Low power and low complexity digital filters design and implementation

Shi, Dong

2010

Shi, D. (2010). Low power and low complexity digital filters design and implementation.
Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/42366>

<https://doi.org/10.32657/10356/42366>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**LOW POWER AND LOW COMPLEXITY DIGITAL FILTERS
DESIGN AND IMPLEMENTATION**

SHI DONG

**SCHOOL OF ELECTRICAL AND ELECTRONIC
ENGINEERING**

2010

LOW POWER AND LOW COMPLEXITY DIGITAL FILTERS DESIGN AND IMPLEMENTATION

SHI DONG

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
In partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2010

Acknowledgements

This thesis would not have been possible without the great supervision and support that has been so generously given to me, which I am heartily thankful for.

In the first place, I would like to express my sincerest gratitude to my supervisor, Professor Yu Ya Jun, for her constant guidance and advice throughout this research. In various ways, she has provided me extraordinary experience and unflinching encouragement. Her truly scientist intuition and enthusiasm in pursuing perfection made her a constant oasis of ideas and passions in scientific research, which exceptionally inspire and enrich my growth as a student, a researcher and a scientist. One simply could not wish for a better or friendlier supervisor.

I would like to gratefully acknowledge Professor Lim Yong Ching, Chang Chip Hong, and Oscar Gustafsson at Linköping University for their fruitful discussions, patient reviews and constructive comments on my work, which triggered and nourished my intellectual maturity that I will benefit from.

Many thanks to Professor Lin Zhi Ping, Chau Lap Pui and Gwee Bah Hwee for organizing plenty of insightful seminars and impressive events and providing me precious opportunities of participating in the IEEE Circuits and Systems Singapore Chapter Students Workshops.

Also, I would like to thank the people in Temasek Laboratories: Dr. Yang Chun Zhu, Dr. Lee Jun Wei, Mr. Leong Jenn Gan, Mr. Zhu Zhi Dong, Ms Xu Sai Hua, Mr. Xu Wei Jing and Mr. Shen Tian for their help during my stay in here. Special

thanks to Dr. Yang Chun Zhu for his guidance and assistance in many ways of my life and his brilliance in culinary artistic.

Finally, I would like to give my warmest thanks to my parents: Shi Qi and Ding Liang Zhen for their inseparable love and persistent confidence in me, without which I would never have gone through these years.

Contents

Acknowledgements	i
Summary	vi
1 Introduction	1
1.1 Contributions	3
1.2 Publications	4
1.3 Organization	5
2 Linear Phase FIR Filters	7
2.1 Basic Concepts	8
2.2 Structure	10
2.3 Complexity	15
2.3.1 Early Works	16
2.3.2 MCM Techniques in the First Category	17
2.3.3 MCM Techniques in the Second Category	22
2.4 Power Consumption	27
3 Residual Compensated Extrapolated Impulse Response Filters	34
3.1 The Extrapolated Impulse Response	36
3.2 Extrapolated Impulse Response with Residual Compensation	39
3.2.1 Structure	39
3.2.2 An Example	42

3.3	Optimizing Filter Coefficients and Residuals in Subexpression Space .	46
3.4	Design Examples	48
3.4.1	Example $L2$	48
3.4.2	Example $L1$	51
3.4.3	Five Additional Benchmark Filters	54
3.5	A Realization of the Extra Delay Elements	54
3.6	Discussion	58
3.6.1	Implementation Complexity	58
3.6.2	Adder Depth	59
3.6.3	Applicability and Limitation	61
3.7	Conclusion	62
4	Design of Linear Phase FIR Filters with High Probability of Achieving Minimum Number of Adders	68
4.1	Problem Formulation	69
4.1.1	Optimization Problem Formulation	70
4.1.2	Choosing the EWL	72
4.2	Traverse of the Discrete Solutions	73
4.3	Detailed Quantization Process	75
4.3.1	Dynamically Expanding Subexpression Basis Set	76
4.3.2	The Quantization Process of a Certain Coefficient	76
4.3.3	Adder Cost Estimation	79
4.3.4	Discrete Values Selection	80
4.3.5	Optimality of the Solution	82
4.4	Design Examples	83
4.4.1	Example A	84
4.4.2	Example B	86
4.4.3	Example C	87
4.5	Computational Complexity	89

4.6	Conclusion	92
5	Linear Phase FIR Filters in Cascade Form	95
5.1	Decomposition of Discrete Coefficient Filters	96
5.2	Proposed Algorithm	99
5.2.1	Problem Formulation	99
5.2.2	Traverse of Discrete Solutions	101
5.3	Some Remarks	102
5.4	An Example	104
5.5	Improving Searching Efficiency	106
5.5.1	Change of Searching Manner	108
5.5.2	Including Dynamically Expanding Subexpression Basis Set . .	109
5.6	Design Examples	110
5.7	Conclusion	113
6	MCM Implementation of Polyphase Linear Phase FIR filters with Restored Symmetry	119
6.1	Polyphase Structures with Restored Coefficient Symmetry	121
6.2	Complexities	125
6.2.1	Fixed Filter Length with Varying M	125
6.2.2	Fixed M with Varying Filter Length	127
6.2.3	Adder Depth	128
6.3	Conclusion	129
7	Conclusion	130
	Bibliography	132

Summary

Decades of technology advancing and rapid evolution of digital signal processors have made available the applications of digital signal processing theories into real integrated circuits with an astonishing boost in implementation cost and power consumption. As one of the essential components in many digital applications, digital filters have attracted much attention in low power and low complexity design and implementation.

In this thesis, the design and implementation of linear phase finite impulse response (FIR) filters are discussed in structural and algorithmic levels. First, a novel structure for synthesizing linear phase FIR filters is proposed. The proposed structure features a lower implementation complexity compared with the conventional FIR filter implementations. Furthermore, a filter optimization problem is formulated based on the proposed structure to produce designs with even lower complexity.

Second, in order to completely solve the problem of designing discrete coefficient linear phase FIR filters with minimum complexity, an algorithm is constructed. This algorithm is capable of producing the optimum design in most cases.

In order to provide better tradeoff between power consumption and implementation complexity, a new algorithm is proposed for the design of linear phase FIR filters in cascade form with discrete coefficients. It is shown that the resultant cascade structure outperforms the single-stage realization in achieving lower complexity, higher circuit speed and probably lower power consumption.

Finally, investigations in the multiple constants multiplication implementation of polyphase linear phase FIR filters with restored symmetry are performed. Comparisons between the proposed and conventional structures in implementation complexity, circuit speed and power consumption are made. It is observed that the proposed MCM polyphase realization leads to reduced implementation complexity with a slightly decrease in circuit speed.

List of Figures

2.1	Frequency response of a lowpass filter.	9
2.2	Digital filter design phases and their interactions	10
2.3	Implementation of an 8-th order linear phase FIR filter. (a) Transposed direct form (b) Direct form	11
2.4	Diagram of sampling rate conversion. (a) The process of decimation, (b) The process of interpolation.	12
2.5	FIR filters in polyphase structure. (a) Type I Decimator, (b) Type II Interpolator.	14
2.6	FIR filters in polyphase structure. (a) Type III Decimator, (b) Type IV Interpolator.	14
2.7	(a) Transposed direct form of an FIR filter. (b) Implementation of the multiplier block.	15
2.8	(a) Filter coefficients expressed in CSD format (b) Without common subexpression sharing (c) With common subexpression sharing	18
2.9	Coefficient set $\{1, 7, 16, 21, 33\}$ synthesized using adder graph.	19
2.10	Multiplier block synthesis using difference method.	20
2.11	Hybrid form of FIR filter with constant matrix multiplication block. .	21
2.12	Diagram of a B -bit (a) Carry Ripple Adder, (b) Carry Save Adder. .	28
2.13	An adder graph with adder depth=2.	30
2.14	Implementation of multiplier block using pipelines.	32

3.1	A typical impulse response of a $2N$ -th order symmetric FIR filter. For simplicity, the impulse response is centered at $n = 0$	36
3.2	Structure of extrapolated impulse response.	37
3.3	Extrapolated impulse response with residual compensation.	41
3.4	Extrapolated impulse response with residual compensation without including tail coefficients.	45
3.5	Modified extrapolated impulse response structure with the center co- efficient extrapolation.	49
3.6	Illustrative of the delay chain.	56
3.7	Circuit diagram of a delay chain.	57
3.8	Circuit diagram of a D-latch.	64
3.9	Circuit diagram of a D-flip-flop.	65
3.10	Circuit diagram of a 28-transistor full adder.	67
4.1	An example of the traverse process.	74
4.2	Diagram of the quantization process of coefficient $h(k)$	77
4.3	Computation time versus increasing filters orders (a) for $S1$ (solid) and (b) for a set of filters with a fixed ripple margin, but decreasing transition width (dashed). $EWL = 7$	90
4.4	Computation time of $G1$ with different EWL . The filter order is 15.	90
5.1	The decomposition process along a single discrete solution.	100
5.2	Magnitude responses of $H(z)$ (solid), $H_1(z)$ (dotted) and $H_2(z)$ (dashed).	105
5.3	Implementation of $H_1(z)$ and $H_2(z)$ in cascade form.	106
6.1	Polyphase structure for (a) Type I polyphase for decimator and (b) Type II polyphase for interpolator. T is a unit delay.	120
6.2	The implementation of Type I polyphase by exploiting the coefficient symmetry. T is a unit delay.	124

6.3	Total number of adders used to realize the MCM blocks with varying M and fixed filter length	126
6.4	Total number of adders used to realize the MCM blocks with varying filter lengths and fixed $M = 4$	127
6.5	Adder depth of conventional and proposed Type I decimators with varying filter length and fixed $M = 4$	128

List of Tables

2.1	Outputs of adding two numbers using a 4-bit CRA. $x = 1111$, $y = 0001$	29
3.1	Impulse Response of A 120-th order filter. $h(n)$ is the original discrete coefficient value obtained in [1], $h_a(n)$ is the extrapolated approximation of the original coefficient value, and $h_r(n)$ is the resulting residual. Coefficients from $h(40)$ to $h(48)$ are used as the prototype lobe.	43
3.2	Number of Adders Used to Synthesize the Filter Coefficients	44
3.3	Impulse Response of Example L2. Coefficients from $h(13)$ to $h(25)$ are chosen as the prototype lobe. Residuals from $h_r(0)$ to $h_r(12)$ and coefficients from $h(13)$ to $h(31)$ are realized in the multiplier block. $h(n) = h_r(n) + \alpha_0 h(n + 13)$ for $n = 0, 1, \dots, 12$, where $\alpha_0 = -4$	50
3.4	Impulse Response of Example L1. Coefficients from $h(37)$ to $h(45)$ are chosen as the prototype lobe. Residuals from $h_r(1)$ to $h_r(36)$, coefficients $h(0)$ and $h(37)$ to $h(60)$ are realized in the multiplier block. $h(n - ld) = h_r(n - ld) + \alpha_{L-1-l} h(n)$ for $n = 37, 38, \dots, 45$, $l = 1, 2, 3, 4$, $L = 5$ and $d = 9$, where $\alpha_0 = 16$, $\alpha_1 = -8$, $\alpha_2 = 4$ and $\alpha_3 = -2$	53
3.5	Number of Adders Used to Implement the Filters	55
3.6	Average power of B -bit delay chain, full adder and tap delay element	58
3.7	Complexity Comparison of Delay Chains and Full Adders. The external signal wordlength is assumed to be 12 bits.	58
3.8	States of the storage unit	64

3.9	The first order estimation of the Power Consumed by the full adder in all transitions	67
4.1	An example of adder cost estimation with the current best result $N_{best} = 13$	79
4.2	Specifications of design examples. ω_p , ω_s , δ_p and δ_p are the passband edge, stopband edge, passband ripple and stopband ripple, respectively.	83
4.3	Specifications of $L3$. δ_p^+ and δ_p^- are the upper bound and lower bound of the passband, respectively. The filter length is 36.	83
4.4	Results and Comparison of Example A. EWL is the effective wordlength of the filter. MBA and SA are the number of multiplier block adders and structural adders respectively.	85
4.5	Impulse response and subexpression basis set of design examples $X1$.	85
4.6	Impulse response of design example $L2$ in residual compensated ex- trapolated impulse response structure. Coefficients from $h(13)$ to $h(25)$ are chosen as the prototype lobe. Residuals from $h_r(0)$ to $h_r(12)$ and coefficients from $h(13)$ to $h(31)$ are realized in the multi- plier block. $h(n) = h_r(n) + \alpha_0 h(n + 13)$ for $n = 0, 1, \dots, 12$, where $\alpha_0 = -4$	88
4.7	Impulse response and subexpression basis set of design examples $S1$ with $MAD = 2$	88
4.8	Impulse responses and subexpression basis sets of design examples $Y1$, $Y2$, $G1$ and $A1$	93
4.9	Impulse responses and subexpression basis sets of design examples $L2$, $L3$, $S1$ and $S2$	94
5.1	Impulse responses of design Example A, $H(z) = H_1(z)H_2(z)$	104
5.2	Specifications of design examples. ω_p , ω_s , δ_p and δ_p are the passband edge, stopband edge, passband ripple and stopband ripple, respectively.	111

5.3	Specifications of $L3$. δ_p^- and δ_p^+ are the upper bound and lower bound of the passband, respectively. The filter length is 36.	111
5.4	Summary of design results. N_i , MBA_i and SA_i are the order, number of multiplier adders and number of structural adders of subfilter i , respectively. SS represents the optimum single-stage design using the algorithm proposed in Chapter 4 and EWL is the effective wordlength of the overall filter.	112
5.5	Summary of design results of $G1$, $S1$ and $L3$ with increased filter lengths. N_i , MBA_i and SA_i are the order, number of multiplier adders and number of structural adders of subfilters i , respectively. SS represents the optimum single-stage design using the algorithm proposed in Chapter 4 and EWL is the effective wordlength of the overall filter.	113
5.6	Impulse responses of design examples $Y1$ and $Y2$	115
5.7	Impulse response of design example $Y3$	116
5.8	Impulse response of design example $A1$	116
5.9	Impulse response of design example $S2$	117
5.10	Impulse response of design example $L2$	117
5.11	Impulse response of design example $G1$	118
5.12	Impulse responses of design examples $S1$ and $L3$	118

Chapter 1

Introduction

The last century witnessed an unprecedented development in digital signal processing (DSP) which was made available as continuous breakthroughs in technology scaling have been achieved [2–6]. Being one of the subfields of signal processing, DSP has found numerous advantages over analog signal processing in stability, error correction, vulnerability to noise and temperature shift, and thus is widely applied in speech and audio coding, statistical signal processing, video and image coding, biomedical signal process and etc [7–12].

As an essential part of many digital signal processing applications, digital filters have received great attention since 1960's, as requirements in increasing sampling rate and resolution have tightened the design constraints of the entire digital system. Tremendous researches have been done in achieving low power and low complexity digital filters, in which the design and implementation of finite impulse response (FIR) filters have been deeply studied. As a result, significant improvements have been made in obtaining power and complexity reduction in practical FIR filters, especially after the general coefficient multipliers, being the most power hungry and complicated part of FIR filters, have been realized using multiplierless techniques [1, 13–22] in very large scale integration (VLSI). Therefore, for the past several decades, much attention has been given to the design of multiplierless FIR filters, in which the general multiplication of the input signal with filter coefficients is realized by using a sequence of shifts and adds.

In practical implementation, filter coefficients are of discrete values, the representations that are used to express the coefficients are of much importance. Therefore, it is more hardware efficient to modify the conventional two's complement representation of filter coefficients by allowing minus 1 as well. The best results ever obtained is the canonic signed digit (CSD) representation, which is also known as canonic signed power of two (SPT) terms. Moreover, instead of realizing the filter coefficients separately, subsequent researches have shown that the partial terms or subexpressions among the CSD or SPT terms of the filter coefficients can be shared during the multiplication with the input data. Thus, for the past two decades, the MCM techniques [23–45] have been an active research area.

To implement FIR filters with low complexity, many algorithms for designing FIR filters with discrete coefficients, mainly to minimize frequency response errors in the sense of least mean squares and minimax, have been proposed [1, 36, 38, 46–60]. One of the most commonly used techniques in optimizing discrete-valued FIR filters is mixed integer linear programming (MILP) [14, 47, 61] which is capable of finding the optimum solution; the optimum is in the sense that the maximum frequency response ripple is minimized for a given discrete coefficient space. Additionally, in order to overcome the high computational complexity of MILP, “sub-optimal” approaches have been proposed to optimizing FIR filters in a reasonable and acceptable amount of time [48, 51, 56, 61–64]. Recently, many algorithms have been constructed to further incorporate MCM techniques in the design process [36, 38, 54–60], resulting in even lower implementation complexity compared with directly applying MCM techniques on a given discrete coefficient set.

The objective of this research is to develop novel methods for the implementation and design of low complexity and low power FIR filters. Attempts have been made mainly in structural and algorithmic level, respectively.

1.1 Contributions

The contributions of this thesis mainly lie in the following:

- An extrapolated impulse response filter with residual compensation is proposed for the design of discrete coefficient FIR filters using MCM techniques. The proposed technique utilizes the quasi-periodic nature of the filter impulse response to approximate the filter coefficients. The reduced degree of freedom of filter coefficients due to the quasi-periodic approximation is perfectly restored by introducing a residual compensation technique. The resulting subexpression sharing synthesis of discrete coefficient FIR filters has lower complexity than that of the conventional synthesis techniques in terms of number of adders. To further reduce the synthesis complexity, filter coefficients and residuals may be optimized in subexpression spaces. Mixed integer linear programming (MILP) is formulated for the optimization.
- An algorithm is proposed for the design of low complexity linear phase FIR filters with optimum discrete coefficients. The proposed algorithm, based on MILP, efficiently traverses the discrete coefficient solutions and searches for the optimum one that results in an implementation using minimum number of adders. During the searching process, discrete coefficients are dynamically synthesized based on a continuously updated subexpression space and, most essentially, a monitoring mechanism is introduced to enable the algorithm's awareness of optimality.
- An algorithm is proposed for the design of linear phase FIR filters in cascade form with discrete coefficients. The proposed algorithm generates the coefficients of the subfilters simultaneously in integer space and finds all the possible discrete valued solutions.

- The MCM implementation of polyphase structures with coefficient symmetry is investigated, where each subfilter MCM block is synthesized separately. It is shown that by using the proposed implementation, it is possible to reduce the number of adders compared with currently existing methods without increasing the storage elements. The proposed techniques can also be extended to the matrix MCM block implementation.

1.2 Publications

The main content of this thesis can be found in the following papers which have been published or will be published or are submitted for consideration of publication.

Conference Papers:

- Y. J. Yu, D. Shi, and Y. C. Lim, "Subexpression encoded extrapolated impulse response FIR filter with perfect residual compensation," in *Proc. IEEE ISCAS'08*, 2008, pp. 2446-2449.
- Y. J. Yu, D. Shi and R. Bregovic, "On the complexity reduction of polyphase linear phase FIR filters with symmetric coefficient implementation," in *Proc. IEEE ISCAS'09*, 2009, pp. 24-27.
- D. Shi and Y. J. Yu, "Low-complexity Linear Phase FIR Filters in Cascade Form," in *Proc. IEEE ISCAS'10*, 2010, pp. 177 - 180.

Journal Papers:

- Y. J. Yu, Y. C. Lim and D. Shi, "Low complexity design of variable bandedge linear phase FIR filters with sharp transition band," in *IEEE Trans. Signal Process.*, vol. 57, no 4, pp. 1328-1338, April, 2009.

- Y. J. Yu, D. Shi, and Y. C. Lim, “Design of Extrapolated Impulse Response FIR Filters With Residual Compensation in Subexpression Space,” in *IEEE Trans. Circuits Syst I.*, vol. 56, no 12, pp. 2623-2633, December, 2009.
- D. Shi and Y. J. Yu, “Design of Linear Phase FIR Filters with High Probability of Achieving Minimum Number of Adders,” to appear in *IEEE Trans. Circuits Syst I.*

1.3 Organization

In Chapter 1, the design and implementation issues of FIR filters are briefly introduced. The goal and main contributions of this thesis are stated. In Chapter 2, literatures in structures, complexity and power consumption of linear phase FIR filters are reviewed.

A novel structure for realizing linear phase FIR filters using MCM techniques is proposed in Chapter 3, which results in even lower implementation complexity compared with the conventional structures. A MILP based algorithm is further proposed to optimize the filter coefficients of the new structure in subexpression space.

An algorithm capable of designing linear phase FIR filters using minimum number of adders was developed and is presented in Chapter 4. Design examples have shown that the proposed algorithm can, in most cases, produce the optimum designs using minimum number of adders for the given specifications. Furthermore, the proposed algorithm can be simply extended for the optimum design with the maximum adder depth (MAD) constraint.

In Chapter 5, a new algorithm is proposed for the design of linear phase FIR filters in cascade form with discrete coefficients. Design examples are given to show the superiority of the proposed algorithm in achieving low implementation complexity

and low adder depth. Improved version of the algorithm is illustrated to efficiently design longer filters.

In Chapter 6, investigations in the MCM implementation of polyphase linear phase FIR filters with restored symmetry are performed. Comparisons in implementation complexity, adder depth, circuit speed and power consumption between the proposed and conventional structures are made.

Chapter 7 concludes this thesis.

Chapter 2

Linear Phase FIR Filters

A digital filter is a filter that operates on digital signals. Digital filters, composed of digital components of multipliers, adders and delays, are in contrast to older analog filters which work entirely in the analog realm and must rely on physical networks of electronic components (such as resistors, capacitors, transistors, etc.) to achieve the desired filtering effect. Among all the types of digital filters, the linear phase FIR filters have found a wide range of applications due to its stability, limited cycles of oscillation and preservation of phase linearity [42, 49, 61, 64–66]. Despite the wide applications of linear phase FIR filters, which can be found in speech coding, image processing, multi-rate systems and etc, the complexity and power consumption of linear phase FIR filters are usually much higher than those of their infinite impulse response (IIR) filters counterparts meeting the same magnitude response specifications. Therefore, many efforts have been dedicated to the design of low complexity and low power linear phase FIR filters. In this chapter, some of the most prevalent and significant researches and works in these two aspects are reviewed. This chapter is begun by the introduction of the very basic concepts and structures of linear phase FIR filters.

2.1 Basic Concepts

Mathematically, the characteristics of an N -th order FIR filter can be completely specified by its impulse response $h(n)$ for $n = 0, 1, 2, \dots, N$. The z -transform function of $h(n)$ is given by

$$H(z) = \sum_{n=0}^N h(n)z^{-n}. \quad (2.1)$$

The frequency response of the filter can be obtained by replacing z in (2.1) with $e^{j\omega}$,

$$H(e^{j\omega}) = \sum_{n=0}^N h(n)e^{-jn\omega}. \quad (2.2)$$

For a linear phase FIR filter, its impulse response $h(n)$ can be either symmetrical or anti-symmetrical. For example, suppose that N is even and $h(n)$ is symmetrical, (2.2) can be further written as

$$H(e^{j\omega}) = e^{-j\frac{N}{2}\omega} \left\{ h\left(\frac{N}{2}\right) + 2 \sum_{n=1}^{\frac{N}{2}} h\left(\frac{N}{2} - n\right) \cos(n\omega) \right\}. \quad (2.3)$$

It can be seen from (2.3) that the term $e^{-j\frac{N}{2}\omega}$ is the linear phase term because the derivative of its phase $(-\frac{N}{2}\omega)$ with respect to the frequency ω is a constant $-\frac{N}{2}$. This feature is essential in many applications where phase linearity is a strict requirement. For expository convenience, $H(e^{j\omega})$ is denoted as the zero-phase frequency response (with the linear phase term removed).

Since the order of a linear phase FIR filter can be even or odd and its impulse response can be symmetrical or anti-symmetrical, there are 4 types of linear phase FIR filters. Their frequency responses are given below.

$$H(e^{j\omega}) = \begin{cases} h\left(\frac{N}{2}\right) + 2 \sum_{n=1}^{\frac{N}{2}} h\left(\frac{N}{2} - n\right) \cos(n\omega) & N \text{ even, symmetrical} \\ 2 \sum_{n=1}^{\frac{N+1}{2}} h\left(\frac{N+1}{2} - n\right) \cos\left((n - \frac{1}{2})\omega\right) & N \text{ odd, symmetrical} \\ 2 \sum_{n=1}^{\frac{N}{2}} h\left(\frac{N}{2} - n\right) \sin(n\omega) & N \text{ even, anti-symmetrical} \\ 2 \sum_{n=1}^{\frac{N+1}{2}} h\left(\frac{N+1}{2} - n\right) \sin\left((n - \frac{1}{2})\omega\right) & N \text{ odd, anti-symmetrical} \end{cases}$$

It is obvious that the frequency responses of these four types linear phase FIR filters are real-valued and are the linear combinations of Trigonometric functions. These two facts provide us with much easiness and convenience in the design of linear phase FIR filters, which will be shown in the sections and chapters that follow.

Figure 2.1: Frequency response of a lowpass filter.

The design of a linear phase FIR filter is usually subject to a given set of specifications of its frequency response. Figure 2.1 shows the frequency response of a typical lowpass filter, where ω_p , ω_s , δ_p , δ_s and Δ are the passband edge, stopband

edge, passband ripple, stopband ripple and transition width, respectively. For the design of highpass, bandpass and bandstop filters, the specification are specified in a similar way.

With the given specifications, the order N of the filter can be estimated using the following equation [3]:

$$N \approx \frac{-20\log_{10}\sqrt{\delta_p\delta_s} - 13}{14.6\Delta/2\pi}. \quad (2.4)$$

It can be seen from (2.4) that the order of the filter increases when the passband and stopband ripple requirement gets stricter. Furthermore, N increases more drastically when the transition bandwidth Δ decreases.

2.2 Structure

The design and implementation of digital filters involves several interacting phases. In each phase there are plenty of freedom for the designer to choose [3]. Therefore, as shown in Figure 2.2, designers usually need to go iteratively among these phases in order to produce a decent final result.

Figure 2.2: Digital filter design phases and their interactions

Hence, the choice of implementation structures is of essence and depends strongly on the specific application. In addition, the structures, in some degree (or to a large extent) determine many of the practical properties of the final implementation such

as the circuit throughput, area cost, power consumption, hardware interconnection and so on.

Figure 2.3: Implementation of an 8-th order linear phase FIR filter. (a) Transposed direct form (b) Direct form

Direct and Transposed Direct Forms

The most conventional structures for implementing digital filters are shown in Figure 2.3. These two structures can be found in most digital filter applications and are the basic components of more complicated digital filters implementation. The structure shown in Figure 2.3(a) is called transposed direct form while the structure shown in Figure 2.3(b) is referred to as direct form. Apparently, there is a clear and simple relationship the structures and the z -transfer function of the impulse response $h(n]$ in (2.1). Also, it can be seen that both structures contain three different circuit elements, which are delay elements, structural adders and multipliers, respectively. Although mathematically equivalent, these two structures, in practical circuit implementation, show different properties. For example, the transposed

direct form has higher throughput than the direct form while the latter has much smaller input capacitance. For linear phase FIR filter, since its impulse response is symmetrical or anti-symmetrical, half multipliers saving can be achieved in both structures.

Polyphase Decomposition

Another important application of FIR filters lies in multirate signal processing, in which different parts work at different sampling rates. In fact, multiple sampling rates are unavoidable for certain signal processing purposes or are introduced to alleviate the computational complexity. The following introduces the polyphase structure of FIR filters in sampling rate conversion where different stages of a system operate at different sampling rates. The operation of decreasing the sampling rate is referred to as decimation, whereas the opposite is called interpolation.

Figure 2.4: Diagram of sampling rate conversion. (a) The process of decimation, (b) The process of interpolation.

Figure 2.4 shows the diagram of the process of decimation and interpolation. It can be seen that for the decimation process in Figure 2.4(a), the sampling rate of the input $x(n)$ is M times of the output $y(n)$. Similarly, for the interpolation process in Figure 2.4(b), the sampling rate of the output $y(n)$ is P times of the input $x(n)$. Here, we only consider the case where M and P are positive integers. It should also be noted that, in either case, there is a need to use digital filter (often FIR filter). For the decimation process, $H_D(z)$ acts as an anti-aliasing filter that prevents the overlapping of frequency components after decimation. On the other hand, for the the interpolation process, $H_I(z)$ is used to remove the periodic frequency contents of $v(n)$, which is the interpolated version of $x(n)$. The detailed principles behind

the design of these two filters can be found in most literature of multirate signal processing [9, 12] and are not discussed here. However, it is of importance to show the structures of $H_D(z)$ and $H_I(z)$ by means of polyphase decomposition.

Recall that the z -transform function of an FIR filter is given by

$$H(z) = \sum_{n=0}^N h(n)z^{-n}, \quad (2.5)$$

where N is the filter order. Suppose that N satisfies $N + 1 = ML$ where M, L are both positive integers. (2.5) can be rewritten as

$$H(z) = \sum_{i=0}^{M-1} \left[\sum_{j=0}^{L-1} h(Mj + i)z^{-Mj} \right] z^{-i}. \quad (2.6)$$

(2.6) indicates that $H(z)$ can be decomposed into M subfilters, each having length of L . Due to the noble identities [5], the interpolation filter $H_I(z)$ and decimation filter $H_D(z)$ of order N in Figure 2.4 can be realized using M subfilters as shown in Figure 2.5, for $N = 7$ and $M = 2$. This results in the fact that $H_D(z)$ and $H_I(z)$ can work at only $\frac{1}{M}$ sampling rate as illustrated in Figure 2.4. Therefore, the computational burden is significantly relieved.

It can be seen from Figure 2.5 that the overall filter is decomposed into two subfilters, each having 4 taps and working at the input or output sampling rate, whichever is lower. The structure in Figure 2.5(a) is referred to as Type I decimator. By transposing Figure 2.5(a), we obtain the Type II interpolator in Figure 2.5(b).

