Algorithms for synthesis and optimization of multiplierless FIR filters

Xu, Fei

2007

Xu, F. (2007). Algorithms for synthesis and optimization of multiplierless FIR filters. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/46954>

<https://doi.org/10.32657/10356/46954>

Nanyang Technological University

Downloaded on 26 Apr 2023 13:44:37 SGT

AT7815T 248

5X

2/8/07

Algorithms for synthesis and Optimization of Multiplierless FIR Filters

Xu Fei

School of Electrical & Electronic Engineering

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

TK
7872
.F5
X8
2007

Acknowledgements

I wish to thank my supervisor, Associate Professor Jong Ching Chuen, for his intellectual guidance, constant attention, unceasing encouragement and strict demands through this research program. In particular, I wish to thank him for teaching me some of the philosophical aspects of research and some intangible skills which I consider are the most invaluable knowledge I have acquired in this research program.

I would like to thank my co-supervisor, Associate Professor Chang Chip Hong for his efficient supervision, active academic motivation and easy-going communication throughout the research program. Especially, I wish to thank him for discussions and collaboration in the theoretical work in this research program.

I would like to thank Ms Ye Zhi, Mr Sharma Udit, Mr Satzoda Ravi Kumar and Mr Chen Jiajia for their friendship, help, discussions and suggestions pertaining to this research work. I would like to thank Ms Gong Yuanyuan and Mr Ru Hong for their tedious but indispensable work to speed up the research schedule.

To the staff and other students in Integrated Systems Research Laboratory and Center for High Performance Embedded Systems, I wish to convey my appreciation to all of them for their kind and friendly assistance.

I would like to express my gratitude to my parents for their consistent support and encouragement during all periods of my study.

Finally, I would like to acknowledge School of Electrical & Electronic Engineering, Nanyang Technological University for providing the research facility and the research scholarship.

Contents

ACKNOWLEDGEMENTS.....I

CONTENTS.....II

SUMMARY..... V

LIST OF FIGURES VIII

LIST OF TABLES X

LIST OF ABBREVIATIONS..... XI

CHAPTER 1 INTRODUCTION 1

1.1 Motivation..... 1

1.2 Objectives 4

1.3 Major Contributions of the Thesis5

1.4 Organization of the Thesis8

CHAPTER 2 LITERATURE REVIEW11

2.1 Overview of Digital FIR Filter.....11

2.1.1 Direct Form Structure.....12

2.1.2 Transposed Direct Form Structure14

2.1.3 Linear Phase Direct Form Structure15

2.2 Number Representations in Digital FIR Filters17

2.2.1 Two’s Complement Representation17

2.2.2 Signed Digit Representation.....18

2.2.3 Canonical Signed Digit and Minimal Signed Digit.....18

2.2.4 Comparisons of Different Number Representations.....21

2.3 Filter Coefficient Synthesis Algorithms22

2.3.1 Problem formulation.....22

2.3.2 Filter Synthesis Approaches24

2.4 Multiple Constant Multiplication Problem.....28

2.4.1 Common Subexpression and Partial Sum.....29

2.4.2 Common Subexpressions Elimination Algorithms (CSE).....31

2.4.3	Graph-Dependence (GD) Algorithms	35
2.5	Summary.....	38
CHAPTER 3 HAMMING WEIGHT PYRAMID		40
3.1	Introduction.....	40
3.2	Generation and Development of Hamming Weight Pyramid	41
3.3	Properties of Hamming Weight Pyramid	44
3.4	Applications of HWP	51
3.4.1	Canonical Signed Digit Conversion algorithm.....	51
3.4.2	Modified RAG-n Algorithm.....	54
3.5	Summary.....	56
CHAPTER 4 FILTER SYNTHESIS ALGORITHM		58
4.1	Introduction.....	58
4.2	Coefficient Synthesis with Common Subexpressions.....	60
4.3	The Proposed Algorithm	63
4.3.1	CSPT Coefficient Allocation.....	65
4.3.2	Quantization Compensation	68
4.3.3	An Example.....	72
4.4	Experimental Results.....	75
4.5	Summary.....	84
CHAPTER 5 CONTENTION RESOLUTION ALGORITHM FOR WEIGHT-2 HORIZONTAL SUBEXPRESSION		85
5.1	Introduction.....	85
5.2	Admissibility Graph.....	87
5.2.1	Notations and Expressions	88
5.2.2	Example.....	90
5.3	Contention Resolution	91
5.4	Contention Resolution Algorithm for Weight-2 Horizontal Subexpressions.....	95
5.4.1	Problem Statement	96
5.4.2	Proposed Algorithm	97
5.4.3	Example.....	100
5.5	Experimental Results.....	104
5.6	Summary.....	109

CHAPTER 6	110
6.1 Introduction.....	110
6.2 Common Subexpressions.....	113
6.3 Contention Resolutions for higher weights.....	119
6.4 Contention Resolution Algorithms	122
6.4.1 General description and classifications of CRAs	122
6.4.2 Contention Resolution Algorithms for weight-X Subexpressions..	124
6.4.3 Contention Resolution Algorithms for weight-N subexpressions	131
6.5 Simulations and Experimental Results	133
6.6 Summary.....	143
CHAPTER 7	145
7.1 Conclusions.....	145
7.2 Suggestions for Further Work	149
7.2.1 Applications of MSD and other SD representations in filter design.....	149
7.2.2 Minimization on switching activity and adder complexity	151
7.2.3 Development of EDA framework.....	152
AUTHOR’S PUBLICATION.....	154
BIBLIOGRAPHY	157

Summary

The use of transformations in high-level synthesis of Very Large Scale Integration (VLSI) circuits has led to the development of powerful optimization methods for many compute intensive digital signal processing problems. One such class of applications is digital filters which are modeled as a generic multiple constant multiplication (MCM) problem by decomposing the computational intensive multiplier in their implementation into shifters and simpler arithmetic operators. This thesis presents the study of synthesis and optimization of multiplierless finite impulse response (FIR) digital filters. In this research, work has been done to scrutinize the characteristics of Canonical Signed Digits (CSD) number system, and to investigate the feasibility of sharing crucial common subexpressions in filter synthesis algorithm and new and efficient algorithms for improving the logic complexity and logic depth of FIR filter implementation. The ultimate goal is to establish a set of technology independent design automation algorithms that allows a quick path to the application specific integrated circuit (ASIC) cores to meet stringent filtering specifications.

The CSD representation, a unique number system with the minimum numbers of Signed-Power-of-Two (SPT) terms widely used in digital filter synthesis and optimization, is explored. In this work, a new Hamming Weight Pyramid (HWP) is proposed to succinctly compress the information of the distribution of the hamming weights of CSD representation in a visually appealing manner for analysis and synthesis. Many interesting properties of CSD are uncovered in this regularly structured HWP. These properties are exploited to develop a novel and elegant algorithm for the direct conversion of decimal number to CSD representation. It is shown that the HWP can also be employed to overcome the limit imposed on the word length of the coefficients for the n -Dimensional Reduced Adder Graph (RAG- n) algorithm and filter coefficient synthesis.

The prevailing algorithms for the design of FIR filter have treated the coefficient synthesis from filter specification and circuit optimization with resource sharing as two correlated but independent problems. In this thesis, a new efficient algorithm is proposed

for the design of low-complexity FIR filters with resource sharing. The proposed algorithm considers the sharing of two most commonly used subexpressions, 101 and $10\bar{1}$ right from the inception of the magnitude response specification. The HWP has been recast to a common subexpression related Hamming Weight Pyramid (CSHWP) so that the original problem statement based on the minimization of SPT terms can be reformulated to account for the physical adder cost oriented Common Signed-Power-of-Two (CSPT) terms. The proposed algorithm first generates an initial CSD coefficient set in the vicinity of the rounded coefficients and then iteratively compensates for the quantization errors of the initial set until it converges to the desired amplitude response specifications. It has been demonstrated that the proposed algorithm is capable of synthesizing filters with reduced hardware cost by minimizing the number of CSPT terms as opposed to the number of SPT terms in existing filter synthesis algorithms.

For solutions that can be extended to the more general form of MCM problems, various redundant computations present implicitly in the fixed coefficients are identified and eliminated. In this perspective, the notion of contention resolution is introduced along with the proposal of a succinct admissibility graph labeling approach to adaptively eliminate the redundant adders in the multiplierless implementation of MCM. The prowess of the idea of contention resolution is its ability to revert a precedential decision of suboptimal common subexpressions through a localized cost function evaluation when there is a conflict between the selections of two competitive subexpressions. Several derivatives of contention resolution algorithms have been proposed to complete a set of versatile design methodologies for optimizing the sharing of both intra- and inter-coefficient common subexpressions, of any hamming weight and in any legitimate composition of horizontal, vertical and oblique common subexpressions. As the algorithms expand the common subexpressions to higher-weight only when there is cost saving, the logic depth can be controlled by constraining the weights of the common subexpressions in the admissibility graph. The fundamental tenet of good heuristic is honored in the proposed algorithms as significant improvement in the solution quality (in terms of logic complexity and logic depth) is attained with increased problem size but the computational time remains well bounded.

Summary

The thesis presents the details of the newly constructed HWP, its properties and its applications. It describes the filter synthesis techniques which incorporate the common subexpression sharing and the contention resolution algorithms which optimize the fixed coefficients. The details of the algorithms, the experimental testing results, their comparisons with the published results and the analysis and discussions of the algorithms and results are included.

List of Figures

Figure 2.1	Direct form FIR filter structure	13
Figure 2.2	Transposed direct form FIR structure.....	14
Figure 2.3	FIR filter with symmetrical coefficients.....	16
Figure 2.4	Multiple Constant Multiplication Block in FIR Filter.....	28
Figure 2.5	CSE Implementations of coefficient 10100101.....	29
Figure 2.6	GD implementation of coefficient 165 (10100101)	30
Figure 3.1	HWP and its corresponding decimal and CSD pyramids.....	43
Figure 3.2	Odd Hamming Weight Pyramid (OHWP).....	44
Figure 3.3	Generation of the core and wings of row 4	47
Figure 3.4	Trace of shifting CSD numbers	48
Figure 3.5	Duality property of OHWP.....	49
Figure 3.6	An add-one HWP generated from OHWP	49
Figure 3.7	Ancestor-descendents relationships in HWP.....	50
Figure 3.8	Ternary tree of CSD numbers.....	50
Figure 3.9	Pseudo code of new conversion algorithm	51
Figure 3.10	Traces of decimal to CSD number conversion from HWP	52
Figure 3.11	CSD Tree	53
Figure 3.12	Pseudo code of the MRAG algorithm	55
Figure 4.1	Common subexpression eliminations.....	60
Figure 4.2	CS-HWP	65
Figure 4.3	Pseudocode of two-stage filter coefficient synthesis algorithm	71
Figure 4.4	Comparison of magnitude responses of FIR filter designs for Example 4.3.....	78
Figure 4.5	Comparisons of frequency responses of Example 4.4.....	79
Figure 4.6	Circuit implementation with common subexpressions, 101 and $10\bar{1}$	82
Figure 4.7	Circuit implementation with higher weight common subexpressions.....	82

List of Figures

Figure 5.1	A 5-vertex Admissibility Graph.	91
Figure 5.2	A scenario of contention resolution.....	92
Figure 5.3	Another scenario of contention resolution.....	93
Figure 5.4	Complicated contention.....	94
Figure 5.5	Algorithm CRAH-2 for weight-2 horizontal common subexpressions.....	98
Figure 5.6	Labeling of order 1 subexpressions.....	101
Figure 5.7	Labeling of order 2 subexpressions.....	101
Figure 5.8	Labeling of order 3 subexpressions.....	101
Figure 5.9	Labeling of order 4 subexpressions.....	102
Figure 5.10	Results of the contention resolution	102
Figure 5.11	AG solution graph of CRAH-2 for Example 2.....	103
Figure 5.12	Circuit implementation.....	104
Figure 5.13	Comparison of LO between CRAH-2 and NRSCSE	107
Figure 5.14	Comparison of average adder cost and computation time of CSD, NRSCSE, Paško, and CRAH-2	108
Figure 6.1	Notations for different types of subexpressions	113
Figure 6.2	Composite common subexpression eliminations	115
Figure 6.3	Higher weight common subexpression eliminations.....	118
Figure 6.4	Generation of weight-4 precedence path from weight-3 precedence paths.....	119
Figure 6.5	Contention Resolution in path generations.....	120
Figure 6.6	Pseudocode of weight-X CRA	125
Figure 6.7	AG after labeling of order 1 subexpressions	128
Figure 6.8	AG after labeling of order 2 subexpressions	128
Figure 6.9	Final AG of Example 6.2.....	129
Figure 6.10	Solution by CRA	129
Figure 6.11	Mapping of solution AG to adder tree.....	130
Figure 6.12	Adder tree topologies for a weight-7 coefficient.....	133
Figure 6.13	Comparison of average adder cost.....	138
Figure 6.14	Distribution of the heights of ICS.....	141
Figure 6.15	Comparison on computation times.....	143

List of Tables

Table 2.1	CSD Generation with lookup table [HWA79]	19
Table 2.2	Comparisons of Number Representations	21
Table 3.1	$N(i)$ and $S(i)$ of HWP	46
Table 3.2	Memory organization of HWP	48
Table 4.1	Adaptive neighborhood for quantization error compensation	69
Table 4.2	Summary of different filter designs for Example 4.3	76
Table 4.3	Comparison of adder costs for Example 4.3.....	77
Table 4.4	Summary of different filter designs for Example 4.4.....	80
Table 4.5	Comparison with merge-search algorithms for CSA-based filters.....	81
Table 4.6	Experimental results for other filter specifications.....	84
Table 5.1	Sorted common ubexpression list, S	100
Table 5.2	Simulation results on FIR filters.....	105
Table 6.1	Classification of CRAs	123
Table 6.2	Mapping between logic depth and path weight	131
Table 6.3	Experimental results comparison for CRAH.....	135
Table 6.4	Adder cost comparison for CRA	136
Table 6.5	Experimental results of CRAH-N with varying logic depth constraints	139
Table 6.6	Statistic Percentages of VCS, OCS and HCS in CRA-2's solutions.....	140

List of Abbreviations

AG	Admissibility Graph
ASIC	Application-Specific Integrated Circuit
CDP	Critical Descend Phenomenon
CP	Candidate Pool
CPA	Carry-Propagation Adders
CRA	Contention Resolution Algorithm
CRAH-2	Contention Resolution Algorithm for Weight-2 Horizontal Subexpressions
CSA	Carry-Save Adders
CSD	Canonical Signed Digit
CSDT	CSD Tree
CSE	Common Subexpression Elimination
CS-HWP	Common Subexpression based Hamming Weight Pyramid
CSPT	Common Signed-Power-of-Two
DSP	Digital Signal Processing
FIR	Finite Impulse Response
GD	Graph-Dependence
HWP	Hamming Weight Pyramid
IC	Integrated Circuit
ICS	Inter-Coefficient Common Subexpressions
IIR	Infinite Impulse response
ILP	Integer Linear Programming
ITM	Iterative Matching
LD	Logic Depth
LO	Logic Operators
LPFIR	Linear Phase Finite Impulse Response Filter
LSD	Least Significant Digit
MAG	Minimum Adder Graph
MCM	Multiple Constant Multiplications
MILP	Mixed Integer Linear Programming

List of Abbreviations

MRAG	Modified n -Dimensional Reduced Adder Graph
MSB	Most Significant Bit
MSD	Minimal Signed Digit
NAS	Number of Adders Saved
NPRM	Normalized Peak Ripple Magnitude
NRSCSE	Nonrecursive Signed Common Subexpression Elimination
OHWP	Odd HWP
POG	Primitive Operator Graph
RAG- n	n -Dimensional Reduced Adder Graph
SD	Signed Digit
SDR	Software Defined Radio
SISO	Single Input Single Output
SoC	System-On-Chip
SPT	Signed-Power-of-Two
VLSI	Very large scale integration

Chapter 1

Introduction

1.1 Motivation

The term digital filter refers to the computational process or algorithm by which a sampled signal or a sequence of numbers (acting as the input) is transformed into a second sequence of numbers as the output signal [KAI66]. Unlike their analog counterparts, digital filters are based on the simple operations of multiplication and addition. Therefore, it outperforms analog filter as it is not susceptible to various effects induced by device nonlinearities, such as DC offset, phase distortions, noises and interferences, reference voltage stability, etc. The processing of discrete signals by linear filters or weighting sequences had its origin in the early 1600's [KAI66]. Mathematicians and astronomers are the pioneers in the development of digital filters, which were used as aids to construct sophisticated mathematical tables and for studying the orbits of heavenly bodies. The proliferation of signal processing techniques and the advent of digital computer further boosted the evolution of digital filters during last century. Due to the rapid expansion of computation capability, digital signal processor technology has become affordable and versatile, making the presence of digital filter in consumer electronics ubiquitous [KAI68, SUZ85, LEI90, LI05].

Digital filters are an important class of linear time-invariant systems. The relationship between the input and output signals of a digital filter can be completely characterized by their unit-sample (or impulse) response and analyzed by convolution of two discrete sequences. Based on the unit-sample response, digital filters can be classified as either recursive or nonrecursive type [KAI66]. The nonrecursive filter has a finite memory (since its unit-sample response is of finite duration), hence the name of Finite Impulse Response

(FIR) filter. Due to its versatility, stability and simplicity over the Infinite Impulse response (IIR) digital filter, FIR filter plays a vital role in modern communication systems [DAV02, FAR00, LI96, PEL04]. The computationally intensity of the applications trigger the implementations of digital filters into application-specific integrated circuit (ASIC). Comparing with analog integrated circuit, digital integrated circuit (IC) technology is more robust with respect to temperature and process variations. The accuracy in digital representation and the impulse response can be controlled better by changing the wordlengths of the signal and the coefficients. One of the popular application of ASIC is the radio transceiver in wireless communications, and the emerging software defined radio (SDR) ensues the digital implementation of the RF and analog front-end signal processing functions in order to eliminate the sophisticated compensation methods needed to provide the repeatability and good signal-to-noise ratio, and to exploit the flexibility and robustness of post-processing in digital domain. Filtering and waveform shaping functions are the main performance stumbling blocks immediately after the analog-to-digital watershed. To achieve an all-digital or a digital-dominant system architecture, the associated kernel functions like convolution, correlation and linear transformations must be processed at very high bit rates, as exemplified by the bandlimited IF signal processing at sampling rate of 50-100 MHz [JAI91,YOS90]. The general trend towards increasing sampling rates and resolutions of analog-to-digital converters has driven the design constraints of digital realization of many complex linear-time invariant systems beyond the traditional consideration of reduced silicon cost. High throughput rate and low power dissipation at reduced design turnaround time are among the desiderata.

One approach to the design of high throughput rate digital filters to meet the stringent area and power constraints is to capitalize on high level design methodologies for ASIC implementation. The use of transformations in high-level synthesis of very large scale integration (VLSI) circuits has led to the development of powerful optimization methods for many compute intensive digital signal processing problems. One general problem for which high-level transformation has been applied with phenomenal success is the multiple constant multiplications (MCM) [POT96], which is the core function of digit

FIR filters. As the name implies, MCM problem involves multiple multiplications of a variable by different constants. As multipliers are generally agreed to be a power hungry device and occupy large silicon area, a clear-cut solution to the MCM problem is to replace the expensive multiplication operations by simpler additions and hardwired shifters, which are almost free in terms of silicon area compared with the use of logic cells [BUL91, DEM95, HAR96, POT96, MAR02, YAO04]. While the multiplications by constants appear only in small fraction of general computation benchmarks, its frequency of encounters has been significant in application specific computations. Many complex linear transformations, convolution and correlation functions can be modeled and solved as an MCM problem. Therefore, good optimization methods developed to automate the design of FIR filters can also be exploited by high-level synthesis tools targeted at a wider spectrum of kernel functions in SDR and real-time multimedia signal processing applications.

There are two major challenges in the design of low complexity FIR filters, which are coefficient synthesis and hardware architectural optimization. The coefficient synthesis aims at mapping the infinite precision filter coefficients (in the form of transfer functions or filter response specifications) into finite length coefficients for optimal implementation within the tolerance of magnitude response specifications. This mapping process introduces quantization errors, and the mapping is not a simple ‘round-off’ task in view of the non uniform contribution of quantization errors from each filter tap. The search for an optimal coefficient set to meet the specification involves complicated iterative search processes, often with integer linear programming of formidable time and space complexity, even under the simplistic optimality criterion of signed-digit coefficients with minimal number of nonzero digits. The hardware architectural level or gate level optimisation is to obtain an optimized filter circuit with minimum silicon area, latency and power consumption. The architectural optimization usually involves the exploration of redundancy in the multiplier block of filter structure to eliminate superfluous computations from a given set of filter coefficients. The optimization objectives are typically driven by the number of arithmetic operators used. For high-speed and low cost consideration, there has been an increased interest to direct the prime optimization goal

towards reducing the logic depth as well. In general, prevailing design automation algorithms [AIT93, AVI61, BEN92, CEM93, YAO01] treat the filter synthesis and architectural optimisation as two correlated but independent research problems. By ignoring the redundancy present in the filter coefficients during the synthesis process, the simplistic assumptions of minimal signed digit representation and uniformly weighted coefficient sensitivity to the final response have forgone the immense potential to achieve better trade-off between the tolerable quantization error and overall performance optimality in VLSI implementation. As canonical signed digit (CSD) representation [REI60] has been used extensively in both processes by many algorithms [CHE99, LEE93, LIM83, LIM90, ROS04, SAM89, SHA91] to reduce the enumeration complexity of the associated search problems, there is also great incentive to further study the properties of CSD numbers and their transformations.

1.2 Objectives

The objective of this research work is to develop a framework of high-level optimization methodologies for the core operation encountered in digital signal processing applications. Digital FIR filter is used as a vehicle to study the VLSI efficient solutions to the general MCM problem. The methodologies and solutions are to be developed with the emphasis to exploit VLSI design at higher level of abstraction than gate and circuit levels. This research will look into the novel formulation of data structure to apply computational transformations to MCM problem. The goal of the transformations is to realize a given set of filter coefficients with minimized VLSI cost metrics such as delay, area and power. The transformation and reduction approaches will explore the potential and flexibility of the re-substitution and re-allocation of operators to address the implementation constraints while maintaining the fundamental saliency of the filter response specifications. With the proposed design automation algorithms, device-dependent parameters can be transformed to logic complexity and logic depth constraints so that changes in filter specifications can be efficiently addressed for the targeted technology of implementation. Being essentially a core generation technology based on the projected constraints of the applications and the base model of the primitive resources (basic

arithmetic circuit from the cell library) available for reuse, the developed algorithms and techniques shall be applicable to other mathematical functions that can be modelled or modified into a convolution-like operation, which involves several different constant multiplications with the same variable.

The following targets have been set towards fulfilling the theme of this research:

- (1) To investigate properties of CSD representation and its transformation to and from other representations in order to exploit them in accelerating the computations involved in FIR filter synthesis and optimization algorithms.
- (2) To define a new cost metric so that an integrated approach can be developed for low-complexity FIR filters by considering architectural optimization opportunity in the early phase of coefficient synthesis to meet the frequency response specification.
- (3) To overcome the local minimal problem due to order-dependent decisions in statistical based common subexpression elimination algorithms, from which new algorithms with emphasis on logic depth constrained optimization or composite area-time constraints are to be derived.
- (4) To explore different types of subexpression sharing in MCM problem in order to extend the versatility of the developed methodologies to other applications.

1.3 Major Contributions of the Thesis

The work reported in this thesis aims to provide better solutions in digital filter design, and the attempt has led to four major contributions and original results:

(1) Proposal of Hamming Weight Pyramid (HWP)

In the process of the study of popular number systems used in digital filter design, some interesting characteristics with regards to the hamming weight distribution of the CSD numbers have been observed, from which a Hamming Weight Pyramid (HWP) is defined. The proposed HWP exhibits a regular structure and provides a new insight into the formation of the CSD representation of integers. Many useful and attractive properties have been extracted from the HWP. A reduced HWP is also proposed. It can be

constructed easily from a reduced sized lookup table. From the HWP, a new method to convert decimal number to its CSD format is derived. Direct conversion from decimal number to CSD form without an intermediate redundant signed-digit number conversion step makes it attractive as preprocessing step of many design automation algorithms can be benefited directly from the substitution of integer operations by their CSD forms. Its computation complexity is comparable to that of the conversion from decimal number to binary number. It is shown that this new conversion process from decimal number to CSD representation has a straightforward view in a unique CSD Tree (CSDT). For architectural optimization, HWP is employed as a search tool to approximate the adder distance in the n -Dimensional Reduced Adder Graph (RAG- n) [DEM95] to overcome the limit in the wordlength of the coefficients. The modified algorithm with this devised cost metric is called the MRAG [XU04a, XU05c]. For filter coefficient synthesis problem, the proposed HWP has been used to narrow down the search range for good CSD coefficients with minimal hamming weight near the rounded coefficients within permissible bound of frequency response error.

(2) A new integrated approach to the synthesis of FIR filter coefficients

A new efficient filter synthesis algorithm is proposed for the design of low-complexity FIR filters with resource sharing in transposed direct form filter structure. The proposed algorithm considers the sharing of two most commonly used subexpressions from the inception of the magnitude response specification. The original problem statement based on the minimization of Signed-Power-of-Two (SPT) terms has been reformulated to account for the physical adder cost oriented Common Signed-Power-of-Two (CSPT) terms. The minimization of CSPT terms considered in the proposed algorithm addresses the simultaneous optimization of the reusability of adders for the two most frequently used common subexpressions as well as the minimization of adders needed for the spare SPT terms. The algorithm uses a Common Subexpression based Hamming Weight Pyramid (CS-HWP) to seek for low cost candidate coefficients with preferential consideration of shared common subexpressions. Experimental results on widely used benchmark design examples demonstrate that the algorithm is capable of synthesizing low-complexity FIR filters with the least CSPT terms compared with the existing filter

synthesis algorithms. The use of common subexpressions in the coefficient synthesis algorithm has shown to be effective in designing multiplierless FIR filters with reduced hardware cost at no penalty to the throughput.

(3) A contention resolution algorithm for minimizing logic depth and adder cost of FIR filter

A dramatically different approach has been developed to the dependence graph and statistical method for common subexpression eliminations. This new algorithm, called the contention resolution algorithm for weight-2 horizontal subexpressions (CRAH-2), is based on an ingenious graph labeling method for detecting conflicting admission criteria of common subexpressions according to their frequency of occurrence. The poor selection of common subexpression is rectified after a localized evaluation of the cost associated with the conflicting common subexpressions. CRAH-2 provides a leeway to break away from the local minimum through a new data structure called the admissibility graph. Since only weight-2 common subexpressions are considered, the primary goal of optimization is the logic depth. The performances of the proposed algorithm demonstrated that CRAH-2 achieves the shortest logic depth with significant reduction in the number of logic operators compared with other reported algorithms. Besides, the proposed CRAH-2 is highly computational efficient as the common subexpressions need to be sorted only once.

(4) Extensions of contention resolution algorithm to more versatile forms of common subexpression elimination

The notion of contention resolution is extended to generate a series of contention resolution algorithms, provide greater flexibility to varying optimization options by considering different types of common subexpressions available in the MCM problem. Besides the ability inherited from CRAH-2 to revert a precedential decision of suboptimal common subexpressions by a localized cost function evaluation, the derivatives of the basic contention resolution algorithm are characterized by their capability in satisfying search for both intra- and inter-coefficient common subexpressions, in any legitimate composition of horizontal, vertical and oblique

common subexpressions. As the algorithms expand the common subexpressions to higher-weight only when there is cost saving, the logic depth can be adaptively controlled by constraining the weights of the common subexpressions in the admissibility graph. The variants of contention resolution algorithm follow an important tenet of good heuristic that significant improvement in the solution quality (in terms of logic complexity and logic depth) is attained with increased problem size but the computational time remains well bounded. Experimental results are analyzed and compared with a number of well-known filter architecture optimization methods to demonstrate the effectiveness and efficiency of the proposed algorithms.

The above contributions have led to the publications listed in the author's publications at the end of the thesis.

1.4 Organization of the Thesis

The thesis is organized into seven chapters. Chapter 1 expositis the motivation and objectives of the research work. The achievements resulted from this work presented in this thesis are summarized as four main contributions in this chapter.

Chapter 2 presents the fundamentals of digital filter and the background information pertaining to its design, synthesis and optimization. The direct form and transposed direct form structures of FIR filters are introduced and discussed. The number systems used in the design of digital filters, including the two's complement representation, the sum of power-of-two terms representation, the CSD representation and the Minimal Signed Digit (MSD) representation are introduced, followed by a comparison of their figure-of-merits. The problem of filter synthesis is stated. Key algorithms to this problem are presented and reviewed along with the projection of incorporating common subexpression sharing in the synthesized solutions. The last part of this chapter is devoted to the survey of a number of important optimization algorithms for the design of low-complexity FIR filter. Prevailing algorithms are categorized into two main groups of Common Subexpression Elimination (CSE) and Graph-Dependence (GD) algorithms for discussion.

Chapter 3 introduces the newly-formulated Hamming Weight Pyramid. The chapter presents the generation of HWP from its corresponding pyramids of natural number and CSD representation. An odd HWP (OHWP) with similar regular structure is derived. Interesting properties observed in HWP and OHWP are presented for their potential exploitation. The construction of HWP from a reduced HWP is explained. From the properties of the HWP, a new method to convert decimal number to its CSD format is derived. The proposed conversion method from the HWP stems a new CSD Tree. The application of HWP to extend the wordlength of the coefficients that can be managed by the RAG- n algorithm is presented.

Chapter 4 explores the integration of sharable common subexpression in filter coefficient synthesis process. The choice of only two common subexpressions, 101 and $10\bar{1}$ for the proposed filter coefficient synthesis algorithm is first justified. A new formulation of the filter coefficient synthesis problem with the consideration of these common subexpressions is established. This is followed by the introduction of CS-HWP as a search tool for low cost CSD coefficients. The proposed two-stage integrated approach to the design of low-complexity digital filter is presented. The operational characteristics and parameters of the first stage called the *CSPT Coefficient Allocation Stage* and the second stage called the *Quantization Compensation Stage* are described and discussed. The solutions of the proposed algorithm are analyzed and compared with those synthesized by other algorithms through two well-known design examples. Additional experimental results are also provided and discussed.

