Linköping Studies in Science and Technology
Dissertations, No 1420

On the Implementation of Integer and Non-Integer Sampling Rate Conversion

Muhammad Abbas

Linköpings universitet
TEKNISKA HÖGSKOLAN

Division of Electronics Systems
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping, Sweden

Linköping 2012

Linköping Studies in Science and Technology
Dissertations, No 1420

Muhammad Abbas
mabbas@isy.liu.se
www.es.isy.liu.se
Division of Electronics Systems
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping, Sweden

Abbas, Muhammad
On the Implementation of Integer and Non-Integer Sampling Rate Conversion
ISBN 978-91-7519-980-1
ISSN 0345-7524

Typeset with LaTeX 2ε
Printed by LiU-Tryck, Linköping, Sweden 2012

To my loving parents

Abstract

The main focus in this thesis is on the aspects related to the implementation of integer and non-integer sampling rate conversion (SRC). SRC is used in many communication and signal processing applications where two signals or systems having different sampling rates need to be interconnected. There are two basic approaches to deal with this problem. The first is to convert the signal to analog and then re-sample it at the desired rate. In the second approach, digital signal processing techniques are utilized to compute values of the new samples from the existing ones. The former approach is hardly used since the latter one introduces less noise and distortion. However, the implementation complexity for the second approach varies for different types of conversion factors. In this work, the second approach for SRC is considered and its implementation details are explored. The conversion factor in general can be an integer, a ratio of two integers, or an irrational number. The SRC by an irrational numbers is impractical and is generally stated for the completeness. They are usually approximated by some rational factor.

The performance of decimators and interpolators is mainly determined by the filters, which are there to suppress aliasing effects or removing unwanted images. There are many approaches for the implementation of decimation and interpolation filters, and cascaded integrator comb (CIC) filters are one of them. CIC filters are most commonly used in the case of integer sampling rate conversions and often preferred due to their simplicity, hardware efficiency, and relatively good anti-aliasing (anti-imaging) characteristics for the first (last) stage of a decimation (interpolation). The multiplierless nature, which generally yields to low power consumption, makes CIC filters well suited for performing conversion at higher rate. Since these filters operate at the maximum sampling frequency, therefore, are critical with respect to power consumption. It is therefore necessary to have an accurate and efficient ways and approaches that could be utilized to estimate the power consumption and the important factors that are contributing to it. Switching activity is one such factor. To have a high-level estimate of dynamic power consumption, switching activity equations in CIC filters are derived, which may then be used to have an estimate of the dynamic power consumption. The modeling of leakage power is also included, which is an important parameter to consider since the input sampling rate may differ several orders of magnitude. These power estimates at higher level can then be used as a feed-back while exploring multiple alternatives.

Sampling rate conversion is a typical example where it is required to determine the values between existing samples. The computation of a value between existing samples can alternatively be regarded as delaying the underlying signal by a fractional sampling period. The fractional-delay filters are used in this context to provide a fractional-delay adjustable to any desired value and are therefore suitable for both integer and non-integer factors. The structure that is used in the efficient implementation of a fractional-delay filter is know as Farrow structure or its modifications. The main advantage of the Farrow struc-

ture lies in the fact that it consists of fixed finite-impulse response (FIR) filters and there is only one adjustable fractional-delay parameter, used to evaluate a polynomial with the filter outputs as coefficients. This characteristic of the Farrow structure makes it a very attractive structure for the implementation. In the considered fixed-point implementation of the Farrow structure, closed-form expressions for suitable word lengths are derived based on scaling and round-off noise. Since multipliers share major portion of the total power consumption, a matrix-vector multiple constant multiplication approach is proposed to improve the multiplierless implementation of FIR sub-filters.

The implementation of the polynomial part of the Farrow structure is investigated by considering the computational complexity of different polynomial evaluation schemes. By considering the number of operations of different types, critical path, pipelining complexity, and latency after pipelining, high-level comparisons are obtained and used to short list the suitable candidates. Most of these evaluation schemes require the explicit computation of higher order power terms. In the parallel evaluation of powers, redundancy in computations is removed by exploiting any possible sharing at word level and also at bit level. As a part of this, since exponents are additive under multiplication, an ILP formulation for the minimum addition sequence problem is proposed.

Populärvetenskaplig sammanfattning

I system där digitala signaler behandlas så kan man ibland behöva ändra datahastigheten (samplingshastighet) på en redan existerande digital signal. Ett exempel kan vara system där flera olika standarder stöds och varje standard behöver behandlas med sin egen datahastighet. Ett annat är dataomvandlare som ut vissa aspekter blir enklare att bygga om de arbetar vid en högre hastighet än vad som teoretiskt behövs för att representera all information i signalen. För att kunna ändra hastigheten krävs i princip alltid ett digitalt filter som kan räkna ut de värden som saknas eller se till att man säkert kan slänga bort vissa data utan att informationen förstörs. I denna avhandling presenteras ett antal resultat relaterat till implementeringen av sådana filter.

Den första klassen av filter är så kallade CIC-filter. Dessa används flitigt då de kan implementeras med enbart ett fåtal adderare, helt utan mer kostsamma multiplikatorer som behövs i många andra filterklasser, samt enkelt kan användas för olika ändringar av datahastighet så länge ändringen av datatakten är ett heltal. En modell för hur mycket effekt olika typer av implementeringar förbrukar presenteras, där den största skillnaden jämfört med tidigare liknande arbeten är att effekt som förbrukas genom läckningsströmmar är medtagen. Läckningsströmmar blir ett relativt sett större och större problem ju mer kretsteknologin utvecklas, så det är viktigt att modellerna följer med. Utöver detta presenteras mycket noggranna ekvationer för hur ofta de digitala värdena som representerar signalerna i dessa filter statistiskt sett ändras, något som har en direkt inverkan på effektförbrukningen.

Den andra klassen av filter är så kallade Farrow-filter. Dessa används för att fördröja en signal mindre än en samplingsperiod, något som kan användas för att räkna ut mellanliggande datavärden och därmed ändra datahastighet gotdtyckligt, utan att behöva ta hänsyn till om ändringen av datatakten är ett heltal eller inte. Mycket av tidigare arbete har handlat om hur man väljer värden för multiplikatorerna, medan själva implementeringen har rönt mindre intresse. Här presenteras slutna uttryck för hur många bitar som behövs i implementeringen för att representera allt data tillräckligt noggrant. Detta är viktigt eftersom antalet bitar direkt påverkar mängden kretsar som i sin tur påverkar mängden effekt som krävs. Utöver detta presenteras en ny metod för att ersätta multiplikatorerna med adderare och multiplikationer med två. Detta är intressant eftersom multiplikationer med två kan ersättas med att koppla ledningarna lite annorlunda och man därmed inte behöver några speciella kretsar för detta.

I Farrow-filter så behöver det även implementeras en uträkning av ett polynom. Som en sista del i avhandlingen presenteras dels en undersökning av komplexiteten för olika metoder att räkna ut polynom, dels föreslås två olika metoder att effektivt räkna ut kvadrater, kuber och högre ordningens heltalsexponenter av tal.

Preface

This thesis contains research work done at the Division of Electronics Systems, Department of Electrical Engineering, Linköping University, Sweden. The work has been done between December 2007 and December 2011, and has resulted in the following publications.

Paper A

The power modeling of different realizations of cascaded integrator-comb (CIC) decimation filters, recursive and non-recursive, is extended with the modeling of leakage power. The inclusion of this factor becomes more important when the input sampling rate varies by several orders of magnitude. Also the importance of the input word length while comparing recursive and non-recursive implementations is highlighted.

⋆ M. Abbas, O. Gustafsson, and L. Wanhammar, "Power estimation of recursive and non-recursive CIC filters implemented in deep-submicron technology," in *Proc. IEEE Int. Conf. Green Circuits Syst.*, Shanghai, China, June 21–23, 2010.

Paper B

A method for the estimation of switching activity in cascaded integrator comb (CIC) filters is presented. The switching activities may then be used to estimate the dynamic power consumption. The switching activity estimation model is first developed for the general-purpose integrators and CIC filter integrators. The model was then extended to gather the effects of pipelining in the carry chain paths of CIC filter integrators. The correlation in sign extension bits is also considered in the switching estimation model. The switching activity estimation model is also derived for the comb sections of the CIC filters, which normally operate at the lower sampling rate. Different values of differential delay in the comb part are considered for the estimation. The comparison of theoretical estimated switching activity results, based on the proposed model, and those obtained by simulation, demonstrates the close correspondence of the estimation model to the simulated one. Model results for the case of phase accumulators of direct digital frequency synthesizers (DDFS) are also presented.

⋆ M. Abbas, O. Gustafsson, and K. Johansson, "Switching activity estimation for cascaded integrator comb filters," *IEEE Trans. Circuits Syst. I*, under review.

A preliminary version of the above work can be found in:

⋆ M. Abbas and O. Gustafsson, "Switching activity estimation of CIC filter integrators," in *Proc. IEEE Asia Pacific Conf. Postgraduate Research in Microelectronics and Electronics*, Shanghai, China, Sept. 22–24, 2010.

Paper C

In this work, there are three major contributions. First, signal scaling in the Farrow structure is studied which is crucial for a fixed-point implementation. Closed-form expressions for the scaling levels for the outputs of each sub-filter as well as for the nodes before the delay multipliers are derived. Second, a round-off noise analysis is performed and closed-form expressions are derived. By using these closed-form expressions for the round-off noise and scaling in terms of integer bits, different approaches to find the suitable word lengths to meet the round-off noise specification at the output of the filter are proposed. Third, direct form sub-filters leading to a matrix MCM block is proposed, which stems from the approach for implementing parallel FIR filters. The use of a matrix MCM blocks leads to fewer structural adders, fewer delay elements, and in most cases fewer total adders.

- ⋆ M. Abbas, O. Gustafsson, and H. Johansson, "On the implementation of fractional delay filters based on the Farrow structure," *IEEE Trans. Circuits Syst. I*, under review.

Preliminary versions of the above work can be found in:

- ⋆ M. Abbas, O. Gustafsson, and H. Johansson, "Scaling of fractional delay filters using the Farrow structure," in *Proc. IEEE Int. Symp. Circuits Syst.*, Taipei, Taiwan, May 24–27, 2009.
- ⋆ M. Abbas, O. Gustafsson, and H. Johansson, "Round-off analysis and word length optimization of the fractional delay filters based on the Farrow structure," in *Proc. Swedish System-on-Chip Conf.*, Rusthållargården, Arlid, Sweden, May 4–5, 2009.

Paper D

The computational complexity of different polynomial evaluation schemes is studied. High-level comparisons of these schemes are obtained based on the number of operations of different types, critical path, pipelining complexity, and latency after pipelining. These parameters are suggested to consider to short list suitable candidates for an implementation given the specifications. In comparisons, not only multiplications are considered, but they are divided into data-data multiplications, squarers, and data-coefficient multiplications. Their impact on different parameters suggested for the selection is stated.

- ⋆ M. Abbas and O. Gustafsson, "Computational and implementation complexity of polynomial evaluation schemes," in *Proc. IEEE Norchip Conf.*, Lund, Sweden, Nov. 14–15, 2011.

Paper E

The problem of computing any requested set of power terms in parallel using summations trees is investigated. A technique is proposed, which first generates

the partial product matrix of each power term independently and then checks the computational redundancy in each and among all partial product matrices at bit level. The redundancy here relates to the fact that same three partial products may be present in more than one columns, and, hence, all can be mapped to the one full adder. The testing of the proposed algorithm for different sets of powers, variable word lengths, and signed/unsigned numbers is done to exploit the sharing potential. This approach has achieved considerable hardware savings for almost all of the cases.

- ⋆ M. Abbas, O. Gustafsson, and A. Blad, "Low-complexity parallel evaluation of powers exploiting bit-level redundancy," in *Proc. Asilomar Conf. Signals Syst. Comp.*, Pacific Grove, CA, Nov. 7–10, 2010.

Paper F

An integer linear programming (ILP) based model is proposed for the computation of a minimal cost addition sequence for a given set of integers. Since exponents are additive for a multiplication, the minimal length addition sequence will provide an efficient solution for the evaluation of a requested set of power terms. Not only is an optimal model proposed, but the model is extended to consider different costs for multipliers and squarers as well as controlling the depth of the resulting addition sequence. Additional cuts are also proposed which, although not required for the solution, help to reduce the solution time.

- ⋆ M. Abbas and O. Gustafsson, "Integer linear programming modeling of addition sequences with additional constraints for evaluation of power terms," manuscript.

Paper G

Based on the switching activity estimation model derived in Paper B, the model equations are derived for the case of phase accumulators of direct digital frequency synthesizers (DDFS).

- ⋆ M. Abbas and O. Gustafsson, "Switching activity estimation of DDFS phase accumulators," manuscript.

The contributions are also made in the following publication but the contents are not directly relevant or less relevant to the topic of thesis.

- ⋆ M. Abbas, F. Qureshi, Z. Sheikh, O. Gustafsson, H. Johansson, and K. Johansson, "Comparison of multiplierless implementation of nonlinear-phase versus linear-phase FIR filters," in *Proc. Asilomar Conf. Signals Syst. Comp.*, Pacific Grove, CA, Oct. 26–29, 2008.

Acknowledgments

I humbly thank Allah Almighty, the Compassionate, the Merciful, who gave health, thoughts, affectionate parents, talented teachers, helping friends and an opportunity to contribute to the vast body of knowledge. Peace and blessing of Allah be upon the Holy Prophet MUHAMMAD (peace be upon him), the last prophet of Allah, who exhort his followers to seek for knowledge from cradle to grave and whose incomparable life is the glorious model for the humanity.

I would like to express my sincere gratitude towards:

- My advisors, Dr. Oscar Gustafsson and Prof. Håkan Johansson, for their inspiring and valuable guidance, enlightening discussions, kind and dynamic supervision through out and in all the phases of this thesis. I have learnt a lot from them and working with them has been a pleasure.
- Higher Education Commission (HEC) of Pakistan is gratefully acknowledged for the financial support and Swedish Institute (SI) for coordinating the scholarship program. Linköping University is also gratefully acknowledged for partial support.
- The former and present colleagues at the Division of Electronics Systems, Department of Electrical Engineering, Linköping University have created a very friendly environment. They always kindly do their best to help you.
- Dr. Kenny Johansson for introducing me to the area and power estimation tools and sharing many useful scripts.
- Dr. Amir Eghbali and Dr. Anton Blad for their kind and constant support throughout may stay here at the department.
- Dr. Kent Palmkvist for help with FPGA and VHDL-related issues.
- Peter Johansson for all the help regarding technical as well as administrative issues.
- Dr. Erik Höckerdal for proving the LaTeX template, which has made life very easy.
- Dr. Rashad Ramzan and Dr. Rizwan Asghar for their generous help and guidance at the start of my PhD study.
- Syed Ahmed Aamir, Muhammad Touqir Pasha, Muhammad Irfan Kazim, and Syed Asad Alam for being caring friends and providing help in proofreading this thesis.
- My friends here in Sweden, Zafar Iqbal, Fahad Qureshi, Ali Saeed, Saima Athar, Nadeem Afzal, Fahad Qazi, Zaka Ullah, Muhammad Saifullah Khan, Dr. Jawad ul Hassan, Mohammad Junaid, Tafzeel ur Rehman, and many more for all kind of help and keeping my social life alive.

- ⋆ My friends and colleagues in Pakistan especially Jamaluddin Ahmed, Khalid Bin Sagheer, Muhammad Zahid, and Ghulam Hussain for all the help and care they have provided during the last four years.
- ⋆ My parent-in-laws, brothers, and sisters for their encouragement and prays.
- ⋆ My elder brother Dr. Qaisar Abbas and sister-in-law Dr. Uzma for their help at the very start when I came to Sweden and throughout the years later on. I have never felt that I am away from home. Thanks for hosting many wonderful days spent there at Uppsala.
- ⋆ My younger brother Muhammad Waqas and sister for their prays and taking care of the home in my absence.
- ⋆ My mother and my father for their non-stop prays, being a great asset with me during all my stay here in Sweden. Thanks to both of you for having confidence and faith in me. Truly you hold the credit for all my achievements.
- ⋆ My wife Dr. Shazia for her devotion, patience, unconditional cooperation, care of Abiha single handedly, and being away from her family for few years. To my little princess, Abiha, who has made my life so beautiful.
- ⋆ To those not listed here, I say profound thanks for bringing pleasant moments in my life.

