Low complexity and low power FIR filter design

Ye, Wenbin

2014

Ye, W. (2014). Low complexity and low power FIR filter design. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/61298>

<https://doi.org/10.32657/10356/61298>

LOW COMPLEXITY AND LOW POWER FIR FILTER DESIGN

YE WENBIN

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2014

Acknowledgment

My Ph.D journey now is coming into an end. I would like to express my deepest appreciations to all who provided me the possibility to complete this journey.

In the first place, I would like to take this opportunity to express my sincere gratitude and appreciation to my supervisor Professor Yu Ya Jun who has given me tremendous guidance and advice on my research work. I am very grateful for her constant patience and enthusiasm over the past four years. Her guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

I want to express my gratitude to all my friends: Mr. Yi Xiang, Mr. Lin Jiafu, Mr. Cao Yuan, Mr. Zhao Bo, Mr. Zhang Xiangyu, Mr. Liu Hang, Mr. Jeremy, Dr. Zhu Xi, Dr. Liao Lei, Mr. Liu Benxu Mr. Li, Renshi, Ms. Hu Quan, Ms. Guo Hui and Ms. Tian Yujing for their help during my stay in Singapore. Special thanks are given to Dr Liao Lei who invited me to be the best man of his wedding, which is really a wonderful experience for me.

I also would like to thank the FYP students: Mr Og Zaiquan and Mr Tan Fuloong that I have guided. It has been quite enjoyable to guide them, and I also have learned much from them.

Furthermore, I want to thank my girlfriend, Ye Yun, for being a companion of my life since I started Ph.d study, for always being so positive and supportive in difficult times,

and for bringing me joyfulness.

Finally, I would like to give my deepest thanks to my parents, Ye Shuixiang and Zhou Jinmei, for their selfless love and the most important thing-giving me the life.

Contents

Acknowledgments	i
List of Figures	viii
List of Tables	x
List of Symbols	xii
Summary	xv
1 Introduction	1
1.1 Contribution	4
1.2 Publications	6
1.3 Organization	7
2 Multiplierless Linear Phase FIR Filters	10
2.1 FIR filters	11
2.1.1 FIR Filter Specifications	12
2.1.2 FIR Filter Structures	13
2.2 Mutliplierless FIR filters	16
2.2.1 SPT Techniques	16

2.2.2	MCM techniques	17
2.2.3	MCM and Filter Design Joint Optimization Techniques	19
2.3	Power Consumption for Mutliplierless FIR Filters	23
2.3.1	Adder Depth	23
2.3.2	Power models for Multiplierless FIR Filters	24
2.4	Summary	30

3 Single-Stage and Cascade Design of High Order Multiplierless linear phase FIR Filters Using Genetic Algorithm 32

3.1	Design Single-Stage FIR Filters Using GA	34
3.1.1	Partitioning the Search Space	35
3.1.2	Determining the Gains	38
3.1.3	Adaptive Crossover and Mutation Rates	39
3.1.4	Encoding and Initial Population	40
3.1.5	Fitness Function	41
3.1.6	Summary of the Proposed GA	43
3.2	Numerical Example of Single-Stage Design	43
3.2.1	Simulations Results	45
3.2.2	Computational Complexity	46
3.2.3	Determining the Parameters λ_1 , λ_2 and λ_0	50
3.2.4	Superiority and Limitation	51
3.3	Design Cascaded FIR Filters using the Proposed GA	52
3.3.1	Integer Coefficient Polynomial Factorization	53
3.3.2	An Example to Illustrate the Factorization Technique	56
3.3.3	Incorporating the Factorization Technique into the Proposed GA	57
3.4	Numerical Examples of Cascade Form Design	58

3.4.1	Example A	59
3.4.2	Example B	61
3.4.3	Robustness for the Design of Cascaded FIR Filters	65
3.5	Conclusion	66
4	A Polynomial Time Algorithm for the Design of Multiplierless Linear Phase FIR Filters with Low Hardware Cost	67
4.1	The Proposed Optimization Technique	69
4.1.1	Motivation	69
4.1.2	A Low Hardware Cost Mechanism	70
4.1.3	Determining the Number of Floating Passband Gain Range	73
4.2	Design Examples	74
4.3	Computation Complexity	76
4.4	Conclusion	79
4.5	Appendix A : Impulse Response of Design Examples	80
5	Multiplierless Multiple-stage Cascaded Linear Phase FIR Filter Design	83
5.1	The Proposed Technique	84
5.2	The Remarks of the Proposed Algorithm	88
5.3	Design Examples	88
5.4	Conclusion	89
5.5	Appendix B : Impulse Response of Design Examples	91
6	Design of Multiplierless FIR filters with Low Average Logic Depth	93
6.1	Review of tree search algorithm for optimization of FIR filters with MCM implementation	94

6.1.1	Problem Formulation for the FIR Filter design	94
6.1.2	Tree Search Procedure	95
6.1.3	Dynamically Expending Subexpression space	98
6.2	A New Synthesis Mechanism	98
6.2.1	Minimum Adder Depth of A Coefficient	99
6.2.2	A New Synthesis Mechanism Ensuring Coefficient Synthesized on its MinAD	100
6.3	Experiment Results	104
6.4	conclusion	106

7 Power Oriented Design of Linear Phase FIR Filters 107

7.1	Proposed Power Model for the Adders in Multiplierless FIR Filter	109
7.1.1	How is the Model to be Proposed	109
7.1.2	Modeling Procedure	111
7.1.3	Verification of the Model	113
7.1.4	Remarks of the Proposed Model	117
7.2	Power Oriented Optimization of Multiplierless FIR Filters	118
7.3	Coefficient Synthesis and Cost Estimation	120
7.3.1	A New Synthesis Mechanism leading to low PI implementation . .	121
7.3.2	Cost of Coefficient Realization	124
7.3.3	An Example to Illustrate the Synthesis Mechanism and the Com- putation of the Power Cost	125
7.4	Design Examples	126
7.4.1	Comparison of Power Consumption	127
7.4.2	Computational Complexity	130
7.5	Conclusion	134

8	Conclusions and Future Works	135
8.1	Conclusions	135
8.2	Future Works	137

List of Figures

2.1	Illustration of the frequency specification of a lowpass FIR filter	12
2.2	Direct form realization of a FIR filter	14
2.3	Transposed direct form realization of a FIR filter	15
2.4	Cascaded form realization of a FIR filter	15
2.5	Diagram of FIR filters with coefficients synthesized as a network of adders and shifts	18
2.6	(a) Filter coefficients expressed in CSD form (b) The implementation in SPT coefficients (c) The implementation with CSE	20
2.7	Filter coefficient set 3,13,6 synthesized using adder graph algorithm . . .	21
2.8	(a) An adder graph with 2 adders, (b) the value change of the output of each adder as the primary input signal changes. $f_{in}(t)$ is the primary input value at time t and $f_i(t)$ represents the output signal value at time t . . .	25
2.9	A simple example of GPC model when (a) 25 is realized as $3 \times 2^3 + 1$ and (b) 25 is realized as $5 \times 2^2 + 5$	27
2.10	Symbol of a full adder	29
3.1	The flow chart of the greedy algorithm in determining the base discrete coefficient set for a given passband gain G_m	37
3.2	The flow chart of the proposed GA	44

3.3	The mean fitness values changes with the number of generations, AGA stands for adaptive GA, and TGA stands for traditional GA	51
4.1	The maximum values of ρ for 20 sets of filters when their feasibilities are kept.	72
5.1	The flow chart to determine the coefficient set of subfilter	87
6.1	The discrete coefficient values 3, 11 and 43 are synthesized (a) without MinAD constraint and (b) with MinAD constraints	100
6.2	Procedure to synthesize a coefficient $h(k)$ on its MinAD	101
7.1	The mean values of normalized power consumption vs. the logic depth of adders d , and the fitting curve.	113
7.2	MCM block constructed by adders 3, 5, 11, 25, 49, 55, 41 and 99	114
7.3	Normalized ratios γ_j of the adders modeled by GPC, bit-level model and the proposed power model for the adders in Figure 7.2 at $f = 500$ KHz	116
7.4	Normalized ratios γ_j of the adders modeled by GPC [1], bit-level model [2] and the proposed power model for the adders in Figure 7.2 at $f = 50$ MHz	116
7.5	Flow-chart to synthesize a coefficient $h(k)$	123

List of Tables

2.1	The Filter Specifications of the Benchmark Filters Designed and Compared in This Thesis	31
3.1	Results and Comparison of the Three Benchmark Filters, where MBA and SA are the Number of Multiplier Block Adders and Structure Adders . .	45
3.2	The Coefficient Values of Filter B	46
3.3	The synthesis of coefficient set of Filter B	46
3.4	The Coefficient Values of Filter L1	47
3.5	The synthesis of coefficient set of Filter L1	47
3.6	An Example to Factorize a Discrete Coefficient Polynomial	57
3.7	Design results of Filters B, L1 and C. MBA and SA are number of multiplier block adders and number of structural adders	59
3.8	The Coefficient Values of Filter B in Cascade Form	62
3.9	The Coefficient Values of Filter L1 in Cascade Form	62
3.10	The Coefficient Values of Filter C in Cascade Form	63
3.11	Design Results of Filter A1 and S2. L_i , MBA_i and SA_i are the Length, Number of Multiplier Block Adders and Number of Structural Adders of Subfilter i , Respectively.	64
3.12	The Coefficient Values of Filter A1 in Cascade Form	64

3.13	The Coefficient Values of Filter S2 in Cascade Form	65
3.14	Design Results of Filter B with Different Filter Orders. L_i , MBA_i and SA_i are the Length, Number of Multiplier Block Adders and Number of Structural Adders of Subfilter i , Respectively.	65
4.1	Results and Comparison of Six Benchmark Filters	77
4.2	The Coefficient Values of Filter A1, S2 and L2	80
4.3	The Coefficient Values of Filter C	81
4.4	The Coefficient Values of Filter L1 and B	82
5.1	Results and Comparison of Benchmark Filters.	89
5.2	The Coefficient Values of Filter Y2 and L2 in Cascade Form	91
5.3	The Coefficient Values of Filter S2, A1 and Y3 in Cascade Form	92
6.1	An Example of the Proposed Coefficient Synthesis Procedure and the Update of Basis Sets	103
6.2	Results and Comparison of eight benchmark filters, Where EWL is effective wordlength, Wl is wordlength, NA is number of adder, MaxLD is Maximum Logic Depth, and ALD is Average Logic Depth.	105
7.1	PIs of Different Models and Simulated Power Consumptions of the Adders in Figure 7.2 at $f=500$ KHz	114
7.2	PIs of Different Models and Simulated Power Consumptions of the Adders in Figure 7.2 at $f=50$ MHz	114
7.3	An Example to Illustrate the Proposed Coefficient Synthesis Procedure, the Update of Basis Sets, and the Power Cost of Each Coefficient	125

7.4	The Result Comparisons between the Proposed Algorithm and the Algorithm in [3]. EWL is the Effective wordlength, MLD is the Maximum Logic Depth, ALD is the Average Logic Depth, NA is the Number of Adders and Power is the Simulated Power Consumption of the Filter.	129
7.5	The Power Comparison between the Proposed Algorithm and Adder-cost Oriented Algorithm with MinAD constraint proposed in Chapter 6 . . .	130
7.6	Impulse Response, Subexpression Bases and Coefficient Synthesis of Design Examples G1, Y1 and Y2	131
7.7	Impulse Response, Subexpression Bases and Coefficient Synthesis of Design Examples A1 and L2	132
7.8	Impulse Response, Subexpression Bases and Coefficient Synthesis of Design Example S2	133

List of Symbol

- α : Switching factor of an adder
- β : Cut-off factor
- ρ : Weighting factor for the control of ripple deterioration in a tree search algorithm
- δ_{opt} : The optimal passband ripple of a filter
- δ_p : Passband ripple specification of a filter
- δ_s : Stopband ripple specification of a filter
- $\lambda_0, \lambda_1, \lambda_2$: Weighting parameters to determine P_c and P_m in adaptive genetic algorithm
- ω_p : Passband edge of a filter
- ω_s : Stopband edge of a filter
- a : A parameter of glitch path score model
- A : A parameter defined the passband gain range
- b : Floating passband gain of a filter
- d : adder depth of an adder
- D : Depth of a tree in a tree search algorithm
- G : The number of generations of proposed genetic algorithm

- $h(n)$: Impulse response of a filter
- $H(\omega)$: Frequency response of a filter
- P_N : Total power consumption of a circuit node
- P_D : Dynamic power consumption of a circuit node
- P_S : Static power consumption of a circuit node
- C_L : Capacitance load of an adder
- P : Population size of proposed genetic algorithm
- P_c : Crossover rate of proposed genetic algorithm
- P_m : Mutation rate of proposed genetic algorithm
- N : Filter order
- L : Width of a tree in a tree search algorithm
- Q : Wordlength of filter coefficient
- r : Width of an adder

Summary

With the advent of information era, the use of digital devices such as hand phones, digital cameras and hearing aids becomes more and more pervasive. Digital finite impulse response (FIR) filters are essential components in these devices. In order to save the cost and increase battery life for these portable digital products, in the past decades, many efforts have been dedicated for the design and implementation of low hardware cost and low power FIR filters. Among many others, one of the most successful techniques for such filter design and implementation is known as multiplierless FIR filters, i.e., the general coefficient multipliers can be replaced with adders and shifts in this technique.

In multiplierless FIR filters, the hardware cost is proportional to the number of adders since the shift can be hard-wired and thus considered cost-free. Reducing the number of adders therefore can reduce the hardware cost directly and it is also generally believed that it can reduce the power consumption indirectly. Existing algorithms thus mainly aim to minimize the adder cost. The limitation of existing algorithms lies in two aspects. First, the computational complexity to achieve the minimum number of adders is too high to be used in the design of long filters, and second, the number of adder criterion is too coarse and inaccurate for the minimization of power consumption.

Therefore, the research in this thesis deals with the issues of designing multiplierless FIR filters to minimize the hardware cost and power consumption. The research is conducted in the following two aspects.

First, several optimization techniques with reduced algorithm complexity are proposed and thus are capable of designing long multiplierless FIR filters. The proposed optimization techniques are improved genetic algorithm and polynomial time tree search algorithm. Design results show that the proposed techniques can efficiently generate the FIR filters with low hardware cost. In addition, in order to further reduce the hardware cost, an algorithm is proposed for the design of FIR filters in multiple-stage cascade form with discrete coefficients.

Second, besides the number of adders, more criteria are considered in the design of FIR filters with minimum power consumption, including the logic depth and operating frequency. In this thesis, an algorithm trading off the minimum number of adders and the minimal logic depth is first proposed. Besides that, a power model that more accurately describes the power consumption is developed and used for the design of FIR filters. The power model also takes both the dynamic power and static power into consideration. Thus, for the first time, the designed filters are frequency aware, i.e., for the same filter specifications, the proposed algorithm may generate different designs for filters operating at different frequencies, and the designed filters consume the minimum power for the respective frequencies.

Chapter 1

Introduction

With the great development of computer technologies, we have now come into a new era, i.e, the information era. The most significant feature of this era is the wide use of digital products such as cellular phones, digital cameras and tablet computers. The core of these digital products lies in the digital signal processor working as the brain of human beings. Compared to analog signal processors, the digital signal processor has many advantages such as high stability, low cost and insensitivity to the noise and temperature shift, and thus has been widely used in sample rate converter, image processing, speech coding and communication systems [4–9].

Filtering is one of the most common processing techniques in the digital signal processing (DSP) systems. According to the impulse response, digital filters can be classified into two categories: finite impulse response filters (FIRs) and infinite impulse response filters (IIRs) [10, 11]. Despite of the fact that IIR filters are computationally more efficient, FIR filters possess desirable properties for a wide range of useful applications. First, the linear phase and stability of the FIR filters can be guaranteed. In communication systems, linear phase is a necessary condition for the minimization of intersymbol interferences. Image processing also requires linear phase to maintain the colors. Second,

FIR filters have desirable numeric properties. In practical hardware implementation, all filters must be implemented using "finite-precision" arithmetic, that is, a limited number of bits in arithmetic operations. The use of finite-precision arithmetic in IIR filters can cause significant and complicated finite wordlength effect when intermediate results are rounded due to the use of feedback. FIR filters have no feedback, and hence the designer have fewer practical problems to solve related to non-ideal arithmetic. In addition, FIR filters are suited to multi-rate applications. In decimating or interpolating operations, the use of FIR filters allows some of the calculations to be omitted, thus providing an significant computational efficiency. In contrast, if IIR filters are used, each output must be individually calculated, even if the output will be discarded. because the discarded outputs are involved in the computation of non-discarded outputs due to the feedback loop in IIR filters. With all the above desirable properties, however, FIR filters also come with a disadvantage, that is the computational complexity and power consumption are much higher than its counterpart IIR filters. Therefore, during the past decades, continuous efforts have been dedicated to the design of low complexity and low power FIR filters. One of the most successful strategies is the multiplierless filtering [12, 13], where the constant filter coefficient multipliers are replaced by a network of adders and shifts. Since the multiplier is the most power-hungry and expensive component in very large scale integration (VLSI) realization of FIR filters, multiplierless filters reduce the implementation complexity and power consumption significantly.

In the past decades, many researchers have paid attentions to the design of multiplierless FIR filters [3, 12–38, 38–54]. Basically, these researches can be divided into three categories. In the first category [12–25, 27, 29], a set of discrete coefficient values is given, and the algorithms encode the set of coefficient values into signed power-of-two (SPT) representations. The algorithms of this category are generally known as SPT techniques. Compared to the conventional two's complement representation, the SPT representation

contains fewer non-zeros digits, leading to efficient implementations of add/subtract networks. Since the hardware cost is mainly proportional to the number of SPT terms, the algorithms in this category aim to minimize the SPT terms for a given filter specification, or to minimize the frequency response ripple when the number of SPT terms is fixed.

Common SPT terms, generally known as common subexpressions, in fact can be shared among filter coefficients. Therefore, in the algorithms of the second category [30–45], instead of synthesizing each coefficient by adders and shifts separately, all coefficient multipliers are synthesized together using an adder-and-shift network to share common subexpressions. Such algorithms are named as multiple constant multiplication (MCM) techniques, because in transposed direct FIR structures, each signal sample is multiplied by multiple constant coefficients.

The MCM techniques consider only a pre-designed discrete coefficient set. However, for a given filter specification, the filter coefficient sets that satisfy the specification are not unique, and there possibly exist other coefficient sets which meet the specification and may be synthesized by using lower hardware cost and consuming lower power. Therefore, the algorithms in the third category [3,48,50–56] incorporate the MCM technique into the coefficient optimization, resulting in further reduction of implementation hardware cost and power consumption. This category of algorithms is named as MCM and coefficient joint optimization techniques. In these algorithms, the search of the coefficients and the synthesis of the coefficients are performed interleavedly. Currently, most of the algorithms in this category are based on tree search algorithms to investigate each potential discrete solution in a specified search space. Global optimal solutions may be obtained, if the search time and computational resources are unlimited. However, for high order filter design problems, the exhaustive tree search algorithms may not be used due to its high computational complexity. To solve this problem, many sub-optimal methods [53–55] with reduced search space or using heuristic algorithms have been developed.

The objective of this research is to develop algorithms which can generate designs and implementations of FIR filters with low hardware cost and low power consumption.

1.1 Contribution

The contributions of this thesis mainly lie in the following:

- A novel genetic algorithm (GA) is proposed for the design of multiplierless linear phase FIR filters. The filters under consideration are of high order and wide coefficient wordlength. Both the single-stage and cascade form are considered. In a practical filter design problem, when the filter specification is stringent, requiring high filter order and wide coefficient wordlength, GAs often fail to find feasible solutions, because the discrete search space thus constructed is huge and the majority of the solution candidates therein can not meet the specification. In the proposed GA, the discrete search space is partitioned into smaller ones. Each small space is constructed surrounding a base discrete coefficient set which is obtained by a proposed greedy algorithm. The partition of the search space increases the chances for the GA to find feasible solutions, but does not sacrifice the coverage of the search. The proposed GA applies to the design of single-stage filters. When a cascade form filter is designed, for each single-stage filter meeting the filter specification generated during the course of GA, an integer polynomial factorization is applied. Design examples show that the proposed GA significantly outperforms existing algorithms dealing with the similar problems in terms of design time, and the hardware cost is saved in most cases.
- A polynomial-time tree search algorithm where each coefficient is fixed to only one discrete value is proposed. Due to the short search time, multiple searches with

floating passband gain become possible, and each search may produce a feasible discrete solution. In such a way, if the whole floating passband gain range is partitioned into many smaller ones, a large amount of feasible discrete solutions may be obtained. In the end, all these feasible solutions are synthesized using a MCM algorithm, and the one using the least number of adders is the final design results. To accelerate the search, a low hardware cost scheme which fixes some coefficient values to 0 prior to each search is proposed. With these techniques, design examples show that the proposed algorithm significantly outperforms the existing algorithms in terms of design time while the hardware cost is kept low.

- A novel algorithm is proposed for the design of linear phase FIR filters in multiple-stage cascade form with discrete coefficients. The number of stages is automatically determined by the algorithm once an overall filter order is given. Design examples have shown that the proposed algorithm achieves significant reduction in hardware cost when compared with the existing best algorithm, and the the design time is also saved.
- An algorithm which aims to minimize the number of adders but with the constraint that each adder is on its minimum adder depth is proposed. The proposed algorithm finds an adder-and-shift network with a small logic depth to realize the coefficient multipliers. The new synthesis mechanism tends to use a few more adders to realize the discrete filter coefficients than the current best algorithms, but with lower average logic depth, resulting in a low power implementation.
- For the first time, a power oriented optimization algorithm for the design of FIR filters is proposed, where a power cost is used as the criteria in the discrete coefficient search and synthesis. The power cost is computed based on a newly proposed

power model, which takes both the static power and dynamic power into consideration. With the new power model, a new coefficient synthesis scheme is proposed such that the synthesized coefficient consumes the minimum power. Compared to the adder-cost oriented algorithm, the proposed power-oriented algorithm has two advantages: first, the algorithm can optimize filters with lower power consumption, and second, the optimal design in the sense of power consumption is frequency aware. Unlike the adder-cost oriented algorithms that generate the same final coefficient set and the same synthesis of the coefficient set regardless of the frequency for a given filter specification, the proposed algorithm searches and synthesizes the coefficients with the awareness of the working frequency; different designs may be resulted for the same filter specification but different working frequency, and each designed filter has lower power consumption in its specified frequency. Transistor level simulations of benchmark filters verified the above claims.

1.2 Publications

This research lead to the following papers which have been published, accepted for publication, or submitted for the consideration of publication in journals and conference proceedings.

Journal Papers:

- W. B. Ye and Y. J. Yu, “Single-Stage and Cascade Design of High Order Multiplierless linear phase FIR Filters Using Genetic Algorithm”, *IEEE Transactions on Circuits and Systems-I*, Vol 60, pp. 2987-2997, Nov. 2013.
- W. B. Ye and Y. J. Yu, “A Greedy Algorithm for the Design of Sparse FIR Filters”,

submitted to *Circuit system and Signal Procoessing*.

- W. B. Ye and Y. J. Yu, “Optimization of Multiplierless FIR Filters at Bit-Level”, submitted to *IEEE Transactions on Circuits and Systems-I*.

Conference Papers:

- W. B. Ye and Y. J. Yu, “An Algorithm for the Design of Low Power Linear Phase FIR filters”, in *Proc.of Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, 2011, pp. 145-148.
- W. B. Ye and Y. J. Yu, “Switching activity analysis and power estimation for multiple constant multiplier block of FIR filters”, in *Proc. IEEE Int. Symp. Circuits Syst.*, 2011, pp. 29-32.
- W. B. Ye and Y. J. Yu, “Design of High Order and Wide Coefficient wordlength multiplierless FIR filters with Low Hardware Cost Using Genetic Algorithm”, in *Proc. IEEE Int. Symp. Circuits Syst.*, 2012, pp. 29-32.
- H. Zhao, W. B. Ye and Y. J. Yu, “Sparse FIR Filter Design Based on Genetic Algorithm”, in *Proc. IEEE Int. Symp. Circuits Syst.*, 2013, pp. 97-100
- W. B. Ye and Y. J. Yu, “A Polynomial time Algorithm for the Design of Multiplierless Linear Phase FIR Filters with Low Hardware Cost”, *Proc. IEEE Int. Symp. Circuits Syst.* to appear.

1.3 Organization

The thesis is organized as follows:

In Chapter 1, the design of multiplierless FIR filters are briefly introduced. The goal, main contributions and organization of the thesis are stated.

Chapter 2 reviews the algorithms for the design of low power and low complexity multiplierless FIR filters. Besides that, exiting power models to estimate the power consumption of the multiplierless FIR filters are also reviewed.

In Chapter 3, a novel genetic algorithm is proposed for the design of multiplierless linear phase FIR filters. The filters under consideration are of high order and wide coefficient wordlength. Both the single-stage and cascade forms are considered.

A polynomial-time algorithm for the design of multiplierless linear phase FIR filters is presented in Chapter 4. Design examples show that the proposed algorithm significantly outperforms the existing algorithms in terms of design time while the hardware cost is kept low.

In Chapter 5, a novel algorithm is proposed for the design of multiplierless multiple-stage cascaded FIR filters. Unlike to the conventional algorithms in which the number of stages is fixed and usually is fixed to two, the number of stages in the proposed algorithm is automatically determined. The design examples show that the proposed algorithm significantly outperforms the best existing algorithm in terms of hardware cost and the design time is also saved.

Chapter 6 proposes an algorithm which aiming to minimize the number of adders but with the constraint that each adder is synthesized on its minimum adder depth. The new synthesis mechanism tends to use a few more adders to successfully realize the discrete filter coefficients than the current best algorithms, but with shorter average logic depths, resulting in a low power implementation. Designs of benchmark filters and simulation results verified the above claims.

Chapter 7 proposes a power oriented optimization algorithm for the design of linear phase FIR filters. A power cost computed based on a newly proposed power model is used as the criteria in the discrete coefficient search and synthesis. The proposed technique is operating frequency aware, i.e., for a given filter specification, the algorithm may generate

different designs for different frequencies, and each design consumes the minimum power for the respective frequency.

Chapter 8 concludes the thesis.

Chapter 2

Multiplierless Linear Phase FIR Filters

FIR filters have many applications in DSP systems for their guaranteed linear phase and stability [10]. However, compared with IIR filters, FIR filters usually suffer from high computational complexity, and hence, consume more power, if the same magnitude response is realized. Therefore, in the past decades, many efforts have been dedicated to the design of FIR filters with low hardware cost and low power, and one of the most successful techniques is multiplierless FIR filters. In this chapter, first, the FIR filters and filter structures are briefly introduced, followed by the introduction to the multiplierless FIR filters. The most significant works on the design of multiplierless linear phase FIR filters are reviewed. In the end, existing power models for the estimation of power consumption of multiplierless FIR filters are also reviewed.

2.1 FIR filters

In digital signal processing, an FIR filter is a digital device whose impulse response is of finite duration, i.e., its impulse response settles to zero in finite time. The z -transform transfer function of an FIR filter is expressed as

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

where $h(n)$ for $n = 0, 1, 2, \dots, N$ is the impulse response, and N is the filter order. The frequency response of the filter thus is expressed as

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

FIR filters can be easily designed to be with linear phase by setting the impulse response $h(n)$ to be symmetrical or anti-symmetrical. Based on the symmetry of the impulse response (symmetrical or anti-symmetrical) and the filter order (even or odd), there are 4 types of linear phase FIR filters. The corresponding frequency responses are given as

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

In the above expression of $H(e^{j\omega})$, $\frac{N}{2}\omega$ is the phase response, and it is proportional to the frequency, i.e., presenting the linear phase. And the factor of proportionality for the phase with respect to the frequency in linear phase filters is determined by the filter order N . If the term of $e^{j\frac{N}{2}\omega}$ is ignored, the rest expression is known as the zero-phase frequency response.

Figure 2.1: Illustration of the frequency specification of a lowpass FIR filter

2.1.1 FIR Filter Specifications

The design of FIR filters, mainly to find the impulse response $h(n)$ of the filter, usually starts from a set of filter specifications. Taking lowpass filter as an example, basically, the filter specifications contain four parameters, the passband edge frequency ω_p , the stopband edge frequency ω_s , the passband ripple δ_p , and the stopband ripple δ_s . An illustration of the filter specification for a lowpass filter is shown in Figure 2.1. The transition bandwidth Δ of the filter is the distance between the passband edge and stopband edge. If the zero-phase frequency response of the filter has smaller fluctuations than δ_p and δ_s in the passband and stopband regions, respectively, we say that the filter meets the specification

For the given specifications, the filter order N can be estimated by [57]

$$N \approx D_{\infty}(\delta_p, \delta_s)/\Delta + f(\delta_p, \delta_s) + 1 \quad (2.3)$$

where

$$\begin{aligned} D_{\infty}(\delta_p, \delta_s) = & 10\delta_s[a_1(\log_{10}\delta_p)^2 + a_2\log_{10}\delta_p + a_3] \\ & -[a_4(\log_{10}\delta_p)^2 + a_5(\log_{10}\delta_p) + a_6] \end{aligned}$$

and

$$f(\delta_p, \delta_s) = 11.01217 + 0.51244[\log_{10}\delta_p - \log_{10}\delta_s].$$

The coefficient a_1 to a_6 are given by

$$\begin{aligned} a_1 &= 0.005309, \quad a_2 = 0.07114, \quad a_3 = -0.4761, \\ a_4 &= 0.00266, \quad a_5 = 0.5914, \quad a_6 = -0.4278. \end{aligned}$$

It can be seen from Eq.(2.3) that the filter order increases if the ripple tolerances become smaller and / or the transition width becomes narrower.

2.1.2 FIR Filter Structures

There are many structures to implement FIR filters. From mathematical perspective, these structures are all equivalent, but in practical implementation, the hardware cost and power consumption of different structures can be greatly different. Three commonly structures used in the realization of FIR filters are: direct form, transposed direct form and cascade form. The direct form structure for the realization of FIR filters is shown in Figure 2.2. Figure 2.2 shows that this structure, the signal samples are fed into a

Figure 2.2: Direct form realization of a FIR filter

delay chain, the delayed samples are then multiplied by the coefficient multipliers, and all the products are finally summed up to produce the output sample. The multiplication constants are the impulse response $h(n)$ for $n = 0, 1, 2, \dots, N$ of the filter. As indicated by the dot line in Figure 2.2, the critical path of such realization is a multiplier plus N adders. As a result, for direct form realization, the critical path may become very long if the filter order N is high.

