INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

**DESIGN AND IMPLEMENTATION
OF A CASCADED INTEGRATOR
COMB (CIC) DECIMATION FILTER**

by

Harry Yang, B.Eng. (Tsinghua University)

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the
requirements for the degree of

Master of Engineering

Ottawa-Carleton Institute of Electrical Engineering
Department of Electronics
Faculty of Engineering
Carleton University
Ottawa, Ontario, Canada
April 2001

© Copyright 2001

**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-61029-2

Canada

The undersigned recommend to the Faculty of Graduate Studies and
Research acceptance of the thesis

**DESIGN AND IMPLEMENTATION
OF A CASCADED INTEGRATOR
COMB (CIC) DECIMATION FILTER**

Submitted by Harry Yang, B.Eng. in partial fulfillment of the
requirements for the degree of Master of Engineering

Thesis Supervisor

Thesis Co-supervisor

Chair, Department of Electronics

CARLETON UNIVERSITY
April 2001

Abstract

The design of an efficient configurable digital decimation filter is presented in this work. This filter is called the Cascaded Integrator-Comb (CIC) filter. It is composed of an equal number of integrator stages and comb stages. It was first presented by Eugene Hogenauer in 1981. It is an economical structure compared with other decimation filter implementations in terms of multipliers and storage.

In this thesis, the basic principles of the CIC decimation filter are presented in detail. A functional model of the CIC decimation filter is developed using the Matlab program. This model is composed of five integrator-comb pairs, and it is based on a floating point mathematical algorithm. A series of tests are carried out on this model. Simulation results show that the CIC decimation filter is a low-pass filter, and its frequency response depends on the number of integrator-comb stages N . As N increases, the CIC decimator has greater stop band attenuation but a less flat pass band. Simulation results conform to the principle analysis of the CIC decimation filter.

A configurable CIC decimation filter is implemented in the VHDL hardware description language. It is also composed of five integrator-comb pairs. "Configurable" means that the maximum decimation ratio and input/output word lengths can be programmed before implementing the code into hardware. This hardware implementation is optimized to fit the configuration defined by a specification. After the hardware implementation, one has the capability of programming the decimation ratio from one to the maximum decimation ratio while running the decimation filter. Simulation results show that the performance of the CIC implementation conforms to that of the functional model.

In order to improve the throughput of the CIC decimation filter, a pipelined structure is adopted in this design. Critical path analysis shows that the CIC decimation filter throughput is determined by the first integrator stage that causes more propagation delay than any other stage. One can use a higher clock rate for a CIC decimation filter if a pipelined structure is used in the integrator stages, as compared to non-pipelined integrator stages. Simulation indicates that for this design, when the maximum decimation ratio is programmed to be 8, input/output word lengths are 8/16 bits, the throughput can be as high as 15.1 MHz. Another novelty of this design besides using the pipelined structure is its flexibility of programming the maximum decimation ratio, operating decimation ratio and the input/output word lengths. The maximum decimation ratio in this design is 1023 and the maximum input/output word lengths are 16/24 bits. A Matlab program is used to optimize the CIC filter hardware implementation based on the specific configuration before it is synthesized onto chips.

The information used in this thesis comes in part from the research program of Dr. T.A. Kwasniewski and his associates in the VLSI in Communications Group and the Integrated Electronics group of the Communications Research Center. The research results appearing in this thesis represents an integral part of the ongoing research programs. All research results in this thesis including tables, graphs and figures but excluding the narrative portions of the thesis are effectively incorporated into the research programs and can be used by Dr. Kwasniewski and his associates at Carleton University and CRC for education and research purposes, including publication in the open literature with the appropriate credits. Matters of intellectual property may be pursued collaboratively with Carleton University and Dr. Kwasniewski or with the Communication Research Center - Canada where and as appropriate.

TABLE OF CONTENTS

List of Tables.....	iv
List of Figures	v
List of Abbreviations	viii
Chapter 1: Introduction	1
1.1 Background.....	1
1.2 Thesis Motivation	4
1.3 Thesis Objective.....	7
1.4 Thesis Outline	8
Chapter 2: Basic Principle of Decimation Filters	9
2.1 Introduction.....	9
2.2 Multirate Digital Signal Processing	9
2.3 Downsampling	10
2.4 Spectrum of the Downsampled Signal.....	11
2.5 Aliasing Effects	13
2.6 Decimation of the Band-pass Signal.....	14
2.7 Conventional Decimation Filter Structures	15
2.8 Decimation Filter Applications	16
2.9 Interpolation Filters.....	19
2.10 Summary.....	20
Chapter 3: The CIC Decimation Filter	21
3.1 Introduction.....	21
3.2 CIC Decimation Filter Description and Mathematical Analysis	21
3.3 Frequency Analysis of the CIC Decimation Filter	23
3.4 Summary.....	30

Chapter 4: Implementation Considerations of the CIC Decimation Filter	32
4.1 Introduction.....	32
4.2 Pipelined Structure in Integrator Sections	32
4.3 Maximum Register Growth With No Truncation.....	34
4.3.1 2's Complement Representation	34
4.3.2 Calculation of Maximum Register Growth	35
4.3.3 Register Growth in CIC Decimation Filters	36
4.4 Truncation Error Considerations.....	38
4.5 Register Width Determination with Truncation.....	44
4.6 Summary.....	45
 Chapter 5: Functional Implementation of the CIC Decimation Filter	 47
5.1 Introduction.....	47
5.2 The Matlab Model of the CIC Decimation Filter.....	47
5.3 Functional Simulation Results	50
5.3.1 Single Tone Test	50
5.3.2 Testing With Two or Multiple Frequency Components	51
5.3.3 Testing a CIC Decimation Filter Composed of Different Numbers of Stages.....	53
5.3.4 Comparison of Matlab Model Performance and Intersil HSP43220 Decimating Digital Filter Development Software Simulation Results.....	54
5.4 Summary.....	55
 Chapter 6: FPGA Technology and the Mentor Graphic Design Environment	 64
6.1 Introduction.....	64
6.2 Altera FPGA Technology.....	65
6.2.1 Altera APEX 20K Device Architecture.....	65
6.2.2 Design Flow of Altera APEX 20K FPGA.....	67
6.2.3 Altera APEX 20K FPGA Device Features	68

6.3	Mentor Graphics Design Environment.....	69
6.4	Summary.....	70

Chapter 7: VHDL Implementation of the CIC Decimation Filter and Its FPGA Realization 71

7.1	Introduction.....	71
7.2	Design Specification	71
7.3	VHDL Implementation.....	72
7.3.1	Block Diagram.....	72
7.3.2	The Process of Designing a Configurable CIC Decimation Filter	73
7.3.3	Verification of the VHDL Implementation	74
7.4	Simulation Results.....	76
7.5	Synthesis and Optimization Consideration	78
7.6	Testability Issues	80
7.7	Summary.....	82

Chapter 8: Summary and Suggested Further Development 90

8.1	Thesis Project Summary	90
8.2	Suggested Further Development.....	91

References 95

LIST OF TABLES

Table 3.1: Pass band and aliasing band attenuation for large rate change factors	29
Table 7.1: Propagation delay of the first integrator stage: Critical block delay of the CIC decimation filter	80
Table 7.2: Synthesis results of the CIC decimation filter based on the Altera device	83

LIST OF FIGURES

Figure 1.1: Digital receiver functional block diagram	5
Figure 2.1: Performing downsampling in time domain	11
Figure 2.2: Signal magnitude spectrum change by downsampling, $D=4$	12
Figure 2.3: Decimation by a factor of D in frequency domain	13
Figure 2.4: Spectrum of a decimated band-pass signal	15
Figure 2.5: A widely used decimation filter structure before the appearance of the CIC decimation filter	16
Figure 2.6: Comparison of a digital and analog receiver	18
Figure 2.7: Interpolation by a factor of I in frequency domain	20
Figure 3.1: CIC decimation filter	22
Figure 3.2: CIC decimation filter structure in Z domain	23
Figure 3.3: Comparison of signal spectrum before and after sampling rate change	25
Figure 3.4: Frequency response of integrator stages	26
Figure 3.5: Amplitude response of the CIC decimation filter	28
Figure 3.6: Testing the CIC decimation filter	31
Figure 4.1: Pipelined and nonpipelined structures	33
Figure 4.2: Functional relationship for 2's complement addition	35
Figure 4.3: Register overflow in a CIC decimator with 2's complement implementation	37
Figure 4.4: Truncation in 2's complement representation	39
Figure 4.5: Probability density function for 2's complement truncation error	40
Figure 4.6: Error propagation in the CIC decimation filter	43
Figure 5.1: Algorithm of pipelined and nonpipelined integrator stages	49
Figure 5.2: Algorithm of comb stages	50
Figure 5.3: CIC decimation filter functional model simulation results for test 1. Configuration: $N=5$, $R=8$. Input signal: $f_1=(1/8)*F_L$	56
Figure 5.4: CIC decimation filter functional model simulation results for test 2. Configuration: $N=5$, $R=16$. Input signal: $f_1=(1/16)*F_L$, $f_2=F_L$	57
Figure 5.5: CIC decimation filter functional model simulation results for test 3. Configuration: $N=5$, $R=16$. Input signal: $f_1=(1/16)*F_L$, $f_2=(1/4)*F_L$	58
Figure 5.6: CIC decimation filter functional model simulation results for test 4. Configuration: $N=5$, $R=1023$. Input signal: $f_1=(1/8)*F_L$, $f_2=(1/4)*F_L$, $f_3=(1/2)*F_L$, $f_4=F_L$	59

Figure 5.7: CIC decimation filter functional model simulation results for test 5. Configuration: $N=3$, $R=16$. Input signal: $f_1=(1/8)*F_L$, $f_2=(3/4)*F_L$	60
Figure 5.8: CIC decimation filter functional model simulation results for test 6. Configuration: $N=4$, $R=16$. Input signal: $f_1=(1/8)*F_L$, $f_2=(3/4)*F_L$	61
Figure 5.9: CIC decimation filter functional model simulation results for test 7. Configuration: $N=5$, $R=16$. Input signal: $f_1=(1/8)*F_L$, $f_2=(3/4)*F_L$	62
Figure 5.10: Intersil HSP43220 decimation digital filter development software simulation results, $N=5$, $R=16$. Input signal: $f_1=(1/8)*F_L$, $f_2=(3/4)*F_L$	63
Figure 6.1: APEX 20K device block diagram	66
Figure 6.2: APEX 20K logic element	67
Figure 6.3: Simplified design flow	69
Figure 7.1: Simplified block diagram of the CIC decimation filter	72
Figure 7.2: Verification of the CIC decimation filter	75
Figure 7.3: Propagation delay in integrator stages of the CIC decimation filter	79
Figure 7.4: Testability of the CIC decimation filter	81
Figure 7.5: Input signal no. 1 to the CIC decimation filter, $f_1=(1/256)*F_H$, $f_2=(1/16)*F_H$, input=12 bits	84
Figure 7.6: Simulation 1 of the CIC decimation filter VHDL code. Configuration: $N=5$, $R_{max}=1023$, $R=16$ bits, output=20 bits	85
Figure 7.7: Input signal no. 2 to the CIC decimation filter, $f_1=(1/64)*F_H$, $f_2=(1/16)*F_H$, input=8 bits	86
Figure 7.8: Simulation 2 of the CIC decimation filter VHDL code. Configuration: $N=5$, $R_{max}=64$, $R=4$, output=16 bits	87
Figure 7.9: Input signal no. 3 to the CIC decimation filter, $f_1=(1/8)*F_L$, $f_2=(1/4)*F_L$, $f_3=(1/2)*F_L$, $f_4=F_L$ input=16 bits	88
Figure 7.10: Simulation 3 of the CIC decimation filter VHDL code. Configuration: $N=5$, $R_{max}=1023$, $R=1023$, output=24 bits	89
Figure 8.1: CIC interpolation filter	93
Figure 8.2: Decimator/Interpolator with rational ratio change	94

ACKNOWLEDGMENTS

The author wishes to thank Professor Tad Kwasniewski for many of his original ideas and remarkable insights into the problems encountered during the research.

I would also like to thank Professor Valek Szwarc for his guidance and detailed instructions throughout my research at the Communication Research Center (CRC). His assistance was crucial to the fulfillment of my thesis project.

As well, I would like to thank Professor Martin Bouchard of Ottawa University for several useful discussions.

Finally, I wish to thank all the members in my research group, especially Mr. Luc Desormeaux, the system administrator, for their help in many different forms.

GLOSSARY

ADC	Analog to Digital Converter
ASIC	Application Specific Integrated Circuit
A/D	Analog to Digital Converter
CAD	Computer Aided Design
CIC	Cascaded Integrator Comb
CMOS	Complementary Metal Oxide Semiconductor
CPLD	Complex Programmable Logic Device
CRC	Communication Research Center
DC	Direct Current
DDC	Digital Downconverter
DSP	Digital Signal Processing
D/A	Digital to Analog Converter
EDA	Electronic Design Automation
EDIF	Electronic Design Interchange Format
EPROM	Erasable Programmable Read Only Memory
ESB	Embedded System Block
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FPLD	Field Programmable Logic Device

I/O	Input/Output
IC	Integrated Circuit
ISOP	Interpolated Second-Order Polynomial
IF	Intermediate Frequency
LAB	Logic Array Block
LC	Logic Cell
LE	Logic Element
LSB	Least Significant Bit
LUT	Look-up Table
MHBF	Modified Half Band Filter
MPGA	Metallization Programmable Gate Array
MSB	Most Significant Bit
NCO	Numerically Controlled Oscillator
OtbT	Object to be Tested
PLD	Programmable Logic Device
RAM	Random Access Memory
RTL	Register Transfer Level
R&D	Research and Development
SDF	Standard Delay Format
SRAM	Static Random Access Memory

VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLSI	Very Large Scale Integration

Chapter 1

INTRODUCTION

1.1 Background

The decimation filter (decimator) is one of the basic building blocks of a sampling rate conversion system. It converts the sample rate of a signal from high to low. It has been widely used in such applications as speech processing, radar systems, antenna systems and communication systems [1],[2]. Considerable attention has been focused in the last few years on the design of high efficiency decimation filters. For example, Henry Samueli and Thu-ji Lin et al. [3],[4] designed a decimation/interpolation filter with a conversion ratio that is a power-of-two. The key to their design is the use of a single decimate/interpolate-by-two halfband filter as the computational engine, with the same block reused repeatedly. A throughput rate in excess of 100MHz has been achieved.