Chapter 5 introduces the concept of contention resolution in MCM and proposes a new algorithm for horizontal common subexpression elimination. An admissibility graph is defined as a unique data structure for the proposed contention resolution algorithm. The terminologies of the admissibility graph are illustrated through an example. Contention resolution, which is the principal idea of the algorithm, is described and two most frequently encountered contentions are discussed. The proposed contention resolution algorithm for weight-two common subexpression (CRAH-2) is described and an example

is used to demonstrate the procedure. Its performance in terms of logic depth and logic complexity are evaluated and analyzed with benchmark filters and randomly generated coefficient sets, and compared with other distinguished common subexpression elimination algorithms.

Chapter 6 extends the basic contention resolution algorithm of Chapter 5 to higher weight common subexpressions in two dimensions. Three types of common subexpressions considered by the CRA variants are described. Then the scenarios of more complex contention in the admissibility graph that lead to the generation of precedence path are explained. The derivatives of contention resolution algorithms are classified according to the hamming different weight constraints of both inter- and intra-coefficient common subexpressions. The chapter ends with the performance evaluation and analysis. The qualities of the solutions generated by the proposed algorithms are compared against competitive common subexpression elimination algorithms.

Finally, Chapter 7 reviews the methods proposed in this thesis by highlighting their features and merits, discusses some of the remaining problems and suggests potential relevant topics worthy of further investigation.

Chapter 2

Literature Review

Digital FIR filter has ubiquitous applications in digital signal processing (DSP), telecommunications, computer graphics and control [AIT93, AVI61, BEN92, CHE99, POT96, VIN05b]. Due to the importance of digital filters in a wide spread of applications, synthesis and optimization of Finite Impulse Response (FIR) filters have attracted the attention of numerous researchers. For the past couples of decades, many techniques and algorithms have been proposed for the design, synthesis and optimisation of FIR filters. This chapter sets the scene of this thesis by providing some background information of the topic discussed and reviews relevant and important previous work. The chapter is organized as follows. Section 2.1 provides an overview of digital FIR filter and presents a few most popularly used architectures for its implementation. The frequently adopted number systems in digital FIR filters are discussed in Section 2.2, followed by the survey and study of the key algorithms on filter synthesis and optimizations in Section 2.3 and 2.4, respectively. The chapter is summarized in Section 2.5.

2.1 Overview of Digital FIR Filter

The frequency response of a linear-phase FIR filter can be described by a real-valued zero-phase frequency response, $H_R(\omega)$ function and a real-valued phase function, $\theta(\omega)$, i.e.,

$$H(\omega) = H_R(\omega) e^{j\theta(\omega)} \quad (2.1)$$

The causality imposes a strong magnitude $|H(\omega)|$ and phase $\theta(\omega)$ dependency on the above frequency response characteristic. Therefore, the realization of practical frequency-

selective filters is generally restricted to those that are characterized by linear time-invariant responses. An FIR filter of filter order N exhibits the following system function

$$H(z) = \sum_{k=0}^N h_k z^{-k} \quad (2.2)$$

which can be equivalently described by the difference equation

$$y(n) = \sum_{k=0}^N h_k x(n-k) \quad (2.3)$$

where $y(n)$ is the filter output, $x(n)$ is the filter input, and h_k are constants, determined by the filter specification. Thus, without regards to the cost of implementation, the design of a practical FIR filter reduces to the determination of a set of coefficients h_k so that the convolution of the unit sample response of the system with the input approximates the ideal frequency characteristics of (2.1) as close as possible.

The implementations of an FIR system vary in different structures, including the direct form, the cascade-form, frequency-sampling realization and lattice realization [PRO96]. Different structures fit different desiderata of applications, and carry different attributes that could be tailored to suit some specific objectives and/or constraints of implementation. Among the structures, the direct form structure is the simplest, which can be further classified as the direct form and transposed direct form with distinct distributions of delay elements. These filter structures will be detailed in what follows, with illustration on how they feature in filter synthesis and optimization.

2.1.1 Direct Form Structure

One of the most intuitive structures for the implementation of FIR filter is the direct form structure, which is a direct mapping of (2.3). In the direct form structure, each of the delayed versions of the input signal is multiplied with its corresponding filter coefficient, and the products are summed up to form the output signal. Generally, for an order- N direct form FIR structure, it requires $N + 1$ multiplications and N additions. The structure is illustrated in Figure 2.1.

Figure 2.1 Direct form FIR filter structure

As shown in Figure 2.1, the number of multiplications and additions in the structure is proportional to the number of filter orders. Each block of “ Z^{-1} ” represents a delay element, or a pipelined stage of the FIR calculation. The delay of the filter increases linearly with the filter length. The critical path delay is given by:

$$t_{direct} = t_{mult} + N \times t_{add} \quad (2.4)$$

where t_{mult} is the maximum tap multiplier delay and t_{add} is the delay of adders used in the final accumulation line. The accumulation line can be replaced by a multi-input adder, in which way the multi-input adder would dominate the speed of the overall system. One of the attractive features of the direct form structure is that the delay elements are separated from the multipliers and accumulators. This has an adverse effect of high area and power consumption. Hawley, et al. [HAW96] observed that by merging the N additions, a direct form structure with a tree of carry-save adders (CSAs) can lower the latency of the accumulator to an order, which is logarithmically instead of linearly increasing with the filter length. i.e.,

$$t_{direct} = t_{mult} + \log_{1.5}(N + 1) \times t_{add} \quad (2.5)$$

By replacing multipliers with adders and shifters, the multiplications of the input and coefficients, along with the final accumulation line can be realized with a carry-saved adder-tree [REU89, HAW96]. Specifically, Ye et al. [YE03, YE04] note that the use of merged arithmetic architecture on direct form structure taking into account the delay profile of the inputs to the final accumulator can speed up the critical path with lower area and power-consumption.

2.1.2 Transposed Direct Form Structure

Another frequently used filter structure is the transposed direct form structure. The transposed direct form structure is derived by applying the transposition theorem, which states that by interchanging the input and the output and reverse all signal flows in a signal-flow graph of single input single output (SISO) system, the transfer function is unchanged [OPP89]. Figure 2.2 shows the transposed direct form structure, in which the input is fed to all filter coefficient multipliers simultaneously. Therefore, all multiplications are performed on the current input sample. The product from each tap is delayed one clock cycle before it is accumulated with the next filter tap product. This structure leads to a balanced N -stage pipelined architecture with the pipe filled over $N + 1$ sample periods, and thereafter one output is produced for each clock cycle.

Figure 2.2 Transposed direct form FIR structure

The main advantage of the transposed direct form structure is its high throughput rate. The critical path delay is given by:

$$t_{\text{transposed}} = t_{\text{mult}} + t_{\text{add}} \quad (2.6)$$

Since the critical path delay is independent of the number of taps, transposed direct form has been the preferred architecture for high-speed, high order FIR filter implementation. However, it has a major drawback, which is the length of the registers increases progressively with their attendant accumulators. Also, the input node has a large fan-out. The distribution of the input to a large number of taps requires more than a single driver. The capacitive loading of the most significant bit (MSB) data line can be reduced significantly by a multilevel buffer tree [YOS90]. The transposed direct form structure differs from the direct form structure in the positions of the delay elements, therefore the

logic complexity involved in realizing the multiplications can be minimized as a Multiple Constant Multiplication (MCM) problem [POT96] to be discussed later. In these multiplierless implementations, the reusable adders are merged so that up to 50% hardware cost may be saved. The prominence hardware savings of reusable components in transposed direct form structure have ignited new research interest to incorporate reusable terms as a quality measure in filter coefficient synthesis.

2.1.3 Linear Phase Direct Form Structure

FIR filters can be implemented with an exact linear phase response, which is desired in many applications such as [JAN02, LIM90]. To obtain a linear phase FIR filter, the impulse response is either symmetric around $n = N/2$

$$h(n) = h(N - n), \quad n = 0, 1, \dots, N \quad (2.7)$$

or antisymmetric around $n = N/2$

$$h(n) = -h(N - n), \quad n = 0, 1, \dots, N \quad (2.8)$$

The symmetry of linear phase filter coefficients makes it possible to reduce the hardware resources by sharing multipliers, although a slight increase in some signal routings would be incurred. Figure 2.3a shows the examples of an even order filter in direct form structure and an odd order filter in transposed direct form structure.

a) Even order linear phase direct form FIR filter utilizing coefficient symmetry

b) Odd order linear phase transposed direct form FIR filter utilizing coefficient symmetry

Figure 2.3 FIR filter with symmetrical coefficients

As shown in Figure 2.3, as many as half of the multipliers can be shared by virtue of symmetry. The multipliers of linear phase structures are shared in direct form and transposed direct form structures according to the positions of their delay elements.

There are other filter structures, many of which are variations of the basic filter structures. The number of filter order, N dictates the amount of memory and logic computation required and this is commensurate with the amount of "filtering" the filter can perform. Therefore, N is determined by the stopband attenuation. The required order for a linear phase FIR filters can be estimated as

$$N = 2\pi \frac{-20 \log_{10}(\sqrt{\delta_p \delta_s}) - 13}{14.6(\omega_s T - \omega_p T)} \quad (2.9)$$

where δ_p , δ_s , $\omega_p T$, and $\omega_s T$ denote the passband ripple, stopband ripple, passband edge, and stopband edge, respectively[#][KAI74]. Equation (2.9) is one of the widely used order estimators and other formulas can be found in [HER73, ICH00]. In general, the transition band is the band of frequencies between the passband and stopband edges. The narrower the transition band, the more taps are required to implement the filter. A smaller transition band results in a sharper filter. Increasing the filter taps will lead to higher stopband attenuation, less ripple, and narrower frequency response.

2.2 Number Representations in Digital FIR Filters

Several number representations are available for filter coefficient synthesis. Different number representations will invoke different topologies of arithmetic operators for the same filter specifications. Each number system has its unique features that make it more suitable than others in certain aspects of synthesis and optimization problems. In hardware implementation, a signal can be scaled by a coefficient using hardwired shifters and adders, but the total number of arithmetic operations is fewer when subtractions are also permitted. To employ both addition and subtraction to reduce the hardware cost, signed integers are usually used to represent the fixed point coefficients. In this section, some of the frequently used number representations in digital system design are reviewed.

2.2.1. Two's Complement Representation

One simple way to represent both positive and negative integers is to dedicate a single bit for sign indication and represent the absolute value of the integer by a binary (radix-2 weighted) number. The problems of this straightforward signed-magnitude representation are that there is two representations of zero and the addition of opposite-sign operands (subtraction) must be handled differently from that of the same-sign operands. These problems can be overcome with the two's complement number representation. Two's complement representation is a radix 2 complement representation, and it is widely used in DSP applications. In two's complement representation, the value of the most significant bit determines the polarity of the number. A integer x can be written as

$$x = -b_0 2^{W-1} + \sum_{i=2}^W b_i 2^{W-i}, b_i \in \{0,1\} \quad (2.10)$$

where W is wordlength of the representation. The conversion of a binary number to its two's complement representation is straightforward, i.e., by adding an *ulp* (unit in least significance position) to its bitwise complementation, as shown in (2.11) [PAR00]:

$$2^k - x = [(2^k - ulp) - x] + ulp = x^{compl} + ulp \quad (2.11)$$

where x^{compl} denotes bitwise complementation of x .

2.2.2 Signed Digit Representation

A signed digit (SD) representation [AVI61] of radix r is a positional number system with a symmetrical digit set, i.e., $a = (a_{n-1}, \dots, a_2, a_1, a_0)$ with $a_i \in \{0, \pm 1, \pm 2, \dots, \pm(r-1)\}$. Signed digit representations of radix 2 are widely used for digital arithmetic operations, such as Booth coding and modified Booth coding [DAS96, HAS96]. By making use of the digit set $\{0, \pm 1\}$, a number can be represented as a sum of Signed-Power-of-Two (SPT) terms as in (2.12):

$$x = \sum_{i=0}^{W-1} s_i 2^i, s_i \in \{0, \pm 1\} \quad (2.12)$$

The signed digit representation is a redundant representation, and one number may have more than one representations. When the number of nonzero digits (termed Hamming weight) is minimal, the representation is called Minimal Signed Digit (MSD), and Canonical Signed Digit (CSD) is a special subset of MSD.

2.3.2 Canonical Signed Digit and Minimal Signed Digit

The CSD and MSD [REI60] representations are two commonly used number representations in digital filter coefficient synthesis and optimizations. Both systems are radix-2 signed digit system with the digit set $\{0, \pm 1\}$ and usually $\bar{1}$ is used for denoting the value -1 . The MSD and CSD representations are crucial in reducing the hardware cost and power consumption in many complex arithmetic operations. The CSD representation has two properties. First, the number of nonzero digits is minimal, which is an inherited property from being a member of the MSD superset. From digital filter implementation perspective, this property guarantees the least number of adders/subtractors used for the multiplication of a variable with a constant represented in CSD and MSD. On average, the CSD and MSD representations can reduce 33% of nonzero digits compared with the binary representation [REI60]. It was also shown in [REI60] that the expected number of nonzero digits in an n -bit CSD number in the range $[-1, 1)$ tends asymptotically to $n/3 + 1/9$. Second, two nonzero digits are never adjacent in the CSD representation, which distinguish the CSD number from its MSD counterparts. Any arbitrary W -bit 2's complement number can be represented in CSD form with no

more than $(W + 1) / 2$ nonzero digits. If the second property is relaxed in the CSD representation, several MSD representations of the same number can be obtained. It is this canonicity that makes the CSD representation more popular in digital systems than the MSD.

To generate a CSD number, usually the decimal number needs to be converted to the binary form first. Hwang [HWA79] presented the traditional approach to generate the CSD number from the binary form. Let $B = b_{n-1} b_{n-2} \dots b_0$ and $CSD = s_{n-1} s_{n-2} \dots s_0$ be the number to be represented in binary form and in CSD form, respectively. $b_i \in \{0, 1\}$ and $s_i \in \{0, \pm 1\}$. Both vectors B and CSD represent the same value. The steps to generate the CSD representation are:

1. Starting at the least significant bit, b_0 of B , set the index $i = 0$ and the initial carry $c_0 = 0$.
2. Generate the next carry c_{i+1} according to the methods of conventional binary arithmetic by examining the two adjacent bits, b_{i+1} and b_i of the vector B according to the value of the carry-in bit, c_i .
3. Use the equation $s_i = b_i + c_i - 2c_{i+1}$ to generate the i^{th} digit s_i of the vector CSD .
4. Increment the index i by one and repeat the procedure at Step 2 until $i = n - 1$.

The procedure is summarized in a lookup table, shown in Table 2.1. The interdependence between the output c_{i+1} , and the input b_{i+1} , b_i and c_i can be described as follows: c_{i+1} will be set if at least two of b_{i+1} , b_i and c_i are 1.

Table 2.1 CSD Generation with lookup table [HWA79]

		Carry-in	CSD code	Carry-out
b_{i+1}	b_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	1	0	1	0
1	0	0	0	0
1	1	0	$\overline{1}$	1
0	0	1	1	0
0	1	1	0	1
1	0	1	$\overline{1}$	1
1	1	1	0	1

Other algorithms for the conversion of the binary to SD [HWA89] representations involve the computation of interim difference and 2's complement to CSD by applying T2I conversion (conversion of a number to its inverse-2's complement format) partially or searching for the nearest SPT term [HAS96, LIM91]. Compared with CSD, MSD conversion is not so straightforward. Some methods have also been developed to generate MSD and signed digit representations with the number of nonzero digits more than the minimal Hamming weight [PAR02, GUS02b, DEM04a]. Due to the vast irregular variants of MSD form, the generation algorithms are by and large based on exhaustive search, which severely restricts the applications of MSD. Some approaches based on look-ahead circuits [DAS96, HER95] and different derivations of CSD [AKO97, HAS96] have been reported to circumvent the conversion problem. The dominance of conversion algorithms to CSD based on normal binary instead of decimal number can be easily understood as binary arithmetic prevails in typical digital computers. Recently, IBM developed the first decimal arithmetic processor, eServer z900 to perform decimal instructions in hardware [BUS01, SCH02]. It is foreseeable that the direct conversion from decimal number to CSD will benefit from the decimal arithmetic unit of IBM eServer z900 or emerging decimal processors with decimal computation capability limited to integer operands. The significance and interest in decimal arithmetic has also led to a proposed revision to the IEEE 754 standard for floating point (FP) arithmetic to include specifications for decimal arithmetic specifications [IEEE04]. On the other hand, in design automation algorithms leading to efficient digital filter design [HAR96, MAR02], the original constant coefficients are normally derived in decimal (integer) form and it would be useful to interpret the CSD number and its conversion directly from the decimal form. Besides, there exist a number of algorithms [CHE99] for synthesizing signed digit coefficients to map the infinite precision filter coefficients into finite length coefficients so that fewest possible SPT terms are used to meet the design specifications. In coefficient synthesis, the Hamming weight of a multiplicand decides the number of adders needed, and hence the complexity of the circuits to be implemented. The characteristics of the Hamming weights of the CSD numbers are worth investigated and exploited in the light of enhancing the computation efficiency and performance of design

automation algorithms. The work on the manipulation of Hamming weight of CSD and a new direct conversion algorithm will be presented in Chapter 3 of this thesis.

2.3.4 Comparisons of Different Number Representations

The abovementioned number systems mentioned are summarized in Table 2.2. The equation, an example of the same numeral, and whether the representation is unique are listed for each system.

Table 2.2 Comparisons of Number Representations

	Representation	Example		Uniqueness
<i>Decimal</i>	x	103	-103	Yes
<i>Signed-magnitude</i>	$\text{sgn}(x) = \begin{cases} 0 & \text{if } x > 0 \\ 1 & \text{if } x < 0 \end{cases}$ $ x = \sum_{i=0}^{W-1} b_i 2^i, \quad b_i \in \{0, 1\}$	0, 1100111	1, 1100111	Yes
<i>2's Complement</i>	$x = -b_0 2^{W-1} + \sum_{i=2}^W b_i 2^{W-i}, \quad b_i \in \{0, 1\}$	01100111	10011001	Yes
<i>Signed Digits</i>	$x = \sum_{i=0}^{W-1} s_i 2^i, \quad s_i \in \{0, \pm 1\}$	11100111 10100111	11100111 10100111	No
<i>Canonical Signed Digit</i>	$x = \sum_{i=0}^{W-1} s_i 2^i, \quad s_i s_{i+1} = 0, \quad s_i \in \{0, \pm 1\}$	10101001	10101001	Yes
<i>Minimal Signed Digit</i>	$x = \sum_{i=0}^{W-1} s_i 2^i, \quad s_i \in \{0, \pm 1\}$	1101001 10011001	1101001 10011001	No

From Table 2.2, SD and MSD are two number systems with inconsistent representations of the same numeral. The representation inconsistency provides flexibility for optimization on one hand but hungers computational efficiency on the other hand due to the search complexity. The conflicting scenarios encountered in there are particularly intricate to resolve. In this sense, CSD owns the popularity in filter synthesis and optimization to its canonicity and minimal number of nonzero digits. Comparing with the 2's complement format, the number of adders required for the multiplication of a CSD constant with a variable is generally lower.

It should be noted that the unit of least (significant) position, ulp is very useful in describing number arithmetic without making a distinction between integers and fixed-point representations that include fractional parts. For integers, $ulp = 1$, and for fractional numbers, the ulp is a negative power of two in any of the above representations.

2.3 Filter Coefficient Synthesis Algorithms

As it is generally agreed that multipliers are power hungry devices and occupy large silicon area, the trend towards the design of fixed point FIR filters is to replace the expensive multiplication operations by simpler additions and shift operations, whereby shifters with fixed amount of shifts can be implemented at almost no cost by hardwired routes [BUL91, DEM95, HAR96, MAR02, VIN03]. Because of the finite word length for fixed point implementation, the basic principle behind the design of FIR filter for optimal multiplierless implementation is to approximate each filter coefficient with a minimal number of SPT terms. The process of finding the desirable SPT terms to represent the real valued coefficient set is called the filter coefficient synthesis. In this way, the filter complexity is determined by the number of additions/subtractions required to implement the multiplications, which in turn is directly related to the number of SPT terms used to synthesize the filter coefficients. Thus, the constrained optimization problem becomes one of finding a set of filter coefficients with a minimal number of SPT terms satisfying a given magnitude response specification [AIT93, AVI61, BEN92, CEM93, CHE99, ROS04].

2.3.1 Problem formulation

The finite wordlength of filter coefficients introduces frequency response error, in that scenario, quantization errors have to be accounted and adjustments are necessary to ensure that the quantized coefficient set still meets the design specification. The dynamic range is application-dependent as the data wordlength is generally dictated by the signal-to-noise ratio and the resolution of the analog-to-digital converter. As each coefficient contributes to the stopband attenuation in an unpredictable degree, the aggregate error of

the quantized coefficients is usually considered as a whole. The truncation or rounding of infinite precision real valued filter coefficients introduces frequency response error of $H'(\omega) - H(\omega)$, where $H'(\omega)$ and $H(\omega)$ are the normalized magnitude responses of the discrete-coefficient-value and continuous-coefficient-value filters, respectively, at frequency ω . The normalized frequency response has $H(\omega) = 1$ in the passband, and $H(\omega) = 0$ in the stopband. However, in most practical applications, the magnitude response $|H(\omega)|$ needs not be zero at stopband nor be constant over the entire passband. A small amount of ripples, δ_p and δ_s at the passband and stopband are tolerable. If the bandedge frequencies ω_p is defined as the end of passband and ω_s the beginning of stopband, then the design of a linear phase FIR filters can be formulated as a constrained optimization problem, which transforms the continuous-valued filter coefficients to finite precision subjected to the following constraints:

$$|H'(\omega_p T) - 1| \leq \delta_p k(\omega_p) \quad (2.13)$$

$$|H'(\omega_s T)| \leq \delta_s k(\omega_s) \quad (2.14)$$

where δ is the ripple value to be minimized and $k(\omega)$ is a positive weighting function. $\delta k(\omega)$ specifies the maximum permissible frequency response error.

In many filter implementations, the absolute value of the passband gain is immaterial. What matters most is the relative attenuation between the passband and stopband, which is specified by the ratio of the values of $k(\omega_p)$ to that of $k(\omega_s)$. This is referred to as the Normalized Peak Ripple Magnitude (NPRM) [LIM90], and it is expressed as:

$$\delta_{NPRM} = \max \left\{ \frac{\delta_p \omega_p}{g}, \frac{\delta_s \omega_s}{g} \right\} \quad (2.15)$$

where g is the average passband gain. The NPRM is often used as a parameter to guide the synthesis of filter coefficients to achieve the desired response while optimizing its implementation cost.

From implementation perspective, the coefficient multiplication in the convolution equation of (2.3) determines the complexity of digital filters. A filter coefficient, $h_{spr}(i)$ can be expressed as a sum of SPT terms in (2.16) [CHE99]:

$$h_{spt}(i) = \sum_{k=1}^{L_i} s_{i,k} 2^{-p_{i,k}} \quad (2.16)$$

where $s_{i,k} \in \{-1, 1\}$ is the k -th SPT term at bit position, $p_{i,k} \in \{1, 2, \dots, B\}$. It is assumed without loss of generality that $0 \leq |h_{spt}(i)| < 1$ and $h_{spt}(i)$ has a wordlength of B bits and comprises L_i SPT terms.

Thus, the fundamental optimization problem becomes one of synthesizing a set of coefficients for a filter of order N that minimizes the total number of SPT terms, J , given in (2.17).

$$J = \sum_{i=0}^N \sum_{k=1}^{L_i} |s_{k,i}| \quad (2.17)$$

subjected to the objective functions of frequency response, such as the minimal NPRM constraint given by (2.16).

2.3.2 Filter Synthesis Approaches

There exist a number of algorithms for synthesizing good CSD coefficients to minimize the number of SPT terms required for the implementation of low complexity FIR filter [AIT93, AVI61, BEN92, CEM93, CHE99, ROS04, YLI01]. The set of coefficients generated must fulfill the frequency response specification of the filter. There are two ways to perform filter synthesis: some algorithms search for the coefficient set with least number of nonzero digits while others fix the total number of nonzero digits and allocate the nonzero digits. The first category focuses on minimizing the hardware cost while the algorithms fall into the latter usually aim at minimizing the ripple subject to hardware specifications (wordlength and filter order). In essence, these two approaches both contribute to the same idea of testing the satisfiability of different coefficient sets but under somewhat different logical constraints. Generally, filter coefficient synthesis methodologies can be categorized into two types, namely, the optimal approaches and the suboptimal approaches. In the following subsections, classic algorithms in these two categories are introduced.

2.3.2.1 Optimal Approaches

As the name implies, methods in this category are claimed to have guaranteed optimality. The optimality is ensured by the use of exhaustive search methods. The Mixed Integer Linear Programming (MILP) is frequently employed in the optimal approach [KOD81, LIM90, YLI01, GUS01a]. As the filter synthesis problem is a NP-complete problem, the main drawback of the optimal approach is its formidable computation time, especially for filters with large number of taps. Y. C. Lim [LIM90] studied the computational complexity of applying MILP to filter coefficient synthesis problem, and he concluded convincingly that the computation time increases at least exponentially with respect to the filter length. This has limited the applicability of MILP to coefficient synthesis problem with filter length of 70 or less. Attempts have been made to reduce the search space in a logical and systematic way in order to shorten the computation time while the optimality is still maintained [GUS01a], but they manage to release only a small portion of the overall computational load. The fraction of redundant computations that could be avoided in an exhaustive search to reach an exact optimal solution is limited, therefore many heuristic techniques have been developed to deal with stringent filter specifications.

2.3.2.2 Suboptimal Approaches

Suboptimal approaches do not guarantee optimal results, but quasi-optimal results are obtained in reasonable time. There are many suboptimal approaches for the filter coefficient synthesis problem, and the most popular one is the local search approach, from which many interesting proposals have been stemmed. The local search approach is based on the observation that the desired solution usually lie in the vicinity of the real valued coefficient set [LIM83]. Hence the heuristic search starts with the real valued coefficient set and search in its neighborhood to obtain the minimal solution within a reasonable time. The computation time for the local search method depends very much on the neighborhood size. Kodek [KOD81] stated in his paper that the bivariate search, where the rounded coefficients are varied by $\pm$ one quantization step size, yields better results than a univariate local search. Thereafter, the bivariate method has become a common method for constructing the neighborhood around the scaled and rounded

coefficients in a local search, although the amounts of quantization enumerated differ in different methods. Besides the definition of neighborhood, the actual search method is another important factor of consideration. Samueli [SAM90] proposed a bivariate local search method whereby the filter response is simulated for every change in a coefficient. The change that results in the minimum value of the objective function is selected. Chen and Willson [CHE99] developed a two-stage local search scheme by applying the trellis search algorithm. The trellis search algorithm has an advantage that instead of rigidly limiting the number of SPT terms per coefficient, it allocates the SPT terms dynamically to the locations that best improve the filter frequency response. The computation time of the local search algorithm does not grow exponentially as the filter length increases. However, it does not guarantee optimal frequency response. One of its drawbacks is that there is a risk of stuck at local minimum. This problem is very difficult to tackle and there is no solution to it according to the authors, other than adjusting the initial set of coefficients. There are many other suboptimal approaches, such as the tree search with least-squares optimization [LEE93, LIM83], stochastic optimization [BEN92, CEM93, SCH93], quantization by coefficient sensitivity [AIT93, SHA91], entropy-directed deterministic annealing [PER05], among many others.

2.3.2.3 Filter Synthesis Algorithms with Common Subexpression Sharing

In multiplierless implementation of high-speed digital filter, the common subexpressions are shared in the transposed direct form structure. As the input signal is multiplied with all coefficients before arriving at the accumulators before the respective delay elements, identical partial sums need not be duplicated but the generated partial sums can be reused to save hardware cost. More on the replication of partial sums will be covered later in Section 2.4 but what prompts the author in this optimization process is a radical paradigm shift of the quality metric used in filter synthesis algorithms. When SPT terms in the coefficients are merged to common subexpressions, the hardware cost is no longer proportional to the number of SPT terms. The hardware cost, simplified to the number of generic adders, is determined by the number of occurrences of common subexpressions and the number of remaining SPT terms. This poses the skepticism to earlier postulation

that coefficient set with less SPT terms will also yield solution with less adder cost. The excess SPT terms may compose more common subexpressions that can be shared across coefficients and hence reduce the adder cost for the implementation of FIR filter. To the best of the author's knowledge, only a few researchers have actually conceived the integration of the synthesis step with the common subexpression elimination (CSE) step. Chen and Willson [CHE99] mentioned additionally that they merged the common subexpressions, 101 and $10\bar{1}$ before trellis search and the results showed that the merge-trellis search realized a saving of 18% to 23% of carry-save adders (CSA) for CSA-tree based transposed direct form architecture. Yli-Kaakinen's algorithm [YLI01] involved the sharing of higher weight common subexpressions, and it first generated a look-up table containing all possible power-of-two numbers for a given wordlength and a given maximum number of SPT terms per coefficient, and then a branch-and-bound procedure was applied to evaluate the coefficient sets. Existing CSE algorithms were applied in Yli-Kaakinen's algorithm. A MILP problem for designing FIR filters that incorporated subexpression sharing was formulated by Gustafsson and Wanhammar [GUS02c]. In [ROS04], Rosa et al. also proposed an algorithm that performed common subexpression elimination on selected coefficient sets of weight- n based on the filter specifications. However, these algorithms either restrict the occurrences of common subexpressions by merging SPT terms before the minimization process based on an initial coefficient set or relying on an exhaustive search for the identification of common subexpressions. Except for Chen and Willson's algorithm, all the other algorithms are optimal algorithms. As discussed before, optimal algorithms become highly inefficient or infeasible for large coefficient sets. In Chapter 4 of this thesis, a new heuristic algorithm is presented to synthesizing good filter coefficient set guided by a completely different quality metric.

2.4 Multiple Constant Multiplication Problem

In high-level synthesis of VLSI circuits, the use of transformations has led to the development of powerful design automation algorithms for many compute intensive DSP problems. Common subexpression sharing is one such computational transformation that has been applied with phenomenal success to the implementation of hardware efficient

digital filters. The interest in exploring high-level synthesis techniques is rekindled by the modeling of the core operation of transposed direct form FIR filter as a general MCM problem in [POT96]. MCM problems are also found in many application kernels like wireless communication, multimedia signal processing and linear transforms for automatic control. MCM refers to a group of scalar multiplications of different constants with the same variable, as shown in the mapping of the architecture of Figure 2.2 to Figure 2.4 which consists of a MCM block and an accumulator-delay line. The efficiency in reducing the cost metric (power, combinational circuit area, and logic depth) of the MCM block directly influences the performance of the compiled silicon solution of an FIR filter. The area-time optimization of MCM problem has often been accomplished through the substitution of multiplications with a reduced number of shift-and-add operations. It is therefore feasible to derive optimization methods that can facilitate automatic generation of compliant silicon-efficient solutions to MCM problems.

