2 Resampling by Decimation 26
2.6.3 Resampling by Interpolation 28
2.6.4 Resampling by a Rational Factor 29
2.7 Introduction to Dual Fixed-point (DFX) Representation 30
2.8 Summary 33

3 Lyon and Mead's Cochlea Model **34**
3.1 Introduction 34
3.2 Digital Cochlea Model: Cascaded IIR Filters 37
3.2.1 Introduction 37
3.2.2 Bandwidth and Centre frequencies 38
3.2.3 Zeros and Poles 39
3.3 Modifications for Decimated Cochlea Model 41
3.3.1 Introduction 41
3.3.2 Aliasing Avoidance 42
3.3.3 Coefficient Modification after Decimation 43
3.4 Summary 47

4 System Architecture **48**
4.1 Introduction 48
4.2 Hardware Platform and CAD Tools 48
4.3 Sequential Processing Electronic Cochlea 51
4.3.1 Pipelining - An Interleaving Scheme 53
4.3.2 Decimation in Sequential Processing Electronic Cochlea . 54
4.3.3 Multiple Sequential Cores 55
4.3.4 Architecture of the DFX Filter Computation Core 55
4.4 Summary 60

5 Experimental Results **61**

5.1 Introduction 61
5.2 Testing Environment 61
5.3 Performance of the Sequential Electronic Cochlea 63
5.3.1 Comparisons 63
5.4 Summary 69

6 Conclusions **70**
6.1 Future Work 72

Bibliography **73**

List of Figures

1.1 Cascaded IIR biquadratic sections used in the Lyon and Mead cochlea model. 2

2.1 The Block diagram of a FIR filter. 14

2.2 The Block diagram of a N input tap, M output tap IIR filter. . 17

2.3 Mapping of the s-plane onto z-plane using bilinear transformation. 20

2.4 Mapping of the continuous-time frequency axis onto the discrete-time frequency axis using bilinear transformation. 20

2.5 Nonlinear relationship between the Ω (s-domain) and ω (z-domain) frequencies. 21

2.6 Mapping of the angular frequencies in the lowpass-to-lowpass transformation. 24

2.7 Spectra for aliasing problem of sample rate reduction by a factor of M: (a) Spectrum of original signal; (b) Spectrum of signal decimated by M. 27

2.8 Avoiding aliasing problem by lowpass filter: (a) Spectrum of original signal; (b) Spectrum filtered by lowpass filter; (c) Spectrum of signal decimated by M. 28

2.9 Block diagram of sample rate reduction by a factor of M. 28

2.10 Interpolation by a factor of L: (a) Spectrum of original signal; (b) Spectrum of signal inserted $L-1$ zeros between each sample; (c) Spectrum filtered by lowpass filter. 29

2.11 Block diagram of sample rate increased by a factor of L. 29
2.12 Block diagram of sample rate changed by a factor of L/M. . . . 30

3.1 Human auditory system. 35
3.2 The basilar membrane inside cochlea. 36
3.3 Centre frequencies of different stages with sampling frequency 16 kHz. 38
3.4 Recursive algorithm of centre frequency calculation 39
3.5 Frequency responses of the electronic cochlea with 169 stages and sampling frequency 200 kHz. 40
3.6 Decimated electronic cochlea. 41
3.7 Normalized computation rate against cutoff value of a system with sampling frequency of 200 kHz (169 stages). 43
3.8 Comparison of different coefficient modification methods: (a) Original frequency response; (b) Modify by spectral transformation approach; (c) Modify by coefficient regeneration approach. . 46
3.9 Frequency responses of the decimated electronic cochlea with 169 stages, sampling frequency of 200 kHz and decimation cutoff value of 50 dB. 47

4.1 The simplified structure of Xilinx slice. 49
4.2 Xilinx Virtex-II Pro CLB structure. 50
4.3 The architecture of a second order IIR filter. 51
4.4 Block diagram of the fully pipelined sequential processing electronic cochlea. 52
4.5 The interleaving scheme for the pipelined electronic cochlea. . . 53
4.6 Multiple sequential processing cochlea cores. 56
4.7 Filter computation module for dual fixed-point $n\_f_0\_f_1$. 57
4.8 Dual fixed-point fractional matcher. 58

5.1 Size and timing comparison of different designs. (FIX x_y refers to a fixed-point system with wordlength x bits and fractional wordlength y bits. DFX x_y_z is explained in Section 2.7) 62
5.2 Relative SNR of 88 cascaded filters systems of different arithmetic and wordlength. 65
5.3 Relative SNR of 169 cascaded filters systems of different arithmetic and wordlength. 66
5.4 Processing rate as a function of decimation level. 68

List of Tables

2.1 Comparison between two different fixed-point scalings in DFX: $Scale_0$ and $Scale_1$ 31

4.1 Calculation stages for different samples. 55

4.2 Rescaling operations for different situations. 60

5.1 Size and timing comparison of different designs. 63

5.2 Multipliers and BlockRAMs usage for different designs. 64

5.3 Maximum processing rate for different designs. 67

Chapter 1

Introduction

1.1 Background and Motivation

The field of neuromorphic engineering, led by Carver Mead [LM88] in 1980s has the objective to use Very Large Scale Integration (VLSI) implementations of biological inspired systems for signal processing. Compared to traditional systems, the biological signal processing systems proposed by the research in this field have led to new applications. As the cochlea serves as the front-end signal processing for all functions of the auditory nervous system such as auditory localization, pitch detection, and speech recognition, research in the biological cochlea system is believed to be potentially important in the study of the auditory system.

The human cochlea acts as a transducer which converts mechanical vibrations from the middle ear into neural electrical discharges, and additionally provides spatial separation of frequency information in manner similar to that of a spectrum analyzer. In many signal processing tasks such as speech recognition, it is clear that the humans can perform better than the most sophisticated computer-based systems so better understanding of the cochlea may lead to improved neuromorphic systems.

Although using software systems to simulate cochlea models is possible, hardware implementations can achieve much better performance. When the

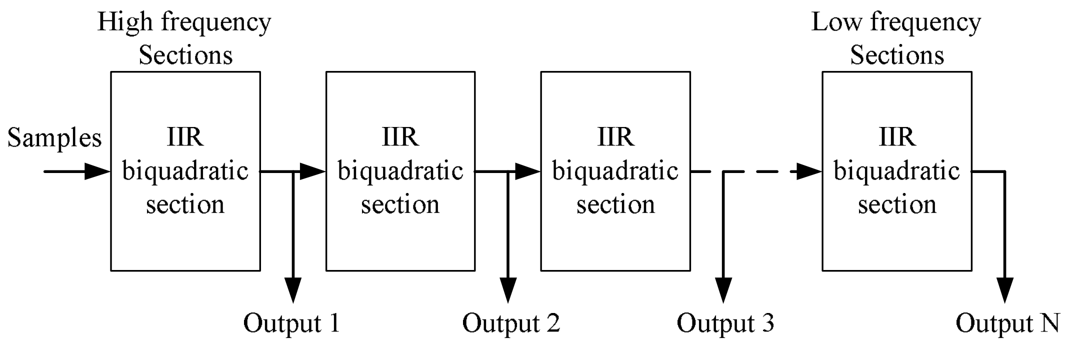


Figure 1.1: Cascaded IIR biquadratic sections used in the Lyon and Mead cochlea model.

target applications are on embedded devices in which power efficiency and small footprint are design considerations, hardware implementations become even more attractive over software simulation systems.

The electronic cochlea, first proposed by Lyon and Mead, models the human cochlea as a cascaded series of biquadratic filter sections (as shown in Figure 1.1). Many audio signal processing systems such as pitch detection [LM89b], spatial localization [LM89a], computer peripheral [LWK94], amplitude modulation detection [vSM99], correlation [MAL91] and speech recognition [LWL97] have successfully used the electronic cochlea model.

In 1988, Lyon and Mead published the original implementation of the electronic cochlea with a cascaded of 480 stages in analog VLSI technology [Lyo91]. Another analog VLSI version was proposed Watts et. al. in 1992 with 50 stages and improved dynamic range and stability [LWM92]. In 1997, van Schaik et. al. used compatible lateral bipolar transistor to implement a 104-stage analog VLSI electronic cochlea with great improvement [AvSV97]. In 2005, Sarpeshkar et. al. proposed and implemented a low-power analog cochlea called the "Bionic Ear" which is q device mimicking the ear to help the deaf [SS05].

Although analog electronic cochlea designs are potentially more efficient as they use the physical current and voltage properties of transistors and avoid digitalization, digital designs have the advantage of insensitivity to environment changes such as temperature and noise from power supply. For digital VLSI electronic cochlea implementations, an ASIC design which used bit-serial second-order filters was reported by Summerfield and Lyon in 1992 [SL92]. In 1997, Lim et. al. used a first-order Butterworth bandpass filter to implement the cochlea filtering in the VHDL-based pitch detection system [SCLJ97]. Brucke et. al., in 1998, implemented a VLSI speech preprocessor which used gammatone filter banks to mimic the cochlea [MBK98]. In 2002, a tenth-order recursive cochlea was implemented by Mishra et. al. using FPGA technology [MH02]. In 2003, an FPGA-based module generator that used distributed arithmetic to implement the biquadratic filters was proposed by M. P. Leong. Using this module generator, designs with different numbers of inputs, filter coefficients and precision could be generated [LJL03].

Field-Programmable Gate Arrays (FPGAs) are hardware devices which can be reconfigured, after fabrication. Functions inside FPGA devices can be changed by downloading different bitstreams for different applications. The following are the advantages of FPGAs:

- The electronic cochlea serves as the front-end signal processing for different functions of the auditory nervous system. For auditory nervous systems with different functionalities, the electronic cochlea should be modifiable with appropriate configurations such as filters coefficients, number of stages and wordlength. With FPGAs, it is possible to reconfigure the cochlea on the chip accordingly.

- Compared with application specific integrated circuit (ASIC) technology, FPGAs have a shorter development period, better stability and faster time to market.

1.2 Objectives

The Lyon and Mead cochlea model is a cascaded series of biquadratic filter sections with exponentially decreasing cutoff frequencies (as shown in Figure 1.1). The signals in the low frequency sections of the cascaded series filters are of lower bandwidth than those in earlier sections since each IIR filter in the cascade has a lowpass transfer function. Thus a lower sampling rate can be tolerated in later sections. Traditionally, implementations of the electronic cochlea have operated at a single sampling frequency, and data is processed at a higher sampling rate than necessary.

The main objective of this work was to employ decimation to reduce computation in the Lyon and Mead cochlea model, and dual fixed-point arithmetic to improve dynamic range. The following features were desired:

- develop a hardware Lyon and Mead cochlea in which decimation is used to reduce computation in low frequency sections.
- design an electronic cochlea with dual fixed-point (DFX) arithmetic [ECC04] to improve dynamic range.
- devise a sequential electronic cochlea architecture which employs pipelined infinite impulse response filter stages.

1.3 Contributions

This thesis presents an improved design for an electronic cochlea filter employing decimation to reduce computation, and dual fixed-point arithmetic to improve dynamic range. The contributions of this work are:

- a study of tradeoffs in the aliasing and redundant filter computation reduction through decimation of the Lyon and Mead electronic cochlea was made.

- Comparisons between a standard fixed-point cochlea implementation and the cochlea with one using dual fixed-point (DFX) arithmetic [ECC04] which employs a single bit exponents to select between two different fixed-point representations were made in terms of accuracy, performance and resource utilization.

- A novel architecture for both pipelined and sequential implementations which use the new ideas introduced.

- Together, these improvements have led to a family of FPGA implementations of the Lyon and Mead electronic cochlea model with improved speed and accuracy over all previous reported designs.

1.4 Thesis Outline

Background information describing digital signal processing which includes the principle of resampling and the dual fixed-point (DFX) arithmetic are presented in Chapter 2. Chapter 3 provides a description of Lyon and Mead's cochlea model as well as the description of human cochlea. In Chapter 4, architectures for the implementation of an electronic cochlea are introduced. Results are presented in Chapter 5 and conclusions are drawn in Chapter 6.

Chapter 2

Digital Signal Processing

2.1 Introduction

In this chapter, the fundamental concepts of discrete-time signals and signal processing systems for one-dimensional signals are introduced. The design methods for discrete-time filters and the digital signal resampling are also detailed.

The discrete-time signals and signal processing systems are introduced at the beginning of this chapter. In next section, the introduction and the design method of FIR filters are discussed and followed by the introduction and the design method of IIR filters. The spectral transformation used in IIR filters is also described. Then FIR and IIR filters are compared to present the advantages and disadvantages of each. The digital signal resampling knowledge involving decimation and interpolation is presented in next section. Finally, a data representation used in digital signal processing, called dual fixed-point is introduced in this chapter and compared with the fixed-point to demonstrate its improved dynamic range.

2.2 Discrete-time Signals and Systems

2.2.1 Discrete-time Signals

The field of signal processing is the science of analyzing and manipulating time varying signals. Signal processing is mainly divided into two categories, analog signal processing and digital signal processing, based on whether the processed signal is continuous-time or discrete-time. The continuous-time signals or analog signals, have continuous waveforms in the time domain and can be represented by a continuous independent variable. Examples of continuous-time signals are the continuous voltage and current representations used in analog circuits. Discrete-time signals are used to describe signals who have quantized time variables and therefore the value of the signals can only be defined at discrete instants of time. A continuous waveform cannot be used to represent a discrete-time signal but used a sequence of values instead. Besides the time variable of signals being discrete, the amplitude can be discrete. A digital signal in which both time and amplitude are discrete [OSB98].

Discrete-time processing systems are systems which deal with the transformation of discrete-time signals: discrete in time but continuous in amplitude. Digital signal processing systems are the systems which manipulate with signals that are discrete in both time and amplitude.

From a mathematically viewpoint, a sequence of numbers are used to represent a discrete-time signal and the time variable is represented as an integer in the range from $-\infty$ to ∞. A sequence of numbers x, in which the value of the nth sample is denoted as $x[n]$ is given as:

$$x = \{x[n]\} \qquad -\infty < n < \infty \qquad \text{for } n \in \mathbf{I} \tag{2.1}$$

As the time variables of discrete-time signals are quantized, $x[n]$ is only defined for integer values of n and is undefined for non-integer values of n. By using the unit function:

$$\delta[t] = \begin{cases} 0, & t \neq 0, \\ 1, & t = 0. \end{cases} \tag{2.2}$$

the value of the nth sample can be generally expressed as:

$$x[n] = \sum_{i=-\infty}^{\infty} x[i]\delta[n-i] \tag{2.3}$$

In fact, discrete-time signals can be obtained by periodically sampling a continuous-time signal. The sequences x can be generated by periodically sampling an analog signal at uniform time intervals. The value of the nth number in the sequence is equal to the value of the analog signal, $x_{analog}(t)$, at time nT:

$$x[n] = x_{analog}(t)|_{t=nT} = x_{analog}(nT) \qquad -\infty < n < \infty \qquad \text{for } n \in \mathbf{I} \tag{2.4}$$

where T is the spacing between two successive samples and is called the sampling interval or sampling period. The reciprocal of the sampling period, T, is called sampling frequency, F_T:

$$F_T = \frac{1}{T} \tag{2.5}$$

The unit of sampling frequency is hertz (Hz), and that of sampling period is second (s). The sampling angular frequency is denoted as Ω_T:

$$\Omega_T = 2\pi F_T = \frac{2\pi}{T} \tag{2.6}$$

For example, if the continuous-time signal is:

$$x_{analog}(t) = A\sin(2\pi f_o t + \phi) = A\sin(\Omega_o t + \phi) \tag{2.7}$$

the discrete-time signal generated by periodically sampling $x_a(t)$ is:

$$\begin{aligned} x[n] &= x_{analog}(t)|_{t=nT} \\ &= A\sin(\Omega_o nT + \phi) \\ &= A\sin(\frac{2\pi\Omega_o}{\Omega_T}n + \phi) \\ &= A\sin(\omega_o n + \phi) \end{aligned} \quad (2.8)$$

where

$$\omega_o = \frac{2\pi\Omega_o}{\Omega_T} = \Omega_o T = \frac{\Omega_o}{F_T} \quad (2.9)$$

ω_o is the normalized digital angular frequency of the discrete-time signal $x[n]$ with the unit of radians per sample, while the unit of the normalized analog angular frequency Ω_o is radians per second [Mit01].

In discrete-time signal processing, the normalized digital angular frequency, ω_o, has the range from $-\pi$ to π:

$$\begin{aligned} -\pi \leq \omega_o &\leq \pi \\ -\pi \leq \frac{\Omega_o}{F_T} &\leq \pi \\ -\frac{F_T}{2} \leq f_o &\leq \frac{F_T}{2} \\ |f_o| &\leq \frac{F_T}{2} \end{aligned} \quad (2.10)$$

According to Eq. (2.10), in order to sample an analog signal with the bandwidth, f_o, the value of sampling frequency used must at least double the signal bandwidth, i.e. $F_T = 2f_o$.

2.2.2 Discrete-time Signal Processing Systems

From a mathematical viewpoint, discrete-time signal processing system is defined as a function that transforms an input sequence with values $x[n]$ into an output sequence with values $y[n]$. It is denoted as:

$$y[n] = F\{x[n]\} \tag{2.11}$$

In Eq. 2.11, the function F is used to represent the transformation operation for computing the output sequence values from the input sequence values. The values of the output sequence at each index n depend on the value of the whole sequence x.

2.2.3 Linear Time-Invariant (LTI) Systems

In the analysis of discrete-time systems, linearity and time-invariance are two important characteristics. The result of any discrete-time signal processing system can be more easily modeled if it is linear and time-invariant.

The linear characteristic is defined by the principle of superposition. Linear systems are systems in which the output is the superposition or sum of the individual outputs obtained by applying the respective individual inputs separately to the system. If $y_1[n]$ and $y_2[n]$ are the responses of a system when $x_1[n]$ and $x_2[n]$ are the respective inputs, the system, L is linear if and only if

$$L\{x_1[n] + x_2[n]\} = L\{x_1[n]\} + L\{x_2[n]\} = y_1[n] + y_2[n] \tag{2.12}$$

$$L\{ax[n]\} = aL\{x[n]\} = ay[n] \tag{2.13}$$

where a is any constant. Eq. (2.12) is called the additivity property and Eq. (2.13) is called scaling property or homogeneity property. The principle of superposition is the combination of these two properties:

$$L\{ax_1[n] + bx_2[n]\} = aL\{x_1[n]\} + bL\{x_2[n]\} \tag{2.14}$$

where a and b any constants. If the input of the linear system is the superposition of many inputs, Eq. (2.14) can be further generalized:

$$x[n] = \sum_k a_k x_k[n] \tag{2.15}$$

and the output of the system would be:

$$\begin{aligned} y[n] &= L\{x[n]\} \\ &= L\{\sum_k a_k x_k[n]\} \\ &= \sum_k a_k L\{x_k[n]\} \\ &= \sum_k a_k y_k[n] \end{aligned} \tag{2.16}$$

where $y_k[n]$ is the corresponding system output to the input $x_k[n]$. According to Eq. (2.16), if the system is linear and a superposed input is applied, the system output would be equal to superposing the outputs of respective input components [OSB98].