On the other hand, the transposed direct form shown in Figure. 2.3 has the critical path containing only a multiplier and an adder, i.e., the length of the critical path is independent of the filter order. Thus, the filter may have high throughput even for long filters. However, the width of the delay, i.e., number of flip flops in the transposed direct form is larger than that in the direct form. Therefore, the hardware cost of the transposed direct form may be higher than that of the direct form. Besides that, the input capacitance of the transposed direct form is much larger than that of direct form.

Besides the direct and transposed direct forms for the realization of FIR filters, another popular structure is the cascade form realization. In a cascade structure, a filter is decomposed into several low-order sections. For example, if a filter with an order N is factorized into two subfilters, and the order of the subfilters are N_1 and N_2 , respectively,

Figure 2.3: Transposed direct form realization of a FIR filter

Figure 2.4: Cascaded form realization of a FIR filter

the z -transform transfer function of the cascade realization is expressed as

$$H(z) = \sum_{n_1=0}^{N_1} h_1(n_1)z^{-n_1} \sum_{n_2=0}^{N_2} h_2(n_2)z^{-n_2}$$

where $N_1 + N_2 = N$. The corresponding cascade structure to realize this transfer function is shown in Figure 2.4, where both subfilters are realized in the direct form. In practical realization, each subfilter can be either in direct or transposed direct form, depending on applications. FIR filters in cascade structure are well-known for their low computational complexity and low coefficient sensitivity, compared with the direct or transposed direct structures [58].

Overall speaking, the selection of implementation structures are determined by many factors and the specific applications. For example, if an implement has a high speed requirement, the transposed direct form is preferred.

2.2 Mutliplierless FIR filters

Since the multiplier is the most expensive component in the VLSI implementation of FIR filters, the power consumption of FIR filters becomes unaffordable when the filter order is high. One of the most successful techniques to design low complexity and low power FIR filters is the multiplierless technique. As stated in Chapter 1, the multiplierless FIR filter design can be classified into three categories. The typical algorithms in each category are discussed in this section.

2.2.1 SPT Techniques

As mentioned in Chapter 1, the hardware cost of implementing a FIR filter is generally proportional to the number of SPT terms. Basically, depending on the applications there are two ways to optimize the filters, i.e, for a given ripple requirements, the number of SPT terms are minimized, or for given filter specifications, the filter ripples are minimized. Typical techniques solving the former one use a polynomial-time algorithm or trellis-search algorithm, as proposed in [27, 28], while the later one is usually optimized using mixed integer linear programming(MILP) [13, 15–19].

Besides the above deterministic algorithms, genetic algorithms (GAs) are also developed to minimize the SPT terms for given SPT term requirements [24, 26, 59]. A GA is an algorithm that mimics the process of natural evolutions. By such process, the solution gradually evolves to the optimum one. Basically, in GAs, potential solutions of the unknowns are encoded to chromosomes and 3 genetic operations including selection, crossover and mutation are applied to these chromosomes. In detail, after initializing the first generation population of the chromosome pool, some individuals, i.e., individual chromosomes in the existing population are selected according to a selection mechanism to breed a new generation. Those selected individuals are called “parents” and they

produce “children” by crossover and mutation operators. The generational process containing selection, crossover and mutation is repeated until a prespecified termination criterion is reached. When GAs are used to minimize the number of SPT terms, each coefficient set is encoded into a CSD string, which in a chromosome in the population. Each chromosome is associated with a fitness value reflecting the number of SPT terms used in the coefficient set. Through the genetic operations, new generation of chromosomes are bred. The chromosomes with higher fitness value and meeting the filter specifications meanwhile in each generation survive. In such a manner, the filter with the minimum number of SPT terms are eventually generated.

2.2.2 MCM techniques

As seen in Figure 2.3 that in the transposed direct form, the input signal is concurrently multiplied by filter coefficient values, the coefficient multiplications become a multiple constant multiplication (MCM) problem [60] for a given set of discrete coefficients. Instead of synthesizing each multiplier using adder and shifts as in the SPT techniques, in MCM problems, all the coefficients multipliers are replaced with a network of adders and shifts [13], as shown in Figure 2.5. The adder-and-shift network realizing the multiplications is called the MCM block. The adders in this block are named as multiplier block adders(MBAs) and the adder in the delay chain are named as structural adders(SAs). In such network, the common subexpression can be shared among different coefficients, for example, the adder 3 is shared between coefficients 6 and 13 in Figure 2.5(b). Therefore, MCM techniques aim to extract as many common subexpressions as possible from give coefficient sets to maximally increases the filter operation speed and reduces the circuit complexity and power consumption. Basically, there are two types of MCM techniques.

The first type technique, named as common subexpression elimination (CSE) [30–

Figure 2.5: Diagram of FIR filters with coefficients synthesized as a network of adders and shifts

33, 35, 36], usually represented the discrete coefficient values as canonical signed digital (CSD) or two's complement form. The common patterns of the coefficients are then extracted and shared. For example, if a filter coefficient set 3, 6, 13 is given, the coefficient set can be represented in CSD forms as shown in Figure 2.6 (a). It can be seen that the common patterns underlined in Figure 2.6(a) can be shared among these 3 coefficients. If each coefficient is synthesized separately using SPT techniques, 4 adders are required as shown in Figure 2.6(b). However, it is noted that there are a common pattern $10\bar{1}$ existing in all the 3 CSD representations, indicated by the underlines in Figure 2.6 (a). If the common pattern $10\bar{1}$ is shared, the implementation is shown in Figure 2.6(c). By such sharing, the total number of adders is reduced by 2. It should be noted that CSE algorithms are coefficient representation dependent, i.e., different representation formats may lead to different implementation cost of MCM blocks.

Unlike the CSE, the second type of MCM technique, the adder-graph based algorithms

[30, 34, 38, 41, 48] is coefficient representation independent. For this type algorithms, the coefficients are realized by shift and add operations on partial product nodes. Each node represents an adder. Figure 2.7 shows the adder-graph synthesis the coefficient set by partial product nodes. In Figure 2.7, the dots represent the partial product nodes, computed as the sum or subtraction of the values represented by the two edges; each edge is an input of an adder; the integer number alongside an edge is the number of binary positions shifted by the partial product; and minus symbol “−” alongside the node indicates that the node is a subtract. Figure 2.7 illustrates the synthesis of coefficient set $\{3, 13, 16\}$. The synthesis starts from the root node 1. A partial product 3 is realized using one adder as $3 = 1 + 1 \ll 1$, where “ $\ll n$ ” represents shift to left by n binary positions. Based on the node 3 the product 13 is realized as $13 = 3 \ll 2 + 1$. BH algorithm [30] is the first adder-graph based algorithm, synthesizing the coefficients from the smallest values. The later developed n-dimensional reduced adder graph (RAG-n) [34] synthesizes the coefficient values based on a pre-calculated table which is used to store the adder distances among different partial products. RAG-n in general outperforms BH algorithm by using less adders to synthesize coefficients. However, the adder depth, i.e., the number of cascade adders, of RAG-n method is generally very large. Algorithm C1 [38] overcomes this problem by adopting minimum-depth decision techniques.

Generally speaking, the adder-graph based algorithms can synthesize the coefficient set with fewer number of adders than that of CSE algorithms, but the adder depth is larger than that of CSE methods.

2.2.3 MCM and Filter Design Joint Optimization Techniques

In the second category, the MCM algorithms are only applied to a given coefficient set. However, for a given filter specification, the filter coefficient set meeting the specification

$$h(0) = \underline{10\bar{1}} \quad h(1) = \underline{10\bar{1}0} \quad h(2) = \underline{10\bar{1}01}$$

(a)

(b)

(c)

Figure 2.6: (a) Filter coefficients expressed in CSD form (b) The implementation in SPT coefficients (c) The implementation with CSE

Figure 2.7: Filter coefficient set 3,13,6 synthesized using adder graph algorithm

is not unique. Therefore, in the third category [3, 48, 50, 52–55, 61], many researchers adopt the design process that combines the filter optimization and MCM techniques together, i.e., searching and synthesizing the coefficient values jointly.

Typical algorithms in this category, constrain the search in a pre-defined space. Tree traversal algorithms are employed to systemically search the coefficients in the defined space and meanwhile the coefficients are synthesized using MCM algorithms. In the algorithms of this category, the efficiency of the tree search determines the algorithm complexity. Among the different tree traverse techniques, the depth-first search is well-received for the reason that a discrete solution is generated quickly. And along with the improvement of the discrete solutions, the cut-off criteria become more and more stringent, accelerating the search procedure.

The earliest depth-first search for the design of multiplierless FIR filters is the branch-and-bound (B&B) technique [15]. In this technique, in each depth along the search, a coefficient is selected for branching, and two linear programmings (LPs) are formulated by adding in bounds to the selected coefficients. The B&B search tree thus is a binary tree and the depth of the tree in general is larger than the number of coefficients to be optimized. When B&B tree search is applied to the MCM filter design [52], the

subexpression space has to be pre-specified, because a coefficient may be selected for branching more than once in a path.

A depth-first width-recursive search [51] solved the above problem by fixing a particular coefficient only at a particular tree depth. In such a manner, the tree depth becomes the number of coefficients to be optimized. However, the tree is no longer a binary tree because each coefficient may be fixed to multiple discrete values. Once a coefficient is fixed to a discrete value, it will not be fixed to another value in the same path. Therefore, a subexpression space can be constructed dynamically along each search path. In [51], the technique to accelerate the search is to fix the coefficients to discrete values that using less adders earlier than that using more adders, such that the search results in designs with low hardware cost faster.

While the technique in [51] sacrifices the completeness of the search to obtain lower hardware cost design faster, a complete search is proposed in [3]. For each coefficient to be quantized, it is first fixed to the rounded continuous value, and subsequently fixed to the next closest discrete values, until the filter specification is violated. By combining a technique aware of the optimum in the synthesis of the MCM coefficients, this approach may produce near optimum design in terms of minimum number of adders. However, due to its complete search of the tree, the technique may design filters with orders up to around 60 to 70.

A similar exhaustive search was earlier proposed in [50], where the feasible range of each coefficient is pre-computed. However, the search range in [3] is smaller than that in [50], because the feasible ranges for coefficients are reduced when more and more coefficients are fixed.

In the technique in [55], once a coefficient is fixed to a particular discrete value, the feasible range of the next coefficient to be fixed is explicitly computed. To accelerate the search, the coefficient is only fixed to 2 discrete values in the feasible range. This

technique shows that when these 2 values are the middle values in the feasible range, better results with lower hardware cost generally can be produced. Since each coefficient is fixed to only two discrete values, the proposed technique may design filters of order more than 300 in 24 hours.

2.3 Power Consumption for Mutliplierless FIR Filters

The hardware cost of multiplierless FIR filters is proportional to the number of adders used (including MBA and SA). However, the power consumption is not completely determined by the number of adders. For the digital CMOS circuit (of IC technologies above 90nm), the major power consumption, i.e., the dynamic power consumption is caused by the transition in the circuit. The dynamic power consumption of a node is given by [62]

$$P_{dynamic} = \frac{1}{2} \alpha C_L V_{dd}^2 f \quad (2.4)$$

where V_{dd} and f are the supply voltage and clock frequency, C_L is the load capacitance and α is the switching activity factor. From Eq. 2.4, it can be seen that the power consumption of a node is very much related to α , i.e., the number of transition in the circuits. One important factor which influences the number of transition in an adder node is the adder depth [63].

2.3.1 Adder Depth

The adder depth of an adder, is the number of adders through which the primary input signal has traveled before it reaches the output of that adder. An example shown in Figure 2.8 (a) is used to illustrate the concept of the adder depth. The adder depth

of the adder with output signal $f_1(t)$ is 1 because to generate $f_1(t)$, the primary input signal $f_{in}(t)$ has passed through an adder, while the adder with output signal $f_2(t)$ is adder depth 2 because one more adder is passed through by $f_{in}(t)$.

During the operation of the physical circuit of each adder takes some time to process the data. Assuming that the time to process an addition by an adder is ΔT , the correct output f_2 can be generated until at $2\Delta T$. From Figure 2.8 (a), it can be seen the $f_1(n) = 3f_{in}(n)$ and $f_2(n) = 2f_1(n) + f_{in}(n)$. Regarding the timing, assuming that initially at time 0^- , the input $f_{in}(0)$ is 5, and all the output nodes are stable with values 15 and 35 for $f_1(0)$ and $f_2(0)$, respectively. The input signal $f_{in}(0^+)$ is changed to 7 at time 0. f_1 changes from 15 to 21 at time ΔT as shown in Figure 2.8 (b). However, the output f_2 at time ΔT , is changed from 35 to 37, which is not the correct output. Only until at $2\Delta T$, f_2 becomes 49, which is then the right output. From Figure 2.8 (b), it can be seen that the adder in the depth 1 only changes once when the primary input changes, while the adder in the adder depth 2 to need to change two times in order to generate the correct output. Since the more changes lead to more transitions and thus result in higher power consumption, from the above example, one can see that generally the adder in higher adder depth consumed higher power.

In viewing of the above, in the filter design, to further reduce the power consumption, recent researches [38,64,65] have restricted the adder depth in the synthesis of coefficient values when the number of adders are minimized.

2.3.2 Power models for Multiplierless FIR Filters

Though constraining the adder depth during the minimization of the number of adders may reduce the power consumption, such optimization is still an indirect minimization of the power consumption. In order to directly minimize the power consumption, instead

Figure 2.8: (a) An adder graph with 2 adders, (b) the value change of the output of each adder as the primary input signal changes. $f_{in}(t)$ is the primary input value at time t and $f_i(t)$ represents the output signal value at time t .

of aiming to minimize the number of adders, a power directly related parameter should be set as the objective function. Therefore, it is essential to develop an accurate power model for multiplierless FIR filters. Since adders are the basic computation unit of multiplierless FIR filters, if the power consumption of adders can be accurately modeled, the whole architecture of the multiplierless FIR filter can also be modeled by summing the power index of each adder. Basically, there are 3 existing power models for specifically developed for the adders in FIR structures. They are glitch path count (GPC) [1], glitch path score (GPS) [66] and switching-activity factor direct estimation method [67].

Glitch Path Count Model

GPC model assumes that the transitions of the preceding adders propagate to all adders in the subsequent stages. A cost function termed as the ‘Glitch Path Count’ is defined to reflect the number of transitions generated and propagated by each adder node. The glitch path (GP) count of a node is the sum of the GP counts entering the node plus one. The additional count is generated by the current node. Mathematically, it is represented as:

$$GP_{output}^i = GP_{input1}^i + GP_{input2}^i + 1 \quad (2.5)$$

for node i , where GP_{input1}^i and GP_{input2}^i are the GP counts at the input of the i^{th} adder node. The total number of GP counts in a MCM block is then given as:

$$GP_{total} = \sum_i GP_{output}^i \quad (2.6)$$

A simple example is shown in Figure 2.9. Figures 2.9 (a) and (b) show two different architectures to realize the coefficient 25. According to (2.6), the total GP of the first realization is 3 and the other one is 5.

Figure 2.9: A simple example of GPC model when (a) 25 is realized as $3 \times 2^3 + 1$ and (b) 25 is realized as $5 \times 2^2 + 5$

The GPC power model is very simple but with lower accuracy. The reason is that the number of glitches cannot exactly reflect the switching activity, and generally it over estimate the transitions. Moreover, the effect of the adder width, i.e., the number of full adders (FAs) in each adder, is ignored. Therefore, this model only provide a rough estimation of the power consumption of adder-and-shift networks.

Glitch Path Score Model

In the GPC model, the wordlength of each adder node is assumed to be the same. However, this assumption is not always true. A refined model called glitch path score (GPS) is proposed in [66]. This model is based on the GPC but takes the adder width into consideration. The procedure for calculating the GP score of a node is as follows:

1. Compute the wordlength, n , of the adder at the node, by:

$$n = \text{floor}(\log_2(x) + 1) + n_{input} \quad (2.7)$$

where x is the partial product, i.e., the value represented by the current node; and n_{input} is the number of bits of the primary input.

2. Let

$$e = \max(s_1, s_2), \quad (2.8)$$

where s_1 and s_2 are the numbers of positions to be shifted by the two inputs.

3. If the operation is a subtraction, the actual adder length, r , is given by:

$$r = n, \quad (2.9)$$

and if the operation is an addition, the adder width is:

$$r = n - e. \quad (2.10)$$

4. Let the GPS's of the two input nodes be GPS_{input_1} and $input_2$. GPS of each node consists of the sum and carry glitches, but only the sum glitch will propagate to the next stage. A scaling factor 0.67 determined empirically is adopted to compute the GPS, i.e., the GPS of the current node is given by $0.67(GPS_{input_1} + GPS_{input_2})$
5. As the logic depth increases, the amount of switching generated by each adder also increases. To reflect this effect, a term related to the logic depth of the current node k is added, given by k^a , where a is a constant of value 0.55.
6. The overall GPS formula for a node is given by:

$$GPS = r \cdot (k^a) + 0.67(GPS_{input_1} + GPS_{input_2}) \quad (2.11)$$

The GPS is more accurate than the GPC model by taking the adder width into consideration. However, the same as GPC, the GPS model is also an indirect estimation of the switching activity of the adder node. In order to improve the accuracy, a model directly computing the switching activity factor is proposed [67].

Switching-Activity Factor Direct Estimation Method

In the switching activity model proposed in the [67], ripple carry adders (RCAs) are considered. The symbol of a FA constructing the RCA is shown in Figure 2.10, where

Figure 2.10: Symbol of a full adder

X_i , Y_i are the two input operands, Z_i is the carry-in from a preceding bit, S_i and C_i are the outputs of sum and carry-out, respectively.

For each full adder in an RCA, any change (i.e, event) of the three inputs (X_i, Y_i, Z_i) may cause transitions of the outputs (C_i, S_i). The probabilities that the carry-out switches and sum output switches are denoted as $P(C_i)$ and $P(S_i)$, respectively. Thus, for given input switching probabilities of $P(X_i)$, $P(Y_i)$ and $P(Z_i)$, $P(C_i)$ and $P(S_i)$ at a time instant are given by [68]:

$$\left\{ \begin{array}{l} P(C_i) = 0.5[P(X_i) + P(Y_i) + P(Z_i)] + P(X_i)P(Y_i)P(Z_i) \\ \quad - 0.5[P(X_i)P(Y_i) + P(Y_i)P(Z_i) + P(X_i)P(Z_i)] \\ P(S_i) = P(X_i) + P(Y_i) + P(Z_i) + 4P(X_i)P(Y_i)P(Z_i) \\ \quad - 2[P(X_i)P(Y_i) + P(X_i)P(Z_i) + P(Y_i)P(Z_i)]. \end{array} \right. \quad (2.12)$$

Based on (2.12), an event-based model is introduced in [67]. To estimate the switching activity of a MCM block, all the possible input events of any full adders in the MCM block are listed along with their time instants. At each time instant of a event, the

switching probabilities of each output of the full adders are computed based on (2.12). The overall switching factors of the output nodes C_i and S_i , denoted as $\alpha(C_i)$ and $\alpha(S_i)$, are obtained by summing all the switching probabilities of the nodes at all time instants in a clock, i.e.,

$$\alpha(C_i) = \sum_t P(C_i), \quad (2.13)$$

$$\alpha(S_i) = \sum_t P(S_i). \quad (2.14)$$

The overall switching activity factor, α , of i -th full adder thus is given as

$$\alpha = \alpha(C_i) + \alpha(S_i) \quad (2.15)$$

This power model is much more accurate than both of GPC and GPS models for the modeling of dynamic power consumption of adder node. However, the computational complexity of this model is too high to be incorporated into filter optimization. In addition, the models reviewed in this section ignore the static power. With the development of the IC technologies, the static power becomes more and more significant in the total power consumption for technologies 90nm or below [62]. Therefore, besides the simplicity and accuracy, a power model taking both dynamic power and static power into consideration is necessary.

2.4 Summary

The review in this chapter revealed that the current researches for the multiplierless FIR filter designs are mainly to minimize the number of adders used in the structures, either by deterministic tree search method or heuristical methods such as GA. This thesis discussed both methods. Improved GA and tree search algorithms are proposed in Chapters 3, 4 and 5, which circumvent the existing problems and outperform the current researches.

Moreover, the above reviews show that the power consumption of FIR filters is not simply proportional to the number of adders used. In Chapter 6 and 7, optimization techniques to minimize the power consumptions, are proposed. To evaluate the performances of these algorithms, benchmark filters from literatures [3, 50–52, 55, 60, 69] are used. The specifications of these filters are listed in Table 2.1 for easy references.

Table 2.1: The Filter Specifications of the Benchmark Filters Designed and Compared in This Thesis

Filters	Filter order	ω_p	ω_s	δ_p	δ_s	references	Chapters referred
G1	15	0.2π	0.5π	0.01	0.01	[3, 50]	6,7
S1	23	0.3π	0.5π	0.0157	0.0066	[3, 51, 52]	6
F11	24	0.25π	0.3π	0.1355	0.1352	[60]	6
Y1	29	0.3π	0.5π	0.00316	0.00316	[3, 51]	5,6,7
Y2	37	0.3π	0.5π	0.001	0.001	[3, 51, 52]	5,6,7
F7	39	0.15π	0.25π	0.011618	0.011764	[60]	6
Y3	49	0.4π	0.51π	0.01	0.001	[69]	5
A1	58	0.125π	0.225π	0.01	0.001	[3, 55]	3,4,5,6,7
S2	59	0.042π	0.14π	0.012	0.001	[3, 51, 52, 55]	5,6,7
L2	62	0.2π	0.28π	0.028	0.001	[3, 51, 52, 55]	5,7
B	104	0.2π	0.24π	0.01	0.01	[55]	3,4
L1	120	0.8π	0.74π	0.0057	0.0001	[52, 55]	3,4
C	324	0.125π	0.14π	0.005	0.005	[55]	3,4

Chapter 3

Single-Stage and Cascade Design of High Order Multiplierless linear phase FIR Filters Using Genetic Algorithm

As reviewed in Section 2.2, the multiplierless FIR filter design methods can be classified into three categories. The algorithms in the third category [3, 48, 50, 52–55, 61] definitely generate results with lower hardware cost than that of the algorithms in other categories. However, the algorithms in this category have a common problem, i.e., requiring extremely long computation time. The problem becomes more serious when the filter order is high and/or the coefficient wordlength is wide, since the computation time of most of these algorithms increases exponentially along with the increasing of filter orders and coefficient wordlengths. A recently developed algorithm [55] confines the search space of each coefficient to the vicinity, for example the 2 closest discrete values, of the middle values in the feasible range, and thus is capable of designing long filters.

To extend the search space and meanwhile restrict the search time to a tolerable level, in this chapter, a novel GA is proposed for the design of long filters and filters requiring coefficients with wide wordlength. GAs are artificial intelligence techniques based on the principle of “survival of the fittest” [70]. As a stochastic algorithm, a GA can be used to solve complicated problem with a huge search space, and thus is suitable for the problem under consideration. However, a drawback of GAs is that they may fail to find a feasible solution if the majority of the solution candidates in the search space are infeasible. In the filter design problem, when the filter specification is stringent and the hardware restriction is high, existing GAs such as [59, 71, 72] often cannot find a feasible solution.

In order to overcome these limitations, in the proposed GA, the discrete search space is partitioned into smaller ones. Each small space is constructed surrounding a base discrete solution which is obtained by a proposed greedy algorithm with different passband gain. This increases the chances for the GA to find feasible solutions, but does not sacrifice the coverage of the search. In addition, the search in the multiple spaces can run in parallel, and thus the computation time for the design of filters under consideration reduces significantly. Although a MILP based algorithm [29] has been developed for the parallel optimization of FIR filters with signed power-of-two coefficients, the programming and the partitioning of the problem are complicated, and therefore, no attempts have been made for the design of filters sharing adder-and-shift network among coefficients. Moreover, in the proposed GA, an encoding technique is tailored for the partitioned search spaces, and a new fitness function putting more weights on the minimization of hardware cost are proposed.

Besides the design of single-stage FIR filters, a novel integer polynomial factorization technique is proposed such that the proposed GA can be used to design cascade form FIR filters; this technique circumvents the difficulty in the determination of the filter orders

and coefficient effective wordlength (EWL) of the subfilters. In this thesis, the EWL of a coefficient refers to the wordlength excluding the sign bit and the leading zero bits of the coefficient value; and the EWL of a filter refers to the EWL of the coefficient with the maximum magnitude in the filter.

Design examples show that the proposed GA uses much less time in the design of long filters and filters with wide coefficient wordlength. The hardware costs of the filters designed by the proposed GA are less than that designed by the other techniques in most cases, for both the single-stage and cascade form filters.

This chapter is organized as follows. In Section 3.1, the proposed GA for the design of the single-stage filters under consideration is discussed in details. Section 3.2 presents three single-stage design examples. The computational complexity, the selection of parameters, and the superiority and limitation of the algorithm are also discussed. Section 3.3 introduces how to extend the proposed GA to design cascaded FIR filters by the introduction of an integer polynomial factorization technique. Two sets of design examples are given in Section 3.4 to show the advantages of the GA in the design of cascade form filters. Finally, the chapter is concluded in Section 3.5.

3.1 Design Single-Stage FIR Filters Using GA

In this section, first, a new partition technique is introduced to divide the whole search space into several smaller spaces based on different passband gains. After that, adaptive crossover and mutation rates controlled by the fitness values are introduced. Finally, a new encoding form of chromosomes and a fitness function of the GA are proposed.

3.1.1 Partitioning the Search Space

In the multiplierless filter design, if the coefficient space is nonlinear, such as the power-of-two space and subexpression space, floating the passband gain away from the unit may result in designs requiring less hardware. However, this enlarges the search space significantly.

Let N and B be the filter order and the EWL of the filter, respectively. Apparently, after scaling the coefficient values to integers, the dynamic range of each coefficient is within $[-2^B \ 2^B - 1]$. Therefore, for $\lceil \frac{N+1}{2} \rceil$ distinct coefficients in a linear phase FIR filter, the number of possible combinations of discrete solutions is as large as $2^{\lceil \frac{N+1}{2} \rceil (B+1)}$. For a moderate long (less than 70) and bit (less than 12) wide filter, for example N and B taken as 49 and 12, respectively, the number of possible discrete solutions in the whole search space will be 2^{325} . However, the majority of these solutions are infeasible, i.e., they do not meet the filter specification, when the specification is reasonably stringent. Therefore, in most cases, it is very difficult to find feasible solutions using conventional GAs. To circumvent this problem, the search spaces are confined to the neighbors of several discrete coefficient sets. The discrete coefficient sets are obtained using a successive reoptimization approach for the selected passband gains. The way to select the passband gains, G_m , for $m=0,1,\dots, M-1$ is discussed in the next subsection. For each G_m , the discrete coefficient set is found in the following two steps.

Step 1: The optimum solution with continuous coefficient values meeting the filter specifications with a passband gain G_m is found using a linear programming formulated as [51]:

$$\begin{aligned} \text{Minimize : } & f = \delta \\ \text{Subject to : } & G_m - \delta \leq H(\omega) \leq G_m + \delta, \quad \text{for } \omega \in [0, \omega_p] \\ & -(\delta_s \delta) / \delta_p \leq H(\omega) \leq (\delta_s \delta) / \delta_p, \quad \text{for } \omega \in [\omega_s, \pi] \end{aligned} \quad (3.1)$$

where ω_p, ω_s are the passband and stopband edges, δ_p, δ_s are the passband and stopband ripple tolerances of the ideal passband and stopband gains away from 1 and 0, respectively, and $H(\omega)$ is the zero-phase frequency response of the filter to be optimized, given by

$$H(\omega) = \sum_{n=0}^{\lfloor \frac{N}{2} \rfloor} h(n) \text{Trig}(\omega, n) \quad (3.2)$$

In (3.2), $h(n)$ is the unknown symmetric impulse response of the filter to be optimized and $\text{Trig}(\omega, n)$ is an appropriate trigonometric function depending on the parity of N and symmetry of the filter. By solving the problem formulated in (3.1), the obtained $h(n)$ is the impulse response of a filter whose frequency response has a passband gain G_m . Divided by G_m , the impulse response can be normalized to unit passband gain. Let an index i be zero.

Step 2: Round $h(i)$ to a prespecified wordlength. If $i = \lfloor \frac{N}{2} \rfloor$, meaning that every coefficient has been rounded to a discrete value, the program stops. Otherwise, reoptimize the rest unfixed coefficient by substituting those fixed coefficient values into $H(\omega)$ and re-solve (3.1); $i = i + 1$ and repeat Step 2.

The above method is a greedy successive reoptimization algorithm. The flow chart of the greedy algorithm to find the discrete coefficient set is shown Figure 3.1. In each iteration of Step 2, the algorithm rounds a coefficient into a discrete value and reoptimizes the rest continuous coefficients to compensate for the degradation of the frequency response. Using the above method, M sets of discrete coefficient values are obtained, one for each passband gain G_m , for $0 \leq m \leq M - 1$. Denoting the discrete coefficient set obtained using gain G_m as $h_m(n)$ for $n=0,1,\dots,\lfloor \frac{N}{2} \rfloor$, to simplify the representation, $h_m(n)$ are scaled to integers and denoted as $h_{q_m}(n)$, i.e.,

Figure 3.1: The flow chart of the greedy algorithm in determining the base discrete coefficient set for a given passband gain G_m

$$h_{q_m}(n) = h_m(n) \times 2^{S_m} \quad (3.3)$$

where S_m is an integer making $\|\mathbf{h}_{q_m}\|_\infty$ containing the prespecified EWL; $\mathbf{h}_{q_m}$ represents the vector $[h_{q_m}(0), h_{q_m}(1), \dots, h_{q_m}(\lfloor \frac{N}{2} \rfloor)]^T$ and $\|\mathbf{h}_{q_m}\|_\infty$ stands for the norm- ∞ of the vector $\mathbf{h}_{q_m}$, i.e, the maximum magnitude of $\mathbf{h}_{q_m}$. Coefficient set $\mathbf{h}_{q_m}$ for $m=0,1,2,\dots,M-1$ is named as the base discrete coefficient sets for the construction of subspaces. Thus, for each passband gain G_m and the base corresponding coefficient set, an $\lceil \frac{N+1}{2} \rceil$ -dimensional search space is constructed as follows. The lower bound and upper bound, denoted as $h_{q_m}^l(n)$ and $h_{q_m}^u(n)$, of n -th dimension, for $n = 0, 1, \dots, \lfloor \frac{N}{2} \rfloor$, respectively, are empirically determined as:

$$\begin{cases} h_{q_m}^u(n) = h_{q_m}(n) + 2^{\text{ceil}(B_m(n)/3)} - 1 \\ h_{q_m}^l(n) = h_{q_m}(n) - 2^{\text{ceil}(B_m(n)/3)} + 1 \end{cases} \quad (3.4)$$

for the trade-off between the feasibility and optimum of the solution, where $B_m(n)$ is the EWL of the n -th coefficient. In such a way, the coefficient with a larger magnitude has a larger range for variation in the discrete coefficient space. For example, for a particular passband gain G_m , if the resultant base discrete coefficient set is $\{3, 9, 21\}$ and the corresponding EWL of each coefficient value is 2, 4 and 5, respectively. The lower bound and upper bound of each discrete coefficient are computed according to (3.4), and therefore, each coefficient value may vary in the range of $[2, \underline{3}, 4]$, $[6, 7, 8, \underline{9}, 10, 11, 12]$ and $[18, 19, \dots, \underline{21}, \dots, 24]$, respectively, where the underlined values are $h_{q_m}(n)$.