Figure 2.4 Multiple Constant Multiplication Block in FIR Filter

Two methods are generally employed to reduce the total hardware cost of the MCM block. The CSE algorithms [HAR96, PAS99, MAR02, POT96] search for common subexpressions and then stochastically maximize the reuse of the products of the input and common subexpressions. The other category is the graph-dependence (GD) algorithms [BUL91, DEM95, DEM00, GUS02b, KAN01], whereby partial sums are symbolically encapsulated in the nodes of the graph and the shift amounts of the partial sums are annotated on the edges. GD algorithms involve the synthesis of a set of minimal cardinality connected graphs from the unity source to the sinks, which are the coefficients to be synthesized. The context of common subexpressions and partial sums in CSE and GD algorithms will be introduced and discussed in following subsections.

2.4.1 Common Subexpression and Partial Sum

The common subexpression used in CSE algorithms and partial sum (also know as fundamental) used in GD algorithms are two semantically equivalent terms for the role they played in the MCM problem. They are, however, different in the context of application. CSE algorithms usually use signed digit coefficients, mostly in CSD and MSD representations. The substrings that replicate themselves in a coefficient or in different coefficients are called common subexpressions. For example, 101 is a common subexpression of the coefficient 10100101. Direct multiplierless implementation of this coefficient requires three adders as shown in Figure 2.5(a) but it reduces to only two adders with the extraction and reuse of the common subexpressions 101 as illustrated in Figure 2.5 (b).

Figure 2.5 CSE Implementations of coefficient 10100101

GD algorithms decompose each inner product in the convolution process into a number of primitive arithmetic operations (additions, shifts and subtractions) [BUL91]. The nodes of the directed acyclic graph synthesized by a typical GD algorithm represent different inner products of the input sample called the partial sums. The directed acyclic graph for the same coefficient 10100101, equivalent to 165 in decimal, is shown in Figure 2.6.

Figure 2.6 GD implementation of coefficient 165 (10100101)

In Figure 2.6, two partial sums, $5x(n)$ and $165x(n)$ each represented by a node “•”, are generated. From Figure 2.5 and Figure 2.6, it is observed that common subexpressions

and partial sums are both redundant products in disguise for two different analytical systems. In signed digit representations, common subexpressions are observable patterns in a given number representations but their existence is explicitly expressed in and implied by the representation. Visually obscure redundant inner product will not be eliminated in the CSE algorithms if no repetitive pattern of its value can be detected in the selected representation. Therefore, common subexpression is representation-dependent. In decimal number system, partial sum is value-dependent and it is implicitly implied without a fixed pattern. More calculations are needed to select a smaller set of partial sums to compose a quality solution in view of the much larger number of possible combinations of inner product values to be enumerated. There existing CSE algorithms that operating on MSD [PAR01], sharing similar benefits and problems as the GD algorithms. The search for good common subexpressions is always based on the overall frequency of occurrences of each subexpression within the targeted coefficient set. Common subexpressions with higher frequency are assumed to be more beneficial. On the contrary, the search for good partial sums is based on the sequentially synthesized coefficients in the procedure. Improvement to computational efficiency has been made to pre-synthesis values from within a restricted range to store the most beneficial partial sums. In this sense, common subexpression can be considered as an entity with ‘global’ optimization outlook whereas partial sum has a somewhat ‘localized’ optimization context. This subtle deviation poses different types of susceptibility to these two classes of algorithms. CSE algorithms are sensitive to the order of frequencies of common subexpressions and GD algorithms are sensitive to the order of coefficients being synthesis. In what follows, the pros and cons of representative algorithms in these two categories will be discussed in details.

2.4.2 Common Subexpressions Elimination Algorithms (CSE)

Elimination of common subexpressions as a lucid mean to reduce the MCM block logic complexity has been widely adopted to realize VLSI efficient FIR filters [GUS02a, HAR96, JAN02, PAR01, PAS99, MAR02, POT96, VIN03, YAO04, YUR99]. Different CSE algorithms lead to different realizations of MCM block of Figure 2.4. The qualities

of the solutions generated by different CSE algorithms for the same filter specification are primarily attributed to the difference in their extracted common subexpressions. The objective function and the representation of the filter coefficients play important roles in determining the quality of the solution. The objective function used for the generation of finite word length filter coefficients is based primarily on the performance error between the ideal infinite precision filter response and the achievable response with discrete coefficients measured at several points in the frequency domain. The representation of the filter coefficients are crucial to the quality of solution generated by CSE algorithms. CSE algorithms usually use SD representations since signed arithmetic operations (adder and subtractor) can be conveniently designed based on two's complement representation with compatible area-time complexity. Among the popular representations used for CSE problems, CSD [GUS02a, HAR96, PAS99, MAR02, YUR99] and MSD [PAR01] representations provide the minimum number of nonzero digits. As mentioned earlier in this chapter, one significant characteristic that differentiates CSD from MSD representations is that CSD is unique and MSD may appear in more than one forms for the same coefficient.

There are three different kinds of common subexpressions or repeated partial sums exploited by CSE algorithms. By representing the set of CSD coefficients in a two-dimensional table [HAR96], common subexpressions can be extracted by identifying horizontal (intra-coefficient), vertical and oblique (inter-coefficient) groups of identical pattern cells. Intra-coefficient common subexpressions are formed by nonzero digits from within the same coefficient and inter-coefficient common subexpressions are formed by nonzero digits from different coefficients. Due to the computational complexity and the fact that linear phase finite impulse response filters (LPFIRs) are symmetrical, the search for redundant computations in MCM block is normally confined to horizontal common subexpressions [GUS02a, PAS99, MAR02, POT96]. Inter-coefficient common subexpressions, which necessitate the insertion of delay elements into the generator of common subexpressions, are seldom considered. The basic procedure of CSE algorithms is to eliminate the most critical common subexpressions within the target coefficient set before the less critical common subexpressions. The criticality of a common

subexpression is usually measured by its frequency of occurrences. Hartley [HAR96] was one of the pioneers to exemplify the use of horizontal, vertical and oblique common subexpressions in digital filter design. However, due to the limited computing power of the machine at that time, he restricted the algorithm to seek for maximal sharings of the two most common subexpressions, 101 and $10\bar{1}$. Multiple coefficients can be optimized simultaneously by this algorithm. Hartley's algorithm demonstrates that by allowing the common subexpressions spanning different delayed copies of the input to be shared, it can lead to better results than are otherwise impossible. Another well known CSE algorithm called the Iterative Matching (ITM) algorithm was proposed by Potkonjak et al. [POT96] who introduced the term, MCM. ITM is a recursive decomposition algorithm based on a specially defined SD format. This method finds the maximum number of coincidences between two SD coefficients. It looks for the same nonzero digits with positions within the coefficients for the least number of shifts. The minimum number of shifts needed has been taken into account in ITM to minimize the number of shifters. Different kinds of number system chosen for this algorithm are very important as it greatly affects the quality of the solution and the performance of the final implementation of the FIR filter. Besides Hartley algorithm, there is another very similar algorithm Paško's algorithm, which freed the fixed shift position requirements of common subexpressions in ITM algorithm. Paško's algorithm also uses CSD representation. These two latter algorithms search for the highest hamming weight common subexpressions to maximize the reuse of higher weight common subexpressions before breaking them down to smaller ones, whereas Hartley searches for weight-2 common subexpressions and merges them to larger ones subsequently. Martínez-Peiró et. al. [MAR02] proposed a Nonrecursive Signed Common Subexpression Elimination (NRSCSE) algorithm. NRSCSE searches for the nonrecursive signed common subexpressions that must be eliminated from the original CSD array. It is called nonrecursive because it does not utilize common subexpressions recursively. This leads to two main consequences. The highest weight of the common subexpressions can be determined, which means that all partial sums generated may be restricted in its maximum nonzero digits. If the weight of subexpressions is confined to 2, solutions with the least logic depth can be achieved. This

algorithm triggered the interest of researchers as the reduction of logic depth in CSE algorithms has an implication on high-speed, low-power circuit FIR filter design.

Until recently, there have been comparatively much fewer CSE algorithms that exploit the inter-coefficient redundancy. After Hartley [HAR96] had proposed the prototype algorithm for all horizontal, vertical and oblique common subexpressions, Macleod [MAC04] extended the algorithm to matrix multipliers. Jang and Yang [JAN02] showed by means of specific filter example that vertical common subexpressions transcend horizontal common subexpressions in LPFIRs as adjacent coefficients have similar pattern in the MSB portion. However, Vinod et al. [VIN03] argued that the elimination of vertical common subexpressions alone does not guarantee greater hardware savings over the conventional horizontal CSE methods in most practical LPFIR filters. Based on their experimentation statistic, they proposed the extraction of the four most common horizontal subexpressions of Hamming weight-2 before the remaining nonzero digits are considered for vertical common subexpressions sharing. To date, none of the algorithms has simultaneously utilized all intra- and inter-coefficient common subexpressions, including the oblique common subexpressions and the higher weight common subexpressions of all sorts to reduce the adder costs of the MCM block.

For coefficients expressed in CSD form, exact optimal subexpression sharing algorithms based on integer linear programming (ILP) model have been proposed [GUS02a, YUR99]. The algorithm of [YUR99] is restricted to the minimization of weight-2 subexpressions only. In [GUS02a], it has been extended to deal with subexpressions of any weight. Both algorithms have high computational complexity as the optimal selection of common subexpressions is by itself an NP-complete problem. The ILP models are thus not suitable for large filters.

Optimal algorithms for non-canonical number systems, such as MSD, have not been proposed because of the computational complexity. Algorithms employing MSD may further reduce the hardware implementation cost of MCM block in digital filters. Park and Kang [PAR01] has attempted to bridge the optimization gap between CSE and GD

algorithms via MSD representation. The algorithm tried to instill the idea of partial sums into MSD coefficients. However, this algorithm included an exhaustive search for all MSD representations to all the coefficients and restricted the values of the partial sums to within those available in the MSD representations. Another interesting algorithm proposed by Yao et. al. [YAO04] makes use of the mutual conversions between SD representations and CSD representation in the process of composing common subexpressions. Actually, Yao's algorithm has partially freed CSD representations to the more versatile SD representations. In the author's opinion, this is likely to be the future research trend in CSE algorithms when more computing power is available. SD representations (including MSD representation) provide a larger scope for finding more opportune common subexpressions for elimination. In [DEM04b], Dempster and Macleod proposed an even larger search space, which does not restrict the representations to have the minimal numbers of nonzero digits. It has been proven by simulation using benchmark filters that it is highly probable to produce solutions with less adders by increasing the numbers of nonzero digits in coefficients representations. However, this method was based on the exhaustive searches for different representations of filter coefficients and it involved enormous trials on every possible combination of the representations of each coefficient. Hence, it is computationally inefficient and not popularly adopted in practice.

Regardless of the number systems, the search for common subexpressions to form the complete set of coefficients with minimal number of adders is the key concept behind the CSE algorithms. For a generic filter specification, unfortunately, there is no obvious answer to the best representation for establishing a well-defined cost function to simultaneously achieve both objective functions of minimal performance error and optimality of VLSI architecture. Due to the complexity of the MCM problem defined in [POT96], there is no exact algorithm that guarantees optimality of the solutions for all kinds of coefficients. Since exact optimal solution is intractable, almost all existing CSE algorithms are heuristic in nature and the solutions generated are possibly suboptimal due to local minima. Many heuristic algorithms [HAR96, MAR02, YAO04] use weight-2 subexpressions as the primitive elements for reuse and then search for higher-weight

common subexpressions from existing lower-weight common subexpressions. However, this process tends to discriminate the higher-weight subexpressions from being selected as their nonzero digits have already been preoccupied by the lower-weight common subexpressions. On the contrary, some algorithms [PAS99, POT96] search for the highest-weight common subexpressions at the outset followed by splitting them into the lower-weight common subexpressions for further sharing. This is to insure that the higher-weight common subexpressions do not always give in to the lower-weight subexpressions when they are overlapped. The experimental results presented later in Section 6.5 show that admission of higher-weight common subexpressions does not always lead to the reduction of total adder cost. Some algorithms [HAR96, POT96] enhance the optimization by including more factors and enlarging the search scope, but the qualities of the solutions improve only marginally. This is because gradient decent approaches, though efficient, are sensitive to the order of execution. There is no feedback and back tracking to allow efficient resubstitution of lower cost common subexpressions in case of conflicts arising from the nullification of potential subexpression elimination caused by early decisions in the execution process.

2.4.3 Graph-Dependence (GD) Algorithms

GD algorithms use integer representation for the filter coefficients [BUL91, DEM95, DEM00, DEM02, GUS02b, KAN01, PAR01] and aim at minimizing the number of partial sums that compose the complete set of coefficients. The notion of primitive operator has been introduced by Bull and Horrocks [BUL91], who are among the pioneers in applying the graph theoretic algorithm to systematically minimize the implementation cost of the multiplication block of the transposed direct form FIR filters. In the BH algorithm [BUL91], a filter is represented as a directed graph whereby partial sums called the fundamentals are symbolically encapsulated in the vertices of the graph and the shift amounts of the partial sums are annotated on the edges. GD algorithms involve the synthesis of a set of minimal cardinality connected graphs from the unity source to the sinks, which are the coefficients to be synthesized. New partial sums are generated one by one from the previously generated partial sums. The filter coefficients

are generated sequentially one after another, employing the existing partial sums. Such primitive operator graph not only reflects the arithmetic relationship between vertices but also the physical delays in integer multiples of clock periods between adjacent vertices. Three main constraints of the original proposed primitive operator graph approach have been identified [BUL91], which have been improved in subsequent research. One of the important modifications made to the basic BH algorithm is the use of subtractors to reduce the number of logic operators needed. This gives yet another strong reason to choose signed number systems. Another two major improvements to the basic BH algorithm are the reduction of each partial sum by factors of 2 until it becomes odd and the order of processing the coefficients. Bull and Horrocks algorithm with the above enhancements is referred to as the BHM algorithm [DEM95]. Based on the concept and data structure of primitive operator graph, a highly regarded algorithm called the n -Dimensional Reduced Adder Graph (RAG- n) algorithm [DEM95] is proposed. The RAG- n algorithm consists of two main parts: the *optimal part* and the *heuristic part*. If one set of coefficients is completely synthesized by the optimal part of the algorithm, minimum adder cost is assured. The main idea is therefore to synthesize as many fundamentals and coefficients as possible with the optimal part of the algorithm. Only when the coefficients cannot be synthesized by the optimal part, the heuristic part will be explored. A new measure of '*adder distance*' is introduced in the heuristic part. The adder distance of a new vertex from an incompletely synthesized graph is the number of extra adders that are needed to generate this vertex from the existing vertices of the incomplete graph. The aim is to find the minimum adder distance between the graph created by the optimal part of the algorithm and the coefficients that have yet to be synthesized. The adder distance needs to be checked by using precomputed lookup tables generated by an exhaustive search of all possible graph topologies. Minimum Adder Graph (MAG) algorithm [DEM94] was typically used to generate the lookup tables. Soon after the introduction of RAG- n algorithm, Dempster and Macleod developed several other algorithms for one-constant [DEM94, GUS02b] and two-constant [DEM05] multiplication problems. The above GD algorithms assumed the use of conventional carry propagation adders (CPAs) for their implementation. Gustafsson et. al. [GUS01b, GUS04] proposed modified algorithms based on RAG- n by considering the number of

carry-saved adders (CSAs) as opposed to the number of CPAs as the criteria for the optimization of fixed-coefficient filters. The use of CSAs speeds up the critical path by avoiding the carry propagations for multi-operand addition except for the final stage. These algorithms are particularly useful for very high-speed and high precision applications. One fundamental issue associated with the GD algorithms is that integer representation makes no assumption on the number format and hence may generate immense possibility for the compositions of coefficients from the intermediate partial sums. As the search space enlarges, the already complex problem becomes daunting with the additional dimension of decision trade-offs. A common way of managing this level of complexity is to use pre-computed lookup table containing the best compositions of partial sums over a range of integer numbers. Since finding the minimum adder distance for an arbitrary number is itself an NP hard problem, there is a limit imposed on the wordlength of the coefficients and the sizes of the lookup tables. This is a major constraint of RAG- n algorithm. In Chapter 3 of this thesis, it is demonstrate that the limit of coefficient wordlength can be lifted by a quasi-minimal adder distance search using the hamming weight pyramid (HWP) [XU04d]. This quasi-minimal adder distance has been sued in the algorithm proposed to improve the utilization rate of the optimal part of RAG- n algorithm. Another problem with the GD algorithms is that they synthesize coefficients sequentially, and greedily reduce the hardware complexity with little or no regard to other optimization criteria. As redundancies among coefficients have not been adequately exploited and the lookup table is designed based on minimal logic complexity with constrained table size, the MCM block optimized by conventional GD algorithms are likely to yield long critical path [KAN01, YAO04], unless some special depth control measure is instilled in the process. The SLBHM and SLRAG- n algorithms proposed by Kang [KAN01] are founded on BHM and RAG- n algorithms by integrating tree reduction method to limit their logic depths. Dempster et. al. [DEM02] tried exhaustively to increase the use of cost-1 partial sums to reduce the logic depth, but the C1 algorithm proposed by them is highly computational intensive. These approaches share a common fundamental idea, which is to convert the serial adding structure of the original GD algorithms to a parallel one. Basically, GD algorithms are restricted by two properties. Firstly, GD algorithms are inherently order-dependent because partial sums are generated

according to the order from which coefficients are processed. From BH to RAG- n algorithms, the basis for the processing order of the coefficients changed from the magnitude of the coefficients to the adder cost, but the choice is heuristic. Another factor is the choice of partial sums. As GD algorithms only consider partial sums from the processed coefficients, to a certain extent it is also vulnerable to local minimum problem. Since RAG- n algorithm uses MAG to guide the selection of minimal-cost partial sums, its memory requirement has restricted the optimality of RAG- n to some limited wordlength coefficients. Furthermore, the optimality is guaranteed only for each individual coefficient but not the filter set as a whole.

2.5 Summary

In this chapter, background knowledge related to the realization of digital FIR filter as well as a comprehensive literature survey on filter synthesis and optimization algorithms are provided.

The fundamentals of digital FIR filters have been introduced. The direct form and transposed direct form filter structures have been reviewed. The linear phase direct form structure has also been presented. To facilitate the explanation of filter coefficient synthesis algorithms, various types of number representations, namely, the two's complement representation, signed digit representation and the CSD/MSD representation are discussed. The latter is commonly used in filter coefficient synthesis and optimization problems as it guarantees the fewest number of nonzero digits and hence lowers the hardware cost.

Some popular filter synthesis algorithms have been introduced. Both the optimal and suboptimal approaches have been discussed. The algorithms pertaining to common subexpression sharing have been studied and this topic will be further discussed in Chapter 4, where a new filter coefficient synthesis algorithm is proposed.

The literature survey also covers the existing algorithms on optimizing the implementation of MCM block in digital FIR filters. The evolution and variants of CSE and GD algorithms as well as the use of different types of common subexpressions have been presented.

Chapter 3

Hamming Weight Pyramid

3.1 Introduction

As the Canonical Signed Digit (CSD) representation is widely used for digital filter coefficients, the study on CSD lead to a new formation of a triangular tree structure for analyzing the hamming weights of CSD numbers. The new structure is named the Hamming Weight Pyramid (HWP). This chapter describes the formation of the HWP, its properties and applications. Hamming weight is defined as the number of nonzero symbols in a symbol sequence. For binary signaling, hamming weight is the number of “1” bits in the binary sequence; and for signed-power-of-two (SPT) numbers, hamming weight is the number of 1 and -1 digits. The hamming weights of some circuit parameters have more often than not, a direct or indirect implication on the implementation complexity. A signed digit (SD) representation is said to be minimum if it possesses the minimal hamming weight. CSD representation, which is a special subset of the minimal signed digit (MSD) representation, is unique in that it has no adjacent nonzero digits in its positional notation. The expression of constants with minimum hamming weight helps reducing the hardware implementation cost and CSD’s unique property facilitates fast search for numerals of matching criteria. In the realization of the MCM blocks of FIR or IIR filters, additions and shifts replace the power hungry and area consuming multipliers. The complexity of the MCM block can be reduced by exploiting the tolerances of the coefficients and knowledge of their hamming weight to eliminate redundant primitive operators corresponding to common subexpressions. Parameters in digital signal processing applications are normally specified in decimal form for precision and to avoid the loss of dynamic range. Therefore, conversion from decimal to CSD is an inevitable step for design automation algorithms that need to exploit the advantages and

conveniences of CSD. Usually, to generate a CSD representation, the decimal number needs to be converted to the binary form first. Direct conversion from decimal to CSD is beneficial and a conversion method is proposed in this chapter.

Since the CSD and MSD representations of a number have the same hamming weight, the same structure for CSD is equally applicable to MSD. Many useful and attractive properties are extracted from the HWP. These properties manifest the significances of HWP and provide new perspectives on the applications of CSD. A reduced HWP is also proposed. It can be constructed easily from a reduced sized lookup table, which is an abstraction of HWP. From the properties of the HWP, a new method to convert decimal number to its CSD format is derived. Its computation complexity is comparable to that of the conversion from decimal number to binary number. Direct conversion from decimal number to CSD form without an intermediate redundant signed-digit number conversion step makes it attractive as the preprocessing step for many design automation algorithms. These algorithms can be benefited from the substitution of integer operations by their CSD forms. The proposed conversion method from the HWP stems a new CSD Tree (CSDT). This CSDT provides a straightforward view of the conversion from decimal number to CSD representation. Besides, with the aid of the HWP, a cost metric is devised for the recently proposed decimal-number-based algorithm, MRAG [XU05c] to extend the word length of the coefficients managed by the n-Dimensional Reduced Adder Graph (RAG-n) algorithm [DEM95].

3.2 Generation and Development of Hamming Weight Pyramid

To reduce the implementation cost of compound arithmetic operations by using simple addition, subtraction and shift operations in conventional binary computer, the 2^{n+1} distinct integers, X in the range of $(-2^{n+1}, 2^{n+1})$ are generally represented with SD representation as follows [GAR65, HWA79]:

$$X = \sum_{i=0}^n a_i 2^i, \quad a_i \in \{0, \pm 1\} \quad (3.1)$$

The hamming weight of the SD is defined as the number of nonzero digits, a_i in (3.1). A CSD representation is a unique SD representation that exhibits the property that at least one of any two consecutive digits must be zero [GAR65, HWA79], i.e.

$$a_i a_{i-1} = 0 \quad \forall i \in (1, n) \quad (3.2)$$

Since negative integers in CSD can be trivially obtained from its positive counterpart by changing the signs of all nonzero digits, it is sufficient to consider only the CSDs of natural numbers without loss of generality. An interesting symmetrical two-dimension ordered sequence is observed on the hamming weights if the CSD is recursively decomposed into a leading nonzero digit, a_L and a smaller CSD, as follows:

$$X = a_L 2^n + \sum_{i=0}^{n-1} a_i 2^i \quad (3.3)$$

Using the first term of (3.3) as pivot and the second term as offset, the natural number sequence, 1, 2, 3 ... in CSD representation can be ordered in an interesting triangular structure. Figure 3.1 (a) shows the Pyramid of the hamming weights of CSD numbers, $2^r + c$, where r and c correspond to the row number and the offset from the center column of the Pyramid, respectively. Figure 3.1 (b) and (c) show the corresponding pyramids of decimal and CSD representations of HWP for ease of exposition. The center column is numbered 0 and the other column numbers are decremented towards its left and incremented towards its right. By reading the weights in a top-down, left-right order, the integers in the HWP correspond to the CSDs of natural number sequence, 1, 2, 3, 4...

		Column c																								
		...	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	...		
Row r	0													1												
	1													1												
	2												2	1	2											
	3											2	2	1	2	2										
	4							3	2	3	2	2	1	2	2	3	2	3								
	5			3	3	2	3	3	3	2	3	2	2	1	2	2	3	2	3	3	3	2	3	3		
...														...												

a) Hamming Weight Pyramid (HWP)

b) Decimal numbers correspondence of the HWP

c) CSD numbers correspondence of the HWP

Figure 3.1 HWP and its corresponding decimal and CSD pyramids

Since the least significant digits (LSDs) of the CSD numbers correspond to even numbered columns are identically zero, their hamming weights can be deduced from their corresponding odd CSD numbers obtained by shifting out the trailing zero digits. By omitting the even numbered columns, an Odd HWP (OHWP) for the hamming weights of CSDs of odd natural number sequence 3, 5, 7, 9 ... is obtained in Figure 3.2. This simplification is of significant interest in digital filter design since only the odd fundamentals of the primitive operator graph [DEM95] need to be considered when it is used to optimize the multiplier block of FIR filters. Comparing with the HWP, the OHWP has no even columns and starts from row 3. In the OHWP, odd number 1 is omitted mainly because it is the only odd number in the even column numbers. 1 is also unique as it is the only odd integer that can be defined by exactly one power-of-two term.

Figure 3.2 Odd Hamming Weight Pyramid (OHWP)

Since both CSD and MSD have the same minimum hamming weight for a given integer, the HWP and the OHWP also represent the hamming weights of MSD numbers. MSD is not unique and can have consecutive nonzero digits. For example, the decimal number 13 has two MSD forms, 1101 and $10\bar{1}01$. $10\bar{1}01$ is a CSD number and its bit pattern exhibits visually appealing characteristic that correlates well with the row and column numbers of HWP. On the contrary, the bit pattern of 1101 has obscure meaning in HWP. Therefore, the following discussions on the HWP emphasize on CSD and some properties are not intended to be used with MSD representations.

3.3 Properties of Hamming Weight Pyramid

The CSD represented numbers, 2^i with the hamming weight of 1 form the middle pillar of the HWP, from which symmetrical number of entries are extended on both sides. The leftmost entry of any arbitrary row represents the hamming weight of a CSD number adjacent to one corresponding to the rightmost entry of the row above it. This formation has led to several interesting properties.

Property 1: The CSD numbers corresponding to row i of the HWP are the range of natural numbers that can be represented with $i+1$ digits and has a leading ‘1’ in the same positions in their CSD representations. For example, the corresponding CSD numbers in row 3 ($r = 3$) of the HWP from left ($c = -2$) to right ($c = 2$) are sequentially $10\bar{1}0$, $100\bar{1}$, 1000, 1001 and 1010 (corresponding to decimal numbers 6, 7, 8, 9 and 10). The centre element in each row of the HWP is 1, which is the only 1 in a row. All hamming weights

in the same column correspond to the CSD represented numbers with their offset terms in (3.3) differ only in the leading nonzero digits, i.e., their offset terms upon further decomposition by (3.3) are identical. For example, the corresponding CSD numbers in column $c = -2$ begin from row 3 are $10\bar{1}0$, $100\bar{1}0$, $1000\bar{1}0$ and so on. Figure 3.1 shows the correspondence between the HWP and the natural numbers and CSD numbers.

Property 2: The number of entries in row i ($i > 0$) of the HWP, denoted as $N(i)$ can be described by any of the following recursive equations:

$$N(i) = N(i-1) + 2N(i-2) \quad (3.4)$$

$$\text{or} \quad N(i) = 2^i - N(i-1) \quad (3.5)$$

where $N(-1) = 0$ and $N(0) = 1$. By unrolling the recursion,

$$\begin{aligned} N(i) &= 2^i - 2^{i-1} + 2^{i-2} - \cdots + (-1)^{i-2} 2^2 + (-1)^{i-1} N(1) \\ &= 2^i - 2^{i-1} + 2^{i-2} - \cdots + (-1)^{i-2} 2^2 + (-1)^{i-1} 2 - (-1)^{i-1} \\ &= 2^i - 2^{i-1} + 2^{i-2} - \cdots + (-1)^{i-2} 2^2 + (-1)^{i-1} 2 + (-1)^i 2^0 \\ &= \sum_{r=0}^i (-1)^{i-r} 2^r \text{ or } \sum_{r=0}^i (-1)^r 2^{i-r} \\ &= \frac{1}{3} (2^{i+1} - (-1)^{i+1}) \end{aligned} \quad (3.6)$$

The expression of (3.6) is obtained from the finite sum of a geometrical progression of $i+1$ terms with a common ratio of $-1/2$ and the first term equals 2^i . $N(i)$ is the number of CSD representations with a leading 1 and have $i+1$ digits.

Property 3: The total number of entries up to and including row i ($i \geq 0$), $S(i)$ is given by:

$$\begin{aligned} S(i) &= \sum_{r=0}^i N(r) = \frac{1}{3} \left(\sum_{r=0}^i 2^{r+1} - \sum_{r=0}^i (-1)^{r+1} \right) \\ &= \frac{1}{3} (2^{i+2} - 2 - \sum_{r=0}^i (-1)^{r+1}) = \left\lfloor \frac{2^{i+2}}{3} \right\rfloor \\ &= \begin{cases} 2N(i) & \text{if } i \text{ is odd} \\ 2N(i)-1 & \text{if } i \text{ is even} \end{cases} \end{aligned} \quad (3.7)$$

In the derivation of (3.7), $\sum_{r=0}^i (-1)^{r+1}$ is equal to 0 if i is odd and -1 if i is even. $S(i)$ is the number of CSD representations with a leading 1 and have not more than $i + 1$ digits. Table 3.1 shows the values of $N(i)$ and $S(i)$ for the first 10 rows of the HWP.

Table 3.1 $N(i)$ and $S(i)$ of HWP

i	0	1	2	3	4	5	6	7	8	9
$N(i)$	1	1	3	5	11	21	43	85	171	341
$S(i)$	1	2	5	10	21	42	85	170	341	682

It is observed that the entries in row i of the HWP are entirely copied into the corresponding columns in row $i + 1$. The core, $C(i)$ of row i is defined as a vector of elements that are copied from row $i - 1$. The remaining elements to the left and right of $C(i)$ are called the left and right wings, respectively. $W_L(i)$ and $W_R(i)$ are the vectors used to denote the order list of elements in the left and right wings, respectively. Based on the above nomenclatures, the following properties are stated:

Property 4: The cardinality of $C(i)$ is given by:

$$|C(i)| = N(i-1) = \sum_{r=0}^{i-1} (-1)^r 2^{i-r-1} \quad (3.8)$$

Property 5: The cardinality of $W_L(i)$, which is equal to that of $W_R(i)$, is given by:

$$|W_L(i)| = |W_R(i)| = \frac{N(i) - N(i-1)}{2} = N(i-2) = \sum_{r=0}^{i-1} (-1)^r 2^{i-r-1} \quad (3.9)$$

Property 6: For all $i \geq 2$,

$$W_L(i) = W_R(i) = C(i-2) + I_{|C(i-2)|} \quad (3.10)$$

where I_n is a $n \times 1$ vector whose elements are all '1's.