Muhammad Abbas
January, 2012,
Linköping Sweden

Contents

1 Introduction **1**
- 1.1 Digital Filters . . . 1
  - 1.1.1 FIR Filters . . . 1
  - 1.1.2 IIR Filters . . . 2
- 1.2 Sampling Rate Conversion . . . 3
  - 1.2.1 Decimation . . . 4
  - 1.2.2 Interpolation . . . 5
  - 1.2.3 Noble Identities . . . 7
- 1.3 Polyphase Representation . . . 8
- 1.4 Fractional-Delay Filters . . . 10
- 1.5 Power and Energy Consumption . . . 13

2 Finite Word Length Effects **17**
- 2.1 Numbers Representation . . . 17
  - 2.1.1 Two's Complement Numbers . . . 18
  - 2.1.2 Canonic Signed-Digit Representation . . . 19
- 2.2 Fixed-Point Quantization . . . 19
- 2.3 Overflow Characteristics . . . 21
  - 2.3.1 Two's Complement Addition . . . 21
  - 2.3.2 Two's Complement Multiplication . . . 21
- 2.4 Scaling . . . 22
- 2.5 Round-Off Noise . . . 23
- 2.6 Word Length Optimization . . . 23
- 2.7 Constant Multiplication . . . 24

3 Integer Sampling Rate Conversion **27**
- 3.1 Basic Implementation . . . 27
  - 3.1.1 FIR Decimators . . . 28
  - 3.1.2 FIR Interpolators . . . 29
- 3.2 Polyphase FIR Filters . . . 29
  - 3.2.1 Polyphase FIR Decimators . . . 30
  - 3.2.2 Polyphase FIR Interpolators . . . 31

3.3 Multistage Implementation . . . 31
3.4 Cascaded Integrator Comb Filters . . . 32
3.4.1 CIC Filters in Interpolators and Decimators . . . 33
3.4.2 Polyphase Implementation of Non-Recursive CIC . . . 35
3.4.3 Compensation Filters . . . 36

4 Non-Integer Sampling Rate Conversion 37
4.1 Sampling Rate Conversion: Rational Factor . . . 37
4.1.1 Small Rational Factors . . . 37
4.1.2 Polyphase Implementation of Rational Sampling Rate Conversion . . . 39
4.1.3 Large Rational Factors . . . 40
4.2 Sampling Rate Conversion: Arbitrary Factor . . . 41
4.2.1 Farrow Structures . . . 41

5 Polynomial Evaluation 45
5.1 Polynomial Evaluation Algorithms . . . 45
5.2 Horner's Scheme . . . 46
5.3 Parallel Schemes . . . 46
5.4 Powers Evaluation . . . 47
5.4.1 Powers Evaluation with Sharing at Word-Level . . . 48
5.4.2 Powers Evaluation with Sharing at Bit-Level . . . 49

6 Conclusions and Future Work 53
6.1 Conclusions . . . 53
6.2 Future Work . . . 53

7 References 55
References . . . 56

Publications 67

A Power Estimation of Recursive and Non-Recursive CIC Filters Implemented in Deep-Submicron Technology 69
1 Introduction . . . 72
2 Recursive CIC Filters . . . 72
2.1 Complexity and Power Model . . . 73
3 Non-Recursive CIC Filters . . . 76
3.1 Complexity and Power Model . . . 76
4 Results . . . 79
5 Conclusion . . . 80
References . . . 83

B Switching Activity Estimation of Cascaded Integrator Comb Filters **85**
1 Introduction . . . 88
2 CIC Filters . . . 89
3 CIC Filter Integrators/Accumulators . . . 90
3.1 One's Probability . . . 91
3.2 Switching Activity . . . 93
3.3 Pipelining in CIC Filter Accumulators/Integrators . . . 97
3.4 Switching Activity for Correlated Sign Extension Bits . . . 101
3.5 Switching Activity for Later Integrator Stages . . . 102
4 CIC Filter Combs . . . 103
4.1 One's Probability . . . 103
4.2 Switching Activity . . . 104
4.3 Arbitrary Differential Delay Comb . . . 106
4.4 Switching Activity for Later Comb Stages . . . 108
5 Results . . . 109
5.1 CIC Integrators . . . 109
5.2 CIC Combs . . . 110
6 Discussion . . . 111
7 Conclusions . . . 112
References . . . 116
A Double Differential Delay Comb . . . 118
A.1 Switching Activity . . . 118

C On the Implementation of Fractional-Delay Filters Based on the Farrow Structure **123**
1 Introduction . . . 126
1.1 Contribution of the Paper . . . 126
1.2 Outline . . . 127
2 Adjustable FD FIR Filters and the Farrow Structure . . . 127
3 Scaling of the Farrow Structure for FD FIR Filters . . . 128
3.1 Scaling the Output Values of the Sub-Filters . . . 129
3.2 Scaling in the Polynomial Evaluation . . . 130
4 Round-Off Noise . . . 133
4.1 Rounding and Truncation . . . 134
4.2 Round-Off Noise in the Farrow Structure . . . 136
4.3 Approaches to Select Word Lengths . . . 137
5 Implementation of Sub-Filters in the Farrow Structure . . . 143
6 Results . . . 144
6.1 Scaling . . . 145
6.2 Word Length Selection Approaches . . . 146
6.3 Sub-Filter Implementation . . . 148
7 Conclusions . . . 150
References . . . 151

D Computational and Implementation Complexity of Polynomial Evaluation Schemes 155
1 Introduction . . . 158
2 Polynomial Evaluation Algorithms . . . 159
2.1 Horner's Scheme . . . 159
2.2 Dorn's Generalized Horner Scheme . . . 159
2.3 Estrin's Scheme . . . 160
2.4 Munro and Paterson's Scheme . . . 161
2.5 Maruyama's Scheme . . . 162
2.6 Even Odd (EO) Scheme . . . 163
2.7 Li et al. Scheme . . . 164
2.8 Pipelining, Critical Path, and Latency . . . 164
3 Results . . . 165
4 Conclusions . . . 167
References . . . 172

E Low-Complexity Parallel Evaluation of Powers Exploiting Bit-Level Redundancy 175
1 Introduction . . . 178
2 Powers Evaluation of Binary Numbers . . . 178
2.1 Unsigned Binary Numbers . . . 178
2.2 Powers Evaluation for Two's Complement Numbers . . . 180
2.3 Partial Product Matrices with CSD Representation of Partial Product Weights . . . 180
2.4 Pruning of the Partial Product Matrices . . . 180
3 Proposed Algorithm for the Exploitation of Redundancy . . . 181
3.1 Unsigned Binary Numbers Case . . . 181
3.2 Examples . . . 182
3.3 Two's Complement Input and CSD Encoding Case . . . 182
4 Results . . . 183
5 Conclusion . . . 184
References . . . 188

F Integer Linear Programming Modeling of Addition Sequences With Additional Constraints for Evaluation of Power Terms 189
1 Introduction . . . 192
2 Addition Chains and Addition Sequences . . . 193
3 Proposed ILP Model . . . 194
3.1 Basic ILP Model . . . 194
3.2 Minimizing Weighted Cost . . . 195
3.3 Minimizing Depth . . . 196
4 Results . . . 196
5 Conclusions . . . 198
References . . . 201

G Switching Activity Estimation of DDFS Phase Accumulators 203
1 Introduction 206
2 One's Probability 207
3 Switching Activity 207
4 Conclusion 209
References 211

Chapter 1

Introduction

Linear time-invariant (LTI) systems have the same sampling rate at the input, output, and inside of the systems. In applications involving systems operating at different sampling rates, there is a need to convert the given sampling rate to the desired sampling rate, without destroying the signal information of interest. The sampling rate conversion (SRC) factor can be an integer or a non-integer. This chapter gives a brief overview of SRC and the role of filtering in SRC.

Digital filters are first introduced. The SRC, when changing the sampling rate by an integer factor, is then explained. The time and frequency-domain representations of the downsampling and upsampling operations are then given. The concept of decimation and interpolation that include filtering is explained. The description of six identities that enable the reductions in computational complexity of multirate systems is given. A part of the chapter is devoted to the efficient polyphase implementation of decimators and interpolators. The fractional-delay filters are then briefly reviewed. Finally, the power and energy consumption in CMOS circuits is described.

1.1 Digital Filters

Digital filters are usually used to separate signals from noise or signals in different frequency bands by performing mathematical operations on a sampled, discrete-time signal. A digital filter is characterized by its transfer function, or equivalently, by its difference equation.

1.1.1 FIR Filters

If the impulse response is of finite duration and becomes zero after a finite number of samples, it is a finite-length impulse response (FIR) filter. A causal

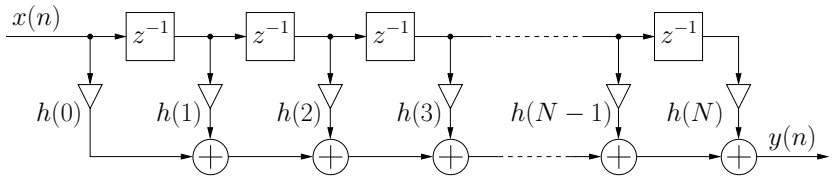


Figure 1.1: Direct-form realization of an N-th order FIR filter.

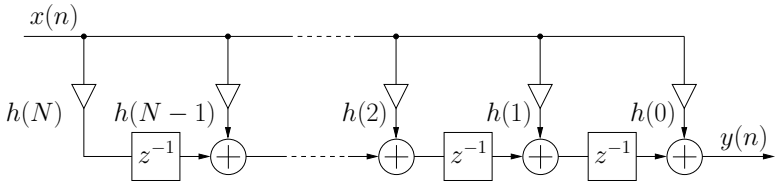


Figure 1.2: Transposed direct-form realization of an N-th order FIR filter.

FIR filter of order N is characterized by a transfer function $H(z)$, defined as [1, 2]

$$H(z) = \sum_{k=0}^{N} h(k) z^{-k}, \tag{1.1}$$

which is a polynomial in z^{-1} of degree N. The time-domain input-output relation of the above FIR filter is given by

$$y(n) = \sum_{k=0}^{N} h(k) x(n-k), \tag{1.2}$$

where $y(n)$ and $x(n)$ are the output and input sequences, respectively, and $h(0), h(1), \ldots, h(N)$ are the impulse response values, also called filter coefficients. The parameter N is the filter order and total number of coefficients, $N + 1$, is the filter length. The FIR filters can be designed to provide exact linear-phase over the whole frequency range and are always bounded-input bounded-output (BIBO) stable, independent of the filter coefficients [1–3]. The direct form structure in Fig. 1.1 is the block diagram description of the difference equation (1.2). The transpose structure is shown in Fig. 1.2. The number of coefficient multiplications in the direct and transpose forms can be halved when exploiting the coefficient symmetry of linear-phase FIR filter. The total number of coefficient multiplications will be $N/2 + 1$ for even N and $(N + 1)/2$ for odd N.

1.1.2 IIR Filters

If the impulse response has an infinite duration, i.e., theoretically never approaches zero, it is an infinite-length impulse response (IIR) filter. This type

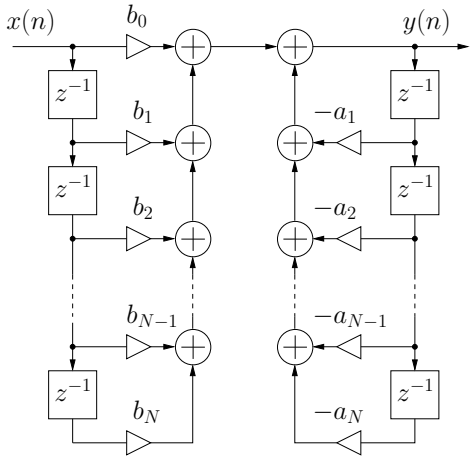


Figure 1.3: Direct-form I IIR realization.

of filter is recursive and represented by a linear constant-coefficient difference equation as [1, 2]

$$y(n) = \sum_{k=0}^{N} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k). \tag{1.3}$$

The first sum in (1.3) is non-recursive and the second sum is recursive. These two parts can be implemented separately and connected together. The cascade connection of the non-recursive and recursive sections results in a structure called direct-form I as shown in Fig. 1.3.

Compared with an FIR filter, an IIR filter can attain the same magnitude specification requirements with a transfer function of significantly lower order [1]. The drawbacks are nonlinear phase characteristics, possible stability issues, and sensitivity to quantization errors [1, 4].

1.2 Sampling Rate Conversion

Sampling rate conversion is the process of converting a signal from one sampling rate to another, while changing the information carried by the signal as little as possible [5–8]. SRC is utilized in many DSP applications where two signals or systems having different sampling rates need to be interconnected to exchange digital signal data. The SRC factor can in general be an integer, a ratio of two integers, or an irrational number. Mathematically, the SRC factor can be defined as

$$R = \frac{F_{out}}{F_{in}}, \tag{1.4}$$

where F_{in} and F_{out} are the original input sampling rate and the new sampling rate after the conversion, respectively. The sampling frequencies are chosen in such a way that each of them exceeds at least two times the highest frequency in the spectrum of original continuous-time signal. When a continuous-time signal $x_a(t)$ is sampled at a rate F_{in}, and the discrete-time samples are $x(n) = x_a(n/F_{in})$, SRC is required when there is need of $x(n) = x_a(n/F_{out})$ and the continuous-time signal $x_a(t)$ is not available anymore. For example, an analog-to-digital (A/D) conversion system is supplying a signal data at some sampling rate, and the processor used to process that data can only accept data at a different sampling rate. One alternative is to first reconstruct the corresponding analog signal and, then, re-sample it with the desired sampling rate. However, it is more efficient to perform SRC directly in the digital domain due to the availability of accurate all-digital sampling rate conversion schemes.

SRC is available in two flavors. For $R < 1$, the sampling rate is reduced and this process is known as decimation. For $R > 1$, the sampling rate is increased and this process is known as interpolation.

1.2.1 Decimation

Decimation by a factor of M, where M is a positive integer, can be performed as a two-step process, consisting of an anti-aliasing filtering followed by an operation known as downsampling [9]. A sequence can be downsampled with a factor of M by retaining every M-th sample and discarding all of the remaining samples. Applying the downsampling operation to a discrete-time signal, $x(n)$, produces a downsampled signal $y(m)$ as

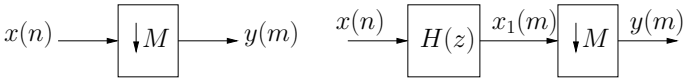
$$y(m) = x(mM). \tag{1.5}$$

The time index m in the above equation is related to the old time index n by a factor of M. The block diagram showing the downsampling operation is shown in Fig. 1.4a The sampling rate of new discrete-time signal is M times smaller than the sampling rate of original signal. The downsampling operation is linear but time-varying operation. A delay in the original input signal by some samples does not result in the same delay of the downsampled signal. A signal downsampled by two different factors may have two different shape output signals but both carry the same information if the downsampling factor satisfies the sampling theorem criteria.

The frequency domain representation of downsampling can be found by taking the z-transform to both sides of (1.5) as

$$Y(e^{j\omega T}) = \sum_{-\infty}^{+\infty} x(mM)e^{-j\omega Tm} = \frac{1}{M}\sum_{k=0}^{M-1} X(e^{j(\omega T - 2\pi k)/M}). \tag{1.6}$$

The above equation shows the implication of the downsampling operation on the spectrum of the original signal. The output spectrum is a sum of M uniformly shifted and stretched versions of $X(e^{j\omega T})$ and also scaled by a factor of



(a) M-fold downsampler. (b) M-fold decimator.

Figure 1.4: M-fold downsampler and decimation.

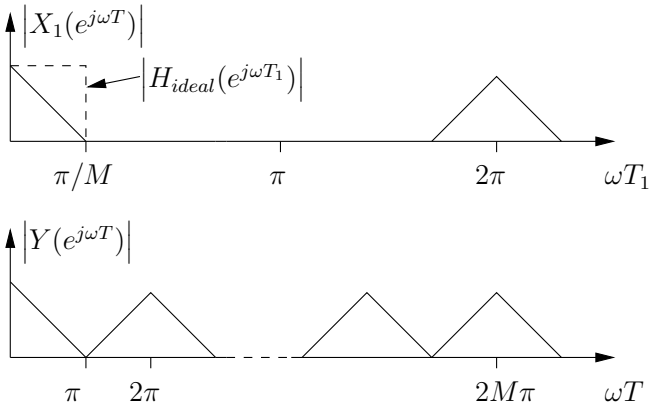


Figure 1.5: Spectra of the intermediate and decimated sequence.

$1/M$. The signals which are bandlimited to π/M can be downsampled without distortion.

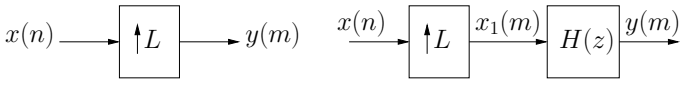
Decimation requires that aliasing should be avoided. Therefore, the first step is to bandlimit the signal to π/M and then downsampling by a factor M. The block diagram of a decimator is shown in Fig. 1.4b. The performance of a decimator is determined by the filter $H(z)$ which is there to suppress the aliasing effect to an acceptable level. The spectra of the intermediate sequence and output sequence obtained after downsampling are shown in Fig. 1.5. The ideal filter, as shown by dotted line in Fig. 1.5, should be a lowpass filter with the stopband edge at $\omega_s T_1 = \pi/M$.

1.2.2 Interpolation

Interpolation by a factor of L, where L is a positive integer, can be realized as a two-step process of upsampling followed by an anti-imaging filtering. The upsampling by a factor of L is implemented by inserting $L-1$ zeros between two consecutive samples [9]. An upsampling operation to a discrete-time signal $x(n)$ produces an upsampled signal $y(m)$ according to

$$y(m) = \begin{cases} x(m/L), & m = 0, \pm L, \pm 2L, \ldots, \\ 0, & \text{otherwise.} \end{cases} \tag{1.7}$$

A block diagram shown in Fig. 1.6a is used to represent the upsampling op-



(a) L-fold upsampler. (b) L-fold interpolator.

Figure 1.6: L-fold upsampler and interpolator.

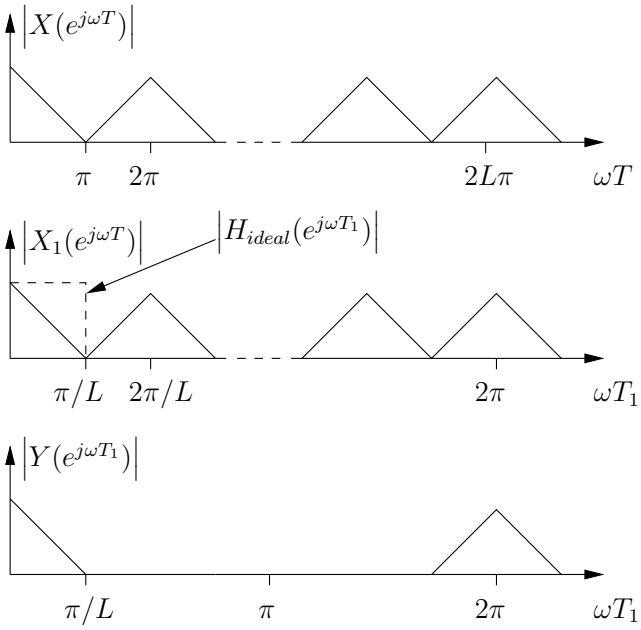


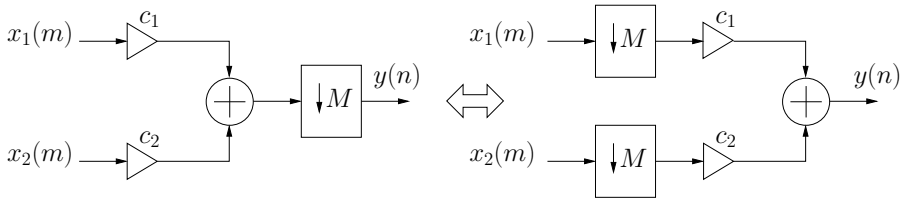
Figure 1.7: Spectra of the original, intermediate, and output sequences.

eration. The upsampling operation increases the sampling rate of the original signal by L times. The upsampling operation is a linear but time-varying operation. A delay in the original input signal by some samples does not result in the same delay of the upsampled signal. The frequency domain representation of upsampling can be found by taking the z-transform of both sides of (1.7) as

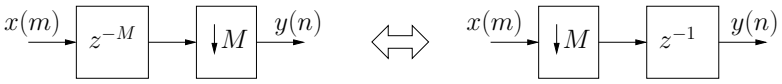
$$Y(e^{j\omega T}) = \sum_{-\infty}^{+\infty} y(m)e^{-j\omega Tm} = X(e^{j\omega TL}). \tag{1.8}$$

The above equation shows that the upsampling operation leads to $L-1$ images of the spectrum of the original signal in the baseband.

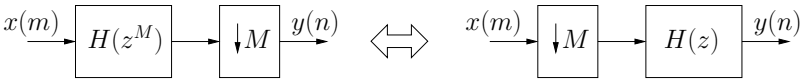
Interpolation requires the removal of the images. Therefore in first step, upsampling by a factor of L is performed, and in the second step, unwanted images are removed using anti-imaging filter. The block diagram of an interpolator is shown in Fig. 1.6b. The performance of an interpolator is determined



(a) Noble identity 1.



(b) Noble identity 3.