3.1.2 Determining the Gains

For a given filter specification, when the passband gain is G_m or $2G_m$, the discrete coefficient sets obtained from the successive reoptimization algorithm are the same for a given EWL. Therefore, only the passband gain varying in a range from A to $2A$ for any

positive A is necessary to be considered. The parameter A is chosen to be 0.7 as [3] and thus G_m may vary in [0.7 1.4]. First, the variation range of the passband gain is divided into to M sections in [0.7 1.4] as that in [3]; M is set to 20 in the proposed algorithm, and thus G_m is confined to a range of Γ_m , given by $[0.7+0.035m \ 0.7+0.035(m+1)]$ for $m=0,1,2,\dots,M-1$. Each range Γ_m is further discretized by a stepsize of 0.001, and thus G_m may take any value from the 35 possible values in Γ_m . Let the 35 possible values be $G_{m,j}$ for $j=0,1,2,\dots,34$. The discrete coefficient set for each $G_{m,j}$, denoted as $h_{qm,j}(n)$, obtained by the proposed greedy algorithm discussed in Section 3.1.2, leads to a filter with normalized magnitude ripples $\delta_{m,j}$ computed by

$$\left\{ \begin{array}{l} H(\omega) = \sum_{n=0}^{\lfloor \frac{N}{2} \rfloor} h_{qm,j}(n) \text{Trig}(\omega, n) \\ \text{gain} = \frac{\max(H(\omega)) + \min(H(\omega))}{2}, \text{ for } \omega \in [0, \omega_p] \\ \hat{\delta}_p = \frac{\max(H(\omega)) - \min(H(\omega))}{2}, \text{ for } \omega \in [0, \omega_p] \\ \hat{\delta}_s = \max(\text{abs}(H(\omega))), \text{ for } \omega \in [\omega_s, \pi] \\ \delta_{m,j} = \max(\frac{\hat{\delta}_p}{\text{gain}}, \frac{\hat{\delta}_s}{\text{gain}} * \frac{\delta_p}{\delta_s}) \end{array} \right. \quad (3.5)$$

Among all designed filters, G_m is selected to be $G_{m,i}$ whose corresponding $\delta_{m,i}$ is minimum among $\delta_{m,j}$ for all $j=0,1,2,\dots,34$. With the selected G_m for $m=0,1,2,\dots,19$, twenty spaces are constructed based on the 20 base discrete coefficient sets according to the method proposed in Section 3.1.2, and the proposed GA is applied to each search space separately.

3.1.3 Adaptive Crossover and Mutation Rates

During the generational process, crossover and mutation are the two basic operations to generate the offsprings. The crossover and mutation rates, denoted as P_c and P_m , play a vital role in the GA algorithm. The higher P_c and P_m are, the more variations

are introduced into the offsprings; this prevents the premature convergence of the GA and increases the search efficiency. However, large P_c and P_m may lead to the solution unstable, i.e., fitter chromosomes have the same probability to be cast away during the generational process as unfitter chromosomes. Therefore, in the proposed GA, instead of keeping P_c and P_m unchanged during the whole process, P_c and P_m are adaptively adjusted according to the fitness values of the chromosomes, expressed as :

$$\begin{aligned} P_c &= \begin{cases} \lambda_1 \frac{F_{max} - F^L}{F_{max} - F_{mean}} + \lambda_0, & \text{for } F^L \geq F_{mean} \\ \lambda_1 + \lambda_0, & \text{otherwise} \end{cases} \\ P_m &= \begin{cases} \lambda_2 \frac{F_{max} - F}{F_{max} - F_{mean}} + \frac{\lambda_0}{100}, & \text{for } F \geq F_{mean} \\ \lambda_2 + \frac{\lambda_0}{100}, & \text{otherwise} \end{cases} \end{aligned} \quad (3.6)$$

where F_{max} and F_{mean} are the maximum and mean values of the overall fitness values in the current generation, respectively, F^L is the larger fitness value of the two parent chromosomes that are selected for crossover operation, and F is the fitness value of the chromosome that is to mutate. λ_1 and λ_2 are weighting parameters and λ_0 is a constant. According to (3.6), the chromosome with smaller fitness value has larger P_m and P_c ; thus, their codons in the chromosomes are recombined more frequently. On the contrary, for the fittest chromosome in the current generation, their P_m and P_c are λ_0 and $\lambda_0/100$, the minimum mutation and crossover rates, so that the good codons are preserved. The selection of parameters in (3.6) is presented in Section 3.2.3.

3.1.4 Encoding and Initial Population

In the proposed algorithm, chromosomes are encoded to ternary strings of $\{\bar{1}, 0, 1\}$. Since the search space in the proposed GA has been confined to a small range around the base discrete coefficient set, each element in the population is a small deviation from the base discrete values. Therefore, to encode the discrete coefficient values in the population,

instead of encoding the discrete coefficient values themselves, the deviations from the base discrete values are encoded to reduce the length of the chromosomes. To illustrate this encoding method, the same example in Section 3.1.2 is used, where the base discrete coefficient values are $\{3, 9, 21\}$, and the lower bound and upper bound for each value are $[2, 4]$, $[6, 12]$ and $[18, 24]$, respectively. Thus, the possible largest deviations are 1, 3 and 3, respectively. Therefore, 5 digits are sufficient to encode all elements in the given search space. For example, an element of $\{4, 8, 20\}$ is encoded to be $10\bar{1}01$. If the coefficient values are directly encoded to a ternary string, 12 digits are needed.

With the above encoding technique, a chromosome with an all-zero string represents the base discrete coefficient set, and such chromosome is named as an “ancestor chromosome”. Design experiences show that for filters with length less than 35, the proposed GA is robust to find feasible solutions even with a randomly produced population in most cases; however, when the filter length is longer, a good initial population is very important for the GA to find a feasible solution. In the proposed GA, half of the total population is produced by perturbing the “ancestor chromosome” and the other half is randomly produced in the respectively constructed space. The randomly produced population is to diversify the gene pool and thus helps to prevent the premature convergence.

3.1.5 Fitness Function

To minimize the hardware cost, the objective function f_g of the proposed GA is set to be the number of adders. After each chromosome is encoded, a MCM algorithm is used to compute the number of adders used to synthesize the coefficient values. The C1 [38] algorithm is used for the reason that this algorithm usually outperforms other MCM algorithms in terms of number adders used to synthesize the discrete coefficient values .

However, as mentioned in Section 3.1.2, the major elements in the search space cannot meet the given filter specification; for such chromosomes, even their corresponding coefficient sets can be synthesized using fewer adders, they should be cast away during the selection process gradually. A common way to cast away infeasible elements is to add a penalty to the objective function when the filter specification is not satisfied, and the fitness values are thus significantly reduced for these cases. The proposed fitness function is given as,

$$F = \begin{cases} 1/f_g & \text{if } \delta_m \leq \delta_p \\ 1/(P_e \times \frac{\delta_m}{\delta_p}) & \text{if } \delta_m > \delta_p \end{cases} \quad (3.7)$$

where, f_g is the total number of adders used to synthesize the coefficient values of the chromosome, δ_m is the ripple of the filter represented by the chromosome calculated as in (3.5), δ_p is the given ripple specification and P_e is a penalty value, given by

$$P_e = \sum_{n=0}^{\lfloor \frac{N}{2} \rfloor} \text{Maxaddercost}(h_{qm}(n)) \quad (3.8)$$

Here, $\text{Maxaddercost}(h_{qm}(n))$ is the maximum adder cost to synthesize all possible values of $h_{qm}(n)$ in its range $[h_{qm}^l(n) \ h_{qm}^u(n)]$ given in (3.4). For example, if a coefficient value is in the range of $[2,3,4]$, the corresponding adder costs to synthesize these discrete values are 0, 1 and 0, respectively. So the maximum adder cost for this coefficient is 1. Computing P_e according to (3.8) guarantees that the fitness values of the infeasible individuals are smaller than that of the feasible individuals. Thus, during the generational process, the infeasible solutions will be gradually cast away. Meanwhile, in (3.7) for the coefficient sets not meeting the ripple specification, the coefficient sets with smaller ripples have bigger fitness values, ensuring that they have bigger chances to survive in the selection process. This helps the program to find out more coefficient sets meeting the given filter specification.

With the introduction of the penalty function, the MCM algorithm is not applied to the coefficient sets not meeting the given filter ripple requirement. When the majority of the coefficient sets in the population are infeasible, the proposed fitness function significantly reduce the runtime of the program.

3.1.6 Summary of the Proposed GA

The step-by-step design procedure is shown in Figure 3.2.

1. Determine the passband gains G_m , for $m=0,1,2,\dots,19$, using the method proposed in Section 3.1.3.
2. For each G_m obtained in step 1, find the base discrete coefficient set and construct the corresponding search space. Run the GA using the fitness function given in (3.7) and the adaptive crossover and mutation rates given in (3.6). When the GA operation is terminated, the chromosome with the biggest fitness value is recorded. Since G_m has 20 different values, in total there are 20 sub-problems of the GA. Because the sub-problems of the GA are independent to each other, they can be run in parallel, resulting in a significant reduction of total runtime.
3. If the final optimum solution does not meet the given filter specification, the filter length N or the EWL has to be increased, and the whole procedure is repeated.

3.2 Numerical Example of Single-Stage Design

Three benchmark filters (B, L1 and C) are designed to show the superiority and limitation of the proposed algorithm in the design of high order and wide coefficient wordlength FIR filters. The specifications of the three filters are listed in Table 2.1. The specifications

Figure 3.2: The flow chart of the proposed GA

Table 3.1: Results and Comparison of the Three Benchmark Filters, where MBA and SA are the Number of Multiplier Block Adders and Structure Adders

	Best published references / proposed GA		
Filters	B [55]	L1 [55]	C [55]
MBA	11 / 10	47 / 45	22 / 31
SA	100 / 98	120 / 120	306 / 306
Total adders	111 / 108	167 / 165	328 / 337

show that the order of the three filters are all high and meanwhile filter L1 with wide coefficient wordlength.

3.2.1 Simulations Results

In the proposed GA, the population size is set to 400 and the running of the GA is terminated when the generation reaches 100. The Roulette Wheel Selection [73] is used to select the chromosomes for crossover and mutation. The determination of the other GA parameters will be discussed in Section 3.2.3. MATLAB is the computer language used to develop the program. The 20 sub-problems of each example are casted to 20 computers, and each computer has two cores with CPU speed of 3.2G Hz.

Table 3.1 lists the simulation results, where the number of adders is given in terms of MBAs and SAs. It can be seen that for the filters with length less than 150, the proposed GA generates designs using less number of adders. However, for the filter with length longer than 300, the proposed GA cannot beat the design obtained in [55] in terms of number of adders. The reasons for this are analyzed in Section 3.2.4. The coefficient values of the filters B and L1 obtained by the proposed GA are listed in Tables 3.2 and 3.4, respectively.

Table 3.2: The Coefficient Values of Filter B

Filter B: $h(n) = h(104 - n)$ for $0 \leq n \leq 52$; passband gain: 907.66, EWL: 8 normalized passband ripple: 0.00995; normalized stopband ripple : 0.00995				
$h(0) = -2$	$h(14) = 3$	$h(28) = -7$	$h(42) = 16$	
$h(1) = -1$	$h(15) = 2$	$h(29) = -2$	$h(43) = -2$	
$h(2) = 0$	$h(16) = -1$	$h(30) = 5$	$h(44) = -24$	
$h(3) = 1$	$h(17) = -4$	$h(31) = 10$	$h(45) = -40$	
$h(4) = 2$	$h(18) = -5$	$h(32) = 11$	$h(46) = -40$	
$h(5) = 2$	$h(19) = -4$	$h(33) = 7$	$h(47) = -18$	
$h(6) = 1$	$h(20) = 0$	$h(34) = -2$	$h(48) = 27$	
$h(7) = -1$	$h(21) = 3$	$h(35) = -11$	$h(49) = 84$	
$h(8) = -2$	$h(22) = 6$	$h(36) = -16$	$h(50) = 142$	
$h(9) = -2$	$h(23) = 6$	$h(37) = -14$	$h(51) = 185$	
$h(10) = -2$	$h(24) = 4$	$h(38) = -5$	$h(52) = 200$	
$h(11) = 0$	$h(25) = -1$	$h(39) = 9$		
$h(12) = 2$	$h(26) = -6$	$h(40) = 20$		
$h(13) = 3$	$h(27) = -9$	$h(41) = 24$		

Table 3.3: The synthesis of coefficient set of Filter B

subexpression: 7,3,5,9,11,27,21,71,185,25			
$7 = -1 \times 1 + 1 \times 8$	$11 = 5 \times 1 + 2 \times 1$	$185 = 3 \times 7 - 64 \times 1$	
$3 = 1 \times 1 + 2 \times 1$	$27 = -5 \times 1 + 1 \times 32$	$25 = 3 \times 1 + 8 \times 1$	
$5 = 1 \times 1 + 4 \times 1$	$21 = 5 \times 1 + 4 \times 1$		
$9 = 1 \times 1 + 8 \times 1$	$71 = 7 \times 1 + 1 \times 64$		

3.2.2 Computational Complexity

The runtime of the proposed GA for the design of the filters L1, B and C are 26 minutes 41 seconds, 16 minutes 24 seconds and 3 hours 49 minutes, respectively, when the partitioned problems are cast to 20 machines. When only one computer is used but the parallel toolbox in MATLAB is exploited, the runtimes of the examples L1, B and C are 4 hours 31 minutes, 2 hours 41 minutes and 37 hours 9 minutes, respectively. However, the runtimes cannot accurately reflect the computational complexity of the algorithms due to various factors such as the efficiency of the computer languages, the speed of the computer, etc. In the following, the computational complexity of the algorithms are

Table 3.4: The Coefficient Values of Filter L1

Filter L1: $h(n) = h(120 - n)$ for $0 \leq n \leq 60$; passband gain:56986.8943, EWL: 14 normalized passband ripple: 0.00537 ; normalized stopband ripple: 0.000094368				
$h(0) = 4$	$h(16) = -61$	$h(32) = 280$	$h(48) = 1144$	
$h(1) = -10$	$h(17) = 108$	$h(33) = -53$	$h(49) = -1518$	
$h(2) = 17$	$h(18) = -111$	$h(34) = -238$	$h(50) = 1171$	
$h(3) = -21$	$h(19) = 59$	$h(35) = 454$	$h(51) = -107$	
$h(4) = 19$	$h(20) = 32$	$h(36) = -472$	$h(52) = -1311$	
$h(5) = -7$	$h(21) = -123$	$h(37) = 251$	$h(53) = 2445$	
$h(6) = -12$	$h(22) = 167$	$h(38) = 134$	$h(54) = -2603$	
$h(7) = 32$	$h(23) = -132$	$h(39) = -512$	$h(55) = 1317$	
$h(8) = -44$	$h(24) = 22$	$h(40) = 686$	$h(56) = 1434$	
$h(9) = 39$	$h(25) = 121$	$h(41) = -533$	$h(57) = -5148$	
$h(10) = -14$	$h(26) = -228$	$h(42) = 76$	$h(58) = 8921$	
$h(11) = -24$	$h(27) = 238$	$h(43) = 502$	$h(59) = -11728$	
$h(12) = 59$	$h(28) = -128$	$h(44) = -923$	$h(60) = 12765$	
$h(13) = -73$	$h(29) = -66$	$h(45) = 941$		
$h(14) = 54$	$h(30) = 259$	$h(46) = -470$		
$h(15) = -3$	$h(31) = -352$	$h(47) = -342$		

Table 3.5: The synthesis of coefficient set of Filter L1

subexpression: 5,17,7,3,33,21,39,73,61,27,111,123,121,57,259,11,35,119,59,67,19,251,143,167,53 227,533,107,1287,923,941,235,171,759,1171,1311,1317,717,733,343,2603,3191,12765,2445,8921			
5 = $1 \times 1 + 4 \times 1$	11 = $5 \times 1 + 2 \times 1$	941 = $59 \times 3 - 16 \times 1$	
17 = $1 \times 1 + 16 \times 1$	35 = $17 \times 1 + 2 \times 1$	235 = $227 \times 1 + 1 \times 8$	
7 = $-1 \times 1 + 1 \times 8$	119 = $-17 \times 17 + 1 \times 8$	171 = $167 \times 1 + 1 \times 4$	
3 = $1 \times 1 + 2 \times 1$	59 = $-5 \times 1 + 1 \times 64$	759 = $119 \times 5 + 1 \times 128$	
33 = $1 \times 1 + 32 \times 1$	67 = $33 \times 1 + 2 \times 1$	1171 = $73 \times 3 + 16 \times 1$	
21 = $5 \times 1 + 4 \times 1$	19 = $17 \times 1 + 1 \times 2$	1311 = $1287 \times 3 + 1 \times 8$	
39 = $7 \times 1 + 1 \times 32$	251 = $-5 \times 1 + 1 \times 256$	1317 = $-27 \times 21 + 1 \times 64$	
73 = $17 \times 5 + 4 \times 1$	143 = $-17 \times 5 + 1 \times 32$	717 = $-259 \times 61 + 1 \times 16$	
61 = $-3 \times 1 + 1 \times 64$	167 = $7 \times 5 + 1 \times 32$	733 = $-35 \times 3 + 1 \times 256$	
27 = $-5 \times 1 + 1 \times 32$	53 = $3 \times 5 + 16 \times 1$	343 = $21 \times 7 + 16 \times 1$	
111 = $-17 \times 1 + 1 \times 128$	227 = $3 \times 7 + 1 \times 32$	2603 = $107 \times 39 + 1 \times 64$	
123 = $-5 \times 1 + 1 \times 128$	533 = $33 \times 5 + 16 \times 1$	3191 = $119 \times 3 + 1 \times 1024$	
121 = $-7 \times 1 + 1 \times 128$	107 = $7 \times 5 - 16 \times 1$	12765 = $3191 \times 1 + 4 \times 1$	
57 = $7 \times 1 + 8 \times 1$	1287 = $7 \times 5 + 1 \times 256$	2445 = $-227 \times 167 + 1 \times 16$	

analyzed based on the number of linear programming to be solved.

In the proposed algorithm, the computation contains two parts: one is to find the base discrete coefficient sets and the other is to search in the subspaces constructed based on the base discrete coefficient sets. Both parts repeatedly compute the filter ripple by solving the linear programming. Let N , P and G be the order of the filter, the population size and the number of generations to be updated in the GA, respectively. The total number of linear programmings for the first part to find the base discrete coefficient sets, expressed as LP_{N1} is given by

$$LP_{N1} = J \left\lceil \frac{N+1}{2} \right\rceil \quad (3.9)$$

where J is the number of passband gains in each section. In the proposed algorithm $J = 35$.

In the second part, the number of linear programmings to be done, denoted as LP_{N2} , is determined by

$$LP_{N2} = PG \quad (3.10)$$

where $P = 400$ and $G = 100$ in the proposed algorithm. The computation time for the selection, crossover and mutations operations is much shorter than that for the linear programming and is ignored in the analysis. Let the average runtime of a linear programming be T . The overall computation time of the proposed algorithm is

$$ComputationTime = M \left(J \left\lceil \frac{N+1}{2} \right\rceil + PG \right) T \quad (3.11)$$

where M is the total number of partitioned subspaces.

In comparison, the total iterations needed to be run in the algorithm [55] is $L^{\lceil \frac{N+1}{2} \rceil}$, where L is the size of the refined value set. L is set to 2 in [55]. Each iteration requires two

linear programmings to compute the lower bound and upper bound of the corresponding coefficient. The overall computation time of the algorithm in [55] can be expressed as

$$ComputationTime = 2\beta L^{\lceil \frac{N+1}{2} \rceil} T \quad (3.12)$$

where β is a factor influenced by many cutoff “mechanisms”.

From (3.11) and (3.12), it can be seen that the computational complexity of the proposed algorithm increases linearly with the increasing of N , whereas that for the algorithm in [55] increases exponentially with the increasing of N . When N is large, the proposed algorithm has much lower computational complexity. The cutoff factor β may alleviate the computational complexity but it can not change the exponential relation by nature. For example, assuming that the cutoff factor β is 10^{-8} (which in practical is much larger), to design filter B (with order 104), the total computation time of the algorithm in [55] is $(1.8 \times 10^8)T$, whereas for the proposed GA it is $(8.364 \times 10^5)T$. The later one is much smaller than the former one. That is the reason why [55] need to be manually terminated after running 24h for filters B and C. If let the program finish the whole search, it may takes several days or even several weeks. Furthermore, the proposed GA can be easily programmed to be run in parallel, but it is much more complicated for algorithm in [55]. The reason that the running time to design filter C by a single computer using the proposed technique shows longer is because the proposed algorithm is developed in Matlab. As an interpreted language, the runtime of Matlab for a problem is about 6 times of that of the compiled language used in [55], according to a rough estimation.

3.2.3 Determining the Parameters λ_1 , λ_2 and λ_0

For traditional GA, it is difficult to exactly determine the optimal values of P_m and P_c , because there are many random factors impacting on the search process. One of the goals of adaptive mutation and crossover rates is to ease the users' burden to specify P_m and P_c , since the performance of the adaptive GA is not significantly influenced by the weighting parameters λ_1 and λ_2 . The reason to introduce a constant λ_0 is because without it, the crossover and mutation rates for the best solution are 0. In such a case, the genes of the best solution in a population do not contribute to the next generation through crossover and mutation. By setting λ_0 to be a very small value (less than 0.01), on one hand the best solution is persevered and on the other hand the good "genes" have chances to be inherited by its children.

The general rule to choose λ_1 and λ_2 is that they are selected to be values relatively larger than the crossover and mutation rates in the traditional GA, respectively. This is to increase the probabilities of the individuals, whose fitness values are less than the average fitness value, to breed new individuals through crossover and mutation. The classical crossover and mutation rates for traditional GA are in the range from 0.8 to 0.95 and from 0.005 to 0.05 [70], respectively. In the proposed GA, λ_1 and λ_2 with values not smaller than 0.8 and 0.05, respectively, are tested. Figure 3.3 shows the resultant mean fitness value vs. the generations for (λ_1, λ_2) to be (0.8, 0.05), (0.9, 0.1) and (1, 0.15) respectively, for the design of filter B with G_m for $m = 10$. For comparison, plots for the traditional GA with P_c and P_m to be (0.8, 0.0098), (0.9, 0.0198) and (1, 0.027), respectively, are also shown in Figure 3.3. It shows that for the three cases in adaptive GA, the convergence speeds are slightly different, but they all converge to the same mean fitness value. However, the traditional GA tends to converge to a local poorer optimal point.

Figure 3.3: The mean fitness values changes with the number of generations, AGA stands for adaptive GA, and TGA stands for traditional GA

However, it should be noted that although the program converges to the same results with different values of λ_1 , λ_2 and λ_0 , it cannot guarantee that the results are optimum by evolutions with limited number of generations.

In the design examples presented in this chapter, λ_1 , λ_2 and λ_0 are set to 0.9, 0.1 and 0.005, respectively.

3.2.4 Superiority and Limitation

The simulation results in Section 3.2.1 and the computational complexity analysis in Section 3.2.2 showed that the proposed algorithm designs filters with reduced hardware cost and reduced design time in most cases. However, when the filter order is as large as 300, such as the example filter C, the proposed GA cannot beat the design obtained in [55].

This is because the computational complexity of the algorithm is kept to be linearly proportional to the filter order as shown in (3.11), where the parameters M , J , P and G

are all set to be constant independent of N . When N increases, the search space for each subproblem thus increases significantly. For example, when the filter order is 120, the size of the search space is around 3^{60} ; when the filter order is increased to 300, the size of the search space is around 3^{150} . Thus the GA cannot obtain optimum solution when the population size (P) and the generation of GA (G) do not increase accordingly. In other words, the optimality is sacrificed to maintain the problem to be tractable, and the algorithm provides a promising approach to design long filters in a tractable time.

In the end, a general speculation to the efficiency of the proposed GA is that the GA is more efficient in the design of filters with order less than 300. This is for the reason that the efficiency of the proposed GA depends on the size of search space and the amount of feasible solutions in the space. The larger search space and less feasible solutions, the lower efficiency of the algorithm. When the filters are of more stringent specifications, the search space does not change according to the proposed algorithm, but the number of feasible solutions is reduced, such that the efficiency is also reduced. This is opposite to MILP which in general is more efficient when the specifications are more stringent.

3.3 Design Cascaded FIR Filters using the Proposed GA

FIR filters in cascade structure are well-known for their low computational complexity, compared with the single-stage design [58]. Many researches [69, 74–76] have attempted to design multiplierless FIR filters in cascade form. However, existing algorithms either can not design long filters [69] or fail to beat the state-of-the-art single-stage design techniques [3, 55].

While GAs have been successfully developed for the design of single-stage FIR filters

[59, 71, 72], they do not show significant advantages in the design of subfilters in cascade form [74]. One of the difficulties in the design of cascade form FIR filter with discrete coefficients is the determination of the subfilter orders. GAs cannot provide an efficient technique to evolve automatically to an order combination of subfilters that can lead to a low hardware cost. In addition, in earlier algorithms for the design of cascade form multiplierless FIR filters [74–76], in order to obtain feasible cascade solutions, the EWL of each subfilter is not properly controlled. In these algorithms considering 2-stage cascade, although the EWL of the coefficients of each subfilter is shorter or equal to the minimum that can be achieved by the single-stage design, due to the cascade, the width of the SAs of the second subfilter significantly increases. Therefore, the cascade form filters designed using such approaches often have higher hardware cost than that of the best single-stage design.

In this section, a GA for the design of 2-stage cascade form filters is proposed. The GA search is performed on single-stage filters, and an integer coefficient polynomial factorization technique which results in two discrete coefficient polynomials is proposed. In such a manner, the above two difficulties in the design of cascade form filters are circumvented, i.e. the GA need not to determine the orders of the subfilters, and the EWLs of the subfilters are controlled by the single-stage design, so that the EWLs of the subfilters are kept low.

3.3.1 Integer Coefficient Polynomial Factorization

For a given integer coefficient $(N-1)$ -th order polynomial, denoted as $H_d(z) = \sum_{n=0}^{N-1} h_d(n)z^{-n}$, where $h_d(n)$ are integers for $n=0,1,2,\dots,N-1$, there are $N-1$ roots (including real and complex conjugated roots). Denoting the j -th real root as R_j and the k -th complex root as $r_k e^{-i\theta_k}$, where r_k and θ_k are the magnitude and phase of the corresponding complex,

respectively, the polynomial $H_d(z)$ can be factorized as:

$$H_d(z) = \beta \prod_{k=1}^K (z^{-1} + r_k e^{-i\theta_k}) \prod_{j=1}^J (z^{-1} + R_j) \quad (3.13)$$

where $K + J = N - 1$, and β is a scaling constant.

If every pair of the factors that are complex conjugated roots are multiplied, $H_d(z)$ can be rewritten as:

$$H_d(z) = \beta \prod_{k=1}^{K/2} (z^{-2} + 2r_k \cos(\theta_k) z^{-1} + r_k^2) \prod_{j=1}^J (z^{-1} + R_j) \quad (3.14)$$

which consists of J first-order factors and $K/2$ second-order factors, all containing real coefficients. Depending on if the coefficient values of a factor are all integers or not, these factors are classified into integer factors and non-integer factors. If the factors can be cast into several groups, and the product of each group is an integer coefficient polynomial, respectively, the original polynomial is successfully factorized. In this work, only the cascade of two subfilters, i.e. the polynomial is factorized to two integer coefficient polynomials, is considered. Therefore, the factorization problem is transferred to how these factors can be allocated to two “baskets”, each corresponding to a sub-polynomial, such that the product of the factors in each basket, scaled by a proper constant, is an integer coefficient polynomial. Denoting the two baskets as B1 and B2 and their scalars as β_1 and β_2 , respectively, $\beta_1 \beta_2$ has to be equal to β .

When the polynomial factors are allocated to a basket, the resultant basket may be an integer basket, which contains only integer factors, or a non-integer basket, which contains only non-integer factors, or a mixed basket, which contains both integer and non-integer factors. Obviously, the product of the factors in an integer basket is an integer polynomial. On the other hand, the product of the factors in a non-integer basket and that in a mixed basket with some proper scalars, also have potentials to be

an integer coefficient polynomial. If all the combinations of the allocation of factors into the two baskets B1 and B2 are tested, the problem is in combination complexity and therefore is intractable. For this reason, the factor allocation is restricted such that B1 can be allocated with integer factors only, and B2 can be allocated with either integer or non-integer factors, i.e, B1 is an integer basket taking scaling factor $\beta_1 = 1$ and B2 may be either a non-integer basket or a mixed basket taking a scaling factor $\beta_2 = \beta$.

Since the number of integer factors in the polynomial factorization is limited, the problem is significantly simplified. First, all the integer factors obtained in the factorization of the original integer coefficient polynomial are listed. Second, all the combinations of the integer factors which can be allocated to B1 are listed. For example, if there are 2 integer factors, B1 thus can be allocated with either of them or both of them, and in total, there are 3 combinations. For each combination in B1, the corresponding rest factors are allocated to B2. Finally, the product of the factors in B2 is checked; if the resultant polynomial contains only integer coefficients, the factorization of the integer coefficient polynomial to two integer coefficient polynomials is successful and the result is recorded.

However, in the factorization of (3.14), all factors are in the normalized form that the coefficient of the highest order term is unit. This excludes the factors such as $2z^{-1} + 5$, $5z^{-1} + 1$ or $2z^{-2} + z^{-1} + 2$, from the integer factors. To secure such integer factors, every non-integer factor is tested to check if the factor can be converted to an integer factor by a converting constant.