A time-invariant system is a system in which a time shift or delay in the input sequence causes a equivalent time shift in the output sequence [Lyo01]. Suppose a system, F, which transforms the input sequence with values,$x[n]$, into the output sequence with values, $y[n]$.

$$y[n] = F\{x[n]\} \tag{2.17}$$

The transformation system is said to be time-invariant if applying the shifted version of the original $x[n]$ input, $x'[n] = x[n-\rho]$, then the output sequence:

$$y'[n] = F\{x'[n]\} = F\{x[n-\rho]\} = y[n-\rho] \tag{2.18}$$

where ρ is an integer representing a ρ sample time shift. Eq. (2.18) must hold for any time shift ρ and any input sequence if the system is time-invariant.

For a linear time-invariant system, by combining its linear property with the representation of the general sequence as a linear combination of delayed impulses as in Eq. (2.3), the system can be completely characterized by its impulse response. Let $h_k[n]$ be the response of the system, F, for the impulse input $\delta[n-k]$, occurring at $n = k$. From Eq. (2.3),

$$y[n] = F\left\{ \sum_{k=-\infty}^{\infty} x[k]\delta[n-k] \right\} \tag{2.19}$$

By using the principle of superposition in Eq. (2.14),

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]F\{\delta[n-k]\} = \sum_{k=-\infty}^{\infty} x[k]h_k[n] \tag{2.20}$$

The system response to any input sequence can be expressed in terms of the system's responses to the sequence $\delta[n-k]$. The time-invariant property of the system supports that if $h[n]$ is the response to $\delta[n]$, the response of $\delta[n-k]$ is $h[n-k]$. Therefore, Eq. (2.20) becomes:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{2.21}$$

The output $y[n]$ can be computed from system's impulse response, $h[n]$ and the input sequence $x[n]$ [OSB98]. Consequently, a linear time-invariant system is completely characterized by its impulse response $h[n]$. Eq. (2.21) is called a convolution sum. The output $y[n]$ is called the convolution of the sequence $x[n]$ with $h[n]$ and the notation is:

$$y[n] = x[n] * h[n] \tag{2.22}$$

The convolution operation is commutative:

$$y[n] = h[n] * x[n] \tag{2.23}$$

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \tag{2.24}$$

According to Eq. (2.21) and (2.24), the convolution sum expresses each sample of the output sequence in terms of all of the samples of the input and impulse response sequences [OSB98].

Filters are particularly important class of linear time-invariant systems. Filters can be divided into two kinds: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters.

2.3 Finite Impulse Response (FIR) Filters

2.3.1 Introduction

FIR digital filters use only the current and past input samples to generate the current output sample value. None of the filter's previous output samples are used and are also called nonrecursive filters. If the duration of nonzero input values is finite, an FIR filter always has a finite duration of nonzero output values. Therefore, if the impulse is used as the input of the FIR filter, the filter's output will eventually be a sequence of all zeros and hence, a finite impulse response. Figure 2.1 shows the basic block diagram of an FIR filter of length N (N-tap). The nth output of a general N-tap FIR filter is:

$$y[n] = \sum_{k=0}^{N-1} b[k]x[n-k] \tag{2.25}$$

Comparing Eq. (2.24) and (2.25), the output of a general FIR filter, Eq. (2.25), is the convolution sum of the input sequence and impulse response

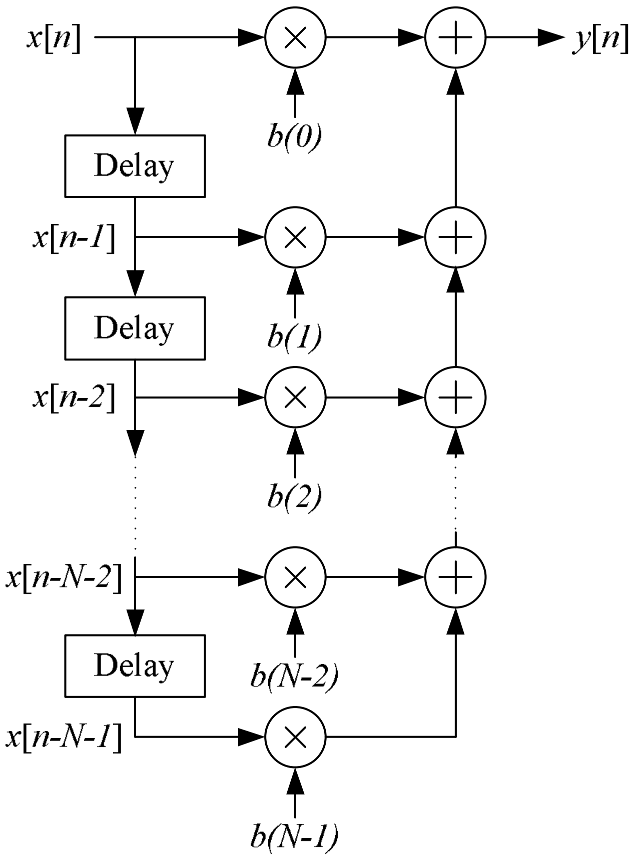


Figure 2.1: The Block diagram of a FIR filter.

sequence of finite length, N. Therefore, the impulse response of a FIR filter is identical to the FIR filter coefficients.

The discrete Fourier transform (DFT) result from the convolution of a filter's impulse response (coefficients) and the input sequence is equal to the product of the spectrum of the input sequence and the DFT of the impulse response. Given two time-domain sequences $h[k]$ and $x[n]$ having the DFTs $H[\omega]$ and $X[\omega]$ respectively, the DFT of $y[n] = h[k] * x[n]$ is $H[\omega] \cdot X[\omega]$:

$$y[n] = h[k] * x[n] \quad \underset{\text{IDFT}}{\overset{\text{DFT}}{\rightleftarrows}} \quad H[\omega] \cdot X[\omega] \tag{2.26}$$

in which IDFT means the inverse DFT. Eq. (2.26) shows the sequence $y[n]$ resulting from $h[k]*x[n]$ and the sequence resulting from $H[\omega]\cdot X[\omega]$ are Fourier transform pairs. Therefore, taking the DFT of $h[k] * x[n]$ would return the product of $H[\omega]$ and $X[\omega]$ which is the frequency spectrum of filter output $Y[\omega]$. In conclusion, convolution in the time-domain is equivalent to multiplication in the frequency-domain [Lyo01].

2.3.2 Windowing FIR Filter Design Method

The design techniques for FIR filters are based on directly approximating the desired frequency response of the discrete-time system. The simplest FIR filter design method is called the window method. The steps to determine the filter's time-domain coefficients are:

1. Define the expression for the discrete frequency response $H[\omega]$
2. Apply the expression to the inverse DFT to obtain the time domain $h[k]$
3. Evaluate the $h[k]$ expression as a function of time index k.

For designing an ideal lowpass filter with cutoff frequency f_c, the frequency response is defined as $H[\omega]$:

$$H[\omega] = \begin{cases} 1, & |\omega| \le Mf_c/f_s, \\ 0, & \text{otherwise.} \end{cases} \tag{2.27}$$

where f_s is the sampling frequency and M is the number of samples covering $-f_s/2$ to $fs/2$. The result of the inverse DFT of $H[\omega]$ is:

$$h[k] = \frac{1}{M} \cdot \frac{\sin(\pi k K/M)}{\sin(\pi k/M)} \qquad \text{where} \quad K = \frac{2Mf_c}{f_s} \tag{2.28}$$

However, this impulse response expression is infinitely long and a practical FIR filter would require truncation of the impulse response. The new truncated impulse response, $h^*[k]$, is:

$$h^*[k] = \begin{cases} h[k], & 0 \leq k \leq N, \\ 0, & \text{otherwise.} \end{cases} \tag{2.29}$$

where N is the number of taps (coefficients) of the FIR filter. $h^*[k]$ in fact is the product of the desired impulse response and a finite-duration 'window' $w[k]$:

$$h^*[k] = h[k]w[k] \tag{2.30}$$

where $w[k]$ is the rectangular window:

$$w[k] = \begin{cases} 1, & 0 \leq k \leq N, \\ 0, & \text{otherwise.} \end{cases} \tag{2.31}$$

Recalling that from Eq. (2.26), multiplication in the time-domain is equivalent to convolution in the frequency-domain, the actual frequency response $H^*[\omega]$ is the convolution of $H[\omega]$ and $W[\omega]$. As the DFT of the rectangular window, $w[k]$, returns a function, $W[\omega]$, which contains side lobes, passband ripples occur in the actual frequency response $H^*[\omega]$. Although increasing the number of taps can make the filter's transition region more narrow, it is impossible to reduce or even eliminate the passband ripples. As long as $w[k]$ is a finite number of values (i.e. a rectangular window of finite width), there are sidelobe ripples in $W[\omega]$ and passband ripples would be induced in the final frequency response $H^*[\omega]$ by the convolution of $H[\omega]$ and $W[\omega]$ [Lyo01].

Although passband ripples cannot be completely eliminated, some window functions which can increase the magnitude ratio between mainlobe and sidelobes are commonly used in FIR filter design. These include Bartlett, Hanning, Hamming and Blackman windows [OSB98].

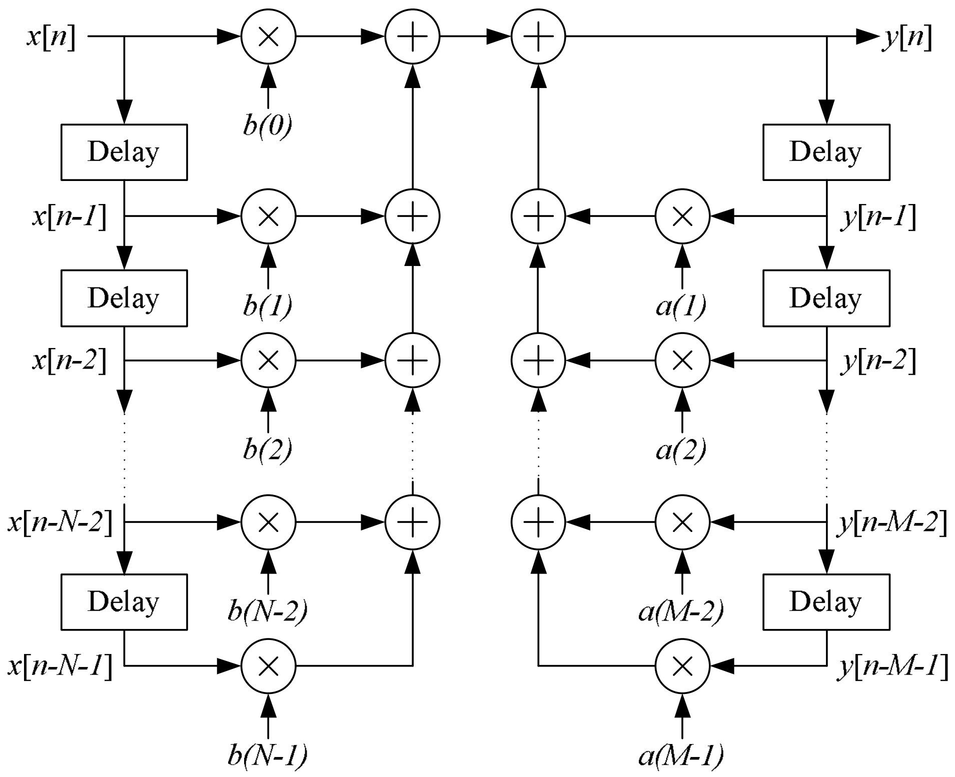


Figure 2.2: The Block diagram of a N input tap, M output tap IIR filter.

2.4 Infinite Impulse Response (IIR) Filters

2.4.1 Introduction

Infinite impulse response (IIR) digital filters are different from finite impulse response (FIR) filters because IIR filters involve feedback of the output. While the output sample of FIR filters depend only on the current and past input samples, each IIR filter output sample depends on previous input samples and previous output samples i.e they are recursive filters. As IIR filters are feedback systems, theoretically, an impulse input can cause the filter to oscillate indefinitely. The output of the system can possibly have an infinite duration even if the input becomes a sequence of all zeros. Therefore, the system can theoretically have an infinite impulse response [Lyo01].

Figure 2.2 shows the basic block diagram of an IIR filter with N-tap input samples and M-tap output samples. That means each output sample of the IIR filter shown in Figure 2.2 is generated by using N number of input samples and M number of previous output samples. The nth output of this IIR filter is:

$$y[n] = \sum_{k=0}^{N-1} b[k]x[n-k] + \sum_{k=1}^{M-1} a[k]y[n-k] \qquad (2.32)$$

2.4.2 Bilinear Transform IIR Filter Design Method

Due to the nonrecursive nature of FIR filters, the desired filter coefficients can be directly determined from the impulse response sequence. However, as IIR filters use feedback, filters' coefficients $a[k]$ and $b[k]$ cannot be directly computed from the impulse response. Instead, the transformation of a continuous-time filter (analog filter) into a discrete-time filter is used to design IIR filters.

The first step in the design of IIR filters is to design a prototype continuous-time filter using the analog specifications obtained from transformation of the desired discrete-time specifications. Some approximation methods in the continuous-time filter design such as Butterworth and Chebyshev are used to obtain the system function $H_c[s]$ of the continuous-time filter. Finally, by applying a continuous-time-to-discrete-time spectral transformation to $H_c[s]$, the system function $H[z]$ for the discrete-time filter would be obtained [OSB98]. Bilinear transformation is one of such transformation method.

The bilinear transformation is an algebraic transformation between the variables s and z: the entire $j\Omega$-axis in the s-plane is mapped to one revolution of the unit circle in the z-plane [Lyo01]. As $-\infty \leq \Omega \leq \infty$ is mapped onto $-\pi \leq \omega \leq \pi$, the transformation between the continuous-time (s-plane) and the discrete-time (z-plane) frequency variables must be nonlinear.

Assume $H_c[s]$ be the continuous-time system function and $H[z]$ be the discrete-time system function, the bilinear transformation is to substitute s

by:

$$s = \frac{2}{T_d}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \tag{2.33}$$

where T_d is called the sampling parameter and should be related to the sampling frequency. But T_d is of no consequence in the design procedures if the design problem is assumed to be always start from the specifications on the discrete-time filter. When the discrete-time specifications are mapped to continuous-time specifications and the continuous-time filter is mapped back to a discrete-time filter, the effect of T_d would be canceled [OSB98]. By solving Eq. (2.33), an expression in terms of the s-domain can be derived:

$$z = \frac{1 + (T_d/2)s}{1 - (T_d/2)s} \tag{2.34}$$

and by substituting $s = \sigma + j\Omega$ into Eq. (2.34), the magnitude of z is given by:

$$|z| = \sqrt{\frac{(1 + \sigma T_d/2)^2 + (\Omega T_d/2)^2}{(1 - \sigma T_d/2)^2 + (\Omega T_d/2)^2}} \tag{2.35}$$

If σ is negative ($\sigma < 0$), the numerator of the ratio of $|z|$ would be less than the denominator and $|z| < 1$. On the other hand, if σ is positive ($\sigma > 0$), the numerator would be larger than the denominator and $|z| > 1$. Therefore, when using the bilinear transformation, any poles located on the left side of the s-plane ($\sigma < 0$) would be mapped to a z-plane inside the unit circle. In other words, any stable s-plane pole of the prototype analog filter would be mapped to a stable z-plane pole for the discrete-time IIR filter (Figure 2.3).

In the bilinear transformation, the $j\Omega$-axis of the s-plane is mapped onto the unit circle. By substituting $s = j\Omega$ i.e. $\sigma = 0$ into Eq. (2.35):

$$|z| = \sqrt{\frac{(1)^2 + (\Omega T_d/2)^2}{(1)^2 + (\Omega T_d/2)^2}} = 1 \tag{2.36}$$

For all values of s on the $j\Omega$-axis, Eq. (2.36) shows that $|z| = 1$. That means,

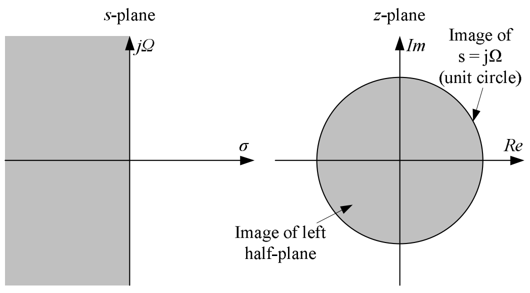


Figure 2.3: Mapping of the s-plane onto z-plane using bilinear transformation.

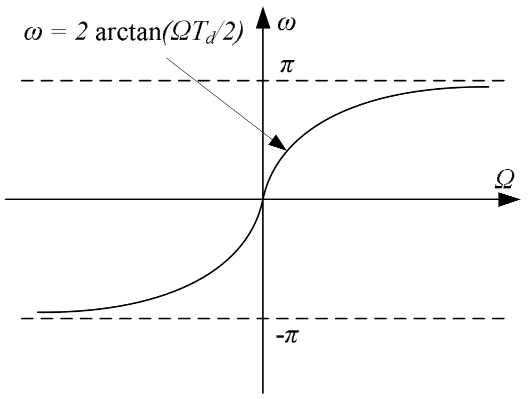


Figure 2.4: Mapping of the continuous-time frequency axis onto the discrete-time frequency axis using bilinear transformation.

the $j\Omega$-axis of the s-plane is mapped onto the unit circle.