In 1981, Eugene Hogenauer [5] invented a new class of economical digital filters for decimation and interpolation (converting the sampling rate from low to high) called a cascaded integrator comb (CIC) filter. This filter was composed of an integrator part and a comb part. No multipliers were required and the storage requirement was reduced when compared with other implementations of decimation filters. This structure has been proven to be very efficient and many researchers have investigated it, a few of whom are noted in the following paragraphs.

Y. Djadi, et al. [6] designed a programmable decimation and interpolation digital filter based on the CIC structure. The circuit was configurable as either a decimation filter or an interpolation filter and the conversion ratio was programmable to any integer value from 10 to 256. Both the decimation filter and the interpolation filter had five integrator-comb stages. The filter was designed with MOSIS 1.2 micron CMOSN standard cell libraries, and simulation results demonstrated that the data input rate of this circuit could be as high as 50MHz.

Alan Kwentus, et al. [7] designed and fabricated a programmable CIC decimation filter using six integrator-comb pairs. The decimation factor was programmable with a power-of-two range of 2 to 1024. The highest throughput was measured at 250MHz when the circuit was manufactured in a 0.8 μm CMOS process. The integrator and comb stages were implemented using carry-save arithmetic in order to get high throughput. Kei-Yong Khoo et al [8] also presented an efficient architecture for the first carry-save integrator stage.

Hyuk Jun Oh and Yong H. Lee [9] used a simple interpolated second-order polynomial filter (ISOP) cascaded with a CIC decimation filter to effectively reduce the pass band distortion caused by CIC filtering with little degradation in aliasing attenuation. Their proposed decimation filter for a programmable downconverter cascaded a CIC filter, an ISOP, MHBFs (modified halfband filters) and a programmable FIR filter. An optimization method was also presented to simultaneously design the ISOP and programmable FIR filter [10]. A design example demonstrated that this leads to a more efficient decimation filter implementation.

Hong-Kui Yang and W. Martin Snelgrove [11],[12] took another approach to implementing a CIC decimation filter. They decomposed the decimation ratio R into two factors, namely, $R_1 * R_2$ (R_1 and R_2 are integers). Instead of using one CIC filter to decimate the

high speed digital signal, they used two CIC decimation filters: one CIC filter with a decimation ratio of R_1 , and one with a decimation ratio of R_2 . The implementation of the first decimator was based on a polyphase decomposition. It has been shown that the polyphase components can be realized with only a small number of adders and shifters.

Another way of implementing the CIC decimation filter was presented by Yonghong Gao, et al. [13]. In this implementation, the decimation factor N was constrained to an M -th power-of-two. The transfer function can be rewritten as the product of M identical low-order FIR filters. The sampling rate in this implementation decreased at every stage by a factor of two. The first stage was crucial to the throughput; the employment of parallel processing techniques enabled a higher clock rate. The polyphase decomposition technique used here is similar to that in reference [12].

Three leading semiconductor companies fabricate ICs with CIC decimation filters, namely, Harris Semiconductor (now called Intersil) [14],[15],[16], Graychip Inc. [17] and Analog Devices Inc. [18]. Their functionality and performance is compared below.

In Harris' HSP50214 Programmable Digital Downconverter (DDC) [14], a five-stage CIC decimation filter is used, and the decimation ratio can be programmed from 4 to 32. The throughput can be as high as 52MHz.

Harris' HSP43220 [15],[16] is a pure digital decimation filter. It is composed of two modules, namely, a five-stage CIC decimation filter followed by a FIR decimation filter. The input word width is fixed at 16 bits, and the output at 24 bits. It accepts samples in 2's complement format with a sample rate of up to 35MHz. The decimation ratio ranges from 10 to 1024.

In Graychip's GC4016 Multi-Standard Quad DDC chip [17], a five-stage CIC decimation filter is also used. The input data sampling rate can be up to 80MHz, and the decimation ratio is programmable as any integer between 8 and 4K. Note: in this work, for throughput or data sampling rate, the unit is normally referred to as MHz.

Analog Device's AD6620 digital receiver [18] also uses a five-stage CIC decimation filter. The decimation ratio range is from 1 to 32 and the highest data input rate for the CIC decimator is 32.5MHz.

1.2 Thesis Motivation

There is a trend in the current design of DSP wireless communication circuits such as digital receivers to implement the whole receiver as an all-digital, single-chip ASIC and to push the signal processor closer to the "front end" (of antenna or sensor). One of the main implications of this trend is the need for flexibly programming or configuring hardware so that it can accommodate multiple communication standards. A digital implementation keeps the IC manufacturing cost low while still providing the same performance as an analog implementation. Though analog stages can run at lower power compared to digital stages, they are normally more costly because of a number of manufacturing problems such as component yield. Analog devices are application specific and do not have the flexibility of being reprogrammed.

Figure 1.1 Digital receiver functional block diagram

In a wireless receiver, pushing the digital signal processor (DSP) to the “front end” also requires the usage of an analog to digital converter (ADC) to convert an intermediate-frequency (IF) analog signal or radio frequency (RF) analog signal into a discrete-time signal. To sample a signal, a sampling rate twice the signal bandwidth is needed (according to the Nyquist theorem). Figure 1.1 shows a block diagram of such a digital receiver. It selects and filters a single channel from a fixed signal bandwidth. For example, if the incoming sampled signal has a bandwidth of 15MHz, containing a channel of interest with a bandwidth of 10KHz, the sample rate of the real and quadrature components of the signal after the A/D conversion must be 15MHz or higher if the input signal is complex. The high resolution Numerically Controlled Oscillator’s (NCO) signal output mixes with the sampled signal and translates the input signal band of interest from IF to baseband. Here the bandwidth of interest is 10KHz but the sampling rate is still 15MHz, which is too fast for a DSP to process and also leads to large power consumption. The CIC (Cascaded Integrator Comb) Filter is a special type of decimation filter used to reduce the initial data sample rate down to a lower

desired sample rate. An additional filter is normally needed to eliminate the side lobes and compensate for the pass band deformations caused by the CIC filter. This is shown as the FIR filter in Figure 1.1. The resulting signal is fed into an “output format” block and processed to generate the required format, such as series or parallel, multi-channel or single-channel. Finally, the signal is ready for further processing by either a DSP processor or a general-purpose processor. From this example, it is clear that the decimation filter plays an important role in a digital receiver implementation.

As described in section 1.1, considerable research has been done on CIC decimation filters and many novel methods have been used to improve their efficiency. However, many limitations are associated with those methods. For example, for methods used in references [7] and [13], the decimation ratio is restricted as an M -th power-of-two. For references [10] and [12], the ratio R must be capable of decomposing into the product of two integers $R_1 * R_2$, since these implementations can not handle decimation ratios that are prime numbers. For reference [6], although the rate change figure could be any integer, the range is constrained between 10 and 256, and there is no means of programming the input/output register widths. For commercial products, such as the Harris and Analog Device products, the decimation ratio ranges are limited as stated previously. The Graychip GC4016 has a wider decimation ratio range than others, but the input/output word-length is fixed. The input and output ports for the Harris HSP43220 are fixed, and although its decimation ratio can be programmed, the decimation ratio range can't be. The importance of programming the decimation ratio range is that it enables the optimization of the hardware implementation. Hardware optimization is normally based on the worst case configuration, so it is of interest to have the flexibility of programming this worst case configuration.

In order to overcome all of the above limitations, a CIC decimation filter needs to be designed with full flexibility in terms of decimation ratio, I/O word length, and circuit optimization. More specifically, the design objectives for this CIC decimation filter are: programmable input/output register widths; optimization for all the intermediate register widths; and removal of any redundant storage. Based on these specifications, the internal hardware of the CIC decimation filter is configured and designed. This work is different from that discussed above in that the design can be optimized for performance and gate-count (which will lead to improved power consumption), and pre-configured to meet requirements of input/output word length, decimation ratio range, and decimation ratio.

This research was carried out in collaboration with the Communication Research Center (CRC) Canada. CRC is the leading government R&D laboratory in the area of communication technologies. It collaborates with Canadian universities and government agencies in a broad range of research programs, among them digital circuits for communication applications. The research on the CIC decimation filter was carried out within the framework of such a collaborative research agreement between Carleton University and CRC.

1.3 Thesis Objectives

In this thesis, the algorithm and implementation of a programmable CIC decimation filter with flexible rate change is investigated. The thesis objectives include the determination of a design and implementation methodology, investigation of circuit architecture and actual VHDL coding of a re-configurable CIC decimation filter.

The detailed specifications that this design must achieve is as follows: input data word length: 8-16 bits; output data word length: 16-24 bits; maximum decimation ratio: 1023; and number of integrator-comb stages: 5.

1.4 Thesis Outline

This thesis is organized as follows: Chapter 2 presents the basic principles of decimation filtering. Chapter 3 specifically deals with the CIC decimation filter and its properties. In Chapter 4, CIC decimation filter implementation considerations are discussed. Chapter 5 brings out a Matlab functional model for the CIC decimation filter. Chapter 6 briefly introduces the Altera FPGA technology used in this work and describes the Mentor Graphics design environment. The actual VHDL implementation and its FPGA realization are described in Chapter 7. Finally, a summary and suggested further research topics are presented in Chapter 8.

Chapter 2

BASIC PRINCIPLES OF DECIMATION FILTERS

2.1 Introduction

In many discrete time systems, signals of different sampling rates have to be processed at the same time. The process of converting a signal with a certain sampling rate to another sampling rate is called “sampling rate conversion”. When a lower sampling rate signal is converted to a higher sampling rate, it is called “interpolation”; the reverse case is called “decimation”. In this chapter, basic principles of multirate DSP are first presented, then structures of different decimation filters are compared, and finally, a few applications are given.

2.2 Multirate Digital Signal Processing

Multirate DSP hardware is an important sub-category of the DSP hardware. It deals with the sampling rate conversion of a digital signal, including interpolation and decimation. There are basically two methods to accomplish this. One method converts a digital signal to analog through a D/A, and then resamples the analog signal using a different sampling rate with an A/D, a process that requires analog filters. The other method operates totally in the digital domain, which has a great advantage over the first method as only digital filters are needed. David B. Chester [19] from Harris Semiconductor has given a comparison between analog filters and digital filters. It is reported that digital filters are more size- and power-efficient than analog filters in applications requiring an extremely linear phase, very high

stopband attenuation, or very low passband ripple. As well, if the filter's response must be programmable or adaptive, digital filters are more appropriate.

2.3 Downsampling

One simple way of decimating a digital signal $x(n)$ by a factor of D is to select every D -th sample of the input signal $x(n)$. The relationship between the resulting signal $y(m)$ and the original $x(n)$ is as follows:

$$y(m) = x(m \cdot D) \quad (2.1)$$

This decimation process is shown in Figure 2.1 for $D=4$. In the z -domain, the z -transform of the original signal (a) is:

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) \cdot z^{-n} \quad (2.2)$$

The signal depicted in Figure 2.1(b) is obtained by discrete sampling, for example, multiplying equation (2.1) by a discrete sampling function given in equation (2.3).

$$h(n) = \begin{cases} 1 & \text{for } n = mD, \quad m \text{ integer,} \\ 0 & \text{for } n \neq mD, \end{cases} \quad (2.3)$$

The z -transform of signal (b) can be expressed as:

$$\begin{aligned} X_0(z^D) &= \sum_{m=-\infty}^{\infty} x(m \cdot D) \cdot z^{-mD} \\ &= \sum_{m=-\infty}^{\infty} x(m) \cdot (z^D)^{-m} \\ &= Y(z^D) \end{aligned} \quad (2.4)$$

We get signal (c) from (b) by leaving out $D-1$ zeros between each sample; by using the relation of equation (2.5) one obtains the z -transform of signal shown in Figure 2.1 (c):

$$Y(z^D) = Y(z') \quad (2.5)$$

$Y(z)$ is $y(m)$ in z -domain.

Figure 2.1 Performing downsampling in time domain, $D=4$:
 (a) input signal; (b) multiplying the input signal by a discrete sampling function; (c) leave out $D-1$ zeros between every D -th sample.

2.4 Spectrum of the Downsampled Signal

The z -transform signal $Y(z^D)$ in (2.4) can be expressed in terms of $X(z)$ as follows [20]:

$$Y(z^D) = \frac{1}{D} \sum_{k=0}^{D-1} X(z W_D^k) \quad (2.6)$$

$$W_D^k = e^{-j2\pi k/D}$$

Substituting z with $\exp(j\Omega)$ one obtains the discrete-time Fourier transform

$$Y(e^{jD\Omega}) = \frac{1}{D} \sum_{k=0}^{D-1} X(e^{j\Omega - j2\pi k/D}) \quad (2.7)$$

This equation gives the spectrum of the downsampled signal. Figure 2.2(a) shows the magnitude spectrum (referred to as “spectrum”) of the original signal $|X(e^{j\Omega})|$. Figure 2.2(b) shows the downsampled signal $y(m)$ based on equation (2.7).

Figure 2.2 Signal magnitude spectrum change by downsampling, $D=4$

Substituting $D\Omega$ with Ω' in equation (2.7), we have equation (2.8):

$$Y(e^{j\Omega'}) = \frac{1}{D} \sum_{k=0}^{D-1} X(e^{j[\Omega' - 2\pi k]/D}) \quad (2.8)$$

equations (2.7) and (2.8) are equivalent except that the output signal spectrum Y in equation (2.7) is expressed in terms of frequency normalized to the original sampling rate and Y in equation (2.8) is expressed in terms of frequency normalized to the reduced sampling rate.

2.5 Aliasing Effects

From Figure 2.2, it can be seen that the decimated signal spectrum will overlap if the original signal spectrum is not within $[-\pi/D, \pi/D]$. This is known as “aliasing” and makes the recovery of the original signal impossible. A low-pass filter called an “anti-aliasing” filter is needed to band-limit the original signal.

Figure 2.3 Decimation by a factor of D in frequency domain:
(a) Block diagram of low-pass filter $b(n)$ and sampler; (b) Spectral relationships of the filter and decimator I/Os.