Based on the above properties, the rows of the HWP can be progressively generated after initializing the first two trivial rows to 1. Figure 3.3 illustrates the construction of row 4

of the HWP, where the cores and wings are represented by the numbers enclosed in the dashed and solid line blocks, respectively.

Figure 3.3 Generation of the core and wings of row 4

By symmetry, the number of entries towards either sides of the central element in row i is trivially given by $\frac{N(i)-1}{2}$.

Property 7: Let $h(i, j)$ denote the hamming weight appeared in row i , column j of the HWP, then

$$h(i, j) = \begin{cases} h(i-2, j)+1 & \text{for } i \geq 2 \text{ and } \frac{N(i-1)+1}{2} \leq j \leq \frac{N(i)-1}{2} \\ h(i-1, j) & \text{for } i \geq 2 \text{ and } 0 \leq j \leq \frac{N(i-1)-1}{2} \end{cases} \quad (3.11)$$

and $h(0, 0) = h(1, 0) = 1$.

By virtue of the above properties and the symmetry of the HWP, it is sufficient to store only the elements of W_L or W_R . With proper indexing to the vectors, the information embedded in the HWP can be encoded succinctly. This reduces the storage of the HWP to less than a quarter of that required if the entries are sequentially stored along with the row and column indices. Table 3.2 illustrates the compressed format used to store the HWP up to row six. The elements in $W_L(i)$ for $i = 2, 3, \dots$ are stored sequentially in the second row of Table 3.2 while the numbers in the first row are the indices, i to the first element of $W_L(i)$ listed in the second row. The third row is added to indicate the starting column numbers of $W_L(i)$.

Table 3.2 Memory organization of HWP

2	3	4	5					6												
2	2	3	2	3	3	3	2	3	3	4	3	4	3	3	2	3	3	4	3	4
-1	-2	-5	-10					-21												

Property 8: Let $csd(i, j)$ denote the CSD number corresponding to the hamming weight, $h(i, j)$ in the HWP. Then

$$csd(i, j) \ll n = csd(i + n, 2^n \times j) \quad (3.12)$$

where “ $\ll n$ ” means shifting the operand by n bits to the left.

Property 8 follows directly from multiplying both sides of (3.3) by 2^n . Multiplying the pivoting term by 2^n is the same as descending the entry in the HWP by n rows down the centre column while multiplying the offset term by 2^n is equivalent to shifting it outwards by 2^n times. Using Property 8, the CSD numbers differ only in the number of trailing zeros are readily identified. It is worth noting the traces of shifting the CSD numbers in the HWP. The sequences of shifts for $(10\bar{1}, 10\bar{1}0, 10\bar{1}00, 10\bar{1}000)$ and $(1010\bar{1}, 1010\bar{1}0)$ are marked in Figure 3.4.

Figure 3.4 Trace of shifting CSD numbers

Property 8 also provides the basis for the reduction of the HWP as the eliminated CSD entries in the reduced HWP can be retrieved from the shifting property. It should be noted that Properties 2 to 7 are still valid for the OHWP. In addition, the following property is unique to the OHWP.

Property 9: The entries in the OHWP appear in duplets of adjacent identical numbers. The corresponding CSDs of the duplet differ only in the signs of their least significant digits.

The duality property of the OHWP is highlighted in Figure 3.5 by enclosing the duplets in brackets. The first pair of hamming weights corresponds to the CSD numbers $10\bar{1}$ and 101 .

Figure 3.5 Duality property of OHWP

If each duplet of hamming weights is replaced by a single number, the result is an interesting add-one HWP structure as shown in Figure 3.6, which is identical to the original HWP with every entry incremented by one.

Figure 3.6 An add-one HWP generated from OHWP

By subtracting one from every entries of Figure 3.6, the structure of the original HWP is obtained. However, some information on the original CSD numbers is lost in this scaling. In this case, the two least significant digits, $0\bar{1}$ and 01 are pruned. For example, when the duplets of $10\bar{1}$ and 101 in Figure 3.5 are replaced by a single hamming weight of 2 in the first entry of Figure 3.6, the difference between the least significant digits ‘ 01 ’ and ‘ $0\bar{1}$ ’ are not distinguished. If these two digits are shifted out (hence reducing their hamming weights by 1), then the two CSD numbers are identically reduced to 1, which correspond to the first entry in the original HWP and has a hamming weight of $2 - 1 = 1$. In the above example, $10\bar{1}$ and 101 are called the descendents and their shifted version, 1 is called the ancestor. Some ancestor-descendent relationships are marked on the HWP in Figure 3.7 with the arrows pointed from the ancestors to their descendents.

Figure 3.7 Ancestor-descendents relationships in HWP

The above ancestor-descendent relationship is best expressed in the form of a ternary tree where each parent has three children. The left, middle and right children are obtained from their parent by appending to it digit strings of ‘ $0\bar{1}$ ’, ‘ 0 ’ and ‘ 01 ’, respectively. The ternary tree to generate the CSD numbers iteratively is shown in Figure 3.8.

Figure 3.8 Ternary tree of CSD numbers

3.4 Applications of HWP

The HWP uncovers the intriguing relationship between decimal numbers and their CSD representations. A direct conversion algorithm can then be readily developed based on the structural property of the HWP. The HWP can also be used to replace the pre-computed lookup table embedded in the Modified n-Dimensional Reduced Adder Graph algorithm (MRAG) to lift the restriction on the range of coefficients. Without considering the sharing of common subexpressions, the adder cost for multiplierless implementation of digital filters can be benchmarked by the number of SPT terms used to represent their coefficient sets. The search for proximal CSDs to a real-valued coefficient with a specific hamming weight limit is therefore a common pursuit in the synthesis of optimal filter coefficients. In conjunction with common subexpression elimination, the HWP can also be modified to the Common Subexpression based HWP (CS-HWP) by treating the two most frequently encountered common subexpressions, 101 and $10\bar{1}$ specially. The CS-HWP is introduced in the next chapter, while presents a new low complexity FIR filter design algorithm that accounts for the reusability of the above two common subexpressions in the coefficient synthesis process.

3.4.1 Canonical Signed Digit Conversion algorithm

The HWP provides a new perspective to the synthesis of CSD numbers. With the help of Property 3 and (3.3), the following algorithm is developed for the conversion of a decimal number, d to its CSD representation, csd . The pseudo code in Figure 3.9 describes the algorithm.

```

1: Dec2CSD( $d$ ) {
2:    $csd = 00...0$ ;
3:    $sign = 1$ ;
4:   while ( $d \neq 0$ ) {
5:      $i = \text{row\_number}(d)$ ;
6:      $csd[i] = sign$ ;
7:      $d = d - 2^i$ ;
8:     if ( $d < 0$ ) {
9:        $sign = -sign$ ;
10:       $d = -d$ ; }
11:  return  $csd$ ; }
```

Figure 3.9 Pseudo code of new conversion algorithm

In Figure 3.9, the function $row_number(d)$ searches for the row index, i such that $S(i - 1) < d \leq S(i)$. The conversion requires h iterations to complete where h is the hamming weight of csd . Thus, the average number of iterations for an n -digit CSD number is approximately $n / 3$. If a lookup table is used to search for the row index in $row_number(d)$ (line 5 of Figure 3.9), the size of the table is proportional to the word length, as observed from Table 3.1 and Property 3. In line 5, the function $row_number(d)$ can be represented by $\left\lceil \log_2 \left(\frac{4d}{3} \right) \right\rceil$ if the lookup table is to be avoided. The search range is progressively narrowed down as the offset term becomes smaller. The worst case computational complexity of the above conversion algorithm is thus bound by $O(n^2)$. The algorithm can be easily extended to the conversion from 2's complement number for hardware implementation.

As an example, the traces of converting the decimal number 45 to its CSD form $10\bar{1}0\bar{1}01$ are shown in Figure 3.10. Let csd_i denote the i -th digit of the final CSD number. In the first iteration, $i = 6$, $csd_6 = sign = 1$ and $d = 45 - 64 = -19$. Since $d < 0$, $sign = -1$ and $d = 19$. In the second iteration, $i = 4$, $csd_4 = \bar{1}$ and $d = 19 - 16 = 3$. In the third iteration, $i = 2$, $csd_2 = \bar{1}$ and $d = 3 - 4 = -1$. Since $d < 0$, $sign = -sign = 1$ and $d = 1$. In the final iteration, $i = 0$, $csd_0 = 1$ and $d = 1 - 1 = 0$.

Figure 3.10 Traces of decimal to CSD number conversion from HWP

The way the HWP is traversed to obtain the CSD representation spins off yet another structure of CSD tree (CSDT) shown in Figure 3.11. In the CSDT, each node is

designated a value from the three legitimate digits $\{0, 1, \bar{1}\}$. The root of the CSDT corresponds to the most significant digit of a CSD number. The height of the tree is determined by the word length of the CSD numbers of interest. Except the leaf nodes, which represent the least significant digits of the CSD numbers, each non-terminal node can have either one child or three children nodes. A nonzero valued parent node can have only one child node which has a node value of 0. A zero valued parent node will have three children, each has a unique node value. The children are ordered according to their node values, i.e., $\bar{1}$ node precedes 0 node then 1 node. This construction of the CSDT conforms to (2), which is the fundamental criterion of the CSD representation. Each node of the CSDT represents a valid CSD number. The CSD number represented by any node can be obtained by a depth first visit of the node values from the root to that node. A breadth first traversal of the CSDT lists its implicitly embedded CSD numbers in natural number order. For example, the first element at level 3, denoted as CSDT(3, 1), represents the natural number 3. By a depth first visit from the root to this node, the ordered node values of the designated path are 1, 0, $\bar{1}$, which gives the exact CSD representation of the decimal number 3. Its hamming weight, 2, is given by the element in the same position of HWP, as shown in Figure 3.11.

Figure 3.11 CSD Tree

Due to the succinctness of the CSDT, the node values can be compactly stored in a small lookup table. Similar decoding logic as that of the HWP can be used to index to the correct position of CSDT for a given number. Therefore, with the aid of the CSDT, only the position of the number in the CSDT is needed and the CSD representation can be

directly read from the CSDT once its position is determined. Due to the similarity in their structures, the CSDT shares many properties of the HWP.

3.4.2 Modified RAG- n Algorithm

Among all the optimization algorithms applied to the synthesis of digital filter coefficients for MCM circuits, the solutions produced by the n -Dimensional Reduced Adder Graph (RAG- n) [DEM95] usually have the least logic complexity. One of the major contributing factors RAG- n possessed over similar algorithms is the concept of adder cost, which is the least number of adders used to implement a single coefficient. To obtain the optimal cost of each coefficient, a cost lookup table is first generated from the minimized adder graph (MAG) [DEM94]. MAG is an exhaustive algorithm and the range of coefficients that can be synthesized is limited by the size of the lookup table.

A modified RAG- n algorithm, MRAG was developed in this work to extend the capability range of RAG- n . The algorithm operates on the primitive operator graph [BUL91]. As the aim of this section is to highlight the use of the HWP and the CSDT in MRAG, only relevant algorithmic description pertaining to the use of the HWP is presented. In MRAG, the cost referred to by the RAG- n is replaced by the more generally used hamming weight. The cost metric is used in two parts of the RAG- n algorithm. In the optimal part, the cost is used to dismiss the repeated coefficients and coefficients of cost-0 and cost-1. In the heuristic part, when there is a tie in the synthesis of the coefficients, the coefficient with the lowest cost is arbitrarily chosen. As the generation of optimal single coefficient adder cost by the MAG algorithm is complicated and restrictive, the hamming weight is used in place of the adder cost. In MRAG, the hamming weights are used in the two corresponding parts as follows. In the construction of the odd fundamental graph, for any number with cost less than 3, its hamming weight, which is the number of nonzero digits in its CSD representation, is one more than the cost. This is because when there are less than four nonzero digits in a number, it is virtually impossible to realize this single coefficient with the minimum number of adders by eliminating common subexpressions. In the heuristic part, when no fundamental is added

to the graph set, the least hamming weight coefficient is subjected to the BHM [DEM95] algorithm for decomposition. The pseudo code of the proposed MRAG is shown in Figure 3.12.

```

1: C: coefficient set  $\{C_1, C_2, \dots, C_N\}$ 
2: reduce  $C_i$  to odd fundamentals;
3: store only the unique fundamental in the incomplete set,  $IS_i$ ;
4: for( $i = 1..|IS|$ ) {
5:    $HW_i = \mathbf{HWP}(IS_i)$ ;
6:   if ( $HW_i \leq 1$ )  $IS = IS - IS_i$ ; }
7: sort  $IS_i \in IS$  in ascending order of hamming weight;
8: Initialize graph set,  $GS = \phi$ ;
9: for( $i = 1..|IS|$ ) {
10:  if ( $HW_i == 2$ ) {
11:     $IS = IS - IS_i$ ;
12:     $GS = GS + IS_i$ ; } }
13:  $newlast = 1$ ;
14:  $newfnw = 0$ ;
15: while( $newlast \neq newfnw$ ) {
16:   $newlast = 0$ ;
17:  for( $i = 1..|GS|$ )
18:    for( $j = i..|GS|$ )
19:      for( $k = 1..|IS|$ )
20:        if( $\mathbf{CSDT}(GS_i) + \mathbf{CSDT}(GS_j) == IS_k$ ) {
21:           $IS = IS - IS_k$ ;
22:           $GS = GS + IS_k$ ;
23:           $newlast = newfnw$ ;
24:           $newfnw = IS_k$ ; } }
25: for( $k = 1..|IS|$ )
26:  for( $i = 1..|GS|$ ) {
27:     $difference = |IS_k \pm GS_i|$ ;
28:    if( $\mathbf{HWP}(difference) == 2$ ) {
29:       $IS = IS - IS_k$ ;
30:       $GS = GS + difference$ ;
31:       $GS = GS + IS_i$ ;
32:      go to line 13; }
33:    for( $j = i..|GS|$ ) {
34:       $difference = |IS_k \pm GS_i \pm GS_j|$ ;
35:      if( $\mathbf{HWP}(difference) == 1$ ) {
36:         $IS = IS - IS_k$ ;
37:         $GS = GS + difference$ ;
38:         $GS = GS + IS_i$ ;
39:        go to line 13; } } }
40:  $IS = IS - IS_i$ ;
41:  $GS = GS + \mathbf{BHM}(IS_i)$ ;
42: go to line 13;
43: return  $GS$ ;

```

Figure 3.12 Pseudo code of the MRAG algorithm

In Figure 3.12, IS and GS are the incomplete set and graph set as defined in [DEM95]. $HWP(n)$ is a function to evaluate the hamming weight of a decimal number n through the HWP. $CSDT(a)$ returns the set of power-of-two multiples of a which can be easily obtained from the CSDT. $BHM(m)$ employs the BHM algorithm to synthesize the selected coefficient m by exploiting the existing fundamentals and returns the computed fundamentals of the coefficient. Each time a new coefficient is synthesized, IS is checked to see if it is empty. If so, the process ends.

The generation of the hamming weight for CSD representation through the HWP is very simple and does not require an exhaustive evaluation of all possible primitive graphs for any number within the range of interest. The storage required by the HWP is much smaller compared with the cost lookup table. For decimal numbers range from 1 to 4096, the highest weight is 7. Hence the entries of HWP are decimal numbers range from 0 to 7. 3 bits are needed for each entry. The total storage needed to index into the hamming weights for the HWP is less than 1024 bytes, whereas the pre-computed cost lookup table requires several times of 4096 bytes of storage. The use of HWP here simplifies the cost evaluation and extends the range of the coefficients that can be synthesized by the RAG- n algorithm.

3.5 Summary

CSD representation plays a major role in MCM problems described in Chapter 2 and optimization algorithms for FIR and IIR filters. A good understanding of the specific properties exhibited by these numbers and their appropriate exploitation can help to improve the quality and efficiency of the existing algorithms, or even lead to new concepts and new algorithms.

In this work, a regularly structured HWP is proposed to assist the exploration the interesting properties of the CSD numbers and the essential relationship between the natural numbers and their hamming weights in the CSD and MSD representation. This formulation gives the motivation to develop a simple and efficient method of converting

decimal number to its CSD form, and a more intuitive and succinct structure for representing all CSD numbers of a given word length. It is demonstrated that the HWP can be employed to calculate the hamming weight in place of the complicated cost metric used by the RAG- n algorithm for implementing digital filters with the minimal logic complexity. It is envisaged that other applications that employ CSD form can be benefited from further exploitation of the derived properties of the HWP. For example, ternary value finite automata [HAS96] and filter coefficient synthesis [CHE99] can be spawn from locating the constant coefficients and through direct manipulation on the proposed HWP. Besides, the HWP is useful for filter coefficient synthesis. The reduction in adder cost associated with the reuse of common subexpressions is factored in a modified HWP presented in the next Chapter. This modified HWP, CS-HWP, is used to guide the synthesis of the finite wordlength filter coefficients with the minimal cost of implementation subjected to some maximal permissible frequency response error.

Chapter 4

Filter Synthesis Algorithm

4.1 Introduction

Canonical signed digit (CSD) representation studied in the previous chapter has been one of the most widely adopted number representations for fixed point coefficients to minimize the implementation cost of high-speed finite-impulse response (FIR) digital filter. For many years, the design of FIR filter has been focused on how to synthesize a set of filter coefficients from its frequency response specification with the least number of Signed-Power-of-Two (SPT) terms. The number of SPT terms of a coefficient is in essence, equivalent to the hamming weight of the coefficient in signed digit (SD) representation. It is therefore self-evident that CSD stands out as the most popular and extensively studied number representation for digital filter synthesis algorithms. In Chapter 3, a Hamming Weight Pyramid (HWP) is presented for CSD numbers and its application to assist the adder cost estimation for RAG-n algorithm is demonstrated. The adder cost assessment, however, has not taken into account the possible adder reduction by common subexpression elimination (CSE).

In this chapter, a Common Subexpression based Hamming Weight Pyramid (CS-HWP) that possesses the same skeleton as the HWP is proposed and described. It is used to seek for low cost candidate coefficients with preferential consideration to numerals with shared common subexpressions. To account for the physical adder cost oriented Common Signed-Power-of-Two (CSPT) terms, the original design problem statement based on the minimization of SPT terms has to be reformulated. With the CS-HWP, a new efficient algorithm is proposed for the design of low-complexity FIR filters with resource sharing in the transposed direct form filter structure. The filter design solution is directly coupled

with the actual circuit optimization criterion right from the inception of the magnitude response specification. One main redundancy to be eliminated in the circuit optimization process is manifested as the exploitation of adder reusability for common subexpressions. The minimization of CSPT terms considered in the algorithm described in this chapter addresses the simultaneous optimization of the reusability of adders for the two most frequently used common subexpressions as well as the minimization of adders needed for the remaining SPT terms. This multiple constraint optimization is achieved in two stages. In the first stage, CSPT terms in the vicinity of the scaled and rounded CSD coefficients are allocated to obtain a CSD coefficient set with the total number of CSPT terms not exceeding the initial coefficient set. The balanced Normalized Peak Ripple Magnitude (NPRM) due to the quantization error is fulfilled in the second stage by a bivariate local search method. Experimental results on widely used benchmark design examples demonstrate that the proposed algorithm is capable of synthesizing low-complexity FIR filters with the least CSPT terms compared with the existing filter synthesis algorithms. The use of common subexpressions in the coefficient synthesis algorithm has proven to be effective to design multiplierless FIR filters with reduced hardware cost at no penalty to their throughput rates.

This chapter is organized as follows: Section 4.2 justifies the consideration of the common subexpressions, 101 and $10\bar{1}$ for the proposed filter coefficient synthesis algorithm and formulates a new model for the filter coefficient synthesis problem associated with the consideration of these common subexpressions. Section 4.3 introduces the notion of CS-HWP as a search tool for low cost CSD coefficients embody the common subexpressions, 101 and $10\bar{1}$. The proposed two-stage integrated approach for the design of low-complexity digital filter is presented in Section 4.4. The solutions of the proposed algorithm are compared with those synthesized by other algorithms through two well-known design examples. Additional results are also provided and discussed in Section 4.5. The chapter is summarized in Section 4.6.

4.2 Coefficient Synthesis with Common Subexpressions

Among the approaches proposed in the literature for fixed point digital filter implementation, common subexpression elimination (CSE) techniques [HAR96, MAR02, PAR01, POT96, VIO03] have received considerable attention. The optimality of CSE algorithms relies on the detection of good common subexpressions to eliminate redundant computations.

Example 4.1: A small coefficient set is used to illustrate the effects of sharing the common subexpressions 101 and $10\bar{1}$ in Figure 4.1. The five coefficients of the filter are represented in 10 signed digit fixed-point CSD format as, $\{h\} = \{-2^{-5} - 2^{-8} + 2^{-10}, 2^{-4} - 2^{-6} + 2^{-8} + 2^{-10}, -2^{-4} - 2^{-7} + 2^{-9}, -2^{-3} - 2^{-5} + 2^{-9}, 2^{-1} - 2^{-3} + 2^{-5} + 2^{-7} - 2^{-10}\}$.

Figure 4.1 Common subexpression eliminations

The CSD coefficients are normalized to the unit of least significant position (*ulp*), 2^{-10} in this example, so that their *ulp* are all weighted 1. The normalized output of the filter can be expressed as

$$\begin{aligned}
 y &= -x \ll 5 - x \ll 2 + x \\
 &+ x[-1] \ll 6 - x[-1] \ll 4 + x[-1] \ll 2 + x[-1] \\
 &- x[-2] \ll 6 - x[-2] \ll 3 + x[-2] \ll 1 \\
 &- x[-3] \ll 7 - x[-3] \ll 5 + x[-3] \ll 1 \\
 &+ x[-4] \ll 9 - x[-4] \ll 7 + x[-4] \ll 5 + x[-4] \ll 3 - x[-4]
 \end{aligned} \tag{4.1}$$

where x is the input, $x[-i]$ denotes the value of x after i sample delays and $\ll j$ the left shift of j digits. There are 18 nonzero digits in this example and 17 adders are needed if common subexpressions are not considered. The output can be denormalized by hardwired shifts without requiring any adder.

By extracting the common subexpressions, $x_{101} = x \ll 2 + x$ and $x_{10\bar{1}} = x \ll 2 - x$ with their combined reuse frequency of 7, (4.1) becomes:

$$\begin{aligned}
 y = & -x \ll 5 - x_{10\bar{1}} \\
 & + x_{10\bar{1}}[-1] \ll 4 + x_{101} \\
 & - x[-2] \ll 6 - x_{10\bar{1}}[-2] \ll 1 \\
 & - x_{101}[-3] \ll 5 + x[-3] \ll 1 \\
 & + x_{10\bar{1}}[-4] \ll 7 + x_{101}[-4] \ll 3 - x[-4]
 \end{aligned} \tag{4.2}$$

In Figure 4.1, the pairs of connected digits are common subexpressions. The same type of common subexpressions needs to be generated once. The benefit of sharing common subexpressions is obvious in this small example. 5 adders are saved, which is 29% less than the direct CSD implementation.

From the above illustration, any replicated cell pattern in a coefficient array can be shared. Since adders and subtractors are assumed to be equivalent, subexpressions with identical pattern but differ only in the signs of every nonzero digit consistently are also reusable. Hartley [HAR96] has shown that substantial gain in circuit area can be achieved very easily by finding the occurrences of only the common subexpressions, 101 and $10\bar{1}$. Asymptotically an n -bit CSD number can be broken down into $n/18 + O(1)$ pairs of $10\bar{1}$ subexpression, $n/18 + O(1)$ pairs of 101 subexpression, and $n/9 + O(1)$ isolated 1 or $\bar{1}$. Referring to a $10\bar{1}$ pair, a 101 pair, or an otherwise isolated nonzero digit as a term, it can be shown that the asymptotic expected number of terms in an n -bit CSD number is $2(n+1)/9$. This represents a 33% saving compared with the total number of nonzero digits, which is equal to $(3n+1)/9$.

In [CHE99], a transposed direct form filter structure (shown in Figure 2.2) with carry-save arithmetic is used to exploit the CSD coefficients for speeding up the coefficient multiplication-accumulation to a few full adder delays without pipelining. As the common subexpressions need to be generated separately in one stage and disseminated to the taps where they are shared, restricting the number of nonzero digits and the bit width of the common subexpressions from appearing in the initial set will prevent the common subexpression generator from becoming the performance bottleneck. By considering the shortest two nonzero digit common subexpressions, 101 and $10\bar{1}$, the operand width and hence the area and time complexity of the adder used for their generation are reduced. Many other filter synthesis algorithms [YAO01], including the optimal MILP method [GUS02c], limit the maximum number of SPT terms for each coefficient in order to limit the logic depth of the filter so as to accelerate the computation rate. The advantage of having filter coefficients with many shared common subexpressions is that under the same limit of adders allowed for each tap, effectively more SPT terms can be assigned to a coefficient to improve the coefficient precision thereby leading to a better approximation to the desired frequency response characteristics. Neglecting other common subexpressions also greatly reduces the complexity of the search scope.

Unfortunately, the minimization criterion, J of (2.17) adopted by most filter coefficient synthesis algorithms does not consider the possibility of sharing common subexpressions to reduce the adder cost. By extracting the common subexpressions, $x_{101} = x \ll 2 + x$ and $x_{10\bar{1}} = x \ll 2 - x$ from (2.16), the general expression for $h_{spt}(i)$ can be rewritten as:

$$h_{cspt}(i) = \sum_{k=1}^{L_{101,i}} cs_{i,k}^{101} \left(2^{-p_{i,k}^{101}+2} + 2^{-p_{i,k}^{101}} \right) + \sum_{k=1}^{L_{10\bar{1},i}} cs_{i,k}^{10\bar{1}} \left(2^{-p_{i,k}^{10\bar{1}}+2} - 2^{-p_{i,k}^{10\bar{1}}} \right) + \sum_{k=0}^{L_i'} s'_{i,k} 2^{-p'_{i,k}} \quad (4.3)$$

where $cs_{i,k}^{101}, cs_{i,k}^{10\bar{1}} \in \{1, -1\}$ are the signs of the common subexpressions, 101 and $10\bar{1}$ present in the i -th coefficient with its least significant digit located at positions $p_{i,k}^{101}$ and $p_{i,k}^{10\bar{1}}$, respectively. $s'_{i,k} \in \{1, \bar{1}\}$ are isolated SPT terms (not belong to any selected common subexpression) in the i -th coefficient located at digit position $p'_{i,k}$. $p_{i,k}^{101}, p_{i,k}^{10\bar{1}}$ and $p'_{i,k} \in \{1, 2, \dots, B\}$ are for a B -bit coefficient. $L_{101,i}$ and $L_{10\bar{1},i}$ are the number of

common subexpressions of types 101 and $10\bar{1}$, respectively. $L_i' = L_i - 2(L_{101,i} + L_{10\bar{1},i})$ is the number of SPT terms of the i -th coefficient that are not present in any common subexpression, while L_i is the total number of SPT terms.

An adder is required to generate each of the common subexpressions 101 and $10\bar{1}$ once before they can be shared among the coefficients that embrace them. Each of the isolated SPT terms which are not part of a common subexpression requires one adder. Therefore, the total number of adders required to implement the multiplication block of a transposed direct form filter is bounded by:

$$Cost \leq 2 + \sum_{i=1}^N (L_i' + L_{101,i} + L_{10\bar{1},i}) = 2 + \sum_{i=1}^N L_i - \sum_{i=1}^N (L_{101,i} + L_{10\bar{1},i}) \quad (4.4)$$

The equality of (4.4) happens when each of 101 and $10\bar{1}$ occurs at least twice. The adder cost is minimized by maximizing the values of $L_{101,i}$ and $L_{10\bar{1},i}$ bounded by the allowable number of SPT terms $\sum_{i=1}^N L_i$ of the coefficient set. Therefore, the objective of the proposed SPT coefficient synthesis algorithm for low complexity FIR filters is recasted to that of reducing the combined number of common subexpressions and non-sharable SPT terms, L' in each coefficient

$$J' = \sum_{i=1}^N L_i' + L_{101,i} + L_{10\bar{1},i} \quad (4.5)$$

subjected to the minimal NPRM constraint of the desired amplitude response specification.

4.3 The Proposed Algorithm

As the desired objective function of minimizing the NPRM is nonlinear [LIM90], heuristic optimization of multiple constraints in a single run is susceptible to the local minimum problem due to poor initial condition. If a reasonable upper bound on the CSPT (i.e., one that considers common subexpression as a single super SPT, to be defined later in this section) terms based on the rounded CSD coefficients is imposed, a quality initial CSD coefficient set can be obtained. This preliminary coefficient set may not meet the

desired amplitude response specification. Its NPRM can be further optimized until the specification is met by slightly relaxing the allowable number of CSPT terms. These two processes require the location of CSD numbers with specific number of CSPT terms or the determination of the number of CSPT terms of a specific CSD number. Therefore, a search tool is devised for the CSD numbers and the subexpressions, 101 and $10\bar{1}$, are treated specially.

The HWP, shown in Figure 3.1 (a), can be served as a useful search tool for the nearest neighbours to a CSD number with similar or lower hamming weight. Its corresponding pyramid of CSD representations is shown in Figure 3.1 (c). By reading the weights in a top-down and left-right order, the integers in the pyramid correspond to the CSD representations of natural number sequence. The CSD number in row r and offset c from the center column represents nature number $2^r + c$. In order to account for the saving of adders by considering 101 and $10\bar{1}$ as common subexpressions, the HWP needs to be revamped so that it can be useful for seeking nearest CSD numbers with equal or lower adder cost. Based on the adder cost of (4.4), it would be logical to reduce the hamming weight by one for each non-overlapping string of either 101 or $10\bar{1}$ encountered in a CSD number. The resultant HWP is shown in Figure 4.2 and it is called CS-HWP where CS is the abbreviation for common subexpression. For example, the CSD number of decimal 19 is $1010\bar{1}$. It corresponds to the entries in row 4 column 3 of CS-HWP. Its hamming weight is 3 in HWP and can be easily obtained in Figure 3.1(a) but is reduced to 2 in Figure 4.2 due to the presence of subexpression either 101 or $10\bar{1}$ (but not both). To avoid ambiguity, the SPT terms with weights corresponding to those in the CS-HWP are referred to as CSPT terms. Each non-overlapping string of 101 and $10\bar{1}$ is treated as a single CSPT term.