There are another two polyphase structures to realize the decimator and interpolator which are shown in Figure 2.6(a) and Figure 2.6(b) as Type III decimator and Type IV interpolator, respectively. It should be pointed out that Type III and Type IV structures require more delay elements than their Type I and Type II counterparts. It should be noted that the matrix MCM technique [67, 68] (discussed later in this chapter) has uses in polyphase FIR filters for sample rate conversions.

Figure 2.5: FIR filters in polyphase structure. (a) Type I Decimator, (b) Type II Interpolator.

Figure 2.6: FIR filters in polyphase structure. (a) Type III Decimator, (b) Type IV Interpolator.

2.3 Complexity

In this section, the actual modern realization schemes of FIR filters are introduced. Furthermore, typical techniques in synthesizing and optimizing linear phase FIR filters are discussed.

Figure 2.7: (a) Transposed direct form of an FIR filter. (b) Implementation of the multiplier block.

Shown in Figure 2.7 is the detailed implementation of a transposed direct form FIR filter, where the input signal is first multiplied by the constant filter coefficients and then goes into the delay elements. This operation is often referred to as MCM problem because the same signal $x(n)$ is multiplied by multiple constant filter coefficients. The constant multipliers can be realized using multiplierless techniques where the general multipliers are replaced by a network of shifts and adders, as indicated in Figure 2.7(b). It can be seen that the adders can be further classified into structural adders (SA) and multiplier block (MB) adders. SA are used to add the temporarily stored values in the delay elements with the products of the input

signal with the filter coefficients, whereas MB adders are used in the multiplier block (MB) to generate the constant filter coefficients. Since the shifts in the MB can be hard-wired and thus are considered cost free and the number of delay elements cannot be reduced, the complexity of FIR filters is therefore very much related to the number of adders (MB adders and SA).

2.3.1 Early Works

Before the application of MCM techniques, filter coefficients were usually expressed in CSD form or signed power-of-two terms (SPT) and realized separately. Therefore, since the number of MB adders is proportional to the number of CSD or SPT terms used to represent the coefficients, many algorithms have been proposed to design filter coefficients within this context [1, 14–17, 21, 48, 51, 61–64, 69], i.e., to reduce the number of CSD or SPT terms.

In [48, 61], the total number of SPT terms used to represent filter coefficients are pre-defined. Polynomial-time algorithms are used to find the coefficient set that best approximates the infinite precision design. In [1, 14, 17], the design of FIR filters is formulated as a linear programming problem or least mean square problem. The filter coefficients are quantized in their own SPT space one by one during the optimization. Every time when a coefficient is fixed, the other unfixed coefficients are optimized again to compensate for the frequency response deterioration.

In [51, 62–64], genetic algorithms have also been adopted in the design of FIR filters with discrete coefficients. Genetic algorithms are developed to emulate the evolution process. Specifically, the algorithms usually start with a pool, containing a lot of initial filter coefficient sets, each representing a filter design. Each coefficient set has a *health score* that represents the fitness of the individual. Coefficient sets in the pool have different probabilities to reproduce new coefficient sets (*offspring*) with others. This operation is called *crossover*. Usually, coefficient sets with high health scores have larger opportunity to crossover than those with low health scores.

The reproduced coefficient sets (offsprings) have some features from their parents and might go through an evolution process called *mutation*. After the reproduction, the parents and offsprings are put back into the pool and the GA cycle is repeated until a certain termination criterion is met.

After the discovery that the partial CSD or SPT terms (or subexpressions) of filter coefficients can be shared in realizing the multiplier block, filter coefficients are no longer synthesized separately. As a result, MCM techniques have evolved as the most efficient ones in realizing filter coefficients. Therefore, many efforts have been made to reduce the number of SA and MB adders within the MCM context [23–28, 30, 33, 37, 38, 41, 43–45, 54, 56–58, 70–72]. Although most MCM algorithms are considered in transposed direct form, it should be noted that the direct form may synthesize the filter in the same complexity. The proof can be found in [73, 74].

The MCM techniques can be classified into two categories.

2.3.2 MCM Techniques in the First Category

In the first category, MCM algorithms are applied on an FIR design with a given set of discrete coefficients. The common subexpression patterns are extracted and shared among the discrete filter coefficients. Among them, there are three types that are most commonly used.

The first type of algorithms is *common subexpression sharing* [24–27, 30, 43, 54, 70, 71] in which the common patterns of the coefficients, usually represented in a CSD form, or binary numbers are found and shared. For instance, in Figure 2.8 the input data is multiplied by three filter coefficients: 3, 6 and 12 and the products are fed into the delay line. When represented in CSD format, these coefficients are: $10\bar{1}$, $10\bar{1}0$ and $10\bar{1}00$, respectively, where $\bar{1}$ represents -1 . It can be seen that they share the same pattern $10\bar{1}$ as circled in Figure 2.8(a). The subexpression of $10\bar{1}$ is calculated three times as shown in Figure 2.8(b). Therefore, by extracting the common pattern $10\bar{1}$, the multiplication of $x(n)$ with the three coefficients can

$$h(2) = \overline{101}, \quad h(1) = \overline{101}0, \quad h(0) = \overline{101}00$$

(a)

(b)

(c)

Figure 2.8: (a) Filter coefficients expressed in CSD format (b) Without common subexpression sharing (c) With common subexpression sharing

be realized in the way as indicated in Figure 2.8(c). As a result, the number of total additions required using common subexpression sharing is reduced by 2 compared to the direct synthesizing scheme in which the common pattern $\overline{101}$ is not shared among all the multiplications. Basically, the common subexpression sharing technique is representation dependent, where a same set of coefficient represent in different formats may use different number of adders.

Figure 2.9: Coefficient set $\{1, 7, 16, 21, 33\}$ synthesized using adder graph.

The second category of algorithms are based on the *adder graph* [23, 33] in which larger coefficients are realized by shifting and adding those already realized smaller coefficients. An example of synthesizing 5 filter coefficient values of 1, 7, 16, 21 and 33 is shown in Figure. 2.9. In the adder graph, each vertex represents an adder (except the left most one) while each edge represents a shift (which is assumed to be cost free). So the number of adders used to synthesize filter coefficients is determined by the number of vertices in the graph. As noted from Figure 2.9, the synthesis process starts from the left where the first vertex is always 1. After the coefficient value 7 is realized using one adder ($7 = 1 \ll 3 + \bar{1}$), the number 21 is further realized using the already synthesized 7 ($21 = 7 + 7 \ll 1$). It is noted that in the adder graph, the synthesis of the coefficient values are representation independent [31, 32].

The third category is the *difference method* [28, 29, 37] of which the cost to realize the differences of the coefficients are minimized to reduce the overall complexity. As is shown in Figure 2.10, the filter coefficients $h(n)$ to be synthesized are 3, 5, 17, 17 and 29 respectively. The differences of adjacent coefficients are first calculated. For instance, $\delta_{34} = h(3) - h(4) = 5 - 3 = 2$ while $\delta_{23} = h(2) - h(3) = 17 - 5 = 12$, where

Figure 2.10: Multiplier block synthesis using difference method.

δ_{ij} is the difference of coefficient $h(i)$ and $h(j)$. Therefore, in the implementation of the coefficients, $h(i) = h(j) + \delta_{ij}$ for $i = 0, 1, 2, \dots, N - 2$. The first coefficient $h(N - 1)$ ($h(4)$ in the case of Figure 2.10) is realized directly. As a result, 4 adders are required using the difference method which is 2 less than directly realizing these coefficients in CSD format.

Matrix MCM

Recent studies have extended the concept of the MCM problem to a matrix MCM scenario, as reported in [68]. The derivation of matrix MCM problem is based on the hybrid form FIR filters [68].

Recall the polyphase decomposition in (2.6), which is equivalent to the form given as

$$H(z) = \sum_{i=0}^{L-1} \left[\sum_{j=0}^{M-1} h(Mi + j) z^{-j} \right] z^{-Mi}. \quad (2.7)$$

Now, denote $\mathbf{h}_i = [h(Mi) \ h(Mi+1) \ \dots \ h(Mi+M-1)]^T$ and $\mathbf{z}_i = [1 \ z^{-1} \ \dots \ z^{-M+1}]^T$.

(2.7) can be written as

$$H(z) = \sum_{i=0}^{L-1} \left[\mathbf{h}_i^T \mathbf{z}_l \right] z^{-Mi}. \quad (2.8)$$

Similarly, denote $\mathbf{z}_r = [1 \ z^{-M} \ \dots \ z^{-(L-1)M}]^T$. (2.8) can be further explicitly expressed as

$$H(z) = \mathbf{z}_l^T \begin{bmatrix} \mathbf{h}_0 & \mathbf{h}_1 & \dots & \mathbf{h}_{L-1} \end{bmatrix} \mathbf{z}_r. \quad (2.9)$$

Obviously, (2.9) shows that the overall transfer function $H(z)$ can be computed as a matrix multiplication with the delayed inputs. And the constant matrix is of the form

$$\begin{bmatrix} h_0 & h_M & \dots & h_{N+1-M} \\ h_1 & h_{M+1} & \dots & h_{N+2-M} \\ \vdots & \vdots & \ddots & \vdots \\ h_{M-1} & h_{2M-1} & \dots & h_N \end{bmatrix}$$

Figure 2.11: Hybrid form of FIR filter with constant matrix multiplication block.

Shown in Figure 2.11 is the hybrid form FIR structure with constant matrix multiplication block. It should be noted that, for $M = 1$, this structure becomes the transposed direct form in Figure 2.3(a) whereas for $M = N + 1$, the structure

becomes the direct form in Figure 2.3(b). Therefore, the transposed direct form and direct form realizations of FIR filter are just two special cases of the general hybrid form in Figure 2.11. Several techniques have been proposed to synthesize the matrix MCM block as shown in Figure 2.11 [37, 67]. In Chapter 3, a concrete application of this idea in synthesizing and optimizing linear phase FIR filters with reduced complexity compared with the transposed direct form or direct form realizations will be presented.

2.3.3 MCM Techniques in the Second Category

In the second category, many designers have returned to the filter design process, trying to incorporate the MCM techniques into the searching process for discrete solutions [38, 56–60, 75]. In this section, two most representative methods, i.e., linear programming method [38, 57, 59] and subexpression space method [58, 60] are reviewed.

Linear Programming Method

In [38, 57, 59], the design of linear phase FIR filters is formulated as a linear programming problem. Simple branch and bound (B&B) search of different combinations of coefficients or more complicated mixed integer linear programming (MILP) were adopted.

Specifically, in [57], the optimization problem is addressed in two step. In the first step, the lower bound and upper bound of each coefficient with infinite precision is found. Recall that the zero-phase frequency response of a linear phase FIR filter can be written as

$$H(\omega) = \sum_{n=0}^K h(n) \text{Trig}(\omega, n), \quad (2.10)$$

where the Trigonometric function could be *cos* or *sin*, depending on order N and symmetry of the filter. There are $K + 1$ independent coefficients to be optimized

while $K = N/2$ if N is even, or $K = (N + 1)/2$ if N is odd. Furthermore, the constraints in the formulation of linear programming can be written as

$$\begin{aligned} 1 - \delta_p &\leq H(\omega)/b \leq 1 + \delta_p, \text{ for } \omega \in [0, \omega_p], \\ -\delta_s &\leq H(\omega)/b \leq \delta_s, \text{ for } \omega \in [\omega_s, \pi], \end{aligned} \quad (2.11)$$

where δ_p , δ_s , ω_p and ω_s are the passband ripple, stopband ripple, passband edge and stopband edge, respectively and they are given in the filter specifications. b is the passband gain (not restricted to 1) which is allowed to float within certain range, for the purpose of better results. The above constraints are for the case of lowpass filter, for other types of filters, the constraints can be constructed in a similar way.

After building up the constraints, according to [57], the lower bound and upper bound of all the coefficients (except $h(0)$) in infinite precision are searched using linear programming. By using the constraints given in (2.11) where $h(0)$ is set to be 1, the objective function f for searching the bounds of each coefficient $h(n)$ for $n = 1, 2, \dots, K$ is set to $f = h(n)$ and $f = -h(n)$ for the lower bound and upper bound, respectively. Collectively, the total number of linear programming problems is $2K$.

In the second step, every coefficient is quantized to finite precision using a fixed number of SPT terms that is prescribed. The quantized value has to be within the lower bound and upper bound obtained in the first step. Then an exhaustive search is performed with different combinations of all the quantized coefficients to check which combinations satisfy the filter specifications in the frequency domain. In order to obtain better results, bounds for the coefficients are further scaled with different factors, i.e.,

$$\begin{aligned} \hat{h}_l(n) &= \alpha h_l(n), \text{ for } n = 0, 1, \dots, K, \\ \hat{h}_u(n) &= \alpha h_u(n), \text{ for } n = 0, 1, \dots, K, \end{aligned} \quad (2.12)$$

where $h_l(n)$ and $h_u(n)$ are the lower bound and upper bound of coefficient n obtained in the first step. $\hat{h}_l(n)$ and $\hat{h}_u(n)$ are the scaled versions. The scaling factor α is chosen from a look-up table while $\alpha \in [1/3, 2/3]$.

When the search for feasible coefficients sets (which satisfy the design criteria) is over, a common subexpression reduction algorithm is used on these feasible sets to further reduce the number of adders and the one that achieves minimum number of adders is the optimum solution.

Different from the algorithm proposed in [57], in [38] subexpression sharing is incorporated in the linear programming problem formulation directly. The major improvement is that the objective function is the number of adders that are used to realize the whole filter given by

$$\text{minimize : } f = (\sum e_q) + (\sum a_{n,s,t,q} - N) + (2 \sum b_n - 1), \quad (2.13)$$

where the first term in (2.13) represents the number of adders used to realize the required subexpressions and the second term is the number of adders used to synthesize all the filter coefficients using these subexpressions and the third term is the number of structural adders. The constraints of the problem are the same as those given in (2.11). For instance, suppose that the subexpressions are (expressed in decimal numbers) 3 and 5. This means that the first term in (2.13) is equal to 2. Furthermore, we have three coefficients 6, 20, 29. Obviously, realizing these three coefficients (based on the subexpressions) would require 1 adder, which means that the second term in (2.13) is 1. Finally, for a linear phase filter with length of 6, the third term is equal to 5. Altogether, it requires 8 adders to realize the whole filter (3 multiplier block adders plus 5 structural adders).

It should be noted that the variables (e_q , $a_{n,s,t,q}$ and b_n) are binary numbers (0/1). Therefore, the number of 0/1 variables to be optimized is really large even for a small filter with short wordlength.

More recently, in [59], the optimization is also formulated as a linear programming problem except that the passband gain is kept to unity. Because the passband gain is not treated as a variable in the optimization (as shown in [38, 57]), the number of feasible solutions is significantly reduced.

Different from the algorithm in [57] which computes the lower bound and upper bound of all the coefficients in the first step, the algorithm in [59] finds the boundaries of a certain coefficient during the MILP process, where coefficients are fixed one by one. This narrows down the searching space because as more and more coefficients are fixed, the boundaries of the rest unfixed ones become tighter. In addition, [59] introduces an SPT terms prediction mechanism which helps to cut off unnecessary branches during the (B&B) searching in MILP. A subexpression elimination algorithm is applied on the best solution (which uses the least SPT terms) ever found to compute the final number of adders used to synthesize the filter.

Subexpression Space Method

In [58], a B&B MILP is proposed to optimize the filter coefficients in subexpression space. We shall first review the concept of subexpression space.

A subexpression space is constructed based on a subexpression basis set [58]. The basis set consists of 0 and odd integers and the order of the basis set is defined as the number of adders needed to generate the values of all the elements in the set. For example, the order of the basis set $\{0, \pm 1, \pm 3, \pm 5, \pm 7\}$ is 3 because there are three adder needed to construct 3, 5 and 7 altogether.

A subexpression space is then constructed on a subexpression basis set by defining its element as:

$$n = \sum_{i=0}^{K-1} y(i)2^{q(i)}, y(i) \in S, \quad (2.14)$$

where S is a predefined subexpression basis set and $q(i)$ is an integer. K is the number of allowed subexpression terms. From the equation, it is obvious that for a

given K and a given subexpression basis set, in general, the constructed subexpression space is a subset of the integer space with the same wordlength.

In optimizing the filter coefficients using B&B MILP in subexpression space [58], the subexpression basis set is first defined and the number of subexpression terms allocated to each coefficient is also specified. The normalized peak ripple magnitude (NPRM) [47] is minimized subject to the filter specification such as band ripple ratios and band edges.

During the branch and bound optimization, filter coefficients are selected for branching and discrete values are assigned to each coefficient. The subexpression space for each coefficient is different because each coefficient is assigned different number of adders, i.e., the more adders are assigned, the larger its space becomes.

The advantages of the introduction of subexpression space are mainly in the following. First, because most algorithms design filter coefficients in SPT terms, it is unclear how many adders will be used until a subexpression elimination algorithm is applied on the final result. Furthermore, the search space is limited by the SPT coefficients obtained in the optimization. However, using subexpression space, the number of adders used can be directly determined before the optimization of filter coefficients. Moreover, the number of adders used in every coefficient are precisely controlled.

Second, instead of searching the entire integer space, the algorithm searches for feasible solutions within the specified subexpression space. The reduced searching space thus results in significant time saving compared with the algorithms in [38, 57].

However, since the subexpression basis set is pre-defined in [58], there is no guarantee that this basis set is the optimum one which results in minimum number of adders. In order to overcome this issue, in [60], the concept of subexpression space is further extended to the dynamically expanding subexpression space, in which the subexpression basis set is initialized to contain only 0 and 1. During the searching process of branch and bound MILP, each node is associated with a

subexpression basis set, and the subexpression basis sets are updated dynamically. When the traverse searches forward along a path with more and more coefficients being quantized to integers, the subexpression basis set expands, while the search is traced back to a coefficient, the original basis set of that coefficient is recovered. As a result, the algorithm based on dynamically expanding subexpression basis set can produce even better results.

By now, we have reviewed several representative methods for the design of linear phase FIR filters with discrete coefficients, i.e., MCM techniques in the second category. However, none of the above algorithms are capable of producing the optimum solution for filters with practical lengths. By optimum solution, we mean the solution that requires minimum number of adders to implement in transposed direct form or direct form. This problem is answered in Chapter 4 where an algorithm capable of finding the optimum solution is proposed.

2.4 Power Consumption

From the aforementioned discussion, it can be summarized that the complexity of an FIR filter is more or less proportional to the number of adders that are used to construct the circuit. However, the power consumption of an FIR filter, on the other hand, is not entirely determined by the number of adders as it seems to be [76, 77]. In order to accurately estimate the power consumption, investigation down to the gate level of the circuit shall be carried out, i.e., the transition activities occurring in the adders in the transistor level must be examined. In digital CMOS circuits, the main source of power consumption comes from the transitions at the circuit nodes, i.e., the dynamic power which is given by

$$P_{dynamic} = \alpha C_L V_{DD}^2 f_{clk}, \quad (2.15)$$

where C_L is the load capacitance, V_{DD} is the source voltage, f_{clk} is the clock frequency and α is the transition activity factor, i.e., the average number of transitions at the node per clock. Apparently, the dynamic power of the circuit is proportional to α (given other factors fixed) and one of the major contributions of α in the circumstance of FIR filters is called glitch.

Glitch

Figure 2.12: Diagram of a B -bit (a) Carry Ripple Adder, (b) Carry Save Adder.

Glitch is an unwanted transition at a node and it occurs because a circuit has different arrival times of signals (which are supposed to arrival at the same time ideally). This asynchronous phenomenon arises due to the factor that physical circuit components actually take time to function or to process data. For instance, it takes a certain amount time to charge or discharge the load capacitor of an inverter. In FIR filters, glitches usually appear within one adder or among adders. Let's take the carry ripple adder (CRA) shown in Figure 2.12(a) as an example. A B -bit CRA is a cascade of B blocks of single-bit CRA. A single bit CRA has three inputs: x , y and c_{in} and two outputs: s and c_{out} . The input c_{in} is usually connected with the carry output of the previous bit while the output c_{out} is passed onto the carry input of the next bit. In this way, the carry propagates from the least significant bit (LSB) to the most significant bit (MSB). Therefore, the outputs of the n th-bit CRA are wrong and unwanted (glitches) until the carry input to the n th-bit CRA has been correctly generated.

Table 2.1: Outputs of adding two numbers using a 4-bit CRA. $x = 1111$, $y = 0001$

Time	S_0	S_1	S_2	S_3	C_0	C_1	C_2	C_3
Δ	0	1	1	1	1	0	0	0
2Δ	0	0	1	1	1	1	0	0
3Δ	0	0	0	1	1	1	1	0
4Δ	0	0	0	0	1	1	1	1

For instance, suppose the inputs of a 4-bit CRA are: 0001 and 1111, respectively. And the time for the carry to propagate through a single bit CRA is Δ and at time $T = 0$ the two inputs are imposed upon the CRA. Table 2.1 lists all the outputs of the CRA from $T = \Delta$ to $T = 4\Delta$. It can be seen that at Δ , 2Δ and 3Δ the carry input of the 4-th bit CRA has not been correctly generated. However, at these instances there are transitions at the outputs which are obviously unwanted. Only after 4Δ do we obtain the final result. Hence, instead of the ideal case where there is only one output, in practice, the particular CRA causes 3 more glitches. In addition, the more bits the CRA has, the longer it takes for the carry to propagate and thus the more glitches. It may be inferred from the above discussion that shorter coefficient wordlength results in higher circuit speed and also less power consumption. However, this is not always the case when the whole adder network is taken into consideration. This is mainly because that using longer wordlength could result in an implementation of lower adder depth (to be discussed next) and smaller number of adders.

In [76], a term called Glitch Path (GP) count has been adopted as a measure of power consumption. The GP count of a node, according to [76] is given by

$$GP_{out}^i = GP_{in_1}^i + GP_{in_2}^i + 1, \quad (2.16)$$

where $GP_{in_1}^i$ and $GP_{in_2}^i$ are the GP counts at the two inputs of adder i . And the total GP count of the whole adder network is the sum of GP count at each node:

$$GP_{total} = \sum_i GP_{out}^i. \quad (2.17)$$

Adder Depth

In order to minimize the GP count of the whole circuit, one might consider the use of carry save adder (CSA) shown in Figure 2.12(b). Similar to CRA, a single-bit CSA has three inputs x , y , z and two outputs s and c , except that the carry out c is not connected to the next bit. The usage of CSA can be found widely in the implementation of direct form FIR filter where delayed versions of signal are added in a big MCM block (which contains both multiplier block adders and structural adders) and other applications [78]. For a stand alone B -bit CSA, there is no glitch at the output since the GP counts at its two inputs are 0. Nevertheless, glitches still exist in between several CSA adders.

Figure 2.13: An adder graph with adder depth=2.

For example, consider the adder connection shown in Figure 2.13, where the adders are assumed to be ideal, i.e., it takes Δ for an adder to compute the sum and only at $T = \Delta$ the output will transit. It can be seen that $v_2(n) = 3v_1(n)$ and $v_3(n) = 8v_1(n) + v_2(n) = 11v_1(n)$. However, the output of the second adder ($v_3(n)$) exhibits transitional values (i.e., glitches) before the expected value $11v_1(n)$ is reached. This can be explained as follows. Suppose that at $T = 0$ $v_1(0)$ is imposed. It should be noted that at this time the output of the first adder has not changed and

remains $v_2(0) = 3v_1(-1)$, where $v_1(-1)$ is the signal in the previous clock. Therefore, at $T = \Delta$, $v_2(\Delta)$ and $v_3(\Delta)$ become $v_2(\Delta) = 3v_1(\Delta)$ while $v_3(\Delta) = 8v_1(\Delta) + 3v_1(-1)$. Clearly, at this instant a glitch is caused at $v_3(\Delta)$. Only after $T = 2\Delta$ do we have the right output. Furthermore, since it takes at least 2Δ for this particular circuit to compute the final result, the clock period must be longer than 2Δ , i.e., $f_{clk} \leq 1/(2\Delta)$ (assume that there are registers right before v_1 and after v_3 ; and the setup time for registers are taken into consideration).

From the above example, we can see that the power consumption is very much related to the adder depth while adder depth is defined as the number of adders that the input signal goes through before reaching the delay element. In Figure 2.13, the adder depth is equal to 2. It is shown in [76] that larger adder depth results in larger GP count and hence more transition activities and power. It is also noted that the throughput of the circuit is decided by the maximum adder depth (MAD), whereas the power consumption is more related to the average adder depth. The net effect of all the filter coefficients, realized with different adder depths, may also affect the power consumption. Therefore, when it comes to the circuit speed and power consumption, low adder depth implementation is usually desirable.

One of the practical implementation techniques in achieving low adder depth is to pipeline the adder network as shown in Figure 2.14. It can be seen that the multiplier block is implemented using pipelines for transposed direct form FIR filters. The extra registers are employed to minimize the adder depth while the number of MB adders can be reduced at the same time using algorithms [26, 28, 33, 44, 54, 72] reviewed in Section 2.3.2. In this particular example, the MAD is equal to 1 so that the circuit can operate with highest throughput. However, if CRA is used to realize the adders, glitches within each CRA can still occur. The direct form implementation using pipeline can be obtained in a similar manner. However, the disadvantages of pipelining are magnified by the facts that the implementation complexity is raised because of the usage of extra registers; and also, the power

consumption of the total registers is increased, which might negate the savings achieved with reduced adder depth.

Figure 2.14: Implementation of multiplier block using pipelines.

Besides the pipeline technique, some studies concerning low adder depth implementation [36, 41, 79, 80] have been carried out. The most recent study is reported in [77] where the transition activity within a single CRA is accurately characterized. However, for more complicated adder network, no concrete progress has been made yet. Generally speaking, tradeoff between the number of MB adders and adder depth, i.e., the tradeoff between implementation complexity and power consumption, has to be made. Although the direct form implementation can be obtained by transposing the transposed direct form using the same number of adders, compared with the transposed direct form, the direct form implementation has larger adder depth, resulting in more power consumption and lower speed (if no pipeline is used).

The actual transition activity and power consumption of FIR filters are much more complicated because it not only involves accurate modeling of the network of adders but also the driving capability of different types of adders that are used. The transition activities at each node, as shown above, are asynchronous with respect

to the clock. On the other hand, they very much depend on the physical properties of the specific components. Therefore, in order to completely study this problem, investigations down to the transistor level should be carried out, and this area is still open for further research.

Nevertheless, based on the above review and analysis, low power of digital filters could be achieved by designs of low implementation complexity **together with** low maximum and average adder depths. Therefore, throughout this thesis, reduced complexity (number of adders) as well as reduced adder depth (maximum and average adder depth) of digital filters are the main focus to achieve the low power and low complexity digital filter design and implementation.

Chapter 3

Residual Compensated Extrapolated Impulse Response Filters

As reviewed in Section 2.3.2, MCM algorithms in the first category are applied on an FIR design with a given set of discrete coefficients. In addition, these algorithms can be further classified into three types. The first type is common subexpression sharing [26,30,54] in which the common digit patterns of the coefficients, usually represented in a CSD form, or binary numbers are found and shared. The second type of algorithms are based on the adder graph [23,33] in which larger coefficients are realized by shifting and adding those already realized smaller coefficients. The third type is the difference method [28,37] of which the differences of the coefficient values, instead of the coefficient values, are synthesized to reduce the overall complexity.

MCM algorithms in the first category, however, suffer from the fact that the searching space is limited by the given discrete coefficient values. Furthermore, it has been proved in [45] that for a filter with L distinct positive non-one coefficient values (after the even and negative coefficient values are transformed to positive odd numbers by scaling the coefficient values with a proper signed power-of-two factor), the lower bound of the number of MB adders of the transposed direct form implementation is equal to the minimum number of adders required to realize the simplest coefficient plus $(L - 1)$. Since there are always very small magnitude coefficients which can be realized by one adder in a practical filter, in most cases,

the number of adders required to realize the simplest coefficient is one. Therefore, the lower bound is basically determined by the number of distinct coefficient values, L . For many benchmark filters, this lower bound has been achieved, for instance, example 1 of reference [1]. The example is a 120th-order filter with 52 distinct coefficient values. Many algorithms [33, 41, 44, 72] have achieved this lower bound. To further reduce the number of MB adders, the only way is to reduce the lower bound by reducing the number of distinct coefficient values.

In order to reduce the filter complexity by further reducing the lower bound of the number of adders, a residual compensated extrapolated impulse response filter structure is presented in this chapter. Using the proposed technique, the dynamic range of the coefficient multipliers is further reduced, which increases the chance of having coefficients with identical values. This results in a reduction in the lower bound of MB adders. In addition, the design of the proposed structure is formulated as a MILP problem where the filter coefficients and residuals are optimized simultaneously.

This chapter is organized as follows. Section 3.1 reviews the extrapolated impulse response filters. The residual compensated extrapolated impulse response filter structure [81] is introduced in Section 3.2. A synthesis example is used to illustrate the proposed technique. The optimization of the coefficient values and residuals in the subexpression space is formulated as a MILP problem in Section 3.3. In Section 3.4 benchmark filters are designed to illustrate the efficiency of the proposed technique. Section 3.5 gives a circuit realization of the extra delay chains. In Section 3.6, the implementation complexity and adder depth of the proposed technique are discussed and compared with the conventional structures.

Figure 3.1: A typical impulse response of a $2N$ -th order symmetric FIR filter. For simplicity, the impulse response is centered at $n = 0$

3.1 The Extrapolated Impulse Response

The impulse response of a typical linear phase FIR filter is quasi-periodic [82–84] as shown in Figure 3.1. Most of the energy of the impulse response is concentrated at the center lobe, whereas the side lobes have decreasing magnitudes. If **lobe 0**, **lobe 1** and **lobe 2** (see Figure 3.1) have the same number of samples, **lobe 1** and **lobe 2** can be approximated as scaled versions of **lobe 0**. **lobe 0** is thus served as a prototype lobe. Once the coefficient multipliers in **lobe 0** is implemented, the coefficient multipliers in **lobe 1** and **lobe 2** do not need to be implemented individually; instead, they could be obtained using a single scalar for each lobe and the implementation complexity of the filters are reduced. **lobe 1** and **lobe 2** are thus realized as the extrapolations of **lobe 0** and this is the origin of the extrapolated impulse response filters [82]. While in the original extrapolated impulse response filters, the lobe closest to the center lobe is usually selected as the prototype filter. However, it is not necessary the case. Any lobe could be the prototype lobe and the other lobes are approximated as scaled versions of the prototype lobe. In this case, it is no longer an extrapolation, but the name of extrapolated filter is still used.