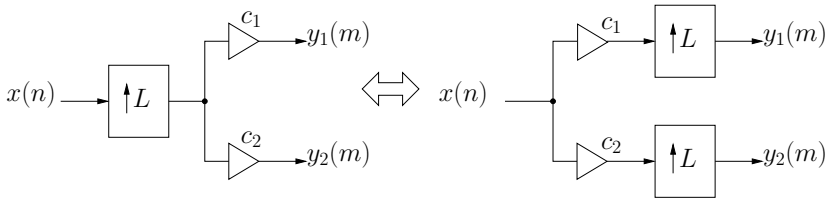
(c) Noble identity 5.

Figure 1.8: Noble identities for decimation.

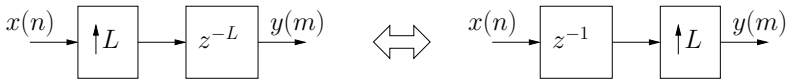
by the filter $H(z)$, which is there to remove the unwanted images. As shown in Fig. 1.7, the spectrum of the sequence, $x_1(m)$, not only contains the baseband of the original signal, but also the repeated images of the baseband. Apparently, the desired sequence, $y(m)$, can be obtained from $x_1(m)$ by removing these unwanted images. This is performed by the interpolation (anti-imaging) filter. The ideal filter should be a lowpass filter with the stopband edge at $\omega_s T_1 = \pi/L$ as shown by dotted line in Fig. 1.7.

1.2.3 Noble Identities

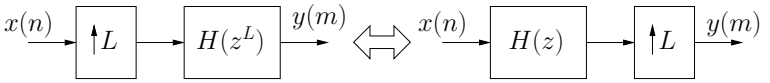
The six identities, called noble identities, help to move the downsampler and upsampler operations to a more desirable position to enable an efficient implementation structure. As a result, the arithmetic operations of additions and multiplications are to be evaluated at the lowest possible sampling rate. In SRC, since filtering has to be performed at the higher sampling rate, the computational efficiency may be improved if downsampling (upsampling) operations are introduced into the filter structures. In the first and second identities, seen in Figs. 1.9a and 1.8a, moving converters leads to evaluation of additions and multiplications at lower sampling rate. The third and fourth identities, seen in Figs. 1.9b and 1.8b, show that a delay of M (L) sampling periods at the higher sampling rate corresponds to a delay of one sampling period at the lower rate. The fifth and sixth identities, seen in Figs. 1.9c and 1.8c, are generalized versions of the third and fourth identities.



(a) Noble identity 2.



(b) Noble identity 4.



(c) Noble identity 6.

Figure 1.9: Noble identities for interpolation.

1.3 Polyphase Representation

A very useful tool in multirate signal processing is the so-called polyphase representation of signals and systems [5, 10]. It facilitates considerable simplifications of theoretical results as well as efficient implementation of multirate systems. To formally define it, an LTI system is considered with a transfer function

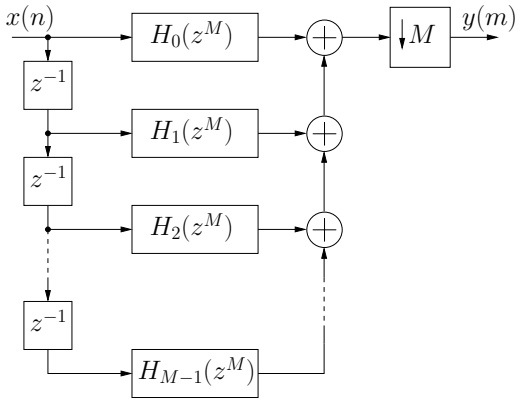
$$H(z) = \sum_{n=-\infty}^{+\infty} h(n) z^{-n}. \tag{1.9}$$

For an integer M, $H(z)$ can be decomposed as

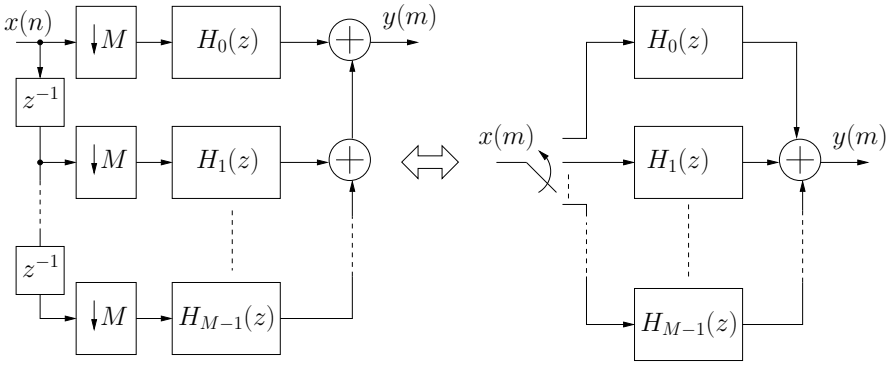
$$H(z) = \sum_{m=0}^{M-1} z^{-m} \sum_{n=-\infty}^{+\infty} h(nM+m) z^{-nM} = \sum_{m=0}^{M-1} z^{-m} H_m(z^M). \tag{1.10}$$

The above representation is equivalent to dividing the impulse response $h(n)$ into M non-overlapping groups of samples $h_m(n)$, obtained from $h(n)$ by M-fold decimation starting from sample m. The subsequences $h_m(n)$ and the corresponding z-transforms defined in (1.10) are called the Type-1 polyphase components of $H(z)$ with respect to M [10].

The polyphase decomposition is widely used and its combination with the noble identities leads to efficient multirate implementation structures. Since each polyphase component contains $M-1$ zeros between two consecutive samples and only nonzero samples are needed for further processing, zeros can be discharged resulting in downsampled-by-M polyphase components. The polyphase components, as a result, operate at M times lower sampling rate.



(a) Polyphase decomposition of a decimation filter.

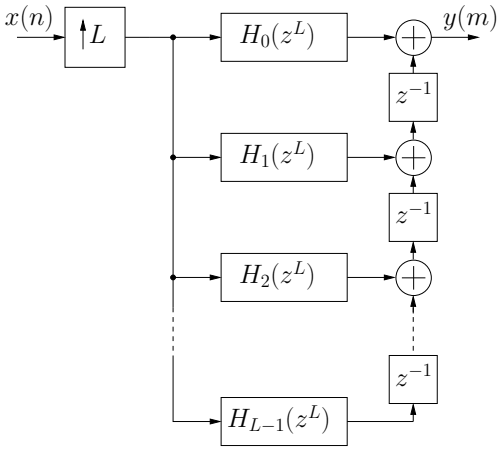


(b) Moving downsampler to before sub-filters and replacing input structure by a commutator.

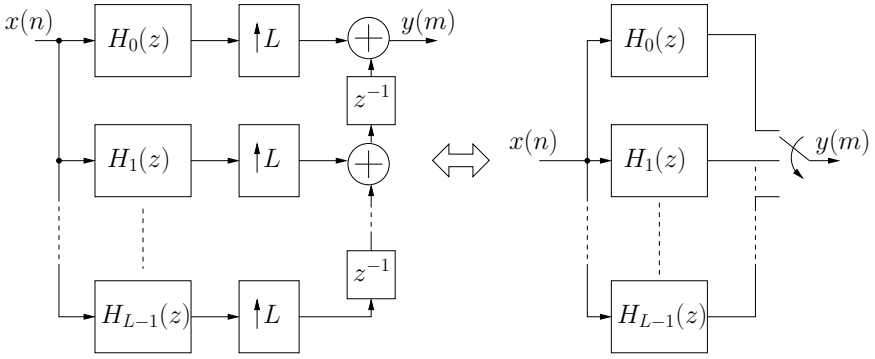
Figure 1.10: Polyphase implementation of a decimator.

An efficient implementation of decimators and interpolators results if the filter transfer function is represented in polyphase decomposed form [10]. The filter $H(z)$ in Fig. 1.4b is represented by its polyphase representation form as shown in Fig. 1.10a. A more efficient polyphase implementation and its equivalent commutative structure is shown in Fig. 1.10b. Similarly, the interpolation filter $H(z)$ in Fig. 1.6b is represented by its polyphase representation form as shown in Fig. 1.11a. Its equivalent structure, but more efficient in hardware implementation, is shown in Fig. 1.11b.

For FIR filters, the polyphase decomposition into low-order sub-filters is very easy. However, for IIR filters, the polyphase decomposition is not so simple, but it is possible to do so [10]. An IIR filter has a transfer function that is a ratio of two polynomials. The representation of the transfer function into the form, (1.10), needs some modifications in the original transfer function in such a way that the denominator is only function of powers of z^M or z^L, where M and L



(a) Polyphase decomposition of an interpolation filter.



(b) Moving upsampler to after sub-filters and replacing output structure by a commutator.

Figure 1.11: Polyphase implementation of an interpolator.

are the polyphase decomposition factors. Several approaches are available in the literature for the polyphase decomposition of the IIR filters. In the first approach [11], the original IIR transfer function is re-arranged and transformed into (1.10). The polyphase sub-filters in the second approach has distinct all-pass sub-filters [12–15].

In this thesis, only FIR filters and their polyphase implementation forms are considered for the sampling rate conversion.

1.4 Fractional-Delay Filters

Fractional-delay (FD) filters find applications in, for example, mitigation of symbol synchronization errors in digital communications [16–19], time-delay estimation [20–22], echo cancellation [23], and arbitrary sampling rate conver-

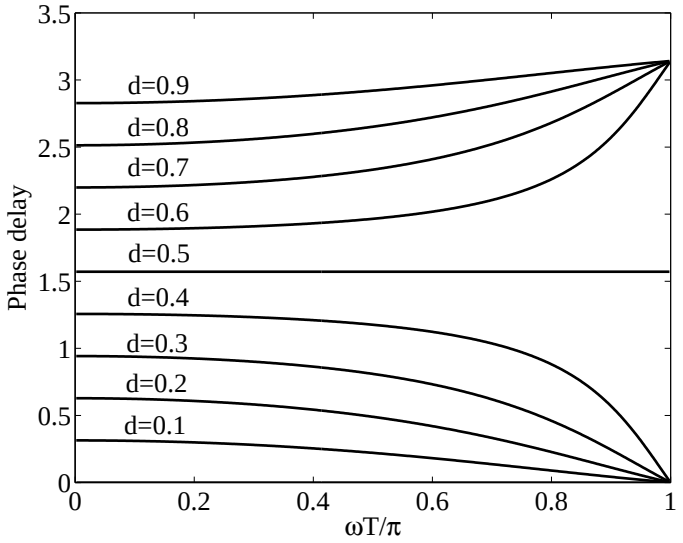


Figure 1.12: Phase-delay characteristics of FD filters designed for delay parameter $d = \{0.1, 0.2, \ldots, 0.9\}$.

sion [24–26]. FD filters are used to provide a fractional delay adjustable to any desired value. Ideally, the output $y(n)$ of an FD filter for an input $x(n)$ is given by

$$y(n) = x(n - D), \tag{1.11}$$

where D is a delay. Equation (1.11) is valid for integer values of D only. For non-integer values of D, (1.11) need to be approximated. The delay parameter D can be expressed as

$$D = D_{\text{int}} + d, \tag{1.12}$$

where D_{int} is the integer part of D and d is the FD. The integer part of the delay can then be implemented as a chain of D_{int} unit delays. The FD d however needs approximation. In the frequency domain, an ideal FD filter can be expressed as

$$H_{\text{des}}(e^{j\omega}) = e^{-j(D_{\text{int}}+d)\omega T}. \tag{1.13}$$

The ideal FD filter in (1.13) can be considered as all-pass and having a linear-phase characteristics. The magnitude and phase responses are

$$|H_{\text{des}}(e^{j\omega T})| = 1 \tag{1.14}$$

and

$$\phi(\omega T) = -(D_{\text{int}} + d)\omega T. \tag{1.15}$$

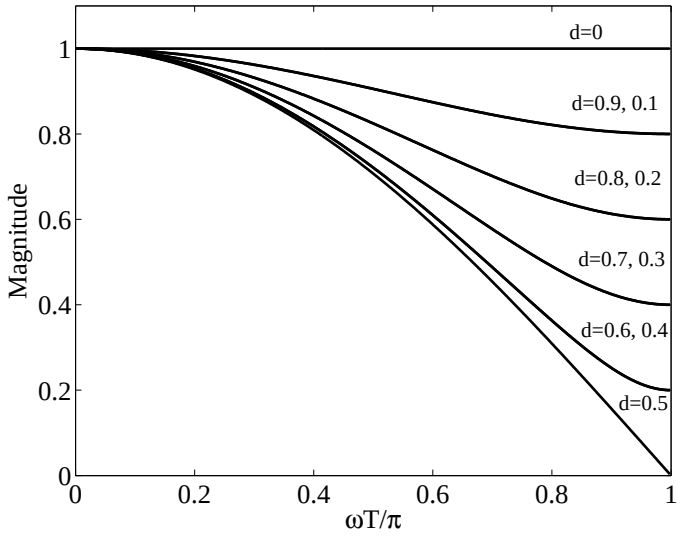


Figure 1.13: Magnitude of FD filters designed for delay parameter $d = \{0.1, 0.2, \ldots, 0.9\}$.

For the approximation of the ideal filter response in (1.11), a wide range of FD filters have been proposed based on FIR and IIR filters [27–34]. The phase delay and magnitude characteristics of the FD filter based on first order Lagrange interpolation [32, 35] are shown in Figs. 1.12 and 1.13. The underlying continuous-time signal $x_a(nT)$, delayed by a fractional-delay d, can be expressed as

$$y(nT) = x_a(nT - D_{\text{int}}T - dT), \tag{1.16}$$

and it is demonstrated for $d = 0.2$ and $d = 0.4$ in Fig. 1.14 .

In the application of FD filters for SRC, the fractional-delay d is changed at every instant an output sample occurs. The input and output rates will now be different. If the sampling rate is required to be increased by a factor of two, for each input sample there are now two output samples. The fractional-delay d assumes the value 0 and 0.5 for each input sample, and the two corresponding output samples are computed. For a sampling rate increase by a factor of L, d will take on all values in a sequential manner between $\{(L-1)/L, 0\}$, with a step size of $1/L$. For each input sample, L output samples are generated. Equivalently, it can be interpreted as delaying the underlying continuous-time signal by L different values of fractional-delays.

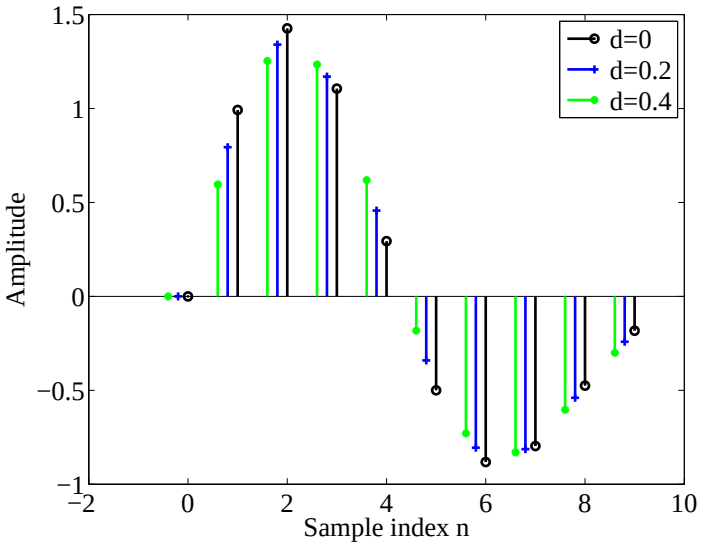


Figure 1.14: The underlying continuous-time signal delayed by $d = 0.2$ and $d = 0.4$ with FD filtering.

1.5 Power and Energy Consumption

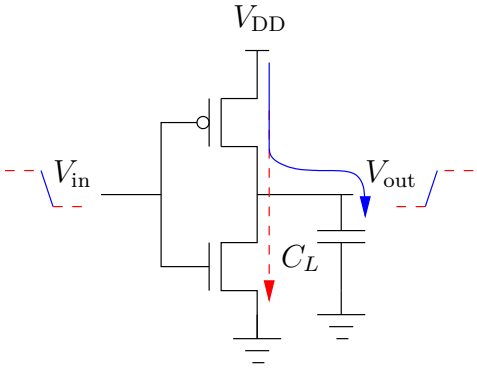
Power is dissipated in the form of heat in digital CMOS circuits. The power dissipation is commonly divided into three different sources: [36, 37]

- ⋆ Dynamic or switching power consumption
- ⋆ Short circuit power consumption
- ⋆ Leakage power consumption

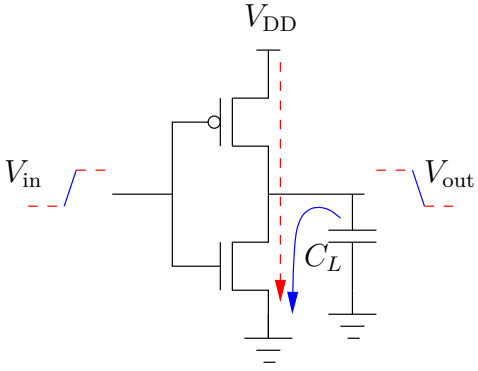
The above sources are summarized in an equation as

$$\begin{aligned} P_{\text{avg}} &= P_{\text{dynamic}} + P_{\text{short-circuit}} + P_{\text{leakage}} \\ &= \alpha_{0\to1} C_L V_{\text{DD}}^2 f_{\text{clk}} + I_{\text{sc}} V_{\text{DD}} + I_{\text{leakage}} V_{\text{DD}}. \end{aligned} \tag{1.17}$$

The switching or dynamic power consumption is related to charging and discharging of a load capacitance C_L through the PMOS and NMOS transistors during low-to-high and high-to low transitions at the output, respectively. The total energy drawn from the power supply for low-to-high transition, seen in Fig. 1.15a, is $C_L V_{\text{DD}}^2$, half of which is dissipated in the form of heat through the PMOS transistors while the other half is stored in the load capacitor. During the pull-down, high to low transition, seen in Fig. 1.15b, the energy stored on C_L which is $C_L V_{\text{DD}}^2/2$ is dissipated as heat by the NMOS transistors. If all these transitions occur at the clock rate of f_{clk} then the switching power consumption



(a) A rising output transition on the CMOS inverter.



(b) A falling output transition on the inverter.

Figure 1.15: A rising and falling output transition on the CMOS inverter. The solid arrows represent the charging and discharging of the load capacitance C_L. The dashed arrow is for the leakage current.

is given by $C_L V_{DD}^2 f_{clk}$. However the switching of the data is not always at the clock rate but rather at some reduced rate which is best defined by another parameter $\alpha_{0\to1}$, defined as the average number of times in each cycle that a node makes a transition from low to high. All the parameters in the dynamic power equation, except $\alpha_{0\to1}$, are defined by the layout and specification of the circuit.

The second power term is due to the direct-path short circuit current, I_{sc}, which flows when both of the PMOS and NMOS transistors are active simultaneously, resulting in a direct path from supply to ground.