Theoretically, the non-integer factors with rational number coefficients can be converted to integer factors if the coefficients are multiplied by a suitable integer, for example, a non-integer factor $z^{-1} + 0.1189$ can be converted to an integer factor if it is multiplied by a converting constant, 10000 in this example. However, when the converting constant

is too large, although the resultant factor is an integer factor, the wordlength of the coefficient is too long to be implemented by using a limited number of adders. Therefore, the converting constant, denoted as C , is bounded to $\max(|h_d(n)|)/2$. If a non-integer factor is converted to an integer factor by converting constant C , the overall scaling factor β is adjusted to β/C .

With all the above considerations, for a given $(N - 1)$ -th order polynomial $H_d(z)$ with integer coefficients $h_d(n)$ for $n = 1, 2, \dots, N - 1$, the details of its factorization are summarized as follows:

1. Factorizing $H_d(z)$ into the form in (3.14) by finding all the roots of the polynomial. Integer factors and non-integer factors are classified.
2. Converting the non-integer factors into integer factors, if possible. If there are Q convertible non-integer factors, and the corresponding converting factors are C_i , for $i = 1, 2, \dots, Q$, adjusting β to $\beta / \prod_{i=1}^Q C_i$.
3. If even after conversion, there is no integer factors, the polynomial of the given coefficient set cannot be factorized into two integer sub-polynomials; otherwise, all the combinations of the integer factors are listed.
4. For each combination of the integer factors, checking whether the product of the rest factors scaled by $\beta_2 = \beta$ is a polynomial with integer coefficients. If yes, the factorization is successful.

3.3.2 An Example to Illustrate the Factorization Technique

A polynomial p with an integer coefficient set $\{ 40, 140, 314, 473, 551, 473, 314, 140, 40 \}$ is given to illustrate the factorization technique in details. The roots of the polynomial p

Table 3.6: An Example to Factorize a Discrete Coefficient Polynomial

B1	B2	Success
$2z^{-2} + z^{-1} + 2$	$10(z^{-2} + 0.0699z^{-1} + 1)(z^{-2} + 1.4301z^{-1} + 1)(2z^{-2} + 3z^{-1} + 2)$	Y
$2z^{-2} + 3z^{-1} + 2$	$10(z^{-2} + 0.0699z^{-1} + 1)(z^{-2} + 1.4301z^{-1} + 1)(2z^{-2} + z^{-1} + 2)$	Y
$(2z^{-2} + z^{-1} + 2)(2z^{-2} + 3z^{-1} + 2)$	$10(z^{-2} + 0.0699z^{-1} + 1)(z^{-2} + 1.4301z^{-1} + 1)$	Y

are computed by using Matlab function “roots”, and they are $-0.25 + 0.9682i$, $-0.25 - 0.9682i$, $-0.0350 + 0.9994i$, $-0.0350 - 0.9994i$, $-0.7150 + 0.6991i$, $-0.7150 - 0.6991i$, $-0.75 + 0.6614i$ and $-0.75 - 0.6614i$, respectively, with a scaling constant $\beta = 40$. By pairing the conjugated roots, all the factors can be obtained, and they are $z^{-2} + 0.5z^{-1} + 1$, $z^{-2} + 0.0699z^{-1} + 1$, $z^{-2} + 1.4301z^{-1} + 1$ and $z^{-2} + 1.5z^{-1} + 1$. Originally, there are no integer factors. By conversion, the non-integer factors $z^{-2} + 0.5z^{-1} + 1$ and $z^{-2} + 1.5z^{-1} + 1$ are converted to integer factors $2z^{-2} + z^{-1} + 2$ and $2z^{-2} + 3z^{-1} + 2$ when they are multiplied by an integer 2 for 2 is less than $473/2$. Thus the scaling constant β becomes 10. After conversion, there are two integer factors and two non-integer factors. According to the proposed factorization technique, B1 contains only integer factors. Therefore, there are in total 3 possible combinations of integer factors that can be allocated to B1 as listed in Table 3.6. The rest factors, together with β , of each combination in B1 are allocated to B2. The coefficients of the products of the factors in B2 are checked; the resultant coefficients of B2 in this example are $\{20, 60, 107, 123, 107, 60, 20\}$, $\{20, 40, 77, 81, 77, 40, 20\}$ and $\{10, 15, 21, 15, 10\}$ for the first, second and third factorization, respectively. Therefore, all of these factorizations are successful.

3.3.3 Incorporating the Factorization Technique into the Proposed GA

When the factorization technique is incorporated into the proposed GA, the GA search is conducted on single-stage filters. The 3 genetic operations including selection, crossover

and mutation are applied as usual, but the fitness is computed in a different way. During the GA search, once a chromosome is obtained, an integer coefficient set of a single-stage filter is obtained. This discrete coefficient set is evaluated to check if the filter specification is satisfied, and the adder cost, f_g , is computed. The factorization technique proposed in Section 3.3.1 is performed only when the filter specification is met. If the polynomial of the single-stage filter is successfully factorized into two integer polynomials, the adder cost to realize the two factorized integer coefficient sets, denoted as $f_{g_{cas}}$, is computed. If $f_{g_{cas}} \leq f_g$, the fitness value of the chromosome is set to $1/f_{g_{cas}}$; otherwise, to $1/f_g$. Note that 1) for an integer coefficient set (Chromosome), the factorization may not be unique, and thus $f_{g_{cas}}$ is set to the smallest adder cost and the corresponding factorization is recorded; 2) the factorization is not necessary leading to a smaller objective value (i.e., adder cost) than the single-stage realization, and the factorization is not recorded in this case.

A simple example to illustrate the fitness evaluation of the chromosomes is given as follows. If a chromosome is decoded to an integer coefficient set $\{5, 29, 52, 62, 52, 29, 5\}$, the adder cost of this integer set (including both MBAs and SAs) is $f_g = 10$. The proposed factorization technique is performed and the polynomial can be factorized into two subpolynomials with coefficient sets $\{1, 4, 1\}$ and $\{5, 9, 11, 9, 5\}$, respectively. The adder cost to realize these two integer coefficient sets is $f_{g_{cas}} = 9$. Since $f_{g_{cas}} \leq f_g$, the fitness value of this chromosome is $1/f_{g_{cas}}$, i.e., $1/9$. The factorization result is recorded.

3.4 Numerical Examples of Cascade Form Design

In this section, two sets of design examples are given to illustrate the superiority of the proposed algorithm.

In Example A, the long filters B, L1 and C are designed using the proposed algorithm

Table 3.7: Design results of Filters B, L1 and C. *MBA* and *SA* are number of multiplier block adders and number of structural adders

Filters	EWL	Cascade / Single Stage Design by Proposed GA / single stage design [55]			
		<i>MBA</i>	<i>SA</i>	Total	Runtime
B	8	8 / 10 / 11	86 / 98 / 100	94 / 108 / 111	22m19s / 16m24s / 24h
L1	14	32 / 45 / 47	120 / 120 / 120	152 / 165 / 167	34m21s / 26m41s / 56m
C	10	20 / 31 / 22	280 / 306 / 306	300 / 337 / 328	4h3m / 3h49m / 24h

in cascade form. Since there is no existing algorithm directly designing long filter in cascade form which aims to minimize the adder cost, the results are first compared with the best results of single-stage designs in [55]. The algorithms in [3, 69] is further repacked in a manner such that they can be used to design long filters in the cascade form, and the results obtained are also compared.

In Example B, two moderately long filters A1 and S2 (with length around 60) taken from [69] are designed using the proposed GA in cascade form. The results are compared with the cascade design in [69]. Note that if enough time is given, the algorithm in [69] is able to find the cascade design with minimum number of adders. However, even for the moderately long filters, it may takes several weeks to finish the search; therefore in [69] the runtime for each design is limited to 24 hours.

3.4.1 Example A

The design results, including the adder cost and design time, of B, L1 and C are listed in Table 3.7. The single-stage design obtained by [55] and by the proposed GA are also listed. It shows that the adder costs obtained by the proposed cascade technique are significant reduced compared with the single stage designs. The reason that the results are compared with the single-stage design in [55] is that no existing multiplierless technique can directly design long FIR filters in cascade form with lower hardware cost than the state-of-the-art single-stage design technique.

However, with existing cascade filter design techniques [69, 74–76] and discrete coefficient filter design technique for moderately long filters [3], alternative approaches indirectly designing the long filters are possible. Here, filter B is designed using such a possible way to compare with results obtained by the proposed algorithm.

The basic idea of the indirect design is to first separate the long filter into a cascade of several short filters with continuous coefficients, and each short filter is then designed by [3] or [69]. Two subfilters are considered in this example, denoted as F1 and F2, respectively. However, a difficulty is to set the specifications for each subfilter, such as filter orders, ripples and EWLs. In [75], two approaches are proposed.

The first method is named as direct approximation method, in which F1 is designed to continuous coefficient values directly using the frequency response specifications of the overall filter and by fixing F1, F2 is then optimized in discrete values. However, in the design of filter B, F1 designed in such a way cannot lead to an overall filter meeting the ripple requirement even if F2 is optimized with continuous coefficients.

The other way is to obtain subfilters using the interleaved zero approach. After obtaining the continuous coefficient values of F1 by using this method, F1 is fixed and F2 is optimized using algorithms [3] or [69] to obtain discrete coefficient values such that the overall filter meets the specification of filter B. Then, F2 is fixed and F1 is optimized using [3] or [69] to meet the overall specification. In the design of Filter B, the subfilter orders for both F1 and F2 are 52, according to the interleaved zero approach.

Using [3], the resultant hardware cost for F1 and F2 are 60 and 55 adders, with EWLs 8 and 5, respectively. Thus, the EWL of the overall filter is as large as 15 due to the convolution of the two subfilters. Note that filter B designed using the proposed algorithm requires only 94 adders with overall EWL 8. In terms of the hardware cost, the filter obtained using the proposed GA is much lower, because of the small number of adders and small EWL. Using [69], each subfilter is further a cascade of two short

subfilters. However, in this case, in the step fixing F1 and optimizing F2, even if the EWL of F2 is increased to 6, no solution can be found. It is possible to further increase the EWL of F2 until a solution is found, but this, on one hand, increases the design time significantly, and, on the other hand, will increase the hardware cost significantly due to the large overall EWL of the filter.

This design example shows that such indirect optimization approaches either generate designs with much higher hardware cost than the proposed GA or cannot produce feasible cascade designs for a given search space. In addition, the runtimes of these algorithms are also much longer because each subproblem (to optimize the subfilter) may take several hours for a filter with orders around 50.

The reason that such indirectly techniques cannot generate better results is due to the difficulty in how to set the orders of the subfilters, and how to allocate zeros to each subfilter. Using the interleaved zero approach, for a given filter orders, the way to allocate zeros to the two subfilters are fixed. Certainly, there are other ways to allocate zeros that may result in feasible and better solutions, but obviously finding a good allocation is not straightforward, and significant additional research may be required.

Overall speaking, the proposed technique is the first one efficiently achieving lower hardware cost for the design of long filters in cascade form. The coefficient values of the cascade designs of B, L1 and C are listed in Tables 3.8, 3.9 and 3.10, respectively.

3.4.2 Example B

In this example, the proposed GA is used for the design of two moderately long filters A1 and S2 in cascade form and the results are compared with that obtained by MILP based algorithm [69]. Refer to Table 2.1 for the specifications of A1 and S2. Table 3.11 shows that for the filters with moderately long length, which can be handled by deterministic

Table 3.8: The Coefficient Values of Filter B in Cascade Form

Filter B $h_1(n) = h_1(1 - n)$ for $n = 0$, $h_2(n) = h_2(103 - n)$ for $0 \leq n \leq 51$ $h(n) = h(104 - n)$ for $0 \leq n \leq 52$, passband gain: 907.66 EWL: 8 normalized passband ripple: 0.00995; normalized stopband ripple: 0.00995				
$h_1(0) = 1$	$h_2(13) = -1$	$h_2(27) = -9$	$h_2(41) = 8$	
$h_2(0) = 2$	$h_2(14) = 4$	$h_2(28) = 2$	$h_2(42) = 8$	
$h_2(1) = -3$	$h_2(15) = -2$	$h_2(29) = -4$	$h_2(43) = -10$	
$h_2(2) = 3$	$h_2(16) = 1$	$h_2(30) = 9$	$h_2(44) = -14$	
$h_2(3) = -2$	$h_2(17) = -5$	$h_2(31) = 1$	$h_2(45) = -26$	
$h_2(4) = 4$	$h_2(18) = 0$	$h_2(32) = 10$	$h_2(46) = -14$	
$h_2(5) = -2$	$h_2(19) = -4$	$h_2(33) = -3$	$h_2(47) = -4$	
$h_2(6) = 3$	$h_2(20) = 4$	$h_2(34) = 1$	$h_2(48) = 31$	
$h_2(7) = -4$	$h_2(21) = -1$	$h_2(35) = -12$	$h_2(49) = 53$	
$h_2(8) = 2$	$h_2(22) = 7$	$h_2(36) = -4$	$h_2(50) = 89$	
$h_2(9) = -4$	$h_2(23) = -1$	$h_2(37) = -10$	$h_2(51) = 96$	
$h_2(10) = 2$	$h_2(24) = 5$	$h_2(38) = 5$		
$h_2(11) = -2$	$h_2(25) = -6$	$h_2(39) = 4$		
$h_2(12) = 4$	$h_2(26) = 0$	$h_2(40) = 16$		

Table 3.9: The Coefficient Values of Filter L1 in Cascade Form

Filter L1 $h_1(n) = h_1(6 - n)$ for $0 \leq n \leq 3$, $h_2(n) = h_2(114 - n)$ for $0 \leq n \leq 57$ $h(n) = h(120 - n)$ for $0 \leq n \leq 60$, passband gain: 69265.174 EWL: 14 normalized passband ripple: 0.005572; normalized stopband ripple: 0.00009775				
$h_1(0) = 1$	$h_2(12) = 15$	$h_2(28) = 50$	$h_2(44) = 38$	
$h_1(1) = -3$	$h_2(13) = 16$	$h_2(29) = -6$	$h_2(45) = -61$	
$h_1(2) = 5$	$h_2(14) = 1$	$h_2(30) = 31$	$h_2(46) = 18$	
$h_1(3) = -6$	$h_2(15) = 20$	$h_2(31) = 36$	$h_2(47) = -90$	
$h_2(0) = 6$	$h_2(16) = 6$	$h_2(32) = -18$	$h_2(48) = 72$	
$h_2(1) = 3$	$h_2(17) = 14$	$h_2(33) = 65$	$h_2(49) = 109$	
$h_2(2) = 4$	$h_2(18) = 23$	$h_2(34) = -6$	$h_2(50) = -161$	
$h_2(3) = 3$	$h_2(19) = -3$	$h_2(35) = 17$	$h_2(51) = 259$	
$h_2(4) = 4$	$h_2(20) = 27$	$h_2(36) = 66$	$h_2(52) = -147$	
$h_2(5) = 13$	$h_2(21) = 10$	$h_2(37) = -36$	$h_2(53) = 35$	
$h_2(6) = 4$	$h_2(22) = 7$	$h_2(38) = 77$	$h_2(54) = 283$	
$h_2(7) = 9$	$h_2(23) = 35$	$h_2(39) = 5$	$h_2(55) = -486$	
$h_2(8) = 9$	$h_2(24) = -4$	$h_2(40) = -9$	$h_2(56) = 745$	
$h_2(9) = 5$	$h_2(25) = 33$	$h_2(41) = 106$	$h_2(57) = -764$	
$h_2(10) = 17$	$h_2(26) = 18$	$h_2(42) = -59$		
$h_2(11) = 5$	$h_2(27) = -5$	$h_2(43) = 81$		

Table 3.10: The Coefficient Values of Filter C in Cascade Form

Filter C: $h_1(n) = h_1(2 - n)$ for $0 \leq n \leq 2$, $h_2(n) = h_2(322 - n)$ for $0 \leq n \leq 161$ $h(n) = h(324 - n)$ for $0 \leq n \leq 162$, passband gain: 5395.25, EWL: 10 normalized passband ripple: 0.00489; normalized stopband ripple: 0.00489				
$h_1(0) = 1$	$h_2(39) = -1$	$h_2(80) = 7$	$h_2(121) = -6$	
$h_1(1) = 1$	$h_2(40) = 1$	$h_2(81) = 0$	$h_2(122) = -7$	
$h_2(0) = -4$	$h_2(41) = 0$	$h_2(82) = 7$	$h_2(123) = -7$	
$h_2(1) = 0$	$h_2(42) = -2$	$h_2(83) = 9$	$h_2(124) = 10$	
$h_2(2) = 2$	$h_2(43) = -2$	$h_2(84) = -3$	$h_2(125) = 9$	
$h_2(3) = -2$	$h_2(44) = -1$	$h_2(85) = 4$	$h_2(126) = 11$	
$h_2(4) = 3$	$h_2(45) = -4$	$h_2(86) = 2$	$h_2(127) = 22$	
$h_2(5) = 1$	$h_2(46) = 0$	$h_2(87) = -8$	$h_2(128) = 14$	
$h_2(6) = -1$	$h_2(47) = 0$	$h_2(88) = -4$	$h_2(129) = 9$	
$h_2(7) = 2$	$h_2(48) = -3$	$h_2(89) = -2$	$h_2(130) = 12$	
$h_2(8) = 2$	$h_2(49) = 4$	$h_2(90) = -12$	$h_2(131) = -4$	
$h_2(9) = -1$	$h_2(50) = 3$	$h_2(91) = -2$	$h_2(132) = -12$	
$h_2(10) = 0$	$h_2(51) = -1$	$h_2(92) = 0$	$h_2(133) = -11$	
$h_2(11) = 1$	$h_2(52) = 6$	$h_2(93) = -6$	$h_2(134) = -23$	
$h_2(12) = -2$	$h_2(53) = 2$	$h_2(94) = 6$	$h_2(135) = -25$	
$h_2(13) = -1$	$h_2(54) = -1$	$h_2(95) = 8$	$h_2(136) = -15$	
$h_2(14) = 0$	$h_2(55) = 3$	$h_2(96) = 0$	$h_2(137) = -15$	
$h_2(15) = -2$	$h_2(56) = 0$	$h_2(97) = 12$	$h_2(138) = -7$	
$h_2(16) = 0$	$h_2(57) = -5$	$h_2(98) = 10$	$h_2(139) = 11$	
$h_2(17) = 0$	$h_2(58) = -1$	$h_2(99) = -3$	$h_2(140) = 16$	
$h_2(18) = -1$	$h_2(59) = -1$	$h_2(100) = 7$	$h_2(141) = 24$	
$h_2(19) = 1$	$h_2(60) = -7$	$h_2(101) = 3$	$h_2(142) = 34$	
$h_2(20) = 1$	$h_2(61) = 0$	$h_2(102) = -14$	$h_2(143) = 31$	
$h_2(21) = 1$	$h_2(62) = 0$	$h_2(103) = -2$	$h_2(144) = 23$	
$h_2(22) = 1$	$h_2(63) = -3$	$h_2(104) = -4$	$h_2(145) = 16$	
$h_2(23) = 1$	$h_2(64) = 3$	$h_2(105) = -18$	$h_2(146) = 0$	
$h_2(24) = 1$	$h_2(65) = 5$	$h_2(106) = -3$	$h_2(147) = -20$	
$h_2(25) = -1$	$h_2(66) = 0$	$h_2(107) = 0$	$h_2(148) = -33$	
$h_2(26) = 0$	$h_2(67) = 5$	$h_2(108) = -9$	$h_2(149) = -48$	
$h_2(27) = 0$	$h_2(68) = 6$	$h_2(109) = 8$	$h_2(150) = -55$	
$h_2(28) = -3$	$h_2(69) = -2$	$h_2(110) = 10$	$h_2(151) = -50$	
$h_2(29) = -1$	$h_2(70) = 3$	$h_2(111) = 2$	$h_2(152) = -41$	
$h_2(30) = -1$	$h_2(71) = 0$	$h_2(112) = 16$	$h_2(153) = -17$	
$h_2(31) = -2$	$h_2(72) = -5$	$h_2(113) = 12$	$h_2(154) = 18$	
$h_2(32) = 1$	$h_2(73) = -3$	$h_2(114) = 1$	$h_2(155) = 54$	
$h_2(33) = -1$	$h_2(74) = -2$	$h_2(115) = 8$	$h_2(156) = 100$	
$h_2(34) = 0$	$h_2(75) = -8$	$h_2(116) = 1$	$h_2(157) = 145$	
$h_2(35) = 4$	$h_2(76) = -2$	$h_2(117) = -13$	$h_2(158) = 182$	
$h_2(36) = 0$	$h_2(77) = 1$	$h_2(118) = -5$	$h_2(159) = 216$	
$h_2(37) = 2$	$h_2(78) = -5$	$h_2(119) = -12$	$h_2(160) = 237$	
$h_2(38) = 3$	$h_2(79) = 4$	$h_2(120) = -19$	$h_2(161) = 241$	

Table 3.11: Design Results of Filter A1 and S2. L_i , MBA_i and SA_i are the Length, Number of Multiplier Block Adders and Number of Structural Adders of Subfilter i , Respectively.

Filters	EWL	The propose GA / Algorithm in [69]							
		L_1	L_2	MBA_1	MBA_2	SA_1	SA_2	Total	Runtime
A1	10	5 / 2	55 / 58	0 / 0	8 / 9	4 / 1	48 / 52	60 / 62	9m48s / 24h
S2	10	3 / 2	59 / 59	0 / 0	13 / 14	2 / 1	54 / 54	69 / 69	21m09s / 24h

Table 3.12: The Coefficient Values of Filter A1 in Cascade Form

Filter A1: $h_1(n) = h_1(4 - n)$ for $0 \leq n \leq 2$, $h_2(n) = h_2(54 - n)$ for $0 \leq n \leq 27$ $h(n) = h(58 - n)$ for $0 \leq n \leq 29$, passband gain: 4894.15, EWL: 10 normalized passband ripple: 0.009908; normalized stop ripple: 0.0009907				
$h_1(0) = 1$	$h_2(5) = -6$	$h_2(13) = 0$	$h_2(21) = -20$	
$h_1(1) = 2$	$h_2(6) = 0$	$h_2(14) = 18$	$h_2(22) = 27$	
$h_1(2) = 2$	$h_2(7) = -9$	$h_2(15) = -8$	$h_2(23) = 26$	
$h_2(0) = 2$	$h_2(8) = 3$	$h_2(16) = 6$	$h_2(24) = 80$	
$h_2(1) = 0$	$h_2(9) = -8$	$h_2(17) = -27$	$h_2(25) = 74$	
$h_2(2) = 2$	$h_2(10) = 9$	$h_2(18) = -9$	$h_2(26) = 120$	
$h_2(3) = -2$	$h_2(11) = -1$	$h_2(19) = -36$	$h_2(27) = 97$	
$h_2(4) = 2$	$h_2(12) = 18$	$h_2(20) = -6$		

algorithm such as [69], the propose heuristic GA use much shorter design time than that of the algorithm in [69], while the adder cost of the final designs is not worse than that in [69]. It has to be noted that, for filter S2, when the EWL and order are kept to be 10 and 59 as in [69], the proposed GA can not find the feasible solution. Therefore, as described in Section 3.1.7, the filter order is increased by 1 and the optimization is re-done. The design times of the proposed GA for filter A1 and S2 are 9 minutes 48 seconds and 21 minutes 18 seconds using multiple computers and 1 hours 36 minutes and 3 hours 11 minutes using one computer. The design time are significantly less than that of the algorithm in [69], where both designs use more than 24 hours. The coefficient values of the cascade design of A1 and S2 are listed in Tables 3.12, and 3.13, respectively.

Table 3.13: The Coefficient Values of Filter S2 in Cascade Form

Filter S2: $h_1(n) = h_1(2 - n)$ for $0 \leq n \leq 1$, $h_2(n) = h_2(58 - n)$ for $0 \leq n \leq 29$ $h(n) = h(60 - n)$ for $0 \leq n \leq 30$, passband gain: 7263.98, EWL: 10 normalized passband ripple:0.011922;normalized passband ripple:0.0009935				
$h_1(0) = 1$	$h_2(6) = -2$	$h_2(14) = -24$	$h_2(22) = 96$	
$h_1(1) = 1$	$h_2(7) = -6$	$h_2(15) = -16$	$h_2(23) = 116$	
$h_2(0) = 2$	$h_2(8) = -11$	$h_2(16) = -10$	$h_2(24) = 143$	
$h_2(1) = 1$	$h_2(9) = -11$	$h_2(17) = -3$	$h_2(25) = 163$	
$h_2(2) = 0$	$h_2(10) = -16$	$h_2(18) = 14$	$h_2(26) = 177$	
$h_2(3) = 2$	$h_2(11) = -22$	$h_2(19) = 31$	$h_2(27) = 195$	
$h_2(4) = 0$	$h_2(12) = -20$	$h_2(20) = 47$	$h_2(28) = 202$	
$h_2(5) = -3$	$h_2(13) = -22$	$h_2(21) = 73$	$h_2(29) = 202$	

Table 3.14: Design Results of Filter B with Different Filter Orders. L_i , MBA_i and SA_i are the Length, Number of Multiplier Block Adders and Number of Structural Adders of Subfilter i , Respectively.

Order	EWL	L_1	L_2	MBA_1	MBA_2	SA_1	SA_2	Total
105	8	2	105	0	10	1	90	101
106	8	3	105	0	10	2	88	100
107	8	2	107	0	8	1	92	101
108	8	No cascade design is found						
109	8	2	109	0	10	1	92	103

3.4.3 Robustness for the Design of Cascaded FIR Filters

Since the proposed GA for the design of cascaded FIR filter is based on the factorization of a given integer coefficient polynomial and the factorization is not guaranteed to be successful, it is possible that no cascade design is produced by the proposed GA. To test the robustness of the proposed GA for the design of cascaded FIR filters, filter B is designed with filter orders from 105 to 109, and the design results are listed in Table 3.14. Table 3.14 shows that for each filter order except 108, cascade designs can be found.

An additional advantage of the proposed GA for the cascade design is its compatibility with the single-stage design, i.e., even if no cascade design is found, in the end of the optimization the proposed GA takes the best single-stage design as the final design, as discussed in Section 3.3.3.

3.5 Conclusion

In this chapter, a novel GA is proposed for the design of low hardware cost multiplierless FIR filters both in single-stage and cascade forms. In order to make the GA capable of the design of high order and wide coefficient wordlength filters, the discrete search space is partitioned into multiple spaces according to the passband gains. Moreover, the search efficiency of the GA is improved by adaptively adjusting the crossover and mutation rates. In addition, instead of minimizing the filter ripple as in conventional GAs, the proposed GA uses the adder cost of the filter as the objective function, and penalties are applied when the ripple requirement is not met, such that the hardware cost and filter ripples are balanced in the fitness values. The cascade design is incorporated into the GA search for the single-stage designs. Every single-stage design obtained during the GA operation, if meeting the filter specification, is applied with an integer coefficient polynomial factorization. Successful factorizations generate the cascade designs if the hardware cost are lower. Design examples show that the proposed GA uses much less computational time and the hardware cost of the resulting designs, for both single-stage and cascade designs, is reduced in most cases.

Chapter 4

A Polynomial Time Algorithm for the Design of Multiplierless Linear Phase FIR Filters with Low Hardware Cost

As reviewed in section 2.2.3, most MCM and filter design joint optimization algorithms [3,51,55] applied the depth-first width-recursive tree search method to traverse potential solutions and find out the best one. To obtain the optimum results, exhaustive search as proposed in [3] is necessary. However, such algorithms are only suitable for the design of filters up to orders around of 60. Many tree search algorithms are developed for the trade off between the efficiency and optimality of the algorithm. To increase the efficiency of the tree search, every coefficient only can be fixed to a limited number of discrete values.

The critical issue therefore in such tree search algorithms is to determine how many and which discrete coefficient values of every coefficient should be fixed in its feasible range. The technique in [55] first time proposed that in depth-first width-recursive tree

search, fixing the coefficients values to the surrounding of the middle value of the coefficient feasible range may result in low hardware cost. The technique suggested several possible solutions and identified implicitly that assigning each coefficient with two discrete values for each node without floating the passband gain is a good trade-off between the algorithm complexity and the designed filters complexity.

Inspired by the technique in [55], this chapter proposes a polynomial time tree search algorithm by constraining each coefficient to only one discrete value, the middle value in its feasible range. Instead of assigning more discrete values to each coefficient, the research shows that floating the passband gain increases the chances to achieve lower hardware cost and the algorithm remains in polynomial computational complexity. Besides a proposed passband gain floating scheme, a low hardware cost scheme which fixes some coefficients to 0 value prior to the tree search is applied. Designs of benchmark filters show that the design time of the proposed algorithm is significantly shorter than that of other tree search based algorithms, and the hardware costs of the designs are close to that of the near optimum designs [3]. The near optimum design technique is only suitable for the design of filters with orders up to about 60 and 70.

The rest of the chapter is organized as follows. Section 4.1 introduces in detail the low hardware cost scheme, i.e., which and how many coefficients are fixed to 0 prior to the tree search, and the passband gain floating scheme. Six design examples are given in Section 4.2 to show the advantages of the proposed algorithm in the design of multiplierless FIR filters. In Section 4.3, the computational complexity of the proposed algorithm is analyzed. Finally, the conclusion is given in Section 4.4.

4.1 The Proposed Optimization Technique

4.1.1 Motivation

All deterministic tree search based optimization techniques [3, 51, 55] target to find a single final discrete solution that costs the minimum hardware. During the tree search, a current best discrete solution in terms of hardware cost is maintained as a reference to cut off the search branches that have no chances to produce the final discrete solution better than the current best one.

Instead of estimating the hardware cost during the tree search, the proposed algorithm targets to find as many feasible discrete solutions as possible. The more feasible discrete solutions we have, the higher chances to obtain a low hardware cost design when these feasible solutions are synthesized using a MCM algorithm.

As practiced in Chapter 3, floating the passband gain may result in different feasible discrete solutions. The basic idea of the proposed technique is also to partition the range of the floating passband gain into smaller ones, but in each smaller ranger, instead of using heuristic GAs as in Chapter 3, a tree determinstic search is performed to find a feasible solution. Under such strategy, it is obvious that the more partitioned passband gain ranges, the more feasible solutions that we may have. However, on the other hand, the more partitioned passband gain ranges, the more tree searches to be performed. This requires a polynomial time tree search technique for each of the partitioned passband gain range.