Figure 3.2: Structure of extrapolated impulse response.

For expository convenience, we suppose that the impulse response is centered at time 0. Thus, the zero phase z -transform transfer function of a $2N$ -th order filter is given by:

$$H(z) = h(0) + \sum_{n=1}^N h(n)(z^n + z^{-n}). \quad (3.1)$$

Assume that $lobe_l$, $l = 0, 1, \dots, L-1$, begins at $n = p_l$ through $n = q_l$ for an L lobe filter, where $q_{L-1} \leq N$, $H(z)$ can be rewritten as:

$$\begin{aligned} H(z) &= h(0) + \sum_{n=1}^M h(n)(z^n + z^{-n}) \\ &+ \sum_{n=p_0}^{q_0} h(n)(z^n + z^{-n}) + \sum_{n=p_1}^{q_1} h(n)(z^n + z^{-n}) + \dots \\ &+ \sum_{n=p_{L-1}}^{q_{L-1}} h(n)(z^n + z^{-n}) + \sum_{n=q_{L-1}+1}^N h(n)(z^n + z^{-n}). \end{aligned} \quad (3.2)$$

The durations of these lobes are $q_l - p_l + 1$ for $l = 0, 1, \dots, L - 1$, and they might not be all equal. However, they can be separated into several groups, and each group consists of lobes with the same duration. Thus, in each group, any lobe can always be approximated as the scaled version of a prototype lobe in that group. For expository convenience, we assume that all the lobes have the same duration d since other cases are just simple extensions. As a result, (3.2) can be written as:

$$\begin{aligned} H(z) &= h(0) + \sum_{n=1}^M h(n)(z^n + z^{-n}) \\ &+ \sum_{l=0}^{L-1} \sum_{m=1}^d h(M + m + ld)(z^{M+m+ld} + z^{-(M+m+ld)}) \\ &+ \sum_{n=M+Ld+1}^N h(n)(z^n + z^{-n}), \end{aligned} \quad (3.3)$$

where d is the duration of each lobe. If the lobe with the smallest magnitude (usually, the $(L - 1)$ -th lobe while $lobe_0$ is considered as the zero-th lobe) is chosen as the prototype lobe, $H(z)$ can be approximated by

$$\begin{aligned} H(z) &\approx \hat{H}(z) = h(0) + \sum_{n=1}^M h(n)(z^n + z^{-n}) \\ &+ \sum_{m=1}^d h(M + m + (L - 1)d) \sum_{l=0}^{L-1} \alpha_l (z^{M+m+ld} \\ &+ z^{-(M+m+ld)}) + \sum_{n=M+Ld+1}^N h(n)(z^n + z^{-n}), \end{aligned} \quad (3.4)$$

where α_l is the l -th scaling factor and $\alpha_{L-1} = 1$.

A realization of a 12th-order ($N = 6$) linear phase FIR filter using the above extrapolation technique is given in Figure 3.2, where $M = 0$, $L = 2$ and $d = 3$. Coefficients $h(1)$ to $h(3)$ are implemented as scaled version of $h(4)$ to $h(6)$ with a scaling factor α_0 .

Optimization techniques have been proposed in [82–84] to optimize the filter coefficients of the prototype lobe as well as the scaling factors in continuous spaces.

The computational complexity, in terms of the number of multiplication of the resulting extrapolated filter is much reduced when compared with the direct form implementation. The price paid for that is that the order of the extrapolated filter may be slightly higher than that of the minimax optimum [82, 83]. The frequency response of the extrapolated filter is generally degraded if the same order as that of the minimax optimum filter is adopted.

3.2 Extrapolated Impulse Response with Residual Compensation

The degradation in the frequency response in the traditional extrapolated filters is due to the fact that the complexity reduction is achieved by reducing the degree of freedom of filter coefficients. Coefficients are only approximations of their optimum values. To meet a given specification, the order of the extrapolated filter is, in general, higher than that of the minimax optimum. In order to overcome this problem, an extrapolated impulse response with residual compensation is proposed in this section.

3.2.1 Structure

Assume that the optimum impulse response of a $2N$ -th order linear phase FIR filter for a given specification in a discrete space is $h(n)$ for $-N \leq n \leq N$. The discrete space, denoted as $\mathcal{D}$, may be the finite-word length space or the signed power-of-two space. Therefore, $h(n) \in \mathcal{D}$, and $h(-n) = h(n)$, for $-N \leq n \leq N$.

Since the coefficients are symmetric, only the coefficients with non-negative index are considered. Assume further that the coefficients are quasi-periodic with duration d from $n = M + 1$ to $n = M + Ld$ for L periods. Thus, $0 \leq M < N$, $L \leq \frac{N-M}{d}$ and both are integers. Following the procedure of the traditional extrapolated filters, the filter coefficients of $h(n)$ for $0 \leq n \leq M$ and $M + Ld + 1 \leq n \leq N$ are implemented

accurately. For $h(n)$ within the range $M + 1 \leq n \leq M + Ld$, if the period with the smallest magnitude is chosen as the prototype lobe, and all the other periods are approximated as the scaled versions of the prototype lobe, the approximated coefficients, denoted as $h_a(n)$, from $n = M + 1$ to $n = M + (L - 1)d$ become

$$\begin{aligned} h_a(n - (L - l - 1)d) &= \alpha_l h(n) \\ \text{for } M + (L - 1)d + 1 &\leq n \leq M + Ld \text{ and} \\ 0 &\leq l \leq L - 2, \end{aligned} \quad (3.5)$$

where the scaling factors, α_l , are integers or power-of-two numbers for multiplierless implementation. Therefore, the coefficient residuals, denoted as $h_r(n)$, due to the approximation are given by

$$h_r(n) = h(n) - h_a(n), \quad \text{for } M + 1 \leq n \leq M + (L - 1)d. \quad (3.6)$$

Thus, the z -transform transfer function $H(z)$ can be precisely represented as:

$$\begin{aligned} H(z) &= h(0) + \sum_{n=1}^M h(n)(z^n + z^{-n}) \\ &+ \sum_{n=M+Ld+1}^N h(n)(z^n + z^{-n}) + \sum_{n=M+1}^{M+(L-1)d} h_r(n)(z^n \\ &+ z^{-n}) + \sum_{n=M+(L-1)d+1}^{M+Ld} \sum_{l=0}^{L-1} \alpha_l h(n)(z^{n-(L-1-l)d} \\ &+ z^{-n+(L-1-l)d}). \end{aligned} \quad (3.7)$$

In the implementation of the extrapolated filter, the coefficient residuals are compensated by adding the products of the signal samples and the residuals into the tap delay line, as shown in Figure 3.3. Thus, the filter impulse response is restored perfectly.

Figure 3.3: Extrapolated impulse response with residual compensation.

Comparing the residual compensated extrapolated filter implementation shown in Figure 3.3 with the transposed direct form implementation shown in Figure 2.3, it can be seen that the numbers of structural adders are the same for both structures, since generally each extrapolated coefficient is compensated by a corresponding residual. The MB adders of the proposed technique can be further classified into the prototype coefficient adders and the residual adders, which are used to realize the prototype coefficients and residuals, respectively. Both the prototype coefficients and residuals can share the same MB using MCM algorithms, since they are multiplied by the same signal, as shown in Figure 3.3. Since the residual is the difference of the optimum coefficient value and the extrapolated approximation, its magnitude is much smaller than the original optimum value. The magnitudes of the prototype coefficients are small, since the lobe with the smallest magnitude of coefficient values is selected as the prototype lobe. Thus, the dynamic range of the

coefficient multiplier is significantly reduced. As a result, the number of distinct odd positive integers is reduced.

It should be noted from Figure 3.2 and Figure 3.3 that, in the extrapolated impulse response structure, besides the structural adders and MB adders, another two types of adders are employed in the residual compensated extrapolated filters. The first type of adder is the extrapolation adders used to add the extrapolated terms into the tap delay line; and the second type is the scaling factor adders used to generate proper scaling factors if they are not power-of-two numbers. To minimize the number of overall adders, the scaling factors are chosen in such a way that as few scaling factor adders as possible are used to reduce the number of distinct residuals as much as possible. If power-of-two numbers are used for the scaling factors, no additional scaling factor adders are required. However, in some cases, using a few scaling factor adders may achieve more saving in the residual adders.

3.2.2 An Example

In spite of the overhead which might be caused by the extrapolation adders and scaling factor adders, the proposed technique still significantly reduce the overall number of adders when compared with any other existing techniques. An example is taken from literature [1] to illustrate the efficiency of the proposed technique.

The discrete filter coefficient values presented in [1] are listed as $h(n)$ in Table 3.1 for easy reference. As we have indicated in the introduction, the number of non-one distinct coefficient values (after they have been transformed to positive odd numbers) of the coefficient set of this filter is 52. Therefore, the lower bound of the MB adders is 52, which have been achieved by many algorithms [33, 41, 44, 72].

By inspecting the coefficient values $h(n)$ in Table 3.1, it is noted that the filter impulse response shows a quasi-periodicity for 5 periods, from $h(4)$ to $h(48)$, with period duration of 9. The period with the minimum coefficient magnitude, i.e., from $h(40)$ to $h(48)$ is chosen as the prototype lobe to approximate the other 4 lobes. The

Table 3.1: Impulse Response of A 120-th order filter. $h(n)$ is the original discrete coefficient value obtained in [1], $h_a(n)$ is the extrapolated approximation of the original coefficient value, and $h_r(n)$ is the resulting residual. Coefficients from $h(40)$ to $h(48)$ are used as the prototype lobe.

n	$h(n)$	$h_a(n)$	$h_r(n)$	n	$h(n)$	$h_a(n)$	$h_r(n)$	n	$h(n)$
0	14686	—	—	21	−600	−552	−48	42	− 130
1	−13494	—	—	22	165	144	21	43	125
2	10267	—	—	23	282	280	2	44	− 68
3	−5927	—	—	24	−541	−520	−21	45	− 8
4	1653	720	933	25	524	500	24	46	68
5	1515	1400	115	26	−277	−272	−5	47	− 89
6	−3000	−2600	−400	27	−59	−32	−27	48	69
7	2823	2500	323	28	323	272	51	49	−24
8	−1522	−1360	−162	29	−408	−356	−52	50	−24
9	−109	−160	51	30	302	276	26	51	55
10	1336	1360	−24	31	−78	−72	−6	52	−61
11	−1739	−1780	41	32	−148	−140	−8	53	45
12	1312	1380	−68	33	277	260	17	54	−18
13	−390	−288	−102	34	−267	−250	−17	55	−8
14	−546	−560	14	35	143	136	7	56	24
15	1090	1040	50	36	23	16	7	57	−28
16	−1071	−1000	−71	37	−152	−136	−16	58	23
17	585	544	41	38	193	178	15	59	−14
18	84	64	20	39	−142	−138	−4	60	6
19	−616	−544	−72	40	36	—	—		
20	798	712	86	41	70	—	—		

scaling factors for lobes $h(4)$ to $h(12)$, $h(13)$ to $h(21)$, $h(22)$ to $h(30)$ and $h(31)$ to $h(39)$ are chosen to be 20, −8, 4, and −2, respectively. Therefore, the approximated coefficient values are given by:

$$h_a(4+k) = 20h(40+k), \quad h_a(13+k) = -8h(40+k),$$

$$h_a(22+k) = 4h(40+k), \quad h_a(31+k) = -2h(40+k),$$

$$\text{for } k = 0, 1, \dots, 8. \quad (3.8)$$

Table 3.2: Number of Adders Used to Synthesize the Filter Coefficients

	Best result obtained in literature [33, 41, 44, 72]	Proposed extrapolated structure
Adders in MB	52	33
Extrapolation Adders	NA	8
Scaling Factor Adders	NA	2
Total Adders	52	43

Thus, the residuals are:

$$\begin{aligned}
h_r(4+k) &= h(4+k) - 20h(40+k), \\
h_r(13+k) &= h(13+k) + 8h(40+k), \\
h_r(22+k) &= h(22+k) - 4h(40+k), \\
h_r(31+k) &= h(31+k) + 2h(40+k), \\
&\text{for } k = 0, 1, \dots, 12.
\end{aligned} \tag{3.9}$$

Both $h_a(n)$ and $h_r(n)$ for $4 \leq n \leq 39$ are listed in Table 3.1.

It can be seen from Table I that the residual coefficient values from $h_r(4)$ to $h_r(39)$ are much smaller than their original values. To implement the filter in residual compensated extrapolated form, the coefficient values to be synthesized are $h(0)$ to $h(3)$, $h_r(4)$ to $h_r(39)$ and $h(40)$ to $h(60)$. Among these coefficient values, the number of distinct non-one coefficient values (after they have been transformed to positive odd numbers) have been reduced to 31. Using RAG- n algorithm [33] to generate the subexpression coefficients, the resulting MB requires 33 adders, which is close to the new lower bound of 31. The reason that 2 more adders are used than the lower bound is that some coefficient values of the center lobe are realized directly, and their magnitudes are large, among which $h(0)$ and $h(2)$, each requires two adders to synthesize.

Figure 3.4: Extrapolated impulse response with residual compensation without including tail coefficients.

Symmetric coefficients of linear phase FIR filter can share the MB. However, the extrapolation adders and scaling factor adders for symmetric lobes cannot be shared with each other as shown in Figure 3.3. Therefore, besides the MB adders, the extrapolated realization requires additional 8 adders to add the extrapolated lobes into the tap delay line, and additional 2 adders to generate the scaling factor 20. Thus, in total 43 adders are required to synthesize the filter, as shown in Table 3.2. The best results obtained by synthesizing the same set of discrete coefficients in published literature require 52 adders [33, 41, 44, 72]; the result is also listed in Table 3.2 for comparison.

In the synthesis of the extrapolated impulse response, it is noted that extra delay chains may be used. As indicated by the dashed circles in Figure 3.3, a delay chain of length d , $d = 3$ in this example, is used when the coefficient symmetry is exploited. Furthermore, if $N > M + Ld$, the tail coefficients, $h(7)$ as shown in Figure 3.4, are not included in any extrapolation lobe, one more delay chain with the same length is required. The delay chain(s) are employed to maintain the delays of tap signals.

The extra delay chains are inherent in the extrapolated impulse response structure. The reason that the extra delay chain was not previously considered in [82–84] is because that the overhead introduced by the delay chain is negligible compared with the saving in multipliers. In our proposed technique, however, the arithmetic complexity is reduced to such an extent that it has become necessary to consider the complexity of the delay chain. Fortunately, each delay chain absorbs one tap delay element in the original filter structure, which compensates for the implementation complexity and power consumption. A realization of the delay chains is given in Section 3.5. A detailed analysis of the power consumed by the delay chain is given in Appendix A. The implementation and power consumption overhead due to the delay chains are discussed in Section 3.5 and Section 3.6, respectively.

3.3 Optimizing Filter Coefficients and Residuals in Subexpression Space

As illustrated in Section 3.2, the proposed technique can significantly reduce the number of adders in the multiplier block. However, the example shown in Section 3.2 synthesized the extrapolated impulse response filter based on a given set of discrete coefficients, therefore belongs to the first category optimization technique for the subexpression sharing. It has been shown in [58] that the second category optimization techniques, where coefficient values are optimized directly in subexpression space, outperformed significantly over the techniques of the first category. In this section, a linear programming problem of the proposed structure is formulated to optimize the filter coefficients and residuals in subexpression space. The concept of subexpression space has been reviewed in Section 2.3.3.

Based on (3.7), the frequency response of a linear-phase residual compensated extrapolated impulse response FIR filter can be expressed as

$$\begin{aligned}
 H(\omega) = & h(0) + \sum_{n=1}^M h(n) \text{Trig}(\omega, n) \\
 & + \sum_{n=M+Ld+1}^N h(n) \text{Trig}(\omega, n) \\
 & + \sum_{n=M+1}^{M+(L-1)d} h_r(n) \text{Trig}(\omega, n) \\
 & + \sum_{n=M+(L-1)d+1}^{M+Ld} \sum_{l=0}^{L-1} \alpha_l h(n) \text{Trig}(\omega, n - (L-1-l)d).
 \end{aligned} \tag{3.10}$$

In (3.10), $h(n)$ for $n = 0, 1, \dots, M, M+Ld+1, \dots, N$ is the filter coefficient that is synthesized directly; $\alpha_l h(n)$ for $n = M+(L-1)d+1, M+(L-1)d+2, \dots, M+Ld$ and $l = 0, 1, \dots, L-1$ is the extrapolated coefficient with the scaling factor α_l pre-fixed and $\alpha_{L-1} = 1$; and $h_r(n)$ for $n = M+1, M+2, \dots, M+(L-1)d$ is the residual used to improve the precision of the filter coefficients. Therefore, the total number of the variables to be optimized is equal to the number of the original filter coefficients. The optimization is formulated to find the filter coefficient values of $h(n)$ for $n = 0, 1, \dots, M, M+Ld+1, \dots, N$, and residuals $h_r(n)$ for $n = M+1, M+2, \dots, M+(L-1)d$, in given spaces, to minimize the frequency response ripples δ :

$$\begin{aligned}
 & \text{minimize : } \delta \\
 & \text{subject to : } 1 - \delta \leq H(\omega) \leq 1 + \delta, \text{ for } \omega \in [0, \omega_p] \\
 & \quad -(\delta_s \delta) / \delta_p \leq H(\omega) \leq (\delta_s \delta) / \delta_p, \text{ for } \omega \in [\omega_s, \pi]
 \end{aligned} \tag{3.11}$$

where ω_p and ω_s are the passband and stopband edges, δ_p and δ_s are the passband and stopband ripples, and $H(\omega)$ is the frequency response of the filter given in

(3.10). This problem can be efficiently optimized by B&B MILP [47, 53, 58]. It is noted that the residuals are usually much smaller than their original filter coefficient values; during the branch and bound search, it is more likely to produce zero-valued residuals. A zero-valued residual contributes to two adders reduction in the structural adders and therefore is preferred.

The criteria of the selection for a coefficient for branching could be found in [14]. In particular, the coefficients in the prototype lobe have more impact on the frequency response and hence should be branched at early stage in the branch and bound optimization. The bound condition is determined by the subexpression space pre-defined for each coefficient [58]. The computation time in optimizing the coefficient values and residuals of the proposed technique is similar to that of MILP in the optimization of filters in other discrete spaces and/or structures. The computation time increases exponentially with the number of variables [14, 58]. For a given number of variables, the computing time required for different specifications differs widely [14]. The typical computation time for designing a 60-th order filter, for instance, ranges from 20 minutes to 2 hours on a Pentium 4 2.4GHz desktop computer with 3GB RAM.

3.4 Design Examples

In this section, several benchmark filters from literature are designed to illustrate the superiority of the proposed technique. The detailed design procedures are given for the first two examples in Sections 3.4.1 and 3.4.2, whereas 5 more examples are collectively presented in Section 3.4.3.

3.4.1 Example $L2$

Example $L2$ is taken from the second example of reference [1]. The filter specification is as follows: a lowpass filter with order 62 has passband and stopband edges at

Figure 3.5: Modified extrapolated impulse response structure with the center coefficient extrapolation.

$\omega_p = 0.2\pi$ and $\omega_w = 0.28\pi$, respectively; the passband and stopband ripples are $\delta_p = 0.028$ and $\delta_s = 0.001$, and the effective wordlength (excluding the sign bit) is 11. By inspection of the optimal continuous coefficients, the 26 coefficients from $h(0)$ to $h(25)$ consist of 2 lobes, each comprising of 13 coefficients. Thus, the coefficients from $h(13)$ to $h(25)$ are chosen as the prototype $lobe_1$. $lobe_0$ (from $h(0)$ to $h(12)$) is approximated by the prototype $lobe_1$ with a scaling factor $\alpha_0 = -4$. This can be expressed mathematically as

$$h_a(k) = -4h(13 + k),$$

$$\text{for } k = 0, 1, 2, \dots, 8. \quad (3.12)$$

It is noted that, in this example, the center coefficient is included in a lobe extrapolated from the prototype lobe. To avoid the extrapolation of the center coefficient being added twice in the implementation where coefficient symmetry is exploited, a modified structure, illustrated by a 10-th order example, is shown in

Table 3.3: Impulse Response of Example *L2*. Coefficients from $h(13)$ to $h(25)$ are chosen as the prototype lobe. Residuals from $h_r(0)$ to $h_r(12)$ and coefficients from $h(13)$ to $h(31)$ are realized in the multiplier block. $h(n) = h_r(n) + \alpha_0 h(n + 13)$ for $n = 0, 1, \dots, 12$, where $\alpha_0 = -4$.

Passband Gain = 3982.402151, Impulse Response $\times 4096$							
n	$h(n)$	$h_a(n)$	$h_r(n)$	n	$h(n)$	n	$h(n)$
0	921	0	$17 \times 2^6 - 5 \times 2^5 - 7 \times 2^0$	13	0	28	7×2^0
1	841	176	$5 \times 2^7 + 7 \times 2^2 - 3 \times 2^0$	14	-11×2^2	29	1×2^3
2	626	236	$3 \times 2^7 + 3 \times 2^1$	15	$-1 \times 2^6 + 5 \times 2^0$	30	3×2^1
3	343	176	$5 \times 2^5 + 7 \times 2^0$	16	-11×2^2	31	3×2^0
4	74	52	11×2^1	17	-13×2^0		
5	-112	-68	-11×2^2	18	17×2^0		
6	-184	-128	-7×2^3	19	1×2^5		
7	-152	-112	-5×2^3	20	7×2^2		
8	-62	-44	-9×2^1	21	11×2^0		
9	34	36	-1×2^1	22	-9×2^0		
10	92	92	0	23	$-3 \times 2^3 + 1 \times 2^0$		
11	96	104	-1×2^3	24	-13×2^1		
12	56	80	-3×2^3	25	-5×2^2		
				26	-9×2^0		
				27	1×2^0		

Figure 3.5, where coefficients $h(0)$ to $h(2)$ are extrapolated from $h(3)$ to $h(5)$. In this structure, $\alpha_0 h(3)$ is added into the delay chain only once.

The subexpression space for the design is constructed on a basis set of order 7:

$$SS = \{\pm 1, \pm 3, \pm 5, \pm 7, \pm 9, \pm 11, \pm 13, \pm 17\}. \quad (3.13)$$

The resulting filter coefficients are listed in Table 3.3. It can be seen that residuals $h_r(0)$ and $h_r(1)$ require 2 adders to construct each. However, since they both use the coefficient 167 ($5 \times 2^5 + 7 \times 2^0$) which is already synthesized in $h_r(3)$, only one adder each is used in the synthesis of $h_r(0)$ and $h_r(1)$. Furthermore, it is noted that $h_r(10)$ and $h(13)$ are 0 which contributes to 4 adders reduction in the structural adders. As listed in Table 3.5, both the MB adders and structural adders are reduced by 4 in comparing with the best result obtained in literature [58]. The overall number

of adders reduced is 6 after the 2 extrapolation adders are compensated. It is also noted from Table 3.3 that the effective wordlength of the filter coefficient is 10 which is one bit less than that in [58].

3.4.2 Example $L1$

Example $L1$ is also taken from reference [1]. The filter specification is as follows: a highpass filter with order $N = 120$ has stopband and passband edges at $\omega_s = 0.74\pi$ and $\omega_p = 0.8\pi$, respectively; the stopband and passband ripples are $\delta_s = 0.0001$ and $\delta_p = 0.0057$ while the effective wordlength (excluding the sign bit) is 14.

Again, by inspecting the optimal continuous coefficient values, it is noted that the filter impulse response shows a quasi-periodicity for 5 lobes, from $h(1)$ to $h(45)$, with each lobe consisting of 9 coefficients. The lobe with the minimum coefficient magnitude, i.e., $lobe_4$ (from $h(37)$ to $h(45)$) is chosen as the prototype lobe to approximate the other 4 lobes. The scaling factors for $lobe_3$ ($h(28)$ to $h(36)$), $lobe_2$ ($h(19)$ to $h(27)$), $lobe_1$ ($h(10)$ to $h(18)$) and $lobe_0$ ($h(1)$ to $h(9)$) are chosen to be -2 , 4 , -8 , and 16 , respectively. Expressed mathematically, we have

$$\begin{aligned} h_a(1+k) &= 16h(37+k), \quad h_a(10+k) = -8h(37+k), \\ h_a(19+k) &= 4h(37+k), \quad h_a(28+k) = -2h(37+k), \\ \text{for } k &= 0, 1, \dots, 8. \end{aligned} \tag{3.14}$$

The other coefficient values are synthesized directly. So, the coefficients to be optimized in this problem are $h(n)$ for $n = 0$ and $n = 37, 38, \dots, 60$ and $h_r(n)$ for $n = 1, 2, \dots, 36$.

The multi-step design approach proposed in [58] is employed to accelerate the optimization procedure, as well as to re-use the subexpression bases. The filter coefficients are split into 3 groups:

Group A : $h_r(28)$ to $h_r(36)$ and $h(37)$ to $h(60)$;

Group B : $h_r(8)$ to $h_r(27)$;

Group C : $h(0)$ and $h_r(1)$ to $h_r(7)$.

In the course of the optimization, the residuals and coefficients in Group A are first optimized in a subexpression space constructed by an initial subexpression basis set, while the residuals and coefficients in Group B and Group C are left continuous. Once a set of discrete values are obtained for the coefficients and residuals in Group A, the initial subexpression basis set is expanded as new bases are generated. The residuals in Group B are then optimized in the expanded subexpression space, while the coefficients and residuals in Group A are fixed to the obtained discrete values, and those in Group C are left continuous. The subexpression space is further expanded . At last, the coefficients and residuals in Group C are optimized, while those of Group A and B are fixed to the obtained discrete values.

The initial subexpression basis set SS_1 and the subsequently updated basis sets SS_2 and SS_3 are as follows:

$$SS_1 = \{\pm 1, \pm 3, \pm 5, \pm 7, \pm 9\}$$

$$SS_2 = \{\pm 1, \pm 3, \pm 5, \pm 7, \pm 9, \pm 13, \pm 15, \\ \pm 19, \pm 23, \pm 25, \pm 29, \pm 33, \pm 41, \\ \pm 63, \pm 73, \pm 135, \pm 157, \pm 197\}.$$

$$SS_3 = \{\pm 1, \pm 3, \pm 5, \pm 7, \pm 9, \pm 13, \pm 15, \\ \pm 19, \pm 23, \pm 25, \pm 29, \pm 33, \pm 41, \\ \pm 63, \pm 73, \pm 89, \pm 111, \pm 135, \\ \pm 145, \pm 157, \pm 171, \pm 177, \pm 197\}$$

i.e., the coefficients and residuals of Group A, B and C are optimized based on the subexpression space constructed from SS_1 , SS_2 and SS_3 , respectively.

The resulting filter coefficients and residuals are listed in Table 3.4. It can be seen that the final basis set of SS_3 requires 22 adders, while additional 8 adders are

Table 3.4: Impulse Response of Example $L1$. Coefficients from $h(37)$ to $h(45)$ are chosen as the prototype lobe. Residuals from $h_r(1)$ to $h_r(36)$, coefficients $h(0)$ and $h(37)$ to $h(60)$ are realized in the multiplier block. $h(n-l_d) = h_r(n-l_d) + \alpha_{L-1-l}h(n)$ for $n = 37, 38, \dots, 45$, $l = 1, 2, 3, 4$, $L = 5$ and $d = 9$, where $\alpha_0 = 16$, $\alpha_1 = -8$, $\alpha_2 = 4$ and $\alpha_3 = -2$.

Passband Gain = 65496.94747, Impulse Response $\times 65536$									
n	$h(n)$	$h_a(n)$	$h_r(n)$	n	$h(n)$	$h_a(n)$	$h_r(n)$	n	$h(n)$
0	14672		111×2^7 $+29 \times 2^4$	19	-619	-628	9×2^0	37	-157×2^0
				20	798	788	5×2^1	38	197×2^0
1	-13481	-2512	-43×2^8 $+1 \times 2^5$ $+7 \times 2^0$	21	-599	-576	-23×2^0	39	-9×2^4
				22	163	144	19×2^0	40	9×2^2
2	10256	3152	111×2^6	23	283	292	-9×2^0	41	73×2^0
				24	-540	-540	0	42	-135×2^0
3	-5920	-2304	-111×2^5 $+1 \times 2^6$	25	521	528	-7×2^0	43	33×2^2
				26	-271	-304	33×2^0	44	-19×2^2
4	1651	576	33×2^5 $+19 \times 2^0$	27	-66	0	-33×2^1	45	0
				28	330	314	1×2^4	46	15×2^2
5	1512	1168	9×2^5 $+7 \times 2^3$	30	306	288	9×2^1	48	63×2^0
				31	-81	-72	-9×2^0	49	-19×2^0
6	-2991	-2160	-63×2^4 $+177 \times 2^0$	32	-146	-146	0	50	-7×2^2
				33	275	270	5×2^0	51	29×2^1
7	2810	2112	177×2^2 -5×2^1	34	-264	-264	0	52	-63×2^0
				35	139	152	-13×2^0	53	23×2^1
8	-1506	-1216	-145×2^1	36	28	0	7×2^2	54	-9×2^1
9	-126	0	-63×2^1					55	-1×2^3
10	1352	1256	3×2^5					56	25×2^0
11	-1753	-1576	-177×2^0					57	-29×2^0
12	1323	1152	171×2^0					58	3×2^3
13	-399	-288	-111×2^0					59	-13×2^0
14	-538	-584	23×2^1					60	7×2^0
15	1083	1080	3×2^0						
16	-1064	-1056	-1×2^3						
17	579	608	-29×2^0						
18	89	0	89×2^0						

used to generate all the coefficients in the MB. Besides, 8 extrapolation adders are used. However, it is noted in Table 3.4 that the values of residuals $h_r(24)$, $h_r(32)$, $h_r(34)$ and coefficient $h(45)$ are 0, which contributes to 8 adders reduction in the structural adders. The number of different type adders are listed in Table 3.5. Thus the total number of adders used to construct the filter using the proposed technique is 14 less than obtained in [58] in which this filter is optimized in transposed direct form. It should also be noted that, compared to the result obtained in Section 3.2, where the extrapolated filter is synthesized based on a given set of discrete coefficient values, 13 more adders are reduced by optimizing the filter coefficients and residuals directly in subexpression space.