Figure 2.3 further shows why an anti-aliasing filter is needed. The decimation filter in Figure 2.3 is composed of a low-pass filter $b(n)$ and a sampler. The relationship between the filter input $x(n)$, sampled at rate F_s , its output $v(n)$ and the sampler output $y(n)$ in the frequency

domain are depicted in Figure 2.3(b). The signal spectrum at the filter output $|V(\omega)|$ is limited between $-\pi/D$ and π/D . The sampler expands the spectrum of its input $V(\omega)$ between $-\pi$ and π by selecting every D -th signal. The spectrum of the decimated signal is $|Y(\omega)|$. It is clear that without the low-pass filter, decimation will cause aliasing of signals whose spectral components lie outside of $[-\pi/D, \pi/D]$.

2.6 Decimation of the Band-pass Signal

Any bandpass signal can be expressed in terms of the modulation components of a baseband signal [20]. The expression is formed from the baseband signal by means of a frequency shift of $\Delta\Omega = 2\pi l/D$, where l is an integer. This is shown as follows:

$$X_l^{(m)}(z) = X(z \cdot e^{-j2\pi l/D}) \quad (2.9)$$

Figure 2.4 shows the spectrum of decimated band-pass signals. The spectrum in Figure 2.4(b) is identical to Figure 2.2(b), since $X(z \cdot e^{j2\pi l/D})$ is periodic in Ω with a period 2π . Thus the decimation filter structure in Figure 2.3 needs to be changed if the input signals are band-pass signals. Specifically, the low-pass filter must be replaced by a band-pass filter.

Figure 2.4 Spectrum of a decimated band-pass signal:
 (a) Spectrum of a band-pass signal; (b) Spectrum of a decimated band-pass signal.

2.7 Conventional Decimation Filter Structure

As discussed above, in a decimation filter a low-pass filter is required to limit the bandwidth of input signals before actually changing the sampling rate. Figure 2.5 shows a traditional implementation of a decimation filter. A direct-form FIR filter of length M is implemented in front of the down-sampler. This filter has M multipliers, M two-input accumulators, and $3M$ storage registers. Assuming the sampling rate of the input signal is F_s , then $M \times F_s$ multiplications are executed every second. To produce one sample at the data output if the decimation ratio is D , $D \times M$ multiplications must be carried out. The next chapter compares the CIC decimation filter structure with this structure and considers why the CIC

structure is favored in terms of computational and hardware requirements.

Figure 2.5 A widely used decimation filter structure before the appearance of the CIC decimation filter

Improvement can be made to this structure; for example, a polyphase filter structure could be used to increase the throughput [21], but there is a cost of greater implementation complexity.

2.8 Decimation Filter Applications

One of the typical applications for decimation filters in telecommunications is a digital receiver. Modern A/D converters can handle sampling rates as high as a few hundred MHz or

even higher [22],[23],[24]. In a receiver, the sampling rate needs only be as high as twice the signal bandwidth according to the Nyquist theorem. Analog signals received by an antenna and amplified by a RF amplifier can be directly converted into the digital domain, but the sampled RF (IF) signal may require extremely high sampling rates which makes it very costly in terms of processing requirements. This signal needs to be down-converted, filtered and decimated by a digital receiver before being processed.

Figure 2.6 shows block diagram architectures of typical analog and digital receivers. A digital receiver is more complex than an analog receiver, since the RF signal is transferred to digital domain and then back to analog domain after being processed. Why is a complex structure preferred? One reason is that digital filters have certain advantages over analog filters in terms of performance, implementation cost, and long term stability. Another is that the precision of many parameters can not be easily controlled for the analog manufacturing process, while it is possible to implement digital filters with well controlled features such as small pass band ripple, sharp transition band characteristics and linear phase.

The digital receiver block diagram in Figure 2.6 shows that the decimation filter plays an important role in processing the sampled RF signal. (Note: diagram in Figure 2.6 is from reference [25]) First, this signal is down-converted to baseband by a digital mixer and also decomposed into In-phase and Quadrature components. Note that the sample rate of the I and Q signals - F_s , is still high. This signal contains sum $(f_1 + f_2)$ and difference $(f_1 - f_2)$ frequencies (assume f_1 is the input signal frequency and f_2 is the Numerically Controlled Oscillator -NCO output frequency. Note: NCO is referred to as digital local oscillator in Figure 2.6). The high frequency components need to be eliminated, and the unnecessarily high sampling rate needs to be brought down since the sampling rate only needs to be greater than twice the bandwidth

of the signal. The decimation filter eliminates frequency components beyond baseband and reduces the sampling rate down to F_s/N . Then the signal with the low sampling rate is ready to be processed by the DSP processor.

Analog Receiver Block Diagram

Digital Receiver Block Diagram

Figure 2.6 Comparison of a digital and analog receiver[25]

Another major advantage of digitizing closer to the antenna is that the receiver can handle many different communication standards. In an analog receiver, dedicated hardware is needed to process different standards, while in the digital world, the hardware can be programmed to handle those standards.

Harris Semiconductor produced the first CIC digital decimation filter (HSP43220) which was available on the market in 1990 [15]. Graychip Co. introduced a digital receiver chip in 1992. Currently, a number of companies manufacture chips with CIC decimation filters, such as digital receiver/downconverter chips: Harris' HSP50214 programmable downconverter [14], Graychip's GC4016/GC4017 Multi-Standard Quad Receiver [17] and Analog Device's AD6620 Digital Receive Signal Processor [18].

Besides being used in digital receivers, decimation filters are widely used in many systems involving multirate processing and conversion. Those applications include FSK Modems, Speech/Audio Coding and Image/Video Coding [2].

2.9 Interpolation Filters

Interpolating by a factor of I corresponds to inserting $I-1$ zeros between two consecutive samples. In the frequency domain, it is a reverse process to that of decimation. The frequency components of input signals are narrowed down from $[-\pi, \pi]$ to $[-\pi/I, \pi/I]$, but at the same time, some high frequency spectral components are introduced. A low-pass filter is needed to eliminate these high frequency components beyond π/I . Figure 2.7 shows the process of interpolation by a factor of I .

Figure 2.7 Interpolation by a factor of I in frequency domain

2.10 Summary

The basic principle of decimation was introduced in this chapter. The structure of decimation filters used before the introduction of the CIC decimation filter was described and will be compared to the CIC structure in the next chapter. The example of a digital receiver showed how and where decimation filters are used.

Chapter 3

THE CIC DECIMATION FILTER

3.1 Introduction

As indicated in Chapter 1, the Cascaded Integrator-Comb (CIC) filter structure was first presented by Eugene Hogenauer in 1981. Today, some commercial digital receiver chips use this structure. For example, Graychip's GC4016/GC4017 Multi-Standard Quad Receiver [17], Analog Device's AD6620 Digital Receive Signal Processor [18] and Harris' HSP50214 Programmable Downconverter [14] are all based on this structure, with five-stage CIC decimation filters in their chips. In this chapter, the CIC decimation filter structure is studied in detail.

3.2 CIC Decimation Filter Description and Mathematical Analysis

Figure 3.1 shows the structure of the CIC decimation filter presented by Hogenauer. The filter consists of N integrator stages and N comb stages. The sampling rate change ratio is R . In Figure 3.1, all the integrator stages are identical and all the comb stages are also identical. Input data is fed into the first integrator stage with a high sampling rate F_s (the integrator stages are operating at the same clock rate as the input sampling rate F_s) and is processed by the N integrator stages. There is a down-sampler between the integrator stages and the comb stages. Only 1 out of R samples is selected from the last integrator stage and fed into the first comb stage. The input sample rate of the comb stages is F_s/R , and the comb stages operate at

a clock rate of F_s/R . The sampling rate of the CIC decimation filter output signal is decreased to F_s/R .

Figure 3.1 CIC decimation filter

As shown in Figure 3.1, the transfer function of an integrator stage in the Z domain is: $1/(1-Z^{-1})$ with a low-pass characteristic, and the transfer function of a comb stage is: $(1-Z^{-1})$ which is a high-pass filter. As the clock rate of comb stages is changed to F_s/R , data is processed R times slower than in the integrator section. Thus, the transfer function for a comb stage becomes: $(1-Z^{-R})$ if normalized to the input sample rate. The transfer function for the circuit in Figure 3.1 is:

$$H(z) = H_I^N(z) H_C^N(z) = \frac{(1-z^{-R})^N}{(1-z^{-1})^N} = \left[\frac{(1-z^{-1})(1+z^{-1}+\dots+z^{-R+1})}{(1-z^{-1})} \right]^N = \left[\sum_{k=0}^{R-1} z^{-k} \right]^N \quad (3.1)$$

In this implementation, $2N$ adders and $4N+2$ registers are needed. Since an adder exists for each stage, two registers are needed to store the two input signals for each stage, and 2 additional registers store the integrator and comb outputs. If this same transfer function is

realized by the implementation described in Figure 2.5, then N cascaded low-pass filters as described in Figure 2.3 would be required. Thus, a total of $N \cdot R$ adders and $N \cdot (2R+1)$ registers is required. Comparing a CIC implementation with a traditional implementation for large R , many adders and registers are saved. For this reason, the CIC decimation filter is an efficient class of digital filters.

3.3 Frequency Analysis of the CIC Decimation Filter

Next, consider how the CIC decimation filter works in the frequency domain. In order to make the analysis easier, the transfer structure in the Z domain is redrawn as shown in Figure 3.2.

Figure 3.2 CIC decimation filter structure in Z domain

To undertake a frequency analysis, we must replace Z with $e^{j\omega}$ (ω is the frequency relative to the input sampling rate F_s). Assuming that the Fourier transform of $x(n)$ is $X(\omega)$, then $x_1(n)$ in

$$\text{frequency domain is: } X_1(\omega) = \left(\frac{1}{1 - e^{-j\omega}} \right)^N \cdot X(\omega) \quad (3.2)$$

$x_2(n)$ is given by:

$$\begin{aligned}
X_2(\omega) &= \frac{1}{R} \sum_{k=0}^{R-1} X_1\left(\frac{\omega - k \cdot 2\pi}{R}\right) \\
&= \frac{1}{R} \sum_{k=0}^{R-1} \left[\frac{1}{1 - e^{-j\left(\frac{\omega - k \cdot 2\pi}{R}\right)}} \right]^N \cdot X\left(\frac{\omega - k \cdot 2\pi}{R}\right) \\
&= \frac{1}{R} \left\{ \left[\frac{1}{1 - e^{-j\left(\frac{\omega}{R}\right)}} \right]^N \cdot X\left(\frac{\omega}{R}\right) + \left[\frac{1}{1 - e^{-j\left(\frac{\omega - 2\pi}{R}\right)}} \right]^N \cdot X\left(\frac{\omega - 2\pi}{R}\right) + \dots \right\} \quad (3.3)
\end{aligned}$$

For $X(\omega)=0$ when $\omega > \pi/R$ or $2\omega_{\max} < 2\pi/R$, $\omega \in [-\pi/R, \pi/R]$, equation (3.3) could be

$$\text{simplified to: } X_2(\omega) = \frac{1}{R} \left(\frac{1}{1 - e^{-j\omega/R}} \right)^N \cdot X(\omega/R) \quad (3.4)$$

Only the first term remains, as the rest of the terms make no contribution to $X_2(\omega)$ in the interval of $[-\pi/R, \pi/R]$. This process is illustrated in Figure 3.3.

Spectrum $X_1(\omega)$ for the discrete-time signal $x_1(n)$ with a sampling rate F_s in Figure 3.2 is shown in Figure 3.3(a). At the output of the down-sampler, the sampling rate changes to F_s/R , the time domain signal representation changes to $x_2(n)$ ($x_2(n)$ is every R -th sample of $x_1(n)$). The spectrum of $x_2(n)$, $X_2(\omega)$ is shown in Figure 3.3(b). This process can be considered in a different way: it is analogous to sampling a discrete time signal $x_1(n)$ with a sampling rate F_s/R and in the frequency domain, sampling a signal is copying the baseband frequency components.

For $\omega_{\max} < \pi/R$ (as shown in Figure 3.3(b)) the signal spectrum centered at $2\pi/R$ or $-2\pi/R$ does not overlap with the region $[-\pi/R, \pi/R]$ and as a result, does not overlap with the signal spectrum centered at 0.

(a) Input signal spectrum before sampling rate change

(b) Output signal spectrum after sampling rate change

Figure 3.3 Comparison of signal spectrum before and after sampling rate change;
 (a) Input signal spectrum (b) Output spectrum after sampling rate change

Thus, it is important to eliminate the high frequency component of $X(\omega)$. The function of the first N integrator stages is to eliminate the high frequency components, which satisfies the condition $\omega_{\max} < \pi/R$. The Matlab plot in Figure 3.4 shows the frequency response for an integrator composed of different numbers of stages.

From Figure 3.4, it can be seen that the goal of the integrators is to emphasize the low-frequency content of the input signal, and attenuate the high frequency component. It can also be seen that as the number of integrator stages increases, the high frequency attenuation increases. So when N is big enough, the simplification from equation (3.3) to (3.4) could be

accomplished in a CIC decimation filter. However, it has to be noted that the integrator itself is not a stable structure as it amplifies the low frequency component. Overflow means that when the input is a stream of “1”, the integrator output will cause overflow. But from equation (3.1), it follows that the whole CIC decimation filter is stable. This is because the comb stage is a high-pass structure and the low frequency component is attenuated, making the whole system stable.

Figure 3.4 Frequency response of integrator stages

Now let us calculate the output signal $y(n)$'s spectrum $Y(\omega)$:

$$\begin{aligned}
 Y(\omega) &= (1 - e^{-j\omega})^N \cdot X_2(\omega) \\
 &= \frac{1}{R} \cdot \frac{(1 - e^{-j\omega})^N}{(1 - e^{-j\omega/R})^N} \cdot X\left(\frac{\omega}{R}\right) \\
 &= \frac{1}{R} \cdot \left[\frac{e^{-j\omega/2}}{e^{-j\omega/(2R)}} \cdot \frac{(e^{j\omega/2} - e^{-j\omega/2})}{(e^{j\omega/(2R)} - e^{-j\omega/(2R)})} \right]^N \cdot X\left(\frac{\omega}{R}\right) \\
 &= \frac{1}{R} \cdot \left(\frac{\sin \omega / 2}{\sin \omega / 2R} \cdot e^{-j\omega(1/2 - 1/2R)} \right)^N \cdot X\left(\frac{\omega}{R}\right) \\
 &= \frac{1}{R} \cdot \left(\frac{\sin \omega / 2}{\sin \omega / 2R} \right)^N \cdot e^{-jN(1/2 - 1/2R)\omega} \cdot X\left(\frac{\omega}{R}\right)
 \end{aligned} \tag{3.5}$$

Here we have the amplitude response and the phase response of the complete CIC decimation filter. Some amplitude distortion is introduced. For increasing N the amplitude frequency response exhibits more attenuation at the pass band edge. The comparison of CIC decimation filter amplitude responses for different N values is shown in Figure 3.5.