Leakage power, on the other hand, is dissipated in the circuits when they are idle, as shown in Figs. 1.15a and 1.15b by a dashed line. The leakage current, $I_{leakage}$, consists of two major contributions: I_{sub} and I_{gate}. The term I_{sub} is the sub-threshold current caused by low threshold voltage, when both NMOS

and PMOS transistors are off. The other quantity, I_{gate}, is the gate current caused by reduced thickness of the gate oxide in deep sub-micron process . The contribution of the reverse leakage currents, due to of reverse bias between diffusion regions and wells, is small compared to sub-threshold and gate leakage currents.

In modern/concurrent CMOS technologies, the two foremost forms of power consumptions are dynamic and leakage. The relative contribution of these two forms of power consumptions has greatly evolved over the period of time. Today, when technology scaling motivates the reduced power supply and threshold voltage, the leakage component of power consumption has started to become dominant [38–40]. In today's processes, sub-threshold leakage is the main contributor to the leakage current.

Chapter 2

Finite Word Length Effects

Digital filters are implemented in hardware with finite-precision numbers and arithmetic. As a result, the digital filter coefficients and internal signals are represented in discrete form. This generally leads to two different types of finite word length effects.

First, there are the errors in the representing of coefficients. The coefficients representation in finite precision (quantization) has the effect of a slight change in the location of the filter poles and zeros. As a result, the filter frequency response differs from the response with infinite-precision coefficients. However, this error type is deterministic and is called coefficient quantization error.

Second, there are the errors due to multiplication round-off, that results from the rounding or truncation of multiplication products within the filter. The error at the filter output that results from these roundings or truncations is called round-off noise.

This chapter outlines the finite word length effects in digital filters. It first discusses binary number representation forms. Different types of fixed-point quantizations are then introduced along with their characteristics. The overflow characteristics in digital filters are briefly reviewed with respect to addition and multiplication operations. Scaling operation is then discussed which is used to prevent overflows in digital filter structures. The computation of round-off noise at the digital filter output is then outlined. The description of the constant coefficient multiplication is then given. Finally, different approaches for the optimization of word length are reviewed.

2.1 Numbers Representation

In digital circuits, a number representation with a radix of two, i.e., binary representation, is most commonly used. Therefore, a number is represented by

a sequence of binary digits, bits, which are either 0 or 1. A w-bit unsigned binary number can be represented as

$$X = x_0 x_1 x_2 ... x_{w-2} x_{w-1}, \tag{2.1}$$

with a value of

$$X = \sum_{i=0}^{w-1} x_i 2^{w-i-1}, \tag{2.2}$$

where x_0 is the most significant bit (MSB) and x_{w-1} is the least significant bit (LSB) of the binary number.

A fixed-point number consists of an integral part and a fractional part, with the two parts separated by a binary point in radix of two. The position of the binary point is almost always implied and thus the point is not explicitly shown. If a fixed-point number has w_I integer bits and w_F fractional bits, it can be expressed as

$$X = x_{w_I-1} \dots x_1 x_0 . x_{-1} x_{-2} \dots x_{-w_F}. \tag{2.3}$$

The value can be obtained as

$$X = \sum_{i=0}^{w_I-1} x_i 2^i + \sum_{i=-w_F}^{-1} x_i 2^i. \tag{2.4}$$

2.1.1 Two's Complement Numbers

For a suitable representation of numbers and an efficient implementation of arithmetic operation, fixed-point arithmetics with a word length of w bits is considered. Because of its special properties, the two's complement representation is considered, which is the most common type of arithmetic used in digital signal processing. The numbers are usually normalized to $[-1, 1)$, however, to accommodate the integer bits, the range $[-2^{w_I}, -2^{w_I})$, where $w_I \in \mathbb{N}$, is assumed. The quantity w_I denotes the number of integer bits. The MSB, the left-most bit in w, is used as the sign bit. The sign bit is treated in the same manner as the other bits. The fraction part is represented with $w_F = w - 1 - w_I$ bits. The quantization step is as a result $\Delta = 2^{-w_F}$.

If X_{2C} is a w-bit number in two's complement form, then by using all definitions considered above, X can be represented as

$$\begin{aligned} X_{2C} &= 2^{w_I} \left(-x_0 2^0 + \sum_{i=1}^{w-1} x_i 2^{-i} \right), x_i \in \{0, 1\}, i = 0, 1, 2, \dots, w-1, \\ &= \underbrace{-x_0 2^{w_I}}_{\text{sign bit}} + \underbrace{\sum_{i=1}^{w_I} x_i 2^{w_I - i}}_{\text{integer}} + \underbrace{\sum_{i=w_I+1}^{w-1} x_i 2^{w_I - i}}_{\text{fraction}}, \end{aligned}$$

or in compact form as

$$X_{2C} = [\underbrace{x_0}_{\text{sign bit}} | \underbrace{x_1 x_2 \dots x_{w_I}}_{\text{integer}} | \underbrace{x_{w_I+1} \dots x_{w-1}}_{\text{fraction}}]_2. \tag{2.5}$$

In two's complement, the range of representable numbers is asymmetric. The largest number is

$$X_{\max} = 2^{w_I} - 2^{-w_F} = [0|1 \dots 1|1 \dots 1|]_2, \tag{2.6}$$

and the smallest number is

$$X_{\min} = -2^{w_I} = [1|0 \dots 0|0 \dots 0|]_2. \tag{2.7}$$

2.1.2 Canonic Signed-Digit Representation

Signed-digit (SD) numbers differ from the binary representation, since the digits are allowed to take negative values, i.e., $x_i \in \{-1, 0, 1\}$. The symbol $\overline{1}$ is also used to represent -1. It is a redundant number system, as different SD representations are possible of the same integer value. The canonic signed-digit (CSD) representation is a special case of signed-digit representation in that each number has a unique representation. The other feature of CSD representation is that a CSD binary number has the fewest number of non-zero digits with no consecutive bits being non-zero [41].

A number can be represented in CSD form as

$$X = \sum_{i=0}^{w-1} x_i 2^i,$$

where, $x_i \in \{-1, 0, +1\}$ and $x_i x_{i+1} = 0$, $i = 0, 1, \dots, w-2$.

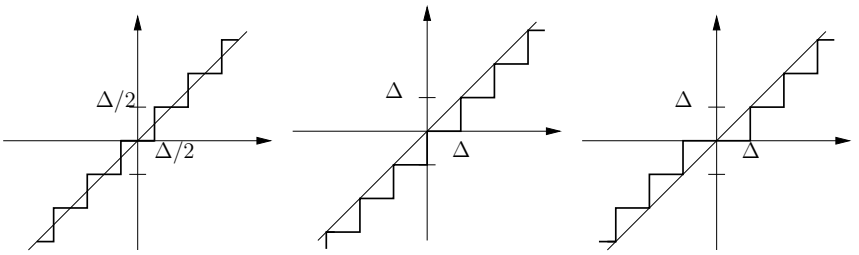
2.2 Fixed-Point Quantization

Three types of fixed-point quantization are normally considered, rounding, truncation, and magnitude truncation [1, 42, 43]. The quantization operator is denoted by $Q(.)$. For a number X, the rounded value is denoted by $Q_r(X)$, the truncated value by $Q_t(X)$, and the magnitude truncated value $Q_{mt}(X)$. If the quantized value has w_F fractional bits, the quantization step size, i.e., the difference between the adjacent quantized levels, is

$$\Delta = 2^{-w_F} \tag{2.8}$$

The rounding operation selects the quantized level that is nearest to the unquantized value. As a result, the rounding error is at most $\Delta/2$ in magnitude as shown in Fig. 2.1a. If the rounding error, ϵ_r, is defined as

$$\epsilon_r = Q_r(X) - X, \tag{2.9}$$



(a) Rounding error. (b) Truncation error. (c) Magnitude truncation error.

Figure 2.1: Quantization error characteristics.

then

$$-\frac{\Delta}{2} \leq \epsilon_r \leq \frac{\Delta}{2}. \tag{2.10}$$

Truncation simply discards the LSB bits, giving a quantized value that is always less than or equal to the exact value. The error characteristics in the case of truncation are shown in Fig. 2.1b. The truncation error is

$$-\Delta < \epsilon_t \leq 0. \tag{2.11}$$

Magnitude truncation chooses the nearest quantized value that has a magnitude less than or equal to the exact value, as shown in Fig. 2.1c, which implies

$$-\Delta < \epsilon_{mt} < \Delta. \tag{2.12}$$

The quantization error can often be modeled as a random variable that has a uniform distribution over the appropriate error range. Therefore, the filter calculations involving round-off errors can be assumed error-free calculations that have been corrupted by additive white noise [43]. The mean and variance of the rounding error is

$$m_r = \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} \epsilon_r d\epsilon_r = 0 \tag{2.13}$$

and

$$\sigma_r^2 = \frac{1}{\Delta} \int_{-\Delta/2}^{\Delta/2} (\epsilon_r - m_r)^2 d\epsilon_r = \frac{\Delta^2}{12}. \tag{2.14}$$

Similarly, for truncation, the mean and variance of the error are

$$m_t = -\frac{\Delta}{2} \quad \text{and } \sigma_t^2 = \frac{\Delta^2}{12}, \tag{2.15}$$

and for magnitude truncation,

$$m_{mt} = 0 \quad \text{and } \sigma_{mt}^2 = \frac{\Delta^2}{3}. \tag{2.16}$$

2.3 Overflow Characteristics

With finite word length, it is possible for the arithmetic operations to overflow. This happens for fixed-point arithmetic ,e.g., when two numbers of the same sign are added to give a value having a magnitude not in the interval $[-2^{w_I}, 2^{w_I})$. Since numbers outside this range are not representable, the result overflows. The overflow characteristics of two's complement arithmetic can be expressed as

$$X_{2C}(X) = \begin{cases} X - 2^{w_I+1}, & X \geq 2^{w_I}, \\ X, & -2^{w_I} \leq X < 2^{w_I}, \\ X + 2^{w_I+1}, & X < -2^{w_I}, \end{cases} \tag{2.17}$$

and graphically it is shown in Fig. 2.2.

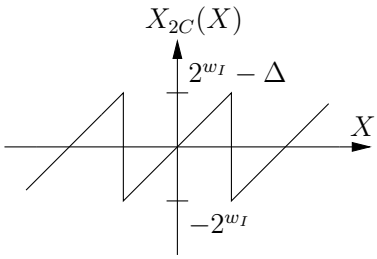


Figure 2.2: Overflow characteristics for two's complement arithmetic.

2.3.1 Two's Complement Addition

In two's complement arithmetic, when two numbers each having w-bits are added together, the result will be $w + 1$ bits. To accommodate this extra bit, the integer bits need to be extended. In two's complement, such overflows can be seen as discarding the extra bit, which corresponds to a repeated addition or subtraction of $2^{(w_I+1)}$ to make the $w + 1$-bit result to be representable by w-bits. This model for overflow is illustrated in Fig. 2.3.

2.3.2 Two's Complement Multiplication

In the case of multiplication of two fixed-point numbers each having w-bits, the result is $2w$-bits. Overflow is similarly treated here as in the case of addition, a repeated addition or subtraction of $2^{(w_I+1)}$. Having two numbers, each with a precision w_F, the product is of precision $2w_F$. The number is reduced to w_F

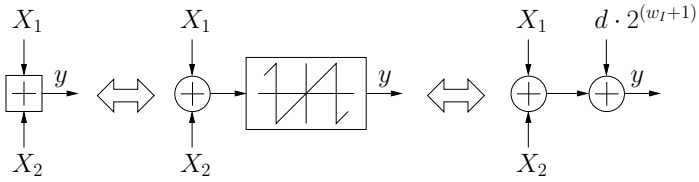


Figure 2.3: Addition in two's complement. The integer $d \in \mathbb{Z}$ has to assign a value such that $y \in [-2^{w_I}, 2^{w_I})$.

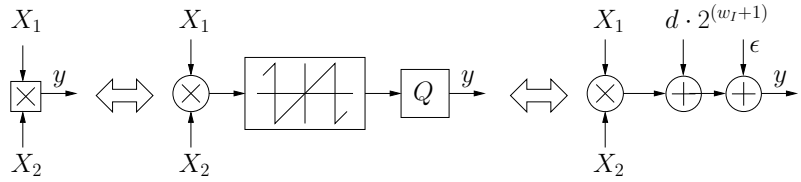


Figure 2.4: Multiplication in two's complement. The integer $d \in \mathbb{Z}$ has to assign a value such that $y \in [-2^{w_I}, 2^{w_I})$.

precision again by using rounding or truncation. The model to handle overflow in multiplication is shown in Fig. 2.4.

2.4 Scaling

To prevent overflow in fixed-point filter realizations, the signal levels inside the filter can be reduced by inserting scaling multipliers. However, the scaling multipliers should not distort the transfer function of the filter. Also the signal levels should not be too low, otherwise, the signal-to-noise (SNR) ratio will suffer as the noise level is fixed for fixed-point arithmetic.

The use of two's complement arithmetic eases the scaling, as repeated additions with an overflow can be acceptable if the final sum lies within the proper signal range [4]. However, the inputs to non-integer multipliers must not overflow. In the literature, there exist several scaling norms that compromise between the probability of overflows and the round-off noise level at the output. In this thesis, only the commonly employed L_2-norm is considered which for a Fourier transform $H(e^{j\omega T})$ is defined as

$$\left\| H(e^{j\omega T}) \right\|_2 = \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega T}) \right|^2 d(\omega T)}.$$

In particular, if the input to a filter is Gaussian white noise with a certain probability of overflow, using L_2-norm scaling of a node inside the filter, or at the output, implies the same probability of overflow at that node.

2.5 Round-Off Noise

A few assumptions need to be made before computing the round-off noise at the digital filter output. Quantization noise is assumed to be stationary, white, and uncorrelated with the filter input, output, and internal variables. This assumption is valid if the filter input changes from sample to sample in a sufficiently random-like manner [43].

For a linear system with impulse response $g(n)$, excited by white noise with mean m_x and variance σ_x^2, the mean and variance of the output noise is

$$m_y = m_x \sum_{n=-\infty}^{\infty} g(n) \tag{2.18}$$

and

$$\sigma_y^2 = \sigma_x^2 \sum_{n=-\infty}^{\infty} g^2(n), \tag{2.19}$$

where $g(n)$ is the impulse response from the point where a round-off takes place to the filter output. In case there is more than one source of roundoff error in the filter, the assumption is made that these errors are uncorrelated. The round-off noise variance at the output is the sum of contributions from each quantization error source.

2.6 Word Length Optimization

As stated earlier in the chapter, the quantization process introduces round-off errors, which in turn measures the accuracy of an implementation. The cost of an implementation is generally required to be minimized, while still satisfying the system specification in terms of implementation accuracy. Excessive bit-width allocation will result in wasting valuable hardware resources, while in-sufficient bit-width allocation will result in overflows and violate precision requirements. The word length optimization approach trades precisions for VLSI measures such as area, power, and speed. These are the measures or costs by which the performance of a design is evaluated. After word length optimization, the hardware implementation of an algorithm will be efficient typically involving a variety of finite precision representation of different sizes for the internal variables.

The first difficulty in the word length optimization problem is defining of the relationship of word length to considered VLSI measures. The possible ways could be closed-form expressions or the availability of precomputed values in the form of a table of these measures as a function of word lengths. These closed-form expressions and precomputed values are then used by the word length optimization algorithm at the word length assignment phase to have an estimate, before doing an actual VLSI implementation.

The round-off noise at the output is considered as the measure of performance function because it is the primary concern of many algorithm designers. The round-off error is a decreasing function of word length, while VLSI measures such as area, speed, and power consumption are increasing functions of word length. To derive a round-off noise model, an LTI system with n quantization error sources is assumed. This assumption allows to use superposition of independent noise sources to compute the overall round-off noise at the output. The noise variance at the output is then written as

$$\sigma_o^2 = \sum_{k=0}^{\infty} \sigma_{e_i}^2 h_i^2(k), \quad 1 \leq i \leq n, \tag{2.20}$$

where e_i is the quantization error source at node i and $h_i(k)$ is the impulse response from node i to the output. If the quantization word length w_i is assumed for error source at node i then

$$\sigma_{e_i}^2 = \frac{2^{-2w_i}}{12}, \quad 1 \leq i \leq n. \tag{2.21}$$

The formulation of an optimization problem is done by constraining the cost or accuracy of an implementation while optimizing other metric(s). For simplicity, the cost function is assumed to be the area of the design. Its value is measured appropriately to the considered technology, and it is assumed to be the function of quantization word lengths, $f(w_i)$, $i = 1, 2, \ldots, n$. The performance function, on other hand, is taken to be the round-off noise value at the output, given in (2.20), due to the limiting of internal word lengths. As a result, one possible formulation of word length optimization problem is

$$\text{minimize } area : f(w_i), \quad 1 \leq i \leq n \tag{2.22}$$

$$\text{s.t. } \sigma_o^2 \leq \sigma_{\text{spec}}, \tag{2.23}$$

where σ_{spec} is the required noise specification at the output.

The problem of word length optimization has received considerable research attention. In [44–48], different search-based strategies are used to find suitable word length combinations. In [49], the word length allocation problem is solved using a mixed-integer linear programming formulation. Some other approaches, e.g., [50–52], have constrained the cost, while optimizing the other metric(s).

2.7 Constant Multiplication

A multiplication with a constant coefficient, commonly used in DSP algorithms such as digital filters [53], can be made multiplierless by using additions, subtractions, and shifts only [54]. The complexity for adders and subtracters is roughly the same so no differentiation between the two is normally considered. A shift operation in this context is used to implement a multiplication by a

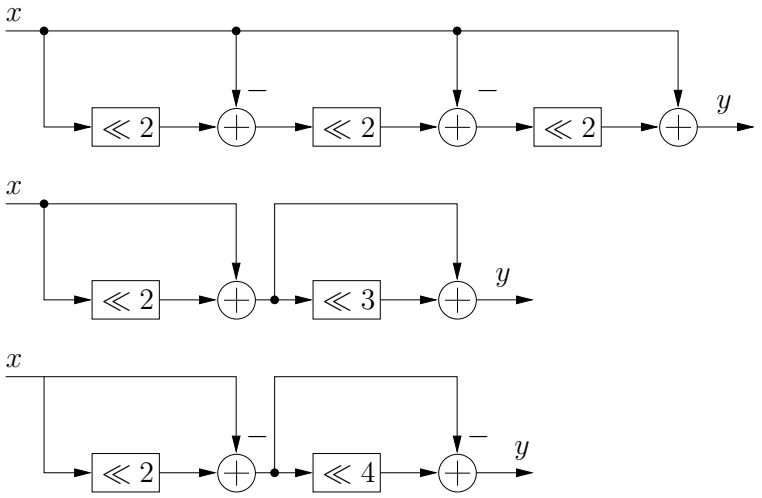


Figure 2.5: Different realizations of multiplication with the coefficient 45. The symbol $\ll i$ are used to represent i left shifts.

factor of two. Most of the work in the literature has focused on minimizing the adder cost [55–57]. For bit parallel arithmetic, the shifts can be realized without any hardware using hardwiring.