3.4.3 Five Additional Benchmark Filters

The proposed technique is further applied on 5 design examples taken from literature. The multiplier block adders, structural adders and total number of adders used to implement the filters are compared with those of the best available designs. Filters $S1$, $S2$ and $L3$ are taken from [58]. $Y1$ is the first example of [57]. $G1$ is given as example 1 in [38]. As listed in Table 3.5, our proposed technique synthesizes the most of the filters using less number of adders. It is noted that for filters of order as low as 24, the proposed technique may further reduce the required number of adders. The reduction is achieved due to the fact that many coefficient values and residuals are reduced to 0 because of the extrapolation; zero save the structural adders. However, for extremely low order and short wordlength filter as $G1$, no further reduction is achieved since the number of adders used in the original design is already very small.

In the implementation of the proposed filters, extra delay chains are used. The number of delay chains required and the length of each delay chain for each design are also shown in Table 3.5. The complexity overhead due to the delay chains is implementation dependent. A realization of the delay chain is proposed in the following section, while the net saving in the implementation is discussed in Section 3.6.

3.5 A Realization of the Extra Delay Elements

A realization of the extra delay chain required in the implementation of the proposed extrapolated filter is introduced in this section. Assume that the data width is B -bits. A delay chain of length d may be realized by using 2 pointers and $d + 1$ storage units, as shown in Figure 3.6. At any clock cycle, a new data is written into the storage unit indicated by pointer W , while the data in storage unit indicated by R is read out for arithmetic operation. The pointers are circulated in the direction indicated by the solid and dashed arrows in Figure 3.6 over the storage units along

Table 3.5: Number of Adders Used to Implement the Filters

	Proposed / Best Published References						
Filters	$L2$ / [58]	$L1$ / [58]	$L3$ / [58]	$S1$ / [58]	$S2$ / [58]	$Y1$ / [57] / [56]	$G1$ / [38]
Filter Order	62	120	35	24	59	29	15
Coefficient Effective Wordlength	10 / 11	14 / 14	6 / 8	7 / 8	10 / 10	10 / 10 / 12	6 / 8
MB Adders	13 / 17	30 / 44	5 / 3	4 / 4	18 / 19	3 / 7 / 9	2 / 2
Structural Adders	58 / 62	112 / 120	27 / 35	16 / 24	53 / 59	21 / 23 / 21	11 / 13
Extrapolation Adders	2 / –	8 / –	4 / –	2 / –	2 / –	4 / – / –	2 / –
Scaling Factor Adders	0 / –	0 / –	0 / –	2 / –	0 / –	0 / – / –	0 / –
Total Adders	73 / 79	150 / 164	36 / 38	24 / 28	73 / 78	28 / 30 / 30	15 / 15
Delay Chain Length d	13 / –	9 / –	4 / –	5 / –	6 / –	5 / – / –	3 / –
Number of Delay Chain(s) k	2 / –	2 / –	2 / –	1 / –	1 / –	1 / – / –	1 / –
Maximum Adder Depth	4 / 4	4 / 4	4 / 3	3 / 3	4 / 4	3 / 3 / 3	3 / 3
Average Adder Depth	2.27 / 2.42	2.47 / 2.86	1.61 / 1.72	1.31 / 1.69	2.5 / 2.53	1.33 / 1.67 / 1.47	1 / 1.75

Figure 3.6: Illustrative of the delay chain.

the clock cycle. The circuit diagram of the delay chain with $d = 3$ is shown in Figure 3.7. The circuit consists of two parts: a control logic and $d + 1$ storage units. The control logic is a shifting register constructed by $d + 1$ 1-bit D Flip-flops (DFF) connected in a circulative manner, whereas each storage unit consists of B -bit D latches connected by the enable signal EN_1 . The shifting registers are initialized to have one output HIGH, while the others LOW. The HIGH output is circulated to enable one of the storage unit to be written, and at the same time select another storage unit to be read out by controlling the enable signals EN_1 and EN_2 in the storage units at any clock cycle. During the operation of the delay chain, at any clock cycle, only two storage units and two DFFs may change states. Therefore, the overall power consumption incurred due to the delay chain is low. When two delay chains are used in the extrapolated filter, the control logic of the shifting register may be shared by the two chains.

A detailed analysis and estimation of the power consumption incurred due to the delay chain(s) in comparison with that of full adders and tap delay elements are given in Appendix A; the results are listed in Table 3.6, where the data width is B -bits. E is a unit of power defined in Appendix A. It is noted that the power

Figure 3.7: Circuit diagram of a delay chain.

consumed by the delay chain(s) is independent of the length of the delay chain. The increased power due to one delay chain and two delay chains, given by $P_2 - P_4$ and $P_3 - 2P_4$ is less than one sixth of the power consumed by an adder with width B , respectively, where P_2 , P_3 and P_4 are given in Table 3.6.

Furthermore, in the estimation of the power consumption, it was assumed that there is no spurious switching in the operation of the circuit. In practice, however, full adders experience substantial spurious switching [76, 77], whereas the delay chains constructed from D-latches and DFFs do not result in much of such switching. Therefore, the overall power consumed due to the implementation of the extra delay chain(s) is negligible in comparing with the saving achieved due to the reduction of adders.

Table 3.6: Average power of B -bit delay chain, full adder and tap delay element

	power (E)
$P_1 = \text{An adder}$	$2.5B$
$P_2 = \text{A delay Chain}$	$3 + 2.125B$
$P_3 = \text{Two delay chains}$	$3 + 3.125B$
$P_4 = \text{A tap delay}$	$1.75B$

Table 3.7: Complexity Comparison of Delay Chains and Full Adders. The external signal wordlength is assumed to be 12 bits.

Filters	$L2$	$L1$	$L3$	$S1$	$S2$	$Y1$	$G1$
Reduction in Number of Adders	6	14	2	4	5	2	0
Intermediate Result Wordlength B	22	26	18	19	22	22	18
Reduction in Number of Transistors for Adders	3696	10192	1008	2128	3080	1232	0
Number of Extra Transistors in Delay Chains	2600	1928	404	362	530	404	136
Reduction in Total Number of Transistors	1096	8264	604	1766	2550	828	-136
Reduction in Equivalent Number of Adders	1.8	11.4	1.2	3.3	4.1	1.3	-0.3

3.6 Discussion

It has been shown in Section 3.4 that the proposed technique synthesizes filters using less adders but employing extra delay elements. To give a sense on the reduction of implementation complexity, in this section, a transistor level comparison is given based on the realization proposed in Section 3.5 and Appendix A. The equivalent number of adders saved is thus obtained. In addition to the implementation complexity, the adder depth of the filters designed using the proposed technique is investigated and the applicability of the proposed technique is discussed.

3.6.1 Implementation Complexity

A simplified comparison is given in this subsection by counting the number of transistors used in the adders and delay chains in the implementation. Assume that the number of adders reduced is A and each adder is of B -bit. Then the number of transistors reduced is equal to $28AB$ (assume that the conventional 28-transistor full adder given in Figure 3.10 in Appendix A is used). On the other hand, the number

of transistors used to implement one delay chain and two delay chains of length d are $(d + 1)(5B + 16) - 16B$ and $(d + 1)(10B + 16) - 32B$, respectively. Thus, the number of transistors reduced due to the reduced number of adders and the number of transistors increased due to the extra delay chain are shown in Table 3.7 for each design presented in Section 3.5. In Table 3.7, the wordlength of the intermediate result, B , is assumed to be the sum of the wordlength of the external signals and the wordlength of the filter coefficients. Assumed also that the wordlength of the external signal is 12 bits. The equivalent number of adders saved after taking the additional delay chains into consideration is listed in the last row of Table 3.7.

From Table 3.7, it is shown that the proposed technique reduces the implementation complexity in most cases. The reduction amount, varying from a few adders to more than 10 adders, depends on many factors such as the filter length, coefficient effective wordlength, the number and length of the delay chains. For very short filters with short coefficient wordlength, for instance $G1$ with filter order 15 and coefficient effective wordlength 6, the net saving in implementation is negative; this is because the MB in such a case is small and the saving in MB adders and structural adders are canceled by the extrapolation adders. In general, in order to achieve implementation complexity reduction, the total number of adders reduced, A , using the proposed algorithm should satisfy

$$A > \frac{5kd}{28} \quad (3.15)$$

where k is the number of delay chains. In (3.15), it is assumed that B is larger than $(d + 1)$, which in general is true.

3.6.2 Adder Depth

Adder depth is defined as the number of adders that a signal goes through from the input to a delay element. It was pointed out in [76, 77] that the power consumption

of a MB is often not proportional to the number of adders but is rather very much dependent on the adder depth of every coefficient. More specifically, the average dynamic power of a circuit can be expressed by [85]

$$P_{dyn} = 0.5V_{DD}^2 f_c C_L \alpha, \quad (3.16)$$

where V_{DD} is the supply voltage, f_c is the clock frequency, C_L is the load capacitance and α is the switching activity. All the parameters, except α , are defined by the physical layout and specifications of the circuit. Apparently, larger α would result in more dynamic power consumption. It is pointed out in [76, 77] that in a network of adders, for instance the MB, one of the main contributions to the increase of α is due to an undesirable effect called “glitch”. In [76] a simple model for estimating the glitch path among all the coefficients is proposed. However, it is shown in [77] that the model in [76] is not always accurate. Nevertheless, it is in general agreed that larger adder depth indicates more glitches and hence higher power consumption [76, 77].

Listed in Table 3.5 are the maximum adder depth (MAD) and the average adder depth of each design. It is shown that the filters designed using our proposed technique have the same MAD in most cases and smaller average adder depth in all cases when compared with the best previously published results. The reduction in the average adder depth is due to the fact that the dynamic range of the residuals is significantly reduced compared with their original coefficients. These results show that the proposed filter structure will not incur any power consumption overhead due to increased adder depth; rather in addition to the power consumption reduced due to the reduced adders, the power consumption may be slightly further reduced because of the smaller average adder depth.

3.6.3 Applicability and Limitation

The proposed technique is applicable to the design of any FIR filter since the quasi-periodic property is inherent in the impulse response. Therefore, for those impulse responses that do not have multiple lobes, as long as the periodicity is significant (which is true in most cases, e.g. see example *L1*), the proposed technique can be employed. While the inaccuracy of the periodicity may degrade the frequency response in the traditional extrapolated filter, the proposed technique is immune to the problem of imperfect periodicity due to the introduction of residual compensation. The magnitude of the residual might be larger than that of the original coefficient value; such situation, however, does not exist in the design of the current examples, whose specifications vary from lowpass to highpass, narrow band to wide band, and moderate ripple requirements to stringent ones. When the length and location of the prototype lobe is properly selected, the magnitude of residual hardly increases. Even if one or two residuals increase the magnitudes in exceptional examples, the overall dynamic range of coefficient values and residuals resulted from the extrapolation decreases for sure; the proposed optimization technique further ensures that the values with increased magnitudes (if any) are synthesized using not more than the required number of adders.

Therefore, in the proposed technique, what actually matters is if the reduction in the MB adders and structural adders is large enough to compensate for the increase in the extrapolation adders and scaling factor adders. In choosing the scaling factors, it is suggested that one start with some numbers of minimum implementation complexity (i.e., scaling factors that require no adders). Otherwise, the reduction of MB adders might be negated by the increase of realizing scaling factors (similar to the case of extrapolation adders). As shown in Section 3.6.1, when both filter length and coefficient wordlength are short, net implementation complexity reduction is not guaranteed. Furthermore, the saving on the adders may be canceled by the additional delay chain if the delay chain is long. For instance, the proposed

design of $L2$ uses 6 adders less than that in [58] as shown in Table 3.5. However, the equivalent saving after compensating the delay chains is only 1.8 adders as shown in Table 3.7 due to the long delay chains of length 13. If the implementation complexity is the major concern, a discrete solution using 4 adders less than that in [58] could be obtained using the proposed technique where a prototype lobe with length 5 is adopted; in this design, the saving on the number of adders is reduced, but the equivalent saving after compensating for the delay chain increases to 3.2 adders. Nevertheless, the design with delay chain length of 13 achieves more power saving since the power incurred due to the delay chains is independent of the length of delay chains and is negligible. Therefore, depending on the design requirements, complexity priority or power priority design could be chosen by selecting different prototype lobes.

3.7 Conclusion

In this chapter, an extrapolated impulse response with residual compensation is proposed for synthesizing linear phase FIR filter in subexpression sharing. The proposed filter structure reduces the dynamic range of the coefficient values to be synthesized and this further reduces the lower bound of the number of adders required to synthesize a filter. Both the filter coefficients and the residuals may be optimized directly in subexpression space. MILP is formulated to optimize the filter coefficients and the residuals. Examples have shown that the number of adders used to synthesize the filter is significantly reduced. The reduction varies from a few adders to more than 10 adders for different designs if the filter order is not lower than 20. Extra delay chains are used in the proposed structure. The power overhead incurred due to the proposed delay chains is negligible, whereas the increase in the implementation complexity is insignificant if the prototype lobe is chosen properly.

Appendix A: Average Power Estimation of Delay Chain, Full Adder and Tap Delay Element

The power consumption of the delay chain described in Section 3.5 as well as the conventional 28-transistor 1-bit full adder [2] and tap delay element is estimated in this section. To simplify the analysis, the first order estimation is used and the following assumptions are made:

1. Static power consumption is much smaller than dynamic power consumption, and therefore ignored.
2. Gate capacitance is considered as load capacitance. Drain and source capacitances are ignored.
3. All p-transistors used in the full adder, delay chain and tap delay element adopt the same gate size and therefore have the same gate capacitance C_p , whereas all n-transistors have the same gate capacitance C_n . $(C_p + C_n)$ is considered as a unit load capacitance.
4. The external load of the delay chain, the tap delay element and the sum output of the full adder is the addend input of the subsequent full adder, whereas the external load of the carry out of the full adder is the carry in input of the subsequent bit of full adder. This assumption is consistent with the filter structure shown in Figure 3.4.
5. Input data is random where 0 and 1 have equal probability.

Based on the above assumptions, the power consumed in a clock cycle by a circuit to charge a node with a unit load capacitance (consisting of a p-gate and an n-gate) from 0 to 1 is considered as a unit of power consumption, and denoted as E . $E = E_n + E_p$, where E_n and E_p are the power consumed when capacitance C_p and C_n are charged from 0 to 1, respectively.

D-latch

Figure 3.8: Circuit diagram of a D-latch.

Table 3.8: States of the storage unit

	EN1	EN2	Remark
ST0	0	0	—
ST1	1	0	A new data is written into the unit
ST2	0	1	The data in the unit is read

Delay Chain

Two types of components, D-latch and D-flip-flop, are used in the realization of the delay chain. Consider a 1-bit storage unit consisting of a D-latch. The gate level diagram is shown in Figure 3.8. According to the structure in Figure 3.7, each storage unit only has three possible states: ST0, ST1 and ST2, as listed in Table 3.8. In order to estimate the power of the whole circuit in one clock cycle, the voltage change of both internal and external nodes are considered.

In the operation of the circuit, at any clock cycle, one latch transits from ST0 to ST2, one from ST2 to ST1 and one from ST1 to ST0. All the other storage units remain the state ST0 unchanged. The storage units remaining the states or changing the state from ST1 to ST0 consume 0 power, regardless the values of the data D .

Figure 3.9: Circuit diagram of a D-flip-flop.

When the storage unit changes the state from ST2 to ST1, only node $\bar{Q}$ with capacitance load C_p may be charged from 0 to 1, if the present data D is 0 and the previous data D is 1. When the present and previous data of D are in other combinations, no power is consumed. In average, $0.25E_p$ is consumed.

When the storage unit changes the state from ST0 to ST2, if the node Q , which is the addend input of a full adder, is changed from 0 to 1, $4E$ is consumed. If Q remains unchanged, or changes from 1 to 0, no power is consumed. Therefore, in average, $1E$ is consumed.

From the above analysis, in one clock cycle, $E + 0.25E_p$ is consumed for 1-bit storage unit. A B -bit storage unit, thus consumes power of $p_1 = (0.25E_p + E)B$.

A master-slave DFF is constructed by using 2 D-latches, connected as shown in Figure 3.9. For a B -bit delay chain, nodes $\bar{Q}_0$, Q_0 and $\bar{Q}$ consist of 1 unit capacitance each. Since Q is connected to the subsequent DFF as well as the control signals of EN_1 and EN_2 of the storage units, as shown in Figure 3.7 and Figure 3.8, node Q consists of 1 unit capacitance plus $2B$ of capacitance C_n . In the operation of the shift register, at any clock cycle, the data stored in one DFF changes from 0

to 1 and another one from 1 to 0, All the other DFFs keep the values unchanged and thus do not consume any power. The two DFFs, which change states, consume $p_2 = E + 2BE_n$ and $p_3 = 2E$, respectively, for a delay chain with data width B . Thus, the total power consumed by one delay chain in a clock cycle is

$$\begin{aligned} P_2 &= p_1 + p_2 + p_3 \\ &= (E + 0.25E_p)B + (E + 2BE_n) + 2E \\ &= 3E + (1.25E_p + 1.75E_n)B. \end{aligned} \tag{3.17}$$

Assume that $E_n = 0.5E$, we have $P_2 = (3 + 2.125B)E$. Note that E_n is in general smaller than $0.5E$ since C_n is generally smaller than C_p .

When two delay chains are employed, node Q in Figure 3.9 consists of 1 unit capacitance plus $4B$ of capacitance C_n . Thus, the power consumed by the shift register for two delay chains is $(3+3.125B)E$. The overall power consumed by the delay chains are listed in Table 3.6.

Tap Delay

Tap delays are realized by DFF too. For each bit of tap delay, $\overline{Q_0}, Q_0, \overline{Q}$ and Q consist of 1, 1, 1 and 4 unit capacitance, respectively. In one clock cycle, if the data in the DFF remains unchange, no power is consumed; if the data changes from 0 to 1, $5E$ is consumed; and if the data changes from 1 to 0, $2E$ is consumed instead. In average, $1.75E$ is consumed for 1-bit tap delay. The value for tap delay listed in Table 3.6 is the power consumption of a B -bit tap delay.

Full Adder

Figure 3.10 shows a conventional 28-transistor ripple carry adder. This circuit has 3 inputs (A, B, C_{in}), 2 outputs (S, C_{out}). Table 3.9 lists all possible state transitions of this adder and the corresponding power consumption where the transitions of both

internal and external nodes are considered. It is shown that in average, a 1-bit full adder consumes power of $2.5E$. The value listed in Table 3.6 is a B -bit adder.

Figure 3.10: Circuit diagram of a 28-transistor full adder.

Table 3.9: The first order estimation of the Power Consumed by the full adder in all transitions

	Power consumed(E)	Final state ABC_{in}								Average
		000	001	010	011	100	101	110	111	
Initial state ABC_{in}	000	0	4	4	3	4	3	3	7	3.5
	001	1	0	0	4	0	4	4	3	2
	010	1	0	0	4	0	4	4	3	2
	011	2	6	6	0	6	0	0	4	3
	100	1	0	0	4	0	4	4	3	2
	101	2	6	6	0	6	0	0	4	3
	110	2	6	6	0	6	0	0	4	3
	111	3	2	2	1	2	1	1	0	1.5
	Overall									2.5

Chapter 4

Design of Linear Phase FIR Filters with High Probability of Achieving Minimum Number of Adders

MCM techniques in the second category, as reviewed in Section 2.3.3, possess a lot of advantages over the ones in the first category mainly in that the complexity of the resultant filter can be even reduced. However, the various techniques reviewed in Section 2.3.3 in [15, 16, 38, 48, 56–61, 75, 86], are not able to guarantee that the results obtained are optimum. By “optimum” in this chapter, we mean a design using minimum number of adders. In addition, there are no mechanisms in these algorithms that can tell whether a specific result is a global optimum one (if these algorithms happen to find such an optimum result). Since for any given filter specifications, there is no known lower bound for the total number of adders, in order to achieve the optimum synthesis, all the feasible discrete solutions must be taken into account.

In this chapter, an algorithm that is capable of traversing the entire discrete solutions efficiently for a given wordlength is proposed. What is of more importance is that during the traverse, the optimality of the synthesis of a set of discrete coefficients is monitored and flagged by a “certainty”, whenever a new coefficient is discretized. In such a manner, when the traverse is completed, the algorithm is able

to be aware of whether an optimum solution is obtained. From our experience on the design of benchmark filters, in most cases, the optimum solution can be claimed.

While the number of adders used to realize an FIR filter is an important criterion of the implementation complexity, the power consumption and circuit speed, on the other hand are much more related to the adder depth of the filter as discussed in Section 2.4. Obviously, low adder depth is preferable for the concerns of low power consumption and high throughput if the same number of adders are required. Maximum adder depth constraint should also be taken into consideration in the algorithmic design.

The remaining of this chapter is organized as follows. Section 4.1 discusses the formulation of the filter design problem with respect to the entire traverse of the discrete solutions. In addition, the effect of coefficient wordlength is briefly discussed. In Section 4.2, the basic traverse process of the proposed algorithm is explained using an example while in Section 4.3, the detailed quantization process for the filter coefficients during the traverse process is explained. Also, the “optimum-awareness” mechanism for determining the optimality of the discrete solution is introduced. In Section 4.4, three sets of examples are designed, each showing the advantages of the proposed algorithm over others from different perspectives. The results are compared with the best published references and the superiority of the proposed algorithm is illustrated by the reduced number of adders, filter length, effective coefficient wordlength (EWL) and adder depth, respectively. A brief analysis of the computational complexity is given in Section 4.5 with 2 examples.

4.1 Problem Formulation

In this section, we formulate the filter design problem as a mixed integer linear programming (MILP) problem. A new objective function is proposed to accelerate

the traverse procedure when MILP is incorporated in a tree search. Furthermore, we briefly discuss the effect of coefficient wordlength.

Before going into the details of the problem formulation, the general steps of the filter design problem using MILP, are given as follows:

1. Formulation the filter optimization problem.
2. Obtaining the continuous optimum solution.
3. Choosing the effective wordlength (*EWL*).
4. Traverse of the discrete solutions.

It should be noted that the second step is not discussed in this thesis because there exist many efficient numerical packages for solving linear programming problems, which is not the focus of this thesis. Steps 1) and 3) are addressed in this section while Step 4) is explained in detail in the next two sections.

4.1.1 Optimization Problem Formulation

The zero-phase frequency response of a Type I linear phase FIR filter with an order of $2N$ can be expressed as [60]:

$$H(\omega) = h(0) + 2 \sum_{n=1}^N h(n) \cos(n\omega). \quad (4.1)$$

The zero-phase frequency responses of Type II, III and IV linear phase FIR filters can be derived similarly. The optimization problem can thus be formulated to find the filter coefficient values of $h(n)$ for $n = 0, 1, \dots, N$, within a given discrete space, such that the normalized peak ripple (NPR) [47] is minimized.

Assume that δ and b are the peak ripple and passband gain, respectively. The NPR, which is defined as δ/b , is non-linear and therefore cannot be optimized by

using linear programming directly. In order to circumvent the problem, the optimization problem is formulated as shown in [47]:

$$\begin{aligned}
 & \text{minimize : } f = \delta - \alpha b & (4.2) \\
 & \text{subject to : } b - \delta \leq H(\omega) \leq b + \delta, \text{ for } \omega \in [0, \omega_p] \\
 & \quad -(\delta_s \delta) / \delta_p \leq H(\omega) \leq (\delta_s \delta) / \delta_p, \text{ for } \omega \in [\omega_s, \pi] \\
 & \quad b_l \leq b \leq b_u, & (4.3)
 \end{aligned}$$

where f is the objective function to be minimized, δ is the ripple, α is a pre-defined constant, δ_p , δ_s , ω_p and ω_s are the given passband ripple, stopband ripple, passband edge and stopband edge, respectively. $H(\omega)$ is the frequency response of the filter given in (4.1). b_l and b_u are two constants, specifying the lower bound and upper bound of the passband gain, respectively. For FIR filters synthesized in binary arithmetic, the lower bound of passband gain is unnecessary to be less than half of the upper bound [47]. In the following of this chapter, b_l and b_u are set as 0.7 and 1.4, respectively, when the gain is floating in the course of optimization.

The above optimization formulation does not directly produce the optimum NPR. Therefore, it is shown in [47] that by using a recursive α replacement technique, a discrete solution that gives the optimum NPR can be obtained after several MILP runs.

The algorithm introduced in [47] is targeted to minimize the NPR. However, to design a filter using minimum number of adders, it is not necessary to minimize the NPR, i.e., finding a solution with NPR less than the specification is sufficient. Therefore a new objective function

$$f = \delta - \delta_p b \quad (4.4)$$

is proposed. This is because the NPR in (4.4) can be expressed as

$$\delta/b = f/b + \delta_p. \quad (4.5)$$

Therefore, as long as $f \leq 0$, we have $\delta/b \leq \delta_p$, which means that the NPR satisfies the passband ripple requirement. Hence, the optimization problem is formulated as:

$$\text{minimize : } f = \delta - \delta_p b \quad (4.6)$$

$$\text{subject to : } b - \delta \leq H(\omega) \leq b + \delta, \text{ for } \omega \in [0, \omega_p]$$

$$-(\delta_s \delta)/\delta_p \leq H(\omega) \leq (\delta_s \delta)/\delta_p, \text{ for } \omega \in [\omega_s, \pi]$$

$$b_l \leq b \leq b_u. \quad (4.7)$$

Thus, if the objective function f is larger than 0 after the running of MILP, it means that no feasible solution satisfying the NPR requirement is available. The recursive α replacement is reduced to a single MILP run. In this way, the optimization is accelerated without optimality compromise. In addition, $f > 0$ can be used as a criterion to terminate a path in the traverse of the discrete solution; further traverse along the path will not produce any solution satisfying the NPR requirement.

4.1.2 Choosing the *EWL*

After the the continuous optimum solution for the problem (6), (7) is obtained, all the filter coefficients are scaled up to be within a certain *EWL*. *EWL* is defined as the number of bits used to represent the filter coefficients (excluding the sign bit). These scaled coefficients are then to be fixed to discrete values during the traverse later. Obviously, longer *EWL* results in less quantization error and hence better frequency response. However, from the perspective of low complexity, low power consumption and high speed implementation, large *EWL* is usually undesirable. Therefore, in this chapter, we choose the *EWL* as small as possible provided that

feasible discrete solutions can be found within it. This is clearly illustrated in the design examples in Section 4.4. Furthermore, *EWL* is much related to the computational complexity of the proposed algorithm as discussed in Section 4.5.

4.2 Traverse of the Discrete Solutions

In this section, we explain roughly the traverse process of the algorithm. Before commencing the traverse, it is beneficial to know the lower bound h_k^l and upper bound h_k^u of coefficient $h(k)$, for $k = 0, 1, 2 \dots, N - 1$. These bounds can be found by solving the following linear programming problem:

$$\begin{aligned}
 & \text{minimize : } f = h(k) \\
 & \text{subject to : } b - \delta_p \leq H(\omega) \leq b + \delta_p, \text{ for } \omega \in [0, \omega_p] \\
 & \quad \quad \quad -\delta_s \leq H(\omega) \leq \delta_s, \text{ for } \omega \in [\omega_s, \pi] \\
 & \quad \quad \quad b_l \leq b \leq b_u.
 \end{aligned} \tag{4.8}$$

The above optimization finds the lower bound of coefficient $h(k)$, denoted as h_k^l . To find the upper bound h_k^u , simply set the objective function to $f = -h(k)$. Therefore, for N coefficients, there are totally $2N$ runs of linear programming for finding the bounds. The main purpose of finding the bounds of the filter coefficients is explained in Subsection 4.3.4.

The traverse process is then performed using MILP to find the optimum discrete solution. Basically, the traverse is a depth-first width-recursive search [52]. Filter coefficients are quantized to certain integers one by one according to the rules explained in detail in Section 4.3. When a filter coefficient is quantized, the rest un-quantized ones are reoptimized to compensate for the loss in frequency response. For example, as shown in Figure 4.1, starting with the optimum continuous design

Figure 4.1: An example of the traverse process.

P_0 , coefficient $h(0)$ is quantized to -3 , then the rest of the coefficients ($h(1)$ to $h(N-1)$) are reoptimized. Then, $h(1)$ is further quantized to 9 and the coefficients (from $h(2)$ to $h(N-1)$) are reoptimized. The process continues, as indicated in Figure 4.1, until the last coefficient $h(N-1)$ is quantized to 984 . In this case, a discrete solution is obtained along Path 1 and the optimality of it will be checked, which is discussed in Section 4.3. After finishing the traverse along Path 1, the program goes back to quantize $h(N-1)$ to other integers in the range $[h_{N-1}^l, h_{N-1}^u]$

one by one and the optimality of these solutions are also checked. Thereafter, the search is traced back to $h(N-2)$ and $h(N-2)$ is then fixed to another integer in the range $[h_{N-2}^l, h_{N-2}^u]$. With the new fixed value of $h(N-2)$, the process of search forwards to reoptimize $h(N-1)$ and the quantization of $h(N-1)$ to discrete values is repeated.