It can be seen from Figure 3.5 that the CIC decimation filter is a low-pass filter in the range $[0, 1]$. Defining $[0, f_c]$ as the pass band, $[f_c, 1]$ as the aliasing/imaging band with the same width as the pass band (f_c is the bandwidth relative to the low sampling rate, $f_A = 1 - f_c$). An increase of N causes aliasing band attenuation but more distortion in the pass band. The pass band and aliasing band attenuation is shown in Table 3.1. It is assumed that the rate change ratio R is large, so that the amplitude response of the CIC decimation filter $P(f)$ in equation (3.6) can be simplified to $P_A(f)$ in (3.7):

$$P(f) = \frac{1}{R^2} \cdot \left[\frac{\sin(\omega / 2)}{\sin(\omega / 2R)} \right]^{2N} \tag{3.6}$$

$$P_A(f) = \frac{1}{R^2} \cdot \left[\frac{\sin(\omega/2)}{\omega/2R} \right]^{2N} \quad (3.7)$$

Figure 3.5 Amplitude response of CIC decimation filter

As stated in the literature [5], the error between $P(f)$ and $P_A(f)$ is less than 1dB for $R \geq 10$, $1 \leq N \leq 7$ and $0 \leq f \leq 255/256$, where f is frequency relative to the low sampling rate.

Table 3.1 also shows that with increasing N , large attenuation in the aliasing band occurs, while pass band distortion increases, a trade-off which has to be considered when designing the CIC decimation filter.

Table 3.1 Pass band and aliasing band attenuation for large rate change factors

Relative Bandwidth (f_D)	Pass band attenuation at f_C in [dB] as a function of stage number N					
	1	2	3	4	5	6
1/128	0.00	0.00	0.00	0.00	0.00	0.01
1/64	0.00	0.01	0.01	0.01	0.02	0.02
1/32	0.01	0.03	0.04	0.06	0.07	0.08
1/16	0.06	0.11	0.17	0.22	0.28	0.34
1/8	0.22	0.45	0.67	0.90	1.12	1.35
1/4	0.91	1.82	2.74	3.65	4.56	5.47

Relative Bandwidth (f_D)	Aliasing band attenuation at f_A in [dB] as a function of stage number N					
	1	2	3	4	5	6
1/128	42.1	84.2	126.2	168.3	210.4	252.5
1/64	36.0	72.0	108.0	144.0	180.0	215.9
1/32	29.8	59.7	89.5	119.4	149.2	179.0
1/16	23.6	47.2	70.7	94.3	117.9	141.5
1/8	17.1	34.3	51.4	68.5	85.6	102.8
1/4	10.5	20.9	31.4	41.8	52.3	62.7

To test the system high frequency signal attenuation, a Matlab model of a CIC decimation filter with five integrator-comb pairs was developed and tested (this model is discussed in Chapter 5). The decimation ratio R was set arbitrarily to 16. A waveform

consisting of two sinusoid waveforms was used to test the filter, $X(t) = \sin(2\pi f_1 t) + \sin(2\pi f_2 t)$. The frequencies of these two sinusoid waveforms were set as follows: $f_1 = (1/16R) * f_H = (1/256) * f_H = (1/16) * f_L$ and $f_2 = (1/R) * f_H = (1/16) * f_H = f_L$, letting f_H represent the high sampling rate, and f_L represent the low sampling rate. The input signal spectrum is shown in Figure 3.6(a), and the decimated output spectrum is shown in Figure 3.6 (b).

In Figure 3.6(a), two peaks are shown at positions $(1/256) * f_H$ and $(1/16) * f_H$. From Figure 3.6(b), the frequency component of $(1/R) * f_H = (1/16) * f_H = f_L$ does not appear in the output signal spectrum, since it has been eliminated by the CIC decimation filter. Only the frequency component of $(1/16R) * f_H = (1/16) * f_L$ remains, as $f_1 < (1/2) * f_L$. This indicates that the CIC decimation filter has the frequency response of a low-pass filter shown in Figure 3.5. Details of the Matlab functional model are presented in Chapter 5. Many harmonic frequency components are present in the input signal in Figure 3.6 due to a quantization process of a periodic signal: the quantization error itself is periodic in the time domain. This happens when the sampling rate is exactly an integer multiple of the sampled signal frequency. In practice, this seldom happens since the sampling rate can not be set precisely to N times the signal frequency. Thus, in the following analysis, a small frequency offset is added to eliminate this effect.

3.4 Summary

The structure of the CIC decimation filter, introduced by Eugene Hogenauer, was considered in this chapter. From the frequency response characteristic, the integrators emphasize the low frequency content and attenuate the high frequency content. While the integrator may lead to integrator stage output overflow, a serial connection of comb stages and

integrator stages guarantee filter stability. The frequency analysis of the CIC decimation filter was also presented in this chapter. Figure 3.6 shows the CIC decimation filter functional testing results.

Figure 3.6 Testing the CIC decimation filter; (a) Input signal spectrum (b) Decimated output signal spectrum

Chapter 4

IMPLEMENTATION CONSIDERATIONS OF THE CIC DECIMATION FILTER

4.1 Introduction

When implementing CIC decimation filters at the RTL level, many detailed problems need to be considered. In this chapter a few implementation issues are discussed. In order to attain the highest possible throughput, a pipelined structure is used in this work. The register width of the CIC decimation filter output is set to a constant value. The register width of all the integrator and comb stages needs to be determined. To do this, truncation errors are estimated. To overcome the overflow problem in the integrator section, 2's complement representation is adopted in this work.

4.2 Pipelined Structure in Integrator Sections

To obtain a high throughput of the CIC decimation filter, the circuit must be designed in such a way that a high frequency system clock can be used. The usable highest clock rate is determined by the maximum delay of any two adjacent (simultaneously clocked) registers. As discussed in the last chapter, the clock rate in the integrator section is R times higher than in the comb section. Thus, the clock rate of the integrator section determines the throughput of the whole CIC decimation filter. The integrator circuit in Figure 3.1 is a nonpipelined structure. Based on the following analysis, optimization is required to achieve higher throughput.

Figure 4.1 shows the circuits for integrator stages with a nonpipelined structure and integrator stages with a pipelined structure [6]. In the pipelined structure, no additional pipeline registers are used, so that the hardware requirement is the same as in the nonpipelined structure. The delay between registers determines the maximum clock rate. If the delay is short, a higher maximum clock rate can be used; if the delay between registers is long, only a slower frequency clock signal can be used. In the nonpipelined structure, there are two adders between two adjacent registers, while in the pipelined structure there is only one. Thus, a pipelined integrator section is preferred for higher throughput. In the comb section, a pipelined structure is not necessary as the clock rate is much slower than the integrator section. A detailed mathematical analysis of nonpipelined and pipelined structures is given in Chapter 5.

Figure 4.1 Pipelined and nonpipelined structures

4.3 Maximum Register Growth with No Truncation

In an ideal single-pole recursive filter, an overflow during the addition of two binary numbers occurs when the sum exceeds the word size available in the digital implementation of the filter. In the CIC decimation filter, since the first N stages are integrators with unity feedback, integrator outputs may grow without limit for unrelated input data (unrelated means statistically independent). This may result in register overflow in the integrator stages.

This can be resolved if the following two conditions are met at the same time [5]:

1. The filter is implemented with 2's complement arithmetic or other number system that allows "wrap-around" between the most positive and the most negative numbers.
2. The range of the number system is equal to or exceeds the maximum magnitude expected at the output of the entire decimation filter (here, the number system range is understood as the maximum possible magnitude used for all the calculation stages). If these two conditions are met, the overflow at the intermediate result does not matter. This will be explained in 4.3.3.

4.3.1 2's Complement Representation

A negative number is obtained by subtracting the corresponding positive number from 2.0 (assuming the absolute value of the number is less than 1), creating the 2's complement of the positive number. In a practical sense, it is formed by inverting every bit of the positive number and adding one LSB (least significant bit) as shown below. The MSB (most significant bit) is 1 indicating that it is a negative number. In Expression (4.1), X is a decimal number and X_{2c} is the two's complement representation of X .

$$X_{2c} = 1\overline{b_1b_2b_3}\cdots\overline{b_g} + 0.0\cdots 01 \quad X < 0 \quad (4.1)$$

The functional relationship between a 2's complement representation and a decimal representation is shown in Figure 4.2. Assuming several decimal numbers are added together, v , the result of the addition, is a decimal number and $g(v)$ is the output after converting v to a 2's complement representation. Note, however, that if v exceeds the maximum magnitude which is 2, truncation is applied, and bits above the MSB are erased. Note also, that when v is between 1 and 2, it goes into the negative region in 2's complement representation. This phenomenon is called "overflow oscillation". For example, when $v=1$, $g(v)=1$; when $v=3$, $g(v)$ is still $=1$; when $v=-1$, $g(v)=1$.

Figure 4.2 Functional relationship for 2's complement addition

4.3.2 Calculation of Maximum Register Growth

The maximum register growth is a function of the maximum output magnitude resulting from the worst possible input conditions. The reason for considering this is to

determine register length requirements necessary to assure no data loss. From the transfer function $H(z)$ in (3.1) :

$$H(z) = \sum_{k=0}^{(R-1)N} h(k)z^{-k} = \left[\sum_{k=0}^{R-1} z^{-k} \right]^N \leq \left| \sum_{k=0}^{R-1} z^{-k} \right|^N \leq \left(\sum_{k=0}^{R-1} |z|^{-k} \right)^N = \left(\sum_{k=0}^{R-1} 1 \right)^N = (R)^N \quad (4.2)$$

From this expression, the maximum register growth/width, $G_{\max}$ can be expressed as:

$$G_{\max} = (R)^N \quad (4.3)$$

If the input data word length is B_{in} , one may calculate the most significant bit (MSB) at the filter output $B_{\max}$ using the register growth (based on amplification of the input data). The result is as follows [5]:

$$B_{\max} = \lceil N \log_2 R + B_{in} - 1 \rceil \quad (4.4)$$

where the LSB (least significant bit) of the input signal is defined as bit number zero and $\lceil x \rceil$ is the smallest integer not less than x .

4.3.3 Register Growth in CIC Decimation Filters

It is clear that in a CIC decimation filter the maximum register growth/bit width is a finite number that can be determined from the sampling rate change ratio R and the number of integrator-comb pairs N . Once these two numbers are determined, $G_{\max}$ is fixed. And if $B_{\max}$ is set to the MSB at the filter output, the filter output register width will be sufficient. Section 4.3.1 stated that if 2's complement representation is used in the CIC decimation filter (which conforms to requirement 1), and the range of the number system is equal to $B_{\max}$ (conforming to requirement 2), the register overflow at the intermediate stages does not impact the accuracy of the output. Figure 4.3 shows why in a more direct way. When the above two conditions are

met, the output represents an accurate result.

Figure 4.3 Register overflow in a CIC decimator with 2's complement implementation

Figure 4.3 shows the register overflow example in a three-stage CIC decimation filter (composed of three integrator-comb pairs). It is assumed that the inputs are in 2's complement format, and all the intermediate calculation results are also calculated in 2's complement format. It is also assumed that the maximum absolute value of output signals is scaled to 1 in decimal format. Note that once the filter architecture is determined, the maximum possible output magnitude can be found by calculating the worst possible input cases, as shown in Section 4.3.2. The output signal is also in 2's complement format, giving an output signal range of 0 to 2. By making these two assumptions, we ensure that the two conditions described in Section 4.3.1 are met (in an actual implementation, these two assumptions are easy to fulfill). Imagine that the solid line in Figure 4.3 represents straight calculation results before

conversion to 2's complement format, and the dotted line is the result after applying 2's complement format (the same as the relationship between v and $g(v)$ in Figure 4.2). Following the dotted line, once the result exceeds the $[0,2]$ range, an overflow occurs, and the value is brought back to $[0,2]$. By comparing the dotted line and the solid line it is seen that while there is a difference in the middle stages, at the output they reach the same point. This can be mathematically demonstrated as follows:

Assume that at the output, the solid line reaches a point " a ", and the dotted line reaches another point " b ", then $0 \leq a \leq 2$, $0 \leq b \leq 2$. They both come from the same input value and they both experience the same calculation process, except that a few overflows have occurred to the process described by the dotted line. Each overflow is equal to adding or subtracting 2. Thus, $b = a + M \times 2$ (M is an integer number) and $0 \leq a + M \times 2 \leq 2$ while $0 \leq a \leq 2$. Both expressions can be true at the same time only if $M=0$ and so $b=a$. This demonstrates that overflow does not affect the final result if the two conditions given in 4.3.1 are met.

4.4 Truncation Error Considerations

From previous discussion, the implementation of a CIC decimation filter requires that the register length of every stage is kept within the range of 0 to B_{max} . This applies to the output register as well. However, in practice, the output data word length is usually predetermined (for example, if B_{max} is 20 and the output word length is fixed at 10, then one can keep only the high 10 output bits). Truncation or rounding is applied to LSBs of the output data and some error is introduced. Truncation or rounding can also be applied to the input of every filter stage to reduce register width. Thus, there are a total of $2N+1$ truncation/ rounding sources that

contribute errors to the output data ($2N$ input registers, one for every filter stage plus one for the output register). Next the errors seen at the filter output are calculated.

It is assumed for the purpose of this discussion that the error sources resulting from truncation/ rounding are independent of each other and that they have a uniform probability of distribution [21]. It will be demonstrated later that for truncation and rounding, the statistical errors are the same, except for the input register for the first filter stage and for the output register. As truncation is easier to implement, it is used throughout the discussion.

For truncation in a 2's-complement representation, suppose that when a number x is truncated from n bits ($x = 0.b_0 b_1 b_2 \dots b_{n-1}$) to v bits ($x = 0.b_0 b_1 b_2 \dots b_{v-1}$), the lower significant bits below b_{v-1} are dropped. In Figure 4.4, $x(n \text{ bits})$ is truncated to $QI(x)$ which is v bits. $QI(x)$ is shown on the y-axis of Figure 4.4:

Figure 4.4 Truncation in 2's complement representation

Unlike rounding, the error caused by truncation has a bias, which is shown in Figure 4.5 below:

Figure 4.5 Probability density function for 2's complement truncation error, $\Delta=2^{-b}$

E_t is the truncation error and $P(E_t)$ is the E_t probability. When a truncation occurs, it distributes evenly between $-\Delta$ and 0.