In constant coefficient multiplication, the hardware requirements depend on the coefficient value, e.g., the number of ones in the binary representation of the coefficient value. The constant coefficient multiplication can be implemented by the method that is based on the CSD representation of the constant coefficient [58], or more efficiently by using other structures as well that require fewer number of operations [59]. Consider, for example, the coefficient 45, having the CSD representation $10\overline{1}0\overline{1}01$. The multiplication with this constant can be realized by three different structures as shown in Fig. 2.5, varying with respect to number of additions and shifts requirement [41].

In some applications, one signal is required to be multiplied by several constant coefficients, as in the case of transposed direct-form FIR filters shown in 1.2. Realizing the set of products of a single multiplicand is known as the multiplier block problem [60] or the multiple constant multiplications (MCM) problem [61]. A simple way to implement multiplier blocks is to realize each multiplier separately. However, they can be implemented more efficiently by using structures that remove any redundant partial results among the coefficients and thereby reduce the overall number of operations. The MCM algorithms can be divided into three groups based on the approach used in the algorithms; sub-expression sharing [61–65], difference methods [66–70], and graph based methods [60, 71–73].

The MCM concepts can be further generalized to computations involving multiple inputs and multiple outputs. This corresponds to a matrix-vector

multiplication with a matrix with constant coefficients. This is the case for linear transforms such as the discrete cosine transform (DCT) or the discrete Fourier transform (DFT), but also FIR filter banks [74], polyphase decomposed FIR filters [75], and state space digital filters [4, 41]. Matrix-vector MCM algorithms include [76–80].

Chapter 3

Integer Sampling Rate Conversion

The role of filters in sampling rate conversion has been described in Chapter 1. This chapter mainly focuses on the implementation aspects of decimators and interpolators due to the fact that filtering initially has to be performed on the side of higher sampling rate. The goal is then to achieve conversion structures allowing the arithmetic operations to be performed at the lower sampling rate. The overall computational workload will as a result be reduced.

The chapter begins with the basic decimation and interpolation structures that are based on FIR filters. A part of the chapter is then devoted to the efficient polyphase implementation of FIR decimators and interpolators. The concept of the basic cascade integrator-comb (CIC) filter is introduced and its properties are discussed. The structures of the CIC-based decimators and interpolators are then shown. The overall two-stage implementation of a CIC and an FIR filter that is used for the compensation is described. Finally, the non-recursive implementation of the CIC filter and its multistage and polyphase implementation structures are presented.

3.1 Basic Implementation

Filters in SRC have an important role and are used as anti-aliasing filters in decimators or anti-imaging filters in interpolators. The characteristics of these filters then correlate to the overall performance of a decimator or of an interpolator. As discussed in Chapter 1, filtering operations need to be performed at the higher sampling rate. In decimation, filtering operation precedes downsampling and in interpolation, filtering operation proceeds upsampling. However, downsampling or upsampling operations can be moved into the filtering structures, providing arithmetic operations to be performed at the lower sampling rate. As a result, the overall computational workload in the sampling rate conversion system can potentially be reduced by the conversion factor M (L).

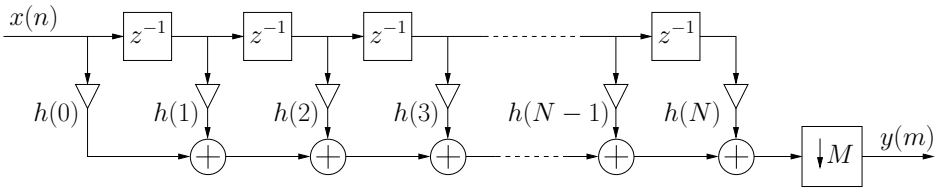


Figure 3.1: Cascade of direct-form FIR filter and downsampler.

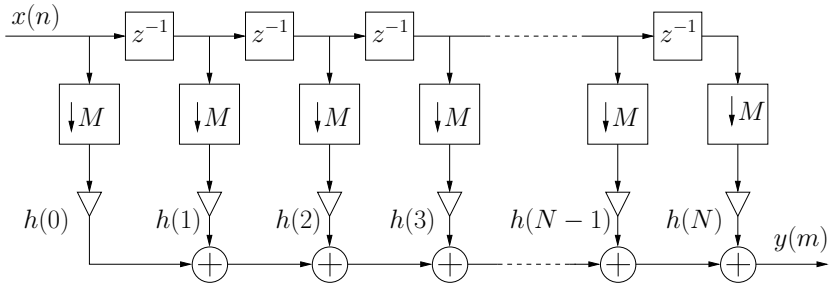


Figure 3.2: Computational efficient direct-form FIR decimation structure.

In the basic approach [81], FIR filters are considered as anti-aliasing or anti-imaging filters. The non-recursive nature of FIR filters provides the opportunity to improve the efficiency of FIR decimators and interpolators through polyphase decomposition.

3.1.1 FIR Decimators

In the basic implementation of an FIR decimator, a direct-form FIR filter is cascaded with the downsampling operation as shown in Fig. 3.1. The FIR filter acts as the anti-aliasing filter. The implementation can be modified to a form that is computationally more efficient, as seen in Fig. 3.2. The downsampling operation precedes the multiplications and additions which results in the arithmetic operations to be performed at the lower sampling rate.

In Fig. 3.2, the input data is now read simultaneously from the delays at every M:th instant of time. The input sample values are then multiplied by the filter coefficients $h(n)$ and combined together to give $y(m)$.

In the basic implementation of the FIR decimator in Fig. 3.1, the number of the multiplications per input sample in the decimator is equal to the FIR filter length $N + 1$. The first modification made by the use of noble identity, seen in Fig. 3.2, reduces the multiplications per input sample to $(N + 1)/M$. The symmetry of coefficients may be exploited in the case of linear phase FIR filter to reduce the number of multiplications to $(N + 1)/2$ for odd N and $N/2 + 1$ for even N. In the second modification, for a downsampling factor M, multiplications per input sample is $(N + 2)/2M$ for even N. Similarly, the

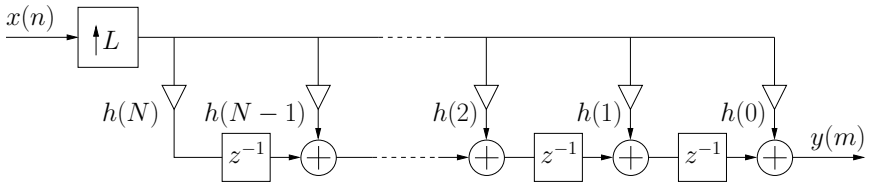


Figure 3.3: Cascade of upsampler and transposed direct-form FIR filter.

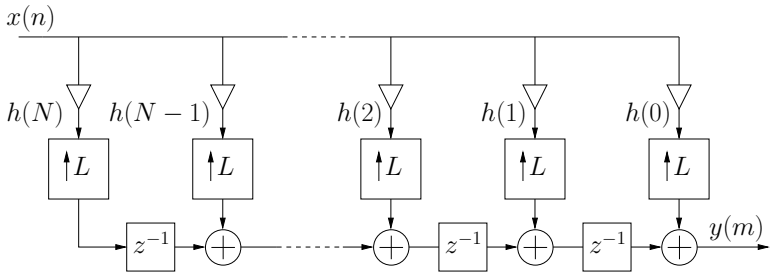


Figure 3.4: Computational efficient transposed direct-form FIR interpolation structure.

multiplications per input sample are $(N+1)/(2M)$ for odd N.

3.1.2 FIR Interpolators

The interpolator is a cascade of an upsampler followed by an FIR filter that acts as an anti-imaging filter, seen in Fig. 3.3. Similar to the case of decimator, the upsampling operation is moved into the filter structure at the desired position such that the multiplication operations are performed at the lower sampling-rate side, as shown in Fig. 3.4.

In the basic multiplication of the FIR interpolator in Fig. 3.3, every L:th input sample to the filter is non-zero. Since there is no need to multiply the filter coefficients by the zero-valued samples, multiplications need to be performed at the sampling rate of the input signal. The result is then upsampled by the desired factor L. This modified implementation approach reduces the multiplications count per output sample from $N+1$ to $(N+1)/L$. Similar to the decimator case, the coefficient symmetry of the linear phase FIR filter can be exploited to reduce the number of multiplication further by two.

3.2 Polyphase FIR Filters

As described earlier in Sec. 3.1, the transfer function of an FIR filter can be decomposed into parallel structures based on the principle of polyphase decomposition. For a factor of M polyphase decomposition, the FIR filter transfer

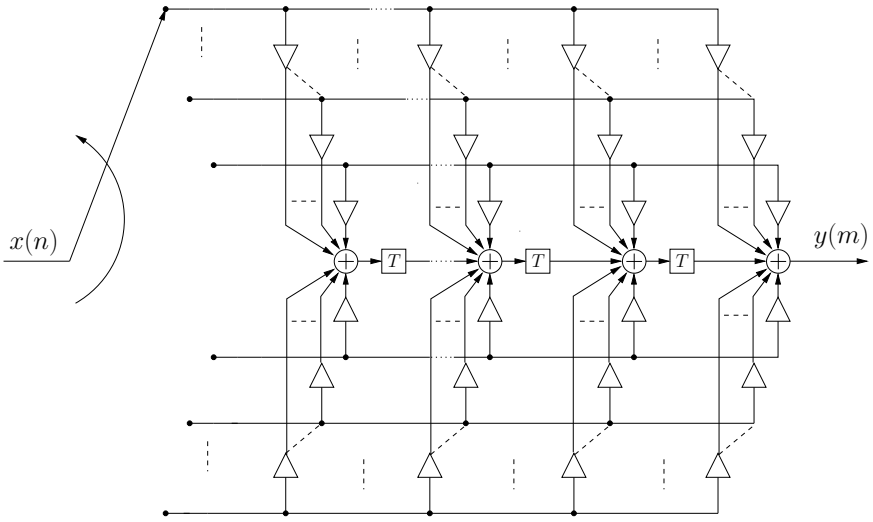


Figure 3.5: Polyphase factor of M FIR decimator structure.

function is decomposed into M low-order transfer functions called the polyphase components [5, 10]. The individual contributions from these polyphase components when added together has the same effect as of the original transfer function.

If the impulse response $h(n)$ is assumed zero outside $0 \leq n \leq N$, then the factors $H_m(z)$ in (1.10) becomes

$$H_m(z) = \sum_{n=0}^{\lfloor (N+1)/M \rfloor} h(nM+m)z^{-n}, \quad 0 \leq m \leq M-1. \tag{3.1}$$

3.2.1 Polyphase FIR Decimators

The basic structure of the polyphase decomposed FIR filter is the same as that in Fig. 1.10a, however the sub-filters $H_m(z)$ are now specifically defined in (3.1). The arithmetic operations in the implementation of all FIR sub-filters still operate at the input sampling rate. The downsampling operation at the output can be moved into the polyphase branches by using the more efficient polyphase decimation implementation structure in Fig. 1.10. As a result, the arithmetic operations inside the sub-filters now operate at the lower sampling rate. The overall computational complexity of the decimator is reduced by a factor of M.

The input sequences to the polyphase sub-filters are combination of delayed and downsampled values of the input which can be directly selected from the input with the use of a commutator. The resulting polyphase architecture for a factor-of-M decimation is shown in Fig. 3.5. The commutator operates at the

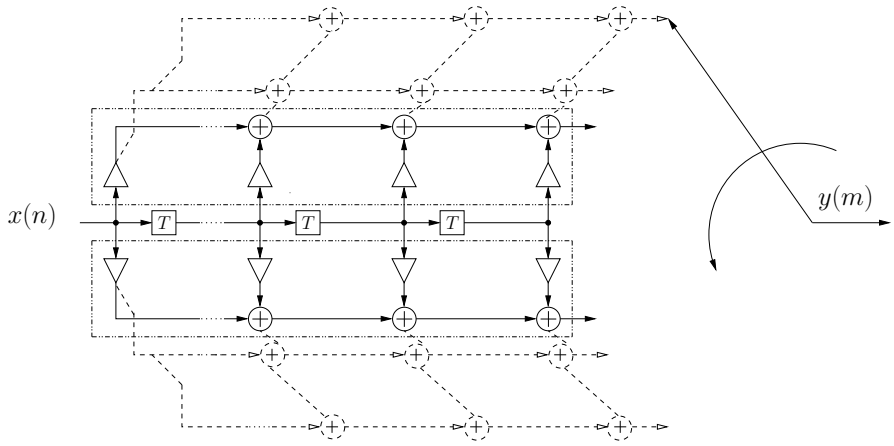


Figure 3.6: Polyphase factor of L FIR interpolator structure.

input sampling rate but the sub-filters operate at the output sampling rate.

3.2.2 Polyphase FIR Interpolators

The polyphase implementation of an FIR filter, by a factor L, is done by decomposing the original transfer function into L polyphase components, also called polyphase sub-filters. The basic structure of the polyphase decomposed FIR filter is same as that in the Fig. 1.11a. The polyphase sub-filters $H_m(z)$ are defined in (3.1) for the FIR filter case. In the basic structure, the arithmetic operations in the implementation of all FIR sub-filters operate at the upsampled rate. The upsampling operation at the input is moved into the polyphase branches by using the more efficient polyphase interpolation implementation structure in Fig. 1.11. More efficient interpolation structure will result because the arithmetic operations inside the sub-filters now operate at the lower sampling rate. The overall computational complexity of the interpolator is reduced by a factor of L.

On the output side, the combination of upsamplers, delays, and adders are used to feed the correct interpolated output sample. The same function can be replaced directly by using a commutative switch. The resulting architecture for factor of L interpolator is shown in Fig. 3.6.

3.3 Multistage Implementation

A multistage decimator with K stages, seen in Fig. 3.7, can be used when the decimation factor M can be factored into the product of integers, $M = M_1 \times M_2 \times, \ldots, M_k$, instead of using a single filter and factor of M downsampler. Similarly, a multistage interpolator with K stages, seen in Fig. 3.8, can be used if

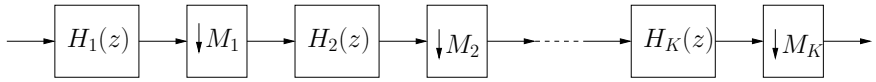


Figure 3.7: Multistage implementation of a decimator.

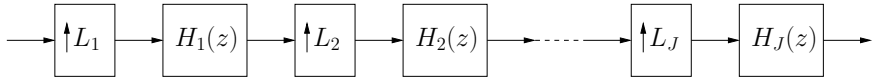


Figure 3.8: Multistage implementation of an interpolator.

the interpolation factor L can be factored as $L = L_1 \times L_2 \times \cdots \times L_K$. The noble identities can be used to transform the multistage decimator implementation into the equivalent single stage structure as shown in Fig. 3.9. Similarly, the single stage equivalent of the multistage interpolator is shown in Fig. 3.10. An optimum realization depends on the proper selection of K and J and best ordering of the multistage factors [82].

The multistage implementations are used when there is need of implementing large sampling rate conversion factors [1, 82]. Compared to the single stage implementation, it relaxes the specifications of individual filters. A single stage implementation with a large decimation (interpolation) factor requires a very narrow passband filter, which is hard from the complexity point of view [3, 81, 82].

3.4 Cascaded Integrator Comb Filters

An efficient architecture for a high decimation-rate filter is the cascade integrator comb (CIC) filter introduced by Hogenauer [83]. The CIC filter has proven to be an effective element in high-decimation or interpolation systems [84–87]. The simplicity of implementation makes the CIC filters suitable for operating at higher frequencies. A single stage CIC filter is shown in Fig. 3.11. It is a cascade of integrator and comb sections. The feed forward section of the CIC filter, with differential delay R, is comb section and the feedback section is called an integrator. The transfer function of a single stage CIC filter is given as

$$H(z) = \frac{1 - z^{-R}}{1 - z^{-1}} = \sum_{n=0}^{R-1} z^{-n}. \tag{3.2}$$

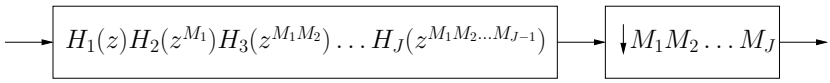


Figure 3.9: Single stage equivalent of a multistage decimator.

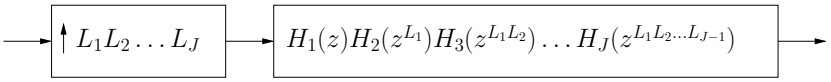


Figure 3.10: Single stage equivalent of a multistage interpolator.

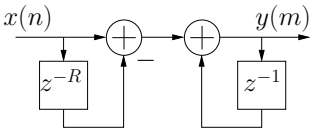


Figure 3.11: Single-stage CIC filter.

The comb filter has R zeros that are equally spaced around the unit circle. The zeros are the R-th roots of unity and are located at $z(k) = e^{j2\pi k/R}$, where $k = 0, 1, 2, \ldots, R-1$. The integrator section, on the other hand, has a single pole at $z = 1$. CIC filters are based on the fact that perfect pole/zero canceling can be achieved, which is only possible with exact integer arithmetic [88]. The existence of integrator stages will lead to overflows. However, this is of no harm if two's complement arithmetic is used and the range of the number system is equal to or exceeds the maximum magnitude expected at the output of the composite CIC filter. The use of two's complement and non-saturating arithmetic accommodates the issues with overflow. With the two's complement wraparound property, the comb section following the integrator will compute the correct result at the output.

The frequency response of CIC filter, by evaluating $H(z)$ on the unit circle, $z = e^{j\omega T} = e^{j2\pi f}$, is given by

$$H(e^{j\omega T}) = \left(\frac{\sin(\frac{\omega TR}{2})}{\sin(\frac{\omega T}{2})} \right) e^{-j\omega T(R-1)/2} \tag{3.3}$$

The gain of the single stage CIC filter at $\omega T = 0$ is equal to the differential delay, R, of the comb section.

3.4.1 CIC Filters in Interpolators and Decimators

In the hardware implementations of decimation and interpolation, cascaded integrator-comb (CIC) filters are frequently used as a computationally efficient narrowband lowpass filters. These lowpass filters are well suited to improve the efficiency of anti-aliasing filters prior to decimation or the anti-imaging filters after the interpolation. In both of these applications, CIC filters have to operate at very high sampling rate. The CIC filters are generally more convenient for large conversion factors due to small lowpass bandwidth. In multistage decimators with a large conversion factor, the CIC filter is the best suited for the first decimation stage, whereas in interpolators, the comb filter is adopted for the last interpolation stage.

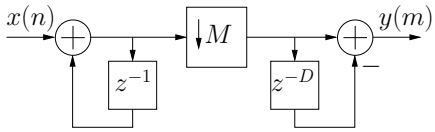


Figure 3.12: Single stage CIC decimation filter with $D = R/M$.

The cascade connection of integrator and comb in Fig. 3.11 can be interchanged as both of these are linear time-invariant operations. In a decimator structure, the integrator and comb are used as the first and the second section of the CIC filter, respectively. The structure also includes the decimation factor M. The sampling rate at the output as a result of this operation is F_{in}/M, where F_{in} is the input sampling rate. However, an important aspect with the CIC filters is the spectral aliasing resulting from the downsampling operation. The spectral bands centered at the integer multiples of $2/D$ will alias directly into the desired band after the downsampling operation. Similarly, the CIC filter with the upsampling factor L is used for the interpolation purpose. The sampling rate as a result of this operation is $F_{\text{in}}L$. However, imperfect filtering gives rise to unwanted spectral images. The filtering characteristics for the anti-aliasing and anti-image attenuation can be improved by increasing the number of stages of CIC filter.