In the existing depth-first width-recursive tree search methods [3, 51, 55], assuming that the numbers of children generated by each node are the same, denoted as L , the numbers of total nodes generated in the tree is $(L^{D+1} - 1)/(L - 1)$ for $L > 1$, where D is the depth of the tree, which is equal to the number of coefficients to be optimized. The

number of LPs to be solved, denoted as N_{LP} , is proportional to the number of nodes, i.e., $N_{LP} \propto (L^{D+1} - 1)/(L - 1) \doteq L^{D+1}/(L - 1)$. L is further proportional to 2^B in the techniques in [3, 51], where B is the EWL of the coefficients, and L is set to 2 in the technique [55] for the best trade-off between the algorithm complexity and designed filter complexity. To have an algorithm with polynomial complexity, L must be constrained to 1 and the number of nodes of the tree search is thus reduced to D .

Inspired by the algorithm in [55], in the proposed polynomial-time algorithm, each coefficient is fixed to only one discrete value, the one closest to the middle value in its feasible range. To accelerate the search and maximally reduce the complexity, a low hardware cost scheme fixing some coefficients to 0 is proposed before the passband gain floating scheme is introduced.

4.1.2 A Low Hardware Cost Mechanism

It is known to all that the coefficient value “0” is the most favorable value in multiplierless FIR filter design, because the zero value coefficient can immediately reduce two adders in the delay chain. Moreover, the adders in the delay chain generally have larger number of full adders than that of the adders used to synthesize the coefficients. Therefore, in the proposed algorithm, some coefficients are first fixed to zeros; after that, the non-zero coefficients are fixed successively to the discrete value that is closest to the middle value of its feasible range as proposed [55]. So the first key problem is how to determine the number of zero coefficients and their corresponding positions. The procedure to determine the zero coefficients is as follows:

1. Set the zero index set, denoted as Z , to empty.
2. Find $h(n)$ and δ by solving the following linear programming

Minimize : δ

$$\text{Subject to : } 1 - \delta \leq H(\omega) \leq 1 - \delta, \quad \text{for } \omega \in [0, \omega_p] \quad (4.1)$$

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

where δ_p , δ_s , ω_p and ω_s are the given passband ripple, stopband ripple, passband edge and stopband edge, respectively. $H(\omega)$ in the formulation is the zero-phase frequency responses given by

$$H(\omega) = \sum_{n=0, n \notin Z}^{\lfloor \frac{N}{2} \rfloor} h(n) \text{Trig}(\omega, n). \quad (4.2)$$

3. Find the minimum absolute value of the obtained $h(n)$ for $n = 0, 1, \dots, \lfloor \frac{N}{2} \rfloor$ and $n \notin Z$. Record its corresponding index, denoted as k . If the resultant δ satisfies $\delta \leq \rho \delta_p + (1 - \rho) \delta_{opt}$, where δ_{opt} is the minimum passband ripple that can be achieved by the continuous value filter with order N and ρ is a weighting factor, set $Z = Z \cup k$, fix $h(k)$ to 0 and go to step 2; otherwise, stop.

In step 3 of the above procedure, only when the resultant ripple δ is not larger than $\rho \delta_p + (1 - \rho) \delta_{opt}$, is the coefficient with minimum absolute value fixed to 0. This is because the frequency response of the filter in the tree search deteriorates for two factors, fixation of coefficients to zero values, or quantization of the other non-zero coefficients. Therefore, the deterioration caused due to the fixed zero coefficients must be restricted to a certain level such that there are still margins for the quantization of other non-zero coefficients. In an extreme case that the EWL is very large, the coefficient values are nearly continuous and ρ can be simply set to 1. However, generally in the multiplierless FIR filter design, the EWL is set to the minimum value because every increment of the coefficient wordlength may increase the bit-level hardware cost significantly. The minimum EWL refers to the wordlength below which no discrete solution can be found.

Figure 4.1: The maximum values of ρ for 20 sets of filters when their feasibilities are kept.

Under such wordlength constraints, a larger ρ may lead to nonexistence of feasible discrete solution. The value of ρ is affected by many factors such as filter order, EWL, ripple requirement, etc., and is difficult to be determined theoretically.

To understand the pattern of the value of ρ that maintains the feasibility of the discrete solutions, a set of 20 filters with random specifications is tested. Under the minimum EWL constraints, the maximum value of ρ when the feasibility is kept is shown in Fig. 4.1.

It can be seen that for different filters, the value of ρ may differ significantly. The smallest value is 0.157 while the largest is 0.897. To always keep the feasibility, a conservative value of ρ is chosen to be 0.15 at first. If under such conservative value, there is still no feasible solution, ρ is decreased by a stepsize 0.05 until feasible solution can be obtained.

Once the zero coefficients are fixed, the unfixed coefficients $h(n)$ for $n \notin Z$ are quantized according to the following tree search algorithm:

1. For a given passband gain range $\Gamma_m = [l_m, u_m]$, set $i = 0$.

2. If $i \in Z$, go to step 4; else, find the feasible range of the i -th coefficient, denoted as F_i^m , by solving the following optimization problem.

$$\begin{aligned}
&\text{Minimize : } f_0 = h(i) \text{ and } f_1 = -h(i) \\
&\text{Subject to : } b_m(1 - \delta_p)2^Q \leq H(\omega) \leq b_m(1 + \delta_p)2^Q, \text{ for } \omega \in [0, \omega_p] \\
&\quad -\delta_s b_m 2^Q \leq H(\omega) \leq \delta_s b_m 2^Q, \quad \text{for } \omega \in [\omega_s, \pi] \\
&\quad l_m \leq b_m \leq u_m
\end{aligned} \tag{4.3}$$

where

$$H(\omega) = \sum_{n=0, n \notin Z}^{i-1} h'(n) \text{Trig}(\omega, n) + \sum_{n=i, n \notin Z}^{\lfloor \frac{N-1}{2} \rfloor} h(n) \text{Trig}(\omega, n). \tag{4.4}$$

In (4.4), $h'(n)$ are the quantized coefficients in the previous iterations, $h(n)$ are variable coefficients to be optimized. The feasible range F_i^m is thus given by $[f_0 - f_1]$.

3. Check whether F_i^m is empty. If yes, no feasible solution is found for this passband gain section under the current search scheme, and go to step 5; otherwise, set $h'(i)$ to the discrete value closest to the middle value in F_i^m .
4. $i = i + 1$. If $i = D$, a feasible discrete solution $\mathbf{h}^{fix}$ is obtained. Go to step 5. Otherwise, go to step 2.
5. Stop.

4.1.3 Determining the Number of Floating Passband Gain Range

The tree search algorithm proposed in section 4.1 is performed for a given floating passband gain range Γ_m . The last issue, therefore, is to determine how to partition the floating passband gain range. For the same reason given in Section 3.1.2, the passband

gain range $[0.7 \ 1.4]$ is considered. Obviously, in this range, if finer ranges are partitioned, more feasible discrete solutions may be obtained, but the complexity of the algorithm is higher as well. Instead of using a constant M as in Section 3.1.2, according to design experiences, a good trade off between the number of feasible solutions and the optimization complexity is to partition the range evenly into M sections and M is related to the wordlength, Q , of the filter, given by

$$M = \begin{cases} 40 & Q \leq 12, \\ 40 + 5(Q - 12) & \text{otherwise.} \end{cases} \quad (4.5)$$

Thus, the m -th range Γ_m is $[0.7 + 0.7(m - 1)/M, 0.7 + 0.7m/M]$, for $m = 1, 2, \dots, M$.

For each of the range Γ_m , the tree search is performed. Therefore, at most M feasible discrete solutions may be obtained, and the MCM algorithm proposed in [77] (the same algorithm used in [55]) is applied to synthesize each set of discrete coefficients to MCM adder-and-shift networks. The one synthesized with the fewest adders is the final solution.

4.2 Design Examples

Six benchmark filters labeled as B, C, L1, A1, S2 and L2 are designed to show the superiority of the proposed algorithm. The filter specifications are listed in Table 2.1. The filters with moderate orders S2, A and L2 are taken from [55] and [3]. The long filters B, C and L1 are taken from [55]. The designs produced by the proposed algorithm are compared with the results obtained from SHI's algorithm [3] and the best results suggested by the FIRGAM algorithm [55]. For the long filters B, L1 and C, the proposed algorithm is also compared to the algorithm proposed in Chapter 3. The design time and the hardware cost in terms of number of adders and EWL are summarized in Table 4.1. For SHI's algorithm [3] and the FIRGAM algorithm [55], the time needed to find the best

solution may be much shorter than the whole runtime. For a complete comparison, the time to find the best solution time are also listed in Table 4.1. It should be noted that SHI's algorithm and the FIRGAM algorithm are developed by C language, and the GA in Chapter 3 and the proposed technique in this Chapter is developed by MATLAB. In addition, the runtimes of filters B, C and L1 in SHI's algorithm as well as that of B and C in the FIRGAM are limited to 24 hours. Table 4.1 shows that both the runtime and best solution time of the proposed technique are significantly shorter than that of two techniques in [55] and [3]. If the programs are not terminated manually, for the filters B and C, the programs in techniques [3,55] may run for weeks or even months to complete the search. In addition, the run time of the proposed technique is also shorter than the GAs proposed in Chapter 3 even if multiple computers are used for parallel computation for the GA. The computation complexity of the proposed algorithm is analyzed in the next section in details.

With such significantly shorter design time, the proposed algorithm produces the designs with fewer adders than that of FIRGAM algorithm in most cases. The number of adders listed for the algorithm in [55] and for the proposed technique are computed using the MCM synthesis algorithm in [77] for a fair comparison. The technique C1 [38] may achieve better synthesis with fewer adders. The number of adders synthesized using C1 for both techniques are also listed in brackets in Table 4.1 for reference, if data are available. C1 is also the MCM algorithm used for the designs obtained by GA algorithm in Chapter 3. Compared with the algorithm proposed in Chapter 3, for the design of long filters L1, B and C, the proposed one generate designs for filter B with one more adder but save 3 and 18 adders for the filters L1 and C, respectively. Compared with SHI's algorithm, when the filter orders are not more than 70, near optimum results are obtained by SHI's algorithm, and the proposed algorithm may requires a few more adders; however, when it comes to higher order filters such as B, C and L1, SHI's algorithm in

24 hours either fails to find a feasible solution or produce a result with higher hardware cost than the proposed algorithm.

For a more complete comparison, the proposed algorithm is also compared with the FIRGAM when L is set to 1. Since the tree width is 1, the FIRGAM now becomes a polynomial-time design algorithms and multiple runs of the tree search over different floating passband gain range become possible. The number of floating passband gain ranges for the FIRGAM is set to the same as the proposed one. The adder cost for filters B, C, L1, A, L2 and S2 are 111, 327, 171, 76, 82 and 83, respectively, higher than that obtained by the proposed one. The lower hardware cost achieved by the proposed algorithm than that of the FIRGRAM with $L = 1$ and floating passband gain is due to the proposed low hardware cost strategy proposed in section 4.1.2.

For results listed in Table 4.1, the proposed algorithm is executed on a PC with 3.2 GHz core 2 Duo CPU.

4.3 Computation Complexity

The search algorithm presented in Section 4.1 shows that each tree search for a floating passband gain range at most traverse D nodes and each node solves 2 LPs, i.e., each tree search requires not more than $2D$ LPs. For M passband gain ranges, the total number of LPs solved in the proposed algorithm is not more than $2MD$, where M is given in (4.5). This number is much smaller than the computation time of the algorithm proposed in Chapter 3 given in (3.11). To analyze further, it is known that the complexity of each LP is a polynomial of the number of unknown variables. When more and more coefficients are fixed to discrete values in the tree search, the number of unknown variables are less and less. The overall complexity of the proposed algorithm T , is given by $T \leq 2M \sum_{v=1}^D T_{lp}(v)$, where $T_{lp}(v)$ is a polynomial of v and v is the number of unknown variables.

Table 4.1: Results and Comparison of Six Benchmark Filters

Filters	Method	Q	EWL	Number of adders	Runtime	Best Solution time
B	SHI [3]			no feasible result		
	FIRGAM [55]	11	8	111 (††)	24h	9m
	GA*	11	8	(108 [†])	16m24s	16m24s
	Proposed	11	8	109 (109 [†])	6m23s	3m47s
C	SHI [3]			no feasible result		
	FIRGAM [55]	13	10	328 (††)	24h	13h7m
	GA*	13	10	(337 [†])	3h49m	3h49m
	Proposed	12	9	319 (319 [†])	2h11m	1h58m
L1	SHI [3]	17	14	167	24h	17h
	FIRGRAM [55]	17	14	167 (162 [†])	56m	32m
	GA*	17	14	(165 [†])	26m41s	26m41s
	Proposed	17	14	169 (162 [†])	15m26s	8m31s
A	SHI [3]	14	10	69	50h	6h31m
	FIRGAM [55]	14	10	76 (††)	4h14m	3h2m
	Proposed	14	10	72 (71 [†])	3m5s	2m42s
L2	SHI [3]	13	10	73	16h28m	4h21m
	FIRGAM [55]	13	10	80 (††)	54m	26m
	Proposed	13	10	78 (76 [†])	4m9s	2m17s
S2	SHI [3]	14	10	77	16h	3h40m
	FIRGAM [55]	15	11	86 (††)	27m	23m
	Proposed	15	11	83 (78 [†])	5m21s	2m47s

[†] synthesized using C1 algorithm [38] for reference

^{††} the coefficient set not available

* Genetic algorithm proposed in Chapter 3

For different algorithms solving LPs, the function $T_{lp}(v)$ may be different. For example, if the Karmarkar's algorithm [78] is used to solve the linear programming, the worst-case solving time can be expressed as $\beta v^{3.5}$, where β is a constant. Thus, T can be expressed as

$$T \leq 2M\beta \sum_{v=1}^D v^{3.5} = \begin{cases} 80\beta \sum_{v=1}^D v^{3.5} & \text{for } Q \leq 12 \\ 10(Q-4)\beta \sum_{v=1}^D v^{3.5} & \text{for } Q > 12 \end{cases} \quad (4.6)$$

Since $\sum_{v=1}^D v^3 < \sum_{v=1}^D v^{3.5} < \sum_{v=1}^D v^4$, and $\sum_{v=1}^D v^3$ and $\sum_{v=1}^D v^4$ can be computed as $\frac{1}{4}D^4 + \frac{1}{2}D^3 + \frac{1}{4}D^2$ and $\frac{1}{5}D^5 + \frac{1}{2}D^4 + \frac{1}{3}D^3 - \frac{1}{30}D$, respectively, the algorithm complexity is a polynomial of D with order not higher than 5. Moreover, it can be seen from (4.6) that the algorithm complexity is independent of wordlength when $Q \leq 12$ or it is a linear function of wordlength when $Q > 12$; this greatly reduces the computation time for the design of filters with long wordlength. In addition, the complexity for the synthesis of the feasible discrete coefficient sets to adder-and-shift network using MCM algorithms is negligible compared with that of LPs.

Overall speaking, the proposed algorithm is a polynomial time algorithm which can be used to design long filters with low hardware cost using short design time. When it is used to design moderate order filters, the hardware cost of the results are slightly higher than that of the existing best algorithms [3]. Compared with [55], besides the low hardware cost mechanism proposed in section 4.1.2, this chapter explicitly shows that width 1 tree search with proper passband gain floating technique outperforms width 2 tree search; the later one is considered as the best trade-off between the algorithm complexity and designed filter complexity in [55].

4.4 Conclusion

In this chapter, instead of finding a single discrete solution with the lowest hardware cost in a tree search, as many as possible discrete feasible solutions are searched. Therefore, a polynomial-time tree search algorithm is proposed, where each coefficient is only fixed to a single discrete value in a search. By floating the passband gain a large number of feasible discrete solutions may be produced in a short time. A MCM algorithm is then applied to each feasible discrete solution. The one with the lowest hardware cost is the final design. Besides the passband gain floating scheme, a low hardware cost mechanism to fix some coefficient values to 0 prior to the tree search is also proposed. Design examples show that the proposed algorithm can generate low hardware cost design in significantly shorter time, specially for long filters.

4.5 Appendix A : Impulse Response of Design Examples

Table 4.2: The Coefficient Values of Filter A1, S2 and L2

Filter A, $h(n) = h(58 - n)$ for $0 \leq n \leq 29$ passband gain: 5363.1836, EWL: 10				
$h(0) = 3$	$h(8) = -24$	$h(16) = 50$	$h(24) = 152$	
$h(1) = 5$	$h(9) = -25$	$h(17) = 7$	$h(25) = 350$	
$h(2) = 7$	$h(10) = -17$	$h(18) = -52$	$h(26) = 556$	
$h(3) = 8$	$h(11) = 0$	$h(19) = -113$	$h(27) = 736$	
$h(4) = 6$	$h(12) = 25$	$h(20) = -157$	$h(28) = 858$	
$h(5) = 0$	$h(13) = 50$	$h(21) = -163$	$h(29) = 902$	
$h(6) = -8$	$h(14) = 68$	$h(22) = -117$		
$h(7) = -17$	$h(15) = 70$	$h(23) = -10$		
Filter S2, $h(n) = h(59 - n)$ for $0 \leq n \leq 29$ passband gain: 14426.144, EWL: 11				
$h(0) = 7$	$h(8) = -42$	$h(16) = -89$	$h(24) = 773$	
$h(1) = 8$	$h(9) = -62$	$h(17) = -39$	$h(25) = 905$	
$h(2) = 8$	$h(10) = -85$	$h(18) = 31$	$h(26) = 1020$	
$h(3) = 8$	$h(11) = -105$	$h(19) = 122$	$h(27) = 1113$	
$h(4) = 3$	$h(12) = -122$	$h(20) = 231$	$h(28) = 1178$	
$h(5) = 0$	$h(13) = -133$	$h(21) = 356$	$h(29) = 1212$	
$h(6) = -12$	$h(14) = -133$	$h(22) = 492$		
$h(7) = -25$	$h(15) = -119$	$h(23) = 634$		
Filter L2, $h(n) = h(62 - n)$ for $0 \leq n \leq 31$ passband gain: 4294.6068, EWL: 10				
$h(0) = 3$	$h(8) = -23$	$h(16) = -64$	$h(24) = -168$	
$h(1) = 6$	$h(9) = -7$	$h(17) = -48$	$h(25) = -201$	
$h(2) = 8$	$h(10) = 15$	$h(18) = 0$	$h(26) = -123$	
$h(3) = 6$	$h(11) = 33$	$h(19) = 60$	$h(27) = 78$	
$h(4) = 0$	$h(12) = 36$	$h(20) = 102$	$h(28) = 368$	
$h(5) = -11$	$h(13) = 19$	$h(21) = 97$	$h(29) = 673$	
$h(6) = -22$	$h(14) = -14$	$h(22) = 33$	$h(30) = 904$	
$h(7) = -28$	$h(15) = -48$	$h(23) = -70$	$h(31) = 990$	

Table 4.3: The Coefficient Values of Filter C

$h(n) = h(324 - n)$ for $0 \leq n \leq 162$, passband gain:5002.9896, EWL: 10				
$h(0) = -3$	$h(41) = 0$	$h(82) = 13$	$h(123) = -19$	
$h(1) = -3$	$h(42) = 0$	$h(83) = 14$	$h(124) = -4$	
$h(2) = 0$	$h(43) = -4$	$h(84) = 12$	$h(125) = 12$	
$h(3) = 0$	$h(44) = -5$	$h(85) = 9$	$h(126) = 27$	
$h(4) = 1$	$h(45) = -5$	$h(86) = 3$	$h(127) = 39$	
$h(5) = 2$	$h(46) = -5$	$h(87) = -3$	$h(128) = 44$	
$h(6) = 3$	$h(47) = -4$	$h(88) = -9$	$h(129) = 42$	
$h(7) = 3$	$h(48) = -2$	$h(89) = -13$	$h(130) = 32$	
$h(8) = 2$	$h(49) = 0$	$h(90) = -16$	$h(131) = 16$	
$h(9) = 2$	$h(50) = 3$	$h(91) = -16$	$h(132) = -4$	
$h(10) = 1$	$h(51) = 5$	$h(92) = -13$	$h(133) = -25$	
$h(11) = 0$	$h(52) = 7$	$h(93) = -8$	$h(134) = -43$	
$h(12) = 0$	$h(53) = 7$	$h(94) = 0$	$h(135) = -54$	
$h(13) = 0$	$h(54) = 6$	$h(95) = 7$	$h(136) = -58$	
$h(14) = -3$	$h(55) = 4$	$h(96) = 14$	$h(137) = -52$	
$h(15) = -3$	$h(56) = 0$	$h(97) = 18$	$h(138) = -34$	
$h(16) = -3$	$h(57) = -2$	$h(98) = 20$	$h(139) = -10$	
$h(17) = 0$	$h(58) = -5$	$h(99) = 17$	$h(140) = 19$	
$h(18) = 0$	$h(59) = -7$	$h(100) = 12$	$h(141) = 46$	
$h(19) = 0$	$h(60) = -8$	$h(101) = 6$	$h(142) = 69$	
$h(20) = 2$	$h(61) = -8$	$h(102) = -3$	$h(143) = 82$	
$h(21) = 3$	$h(62) = -6$	$h(103) = -12$	$h(144) = 81$	
$h(22) = 3$	$h(63) = -3$	$h(104) = -19$	$h(145) = 66$	
$h(23) = 3$	$h(64) = 0$	$h(105) = -23$	$h(146) = 36$	
$h(24) = 3$	$h(65) = 5$	$h(106) = -23$	$h(147) = -4$	
$h(25) = 2$	$h(66) = 7$	$h(107) = -19$	$h(148) = -50$	
$h(26) = 0$	$h(67) = 9$	$h(108) = -11$	$h(149) = -93$	
$h(27) = -1$	$h(68) = 10$	$h(109) = -2$	$h(150) = -127$	
$h(28) = -3$	$h(69) = 8$	$h(110) = 9$	$h(151) = -143$	
$h(29) = -3$	$h(70) = 6$	$h(111) = 19$	$h(152) = -134$	
$h(30) = -4$	$h(71) = 2$	$h(112) = 25$	$h(153) = -100$	
$h(31) = -4$	$h(72) = -3$	$h(113) = 28$	$h(154) = -37$	
$h(32) = -3$	$h(73) = -7$	$h(114) = 26$	$h(155) = 51$	
$h(33) = 0$	$h(74) = -10$	$h(115) = 20$	$h(156) = 160$	
$h(34) = 0$	$h(75) = -12$	$h(116) = 9$	$h(157) = 277$	
$h(35) = 3$	$h(76) = -11$	$h(117) = -4$	$h(158) = 396$	
$h(36) = 4$	$h(77) = -9$	$h(118) = -17$	$h(159) = 503$	
$h(37) = 5$	$h(78) = -5$	$h(119) = -27$	$h(160) = 588$	
$h(38) = 5$	$h(79) = 0$	$h(120) = -34$	$h(161) = 644$	
$h(39) = 4$	$h(80) = 5$	$h(121) = -34$	$h(162) = 664$	
$h(40) = 2$	$h(81) = 10$	$h(122) = -29$		

Table 4.4: The Coefficient Values of Filter L1 and B

Filter L1, $h(n) = h(120 - n)$ for $0 \leq n \leq 60$				
passband gain: 46379.3029 , EWL: 14				
$h(0) = 3$	$h(16) = -51$	$h(32) = 227$	$h(48) = 928$	
$h(1) = -7$	$h(17) = 88$	$h(33) = -43$	$h(49) = -1231$	
$h(2) = 12$	$h(18) = -89$	$h(34) = -192$	$h(50) = 948$	
$h(3) = -15$	$h(19) = 46$	$h(35) = 366$	$h(51) = -83$	
$h(4) = 13$	$h(20) = 28$	$h(36) = -380$	$h(52) = -1069$	
$h(5) = -4$	$h(21) = -101$	$h(37) = 201$	$h(53) = 1989$	
$h(6) = -11$	$h(22) = 136$	$h(38) = 110$	$h(54) = -2115$	
$h(7) = 26$	$h(23) = -107$	$h(39) = -414$	$h(55) = 1066$	
$h(8) = -35$	$h(24) = 18$	$h(40) = 552$	$h(56) = 1173$	
$h(9) = 30$	$h(25) = 97$	$h(41) = -425$	$h(57) = -4195$	
$h(10) = -10$	$h(26) = -183$	$h(42) = 52$	$h(58) = 7264$	
$h(11) = -20$	$h(27) = 190$	$h(43) = 417$	$h(59) = -9547$	
$h(12) = 47$	$h(28) = -101$	$h(44) = -758$	$h(60) = 10390$	
$h(13) = -57$	$h(29) = -56$	$h(45) = 770$		
$h(14) = 41$	$h(30) = 212$	$h(46) = -385$		
$h(15) = 0$	$h(31) = -286$	$h(47) = -276$		
Filter B, $h(n) = h(104 - n)$ for $0 \leq n \leq 52$				
passband gain: 1146.7315, EWL: 8				
$h(0) = -3$	$h(14) = 4$	$h(28) = -9$	$h(42) = 20$	
$h(1) = 0$	$h(15) = 2$	$h(29) = -2$	$h(43) = -2$	
$h(2) = 0$	$h(16) = -1$	$h(30) = 6$	$h(44) = -30$	
$h(3) = 2$	$h(17) = -5$	$h(31) = 13$	$h(45) = -51$	
$h(4) = 3$	$h(18) = -6$	$h(32) = 14$	$h(46) = -51$	
$h(5) = 2$	$h(19) = -5$	$h(33) = 9$	$h(47) = -22$	
$h(6) = 1$	$h(20) = -1$	$h(34) = -2$	$h(48) = 33$	
$h(7) = 0$	$h(21) = 4$	$h(35) = -14$	$h(49) = 106$	
$h(8) = -3$	$h(22) = 7$	$h(36) = -20$	$h(50) = 180$	
$h(9) = -4$	$h(23) = 8$	$h(37) = -18$	$h(51) = 232$	
$h(10) = -2$	$h(24) = 4$	$h(38) = -6$	$h(52) = 251$	
$h(11) = 0$	$h(25) = -2$	$h(39) = 11$		
$h(12) = 2$	$h(26) = -8$	$h(40) = 26$		
$h(13) = 4$	$h(27) = -11$	$h(41) = 30$		

Chapter 5

Multiplierless Multiple-stage Cascaded Linear Phase FIR Filter Design

As reviewed in Section 2.1, an attractive way to further reduce the hardware cost of FIR filters is to design the filters in multiple-stage cascade forms. In two-stage cascaded FIR filters, the overall filter $H(\omega)$ is decomposed into two subfilters $H_1(\omega)$ and $H_2(\omega)$, i.e, $H(\omega) = H_1(\omega)H_2(\omega)$. Many algorithms [69, 74–76] have been proposed to design the two-stage cascaded FIR filters with discrete value coefficients. Among them, the latest algorithm proposed in [69] can design the two subfilters simultaneously and achieve significant hardware reduction compared to that of single stage and other two-stage cascaded designs [75, 76]. However, the computational complexity of the algorithm in [69] is high. Even for the filter with orders around 60, the design time may last more than one week if the program is not manually terminated. On the other hand, if the filter can be decomposed into more than two stages, the hardware complexity may be further reduced, but currently no algorithm can do this efficiently. In this chapter, an algorithm

is proposed for the design of multiple-stage cascaded FIR filters. The number of stages is automatically determined by the algorithm and examples show that the number of filter stages can be decomposed to as high as 6. The designed filters achieve lower hardware cost than that of the conventional algorithms.

5.1 The Proposed Technique

For expression convenience, the zero-phase frequency response of a linear phase FIR filter with order of N given in (3.2) is re-expressed here

$$H(\omega) = \sum_{n=0}^{\lfloor \frac{N}{2} \rfloor} h(n) \text{Trig}(\omega, n) \quad (5.1)$$

where $h(n)$ are the filter coefficients and $\text{Trig}(\omega, n)$ is an appropriate trigonometric function depending on the parity of N and symmetry of the filter.

If the overfall filter $H(\omega)$ is in cascade form with M stages, i.e., $H(\omega) = H_1(\omega)H_2(\omega)\dots H_M(\omega)$, the goal in the multiplierless multiple-stage cascaded FIR filter design is to find the coefficient sets of all subfilters $H_1(\omega), H_2(\omega), \dots, H_M(\omega)$ in discrete values so that the hardware cost measured in the number of full adders (FAs) is minimized. In the proposed algorithm, the orders of the subfilters except for $H_M(\omega)$ are restricted to be up to 2 and the values of the coefficients of these subfilters are constrained to integer values 1, 0 or -1 . Thus the coefficient sets of $H_1(\omega), H_2(\omega), \dots, H_{M-1}(\omega)$ can only be one of $\{[1 \ 1], [1 \ 0 \ 1], [1 \ -1 \ 1], [1 \ 1 \ 1]\}$ when symmetric filter is considered (Anti-symmetric case can be developed similarly). Here, the set $\{[1 \ 1], [1 \ 0 \ 1], [1 \ -1 \ 1], [1 \ 1 \ 1]\}$ are denoted as h_{base} , and for $k = 1, 2, 3, 4$, $h_{base}\{k\}$ is $[1 \ 1]$, $[1 \ 0 \ 1]$, $[1 \ -1 \ 1]$ and $[1 \ 1 \ 1]$, respectively. When the subfilters $H_1(\omega), H_2(\omega), \dots, H_{M-1}(\omega)$ are all fixed, the last subfilter $H_M(\omega)$ is optimized using the single-stage design method proposed in [3]. The reason to restrict the coefficient values of subfilters $H_1(\omega), H_2(\omega), \dots, H_{M-1}(\omega)$ to values of 1, 0 and -1 is that the realization

of these 3 values are cost free. Therefore, the overall hardware cost of the whole filter is mainly determined by the last subfilter $H_M(\omega)$. Now the problem is transferred to how to select the coefficient sets of subfilters $H_1(\omega), H_2(\omega), \dots, H_{M-1}(\omega)$ from the base coefficient set h_{base} , and leading to a low hardware cost realization of $H_M(\omega)$.

The basic procedure is to determine $H_i(\omega)$ for $i = 1, \dots, M-1$ one by one starting from $H_1(\omega)$. When determining each $H_i(\omega)$, the overall frequency response can be written as $\left(\prod_{j=1}^{i-1} H_j(\omega)\right) H_i(\omega) H_{rest}^i(\omega)$, of which $H_j(\omega)$ for $j = 1, \dots, i-1$ have been fixed to a coefficient set in h_{base} , each $H_i(\omega)$ is replaced by $h_{base}\{k\}$ for $k = 1, 2, 3, 4$ one by one, and $H_{rest}^i(\omega)$ is optimized with continuous coefficient values for each replacement of $H_i(\omega)$ by $h_{base}\{k\}$. Denoting the resulted optimized filters as $H_{rest}^{i,k}(\omega)$ for $k = 1, 2, 3, 4$, $h_{base}\{m\}$ is selected to be the impulse response of $H_i(\omega)$ if the average of the magnitude of the impulse response of $H_{rest}^{i,m}(\omega)$, denoted as $S_{mean}^{i,m}$, is minimum among all $H_{rest}^{i,k}(\omega)$.