Figure 4.2 CS-HWP

In what follows, a two-stage algorithm for the design of low-complexity FIR filter is presented. In the first stage, a set of CSD coefficients in the continuum of the rounded CSD coefficients with minimal filter response error and yet contains CSPT terms not more than the rounded CSD coefficient set is generated. The balanced NPRM budget to the desired response due to the quantization error of the obtained CSD coefficient set is compensated in the second stage with a reduction or minimal increment to the total number of CSPT terms. The first stage is named the *CSPT Coefficient Allocation Stage* and the second stage is named the *Quantization Compensation Stage*.

4.3.1 CSPT Coefficient Allocation

In [LIM83], Lim and Parker mentioned that for a minimax design the optimal coefficient set is usually not the one obtained by rounding but the optimal CSD coefficients do lie near the rounded ones. During the experiments, it is found that the rounding process has a strong influence on the filter frequency response. Rounding a real-value coefficient set fulfilling an amplitude response specification to its finite wordlength CSD coefficients is quite likely to fail under the same specification. The aim of this allocation stage is to search for a good initial CSD coefficient set with reduced number of CSPT terms by considering common subexpressions.

First, the real-value coefficient set $\{h\}$ fulfilling the frequency response specification is quantized to the desired wordlength in CSD representation. From the rounded CSD coefficient set $\{h_{csd}\}$, new CSD coefficients are added to expand the candidate pool (CP)

by finding the nearest CSD coefficients to each $h_{csd}(i)$ for $i = 1, 2, \dots, N$ with the number of CSPT terms equal to or smaller than those of $h_{csd}(i)$. With the help of the CS-HWP, the required nearest CSD coefficients can be easily found by searching from two sides of the original coefficient location in the CS-HWP for entries with equal or lower number of CSPT terms. Let the coefficients in the CP be designated as $h_{cspt}(i)(j)$ for $i = 1, 2, \dots, N, j = 1, 2, \dots, M$. $M = 2ncspt(i)$ where $ncspt(i)$ is the number of CSPT terms of $h_{csd}(i)$. In other words, one nearest CSD number with equal number of CSPT terms and two nearest CSD numbers with smaller number of CSPT terms are sought for each coefficient $h_{csd}(i)$. Thus, for a 4-CSPT-term coefficient, its two nearest 4-CSPT-term (including itself), 3-CSPT-term, 2-CSPT-term and 1-CSPT-term numbers are added to the CP. It is found from experimentation that this number of nearest CSD coefficients is adequate for most amplitude response specifications. By having a larger value of M , more coefficients with larger quantization errors will be included but these additional CSD coefficients are rarely selected eventually as the achievable NPRM error has already reached its margin of diminishing return. Even if some of them have been included in the final solution of this stage, the improvement they made to the NPRM of the solution is insignificant to warrant the extra computation.

Next, a preliminary minimal filter coefficient set, $\{h_p\}$ is established by setting all its coefficient values to zero. The initial coefficients of $\{h_p\}$ either remain as zero or are gradually substituted by the coefficients of the CP, $\{h_{cspt}\}$, in an iterative process to reduce the NPRM of $\{h_p\}$. Each iteration increments the total number of CSPT terms of $\{h_p\}$ by one. At any iteration r , a number of coefficient sets can be formed by replacing an entry in the current $\{h_p\}$ by one from the CP such that the total number of CSPT terms of the coefficient set so formed is equal to r . The best coefficient set with the lowest NPRM error from among the eligible sets is selected to replace $\{h_p\}$. At the end of each iteration, all candidate coefficients with the same number of CSPT terms from the same tap as the substituting coefficient, including the substituting coefficient itself, are deleted from the CP. The process continues until the CP is exhausted.

This search for better quality CSPT terms for each coefficient is in congruence with the practice of many popular search methods in filter synthesis [AIT93, CHE99, SHA91] except that common subexpressions have been considered and the proposed approach is more elegant and efficient. In this process, the NPRM may not be monotonically decreasing although the overall NPRM is trending down. To avoid stuck at local minimum, temporary increase in NPRM is allowed from one iteration to the next, but a record of the minimum NPRM, $NPRM_{min}$ and its corresponding coefficient set $\{h_{pmin}\}$ is kept. Upon the termination of CSPT coefficient allocation stage, the coefficient set $\{h_{pmin}\}$ with the minimum NPRM throughout the stage is selected and used for the second stage. This stage is highly efficient because $\{h_p\}$ is small and the coefficients in the CP have been prudently chosen with the aid of the CS-HWP. The computation time is $O(MN/2)$, which is well bounded by the cardinality of $|h_{cspt}|$.

The size of the CP marks the frequency characteristics of initial coefficient set for the second stage. Hence, an error margin that is small enough to ensure that the second stage could converge to the desired NPRM but sufficiently large should be defined so that the first stage does not incur excessive adder cost. Usually, there is no deterministic answer to what constitutes the best starting coefficient set for the local search. Chen and Willson [CHE99] allocated up to about 75% of the SPT terms in the first pass of local search without having an adverse effect on the final result. In contrary to his heuristic local search methods, the criterion for the initial coefficient set is defined based on the mediation of amplitude response specification and the number of CSPT terms. It is normal that the $\{h_{pmin}\}$ obtained in the first stage based on the rounded CSD coefficients will not meet the NPRM requirement. Therefore, an adaptive quantization compensation is applied in the second stage based on the balanced NPRM budget while suppressing the growth of the number of CSPT terms. The amount of compensation needed and hence the effort required by the optimization method depends on the quality of its preliminary coefficient set. A sensible NPRM budget can be empirically determined in the first stage. Given the trade-off between the number of CSPT terms and the acceptable quantization error, high adder cost is avoided at this stage by setting an upper bound on the maximum number of CSPT terms to that of the rounded coefficient set and the size of the CP is

further limited to MN to reduce inefficient computation for an insignificant improvement in NPRM.

4.3.2 Quantization Compensation

In the second stage, the frequency response characteristic of the optimal coefficient set $\{h_{pmin}\}$ with $NPRM_{min}$ is further improved by modifying the least significant digits of the coefficients until the desired NPRM has been reached. The optimization of frequency response shall be accomplished with little or no penalty to the adder cost. In fact, with the consideration of CSPT terms, the number of adders may even be reduced from the primitive set $\{h_{pmin}\}$ upon achieving the desired amplitude response specification. There are two main concerns in this stage: the definition of neighborhood and the search strategy.

The neighborhood defines a boundary within which small discrete SPT terms are introduced to adjust the coefficient values to compensate for the frequency response error. It determines the degree of freedom of the search procedure. A larger neighborhood provides a better chance of finding the optimal coefficient set, but it also leads to a longer computation time. Earlier researchers usually define the neighborhood to be wordlength independent. Samueli [SAM90] constructed the neighborhood by varying the rounded filter coefficients by very small quantization steps and the chance of finding quality solutions has been slim. In Chen and Willson's approach [CHE99], the neighborhood is made up of the last two Least Significant Digits (LSDs), which form the candidate pool along with those SPT terms that are present in the rounded coefficients but not in the initial set. The LSDs are essentially SPT terms, expressible in the form of $\pm 2^{-m}$, where m denotes the positions of the LSDs. It is found that this definition of neighborhood works fine with short wordlength coefficients, but it restricts the search space for long wordlength coefficients. For short wordlength coefficients, allowing too many LSDs may deficit the purpose of having local search as opposed to exhaustive search. However, there is no reason to constrain the search space to merely the last two LSDs for long wordlength coefficients. For example, for wordlength of 10 bits, the neighborhood

defined by [CHE99] is a circumference of radius of 2^{-9} around the preliminary coefficient set. This is sufficient for many coefficient sets but deficient for some other situations. If the desired wordlength is 14 bits, the neighborhood will be a circumference of radius of 2^{-13} . This search space is too trivial to improve the quality of solution. In the proposed method, the number of LSDs for the neighborhood is wordlength dependent. More LSDs are chosen for longer wordlength to ensure that the search space is sufficiently large for optimization but small enough to maintain the efficiency of the algorithm. The proposed algorithm followed the statistical adaptive neighborhood provided by Ye, et. al. [YE04]. The number of LSDs is determined empirically. To find the suitable neighborhood size, filters of different coefficient wordlengths have been tested, each optimized with different neighborhood sizes, from 2 LSDs to 5 LSDs. It is found that for some filters with longer coefficient wordlength, the search process converges faster when the neighborhood size reaches a critical number of LSDs than when a smaller search space is used. There are also cases where no solution can be found or the algorithm fails to converge when the size of the neighborhood is too small. Table 4.1 lists all LSDs from $\pm 2^{-m}$ to $\pm 2^{-B}$ in the circumference of critical radius m , which is determined empirically for coefficients of wordlength, B for $B = 8$ to 14. The neighborhoods in Table 4.1 are presented in discrete SPT terms Δq , and the maximum absolute quantization step, Δq_{max} . The neighborhood for coefficients of wordlengths 8 to 14 bits is considered, as it is found that the wordlengths of filter coefficients in the range of 8 to 14 bits are sufficient to meet most filter specifications [SAM90].

Table 4.1 Adaptive neighborhood for quantization error compensation

WORDLENGTH B	DISCRETE SPT TERMS Δq	Δq_{MAX}
8	$\pm 2^{-7}, \pm 2^{-8}$	2^{-7}
9	$\pm 2^{-8}, \pm 2^{-9}$	2^{-8}
10	$\pm 2^{-8}, \pm 2^{-9}, \pm 2^{-10}$	2^{-8}
11	$\pm 2^{-9}, \pm 2^{-10}, \pm 2^{-11}$	2^{-9}
12	$\pm 2^{-9}, \pm 2^{-10}, \pm 2^{-11}, \pm 2^{-12}$	2^{-9}
13	$\pm 2^{-10}, \pm 2^{-11}, \pm 2^{-12}, \pm 2^{-13}$	2^{-10}
14	$\pm 2^{-11}, \pm 2^{-12}, \pm 2^{-13}, \pm 2^{-14}$	2^{-11}

The filter frequency response of the compensated coefficient set, $\{h_{qc}\}$ is iteratively improved as discrete SPT terms in Table 4.1 are added to compensate for the quantization

error. As the coefficient values change dynamically in this process, unrestrictive compensation can lead to the over-flow or under-flow of coefficient values beyond the maximum compensation defined by the neighborhood of the initial coefficient set, $\{h_{pmin}\}$. Therefore, the range of the i -th compensated coefficient is limited to the following upper and lower bounds at all time.

$$h_{pmin}(i) - 2\Delta q_{max} \leq h_{qc}(i) \leq h_{pmin}(i) + 2\Delta q_{max} \quad (4.6)$$

The upper and lower bounds in (4.6) are calculated once from $\{h_{pmin}\}$ of the first stage and out-of-range coefficients are pruned in each iteration. The restrictive search narrows down the search space substantially. Therefore the computational time for each iteration reduces successively until convergence.

The procedure for quantization compensation is detailed as follows. To each coefficient $h_{pmin}(i)$, a number of compensated coefficients are created by adding the discrete SPT terms defined by the adaptive neighborhood of Table 4.1. Compensated coefficient values falling outside the range of (4.6) are discarded. The NPRM of every combination of coefficient set that can be generated from the pool of compensated coefficients is evaluated and the five best NPRM coefficient sets are stored as $\{h_{qc1}\}, \{h_{qc2}\}, \dots, \{h_{qc5}\}$. The numbers of CSPT terms for these coefficient sets can be extracted from the CS-HWP and stored in a list $\{ncspt_1, ncspt_2, \dots, ncspt_5\}$. The increment (or decrement) in the number of CSPT terms of these five best compensated coefficient sets from that of $\{h_{pmin}\}$ is recorded as $\{\Delta ncspt_1, \Delta ncspt_2, \dots, \Delta ncspt_5\}$. $\Delta ncspt$ is positive for an increment and negative for a decrement in the number of CSPT terms. The five compensated coefficient sets are sorted in ascending order of $\Delta ncspt$. If there is a tie, the sets are further sorted in ascending order of NPRM. The coefficient set with the minimum $\Delta ncspt$ is selected to replace $\{h_{pmin}\}$ as the starting coefficient set for the next iteration until the desired NPRM is fulfilled by the selected $\{h_{pmin}\}$.

The pseudocode of the two-stage algorithm is given in Figure 4.3.

two-stage search($\{h\}$, $desired_NPRM$)

```

1.  $\{h_{csd}\} = \mathbf{CSD}(\{h\});$ 
2.  $\{h_{cspt}\} = \mathbf{CS-HWP}(\{h_{csd}\});$ 
3.  $r = 0$ ; initialize  $\{h_p\}$  to a list of all zero coefficients;
4.  $\{h_{pmin}\} = \{h_p\}$ ;  $min\_NPRM = 0$ ;
   % Coefficient Allocation Stage
5. for ( $r = 1$  to  $\mathbf{NCSPT}(\{h_{cspt}\})$ ) {
6.   for all  $h_{cspt}(i)(j)$  with  $\mathbf{NCSPT}(\{h_{cspt}(i)(j)\}) = \mathbf{NCSPT}(\{h_p(i)\}) + 1$ 
7.      $\{h_p\}_{i,j} = \{h_p\} + h_{cspt}(i)(j) - h_p(i);$ 
8.      $\{h_p\} = \left\{ \{h_p\}_{i,j} \mid (i,j) = \arg \left[ \mathbf{NPRM}(\{h_p\}_{i,j}) \right] \right\};$ 
9.   delete from  $\{h_{cspt}\}$  selected coefficient and coefficients with same  $ncspt$  terms in the same tap;
10.  if ( $\mathbf{NPRM}(\{h_p\}) < min\_NPRM$ ) {
11.     $\{h_{pmin}\} = \{h_p\};$ 
12.     $min\_NPRM = \mathbf{NPRM}(\{h_p\});$  % end of if loop starting in line 10
13.  } % end of for loop starting in line 5
   % Quantization Compensation Stage
14. for ( $i = 1$  to  $N$ ) {
15.    $lower\_limit(i) = h_{pmin}(i) - 2\Delta q_{max};$ 
16.    $upper\_limit(i) = h_{pmin}(i) + 2\Delta q_{max};$ 
17.   while ( $\mathbf{NPRM}(\{h_{pmin}\}) > desired\_NPRM$ ) {
18.     for ( $i = 1$  to  $N$ )
19.       for ( $\Delta q = \pm 2^{-B}$ ,  $|\Delta q| \leq \Delta q_{max}$ ,  $\Delta q = \Delta q \times 2$ ) %  $\Delta q$  are discrete SPT terms from Table 4.1.
20.         if ( $h_{pmin}(i) + \Delta q > lower\_limit(i)$  and  $h_{pmin}(i) + \Delta q < upper\_limit(i)$ )
21.            $\{h_{pmin}(i)\} = \{h_{pmin}(i)\} + \{h_{pmin}(i) + \Delta q\} - h_{pmin}(i);$ 
22.         Find five best NPRM coefficient sets,  $\{\{h_{qc1}\}, \{h_{qc2}\}, \{h_{qc3}\}, \{h_{qc4}\}, \{h_{qc5}\}\}$ ;
23.         for ( $i = 1$  to 5)
24.            $\Delta ncspt_i = \mathbf{NCSPT}(\{h_{qc_i}\}) - \mathbf{NCSPT}(\{h_{pmin}\});$ 
25.         Sort the five coefficient sets in ascending order of  $\Delta ncspt$  then NPRM;
26.          $\{h_{pmin}\} =$  first coefficient set in the sorted list;
27.       } % end of while loop starting in line 17
18.   } return  $\{h_{pmin}\};$ 

```

Figure 4.3 Pseudocode of two-stage filter coefficient synthesis algorithm

In Figure 4.3, $\mathbf{CSD}(\{h\})$ rounds the coefficients of $\{h\}$ from real values to the nearest CSD numbers and $\mathbf{CS-HWP}(\{h_{csd}\})$ searches in the CS-HWP for the nearest CSD numbers with CSPT terms less than or equal to those of $\{h_{csd}\}$. The NPRM specification is given by the constant, $desired_NPRM$. $\mathbf{NPRM}(\{h\})$ returns the NPRM of the coefficient set, $\{h\}$ and $\mathbf{NCSPT}(\{h\})$ returns the total number of CSPT terms of $\{h\}$. The statement, $\{h_p\} + h_{cspt}(i)(j) - h_p(i)$ means replacing the coefficient, $h_p(i)$ in $\{h_p\}$ with $h_{cspt}(i)(j)$. Compared with the exhaustive search or optimal methods, the proposed algorithm avoids the search space from growing exponentially. The shortcoming of any heuristic algorithm is that for coefficient set with large quantization error in the conversion from real value

coefficients to finite wordlength coefficients, particularly for small wordlength, the algorithm may converge to a NPRM that misses the desired amplitude response specification by a small margin. The proposed algorithm relies on the optimality of each choice greedily and is not absolutely free from local minimum problem. One possible way to break away from the occasional local minimum problem is to expand the search scope in each iteration, but this will inevitably introduce more computations and many of which might be fruitless for the majority of coefficient sets that have already been easily managed by the present approach. Conditional backtracking with expansion of search scope based on the assessment of the NPRM margin of first stage and the descending rate of NPRM in the second stage may be feasible.

4.3.3 An Example

Example 4.2: A simple three-tap filter is used to illustrate the proposed two-stage design procedure. Suppose that an arbitrary set of coefficients $h(1) = 0.0051$, $h(2) = 0.0341$ and $h(3) = -0.1419$ that meets or nearly meets the required frequency response is given. The proposed algorithm is applied on this filter to generate another set of coefficients that will also fulfill the frequency response specification but with a lower CSPT cost. For a wordlength of 10-bit, its rounded CSD coefficients are $h_{csd}(1) = 2^{-8} + 2^{-10}$, $h_{csd}(2) = 2^{-5} + 2^{-8} - 2^{-10}$, and $h_{csd}(3) = -2^{-3} - 2^{-6} - 2^{-10}$. The normalized CSD coefficient array is given by:

$$h_{csd} = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & \frac{1}{1} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \frac{1}{1} \\ 0 & 0 & \overline{1} & 0 & 0 & \overline{1} & 0 & 0 & 0 & \frac{1}{1} \end{Bmatrix} \quad (4.7)$$

Since $ncspt = \{1, 2, 3\}$, new CSD coefficients are sought from the CS-HWP to form the following candidate pool (CP) in the *CSPT Coefficient Allocation* stage.

$$\begin{aligned}
h_{cspt}(1) &= \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{Bmatrix} \\
h_{cspt}(2) &= \begin{Bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & \overline{1} \end{Bmatrix} \\
h_{cspt}(3) &= \begin{Bmatrix} 0 & 0 & \overline{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \overline{1} & 0 & \overline{1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \overline{1} & 0 & 0 & \overline{1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \overline{1} & 0 & 0 & \overline{1} & 0 & \overline{1} & 0 & 0 \\ 0 & 0 & \overline{1} & 0 & 0 & \overline{1} & 0 & 0 & \overline{1} & 0 \\ 0 & 0 & \overline{1} & 0 & 0 & \overline{1} & 0 & 0 & 0 & \overline{1} \end{Bmatrix} \tag{4.8}
\end{aligned}$$

The cardinality of $h_{cspt}(i)$ for the i th coefficient in the CP is twice that of $ncspt(i)$. The preliminary coefficient set, $\{h_p\}$ is initialized to $\{0, 0, 0\}$. For the first iteration, six candidate coefficient sets with one CSPT term are formed by substituting a coefficient in $\{h_p\}$ by a coefficient from the CP with one CSPT term. The qualified coefficients from the CP include two candidates in $\{h_{cspt}(1)\}$ to substitute for $h_p(1)$, the first two candidates in $\{h_{cspt}(2)\}$ for $h_p(3)$ and $\{h_{cspt}(3)\}$ for $h_p(3)$. As $\{h_{cspt}(i)\}$ is sorted in ascending order of the number of CSPT terms, the first two terms of $\{h_{cspt}(i)\}$ are evaluated for the prospective $\{h_p\}_{ij}$ in each iteration. The coefficient set containing $h_{cspt}(3)(2)$ has the least NPRM and it is selected to replace $\{h_p\}$ for the next iteration. At this point, $\{h_p\} = \{0, 0, -2^{-3} - 2^{-5}\} = \{0, 0, -0.1536\}$. The coefficient, $h_p(3) = -2^{-3} - 2^{-5}$ is a common subexpression, $\overline{101} \equiv 101$ and it is deemed to have only one CSPT term. The two coefficients, $h_{cspt}(3)(1)$ and $h_{cspt}(3)(2)$ of one CSPT term are then deleted from $\{h_{cspt}(3)\}$. In the second iteration, the number of CSPT terms of $\{h_p\}$ is incremented to two. Two possibilities exist: either $h_p(1)$ or $h_p(2)$ is replaced with a candidate coefficient of one CSPT term from $\{h_{cspt}(1)\}$ or $\{h_{cspt}(2)\}$, or $h_p(3)$ is replaced with a candidate coefficient of two CSPT terms from $\{h_{cspt}(3)\}$. Altogether 6 candidate coefficient sets can be generated from the first two candidates of $\{h_{cspt}(1)\}$, $\{h_{cspt}(2)\}$ and $\{h_{cspt}(3)\}$. The candidate coefficient set containing the second candidate of $\{h_{cspt}(3)\}$ is found to have the minimum NPRM. Consequently, $\{h_p\} = \{0, 0, -2^{-3} - 2^{-6} - 2^{-8}\} = \{0, 0, -0.1445\}$. Two more coefficients in $\{h_{cspt}(3)\}$ of two CSPT terms are deleted and the updated candidate pool is given by (4.9):

$$\begin{aligned}
h_{cspt}(1) &= \begin{Bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{Bmatrix} \\
h_{cspt}(2) &= \begin{Bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{Bmatrix} \\
h_{cspt}(3) &= \begin{Bmatrix} 0 & 0 & \bar{1} & 0 & 0 & \bar{1} & 0 & 0 & \bar{1} & 0 \\ 0 & 0 & \bar{1} & 0 & 0 & \bar{1} & 0 & 0 & 0 & \bar{1} \end{Bmatrix} \quad (4.9)
\end{aligned}$$

The process continues until $\{h_{cspt}\}$ is exhausted. The *CSPT Coefficient Allocation* stage terminates with the set of coefficients, $\{h_p\}$ that has the smallest NPRM value for the given specification. $\{h_{pmin}\} = \{2^{-8}, 2^{-5} + 2^{-8} - 2^{-10}, -2^{-3} - 2^{-6} - 2^{-8}\} = \{0.0039, 0.0341, -0.1445\}$. The value of the i -th coefficient in the *Quantization Compensation* stage is restricted to the range $(h_{pmin}(i) - 2^{-7}, h_{pmin}(i) + 2^{-7})$ as given by (4.6). The lower and upper limits for each coefficient are given by $lower_limit = \{-0.0039, 0.0263, -0.1523\}$ and $upper_limit = \{0.0117, 0.0419, -0.1367\}$. The $\{ncspt\}$ for $\{h_{pmin}\}$ is $\{1, 2, 2\}$. Each coefficient is allowed to be compensated by six different discrete SPT terms, $\pm 2^{-8}$, $\pm 2^{-9}$ and $\pm 2^{-10}$ according to Table 4.1. The NPRMs of 18 compensated coefficient sets are evaluated from which the five least NPRM coefficient sets are extracted. These coefficient sets, sorted in ascending order of NPRM, are: $\{h_{pmin}(1), h_{pmin}(2), h_{pmin}(3) - 2^{-10}\}$, $\{h_{pmin}(1), h_{pmin}(2) - 2^{-10}, h_{pmin}(3)\}$, $\{h_{pmin}(1), h_{pmin}(2) - 2^{-9}, h_{pmin}(3)\}$, $\{h_{pmin}(1), h_{pmin}(2), h_{pmin}(3) + 2^{-10}\}$, and $\{h_{pmin}(1) + 2^{-10}, h_{pmin}(2), h_{pmin}(3)\}$. Their numbers of CSPT terms are respectively, $\{1, 2, 2\}$, $\{1, 2, 2\}$, $\{1, 2, 2\}$, $\{1, 2, 3\}$, and $\{1, 2, 3\}$. Therefore, $\{\Delta ncspt\} = \{1, 0, 0, 1, 0\}$. The second, third and fifth sets have maintained the same number of CSPT terms because the compensations have either generated new common subexpressions or relocated some existing common subexpressions. Among these three least $\Delta ncspt$ sets, the second set with the smallest NPRM is selected. $\{h_{pmin}\}$ is updated to $\{2^{-8}, 2^{-5} + 2^{-9}, -2^{-3} - 2^{-6} - 2^{-8}\} = \{0.0039, 0.0332, -0.1445\}$ and $\{ncspt\} = \{1, 2, 2\}$. The process iterates until the required NPRM is achieved.

4.4 Experimental Results

In this section, the capability of the proposed algorithm is demonstrated through the implementation of two practical design examples. The specifications of these two filters are frequently used as benchmarks for evaluating the optimality of different digital filter coefficient synthesis algorithms [CHE99, GUS03, YAO01, YLI01]. The results are compared with solutions synthesized by Chen's algorithm [CHE99], the MILP algorithm [LIM83], Samueli's local search algorithm [SAM89], Li's algorithm [LI93] and Yao's algorithm [YAO01]. The comparisons are based on the results taken from [CHE99, GUS03]. The metric used for benchmarking the performances of different solutions are the number of CSPT terms, which provides an indication of the adder cost and the NPRM, which provides an indication of the quality of fitness to the actual frequency responses. For fair comparison, the sharings of common subexpressions, 101 and $10\bar{1}$ are explicitly accounted for the number of CSPT terms of the coefficient sets generated by those algorithms that have not considered common subexpressions.

Example 4.3: The halfband filter is part of a programmable digital decimation filter using the sharpened (CIC) architecture with the frequency response specification designed for wideband satellite communication systems [KWE97]. The normalized passband and stopband edge frequencies are 0.2π and 0.8π , respectively. The coefficient wordlength is 14 bits and the filter length is 15. The stopband attenuation is -80 dB. An equiripple lowpass filter is first designed using Matlab Simulink based on the Parks-McClellan algorithm [IEEE79] and use the obtained real value filter coefficient set as the input $\{h\}$ to the proposed algorithm. The final coefficient set is given as follows.

$$\begin{aligned}
 h(0) &= h(14) = -2^{-9} - 2^{-11} \\
 h(1) &= h(13) = 0 \\
 h(2) &= h(12) = 2^{-6} + 2^{-10} + 2^{-12} \\
 h(3) &= h(11) = 0 \\
 h(4) &= h(10) = -2^{-4} - 2^{-8} - 2^{-10} - 2^{-13} \\
 h(5) &= h(9) = 0 \\
 h(6) &= h(8) = 2^{-2} + 2^{-4} - 2^{-7} - 2^{-9} + 2^{-11} - 2^{-13} \\
 h(7) &= 2^{-1}
 \end{aligned}$$

The NPRM and the number of CSPT terms are -72.28 dB and 28 for the rounded CSD coefficient set, $\{h_{csd}\}$, generated by Matlab. The NPRM and the number of CSPT terms

reduce to -78.45 dB and 21 after the CSPT Coefficient Allocation stage. The NPRM reaches -83.63 dB after the Quantization Compensation stage and the number of CSPT terms of the final solution is 19. The NPRM per CSPT term from the initial rounded coefficient set to the final solution are progressively reduced from -2.58 dB, -3.74 dB to -4.40 dB. In fact, the quantization compensation stage takes only two iterations to achieve the amplitude response requirement.

Table 4.2 gives the filter characteristics designed by different algorithms to attain -80 dB stopband attenuation with minimal possible number of CSPT or SPT terms.

Table 4.2 Summary of different filter designs for Example 4.3

Algorithms	NPRM (dB)	N_{SPT}	N_{CSPT}	N_{101}	$N_{10\bar{1}}$
Kwentus [KWE97]	-74.99	31	21	5	0
Samueli [SAM89]	-83.22	28	-	-	-
Li [LI93]	-82.35	27	-	-	-
Chen [CHE99]	-85.62	23	22	1	0
Proposed	-83.63	31	19	5	1

The original coefficient set provided by Kwentus [KWE97] aims at only -75 dB NPRM and its results are used as a reference in the first row of Table 4.2. In Table 4.2, N_{SPT} , N_{CSPT} , N_{101} and $N_{10\bar{1}}$ denote the number of SPT terms, the number of CSPT terms and the numbers of common subexpressions, 101 and $10\bar{1}$, respectively. The common subexpressions 101 and $10\bar{1}$ have been searched in the solution sets listed in Kwentus's [KWE97] and Chen's [CHE99] papers to account for the numbers, N_{CSPT} , N_{101} and $N_{10\bar{1}}$. For the other two algorithms, the implementation are unable to obtain from their papers [LI93, SAM89], only the published results of N_{SPT} and NPRM are listed. Based on the NPRM measurement, the proposed algorithm converges to a solution which has a better amplitude response than Kwentus's, Samueli's and Li's solutions. Chen's trellis search algorithm produces the least number of SPT terms but the actual implementation cost is reflected in the number of CSPT terms as adders saved by operator sharing have to be accounted. The high N_{SPT} of the proposed algorithm is expected as operator sharing is encouraged. Elimination of redundant computations by maximizing the number of

common subexpressions is evincible in the highest aggregate of N_{101} and $N_{10\bar{1}}$. The solution from the proposed algorithm has the least adder cost among the algorithms that have the N_{CSPT} listed. Three CSPT terms are saved over the trellis search algorithm due to the extraction of five more common subexpressions than Chen's algorithm. In Chen's solution, the maximum number of SPT terms per coefficient was three, and the solution obtained by the proposed algorithm also employs no more than three CSPT terms in any coefficient. The saving in adder cost is achieved at no performance penalty to logic depth for high-speed digital filter. Table 4.3 lists the actual total number of carry propagation adders needed to implement the published coefficient sets generated by the three algorithms in the transposed direct form structure. Since this is an even symmetric filter, the first half of the filter taps is duplicated in the second half. The adders used to generate the common subexpressions are treated as being equally complex as other adders although the wordlength of the adders is generally much shorter (a (input wordlength+2)-bit adder is sufficient to implement the 101 or $10\bar{1}$ subexpression). As only one subexpression of 101 appears in the solution of the trellis search algorithm, the actual adder cost is the same for its solution with or without the consideration of common subexpressions. With the sharing of common subexpression, 101 , Kwentus's solution can be realized with one less adder than that of the trellis search algorithm. As there is only one occurrence of the subexpression, $10\bar{1}$, the proposed implementation cost is the same as Kwentus's but the proposed amplitude response error is much smaller.