The comb and integrator sections are both linear time-invariant and can follow or precede each other. However, it is generally preferred to place comb part on the side which has the lower sampling rate. It reduces the length of the delay line which is basically the differential delay of the comb section. When the decimation factor M is moved into the comb section using the noble identity and considering $R = DM$, the resulting architecture is the most common implementation of CIC decimation filters shown in Fig. 3.12. The delay length is reduced to $D = R/M$, which is now considered as the differential delay of comb part. One advantage is that the storage requirement is reduced and also the comb section now operates at a reduced clock rate. Both of these factors result in hardware saving and also low power consumption. Typically, the differential delay D is restricted to 1 or 2. The value of D also decides the number of nulls in the frequency response of the decimation filter. Similarly, the interpolation factor L is moved into the comb section using the noble identity and considering $R = DL$. The resulting interpolation structure is shown in Fig. 3.13. The delay length is reduced to $D = R/L$. Similar advantages can be achieved as in the case of CIC decimation filter. The important thing to note is that the integrator section operate at the higher sampling rate in both cases.

The transfer function of the K-stage CIC filter is

$$H(z) = H_I^K(z)H_C^K(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^K. \tag{3.4}$$

In order to modify the magnitude response of the CIC filter, there are only two

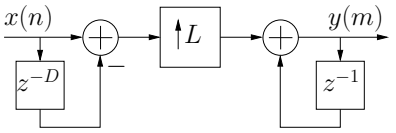


Figure 3.13: Single stage CIC interpolation filter with $D = R/L$.

parameters; the number of CIC filter stages and the differential delay D. The natural nulls of the CIC filter provide maximum alias rejection but the aliasing bandwidths around the nulls are narrow, and are usually not enough to provide sufficient aliasing rejection in the entire baseband of the signal. The advantage of increased number of stages is improvement in attenuation characteristics but the passband droop normally needs to be compensated.

3.4.2 Polyphase Implementation of Non-Recursive CIC

The polyphase decomposition of the non-recursive CIC filter transfer function may achieve lower power consumption than the corresponding recursive implementation [89, 90]. The basic non-recursive form of the transfer function is

$$H(z) = \frac{1}{D} \sum_{n=0}^{D-1} z^{-n}. \tag{3.5}$$

The challenge to achieve low power consumption are those stages of the filter which normally operate at higher input sampling rate. If the overall conversion factor D is power of two, $D = 2^J$, then the transfer function can be expressed as

$$H(z) = \prod_{i=0}^{J-1} (1 + z^{-2^i}) \tag{3.6}$$

As a result, the original converter can be transformed into a cascade of J factor-of-two converters. Each has a non-recursive sub-filter $(1 + z^{-1})^K$ and a factor-of-two conversion.

A single stage non-recursive CIC decimation filter is shown in Fig. 3.14. In the case of multistage implementation, one advantage is that only first decimation stage will operate at high input sampling rate. The sampling rate is successively reduced by two after each decimation stage as shown in Fig. 3.15. Further, the polyphase decomposition by a factor of two at each stage can be used to reduce the sampling rate by an additional factor of two. The second advantage of the non-recursive structures is that there are no more register overflow problems, if properly scaled. The register word length at any stage j for an input word length w_{in}, is limited to $(w_{\text{in}} + K \cdot j)$.

In an approach proposed in [89], the overall decimator is decomposed into a first-stage non-recursive filter with the decimation factor J_1, followed by a cascade of non-recursive $(1 + z^{-1})^K$ filters with a factor-of-2 decimation. The

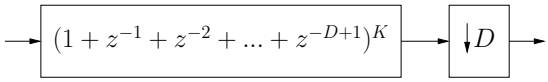


Figure 3.14: Non-recursive implementation of a CIC decimation filter.

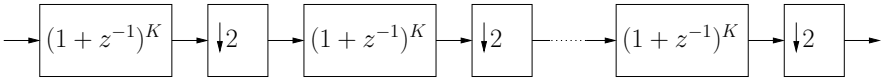


Figure 3.15: Cascade of J factor-of-two decimation filters.

polyphase decomposition along with noble identities is used to implement all these sub-filters.

3.4.3 Compensation Filters

The frequency magnitude response envelopes of the CIC filters are like $\sin(x)/x$, therefore, some compensation filter are normally used after the CIC filters. The purpose of the compensation filter is to compensate for the non-flat passband of the CIC filter.

Several approaches in the literature [91–95] are available for the CIC filter sharpening to improve the passband and stopband characteristics. They are based on the method proposed earlier in [96]. A two stage solution of sharpened comb decimation structure is proposed in [93–95, 97–100] for the case where sampling rate conversion is expressible as the product of two integers. The transfer function can then be written as the product of two filter sections. The role of these filter sections is then defined. One filter section may be used to provide sharpening while the other can provide stopband attenuation by selecting appropriate number of stages in each filter section.

Chapter 4

Non-Integer Sampling Rate Conversion

The need for a non-integer sampling rate conversion may appear when the two systems operating at different sampling rates have to be connected. This chapter gives a concise overview of SRC by a rational factor and implementation details.

The chapter first introduces the SRC by a rational factor. The technique for constructing efficient SRC by a rational factor based on FIR filters and polyphase decomposition is then presented. In addition, the sampling rate alteration with an arbitrary conversion factor is described. The polynomial-based approximation of the impulse response of a resampler model is then presented. Finally, the implementation of fractional-delay filters based on the Farrow structure is considered.

4.1 Sampling Rate Conversion: Rational Factor

The two basic operators, downsampler and upsampler, are used to change the sampling rate of a discrete-time signal by an integer factor only. Therefore, the sampling rate change by a rational factor, requires the cascade of upsampler and downsampler operators; Upsampling by a factor of L, followed by downsampling by a factor of M [9, 82]. For some cases, it may be beneficial to change the order of these operators, which is only possible if the L and M factors are relative prime, i.e., M and L do not share a common divisor greater than one. A cascade of these sampling rate alteration devices, results in sampling rate change by a rational factor L/M. The realizations shown in Fig. 4.1 are equivalent if the factors L and M are relative prime.

4.1.1 Small Rational Factors

The classical design of implementing the ratio L/M is upsampling by a factor of L followed by appropriate filtering, followed by a downsampling by a factor of M.

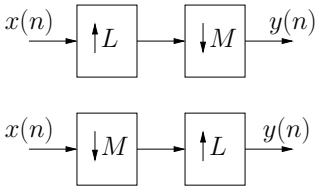


Figure 4.1: Two different realizations of rational factor L/M.

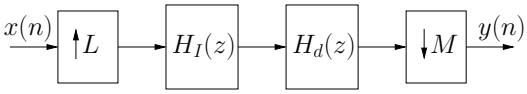


Figure 4.2: Cascade of an interpolator and decimator.

The implementation of a rational SRC scheme is shown in Fig. 4.2. The original input signal $x(n)$ is first upsampled by a factor of L followed by an anti-imaging filter also called interpolation filter. The interpolated signal is first filtered by an anti-aliasing filter before downsampling by a factor of M. The sampling rate of the output signal is

$$F_{\text{out}} = \frac{L}{M} F_{\text{in}}. \tag{4.1}$$

Since both filters, interpolation and the decimation, are next to each other, and operate at the same sampling rate. Both of these lowpass filters can be combined into a single lowpass filter $H(z)$ as shown in Fig. 4.3. The specification of the filter $H(z)$ needs to be selected such that it could act as both anti-imaging and anti-aliasing filters at the same time.

Since the role of the filter $H(z)$ is to act as an anti-imaging as well as an anti-aliasing filter, the stopband edge frequency of the ideal filter $H(z)$ in the rational SRC of Fig. 4.3 should be

$$\omega_s T = \min\left(\frac{\pi}{L}, \frac{\pi}{M}\right). \tag{4.2}$$

From the above equation it is clear that the passband will become more narrow and filter requirements will be hard for larger values of L and M. The ideal specification requirement for the magnitude response is

$$|H(e^{j\omega T})| = \begin{cases} L, & |\omega T| \leq \min\left(\frac{\pi}{L}, \frac{\pi}{M}\right) \\ 0, & \text{otherwise,} \end{cases} \tag{4.3}$$

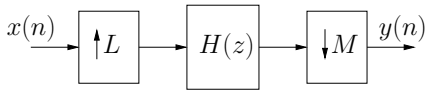


Figure 4.3: An implementation of SRC by a rational factor L/M.

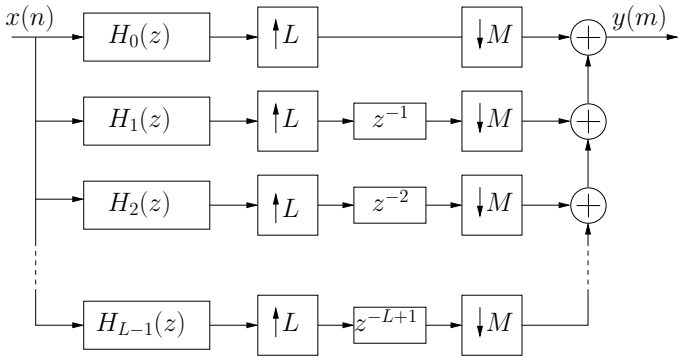


Figure 4.4: Polyphase realization of SRC by a rational factor L/M.

The frequency spectrum of the re-sampled signal, $Y(e^{j\omega T})$, as a function of spectrum of the original signal $X(e^{j\omega T})$, the filter transfer function $H(e^{j\omega T})$, and the interpolation and decimation factors L and M, can be expressed as

$$Y(e^{j\omega T}) = \frac{1}{M}\sum_{k=0}^{M-1} X(e^{j(L\omega T-2\pi k)/M})H(e^{j(\omega T-2\pi k)/M}). \tag{4.4}$$

It is apparent that the overall performance of the resampler depends on the filter characteristics.

4.1.2 Polyphase Implementation of Rational Sampling Rate Conversion

The polyphase representation is used for the efficient implementation of rational sampling rate converters, which then enables the arithmetic operations to be performed at the lowest possible sampling rate. The transfer function $H(z)$ in Fig. 4.3 is decomposed into L polyphase components using the approach shown in Fig. 1.11a. In the next step, the interpolation factor L is moved into the sub-filters using the computationally efficient polyphase representation form shown in Fig. 1.11. An equivalent delay block for each sub-filter branch is placed in cascade with the sub-filters. Since the downsampling operation is linear, and also by the use of noble identity, it can be moved into each branch of the sub-filters. The equivalent polyphase representation of the rational factor of L/M as a result of these modifications is shown in Fig. 4.4 [5, 10].

By using polyphase representation and noble identities, Fig. 4.4 has already attained the computational saving by a factor of L. The next motivation is to move the downsampling factor to the input side so that all filtering operations could be evaluated at $1/M$-th of the input sampling rate. The reorganization of the individual polyphase branches is targeted next. If the sampling conversion

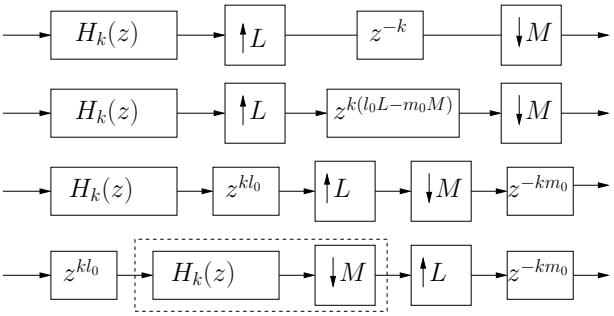


Figure 4.5: Reordering the upsampling and downsampling in the polyphase realization of k-th branch.

factors, L and M, are relative prime, then

$$l_0 L - m_0 M = -1 \tag{4.5}$$

where l_0 and m_0 are some integers. Using (4.5), the k-th delay can be represented as

$$z^{-k} = z^{k(l_0 L - m_0 M)} \tag{4.6}$$

The step-by-step reorganization of the k-th branch is shown in Fig. 4.5. The delay chain z^{-k} is first replaced by its equivalent form in (4.6), which then enables the interchange of downsampler and upsampler. Since both upsampler and downsampler are assumed to be relative prime, they can be interchanged. The fixed independent delay chains are also moved to the input and the output sides. As can be seen in Fig. 4.5, the sub-filter $H_k(z)$ is in cascade with the downsampling operation, as highlighted by the dashed block. The polyphase representation form in Fig. 1.10a and its equivalent computationally efficient form in Fig. 1.10 can be used for the polyphase representation of the sub-filter $H_k(z)$. The same procedure is followed for all other branches. As a result of these adjustments, all filtering operations now operate at $1/M$:th of the input sampling rate. The intermediate sampling rate for the filtering operation will be lower, which makes it a more efficient structure for a rational SRC by a factor L/M. The negative delay z^{kl_0} on the input side is adjusted by appropriately delaying the input to make the solution causal. The rational sampling rate conversion structures for the case $L/M < 1$ considered can be used to perform $L/M > 1$ by considering its dual.

4.1.3 Large Rational Factors

The polyphase decomposition approach is convenient in cases where the factors L and M are small. Otherwise, filters of a very high order are needed. For example, in the application of sampling rate conversion from 48 kHz to 44.1 kHz,

the rational factors requirement is $L/M = 147/160$, which imposes severe requirements for the filters. The stopband edge frequency from (4.2) is $\pi/160$. This will result in a high-order filter with a high complexity. In this case, a multi-stage realization will be beneficial [41].

4.2 Sampling Rate Conversion: Arbitrary Factor

In many applications, there is a need to compute the value of underlying continuous-time signal $x_a(t)$ at an arbitrary time instant between two existing samples. The computation of new sample values at some arbitrary points can be viewed as interpolation.

Assuming an input sequence, $\ldots, x((n_b-2)T_x), x((n_b-1)T_x), x(n_bT_x), x((n_b+1)T_x), x((n_b+2)T_x), \ldots$, uniformly sampled at the interval T_x. The requirement is to compute new sample value, $y(T_ym)$ at some time instant, T_ym, which occurs between the two existing samples $x(n_bT_x)$ and $x((n_b+1)T_x)$, where $T_ym = n_bT_x + T_xd$. The parameter n_b is some reference or base-point index, and d is called the fractional interval. The fractional interval can take on any value in the range $|T_xd| \leq 0.5$, to cover the whole sampling interval range. The new sample value $y(T_ym)$ is computed, using the existing samples, by utilizing some interpolation algorithm. The interpolation algorithm may in general be defined as a time-varying digital filter with the impulse response $h(n_b, d)$.

The interpolation problem can be considered as resampling, where the continuous-time function $y_a(t)$ is first reconstructed from a finite set of existing samples $x(n)$. The continuous-time signal can then be sampled at the desired instant, $y(T_ym) = y_a(T_ym)$. The commonly used interpolation techniques are based on polynomial approximations. For a given set of the input samples $x(-N_1 + n_b), \ldots, x(n_b), \ldots, x(n_b + N_2)$, the window or interval chosen is $N = N_1 + N_2 + 1$, A polynomial approximation $y_a(t)$ is then defined as

$$y_a(t) = \sum_{k=-N_1}^{N_2} P_k(t)x(n_b + k) \tag{4.7}$$

where $P_k(t)$ are polynomials. If the interpolation process is based on the Lagrange polynomials, then $P_k(t)$ is defined as

$$P_k(t) = \prod_{i=-N1, i\neq k}^{N_2} \frac{t - t_k}{t_k - t_i}, k = -N_1, -N_1 + 1, \ldots, N_2 - 1, N_2. \tag{4.8}$$

The Lagrange approximation also does the exact reconstruction of the original input samples, $x(n)$.

4.2.1 Farrow Structures

In conventional SRC implementation, if the SRC ratio changes, new filters are needed. This limits the flexibility in covering different SRC ratios. By utiliz-

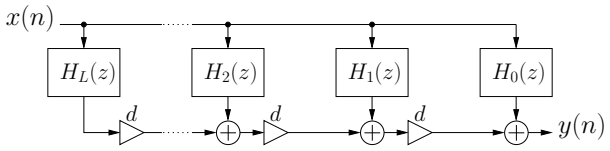


Figure 4.6: Farrow structure.

ing the Farrow structure, shown in Fig. 4.6, this can be solved in an elegant way. The Farrow structure is composed of linear-phase FIR sub-filters, $H_k(z)$, $k = 0, 1, \ldots, L$, with either a symmetric (for k even) or antisymmetric (for k odd) impulse response. The overall impulse response values are expressed as polynomials in the delay parameter. The implementation complexity of the Farrow structure is lower compared to alternatives such as online design or storage of a large number of different impulse responses. The corresponding filter structure makes use of a number of sub-filters and one adjustable fractional-delay as seen in Fig. 4.6. Let the desired frequency response, $H_{\text{des}}(e^{j\omega T}, d)$, of an adjustable FD filter be

$$H_{\text{des}}(e^{j\omega T}, d) = e^{-j\omega T(D+d)}, \quad |\omega T| \leq \omega_c T < \pi, \tag{4.9}$$

where D and d are fixed and adjustable real-valued constants, respectively. It is assumed that D is either an integer, or an integer plus a half, whereas d takes on values in the interval $[-1/2, 1/2]$. In this way, a whole sampling interval is covered by d, and the fractional-delay equals d $(d + 0.5)$ when D is an integer (an integer plus a half). The transfer function of the Farrow structure is

$$H(z, d) = \sum_{k=0}^{L} d^k H_k(z), \tag{4.10}$$

where $H_k(z)$ are fixed FIR sub-filters approximating kth-order differentiators with frequency responses

$$H_k(e^{j\omega T}) \approx e^{-j\omega TD} \frac{(-j\omega T)^k}{k!}, \tag{4.11}$$

which is obtained by truncating the Taylor series expansion of (4.9) to $L + 1$ terms. A filter with a transfer function in the form of (4.11) can approximate the ideal response in (4.10) as close as desired by choosing L and designing the sub-filters appropriately [30, 101]. When $H_k(z)$ are linear-phase FIR filters, the Farrow structure is often referred to as the modified Farrow structure. The Farrow structure is efficient for interpolation whereas, for decimation, it is better to use the transposed Farrow structure so as to avoid aliasing. The sub-filters can also have even or odd orders N_k. With odd N_k, all $H_k(z)$ are general filters whereas for even N_k, the filter $H_0(z)$ reduces to a pure delay. An alternative

representation of the transfer function of Farrow structure is

$$\begin{aligned} H(z,d) &= \sum_{k=0}^{L}\sum_{n=0}^{N} h_k(n) z^{-n} d^k, \\ &= \sum_{n=0}^{N}\sum_{k=0}^{L} h_k(n) d^k z^{-n}, \\ &= \sum_{n=0}^{N} h(n,d) z^{-n}. \end{aligned} \tag{4.12}$$

The quantity N is the order of the overall impulse response and

$$h(n,d) = \sum_{k=0}^{L} h_k(n) d^k. \tag{4.13}$$

Assuming T_{in} and T_{out} to be the sampling period of $x(n)$ and $y(n)$, respectively, the output sample index at the output of the Farrow structure is

$$n_{\text{out}} T_{\text{out}} = \begin{cases} (n_{\text{in}} + d(n_{\text{in}}))T_{\text{in}}, & \text{Even } N_k \\ (n_{\text{in}} + 0.5 + d(n_{\text{in}}))T_{\text{in}}, & \text{Odd } N_k, \end{cases} \tag{4.14}$$

where $n_{\text{in}}(n_{\text{out}})$ is the input (output) sample index. If d is constant for all input samples, the Farrow structure delays a bandlimited signal by a fixed d. If a signal needs to be delayed by two different values of d, in both cases, one set of $H_k(z)$ is used and only d is required to be modified.