In such a way, the traverse (containing back tracing, switching to new fixed values, searching forward and checking optimality) is repeated until all solutions which has the possibility of yielding a feasible discrete solution using less adders are searched. The search along any path may be terminated at some node as shown in Figure 4.1, where Path m is terminated at node $h(2)$ when it is quantized to 29. This termination can be due to two criteria, which are addressed in Section 4.3.

Apparently, the maximum depth of such a tree is N and the total number of paths to be traversed is proportional to $\prod_{k=0}^{N-1} (h_k^u - h_k^l)$. However, due to the criteria for terminating the nodes, the actual number of paths to be traversed is much smaller.

4.3 Detailed Quantization Process

In this section, we explain in detail the quantization process of filter coefficients during the traverse of discrete solutions which is discussed in Section 4.2. First, the detailed quantization process of filter coefficients is explained, in which a “optimum-awareness” mechanism measured by “certainty” is introduced to monitor the optimality of discrete solutions. Secondly, some remarks are made about the manner of selecting discrete values and optimality of the solution. It should be noted that the concept of dynamically expanding subexpression basis set is used here. Readers are to refer to the introduction in Section 2.3.3 for details.

4.3.1 Dynamically Expanding Subexpression Basis Set

A subexpression basis set is defined as a set that contains zero and odd-valued integers [58]. The order of a subexpression basis set is defined as the number of adders that is required to realize the elements in the set. For instance, the order of the subexpression basis set $\{0, \pm 1, \pm 3, \pm 5, \pm 7\}$ is 3. Here, it is assumed that if a positive odd number is realized, its negative value is automatically realized and the elements 0 and ± 1 are always included. Therefore, for expository convenience, we omit the negative values and 0. Thus, the above subexpressions basis set is simplified as $\{1, 3, 5, 7\}$. The concept of dynamically expanding subexpression basis set is introduced in [60], where the subexpression basis set is initialized to contain only 0 and 1. During the traverse process of the algorithm, each node is associated with a subexpression basis set, and the subexpression basis sets are updated dynamically. When the traverse searches forward along a path with more and more coefficients being quantized to integers, the subexpression basis set expands, while the search is traced back to a coefficient, the original basis set of that coefficient is recovered.

4.3.2 The Quantization Process of a Certain Coefficient

Figure 4.2 shows the basic operations that a coefficient $h(k)$ is to be quantized during the traverse of discrete solutions. In this scenario, $h(k-1)$ has been quantized to a discrete value. The rest unquantized coefficients (including $h(k)$) are reoptimized. $h(k)$ is of continuous value and therefore needs to be fixed to certain integer. The detailed descriptions of the operations are as follows:

1. If all the coefficients have been fixed (recall Path 1 in Figure 4.1), a discrete solution has been found. The optimality of the solution needs to be checked while the solution is to be recorded; the right branch of the diagram in Figure 4.2 is taken. Let $N_{total} = N_{sub} + N_{filter} - 2N_{zero}$ be the total number of adders required to realize the filter, where N_{filter} is the order of the filter,

Figure 4.2: Diagram of the quantization process of coefficient $h(k)$.

N_{zero} is the number of zero coefficients in the discrete solution and N_{sub} is the order of the corresponding subexpression basis set (which is equivalent to the number of multiplier block adder adders used to synthesize this discrete solution). The term $-2N_{zero}$ included in the computation of N_{total} is because a 0 coefficient in a linear phase FIR filter results in 2 SA reduction. Also, let N_{best} be the current best result. If N_{sub} for this discrete solution is “certain” and $N_{total} < N_{best}$, N_{best} is updated to N_{total} . However, it may happen that N_{sub} is “uncertain”. In that case, this solution is simply recorded without updating N_{best} . The “certain” and “uncertain” of a N_{sub} is the mechanism introduced to monitor the optimality of a solution and is explained in Subsection 4.3.3. Since all coefficients have been discrete values, the program goes back to the quantization process of $h(k-1)$.

2. If not all the coefficients are fixed, as indicated in Figure 4.2, the left branch of the diagram is taken. $h(k)$ is to be fixed to some integers, say $h^o(k)$. The adder cost for realizing $h^o(k)$ is estimated based on the current subexpression basis set of the path (which is inherited from the quantization process of coefficient $h(k-1)$). The details of adder cost estimation is discussed in Subsection 4.3.3 and the manner to select the integer values of $h(k)$ is presented in Subsection 4.3.4.
3. If realizing $h^o(k)$ results in $N_{total} < N_{best}$, the program goes on to check the feasibility of fixing $h(k)$ to $h^o(k)$. Otherwise, the path to $h(k)$ with discrete value $h^o(k)$ is terminated and other possible integers of $h(k)$ are to be checked, i.e., $N_{total} \geq N_{best}$ is the first criterion to terminate a path. If no more integers of $h(k)$ results in an adder cost $N_{total} < N_{best}$, the program goes back to the quantization process of $h(k-1)$.
4. Assume that fixing $h(k)$ to $h^o(k)$ results in an adder cost lower than N_{best} , the objective function f in (6) is then minimized by reoptimizing the rest of the

Table 4.1: An example of adder cost estimation with the current best result $N_{best} = 13$

Coefficients	Set	Adder Cost	$N_{sub} - 2N_{zero}$	Return
$h(0) = -25$	$\{1, \mathbf{25}\}$	2^*	2^*	Y
$h(1) = 96$	$\{1, 25, \mathbf{3}\}$	1	2	Y
$h(2) = 8$	$\{1, 25, 3\}$	0	2	Y
$h(3) = 74$	$\{1, 25, 3, \mathbf{37}\}$	1	3	N

coefficients that are not fixed to integers yet. If $f > 0$, it means that fixing $h(k)$ to $h^o(k)$ will not produce any feasible solution and the path is therefore terminated, i.e., $f > 0$ is the second criterion for terminating a node.

5. If $f < 0$, it means that the solution so far still meets the filter specifications.

The program then goes to the quantization process of $h(k+1)$ which has the current subexpression basis set inherited. The quantization process of $h(k+1)$ is the same as that of $h(k)$ indicated in Figure 4.2. When no more discrete values are available for $h(k+1)$ to fix, the algorithm is traced back from the quantization process of $h(k+1)$ to that of $h(k)$ with subexpression basis set recovered; $h(k)$ is fixed to other integers. The same loop is gone through again.

4.3.3 Adder Cost Estimation

In this subsection, the adder cost estimation block is explained by using an instance listed in Table 4.1, where the initial subexpression basis set is always $\{1\}$. The subexpression basis set expands with more and more coefficients ($h(0), h(1), h(2), \dots$) are fixed. Assume that the order of the filter is 10, i.e., $N_{filter} = 10$ and the current best result $N_{best} = 13$. Since N_{filter} is a fixed value, in the following discussion, we shall only focus on the term $N_{sub} - 2N_{zero}$.

As shown in Table 4.1, when $h(0)$ is fixed to -25 , the adder cost is estimated based on basis set 1. Since 25 cannot be realized from the elements in the set using only one adder and thus the cost for realizing it must be equal to or greater than 2.

The adder cost is thus assumed to be 2 temporarily with * marked over the number to represent the uncertainty of the adder cost. With more and more numbers added into the subexpression basis set, it is likely to realize 25 using only one adder based on those numbers added later. Therefore, in our technique, at the current stage we assume that the number of adders required to construct 25 is two and 25 is thus added to the subexpression basis set. Apparently, at the current stage, the actual number of adders used to synthesize the coefficients is “uncertain” because of 25. After the quantization of $h(0)$, $h(1)$ is then fixed to 96 which can be expressed as 3×2^5 . Clearly, 3 can be realized using only one adder ($3 = 1 + 1 \times 2^1$) based on the current basis set $\{1, 25\}$ and 3 is thus added to the set. The adder cost for realizing 3 is 1. After adding 3 to the set, the program tries to synthesize 25 again with the new added number 3. It turns out that 25 can be realized using only one adder ($25 = 3 \times 2^3 + 1$). Therefore, the * is removed and the order of this subexpression basis set becomes “certain”. For $h(2)$ and $h(3)$, one more adder is required to synthesize $h(3)$ (37×2^1). However, with 37 is added, $N_{sub} - 2N_{zero} + N_{filter}$ is equal to the best obtained result N_{best} . Therefore, the $N_{total} < N_{best}$ test in Figure 4.2 returns NO.

4.3.4 Discrete Values Selection

In this subsection, the manner to select the discrete values of coefficient $h(k)$ is presented. The important roles played by initial boundaries (obtained in Section 4.2) are revealed meanwhile.

In order to facilitate the searching process, coefficients whose boundaries include 0 should be fixed at first place. The reason is as follows. Coefficients that may be fixed to 0 result in the reduction of structural adders (SA) and hence affect $N_{total} = N_{sub} - 2N_{zero} + N_{filter}$. It is clear from the ongoing discussion that the first criterion for terminating a node is under the assumption that N_{total} does not decrease with more and more coefficients are fixed. Therefore, it is required that

all the coefficients whose initial boundaries include 0 should be fixed at first so that N_{zero} does not vary later on. Furthermore, assume that the number of coefficients that may be fixed to 0 is q ; N_{zero} is thus initialized to $2q$. As these coefficients are quantized one by one, N_{zero} decreases accordingly in a monotonous manner along each path until all of them are fixed at the very early stages during the searching process.

Thereafter, the discrete values of $h(k)$ are selected in the following manner. Let $\lfloor x \rfloor$ and $\lceil x \rceil$ be the floor and ceiling functions, respectively. Integers $\lfloor h(k) \rfloor, \lfloor h(k) \rfloor - 1, \lfloor h(k) \rfloor - 2, \dots$ towards the lower bound $h^l(k)$ or $\lceil h(k) \rceil, \lceil h(k) \rceil + 1, \lceil h(k) \rceil + 2, \dots$ towards the upper bound $h^u(k)$ are to be enumerated. In other words, our proposed algorithm quantizes $h(k)$ in one direction towards the initial bound or until $f > 0$ and then switches to the opposite direction.

It should be noted that the actual boundary of $h(k)$ is much narrower than the initial one ($[h^l(k) \ h^u(k)]$) with more and more coefficients quantized. However, from the computational complexity point of view, extra loops of adder cost estimation is preferred rather than two more MILP runs to specify the actual lower bound and upper bound of $h(k)$. Furthermore, in our proposed algorithm, the adder cost checking ($N_{total} < N_{best}$ in Figure 4.2) is executed before the NPR checking ($f \leq 0$ in Figure 4.2). Thus, if the first criterion ($N_{total} < N_{best}$) is not satisfied, the execution of MILP for the second criterion is saved. From our design experience, the first criterion for terminating a node is met more often than the second one as better and better N_{best} is obtained. Hence, many unnecessary MILP runs can be eliminated. However, once a MILP is run and f is larger than 0, fixing $h(k)$ to integer values beyond the current fixed value will not produce any solution with $f \leq 0$. $h(k)$ is then fixed to the other integer values (if any) in the opposite direction.

4.3.5 Optimality of the Solution

The optimality of the discrete solution is very much related to the “certainty” of the corresponding subexpression basis set. If the discrete solution with smallest N_{total} lies in the set of discrete solutions whose corresponding subexpression basis sets are “certain”, the optimum solution with $N_{total} = N_{best}$ is found. Through the “certainty” of the best solution’s subexpression basis, the proposed algorithm is aware of the optimality if an optimum solution is obtained. However, it could happen that the discrete solution with smallest N_{total} lies in the set of discrete solutions whose certainty is not specified. In this case, the optimality of the discrete solution with $N_{total} = N_{best}$ is not certain, but the optimum solution will use at most N_{best} adders to synthesize; the optimum solution may lie in those uncertain discrete solutions with $N_{total} < N_{best}$.

For the “uncertain” discrete solutions, extra elements should be inserted into the subexpression basis set to determine the actual N_{total} . Many algorithms have been proposed to deal with this case [44, 72]. The reason that we do not include those algorithms during the adder cost estimation block is because: 1) this kind of problem is NP hard and increases the computation time of the algorithm; 2) the optimality of the element inserted is not guaranteed and the insertion of such elements spoils the algorithms’s awareness of optimality; 3) from our experience, it is very often that all the elements in the subexpression basis set of the optimum solution can be synthesized using only one adder when more and more coefficients are quantized. Therefore, in our proposed algorithm, if there are such uncertain solutions, we simply record them and compare them with the best certain ones ever found after the program ends. If such situation happens, the optimality of the final results is not assured. However, in most cases, the proposed algorithm generates the best result with “certainty” and thus optimality is assured.

Table 4.2: Specifications of design examples. ω_p , ω_s , δ_p and δ_s are the passband edge, stopband edge, passband ripple and stopband ripple, respectively.

Filters	Filter length	ω_p	ω_s	δ_p	δ_s
$L2$	63	0.2π	0.28π	0.028	0.001
$S1$	24	0.3π	0.5π	0.0157	0.0066
$S2$	60	0.042π	0.14π	0.012	0.001
$Y1$	30	0.3π	0.5π	0.00316	0.00316
$Y2$	38	0.3π	0.5π	0.001	0.001
$G1$	16	0.2π	0.5π	0.01	0.01
$A1$	59	0.125π	0.225π	0.01	0.001
$X1$	15	0.2π	0.8π	0.0001	0.0001

Table 4.3: Specifications of $L3$. δ_p^+ and δ_p^- are the upper bound and lower bound of the passband, respectively. The filter length is 36.

Bands(π)	δ_p^+ (db)	δ_p^- (db)	δ_s (db)
$0 \sim 0.15$	0.2	-0.2	-
$0.15 \sim 0.1875$	0.2	-0.5	-
$0.1875 \sim 0.2125$	0.2	-0.9	-
$0.2875 \sim 1$	-	-	-30

4.4 Design Examples

In this section, three sets of design examples are given to illustrate the superiority of the proposed algorithm. In Example A, 9 benchmark filters are designed using the proposed algorithm in transposed direct form. The results are compared with those of the best published ones. In Example B, one of the benchmark filters is further designed using the proposed algorithm in a residue compensated extrapolated impulse response filter structure [86] (instead of the traditional transposed direct form). In Example C, the constraint of MAD is also incorporated in the algorithm so that the algorithm is able to design optimum filters with low MAD requirement. All the examples are designed using a Pentium 2.4GHz desktop PC.

4.4.1 Example A

The specifications for the 9 benchmark filters are listed in Table 4.2 and Table 4.3, respectively. Filters $L2$ and $L3$ are the second and third examples from [1] while $S1$ and $S2$ are the first and second examples from [16], respectively. $Y1$ and $Y2$ are the first two examples in [57]. $G1$ is the first example in [38] and $A1$ is the example filter marked as “Filter A” in [59]. $X1$ is the first example given in [56]. In Table 4.2, ω_p , ω_s , δ_p and δ_s are the passband edge, stopband edge, passband ripple and stopband ripple, respectively. In Table 4.3, δ_p^+ and δ_p^- are the upper bound and lower bound of the passband gain, respectively.

The design results of the proposed algorithm and the best results from literature are listed in Table 4.4. EWL is the effective wordlength of the coefficients excluding the sign bit. It should be noted that in this particular example, the proposed algorithm successfully finds the optimum solutions for all the filters (except $X1$) and every benchmark filter is first designed using the shortest EWL (for low complexity and high throughput concern). However, for easy comparison, some of them are further designed using longer EWL so that the total number of adders of the corresponding optimum solutions could be compared with those in the references.

It can be seen from Table 4.4 that for filters $L2$, $L3$, $S2$, $Y2$ and $A1$, our proposed algorithm can produce designs with shorter EWL and less total number of adders. For filter $S1$, although the total number of adders is the same as given in the reference, our design has shorter EWL . For $Y1$ and $G1$, the optimum solutions have the same total number of adders and EWL , which means that the solutions given in [60] and [38] are also optimum for $EWL = 10$ and $EWL = 7$, respectively.

For filter $X1$, the proposed algorithm does not find a certain best solution after the traverse of the tree. Instead, three “uncertain” solutions are recorded. Therefore, by using the algorithm in [72], an extra element 3 is inserted into one of the solutions and it turns out that this “uncertain” solution is the optimum one. The coefficients of $X1$ is listed separately in Table 4.5. The total number of adders required to

Table 4.4: Results and Comparison of Example A. *EWL* is the effective wordlength of the filter. MBA and SA are the number of multiplier block adders and structural adders respectively.

Filters	Proposed algorithm / Best published references				
	<i>EWL</i>	MBA	SA	Total Adders	Runtime
<i>L2</i> / [58]	10 / 11	17 / 17	56 / 62	73 / 79	16h28m / 1h8m
<i>L3</i> / [60]	6,7 / 9	4,4 / 5	33,31 / 31	37,35 / 36	3s,16m45s / 45m
<i>S1</i> / [60]	7 / 9	4 / 4	19 / 19	23 / 23	<1s / 40m6s
<i>S2</i> / [58]	10 / 11	17 / 19	59 / 59	76 / 78	16h42m / 3m28s
<i>Y1</i> / [60]	9,10 / 10	7,6 / 6	23,23 / 23	30,29 / 29	6s,5m9s / 21m30s
<i>Y2</i> / [57]	10,11 / 12	10,10 / -	37,27 / -	47,37 / 39	11s, 40m46s / -
<i>G1</i> / [38,60]	6,7 / 7,8	2,2 / 2,2	15,13 / 13,13	17,15 / 15,15	<1s,<1s / -, 3m45s
<i>A1</i> / [59]	10 / 10	15 / 18	54 / 58	69 / 76	50h34m / 4h14m
<i>X1</i> [56]	10 / 13	5 / 7	8 / 8	13* / 15	<1s / -

realize this filter is 13, which is better than the one reported in [56]. Furthermore, the *EWL* is also shorter.

Table 4.5: Impulse response and subexpression basis set of design examples *X1*

Passband gain: 1680.000000; $h(n) = h(14 - n)$ for $0 \leq n \leq 7$			
Subexpression basis set: $\{ 7, 113, 509, 105, \mathbf{3} \}$, $EWL = 10$			
$h(0) = -4$	$h(3) = 0$	$h(6) = 509$	
$h(1) = 0$	$h(4) = -113$	$h(7) = 840$	
$h(2) = 28$	$h(5) = 0$		

The total runtime for all the filters using the proposed algorithm is listed in the last column of Table 4.4. The runtime for the corresponding best results, if available, is also listed. It can be seen that the proposed algorithm is faster than that in [60], but slower than those in [58] and [59]. It is mainly because that the algorithms in [58,59] only search in much smaller spaces and therefore are not capable of finding the optimum solution, nor are they aware of the optimality if optimum solutions happen to be obtained.

Furthermore for [58], generally speaking, multiple runs should be carried out with different pre-defined subexpression basis sets so the actual runtime of the algorithm

in [58] for finding a decent solution could be much longer than that listed in Table 4.4. Multiple runs are also required in [59] to choose a proper number of integers to be searched from the “mid-point” such that the search could generate a favored solution, while the runtime is still acceptable. In addition, it is noted from Table 4.4 that the *EWL* has a significant effect on the total runtime in our proposed algorithm. For instance, increasing the *EWL* of *L3* by 1 results in almost 335 times increase in the runtime. An analysis of the significant effect of *EWL* on the total runtime is given in Section 4.5.

The coefficients of the optimum designs and their corresponding subexpression basis sets are selectively listed in Table 4.9 and Table 4.8 in Appendix B, respectively. For simplicity, all the 1s in subexpression basis sets are omitted. The frequency responses of these benchmark filters can be evaluated using the coefficients in these two tables. It should be pointed out that, the length of the optimum solution of *Y2* given in Table 4.8 is actually 4 less than the design specifications in Table 4.2.

4.4.2 Example B

In this example, the proposed algorithm is further applied on a residual compensated extrapolated impulse response structure proposed in [81]. Different from the conventional transposed direct form, the frequency response of the structure in [81] is expressed as

$$\begin{aligned}
 H(\omega) = & h(0) + \sum_{n=1}^M h(n) \cos(n\omega) \\
 & + \sum_{n=M+Ld+1}^N h(n) \cos(n\omega) + \sum_{n=M+1}^{M+(L-1)d} h_r(n) \cos(n\omega) \\
 & + \sum_{n=M+(L-1)d+1}^{M+Ld} \sum_{l=0}^{L-1} \alpha_l h(n) \cos[(n - (L-1-l)d)\omega],
 \end{aligned} \tag{4.9}$$

where $h(n)$ for $n = 0, 1, \dots, M, M + Ld + 1, \dots, N$ is the filter coefficient that is the same as that in the conventional structure; α_l for $l = 0, 1, \dots, L - 1$ is the pre-fixed scaling factor and $\alpha_0 = 1$; and $h_r(n)$ for $n = M + 1, M + 2, \dots, M + (L - 1)d$ is the residual used to improve the precision of the filter coefficients. The total number of variables to be optimized is equal to the number of the original filter coefficients. Detailed explanation and structures can be found in [81] while in this chapter, we only demonstrate that the proposed algorithm can be used to design filters in such a structure. The optimization problem can be formulated simply by replacing the frequency response $H(\omega)$ in (3) with $H(\omega)$ in (4.9).

Here, benchmark filter *L2* is designed again using this residual compensated extrapolated impulse response structure. The proposed algorithm produces an optimum solution which requires 70 adders in total (56 SA, 12 MB adders and 2 extrapolation adders [81]). Compared with the result in Table 4.9, 3 more adders are reduced. For the same filter with the same structure designed in Table 3.5, the proposed algorithm also results in 3 more adders reduction. The coefficients of the impulse response of the resultant filter is listed in Table 4.6.

4.4.3 Example C

As discussed in the Introduction, the power consumption of FIR filters is very much related to the adder depth. Therefore, it is worthwhile to include adder depth control in the proposed algorithm. This can be easily done by checking the maximum adder depth (MAD) of the current discrete solution before updating the best result (N_{best}) as indicated in Figure 4.2. If the MAD of the current discrete solution exceeds the requirement, this solution is discarded. However, it may happen that the order of the subexpression basis set of the currently discrete solution is uncertain. As a result, the MAD of this discrete solution is uncertain too. In this case, as discussed in Section 4.3.5, it is assumed that the MAD= 2.

Table 4.6: Impulse response of design example $L2$ in residual compensated extrapolated impulse response structure. Coefficients from $h(13)$ to $h(25)$ are chosen as the prototype lobe. Residuals from $h_r(0)$ to $h_r(12)$ and coefficients from $h(13)$ to $h(31)$ are realized in the multiplier block. $h(n) = h_r(n) + \alpha_0 h(n+13)$ for $n = 0, 1, \dots, 12$, where $\alpha_0 = -4$.

Passband gain: 4412.739313, $EWL = 10$								
Subexpression basis set: $\{3, 7, 5, 15, 37, 39, 21, 13, 23, 47, 55, 743\}$								
$h_r(0)$	$=$	1024	$h(13)$	$=$	0	$h(26)$	$=$	-4
$h_r(1)$	$=$	743	$h(14)$	$=$	-48	$h(27)$	$=$	5
$h_r(2)$	$=$	440	$h(15)$	$=$	-64	$h(28)$	$=$	10
$h_r(3)$	$=$	188	$h(16)$	$=$	-48	$h(29)$	$=$	10
$h_r(4)$	$=$	28	$h(17)$	$=$	-13	$h(30)$	$=$	7
$h_r(5)$	$=$	-46	$h(18)$	$=$	21	$h(31)$	$=$	3
$h_r(6)$	$=$	-56	$h(19)$	$=$	39			
$h_r(7)$	$=$	-32	$h(20)$	$=$	37			
$h_r(8)$	$=$	0	$h(21)$	$=$	20			
$h_r(9)$	$=$	23	$h(22)$	$=$	-1			
$h_r(10)$	$=$	30	$h(23)$	$=$	-16			
$h_r(11)$	$=$	21	$h(24)$	$=$	-20			
$h_r(12)$	$=$	0	$h(25)$	$=$	-15			

Hence, our proposed algorithm is modified to take into account the MAD constraint and the benchmark filter $S1$ in Section 4.4.1 is designed with the requirement that MAD is less than or equal to 2 and using $EWL = 7$. The proposed algorithm successfully produces the optimum design using 28 adders. Obviously, the prices paid for the low MAD is 5 more adders used than the design without MAD constraints. The coefficients are listed in Table 4.7.

Table 4.7: Impulse response and subexpression basis set of design examples $S1$ with $MAD = 2$

Passband gain: 325.568345; $h(n) = h(23 - n)$ for $0 \leq n \leq 11$					
Subexpression basis set: $\{3, 5, 7, 65, 15\}$, $EWL = 7$					
$h(0) = 2$	$h(4) = 1$	$h(8) = -24$			
$h(1) = 1$	$h(5) = 10$	$h(9) = 2$			
$h(2) = -3$	$h(6) = 7$	$h(10) = 65$			
$h(3) = -5$	$h(7) = -12$	$h(11) = 120$			

4.5 Computational Complexity

In this section, we briefly discuss the computational complexity of the proposed algorithm. Let N and B be the number of filter coefficients to be optimized and the effective wordlength (EWL), respectively. Apparently, after scaling, the dynamic range of every coefficient is within $[-2^B, 2^B]$. Therefore, for N coefficients, the number of discrete solutions to be investigated is proportional to 2^{NB} . Also, let the average run time of a linear program problem be T . Therefore, the computation time of our proposed algorithm for an optimization problem with N coefficients and $EWL = B$ is:

$$\text{Computation Time} \propto \beta 2^{NB} T, \quad (4.10)$$

where β is a factor influenced by many “cut-off” mechanisms such as the criteria for terminating a branch discussed in Section 4.3.2. From our experience, decent cut-off mechanisms may reduce the computation time by more than 100 times. However, it is clear that the dominant part of (4.10) is the number of coefficients N and B . Therefore, for high-ordered filters or longer EWL , the computational complexity can be too high to be acceptable because of the exponential relationship.

The computation time of benchmark filter $S1$ with different filter orders and benchmark filter $G1$ with different EWL is shown in Figure 4.3 and Figure 4.4, respectively.

The solid line curve in Figure 4.3 shows a drastic increment in design time of $S1$ with the increase of filter orders. However, the increased filter order, while the filter specifications are fixed, results in an increase in the design margins for the passband and stopband ripples. This increase of ripple margin thus leads to larger searching range for the filter coefficients when they are to be quantized.

In order to avoid the possible bias introduced by the enlarged searching range, a new set of filters are designed where the ripple requirements and stopband edge of $S1$ are adopted, but the transition widths are narrowed (by increasing the passband

Figure 4.3: Computation time versus increasing filters orders (a) for $S1$ (solid) and (b) for a set of filters with a fixed ripple margin, but decreasing transition width (dashed). $EWL = 7$.

Figure 4.4: Computation time of $G1$ with different EWL . The filter order is 15.

edge) with increased filter orders, thus the ripple margin of the filter before the coefficients are quantized are comparable. For the data given in the dashed line curve in Figure 4.3, the passband edge for the filter with orders from 24 to 38 are 0.3π , 0.3038π , 0.3280π , 0.3308π , 0.3493π , 0.3515π , 0.3660π and 0.3674π , respectively. The dashed line curve shows that the computation time is slightly increased with increasing filter orders.

It is clear from the above discussions that the later case, where the ripple margin is fixed, is more practical for showing the relationship between design time and filter order. This is because it is usually unnecessary to adopt a much longer order than the minimum required one for a given set of design specifications. In addition, it can be seen that the computation time for this case increases slightly with the increase of filter orders. This is due to the fact that increasing the filter order while keeping the coefficient searching range merely results in a few more constraints in the LP problem. The additional computation complexity contributed by these new constraints are relatively small.

On the other hand, for Figure 4.4, it is noted that increasing EWL results in more boost of computation time than the filter order does (even for the solid line case in Figure 4.3). Moreover, for the test bench filter $G1$, increasing the EWL beyond 13 results in a manual termination after several days.

From the above discussion, it is obvious that the EWL has much more impact on the computation time than the number of coefficients N does (whether it is the variable or the fixed ripple margin case). This can be inferred from (4.10), in which the magnitude of N is usually larger than that of B . Assume that there is an increase of Δ in either value, i.e., $N + \Delta$ or $B + \Delta$. The resultant product of the two values would be $NB + \Delta B$ or $NB + \Delta N$. Since $N \geq B$, we have $\Delta N \geq \Delta B$. Therefore, from the perspective of computational complexity, it is recommended to increase the filter order rather than the EWL to achieve better frequency performance if both are tunable.

4.6 Conclusion

In this chapter, a novel algorithm is proposed for the design of low complexity and low power linear phase FIR filters. The algorithm is based on mixed integer linear programming which traverses through a given integer space, searching for the optimum discrete solution that requires minimum number of adders. An “optimum-awareness” mechanism for checking the optimality of the discrete solutions is introduced into the traverse process with dynamically expanding subexpression spaces. Design examples have shown that the proposed algorithm can produce the optimum discrete solution in most cases. Maximum adder depth constraint can be incorporated into the algorithm and optimum solution is also achievable.