Table 4.3 Comparison of adder costs for Example 4.3

Algorithms	Kwentus [KWE97]	Chen [CHE99]	Proposed
Adder Cost	15	16	15

The amplitude response of Chen's, Kwentus's and the proposed solutions are shown in Figure 4.4.

Figure 4.4 Comparison of magnitude responses of FIR filter designs for Example 4.3

Example 4.4: This popular filter specification is taken from [CHE99, GUS03, YAO01, YLI01]. The normalized passband and stopband edge frequencies are 0.3π and 0.5π , respectively, with equal weighting on the passband and stopband ripples. The coefficient wordlength is 12 bits and the filter length is 28. The minimum numbers of SPT terms required by the MILP algorithm [LIM83], Samueli's local search algorithm [SAM89], Li's algorithm [LI93], Chen's algorithm [CHE99], Yao's [YAO01], Yli-Kaakinen's [YLI01] and Gustafsson's [GUS02c] algorithms to achieve -50dB normalized peak ripple values are compared. As no original coefficient set of this filter is published in any paper, the infinite precision coefficient set generated from Matlab Simulink based on the Parks-McClellan algorithm [IEEE79] is used as reference.

The final coefficient set generated by the proposed algorithm is given as follows.

$$\begin{aligned}
 h(0) &= h(27) = -2^{-8} + 2^{-10} + 2^{-12} \\
 h(1) &= h(26) = 0 \\
 h(2) &= h(25) = 2^{-7} - 2^{-9} + 2^{-12} \\
 h(3) &= h(24) = 2^{-7} - 2^{-9} - 2^{-12} \\
 h(4) &= h(23) = -2^{-7} \\
 h(5) &= h(22) = -2^{-6} - 2^{-9} + 2^{-11} \\
 h(6) &= h(21) = 0 \\
 h(7) &= h(20) = 2^{-5} \\
 h(8) &= h(19) = 2^{-5} - 2^{-7} + 2^{-9} + 2^{-11} \\
 h(9) &= h(18) = -2^{-5} - 2^{-8} + 2^{-10} \\
 h(10) &= h(17) = -2^{-4} - 2^{-6} + 2^{-10} \\
 h(11) &= h(16) = 0 \\
 h(12) &= h(15) = 2^{-2} - 2^{-4} + 2^{-6} - 2^{-8} - 2^{-11}
 \end{aligned}$$

$$h(13) = h(14) = 2^{-1} - 2^{-3}$$

The desired NPRM has been achieved directly with 56 CSPT terms for the rounded CSD coefficient set, $\{h_{csd}\}$. The number of CSPT terms has been brought down to 44 after the CSPT Coefficient Allocation stage with a slight increase in NPRM and the Quantization Compensation stage further reduces the number of CSPT terms to 40 in 93 iterations before it converges to an NPRM of -50.05 dB. The NPRM per CSPT term from the initial rounded coefficient set to the final solution are progressively reduced from -0.89 dB, -1.10 dB to -1.25 dB.

The frequency responses of the coefficient sets generated by different algorithms including the proposed algorithm are shown in Figure 4.5.

Figure 4.5 Comparisons of frequency responses of Example 4.4

Table 4.4 gives the comparisons of the NPRM, N_{SPT} , N_{CSPT} , N_{101} and $N_{10\bar{1}}$ achieved by the solutions of various algorithms. L_{max} is the maximum allowable number of SPT terms per coefficient. It should be noted that two published solutions that have also been included in Table 4.4 use different numbers of taps, N and wordlengths, B for the same amplitude response. The results used for the comparison are taken from [CHE99, GUS03]. For those algorithms which do not provide the final coefficient sets in the papers, the unavailable information is marked with '-' in Table 4.4. The results show that the proposed algorithm produces a solution with the least CSPT terms. Altogether 11

common subexpressions of 101 and $10\bar{1}$ have been generated by the algorithm, which is the second highest number. The original decimal coefficient set generated by Matlab has the most common subexpressions but it has also a high number of non-sharable SPT terms. The proposed algorithm achieves the least number of CSPT terms at the expense of a comparatively higher number of SPT terms due to the physical adder cost oriented objective of optimization.

Table 4.4 Summary of different filter designs for Example 4.4

Algorithm	N	B	NPRM (dB)	L_{max}	N_{SPT}	N_{CSPT}	N_{101}	$N_{10\bar{1}}$
Matlab	28	12	-51.12	-	80	56	5	7
MILP [LIM83]	28	12	-50.18	4	68	-	-	-
MILP [LIM83]	28	12	-50.18	3	68	-	-	-
Li [LI93]	28	12	-50.35	4	60	-	-	-
Gustafsson [GU02c]	28	12	-	-	54	-	-	-
Gustafsson [GU02c]	28	12	-	3	56	-	-	-
Yli-Kaakinen [YLI01]	28	12	-50.01	-	56	-	-	-
Yli-Kaakinen [YLI01]	30	10	-50.71	-	52	42	1	4
Yao [YAO01]	28	12	-50.08	4	54	42	2	4
Yao [YAO01]	28	12	-50.02	3	56	42	3	4
Chen [CHE99]	28	12	-50.17	4	58	42	5	3
Chen [CHE99]	28	11	-50.12	3	60	46	4	3
Proposed	28	12	-50.05	-	62	40	3	8

In [CHE99], Chen and Willson proposed a merge-search method to reduce the number of carry-save adders (CSAs) for a CSA-tree based transposed direct form FIR filter architecture. An additional precomputation cycle involving two fast carry-propagation adders (of only three bits each) for the common subexpressions, 101 and $10\bar{1}$, is incurred in this architecture. A constraint on L_{max} is imposed to limit the maximum number of CSAs for high throughput rate. The solutions of merging 101 and $10\bar{1}$ within the CSD coefficients before the optimization by MILP, called the MILP-merge, are also provided in [CHE99]. The solution based on the transposed direct form architecture with CSAs is

implemented and the results are tabulated in Table 4.5 along with the Trellis-merge and MILP-merge algorithms from [CHE99] for comparison.

Table 4.5 Comparison with merge-search algorithms for CSA-based filters

Algorithms	Limit of CSAs per coefficient	N	N _{CSA}
Trellis-merge	≤ 3	28	40
	≤ 2	28	40
MILP-merge	≤ 3	28	49
	≤ 2	29	52
Proposed	-	28	38

Although the number of CSPT terms per coefficient is not restricted in the algorithm, the solution generated by the proposed algorithm has no more than 5 SPT terms and 3 CSPT terms in any coefficient. Therefore, no more than 2 carry-save adders are needed for every coefficient. Without degrading the throughput rate, the solution requires a slightly smaller number of carry-save adders.

It should be noted, however, that the CSA-based transposed direct form architecture cannot benefit from any duplicated coefficient as there are at least two carry-save adders needed per tap to accumulate the stored sum and carry from the previous tap to those of the current tap, in addition to doubling the number of registers needed to store the sum and carry signals. If the conventional carry-propagation adders (CPA) are used for the transposed direct form filter, the saving in adder cost is prominent for symmetric filter, as half of coefficients are duplicated and incur no additional adder cost. The implementation of the solution generated by the proposed algorithm in the symmetrical transposed direct form filter structure is shown in

Figure 4.6. Two three-bit adders are employed for the generation of common subexpressions, 101 and $10\bar{1}$, 11 adders are utilized to generate the coefficients, and 21 adders are used in the accumulation line.

Figure 4.6 Circuit implementation with common subexpressions, 101 and 101.

The solution of the proposed algorithm is further compared with the solution of Yli-Kaakinen’s algorithm [YLI01], which is one that involves sharing of higher weight common subexpressions. Yli-Kaakinen has provided two circuits with higher weight common subexpressions for this filter specification using different numbers of taps and wordlengths. The best implementation is obtained with $L = 30$ and $B = 10$. The higher weight common subexpressions are extracted from the solution and the final circuit implementation of the proposed design is shown in Figure 4.7. Two weight-3 common subexpressions have been identified, which are $10\bar{1}00\bar{1}$ and $\bar{1}00\bar{1}01$. Since both weight-3 common subexpressions appear twice, two more adders are saved. This solution needs only 30 adders, which is the same as Yli-Kaakinen’s best solution.

Figure 4.7 Circuit implementation with higher weight common subexpressions

To show that the proposed algorithm is also capable of synthesizing desirable solutions to a wide variety of FIR filter response specifications, the algorithm is evaluated using a number of other frequently used filter specifications for digital filter synthesis algorithms.

The filters are from [LIM90, SAM89, ATI93, LEE93, YON03 and GUS04], with some of the specifications are modified to fulfill the proposed algorithm's requirements. The results are shown in Table 4.6. In Table 4.6, the filter specifications, the target NPRM requirements and the quality of solutions obtained from the proposed algorithm are listed.

In Table 4.6, ω_p and ω_s are the normalized passband and stopband edge frequencies, δ_p and δ_s are the passband and stopband ripple magnitudes, respectively, N is the filter length and B is the wordlength of the coefficients. The last column of Table 4.6 provides the ratios of NCSPT to the product of filter length and coefficient wordlength. This normalized number of CSPT terms serves as a useful figure-of-merit to evaluate the overall performance of the algorithm in term of the average adder cost per tap per bit. As mentioned earlier in the introduction, the maximum number of nonzero digits for an n -bit binary number expressed in CSD format is bounded by $(n + 1)/2$ and the expected number of nonzero digits approaches $n/3 + 1/9$ asymptotically [HAR96]. The expected number of nonzero digits varies from one filter to another and the nonzero digits of the FIR filter are not uniformly distributed among different CSD coefficients. The figure-of-merit shows that on average, for every n -bit binary coefficient, the algorithm can reduce the number of CSPT terms to less than $n/8$. This is a noteworthy reduction. The NCSPT is highly correlated to the actual adder cost incurred than the number of SPT terms, with an overhead of only two (and sometimes one) adders to generate the two common subexpressions, 101 and $10\bar{1}$.

Table 4.6 Experimental results for other filter specifications

	ω_p	ω_s	$\frac{\delta_p}{\delta_s}$	N	B	Desired NPRM (dB)	NPRM Achieved (dB)	NCSPT	$\frac{NCSPT}{N \times B}$
FIR1	0.3π	0.5π	1	25	12	-46	-46.16	39	0.13
FIR2	0.3π	0.5π	3	35	11	-50	-50.06	44	0.11
FIR3	0.256π	0.4096π	32/23	28	10	-40	-40.18	44	0.16
FIR4	0.3π	0.44π	1	53	9	-40	-40.27	46	0.10
FIR5	0.6π	0.69π	1	64	13	-50	-50.08	98	0.12
FIR6	0.47π	0.5π	1	131	12	-40	-40.00	195	0.12

4.5 Summary

An efficient algorithm has been proposed for the design of low-complexity FIR filter by maximizing the sharing of adders in the synthesis of filter coefficients to meet the desired magnitude response specification. The proposed algorithm considers the reuse of the two most frequently used and most profitable subexpressions, 101 and $10\bar{1}$. By dividing the multiple-constraint problem of minimizing the CSPT terms into two stages, the problem is tackled heuristically in an elegant and logical approach. With the help of a modified Hamming Weight Pyramid (CS-HWP), a minimal CSD coefficient set with the minimal NPRM in the continuum of the rounded CSD coefficient set is found in the first stage. The search has been highly efficient as it is confined to only worthy candidate coefficients with equal or smaller number of CSPT terms as the initial set. Depending on the wordlength of the coefficients, and the mismatch between the desired response and that obtained in the *CSPT Coefficient Allocation stage*, bivariate quantization steps in terms of discrete SPT terms are iteratively added in the *Quantization Compensation stage* to compensate for the quantization errors until the desired NPRM has been met. By means of benchmark design examples, the proposed algorithm is shown to be indeed capable of synthesizing transposed direct form FIR filters with the least adder cost in both the CSA-tree and the conventional adder based architecture among the existing filter synthesis algorithms. As it is a local search algorithm and the search range is well defined, the proposed algorithm finds a set of filter coefficients that meets the filter specification in a reasonable time.

Chapter 5

Contention Resolution Algorithm for Weight-2 Horizontal Subexpressions

5.1 Introduction

In the previous chapter, a new method has been developed to synthesis coefficients for linear phase equiripple FIR filters that contain maximally reusable common subexpressions of types 101 and $10\bar{1}$. In this chapter, the same optimization problem will be further studied but with a different starting point and an emphasis on high-level synthesis transformation. The use of transformations in high-level synthesis of VLSI circuits aims at reducing the cost metrics such as area and power consumption via operator re-scheduling and re-allocation to meet the timing constraints. Common subexpression elimination (CSE) [HAR96, PAR02, PAS99, MAR02, POT96] has emerged as one such powerful transformation that has been successfully employed in algebraic speed-up procedure [IQB93] to minimize the critical path delay. Perhaps, the most phenomenal advancement of CSE in high-level transformation has been the effective reduction of the number of arithmetic operators, and hence the area and power of the multiple constant multiplication (MCM) problem. MCM was first coined by [POT96] as a generalization of the decomposition of simultaneous multiplication of one variable with multiple constants into a reduced number of shift and add operations. While the multiplication by a constant appears only in small fraction of general computation benchmarks, its frequency of encounters has been significant in application specific computations, such as the fixed point implementation of generic digital filters. In general, greater savings in operators and more flexibility in critical path re-timing can be achieved for the design of more complex fixed filters by considering different lengths of weight-2

common subexpressions than merely 101 and $10\bar{1}$. However, from the algorithm proposed in the previous chapter, if the reusability of common subexpressions beyond the types 101 and $10\bar{1}$ is to be optimized simultaneously with the fulfillment of magnitude response specification, the computational complexity increases tremendously. Besides, the NPRM measure of the amplitude response alone may not be applicable or feasible for the design of all digital filters. Therefore, this chapter focuses on the development of new method to minimize the implementation cost of a given set of finite wordlength coefficients obtained from the filter transfer function through any means.

As surveyed in Chapter 2, the CSE and GD algorithms are the two categories generally utilized in high-level synthesis transformation to reduce the total hardware cost of the MCM block. They leverage on different number systems and both approaches have their own merits and limitations. Regardless of the differences in algorithms and number systems used, the search for common subexpressions (or the synthesis of repeated partial sums) to form the complete set of coefficients with minimal number of basic arithmetic operators (henceforth referred to as adders unambiguously) is the key concept behind the CSE and GD algorithms. Different optimizing algorithms lead to different realizations of the MCM block. The objective functions (whether logic depth or logic complexity is a premier consideration) and filter coefficient representations play important roles in the quality of solutions. Due to the complexity of the MCM problem defined in [POT96], there is no exact algorithm that guarantees optimality of solutions for all kinds of coefficients. For coefficients expressed in the Canonical Signed Digit (CSD) form, optimal subexpression sharing algorithms based on integer linear programming (ILP) model have been proposed [GUS02a, YUR99]. Both algorithms have high computational complexity as the optimal selection of common subexpressions is an NP-complete problem. Since exact optimal solution is intractable, almost all existing algorithms are heuristic in nature and the solutions generated are highly possibly suboptimal due to local minima. There is no feedback to allow efficient resubstitution of lower cost common subexpressions in case of conflicts of potential subexpression elimination caused by early decision in the execution process. In this chapter, a new contention resolution algorithm, named CRAH-2, featuring the ability to change the precedence of subexpressions to

resolve the local minima problem is proposed. CRAH-2 is applied on the CSD coefficients to reduce the implementation cost and maximize the throughput performance via reduced logic depth of critical path but it is also applicable to coefficients represented in other forms. The fast evolving software defined radio and contemporary communication systems include customized ASICs tailor for wave shaping functions like correlators and high-speed channelizers [GUN99]. These ASICs are designed for low die area, high throughput and low power. The throughput and power consumption requirements exert an upper bound on the logic depth of the multiplier block for a given supply voltage and cell library. Since only horizontal weight-2 common subexpressions are synthesized in CRAH-2, minimal logic depth is assured. Reducing the logic depth has the added advantage that shallow taps can shorten the propagation paths of spurious switching activities through long chain of adders, contributing to lower power dissipation.

The chapter is organized as follows. Section 5.2 introduces the notions and expressions of the admissibility graph. A numeric example is presented to illustrate the terminologies of the admissibility graph. The concept of contention resolution, which is the principle idea of the algorithm, is introduced in section 5.3. Two most frequently encountered contentions are discussed. The proposed CRAH-2 is described in Sections 5.4, followed by the performance analysis and comparison of simulation results with other distinguished algorithms in Section 5.5. The chapter is summarized in Section 5.6.

5.2 Admissibility Graph

Among the many computational models that have been proposed for the CSE problem, the most intuitive and elegant graph theoretic model is the primitive operator graph (POG) [BUL91], where each vertex of the graph represents a partial sum (called fundamental) and each edge represents a fixed number of shifts. This data structure allows the multiplication-free linear transform associated with the fixed coefficient FIR filter realization to be mapped to a NP-complete problem of minimizing the number of vertices of the dependence graph representing the set of coefficients. The processing of the coefficients in order of their magnitudes, and the inclusion of redundant even-valued

partial sums has also limited the achievable quality of the solutions. In this section, we introduce the notion of admissibility graph, which is a radically different data structure that is well suited for minimizing the logic complexity and logic depth of FIR filter implemented with the signed digit coefficients. To facilitate identification and resolution of contentions, a unique data structure named *Admissibility Graph*, $G(V, E)$, where V is a set of vertices and E is a set of edges, is used to describe a set of fixed point coefficients. The following terminologies are used in explaining the construction of the Admissibility Graph and the concept of contention resolution.

5.2.1 Notations and Expressions

Definition 5.1: Every vertex $v \in V$ in an Admissibility Graph $G(V, E)$ represents a nonzero digit. The vertex set V is a collection of all the nonzero digits of the coefficient set and $V_i \subset V$ is the vertex set for the i -th coefficient in the subgraph, G_i . Two types of vertices are defined, a signed vertex corresponds to the digit $\bar{1}$ (-1) and an unsigned vertex corresponds to the digit 1 .

Definition 5.2: Each vertex, v of the M -vertex admissibility graph is associated with a unique positional index, $index(v)$, determined by its position in the fixed point coefficient and the *index* is a non-negative integer.

Definition 5.3: A *subexpression* (or partial sum) is a signed digit string begins and ends with nonzero digits. The *weight* of a subexpression is defined as the number of nonzero digits of the subexpression.

Two subexpressions are said to be *common* if one can be obtained from another without incurring adder costs, i.e., only shifters and inverters are needed.

Definition 5.4: An edge $e = (v_i, v_j)$, $e \in E$ is a connection of two vertices v_i and v_j which are called the endpoints of edge e . An edge, $e = (v_i, v_j)$ forms a weight-2 subexpression separated by a *distance* $= |index(v_i) - index(v_j)| - 1$ of zeros.

The edge set, E of an admissibility graph is a collection of all weight-2 subexpressions of the coefficient set.

Definition 5.5: A path is a succession of adjacent edges and has two endpoints. A path η of weight- w consists of w vertices $(v_1, v_2, \dots, v_w) \subseteq V$ and $w - 1$ edges $(e_1, e_2, e_3, \dots, e_{w-1}) \subset E$. A path η , denoted by $e_1, e_2, \dots, e_{w-1}$, is an alternative sequence of vertices and edges, $v_1, e_1, v_2, e_2, \dots, v_{w-1}, e_{w-1}, v_w$ if the endpoints of the path is v_1 and v_w , and successive vertices v_i and v_{i+1} are endpoints of the intermediate edge e_i .

Definition 5.6: Two edges incident with a common vertex are said to be *adjacent*.

Definition 5.7: The degree of a vertex v , denoted by $\deg(v)$, is the number of edges incident with v .

Definition 5.8: A *precedence edge*, $e_p = (v_i, v_j)$ is used to link two vertices, $v_i, v_j \in V$ to admit a common (i.e., sharable) subexpression and a *contention edge*, $e_c = (v_i, v_j)$ is used to indicate a subexpression that is denied admission as a common subexpression by another subexpression that has already been admitted as common subexpression. The sets of precedence and contention edges are denoted by E_p and E_c , respectively.

In other words, each contention edge implies a conflict (contention) between two common subexpressions, as there exists a common vertex between two edges. Selection of one edge into the final common subexpression list will automatically nullify the admissibility of the other. The definitions of precedence and contention edges are extended to the precedence and contention paths for subexpressions of weight greater than two. This extension will be useful for the higher weight common subexpressions to be considered in the next chapter.

Definition 5.9: A vertex is *free* if it is not connected by a precedence edge (or path) and on the contrary, vertices connected by precedence edges (or paths) are *fixed*.

Based on Definitions 8 and 9, two vertices, v_i and v_j are linked via a *contention edge* $e_c = (v_i, v_j)$ if at least one vertex v_i or v_j is fixed.

Definition 5.10: The subexpressions are ordered in descending order of their number of occurrences in the coefficient set and each edge representing a distinct subexpression is given a unique order number, $order(e)$, starting with 1 for the highest number of occurrences. Two edges are said to be equivalent if they have the same *order*. Equivalent edges are common subexpressions.

Definition 5.11: Two paths $\eta_1 = e_1, e_2 \dots e_{w-1}$ and $\eta_2 = e_1', e_2', \dots e_{w-1}'$ are said to be equivalent, denoted as $\eta_1 \equiv \eta_2$ if all their corresponding edges are equivalent, i.e., $order(e_i) = order(e_i') \forall i = 1, 2, \dots, w-1$.

5.2.2 Example

As an illustration, Figure 5.1 shows a 5-vertex admissibility subgraph, $G_1 \subset G$ for the first coefficient, $C = \bar{1}00\bar{1}010\bar{1}001$ of the coefficient set, $\{\bar{1}00\bar{1}010\bar{1}001, 10\bar{1}00100\bar{1}0\bar{1}, 1010\bar{1}00\bar{1}000\}$. This subgraph represents a specific state of G for the entire coefficient set in the process of contention resolution algorithm. In Figure 5.1, the unsigned and signed vertices are drawn as solid and hollow circles, respectively. The precedence edges are drawn with solid lines while the contention edges are drawn with dash lines. Each vertex is annotated with its position (within the coefficient), and each precedence edge is annotated with a number indicating its order of precedence. The orders are obtained from the statistical analysis of the full set of coefficients, and the state of Figure 5.1 occurs at some stage of the contention resolution procedure to be presented in following sections. In this case, $10\bar{1}$, $\bar{1}00001$, 101 and $\bar{1}00\bar{1}$ are the four topmost frequently occurred common subexpressions with the following frequencies of occurrences: 3, 3, 2 and 2, respectively. These common subexpressions are sorted according to their frequency of occurrences followed by the number of zeros in their string. Each of these common subexpressions in the sorted list is assigned with a unique

order number from 1 to 4. This admissibility graph features one precedence edge, $v_{13}v_{14}$, and three contention edges, $v_{12}v_{13}$, $v_{13}v_{15}$ and $v_{14}v_{15}$. Edges $v_{12}v_{13}$ and $v_{13}v_{14}$ represent respectively the subexpressions, $10\bar{1}$ and $\bar{1}01$ of order 1, edge $v_{13}v_{15}$ represents the subexpression, $\bar{1}00001$ of order 2 and edge $v_{14}v_{15}$ represents the subexpression, $\bar{1}00\bar{1}$ of order 4. Vertices v_{11} , v_{12} and v_{15} are all free vertices. The status of the precedence and contention edges marked at this stage implies that the string $\bar{1}01$ ($v_{13}v_{14}$) in C_1 has been elected to be a common subexpression, making it impossible to share other strings in C_1 , like $10\bar{1}$ ($v_{12}v_{13}$), $\bar{1}00001$ ($v_{13}v_{15}$) and $\bar{1}00\bar{1}$ ($v_{14}v_{15}$) as common subexpressions.

Figure 5.1 A 5-vertex Admissibility Graph.

An admissibility graph is constructed by connecting vertices with precedence or contention edges. Contention resolution algorithms are proposed to resolve the conflict when the endpoints of a precedence edge are incident with contention edges that have a free endpoint. For example, the pattern graph represented by the path $v_{12}v_{13}v_{14}v_{15}$ requires a contention resolution. The contention resolution decides if the pattern graph should be remained as it is or to promote both contention edges to precedence edges and downgrade the precedence edge to a contention edge. Every precedence edge in the final solution graph is a common subexpression. The final precedence edges determine the logic complexity and provide the information necessary for the circuit implementation.

5.3 Contention Resolution

Many CSE algorithms are based on heuristic search technique [HAR96, PAR02, PAS99, MAR02, POT96] to select the common subexpressions for elimination. These algorithms

suffer from a common problem that once a subexpression is identified as the most profitable common subexpression, the decision cannot be reverted. Each candidate subexpression is independently selected based on some cost function to form the final solution progressively. The gross effect of interdependency of coefficients to the cost of implementation is not addressed adequately. For example, consider the subexpressions of $100\bar{1}$, 101 and $\bar{1}01$ of a coefficient $10100\bar{1}01$. If the number of occurrences (henceforth refer to as the frequency) of the subexpressions $100\bar{1}$, 101 and $\bar{1}01$ in a set of coefficients describing a FIR filter are, respectively, f_1, f_2 and f_3 with $f_1 > f_2 > f_3$, then it is highly probable that most algorithms will select $100\bar{1}$ into the candidate pool of common subexpressions. However, the selection of $100\bar{1}$ rules out the other two subexpressions. If the total induced common subexpressions of 101 and $\bar{1}01$ are higher, some adders may be saved by selecting 101 and $\bar{1}01$ instead $100\bar{1}$ as the common subexpressions. To handle this kind of situation, the rigidity of the search for common subexpressions is relaxed by allowing the earlier chosen candidate subexpressions marked by the precedence edges to be substituted by its adjacent contention edges determined by a specific cost metric. This provides the algorithm with the ability to correct choices made earlier to avoid local minima.

The contention resolution described above is illustrated in Figure 5.2. This type of contention is the most representative one and occurs frequently. It occurs when both endpoints of a precedence edge are connected to contention edges, and the other endpoints of the adjacent contention edges are free. At least one adder will be saved by this contention resolution.

Figure 5.2 A scenario of contention resolution

Contention is identified by shared (overlapping) vertices of different common subexpressions and resolved through adder cost evaluation of a localized pattern graph. The number of occurrences of each common subexpression is enumerated only once at the beginning and it is used to order the precedence of common subexpressions so that the lower order edges, corresponding to the most frequently occurred common subexpression are admitted into the admissibility graph first. This list of common subexpression frequencies will not be modified during the admission process. Common subexpressions are admitted as either precedence or contention edges or paths and the frequency of occurrences of the precedence edges or paths for each order number are updated dynamically as they are admitted and when they have been changed as a consequence of contention resolution. The resolution of contention may downgrade certain precedence edges to contention edges. Depending on the number of contentions and the frequency of downgradings, some subexpressions may end up with one or no precedence edge along the contention resolution process and are no longer common subexpressions any more. This led to another frequently encountered scenario of contention resolution as shown in Figure 5.3 whereby one endpoint of the precedence edge is connected to at least one contention edge.

Figure 5.3 Another scenario of contention resolution

Two subexpressions $100\bar{1}$ and 101 of a coefficient $10100\bar{1}$ are shown in Figure 5.3, with 101 assuming a lower order number than $100\bar{1}$. Therefore the higher order common subexpression $100\bar{1}$ admitted after 101 creates a contention due to the sharing of vertex v_2 . Despite having a lower order, it is still possible that due to contention and contention resolution before reaching this point, v_1v_2 is now the only precedence edge of subexpression 101 in the admissibility graph. If the subexpression $100\bar{1}$ has more than one precedence edges at this state, then swapping of precedence-contention edge status of

the two edges involved in Figure 5.3 will save one adder as 101 is no longer a common subexpression.

Two types of contention resolutions shown in Figure 5.2 and Figure 5.3 are the most fundamental ones and other complicated contentions may be reduced to these two types and resolved accordingly. For example, Figure 5.4 gives a more complicated contention. $v_1v_2v_3$ is evaluated as the second type of contention in Figure 5.4. If there is no adder saving, $v_1v_2v_3v_4$ is evaluated as the first type of contention. Precedence edge v_4v_5 should also be considered in the evaluation because if we resolve the contention $v_1v_2v_3v_4$ as Figure 5.2 shown, v_4v_5 will be changed to contention edges. In this way, higher weight contentions are decomposed to the basic two contentions and resolved.

Figure 5.4 Complicated contention

A cost metric, *NAS* (Number of Adders Saved) is defined to evaluate if the status of a precedence edge and contention edge(s) need to be swapped whenever a contention is encountered. *NAS* can be interpreted as an anticipated minimal number of adders saved due to swapping based on a localized model. Without ambiguity, the term adder implies either an adder or a subtractor. Figure 5.2 is used as an example to show how *NAS* is calculated. Let $e_p = (v_2, v_3)$ be the precedence edge (100 $\bar{1}$), $e_{c1} = (v_1, v_2)$ (101) and $e_{c2} = (v_3, v_4)$ ($\bar{1}$ 01) be the two contention edges of the first type. Assume that a coefficient is represented by a simple pattern graph of only these four vertices, v_1, v_2, v_3 and v_4 . If e_p is chosen as the final common subexpression, two adders are required to realize the coefficient for the free vertices v_1 and v_4 . On the other hand, if e_{c1} and e_{c2} replace e_p to be the final subexpression, there is no free vertex but an adder is required to sum the two subexpressions. Therefore, the number of adders saved due to swapping is one. $NAS = 1$ is set initially for this simplistic case and the cost is adjusted by considering other edges in E_p and E_c . If there exists equivalent precedence edges of e_{c1} and e_{c2} , i.e., $\exists e_{p1}, e_{p2} \in E_p$

with $order(e_{p1}) = order(e_{c1})$ and $order(e_{p2}) = order(e_{c2})$, then the adders required for the partial sums of e_{c1} and e_{c2} have already been allocated elsewhere. NAS remains unchanged at this stage as no additional adder is incurred. Otherwise, the number of equivalent precedence edges of e_{c1} , e_{c2} and e_p are independently examined. If there is no edge in E_P equivalent to e_{c1} , then one adder is required to realize this subexpression and $NAS = NAS + 1$. The same evaluation is carried out for e_{c2} as well. If there is no other equivalent precedence edge of e_p , then $NAS = NAS + 1$ because it is assumed that there are at least one precedence edges equivalent to e_p at the outset. If the final value of NAS is greater than zero, it will be profitable to change the status of the edges involved.