In general, SRC can be seen as delaying every input sample with a different d. This delay depends on the SRC ratio. For interpolation, one can obtain new samples between any two consecutive samples of $x(n)$. With decimation, one can shift the original samples (or delay them in the time domain) to the positions which would belong to the decimated signal. Hence, some signal samples will be removed but some new samples will be produced. Thus, by controlling d for every input sample, the Farrow structure performs SRC. For decimation, $T_{\text{out}} > T_{\text{in}}$, whereas interpolation results in $T_{\text{out}} < T_{\text{in}}$.

Chapter 5

Polynomial Evaluation

The Farrow structure, shown in Fig. 4.6, requires evaluation of a polynomial of degree L with d as an independent variable. Motivated by that, this chapter reviews polynomial evaluation schemes and efficient evaluation of a required set of powers terms. Polynomial evaluation also has other applications [102], such as approximation of elementary functions in software or hardware [103, 104].

A uni-variate polynomial is a polynomial that has only one independent variable, whereas multi-variate polynomials involves multiple independent variables. In this chapter, only uni-variate polynomial, $p(x)$, is considered with an independent variable x. At the start of chapter, the classical Horner scheme is described, which is considered as an optimal way to evaluate a polynomial with minimum number of operations. A brief overview of parallel polynomial evaluation schemes is then presented in the central part of the chapter. Finally, the computation of the required set of powers is discussed. Two approaches are outlined to exploit potential sharing in the computations.

5.1 Polynomial Evaluation Algorithms

The most common form of a polynomial, $p(x)$, also called power form, is represented as

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n, \quad a_n \neq 0.$$

Here n is the order of the polynomial, and $a_0, a_1, \ldots, a_n$ are the polynomial coefficients.

In order to evaluate $p(x)$, the first solution that comes into mind is the transformation of all powers to multiplications. The $p(x)$ takes the form

$$p(x) = a_0 + a_1 \times x + a_2 \times x \times x + \cdots + a_n \times x \times x \cdots \times x, \quad a_n \neq 0.$$

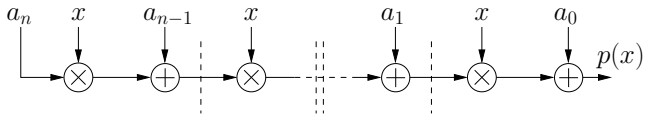


Figure 5.1: Horner's scheme for an n-th order polynomial.

However, there are more efficient ways to evaluate this polynomial. One possible way to start with is the most commonly used scheme in software and hardware named as Horner's scheme.

5.2 Horner's Scheme

Horner's scheme is simply a nested re-write of the polynomial. In this scheme, the polynomial $p(x)$ is represented in the form

$$p(x) = a_0 + x\Big(a_1 + x\big(a_2 + \cdots + x(a_{n-1} + a_n x)\ldots\big)\Big). \tag{5.1}$$

This leads to the structure in Fig. 5.1.

Horner's scheme has the minimum arithmetic complexity of any polynomial evaluation algorithm. For a polynomial order of n, n multiplications and n additions are used. However, it is purely sequential, and therefore becomes unsuitable for high-speed applications.

5.3 Parallel Schemes

Several algorithms with some degree of parallelism in the evaluation of polynomials have been proposed [105–110]. However, there is an increase in the number of operations for these parallel algorithms. Earlier works have primarily focused on either finding parallel schemes suitable for software realization [105–111] or how to find suitable polynomials for hardware implementation of function approximation [47, 112–116].

To compare different polynomial evaluation schemes, varying in degree of parallelism, different parameters such as computational complexity, number of operations of different types, critical path, pipelining complexity, and latency after pipelining may be considered. On the basis of these parameters, the suitable schemes can be shortlisted for an implementation given the specifications. Since all schemes have the same number of additions n, the difference is in the number of multiplication operations, as given in Table 5.1. Multiplications can be divided into three categories, data-data multiplications, squarers, and data-coefficient multiplications.

In squarers, both of the inputs of the multipliers are same. This leads to that the number of partial products can be roughly halved, and, hence, it is

reasonable to assume that a squarer has roughly half the area complexity as of a general multiplier [117, 118]. In data-coefficient multiplications, as explained in Chapter 2, the area complexity is reduced but hard to find as it depends on the coefficient. For simplicity, it is assumed that a constant multiplier has an area which is 1/4 of a general multiplier. As a result, two schemes having the same total number of multiplication but vary in types of multiplications are different with respect to implementation cost and other parameters. The scheme which has the lower number of data-data multiplications is cheaper to implement.

The schemes which are frequently used in literature are Horner and Estrin [105]. However, there are other polynomial evaluation schemes as well like Dorn's Generalized Horner Scheme [106], Munro and Paterson's Scheme [107, 108], Maruyama's Scheme [109], Even Odd (EO) Scheme, and Li et al. Scheme [110]. All the listed schemes have some good potential and useful properties. However, some schemes do not work for all polynomial orders.

The difference in all schemes is that how the schemes efficiently split the polynomial into sub-polynomials so that they can be evaluated in parallel. The other factor that is important is the type of powers required after splitting of the polynomial. Some schemes split the polynomial in such a way that only powers of the form x^{2^i} or x^{3^i} are required. The evaluation of such square and cube powers are normally more efficient than conventional multiplications. The data-coefficient multiplication count also gives the idea of degree of parallelism of that scheme. For example, Horner's scheme has only one data-coefficient multiplication, and hence, has one sequential flow; no degree of parallelism. The EO scheme, on other hand, has two data-coefficient multiplications, which result in two branches running in parallel. The disadvantage on the other hand is that the number of operations are increased and additional need of hardware to run operations in parallel. The applications where polynomials are required to be evaluated at run-time or have extensive use of it, any degree of parallelism in it or efficient evaluation relaxes the computational burden to satisfy required throughput.

Other factors that may be helpful in shortlisting any evaluation scheme are uniformity and simplicity of the resulting evaluation architecture and linearity of the scheme as the order grows; some schemes may be good for one polynomial order but not for the others.

5.4 Powers Evaluation

Evaluation of a set of powers, also known as exponentiation, not only finds application in polynomial evaluation [102, 119] but also in window-based exponentiation for cryptography [120, 121]. The required set of power terms to be evaluated may contain all powers up to some integer n or it has some sparse set of power terms that need to be evaluated. In the process of evaluating all powers at the same time, redundancy in computations at different levels is expected

Table 5.1: Evaluation schemes for fifth-order polynomial, additions (A), data-data multiplications (M), data-coefficient multiplications (C), squarers (S).

Scheme	Evaluation	A	M	C	S
Direct	$a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$	5	2	5	2
Horner	$((((a_5x + a_4)x + a_3)x + a_2)x + a_1)x + a_0$	5	1	4	0
Estrin	$(a_5x^2 + a_4x + a_3)x^3 + a_2x^2 + a_1x + a_0$	5	4	2	1
Li	$(a_5x + a_4)x^4 + (a_3x + a_2)x^2 + (a_1x + a_0)$	5	3	2	2
Dorn	$(a_5x^3 + a_2)x^2 + (a_4x^3 + a_1)x + (a_3x^3 + a_0)$	5	3	3	1
EO	$((a_5x^2 + a_3)x^2 + a_1)x + ((a_4x^2 + a_2)x^2 + a_0)$	5	2	3	1
Maruyama	$(a_5x + a_4)x^4 + a_3x^3 + (a_2x^2 + a_1x + a_0)$	5	4	2	2

which need to be exploited. Two independent approaches are considered. In the first, redundancy is removed at word level, while in the second, it is exploited at bit level.

5.4.1 Powers Evaluation with Sharing at Word-Level

A direct approach to compute x^i is the repeated multiplication of x with itself $i-1$ times, however, this approach is inefficient and becomes infeasible for large values of i. For example, to compute x^{23}, 22 repeated multiplications of x are required. With the binary approach in [102], the same can be achieved in eight multiplications, $\{x^2, x^4, x^5, x^{10}, x^{11}, x^{12}, x^{23}\}$, while the factor method in [102] gives a solution with one less multiplication, $\{x^2, x^3, x^4, x^7, x^8, x^{16}, x^{23}\}$. Another efficient way to evaluate a single power is to consider the relation to the addition chain problem. An addition chain for an integer n is a list of integers [102]

$$1 = a_0, a_1, a_2, \ldots, a_l = n, \tag{5.2}$$

such that

$$a_i = a_j + a_k, \quad k \leq j < i, i = 1, 2, \ldots, l. \tag{5.3}$$

As multiplication is additive in the logarithmic domain, a minimum length addition chain will give an optimal solution to compute a single power term. A minimum length addition chain for $n = 23$ gives the optimal solution with only six multiplications, $\{x^2, x^3, x^5, x^{10}, x^{20}, x^{23}\}$.

However, when these techniques, used for the computation of a single power [102, 122, 123], are applied in the evaluation of multiple power terms, it does not combine well in eliminating the overlapping terms that appear in the evaluation of individual powers as explained below with an example.

Suppose a sparse set a power terms, $T = \{22, 39, 50\}$, need to be evaluated.

A minimum length addition chain solution for each individual power is

$$22 \rightarrow \{1, 2, 3, 5, 10, 11, \mathbf{22}\}$$
$$39 \rightarrow \{1, 2, 3, 6, 12, 15, 27, \mathbf{39}\}$$
$$50 \rightarrow \{1, 2, 3, 6, 12, 24, 25, \mathbf{50}\}.$$

As can be seen, there are some overlapping terms, $\{2, 3, 6, 12\}$, which can be shared. If these overlapping terms are removed and the solutions are combined for the evaluation of the required set of powers, $\{22, 39, 50\}$, the solution is

$$\{22, 39, 50\} \rightarrow \{1, 2, 3, 5, 6, 10, 11, 12, 15, \mathbf{22}, 24, 25, 27, \mathbf{39}, \mathbf{50}\}. \tag{5.4}$$

However, the optimal solution for the evaluation of the requested set of powers, $\{22, 39, 50\}$, is

$$\{22, 39, 50\} \rightarrow \{1, 2, 3, 6, 8, 14, \mathbf{22}, 36, \mathbf{39}, \mathbf{50}\}, \tag{5.5}$$

which clearly shows a significant difference. This solution is obtained by addressing addition sequence problem, which can be considered as the generalization of addition chains. An addition sequence for the set of integers $T = n_1, n_2, \ldots, nr$ is an addition chain that contains each element of T. For example, an addition sequence computing $\{3, 7, 11\}$ is $\{1, 2, \mathbf{3}, 4, \mathbf{7}, 9, \mathbf{11}\}$.

The removal of redundancy at word-level does not strictly mean for the removal of overlapping terms only but to compute only those powers which could be used for evaluation of other powers as well in the set.

Note that the case where all powers of x, from one up to some integer n, are required to be computed is easy compared to the sparse case. Since every single multiplication will compute some power in the set, and it is assumed that each power is computed only once, any solution obtained will be optimal with respect to the minimum number of multiplications.

5.4.2 Powers Evaluation with Sharing at Bit-Level

In this approach, the evaluation of power terms is done in parallel using summations trees, similar to parallel multipliers. The PP matrices for all requested powers in the set are generated independently and sharing at bit level is explored in the PP matrices in order to remove redundant computations. The redundancy here relates to the fact that same three partial products may be present in more than one columns, and, hence, can be mapped to the same full adder.

A w-bit unsigned binary number can be expressed as following

$$X = x_{w-1}x_{w-2}x_{w-3}...x_2x_1x_0, \tag{5.6}$$

with a value of

$$X = \sum_{i=0}^{w-1} x_i 2^i, \tag{5.7}$$

where x_{w-1} is the MSB and x_0 is the LSB of the binary number. The n^{th} power of an unsigned binary number as in (5.7) is given by

$$X^n = \left(\sum_{i=0}^{w-1} x_i 2^i \right)^n . \tag{5.8}$$

For the powers with $n \geq 2$, the expression in (5.8) can be evaluated using the multinomial theorem. For any integer power n and a word length of w bits, the above equation can be simplified to a sum of weighted binary variables, which can further be simplified by using the identities $x_i x_j = x_j x_i$ and $x_i x_i = x_i$. The PP matrix of the corresponding power term can then be deduced from this function. In the process of generation of a PP matrix from the weighted function, the terms, e.g., $x_0 2^0$ will be placed in the first and $x_2 2^6$ in the 7-th column from the left in the PP matrix.

For the terms having weights other than a power of two, binary or CSD representation may be used. The advantage of using CSD representation over binary is that the size of PP matrix is reduced and therefore the number of full adders for accumulating PP are reduced. To make the analysis of the PP matrix simpler, the binary variables in the PP matrix can then be represented by their corresponding representation weights as given in Table 5.2.

Table 5.2: Equivalent representation of binary variables in PP matrix for $w = 3$.

Binary variable	Word (X)			Representation
	x_2	x_1	x_0	
x_0	0	0	1	1
x_1	0	1	0	2
$x_1 x_0$	0	1	1	3
x_2	1	0	0	4
$x_2 x_0$	1	0	1	5
$x_2 x_1$	1	1	0	6
$x_2 x_1 x_0$	1	1	1	7

In Fig. 5.2, an example case for $(n, w) = (5, 4)$, i.e., computing the square, cube, and fourth power of a five-bit input, is considered to demonstrate the sharing potential among multiple partial products. All partial products of the powers from x^2 up to x^5 are shown. As can be seen in Fig. 5.2, there are savings, since the partial product sets $\{9, 10, 13\}$ and $\{5, 7, 9\}$ are present in more than one column. Therefore, compared to adding the partial products in an arbitrary order, three full adders (two for the first set and one for the second) can be saved by making sure that exactly these sets of partial products are added.

$n=2$	$n=3$					$n=4$							
4	6	4	5	3	2	8	14	10	10	4	7	3	5
6	9	9	7	6	5	10	9	6	14	5	9	5	7
9	10	10	10	5	9	13	11	15	13	9	13	11	9
	13	13	11	7					15	10			
	12	11		9					17	13			
	14												

Figure 5.2: Partial product matrices with potential sharing of full adders, $n = 5$, $w = 4$.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this work, contributions beneficial for the implementation of integer and non-integer SRC were presented. By optimizing both the number of arithmetic operations and their word lengths, improved implementations are obtained. This not only reduces the area and power consumption but also allows a better comparisons between different SRC alternatives during a high-level system design. The comparisons are further improved for some cases by accurately estimating the switching activities and by introducing leakage power consumption.

6.2 Future Work

The following ideas are identified as possibilities for future work:

- ⋆ It would be beneficial to derive corresponding accurate switching activity models for the non-recursive CIC architectures. In this way an improved comparison can be made, where both architectures have more accurate switching activity estimates. Deriving a switching activity model for polyphase FIR filters would naturally have benefits for all polyphase FIR filters, not only the ones with CIC filter coefficients.
- ⋆ For interpolating recursive CIC filters, the output of the final comb stage will be embedded with zeros during the upsampling. Based on this, one could derive closed form expressions for the integrator taking this additional knowledge into account.
- ⋆ The length of the Farrow sub-filters usually differ between different sub-filters. This would motivate a study of pipelined polynomial evaluation

schemes where the coefficients arrive at different times. For the proposed matrix-vector multiplication scheme, this could be utilized to shift the rows corresponding to the sub-filters in such a way that the total implementation complexity for the matrix-vector multiplication and the pipelined polynomial evaluation is minimized.

- ⋆ The addition sequence model does not include pipelining. It would be possible to reformulate it to consider the amount of pipelining registers required as well. This could also include obtaining different power terms at different time steps.

Chapter 7

References

References

[1] S. K. Mitra, *Digital Signal Processing: A Computer Based Approach*, 2nd ed. New York: McGraw-Hill, 2001.

[2] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*. Prentice Hall, 1999.

[3] S. K. Mitra and J. F. Kaiser, *Handbook for Digital Signal Processing*. New York: Wiley, 1993.

[4] L. Wanhammar and H. Johansson, *Digital Filters Using MATLAB*. Linköping University, 2011.

[5] N. J. Fliege, *Multirate Digital Signal Processing: Multirate Systems, Filter Banks, Wavelets*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[6] R. Crochiere and L. Rabiner, "Optimum FIR digital filter implementations for decimation, interpolation, and narrow-band filtering," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 23, no. 5, pp. 444–456, Oct. 1975.

[7] R. E. Crochiere and L. R. Rabiner, "Interpolation and decimation of digital signals–a tutorial review," *Proc. IEEE*, vol. 69, no. 3, pp. 300–331, Mar. 1981.

[8] T. Ramstad, "Digital methods for conversion between arbitrary sampling frequencies," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 3, pp. 577–591, Jun. 1984.

[9] M. Bellanger, *Digital Processing of Signals*. John Wiley & Sons, Inc., 1984.

[10] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Prentice Hall, 1993.

[11] M. Bellanger, G. Bonnerot, and M. Coudreuse, "Digital filtering by polyphase network:application to sample-rate alteration and filter banks," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 24, no. 2, pp. 109–114, Apr. 1976.

[12] R. A. Valenzuela and A. G. Constantinides, "Digital signal processing schemes for efficient interpolation and decimation," *IEE Proc. Electr. Circuits Syst.*, vol. 130, no. 6, pp. 225–235, Dec. 1983.

[13] W. Drews and L. Gazsi, "A new design method for polyphase filters using all-pass sections," *IEEE Trans. Circuits Syst.*, vol. 33, no. 3, pp. 346–348, Mar. 1986.

[14] M. Renfors and T. Saramäki, "Recursive Nth–band digital filters – part I: Design and properties," *IEEE Trans. Circuits Syst.*, vol. 34, no. 1, pp. 24–39, Jan. 1987.

[15] ——, "Recursive Nth-band digital filters – part II: Design of multistage decimators and interpolators," *IEEE Trans. Circuits Syst.*, vol. 34, no. 1, pp. 40–51, Jan. 1987.

[16] F. M. Gardner, "Interpolation in digital modems. I. Fundamentals," *IEEE Trans. Commun.*, vol. 41, no. 3, pp. 501–507, Mar. 1993.