The error distribution has a “width” (the same as in the case of rounding) of:

$$E = 2^{-b} \quad (4.5)$$

The mean of the error (for rounding, it is 0):

$$\mu = \frac{1}{2} E = 2^{-b-1} \quad (4.6)$$

The variance of the error [21] is:

$$\sigma^2 = \frac{1}{12} E^2 \quad (4.7)$$

In a CIC decimation filter composed of N integrator-comb pairs, there are $2N+1$ error sources and the j -th error source will propagate through a system composed of the j -th stage

up to the last comb stage. The following section covers the calculation of the transfer function of the CIC decimation filter from j -th stage up to the last stage.

There are two possibilities when considering the range of j [5]: the first is from 1 to N ; the other one is from $N+1$ to $2N$. The two cases are treated differently.

Case 1: The transfer function is:

$$H_j(z) = H_i^{N-j+1} H_c^N = \frac{(1-z^{-R})^N}{(1-z^{-1})^{N-j+1}} = (1-z^{-R})^{j-1} \left[\sum_{k=0}^{R-1} z^{-k} \right]^{N-j+1}, \quad j = 1, \dots, N \quad (4.8)$$

The order of the polynomial in terms of z^{-1} is:

$$R(j-1) + (R-1)(N-j+1) = (R-1)N + j - 1 \quad (4.9)$$

Thus, the system's transfer function can be expanded as:

$$H_j(z) = \sum_{k=0}^{(R-1)N+j-1} h_j(k) z^{-k} \quad (4.10)$$

Another way of expressing the system's transfer function is to expand the integrator and comb sections in terms of their binomial expansion:

$$H_j(z) = \left[\sum_{l=0}^N (-1)^l \binom{N}{l} z^{-Rl} \right] \cdot \left[\sum_{v=0}^{N-j+v} \binom{N-j+v}{v} z^{-v} \right] \quad (4.11)$$

Calculation of the cross product yields:

$$H_j(z) = \sum_{l=0}^N \sum_{v=0}^{N-j+v} (-1)^l \binom{N}{l} \binom{N-j+v}{v} z^{-(Rl+v)} \quad (4.12)$$

Letting $k = Rl + v$, l be an integer in the range of 0 to $\lfloor k/R \rfloor$, the above equation can be rewritten as:

$$H_j(z) = \sum_{k=0}^{\lfloor k/R \rfloor} \left[\sum_{l=0}^{\lfloor k/R \rfloor} (-1)^l \binom{N}{l} \binom{N-j+k-Rl}{k-Rl} \right] z^{-k} \quad (4.13)$$

Comparing this to the transfer function above, we get the result:

$$H_j(z) = \sum_{k=0}^{(R-1)N+j-1} h_j(k) z^{-k} \quad (4.14)$$

$$h_j(k) = \sum_{l=0}^{\lfloor k/R \rfloor} (-1)^l \binom{N}{l} \binom{N-j+k-Rl}{k-Rl}, \quad j=1,2,\dots,N$$

Case 2: The transfer function is:

$$H_j(z) = H_c^{2N+1-j} = (1 - z^{-R})^{2N+1-j}, \quad j = N+1, \dots, 2N \quad (4.15)$$

By simply expanding this, we get:

$$H_j(z) = \sum_{k=0}^{2N+1-j} h_j(k) z^{-kR}, \quad j = N+1, \dots, 2N \quad (4.16)$$

$$h_j(k) = (-1)^k \binom{2N+1-j}{k}, \quad j = N+1, \dots, 2N$$

As stated above, the transfer function from the j -th stage up to and including the final stage is as follows:

$$H_j(z) = \begin{cases} H_1^{N-j+1} H_c^N = \sum_{k=0}^{(R-1)N+j-1} h_j(k) z^{-k}, & j = 1, 2, \dots, N \\ H_c^{j-N} = \sum_{k=0}^{2N+1-j} h_j(k) z^{-kR}, & j = N+1, \dots, 2N \end{cases} \quad (4.17)$$

while

$$h_j(k) = \begin{cases} \sum_{l=0}^{\lfloor k/R \rfloor} (-1)^l \binom{N}{l} \binom{N-j+k-Rl}{k-Rl}, & j = 1, 2, \dots, N \\ (-1)^k \binom{2N+1-j}{k}, & j = N+1, \dots, 2N \end{cases} \quad (4.18)$$

From the system's transfer function (4.17) (4.18), every $h_j(k)$ contributes to the error gain independently. Figure 4.6 shows how an error source E_j propagates through the CIC decimation filter. Based on the (4.17) (4.18), when $j=1,2,\dots,N$ the delay element is z^{-1} , for

$j=N+1, \dots, 2N$ the delay element can be represented by z^{-R} . Assume all E_j are independent error sources, and all error sources will have to go through a network as shown in Figure 4.6. Note that the b_j parameters for different E_j are different, as shown in equation (4.18).

Figure 4.6 Error propagation in the CIC decimation filter

Thus, the error mean μ and variance σ^2 at the output due to the j -th error source are given by [5]:

$$\mu_{out} = \mu_j D_j$$

$$D_j = \begin{cases} \sum_k h_j(k), & j = 1, 2, \dots, 2N \\ 1, & j = 2N + 1 \end{cases} \quad (4.19)$$

$$\sigma_{out}^2 = \sigma_j^2 F_j^2$$

$$F_j^2 = \begin{cases} \sum_k h_j^2(k), & j = 1, 2, \dots, 2N \\ 1, & j = 2N + 1 \end{cases} \quad (4.20)$$

in (4.19) and (4.20) k ranges from 0 to $((R-1)N+j-1)$.

D_j could be simplified to:

$$D_j = \begin{cases} (RM)^N, & j = 1 \\ 0, & j = 2, 3, \dots, 2N \\ 1, & j = 2N + 1 \end{cases} \quad (4.21)$$

D is non-zero for only the first and last of the error sources, so the difference in μ between truncation and rounding does not affect μ_{out} except for the first and last of the error sources.

Further for σ_{out}^2 , truncation and rounding error variances are all the same. The total mean and variance values of the output signal due to truncation are:

$$\begin{aligned} \mu_{out} &= \sum_{j=1}^{2N+1} \mu_{outj} = \mu_{out1} + \mu_{out(2N+1)} \\ \sigma_{out}^2 &= \sum_{j=1}^{2N+1} \sigma_{outj}^2 \end{aligned} \quad (4.22)$$

4.5 Register Width Determination with Truncation

In practical applications, the output register width is truncated. Assuming that the output register width is B_{out} , the number of bits discarded at the output is:

$$B_{2N+1} = B_{max} - B_{out} + 1$$

Accordingly, the error variance caused by this output truncation is (propagation function is 1):

$$\sigma_{out(2N+1)}^2 = \sigma_{2N+1}^2 \quad (4.23)$$

As there is always an error introduced at this stage, it is not necessary to keep the intermediate stages at full B_{max} width, but one needs to decide what word length of every stage we have to keep. Hogenauer [5] used the following criteria: the sum of all the other $2N$ error sources at the output is less than or equal to this σ_{2N+1}^2 , and this sum of error is distributed equally among these $2N$ sources. As a result:

$$\sigma_{outj}^2 \leq \frac{1}{2N} \sigma_{2N+1}^2, \quad j = 1, 2, \dots, 2N. \quad (4.24)$$

Assuming b_j is the number of LSB's discarded at the j -th source, the truncation error is 2^{b_j} .

Combining this with equations (4.7) and (4.20), one obtains:

$$\sigma_{outj}^2 = \frac{1}{12} 2^{2b_j} F_j^2 \quad (4.25)$$

F_j is expressed in (4.20). From (4.24) and (4.25), one obtains:

$$2^{2b_j} F_j^2 \leq \frac{6}{N} \sigma_{2N+1}^2 \quad (4.26)$$

$$b_j \leq -\log_2 F_j + \log_2 \sigma_{2N+1} + \frac{1}{2} \log_2 \frac{6}{N}$$

b_j must be an integer, and for it we choose the largest integer not greater than the right hand side expression. This gives us the expression on the basis of which one can choose b_j :

$$b_j = \left\lfloor -\log_2 F_j + \log_2 \sigma_{2N+1} + \frac{1}{2} \log_2 \frac{6}{N} \right\rfloor \quad (4.27)$$

This equation is used to choose the LSB to be discarded (B_j) at every CIC decimation filter stage in the VHDL implementation.

4.6 Summary

In this chapter, two practical problems were discussed. The first problem concerned achieving a high throughput for the CIC decimation filter: a pipelined structure was proposed in all integrator stages to achieve this. The other problem was the determination of register width for each filter stage. As truncation at every intermediate register causes error, criteria must be adopted for this determination. In this work, criteria presented by Hogenauer is used. The criteria are as follows: 1) The filter is implemented with 2's complement arithmetic or

another number system that allows “wrap-around” between the most positive and the most negative numbers. In the CIC decimation filter implementation, 2’s complement representation is used; 2) The range of the number system is equal to or exceeds the maximum magnitude expected at the output of the composite filter. In the CIC filter implementation, the maximum magnitude expected at the filter output, namely $B_{\max}$ is a finite number, which can be calculated by means of equation (4.4). The maximum magnitude of all CIC filter stages are set to $B_{\max}$.

*Chapter 5***FUNCTIONAL IMPLEMENTATION OF THE CIC DECIMATION FILTER****5.1 Introduction**

Before actually implementing the CIC decimation filter, a functional verification of the proposed algorithm is needed. In this chapter, the design of a Matlab functional model for the CIC decimation filter is presented. This five-stage CIC decimation filter model is based on a pipelined structure rather than on Hogenauer's original structure. A series of test waveforms are used as inputs to this model and the output signal spectrum is analyzed and compared with that of the input. The functional performance of this model is also compared with that of the CIC structure in the Intersil HSP43220 [26] decimation digital filter modeled in Intersil's filter development software.

5.2 The Matlab Model of the CIC Decimation Filter

An accurate Matlab model of the CIC decimation filter enables functional verification of the proposed circuit structure. The CIC decimation filter in this model is composed of five integrator stages and five comb stages, and is a direct implementation of the structure in Figure 3.1. The decimation ratio is predefined in the program and can be modified.

A floating point mathematical algorithm which precisely describes the circuit is designed and incorporated into the model. On the other hand, as the model is intended to provide functional verification of the filtering and decimation functions, some practical

features of the actual implementation are neglected. In this model, no truncation or rounding is considered for the intermediate calculation results, which implies that all the internal register widths are of an arbitrarily large size. To make the algorithm simpler to implement, the calculations (addition and subtraction) are based on floating-point decimal representations instead of the 2's complement representation described in Chapter 4.

Decimal numbers representing input signals are first scaled and rounded to corresponding integers over the range of 0 to $2^{16}-1$ (equivalent to a 16 bit wide input). These decimal numbers are fed into the model and processed in accordance with the CIC algorithm, and no truncation or rounding is introduced during this process. After going through all five integrator-comb stages, the output signals are scaled and rounded to yield integers over a range of 0 to $2^{16}-1$ (equivalent to a 16 bit wide output). The input signal spectra and the corresponding output signal spectra are analyzed and compared.

The algorithms for the integrator and comb stages form the computational cores for the CIC decimation filter. Hogenauer used a non-pipelined structure shown in Figure 3.1. But as shown in Section 4.2, a pipelined structure has an advantage over a non-pipelined structure in terms of throughput. A pipelined structure is therefore used for the integrator section for this model. The algorithms for both structures are different. Figure 5.1 shows the algorithms for the pipelined and non-pipelined integrator stages.

Figure 5.1 Algorithm of pipelined and nonpipelined integrator stages

From the figure it can be observed that every integrator stage in both the pipelined and non-pipelined structures handle the same series of input data, but at different time instants. The outputs, therefore, should be the same. Comparing the two expressions for $V2(t)$, one can see that they are different, but $V1(t)$ and $V1(t-1)$ are actually the same set of data except for a one time unit delay. So, considering the latency of the pipelined structure both structures give the same results at the outputs. Simulation results on both algorithms also show that they are the same, without considering the initial values.

As the throughput problem is not critical for the comb stages, a non-pipelined structure is used. The algorithm block diagram representation for the comb stages is shown in Figure 5.2. Assuming the sampling rate change ratio is R , and every R -th output signal of the

integrator stages is fed into comb stages, the data signal enters into another clock domain, the clock rate of comb stages is R times slower than the integrator stages. $X(p)$ in Figure 5.2 is the output from the integrator stages.

Figure 5.2 Algorithm of comb stages

5.3 Functional Simulation Results

5.3.1 Single Tone Test

A series of input signals is used to test the functional model. One simple way of testing a decimation filter uses a sinusoidal waveform with a single frequency component. A separate Matlab program is used to generate these signals. In the next section, seven tests with different input signals are considered. The following is a summary of CIC decimation structures and input signal definitions used in these tests.

Test 1 is a single tone test, $N=5$, $R=8$.

Test 2 is a two tone test, $N=5$, $R=16$.

Test 3 is a two tone test, $N=5$, $R=4$.

Test 4 is a four tone test, $N=5$, $R=1023$.

Test 5 is a two tone test, with $N=3$, $R=16$.

Test 6 is a two tone test, with $N=4$, $R=16$.

Test 7 is a two tone test, with $N=5$, $R=16$.

In test 1, the decimation ratio (or the sampling rate change ratio) R is set to 8.

Assuming the input high sampling rate is F_s , the output low sampling rate F_L is F_s/R . The input signal is chosen as : $X(n) = \sin(2\pi f n)$, with n an integer from 0 to $16k$, and f is set to a representative value relative to sampling rate F_s .

In test 1, f is set to $(1/8R)*F_s = (1/64)*F_s = (1/8)*F_L$. As $f < 1/2*F_L$, the frequency component will be passed through the CIC decimation filter, and the theoretical frequency response is shown in Figure 3.5. The input signal and the decimated output signal spectra are shown in Figure 5.3. In Figure 5.3(a), the test signal is shown at $(1/64)*F_s = 0.0156F_s$, and the corresponding frequency in Figure 5.3(b) is shown at the position of $0.125F_L = (1/R)*F_s = (1/64)*F_s$.