The detailed procedure is shown in Figure 5.1 and described as follows, for given δ_p , δ_s , ω_p , ω_s as the passband, stopband ripple tolerance, passband edge, stopband edge, and N as the order of the overall filter.

Step 1: Set the stage index i to 1.

Step 2: For $k = 1, 2, 3, 4$, the coefficient set of the i -th stage subfilter $h_i(n)$ is replaced by $h_{base}\{k\}$, and solve the following linear programming problem for each k :

$$\begin{aligned} &\text{Minimize : } \delta_{i,k} \\ &\text{Subject to : } 1 - \delta_{i,k} \leq \left(\prod_{j=1}^{i-1} H_j(\omega)\right) H_i^k(\omega) H_{rest}^{i,k}(\omega) \leq 1 - \delta_{i,k}, \quad \text{for } \omega \in [0, \omega_p] \quad (5.2) \\ &\quad -\frac{\delta_s \delta_{i,k}}{\delta_p} \leq \left(\prod_{j=1}^{i-1} H_j(\omega)\right) H_i^k(\omega) H_{rest}^{i,k}(\omega) \leq \frac{\delta_s \delta_{i,k}}{\delta_p}, \quad \text{for } \omega \in [\omega_s, \pi] \end{aligned}$$

where $\delta_{i,k}$ is the peak ripple to be minimized for the impulse response of $H_i(\omega)$ being replaced by $h_{base}(k)$. $H_j(\omega)$ for $j = 1, 2, \dots, i$ are zero phase frequency response of subfilters from stage 1 to $i-1$ and are expressed as $H_j(\omega) = \sum_{n=0}^{\lfloor \frac{N_j}{2} \rfloor} h_i(n) \text{Trig}(\omega, n)$, where N_j are the orders of respective filters, and $H_{rest}^{i,k}(\omega)$ is the overall zero phase frequency of the

remaining subfilter and is expressed as $H_{rest}^{i,k}(\omega) = \sum_{n=0}^{\left\lfloor \frac{N_{rest}^{i,k}}{2} \right\rfloor} h_{rest}^{i,k}(n) \text{Trig}(\omega, n)$, where $N_{rest}^{i,k}$ is given by $N - \sum_{j=1}^i N_j$. In the formulation (5.2), the variables to be optimized are the coefficient set $h_{rest}^{i,k}(n)$ and ripple $\delta_{i,k}$. Frequency ω is discretized to form the constraints. After solving (5.2), the absolute mean value of $h_{rest}^{i,k}(n)$, denoted as $S_{mean}^{i,k}$ is computed by

$$S_{mean}^{i,k} = \begin{cases} \left\lfloor \frac{N_{rest}^{i,k}}{2} \right\rfloor \sum_{n=0}^{\left\lfloor \frac{N_{rest}^{i,k}}{2} \right\rfloor} |h_{rest}^{i,k}(n)| / N_{rest}^{i,k}, & \text{if } \delta_{i,k} \leq 0.08\delta_p + 0.92\delta_{opt} \\ 100, & \text{otherwise} \end{cases} \quad (5.3)$$

where δ_{opt} is the minimum passband ripple that can be achieved by the overall filter of order N with continuous coefficients. δ_{opt} can be easily obtained by solving a linear programming given by

$$\begin{aligned} &\text{Minimize : } \delta_{opt} \\ &\text{Subject to : } 1 - \delta_{opt} \leq H(\omega) \leq 1 - \delta_{opt}, \quad \text{for } \omega \in [0, \omega_p] \\ &\quad -(\delta_s \delta_{opt}) / \delta_p \leq H(\omega) \leq (\delta_s \delta_{opt}) / \delta_p, \quad \text{for } \omega \in [\omega_s, \pi] \end{aligned}$$

From the above obtained four designs in the i -th stage, denote $S_{mean}^{i,m}$ as $S_{mean}^{i,min}$ if $S_{mean}^{i,m}$ is minimum among all $S_{mean}^{i,k}$ for $k = 1, 2, 3, 4$.

Step 3: If $S_{mean}^{i,min} \geq S_{mean}^{i-1,min}$ and $i \geq 2$, set $M = i$ and go to step 4; otherwise, the coefficient set of i -th stage subfilter $h_i(n)$ is selected to be $h_{base}\{m\}$. $i = i + 1$ and go to step 2.

Step 4: After the subfilters $H_1(\omega), H_2(\omega), \dots, H_{M-1}(\omega)$ have been determined, the only subfilter left, $H_M(\omega)$, is optimized using the mixed integer linear programming method proposed in [3].

Figure 5.1: The flow chart to determine the coefficient set of subfilter

5.2 The Remarks of the Proposed Algorithm

First, the reason that the coefficient set of the i -th stage subfilter $h_i(n)$ is selected to be $h_{base}\{m\}$ whose corresponding $S_{mean}^{i,m}$ is minimum among $S_{mean}^{i,k}$ for $k = 1, 2, 3, 4$ is because the smaller $S_{mean}^{i,k}$, the smaller the coefficient magnitudes. It is obvious that the smaller coefficient magnitudes, the shorter the wordlength to represent the coefficients. In general, shorter wordlength leads to lower hardware costs. That is also the reason that the decomposition process is stopped when $S_{mean}^{i,min}$ can not be further reduced.

Second, $S_{mean}^{i,k}$ is set to a very large value (100 in practical) if the corresponding $\delta_{i,k}$ is larger than $0.08\delta_p + 0.92\delta_{opt}$ in (5.3); this is to restrict the deterioration of frequency response incurred due to the decomposition process. Once the deterioration is larger than the empirical value $0.08\delta_p + 0.92\delta_{opt}$, the penalty value 100 is assigned to $S_{mean}^{i,k}$.

5.3 Design Examples

In this section, 5 benchmark filters labeled as Y2, L2, A1, S2 and Y3 are given to illustrate the superiority of the proposed algorithm in the sense of hardware cost and design time. All these filters are taken from literature [3] and their specifications are listed in Table 2.1. The hardware cost is measured in the numbers of FAs. The numbers of FAs for each filter designed using the proposed algorithm and using the best existing algorithm [69] is listed in Table 5.1, where B_{in} is the wordlength of the input signal. If B_{in} is set to 16, the hardware reduction of the proposed algorithm for filters Y2, L2, A1, S2 and Y3 are 10.7%, 11.7%, 16.4%, 17.7% and 8.6%, respectively. If only the number of adders are compared, the scale factors for B_{in} in Table 5.1 indicate the number of adders in each design. For the moderate long filters A1 and S2, the proposed algorithm is also compared with the GA proposed in Chapter 3. The numbers of FAs obtained by using the GA for

Table 5.1: Results and Comparison of Benchmark Filters.

Proposed algorithm / Algorithm in [69] / GA in Chapter 3						
Filters	M	NFA *	Runtime			
Y2	4 / 2	331+42 B_{in} / 370+47 B_{in}	2s	/	2m41s	
L2	4 / 2	585+57 B_{in} / 623+67 B_{in}	59m	/	24h	
A1	5 / 2 / 2	597+52 B_{in} / 637+63 B_{in} / 621+60 B_{in}	14h36m	/	24h	/ 9m48s
S2	6 / 2 / 2	539+59 B_{in} / 697+69 B_{in} / 701+69 B_{in}	17h21m	/	24h	/ 21m09s
Y3	3 / 2	482+55 B_{in} / 530+60 B_{in}	11m	/	49m	

* Number of full adders

filter A1 and S2 are worse than that of the algorithm proposed in this chapter. Table 5.1 and 3.7 also shows that the design time of the proposed algorithm is much shorter than that of the algorithm proposed in [69] but longer than that of the GA proposed in Chapter 3.

It should be noted that although the proposed algorithm generated the designs with lowest hardware cost in terms of FA among the comparison algorithm, the design time of the proposed algorithm is exponentially increased with increasing of filter order. In other words, the proposed algorithm may not find optimal solution for long filters such as the filters L1, B and C designed in Chapter 3. According to the design experience, the algorithm proposed in this Chapter is suited to design the filter with order less than 70. If the filter order is larger than 70, the algorithms suited to design long filters, such as those proposed in [69] or in Chapter 3 are better choices.

5.4 Conclusion

In this chapter, a novel algorithm is proposed for the design of linear phase FIR filters in multiple-stage cascade form with discrete coefficients. The number of stages is automatically determined by the algorithm once an overall filter order is given. Design examples have shown that the proposed algorithm is able to achieve significant reduction in hardware cost when compared with the existing best algorithms, and the design time

is also saved.

5.5 Appendix B : Impulse Response of Design Examples

Table 5.2: The Coefficient Values of Filter Y2 and L2 in Cascade Form
 Filter Y2: $h(n) = h(29 - n)$ for $0 \leq n \leq 15$
 passband gain: 2880.523525, NPR: 0.000982

$h_1(0) = 1$	$h_4(1) = 5$	$h_4(7) = 0$	$h_4(13) = -43$
$h_2(0) = 1$	$h_4(2) = -1$	$h_4(8) = -18$	$h_4(14) = -15$
$h_2(1) = 1$	$h_4(3) = -6$	$h_4(9) = -5$	$h_4(15) = 71$
$h_3(0) = 1$	$h_4(4) = 0$	$h_4(10) = 28$	$h_4(16) = 124$
$h_3(1) = 1$	$h_4(5) = 7$	$h_4(11) = 20$	
$h_4(0) = -2$	$h_4(6) = 3$	$h_4(12) = -26$	

Filter L2: $h(n) = h(62 - n)$ for $0 \leq n \leq 31$ passband gain: 4369.182751, NPR: 0.027958			
$h_1(0) = 1$	$h_4(4) = -2$	$h_4(13) = -5$	$h_4(22) = -19$
$h_2(0) = 1$	$h_4(5) = -2$	$h_4(14) = -9$	$h_4(23) = -22$
$h_2(1) = 1$	$h_4(6) = 0$	$h_4(15) = 2$	$h_4(24) = 0$
$h_3(0) = 1$	$h_4(7) = -2$	$h_4(16) = 0$	$h_4(25) = 6$
$h_3(1) = 1$	$h_4(8) = 3$	$h_4(17) = 2$	$h_4(26) = 26$
$h_4(0) = 2$	$h_4(9) = 4$	$h_4(18) = 16$	$h_4(27) = 60$
$h_4(1) = -2$	$h_4(10) = 2$	$h_4(19) = 5$	$h_4(28) = 60$
$h_4(2) = 2$	$h_4(11) = 0$	$h_4(20) = -5$	
$h_4(3) = -2$	$h_4(12) = 0$	$h_4(21) = 0$	

Table 5.3: The Coefficient Values of Filter S2, A1 and Y3 in Cascade Form

Filter S2: $h(n) = h(59 - n)$ for $0 \leq n \leq 30$ passband gain: 18556.327435, NPR: 0.011873				
$h_1(0) = 1$	$h_6(1) = -10$	$h_6(10) = -5$	$h_6(19) = 28$	
$h_2(0) = 1$	$h_6(2) = 0$	$h_6(11) = -15$	$h_6(20) = 48$	
$h_3(0) = 1$	$h_6(3) = 0$	$h_6(12) = 8$	$h_6(21) = 48$	
$h_3(1) = 1$	$h_6(4) = 5$	$h_6(13) = -18$	$h_6(22) = 55$	
$h_4(0) = 1$	$h_6(5) = -12$	$h_6(14) = 0$	$h_6(23) = 64$	
$h_4(1) = -1$	$h_6(6) = 4$	$h_6(15) = 4$	$h_6(24) = 76$	
$h_5(0) = 1$	$h_6(7) = -15$	$h_6(16) = 18$	$h_6(25) = 62$	
$h_5(1) = 0$	$h_6(8) = 0$	$h_6(17) = 10$		
$h_6(0) = 10$	$h_6(9) = -14$	$h_6(18) = 31$		
Filter A1: $h(n) = h(58 - n)$ for $0 \leq n \leq 29$ passband gain: 6946.803694, NPR: 0.009818				
$h_1(0) = 1$	$h_5(2) = 0$	$h_5(11) = 3$	$h_5(20) = 0$	
$h_2(0) = 1$	$h_5(3) = 3$	$h_5(12) = 1$	$h_5(21) = 1$	
$h_2(1) = 1$	$h_5(4) = -2$	$h_5(13) = 5$	$h_5(22) = 22$	
$h_3(0) = 1$	$h_5(5) = -5$	$h_5(14) = 1$	$h_5(23) = 27$	
$h_3(1) = 1$	$h_5(6) = 2$	$h_5(15) = -8$	$h_5(24) = 27$	
$h_4(0) = 1$	$h_5(7) = 0$	$h_5(16) = 0$	$h_5(25) = 40$	
$h_4(1) = 0$	$h_5(8) = -2$	$h_5(17) = -11$		
$h_5(0) = 3$	$h_5(9) = 0$	$h_5(18) = -12$		
$h_5(1) = -3$	$h_5(10) = 6$	$h_5(19) = -1$		
Filter Y3: $h(n) = h(49 - n)$ for $0 \leq n \leq 25$ passband gain: 4182.439491, NPR: 0.009937				
$h_1(0) = 1$	$h_3(5) = 4$	$h_3(12) = -12$	$h_3(19) = 23$	
$h_2(0) = 1$	$h_3(6) = 0$	$h_3(13) = 23$	$h_3(20) = -130$	
$h_3(0) = 6$	$h_3(7) = -20$	$h_3(14) = 41$	$h_3(21) = -122$	
$h_3(1) = -3$	$h_3(8) = 8$	$h_3(15) = -31$	$h_3(22) = 184$	
$h_3(2) = -1$	$h_3(9) = 16$	$h_3(16) = -56$	$h_3(23) = 512$	
$h_3(3) = -8$	$h_3(10) = 12$	$h_3(17) = 6$		
$h_3(4) = 12$	$h_3(11) = -31$	$h_3(18) = 96$		

Chapter 6

Design of Multiplierless FIR filters with Low Average Logic Depth

In Chapters 3, 4 and 5, algorithms for the design of low computational complexity FIR filters are proposed. The number of adders is the criterion to be minimized in those algorithms. However, as reviewed in Section 2.3, when low power is concerned, the average logic depth of the adder-and-shift network is also an important factor besides the number of adders. While many MCM algorithms applied on a given set of discrete coefficients (i.e., the algorithms in the second category) have considered the low average logic depth [38, 64, 79], the algorithms in the third category [37, 50–52, 61] so far aimed only at the minimum adder cost. Although the technique in [3] can impose a maximum logic depth constraint to the design, this technique has three obvious limitations. First of all, it is not known what is the best tradeoff between the maximum logic depth and the number of adders. Second, if a design is imposed a maximum logic depth which is smaller than that of a optimum design without such constraints, the probability of obtaining the assured minimum adder cost decreases. Last, even if a design with minimum number of adders is achieved under the maximum logic depth constraint, it does not imply directly

the minimum average logic depth of the overall design, while the minimum average logic depth is generally believed a more relevant criterion for low power design when the number of adders are comparable.

Therefore, this chapter proposes an algorithm aiming at designs with low average logic depth. This is achieved by constraining the synthesis of each coefficient on its minimum adder depth (MinAD) during the tree search of discrete coefficients. With such constraints, designs obtained by the proposed algorithm tend to use a few more adders to synthesize the filters, but always result in a reduction of the average logic depth and power dissipation compared with the best algorithms currently available.

The remaining of this chapter is organized as follows. Section 6.1 reviews the B&B depth-first tree search algorithm to optimize FIR filters in discrete space. In Section 6.2, a novel synthesis mechanism which aims at low average logic depth is introduced. Experimental results are presented in Section 6.3, and Section 6.4 concludes the chapter.

6.1 Review of tree search algorithm for optimization of FIR filters with MCM implementation

In this section, the optimization of filters is first formulated as a linear programming problem to minimize the filter ripples. Secondly, B&B depth-first tree search algorithm is reviewed for the search of discrete coefficients. The dynamic expanding subexpression space for the synthesis of discrete coefficients in MCM form is also introduced.

6.1.1 Problem Formulation for the FIR Filter design

The zero-phase frequency responses of a linear phase FIR filter of order N has been expressed in (3.2), where $h(n)$ are the filter coefficients. To find the coefficient set $h(n)$,

a linear programming is formulated to [3]

$$\begin{aligned}
& \text{Minimize: } f = \delta - \delta_p b \\
& \text{Subject to : } b - \delta \leq H(\omega) \leq b + \delta, \quad \text{for } \omega \in [0, \omega_p] \\
& \quad \quad \quad -(\delta_s \delta)/\delta_p \leq H(\omega) \leq -(\delta_s \delta)/\delta_p, \text{ for } \omega \in [\omega_s, \pi] \\
& \quad \quad \quad b_l \leq b \leq b_u
\end{aligned} \tag{6.1}$$

where δ is the peak ripple and b is a floating passband gain. δ_p , δ_s , ω_p and ω_s are the given passband ripple, stopband ripple, passband edge and stopband edge, respectively. b_l and b_u are two constants, defining the lower bound and upper bound of the passband of gain. They are chosen to be 0.7 and 1.4, respectively. Note that the objective function of the linear programming is not the normalized peak ripple (NPR) defined as δ/b [18]; instead, $\delta - \delta_p b$ is minimized. With such formulation, the algorithm may not find the optimum solution ($h(n)$ together with δ and b) in the sense of minimum NPR, but it always manages to find a feasible solution making $\delta - \delta_p b < 0$, if such solution exists [3].

6.1.2 Tree Search Procedure

For a given effect wordlength (EWL), the discrete value for each coefficient is in the range $[-2^{EWL} \quad 2^{EWL} - 1]$. Potential solutions are combinations of these discrete values. In the proposed tree search algorithm, the root of the tree is a feasible continuous solution of the FIR filter. The root produces its child nodes by quantizing a selected coefficient to a discrete value. All child nodes will further produce their own child nodes by quantizing another coefficient to a discrete value. The process continues until it reaches a leaf which contains only coefficients with discrete values. The nodes having the same parent and generated by quantizing the same coefficient to different discrete values are called siblings. If a node cannot further produce child nodes to forward the search for whatever reason, for example for the reason that all coefficients of the node have been quantized,

the search will switch to its sibling; and if no more sibling exists, the search will track back to its parent node to continue the forward search. In such a manner, the program traverses all the nodes which have a possibility of yielding a feasible solution, and is able to find a solution with less number of adders. When a node is generated, the quantized coefficients are synthesized to MCM form to minimize the number of adders in existing algorithm [3]. This Chapter proposes a new synthesis such that the adders are ensured to be synthesized to its MinAD. The discrete coefficient synthesis is discussed in the next section. The search is a recursive procedure in the following steps:

1. The root is generated by solving the linear programming formulated in (6.1). The feasible range of each coefficient is obtained as proposed in [3]. The root is the current node and the search depth d_s is set to 0;
2. A child of current node is generated by quantizing the coefficient $h(d_s)$ to a new discrete value in its range found in step 1.
 - If the child is successfully generated, the generated child node becomes the current node, $d_s = d_s + 1$, and go to step 3.
 - If such child node does not exist, i.e., $h(d_s)$ has been fixed to all its possible discrete values, there are two cases: a) if $d_s > 0$, the search backtracks to the parent of the current node and $d_s = d_s - 1$ and the parent of the current node becomes the current node and repeat step 2; b) if $d_s = 0$, the traverse of the tree is completed and the program is stopped.
3. The quantized discrete value is synthesized to an adder-and-shift network based on the technique described in Section 6.2, where a subexpression basis set is employed. The subexpression basis set of the current node is updated if new bases are

generated.

4. The first pruning checking: compute the minimal adder cost of the realization of the current node and check whether this node or this node's descendent can grow to be the optimum solution (i.e. if the adder cost of the node is smaller than that of the current best solution). If yes, go to step 5; otherwise, the current node is cut off and the search backtracks to its parent node which becomes the new current node, $d_s = d_s - 1$, and go to step 2. The details of computing the adder cost of the synthesis of the coefficient is discussed in Section 6.2.2.
5. Reoptimize the unquantized coefficients by solving (6.1) to obtain the objective value f .
6. The second pruning checking: it is to check the feasibility of the node. If $f \leq 0$, the current node meets the ripple specification and go to step 7. Otherwise, the current node is cut off; the search backtracks to its parent node which becomes the new current node and $d_s = d_s - 1$, go to step 2.
7. If $d_s = \lfloor \frac{N}{2} \rfloor$, all coefficients of the current node are discrete values, i.e., a leaf is reached. Go to step 8; otherwise, search forwards by going to step 2.
8. If the leaf is better than the current best solution in the sense of adder cost, the best solution is updated.
9. The search backtracks to its parent and the parent node becomes the current node. Set $d_s = d_s - 1$ and go to step 2.

6.1.3 Dynamically Expending Subexpression space

In the above B&B depth-first algorithm, the MCM coefficients are synthesized in step 3 when the selected coefficient is quantized. The synthesis is based on the subexpression space constructed from a subexpression basis set. The root has an initial zero order set including elements 0 and 1; the order of the set is defined by the number of adders required to synthesize all elements in the set. All other nodes inherit the basis sets from their parents, and the sets are expanded by including the current quantized coefficient (after repeatedly divided by 2 until an odd number) and other intermediate bases if additional adders are required to synthesize the current quantized coefficient. In [3], if the synthesis of a quantized coefficient value requires additional intermediate bases, the intermediate bases are not inserted to the child's basis set; instead, the basis set is flagged with "uncertain". With the introduction of the uncertainty, the optimum is guaranteed if the best leaf node is associated with a certain basis set.

6.2 A New Synthesis Mechanism

As discussed at the beginning of this Chapter, the synthesis technique proposed in [3,37, 50–52,61] and in Chapter 3, 4 only aim to minimize the number of adders; however, the average adder depth of the MCM blocks is not controlled, resulting in solutions that may consume higher powers. In this section, a synthesis technique that constraints the average adder depth is proposed. First, the concept of the MinAD of the adder of a discrete value is introduced. Based on this concept, a synthesis mechanism which ensures that each adder is synthesized on its MinAD is proposed. The MCM block synthesized in such a way is guaranteed with low average logic depth.

6.2.1 Minimum Adder Depth of A Coefficient

According to [64], if the number of the non-zero digits of a coefficient expressed in CSD is N_{zero} , irrespective of how this coefficient is decomposed into shift and add operations, the minimum achievable adder depth of the adder network to realize this coefficient is confined as:

$$S = \lceil \log_2 N_{zero} \rceil \quad (6.2)$$

This number S is defined as the minimum adder depth (MinAD) of the coefficient. Figure 6.1 shows two different adder networks synthesizing coefficient values 3, 11, and 43. In Figure 6.1 (a) each coefficient value requires an additional adder resulting in a maximum logic depth of 3. According to (6.2), the MinAD of 3, 11 and 43 are 1, 2 and 2, respectively. This indicates that an adder network with a maximum logic depth of 2 can synthesize the three coefficients, though more adders may be required. Figure 6.1 (b) shows an implementation with maximum logic depth 2, but one more adder is used than that in Figure 6.1 (a). In this chapter, if a coefficient is synthesized with the logic depth of MinAD, this coefficient is said to be synthesized on its MinAD. In the next subsection, a synthesis technique that ensures the realization of each coefficients on its MinAD is proposed.

In general, the terms of logic depth and adder depth are the same in an MCM block and referring to the number of adders in the longest path of a synthesis. In the following part of this thesis, to distinct the difference between the synthesis of a particular coefficient and the synthesis of the entire MCM block, “adder depth” is used to refer the former, and “logic depth” is used to refer the later.

Figure 6.1: The discrete coefficient values 3, 11 and 43 are synthesized (a) without MinAD constraint and (b) with MinAD constraints

6.2.2 A New Synthesis Mechanism Ensuring Coefficient Synthesized on its MinAD

In the proposed technique, besides the expanding subexpression basis set associated with each node, each node is also associated with an expanding auxiliary basis set. The auxiliary set for the root is initialized to be null, and each node inherits the set from its parent; the update of the set will be described incorporating with the coefficient synthesis. When a node is generated in step 3 in the tree search described in section 6.1.1, a selected coefficient $h(k)$ is fixed to a discrete value in its range; the discrete value is denoted as $h^0(k)$. Now this coefficient is to be synthesized based on its associated subexpression basis set. The detailed procedure (as shown in Figure 6.2) is given as follows.

1. The MinAD to realize $h^0(k)$ is computed according to (6.2).

Figure 6.2: Procedure to synthesize a coefficient $h(k)$ on its MinAD

2. If $h^0(k)$ can be synthesized based on the current basis set and the auxiliary set using not more than an adder, and the synthesis is on its MinAD, the synthesis is deemed optimum in the sense of minimum number of adders and the lowest power consumption.

3. If $h^0(k)$ can not be successfully synthesized on its MinAD using not more than one adder, a first order basis is inserted into the auxiliary basis set to help the synthesis of this coefficient on its MinAD. The basis set is also updated to include $h^0(k)$. Thus, the basis set itself is not certain, but the union of the basis set and auxiliary basis set is still certain. This differentiates the proposed technique from that in [3], where, in such cases, the discrete value is just inserted into the basis set and the set is flagged to be uncertain. With the basis set dynamically expanding, the uncertainty may be subsequently eliminated when more coefficients are synthesized. However, the price paid for that may be the increase of the adder depth for the coefficient. If the maximum logic depth constraint is adopted, it produces uncertain results in most cases. In order to produce certain solution without high power and area penalty, the above auxiliary basis set is introduced. Though this sacrifices the optimality in the sense of minimum number of adders in some cases, experiment results show that the increased number of adders is not significant and power saving is achieved.

4. If even by inserting a first order basis in the auxiliary basis set, the discrete value still can not be synthesized on its MinAD, this value is then just added to the basis set and the basis set is flagged as uncertain without updating the auxiliary set. The reason that not allowing more than one basis to be inserted into the auxiliary set is a tradeoff between the certainty of the results and the guarantee of the minimum number of adders. From the design experiences, adding not more than an auxiliary

Table 6.1: An Example of the Proposed Coefficient Synthesis Procedure and the Update of Basis Sets

Coefficients	main basis Set	auxiliary basis set	basis set certain
$h(0)=3$	$\{0,1,3\}$	null	Y
$h(1)=11$	$\{0,1,3,11\}$	null	Y
$h(2)=43$	$\{0,1,3,11,43\}$	$\{5\}$	Y

basis for each node is enough to make the final results certain, while maintain of each coefficient on its MinAD.

A simple example is given to illustrate the synthesis mechanism in details. As shown in Table 6.1, first, $h(0)$ is fixed to 3 and it can be synthesized on it MinAD based on the initial zeroth order basis set. The basis set is updated to include 3, and the auxiliary basis set is Null. Thereafter $h(1)$ is fixed to 11 whose CSD representation is $10\bar{1}0\bar{1}$. According to (6.2), the MinAD of 11 is 2. Clearly, $11=3 \times 2^2 - 1$ can be realized on its MinAD based on the current basis set. Therefore, when $h(2)$ is to be fixed to 43, the basis set is $\{0,1,3,11\}$ with certainty and the auxiliary basis set is still null. $h(2)$ can be synthesized based on the current basis set with one adder as $(43 = 11 \times 2^2 - 1)$, but the adder depth of such synthesis of 43 is 3 which is larger than its MinAD of 2. Thus, a first order basis 5 is inserted into the auxiliary basis set, such that 43 can be synthesized as $5 \times 2^3 + 3$ on its MinAD. Therefore the basis set and the auxiliary set are updated to be $\{0, 1, 3, 11, 43\}$ and $\{5\}$, respectively. Without the auxiliary basis set, the basis set is still certain, but not all members can be synthesized on its MinAD. With the price of one more adder to synthesize the auxiliary basis, the MinAD for each coefficient is achieved.

6.3 Experiment Results

Eight benchmark filters from literature are synthesized to show the superiority of the proposed algorithm in low power implementation. S1, S2, Y1, Y2, G1, A1 are the examples taken from [3], while F7 and F11 are the 7th and 11th examples of [60], respectively. All the filter specifications are listed in Table 2.1. The filters are synthesized using UMC 0.18 μm digital libraries, and the Synopsys Tool, Prime Power, is adopted to simulate the power consumption. The clock rate is 50M Hz, the power supply is 3.3V and the unit of power is milliwatt (mw).

The proposed method is compared with the method proposed in [3] without maximal adder depth constraint. If the final solution obtained in [3] is uncertain, i.e., the synthesis based on the subexpression basis set for the coefficient sets is not successful, some other bases needed to be inserted. For these cases, the MCM algorithm C1 [38] is applied to the uncertain coefficient set to synthesize the coefficients with the minimum adder cost. The uncertain results are marked with a “*” for the power value. Results show, as listed in Table 6.2, that for filters with short EWL, such as G1 and S1 (both with EWL of 7), the proposed algorithm generates exactly the same solutions as that produced by the algorithm proposed in [3]. For the filters with relatively longer EWL, the proposed algorithm produces results outperforming those produced by the algorithm in [3], in terms of maximum logic depth (MaxLD), average logic depth (ALD) and power consumption, although the proposed algorithm generally uses one or two more adders to synthesize the filters.

Table 6.2: Results and Comparison of eight benchmark filters, Where EWL is effective wordlength, W1 is wordlength, NA is number of adder, MaxLD is Maximum Logic Depth, and ALD is Average Logic Depth.

Name	EWL	Output WL	A:the method proposed in [3]				B: [3] with logic depth constraint				C:the proposed method			
			NA	MaxLD	ALD	Power	NA	MaxLD	ALD	Power	NA	MaxLD	ALD	Power (mw)
S1	7	17	23	3	1.48	3.09	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	25.50
G1	7	17	15	3	1.93	1.75	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	21.72
S2	10	22	76	4	2.56	27.30	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	5.98
A1	10	22	68	4	2.41	23.88	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	the same as that of algorithm A	6.22*
Y1	9	21	30	5	2.76	6.60	31	4	2.58	6.04	31	4	2.4	17.80
Y2	11	22	37	5	2.73	7.01	the same as that of algorithm C	the same as that of algorithm C	the same as that of algorithm C	the same as that of algorithm C	38	4	2.31	11.20*
F7	11	22	53	5	2.49	20.24	54	4	2.57	20.22	54	4	2.35	
F11	10	21	35	5	3.45	12.91	36	4	2.71	11.97	37	4	2.43	

* The final result is uncertain

For a fairer comparison, the proposed algorithm is further compared with that in [3] with maximum logic depth constraints. The maximum logic depth of each filter obtained in our technique is used as the constraints for the algorithm in [3]. Even with such information which is usually not prior known to algorithm in [3], only two filters generated by [3] is as good as that of the proposed algorithm, while the others are all worse than that of the proposed algorithm.