[17] L. Erup, F. M. Gardner, and R. A. Harris, "Interpolation in digital modems. II. Implementation and performance," *IEEE Trans. Commun.*, vol. 41, no. 6, pp. 998–1008, Jun. 1993.

[18] V. Tuukkanen, J. Vesma, and M. Renfors, "Combined interpolation and maximum likelihood symbol timing recovery in digital receivers," in *Proc. IEEE Int. Universal Personal Commun. Record Conf.*, vol. 2, 1997, pp. 698–702.

[19] M.-T. Shiue and C.-L. Wey, "Efficient implementation of interpolation technique for symbol timing recovery in DVB-T transceiver design," in *Proc. IEEE Int. Electro/Information Technol. Conf.*, 2006, pp. 427–431.

[20] S. R. Dooley and A. K. Nandi, "On explicit time delay estimation using the Farrow structure," *Signal Process.*, vol. 72, no. 1, pp. 53–57, Jan. 1999.

[21] M. Olsson, H. Johansson, and P. Löwenborg, "Time-delay estimation using Farrow-based fractional-delay FIR filters: filter approximation vs. estimation errors," in *Proc. Europ. Signal Process. Conf.*, 2006.

[22] ——, "Simultaneous estimation of gain, delay, and offset utilizing the Farrow structure," in *Proc. Europ. Conf. Circuit Theory Design*, 2007, pp. 244–247.

[23] C. W. Farrow, "A continuously variable digital delay element," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1988, pp. 2641–2645.

[24] T. Saramäki and T. Ritoniemi, "An efficient approach for conversion between arbitrary sampling frequencies," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2, 1996, pp. 285–288.

[25] D. Babic and M. Renfors, "Power efficient structure for conversion between arbitrary sampling rates," *IEEE Signal Process. Lett.*, vol. 12, no. 1, pp. 1–4, Jan. 2005.

[26] J. T. Kim, "Efficient implementation of polynomial interpolation filters or full digital receivers," *IEEE Trans. Consum. Electron.*, vol. 51, no. 1, pp. 175–178, Feb. 2005.

[27] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine, "Splitting the unit delay," *IEEE Signal Process. Mag.*, vol. 13, no. 1, pp. 30–60, Jan. 1996.

[28] J. Vesma, "A frequency-domain approach to polynomial-based interpolation and the Farrow structure," *IEEE Trans. Circuits Syst. II*, vol. 47, no. 3, pp. 206–209, Mar. 2000.

[29] R. Hamila, J. Vesma, and M. Renfors, "Polynomial-based maximum-likelihood technique for synchronization in digital receivers," *IEEE Trans. Circuits Syst. II*, vol. 49, no. 8, pp. 567–576, Aug. 2002.

[30] H. Johansson and P. Löwenborg, "On the design of adjustable fractional delay FIR filters," *IEEE Trans. Circuits Syst. II*, vol. 50, no. 4, pp. 164–169, Apr. 2003.

[31] J. Vesma and T. Saramäki, "Polynomial-based interpolation filters – part I: Filter synthesis," *Circuits Syst. Signal Process.*, vol. 26, no. 2, pp. 115–146, Apr. 2007.

[32] V. Välimäki and A. Haghparast, "Fractional delay filter design based on truncated Lagrange interpolation," *IEEE Signal Process. Lett.*, vol. 14, no. 11, pp. 816–819, Nov. 2007.

[33] H. H. Dam, A. Cantoni, S. Nordholm, and K. L. Teo, "Variable digital filter with group delay flatness specification or phase constraints," *IEEE Trans. Circuits Syst. II*, vol. 55, no. 5, pp. 442–446, May 2008.

[34] E. Hermanowicz, H. Johansson, and M. Rojewski, "A fractionally delaying complex Hilbert transform filter," *IEEE Trans. Circuits Syst. II*, vol. 55, no. 5, pp. 452–456, May 2008.

[35] C. Candan, "An efficient filtering structure for Lagrange interpolation," *IEEE Signal Process. Lett.*, vol. 14, no. 1, pp. 17–19, Jan. 2007.

[36] A. P. Chandrakasan and R. W. Brodersen, *Low-Power Digital CMOS Design*. Kluwer Academic Publishers, Boston-Dordrecht-London, 1995.

[37] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.

[38] N. S. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. S. Hu, M. J. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68–75, Dec. 2003.

[39] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proc. IEEE*, vol. 91, no. 2, pp. 305–327, Feb. 2003.

[40] International technology roadmap for semiconductors, http://public.itrs.net.

[41] L. Wanhammar, *DSP Integrated Circuits*. Academic Press, 1998.

[42] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. John Wiley and Sons, 1999.

[43] L. B. Jackson, "On the interaction of round-off noise and dynamic range in digital filters," *Bell Syst. Tech. J.*, vol. 49, pp. 159–184, Feb. 1970.

[44] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Trans. Signal Process.*, vol. 43, no. 12, pp. 3087–3090, Dec. 1995.

[45] K.-I. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 8, pp. 921–930, Aug. 2001.

[46] K. Han and B. L. Evans, "Optimum wordlength search using sensitivity information," *EURASIP J. Applied Signal Process.*, vol. 2006, pp. 1–14, Jan. 2006.

[47] D.-U. Lee and J. D. Villasenor, "A bit-width optimization methodology for polynomial-based function evaluation," *IEEE Trans. Comput.*, vol. 56, no. 4, pp. 567–571, Apr. 2007.

[48] P. D. Fiore, "Efficient approximate wordlength optimization," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1561–1570, Nov. 2008.

[49] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, *Synthesis And Optimization Of DSP Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 2004.

[50] P. D. Fiore and L. Lee, "Closed-form and real-time wordlength adaptation," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Process.*, vol. 4, 1999, pp. 1897–1900.

[51] N. Herve, D. Menard, and O. Sentieys, "Data wordlength optimization for FPGA synthesis," in *Proc. IEEE Workshop Signal Process. Syst. Design Implementation*, 2005, pp. 623–628.

[52] S. C. Chan and K. M. Tsui, "Wordlength determination algorithms for hardware implementation of linear time invariant systems with prescribed output accuracy," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2005, pp. 2607–2610.

[53] M. Vesterbacka, "On implementation of maximally fast wave digital filters," Ph.D. dissertation, Diss., no. 487, Linköping university, Sweden, June 1997.

[54] G.-K. Ma and F. J. Taylor, "Multiplier policies for digital signal processing," *IEEE ASSP Mag.*, vol. 7, no. 1, pp. 6–20, Jan. 1990.

[55] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proc. Circuits Devices Syst.*, vol. 141, no. 5, pp. 407–413, Oct. 1994.

[56] O. Gustafsson, A. G. Dempster, and L. Wanhammar, "Extended results for minimum-adder constant integer multipliers," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 1, 2002.

[57] O. Gustafsson, A. Dempster, K. Johansson, M. Macleod, and L. Wanhammar, "Simplified design of constant coefficient multipliers," *Circuits Syst. Signal Process.*, vol. 25, no. 2, pp. 225–251, Apr. 2006.

[58] S. He and M. Torkelson, "FPGA implementation of FIR filters using pipelined bit-serial canonical signed digit multipliers," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1994, pp. 81–84.

[59] K. Johansson, O. Gustafsson, and L. Wanhammar, "Low-complexity bit-serial constant-coefficient multipliers," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3, 2004.

[60] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proc. Circuits, Devices and Syst.*, vol. 138, no. 3, pp. 401–412, Jun. 1991.

[61] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 2, pp. 151–165, Feb. 1996.

[62] A. G. Dempster and M. D. Macleod, "Digital filter design using subexpression elimination and all signed-digit representations," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3, 2004.

[63] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II*, vol. 43, no. 10, pp. 677–688, Oct. 1996.

[64] M. Martinez-Peiro, E. I. Boemo, and L. Wanhammar, "Design of high-speed multiplierless filters using a non-recursive signed common subexpression algorithm," *IEEE Trans. Circuits Syst. II*, vol. 49, no. 3, pp. 196–203, Mar. 2002.

[65] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova, "A new algorithm for elimination of common subexpressions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 1, pp. 58–68, Jan. 1999.

[66] O. Gustafsson and L. Wanhammar, "A novel approach to multiple constant multiplication using minimum spanning trees," in *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, vol. 3, 2002.

[67] O. Gustafsson, H. Ohlsson, and L. Wanhammar, "Improved multiple constant multiplication using a minimum spanning tree," in *Proc. Asilomar Conf. Signals Syst. Comput.*, vol. 1, 2004, pp. 63–66.

[68] O. Gustafsson, "A difference based adder graph heuristic for multiple constant multiplication problems," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2007, pp. 1097–1100.

[69] K. Muhammad and K. Roy, "A graph theoretic approach for synthesizing very low-complexity high-speed digital filters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 2, pp. 204–216, Feb. 2002.

[70] H. Ohlsson, O. Gustafsson, and L. Wanhammar, "Implementation of low complexity FIR filters using a minimum spanning tree," in *Proc. IEEE Mediterranean Electrotechnical Conf.*, vol. 1, 2004, pp. 261–264.

[71] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst. II*, vol. 42, no. 9, pp. 569–577, Sep. 1995.

[72] A. G. Dempster, S. S. Dimirsoy, and I. Kale, "Designing multiplier blocks with low logic depth," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 5, 2002.

[73] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Trans. Algorithms*, vol. 3, no. 2, p. article 11, Apr. May 2007.

[74] A. G. Dempster and N. P. Murphy, "Efficient interpolators and filter banks using multiplier blocks," *IEEE Trans. Signal Process.*, vol. 48, no. 1, pp. 257–261, Jan. 2000.

[75] O. Gustafsson and A. G. Dempster, "On the use of multiple constant multiplication in polyphase FIR filters and filter banks," in *Proc. IEEE Nordic Signal Process. Symp.*, 2004, pp. 53–56.

[76] A. G. Dempster, O. Gustafsson, and J. O. Coleman, "Towards an algorithm for matrix multiplier blocks," in *Proc. Europ. Conf. Circuit Theory Design*, Sep. 2003.

[77] O. Gustafsson, H. Ohlsson, and L. Wanhammar, "Low-complexity constant coefficient matrix multiplication using a minimum spanning tree approach," in *Proc. IEEE Nordic Signal Process. Symp.*, 2004, pp. 141–144.

[78] M. D. Macleod and A. G. Dempster, "Common subexpression elimination algorithm for low-cost multiplierless implementation of matrix multipliers," *Electron. Lett.*, vol. 40, no. 11, pp. 651–652, May 2004.

[79] N. Boullis and A. Tisserand, "Some optimizations of hardware multiplication by constant matrices," *IEEE Trans. Comput.*, vol. 54, no. 10, pp. 1271–1282, Oct. 2005.

[80] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Optimization algorithms for the multiplierless realization of linear transforms," *ACM Trans. Des. Autom. Electron. Syst.*, accepted.

[81] L. Milic, *Multirate Filtering for Digital Signal Processing: MATLAB Applications.* Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, 2008.

[82] R. E. Crochiere and L. Rabiner, *Multirate Digital Signal Processing.* Prentice Hall, Englewood Cliffs, N.J., 1983.

[83] E. Hogenauer, "An economical class of digital filters for decimation and interpolation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 29, no. 2, pp. 155 –162, Apr. 1981.

[84] D. Babic, J. Vesma, and M. Renfors, "Decimation by irrational factor using CIC filter and linear interpolation," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, vol. 6, 2001, pp. 3677–3680.

[85] S. Chu and C. Burrus, "Multirate filter designs using comb filters," *IEEE Trans. Circuits Syst.*, vol. 31, no. 11, pp. 913–924, Nov. 1984.

[86] J. Candy, "Decimation for Sigma Delta modulation," *IEEE Trans. Commun.*, vol. 34, no. 1, pp. 72–76, Jan. 1986.

[87] Y. Gao, L. Jia, and H. Tenhunen, "A fifth-order comb decimation filter for multi-standard transceiver applications," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3, 2000, pp. 89–92.

[88] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays (Signals and Communication Technology).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2004.

[89] H. Aboushady, Y. Dumonteix, M.-M. Louerat, and H. Mehrez, "Efficient polyphase decomposition of comb decimation filters in $\Sigma\Delta$ analog-to-digital converters," *IEEE Trans. Circuits Syst. II*, vol. 48, no. 10, pp. 898–903, Oct. 2001.

[90] A. Blad and O. Gustafsson, "Integer linear programming-based bit-level optimization for high-speed FIR decimation filter architectures," *Circuits Syst. Signal Process.*, vol. 29, no. 1, pp. 81–101, Feb. 2010.

[91] A. Y. Kwentus, Z. Jiang, and J. Willson, A. N., "Application of filter sharpening to cascaded integrator-comb decimation filters," *IEEE Trans. Signal Process.*, vol. 45, no. 2, pp. 457–467, Feb. 1997.

[92] M. Laddomada and M. Mondin, "Decimation schemes for Sigma Delta A/D converters based on kaiser and hamming sharpened filters," *IEE Proc. –Vision, Image Signal Process.*, vol. 151, no. 4, pp. 287–296, Aug. 2004.

[93] G. Jovanovic-Dolecek and S. K. Mitra, "A new two-stage sharpened comb decimator," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 7, pp. 1414–1420, Jul. 2005.

[94] M. Laddomada, "Generalized comb decimation filters for Sigma Delta A/D converters: Analysis and design," *IEEE Trans. Circuits Syst. I*, vol. 54, no. 5, pp. 994–1005, May 2007.

[95] ——, "Comb-based decimation filters for Sigma Delta A/D converters: Novel schemes and comparisons," *IEEE Trans. Signal Process.*, vol. 55, no. 5, pp. 1769–1779, May 2007.

[96] J. Kaiser and R. Hamming, "Sharpening the response of a symmetric nonrecursive filter by multiple use of the same filter," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 5, pp. 415–422, Oct. 1977.

[97] T. Saramäki and T. Ritoniemi, "A modified comb filter structure for decimation," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 4, 1997, pp. 2353–2356.

[98] L. Lo Presti, "Efficient modified-sinc filters for sigma-delta A/D converters," *IEEE Trans. Circuits Syst. II*, vol. 47, no. 11, pp. 1204–1213, Nov. 2000.

[99] G. Stephen and R. W. Stewart, "Sharpening of partially non-recursive CIC decimation filters," in *Proc. Asilomar Conf. Signals Syst. Comput.*, vol. 2, 2004, pp. 2170–2173.

[100] G. J. Dolecek, "Modified CIC filter for rational sample rate conversion," in *Proc. Int. Symp. Commun. and Information Tech.*, 2007, pp. 252–255.

[101] J. Vesma, "Optimization and applications of polynomial-based interpolation filters," Ph.D. dissertation, Diss. no. 254, Tampere University of Technology, 1999.

[102] D. E. Knuth, *The Art of Computer Programming, vol. 2: Seminumerical Algorithms*, 1st ed. Addison-Wesley, Reading, Massachusetts, 1998.

[103] S. Lakshmivarahan and S. K. Dhall, *Analysis and Design of Parallel Algorithms: Arithmetic and Matrix Problems.* McGraw-Hill, 1990.

[104] J.-M. Muller, *Elementary Functions: Algorithms and Implementation*. Birkhauser, 2005.

[105] G. Estrin, "Organization of computer systems: the fixed plus variable structure computer," in *Proc. Western Joint IRE-AIEE-ACM Computer Conf.*, ser. IRE-AIEE-ACM '60 (Western). New York, NY, USA: ACM, 1960, pp. 33–40.

[106] W. S. Dorn, "Generalizations of Horner's rule for polynomial evaluation," *IBM J. Res. Dev.*, vol. 6, no. 2, pp. 239–245, Apr. 1962.

[107] I. Munro and A. Borodin, "Efficient evaluation of polynomial forms," *J. Comput. Syst. Sci.*, vol. 6, no. 6, pp. 625–638, Dec. 1972.

[108] I. Munro and M. Paterson, "Optimal algorithms for parallel polynomial evaluation," *J. Comput. Syst. Sci.*, vol. 7, no. 2, pp. 189–198, Apr. 1973.

[109] K. Maruyama, "On the parallel evaluation of polynomials," *IEEE Trans. Comput.*, vol. 22, no. 1, pp. 2–5, Jan. 1973.

[110] L. Li, J. Hu, and T. Nakamura, "A simple parallel algorithm for polynomial evaluation," *SIAM J. Sci. Comput.*, vol. 17, no. 1, pp. 260–262, Jan. 1996.

[111] J. Villalba, G. Bandera, M. A. Gonzalez, J. Hormigo, and E. L. Zapata, "Polynomial evaluation on multimedia processors," in *Proc. IEEE Int. Applicat.-Specific Syst. Arch. Processors Conf.*, 2002, pp. 265–274.

[112] D.-U. Lee, A. A. Gaffar, O. Mencer, and W. Luk, "Optimizing hardware function evaluation," *IEEE Trans. Comput.*, vol. 54, no. 12, pp. 1520–1531, Dec. 2005.

[113] D.-U. Lee, R. C. C. Cheung, W. Luk, and J. D. Villasenor, "Hardware implementation trade-offs of polynomial approximations and interpolations," *IEEE Trans. Comput.*, vol. 57, no. 5, pp. 686–701, May 2008.

[114] N. Brisebarre, J.-M. Muller, A. Tisserand, and S. Torres, "Hardware operators for function evaluation using sparse-coefficient polynomials," *Electron. Lett.*, vol. 42, no. 25, pp. 1441–1442, Dec. 2006.

[115] F. de Dinechin, M. Joldes, and B. Pasca, "Automatic generation of polynomial-based hardware architectures for function evaluation," in *Proc. IEEE Int. Applicat.-Specific Syst. Arch. Processors Conf.*, 2010, pp. 216–222.

[116] S. Nagayama, T. Sasao, and J. T. Butler, "Design method for numerical function generators based on polynomial approximation for FPGA implementation," in *Proc. Euromicro Conf. Digital Syst. Design Arch., Methods and Tools*, 2007, pp. 280–287.

[117] J. Pihl and E. J. Aas, "A multiplier and squarer generator for high performance DSP applications," in *Proc. IEEE Midwest Symp. Circuits Syst.*, vol. 1, 1996, pp. 109–112.

[118] J.-T. Yoo, K. F. Smith, and G. Gopalakrishnan, "A fast parallel squarer based on divide-and-conquer," *IEEE J. Solid-State Circuits*, vol. 32, no. 6, pp. 909–912, Jun. 1997.

[119] M. Wojko and H. ElGindy, "On determining polynomial evaluation structures for FPGA based custom computing machines," in *Proc. Australasian Comput. Arch. Conf.*, 1999, pp. 11–22.

[120] H. C. A. van Tilborg, *Encyclopedia of Cryptography and Security.* New York, USA,: Springer, 2005.

[121] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[122] A. C.-C. Yao, "On the evaluation of powers," *SIAM J. Comput.*, vol. 5, no. 1, pp. 100–103, Aug. 1976.

[123] D. Bleichenbacher and A. Flammenkamp, "An efficient algorithm for computing shortest addition chains," *SIAM J. Comput.*, 1998.