In rare cases when there are more than one contention edges adjacent to an endpoint of a precedence edge, the one with at least one other equivalent precedence edge will be considered for the above swapping. If none exists, then the priority is given to the one that has the highest number of equivalent contention edges followed by the contention edges with the lowest order number. As all edges of the same order are evaluated in the same iteration, more than one contention may be encountered concurrently. Under this complex situation, all pattern graphs involving in the contention resolution have to be considered as a whole. Swappings of some or all of the contentions are performed when their aggregate NAS is greater than the cost of generating the hitherto nonexistent precedence edges.

5.4 Contention Resolution Algorithm for Weight-2 Horizontal Subexpressions

In this section, a new algorithm is proposed to realize a set of filter coefficients with reduced number of logic operators constrained by throughput rate. The proposed algorithm leverages on the admissibility graph structure presented in Section 5.2 to detect and resolve the contentions explained in Section 5.3. The rigidity of the search for common subexpressions is relaxed by allowing the earlier chosen candidate subexpressions marked by the precedence edges to be substituted by the adjacent contention edges determined by a specific cost metric. This provides the algorithm with

the ability to correct choices made earlier to avoid local minimum. The final solution is generated at the last stage when all the contentions have been resolved. Minimal logic depth is ensured as we consider only the sharing of weight-2 horizontal common subexpression. This objective of minimizing the critical path delay is important particularly of channel filters in radio receiver which have to operating in high throughput due to the high sampling frequency.

5.4.1 Problem Statement

For ease of reference, we reiterate the expression of filter coefficient, $h_{spt}(i)$ in terms of the sum of SPT terms from Chapter 4 [Chen99]:

$$h_{spt}(i) = \sum_{k=1}^{L_i} s_{i,k} 2^{-p_{i,k}} \quad (5.1)$$

where $s_{i,k} \in \{-1, 1\}$ is the k -th SPT term at bit position, $p_{i,k} \in \{1, 2, \dots, B\}$ and L_i is the number of SPT terms in i -th coefficient. It is assumed without loss of generality that $0 \leq |h_{spt}(i)| < 1$ and $h_{spt}(i)$ has a wordlength of B bits and comprises L_i SPT terms.

The cost of implementing a generic MCM block is the number of adders needed to sum up all SPT terms, i.e.,

$$C_MCM = \sum_{i=1}^N (L_i - 1) \quad (5.2)$$

The outputs of the taps are delayed and amassed in an alternate register and accumulator line. The adder cost of this accumulation line is equal to the number of nonzero valued coefficients excluding the first coefficient:

$$C_ACC = \sum_{i=2, h(i) \neq 0}^N 1 \quad (5.3)$$

For the special case of a linear phase symmetric FIR filter, C_MCM is reduced by half because half of the coefficients are identical but C_ACC remains unchanged:

$$C_MCM = \sum_{i=1}^{\lceil N/2 \rceil} (L_i - 1) \quad (5.4)$$

The overall adder cost is given by

$$C = C_MCM + C_ACC \quad (5.5)$$

Multiple SPT terms are merged to form a subexpression. With the elimination of common subexpressions, the calculation of C_MCM can be divided into two parts – the adders required for the generation of common subexpressions and the adders required summing the common subexpressions and the spare SPT terms in the MCM block. If only weight-2 common subexpressions are considered, the adders needed for merging two SPT terms into a common subexpression equal to the number of distinct weight-2 common subexpressions. In the context of the admissibility graph, each precedent edge with a distinct order number in the solution graph requires one adder. If N_CS common subexpressions have been selected, then the number of adders needed to sum the common subexpressions, C_CS is

$$C_CS = N_CS \quad (5.6)$$

We can consider the common subexpressions and the spare SPT terms that are not used to form any common subexpression as being of equal cost. Therefore, the CSPT term defined in Chapter 4 is still applicable here. As a result, (5.1) can be modified to

$$h_{cspt}(i) = \sum_{k=1}^{L_i'} c_{S_{i,k}} 2^{-P_{i,k}} \quad (5.7)$$

$$L_i' = N_CS_i + SL_i \quad (5.8)$$

Comparing with (5.1), L_i' is the number of CSPT terms in the i th coefficient, which include N_CS_i weight-2 common subexpressions and SL_i number of remaining SPT terms ($\deg(v) = 0$). Therefore

$$C_MCM = C_CS + \sum_{i=1}^N (L_i' - 1) \quad (5.9)$$

C_ACC remains unaffected as there are common subexpressions exist only in the MCM block. For the case of symmetric filters, C_MCM is reduced to

$$C_MCM = C_CS + \sum_{i=1}^{\lceil N/2 \rceil} (L_i' - 1) \quad (5.10)$$

The goal of the proposed algorithm is to search for quality horizontal weight-2 common subexpressions that best minimize C_MCM .

5.4.2 Proposed Algorithm

The proposed algorithm, CRAH-2 searches for optimal horizontal common subexpressions of weight-2. Let H be the set of unique CSD coefficients that have more than one nonzero digits. A list of common subexpressions of weight two, S is generated from H and sorted in descending order of frequency of occurrences. In computing the frequency, the sign of the subexpressions are not differentiated, and identical overlapping subexpressions are counted only once. When there is a tie, the subexpressions are further sorted in ascending order of *distance* (see Definition 5.4). A unique order number is assigned to each distinct subexpression and subexpressions of *order* = 1 have the highest precedence. Figure 5.5 lists the pseudo codes of the CRAH-2.

```

1: for (each  $C_i$  from  $L$ ,  $i = 1 \dots |H|$ )
2:   generate admissibility graph  $G_i$  from the nonzero digits of  $C_i$ ;
3:    $E_P = \emptyset$ ;
4:    $E_C = \emptyset$ ;
5:   for ( $i = 1 \dots \text{max\_order}$ ) {
6:     each  $S_j$  from  $S$ , with  $\text{order}(S_j) = i$ ;
7:      $v_1, v_2 \in S_j$ ;
8:     if ( $v_1, v_2$  are both free) {
9:       connect ( $v_1, v_2$ ) with precedence edge,  $e_p$ ;
10:       $\text{order}(e_p) = \text{order}(S_j)$ ;
11:       $E_P = E_P \cup e_p$ ; }
12:    else {
13:      connect ( $v_1, v_2$ ) with contention edge,  $e_c$ ;
14:       $\text{order}(e_c) = \text{order}(S_j)$ ;
15:       $E_C = E_C \cup e_c$ ; }
16:    for (each  $e_p = (v_1, v_2) \in E_P$ )
17:      if (contention) { -- contention resolution required
18:        compute exchange cost,  $NAS$ ;
19:        if ( $NAS > 0$ ) { -- potential saving by exchange
20:          let  $e_1 \in E_C$  be the minimal-cost edge adjacent to  $v_1$ ;
21:          let  $e_2 \in E_C$  be the minimal-cost edge adjacent to  $v_2$ ;
22:           $E_P = (E_P \cup e_1 \cup e_2) - e_p$ ;
23:           $E_C = (E_C \cup e_p) - e_1 - e_2$ ;
24:        }
25:      }
26:    }
27: return  $E_P$ ;

```

Figure 5.5 Algorithm CRAH-2 for weight-2 horizontal common subexpressions.

Once the order list S is established, we can begin to construct the admissibility graph $G_i(V_i, E_i)$ for each coefficient C_i from H , $i = 1, 2, \dots |H|$. E_P and E_C are the lists storing the

precedence and contention edges, respectively. The degrees of two endpoints of a precedence edge are used to check if a contention has occurred. Lines 19 to 24 of Figure 5.5 indicate the resolution of contention.

At the end of the process, the precedence edges in E_P form the partial sums. The cost of implementation in terms of the total number of adders used can be determined from E_P . For a fixed coefficient set, shifters can be hardwired directly, and the complexity and throughput are dictated by the number of adders used. The calculation can be divided into two parts: C_MCM and C_ACC as addressed in section 5.4.1. Let N_CS denote the number of distinct weight-2 common subexpressions in E_P , then the cost of realizing the common subexpressions, $C_CS = N_CS$. Nonzero digits that are not included in the final common subexpressions (vertices, v with $\deg(v) = 0$) need to be summed with the partial sums to produce the final coefficient. In this calculation, the adders in the final linear accumulator are included. While in some publications [BUL91, DEM95, PAR02, PAS99, MAR02], this accumulator cost is not considered because it is not part of the MCM block. This difference will be considered when the results of different algorithms are compared in section 5.5. The reason of its consideration here is for consistency because different adder tree structure may be resulted when vertical and oblique common subexpressions are involved as discussed in Chapter 2. So the total cost of adders is given by (5.11), which can be derived from (5.5) – (5.10).

$$\begin{aligned}
 C &= C_MCMB + C_ACCU \\
 &= C_CS + \sum_{i=1}^N (L_i' - 1) + \sum_{i=1, h(i) \neq 0}^N 1 - 1 \\
 &= C_CS + \sum_{i=1}^{\lceil N/2 \rceil} (L_i' - 1) + \sum_{i=1, h(i) \neq 0}^N 1 - 1 \quad \text{for symmetric filter} \quad (5.11)
 \end{aligned}$$

Another factor that influences the performance of an implemented filter is logic depth. The logic depth is contributed by the coefficients with the greater number of the CSPT terms. Since the logic depth of a common subexpression of weight two is equal to one adder delay, the logic depth, D is given by:

$$D = \max(\lceil \log_2 |V_1| \rceil, \max_{i=2}^{|L|} \lceil \log_2 |V_i| \rceil + 1) \quad (5.12)$$

where $|V_i|$ is the number of vertices of the subgraph G_i and $\lceil a \rceil$ denotes the smallest integer larger than a . As the critical path will traverse an adder of the accumulator line, the above logic depth calculation has included that adder delay of the accumulator. For V_i ($i \geq 2$) one depth is added as the accumulator cannot be merged with other adders.

5.4.3 Example

Consider the application of CRAH-2 to a small coefficient set, $H = \{10\bar{1}010\bar{1}000\bar{1}, 010010\bar{1}0\bar{1}00, 100100\bar{1}0\bar{1}00\}$. The sorted common subexpression list, S is shown in Table 5.1. The first column “order” refers to the order number of common subexpression defined in *Definition 5.10* of Section 5.2.1. The pairs of indices, $(index(v_i), index(v_j))$ of the edges, $e = v_i v_j \in E$ on the admissibility graph corresponding to the subexpressions are also listed for ease of reference.

Table 5.1 Sorted common ubexpression list, S

Order	Frequency	Common Subexpression	Expression	$(index(v_i), index(v_j))$
1	3	$10\bar{1}$ or $\bar{1}01$	$x_{h1} = x - x \ll 2$	h_1 (4, 6), (6, 8) or (8, 10)
				h_2 (4, 6)
2	3	$100000\bar{1}$ or $\bar{1}000001$	$x_{h2} = x - x \ll 6$	h_1 (0, 6) and (4, 10)
				h_3 (4, 10)
3	2	101 or $\bar{1}0\bar{1}$	$x_{h3} = x + x \ll 2$	h_2 (2, 4)
				h_3 (2, 4)
4	2	1001 or $\bar{1}00\bar{1}$	$x_{h4} = x + x \ll 3$	h_2 (6, 9)
				h_3 (7, 10)
5	2	10001 or $\bar{1}000\bar{1}$	$x_{h5} = x + x \ll 4$	h_1 (4, 8) and (6, 10)
6	2	$10000\bar{1}$ or $\bar{1}00001$	$x_{h6} = x - x \ll 5$	h_2 (4, 9)
				h_3 (2, 7)

Figure 5.6 shows the labeling of all order 1 subexpressions, $10\bar{1}$ ($\bar{1}01$), on the admissibility graph. An arc is used to cover all vertices of a subgraph representing a single coefficient. The precedence edges are recorded as $E_P = \{v_{12}v_{13}, v_{14}v_{15}, v_{22}v_{23}\}$. Similarly, the contention edges are recorded as $E_C = \{v_{13}v_{14}\}$. The contention edge, $v_{13}v_{14}$ bridging the two precedence edges does not fulfill the criterion of contention resolution

since both endpoints of $v_{13}v_{14}$ are fixed.

Figure 5.6 Labeling of order 1 subexpressions

Figure 5.7 shows the labeling of the order 2 subexpressions. Since the endpoints of contention edges, $v_{13}v_{14}$ and $v_{12}v_{15}$ are fixed, the precedence edges $v_{14}v_{15}$ do not meet the resolution criterion. There is a contention between $v_{12}v_{13}$, however, no change is made since $NAS = 0$.

Figure 5.7 Labeling of order 2 subexpressions

We continue with the labeling of order 3 subexpressions in Figure 5.8.

Figure 5.8 Labeling of order 3 subexpressions

There are contentions in h_2 and h_3 but no resolution is made, as again $NAS = 0$ for these two contentions. We proceed with the labeling of order 4 subexpressions in Fig. 5.8. Two

interesting contentions occurred in order 4 subexpressions of Figure 5.9.

Figure 5.9 Labeling of order 4 subexpressions

One occurs between the precedence edge $v_{22}v_{23}$ and contention edges $v_{21}v_{22}$ and $v_{23}v_{24}$ in h_2 and the other between the precedence edge $v_{32}v_{34}$ and contention edges $v_{31}v_{32}$ and $v_{33}v_{34}$ in C_3 . At this point, $E_P = \{v_{12}v_{13}, v_{14}v_{15}, v_{22}v_{23}, v_{32}v_{34}\}$, $E_C = \{v_{11}v_{13}, v_{12}v_{15}, v_{13}v_{14}, v_{21}v_{22}, v_{23}v_{24}, v_{31}v_{32}, v_{33}v_{34}\}$. $NAS(v_{22}v_{23}) = NAS(v_{32}v_{34}) = 1$ are set. Since there is no equivalent edge in E_P with order 3 and 4, $NAS(v_{22}v_{23})$ and $NAS(v_{32}v_{34})$ remain as 1. As there exists equivalent precedence edges of order 1 in E_P , $NAS(v_{22}v_{23})$ remains unchanged. Since there is no equivalent edge of order 2 in E_P to $v_{32}v_{34}$, an adder is needed to realize $v_{32}v_{34}$ and $NAS(v_{32}v_{34}) = 1 + 1 = 2$. The cost of constructing the non-existing precedence edges of order 3 and 4 is 2. Therefore, if the contentions are considered independently, $NAS(v_{22}v_{23}) < 2$ and $NAS(v_{32}v_{34}) = 2$, no swapping will be carried out. However, as the aggregate $NAS = NAS(v_{22}v_{23}) + NAS(v_{32}v_{34}) = 3 > 2$, one adder can be saved by simultaneously exchanging the precedence edges, $v_{22}v_{23}$ and $v_{32}v_{34}$ with their respective adjacent contention edges. The result is $E_P = \{v_{12}v_{13}, v_{14}v_{15}, v_{21}v_{22}, v_{23}v_{24}, v_{31}v_{32}, v_{33}v_{34}\}$ and $E_C = \{v_{11}v_{13}, v_{12}v_{15}, v_{13}v_{14}, v_{22}v_{23}, v_{32}v_{34}\}$, as shown in Figure 5.10.

Figure 5.10 Results of the contention resolution

The final list of common subexpressions is given by the precedence edges of E_p . The contention edges are removed from the graph as shown in the final AG in Figure 5.11. The number of adders, C required to implement the coefficient set, H can be calculated from (5.11). There are three distinct common subexpressions of order numbers 1, 3 and 4, so $N_{CS} = 3$. $C_{MCM} = 3 + 2 + 1 + 1 = 7$ and $C_{ACC} = 2$. Thus, the overall adder cost $C = 7 + 2 = 9$. From (5.12), the logic depth, $D = 3$.

Figure 5.11 AG solution graph of CRAH-2 for Example 2

The AG solution can be directly mapped to the hardware architecture. The steps involved are:

- (i) The weight-2 common subexpressions corresponding to the distinct precedence edges are generated. One adder is needed to generate the partial sum for each order number.
- (ii) Free vertices are summed with the generated partial sums to complete the MCM block. The adder tree of partial sums generated in (i) and the free vertices are balanced to obtain the minimum depth tree for speed.
- (iii) The products of the input and coefficients are accumulated to the final output.

To balance the adder tree in Step (ii), free vertices of each G_i are arbitrarily paired and added at level 0. As the partial sums reach adder level 1, they are then added with the partial sums of the precedence edges. This way of allocating the sparse vertices and edges according to the arrival time of partial sum outputs will minimize the logic depth without increasing the number of operators. For vertical and oblique subexpressions (discussed in Chapter 6), delay elements are embedded in the subexpression generation step (i). Figure 5.12 shows the architecture of mapping the AG of Figure 5.11.

Figure 5.12 Circuit implementation.

5.5 Experimental Results

Two methods were used to evaluate the performances of CRAH-2. First, several benchmark filters commonly used for performance evaluation were tested. The qualities of the generated solutions, in terms of the number of logic operators (LO) and logic depth (LD) were compared with the reported results in the literature. Second, a more rigorous experimentation was also performed using randomly generated coefficient sets composed of different filter lengths, N and coefficient word lengths, l . The direct canonical signed digit (CSD) implementation, Paško [PAS99] and nonrecursive signed common subexpression elimination (NRSCSE) [MAR02] algorithms were prototyped and run on the same machine to show the efficiency of CRAH-2.

The experimental results for the comparison of various algorithms are shown in Table 5.2. FIR4 to FIR6 are symmetrical filters and the values of N refer to the total number of taps.

BHM, RAG- n , C1 and Hartley refer to the Modified Bull and Horrock [DEM95], the n -Dimensional Reduced Adder Graph [DEM95], Logic depth minimized RAG- n (for short wordlength) or BHM (for long wordlength) [DEM02] and Hartley [HAR96] algorithms, respectively. For those algorithms that are not implemented, their results are taken from [MAR02] directly. For consistency, the number of logic operators presented in Table 5.2 has included the accumulator associated with each filter tap.

Table 5.2 Simulation results on FIR filters.

Filter	N	l	CSD		Hartley		BHM		RAG-n		C1		Paško		NRSCSE		CRAH-2	
			LO	LD	LO	LD	LO	LD	LO	LD	LO	LD	LO	LD	LO	LD	LO	LD
FIR1	3	11	12	3	10	3	9	6	9	-	9	4	10	3	10	3	9	3
FIR2[MAR02]	4	8	14	3	10	3	9	6	9	-	10	3	10	4	9	3	9	3
FIR3[POT96]	4	11	17	4	13	4	11	6	10	-	11	4	11	5	13	4	13	4
FIR4[PAS99]	25	8	35	3	33	3	30	3	30	-	30	3	30	3	30	3	30	3
FIR5[PAS99]	60	13	115	3	99	3	85	5	84	-	88	3	91	3	94	3	94	3
FIR6[HAR96]	121	14	265	4	204	4	172	8	171	-	172	8	179	5	189	4	183	4

It can be seen from Table 5.2, the proposed CRAH-2 always achieves the same lowest logic depth as NRSCSE and Hartley, but CRAH-2 outperforms NRSCSE and Hartley in the number of logic operators. When the filter taps and the wordlengths increase, the number of logic operators saved by CRAH-2 over NRSCSE increases with a shorter computation time, as evinced in Figures 5.13 and 5.14 to be shown later. All the other algorithms have higher logic depths. The number of logic operators of CRAH-2 is also comparable to Paško, which is an algorithm that seeks for optimal common subexpressions of all weights. Comparing with the GD algorithms, CRAH-2 may not produce lower adder cost solutions in some cases but the logic depths of its solutions are always lower. It is evident from Table 5.2 that the logic depth of its solutions can be as low as halve those of the solutions provided by the GD algorithms. With around 5% overhead in adder cost, the speed of FIR6 produced by CRAH-2 is doubled that of C1. Another advantage of CRAH-2 is that the run-time is significantly faster than those of the GD algorithms. Being a heuristic algorithm, CRAH-2 generates results which are asymptotically close to the minimal achievable logic complexity modeled and solved by the ILP algorithms [YUR99] for coefficients represented in CSD form. For the three

examples given in [GUS02a], CRAH-2 gives exactly the same number of logic operators as [YUR99] as both optimization algorithms use only weight-2 common subexpressions. In short, CRAH-2 promises good overall performance in view of minimal logic depth for solutions with commensurate optimality of logic complexity.

For more general comparisons, the performances of CRAH-2, NRSCSE and Paško algorithms were tested on randomly generated data sets. Every data set used in this test was taken from 50 sets of randomly generated coefficients. First, CRAH-2 and NRSCSE were compared under different word lengths and different number of taps, as shown in Figure 5.13. NRSCSE is one of the latest algorithms based on CSD representation and produce excellent solutions with special emphasis on minimizing the logic depth. It is evident from the results of Figure 5.13 that CRAH-2 performs well in terms of logic complexity. In general, CRAH-2 saves 1% to 3% more logic operators than NRSCSE. As the size of the coefficient set increases, the predominance of CRAH-2 over NRSCSE becomes discernible.

Figure 5.13 Comparison of LO between CRAH-2 and NRSCSE.

The average adder costs for filters with different number of taps at a fixed word length were simulated and average adder cost and computation time were charted in Figure 5.14. The MATLAB programs were run on a Pentium IV 1.9 GHz personal computer with 256 MB of system memory. The word lengths of the coefficients were fixed at 11. It is obvious that CRAH-2 works well at the common word length. The least logic depth CRAH-2 has adder cost comparable to Paško algorithm while it consumes the least CPU time in all the test cases.

Figure 5.14 Comparison of average adder cost and computation time of CSD, NRSCSE, Paško, and CraH-2

5.6 Summary

A new contention resolution algorithm CRAH-2 has been proposed and presented in this chapter. CRAH-2 is able to circumvent the local minimum problem encountered by the existing CSE algorithms. A novel admissibility graph is employed as an efficient data structure for the proposed algorithm, and this synthesis approach is unique in its fundamental concept of operations. The benefit of the algorithm is derived from their ability to appraise and substitute the chosen subexpressions when better alternatives emerge. Reduction in the number of logic operators is used directly as the criterion of the appraisal. Logic depth is also minimized as only horizontal weight-2 common subexpressions are applied. The proposed algorithm CRAH-2 provides promising results. Comparison with related work based on randomly generated filter coefficients shows that the logic depth driven algorithm produces low cost circuits with shorter critical path. Moreover, our algorithms are efficient. The runtime of CRAH-2 is comparatively lower than NRSCSE and significantly lower than Paško algorithm.

In the next chapter, derivatives of contention resolutions are developed to design low complexity FIR filters with greater adder cost savings through versatile manipulation of higher weight common subexpressions based on the same admissibility graph and contention resolution concept.

Chapter 6

Versatile Contention Resolution Algorithms

6.1 Introduction

Common subexpression elimination has been an eminent means to eliminate redundant computations in the MCM blocks of digital filters. Prevailing common subexpression elimination (CSE) algorithms use horizontal common subexpressions predominantly. In Chapter 5, a new data structure and a fresh approach called the contention resolution have been introduced to effectively eliminate horizontal common subexpressions of weight-2. This chapter will leverage on the concept of contention resolution algorithm (CRA) to exploit both the intra- and inter-coefficient redundancy to further mitigate the logic depth and logic complexity of the MCM block. Intra-coefficient common subexpressions are formed by nonzero digits from within the same coefficient and inter-coefficient common subexpressions are formed by nonzero digits from different coefficients. By representing the set of canonical signed digit (CSD) coefficients in a two-dimensional table [HAR96], horizontal (intra-coefficient), vertical and oblique (inter-coefficient) groups of identical pattern cells can be identified. The prowess of CRA presented in this chapter lies in its versatility to handle common subexpressions of different weights in both dimensions of the coefficient array in a comprehensive manner. The basic CRA presented in the last chapter can spawn a number of derivatives by noting that higher-weight intra- and inter-coefficients correspond to longer paths within a coefficient or spanning across coefficients in the admissibility graph.

Higher weight common subexpressions have been explored in many algorithms recently, including the CSE and the GD algorithms. Many heuristic algorithms [HAR96, PAR02, MAR02, YAO04] use weight-2 subexpressions as the primitive elements for reuse and then search for higher weight common subexpressions from existing lower weight common subexpressions. However, this process tends to discriminate the higher weight subexpressions from being selected as their nonzero digits have already been preoccupied by the lower weight common subexpressions. On the contrary, some algorithms [PAS99, POT96] search for the highest weight common subexpressions at the outset followed by splitting them into the lower weight common subexpressions for further sharing. This is to insure that the higher weight common subexpressions do not always give in to the lower weight subexpressions when they are overlapped. It is noted later from the experimental results presented in Section 6.5 that admission of higher weight common subexpressions does not always lead to the reduction of total adder cost. Some algorithms [HAR96, POT96] enhance the optimization by including more factors and enlarging the search scope, but the qualities of the solutions improve only marginally. This is because gradient decent approaches, though efficient, are sensitive to the order of execution. There is no feedback and back tracking to allow efficient resubstitution of lower cost common subexpressions in case of conflicts arising from the nullification of potential subexpression elimination caused by early decisions in the execution process.

As discussed in Chapter 5, the proposed contention resolution algorithms are able to unweave a precedential decision of suboptimal common subexpressions by a localized cost function evaluation when there is a conflict between the selections of two competitive subexpressions. The contention resolution algorithms start with weight-2 common subexpressions, but seek for greater adder cost saving through higher weight common subexpressions when opportunity arrives. For every generation of higher weight common subexpressions, saving of at least one adder is ensured. Therefore, the

contention resolution algorithms target strictly at lowering complexity without rigid restriction imposed by previous decision or preference to higher weight common subexpression. The proposed derivatives of the basic contention resolution algorithm CRAH-2 are versatile in that they are capable of satisfying search for both intra- and inter-coefficient common subexpressions, in any legitimate composition of horizontal, vertical and oblique common subexpressions. Without unduly extending its logic depth, minimal logic depth can also be achieved by weight constrained derivatives of the proposed CRA. In fact, the logic depth can be controlled by constraining the length of the precedence paths in the admissibility graph. To get away from the inferior local minimal solution, the contentions are resolved through evaluation of adder cost of localized pattern graphs in admissibility graph. The variants of contention resolution algorithm complies with an important tenet of good heuristic that significant improvement in the solution quality (in terms of logic complexity and logic depth) is attained with increased problem size but the computational time remains well bounded. Although the algorithms proposed in this chapter use the CSD coefficients synthesized by any of the existing techniques [CHE99, HEW00, LIM83, LIM90] to reduce the implementation cost (logic complexity) of MCM block, they are equally applicable to coefficients represented in other signed digit forms.

The rest of this chapter is organized as follows: Section 6.2 describes the three types of common subexpressions considered by the CRAs of this chapter. Section 6.3 introduces the scenarios for which precedence paths in the admissibility graph are generated via contention resolution. Section 6.4 presents the derivatives of contention resolution algorithms with different weight constraints. Simulation results and performance comparisons are presented and discussed in Section 6.5. The chapter is summarized in Section 6.6.

6.2 Common Subexpressions

The optimality of CSE algorithms relies on the identification of good common subexpressions. There are two main types of subexpressions: an intra-coefficient (horizontal) subexpression is a partial sum of two or more nonzero digits from the same coefficient; and an inter-coefficient subexpression is formed from nonzero digits in different coefficients. Based on the orientation of the nonzero digits of the subexpressions in the coefficient array, inter-coefficient subexpressions can be further classified as vertical or oblique subexpressions. In vertical subexpression, the nonzero digits are located in the same column whereas the oblique subexpression has its nonzero digits located in different columns. Figure 6.1 illustrates the three different types of subexpressions. For ease of exposition, $x[-i]$ denotes the value of x after i sample delays and $\ll j$ the left shift of j digits. In a two dimensional (2-D) view, i and j can be deemed as the height and length of the subexpressions. $i = 0$ for horizontal subexpression, $j = 0$ for vertical subexpression, and i and j are both nonzero for oblique subexpression.

Figure 6.1 Notations for different types of subexpressions

Any type of subexpression is said to be common if its cell pattern is not unique in a coefficient array. Since adders and subtractors are assumed to be equivalent, subexpressions with identical pattern but differ only in the signs of every nonzero digit

consistently are also said to be common. It is assumed that shift operations are essentially free, since they can be hard-wired.

The above three kinds of common subexpressions were coined by Hartley [HAR96], but he chose to base only on the two most frequently used horizontal subexpressions (101 and $10\bar{1}$) for optimization.

Example 6.1: A small coefficient set from [HAR96] is used to illustrate the effects of sharing different types of common subexpressions. For expediency, only weight-2 common subexpressions are identified. The four solution sets shown in Figure 6.2 are respectively, the CSEs for horizontal common subexpressions only, vertical common subexpressions only, oblique common subexpressions only, and combined horizontal and vertical common subexpressions. It is not easy to tell from this small example which kinds of common subexpressions are more beneficial, but it provides a good perspective on the reuse of different types of common subexpressions and the distributions of delay elements for each structure.