By substituting $s = \sigma + j\Omega$ and $z = e^{j\omega}$ into Eq. (2.33) and using Euler's relationship:

$$\begin{aligned}
s &= \sigma + j\Omega \\
s &= \frac{2}{T_d} \cdot \frac{e^{-j\omega/2}}{e^{-j\omega/2}} \cdot \left[\frac{e^{j\omega/2} - e^{-j\omega/2}}{e^{j\omega/2} + e^{-j\omega/2}} \right] \\
\sigma + j\Omega &= \frac{2}{T_d} \left[\frac{2e^{-j\omega/2}(j \sin \omega/2)}{2e^{-j\omega/2}(\cos \omega/2)} \right] \\
&= \frac{2j}{T_d} \tan(\omega/2) \qquad (2.37)
\end{aligned}$$

Therefore,

$$\sigma = 0 \qquad \text{and} \qquad \Omega = \frac{2}{T_d} \tan(\omega/2) \qquad (2.38)$$

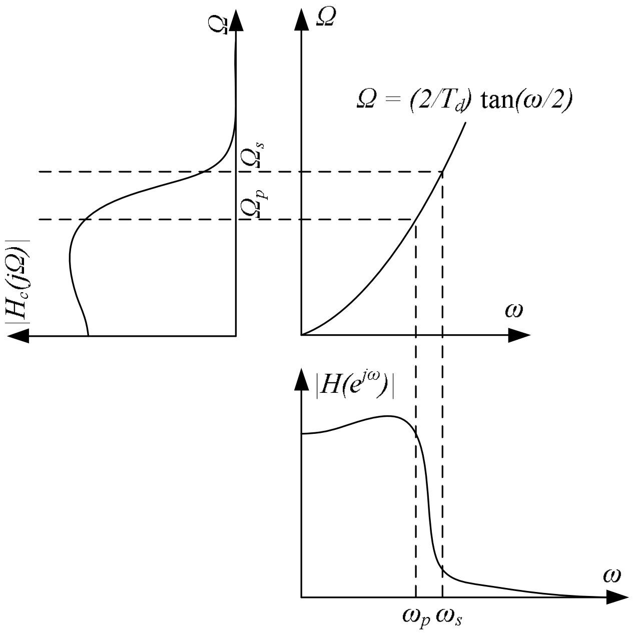


Figure 2.5: Nonlinear relationship between the Ω (s-domain) and ω (z-domain) frequencies.

$$\omega = 2\arctan\left(\frac{\Omega T_d}{2}\right) \tag{2.39}$$

Eq. (2.38) and Eq. (2.39) show the bilinear transformation is a mapping from the s-plane to the z-plane. From Figure 2.4, it shows that the range of frequencies $0 \le \Omega \le \infty$ in the continuous-time s-domain is mapped onto the discrete-time z-domain with the range $0 \le \omega \le \pi$ and the range $-\infty \le \Omega \le 0$ is mapped onto $-\pi \le \omega \le 0$. Although the bilinear transformation forms a nonlinear relationship between the s-domain frequency, Ω, and the z-domain frequency, ω, it can avoid the aliasing since the entire $j\Omega$-axis on the s-plane is mapped onto the unit circle of z-plane. By using this nonlinear transformation, the frequency response of the prototype continuous-time filter in s-domain can be transformed to the frequency response of the desired discrete-time filter in z-domain [OSB98].

In analog systems, the intersection of the vertical $\sigma = 0$ plane and the $|H_c[s]|$ surface in the three-dimensional s-plane gives the frequency magnitude response $|H_c[\Omega]|$. As the bilinear transformation maps the the $s = j\Omega$-axis to the unit circle $|z| = 1$, the frequency magnitude response $|H[\omega]|$ of the discrete-time system would be the intersection of the unit circle $|z| = 1$ and the $|H[z]|$ surface in the three-dimensional z-plane.

2.4.3 Spectral Transformations of IIR Filters

In addition to repeating the filter design procedures, there is a method to modify the characteristics of an existing IIR filter to fulfill the new specifications. For example, it is possible to modify an existing lowpass filter with 500 Hz cutoff frequency to a new lowpass filter with the cutoff frequency of 1 kHz. Besides modifying its cutoff frequency or passband frequency, it is possible to transform an existing lowpass IIR filter to a new IIR filter with highpass, bandpass or bandstop characteristic. This spectral transformation method is a mapping between the original lowpass transfer function, $H_o[z]$, and the desired transfer function, $H_n[z]$, that can be a lowpass, highpass, bandpass or bandstop filter.

Let z^{-1} be the unit delay in the original lowpass filter $H_o[z]$ and $\tilde{z}^{-1}$ be the notation of the unit delay in the transformed filter $H_n[\tilde{z}]$. The unit circles in the z-plane and $\tilde{z}$-plane are defined by:

$$z = e^{j\omega} \qquad \text{and} \qquad \tilde{z} = e^{j\tilde{\omega}}$$

The transformation from z-domain to $\tilde{z}$-domain is denoted as:

$$z = T[\tilde{z}] \tag{2.40}$$

The transform operation, T, must fulfill the following constraints:

- In order to transform a rational $H_o[z]$ into a rational $H_n[\tilde{z}]$, $T[\tilde{z}]$ must be a rational function of $\tilde{z}$.
- The inside of the unit circle of the z-plane must be mapped into the

inside of the unit circle of the $\tilde{z}$-plane. Otherwise, $H_n[\tilde{z}]$ may become unstable.

- The points on the unit circle of the z-plane must be mapped to the points on the unit circle of the $\tilde{z}$-plane in order to map the original magnitude response to the magnitude response of the desired filter.

In the z-plane, points on the unit circle are given by $|z| = 1$, points inside the unit circle are defined by $|z| < 1$ and points outside the unit circle are characterized by $|z| > 1$. The above constraints can be written as:

$$|T[\tilde{z}]| = \begin{cases} > 1, & \text{if } |z| > 1, \\ = 1, & \text{if } |z| = 1, \\ < 1, & \text{if } |z| < 1. \end{cases} \tag{2.41}$$

and the general form of $T^{-1}[\tilde{z}]$ with real coefficients is defined by:

$$T^{-1}[\tilde{z}] = \pm \prod_{\ell=1}^{L} \left(\frac{1 - \alpha_\ell \tilde{z}}{\tilde{z} - \alpha_\ell} \right), \tag{2.42}$$

where $|\alpha_\ell|$ is either real or occurs in complex conjugate pairs with $|\alpha_\ell| < 1$ for stability [Mit01]. For lowpass to lowpass transformation,

$$z^{-1} = T^{-1}[\tilde{z}] = \frac{1 - \alpha\tilde{z}}{\tilde{z} - \alpha} = \frac{\tilde{z}^{-1} - \alpha}{1 - \alpha\tilde{z}^{-1}} \tag{2.43}$$

On the unit circle, the transformation becomes:

$$e^{-j\omega} = \frac{e^{-j\tilde{\omega}} - \alpha}{1 - \alpha e^{-j\tilde{\omega}}} \tag{2.44}$$

Finally, it would become:

$$\tan\left[\frac{\omega}{2}\right] = \left(\frac{1 + \alpha}{1 - \alpha}\right) \tan\left[\frac{\tilde{\omega}}{2}\right] \tag{2.45}$$

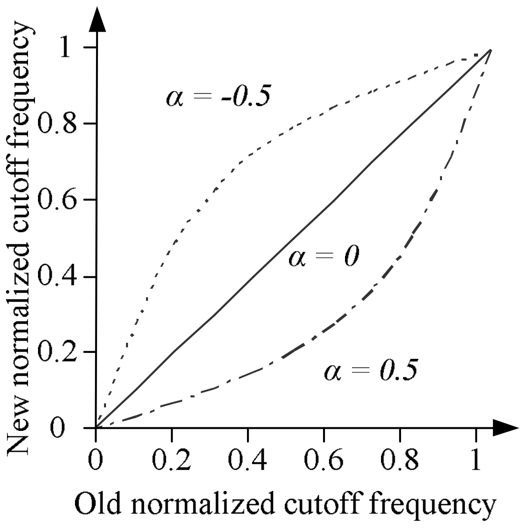


Figure 2.6: Mapping of the angular frequencies in the lowpass-to-lowpass transformation.

Figure 2.6 plots the relationship between ω and $\tilde{\omega}$ for three different values of α. The transformation is linear only for $\alpha = 0$ and for nonzero values of α, the transformation is nonlinear and a warping of the frequency scale occurs. From Eq. (2.45), the relation between the cutoff frequency, ω_c of $H_o[z]$ and the cutoff frequency, $\tilde{\omega}_c$ of $H_n[\tilde{z}]$ is:

$$\tan\left[\frac{\omega_c}{2}\right] = \left(\frac{1+\alpha}{1-\alpha}\right)\tan\left[\frac{\tilde{\omega_c}}{2}\right] \tag{2.46}$$

By solving α:

$$\alpha = \frac{\tan(\omega_c/2) - \tan(\tilde{\omega}_c/2)}{\tan(\omega_c/2) + \tan(\tilde{\omega}_c/2)} = \frac{\sin\left(\frac{\omega_c - \tilde{\omega}_c}{2}\right)}{\sin\left(\frac{\omega_c + \tilde{\omega}_c}{2}\right)} \tag{2.47}$$

and $H_o[z]$ can be transformed to $H_n[\tilde{z}]$:

$$H_n[\tilde{z}] = H_o[z] = H_o(T[\tilde{z}]) = H_o(\frac{\tilde{z} - \alpha}{1 - \alpha\tilde{z}}) \tag{2.48}$$

Although the spectral transformation is convenient in IIR filter design by using lowpass-to-highpass, lowpass-to-bandpass and lowpass-to-bandstop transformations, it can map only one frequency point, ω_c, in the magnitude response of the original lowpass filter into a new frequency position, $\tilde{\omega}_c$, of the same magnitude response value for the transformed lowpass and highpass filters. For transforming bandpass and bandstop filters, the frequency point, ω_c, is mapped into two new frequency positions, $\tilde{\omega}_{c1}$ and $\tilde{\omega}_{c2}$, of the same magnitude response values. It has the disadvantage that it can only map either passband or stopband edges of the original lowpass filter to the positions of the desired filters, but not both [Mit01].

2.5 Comparison on FIR and IIR Filters

In the implementation of digital signal processing systems, the choice between using FIR filters and IIR filters base on the importance in the design problem of the advantages of each filter type. IIR filters, for example, have a design flexibility advantage as they can simulate the predefined prototype analog filter. IIR filters also can easily be designed by using the transformation from the well developed approximate methods in the continuous-time filters such as Butterworth, Chebyshev or elliptic. However, the transformation results from the approximate methods only focus on the magnitude in the frequency response and are limited to frequency-selective filters. If the design specification requires a phase-delay or group-delay response, IIR filter design is difficult.

FIR filters can have a linear phase and are the only solution for the systems having a exact linear phase specification. On the other hand, given the same quality in magnitude response, IIR filters require fewer taps than FIR filters. That means the number of multipliers used in IIR filters are smaller than that of FIR filters with the same magnitude response. If the hardware limitation is important, the slight phase nonlinearity is tolerable, IIR filters might be the best choice [Lyo01].

2.6 Digital Signal Resampling

2.6.1 Introduction

Digital signal resampling is used to change the effective sampling rate of a discrete-time signal that has already been sampled. As the sampling period takes an important role in many signal processing techniques and applications, resampling is a useful method. For example, it can be used to minimize computations by reducing data rates when signal bandwidth is already narrowed by lowpass filters. Also, in discrete-time signal processing, the signal bandwidth is limited by the sampling frequency:

$$B < \frac{f_s}{2} \tag{2.49}$$

where B is the signal bandwidth and f_s is the sampling frequency. Increasing the sampling frequency can increase the bandwidth and enhance the auditory quality and image quality in digital auditory and image processing systems. Resampling is also imperative in the data transfer between high data rate processors and low data rate devices.

2.6.2 Resampling by Decimation

Resampling changes the digital signal by either increasing or decreasing the sample rate. Decreasing the sample rate ($f_s \downarrow$ and $T_s \uparrow$) is called decimation. For increasing the sample rate ($f_s \uparrow$ and $T_s \downarrow$), the process is called interpolation. Although, theoretically, the decrease and increase factor in resampling can be a non-integral number, decimation and interpolation are the term used for reducing and increasing the sample rate by an integral factor.

Decimating a sequence of sampled values by a factor M cannot in general be done by simply extracting every Mth sample as this would cause aliasing. Figure 2.7 shows how the aliasing occurs. As the frequency spectrum of a

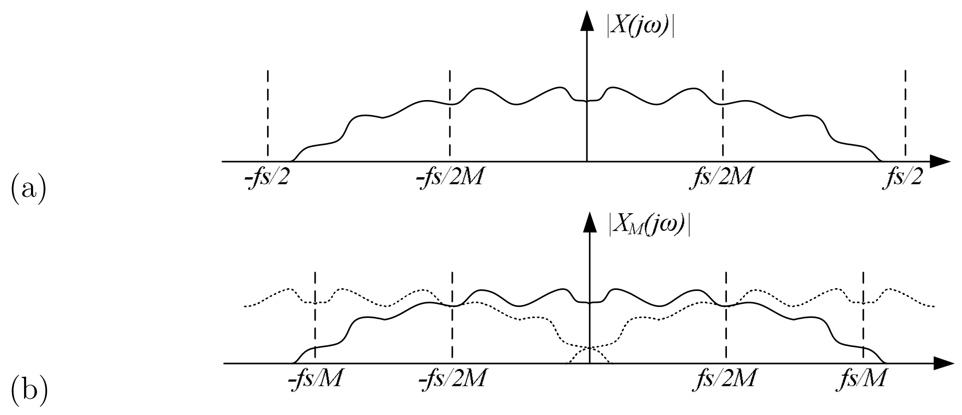


Figure 2.7: Spectra for aliasing problem of sample rate reduction by a factor of M: (a) Spectrum of original signal; (b) Spectrum of signal decimated by M.

discrete-time signal replicates every f_s frequency intervals, directly extracting every Mth sample would cause the spectra, $X_M[j\omega]$, to overlap and aliasing occurs.

To avoid aliasing, it is necessary to first filter the discrete-time signal with a lowpass filter which approximates the ideal characteristic:

$$H[e^{j\omega}] = \begin{cases} 1, & |\omega| \leq \pi/M \\ 0, & \text{otherwise.} \end{cases} \tag{2.50}$$

For the decimation of the signal $x[n]$ by a factor of M (Figure 2.8a), $x[n]$ is first filtered by a lowpass filter with cutoff frequency of $f_s/2M$ (Figure 2.8b). After filtering, the sample rate reduction is implemented by forming the new signal $y[n]$ by extracting every Mth sample of the filtered output (Figure 2.8c). This lowpass filter is called a decimator filter. However, if the frequency response of the lowpass filter is not sharp enough, aliasing will still occur as signals of frequency higher than $f_s/2M$ would corrupt the low frequency components [CR81].

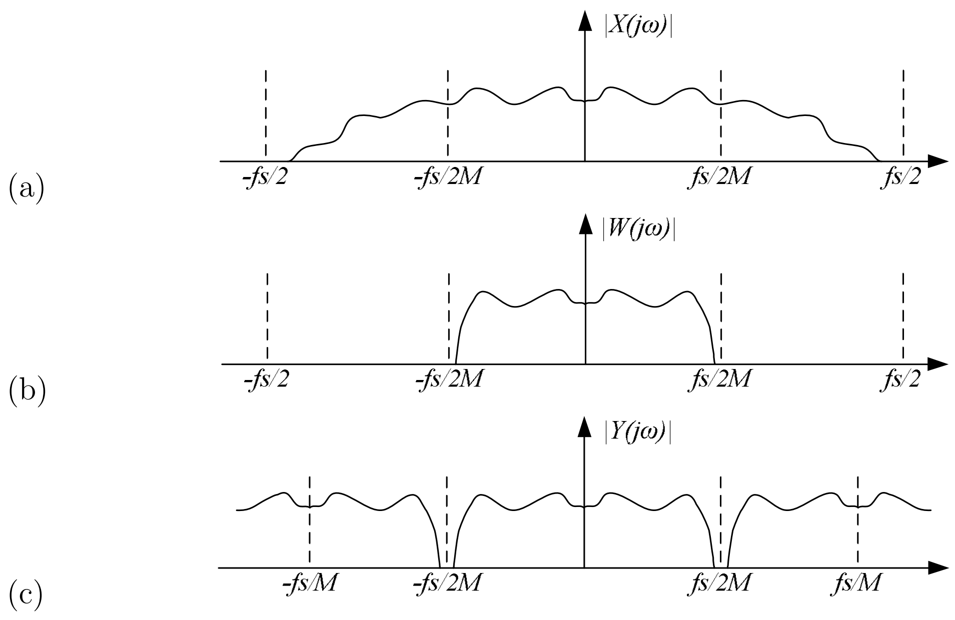


Figure 2.8: Avoiding aliasing problem by lowpass filter: (a) Spectrum of original signal; (b) Spectrum filtered by lowpass filter; (c) Spectrum of signal decimated by M.

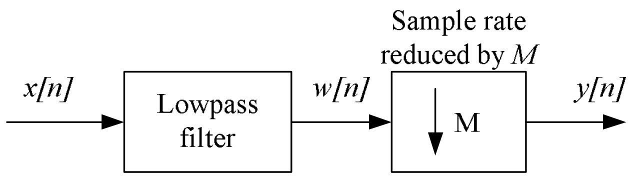


Figure 2.9: Block diagram of sample rate reduction by a factor of M.

2.6.3 Resampling by Interpolation

For the interpolation of the signal $x[n]$ (Figure 2.10a) by a factor of L, $L-1$ zeros must be first inserted between each sample of the signal $x[n]$ to generate a new sequence $v[n]$. By doing such zero padding, the sequence would be up-sampled by a factor of L. However, as the spectrum of the original signal $X[j\omega]$ is replicated with period f_s, the spectrum of the up-sampled signal $V[j\omega]$ would contain many replicated spectra (Figure 2.10b). To remove those replicated spectra, the up-sampled signal $v[n]$ is then filtered by a lowpass filter with cutoff frequency $f_s/2$ (the current sampling frequency is Lf_s). The filtered signal $y[n]$ would contain the same frequency components as the original signal

(a)
(b)
(c)

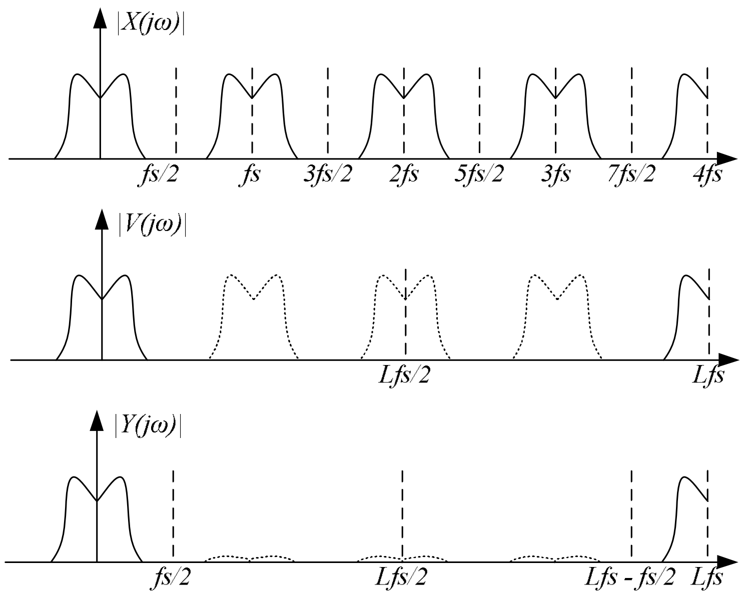


Figure 2.10: Interpolation by a factor of L: (a) Spectrum of original signal; (b) Spectrum of signal inserted $L-1$ zeros between each sample; (c) Spectrum filtered by lowpass filter.