6.4 conclusion

In this chapter, a FIR filter optimization technique is proposed for the low power implementation of the MCM coefficients. The proposed method ensures that each adder in the MCM block is synthesized on its MinAD while keeps the number of adders low. Thus, the average logic depth of the resulting MCM block is ensured to be low without any maximum logic depth constraints enforced, and therefore the power consumption is reduced.

Chapter 7

Power Oriented Design of Linear Phase FIR Filters

As reviewed in Section 2.3, while reducing the number of adders directly reduces the synthesis complexity and furthermore indirectly reduces the power consumption, accurate power models show that the power consumption of FIR filters is more involved [1, 2, 66, 67, 79]. In Chapter 6, an algorithm which minimize the average logic depth to indirectly reduce the power consumption is proposed. To more accurately measure the power consumption during the course of the filter design, using a direct power related parameter as an optimization criterion is necessary.

Section 2.3 reviewed that many power models [1, 66, 67] have been proposed for the estimation of power consumption of the adder-and-shift networks used in FIR filters. These models can be classified into two categories: the word-level model and the bit-level model, and in generally, the bit-level models are more accurate but with higher computational complexity than word-level models. Moreover, all these models considered only the dynamic power consumption, i.e., the static power of adder nodes is ignored. However, with the development of IC technologies, the static power becomes more and

more significant in the total power consumption for technologies 90nm or below.

In this Chapter, a new word-level power model of adder network is proposed. The working frequency is included in the power model to taking the both the dynamic power and static power into consideration, and thus makes the proposed algorithm becomes frequency-aware. The parameter used to describe the dynamic power is the adder depth of an adder. The proposed model is simple but relatively accurate and can be served as an optimization criterion in the filter design process. In the following of this chapter, the values computed according to power models are denoted as power index (PI) and the PI is used in the rest of the chapter as a power consumption indicator.

As reviewed in Chapter 6, in efficient MCM and filter coefficient joint optimizations, generally a tree search algorithm is employed to systematically traverse the possible discrete coefficient values [3, 51, 52], and these discrete coefficient values are synthesized to adder networks. During this procedure, the optimization criterion — the number of adders in earlier researches and the PI in the current research — plays a crucial role in both tree traverse and coefficient synthesis. In this chapter, the optimum of a filter is defined as the adder network of the filter with minimum PI. To achieve the minimum PI of the adder network, a MILP based tree search introduced in Section 6.1 is used to traverse all the potential solutions. Furthermore, based on the proposed power model, a coefficient synthesis scheme is proposed to ensure the minimum PI for each discrete coefficient value. Finally, in earlier techniques, the SAs in the FIR filters are considered separately when the corresponding coefficient value is non-zero. However, in the proposed technique, the PIs of the SAs are determined by the corresponding coefficient values and their synthesis. Therefore, a significant difference of the proposed technique from the previous ones is that the PI of the SA is counted into the corresponding coefficient adder networks and is computed when the coefficient value is synthesized. The calculation of the PI of each coefficient is presented together with the coefficient synthesis scheme.

The remaining of the chapter is organized as follows. In Section 7.1, the new power model is described in detail and the comparison with other models is presented. Section 7.2 incorporates the power model into the tree search optimization algorithm presented in Section 7.1. The details of the proposed coefficient synthesis scheme and the evaluation of the synthesis cost are described in Section 7.3. Numerical results are provided in Section 7.4.

7.1 Proposed Power Model for the Adders in Multiplierless FIR Filter

In this section a word-level power model is proposed for the purpose to serve as the filter design criterion. The reason that not a bit-level model is proposed is illustrated in Section 7.2, although mathematically simple and accurate model may exist.

7.1.1 How is the Model to be Proposed

The power consumption of a node, P_N , is given by

$$P_N = P_D + P_S \quad (7.1)$$

where P_D is the dynamic power and P_S is the static power. The dynamic power of a node in a digital circuit is given by [62],

$$P_D = 0.5\alpha C_L V_{dd}^2 f \quad (7.2)$$

where α is the switching activity factor, f is the clock frequency, V_{dd} is the supply voltage and C_L is the load capacitance. The static power of a node in a digital circuit is given by [80],

$$P_S = V_{dd} I_{leak} N_T K_{design} \quad (7.3)$$

where N_T is the number of transistors, I_{leak} is a technology dependent parameter and K_{design} is a design dependent parameter.

In (7.2) and (7.3), for given fabrication technologies and design methodology, V_{dd} , I_{leak} and K_{design} are constants. Furthermore, in word-level power model, the width of all adders are assumed to be the same. Therefore, the number of transistor N_T , for different adders are considered the same, and the load capacitance C_L , i.e., the capacitance of the subsequent stage adder are the same as well. With such simplification, the total power consumption can be expressed as

$$\begin{aligned} P_N &= P_D + P_S \\ &= W_D \alpha f + W_S \\ &= W_D f (\alpha + W_S / (W_D f)) \end{aligned} \tag{7.4}$$

where W_D and W_S are constants for P_D and P_S , respectively, for given technologies and designs. Furthermore, in a power model, the absolute values of P_D and P_S are not important. What important is the relative power index for different adders. Therefore, (7.4) is normalized by a factor of $W_D f$, and the power model of an adder is proposed as

$$P(\alpha, f) = \alpha + W_P / f \tag{7.5}$$

where $W_P = \frac{W_S}{W_D}$. In this power model, α serves as the modeling for the dynamic power, while W_P / f serves as the modeling for the static power. With increasing f , the proportion of the static power in the total power consumption decreases.

In (7.5), for a given frequency, there are two parameters to be determined, i.e., W_P and α . As long as P_D and P_S for any adder at any frequency, f_a , is obtained, W_P may be calculated as

$$W_P = W_S / W_D = P_S / (P_D / \alpha f_a), \tag{7.6}$$

if α can be modeled. Therefore, the key factor for the overall power modeling is how to model the switching activity factor α for different adder nodes.

Basically, there are two ways to reflect the switching activity: one is to directly compute the probability of switching activities based on the switching propagation rules and knowledge of input signal patterns [67], leading to a bit-level model; the other is to use other parameters to indirectly reflect the switching activities such as that in [1]. Many researches have shown that the adder depth is an important factor affecting the switching activity [38,64]. Therefore, in the proposed word-level model, we use the adder depth as the parameter to indirectly estimate the switching activity factor α .

7.1.2 Modeling Procedure

The general procedure to obtain the switching activity α is described as follows:

1. Randomly generate several sets (≥ 20) of filter coefficient values, and an arbitrary MCM algorithm (in my case, the Bull-Horrocks modified (BHM) algorithm [34]) is used to synthesize the coefficient values into adder-and-shift networks.
2. Use HDL language to describe these filter architectures and map the circuits into ASIC using a particular IC technology .
3. Use Nanosim, a transistor level simulation tool, to simulate and estimate the dynamic and static power consumptions of each adder in the filter structure. Compute the mean dynamic power dissipation of the adders in the same adder depth and mean static power of all adders.
4. Use curve fitting tool to fit the mean values of dynamic power consumption into a function of logic depth and determine the corresponding parameters of this function.

In the experiment, the circuits are mapped into ASIC using TSMC 65 nm CMOS process. The operating frequency f_a of the FIR filter is set to 50 MHz and the input signal of all FIR filters are 8 bits random samples with uniform amplitude distribution. The mean values of the dynamic power dissipation of the adders in each logic depth is shown in Figure 7.1, where the values are normalized such that the dynamic power dissipation of the adders with adder depth one is unit. From Figure 7.1, it can be seen that the switching activity factor α of the adders in different adder depths can be fitted to using a linear function in a form of

$$\alpha(d) = kd + c, \quad (7.7)$$

where d is the logic depth and k and c are parameters to be determined. Matlab curve fitting tool “cftool” is used to fit those mean values into the function $kd + c$, and k and c are 0.82 and 0.175, respectively. Therefore, the switching activity model is given by:

$$\alpha(d) = 0.82d + 0.175. \quad (7.8)$$

Once $\alpha(d)$ is obtained, W_P is computed according to (7.6), by setting P_S to the mean static power consumption of all adders, P_S^m , setting P_D to the mean dynamic power of adders in adder depth 1, $P_D^m(1)$, and setting $\alpha(d)$ to $\alpha(1)$. Both $P_D^m(1)$ and P_S^m have been obtained in step 3 in the aforementioned procedure, and $\alpha(1)$ is computed according to (7.8). Thus, the resulting W_P is 296280. The overall power consumption of an adder can be modeled as

$$P(d, f) = 0.82d + 0.175 + 296280/f \quad (7.9)$$

In the above computation of W_P , the adders with adder depth 1 are arbitrarily chosen as the measure of dynamic power. When adders at other logic depths are used for the calculation, the difference for the resulting W_P is insignificant. With the power model for

Figure 7.1: The mean values of normalized power consumption vs. the logic depth of adders d , and the fitting curve.

an adder, the overall power consumption of a MCM FIR filter at a particular frequency is modeled as the sum of the PIs given in (7.9) of all adders involved in the FIR filter, i.e.,

$$P_{total} = \sum_j P(d_j, f), \quad (7.10)$$

where j is the index of adders (including both MBAs and SAs) in the FIR filter and d_j is the adder depth of adder j .

7.1.3 Verification of the Model

The accuracy of the above model is verified by a set of adder-and-shift networks synthesized and simulated at the frequencies 500 KHz and 50 MHz, respectively.

The testbench MCM block is shown in Figure 7.2. The simulated power consumption and the computed PIs by different models for the structure in Figure 7.2 for frequencies at 500 KHz and 50 MHz are given in Table 7.1 and 7.2, respectively. A normalized ratio γ [66, 81] is used to verify the accuracy of the power estimation. For a MCM with J

Figure 7.2: MCM block constructed by adders 3, 5, 11, 25, 49, 55, 41 and 99

Table 7.1: PIs of Different Models and Simulated Power Consumptions of the Adders in Figure 7.2 at $f=500$ KHz

Adder name	GPC [1]	bit-level model [2]	Proposed model	simulated power(mw)
3	1	43.5	1.588	1.34e-004
5	1	42.01	1.588	1.12e-004
11	2	72.0	2.407	2.27e-004
25	3	78.3	2.407	1.92e-004
41	4	102.3	3.228	2.33e-004
49	6	115.8	3.228	2.58e-004
55	8	132.3	4.048	3.23e-004
99	10	134.1	4.048	3.16e-004

Table 7.2: PIs of Different Models and Simulated Power Consumptions of the Adders in Figure 7.2 at $f=50$ MHz

Adder name	GPC [1]	bit-level model [2]	Proposed model	simulated power(mw)
3	1	43.5	1.001	8.89e-003
5	1	42.01	1.001	7.22e-003
11	2	72.0	1.821	1.44e-002
25	3	78.3	1.821	1.28e-002
41	4	102.3	2.641	1.74e-002
49	6	115.8	2.641	2.32e-002
55	8	132.3	3.461	2.57e-002
99	10	134.1	3.461	2.43e-002

adders, γ_j for adder j is defined as

$$\gamma_j = \frac{J \frac{P(d_j, f)}{S_j}}{\sum_{k=1}^J \frac{P(d_k, f)}{S_k}}, \quad (7.11)$$

where $P(d_j, f)$ is the PI obtained from the proposed model (7.9), S_j is the power consumption obtained from simulation, for adder j . Under such definition, a γ_j closer to unit indicates a higher estimation accuracy [66, 81].

Figures 7.3 and 7.4 show the normalized ratios of GPC model [1], that of a bit-level model [2], and that of the proposed model for all the adders in the structure of Figure 7.2 at frequencies 500 KHz and 50 MHz, respectively.

In Figure 7.3 showing the case at frequency 500 KHz, the maximum estimation errors deviated from unit for the three models, i.e., the GPC model, the bit-level model and the proposed model, are 0.83, 0.13 and 0.10, respectively. The corresponding mean estimation errors are 0.4064, 0.0862 and 0.0378, respectively. It can be seen that the proposed word-level power model is more accurate than the GPC model and the bit-level model [2]. The reason is that both models in [1, 2] ignored the static power. However, at 500 KHz, the proportion of static power is around 40% of the total power consumption. Ignoring the static power obviously will result in a significant error for filters operating at low frequency. Figure 7.4 shows the case at frequency 50 MHz. The maximum estimation errors deviated from unit for the three models are 0.8, 0.11 and 0.15, respectively, and the corresponding mean estimation errors are 0.3261, 0.0623 and 0.0902, respectively. It can be seen that the proposed word-level power model is much more accurate than the GPC model; compared with the bit-level model, the proposed model is slightly less accurate than the bit-level model proposed in [2]. This is because at 50 MHz, the dynamic power consumption consists of 99% of the total power consumption and the static power is negligible. The accuracy of the bit-level model [2] for modeling the dynamic power is

Figure 7.3: Normalized ratios γ_j of the adders modeled by GPC, bit-level model and the proposed power model for the adders in Figure 7.2 at $f = 500$ KHz

Figure 7.4: Normalized ratios γ_j of the adders modeled by GPC [1], bit-level model [2] and the proposed power model for the adders in Figure 7.2 at $f = 50$ MHz

higher than that of the proposed model, and that is the reason why the bit-level model has a smaller estimation error than that of the proposed model at frequency 50 MHz. However, the computation of the proposed PI is much simpler than that of the bit-level model.

7.1.4 Remarks of the Proposed Model

First, the proposed power model for the adder is a rough word-level model, where the number of FAs (i.e., the wordlength of the adder) and the load capacitances of the adder are not taken into consideration. It is not guaranteed that the architectures with smaller PI computed by the proposed model definitely have lower power consumption. At high frequency, this model is more accurate than the GPC and slightly less accurate than the bit-level model with significantly lower complexity. At low frequency, it has higher accuracy than the GPC and the bit-level model.

Second, if the frequency increases, the power consumption of an adder node certainly will increase. However, according to (7.9), the PI value of the adder node in a logic depth is reduced if f increases. The reason is in the proposed model, only the relative ratios of power consumption among the adders in different adder depth is considered, because in the optimization process, only the ratios among the different adders is used to guide the search program.

Last, the parameters k , c and W_P may vary when different IC technology and design methodology are applied, but the procedure to determine the values of those parameters is the same as that discussed in Section 7.1.2. Furthermore, the parameters are obtained for the filters operating at frequency 50 MHz. Tests as well at other frequencies such as 500 KHz and 5 MHz are also conducted, and the maximum variations of the parameters k , c and W_P are 0.21%, 0.32% and 0.57%, respectively. Such small variations are negligible

and ignored.

Overall, the key idea behind this model is to provide a systematic approach to determine the relation between the adder depth, the operating frequency, and the power consumption of the adder.

7.2 Power Oriented Optimization of Multiplierless FIR Filters

The proposed power model is incorporated in the tree search algorithm proposed in the Section 6.1.2 to optimize the FIR filters achieving minimum power consumption. In the optimization, the coefficients are selected to be quantized to discrete values and further synthesized to adder-and-shift network. Cut-off schemes are exploited to accelerate the search. Both the coefficient synthesis and tree cut-off are based on the proposed PI. A new cut-off scheme is proposed in this section, while the synthesis of the coefficient values will be discussed in next section.

In the tree search procedure described in section 6.1.2, two cut-off schemes are adopted to accelerate the search. The second cut off mechanism in step 6 to check the feasibility of a node keeps unchanged in the proposed search.

The first cut-off mechanism in step 4 is refined to cater for the minimum power design. In step 4, instead of computing the minimal adder cost as that in Section 6.1.2, the minimum PI cost of the realization of the current node is computed, and whether the node or the node's descendent can grow to be the optimum solution, (i.e., if the overall PI upto the node, denoted as P_{all} , is smaller than that of the best solution P_{best}) is checked. If all coefficients of the current node have been quantized and synthesized, P_{all} is simply the sum of PIs of all coefficients P_{fix} , i.e., $P_{all} = P_{fix}$. However, in the search procedure,

most nodes have partial fixed and partial unfixed coefficients. If P_{fix} is used to compare with P_{best} as the cut-off criteria, the criteria is too conservative to search efficiently. In this section, a more efficient cut-off criteria is proposed that P_{all} is determined as

$$P_{all} = P_{fix} + P_{unfix} \quad (7.12)$$

where P_{unfix} is estimated as the lower bound of the PIs contributed by the unfixed coefficients. Assuming that there are N_{unfix} unfixed coefficients $h(i_k)$ for $1 \leq k \leq N_{unfix}$, among which N_{zero} coefficients, $h(j_l)$ for $1 \leq l \leq N_{zero}$, may have the possibility to be zero in the subsequent quantization process, P_{unfix} is given by

$$P_{unfix} = \left(\sum_{k=1}^{N_{unfix}} N_s(i_k) - \sum_{l=1}^{N_{zero}} N_s(j_l) \right) P(1, f) \quad (7.13)$$

where $N_s(i_k)$ and $N_s(j_l)$ are the number of SAs for the corresponding unfixed coefficients $h(i_k)$ and $h(j_l)$, respectively. $N_s(i)$ for coefficient $h(i)$, for $0 \leq i \leq \lfloor \frac{N}{2} \rfloor$, i.e., the number of SAs that a coefficient $h(i)$ is connected to, for an N -th order symmetric FIR filters in direct transposed form, is given as:

$$N_s(i) = \begin{cases} 1 & \text{for } i = 0, \frac{N}{2} \\ 2 & \text{otherwise} \end{cases} \quad (7.14)$$

if N is even, and

$$N_s(i) = \begin{cases} 1 & \text{for } i = 0 \\ 2 & \text{otherwise} \end{cases} \quad (7.15)$$

if N is odd.

The above estimation on the term P_{unfix} is based on the fact that the unfixed coefficients can be either 0 or non-zero in the future optimization. When the coefficient is non-zero, even it can be realized by using 0 MBA, $N_s(i_k)$ SAs are still needed for the realization of $h(i_k)$. Therefore, when they are all non-zero, the minimum power consumption of these unfixed coefficients are $\sum_{k=1}^{N_{unfix}} N_s(i_k)P(1, f)$, since these SAs are adders on adder depth 1. Considering that N_{zero} coefficients may be zero in the future optimization, the number of corresponding SAs that can be removed is $\sum_{l=1}^{N_{zero}} N_s(j_l)$.

The cut-off scheme based on such defined P_{all} is much more efficient than using P_{fix} only, and makes the design of relatively long filter possible. However, due to this cut-off scheme, although mathematically simple and accurate bit-level power model may be developed, bit-level models are not suitable to be used in the proposed tree search. The reason is because the width of the SAs for those unfixed coefficients are difficult to be estimated. If bit-level model is used, only a very conservative low bound of the adder-width of SAs for those unfixed coefficients can be used, but this makes the cut-off schemes highly inefficient and the optimization process becomes unaffordably time consuming.

7.3 Coefficient Synthesis and Cost Estimation

Section 7.2 introduced how to estimate the PI of the unfixed coefficients. The remaining problems are 1) how to evaluate the PI cost when a coefficient is fixed to a discrete value and synthesized, and 2) how to synthesize a discrete value such that the cost in synthesizing the coefficient is minimum. For the first problem, if we consider only the MBAs, the PI cost of the coefficient can be evaluated using (7.9) directly once the number of adders used to synthesize the coefficient and the adder depths of these adders are available. However, in FIR filter implementations, each coefficient corresponds to one or two SAs as discussed in Section 7.2. PIs occur also for these SAs. Furthermore, the

synthesis of the coefficient affects the PI of the corresponding SAs. Therefore, to reflect the entire power consumption in real time during the optimization when a coefficient is synthesized, the PI of the corresponding SAs are also counted into the cost of the synthesis of the coefficient.

Since the PI for an adder is proportional to the only factor — the adder depth as in (7.9) for a given operating frequency, synthesizing each necessary adder on its minimum adder depth (MinAD) guarantees the minimum power consumption. Therefore, in the synthesis of coefficients, if the number of adders used is minimum and meanwhile the coefficient is synthesized on its MinAD, the synthesis is optimum in the sense of PI. However, when the coefficient cannot be synthesized under such optimum condition, more complicated and synthesis mechanisms rules based on the PI values have to be developed.

In this section, first, a synthesis mechanism which trades off the adder depth and the number of adders used to synthesize the coefficient is proposed. Second, the approach to compute the overall PI cost to synthesize the coefficient, including that of the corresponding SAs and extra adders if any, is proposed. Last, an example to illustrate the synthesis mechanism and the computation of the power cost is presented.

7.3.1 A New Synthesis Mechanism leading to low PI implementation

In the tree search described in Section 6.1.2, each node is associated with an expanding subexpression basis set. Besides that, each node is also associated with an expanding auxiliary basis set. The auxiliary set for the root is initialized to be null, and each node inherits the set from its parent as well. When a node is generated in the tree search, $h(d_s)$ is fixed to a discrete value in its range. Let the discrete value be denoted as

$h^0(k)$. The procedure (as shown in Figure 7.5) to synthesize $h^0(k)$ based on its associated subexpression basis set and auxiliary set is given as follows.

1. The MinAD to realize $h^0(k)$ is computed according to (6.2).
2. If $h^0(k)$ can be realized based on the current basis set and the auxiliary set using not more than an adder, and the synthesis is on its MinAD, the synthesis is optimum in the sense of minimum PI. The subexpression basis set is updated by including $h^0(k)$ if an adder is used.
3. If $h^0(k)$ cannot be synthesized as in step 2, the coefficient is synthesized in the following 2 ways: (1) Synthesize $h^0(k)$ using not more than an adder based on the current basis set but on the adder depth higher than MinAD, and (2) synthesize $h^0(k)$ on its MinAD with an inserting of a first order basis. If only one synthesis is successful, the successful synthesis is adopted. If both synthesis are successful, the PIs of the 2 ways are computed. The computation of the PIs is introduced in Section 7.3.2. The synthesis with lower PI is adopted.
4. If $h^0(k)$ can not be successfully synthesized using aforementioned ways, this value is just added to the basis set without updating the auxiliary set. The basis set is flagged to indicate that some coefficient values are not synthesized. The reason that not allowing more than one basis or higher order basis to be inserted into the auxiliary set is that from design experiences, adding not more than a first order auxiliary basis for each node is enough to successfully synthesize all the coefficients while the PI of the synthesis of the coefficient set keeps low.

Figure 7.5: Flow-chart to synthesize a coefficient $h(k)$

7.3.2 Cost of Coefficient Realization

As introduced in the beginning of this section, the PI of the corresponding SAs of a coefficient has to be counted into the cost of the coefficient when it is synthesized. In addition, if the coefficient is synthesized by using existing subexpression bases from the basis set, the cost to realize these bases has been counted when they are synthesized and they are not counted repeatly.

With these considerations, the cost to synthesize a fixed coefficient is computed as:

$$P_{fix} = P_{MA} + P_{SA} + P_{extra} \quad (7.16)$$

where P_{MA} , P_{SA} and P_{extra} are the PI of the MBA directly synthesizing the coefficient, the PI of the corresponding SA(s) and the PI of the MBA to synthesize the extra basis if used, respectively. Depending on the values of the discrete coefficients and how they are synthesized, P_{MA} , P_{SA} and P_{extra} are computed as follows:

Case 1: If the coefficient value is 0, $P_{MA} = P_{SA} = P_{extra} = 0$, since neither MA nor SA is needed.

Case 2: If the coefficient value $h(n)$ can be expressed as an existing subexpression basis scaled by a power-of-two term, then $P_{MA} = P_{extra} = 0$ and $P_{SA} = N_s(n)P(d_{basis} + 1, f)$, where d_{basis} is the MinAD of the subexpression basis used, and $N_s(n)$ is given in (7.14) or (7.15).

Case 3: If the coefficient value $h(n)$ is synthesized without inserting any extra basis, then $P_{MA} = P(d_{h(n)}, f)$, $P_{SA} = N_s(n)P(d_{h(n)} + 1, f)$ and $P_{extra} = 0$, where $d_{h(n)}$ is the adder depth of the synthesized $h(n)$.

Case 4: If the coefficient value $h(n)$ is synthesized by using a first order extra basis, then $P_{MA} = P(d_{h(n)}, f)$, $P_{SA} = N_s(n)P(d_{h(n)} + 1, f)$ and $P_{extra} = P(1, f)$, since only adder depth 1 extra basis may be used.

Table 7.3: An Example to Illustrate the Proposed Coefficient Synthesis Procedure, the Update of Basis Sets, and the Power Cost of Each Coefficient

Coefficients	MinAD	Main basis	Auxiliary basis	Power Cost
$h(0)=3$	1	$\{0,1,3\}$	Null	2.810
$h(1)=11$	2	$\{0,1,3,11\}$	Null	7.103
$h(2)=43$	2	$\{0,1,3,11,43\}$	$\{5\}$	8.104
$h(3)=6$	1	$\{0,1,3,11,43\}$	$\{5\}$	3.642

7.3.3 An Example to Illustrate the Synthesis Mechanism and the Computation of the Power Cost

A FIR filter with filter length 8 (i.e., 4 distinct coefficients) is given to illustrate the synthesis mechanism in details. The frequency f is assume to be 50 MHz. As shown in Table 7.3, first, $h(0)$ is fixed to 3 and it can be synthesized on its MinAD based on the initial zeroth order basis set. The basis set is updated to include 3, and the auxiliary basis set is Null. The synthesis of 3 corresponds to the Case 3 in Section 7.3.2. Therefore the total power cost is computed as $P(1, f) + P(2, f)$, since the MinAD of 3 is 1; the cost is computed to 2.810. Thereafter, $h(1)$ is fixed to 11 whose CSD representation is $10\bar{1}0\bar{1}$. According to (6.2), the MinAD of 11 is 2. Clearly, $11=3 \times 2^2 - 1$ can be realized on its MinAD based on the current basis set. After the synthesis, the basis set is $\{0,1,3,11\}$ with the auxiliary basis set being Null. The total power cost for $h(1)$ is computed as $P(2, f) + 2P(3, f)$, since the synthesis of 11 is also the Cases 3 synthesis. The resulting cost is 7.103.

$h(2)$ can be synthesized based on the current basis set using one adder as $(43 = 11 \times 2^2 - 1)$, but the adder depth of such synthesis is 3 which is larger than its MinAD of 2. The PI cost of realization $h(2)$ on adder depth 3 is computed as $P(3, f) + 2P(4, f) = 9.563$. If a first order basis 5 is inserted into the auxiliary basis set, such that 43 can be synthesized as $5 \times 2^3 + 3$ on its MinAD, the total PI cost for $h(2)$ is computed as $P(2, f) + 2P(3, f) + P(1, f) = 8.104$. Since $8.103 < 9.563$, the first order basis 5 is

inserted into auxiliary set. The basis set and the auxiliary basis set are updated to be $\{0, 1, 3, 11, 43\}$ and $\{5\}$, respectively.

The last coefficient $h(3)$ can be realized as Case 2 in Section 7.3.2 and the adder depth for the subexpression basis used in the synthesis is 1. Therefore, the power cost is $P_{SA} = 2P(2, f)$, resulting in 3.642.

Therefore, P_{total} of this filter is the sum of the power costs of all coefficients, i.e., 21.659.

7.4 Design Examples

In this section, 6 benchmark filters with labels G1, Y1, Y2, S2,L2 and A1 are given to illustrate the superiority of the proposed algorithm. All these filters' specifications have been listed in Table 2.1. The results obtained by the proposed algorithm are compared with the best published adder-cost oriented algorithm [3] and also compared with the adder-cost algorithm with MinAD constraint proposed in Chapter 6, where the MinAD constraint is imposed.

The actual power consumptions of different filter architectures are simulated. The IC technology used is TSMC 65nm standard cell, and Nanosim, a transistor level tool, is used to measure the power consumption of each architecture. The clock of each FIR filter is set to 500 KHz and 50 MHz and the input signal of all FIR filter are 8 bits random samples with uniform amplitude distribution. All the examples are designed using a 2.4-GHz desktop PC and the runtime of the program is restricted to the same as that of using algorithm in [3], if the program need to run more than 10 hours.

7.4.1 Comparison of Power Consumption

The power consumption and the relevant information of the designs obtained using the proposed algorithm and the algorithm in [3] are listed in Table 7.4 for comparison.

Table 7.4 shows that all designs obtained by using the proposed algorithm have lower PI cost than that obtained in [3] except the two designs X1 and S1 of which both algorithms produce the same coefficient set and synthesis architecture. Furthermore, the PI cost of different designs are consistent with the measured power consumption by simulation. The maximum saving in power consumption is as high as 16.01% for G1 filter at frequency 50 MHz.

It can also be seen from Table 7.4 that at frequency 50 MHz, the power saving achieved by the proposed algorithm is larger than that achieved at 500 KHz. The reason is because at lower frequency, the proportion of the static power in the total power is higher. Since the static power is directly related to the number of adders (more accurately related to the number of transistors N_T), at low frequency, minimizing the adder cost is close to the minimization of the power consumption. In fact, when the frequency is reduced to extremely low, the proposed algorithm results the same design as the algorithm in [3]. However, at higher frequency, the dynamic power dominates, and the total power consumption is no longer depending on the number of adders. Thus the proposed algorithm achieves more significant power saving.

The proposed power-oriented algorithm is also compared with the technique proposed in Chapter 6. The comparisons results are listed in Table 7.5. From Table 7.5, it can be seen that the proposed algorithms either generate the same designs as or slightly outperforms that obtained in Chapter 6 in the terms of power consumption. This affirms that the technique proposed in Chapter 6 constraining the synthesis of coefficients on

their MinAD is a very effective algorithm for the design of low power FIR filters. However, compared with the algorithm proposed in Chapter 6, a significant advantage of the proposed algorithm in this Chapter is its frequency-awareness.

Overall, the proposed algorithm is adaptive to the frequency, and the resultant design and synthesis of the filter may be different for different frequencies, and each design achieves the minimum PI in the respective frequency. The coefficient sets for operating frequency 50 MHz obtained by the proposed algorithm and their synthesis are listed in Table 7.6, 7.7 and 7.8 for references.

Table 7.4: The Result Comparisons between the Proposed Algorithm and the Algorithm in [3]. EWL is the Effective wordlength, MLD is the Maximum Logic Depth, ALD is the Average Logic Depth, NA is the Number of Adders and Power is the Simulated Power Consumption of the Filter.