5.3.2 Test With Two or Multiple Frequency Components

By testing with multiple frequency components we can see the relative magnitude of different frequency components after they are processed by the CIC filter- another way of showing the filter frequency response. In the next test the decimation ratio is set to an integer number, and two sinusoidal components of the same magnitude are used. The input signal is given by $X(n) = \sin(2\pi f_1 n) + \sin(2\pi f_2 n)$.

In test 2, $R=16$, f_1 is set to $(1/16)*F_L = (1/16R)*F_S = (1/256)*F_S \approx 0.0039F_S$, and f_2 is set to $F_L = (1/R)*F_S = (1/16)*F_S \approx 0.0625F_S$. Input and output spectra are shown in Figure 5.4. Note that the two input spectral components have the same magnitude. At the output the spectral component of frequency f_1 is shown at the position of $0.125F_S$, and component at f_2 is attenuated severely as $f_2 > (1/2)*F_L$. The attenuation corresponds to the low-pass frequency response of the CIC decimator shown in Figure 3.5.

In test 3, $R=4$, and f_1 and f_2 have the same value as in the first two tone test. f_1 is set to $(1/256)*F_S \approx 0.0039F_S$, and f_2 is set to $(1/16)*F_S \approx 0.0625F_S$. Input and output spectra are shown in Figure 5.5. The two input test signals have the same magnitude, and at the output the test signals have frequency components at $0.03125(4*1/256 = 1/64)*F_L$ and $0.25(4*1/16 = 1/4)*F_L$ with almost the same amplitude. This means that the pass band of the CIC low-pass filter is fairly flat, and little amplitude drop is introduced in the range of $(1/64) \sim (1/4)*F_L$. From Figure 3.5, note that when the frequency relative to low sampling rate is small, the gain variation is quite small even for pass band drop. From Table 3.1, it can be observed that for $N=5$, $f_c = (1/16)*F_L$ the pass band attenuation is only $0.28dB$. The size of the pass band droop depends on how many stages the CIC decimation filter is composed of. This topic is discussed in more detail in Section 5.3.3.

Test 4 is an extreme case, when R is set to 1023, the maximum decimation ratio implementation. Four frequency components with the same magnitude are used in this test. In this test f_1 is still set to $(1/8)*F_L = (1/8R)*F_S = (1/8184)*F_S \approx 0.12 \times 10^{-3} F_S$, f_2 is set to $(1/4)*F_L = (1/4R)*F_S = (1/4092)*F_S \approx 0.24 \times 10^{-3} F_S$, f_3 is set to $(1/2)*F_L = (1/2R)*F_S = (1/2046)*F_S \approx 0.49 \times 10^{-3} F_S$, and f_4 is set to $F_L = (1/R)*F_S = (1/1023)*F_S \approx 0.98 \times 10^{-3} F_S$. The simulation results

are presented in Figure 5.6. Here, the component at frequency f_1 is shown at $0.125F_s$, and component at f_2 is shown at $0.25F_s$ because it is in the pass band. The frequency component f_3 is right in the middle of the transition band, so it can be identified on the output spectrum, but with a lower magnitude than components at frequency f_1 and f_2 . In Figure 5.6 (b), the magnitude of the decimated signal at frequency f_3 is around 15dB lower than the decimated signals at frequency f_1 and f_2 , and is shown at $0.5F_L$. The frequency of signal f_4 is much larger than $(1/2)*F_L$, so it is attenuated by the CIC low-pass filter and is not shown in Figure 5.6(b). This is consistent with the CIC decimation filter frequency response in Figure 3.5.

The simulation results from test 1 to test 4 demonstrate that the functional CIC decimation filter model has a frequency response as depicted in Figure 3.5. Low frequency components in the input signals are kept and high frequency components are attenuated. The CIC decimation filter exhibits low-pass filter characteristics.

5.3.3 Testing a CIC Decimation Filter Composed of Different Numbers of Stages

The model discussed in sections 5.3.2 and 5.3.3 is a CIC decimation filter composed of five stages. In this section, a four-stage model and a three-stage model are built up and compared with the five-stage model by simulating with the same input signals.

The three-stage and the four-stage CIC decimation models are identical to the five-stage model, except for the number of integrator-comb pairs. The decimation ratio R is set to 16. An input signal with two frequency components of equal amplitude is used, $f_1 = (1/8)*F_L$, $f_2 = (3/4)*F_L$. Thus, components at frequency f_1 are in the pass band and components at frequency f_2 are in the transition band.

In test 5, the input signal is fed into a three-stage filter. The simulation results are shown in Figure 5.7. In test 6, the input is applied to a four-stage filter. The simulation results are shown in Figure 5.8. In test 7, the same input signal is applied to the five-stage filter. The results are shown in Figure 5.9. Comparing the decimated output signal spectra in Figures 5.7(b), 5.8(b) and 5.9(b), we note that as the number of stages increases, the frequency component at $0.25F_L$ (in the transition band) is more attenuated. When $N=3$, the magnitude of signal at frequency f_2 is about $-30dB$, when $N=4$, the magnitude is $-40dB$, and when $N=5$, the magnitude is $-50dB$. This means attenuation in the transition band is greater. This also conforms with Figure 3.5, where frequency responses of CIC decimation filters with different number of stages are compared. When N increases, higher transition band and stop band attenuation is observed, but at the same time, the flatness of pass band gets worse and worse. This is a trade-off that must be considered when designing a CIC decimation filter.

5.3.4 Comparison of Matlab Model Performance and Intersil HSP43220 Decimating Digital Filter Development Software Simulation Results

Intersil DECIMATE Development Software [26] is used to help DSP designers to select parameters for the HSP43220 Decimating Digital Filter (DDF). This DDF is composed of a CIC decimation filter and a FIR filter. The reason for investigating this software is to compare the performance of this CIC decimation filter model with the developments of this work. The software is used as follows: the user selects the design requirements such as input/output sampling rates, pass band width, transition band width, and pass band/stop band attenuation. The software automatically calculates the decimation ratio, the CIC filter order and the lowpass FIR filter parameters.

To make the comparison more accurate, the FIR filter was disabled. The design parameters were set as follows: the CIC decimation filter order was 5 and the decimation ratio was 16 (the same configuration as for test 7). The same input signal as used in test 7 was used with this software, and the output signal was saved and processed by the Matlab software. Note that the input signal has the same frequency components as the input signal used in test 7, and is generated by the HSP43220 software internally. The noise levels are different from the input power spectra in Figure 5.9(a). Figure 5.10 shows the input/output signal spectra generated by the DECIMATE software. The output spectrum depicted in Figure 5.10(b) is exactly the same as that in Figure 5.9(b), and the magnitude of the f_2 signal is also $-50dB$.

5.4 Summary

In this chapter, a functional model for the CIC decimation filter developed with the Matlab program was presented. The model provides an ideal estimation of the CIC decimation filter implementation. Internal register truncation was not considered and floating point arithmetic was used instead of 2's complement representation. A series of simulations was carried out using this model. The results of these tests are shown in Figures 5.3 – 5.10.

Figure 5.3 CIC decimation filter functional model simulation results for test 1. Configuration: $N=5$, $R=8$, $f_i = (1/8) \cdot F_L$: (a) Input signal spectrum; (b) Decimated output signal spectrum.

Figure 5.4 CIC decimation filter functional model simulation results for test 2. Configuration: $N=5$, $R=16$, $f_1 = (1/16)*F_L$, $f_2 = F_L$: (a) Input signal spectrum; (b) Decimated output signal spectrum.

Figure 5.5 CIC decimation filter functional model simulation results for test 3. Configuration: $N=5$, $R=4$, $f_1 = (1/16)*F_L$, $f_2 = (1/4)*F_L$:
 (a) Input signal spectrum; (b) Decimated output signal spectrum.

Figure 5.6 CIC decimation filter functional model simulation results for test 4. Configuration: $N=5$, $R=1023$, $f_1 = (1/8)*F_L$, $f_2 = (1/4)*F_L$, $f_3 = (1/2)*F_L$, $f_4 = F_L$: (a) Input signal spectrum; (b) Decimated output signal spectrum.

Figure 5.7 CIC decimation filter functional model simulation results for test 5. Configuration: $N=3$, $R=16$, $f_1 = (1/8)*F_L$, $f_2 = (3/4)*F_L$:
 (a) Input signal spectrum; (b) Decimated output signal spectrum.

Figure 5.8 CIC decimation filter functional model simulation results for test 6. Configuration: $N=4$, $R=16$, $f_1 = (1/8)*F_L$, $f_2 = (3/4)*F_L$:
 (a) Input signal spectrum; (b) Decimated output signal spectrum.

Figure 5.9 CIC decimation filter functional model simulation results

for test 7. Configuration: $N=5$, $R=16$, $f_1 = (1/8)*F_L$, $f_2 = (3/4)*F_L$:

(a) Input signal spectrum; (b) Decimated output signal spectrum.

Figure 5.10 Intersil HSP43220 decimation digital filter development software simulation results. Configuration: $N=5$, $R=16$, $f_1 = (1/8) \cdot F_L$, $f_2 = (3/4) \cdot F_L$: (a) Input signal spectrum; (b) Decimated output signal spectrum.

Chapter 6**FPGA TECHNOLOGY AND THE MENTOR GRAPHICS DESIGN ENVIRONMENT****6.1 Introduction**

A programmable logic device (PLD), also called a Field Programmable Logic Device (FPLD) is any type of integrated circuit that can be configured in the field by the end user for a logic design implementation [27]. User configurability of PLDs is the major difference between a PLD and a metallization programmable gate array (MPGA) or ASIC. A field programmable gate array (FPGA) is a narrower category of PLD which also includes complex PLDs (CPLDs). Modern FPGA technology provides a digital IC designer with powerful capabilities, shortens the design cycle, reduces costs and can support designs composed of a million gates or more. As circuits become larger and larger, it is becoming more and more difficult to design circuits manually, and hardware description languages such as VHDL and Verilog are being introduced and combined with other CAD tools.

In this chapter, the Mentor Graphics Design Environment and Altera FPGA technology are discussed. Details about the Altera APEX 20K Programmable Logic Device (PLD) family are also provided.

6.2 Altera FPGA Technology

Altera Corporation is a supplier of programmable logic devices (PLD). It delivers PLD devices and associated development tools. At present, Altera's System-on-a-Programmable-Chip™ solutions offer multi-million-gate PLD devices, and allow the integration of a large system in a single chip. It should be noted that Altera prefers using the term PLD rather than FPGA (Field Programmable Gate Array) for its products in order to differentiate them from other companies' products.

6.2.1 Altera APEX 20K Device Architecture

The Altera APEX 20K PLD family [28] is one of the industry's first PLDs to have the capability of System-on-a-Programmable-Chip™ integration. It currently offers high density devices with maximum gate numbers from 200,000 up to one million. It is also the first PLD with MultiCore architecture, which integrates look-up table (LUT) logic, product-term logic and embedded memory. LUT is suitable for data-path, register-intensive, or DSP designs. Product-term-based logic is optimized for complex combinatorial paths such as those found in state machines. Figure 6.1 (from reference [28]) shows the block diagram of the APEX 20K family device. The APEX 20K devices are fabricated on a 0.22 micron, six metal layer CMOS process with an operating voltage of 2.5V. APEX 20KE devices are manufactured on a 0.18/0.15 micron, six/seven metal layer process with an supply voltage of 1.8V.

Figure 6.1 APEX 20K device block diagram [28]

There are three major categories of FPGAs: Complex Programmable Logic Devices (CPLDs), Static RAM Field Programmable Gate Arrays (SRAM FPGAs) and Antifuse FPGAs. The Altera APEX 20K device uses the static RAM technology. Figure 6.2 shows the logic elements of the APEX 20K devices.

Antifuse technology allows an anti-fuse to stay at high impedance or be programmed into low impedance; this is a program-once device. CPLDs are composed of many microcells interconnecting with each other, where each cell has programmable and-or-term architecture to implement combinational logic, programmable registers and programmable I/Os. All the settings are stored in the on-chip programmable EPROM. Compared to CPLDs, the SRAM FPGA has a different structure. It has a matrix of logic elements (LEs) interconnected by configurable connections and surrounded by configurable I/Os. Altera APEX 20K devices use this kind of structure. APEX is composed of many MegaLAB structures, with each MegaLAB containing 16 LAB (logic array block) and 1 ESB (embedded system block, performing product-term logic). Each LAB has 10 LEs. In Figure 6.2, each LE has four-input look-up tables (LUTs), a carry chain, a cascade chain and a programmable flip-flop. The LUTs

implement combinational and sequential logic functions and a network that can be programmed by the user. As the system powers up, configurable data from EPROM, RAM or an external device is downloaded to the specific devices.

Figure 6.2 APEX 20K logic element

6.2.2 Design Flow of APEX 20K FPGA

LeonardoSpectrum is a product of Exemplar Logic Inc., a Mentor Graphics Company. It is a high-level design tool for FPGAs or ASICs. It provides users with many functions including design capture, HDL entry, logic synthesis, circuit optimization, timing analysis, and placement and routing. In brief, LeonardoSpectrum accepts HDL hierarchical descriptions, optimizes them, and then maps them into APEX 20K FPGA architecture.

Figure 6.3 (from reference [28]) shows the design flow for the Altera APEX 20K with LeonardoSpectrum. Input design is translated into intermediate data structure format, and optimization is performed, to make sure that the inputs of combinational functions are suitably limited in number. This is called fanin-limited optimization. Then, the design is subsequently mapped into lookup tables (LUTs). Most synthesis tools stop here, but LeonardoSpectrum allows the user to set time constraints and do timing optimization. After that, a hierarchical EDIF netlist is created, and vendor tools are used to do the place-and-route. For Altera APEX 20K, MAX+PLUS Place-and-Route tools are used for this purpose. LeonardoSpectrum can also back-annotate the post place-and-route timing, and generate a SDF (Standard Delay Format) timing file and VHDL netlist at the same time. These files can be re-simulated for functional verification.

6.2.3 Altera APEX 20K FPGA Device Features

For this research, the Altera APEX 20K family, EP20K400 FPGA is used. Its main features include:

Maximum system gate count: 1,052,000

Typical gate count: 400,000

LEs: 16,640

ESBs: 104

Maximum RAM bits: 212,992

Maximum macrocells: 1,664

Maximum user I/O pins: 502

Internal supply voltage: 2.5V

MultiVoltage I/O interface voltage levels: 2.5V, 3.3V

Figure 6.3 Simplified design flow

6.3 Mentor Graphics Design Environment

Mentor Graphics Corp., established in 1981, is a major electronics design automation (EDA) companies in the world. It offers design tools and consulting services to the electronics

and semiconductor industry. Its products include Design Architect, Renoir, QuickSim, ModelSim, and Leonardo.