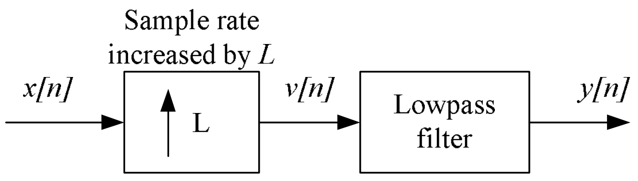


Figure 2.11: Block diagram of sample rate increased by a factor of L.

$x[n]$. The lowpass filter used in interpolation is called an interpolation filter which approximates the ideal characteristic:

$$H[e^{j\omega}] = \begin{cases} 1, & |\omega| \leq f_{s(old)}/2 = \pi/L \\ 0, & \text{otherwise.} \end{cases} \tag{2.51}$$

2.6.4 Resampling by a Rational Factor

By combining the processes of decimator and interpolation, discrete-time signals are be resampled by any rational fraction L/M. The sampled signal, $x[n]$, is first up-sampled by a factor of L by padding $L-1$ zeros between each sample

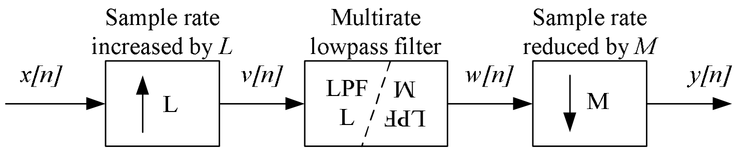


Figure 2.12: Block diagram of sample rate changed by a factor of L/M.

to form a up-sampled signal, $v[n]$. Then, it is filtered by the interpolation filter and decimation filter. As the purpose of interpolation filters and decimation filters are the same, a single combined filter which is called a multirate filter is used instead. Finally, the filtered signal, $w[n]$, is used to form the new signal, $y[n]$, by extracting every Mth sample of the filtered output.

2.7 Introduction to Dual Fixed-point (DFX) Representation

Fixed-point arithmetic is commonly used in digital signal processing system. The position of the radix point in this representation is fixed. The fixed-point number is a binary word that has a fixed number of digits before and after the radix point, namely the integral part and fractional part. Given a number, X, with wordlength n, m digits in the integral part and f digits in the fractional part ($n = m + f$), the value of that number assuming a two's complement representation is:

$$X = -x_{m-1} \cdot 2^{(m-1)} + \sum_{i=-f}^{m-2} x_i 2^i \tag{2.52}$$

in which $x_{(m-1)}$ is the most significant bit and $x_{(-f)}$ is the least significant bit. The notation for the fixed-number representation is n_f. In fixed-point representation, the position of the radix point takes an important role in the range and accuracy of the fixed-point number. Given a fixed-point number with fixed wordlength, increasing its representation range ($-2^{(m-1)} \leq X < 2^{(m-1)}$) will, at the same time, decrease its accuracy ($2^{(-f)}$) as $n = m + f$.

Scaling	Value of E	Range	Precision
$Scale_1$	1	$2^{(n-2-f_0)} \leq \text{value} < 2^{(n-2-f_1)}$ $-2^{(n-2-f_1)} \leq \text{value} < -2^{(n-2-f_0)}$ Larger Range[a]	$2^{(-f_1)}$ Lower precision
$Scale_0$	0	$-2^{(n-2-f_0)} \leq \text{value} < 2^{(n-2-f_0)}$ Smaller Range	$2^{(-f_0)}$ Higher precision

[a]since $f_0 > f_1$

Table 2.1: Comparison between two different fixed-point scalings in DFX: $Scale_0$ and $Scale_1$

In order to get a large dynamic range, a floating-point representation is used in some signal processing systems. The value of a floating-point number is represented by the pair of significand (or mantissa), M, and exponent, E:

$$X = M \cdot \beta^E \tag{2.53}$$

where β is the base of the exponent and commonly equals to 2. Although floating-point representation can provide a large dynamic range, the computation of its arithmetic has high complexity. Therefore, the floating-point representation normally would not be used in a hardware digital signal processing system.

In 2004, Ewe et. al. proposed a dual fixed-point (DFX) representation which has an improved dynamic range over a fixed-point representation and the computation complexity is similar to that of fixed-point [ECC04]. In dual fixed-point (DFX) data representation, a single bit exponent E is used to select between two different fixed-point scalings, $Scale_0$ and $Scale_1$. The notation $n\_f_0\_f_1$ is used to describe a DFX number system where n is the word length including the exponent bit E and $f_0 > f_1$. The value of a dual fixed-point number is:

$$\text{value} = \begin{cases} X \cdot 2^{(-f_0)}, & \text{if} \quad E = 0 \\ X \cdot 2^{(-f_1)}, & \text{if} \quad E = 1. \end{cases} \tag{2.54}$$

In order to choose the best scaling, $Scale_0$ or $Scale_1$ for DFX numbers, a boundary value $Bound$, which equals the range limit of the fixed-point number $(n-1)$-f_0 (i.e. $Bound = 2^{(n-2-f_0)}$), is used and the value of E is calculated by:

$$E = \begin{cases} 0, & \text{if} \quad -2^{(n-2-f_0)} \leq \text{ value } \leq 2^{(n-2-f_0)} - 2^{(-f_0)} \\ 1, & \text{if} \quad \text{value} < -2^{(n-2-f_0)} \text{ or value } > 2^{(n-2-f_0)} - 2^{(-f_0)}. \end{cases} \tag{2.55}$$

Table 2.1 shows the comparison between the two different fixed-point scalings in DFX representation.

The dynamic range of any number representation is defined as the ratio between the largest and smallest absolute value in that number representation format. For fixed-point representation, the smallest absolute value is $2^{(-f)}$ and the largest absolute value is $2^{(m-1)} = 2^{(n-1-f)}$. Its dynamic range is:

$$\begin{aligned} \text{Dynamic range} &= 20\log\left(\frac{2^{(n-1-f)}}{2^{(-f)}}\right) \\ &= 20\log(2^{(n-1)})dB \end{aligned} \tag{2.56}$$

For DFX, the largest and smallest absolute values are 2^{n-2-f_1} of $Scale_1$ and 2^{-f_0} in $Scale_0$ respectively. Its dynamic range is:

$$\begin{aligned} \text{Dynamic range} &= 20\log\left(\frac{2^{(n-2-f_1)}}{2^{(-f_0)}}\right) \\ &= 20\log(2^{(n-2+f_0-f_1)})dB \end{aligned} \tag{2.57}$$

As $f_0 > f_1$, the dynamic range of DFX is larger than that of fixed-point [ECC04].

2.8 Summary

In this chapter, background knowledge in digital signal processing was introduced and some design techniques for both FIR and IIR filters were discussed. After comparing FIR and IIR filters, the advantages and disadvantages of each were shown. Digital signal resampling methods, called decimation and interpolation, were alos detailed in this chapter. The dual fixed-point representation was introduced and it has the advantage of improved dynamic range and low computational complexity.

Chapter 3

Lyon and Mead's Cochlea Model

3.1 Introduction

The first cochlea model, which was proposed by Lyon and Mead in 1988 [LM88] was implemented in analog VLSI and mimics the behavior of the human cochlea by using a cascaded series of second order filters (Figure 1.1). In order to understand how the behavior of the human cochlea is mimicked by a cascade series of filters, a detail description of the human cochlea is given in this section. This hearing mechanism of the human ear was introduced by Pickles at. el. in 1988 [Pic88].

The human cochlea is three dimensional fluid-dynamic system which acts as a transducer in the human auditory system. It is used to convert mechanical vibrations from the middle ear into neural electrical signals. It is composed of the basilar membrane, inner hair cells and outer hair cells. The converted neural electrical signals are then sent to higher levels in the auditory pathway for further processing.

Figure 3.1 shows a simplified human ear system in which the oval window is the input to the cochlea. Sound that enters the outer ear would cause mechanical vibrations in the eardrum and the vibrations are propagated by

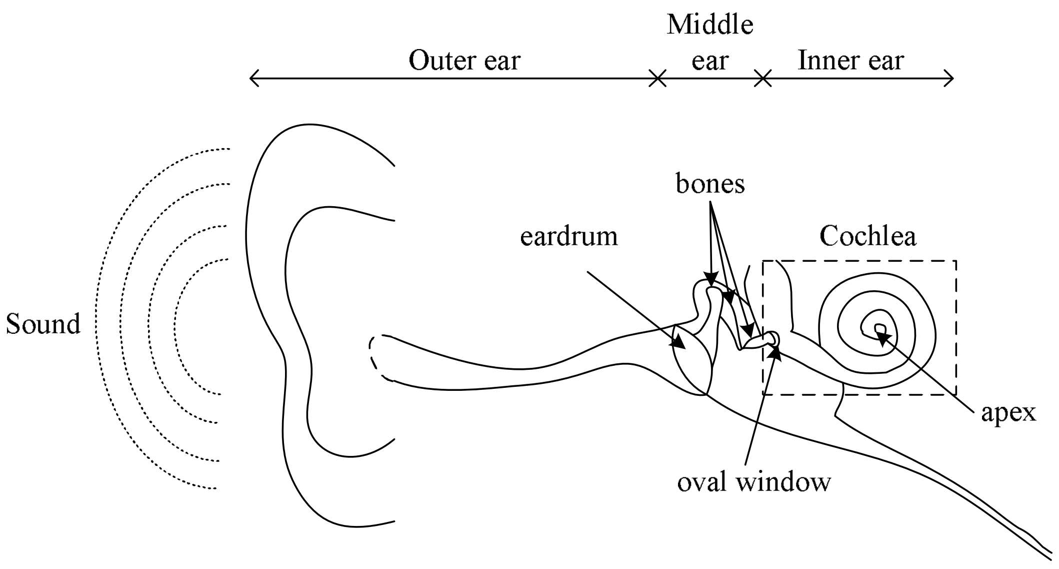


Figure 3.1: Human auditory system.

the bones in the middle ear. The mechanical pressure wave is conducted to the oval window into the cochlea. The wave is then propagated from base to apex along the basilar membrane of the cochlea duct.

The basilar membrane is a longitudinal membrane inside the cochlea duct. The stiffness of the basilar membrane varies smoothly over its length and at any point will resonate most strongly with a pressure wave of a particular frequency. The displacement is sensed and converted into neural signals by several thousand inner hair cells that are distributed along the basilar membrane (as shown in Figure 3.2). The hair cells also act as a half-wave rectifying system as they can only sense displacements in one direction.

An important characteristic of the cochlea is that the energy in the auditory wave is separated by frequency and each point of the cochlea respond best to one frequency. That means the frequency content of the signal is mapped into the spatial domain by the cochlea in a manner similar to that of a spectrum analyzer. Due to the variety of stiffness of basilar membrane, the hair cells near the base (oval window) are most sensitive to high-frequency auditory waves and as the waves travel down the cochlea, lower frequencies are sensed.

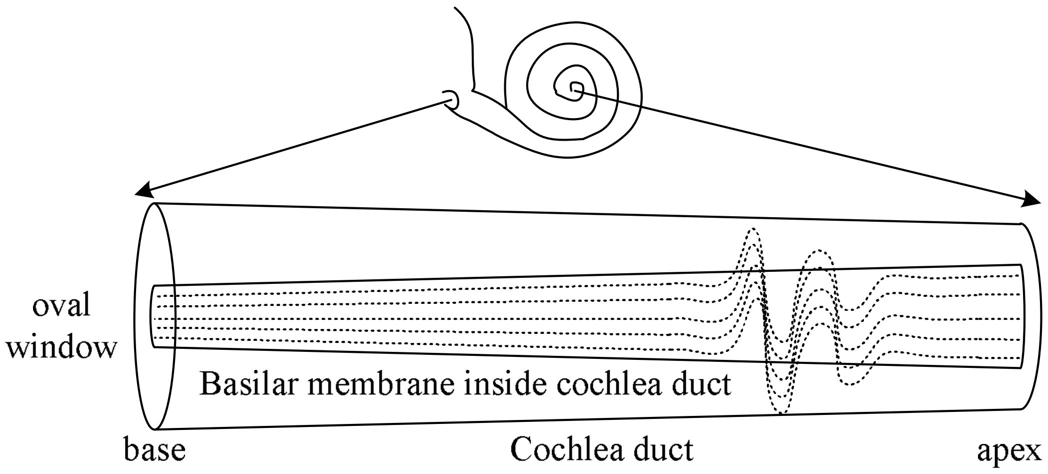


Figure 3.2: The basilar membrane inside cochlea.

To simulate the characteristics of the basilar membrane, Lyon and Mead used a cascaded series of second order lowpass filters, each filter having the following transfer function:

$$H[s] = \frac{1}{\tau^2 s^2 + \frac{1}{Q}\tau s + 1} \tag{3.1}$$

where τ is the time constant and Q is the damping quality factor. The values of τ are varied along the cascade so that the cutoff frequencies of the filters decrease exponentially. The quality factors, Q of all filters are the same in order to maintain a constant damping characteristic.

In Lyon and Mead's cochlea model, each stage of the cascaded series has the following properties:

- acts as a resonant filter at a given frequency.
- acts as a lowpass filter which filters out the high frequency components already handled by the earlier stages of the cascade.

3.2 Digital Cochlea Model: Cascaded IIR Filters

3.2.1 Introduction

As the original Lyon and Mead's cochlea model is a cochlea model using analog filters, the analog filter model is converted into a discrete-time filter model by using the bilinear transformation introduced in Section 2.4.2 of Chapter 2 [Sla88] [Sla98]. In the bilinear transformation, the continuous-time variable, s, is substituted by discrete-time variable z as:

$$s = \frac{2}{T_d}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right)$$

The transfer function, $H[s]$ in Eq. (3.1), for each analog filter is converted into a second order discrete-time transfer function:

$$H[z] = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \tag{3.2}$$

$$H[z] = \frac{\sum_{i=0}^{2} b_i z^{-i}}{\sum_{i=0}^{2} a_i z^{-i}} \tag{3.3}$$

where a_i and b_i can be expressed in terms of τ and Q of $H[s]$ from the bilinear transformation. Therefore, a_i and b_i can be calculated after determining the τ and Q of $H[s]$ in Eq.(3.1).

As the value of the time constant, τ, depends on the cutoff and pass frequencies of its own filter stage, it can be expressed as the ratio between the cutoff frequency and the pass frequency. Furthermore, the cutoff and pass frequencies of each filter stage are based on its filter bandwidth. The bandwidth of each second order filter is first determined.

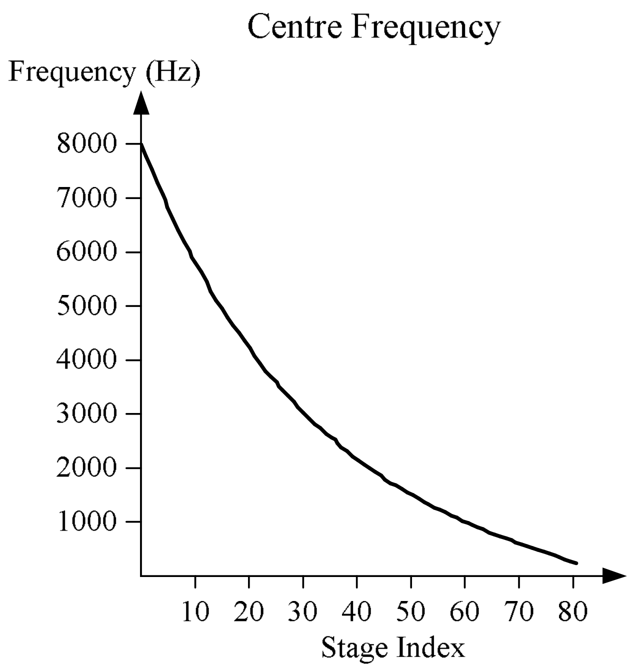


Figure 3.3: Centre frequencies of different stages with sampling frequency 16 kHz.

3.2.2 Bandwidth and Centre frequencies

The bandwidth of each filter is calculated from its centre frequency. At the high frequency sections of the cascaded series, the filter's bandwidth is approximately equal to the centre frequency divided by a constant $EarQ$ while the bandwidth of filters at the low frequency sections approaches to a constant of $BreakFreq/EarQ$. Therefore, the bandwidth is calculated as:

$$Bandwidth[cf] = \frac{\sqrt{cf^2 + BreakFreq^2}}{EarQ} \tag{3.4}$$

where cf is the centre frequency of that filter stage [Sla88]. Normally, the values of $BreakFreq$ and $EarQ$ are 1000 Hz and 8 respectively so that the bandwidth of the last filter stage is approximately equal to 125 Hz.

In the cascaded series of the cochlea model, the frequency responses of successive cochlea filters are overlapped with each other by a fraction of their bandwidths. Although this overlapped parameter is arbitrary, smaller values brings more stages and computations to the cascade. A typical choice is for 4 successive stages to overlap and as a result, the factor *EarStep* is designed to be 0.25.

In order to produce exponentially decreasing centre frequencies of the filter stages, a recursive algorithm is used [Sla88]:

```
Algorithm $EarStageCF$
Input: stage index $i$ and sampling frequency $f_s$
Output: centre frequency $cf_i$
1.  if $i = 1$ /* first stage */
2.     then
3.           $cf_i \leftarrow f_s/2$
4.     else
5.           $cf_{i-1} \leftarrow EarStageCF(i-1, f_s)$
6.           $cf_i \leftarrow (cf_{i-1} - EarStep \cdot Bandwidth(cf_{i-1}))$
7.  return $cf_i$
```

Figure 3.4: Recursive algorithm of centre frequency calculation

Figure 3.3 shows centre frequencies of different stages in an example of cascaded cochlea filter series with sampling frequency 16 kHz.

3.2.3 Zeros and Poles

The cutoff frequencies (zeros) and pass frequencies (poles) of each filter stage can be determined from their centre frequencies and bandwidths.