Appendix B: Impulse Responses of Design Example A

Table 4.8: Impulse responses and subexpression basis sets of design examples Y1, Y2, G1 and A1

Filter Y1, $h(n) = h(29 - n)$ for $0 \leq n \leq 14$ Passband gain: 2660.942275, $EWL = 10$ Subexpression basis set: {17, 21, 23, 205, 527, 497}			
$h(0) = -2$	$h(4) = 16$	$h(8) = 84$	$h(12) = 0$
$h(1) = -8$	$h(5) = -21$	$h(9) = 68$	$h(13) = 527$
$h(2) = 0$	$h(6) = -46$	$h(10) = -92$	$h(14) = 994$
$h(3) = 17$	$h(7) = 0$	$h(11) = -205$	
Filter Y2, $h(n) = h(33 - n)$ for $0 \leq n \leq 16$ Passband gain: 5141.653451, $EWL = 11$ Subexpression basis set: {3, 9, 11, 43, 101, 171, 137, 91, 255, 15}			
$h(0) = 6$	$h(5) = 43$	$h(10) = 171$	$h(15) = 1020$
$h(1) = 6$	$h(6) = 36$	$h(11) = 137$	$h(16) = 1920$
$h(2) = -9$	$h(7) = -48$	$h(12) = -182$	
$h(3) = -22$	$h(8) = 101$	$h(13) = -40$	
$h(4) = 0$	$h(9) = 0$	$h(14) = 0$	
Filter G1, $h(n) = h(15 - n)$ for $0 \leq n \leq 7$ Passband gain: 376.999805, $EWL = 7$ Subexpression basis set: {3, 19}			
$h(0) = 3$	$h(2) = 0$	$h(4) = -19$	$h(6) = 76$
$h(1) = 6$	$h(3) = -16$	$h(5) = 12$	$h(7) = 128$
Filter A1, $h(n) = h(58 - n)$ for $0 \leq n \leq 28$ Passband gain: 6007.285543, $EWL = 10$ Subexpression basis set: {7, 9, 5, 21, 29, 15, 39, 177, 23, 131, 171, 393, 311, 823, 63}			
$h(0) = 4$	$h(8) = -29$	$h(16) = 56$	$h(24) = 171$
$h(1) = 7$	$h(9) = -30$	$h(17) = 8$	$h(25) = 393$
$h(2) = 9$	$h(10) = -20$	$h(18) = -601$	$h(26) = 622$
$h(3) = 10$	$h(11) = 0$	$h(19) = -128$	$h(27) = 823$
$h(4) = 7$	$h(12) = 29$	$h(20) = -177$	$h(28) = 960$
$h(5) = 0$	$h(13) = 58$	$h(21) = -184$	$h(29) = 1008$
$h(6) = -10$	$h(14) = 78$	$h(22) = -131$	
$h(7) = -21$	$h(15) = 80$	$h(23) = -10$	

Table 4.9: Impulse responses and subexpression basis sets of design examples $L2$, $L3$, $S1$ and $S2$

Filter $L2$, $h(n) = h(62 - n)$ for $0 \leq n \leq 30$ Passband gain: 4299.842142, $EWL = 10$ Subexpression basis set: $\{3, 13, 9, 5, 19, 35, 15, 49, 51, 85, 203, 31, 79, 371, 339, 911, 499\}$			
$h(0) = 4$	$h(8) = -13$	$h(16) = -64$	$h(24) = -170$
$h(1) = 8$	$h(9) = 0$	$h(17) = -48$	$h(25) = -203$
$h(2) = 12$	$h(10) = 19$	$h(18) = 0$	$h(26) = -124$
$h(3) = 13$	$h(11) = 35$	$h(19) = 60$	$h(27) = 79$
$h(4) = 9$	$h(12) = 36$	$h(20) = 102$	$h(28) = 371$
$h(5) = 0$	$h(13) = 18$	$h(21) = 96$	$h(29) = 678$
$h(6) = -10$	$h(14) = -15$	$h(22) = 32$	$h(30) = 911$
$h(7) = -16$	$h(15) = -49$	$h(23) = -72$	$h(31) = 998$
Filter $L3$, $h(n) = h(35 - n)$ for $0 \leq n \leq 17$ Passband gain: 336.314299, $EWL = 7$ Subexpression basis set: $\{3, 5, 7, 65\}$			
$h(0) = 3$	$h(5) = 0$	$h(10) = -6$	$h(15) = 40$
$h(1) = 0$	$h(6) = 3$	$h(11) = -14$	$h(16) = 65$
$h(2) = -2$	$h(7) = 7$	$h(12) = -16$	$h(17) = 80$
$h(3) = -5$	$h(8) = 8$	$h(13) = -7$	
$h(4) = -5$	$h(9) = 3$	$h(14) = 12$	
Filter $S1$, $h(n) = h(23 - n)$ for $0 \leq n \leq 11$ Passband gain: 239.223425, $EWL = 7$ Subexpression basis set: $\{5, 3, 13, 17\}$			
$h(0) = 2$	$h(3) = -5$	$h(6) = 8$	$h(9) = 0$
$h(1) = 2$	$h(4) = 0$	$h(7) = -12$	$h(10) = 68$
$h(2) = -2$	$h(5) = 10$	$h(8) = -26$	$h(11) = 128$
Filter $S2$, $h(n) = h(59 - n)$ for $0 \leq n \leq 29$ Passband gain: 10738.629695, $EWL = 10$ Subexpression basis set: $\{5, 3, 39, 23, 49, 87, 65, 13, 93, 267, 59, 575, 21, 757, 413, 437, 449\}$			
$h(0) = 5$	$h(8) = -32$	$h(16) = -65$	$h(24) = 575$
$h(1) = 5$	$h(9) = -48$	$h(17) = -26$	$h(25) = 672$
$h(2) = 6$	$h(10) = -64$	$h(18) = 26$	$h(26) = 757$
$h(3) = 5$	$h(11) = -78$	$h(19) = 93$	$h(27) = 826$
$h(4) = 2$	$h(12) = -92$	$h(20) = 174$	$h(28) = 874$
$h(5) = -2$	$h(13) = -98$	$h(21) = 267$	$h(29) = 898$
$h(6) = -10$	$h(14) = -98$	$h(22) = 368$	
$h(7) = -20$	$h(15) = -87$	$h(23) = 472$	

Chapter 5

Linear Phase FIR Filters in Cascade Form

As reviewed in Section 2.4, the power consumption of an FIR filter, is a much complicated issue which largely depends on the adder depth of the detailed implementation of circuit components. Therefore, adder depth is one of the most important criteria for low power and high speed implementation. A straightforward way of achieving low adder depth is to insert pipelines in between the MB adders as shown in Figure 2.14 in Section 2.4.

An alternative attractive approach to achieving lower adder depth as well as lower implementation cost is to implement FIR filters in multi-stages or cascade form. There is a substantial amount of work on cascaded digital filters in early literatures [87–92]. However, algorithms for the design of discrete-valued cascaded FIR filters were rarely found during that period. Existing techniques either fix one subfilters to optimize the other one using MILP [93–97] or jointly optimize the two subfilters using some non-linear algorithms [98] to meet the given specifications. In this chapter, a MILP technique is proposed to simultaneously optimize the coefficients of the two subfilters. The basic idea is to decompose a single discrete coefficient filter into two subfilters whose coefficient values remain discrete numbers. The single discrete coefficient filter is optimized by using MILP, during which coefficient values are fixed one by one. As more and more coefficients are fixed to integers, the possible choices of the decomposed subfilters coefficients becomes less.

Eventually, either a successful decomposition with two subfilters is obtained; or the single discrete coefficient filter cannot be decomposed anymore. The proposed MILP search explores all decomposable filters to find the cascade design using minimum number of adders.

Shown by design examples, the overall implementation complexity of the filters designed using the proposed method is reduced compared with that of single-stage designs. Meanwhile, the EWL (or dynamic range) of the subfilters coefficients is small, resulting in a lower adder depth and smaller width adders in the implementation of the subfilters.

The rest of this chapter is organized as follows. In Section 5.1, the basic strategy of the decomposition of a single-stage discrete coefficient filter into two subfilters is discussed. An algorithm based on MILP is proposed in Section 5.2 to traverse through all the discrete solutions for a given design specification. In Section 5.3, some important observations of the proposed algorithm are discussed. An example is given in Section 5.4 to illustrate the advantages of the proposed algorithm, both in implementation cost and adder depth. In order to design longer filters, in Section 5.5 modifications of the searching manner of the proposed algorithm are given to improve its efficiency. Design examples with relatively longer lengths are given in Section 5.6 to illustrate the efficiency of the modified algorithm.

5.1 Decomposition of Discrete Coefficient Filters

This section explains the basic strategy of decomposing a linear phase FIR filter into two subfilters in cascade form. Throughout this chapter, it is assumed that the filter coefficients, when realized in hardware, are of integers.

Assume that the length of a linear phase FIR filter is $2N + 1$. The corresponding z -transform is thus given by

$$H(z) = \sum_{n=0}^{2N} h(n)z^{-n}, \quad (5.1)$$

where $h(n) = h(2N - n)$ for $n = 0, 1, 2, \dots, N$. Let $H_1(z)$ and $H_2(z)$ be the two symmetric subfilters that $H(z)$ is decomposed into, and N_1, N_2 be the orders of $H_1(z)$ and $H_2(z)$, respectively. $h_1(n) = h_1(N_1 - n)$, for $n = 0, 1, 2, \dots, \lfloor N_1/2 \rfloor$ and $h_2(n) = h_2(N_2 - n)$, for $n = 0, 1, 2, \dots, \lfloor N_2/2 \rfloor$, whereas $\lfloor x \rfloor$ denotes the largest integer less than or equal to x . Obviously, we have $2N = N_1 + N_2$ and $H(z) = H_1(z)H_2(z)$. In addition, $h(n) = 0$ for $n < 0$ or $n > N$, $h_1(n) = 0$ for $n < 0$ or $n > N_1$ and $h_2(n) = 0$ for $n < 0$ or $n > N_2$.

Assume that $N_1 \leq N_2$, the overall filter coefficients $h(n)$, for $n = 0, 1, 2, \dots, N$, can thus be expressed as the convolution of the subfilters coefficients $h_1(n)$ and $h_2(n)$ as:

$$h(n) = \sum_{k=0}^n h_1(k)h_2(n-k), \quad (5.2)$$

when none of the values of $h(n)$ is fixed and $h_1(n)$ and $h_2(n)$ could be of any values. Assume that $h(0)$ is fixed to an integer, say -2 , according to (5.2), we have

$$h(0) = h_1(0)h_2(0). \quad (5.3)$$

For a given EWL of the subfilters, the possible value pairs of $(h_1(0), h_2(0))$ are enumerable. For example, if $\text{EWL} = 3$, $h_1(n)$ and $h_2(n)$ can take integers ranging from -8 to 8 . Define P_{n,v_n} as the set that contains all the possible pairs of $h_1(n)$ and $h_2(n)$ making $h(n)$ equal to v_n . Clearly, if $h(0) = -2$, we have $P_{0,-2} = \{(1, -2), (-2, 1)\}$ which contains two pairs, satisfying (5.3). Therefore, fixing $h(0)$ to -2 narrows the choices of $h_1(0)$ and $h_2(0)$ down to 2 pairs, while all the other

$h_1(n)$ for $n = 1, 2, \dots, \lfloor N_1/2 \rfloor$ and $h_2(n)$ for $n = 1, 2, \dots, \lfloor N_2/2 \rfloor$ are not specified yet.

Since (5.2) can be rewritten as

$$h(n) = h_1(0)h_2(n) + h_1(n)h_2(0) + \sum_{k=1}^{n-1} h_1(k)h_2(n-k), \quad (5.4)$$

it implies that if $h_1(k)$ and $h_2(k)$ for $0 \leq k \leq n-1$ have been determined by fixing the value of $h(k)$ for $0 \leq k \leq n-1$, the possible pairs of $h_1(n)$ and $h_2(n)$ can also be determined when $h(n)$ is fixed to an integer, i.e., P_{n,v_n} is determined once $h(n)$ is fixed to v_n . Following the above example, once we have $P_{0,-2}$ and $h(1)$ is then fixed to an integer, say 11, if we select $h_1(0)$ and $h_2(0)$ as the first pair of $P_{0,-2}$, i.e., $h_1(0) = 1$ and $h_2(0) = -2$, according to (5.4), we have $h(1) = h_2(1) - 2h_1(1)$. Therefore

$$P_{1,11} = \{(-8, -5), (-7, -3), (-6, -1), \\ (-5, 1), (-4, 3), (-3, 5), (-2, 7)\}$$

is generated. If $h_1(0)$ and $h_2(0)$ are selected as the second pair of $P_{0,-2}$, the corresponding $P_{1,11}$ can be generated as

$$P_{1,11} = \{(-5, -8), (-3, -7), (-1, -6), \\ (1, -5), (3, -4), (5, -3), (7, -2)\}.$$

The generation of coefficient pairs continues with more and more filter coefficients $h(n)$ are fixed to certain integers. A tree of P_{n,v_n} for $n = 0, 1, 2, \dots, \lfloor N_1/2 \rfloor$ is thus generated until $h(\lfloor N_1/2 \rfloor)$ is fixed, as shown in part A of Figure 5.1. It is possible that for certain path in Figure 5.1, no possible P_{n,v_n} is available to satisfy (5.4) for a given EWL; those paths are then terminated. Each non-terminated path corresponds to a decomposed subfilters $h_1(n)$ for $n = 0, 1, 2, \dots, \lfloor N_1/2 \rfloor$. Since

$N_1 \leq N_2$, in $h_2(n)$, there may be some more coefficients to be determined. Since subfilters $h_1(n)$ has been determined, (5.4) can be further written as:

$$h(n) = h_1(0)h_2(n) + \sum_{k=1}^n h_1(k)h_2(n-k), \quad (5.5)$$

where $h_1(k) = 0$ for N_1 . (5.5) implies that once all the coefficients of $h_1(n)$ have been determined, the value of $h_2(n)$ can be uniquely determined when $h(n)$ is fixed to an integer. The tree of P_{n,v_n} is extended to part B in Figure 5.1 until $h(\lfloor N_2/2 \rfloor)$ is fixed. Some paths may also be terminated during this procedure. Up to now, both subfilters $h_1(n)$ and $h_2(n)$ are determined. The remaining unfixed coefficients of $h(n)$ can be determined by (5.2) for $n = \lfloor N_2/2 \rfloor + 1, \dots, N$. The decomposition of one single discrete filter is completed. Since numerous single discrete filters can meet a given filter specifications, a MILP search presented in the next section is therefore proposed to search the discrete solutions efficiently.

It should be noted that the overall filter given in (5.1) is assumed to be of even order and symmetric. For other types of filters (odd order, anti-symmetric), similar equations can be derived.

5.2 Proposed Algorithm

In this section, the filter optimization problem is formulated and an algorithm is proposed based on the backgrounds given in Section 5.1.

5.2.1 Problem Formulation

The frequency response of a linear phase FIR filter given in (5.1) can be expressed as:

$$H(\omega) = \sum_{n=0}^{2N} h(n)e^{-jn\omega}. \quad (5.6)$$

Figure 5.1: The decomposition process along a single discrete solution.

Based on (5.6), a linear programming (LP) problem can be formulated as:

$$\text{minimize : } f = \delta - \delta_p b \quad (5.7)$$

$$\text{subject to : } b - \delta \leq H(\omega) \leq b + \delta, \text{ for } \omega \in [0, \omega_p]$$

$$-(\delta_s \delta) / \delta_p \leq H(\omega) \leq (\delta_s \delta) / \delta_p, \text{ for } \omega \in [\omega_s, \pi]$$

$$b_l \leq b \leq b_u, \quad (5.8)$$

where f is the objective function, δ is the ripple, δ_p , δ_s , ω_p and ω_s are the given passband ripple, stopband ripple, passband edge and stopband edge, respectively. $H(\omega)$ is given in (5.6). b_l and b_u are the lower bound and upper bound of the passband gain, respectively. b_l and b_u could be chosen as 0.7 and 1.4, respectively, as discussed in [47].

5.2.2 Traverse of Discrete Solutions

The above optimization problem finds the continuous optimal solution of the overall filter. During the integer programming part, filter coefficients are to be fixed to integers one by one while the rest unfixed ones are optimized to compensate for the loss in frequency response. The traverse of discrete solutions is conducted in a recursive way and the whole search process is terminated when all possible discrete solutions are investigated. For a better understanding of the traverse process of MILP, [52] is to be referred.

The proposed algorithm is also mainly based on the recursive search process in [52]. However, in order to incorporate the decomposition procedures discussed in Section 5.1, the following operations are carried out:

1. For every fixed $h(n)$, $n = 0$, P_{0,v_n} is generated according to (5.3) discussed in Section 5.1.

2. For every fixed $h(n)$, for $1 \leq n \leq \lfloor N_1/2 \rfloor$ and every pair in $P_{n-1, v_{n-1}}$, a corresponding P_{n, v_n} is to be further generated according to (5.4)
3. For $\lfloor N_1/2 \rfloor < n \leq \lfloor N_2/2 \rfloor$, once $h(n)$ is fixed, for every path $h_2(n)$ is determined using (5.5).
4. If $n > \lfloor N_2/2 \rfloor$, it is unnecessary to fix $h(n)$ one by one anymore. Instead, all the rest unfixed coefficients $h(n)$ for $n = \lfloor N_2/2 \rfloor, \dots, N$ are computed at one time using (5.2) and one linear programming is required to check if the discrete solution meets the specifications. Therefore, in the proposed algorithm, the “depth” of the traverse is $\lfloor N_2/2 \rfloor$, instead of the number of the variable of the overall filter N as shown in [52].

5.3 Some Remarks

In this section, some important observations of the proposed algorithm are discussed.

1. The effective wordlength (EWL) of the subfilters is predefined so as to confine the magnitudes of $h_1(n)$ and $h_2(n)$ in generating all the possible coefficient pairs in generating P_{n, v_n} . It should be noted that larger EWL results in more combinations of $h_1(n)$ and $h_2(n)$ but also heavier computational complexity.
2. During the MILP search, all the possible integers of $h(n)$ that would result in a feasible solution should be investigated. In order to facilitate the search, the lower bound and upper bound of $h(n)$ for $n = 0, 1, \dots, N$ might be computed before fixing $h(n)$ to certain integers. The bound can be found by using linear programming.
3. When $h(0)$ is fixed at the very beginning of the MILP search, the pairs of $h_1(0)$ and $h_2(0)$ are to be generated. It should be noted that the sign of $h_1(0)$ or $h_2(0)$ does not affect the final design of the subfilters. In other words, for instance, the pair $h_1(0) = -1, h_2(0) = 2$ and the pair $h_1(0) = 1, h_2(0) = -2$ would lead

to the same design regardless of their signs. The proposed algorithm can be accelerated by simply discarding either of the above pairs in $P_{0,-2}$, which is shown in Figure 5.1 where $P_{0,-2}$ only contains two pairs instead of four.

4. From our design experience, the orders of the subfilters, N_1 and N_2 , are of much importance in the following two factors. First, they have significant impact on the computational complexity of the algorithm. It is observed that the computation time increases dramatically as the difference between N_1 and N_2 decreases. For instance, it takes around 1 minute to fully run the proposed algorithm when $N_1 = 2$ and $N_2 = 13$ in the design example given in the next section. However, when $N_1 = 5$ and $N_2 = 10$, the whole computation time rises to 20 minutes. This huge increase in computational complexity is due to the generation of P_{n,v_n} . Apparently, according to (5.4), for larger N_1 , more filter coefficients $h(n)$ are to be used in generating P_{n,v_n} . For example, if $N_1 = 2$, only P_{0,v_0} and P_{1,v_1} are generated when $h(0)$ and $h(1)$ are fixed to integers according to the proposed algorithm; however if $N_1 = 6$, P_{0,v_0} , P_{1,v_1} , P_{2,v_2} and P_{3,v_3} will be generated, which implies more loops or branches that the algorithm has to execute.
5. It is observed that the number of feasible designs tends to drop as N_1 increases (or when N_1 and N_2 tend to be equal to each other). It is further noted that for every specific design, there exists an $\tilde{N}_1$, increasing beyond which does not result in any feasible design. It should be pointed out that in the above discuss, the order of the total filter is fixed so that when N_1 increase, N_2 decreases. Also, N_1 is kept smaller than N_2 in order to cut off duplicate designs because $N_1 = 2, N_2 = 10$ yields the same design as $N_1 = 10, N_2 = 2$ does.

5.4 An Example

In this section, an example is used to show the effectiveness of the proposed algorithm in achieving low implementation complexity and low adder depth. Due to the lack of concrete design specifications in the references, an example, designed in single stage, is selected to illustrate the merits of the proposed algorithm. This example is taken from [38], i.e., a low pass filter with length of 16, passband edge ω_p and stopband edge ω_s are 0.2π and 0.5π , respectively. Passband ripple δ_p and stopband ripple δ_s are both 0.01. In this case, we increase the filter order by one and set the orders of the subfilters as $N_1 = 4$ and $N_2 = 12$, respectively. The EWL of each subfilters is set to 4.

The proposed algorithm successfully finds several pairs of $H_1(z)$ and $H_2(z)$, among which one design that uses minimum number of adders is selected. Here, the algorithm proposed in [33] is employed to compute the number of adders used to synthesize the multiplier block.

Table 5.1: Impulse responses of design Example A, $H(z) = H_1(z)H_2(z)$.

subfilters $H_1(z)$, $h_1(n) = h_1(4 - n)$ for $0 \leq n \leq 2$					
$h_1(0)$	=	1	$h_1(1)$	=	2
$h_1(2)$	=	2			
subfilters $H_2(z)$, $h_2(n) = h_2(12 - n)$ for $0 \leq n \leq 6$					
$h_2(0)$	=	1	$h_2(1)$	=	0
$h_2(2)$	=	-2			
$h_2(3)$	=	-3	$h_2(4)$	=	0
$h_2(5)$	=	9			
$h_2(6)$	=	12			
Overall filter $H(z)$, $h(n) = h(16 - n)$ for $0 \leq n \leq 8$					
Passband gain: 177.410954					
$h(0)$	=	1	$h(1)$	=	2
$h(2)$	=	0			
$h(3)$	=	-5	$h(4)$	=	-9
$h(5)$	=	-1			
$h(6)$	=	22	$h(7)$	=	48
$h(8)$	=	60			

The coefficients of subfilters $H_1(z)$, $H_2(z)$ and the overall filter $H(z)$ are listed in Table 5.1. It should be noted that $h(n)$ is exactly the convolution of $h_1(n)$ and $h_2(n)$. In addition, it can be seen that if the filter is implemented using a single-

Figure 5.2: Magnitude responses of $H(z)$ (solid), $H_1(z)$ (dotted) and $H_2(z)$ (dashed).

stage transposed direct form, i.e., $H(z)$ is directly realized, the total number of adders used is 19. However, when decomposing $H(z)$ into two subfilters, the total number of adders required to synthesize $H_1(z)$ and $H_2(z)$ is 14, which is 5 less than the direct realization. Furthermore, the maximum adder depths (MAD) of $H_1(z)$ and $H_2(z)$ are 1 and 2, respectively and is less than that of $H(z)$. It should also be noted that, the best single-stage design for this example also achieve 15 total number of adders [38]. However, our design has a smaller overall EWL of 6 (compared with 7 in [38]). Therefore, in this example, the cascade form filter design produced by the proposed algorithm is superior to the direct single-stage design in both implementation complexity and adder depth (or even EWL). The magnitude responses of $H_1(z)$, $H_2(z)$ and $H(z)$ are plotted in Figure 5.2.

Figure 5.3 shows an implementation of $H_1(z)$ and $H_2(z)$, where an extra delay element is inserted between the subfilters as indicated. Obviously, without the

Figure 5.3: Implementation of $H_1(z)$ and $H_2(z)$ in cascade form.

extra delay, the MAD or critical path of this circuit would be larger as shown in Figure 5.3. One alternatively way of circumventing this issue is to realize $H_2(z)$ in direct structure. However, the multiplier block of the direct structure has larger adder depth than the transposed direct structure, which might result in lower circuit speed and higher power consumption.

5.5 Improving Searching Efficiency

Although the algorithm proposed above can successfully find the discrete-valued cascade design, its efficiency is rather low and can not be used to design longer filters (with order more than 20). In this section, we shall examine the cause of inefficiency and try to improve it.

The main redundancy of the aforementioned algorithm lies in the repeated runs of MILP for the same coefficient. This is illustrated in the first two parts marked as A and B in Figure 5.1. Let h_1 and h_2 be the coefficients of subfilter 1 and subfilter 2, respectively.

For $k = 0, \dots, \lfloor \frac{N_1}{2} \rfloor$, suppose that the overall filter coefficient $h(k)$ is quantized to v_k , according to the algorithm, the set that contains all the possible subfilters coefficient pairs are generated, i.e., P_{k,v_k} is generated. And what the algorithm

does is that: for every pair in P_{k,v_k} , fix the subfilters coefficients $h_1(k)$ and $h_2(k)$ as that pair, say pair 0, and go one step deeper to quantize the next overall filter coefficient $h(k+1)$. It is clear that several MILP runs need to be executed in order to verify if quantizing $h(k+1)$ to certain integers would satisfy the design specification. However, when the algorithm traces back to fix $h_1(k)$ and $h_2(k)$ to another pair in P_{k,v_k} , say pair 1, the algorithm again goes one step deeper into the quantization of $h(k+1)$. The same MILP runs will be executed to check the feasibility of fixing $h(k+1)$ to some integers. Obviously, these MILP runs are unnecessary. The reason is because: although $h_1(k)$ and $h_2(k)$ are fixed to different values when different pairs in P_{k,v_k} are selected, in both cases $h(k)$ is fixed to the same value v_k ; the MILP runs for checking the feasibility of fixing $h(k+1)$ when pair 1 is selected for $h(k)$ are the same as those when pair 0 is selected. This is the major redundancy in Part A in Figure 5.1. And it is clear that the number of redundant MILP runs grows exponentially with N_1 , which is the reason why (as pointed out in the fourth remark in Section 5.3) the algorithm takes more and more time to run as N_1 is increased.

For $k = \lfloor \frac{N_1}{2} \rfloor + 1, \dots, \lfloor \frac{N_2}{2} \rfloor$, redundancy exists in a similar way. Suppose that $h(k)$ is fixed to an integer. Since all the coefficients in subfilter 1 are already fixed (i.e., h_1 is totally determined), the value of $h_2(k)$ can be uniquely calculated once $h(k)$ is determined according to (5.5). If $h_2(k)$ is an integer, then the algorithm goes down to the quantization process of $h(k+1)$. However, once the algorithm traces back to Part A in Figure 5.1 and brings another set of h_1 and h_2 to Part B (with h_2 partially determined), all the filter coefficients $h(k)$ in Part B are to be quantized again to the same values that they have already been quantized to earlier and MILP runs are needed to check feasibility. This is the major redundancy in Part B and the number of redundant MILP runs is proportional to the number of coefficients sets (h_1 and h_2), which in turn grows exponentially with N_1 .

Therefore, the number of extra linear programming problems can be significantly more than what is necessary for the searching process due to the two parts mentioned

above. For instance, for a 24-order filter, the single-stage optimization algorithm proposed in Chapter 4 requires a total number of about 4000 MILP runs while the total number of MILP runs is more than 4 million for a 3-order and 21-order cascade design.

5.5.1 Change of Searching Manner

The manner of searching is modified in this section so that the number of MILP runs is of the same order of the single-stage design algorithm in Chapter 4. In order to achieve this goal, the searching manner should be modified as follows.

For $k = 0, \dots, \lfloor \frac{N_1}{2} \rfloor$, the algorithm traverses the discrete space of the filter coefficients $h(k)$ same as the single-stage design algorithm in Chapter 4. When the search along a path reaches $h(\lfloor \frac{N_1}{2} \rfloor)$, i.e., the coefficients from $h(0)$ to $h(\lfloor \frac{N_1}{2} \rfloor)$ have been fixed one by one by reoptimizing the rest unfixed coefficients, the discrete coefficient values along the path are recorded. $h(\lfloor \frac{N_1}{2} \rfloor + 1)$ is to be fixed.

For $k = \lfloor \frac{N_1}{2} \rfloor + 1$, find all the feasible sets of h_1 and h_2 (h_1 completely determined at this point while h_2 partially determined) that satisfy (5.5) for a certain integer value of $h(\lfloor \frac{N_1}{2} \rfloor + 1)$. More specifically, the algorithm does the following steps:

1. According to the procedure described in Section 5.1, generate all the possible sets of h_1 and h_2 satisfying $h(i) = v_i$ using (5.3) and (5.4), where v_i for $i = 0, 1, \dots, \lfloor \frac{N_1}{2} \rfloor$ are the fixed coefficient values obtained in the above tree search. Then $h(\lfloor \frac{N_1}{2} \rfloor + 1)$ is fixed to an integer value.
2. For all the sets generated in Step 1, find the sets that satisfy (5.5) for the fixed value of $h(\lfloor \frac{N_1}{2} \rfloor + 1)$ and record them in a pool $Q(\lfloor \frac{N_1}{2} \rfloor + 1)$.
3. Do an MILP run to check the feasibility.
4. If feasible, then go to the quantization of $h(\lfloor \frac{N_1}{2} \rfloor + 2)$. Else, find another integer value for $h(\lfloor \frac{N_1}{2} \rfloor + 1)$ and go to Step 2.

It should be noted that same as discussed in Chapter 4, if both the upper bound and lower bound have been exceeded, the node is terminated and the algorithm traces back to the upper nodes.

For $\lfloor \frac{N_1}{2} \rfloor + 1 < k \leq \lfloor \frac{N_2}{2} \rfloor$, when a coefficient $h(k)$ is fixed to an integer, by applying (5.5), only those sets of h_1 and h_2 in $Q(k-1)$ that satisfy (5.5) will be inherited and stored in $Q(k)$. Then the algorithm goes one step deeper. As more and more coefficients are fixed, the number of possible sets of h_1 and h_2 at each level decreases.

For $k > \lfloor \frac{N_2}{2} \rfloor$, since all the coefficient of h_1 and h_2 are fixed, for all the sets in $Q(k-1)$, calculate their convolutions for $h(k)$ and check if the frequency response specifications are satisfied. This part is actually the same as that in Section 5.2.2.

The above searching manner is based on the traverse of the “tree” of the overall filter coefficients, instead of the searching manner discussed in Section 5.2, which is based on the “tree” of subfilters coefficient pairs. As a result, using the modified algorithm, the total number of MILP runs (which consume the major part of the computation time) is kept as the same order as that of the single-stage optimization algorithm. The main overhead of the modified algorithm lies in generating all the possible sets of h_1 and h_2 when $k = \lfloor \frac{N_1}{2} \rfloor + 1$. If N_1 is large, the number of iterations required to generate these sets is also large. However, compared with the time it takes to do MILP runs, this overhead is negligible.