Renoir is a HDL graphical design environment that can generate VHDL/Verilog code from state machines, flow charts, truth tables, and block diagrams and more.

ModelSim is a digital simulation tool that can be combined with Mentor Graphics Renoir to provide an integrated design capture/simulation environment. Today, the ModelSim simulator occupies much of the VHDL simulation market for ASICs and FPGAs.

LeonardoSpectrum is one of the few tools that can design both FPGAs and ASICs. As noted in 6.2.2, it is mainly used for FPGA logic synthesis in this work.

6.4 Summary

This chapter introduced the design process of the Altera APEX 20K FPGA based on the Mentor Graphics design environment. The design process consists of VHDL design capture by Renoir, digital simulation with ModelSim, LeonardoSpectrum for logic synthesis (generating EDIF netlist), and finally, place-and-route with MAX+PLUS tools.

Chapter 7

VHDL IMPLEMENTATION OF THE CIC DECIMATION FILTER AND ITS FPGA REALIZATION

7.1 Introduction

In this chapter, the CIC decimation filter implementation using VHDL code is presented. The design is based on a fixed-point mathematical representation. The design is partitioned into a few building blocks. RTL coding is done on each individual building block and verification is performed on the whole CIC decimation filter design for different configurations. Finally the design is synthesized and mapped onto an Altera FPGA device and testability of the chip is discussed.

7.2 Design Specification

The goal of this research is to implement a configurable CIC decimation filter. “Configurable” means that the input/output word-length can be programmed in the VHDL code, as can the maximum decimation ratio R_{max} , and the implementation of all the filter stages is optimized based on the specific configuration. After the VHDL code is synthesized into an actual chip, the decimation ratio must be programmed in the range from 1 to R_{max} .

Specification of the configurable CIC decimation filter:

Number of integrator-comb pairs: 5 (5 is chosen to provide a good trade-off between the pass-band droop and the stop-band attenuation according to Hogenauer [5]. Harris

HSP50214 [14], Graychip GC4016 [17] and Analog Devices AD6620 [18], all use a 5th order CIC decimation filter).

Input data word length: 8-16 bits; Output data word length: 16-24 bits; Maximum decimation ratio: 1023; Programming range of decimation ratios: $1-R_{\max}$ (the biggest possible $R_{\max}$ value is 1023)

7.3 VHDL Implementation

7.3.1 Block Diagram

The CIC decimation filter design was partitioned into a few functional blocks. Figure 7.1 shows the block diagram of the design.

Figure 7.1 Simplified block diagram of the CIC decimation filter

The “clock divider” block generates a clock $CLKR$ with $1/R$ frequency of the input clock CLK . The integrator section runs at the normal clock rate CLK , and the comb section

runs at the generated clock rate $CLKR$. The shifter block matches the input data with the integrator section when the decimation ratio is changed. For example, the width of the first integrator stage is 62 bits, and the input signal word length is 12 bits. When running under different decimation ratios, this 12 bit input signal is placed in different positions with respect to the 62 bit first integrator stage. Another *RESET* signal is used to reset the registers in the data path to zero. The standard working process for running the CIC decimation filter chip is as follows: first, add a signal to R input port for the decimation ratio programming. Then a high *Load* signal enables the R input write into the clock divider circuit. Once the *Load* signal becomes low, the clock divider begins to run, and the $CLKR$ clock is generated. Now the CIC decimation filter is ready to accept input data signals from *Data_in* ports. Data output is observed at the *Data_out* ports.

7.3.2 The Process of Designing a Configurable CIC Decimation Filter

There is a general process that must be followed when designing a configurable CIC decimation filter using FPGA technology:

- 1) Select the proper number of stages based on the frequency response requirements. In this design this step is skipped as the number of stages has been fixed at 5. Generally speaking, when designing a configurable CIC decimation filter, this step should be considered. The Harris/Intersil HSP43220 Decimating Digital Filter Development Software [26] calculates the number of CIC stages and the parameters of a FIR filter which is connected in series with the CIC decimation filter, based on the user's requirement for decimation ratio, pass band width, transition band width and pass band/ stop band attenuation.
- 2) Calculate the internal register widths. As discussed in Chapter 4, truncation in the internal registers will cause error, so proper selection of the internal register widths is important to

control this error. A Matlab program to do this calculation has been designed based on the algorithm described in Chapter 4. The program requires the maximum decimation ratio, input/output word length, and the number of integrator-comb pairs to calculate the internal register widths.

- 3) Set the input/output word lengths and all the internal register word lengths in the original VHDL/Verilog code. Compile and synthesize the design using CAD tools, and finally map it onto an FPGA chip. For example, in this implementation, $N=5$ (there are five integrator stages and five comb stages), $R_{max}=1000$, Input word length=12 bits and Output word length=20 bits. The Matlab program calculates the internal register widths as: Int1=62, Int2=57, Int3=47, Int4=39, Int5=30, Comb1=26, Comb2=25, Comb3=24, Comb4=23, Comb5=23. These numbers must then be written into the VHDL code. Every time the configuration changes (R_{max} and/or input/output word length change), these numbers must be re-calculated and re-programmed in VHDL code.

7.3.3 Verification of the VHDL Implementation

Verification is extremely important in an ASIC/FPGA design flow. In this work, verification was performed on every individual building block, for example, the integrator section, comb section, and clock divider. A testbench was written for each block, and simulation results were compared to calculated results. After the separate blocks were integrated, a series of tests was carried out on the whole design to verify the functioning of the design. The process is shown in Figure 7.2.

components of the input and output signal spectra, we can easily tell if the CIC decimation filter is working or not, no matter what the configuration is. The author believes that there is still work that could be done to improve the testing of this design, for example, through the implementation of a built-in self-test and algorithms to determine the best test vectors. Examples of CIC filter tests will be presented in the next section.

7.4 Simulation Results

Simulation 1: The configuration of the CIC decimation filter to be tested is as follows: $N=5$, $R_{\max}=1000$, input word length=12 bits, and output word length=20 bits. The test signal consists of the sum of two sinusoidal waveforms with different frequencies. The two waveforms have the same amplitude, $f_1 = (1/16)*F_L$, $f_2 = F_L$ (F_L stands for low sampling rate, F_H stands for high sampling rate). The input signal and its spectrum are shown in Figure 7.5 (at the end of this chapter). “Quantized Signal” in this simulation means that the signal has been rounded in the time domain. The actual decimation ratio R is set to 16, and the two frequency components are present at $(1/256)*F_H \approx 0.004 F_H$, and at $(1/16)*F_H = 0.0625 F_H$ in the frequency domain. The output signal and its spectrum are shown in Figure 7.6. It is shown that only one frequency component f_1 is present. The decimation filter eliminates f_2 because f_2 is much bigger than $F_L/2$. The peak at $(1/16)*F_L = 0.0625 F_L$ in Figure 7.6(b) shows the position of the frequency component f_1 . By comparing this simulation result with Figure 5.4, the VHDL code simulation gives the same result as the functional model under this specific configuration.

Simulation 2: The configuration of CIC decimation filter is changed to: $N=5$, $R_{\max}=64$, input word length=8, and output word length=16. A decimation ratio R of 4 is selected to test

the system when R is small. The test signal is the addition of two sinusoid waveforms with different frequencies, $f_1 = (1/16) * F_L = (1/64) * F_H \approx 0.016 F_H$, $f_2 = (1/4) * F_L = (1/16) * F_H = 0.0625 F_H$. The input signal and its spectrum are shown in Figure 7.7. In this simulation, the input signal has been rounded to 8 bits, and its spectrum is shown in Figure 7.7(b). The output signal and its spectrum are shown in Figure 7.8. It is shown that both frequency components are present because $f_1, f_2 < F_L/2$. f_1 is at the position of $(1/16) * F_L = 0.0625 F_L$, f_2 is at the position of $(1/4) * F_L = 0.25 F_L$ in the frequency domain. Figure 7.8(b) shows the same result as in Figure 5.5(b).

Simulation 3: the configuration of the CIC decimation filter is changed to: $N=5$, $R_{\max}=1023$, input word length=16, and output word length=24. The test signal is the sum of four sinusoid waveforms with different frequencies, $f_1 = (1/8) * F_L$, $f_2 = (1/4) * F_L$, $f_3 = (1/2) * F_L$, and $f_4 = F_L$. These four sinusoid waveforms have the same amplitude. The input signal and its spectrum are shown in Figure 7.9. The input signal has been rounded to 16 bits. R is selected to be $R_{\max}=1023$ to test the extreme case. The four frequency components show their positions at $(1/8R) * F_H = 1/8184 * F_H \approx 1.22 \times 10^{-4} F_H$, $(1/4R) * F_H = (1/4092) * F_H \approx 2.44 \times 10^{-4} F_H$, $(1/2R) * F_H = (1/2046) * F_H \approx 4.89 \times 10^{-4} F_H$ and $(1/R) * F_H = (1/1023) * F_H \approx 9.78 \times 10^{-4} F_H$ respectively in Figure 7.9(b). The output from the CIC decimation filter is shown in Figure 7.10. The signals f_1 and f_2 as shown in Figure 7.10(b), are located at $1/8 F_L = 0.125 F_L$ and $1/4 F_L = 0.25 F_L$ with the same amplitude. As $f_1, f_2 < F_L/2$ both signals are in the pass band. Frequency component f_3 could be identified at $0.5 F_L$ but with lower amplitude than f_1 and f_2 as it is in the transition band. Frequency component f_4 is filtered out from the spectrum as it is in the stop band. This result also conforms to the functional simulation result in Figure 5.6. As the decimation ratio is 1023, every 1023rd input signal generate one output signal, so

calculating many output signals will be very time-consuming. It is for this reason that a small number of output signal samples were used in Figure 7.10 (b), and the resulting waveform shows poor resolution.

7.5 Synthesis and Optimization Consideration

After verification, Mentor Graphics' Leonardo is used to synthesize the RTL code into an Altera APEX FPGA. The process of using the Leonardo tool can be described in five steps:

Step 1: Set up input files and assign a working directory. There could be several input files, but they must be read in proper sequence.

Step 2: Select the technology and device. In this case, Altera APEX20K technology is selected.

Step 3: Set global constraints. For example, specify clock frequency and so on.

Step 4: Set output files. Specify the location and format of the FPGA netlist and synthesis summary file.

Step 5: Review and start running Leonardo.

The critical path of the integrator stages in the CIC decimation filter determines throughput. "Throughput" here means the highest clock rate that can be used under this specific configuration. The following is the critical path analysis of the CIC decimation filter. The greater the circuit delay between two adjacent registers (denoted as Z^{-1} blocks in Figure 4.1), the lower the throughput of the CIC decimation filter. In the pipelined structure shown in

Figure 4.1, circuits between adjacent registers are basically multi-bit adders. When $R_{\max}$ or In/Out word length increases (both of them will increase the word length of internal multi-bit adders), throughput decreases, which means that only a slower clock can be used. This results since larger size adders introduce more propagation delay. Figure 7.3 shows the propagation delay in the integrator stages when $R_{\max} = 1000$, and In/Out = 12/20 bits. Based on Matlab calculation results, Int1=62, Int2=57, Int3=47, Int4=39, Int5=30 (Int k is the register width of the k th integrator stage). So the adder in the first integrator stage is a 62 bit adder, the second is a 57 bit adder and so on. A 62 bit adder causes more delay than a 57 bit adder, the first integrator stage is the bottleneck for throughput improvement.

Figure 7.3 Propagation delay in integrator stages of the CIC decimation filter

In Figure 7.3, if the propagation delay of the first integrator stage is T_1 , the propagation delay of the second integrator stage is T_2 and so on, the minimum clock cycle T equals to T_1 as T_1 is the largest delay of all integrator stages. Table 7.1 shows the critical block delay, which is also the propagation delay of the first integrator stage. As $R_{\max}$ and In/Out word lengths increase, the delay of the critical block becomes longer and longer, since the adder size of the first integrator stage also increases.

Table 7.1 Propagation delay of the first integrator stage:
Critical block delay of the CIC decimation filter

$R_{\max}$ In/Out	8	16	64	256	1023
16/24 (bits)	9	11	14	17	20
12/20 (bits)	8	9	12	15	19
8/16 (bits)	7	8	11	14	17

Critical Block Delay (ns)

Table 7.2 shows the synthesis results under different configurations. Table 7.2(a) shows the throughput, resulting from the timing analysis done by Leonardo. The worst condition occurs when $R_{\max} = 1023$, and In/Out = 16/24 bits, for which the throughput is only 5MHz. A clock rate higher than this can not be used. Table 7.2(b) shows the number of Logic Cells (LC) used in the Altera APEX20K by the CIC decimation filter design. This gives some idea of the design size. The total number of LCs in an Altera EP20K400BC652 FPGA is 16640. So, for the worst case ($R_{\max} = 1023$, In/Out = 16/24 bits), 1847 LCs are used, which is only 11.10% of all the LCs. For I/O pins, only 53 out of 652 are used (8.13%). This indicates that the FPGA chip can hold larger designs.

7.6 Testability Issues

Testing is very important in VLSI circuit and system design. Tsui[29] has stated, "Testing, in its broadest sense, means to examine (the whole and all parts of) a product, to ensure that it functions and exhibits the properties and capabilities it was designed to possess."

This work presented certain configurations of the CIC decimation filter that were very difficult to test in software simulations. When the decimation ratio R is large, say 1000, then 1000 input samples are needed to get one output signal sample. Thus it will take a long time to compute and store the result of inputting 1000 signals, so testing the whole CIC decimation filter becomes a time consuming process. In other words, the observability of some parts of the CIC decimation filter (for example, the integrator section) is poor. Thus, it is worthwhile to discuss methods that can improve the testability of the CIC decimation filter.

Usually, testability can be improved by adding additional circuitry, and partitioning a system into parts and establishing observable test points within the system. We can partition the CIC decimation filter into several functional blocks (the CIC filter has been divided into blocks such as the integrator, comb, shifter, and clock divider), and increase the observability of the integrator section by adding some simple multiplexers. This is shown in Figure 7.4.