As each stage of the cochlea acts as a resonant filter, the pole is placed at the centre frequency to provide a peak to the filter's frequency response at its centre frequency so that auditory signal components whose frequencies are close to that centre frequency will resonate. In order to act as a lowpass filter and reject the high frequency components, the zero is designed slightly above the centre frequency in the frequency response. Therefore, the zero and pole

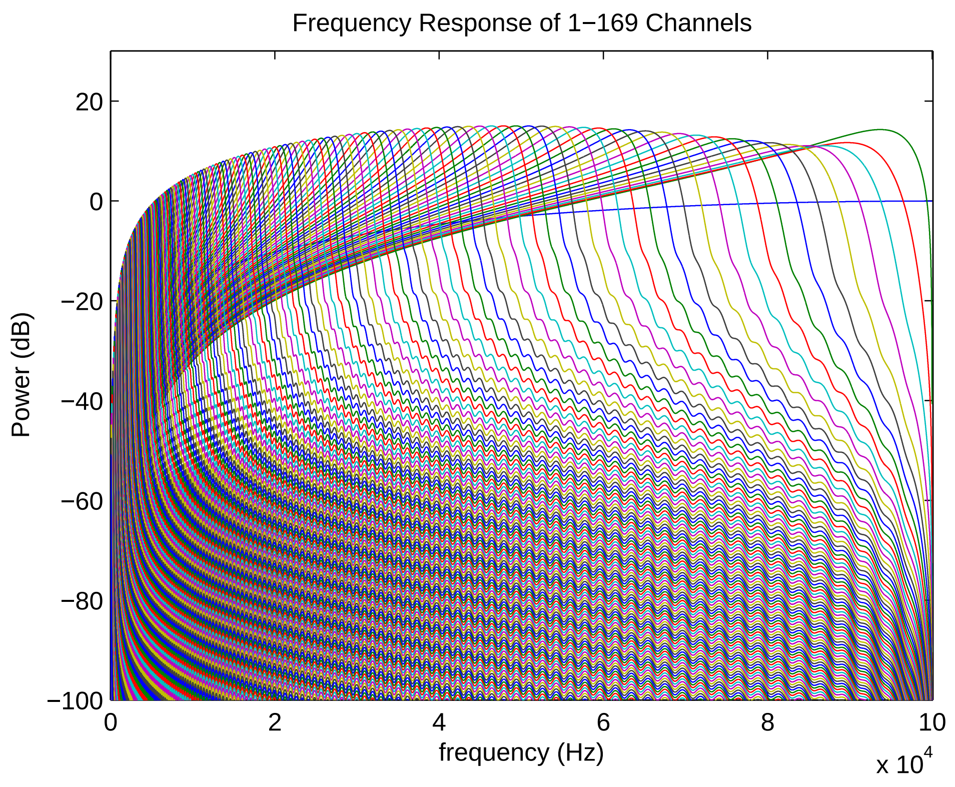


Figure 3.5: Frequency responses of the electronic cochlea with 169 stages and sampling frequency 200 kHz.

frequencies are calculated as [Sla88]:

$$PoleFreq[cf] = cf \tag{3.5}$$

$$\begin{aligned} ZeroFreq[cf] &= cf + 1.5 EarStep \cdot Bandwidth[cf] \\ &= cf + 0.375 \cdot Bandwidth[cf] \end{aligned} \tag{3.6}$$

where cf is the centre frequency. The quality factor of zeros and poles are calculated by [Sla88]:

$$PoleQ[cf] = \frac{PoleFreq[cf]}{Bandwidth[cf]} \tag{3.7}$$

$$ZeroQ[cf] = \frac{5 \cdot ZeroFreq[cf]}{Bandwidth[cf]} \tag{3.8}$$

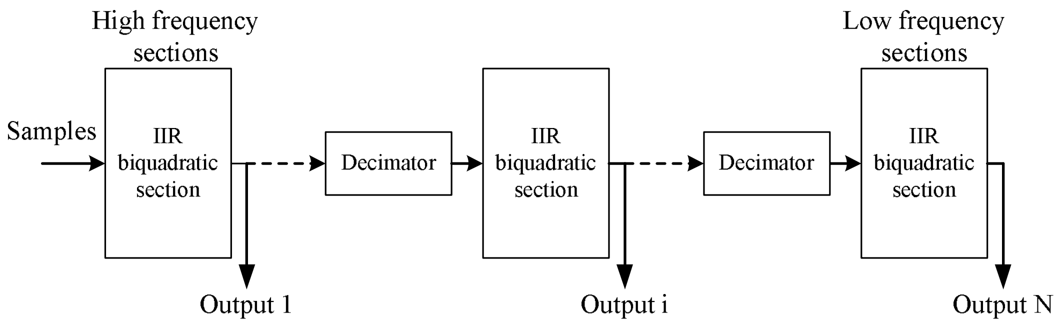


Figure 3.6: Decimated electronic cochlea.

where cf is the centre frequency. Finally, the coefficients a_i and b_i of the second order transfer function in Eq. (3.2) can be determined by using the bilinear transformation of the analog transfer function, $H[s]$ (Eq. (3.1)), in which the time constant τ and the damping quality factor Q are calculated from $PoleFreq$, $ZeroFreq$, $PoleQ$ and $ZeroQ$. Figure 3.5 shows the frequency responses of a cochlea example with 169 stages and sampling frequency 200 kHz.

3.3 Modifications for Decimated Cochlea Model

3.3.1 Introduction

Decimation can be used to avoid redundant filter computations in the low frequency sections of the Lyon and Mead model. The signals in the low frequency sections of the cascaded series filters are of lower bandwidth than those in earlier sections since each second order IIR filter in the cascaded has a lowpass transfer function. Thus a lower sampling rate can be tolerated in later sections. Traditionally, implementations of the electronic cochlea have operated at a single sampling frequency, and data is processed at a higher sampling rate than necessary.

In order to reduce the computation rate of the low frequency sections, decimation can be used (as shown in Figure 3.6). After decimation, the sampling frequency is reduced and the computation rate of subsequent filters is also reduced. For a parallel, pipelined implementation, a multirate system with high sampling frequencies for the sections near the base and lower operating frequencies at the apex can be employed. For a sequential implementation, decimation can be implemented by time-sharing IIR filters in such a way that computations for the high frequency contents are more frequent than for the low frequency sections.

Although decimation can reduce redundant filter computations in the low frequency sections, aliasing may occur and in addition, the filter coefficients need to be recomputed to reflect the change in sample rate.

3.3.2 Aliasing Avoidance

As described in Section 2.6.2 in Chapter 2, it is necessary to first filter earlier stages with a lowpass filter which approximates the ideal characteristic in order to avoid aliasing caused by decimation.

Since the Lyon and Mead's cochlea model is composed of a series of lowpass filters with exponentially decreasing cutoff frequencies, they can also act as the lowpass filter for the decimator. The remaining task for decimation is to determine suitable positions along the filter cascade to perform decimation.

In this work, the cochlea filter has been simulated with noise inputs to determine the decimation position. Each stage of the filter cascade is examined in sequence and if its signal power at frequency $f_s/2^n$ is smaller than the user-defined cutoff value in dB, that stage should be decimated by a factor of 2^{n-1}. Figure 3.7 shows the relationship between cutoff value and the computation required.

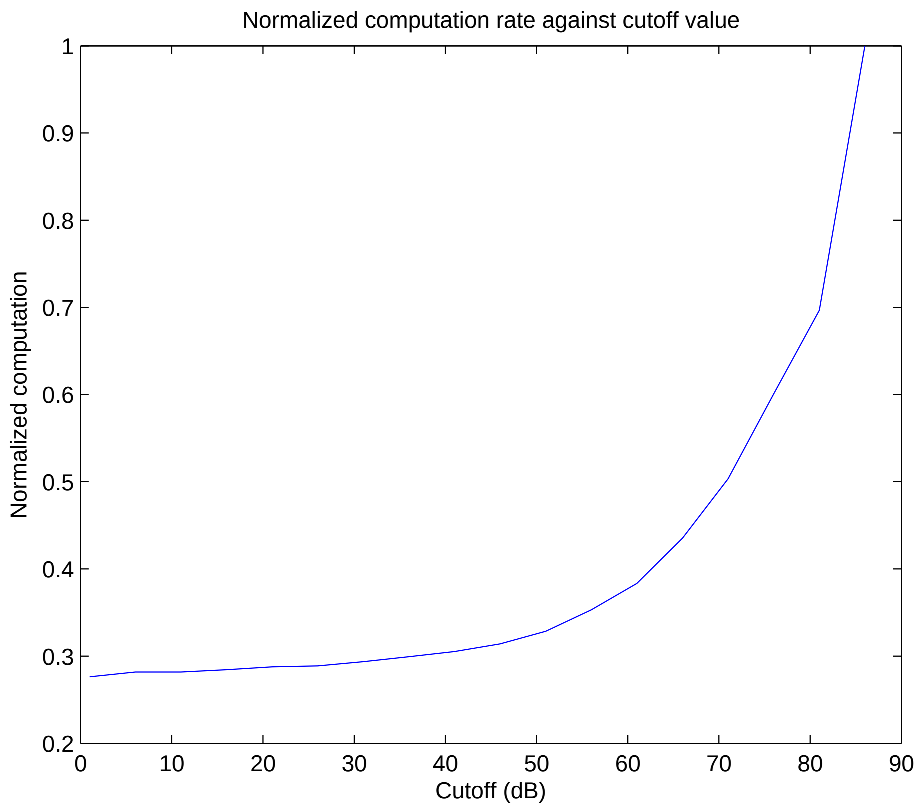


Figure 3.7: Normalized computation rate against cutoff value of a system with sampling frequency of 200 kHz (169 stages).

3.3.3 Coefficient Modification after Decimation

After decimation, the filter coefficients need to be changed for the new operating frequency since the cutoff and pass frequencies, f_c and f_p, stored in the lowpass transfer function, $H[z]$, are in the form of ω_c and ω_p, which are normalized in the range of $-\pi$ and π.

$$f_c = \frac{f_s \cdot \omega_c}{2\pi} \tag{3.9}$$

Therefore, after decimation, for those stages behind decimators in the filter cascade, their cutoff frequencies would be reduced by 2^n where n is the number of previous decimations. In order to restore their cutoff frequencies and maintain their frequency responses as original, two coefficient modification approaches have been studied: the spectral transformation and coefficient regeneration approaches.

Spectral Transformation Approach

The spectral transformation approach is based on the spectral transformation of IIR filters introduced in Section 2.4.3 of Chapter 2. By substituting Eq. (2.43) into the second order IIR filter transfer function, $H[z]$, in Eq. (3.3):

$$\begin{aligned} H[z] &= \frac{\sum_{i=0}^{2} b_i z^{-i}}{\sum_{i=0}^{2} a_i z^{-i}} \\ H[\tilde{z}] &= \frac{\sum_{i=0}^{2} b_i \left(\frac{\tilde{z}^{-1}-\alpha}{1-\alpha\tilde{z}^{-1}}\right)^i}{\sum_{i=0}^{2} a_i \left(\frac{\tilde{z}^{-1}-\alpha}{1-\alpha\tilde{z}^{-1}}\right)^i} && (3.10) \\ &= \frac{b_0\left(1-\alpha\tilde{z}^{-1}\right)^2 + b_1\left(1-\alpha\tilde{z}^{-1}\right)\left(\tilde{z}^{-1}-\alpha\right) + b_2\left(\tilde{z}^{-1}-\alpha\right)^2}{a_0\left(1-\alpha\tilde{z}^{-1}\right)^2 + a_1\left(1-\alpha\tilde{z}^{-1}\right)\left(\tilde{z}^{-1}-\alpha\right) + a_2\left(\tilde{z}^{-1}-\alpha\right)^2} && (3.11) \end{aligned}$$

where

$$\alpha = \frac{-\sin\left(\frac{\omega_c}{2}\right)}{\sin\left(\frac{3\omega_c}{2}\right)} \qquad (3.12)$$

As a result, the coefficient modification by spectral transformation can be represented as a matrix transformation function as:

$$\begin{aligned} \begin{pmatrix} b_0' & b_1' & b_2' \end{pmatrix} &= \begin{pmatrix} b_0 & b_1 & b_2 \end{pmatrix} \cdot \begin{pmatrix} 1 & -2\alpha & \alpha^2 \\ -\alpha & \alpha^2+1 & -\alpha \\ \alpha^2 & -2\alpha & 1 \end{pmatrix} && (3.13) \\ \begin{pmatrix} a_0' & a_1' & a_2' \end{pmatrix} &= \begin{pmatrix} a_0 & a_1 & a_2 \end{pmatrix} \cdot \begin{pmatrix} 1 & -2\alpha & \alpha^2 \\ -\alpha & \alpha^2+1 & -\alpha \\ \alpha^2 & -2\alpha & 1 \end{pmatrix} && (3.14) \\ H[\tilde{z}] &= \frac{\sum_{i=0}^{2} b_i' \tilde{z}^{-i}}{\sum_{i=0}^{2} a_i' \tilde{z}^{-i}} \end{aligned}$$

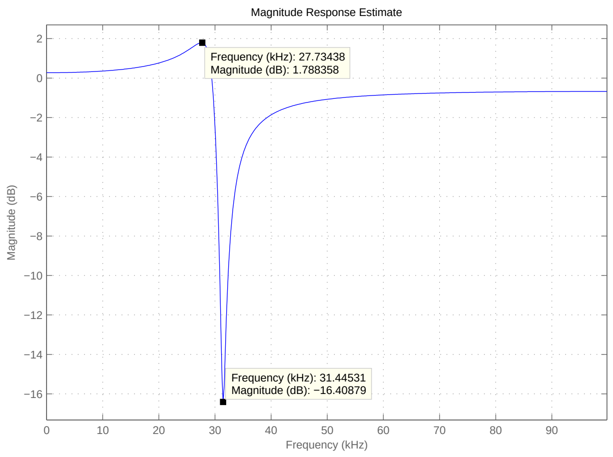
Coefficient Regeneration Approach

This coefficient regeneration approach simply involves recomputing the filter coefficients of affected filters by using the same zero and pole positions at decimated sampling frequency $fs/2^n$ where n is the decimation level (number of decimator stages).

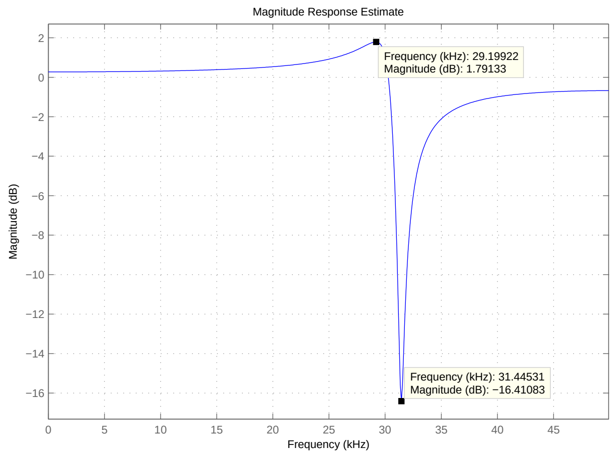
Comparison between Different Methods

Figure 3.8 shows the result of applying two different coefficient modification methods to a lowpass filter of the original pole and zero at 28 kHz and 31.4 kHz. As the cutoff frequency is used as the reference point for spectral transformation mapping, the cutoff frequency can be restored by using any of the two modification methods. However, Figure 3.8b shows that the new position of the pole is moved and therefore the spectral transformation cannot maintain the pole as original.

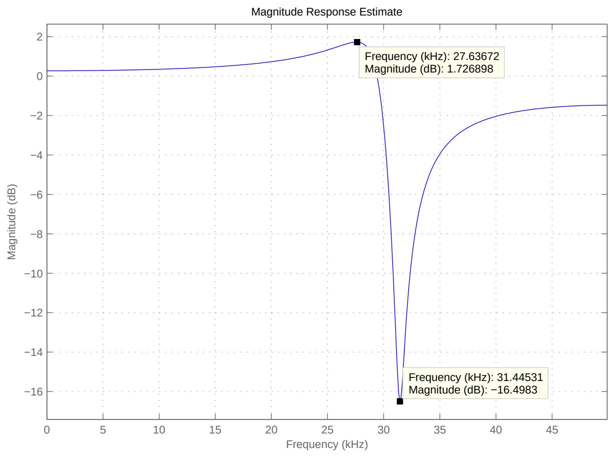
Although the spectral transformation is a simpler modification method and only involves two matrix multiplications, it can only map either the passband edge (pass frequency) or the stopband edge (cutoff frequency) of the original lowpass filter to that of the desired lowpass filter, but not both. Only the zeros or poles of the affected filters can be restored in the spectral transformation while the coefficient regeneration approach can restore both the pass frequency and the cutoff frequency as it is based on re-designing the filter. Consequently, the coefficient regeneration method is chosen to modify the coefficients of affected filters. Figure 3.9 shows the frequency responses of a decimated cochlea example with 169 stages, sampling frequency of 200 kHz and using a 50 dB decimation cutoff value.



(a)



(b)



(c)

Figure 3.8: Comparison of different coefficient modification methods: (a) Original frequency response; (b) Modify by spectral transformation approach; (c) Modify by coefficient regeneration approach.

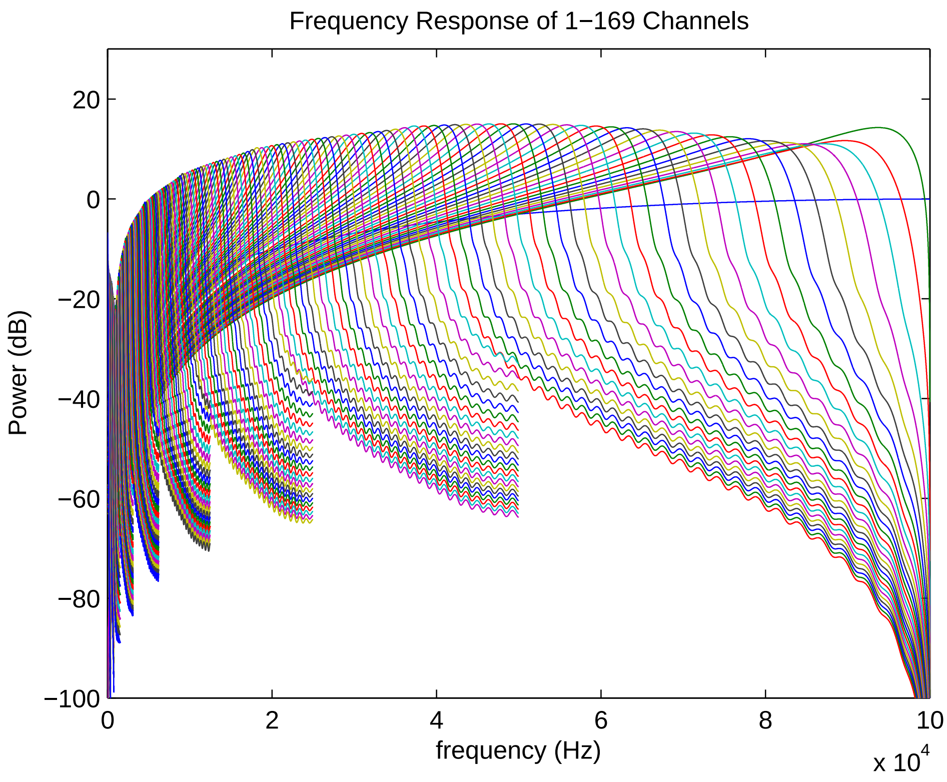


Figure 3.9: Frequency responses of the decimated electronic cochlea with 169 stages, sampling frequency of 200 kHz and decimation cutoff value of 50 dB.