5.5.2 Including Dynamically Expanding Subexpression Basis Set

In order to further speed up the searching process, the concept of dynamically expanding subexpression basis set is incorporated into the algorithm. The major difference from Chapter 4 is that here two basis sets are required for h_1 and h_2 , respectively. As the decomposition proceeds, the subexpression basis sets of h_1 and h_2 grow accordingly while the numbers of adders used to realize their coefficients

are monitored. The basic mechanism is the same as that in Chapter 4, except that here the sum of the orders of the two basis sets are compared with the best result ever achieved. Let b_1 and b_2 be the orders of the corresponding basis sets of h_1 and h_2 , respectively, and let N_{best} be the total number of adders used to implement the multiplier block adders of the best solution so far. Then, when $h(k)$ for $k \geq \lfloor \frac{N_1}{2} \rfloor + 1$ is fixed, in finding all the possible sets of h_1 and h_2 from $Q(k-1)$, as discussed in Section 5.5.1, the criterion $b_1 + b_2 < N_{best}$ becomes a condition to terminate the decomposition procedure of the search in Part B.

5.6 Design Examples

In the section, the capacity of the improved algorithm in designing relatively longer filters is illustrated. Nine benchmark filters taken from literature are designed. Most of the filters are the same as those in the design example A in Chapter 4. For convenience, the specifications are re-listed in Table 5.2 and Table 5.3, respectively. It should be noted that the filter marked as Y3 is the second example in [60]. ω_p , ω_s , δ_p and δ_s are the passband edge, stopband edge, passband ripple and stopband ripple, respectively. In Table 4.3, δ_p^+ and δ_p^- are the upper bound and lower bound of the passband gain, respectively. For all the filters, every possible order combination of the subfilters has been tried.

The design results are listed in Table 5.4, where N_i , MBA_i and SA_i are the order, number of multiplier adders and number of structural adders of subfilters i , respectively. SS represents the single-stage optimum solution using the algorithm proposed in Chapter 4. EWL is the effective wordlength of the overall filter. It can be seen that for filters G1, L3, Y3, A1, S2 and L2, the proposed algorithm outperforms the single-stage optimum design. The maximum reductions in the total numbers of adders designed in cascade form for those filters are 1, 1, 1, 6, 7 and 6 adders, respectively. Also, for filters S1 and Y2, the cascade form design

Table 5.2: Specifications of design examples. ω_p , ω_s , δ_p and δ_s are the passband edge, stopband edge, passband ripple and stopband ripple, respectively.

Filters	Filter length	ω_p	ω_s	δ_p	δ_s
$L2$	63	0.2π	0.28π	0.028	0.001
$S1$	24	0.3π	0.5π	0.0157	0.0066
$S2$	60	0.042π	0.14π	0.012	0.001
$Y1$	30	0.3π	0.5π	0.00316	0.00316
$Y2$	38	0.3π	0.5π	0.001	0.001
$Y3$	50	0.4π	0.51π	0.01	0.001
$G1$	16	0.2π	0.5π	0.01	0.01
$A1$	59	0.125π	0.225π	0.01	0.001

Table 5.3: Specifications of $L3$. δ_p^+ and δ_p^- are the upper bound and lower bound of the passband, respectively. The filter length is 36.

Bands(π)	δ_p^+ (db)	δ_p^- (db)	δ_s (db)
$0 \sim 0.15$	0.2	-0.2	-
$0.15 \sim 0.1875$	0.2	-0.5	-
$0.1875 \sim 0.2125$	0.2	-0.9	-
$0.2875 \sim 1$	-	-	-30

does not yield any reduction in the total number of adders. Furthermore, for $Y1$ the total number of adders realized using cascade form surpasses the single-stage design. It should also be noted that, for the combination of $N_1 = 2$ and $N_2 = 27$, no feasible discrete solutions were found for $Y2$. However, for $N_1 = 3$ and $N_2 = 26$, there exist feasible solutions. Nevertheless, for every specific filter, there does exist a combination of filter orders beyond which there are no feasible solutions.

Furthermore, it can be noted that the possible combinations of subfilters is rather limited, especially for filters with longer lengths. This is because most of the filters are designed using minimum lengths for the given EWL. Therefore, it is not likely to decompose them into small subfilters that have different order combinations. It is also interesting to note that some of the cases (where $h_1(0) = h_1(1) = 1$) turn out to be the standard difference FIR filters.

Table 5.4: Summary of design results. N_i , MBA_i and SA_i are the order, number of multiplier adders and number of structural adders of subfilter i , respectively. SS represents the optimum single-stage design using the algorithm proposed in Chapter 4 and EWL is the effective wordlength of the overall filter.

Filters	EWL	N_1	N_2	MBA_1	SA_1	MBA_2	SA_2	Total	SS
$G1$	6	1	14	0	1	3	12	16	17
		2	13	0	2	3	11	16	
		3	12	0	3	3	10	16	
$S1$	7	1	22	0	1	4	18	23	23
		2	21	0	1	4	19	24	
		3	20	0	3	3	18	24	
		4	19	1	4	3	19	27	
		5	18	2	5	2	16	25	
$Y1$	9	1	28	0	1	6	24	31	30
		2	27	-	-	-	-	-	
		3	26	0	3	5	26	34	
$Y2$	10	1	36	0	1	10	36	47	47
		2	35	0	2	12	33	47	
		3	34	0	3	10	34	47	
$L3$	6	1	34	0	1	5	30	36	37
$Y3$	10	1	48	0	1	12	48	60	61
		2	47	0	2	15	45	60	
$A1$	10	1	57	0	1	12	55	68	69
		2	56	0	2	9	52	63	
		3	55	0	3	9	53	65	
$S2$	10	1	58	0	1	14	54	69	76
		2	57	0	2	15	53	70	
$L2$	10	1	61	0	1	11	55	67	73
		2	60	0	2	12	54	68	

In order to see the effect of filter length on the cascade form design, we increased the orders of $G1$, $S1$ and $L3$ by one and re-designed them using all possible combinations. The results are listed in Table 5.5.

It can be seen that, increasing the filter length by one results in not only increased possible combinations but also even better reduction in the total number of adders. For $G1$, the best solution requires only 12 adders to implement the whole filter, which is 25% reduction compared with the single-stage optimum design with the same length. For $S1$, the total number of adders of the best result is also reduced from 23 to 21. For $L3$, only 27 adders are required in the best case to implement

Table 5.5: Summary of design results of $G1$, $S1$ and $L3$ with increased filter lengths. N_i , MBA_i and SA_i are the order, number of multiplier adders and number of structural adders of subfilters i , respectively. SS represents the optimum single-stage design using the algorithm proposed in Chapter 4 and EWL is the effective wordlength of the overall filter.

Filters	EWL	N_1	N_2	MBA_1	SA_1	MBA_2	SA_2	Total	SS
$G1$	6	1	15	0	1	2	13	16	16
		2	14	0	2	2	8	12	
		3	13	0	3	4	13	20	
		4	12	0	4	2	8	14	
$S1$	7	1	23	0	1	3	21	25	25
		2	22	0	2	3	16	21	
		3	21	0	3	2	19	24	
		4	20	1	4	3	16	24	
		5	19	2	5	2	15	24	
		6	18	2	6	2	14	24	
		7	17	1	7	3	15	26	
$L3$	6	1	35	0	1	4	27	32	32
		2	34	0	2	3	22	27	
		3	33	0	3	3	25	31	
		4	32	0	4	3	26	33	
		5	31	0	5	3	25	33	
		6	30	1	6	3	24	34	

the cascade form realization, which is 5 less than the single-stage design and 10 less than that in Table 5.4. It should be pointed out that the EWL listed in Table 5.4 and Table 5.5 is the effective wordlength of the overall filter, while the EWL for the subfilters is much smaller. As reviewed earlier in Section 2.4, reduced wordlength of filter coefficients is beneficial for achieving lower power consumption and higher circuit throughput.

The impulse responses of all the design examples (with the combination that produce the least total number of adders) are listed in Appendix C for verification.

5.7 Conclusion

In this chapter, an algorithm is proposed for the design of linear phase FIR filters in cascade form with low complexity and low adder depth. The proposed algorithm

decomposes the overall filter into subfilters during the traverse of MILP search of the overall filter. The conventional recursive search process is modified to cope with the decomposition procedures. In order to further improve the efficiency of the proposed algorithm, the searching manner is modified so as to keep the number of MILP iterations as the same order as that of a single-stage algorithm. Dynamically expanding subexpression basis sets are introduced to speed up the traverse and keep track of the total number of adders. Design examples are given to illustrate the superiority of the algorithm in achieving both low implementation complexity and low adder depth. Several important observations concerning computational complexity and choice of subfilters effective wordlength and orders are discussed.

Appendix C: Impulse Responses of Design Examples

Table 5.6: Impulse responses of design examples Y1 and Y2

Filter Y1: $h_1(0) = h_1(1) = 1$, $h_2(n) = h_2(28 - n)$ for $0 \leq n \leq 14$, $h(n) = h(29 - n)$ for $0 \leq n \leq 14$. Passband gain: 827.425810, $EWL = 9$.			
$h_1(0) = 1$	$h_2(3) = 4$	$h_2(7) = 8$	$h_2(11) = -32$
$h_2(0) = -1$	$h_2(4) = 0$	$h_2(8) = 19$	$h_2(12) = 35$
$h_2(1) = -1$	$h_2(5) = -8$	$h_2(9) = 0$	$h_2(13) = 130$
$h_2(2) = 2$	$h_2(6) = -6$	$h_2(10) = -31$	$h_2(14) = 176$
$h(0) = -1$	$h(4) = 4$	$h(8) = 27$	$h(12) = 3$
$h(1) = -2$	$h(5) = -8$	$h(9) = 19$	$h(13) = 165$
$h(2) = 1$	$h(6) = -14$	$h(10) = -31$	$h(14) = 306$
$h(3) = 6$	$h(7) = 2$	$h(11) = -63$	
Filter Y2: $h_1(n) = h_1(3 - n) = 1$ for $0 \leq n \leq 1$, $h_2(n) = h_2(34 - n)$ for $0 \leq n \leq 17$, $h(n) = h(37 - n)$ for $0 \leq n \leq 18$. Passband gain: 2530.445200, $EWL = 10$.			
$h_1(0) = 1$	$h_2(3) = 1$	$h_2(8) = -11$	$h_2(13) = -64$
$h_1(1) = 1$	$h_2(4) = -11$	$h_2(9) = 54$	$h_2(14) = -177$
$h_2(0) = -2$	$h_2(5) = -9$	$h_2(10) = -17$	$h_2(15) = -42$
$h_2(1) = 1$	$h_2(6) = 16$	$h_2(11) = 76$	$h_2(16) = 275$
$h_2(2) = 5$	$h_2(7) = 27$	$h_2(12) = 80$	$h_2(17) = 444$
$h(0) = -2$	$h(5) = -14$	$h(10) = -55$	$h(15) = -203$
$h(1) = -1$	$h(6) = -3$	$h(11) = -6$	$h(16) = -8$
$h(2) = 4$	$h(7) = 23$	$h(12) = 85$	$h(17) = 500$
$h(3) = 5$	$h(8) = 23$	$h(13) = 75$	$h(18) = 952$
$h(4) = -4$	$h(9) = -22$	$h(14) = -85$	

Table 5.7: Impulse response of design example Y3

Filter Y3: $h_1(n) = h_1(2 - n) = 1$ for $0 \leq n \leq 1$, $h_2(n) = h_2(47 - n)$ for $0 \leq n \leq 23$, $h(n) = h(49 - n)$ for $0 \leq n \leq 24$. Passband gain: 2363.828914, $EWL = 10$.			
$h_1(0) = 1$	$h_2(5) = 8$	$h_2(12) = -20$	$h_2(19) = 35$
$h_1(1) = 1$	$h_2(6) = -4$	$h_2(13) = 26$	$h_2(20) = -116$
$h_2(0) = 3$	$h_2(7) = -13$	$h_2(14) = 36$	$h_2(21) = -123$
$h_2(1) = 1$	$h_2(8) = 0$	$h_2(15) = -21$	$h_2(22) = 132$
$h_2(2) = -6$	$h_2(9) = 20$	$h_2(16) = -59$	$h_2(23) = 419$
$h_2(3) = -4$	$h_2(10) = 7$	$h_2(17) = 3$	
$h_2(4) = 7$	$h_2(11) = -26$	$h_2(18) = 86$	
$h(0) = 3$	$h(7) = -9$	$h(14) = 42$	$h(21) = -204$
$h(1) = 4$	$h(8) = -17$	$h(15) = 41$	$h(22) = -107$
$h(2) = -2$	$h(9) = 7$	$h(16) = -44$	$h(23) = 428$
$h(3) = -9$	$h(10) = 27$	$h(17) = -77$	$h(24) = 970$
$h(4) = -3$	$h(11) = 1$	$h(18) = 30$	
$h(5) = 11$	$h(12) = -39$	$h(19) = 124$	
$h(6) = 11$	$h(13) = -20$	$h(20) = 5$	

Table 5.8: Impulse response of design example A1

Filter A1: $h_1(n) = h_1(2 - n) = 1$ for $0 \leq n \leq 1$, $h_2(n) = h_2(56 - n)$ for $0 \leq n \leq 28$, $h(n) = h(58 - n)$ for $0 \leq n \leq 29$. Passband gain: 6067.190254, $EWL = 10$.			
$h_1(0) = 1$	$h_2(6) = -10$	$h_2(14) = 14$	$h_2(22) = -12$
$h_1(1) = 2$	$h_2(7) = -4$	$h_2(15) = 23$	$h_2(23) = 48$
$h_2(0) = 4$	$h_2(8) = -14$	$h_2(16) = -5$	$h_2(24) = 92$
$h_2(1) = 1$	$h_2(9) = 0$	$h_2(17) = -8$	$h_2(25) = 167$
$h_2(2) = 2$	$h_2(10) = -7$	$h_2(18) = -41$	$h_2(26) = 204$
$h_2(3) = 3$	$h_2(11) = 14$	$h_2(19) = -41$	$h_2(27) = 256$
$h_2(4) = -3$	$h_2(12) = 8$	$h_2(20) = -56$	$h_2(28) = 253$
$h_2(5) = 0$	$h_2(13) = -28$	$h_2(21) = -31$	
$h(0) = 4$	$h(8) = -32$	$h(16) = 55$	$h(24) = 176$
$h(1) = 6$	$h(9) = -32$	$h(17) = 5$	$h(25) = 399$
$h(2) = 8$	$h(10) = -21$	$h(18) = -62$	$h(26) = 630$
$h(3) = 8$	$h(11) = 0$	$h(19) = -131$	$h(27) = 831$
$h(4) = 5$	$h(12) = 29$	$h(20) = -179$	$h(28) = 969$
$h(5) = -3$	$h(13) = 58$	$h(21) = -184$	$h(29) = 1018$
$h(6) = -13$	$h(14) = 78$	$h(22) = -130$	
$h(7) = -24$	$h(15) = 79$	$h(23) = -7$	

Table 5.9: Impulse response of design example $S2$

Filter $S2$: $h_1(0) = h_1(1) = 1$, $h_2(n) = h_2(56 - n)$ for $0 \leq n \leq 29$, $h(n) = h(59 - n)$ for $0 \leq n \leq 29$. Passband gain: 11597.187435, $EWL = 10$.				
$h_1(0) = 1$	$h_2(7) = -16$	$h_2(15) = -42$	$h_2(23) = 282$	
$h_2(0) = 6$	$h_2(8) = -20$	$h_2(16) = -25$	$h_2(24) = 340$	
$h_2(1) = 0$	$h_2(9) = -32$	$h_2(17) = -1$	$h_2(25) = 385$	
$h_2(2) = 5$	$h_2(10) = -37$	$h_2(18) = 32$	$h_2(26) = 432$	
$h_2(3) = 0$	$h_2(11) = -48$	$h_2(19) = 71$	$h_2(27) = 458$	
$h_2(4) = 2$	$h_2(12) = -50$	$h_2(20) = 120$	$h_2(28) = 484$	
$h_2(5) = -5$	$h_2(13) = -55$	$h_2(21) = 170$	$h_2(29) = 484$	
$h_2(6) = -6$	$h_2(14) = -50$	$h_2(22) = 229$		
$h(0) = 6$	$h(8) = -36$	$h(16) = -67$	$h(24) = 622$	
$h(1) = 6$	$h(9) = -52$	$h(17) = -26$	$h(25) = 725$	
$h(2) = 5$	$h(10) = -69$	$h(18) = 31$	$h(26) = 817$	
$h(3) = 5$	$h(11) = -85$	$h(19) = 103$	$h(27) = 890$	
$h(4) = 2$	$h(12) = -98$	$h(20) = 191$	$h(28) = 942$	
$h(5) = -3$	$h(13) = -105$	$h(21) = 290$	$h(29) = 968$	
$h(6) = -11$	$h(14) = -105$	$h(22) = 399$		
$h(7) = -22$	$h(15) = -92$	$h(23) = 511$		

Table 5.10: Impulse response of design example $L2$

Filter $L2$: $h_1(0) = h_1(1) = 1$, $h_2(n) = h_2(61 - n)$ for $0 \leq n \leq 30$, $h(n) = h(62 - n)$ for $0 \leq n \leq 31$. Passband gain: 4377.325640, $EWL = 10$.				
$h_1(0) = 1$	$h_2(7) = -15$	$h_2(15) = -34$	$h_2(23) = -65$	
$h_2(0) = 3$	$h_2(8) = -9$	$h_2(16) = -33$	$h_2(24) = -108$	
$h_2(1) = 4$	$h_2(9) = 0$	$h_2(17) = -16$	$h_2(25) = -100$	
$h_2(2) = 6$	$h_2(10) = 12$	$h_2(18) = 17$	$h_2(26) = -28$	
$h_2(3) = 3$	$h_2(11) = 18$	$h_2(19) = 48$	$h_2(27) = 108$	
$h_2(4) = 0$	$h_2(12) = 15$	$h_2(20) = 60$	$h_2(28) = 272$	
$h_2(5) = -8$	$h_2(13) = 0$	$h_2(21) = 43$	$h_2(29) = 423$	
$h_2(6) = -12$	$h_2(14) = -18$	$h_2(22) = -6$	$h_2(30) = 512$	
$h(0) = 3$	$h(8) = -24$	$h(16) = -67$	$h(24) = -173$	
$h(1) = 7$	$h(9) = -9$	$h(17) = -49$	$h(25) = -208$	
$h(2) = 10$	$h(10) = 12$	$h(18) = 1$	$h(26) = -128$	
$h(3) = 9$	$h(11) = 30$	$h(19) = 65$	$h(27) = 80$	
$h(4) = 3$	$h(12) = 33$	$h(20) = 108$	$h(28) = 380$	
$h(5) = -8$	$h(13) = 15$	$h(21) = 103$	$h(29) = 695$	
$h(6) = -20$	$h(14) = -18$	$h(22) = 37$	$h(30) = 935$	
$h(7) = -27$	$h(15) = -52$	$h(23) = -71$	$h(31) = 1024$	

Table 5.11: Impulse response of design example G1

Filter G1: $h_1(n) = h_1(2 - n) = 1$ for $0 \leq n \leq 1$, $h_2(n) = h_2(14 - n)$ for $0 \leq n \leq 7$, $h(n) = h(16 - n)$ for $0 \leq n \leq 8$. Passband gain: 150.622864, $EWL = 6$.			
$h_1(0) = 1$	$h_2(1) = 0$	$h_2(4) = 0$	$h_2(7) = 12$
$h_1(1) = 2$	$h_2(2) = 0$	$h_2(5) = 2$	
$h_2(0) = 1$	$h_2(3) = -4$	$h_2(6) = 14$	
$h(0) = 1$	$h(3) = -4$	$h(6) = 18$	
$h(1) = 2$	$h(4) = -8$	$h(7) = 42$	
$h(2) = 1$	$h(5) = -2$	$h(8) = 52$	

Table 5.12: Impulse responses of design examples S1 and L3

Filter S1: $h_1(n) = h_1(2 - n) = 1$ for $0 \leq n \leq 1$, $h_2(n) = h_2(22 - n)$ for $0 \leq n \leq 11$, $h(n) = h(24 - n)$ for $0 \leq n \leq 12$. Passband gain: 316.554754, $EWL = 7$.			
$h_1(0) = 1$	$h_2(2) = 0$	$h_2(6) = 5$	$h_2(10) = 32$
$h_1(1) = 2$	$h_2(3) = -4$	$h_2(7) = -12$	$h_2(11) = 30$
$h_2(0) = 1$	$h_2(4) = 5$	$h_2(8) = -2$	
$h_2(1) = -1$	$h_2(5) = 0$	$h_2(9) = 0$	
$h(0) = 1$	$h(4) = -3$	$h(8) = -21$	$h(12) = 124$
$h(1) = 1$	$h(5) = 6$	$h(9) = -16$	
$h(2) = -1$	$h(6) = 10$	$h(10) = 30$	
$h(3) = -5$	$h(7) = -2$	$h(11) = 94$	
Filter L3: $h_1(n) = h_1(2 - n) = 1$ for $0 \leq n \leq 1$, $h_2(n) = h_2(34 - n)$ for $0 \leq n \leq 17$, $h(n) = h(36 - n)$ for $0 \leq n \leq 18$. Passband gain: 246.813736, $EWL = 6$.			
$h_1(0) = 1$	$h_2(3) = 0$	$h_2(8) = 0$	$h_2(13) = 2$
$h_1(1) = 1$	$h_2(4) = 0$	$h_2(9) = 0$	$h_2(14) = 3$
$h_2(0) = 2$	$h_2(5) = -2$	$h_2(10) = 0$	$h_2(15) = 15$
$h_2(1) = 0$	$h_2(6) = 2$	$h_2(11) = -8$	$h_2(16) = 20$
$h_2(2) = -3$	$h_2(7) = 4$	$h_2(12) = -4$	$h_2(17) = 20$
$h(0) = 2$	$h(5) = -2$	$h(10) = 0$	$h(15) = 20$
$h(1) = 1$	$h(6) = 0$	$h(11) = -8$	$h(16) = 38$
$h(2) = -1$	$h(7) = 4$	$h(12) = -12$	$h(17) = 55$
$h(3) = -3$	$h(8) = 6$	$h(13) = -10$	$h(18) = 60$
$h(4) = -3$	$h(9) = 4$	$h(14) = 1$	

Chapter 6

MCM Implementation of Polyphase Linear Phase FIR filters with Restored Symmetry

As reviewed in Chapter 2, polyphase structures have wide applications in multirate systems such as decimators, interpolators and filter banks [5]. The advantage of a polyphase structure is the efficiently reduced arithmetic complexity and data storage. Several researches [73, 99, 100] have investigated the multiplierless implementation of the polyphase structure to further reduce the computational complexity. In these techniques, the filter coefficient multiplications are collectively considered as a MCM problem, where the multiplication of one data with several constant coefficients is realized by using shared shifter and addition blocks.

The storage efficient polyphase implementations for decimators (in transposed direct form) and interpolators (in direct form), denoted as Type I and Type II structures [9], are shown in Figure 6.1(a) and Figure 6.1(b), respectively. In these figures a 22nd order filter is decomposed to its 5 polyphase representation. Either structure of Type I or Type II has its own transposed counterpart implementation, denoted as Type III and Type IV structures, respectively. Although in Type III and Type IV structures, a single MCM block could be shared by all polyphase subfilters [101], it requires a factor M more storage elements than that required by Type I and Type II structures in an M polyphase implementation. In [73, 99], the MCM implementations of the above 4 structures are investigated. It is shown that

Figure 6.1: Polyphase structure for (a) Type I polyphase for decimator and (b) Type II polyphase for interpolator. T is a unit delay.

although Type III and IV structures use less adders [73, 99], the silicon areas are larger than that of Types I and II due to the large number of storage elements. In the implementation of Type I and II structures in [73, 99], the problem is formulated as one MCM block for each subfilter, or as a matrix MCM block for all subfilters. However, the coefficient symmetry of the linear phase FIR filters is not exploited in either approach.

Recent studies [102–104] show that the coefficient symmetry can be restored in the polyphase implementation of Type I and Type II structures without any increase in storage elements.

In this chapter, the MCM implementation of the polyphase structure for Type I and Type II are investigated, when the coefficient symmetry is exploited. In the proposed technique, each subfilter uses one separate MCM block. The number of adders used for the synthesis of polyphase subfilters, as well as the adder depth in the implementation of decimators, of the proposed implementation is compared with the conventional polyphase implementations, where the coefficient symmetry is not utilized. Results show that the number of adders used in synthesizing the proposed polyphase filter is significantly reduced compared with the conventional implementations [73, 99] of Type I and Type II structures with a slight increase in adder depth. The proposed technique could also be extended to formulate the problem using a matrix MCM block.

6.1 Polyphase Structures with Restored Coefficient Symmetry

Assume that a symmetric FIR filter of length N has the z -transform given by

$$H(z) = \sum_{n=0}^{N-1} h(n)z^{-n}, \quad (6.1)$$

where $h(n) = h(N - 1 - n)$ for $n = 0, 1, 2, \dots, N - 1$. When $H(z)$ is decomposed into M polyphase subfilters, P subfilters have filter length $\lceil N/M \rceil$ and the other $M - P$ subfilters have filter length $\lfloor N/M \rfloor$, where $P = N - \lfloor N/M \rfloor M$. $\lceil x \rceil$ denotes the smallest integer greater than or equal to x and $\lfloor x \rfloor$ denotes the largest integer

less than or equal to x . $H(z)$ can thus be written as

$$H(z) = \sum_{m=0}^{P-1} z^{-m} H_m^L(z^M) + \sum_{m=P}^{M-1} z^{-m} H_m^S(z^M), \quad (6.2)$$

where $H_m^L(z)$ and $H_m^S(z)$ are the z -transforms of the subfilters with lengths of $\lceil N/M \rceil$ and $\lfloor N/M \rfloor$, respectively, and are given by

$$\begin{aligned} H_m^L(z) &= \sum_{n=0}^{\lceil N/M \rceil} h(m+nM)z^{-n} \\ H_m^S(z) &= \sum_{n=0}^{\lfloor N/M \rfloor} h(m+nM)z^{-n}. \end{aligned} \quad (6.3)$$

It has been shown in [102–104] that the subfilters of $H_m^L(z)$ for $m = 0, 1, \dots, \lfloor P/2 \rfloor - 1$, $\lfloor (P+1)/2 \rfloor, \dots, P-1$ and $H_m^S(z)$ for $m = P, P+1, \dots, P + \lfloor (M-P)/2 \rfloor - 1$, $P + \lfloor (M-P+1)/2 \rfloor, \dots, M-1$ are asymmetric. This implies that the subfilter $H_m^L(z)$ has symmetric coefficients only when P is odd and $m = \lfloor P/2 \rfloor$, whereas subfilter $H_m^S(z)$ has symmetric coefficients only when $M-P$ is odd and $m = P + \lfloor (M-P)/2 \rfloor$. However, it is further shown in [103, 104] that the subfilter pairs of $H_m^L(z)$ and $H_{P-1-m}^L(z)$ have mirror image coefficients for $m = 0, 1, \dots, \lfloor P/2 \rfloor - 1$. Similarly, $H_m^S(z)$ and $H_{M-1-m}^S(z)$ have mirror image coefficients for $m = P, P+1, \dots, P + \lfloor (M-P)/2 \rfloor - 1$. Thus, if transforms

$$h'_m(n) = [h_m(n) + h_{P-1-m}(n)]/2 \quad (6.4)$$

$$h'_{P-1-m}(n) = [h_m(n) - h_{P-1-m}(n)]/2 \quad (6.5)$$

for $m = 0, 1, \dots, \lfloor P/2 \rfloor - 1$, $n = 0, 1, \dots, \lceil N/M \rceil$, and

$$h'_m(n) = [h_m(n) + h_{M-1-m}(n)]/2 \quad (6.6)$$

$$h'_{M-1-m}(n) = [h_m(n) - h_{M-1-m}(n)]/2 \quad (6.7)$$

for $m = P, P + 1, \dots, P + \lfloor (M - P)/2 \rfloor - 1$, $n = 0, 1, \dots, \lfloor N/M \rfloor$ are employed, $H(z)$ in (6.2) can be rewritten as

$$\begin{aligned} H(z) = & \sum_{m=0}^{\lfloor P/2 \rfloor - 1} z^{-m} \left[(1 + z^{-(P-1-2m)}) H_m'^L(z^M) \right. \\ & \left. + (1 - z^{-(P-1-2m)}) H_{P-1-m}'^L(z^M) \right] \\ & + \sum_{m=P}^{P + \lfloor (M-P)/2 \rfloor - 1} z^{-m} \left[(1 + z^{-(M-1-2m)}) H_m'^S(z^M) \right. \\ & \left. + (1 - z^{-(M-1-2m)}) H_{M-1-m}'^S(z^M) \right] \\ & + \text{symmetric } H_m^L(z) \text{ and } H_m^S(z) \text{ terms,} \end{aligned} \quad (6.8)$$

where

$$H_m'^L(z) = \sum_{n=0}^{\lfloor N/M \rfloor} h_m'(n) z^{-n} \quad (6.9)$$

for $m = 0, 1, \dots, P - 1$ and

$$H_m'^S(z) = \sum_{n=0}^{\lfloor N/M \rfloor} h_m'(n) z^{-n} \quad (6.10)$$

for $m = P, P + 1, \dots, M - 1$, and the symmetric $H_m^L(z)$ and $H_m^S(z)$ terms are given by

$$\left\{ \begin{array}{ll} 0 & \text{if both } P \text{ and } M - P \text{ are even;} \\ z^{-\lfloor P/2 \rfloor} H_{\lfloor P/2 \rfloor}^L(z^M) & (11) \\ & \text{if } P \text{ is odd and } M - P \text{ is even;} \\ z^{-(P + \lfloor (M-P)/2 \rfloor)} H_{P + \lfloor (M-P)/2 \rfloor}^S(z^M) & (12) \\ & \text{if } P \text{ is even and } M - P \text{ is odd;} \\ (11) + (12) & \text{if both } P \text{ and } M - P \text{ are odd.} \end{array} \right.$$

In (6.9) and (6.10), $H_m'^L(z)$ and $H_m'^S(z)$ have either symmetric or anti-symmetric coefficients due to the transformations (6.4)–(6.7).