Figure 7.4 Testability of the CIC decimation filter

Three multiplexers can be added to the CIC decimation filter circuit, controlled by signals T_1 , T_2 and T_3 respectively. When signal T_1 is high, T_2 is low and T_3 is high, the CIC filter is functioning under normal operating conditions. If T_2 is high, then the shifter and integrator blocks can be tested. If signals T_1 , T_2 , and T_3 are low, the comb section can be tested, and the clock of the comb is the input clock CLK , instead of $CLKR$. This ensures that the testing of the comb section has no relation to the programming of R . The $CLKR$ port is used to examine the working condition of the clock divider. Although very simple, this proposed structure enables the user to test the functional blocks separately. Users can observe the integrator block directly from the output port *Data_out*.

7.7 Summary

This chapter discussed the realization of the CIC decimation filter by RTL coding, and the verification performed on the design before synthesis onto an Altera APEX20K FPGA chip. It was shown that the CIC filter design operates correctly on the basis of simulation results. Different configurations of the CIC filter lead to different performance levels (throughput and gate-count). Methods to improve the testability of this design were also discussed.

Table 7.2 Synthesis results of the CIC decimation filter based on the Altera device

$\begin{matrix} R_{\text{max}} \\ \text{In/out} \end{matrix}$	8	16	64	256	1023
16/24 (bits)	11	9.4	7.3	6.0	5.0
12/20 (bits)	12.7	10.6	8.0	6.5	5.4
8/16 (bits)	15.1	12.3	8.9	7.0	5.8

(a) Throughput (MHz)

$\begin{matrix} R_{\text{max}} \\ \text{In/out} \end{matrix}$	8	16	64	256	1023
16/24 (bits)	1146	1388	1587	1724	1847
12/20 (bits)	954	1081	1351	1566	1702
8/16 (bits)	769	871	1101	1292	1414

(b) Number of LCs used

Figure 7.5 Input signal no.1 to the CIC decimation filter, $f_1 = (1/256) \cdot F_H$, $f_2 = (1/16) \cdot F_H$, input=12 bits: (a) Input signal in time domain; (b) Quantized input signal in frequency domain.

Figure 7.6 Simulation 1 of the CIC decimation filter VHDL code.

Configuration: $N=5$, $R_{\text{max}}=1023$, $R=16$ bits, output=20 bits: (a)

Decimated output signal in time domain; (b) Decimated output signal in frequency domain.

Figure 7.7 Input signal no.2 to the CIC decimation filter, $f_1 = (1/64) * F_H$, $f_2 = (1/16) * F_H$, input=8 bits: (a) Input signal in time domain; (b) Quantized input signal in frequency domain.

Figure 7.8 Simulation 2 of the CIC decimation filter VHDL code.

Configuration: $N=5$, $R_{\text{max}}=64$, $R=4$, output=16 bits: (a) Decimated output signal in time domain; (b) Decimated output signal in frequency domain.

Figure 7.9 Input signal no.3 to the CIC decimation filter, $f_1 = (1/8) \cdot F_L$, $f_2 = (1/4) \cdot F_L$, $f_3 = (1/2) \cdot F_L$, $f_4 = F_L$ input=16 bits: (a) Input signal in time domain; (b) Quantized input signal in frequency domain.

Figure 7.10 Simulation 3 of the CIC decimation filter VHDL code.

Configuration: $N=5$, $R_{\text{max}}=1023$, $R=1023$, output=24 bits: (a)

Decimated output signal in time domain; (b) Decimated output signal in frequency domain.

*Chapter 8***SUMMARY AND SUGGESTED FURTHER DEVELOPMENT****8.1 Thesis Project Summary**

The cascaded integrator comb (CIC) decimation filter structure was the research topic in this thesis project. The principle of the CIC decimation filter was investigated, a programmable CIC decimation filter with flexible rate change was implemented, and the design objectives described in section 1.3 were achieved. The specification of this design is listed as follows: Number of integrator-comb pairs: $N=5$; Maximum decimation ratio: $R_{\max}=1023$; Range of input data word length: 8-16 bits; Range of output data word length: 16-24 bits. The process of using this CIC decimation filter is as follows:

1. Select the maximum decimation ratio $R_{\max}$ and input/output data word length based on the user's design requirement.
2. Calculate all the intermediate register widths using the Matlab program.
3. Incorporate the input/output data word length and all the intermediate register widths into the VHDL code.
4. Synthesize this code into FPGA/ASIC, with the actual decimation ratio. The decimation ratio chosen can be any integer between 1 and $R_{\max}$.

The novelty of the design presented in this thesis is that it has the full flexibility to program the decimation ratio and the Input/Output word length. There is also no limitation on the decimation ratio, which can be any integer between 1 and $R_{\max}$. Previously reported CIC

decimation filter designs typically have limited decimation ratio range. Another advantage of the proposed design is that the circuit can be optimized for different configurations. Different configurations lead to different hardware realizations, meaning that the values of the I/O word length and the maximum decimation ratio R_{max} affect the internal register word lengths in the CIC decimation filter. This optimizes the circuit and redundant hardware is eliminated.

This design was synthesized onto an Altera APEX 20K FPGA by the Mentor Graphics Leonardo synthesis tool. Simulations show that CIC decimation filters with different configurations give different performances. The best performance occurs when both the In/Out word length and the decimation ratio are small. In a Leonardo simulation, when $R_{max}=8$ and In/Out=8/16 bits, the throughput can be as high as 15.1MHz. In comparison, when $R_{max}=1023$ and In/Out=16/24 bits, the throughput is only 5MHz.

8.2 Suggested Further Development

Though the objectives of the work have been achieved, there are several areas in which additional work could be done to improve the current design.

One improvement would be to extend the capability of the current design. The maximum decimation ratio of the current design is 1023, and the maximum In/Out word length is 16/24 bits. Both of these parameters could be increased, but this would require design modifications, such as additional calculations by the Matlab program.

Improving the throughput of the current design is also very important. The current throughput is not satisfactory under certain conditions. Compared with data sheets of some of the commercial products available on the market, a throughput of 5M-15MHz is relatively low.

For example, the throughput of the HSP43220 digital decimation filter can reach 35MHz [15]. There are several ways to improve the throughput, one of which is to synthesize and map the current design into different FPGA technologies or even into an ASIC technology. Normally, logic circuits realized in FPGAs are slower by two or three orders of magnitude than those realized in full-custom design. The Altera APEX 20K FPGA technology may not be the best for this specific CIC decimation filter implementation, since the chip selected was too large. Based on the synthesis results, less than 20% of all logic cells (LCs) are used. The mapping of a small design onto a big FPGA chip can result in longer connections between devices which may cause longer delays. Previous analysis on critical path of the CIC decimation filter has indicated that the overall throughput of the CIC decimation filter is determined by the first integrator stage which is a multi-bit adder. An improvement in the speed of this multi-bit adder will improve the throughput of the filter. Current research indicates that a 64-bit adder could achieve a computational delay as low as 470-670ps using CMOS technology [30] [31], compared with 20ns delay in the design (from Table 7.1, the first integrator stage is a 62-bit adder) presented here. From this comparison, investigation on the usage of high-speed adders in the CIC decimation filter implementation would be advantageous.

Another way to implement a decimation filter is to cascade one or several very simple half-band filters with the CIC decimation filter. The decimation process will be more efficient when decimation occurs in stages rather than in one step. Half-band filters or Equiband filters are particularly efficient in implementing sampling rate change [32]; they admit 2:1 sampling rate changes, though this could be a limitation for the total decimation ratio as it has to be an integer multiple of a power of 2. Goodman and Carey [33] developed a “nine filter structure” for decimation where eight of the nine filters are half-band filters. These half-band filters are

used as components of multistage interpolation or decimation filters. In their structure, a simple resampling technique is used to extend the range of designs to conversions between any two rates. Their techniques are considered useful for the improvement of the current design to increase throughput.

The methods used in this work can be applied to investigate the implementation of a CIC interpolation filter. The structure of the CIC interpolation filter presented by Hogenauer [5] is shown in Figure 8.1.

Figure 8.1 CIC interpolation filter

The structure of a CIC interpolation filter is very similar to a CIC decimation filter except that the comb section and the integrator section exchange positions. All the methods used to analyze a CIC decimation filter can be used for a CIC interpolation filter. It would be of interest to design a CIC interpolation filter with flexible interpolation ratio range and In/Out word lengths.

Once the design for an interpolation filter is implemented, a decimation filter or interpolation filter with a more flexible sample ratio change could be developed. The decimation /interpolation ratio would no longer need to be an integer, a rational number would be valid. The structure of the proposed decimation/interpolation filter could be as indicated in Figure 8.2:

Figure 8.2 Decimator/Interpolator with rational ratio change

The decimator/interpolator shown in Figure 8.2 is composed of two parts. One is a decimator with decimation ratio D and the other is an interpolator with interpolation ratio I , where D and I are integers and I/D is a rational number. The design and the implementation of such a filter would require investigation of such problems as the optimization of the overall circuits, and the design of additional FIR filters in front of the decimator, following the interpolator, or between the decimator and interpolator, to improve the frequency response of the whole structure.

References

1. R.E. Crochiere and L.R. Rabiner, *Multirate Digital Signal Processing*, Englewood Cliffs, Prentice Hall, New Jersey, 1983.
2. P.P. Vaidyanathan, *Multirate Systems and Filter Banks*, Englewood Cliffs, Prentice Hall, New Jersey, 1993.
3. H. Samuelli and T. Lin, "A VLSI Architecture for a Universal High Speed Multirate FIR Digital Filter with Selectable Power-of-Two Decimation/Interpolation Ratios", *Proceeding of the 1991 IEEE ICASSP*, pp.1813-1816.
4. H.Samuelli, T. Lin, R. Hawley and S. Olafson, "VLSI Architectures for a High-Speed Tunable Digital Modulator/Demodulator/Bandpass-Filter Chip Set", *Proceeding of the 1992 IEEE International Symposium on Circuits and Systems*, pp. 1065-1068.
5. E. B. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-29, pp. 155-162, April 1981.
6. Y. Djadi, T. A. Kwasniewski, C. Chan and V. Szwarc, "A High Throughput Programmable Decimation and Interpolation Filter", *Proceeding of the 1994 International Conference on Signal Processing Applications and Technology*, pp. 1743-1748.
7. A. Kwentus, O. Lee, and A. N. Willson, Jr., "A 250 Msample/sec Programmable Cascaded Integrator-Comb Decimation Filter", in *Proceeding VLSI Signal Processing*, IX, pp. 231-240, Oct.-Nov. 1996.
8. Kei-Yong Khoo, Zhan Yu and Alan N. Willson, Jr. , "Efficient High-Speed CIC Decimation Filter", *Proceedings Eleventh Annual IEEE International ASIC Conference*, pp. 251-254, June 1998.
9. H. J. Oh, S. Kim, G. Choi, and Y. H. Lee, "On the Use of Interpolated Second-Order Polynomials for Efficient Filter Design in Programmable Downconversion ", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 4, pp. 551-560, April 1999.
10. H. J. Oh and Y. H. Lee, "Multiplierless FIR Filters Based on Cyclotomic and Interpolated Second-order Polynomials with Powers-of-two Coefficients", in *Proceeding Midwest. Symp. Circuits Syst.*, Sacramento, CA, Aug. 1997.
11. H.-K. Yang and W. M. Snelgrove, "High Speed Polyphase CIC Decimation Filters", *Proceeding of 1996 IEEE International Conference On Communications*, pp. II.229-II.233, Atlanta, US, May 1996.

12. H.-K. Yang, "Low-Power Oversampled Signal Processing for Digital Radio Receivers", Ph.D. Thesis Report, Carleton University, March 1998.
13. Yonghong Gao, Lihong Jia and Hannu Tenhunen, "An Improved Architecture and Implementation of Cascaded Integrator-Comb Decimation Filters", 1999 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM 1999), pp. 317-320, Victoria, B.C., Canada, August 1999.
14. HSP50214 Programmable Downconverter Data Sheet, Harris Semiconductor, 1998.
15. HSP43220 Digital Decimation Filter Data Sheet, Harris Semiconductor, 1990.
16. Callie Riley, David Chester, Abbas Razavi, Fred Taylor and William Ricker, "High-Decimation Digital Filters", Proceeding of the 1991 IEEE ICASSP, pp. 1613-1616.
17. GC4014 Quad Receiver Chip Data Sheet, Graychip, 1996.
18. AD6620 65 MSPS Digital Receive Signal Processor Data Sheet, Analog Devices, 1998.
19. David B. Chester, "Digital IF Filter Technology for 3G Systems: An Introduction", IEEE Communications Magazine, pp. 102-107, February 1999.
20. N. J. Fliege, *Multirate Digital Signal Processing*, John Wiley&Sons, Chichester, England, 1994.
21. Proakis and Manolakis, *Digital Signal Processing, Principles, Algorithms, and Applications*, Prentice Hall, New Jersey, 1996.
22. MAX106 Data Sheet, Maxim Integrated Products, 1999.
23. AD9410 A/D Converter Data Sheet, Analog Device, 1999.
24. AD9203 A/D Converter Data Sheet, Analog Device, 1999.
25. Digital Signal Processing and Data Acquisition, PENTEK Product Catalog, 1999.
26. Intersil HSP43220 Decimating Digital Filter Development Software Data Sheet, May 1999 File Number: 3368.1.
27. Ashok K. Sharma, *Programmable Logic Handbook*, McGraw-Hill, 1998.
28. Altera APEX20K Programmable Logic Device Family Data Sheet, Altera Corporation, August, 1999.
29. Frank F. Tsui, *LSI/VLSI Testability Design*, McGraw-Hill, 1987.

30. Park Jaehong, H.C. Ngo, J.A. Silberman and S.H. Dhong, "470 ps 64-bit parallel binary adder [for CPU chip]", 2000 Symposium on VLSI Circuits, Digest of Technical Papers, pp. 192-193, June 2000.
31. Ramchan Woo, Se-Joong Lee and Hoi-Jun Yoo, "A 670 ps, 64 bit dynamic low-power adder design", 2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings, Vol.1, pp. 28-31, May 2000.
32. Darold W. Rorabacher, "Efficient FIR Filter Design for Sample Rate Reduction or Interpolation", IEEE Proceeding '75 ISCAS, pp. 396-399, 1975.
33. David J. Goodman, "Nine Digital Filters for Decimation and Interpolation", IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-25, No.2, April 1977.