3.4 Summary

In this chapter, the Lyon and Mead's cochlea model was introduced and in order to avoid redundant computations in the low frequency sections of the cochlea model, decimation is applied. Furthermore, the tradeoff among aliasing and redundant filter computation reduction was discussed. A comparison between two coefficient modification methods: spectral transformation and coefficient regeneration was also shown. Although the spectral transformation is simpler, it cannot map both the zeros and poles of filters as original. The filter coefficients, therefore, would not be transformed but regenerated instead after decimation.

Chapter 4

System Architecture

4.1 Introduction

In this chapter, the hardware platform and tools used to develop the electronic cochlea system is introduced. The implementation of electronic cochlea system and its architecture are also detailed.

At the beginning of this chapter, the computer-aided design (CAD) tools and the hardware platform that used to implement the cochlea are introduced. In next section, the implementations of the cochlea is discussed and followed by the pipelining processing scheme and decimation implementation in the system. Then the multiple core version of the cochlea is demonstrated. Finally, the architecture of DFX filter computation core of the cochlea is introduced in this chapter.

4.2 Hardware Platform and CAD Tools

The electronic cochlea was developed using the Very High Speed Integrated Circuit Hardware Description Language (VHDL) [Ska96] which is a language that is used to describe hardware architectures. One of the advantages of using VHDL is that simulation can be done before the real hardware is implemented.

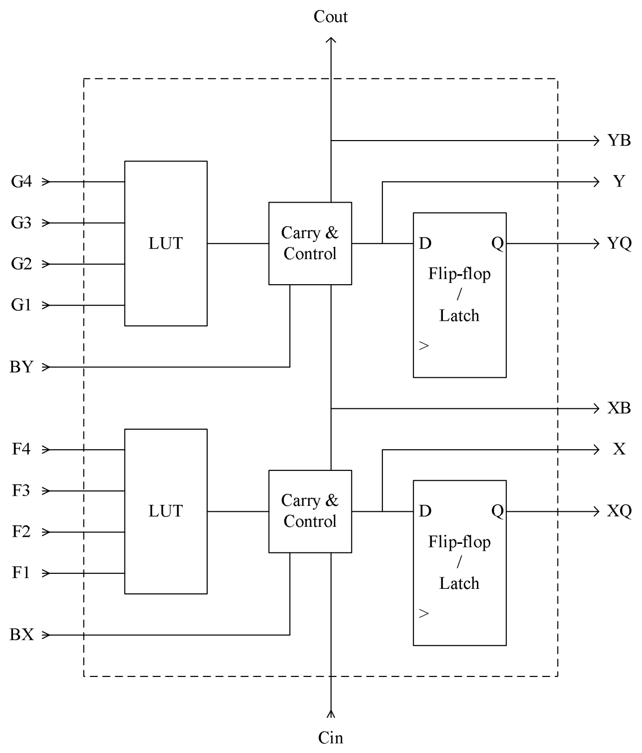


Figure 4.1: The simplified structure of Xilinx slice.

A Field Programmable Gate Array (FPGA) was chosen to be the implementation platform. An FPGA is an integrated circuit (IC) that is programmable after manufacture. FPGAs are reconfigurable hardware that can be reprogrammed via a software downloadable bitstream. They provide the advantage of a short turn around time and low cost and are widely used as a prototype before fabricating a VLSI design or used directly in a product.

The hardware platform that was used to implement the cochlea system is Alpha-data ADM-XP board. It is a PCI board containing the Xilinx Virtex-II Pro 2VP100 FPGA element and 64 MB DDR SDRAMs.

The basic building block of the Virtex-II Pro FPGA is the slice. A Virtex-II Pro slice consists of two 4-input function generators, carry logic and two storage elements (Figure 4.1). As shown in Figure 4.2, each Virtex-II Pro CLB contains 4 slices. The 4-input function generators are implemented as 4-input look-up tables (LUTs). Each of them can be programmed as a function of a

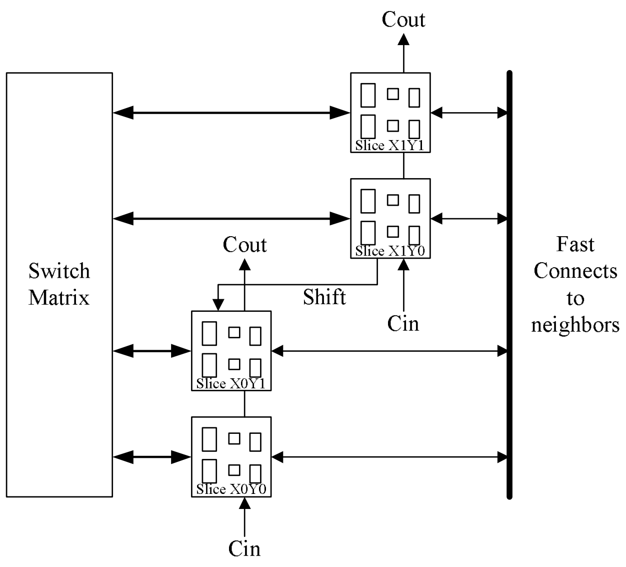


Figure 4.2: Xilinx Virtex-II Pro CLB structure.

4-input LUT, a 16-bit distributed SelectRAM memory or a 16-bit variable-tap shift register [Xil05].

The Virtex-II Pro chip is dedicated large amounts of large blocks of special memories, called BlockRAMs. Each BlockRAM has 18-kbit storage and can be configured to single-port or dual-port mode. The ports can be configured in width from 1 to 36 bits [Xil05].

Although multipliers can be constructed by combining a large amount of LUTs, the constructed multipliers have low performance. The Virtex-II Pro devices provide many embedded multiplier blocks. These multipliers have 18-bit wide inputs and the outputs are 36 bits. They can perform high-speed operations as the 36-bit product is calculated in a single cycle [Xil05]. These were used in digital signal processing systems for improving the performance.

The Xilinx Virtex-II Pro 2VP100 has 7992 kbits of BlockRAM (arranged as 444$\times$18-kbit blocks), 88192 4-input look-up tables (44096 slices) and 444 18-bit-$\times$-18-bit multiplier blocks.

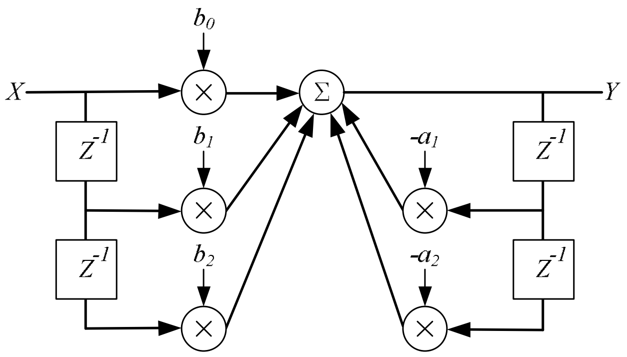


Figure 4.3: The architecture of a second order IIR filter.

4.3 Sequential Processing Electronic Cochlea

The basic architecture of a second order IIR filter is shown in Figure 4.3 which implements the filter with transfer function:

$$H[z] = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

The parallel architecture of Figure 1.1 can be used only if the FPGA device has sufficient resources (multipliers in particular) to perform all the operations in parallel. For a cochlea system with S stages, $5 \times S$ multipliers are required for a parallel implementation.

If insufficient resources are available, or if maximum throughput is not the priority, a sequential approach can be used. For such a design, only 5 multipliers are required. Figure 4.4 shows the architecture of a pipelined sequential processing electronic cochlea which uses 5 dual-port BlockRAMs for storing intermediate signals: $x[t]$, $x[t-1]$, $x[t-2]$, $y[t-1]$ and $y[t-2]$. For every cycles, the intermediate signals are read from the BlockRAMs and the calculated result is later written back to the BlockRAMs. However, even if the IIR filter is fully pipelined and has a latency of k, there would be $k-1$ idle cycles between the signal computations of time sample t in stage $n-1$ and stage n as there are data dependencies between successive stages (the input x of time

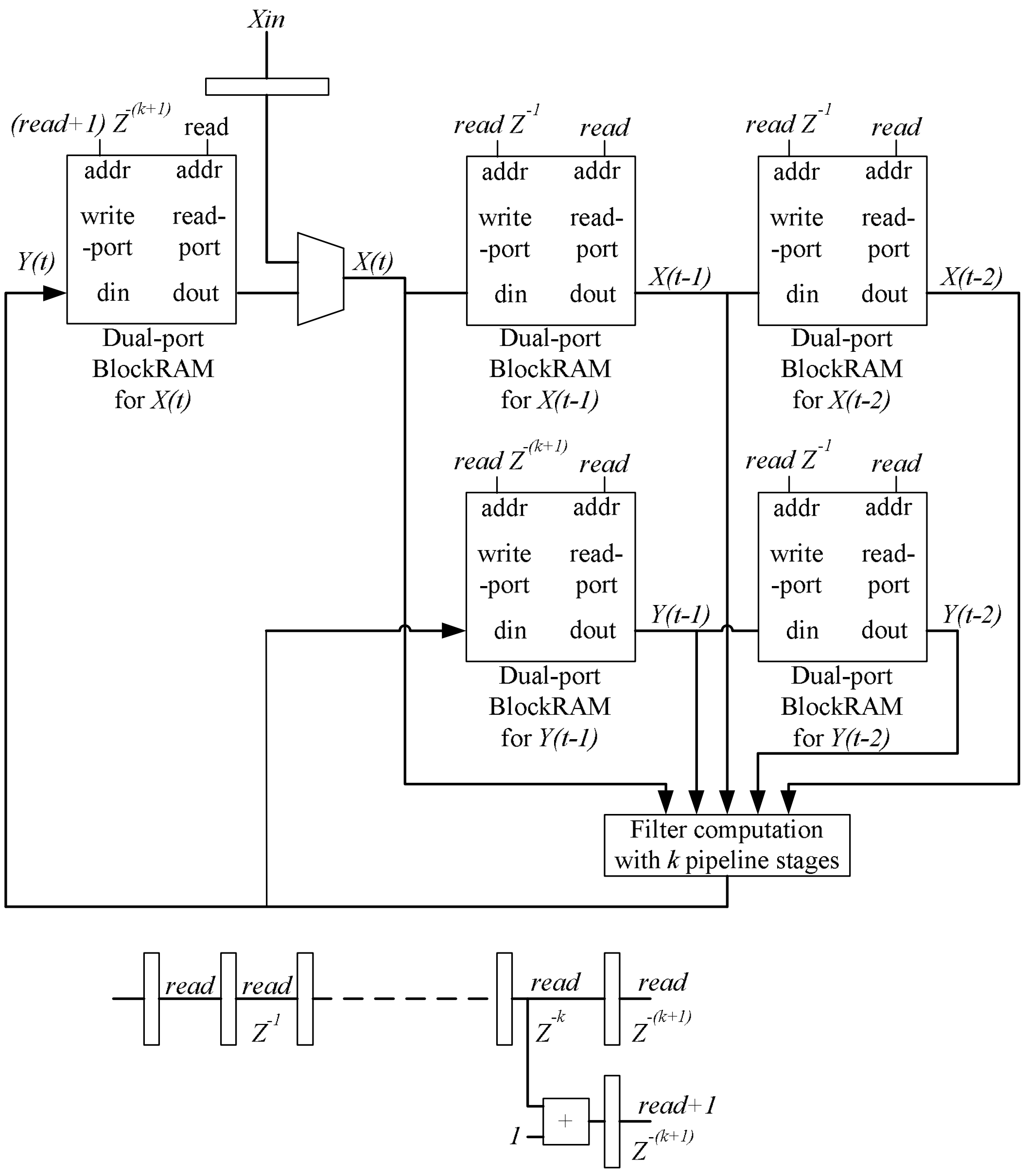

Xin
(read+1) Z^-(k+1)
read
addr
addr
write -port
read- port
din
dout
Y(t)
X(t)
Dual-port BlockRAM for X(t)
read Z^-1
read
X(t-1)
Dual-port BlockRAM for X(t-1)
X(t-2)
Dual-port BlockRAM for X(t-2)
read Z^-(k+1)
Y(t-1)
Dual-port BlockRAM for Y(t-1)
Y(t-2)
Dual-port BlockRAM for Y(t-2)
Filter computation with k pipeline stages
Z^-1
Z^-k
Z^-(k+1)
read+1
1
+

Figure 4.4: Block diagram of the fully pipelined sequential processing electronic cochlea.

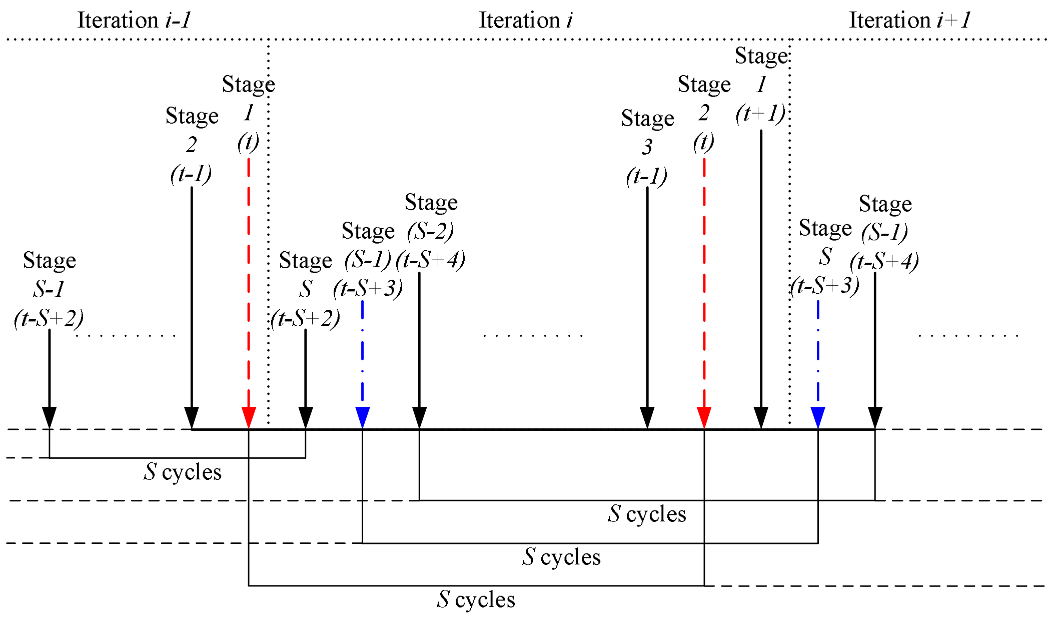


Figure 4.5: The interleaving scheme for the pipelined electronic cochlea.

t of stage n is the output y of time t of stage $n-1$). In order to avoid these idle cycles, an interleaving scheme is implemented in the sequential processing electronic cochlea so that a new computation can be started every cycle.

4.3.1 Pipelining - An Interleaving Scheme

Figure 4.5 shows an interleaving scheme for the sequential processing electronic cochlea. It expands the time between computations of stage $n-1$ and stage n of the same time sample t, to S cycles, where S is the total number of stages. Since $S \geq k-1$, the dependencies are met, and computations of the previous samples of other stages are interleaved. This scheme avoids idle cycles and make the cochlea system to be fully pipelined. It requires a total of S cycles to process the entire cochlea filter.

4.3.2 Decimation in Sequential Processing Electronic Cochlea

When decimation is applied to the sequential processing cochlea, the computation schedule is further modified as computation is not required for all of the stages in each sample. Let G_i be the number of stages operating at the decimated sampling frequency $f_s/2^i$, i.e. there are G_0 stages operating at f_s and G_1 stages at $f_s/2$. That means:

G_0 stages are calculated every sample.
G_1 stages are calculated every 2 samples.
G_2 stages are calculated every 4 samples.
G_3 stages are calculated every 8 samples.
$\vdots$
G_n stages are calculated every 2^n samples.
(where n is the maximum decimation level).

In other words, the calculations for the first 2^n samples can be represented as shown in Table 4.1.

G_i stages are calculated at sample k when $k \mod 2^i = 0$. Let $h = (k \mod 2^n) + 1$. The evaluation of calculation stages at sample k is similar to a priority encoder which determines the bit position of the least significant set-bit of h.

For example:

For the 36th sample, as the value of h is $100100_2 = 36$, the least significant set-bit is the bit of value 2^2. Stages $1 \rightarrow (G_0 + G_1 + G_2)$ are processed in this iteration.

For the 48th sample, h is 110000_2. The least significant set-bit is at the position of 2^4 and therefore stages $1 \rightarrow \sum_{k=0}^{4} G_k$ are calculated

Sample $k \pmod{2^n}$	$h = k + 1$	Stages for Calculation	Number of Stages
0	1	$1 \rightarrow G_0$	G_0
1	2	$1 \rightarrow (G_0 + G_1)$	$G_0 + G_1$
2	3	$1 \rightarrow G_0$	G_0
3	4	$1 \rightarrow (G_0 + G_1 + G_2)$	$G_0 + G_1 + G_2$
4	5	$1 \rightarrow G_0$	G_0
5	6	$1 \rightarrow (G_0 + G_1)$	$G_0 + G_1$
6	7	$1 \rightarrow G_0$	G_0
7	8	$1 \rightarrow (G_0 + G_1 + G_2 + G_3)$	$G_0 + G_1 + G_2 + G_3$
8	9	$1 \rightarrow G_0$	G_0
9	10	$1 \rightarrow (G_0 + G_1)$	$G_0 + G_1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2^n - 1$	2^n	$1 \rightarrow (\sum_{k=0}^{n} G_k)$	$\sum_{k=0}^{n} G_k$

Table 4.1: Calculation stages for different samples.

in this iteration.