Figure 6.2: The implementation of Type I polyphase by exploiting the coefficient symmetry. T is a unit delay.

Based on the above transformations, the polyphase structure of the decimator in Figure 6.1(a) can be realized by exploiting the coefficient symmetry as shown in Figure 6.2. In this example, the number of longer subfilters, i.e., P is equal to 3 while the number of shorter subfilters $M - P$ is equal to 2. Since P is odd whereas $M - P$ is even, $H_1^L(z)$ has symmetric coefficients while all the other subfilters have asymmetric coefficients. By applying the transformations (6.4)–(6.7), each subfilter has either symmetric or anti-symmetric coefficients and thus can be implemented as shown in Figure 6.2. In the implementation of the decimator in Figure 6.2, each subfilter employs an MCM block to synthesize the subfilter coefficients, as indicated. The adders along the delay chain are called structural adders in the transposed direct form filters, as shown in Figure 6.2. In addition to the MCM block adders and structural adders, it is also noted from Figure 6.2 that extra adders are required next to the down samplers to add or subtract the signals, corresponding

to the transformations (6.4)–(6.7). The number of the extra adders, referred to as transformation adders as indicated in Figure 6.2, is given by $2(\lfloor P/2 \rfloor + \lfloor (M-P)/2 \rfloor)$.

The Type II interpolator implementation that utilizes the coefficient symmetry is a transposed structure of that in Figure 6.2. The total number of adders for Type I decimator and Type II interpolator are the same when a filter is decomposed to M polyphase subfilters. However, in the structure of Type II interpolator, the adders are not differentiated as the MCM block adders and structural adders.

6.2 Complexities

As is discussed in Section 6.1, for Type I decimator and Type II interpolator the total number of adders for the proposed structures are the same. Therefore, in this section only the Type I decimator structure in Figure 6.2 is used to compare with the conventional Type I implementation in Figure 6.1(a). Since the structural adders for both conventional and proposed Type I implementations are the same, the total number of adders except the structural adders are compared. Hence, in the conventional Type I implementation, the number of MCM block adders is defined as the number of adders used to implement the multipliers, whereas in the proposed Type I implementation, the number of MCM block adders is defined as the number of adders used to implement the multipliers plus the transformation adders. In addition, the adder depths of the two implementations are also compared.

Since there have been many MCM algorithms concerning the reduction of adders in MCM block [23, 26, 33, 37, 44, 54], the RAG- n algorithm in [33] is used in this chapter to synthesize the MCM block of each subfilter.

6.2.1 Fixed Filter Length with Varying M

In this example, a linear phase FIR filter with the length of 150 is synthesized using *remez.m* in Matlab and rounded arbitrarily to 13 bit precision, excluding the sign

Figure 6.3: Total number of adders used to realize the MCM blocks with varying M and fixed filter length

bit. The filter is then decomposed into M subfilters with M varying from 2 to 16. The MCM block of each subfilter is synthesized using RAG- n for the conventional and the proposed Type I decimator implementations. The curves in Figure 6.3 show the adder count in the synthesis of all the MCM blocks of the polyphase filters. In addition, for the lower curve, i.e., the proposed Type I decimator in Figure 6.2, the number of the transformation adders is also included. It is observed that as M increases, the number of adders for both of the implementations increases. This is because the number of coefficients within each subfilter decreases as M increases, resulting in less sharing of the partial products among the coefficients in each MCM block. Also, for the proposed implementation, the number of transformation adders increases with M . However, the total number of adders used to implement the MCM blocks for the proposed Type I implementation is approximately one third less than that of the conventional Type I implementation. Clearly, in terms of number of adders, the proposed implementation brings significant reduction.

Figure 6.4: Total number of adders used to realize the MCM blocks with varying filter lengths and fixed $M = 4$.

6.2.2 Fixed M with Varying Filter Length

In this example, the MCM blocks of the polyphase decomposition of 13 linear phase FIR filters with different lengths ranging from 30 to 150 are synthesized in the same way as that of the examples in Section 6.2.1. Each filter is decomposed into 4 subfilters. The number of adders used to synthesize the MCM blocks for both implementations is shown in Figure 6.4. In addition, the number of adders used to synthesize an entire MCM block for Type III decimator is also shown for reference. It can be seen from Figure 6.4 that as the filter length increases, the reduction of adders by exploiting the coefficient symmetry becomes more significant. This can be explained by assuming that applying the MCM results in the same reduction factor for both implementations, i.e., the number of adders used to synthesize the conventional and proposed structure is N/C and $0.5N/C$, respectively, where N is the filter length and C is the reduction factor. Obviously, the difference between the two structures, in terms of the number of adders, is proportional to N .

It can also be seen that synthesizing an entire MCM block of Type III implementation always results in less adders than synthesizing the MCM blocks of each subfilter separately in Type I implementation. However, as stated at the beginning

Figure 6.5: Adder depth of conventional and proposed Type I decimators with varying filter length and fixed $M = 4$.

of this chapter, Type III implementation requires a large number of storage elements and therefore uses more silicon area even than the Type I implementation without utilizing the coefficient symmetry. Furthermore, it has been shown in [68, 99] that the Type I implementation can be formulated as a matrix MCM algorithm, and the required number of adders is much fewer than that of the implementation where each subfilter uses a MCM block separately.

6.2.3 Adder Depth

Figure 6.5 shows the comparison of the maximum adder depth (MAD) and average adder depth of the conventional and proposed Type I decimator implementations with $M = 4$. The adder depth is defined as the number of adders that a signal goes through before reaching a delay element. The corresponding structural adders are organized to minimize the adder depth. While MAD decides the throughput of the circuit, the power consumption is very much related to adder depth. It can be seen that by exploiting coefficient symmetry, the MAD and adder depth of the proposed

structure is generally longer than the original polyphase structure because it is shown in Figure 6.2 that the actual input signals of the subfilters are the sum and difference of the down sampled signals instead of the original down sampled ones. As reported in [105], larger adder depth usually results in higher power dissipation and lower throughput of the circuit. However, since the proposed structure can achieve more than one third MCM block adders reduction, the actual power consumption of the circuit due to the overall effect of increased adder depth and reduced adders is not clear in the system level.

6.3 Conclusion

In this chapter, the MCM implementation of Type I and Type II polyphase structures with coefficient symmetry is investigated. The mirror image nature of the subfilters are utilized to construct pairs of new subfilters with symmetric and anti-symmetric coefficients. The coefficient symmetry within each new subfilter results in more than one third reduction in MCM block adders compared with the conventional polyphase implementations. In spite of the slight increase in the adder depth, the reduction in the number of adders is significant.

Chapter 7

Conclusion

In this thesis, various techniques for designing low power and low complexity linear phase FIR filters from structural and algorithmic level have been developed.

Different from the conventional transposed direct form of direct form implementation, an extrapolated impulse response with residual compensation is proposed for synthesizing linear phase FIR filters using MCM techniques. This structure is developed based on the observation that the impulse response of an FIR filter exhibits a quasi-periodicity property, in which lobes with large magnitudes can be approximated by scaled versions lobes with small magnitudes. Residuals compensation is introduced to restore the impulse response loss caused by the approximation. As a result, the dynamic range of the coefficient values and residuals are reduced. This reduced dynamic range decreases the lower bound of the number of adders to synthesize the multiplier block. To further reduce the complexity of the structure, MILP is formulated to optimize the filter coefficients and the residuals. It is shown, by examples, that FIR filters implemented using the proposed structure can be synthesized using with reduced complexity and the effect of applying this structure tends to be more significant provided that the length of the filter is long or the quasi-periodicity is obvious.

In addition, a novel algorithm is proposed for the goal of finding the globally optimum discrete design, which corresponds to a transposed direct form synthesis using minimum number of adders. The algorithm is based on mixed integer linear

programming tree search which traverses through a given integer space, searching for the optimum discrete solution that requires minimum number of adders. An “optimum-awareness” mechanism for checking the optimality of the discrete solutions is introduced into the traverse process with dynamically expanding subexpression spaces. It is shown by design examples that the proposed algorithm can produce the optimum discrete solution in most cases. Furthermore, maximum adder depth (MAD) constraint can be incorporated into the algorithm and optimum solution is also achievable.

Other than using pipelines in real circuit implementation to achieve low adder depth while sacrificing complexity and group delay, an algorithm for the design of cascade form linear phase FIR filters with low complexity and low adder depth is proposed. In the searching process of MILP, the proposed algorithm decomposes the overall filter with larger coefficients magnitudes into subfilters with smaller magnitudes. It is observed that by splitting a single filter into several subfilters, it is possible to achieve both lower complexity and smaller adder depth, which is of much practical importance. In order to reduce the high computational complexity, an improved version of the algorithm is proposed for the design of longer filters. Filters with relatively longer lengths were designed. Results have shown that in most cases, realizing filters in cascade form produced by the algorithm can result in even lower complexity and smaller wordlength than the single-stage optimum design.

Finally, for polyphase FIR filters implementation, the MCM implementation of Type I and Type II polyphase structures, with restored coefficient symmetry is investigated. The mirror image nature of the subfilters are utilized to construct pairs of new subfilters with symmetric and anti-symmetric coefficients. In spite of the minor increase in adder depth, the number of adders is significantly reduced.

Bibliography

- [1] Y. C. Lim and S. R. Parker, “Discrete coefficient FIR digital filter design based upon an LMS criteria,” *IEEE Trans. Circuits Syst.*, vol. 30, pp. 723–739, Oct. 1983.
- [2] N. H. E. Weste and K. Eshraghian, “Principles of CMOS VLSI design: A systems perspective.” Reading, MA: Addison-Wesley, 1993.
- [3] T. Saramäki, “Finite impulse response filter design,” in *Handbook for digital signal processing*, S. K. Mitra and J. F. Kaiser, Eds. New York: John Wiley and Sons, 1993, ch. 4.
- [4] K. K. Parhi, *VLSI digital signal processing system, design and implementation*. John Wiley & Sons, 1999.
- [5] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Prentice-Hall, 1993.
- [6] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Prentice-Hall, 1983.
- [7] P. P. Vaidyanathan and P. Q. Hoang, “Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction qmf banks,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, pp. 81–94, Jan. 1988.
- [8] T. Saramäki and S. K. Mitra, “Multiple branch FIR filters for sampling rate conversion,” in *Proc. IEEE ISCAS’92*, 1992, pp. 1007–1010.

- [9] S. K. Mitra, *Digital Signal Processing, A Computer-Based Approach*. McGraw-Hill, 1998, ch. 10.
- [10] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial," *Proc. IEEE*, vol. 78, pp. 56–93, Jan. 1990.
- [11] P. Arian, T. Saramäki, and S. K. Mitra, "A systematic technique for optimization multiple branch FIR filters for sampling rate conversion," in *Proc. IEEE ISCAS'02*, vol. IV, Phoenix, AZ, May 2002, pp. 1–4.
- [12] S. K. Mitra, A. Mahalanobis, and T. Saramäki, "Generalized structural sub-band decomposition of FIR filters and its application in efficient FIR filter design and implementation," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 363–374, June 1993.
- [13] Y. C. Lim and A. G. Constantinides, "Linear phase FIR digital filter without multipliers," in *Proc. IEEE ISCAS'79*, 1979, pp. 185–189.
- [14] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete power-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-31, pp. 583–591, June 1983.
- [15] Q. Zhao and Y. Tadakoro, "A simple design of FIR filters with power-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 566–570, May 1988.
- [16] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044–1047, July 1989.
- [17] Y. C. Lim, J. B. Evans, and B. Liu, "Decomposition of binary integers into signed power-of-two terms," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 667–672, June 1991.

- [18] H. Shaffeu, M. M. Jones, H. D. Griffiths, *et al.*, “Improved design procedure for multiplierless FIR digital filters,” *Electron. Lett.*, vol. 27, pp. 1142–1144, June 1991.
- [19] B.-R. Horng, H. Samueli, and A. N. Willson, Jr., “The design of two-channel lattice structure perfect-reconstruction filter banks using power-of-two coefficients,” *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 497–499, July 1993.
- [20] Y. C. Lim, R. Yang, D. N. Li, *et al.*, “Signed power-of-two term allocation scheme for the design of digital filters,” *IEEE Trans. Circuits Syst. II*, vol. 46, pp. 577–584, May 1999.
- [21] Y. J. Yu, “Multiplierless multirate FIR filter design and implementation,” Ph.D. dissertation, National University of Singapore, Singapore, Nov. 2003.
- [22] Y. J. Yu and Y. C. Lim, “Roundoff noise analysis of signals represented using signed power-of-two terms,” *IEEE Trans. Signal Process.*, vol. 55, pp. 1409–1416, May 2007.
- [23] D. R. Bull and D. H. Horrocks, “Primitive operator digital filters,” *IEE Proc. G*, vol. 138, pp. 401–412, June 1991.
- [24] R. I. Hartley, “Optimization of canonic signed digit multipliers for filter design,” *Proc. IEEE Inf. Symp. Circuits Syst., Singapore*, pp. 1992–1995, June 1991.
- [25] R. I. Hartley and A. E. Casavant, “Optimizing pipelined networks of associative and commutative operators,” *IEEE Trans. Computer-aided Design of Integrated Circuits and Systems*, vol. 13.
- [26] R. I. Hartley, “Subexpression sharing in filters using canonic signed digit multipliers,” *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 677–688, Oct. 1996.

- [27] M. Potkonjak, M. B. Shrivasta, and A. P. Chandrakasan, "Multiple constant multiplication: Efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, pp. 151–161, Feb. 1996.
- [28] N. Sankarayya and K. Roy, "Algorithms for lower power and high speed FIR filter realization using differential coefficients," *IEEE Trans. Circuits Syst. II*, vol. 43, pp. 488–497, 1997.
- [29] N. Sankarayya, K. Roy, and D. Bhattacharya, "Algorithms for lower power and high speed FIR filter realization using differential coefficients," *10th Int. Conf. VLSI Design*, pp. 174–178, Jan. 1997.
- [30] I.-C. Park and H.-J. Kang, "Digital filter synthesis based on an algorithm to generate all minimal signed digit representation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, pp. 1525–1529, Dec. 2002.
- [31] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proc. Circuits, Devices, Syts.*, vol. 141, pp. 407–413, Oct. 1994.
- [32] —, "Multiplication by an integer using minimum adders," *IEE Mathematical Aspects of Digital Signal Processing Colloquium*, Feb. 1994.
- [33] —, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 569–577, Sept. 1995.
- [34] M. Martinez-Peiró, E. Boemo, and L. Wanhammar, "Design of high-speed multiplierless filters using a nonrecursive signed common subexpression algorithm," *IEEE Trans. Circuits Syst. I*, vol. 49, pp. 196–203, Mar. 2002.

- [35] R. Paško, P. Schaumont, V. Derudder, *et al.*, “A new algorithm for elimination of common subexpressions,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, pp. 58–68, Jan. 1999.
- [36] O. Gustafsson and L. Wanhammar, “ILP modeling of the common subexpression sharing problem,” in *Proc. IEEE ICECS’02*, Dubrovnic, Croatia, 2002, pp. 1171–1174.
- [37] —, “Improved multiple constant multiplication using minimum spanning trees,” in *Proc. Asilomar Conf. Sigals, Syst., Comp.*, 2004, pp. 63–66.
- [38] —, “Design of linear-phase FIR filters combining subexpression sharing with MILP,” in *Proc. MWSCAS’02*, 2002, pp. 9–12.
- [39] S.-F. Lin, S.-C. Huang, F.-S. Yang, *et al.*, “Power-efficient FIR filter architecture design for wireless embedded system,” *IEEE Trans. Circuits Syst. II*, vol. 51, pp. 21–25, Jan. 2004.
- [40] A. G. Dempster and M. D. Macleod, “Generation of signed-digit representations for integer multiplication,” *IEEE Signal Process. Lett.*, vol. 11, pp. 663–665, Aug. 2004.
- [41] C.-Y. Yao, H.-H. Chen, T.-F. Lin, *et al.*, “A novel common-subexpression-elimination method for synthesizing fixed-point FIR filters,” *IEEE Trans. Circuits Syst. I*, vol. 51, pp. 2215–2221, Nov. 2004.
- [42] H. Choo, K. Muhammad, and K. Roy, “Complexity reduction of digital filters using shift inclusive differential coefficients,” *IEEE Trans. Signal Process.*, vol. 52, pp. 1760–1772, June 2004.
- [43] A. P. Vinod, E. M.-K. Lai, A. B. Premkuntar, *et al.*, “FIR filter implementation by efficient sharing of horizontal and vertical common subexpressions,” *Electronics Letters*, vol. 39, pp. 251–253, Jan. 2003.

- [44] Y. Voronenko and M. Püschel, “Multiplierless multiple constant multiplication,” *ACM Trans. Algorithms*, vol. 3, pp. 1–38, May 2007.
- [45] O. Gustafsson, “Lower bounds for constant multiplication problems,” *IEEE Trans. Circuits Syst. II*, vol. 54, pp. 974–978, Nov. 2007.
- [46] J. J. Nielsen, “Design of linear-phase direct-form digital filters with quantized coefficients using error spectrum shaping techniques,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, pp. 1020–1026, July 1989.
- [47] Y. C. Lim, “Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude,” *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1480–1486, Dec. 1990.
- [48] D. Li, J. Song, and Y. C. Lim, “A polynomial-time algorithm for designing digital filters with power-of-two coefficients,” in *Proc. IEEE ISCAS’93*, May 1993, vol. 1, pp. 84–87.
- [49] T. W. Fox and L. E. Turner, “The design of peak-constrained least square fir filters with low-complexity finite-precision coefficient,” *IEEE Trans. Circuits Syst. II*, vol. 49, pp. 151–154, Feb. 2002.
- [50] C. K. C. J. H. Lee and Y. C. Lim, “Design of discrete coefficient FIR digital filters with arbitrary amplitude and phase responses,” *IEEE Trans. Circuits Syst.*, vol. 40, pp. 444–448, July 1993.
- [51] A. Lee, M. Ahmadi, G. A. Jullien, *et al.*, “Digital filter design using genetic algorithm,” in *Proc. IEEE Symp. Advances in Digital Filtering and Signal Processing*, June 1998, pp. 34–38.
- [52] Y. C. Lim and Y. J. Yu, “A width-recursive depth-first tree search approach for the design of discrete coefficient perfect reconstruction lattice filter bank,” *IEEE Trans. Circuits Syst. II*, vol. 50, pp. 257–266, June 2003.

- [53] Y. C. Lim, Y. Sun, and Y. J. Yu, "Design of discrete-coefficient FIR filters on loosely connected parallel machines," *IEEE Trans. Signal Process.*, vol. 50, pp. 1409–1416, June 2002.
- [54] F. Xu, C.-H. Chang, and C.-C. Jong, "Contention resolution algorithm for common subexpression elimination in digital filter design," *IEEE Trans. Circuits Syst. II*, vol. 52, pp. 695–700, Oct. 2005.
- [55] —, "A new integrated approach to the design of low-complexity FIR filters," in *Proc. IEEE ISCAS'06*, Kos, Greece, 2006, pp. 601–604.
- [56] —, "Design of low-complexity FIR filters based on signed-powers-of-two coefficients with reusable common subexpressions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, pp. 1898–1907, Oct. 2007.
- [57] J. Yli-Kaakinen and T. Saramäki, "A systematic algorithm for the design of multiplierless FIR filters," in *Proc. IEEE ISCAS'01*, Sydney, Australia, 2001, pp. 185–188.
- [58] Y. J. Yu and Y. C. Lim, "Design of linear phase FIR filters in subexpression space using mixed integer linear programming," *IEEE Trans. Circuits Syst. I*, vol. 54, pp. 2330–2338, Oct. 2007.
- [59] M. Aktan, A. Yurdakul, and G. Dünder, "An algorithm for the design of low-power hardware-efficient fir filters," *IEEE Trans. Circuits Syst. I*, vol. 55, pp. 1536 – 1545, July 2008.
- [60] Y. J. Yu and Y. C. Lim, "Optimization of linear phase FIR filters in dynamically expanding subexpression space," *Circuit Syst. Signal Process*, vol. 29, pp. 65 – 80, Feb. 2010.

- [61] C.-L. Chen and A. N. Willson, Jr., "A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients," *IEEE Trans. Circuits Syst. II*, vol. 46, pp. 29–39, Jan. 1999.
- [62] A. Fuller, B. Nowrouzian, and F. Ashrafzadeh, "Optimization of FIR digital filters over the canonical signed-digit coefficient space using genetic algorithms," in *Proc. Midwest Symp. Circuits Syst.*, Aug. 1998, pp. 456–459.
- [63] Y. J. Yu and Y. C. Lim, "A novel genetic algorithm for the design of a signed power-of-two coefficient quadrature mirror filter lattice filter bank," *Circuit Syst. Signal Process.*, vol. 21, pp. 263–276, May/June 2002.
- [64] R. Cemes and D. Ait-Boudaoud, "Genetic approach to design of multiplierless FIR filters," *Electron. Lett.*, vol. 29, pp. 2090–2091, Nov. 1993.
- [65] D. Ait-Boudaoud and R. Cemes, "Modified sensitivity criterion for the design of powers-of-two FIR filters," *Electron. Lett.*, vol. 29, pp. 1467–1469, Aug. 1993.
- [66] N. Benvenuto, M. Marchesi, and A. Uncini, "Applications of simulated annealing for the design of special digital filters," *IEEE Trans. Signal Process.*, vol. 40, pp. 323–332, Feb. 1992.
- [67] A. G. Dempster, O. Gustafsson, and J. O. Coleman, "Towards an algorithm for matrix multiplier blocks," in *Proc. European Conf. Circuit Theory Design*, 2003.
- [68] O. Gustafsson, J. O. Coleman, A. G. Dempster, *et al.*, "Low-complexity hybrid form FIR filters using matrix multiple constant multiplication," in *Proc. Asilomar Conf. Signals, Syst., Comp.*, 2004, pp. 77–80.

- [69] D. N. Li, Y. C. Lim, Y. Lian, *et al.*, “A polynomial-time algorithm for design digital filters with power-of-two coefficients,” *IEEE Trans. Signal Process.*, vol. 50, pp. 1935–1941, Aug. 2002.
- [70] P. Flores, J. Monteiro, and E. Costa, “An exact algorithm for the maximal sharing of partial terms in multiple constant multiplications,” *Proc. IEEE Int. Conf. Computer Aided Design*, 2005.
- [71] N. Boullis and T. Tisserand, “Some optimizations of hardware multiplication by constant matrices,” *IEEE Trans. Comput.*, vol. 54, pp. 1271–1282, Oct. 2005.
- [72] O. Gustafsson, “A difference based adder graph heuristic for multiple constant multiplication problems,” in *Proc. IEEE ISCAS’07*, New Orleans, LA, 2007, pp. 1097–1100.
- [73] O. Gustafsson and A. G. Dempster, “On the use of multiple constant multiplication in polyphase FIR filters and filter banks,” in *Proc. IEEE NORISIG’04*, June 2004, pp. 53–56.
- [74] A. N. Willson, Jr., “Desensitized half-band filters,” *IEEE Trans. Circuits Syst. I*, vol. 57, pp. 152 – 165, Jan. 2010.
- [75] C. Y. Yao and C. J. Chien, “A partial MILP algorithm for the design of linear phase FIR filters with SPT coefficients,” *IEICE Trans. Fundamentals*, vol. E85-A, pp. 2302–2310, 2002.
- [76] S. S. Demirsoy, A. G. Dempster, and İ. Kale, “Transition analysis on FPGA for multiplier-block based fir filter structures,” in *Proc. IEEE ICECS’00*, vol. 2, Dec. 2000, pp. 862–865.

- [77] K. Johansson, O. Gustafsson, and L. Wanhammar, "Switching activity estimation for shift-and-add based constant multipliers," in *Proc. IEEE ISCAS'08*, 2008, pp. 676–679.
- [78] A. Blad and O. Gustafsson, "Integer linear programming-based bit-level optimization for high-speed FIR decimation filter architectures," *Circuits, Systems and Signal Processing*, vol. 29, pp. 81–101, Dec. 2010.
- [79] H. J. Kang, H. Kim, and I. C. Park, "FIR filter synthesis algorithms for minimizing the delay and the number of adders," in *Proc. IEEE/ACM Int. Conf. on Computer Aided Design*, Nov. 2000, pp. 51–55.
- [80] A. G. Dempster, S. S. Demirsoy, and I. Kale, "Designing multiplier blocks with low logic depth," in *Proc. IEEE ISCAS'02*, vol. 5, May 2002, pp. 773–776.
- [81] Y. J. Yu, D. Shi, and Y. C. Lim, "Subexpression encoded extrapolated impulse response FIR filter with perfect residual compensation," in *Proc. IEEE ISCAS'08*, 2008, pp. 2446 – 2449.
- [82] Y. C. Lim and B. Liu, "Extrapolated impulse response FIR filters," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1548–1551, Dec. 1990.
- [83] Y. J. Yu, K. L. Teo, Y. C. Lim, *et al.*, "Extrapolated impulse response filter and its application in the synthesis of digital filters using the frequency-response masking technique," *Signal Processing (Elsevier)*, vol. 85/3, pp. 581–590, Mar. 2005.
- [84] L. Zhou, W. Pei, and Z. He, "Extrapolated impulse response filter design utilizing principal components analysis," *Electronics Letters*, vol. 43, pp. 483 – 484, 2007.
- [85] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 446–455, Dec. 1994.

- [86] Y. J. Yu, D. Shi, and Y. C. Lim, "Design of extrapolated impulse response FIR filters with residual compensation in subexpression space," *IEEE Trans. Circuits Syst. I*, vol. 56, pp. 2621–2633, Dec. 2009.
- [87] D. S. K. Chan and L. R. Rabiner, "Theory of roundoff noise in cascade realizations of finite impulse response digital filters," *Bell Syst. Tech. J.*, vol. 52, pp. 229–245, Mar. 1973.
- [88] —, "An algorithm for minimizing roundoff noise in cascade realizations of finite impulse response digital filters," *Bell Syst. Tech. J.*, vol. 52, pp. 347–385, Mar. 1973.
- [89] R. Shively, "On multistage finite impulse response FIR filters with decimation," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 23, pp. 353–357, Aug. 1975.
- [90] R. Crochiere and L. Rabiner, "Optimum FIR digital filter implementations for decimation, interpolation, and narrow-band filtering," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 23, pp. 444–456, Oct. 1975.
- [91] M. Smith and D. Farden, "Statistical design of cascade finite wordlength FIR digital filters," in *Proc. IEEE ICASSP'84*, Mar. 1984, pp. 583–585.
- [92] W. Sung and S. Mitra, "Efficient FIR filter design using differential coding of filter coefficients," in *Proc. IEEE ICASSP'85*, Apr. 1985, pp. 45–48.
- [93] R. J. Hartnett and G. F. Boudreaux-Bartels, "On the use of cyclotomic polynomial prefilters for efficient FIR filter design," *IEEE Trans. Signal Process.*, vol. 41, pp. 1766–1779, May 1993.
- [94] Y. Lian, "Complexity reduction for frequency-response masking based FIR filters via prefilter-equalizer technique," *Circuits systems and signal processing*, vol. 22, pp. 137–155, Mar. 2003.

- [95] H. J. Oh and Y. H. Lee, "Design of discrete coefficient FIR and IIR digital filters with prefilter-equalizer structure using linear programming," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 562–565, June 2000.
- [96] Y. C. Lim and B. Liu, "Design of cascade form FIR filters with discrete valued coefficients," *IEEE Trans. Signal Process.*, vol. 36, pp. 1735–1739, Nov. 1988.
- [97] Y. C. Lim, R. Yang, and B. Liu, "The design of cascaded FIR filters," in *Proc. IEEE ISCAS'96*, vol. 2, May 1996, pp. 181–184.
- [98] S. U. Ahmad and A. Antoniou, "Cascade-form multiplierless FIR filter design using orthogonal genetic algorithm," in *Proc. 2006 IEEE Symposium on Signal Processing and Information Technology*, Aug. 2006, pp. 932–937.
- [99] O. Gustafsson, K. Johansson, H. Johansson, *et al.*, "Implementation of polyphase decomposed FIR filters for interpolation and decimation using multiple constant multiplication techniques," in *Proc. IEEE APCCAS'06*, Dec. 2006, pp. 924–927.
- [100] O. Gustafsson and H. Johansson, "Efficient implementation of FIR filter based rational sampling rate converters using constant matrix multiplication," in *Proc. IEEE ACSSC'06*, 2006, pp. 888–891.
- [101] A. G. Dempster and N. P. Murphy, "Efficient interpolators and filter banks using multiplier blocks," *IEEE Trans. Signal Process.*, vol. 48, pp. 257–261, Jan. 2000.
- [102] Z. J. Mou, "Symmetry exploitation in digital interpolators/decimators," *IEEE Trans. Signal Process.*, vol. 44, pp. 2611–2615, 1996.
- [103] Y. J. Yu, Y. C. Lim, and T. Saramäki, "Restoring coefficient symmetry in polyphase implementation of linear-phase FIR filters," *Circuits Systems Signal Processing*, vol. 25, pp. 253–264, 2006.

- [104] R. Bregovic., T. Saramäki, Y. J. Yu, *et al.*, “An efficient implementation of linear-phase FIR filters for a rational sample rate conversion,” in *Proc. IEEE ISCAS’06*, Kos, Greece, May 2006, pp. 5395–5398.
- [105] S. S. Demirsoy, A. Dempster, and I. Kale, “Transition analysis on FPGA for multiplier-block based FIR filter structures,” in *Proc. IEEE ICECS 2000.*, vol. 2, Dec. 2000, pp. 862–865.