Filter	EWL	The proposed algorithm						The Algorithm in [3]					
		MLD	ALD	NA	PI	Power	runtime	MLD	ALD	NA	PI	Power	runtime
G1 (500 KHz)	6	3	1.29	17	31.2	0.0053	0.43s	3	1.58	17	36.7	0.0059	0.39s
G1 (50 MHz)					21.2	0.361	0.47s				24.8	0.431	
X1	10							Both algorithms produce the same result					
Y1 (500 KHz)	9	3	2	31	72.9	0.0133	3m01s	4	2.33	30	78.8	0.0143	1m40s
Y1 (50 MHz)					57.4	0.988	3m11s				64.6	1.071	
Y2 (500 KHz)	11	4	2.31	38	101.6	0.0190	1m31s	5	2.73	37	111.7	0.0204	1m20s
Y2 (50 MHz)					80.1	1.501	2m24s				91.4	1.604	
S1	7							Both algorithms produce the same result					
S2 (500 KHz)	10	4	2.44	79	212.6	0.0393	11h24m*	5	2.56	76	217.2	0.0405	11h24m
S2 (50 MHz)			2.29	80	161.8	2.966					170.9	3.089	
L2 (500 KHz)	10	4	2.02	76	185.9	0.0350	11h19m*	4	2.31	73	194.1	0.0376	11h19m
L2 (50 MHz)					140.4	2.587					154.4	2.847	
A1 (500 KHz)	10	3	2.17	71	181.6	0.0345	35h47m*	5	2.41	68	193.1	0.0375	35h47m
A1 (50 MHz)		3	2.15	72	140.8	2.565					148.6	2.893	

* manually stopped

Table 7.5: The Power Comparison between the Proposed Algorithm and Adder-cost Oriented Algorithm with MinAD constraint proposed in Chapter 6

Filters	MinAD algorithm / Proposed algorithm				
	G1	X1,Y1,Y2,S1	S2	L2	A1
500 KHz	0.0059 / 0.0053	same result	0.0393 / 0.0393	0.0363 / 0.0350	0.0363 / 0.0345
50 MHz	0.431 / 0.361	— same result	3.031 / 2.966	2.747 / 2.587	2.783 / 2.565

7.4.2 Computational Complexity

Same as other linear programming based search algorithms [3, 51, 52], the major complexity of the proposed algorithm is to solve the linear programming formulated in (6.1). Other computations such as finding the MinAD or PI of each coefficient are negligible compared with solving (6.1). Therefore, the computation time can be expressed as [3]

$$ComputationTime = \beta 2^{NB}T \quad (7.17)$$

where N is the number of coefficients to be optimized, B is the EWL, T is the average runtime of solving a linear program and β is a “cutoff” factor.

While the computational complexity of both the proposed algorithm and the algorithm in [3] have the exponential relation with N and B , the proposed one has a larger value of β than that of [3], although a cut-off mechanism tailored for the proposed algorithm is proposed. This can be seen from Table 7.4, where the runtime of the proposed algorithm is longer than that in [3]. The reason for the less efficiency in cut-off is due to the nature of the problem: the PIs of unfixed coefficient P_{unfix} in (7.12) is estimated using its lower bound, i.e., $N_s(i_k)P(1)$. Only the coefficients without using any MBA but an adder depth 1 SAs can have such low PI. However, a large amount of unfixed coefficients in fact need more adders when they are fixed, but they are kept due to the underestimation of P_{unfix} and they are growing further to generate more children before they are cut off. Meanwhile, since the most kept nodes cannot lead to a solution with lower overall PI, such that P_{best} remains at high level resulting in inefficient in cut-off.

Table 7.6: Impulse Response, Subexpression Bases and Coefficient Synthesis of Design Examples G1, Y1 and Y2

Filter G1 $h(n) = h(15 - n)$ for $0 \leq n \leq 7$ passband gain:193.5701 EWL: 6 subexpression set {39} auxiliary basis set {5}		
$h(0) = 1$ $h(1) = 1 \times 2^1$ $h(2) = -1$	$h(3) = -1 \times 2^3$ $h(4) = -1 \times 2^3$ $h(5) = 1 \times 2^3$	$h(6) = 39$ $h(7) = 1 \times 2^6$
synthesis of bases in the basis sets		
$5 = 1 \times 2^2 + 1$	$39 = 5 \times 2^3 - 1$	
Filter Y1 $h(n) = h(29 - n)$ for $0 \leq n \leq 14$ passband gain:1372.1166 EWL: 9 subexpression set { 9, 11, 3, 43, 35, 53, 271 } auxiliary basis set {17}		
$h(0) = -1$ $h(1) = -1 \times 2^2$ $h(2) = 0$ $h(3) = 9$ $h(4) = 1 \times 2^3$	$h(5) = -5 \times 2^1 - 1$ $h(6) = -3 \times 2^3$ $h(7) = 0$ $h(8) = 17 \times 2^1 + 9$ $h(9) = 17 \times 2^1 + 1$	$h(10) = -3 \times 2^4$ $h(11) = -53 \times 2^1$ $h(12) = 0$ $h(13) = 17 \times 2^4 - 1$ $h(14) = 1 \times 2^9$
synthesis of bases in the basis sets		
$9 = 1 \times 2^3 + 1$ $11 = 1 \times 2^1 + 9$ $3 = 1 \times 2^1 + 1$	$43 = 17 \times 2^1 + 9$ $35 = 1 \times 2^5 + 3$ $53 = 9 \times 2^2 + 17$	$271 = 17 \times 2^4 - 1$
auxiliary basis set {17}		
Filter Y2 $h(n) = h(33 - n)$ for $0 \leq n \leq 16$ passband gain:5141.6534 EWL: 11 subexpression set {3,9,11,43,101,171,137,91,255,15 } auxiliary basis set {5}		
$h(0) = 3 \times 2^1$ $h(1) = 3 \times 2^1$ $h(2) = -9^0$ $h(3) = -11 \times 2^1$ $h(4) = 0$ $h(5) = 43$	$h(6) = 9 \times 2^2$ $h(7) = -3 \times 2^4$ $h(8) = -101$ $h(9) = 0$ $h(10) = 171$ $h(11) = 137$	$h(12) = -91 \times 2^1$ $h(13) = -101 \times 2^2$ $h(14) = 0$ $h(15) = 255 \times 2^2$ $h(16) = 15 \times 2^7$
synthesis of bases in the basis sets		
$3 = 1 \times 2^1 + 1$ $9 = 1 + 1 \times 2^3$ $11 = 3 + 1 \times 2^3$ $255 = 1 \times 2^8 - 1$	$5 = 1 \times 2^2 + 1$ $43 = 5 \times 2^3 + 3$ $101 = 3 \times 2^5 + 5$ $15 = 1 \times 2^4 - 1$	$171 = 43 \times 2^2 - 1$ $137 = 1 \times 2^7 + 3$ $91 = 3 \times 2^5 - 5$

Table 7.7: Impulse Response, Subexpression Bases and Coefficient Synthesis of Design Examples A1 and L2

Filter A1 $h(n) = h(58 - n)$ for $0 \leq n \leq 28$ passband gain:6066.4133 EWL: 10 subexpression set: $\{7, 9, 5, 21, 15, 29, 39, 129, 89, 23, 65, 175, 399, 315, 13, 485, 255\}$ auxiliary basis set $\{3\}$		
$h(0) = 1 \times 2^2$ $h(1) = 7$ $h(2) = 9$ $h(3) = 5 \times 2^1$ $h(4) = 7$ $h(5) = 0$ $h(6) = -5 \times 2^1$ $h(7) = -7 \times 2^1 - 7$ $h(8) = -15 \times 2^1$ $h(9) = -15 \times 2^1$	$h(10) = -7 \times 2^1 - 7$ $h(11) = 0$ $h(12) = 7 \times 2^2$ $h(13) = 29 \times 2^1$ $h(14) = 5 \times 2^4 - 2$ $h(15) = 5 \times 2^4$ $h(16) = 7 \times 2^3$ $h(17) = 7$ $h(18) = -15 \times 2^2$ $h(19) = -129$	$h(20) = -89 \times 2^1$ $h(21) = -23 \times 2^3$ $h(22) = -65 \times 2^1$ $h(23) = -1 \times 2^3$ $h(24) = 175$ $h(25) = 399$ $h(26) = 315 \times 2^1$ $h(27) = 13 \times 2^6$ $h(28) = 485 \times 2^1$ $h(29) = 255 \times 2^2$
synthesis of bases in the basis sets		
$7 = 1 \times 2^3 - 1$ $9 = 1 + 1 \times 2^3$ $5 = 1 + 1 \times 2^2$ $21 = 7 \times 2^1 + 7$ $15 = 1 \times 2^4 - 1$ $29 = 7 \times 2^2 + 1$ $39 = 1 \times 2^5 + 7$	$129 = 1 \times 2^7 + 1$ $69 = 1 \times 2^6 + 5$ $23 = 1 \times 2^4 + 7$ $65 = 1 \times 2^6 + 1$ $175 = 5 \times 2^5 + 15$ $399 = 3 \times 2^7 + 15$ $315 = 5 \times 2^6 - 5$	$13 = 7 \times 2^1 - 1$ $485 = 15 \times 2^5 + 5$ $255 = 1 \times 2^8 - 1$

Filter L2 $h(n) = h(58 - n)$ for $0 \leq n \leq 28$ passband gain:4266.8398 EWL: 10 subexpression set: $\{11, 3, 15, 21, 9, 17, 25, 65, 51, 85, 127, 45, 663, 223, 489\}$ auxiliary basis set $\{5\}$		
$h(0) = 1 \times 2^2$ $h(1) = 1 \times 2^3$ $h(2) = 11$ $h(3) = 11$ $h(4) = 3 \times 2^1$ $h(5) = -1 \times 2^2$ $h(6) = -15$ $h(7) = -1 \times 2^4 - 5$ $h(8) = -9 \times 2^1$ $h(9) = -1 \times 2^2$ $h(10) = 1 \times 2^4$	$h(11) = 1 \times 2^5$ $h(12) = 17 \times 2^1$ $h(13) = 1 \times 2^4$ $h(14) = -17$ $h(15) = -25 \times 2^1$ $h(16) = -65$ $h(17) = -3 \times 2^4$ $h(18) = 0$ $h(19) = 15 \times 2^2$ $h(20) = 51 \times 2^1$ $h(21) = 3 \times 2^5$	$h(22) = 1 \times 2^5$ $h(23) = -9 \times 2^3$ $h(24) = -85 \times 2^1$ $h(25) = -51 \times 2^2$ $h(26) = -127$ $h(27) = 9 \times 2^3$ $h(28) = 45 \times 2^3$ $h(29) = 663$ $h(30) = 223 \times 2^2$ $h(31) = 489 \times 2^1$
synthesis of bases in the basis sets		
$5 = 1 \times 2^2 + 1$ $11 = 1 + 5 \times 2^1$ $3 = 1 + 1 \times 2^1$ $15 = 1 \times 2^4 - 1$ $21 = 5 \times 2^2 + 1$ $9 = 1 \times 2^3 + 1$ $17 = 1 \times 2^4 + 1$	$25 = 3 \times 2^3 + 1$ $65 = 1 \times 2^6 + 1$ $51 = 3 \times 2^4 + 1$ $85 = 5 \times 2^4 + 5$ $127 = 1 \times 2^7 - 1$ $45 = 3 \times 2^4 - 3$ $663 = 21 \times 2^5 - 9$	$223 = 15 \times 2^4 - 17$ $489 = 15 \times 2^5 + 9$

Table 7.8: Impulse Response, Subexpression Bases and Coefficient Synthesis of Design Example S2

Filter S2 $h(n) = h(59 - n)$ for $0 \leq n \leq 29$ passband gain:11817.8847 EWL: 10 subexpression set: {3, 7, 9, 5, 35, 51, 17, 43, 99, 27, 15, 101, 191, 293, 65, 317, 741, 209, 911, 965, 31} auxiliary basis set {Null}		
$h(0) = 3 \times 2^1$	$h(10) = -17 \times 2^2$	$h(20) = 3 \times 2^6 - 1$
$h(1) = 7$	$h(11) = -43 \times 2^1$	$h(21) = 9 \times 2^5 + 5$
$h(2) = 3 \times 2^1$	$h(12) = -99$	$h(22) = 101 \times 2^2$
$h(3) = 3 \times 2^1$	$h(13) = -27 \times 2^2$	$h(23) = 65 \times 2^3$
$h(4) = 1 \times 2^2$	$h(14) = -27 \times 2^2$	$h(24) = 317 \times 2^1$
$h(5) = -1$	$h(15) = -3 \times 2^5$	$h(25) = 99 \times 2^3 - 51$
$h(6) = -9$	$h(16) = -9 \times 2^3$	$h(26) = 209 \times 2^2$
$h(7) = -5 \times 2^2$	$h(17) = -15 \times 2^1$	$h(27) = 911$
$h(8) = -35$	$h(18) = 27$	$h(28) = 965$
$h(9) = -51$	$h(19) = 101$	$h(29) = 31 \times 2^5$
synthesis of bases in the basis sets		
$3 = 1 + 1 \times 2^1$	$7 = 1 \times 2^3 - 1$	$9 = 1 + 1 \times 2^3$
$5 = 1 + 1 \times 2^2$	$35 = 1 \times 2^5 + 3$	$51 = 3 + 3 \times 2^4$
$17 = 1 + 1 \times 2^4$	$43 = 5 \times 2^3 + 3$	$99 = 3 + 3 \times 2^5$
$27 = 3 + 3 \times 2^3$	$15 = 1 \times 2^4 - 1$	$101 = 3 \times 2^5 + 5$
$191 = 3 \times 2^6 - 1$	$293 = 9 \times 2^5 + 5$	$65 = 1 \times 2^6 + 1$
$317 = 5 \times 2^6 - 3$	$741 = 99 \times 2^3 + 51$	$209 = 3 \times 2^6 + 17$
$911 = 7 \times 2^7 + 15$	$965 = 15 \times 2^6 + 5$	$31 = 1 \times 2^5 - 1$

On the contrary, in [3], more kept nodes may lead to a solution with less overall adders, because as long as the coefficients are fixed to integers without using any MBA and the filter is still feasible, even if the corresponding SA is on adder depth higher than 1, better solution is produced for more efficient cut-off.

So, in summary, in the proposed algorithm, the cost of unfixed coefficients is generally more severely underestimated than that in [3] to maintain the optimality of the search and this leads to less efficiency in cut-off. Taking G1 filter as an example, the proposed algorithm did 327 linear programming, while the adder-oriented algorithm [3] only did 284 times. If a bit-level model is used, the cut-off mechanism will become more inefficient, as analyzed in section 7.2. In other words, for the design of long FIR filters, the global optimal solution may not be obtained within the limited time due to the high computational complexity of the proposed algorithm. Nevertheless, although no theoretical

analysis is available, design experiences show that the proposed algorithm generates the designs with lower power consumption than that of the algorithm in [3] in most cases for the filter orders less than 60 when the runtime is set to an acceptable level, for example 24 hours.

7.5 Conclusion

This chapter proposed a power oriented FIR filter optimization technique. While the traditional MILP is employed in the search of discrete coefficient values, the novelty of the technique lies on the fact that a power cost is for the first time introduced into the MILP as a search cut-off criterion. The power cost is calculated based on a new proposed word-level power model which considers both the dynamic and static powers. This power model is also used for the synthesis of discrete values. The algorithm based on the proposed model may generate different designs for different operating frequencies, and the designed filters consume the minimum power for the respective frequency. An additional distinct conceptual difference of the proposed technique from the previous techniques is that the cost of SAs are counted into the corresponding coefficient values when they are synthesized. The designs show that at low frequency, the PI indicator is slightly better than and is close to the indicator of the number of adders, while at high frequency, the PI indicator is much more accurate than that of the number of adders. By using the above new techniques, the power consumptions of the benchmark filters designed are further reduced compared with existing design techniques.

Chapter 8

Conclusions and Future Works

8.1 Conclusions

This thesis, discussed the design of low hardware cost and low power consumption multiplierless FIR filters. Early researches in this area have well solved the single-stage filter design with orders around 70 or less for low hardware cost [3, 52, 61]. Long filter and multiple-stage filter designs were also considered by some researches [55, 69]. However, there are still spaces for improvement in the design of these type of filters. On the other hand, the researches for the design of low power multiplierless FIR filter is even less [64]. This research proposed 5 algorithms and they are suited to different scenarios.

First, the design of long filters with order higher than 70 is discussed in Chapters 3 and 4, where an improved GA and a tree search algorithm are proposed. The difficulty in the design of long filter is mainly because of the large search space but with only sparse feasible solutions available. To overcome this difficulty, both proposed algorithms partitioned the search spaces into smaller ones according to the filters' floating passband gain. The GA or the tree search deal with only a subspace initially. This increases the chances to find feasible solutions significantly, and also is the reason of better results compared

with other algorithms. On the other hand, comparing the improved GA proposed in Chapter 3 with the polynomial time tree search method proposed in Chapter 4, the later one is not only more efficient but also can generate designs with lower hardware cost. The shortcoming of the tree search method however, is that it cannot be used to design FIR filters in cascaded form as the improved GA did.

The second scenario considers the design of multi-stage FIR filters. The GA in Chapter 3 and a MILP based algorithm in Chapter 5 can be used, for the design of multi-stage FIR filters. The improved GA is more computational efficient and thus is more suitable for the design long filters in multi-stage, while the MILP based algorithm has good performance only when the filter order is not much larger than 70. The major difference between these two algorithm is how to determine the subfilters. The GA employed a factorization technique to automatically determine the order and EWL of subfilters, while for the MILP based algorithm, each subfilter is restricted to one of 4 SPT coefficient filters with orders up to 2, except the last subfilter. The GA is more computational efficient and thus are more suitable for the design of long filter while the MILP based algorithm is more effective in terms of hardware cost but with higher computational complexity and thus are more suitable for the design of short or moderate orders filters.

While the above 3 algorithms all aim to minimize the hardware cost, the rest 2 algorithms proposed in Chapters 6 and 7 consider the scenario to minimize the power consumption of FIR filters. In the algorithm in Chapter 6, the adder cost is minimized and meanwhile the coefficient synthesis is constrained to its MinAD. Compared to the algorithms without adder depth constraints, the proposed one generates the designs with a few more number of adders but with much shorter average logic depth, and thus the overall power consumption is reduced. To estimate the power consumption more accurately, in the algorithm proposed in Chapter 7, a word-level power model is proposed and serves as the criteria for the optimization of filters. Besides the parameterization

of by the adder depth, the power model considers both dynamic and static powers. In such a way, the proposed algorithm may generate different designs for different operating frequencies to achieve the minimum power consumption for the respective frequencies. Due to the more accurate measurement of power consumption, the algorithms proposed in Chapter 7 can generate designs with even lower power than that of the algorithm proposed in Chapter 6. However, it requires the detailed information of IC technology to build the power model. If the information of IC technology is not known prior, the algorithm proposed in Chapter 6 is a good choice for the design of low power consumption FIR filters.

8.2 Future Works

This thesis proposes algorithms for the design of low hardware cost and low power consumption FIR filters. However, all existing algorithms optimize the filter at word-level, i.e., the width of adders is ignored. To have a more accurate measurement of the hardware cost and power consumption, the width of adder should be taken into consideration. Therefore, in the future, efforts may be devoted to the development of algorithms aiming to minimize the hardware cost and power consumption at bit-level.

In this thesis, algorithms for simple cascade structures are also proposed. Another promising low hardware cost structure is the frequency response masking (FRM) technique. However, no efficient multiplierless optimization algorithm exists because the network involved in the FRM is more complicated than the cascade structure. It is a more challenging problem for the future work.

Bibliography

- [1] S. S. Demirsoy, A. Dempster, and I.Kale, “Transition analysis on FPGA for multiplier-block based FIR filter structures,” in *Proc. The 7th IEEE International Conference on Electronics, Circuits and Systems*, Lebanon, Dec. 2000, pp. 862–865.
- [2] W. B. Ye and Y. J. Yu, “Switching activity analysis and power estimation for multiple constant multiplier block of FIR filters,” in *Proc. IEEE Int. Symp. Circuits Syst.*, Rio de Janeiro, Brazil, May 2011, pp. 145–148.
- [3] D. Shi and Y. J. Yu, “Design of linear phase FIR filters with high probability of achieving minimum number of adders,” *IEEE Trans. Circuits Syst. I*, vol. 58, pp. 126–136, Jan. 2011.
- [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] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice-Hall, 2002.

- [8] N. H. E. Weste and K. Eshraghian, *Principles of CMOS VLSI Circuit*. Wiley India Pvt. Limited, 1988.
- [9] 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, Pheonix, AZ, May 2002, pp. 1–4.
- [10] T. Saramäki, "Finite impluse 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.
- [11] S. K. Mitra, *Digital Signal Processing, A Computer-Based Approach*. McGraw-Hill, 1998, ch. 10.
- [12] Y. C. Lim and A. G. Constantinides, "Linear phase FIR digital filter without multipliers," in *Proc. IEEE ISCAS'79*, 1979, pp. 185–189.
- [13] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete power-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 583–591, June 1983.
- [14] G. W. Reitwiesner, "Binary arithmetic," in *Advances in Comoputers*, 1960, vol. 1, pp. 231–308.
- [15] 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.
- [16] 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.

- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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.

- [25] C. Y. Yao, "A study of SPT-term distribution of csd numbers and its application for designing fixed-point linear phase FIR filter," in *Proc. IEEE ISCAS'01*, vol. II, 2001, pp. 301–304.
- [26] Y. C. Lim and Y. J. YU, "A successive reoptimization approach for the design of discrete coefficient perfect reconstruction lattice filter bank," in *Proc. IEEE Int. Symp. Circuits Syst.*, Geneva, Switzerland, May 2000, pp. 69–72.
- [27] 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 Processing*, vol. 50, pp. 1935–1941, Aug. 2002.
- [28] 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.
- [29] Y. C. Lim, Y. Sun, and Y. J. Yu, "Design of discrete-coefficient fir filters on loosely connected parallel machines," *IEEE Trans. Signal Processing*, vol. 50, pp. 1409–1416, June 2002.
- [30] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proceedings, Part G: Circuits, Devices and Systems*, vol. 138, pp. 401–412, June 1991.
- [31] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Circuits and Syst. II, Analog Digit. Signal Process*, vol. 43, pp. 677–688, Oct. 1996.
- [32] 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. Computer-Aided Design*, vol. 15, pp. 151–161, Feb. 1996.
- [33] I.-C. Park and H.-J. Kang, “Digital filter synthesis based on an algorithm to generate all minimal signed digit representation,” *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 1525–1529, Dec. 2002.
- [34] A. G. Dempster and M. D. Macleod, “Use of minimum-adder multiplier blocks in FIR digital filters,” *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 569–577, Sept. 1995.
- [35] 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.
- [36] R. Paško, P. Schaumont, V. Derudder, *et al.*, “A new algorithm for elimination of common subexpressions,” *IEEE Trans. Computer-Aided Design*, vol. 18, pp. 58–68, Jan. 1999.
- [37] O. Gustafsson and L. Wanhammar, “ILP modeling of the common subexpression sharing problem,” in *Proc. IEEE ICECS’02*, Dubrovnic, Croatia, 2002, pp. 1171–1174.
- [38] A. G. Dempster, S. S. Demirsoy, and I. Kale, “Designing multiplier blocks with low logic depth,” in *Proc. IEEE Int. Symp. Circuits Syst.*, Scottsdale, AZ, May 2002, pp. 773–776.
- [39] A. G. Dempster and M. D. Macleod, “Generation of signed-digit representations for integer multiplication,” *IEEE Signal Processing Lett.*, vol. 11, pp. 663–665, Aug. 2004.

- [40] 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.
- [41] M. D. Macleod and A. G. Dempster, “Multiplierless FIR filter design algorithms,” *IEEE Signal Processing Lett.*, vol. 12, pp. 186–189, Mar. 2005.
- [42] Y. Wang and K. Roy, “CSDC: A new complexity reduction technique for multiplierless implementation of digital FIR filters,” *IEEE Trans. Circuits Syst. I*, vol. 52, pp. 1845–1853, Sept. 2005.
- [43] 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.
- [44] F. Xu, C.-H. Chang, and C.-C. Jong, “A new integrated approach to the design of low-complexity FIR filters,” in *Proc. IEEE ISCAS’06*, Kos, Greece, 2006, pp. 601–604.
- [45] F. Xu, C.-H. Chang, and C.-C. Jong, “Design of low-complexity FIR filters based on signed-powers-of-two coefficients with reusable common subexpressions,” *IEEE Trans. Computer-Aided Design*.
- [46] Y. Voronenko and M. Püschel, “Multiplierless multiple constant multiplication,” *ACM Trans. Algorithms*, vol. 3, pp. 1–38, May 2007.
- [47] 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.

- [48] 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.
- [49] H.-J. Kang and I.-C. Park, “FIR filter synthesis algorithms for minimizing the delay and the number of adders,” *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 770–777, Aug. 2001.
- [50] 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.
- [51] Y. J. Yu and Y. C. Lim, “Optimization of linear phase FIR filters in dynamically expanding subexpression space,” *Circuits, Systems, and Signal Processing*, vol. 29, pp. 65–80, June 2009.
- [52] 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.
- [53] A. Shahein, Q. Zhang, and Y. Manoli, “A novel hybrid monotonic local search algorithm for FIR filter coefficients optimization,” *IEEE Trans. Circuits and Systems*, vol. 59, no. 3, pp. 616–627, Mar. 2012.
- [54] B. Y. Kong and I.-C. Park, “FIR filter synthesis based on interleaved processing of coefficient generation and multiplier-block synthesis,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 8, pp. 1169–1179, Aug. 2012.

- [55] M. Aktan, A. Yurdakul, and G. Dnda, "An algorithm for the design of low-power hardware-efficient FIR filters,," *IEEE Trans. Circuits and Systems*, vol. 55, pp. 1536–1545, July 2008.
- [56] J. J. Chen and C. H. Chang, "High-level synthesis algorithm for the design of reconfigurable constant multiplier," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 12, pp. 1844–1856, December 2009.
- [57] L. R. R. O. Herrmann and D. S. K. Chan, "Practical design rules for optimum finite response lowpass digital filters,," *Bell Syst. Tech. J*, vol. 52, pp. 769–799, 1974.
- [58] J. W. Adams and J. A. N. Willson, "A new approach to FIR digital filters with fewer multipliers and reduced sensitivity," *IEEE Trans Circuits Systems*, vol. 30, pp. 277–283, May 1983.
- [59] L. Cen, "A hybrid genetic algorithm for the design of FIR filters with SPoT coefficients," *Signal Processing*, vol. 87, pp. 528–540, 2007.
- [60] L. Aksoy, E. Gunes, and P. Flores, "An exact breadth-first search algorithm for the multiple constant multiplications problem," in *Proc. IEEE NORCHIP*, Tallinn, Nov. 2008, pp. 41–46.
- [61] 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.
- [62] N. H. E. Weste and K. Eshraghian, *Digital Communications*. Addison-Wesley, 1993.
- [63] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filter," *IEEE Trans. Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, pp. 569–577, Sept. 1995.

- [64] M. Faust and C. H. Chang, “Minimal logic depth adder tree optimization for multiple constant multiplication,” in *Proc. IEEE Int. Symp. Circuits Syst.*, Paris, May 2010, pp. 457–460.
- [65] W. B. Ye and Y. J. Yu, “An algorithm for the design of low power linear phase FIR filters,” in *Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, Macau, China, oct. 2011, pp. 29–32.
- [66] S. S. Demirsoy, A. G. Dempster, and I. Kale, “Power analysis of multiplier blocks,” in *Proc. IEEE International Symposium on Circuits and Systems*, Phoenix, Arizona, May 2002, pp. 297–300.
- [67] K. Johansson, O. Gustafsson, and L. S. DeBrunner, “Estimation of the switching activity in shift-and-add based computations,” in *Proc. IEEE International Symposium on Circuits and Systems*, Taipei, May 2009, pp. 3054–3057.
- [68] K. Johansson, O. Gustafsson, and L. Wanhammar, “Power estimation ripple-carry adders with correlated input data,” in *Proc. IEEE Int. Workshop Power Timing Modeling ,Optimizaiton Simulation*, Santorini, Greece, Sept. 2004, pp. 662–674.
- [69] D. Shi and Y. J. Yu, “Design of discrete-valued linear phase FIR filters in cascade form,” *IEEE Trans. Circuits Syst. I*, vol. 58, pp. 1627–1636, 2011.
- [70] J. H. Holland, *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.
- [71] P. Gentili, F. Piazza, and A. Uncini, “Efficient genetic algorithm design for power-of-two FIR filters,” in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, Detroit, Michigan, USA, May 1995, pp. 9–12.

- [72] L. Cen and Y. Lian, "A modified micro-genetic algorithm for the design of multiplierless digital FIR filters," in *Proc. IEEE Region 10 Conference*, Chiang Mai, Thailand, Nov. 2004, pp. 52–55.
- [73] D. E. Goldberg, *Genetic algorithm in search and optimization*. Addison-Wesley, 1989.
- [74] S. U. Ahmad and A. Antoniou, "Cascade-form multiplierless FIR filter design using orthogonal genetic algorithm," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol.*, Aug 2006, pp. 932–937.
- [75] Y. C. Lim and B. Liu, "Design of cascade form FIR filters with discrete valued coefficients," *IEEE Transactions on Signal Processing*, vol. 36, pp. 1735–1939, Nov. 1988.
- [76] Y. C. Lim, R. Yang, and B. Liu, "The design of cascaded FIR filters," in *Proc. IEEE Int. Symp. Circuits Syst*, May 1996, pp. 181–184.
- [77] A. Yurdakul and G. Diznda, "Fast and efficient algorithm for the multiplierless realisation of linear DSP transforms," *IEE Proc. Circuits Devices Syst.*, vol. 149, pp. 205–211, Dec. 2002.
- [78] N. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica*, vol. 4, pp. 373–395, 1984.
- [79] K. Johansson, "Low power and low complexity shift-and-add based computations," Ph.D. dissertation, Linkping University, 2008.
- [80] J. A. Butts and G. S. Sohi, "A static power model for architects," in *Proc. Annual International Symposium on Microarchitecture*, Monterey, CA, Dec. 2000, pp. 69–72.

- [81] J. J. Chen, C. H. Chang, and H. H. Qian, “New power index model for switching power analysis from adder graph of FIR filter,” in *Proc. IEEE International Symposium on Circuits and Systems*, Taipei, May 2009, pp. 2197–2200.