4.3.3 Multiple Sequential Cores

A parallel electronic cochlea can be implemented by using multiple numbers of sequential cochlea cores described in Figure 4.4. If P cochlea blocks are used to implement an electronic cochlea system, they are interconnected as shown in Figure 4.6. The schedules of each cochlea block can be modified so that the nth cochlea block, only processes stages i, where $i \mod P = n$. The input signal $x[t]$ of the nth cochlea block is forwarded from the $(n-1)$th cochlea block, which calculates the $y[t]$ signal of the previous stages. Hence, using multiple cores, it has the benefit of increasing the maximum sample rate of the system by a factor of P.

4.3.4 Architecture of the DFX Filter Computation Core

Instead of using the standard dual fixed-point adder and multiplier modules proposed by Ewe et. al. [ECC04], an improved filter computation module,

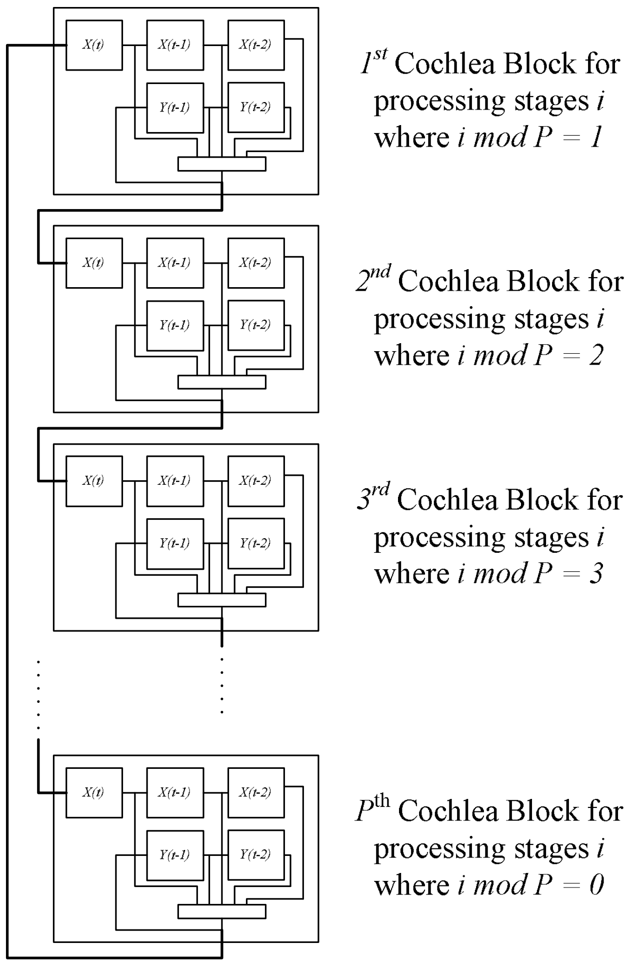

X(t)
X(t-1)
X(t-2)
Y(t-1)
Y(t-2)
1st Cochlea Block for processing stages i where $i \bmod P = 1$
2nd Cochlea Block for processing stages i where $i \bmod P = 2$
3rd Cochlea Block for processing stages i where $i \bmod P = 3$
Pth Cochlea Block for processing stages i where $i \bmod P = 0$

Figure 4.6: Multiple sequential processing cochlea cores.

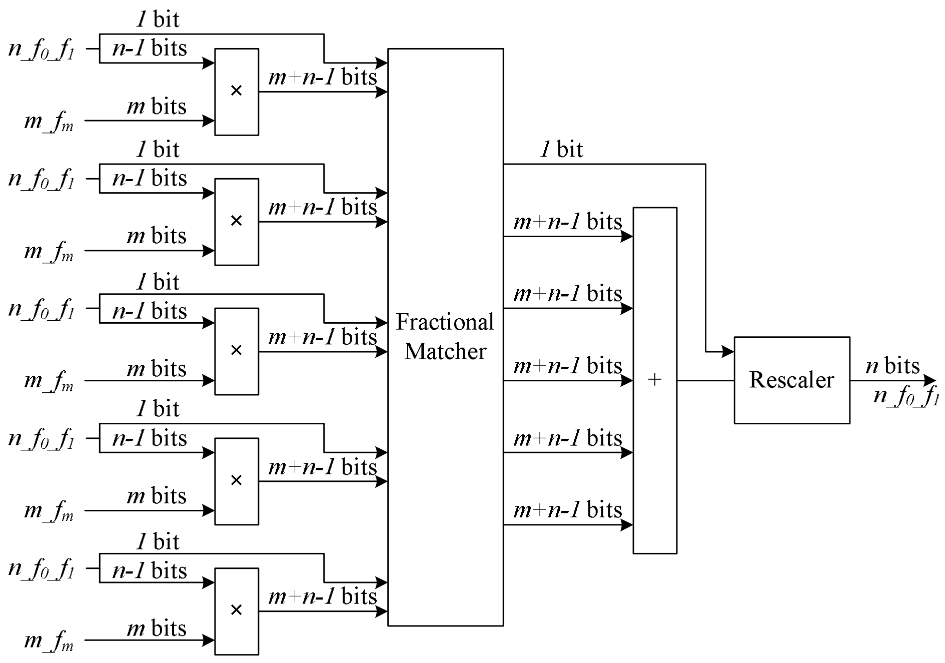


Figure 4.7: Filter computation module for dual fixed-point $n\_f_0\_f_1$.

which is customized for second order IIR filters is used to implement the IIR filter core inside the electronic cochlea.

Figure 4.7 and Figure 4.8 show how the DFX filter computation module is organized. In this filter module, 5 signals in the DFX representation of $n\_f_0\_f_1$ are multiplied with 5 coefficients in the fixed-point representation of m_f_m. The 5 intermediate products are stored in the non-rescaled DFX format of $(n+m)\_(f_0+f_m)\_(f_1+f_m)$. Finally, they are added together and rescaled back to $n\_f_0\_f_1$.

The advantages of using this customized filter module are:

- If the standard DFX adder and multiplier modules were used to construct the filter core, the intermediate products would be rescaled back to the $n\_f_0\_f_1$ format. Further preprocessing normalization and postprocessing rescaling are taken by the standard DFX adders. In the case of 5 products' summation, 4 more normalization and rescaling units would

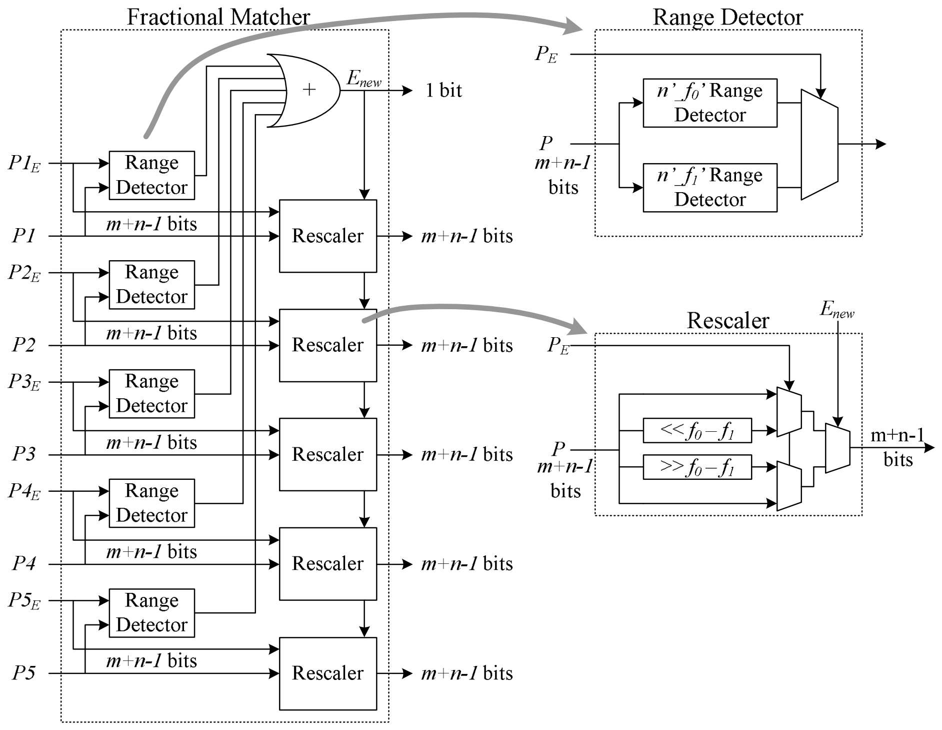


Figure 4.8: Dual fixed-point fractional matcher.

be added to the datapath. Although the customized filter module requires m more storage bits for each intermediate product, it can reduce the hardware resources used by those additional preprocessing normalization and postprocessing rescaling units.

- In some cases, the summation operation in filter calculations may involve adding a very small number, X_{s_0}, in $Scale_0$ scaling with 2 very large $Scale_1$ scaled numbers of the same magnitude but opposite sign, $-Y_{s_1}$ and Y_{s_1}. The addition order would affect the result in these cases if the standard DFX adder modules were used. If the small number X_{s_0} was first added with either of the large numbers, for example Y_{s_1}, and then with another one, $-Y_{s_1}$, the final result would be 0 instead of X_{s_0} as the intermediate sum $(X_{s_0} + Y_{s_1})$ would be rescaled back to $Scale_1$ scaling and the bits containing the value of X_{s_0} would be truncated. If a longer wordlength and more precision (namely a DFX representation

$(n+m)\_(f_0+f_m)\_(f_1+f_m))$ is used for the intermediate sum, this type of error is eliminated.

In the DFX filter, $(n-1)$-bit-$\times$-m-bit fixed-point multipliers and $(m+n-1)$-bit fixed-point adders are used. Compared with a fixed-point core, the additional blocks required to implement DFX are range detectors, a fractional matcher and a rescaler as illustrated in Figure 4.7 and Figure 4.8.

Range Detector

Range detectors are used to determine the exponent bit, E, of the input. If the value of the input is in the range of $Scale_0$, all the bits above the boundary, $B = 2^{(n-2-f_0)}$, would be all 0's if the input is positive, or would be all 1's if the input is negative [ECC04]. Therefore, the output Boolean expression of E is:

$$E = \overline{(b_{n_{in}-1} \cdot b_{n_{in}-2} \cdot \ldots \cdot b_{f_{in}+(n-2-f_0)} + \overline{b_{n_{in}-1}} \cdot \overline{b_{n_{in}-1}} \cdot \ldots \cdot \overline{b_{f_{in}+(n-2-f_0)}})} \tag{4.1}$$

where n_{in} and f_{in} are the wordlength and the number of bits used in the fractional part.

Fractional Matcher

The fractional matcher is used to normalize the products to the same exponent. It first generates the correct exponent bits for each product by using the range detectors. All the products are then normalized by the rescalers in order to have the same new exponent, E_{new}. The new exponent, E_{new}, is generated from the logic OR operation of all the correct product's exponent bits as if one of the products should be in the $Scale_1$ range, all the products should be rescaled to $Scale_1$ scaling.

Rescaler

Rescalers are used to change the scaling of the DFX number from $Scale_0$ to $Scale_1$ or reverse. The table 4.2 shows the operations for different situations.

Original Exponent	Target Exponent	Explanation	Operation
0	0	$Scale_0 \rightarrow Scale_0$	no change
0	1	$Scale_0 \rightarrow Scale_1$	shifting right by $(f_0 - f_1)$ bits
1	0	$Scale_1 \rightarrow Scale_0$	shifting left by $(f_0 - f_1)$ bits
1	1	$Scale_1 \rightarrow Scale_1$	no change

Table 4.2: Rescaling operations for different situations.

4.4 Summary

In this chapter, an overview of hardware platform used in this work was described. The architecture of a sequential processing electronic cochlea was also detailed. The cochlea was developed on a reconfigurable hardware computing platform which contains a Xilinx Virtex-II Pro FPGA. By using the interleaving scheme, the idle cycles during the computation of successive stages can be avoided. In order to apply decimation, the computation stages for each iteration are different. Besides fixed-point arithmetic, the cochlea was implemented with the dual fixed-point arithmetic. The additional blocks used in implementing the DFX second order IIR filter were given to be range detectors, a fractional matcher and a rescaler.

Chapter 5

Experimental Results

5.1 Introduction

In this chapter, results obtained from the cochlea system with different arithmetic are presented. Firstly, the testing environment and the analyzing tools are introduced. It is followed by performance measurements of the cochlea in different designs. Finally, comparisons between cochlea designs for different arithmetic and different decimation are presented.

5.2 Testing Environment

The sequential processing electronic cochlea was verified using the ModelSim Se 6.1d simulator, and was synthesized using Xilinx ISE 8.1i, with Xilinx Virtex-II Pro XC2VP100-6 as the target device. The results were analyzed in Matlab 7.2.

Several designs of two versions of cochleas that consist of 88 stages and 169 stages respectively were implemented in order to show the comparison between fixed-point and dual fixed-point implementations. A random signal was used as input testcase since it has energy at all frequencies. The results of different implementations were analyzed in terms of signal to noise ratio (SNR).

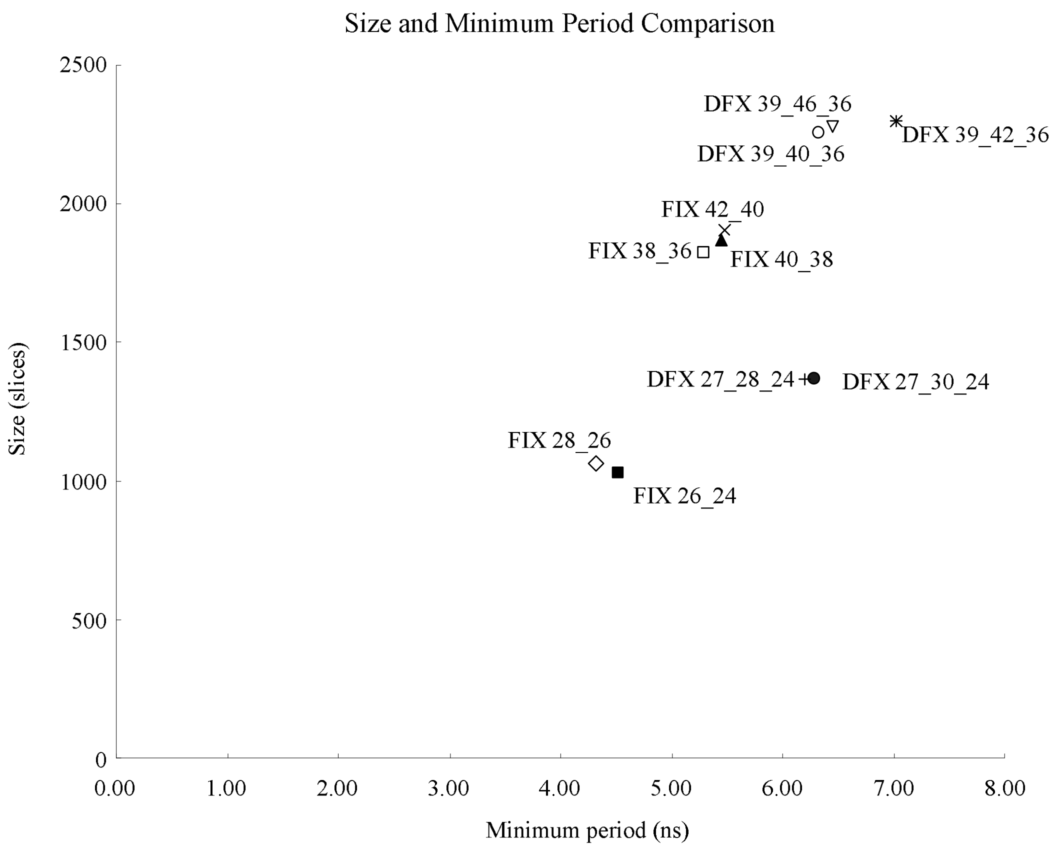


Figure 5.1: Size and timing comparison of different designs. (FIX x_y refers to a fixed-point system with wordlength x bits and fractional wordlength y bits. DFX x_y_z is explained in Section 2.7)

Arithmetic	Design	Size (slices)	Minimum Period (ns)
Fixed-point	FIX 26_24	1034	4.526
	FIX 28_26	1067	4.352
Dual Fixed-point	DFX 27_28_24	1366	6.214
	DFX 27_30_24	1370	6.279
Fixed-point	FIX 38_36	1826	5.293
	FIX 40_38	1862	5.442
	FIX 42_40	1900	5.482
Dual Fixed-point	DFX 39_40_36	2276	6.342
	DFX 39_42_36	2282	7.027
	DFX 39_46_36	2281	6.385

Table 5.1: Size and timing comparison of different designs.

5.3 Performance of the Sequential Electronic Cochlea

Figure 5.1 and Table 5.1 show the size and performance of different cochlea designs. The maximum processing rate of different designs can be calculated from their minimum period:

$$MaxProcessingRate = \frac{1}{NumberofStages \cdot MinimumPeriod} \tag{5.1}$$

5.3.1 Comparisons

Besides comparing the resource usage and timing, the results' accuracy of different implementations were compared. The SNR of a system's output

Design	Length of Cascade	Number of Pipeline Stages	BlockRAM (18-kbit)	Block Multiplier (18-bit-×-18-bit)
FIX 26_24	88	8	22	20
FIX 28_26				
DFX 27_28_24		10		
DFX 27_30_24				
FIX 38_36	169	9	28	30
FIX 40_38				
FIX 42_40				
DFX 39_40_36		11		
DFX 39_42_36				
DFX 39_46_36				

Table 5.2: Multipliers and BlockRAMs usage for different designs.

signal, $a[t]$, to the reference signal, $s[t]$, is calculated as:

$$\begin{aligned} noise[t] &= a[t] - s[t] && (5.2) \\ SNR &= 20\log\left[\frac{\frac{\sum_{t=1}^{N} s[t]^2}{N}}{\frac{\sum_{t=1}^{N} noise[t]^2}{N}}\right] \\ &= 20\log\left[\frac{\sum_{t=1}^{N} s[t]^2}{\sum_{t=1}^{N}(a[t]-s[t])^2}\right] \\ &= 20\left[\log\Big(\sum_{t=1}^{N} s[t]^2\Big) - \log\Big(\sum_{t=1}^{N}(a[t]-s[t])^2\Big)\right] && (5.3) \end{aligned}$$

Fixed-point arithmetic vs. Dual Fixed-point arithmetic

The resulting SNR of different implementations are shown in Figure 5.2 and Figure 5.3. The SNR calculated in this work is with reference to double precision floating-point results. The relative SNR is calculated as the difference in SNR compared with a double precision electronic cochlea that uses fixed-point coefficients.

As the wordlength is increased, the SNR of the fixed-point designs (FIX) increase as expected. For implementations with 88 cascaded filter stages, DFX

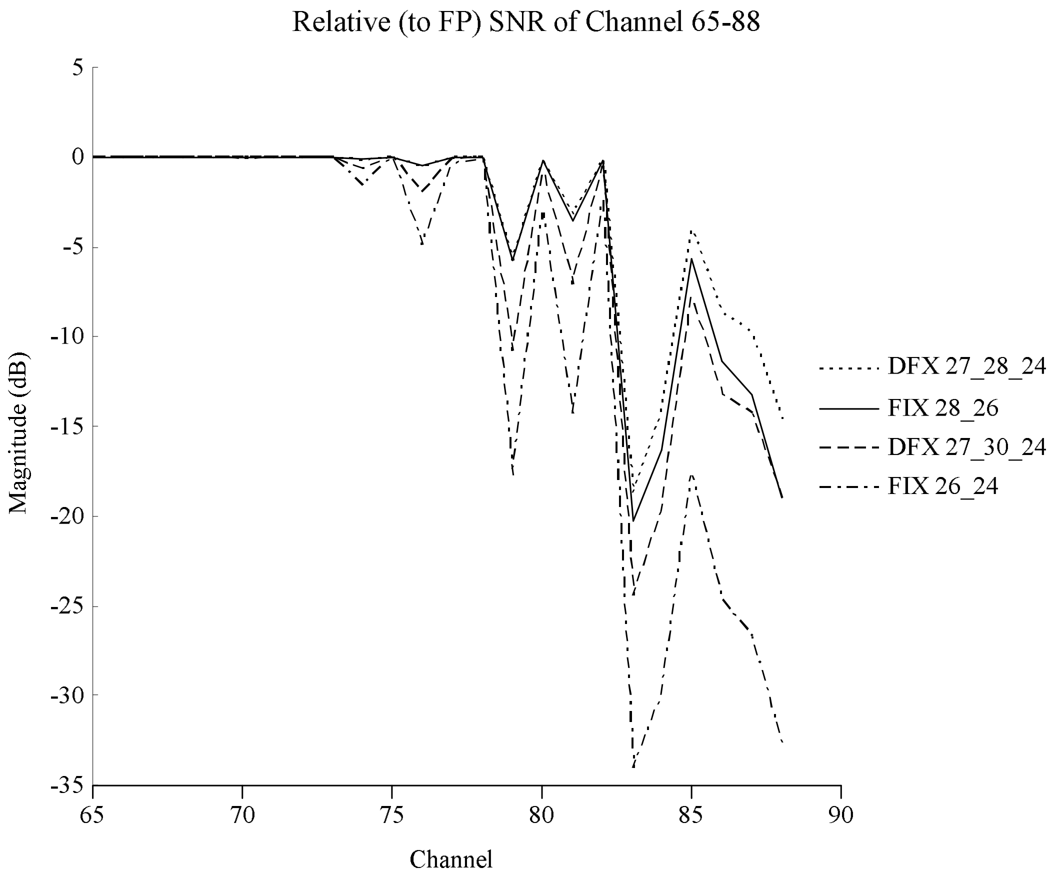


Figure 5.2: Relative SNR of 88 cascaded filters systems of different arithmetic and wordlength.

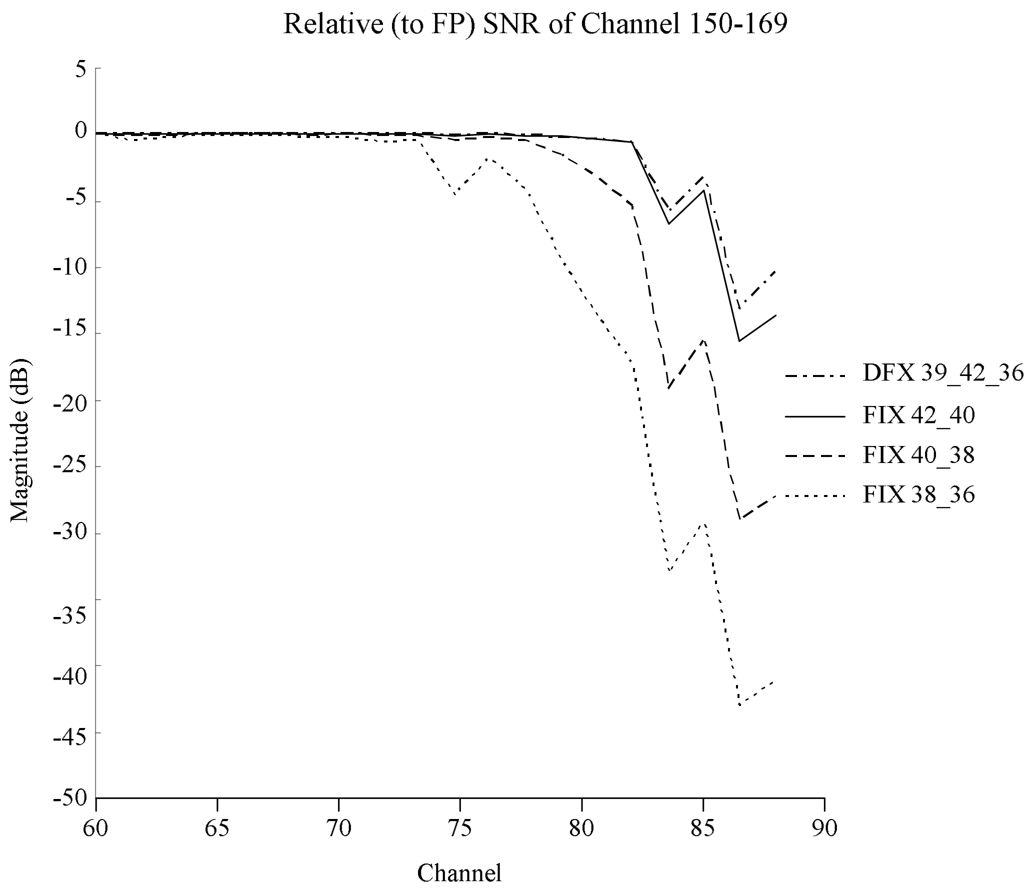


Figure 5.3: Relative SNR of 169 cascaded filters systems of different arithmetic and wordlength.

Arithmetic	Design	Maximum Processing Rate (kHz)
Fixed-point	FIX 26_24	2510
	FIX 28_26	2611
Dual Fixed-point	DFX 27_28_24	1828
	DFX 27_30_24	1809
Fixed-point	FIX 38_36	1117
	FIX 40_38	1087
	FIX 42_40	1079
Dual Fixed-point	DFX 39_40_36	933
	DFX 39_42_36	842
	DFX 39_46_36	926

Table 5.3: Maximum processing rate for different designs.

27_28_24 has an SNR which is 20 dB better than the FIX 26_24 implementation at the same effective wordlength (Figure 5.2). For 169 stages, the SNR of the DFX 39_42_36 is 30 dB higher than the FIX 38_36 which uses the same numbers of multipliers and adders (Figure 5.3). DFX 39_42_36 also has a better SNR than FIX 40_38 in which the effective wordlength is larger than that of DFX 39_42_36.

The resource overhead for the DFX implementation is approximately 450 slices, this being mainly for registers in the additional 2 pipeline stages in the filter computation module.

From Figure 5.3, the FIX 42_40 and DFX 39_42_36 designs have similar SNR. The area-speed product of the DFX 39_42_36 implementation is 16035 and FIX 42_40 is 10415, so if the design is constrained by logic resources, the fixed-point implementation may be advantageous. Both designs used the same number of 18-bit-$\times$-18-bit multipliers and so if that is the main resource constraint, DFX offers better SNR.

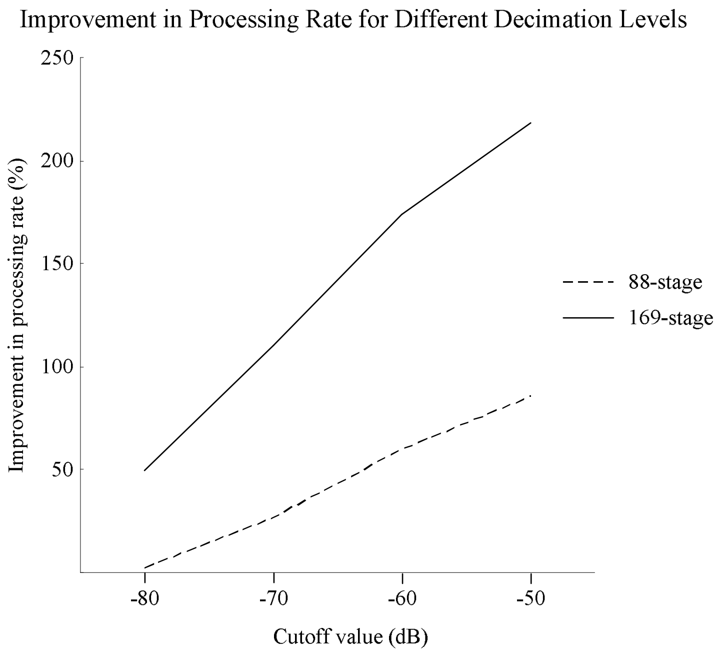


Figure 5.4: Processing rate as a function of decimation level.

Different Decimation Designs

Figure 5.4 shows how the decimation can affect the maximum processing rate for different designs. By using decimation, the maximum processing rate of each design can be increased. For decimation with a cutoff value of -80 dB, the maximum processing rate of the 88-stage systems and the 169-stage systems are increased by 1.7% and 49% respectively. If cutoff value is chosen to be -70 dB, higher levels of decimation can be used and the maximum processing rates are 26% and 110% higher than the non-decimated 88-stage and 169-stage systems respectively. The 88-stage systems can increase the maximum processing rate by 59% and 85% for decimation with cutoff values of -60 dB and -50 dB respectively. The corresponding increase for 169-stage systems are 173% and 218%.

5.4 Summary

In this chapter, the performance of the cochlea in different designs were presented. Also, the comparisons between fixed-point cochlea implementations and the cochleas using dual fixed-point arithmetic were discussed in term of resource usage, timing and SNR.

The dual fixed-point cochlea implementation can achieve a better SNR than the fixed-point cochlea that has the same effective wordlength and used the same number of multipliers. Although decimation cannot reduce the resource usage of the sequential processing electronic cochlea, it can be used to increase the maximum processing rate of the system.

Chapter 6

Conclusions

In this work, a novel architecture for a sequential processing electronic cochlea using decimation and dual fixed-point arithmetic was developed. Through design examples, it showed that decimation can be used to reduce computations in the low frequency sections of the Lyon and Mead's cochlea model. This work illustrated that the dual fixed-point arithmetic can be applied to the cochlea in order to provide more precision results than fixed-point arithmetic. Several problems were addressed as follows:

Aliasing and Computation Tradeoff

For both parallel and sequential implementations of the cochlea, the computation in low frequency sections was reduced by employing decimation. Decimation can provide a higher processing rate to the system but causes aliasing. For the 169-stage system decimated with cutoff values of -50 dB, -60 dB, -70 dB and -80 dB, the maximum processing rate was increased by 218%, 173%, 110% and 49% respectively.

It is important to study tradeoffs between aliasing and computation reduction. Designers always want to increase the processing rate of the system to achieve the highest performance in the given implementation platform. Using decimated designs with different cutoff values, designers can achieve higher processing rates by defining a suitable cutoff value based on precision constraints.

More Accurate Dual Fixed-point Filter

An electronic cochlea utilizing a dual fixed-point second order IIR filter computation core was implemented. A DFX design presented in this thesis had a SNR which is 20-30 dB better than the fixed-point design at the same wordlength. The resource overhead for the DFX implementation was over 400 slices but the number of multipliers used in both implementations were the same.

At the same wordlength, DFX filter cores used more logic blocks than the fixed-point, however, the main resource constraint should be the number of multipliers since filter operations in digital signal processing are always in sum-of-product form. Multipliers in an FPGA chip are more fewer than logic blocks. Therefore for multiplier constrained designs, the resource usage of DFX and fixed-point designs are the same. On the other hand, DFX designs can offer more accurate results than fixed-point designs due to the improved dynamic range of DFX arithmetic. As a result, the quality advantage of DFX designs is more significant than their resource overhead as both designs used the same number of multipliers.

Larger Signal Bandwidth

The need for high quality and efficient electronic cochleas are important and a family of implementations is useful since the electronic cochlea serves as the front-end signal analyzer in auditory systems serving different purposes. In the design presented, decimation was used to increase the processing rate and as the signal bandwidth is limited by the processing rate, decimated designs can be used in systems that require larger signal bandwidth. An example is in modeling bat echolocation systems as they use frequencies in the range of 20-200 kHz [DZM95]. Moreover, as the electronic cochlea presented in this work used a sequential processing architecture, any member of channels can be implemented with a fixed amount of hardware. This is in contrast to parallel designs where resources are propositional to the number of channels. The sequential design presented can also be accelerated by using multiple cores.

6.1 Future Work

The electronic cochlea presented in this thesis can be used as the front-end signal processing stage for auditory analyzing systems such as speech recognition and bat echolocation. As the cochlea only used a small number of logic gates in an FPGA, it is feasible to implement entire auditory applications in a single chip.

Decimation and DFX arithmetic are general tools that can be applied to many other signal processing systems. Opportunities are likely to exist for their use in many similar applications.

Bibliography

[AvSV97] E. Fragnière A. van Schaik and E. Vittoz. Improved silicon cochlea using compatible lateral bipolar transistors. In *Advances in Neural Information Processing Systems*, volume 8, 1997.

[CR81] R. E. Crochiere and L. R. Rabiner. Interpolation and Decimation of Digital Signals - A Tutorial Review. In *Proc. of IEEE*, volume 69, pages 300–331, March 1981.

[DZM95] Itiel E. Dror, Mark Zagaeski, and Cynthia F. Moss. Three-Dimensional target recognition via sonar: A neural network model. *Neural Networks*, 8(1):149–160, 1995.

[ECC04] Chun Te Ewe, Peter Y. K. Cheung, and George A. Constantinides. Dual Fixed-Point: An Efficient Alternative to Floating-Point Computation. In *Field-Programmable Logic and Applications 2004*, pages 200–208, 2004.

[LJL03] M. P. Leong, Craig T. Jin, and Philip H. W. Leong. An FPGA-based Electronic Cochlea. *EURASIP Journal on Applied Signal Processing*, (7):629–638, 2003.

[LM88] R. F. Lyon and C. Mead. An analog electronic cochlea. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 36(7):1119–1134, 1988.

[LM89a] J. P. Lazzaro and C. Mead. Silicon models of auditory localization. *Neural Computation*, 1:47–57, Spring 1989.

[LM89b] J. P. Lazzaro and C. Mead. Silicon models of pitch perception. In *Proc. National Academy of Sciences*, volume 86, pages 9597–9601, 1989.

[LWK94] J. P. Lazzaro, J. Wawrzynek, and A. Kramer. Systems technologies for silicon auditory models. *IEEE Micro*, 14(3):7–15, 1994.

[LWL97] J. P. Lazzaro, J. Wawrzynek, and R. P. Lippmann. Micro power analog circuit implementation of hidden Markov model state decoding. *IEEE Journal Solid State Circuits*, 32(8):1200–1209, 1997.

[LWM92] R. F. Lyon L. Watts, D. A. Kerns and C. A. Mead. Improved implementation of the silicon cochlea. In *IEEE Journal Solid State Circuits*, volume 27, pages 692–700, 1992.

[Lyo91] R. F. Lyon. Analog implementations of auditory models. In *Proc. DARPA Workshop on Speech and Natural Language*, February 1991.

[Lyo01] Richard G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall PTR, 2001.

[MAL91] C. A. Mead, X. Arreguit, and J. P. Lazzaro. Analog VLSI model of binaural hearing. *IEEE Transactions on Neural Networks*, 2(2):230–236, 1991.

[MBK98] A. Schwarz B. Mertsching M. Hansen M. Brucke, W. Nebel and B. Kollmeier. Digital VLSI-implementation of a psychoacoustically and physiologically motivated speech preprocessor. In *Proc. NATO Advanced Study Institute on Computational Hearing*, pages 157–162, 1998.

[MH02] A. Mishra and A. E. Hubbard. A cochlear filter implemented with a field-programmable gate array. In *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, number 1, pages 54–60, 2002.

[Mit01] Sanjit K. Mitra. *Digital Signal Processing A Computer-based Approach*, chapter 7, pages 441–446. McGraw-Hill, 2001.

[OSB98] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time signal processing.* Prentice Hall, Upper Saddle River, New Jersey, second edition, 1998.

[Pic88] James O. Pickles. *An Introduction to the Physiology of Hearing.* Academic Press, London, UK, second edition, 1988.

[SCLJ97] A. R. Temple S. C. Lim and S. Jones. VHDL-based design of biologically inspired pitch detection system. In *Proc. IEEE International Conference on Neural Network*, volume 2, pages 922–927, June 1997.

[Ska96] Kevin Skahill. *VHDL for Programmable Logic.* Prentice Hall, 1996.

[SL92] C. D. Summerfield and R. F. Lyon. ASIC implementation of the Lyon cochlea model. In *Proc. IEEE International Conference Acoustics, Speech, Signal Processing*, pages 673–676, March 1992.

[Sla88] Malcolm Slaney. Lyon's Cochlear Model. Apple Technical Report 13, Advanced Technology Group, Apple Computer, 1988.

[Sla98] Malcolm Slaney. Auditory Toolbox: A Matlab Toolbox for Auditory Modeling Work. Technical Report 010, Interval Research Coporation, Palo Alto, Califm, USA, 1998. Version 2.

[SS05] Rahul Sarpeshkar and C. Salthouse. An Ultra-low-power Programmable Analog Bionic Ear Processor. In *IEEE Transactions on Biomedical Engineering*, volume 52, pages 711–727, 2005.

[vSM99] A. van Schaik and R. Meddis. Analog very large-scale integrated (VLSI) implementation of a model of amplitude modulation sensitivity in the auditory brainstem. *Journal of the Acoustical Society of America*, 105(2):811–821, 1999.

[Xil05] Xilinx. *Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet*, 4.5 edition, October 2005.

Publications

Full length conference paper

C.K. Wong and P.H.W. Leong: An FPGA-based Electronic Cochlea with Dual Fixed-point Arithmetic, Proceedings of the Eleventh International Workshop on Field Programmable Logic and Applications (FPL'06), Tampere 2006