Based on the CSD representation, the original coefficient set of Figure 6.2 can be expressed as:

$$\begin{aligned}
 y = & -x \ll 11 + x \ll 9 + x \ll 7 + x \ll 1 \\
 & -x[-1] \ll 10 - x[-1] \ll 6 - x[-1] \ll 4 - x[-1] \ll 2 - x[-1] \\
 & -x[-2] \ll 10 + x[-2] \ll 7 + x[-2] \ll 1 \\
 & +x[-3] \ll 11 + x[-3] \ll 5 - x[-3] \ll 3
 \end{aligned} \tag{6.1}$$

There are 15 nonzero digits in this example and 14 adders are needed.

a. Horizontal common subexpressions only

b. Vertical common subexpressions only

c. Oblique common subexpressions only

d. Horizontal and vertical common subexpressions

Figure 6.2 Composite common subexpression eliminations

For elimination of only horizontal common subexpression as in Figure 6.2a, the best solution finds only one common subexpression, x_{hc} with a reuse frequency of 5. Therefore, four adders are saved.

$$x_{hc} = x \ll 6 + x \tag{6.2}$$

$$\begin{aligned}
y &= -x \ll 11 + x \ll 9 + x_{hc} \ll 1 \\
&\quad -x_{hc}[-1] \ll 4 - x_{hc}[-1] - x[-1] \ll 2 \\
&\quad -x[-2] \ll 10 + x_{hc}[-2] \ll 1 \\
&\quad + x_{hc}[-3] \ll 5 - x[-3] \ll 3
\end{aligned} \tag{6.3}$$

For elimination of only vertical common subexpressions as in Figure 6.2b, the best solution detects only one common subexpression, x_{vc} with a reuse frequency of 2. As the number of filter taps increases, more vertical common subexpressions are likely to be found. The common subexpression and the solution are expressed in (6.4) and (6.5), respectively. As delay elements are embedded in the subexpression implementation, it changes the structure of the filter.

$$x_{vc} = x + x[-2] \tag{6.4}$$

$$\begin{aligned}
y &= x_{vc} \ll 7 + x_{vc} \ll 1 \\
&\quad -x \ll 11 + x \ll 9 \\
&\quad -x[-1] \ll 10 - x[-1] \ll 6 - x[-1] \ll 4 - x[-1] \ll 2 - x[-1] \\
&\quad -x[-2] \ll 10 \\
&\quad + x[-3] \ll 11 + x[-3] \ll 5 - x[-3] \ll 3
\end{aligned} \tag{6.5}$$

For elimination of only oblique common subexpressions as in Figure 6.2c, the best solution finds only one oblique common subexpression and it is reused 5 times. The subexpression and the solution are expressed in (6.6) and (6.7), respectively. Like the case for vertical common subexpressions, oblique common subexpressions also reallocate latches or introduce new embedded latches.

$$x_{oc} = x - x[-1] \ll 1 \tag{6.6}$$

$$\begin{aligned}
y &= x_{oc} \ll 9 + x_{oc} \ll 1 - x_{oc}[-1] - x_{oc}[-1] \ll 6 - x_{oc}[-2] \ll 10 \\
&\quad -x \ll 11 + x \ll 7 \\
&\quad -x[-1] \ll 4 \\
&\quad + x[-3] \ll 5 - x[-3] \ll 3
\end{aligned} \tag{6.7}$$

For composite horizontal and vertical common subexpression elimination as in Figure 6.2d, the two horizontal common subexpressions, as well as the only vertical common subexpression are reused twice each. Three adders are saved altogether. The common subexpressions and the solution are expressed in (6.8) to (6.11) accordingly.

$$x_{hc1} = x - x \ll 2 \quad (6.8)$$

$$x_{hc2} = x + x \ll 2 \quad (6.9)$$

$$x_{vc} = x + x[-2] \quad (6.10)$$

$$\begin{aligned} y &= -x_{hc1} + x_{vc} \ll 7 + x_{vc} \ll 1 \\ &\quad -x[-1] \ll 10 - x_{hc2}[-1] \ll 4 - x_{hc2}[-1] \\ &\quad -x[-2] \ll 10 \\ &\quad +x[-3] \ll 11 + x_{hc1} \ll 3 \end{aligned} \quad (6.11)$$

Including all types of subexpressions into consideration will generally but not always enhance the probability of getting the least cost solution. For this small example, it is not possible to comprehend the intricate contentions and competitive effects of the simultaneous elimination of all three types of common subexpressions. For example, oblique common subexpressions are not so useful for symmetrical filter structures, as their elimination will devastate the symmetry. This is because oblique subexpressions are not repeatable in the symmetrical structure. For example, the symmetric term for $x_{oc} = x - x[-1] \ll 1$ is $x_{sym} = -x \ll 1 + x[-1]$, which can not be obtained from the output of the same adder. Owing to the retiming caused by latch insertion or reallocation, the first half of coefficients that are generated from oblique common subexpressions can not be reused in the second half. For filters with high number of repeated coefficients, inter-coefficient common subexpression eliminations tend to inhibit the reuse of these coefficients.

The same set of coefficients is optimized with higher weight common subexpressions. Three weight-3 common subexpressions have been identified in Figure 6.3. The weight-3 common subexpression contains a weight-2 horizontal subexpression that can itself be reused for the weight-2 horizontal subexpression of the first coefficient. The common subexpressions and the solution are expressed in (6.12) to (6.14) accordingly.

$$x_{hc} = x \ll 6 + x \quad (6.12)$$

$$x_p = x \ll 5 - x_{hc}[-1] \quad (6.13)$$

$$\begin{aligned} y &= -x \ll 11 + x_p \ll 4 + x_{hc} \ll 1 \\ &\quad -x_p[-1] \ll 1 - x[-1] \ll 2 - x[-1] \\ &\quad -x_p[-2] \ll 5 \\ &\quad -x[-3] \ll 3 \end{aligned} \quad (6.14)$$

Altogether 9 adders are required for the implementation of Figure 6.3. Compared with the above solutions which involve only weight-2 common subexpressions, the use of higher weight common subexpressions can reduce the adder cost further. In order to explore higher weight common subexpressions for greater savings, some weight-2 common subexpressions may have to be dismantled to pave the way for the generation of higher weight common subexpressions. Under those circumstances, contention resolutions are needed. In fact, the weight-3 common subexpressions of Figure 6.3 are weight-3 path formed by a horizontal edge and an oblique edge in the admissibility graph of the coefficient set. The horizontal edge is first created for the two isolated vertices in the subgraph of the first coefficient.

Figure 6.3 Higher weight common subexpression eliminations

6.3 Contention Resolutions for higher weights

A path in the admissibility graph represents a higher weight subexpression. From Definition 5.5 in Chapter 5, a path η of weight- w consists of w vertices $(v_1, v_2, \dots, v_w) \subset V$ and $w - 1$ edges $(e_1, e_2, e_3, \dots, e_{w-1}) \subset E$. A path η , denoted by $e_1, e_2, \dots, e_{w-1}$, presents for the sequence of vertices and edges, $v_1, e_1, v_2, e_2, \dots, v_{w-1}, e_{w-1}, v_w$ if the endpoints of the path η is v_1 and v_w , and successive vertices v_i and v_{i+1} are endpoints of the intermediate edge e_i . A precedence path is a path within which every edge is a precedence edge. Any path that contains a contention edge and has potential to become precedence path is called a candidate path. A weight- n precedence path will be generated from a weight- $(n-1)$ precedence path and an adjacent edge if it is profitable. Figure 6.4 shows the conversion of two candidate paths to a weight-4 precedence path.

Figure 6.4 Generation of weight-4 precedence path from weight-3 precedence paths

In Figure 6.4, the two paths on the left are candidate paths. An evaluation will be carried out for each candidate path to determine if they will be converted to a higher weight precedence path. When it is resolved that a candidate path should be converted to a precedence path, some adders will be saved. Precedence path $v_1v_2v_3v_4$ on the right of Figure 6.4 may be developed from either precedence paths of $v_1v_2v_3$ or $v_2v_3v_4$ on the left part of Figure 6.4 as a result of contention resolution. In some situations, contention resolutions are needed when there are precedence edges emanating from the candidate path, and the resolution may require that the precedence edges be abandoned by changing

them to contention edges to avoid conflict. This is illustrated by an example in Figure 6.5. The precedence edge v_3v_4 emanating from an endpoint v_3 of the candidate path $v_1v_2v_3$ is converted to a contention edge if there is adder saved in forming a precedence path $v_1v_2v_3$ of weight-3. The calculation of NAS for the candidate paths is more complicated because all equivalent paths in the entire admissibility graph need to be considered simultaneously. Sometimes one candidate path alone may not bring adder saving due to contention or the saving is not sufficient to offset the additional adders needed for generating the higher weight precedence path. However, the chance of saving adders will generally be enhanced by considering the reusability of common precedence paths.

Figure 6.5 Contention Resolution in path generations

Contention path resolution is more complicated than contention edge resolution described in Chapter 5. Let SG_1 and SG_2 be the subgraphs comprising the candidate path and its emanating paths before and after the conversion. The number of adders saved, NAS is equal to the difference in adder costs between SG_1 and SG_2 . Only precedence edges and paths emanating from the candidate path that can have their attributes changed after the conversion need to be considered. Each additional contention edge generated by the conversion induces an adder. Breaking an emanating precedence path creates a hybrid path (one that contains at least one contention edge) and introduces an adder if the frequency of equivalent emanating precedence paths is greater than two. If the frequency of the equivalent emanating precedence path is two, changing it to a hybrid path will cause the remaining equivalent path to become hybrid as well, which incurs one more adder. Let Δe_p be the number of precedence edges in SG_1 that have been changed to

contention edges in SG_2 . Let $n_{p \rightarrow h}(f=2)$ and $n_{p \rightarrow h}(f>2)$ be the number of precedence paths of frequency two and of frequency greater than two, respectively in SG_1 that have been changed to hybrid paths in SG_2 . If $n(e_c)$ is the number of contention edges of the candidate path η ,

$$NAS(\eta) = n(e_c) - \Delta e_p - n_{p \rightarrow h}(f>2) - 2 n_{p \rightarrow h}(f=2) \quad (6.15)$$

Equivalent candidate paths with positive NAS cost metric will be converted to precedence path by promoting their contention edges to precedence edges if their aggregate saving of adders is greater than the cost of generating the final precedence path, τ , i.e.,

$$\sum_{i=1}^{|\eta_c|} NAS_i > \tau \quad (6.16)$$

where η_c is the set of equivalent candidate paths that have positive NAS . The threshold τ for promoting the equivalent candidate paths, $\eta = e_1 \ e_2 \ \dots \ e_{w-1}$ to precedence paths is given by:

$$\tau = \sum_{j=1}^{w-1} \tau_j \quad \begin{cases} \tau_j = 0, & \exists e_p \in E_p, \text{order}(e_p) = \text{order}(e_j) \\ \tau_j = 1, & \text{otherwise} \end{cases} \quad (6.17)$$

Edge contention resolutions of Chapter 5 will still preformed before contention resolution for candidate paths described above are carried out. Edge contention resolutions are given higher priority because for the same amount of adders saved, solutions involving lower weight common subexpressions always simplify the circuit implementation and minimize logic depth for higher throughput rate. Precedence paths will only be generated from the contention resolutions of candidate paths when their edge contentions have been resolved.

6.4 Contention Resolution Algorithms

6.4.1 General description and classifications of CRAs

Application of contention resolution to eliminate different weight and different types and of common subexpressions described in Section 6.2 and their composition leads to several derivatives of the basic contention resolution algorithm presented in Chapter 5. These heuristic algorithms all operate on the same base data structure of the admissibility graph of the signed digit coefficient set. They possess the ability to rectify poor decisions made in earlier iterations or postpone ambiguous decisions to later stage to avoid poor suboptimal solution. Generalization of contention resolution algorithm based on a common data structure to handle different types and weights of common subexpressions make the proposed concept versatile to different application constraints. The main steps involved in the basic CRA are:

- Step 1:* A list of weight-2 common subexpressions based on their recurrences in the target coefficient set is extracted. The common subexpressions are sorted in descending order of their frequencies of occurrences. The vertices of an unconnected admissibility graph (AG) based on the nonzero digits of the coefficient set are generated.
- Step 2:* The first common subexpression is removed from the sorted list. All its equivalent edges are labeled as either precedence edges or contention edges onto the AG.
- Step 3:* If there is any contention, the costs, NAS, are evaluated to resolve the contention.
- Step 4:* Steps 2 and 3 are repeated until the common subexpressions in the list have been exhausted

By extending edges to paths, and considering path resolution, higher-weight common subexpressions of all types can be found. Since higher-weight subexpressions are evolved

from lower-weight subexpressions in this process, the basic CRA described above can be weight-constrained to control the logic depth. By incorporating the considerations of different types and weight constraints of the common subexpressions, the CRA variants are classified with the abbreviations given in Table 6.1 and described in details in the following subsections.

Table 6.1 Classification of CRAs

Types of subexpressions considered	Weight (w_{max}) considered		
	$w_{max} = 2$	$w_{max} \leq N$	No limit
Horizontal	CRAH-2	CRAH- N	CRAH
Vertical	CRAV-2	CRAV- N	CRAV
Horizontal and vertical	CRAHV-2	CRAHV- N	CRAHV
Horizontal, vertical and oblique	CRA-2	CRA- N	CRA

The weight-2 algorithms aim at minimizing the adder cost with the least logic depth. The search for partial sums or common subexpressions to form the complete set of coefficients with minimal number of adders remains the key focus of most CSE algorithms today, with only few exceptions [DEM02, PAR01, MAR02] which have also considered logic depth as a secondary criterion in their optimization processes. Recent interest in low power research has added an incentive to reducing the logic depth of digital filter design to avoid accruing to high power-delay product. Some results reported in [DEM02] have revealed that reducing the logic depth of a circuit could potentially reduce the switching activities due to the shortening of paths via which glitches propagate. CRAH-2 in Table 6.1 for horizontal weight-2 common subexpressions has already been introduced in Chapter 5. Other CRAs in the same column with the hamming weight of different common subexpressions restricted to two follow similar procedures as CRAH-2. The weight- N algorithms are designed to flexibly limit the weight of the common subexpressions to only N , which trades logic depth for logic complexity. The value of N

can be used to strike a balance between logic complexity and logic depth under the throughput rate constraint. The weight- X (X denotes don't care) algorithms impose no constraint on the weights of the common subexpressions. The algorithms in this class strive for the maximal adder cost reduction by sharing most profitable common subexpressions irregardless of their weights. In what follows, the CRAs for weight- X and weight- N are presented below.

6.4.2 Contention Resolution Algorithms for weight- X Subexpressions

Compared with weight-2 common subexpressions, elimination of higher-weight common subexpressions offers greater potential savings in adder cost. In the weight- X CRAs, although there is no weight limit imposed on the common subexpressions, precedence paths are only formed when there is reduction in adder cost. The weight- X algorithms comprise two parts of contention resolutions. In the first part, contentions among the edges are resolved and no new path is generated. In the second part, paths are extended from edges and contentions involving paths are resolved. Precedence paths of progressively increased weights are evolved when there is saving in adder cost. Figure 6.6 gives the generic pseudocode of the weight- X CRAs.

```

1:  for (each  $C_i$  in  $H$ ,  $i = 1 \dots |H|$ )
2:      generate  $G_i$  from the nonzero digits of  $C_i$ ;
3:   $E_P = \emptyset$ ;  $E_C = \emptyset$ ;
4:  for ( $i = 1 \dots \max\_order$ ) {
5:      if ( $v_1, v_2 \in S_j$ , are both free, with  $order(S_j) = i$ ,  $S_j \in S$ ) {
6:          connect ( $v_1, v_2$ ) with precedence edge,  $e_p$ ;
7:           $order(e_p) = order(S_j)$ ;
8:           $E_P = E_P \cup e_p$ ; }
9:      else {
10:         connect ( $v_1, v_2$ ) with contention edge,  $e_c$ ;
11:          $order(e_c) = order(S_j)$ ;
12:          $E_C = E_C \cup e_c$ ; }
13:  for (each  $e_p = (v_1, v_2) \in E_P$ )

```

```

14:      if (contention) {          -- contention resolution required
15:          compute exchange cost,  $NAS$ ;
16:          if ( $NAS > 0$ ) { -- potential saving by exchange
17:              let  $e_1 \in E_C$  be the minimal-cost edge adjacent to  $v_1$ ;
18:              let  $e_2 \in E_C$  be the minimal-cost edge adjacent to  $v_2$ ;
19:               $E_P = (E_P \cup e_1 \cup e_2) - e_p$ ;
20:               $E_C = (E_C \cup e_p) - e_1 - e_2$ ; } } -- end for loop starts in line 4
21:      update  $P_C$ , and order for  $P_C$ ; -- If there is change in candidate paths
22:      for ( $q = 1 \dots \max\_order\_path$ ) {
23:          calculate  $\tau$ ,          --  $\tau$  is given in (6.17)
24:          for (each  $\eta_i$  from  $P_C$ , with  $order(\eta_i) = q$ ) compute  $NAS(\eta_i)$ ;
25:           $sum = \sum_{NAS(\eta_i) > 0} NAS(\eta_i)$ ;
26:          if ( $sum > \tau$ ) { -- potential saving in creating path with higher degree
27:              for (all  $\eta_i$ , with  $NAS(\eta_i) > 0$ ) {
28:                  Let  $E_p'$  be the set of emanating precedence edges from  $\eta_i$  to be converted;
29:                  Let  $E_c'$  be the set of contention edges in  $\eta_i$  before changed.
30:                   $E_P = (E_P \cup E_c') - E_p'$ ;
31:                   $E_C = (E_C \cup E_p') - E_c'$ ;
32:                  Update  $P_P$ ; } } }
33:      return  $E_P, P_P$ ;

```

Figure 6.6 Pseudocode of weight-X CRA

The lists, H and S have the same meanings as before (see Section 5.4.2). In addition, a separate order list of equivalent candidate paths, P_C and an order list of precedence paths, P_P are created in a similar manner as the order list S . P_C and P_P are initially empty and they are updated in the second part on every iteration (line 21). The frequencies of the equivalent paths are accounted each time a path is generated or an existing path is degraded. These paths are sorted dynamically by their weights and frequencies.

At the end of the process described in Figure 6.6, the precedence edges and paths in E_P and P_P form the common subexpressions. The implementation cost in terms of the total number of adders used can be determined from E_P , P_P and V_{free} . Let N_CS_w denote the number of distinct weight- w common subexpressions in E_P . Since unique common subexpression of weight-2 requires an adder, the cost of realizing the common

subexpressions of weight-2 is $C_CS_2 = N_CS_2$. The number of adders required to implement a path is not fixed since the weight-2 common subexpressions generated may be reused. A subexpression represented by a weight- w path, η_w may require from one to $w-1$ adders to construct depending on whether there exists other lower-weight paths equivalent to some segments of η_w . From experiments (see Table 6.5 of Section 6.5), it is observed that majority of the precedence paths are of weight-3. Two adders are needed to form this type of subexpressions with no existing weight-2 common subexpressions inside, one for forming an edge from the three connected vertices and one for adding this edge to the isolated vertex. There are only three possibilities of generating the edge from a weight-3 path: either two vertices is merged and then the remaining vertex are added. From the order numbers of the edges, it is easy to identify which formation actually contains a precedence edge that has already been implemented so that an adder can be saved. For higher-weight precedence paths η_w with $w > 3$, the algorithms look for $\eta_{w-1}(i)$ for $i = 1, 2$ from two decompositions of $\eta_w = \eta_{w-1}(i) + v(i)$, where $v(1)$ and $v(2)$ are the two endpoints of η_w . If $\eta_{w-1}(1)$ and $\eta_{w-1}(2)$ have not been generated, their decomposition to $\eta_{w-2}(i)$ continue as long as $w > 2$. If no lower-weight paths are found, the number of possibilities to realize an adder tree for the required subexpression is formidable. Hence, only two different decompositions are considered, either all odd or all even indexed edges are evaluated for the first level of adders. For a weight- w path $\eta_w = e_1 e_2 \dots e_{w-1}$, the odd decomposition consists of a set of subexpressions corresponding to all edges, e_i with odd index i while the even decomposition contains edges of even i . These two decompositions are evaluated in conjunction with the existing partial sums. The set of edges with the lower adder cost is chosen. C_CS_w ($w > 2$) is used to denote the adder cost for the generation of higher weight common subexpressions. Nonzero digits of each coefficient that are not parts of the final common subexpressions need to be summed with the subexpressions to produce the final coefficient. This cost for the MCM block is equal to the number of precedence edges or paths plus the number of free vertices less one. L_i is

the number of CSPT terms in i th coefficient as shown earlier in (5.7). Thus, the total number of adders of the final AG is given by:

$$\begin{aligned}
 C &= C_MCMB + C_ACCU \\
 &= \sum_{w \geq 2} C_CS_w + \sum_{i=1}^N (L_i' - 1) + \sum_{i=1, h(i) \neq 0}^N 1 - 1 \\
 &= \sum_{w \geq 2} C_CS_w + \sum_{i=1}^{\lceil N/2 \rceil} (L_i' - 1) + \sum_{i=1, h(i) \neq 0}^N 1 - 1 \quad \text{for symmetric filter} \quad (6.18)
 \end{aligned}$$

The generalized adder cost equation, (6.18) is applicable to all variants of CRA, including CRAH-2 of Chapter 5.

Since minimizing the logic depth under a fixed minimal number of adders offered by E_P , P_P and V_{free} is itself an optimization problem, especially for vertical and oblique subexpressions, it is not likely to provide a closed form estimation of the exact minimal logic depth until the implementation of the precedence paths is fixed. The following equation gives the upper bound of the logic depth from circuits synthesized by CRAH-X.

$$\begin{aligned}
 D &= \max(\max_{i=1}^{|L|-1} D_i + 1, D_L) \\
 D_i &= \left\lceil \log_2 \left(\frac{V_{free}}{\max_{\eta_i}(\text{weight}(\eta_i) - 1)} + |\eta_i| \right) \right\rceil + \max_{\eta_i}(\text{weight}(\eta_i)) - 1 \quad (6.19)
 \end{aligned}$$

Example 6.2: The same coefficient set as Example 6.1 is used as an example algorithm to illustrate CRA-X. The order list of subexpressions, S , is not shown due to the large number of combinations of any two nonzero digits when vertical and oblique subexpressions are also considered. There are two subexpressions appearing 5 times in the set, one horizontal and one oblique subexpressions given by $x_{hc} = x \ll 6 + x$ and $x_{oc} = x - x[-1] \ll 1$. As mentioned before, the horizontal subexpression is given higher

precedence. Order-1 edges are labeled in Figure 6.7. There is no contention at this stage. The precedence edges of the admissibility graph G are recorded as $E_P = \{v_{11}v_{12}, v_{21}v_{24}, v_{23}v_{25}, v_{31}v_{32}, v_{42}v_{43}\}$ and $E_C = \phi$.

Figure 6.7 AG after labeling of order 1 subexpressions

For order-2 subexpressions, Figure 6.8 is obtained after labeling the 5 oblique edges. All five order-2 edges are contention edges but three of them are constituents of candidate paths. The set of candidate paths in Figure 6.8 is given by $P_C = \{v_{13}v_{25}v_{23}, v_{24}v_{32}v_{31}, v_{33}v_{42}v_{43}\}$.

Figure 6.8 AG after labeling of order 2 subexpressions

For the three candidate paths, the NAS is calculated using (6.15): $NAS(v_{13}v_{25}v_{23}) = 1 - 0 - 0 - 0 = 1$, $NAS(v_{24}v_{32}v_{31}) = 1 - 1 - 0 - 0 = 0$ and $NAS(v_{33}v_{42}v_{43}) = 1 - 0 - 0 - 0 = 1$. The promotion threshold, $\tau = 1$ according to (6.17). Since $NAS(v_{13}v_{25}v_{23}) + NAS(v_{24}v_{32}v_{31}) + NAS(v_{23}v_{24}v_{25}) > \tau$, the candidate paths are promoted to precedence paths except $v_{24}v_{32}v_{31}$ because no adder will be saved by converting the candidate path $v_{24}v_{32}v_{31}$ to precedence path. The newly generated path is $x_p = x \ll 5 - x[-1] \ll 6 - x[-1] = x \ll 5 - x_{hc}[-1]$.

The same set of precedence edges and paths remains unchanged to the end of the process and the final AG is shown in Figure 6.9.

Figure 6.9 Final AG of Example 6.2

There are two common subexpressions found: $x_{hc} = x \ll 6 + x$ and $x_p = x \ll 5 - x_{hc}[-1]$.

The total number of adders used can be calculated from (6.17). Since $C_CS_2 = 1$, $C_CS_3 = 1$, $N_CS_2 = 3$, $N_CS_3 = 2$ and $SL = 3$, $C = 9$ adders are required to implement this set of coefficients. The common subexpressions are circled in the coefficient set shown in Figure 6.9.

Figure 6.10 Solution by CRA

The direct implementation of *Example 6.1* from CSD uses 14 adders/subtractors as shown in (6.1). Compared with the CRA solution given by (6.20), 5 adders are saved and the expression is more succinct.

$$\begin{aligned}
 y &= -x \ll 11 + x_{hc} \ll 1 + x_p \ll 4 \\
 &\quad -x[-1] \ll 2 - x_{hc}[-1] \\
 &\quad + x_{hc}[-2] \ll 1 - x_p[-2] \ll 5 \\
 &\quad -x[-3] \ll 3
 \end{aligned} \tag{6.20}$$

Comparing this result with the solution given by (6.3), the solution of CRA requires the same number of adders and shares the same edges and paths. The difference between the two designs is that the CRA solution has less higher-weight common subexpressions. This is due to the CRA's principle of expending computational effort only when it is worth. In CRA, higher-weight common subexpressions will only be generated only when there is adder saving.

To map the AG of Figure 6.8 to hardware architecture, the weight-2 common subexpressions corresponding to the distinct precedence edges are to be generated first. For precedence edges that bridge across two subgraphs, G_i and G_j , $j - i$ unit delay elements are inserted before the adder. Higher-weight common subexpressions are generated with endeavor to use shorter subexpressions. Finally the free vertices are summed with the generated subexpressions to complete the MCM block. The adder tree of the subexpressions and the free vertices are balanced to have the minimum depth. Figure 6.11 shows the mapping of Figure 6.9 to an adder tree following the above strategy.

Figure 6.11 Mapping of solution AG to adder tree

It should be noted that due to the following considerations, the already formed precedence path will not be decomposed or broken into lower-weight path in subsequent iterations. First, from rigorous experimental simulation, it has been found that the probability of saving adder costs and the amount of savings are very low by breaking the established precedence paths. Second, the complexity for evaluating if a precedence path shall be decomposed is high and because of the first reason, most of these extra computations end up to be redundant.

6.4.3 Contention Resolution Algorithms for weight- N subexpressions

With weight constraint, the algorithms for weight- N subexpressions will check the weight of each candidate path and process it only if it is less than or equal to N . The procedures for weight- N CRAs are similar to those for weight- X . These algorithms are useful for evaluating the tentative benefits of higher-weight subexpressions and the hypothetical relationship between MCM block characteristics, logic depth and different types of subexpressions. A study on the relations between the Hamming Weight of CSD coefficients, logic depth and the weights of paths was carried out and the observations are discussed as follows.

For horizontal common subexpressions, the logic depth of the MCM block can be constrained by limiting the maximum weight of the common subexpressions for each coefficient according to Table 6.2.

Table 6.2 Mapping between logic depth and path weight

Hamming weight of CSD coefficient	Upper bound of logic depth	Maximum permissible weight of precedence path
2	1	2
3	2	3
4	2	2
	3	4
5	3	3

	4	5
6	3	3
	4	4
	5	6
7	3	3 ($\times 1$ only)
	4	4
	5	5
	6	7
8	3	2
	4	4
	5	5
	6	6
	7	8

Table 6.2 is obtained by an exhaustive exploration of all possible adder tree topologies of CSD numbers with 2 to 8 nonzero digits. For a coefficient with two or three nonzero digits, the logic depth is fixed irrespective of how the coefficient is synthesized. Hence, the permissible weight of the precedence path is set equal to the hamming weight of the coefficient. As the hamming weight of the coefficient increases, adder trees of different depths can be used to synthesize the coefficient. If a coefficient is bounded by the logic depth given in Table 6.2, the weights of the precedence paths synthesized for that coefficient will not exceed the corresponding numbers given in the last column. For example, if the logic depth is constrained to 4, the coefficients of hamming weight 5 are allowed to have a precedence path of up to weight-5, and for the higher hamming weight coefficients, only weight-4 or lower-weight precedence paths are allowed. The coefficients of hamming weights lower than 5 need not be considered as their logic depths can never exceed 4. Figure 6.12 shows three different adder tree topologies for a CSD coefficient with a Hamming weight of 7. In Figure 6.12(a), $v_1v_2v_5v_7$ represents a weight-4 precedence path, which is generated from a precedence path $v_1v_2v_5$. Figure 6.12(b) shows a weight-4 precedence path, $v_1v_2v_3v_4$ generated from two weight-2 edges, v_1v_2 and v_3v_4 of the admissibility graphs. Although the coefficient of hamming weight 7 can also be synthesized at a lower logic depth of 3 as in Figure 6.12(b), it will refrain the weight-3 precedence path from being reused if one exists and can be expanded to the

desired weight-4 precedence path. Therefore, the maximum permissible path weights stipulated in Table 6.2 are stringent enough to guarantee the fulfillment of the logic depth constraints but some slacks are allowed for the reduction of adder costs. The topology of Figure 6.12(b) can also be used to illustrate the only exceptional case of Table 6.2 corresponding to the synthesis of a weight-7 coefficient under a logic depth constraint of 3. In this case, only one weight-3 precedence path is allowed (as annotated by “ $\times 1$ ” in Table 3) even if there exist two weight-3 precedence paths. If there is no logic depth constraint, two weight-3 precedence paths, $v_1v_2v_5$ and $v_3v_4v_6$, can be fitted into either adder tree topologies of Figure 6.12(a) or (c). Hence, by mapping the logic depth constraint to the maximum permissible weight of the precedence paths for each coefficient according to its hamming weight in Table 6.2, CRAH optimization can be driven by the logic depth constraint.

Figure 6.12 Adder tree topologies for a weight-7 coefficient

6.5 Simulations and Experimental Results

In this section, the performances of CRAs are evaluated and compared against benchmark filters and randomly generated coefficient sets. Based on the experimental results, the pro and cro of CRAs are analyzed and discussed.

The five practical filters that are chosen for benchmarking are widely used for comparison in the literature in this research area. FIR1 [PAS99, SAM89] is a 25-tap

lowpass filter with normalized passband and stopband edge frequencies of 0.15 and 0.25 respectively, and the ripple weighting factor, W is 1. The passband and stopband ripples are 0.005(or -46.0dB). FIR2 [PAS99, SAM89] is a 60-tap filter with normalized passband and stopband edge frequencies of 0.021 and 0.07, respectively. The passband ripple and stopband attenuation are 0.2 and 60 dB, respectively. FIR3 [LIM83] is 121-tap highpass filter with normalized stopband and passband edge frequencies of 0.37 and 0.4. Passband peak to peak ripple is 0.080dB and peak stopband ripple is -80.3dB . FIR1, FIR2 and FIR3 have the same specifications as FIR4, FIR5 and FIR6 introduced in Table 5.2. FIR 4 [VIN05b] is a LPFIR filter in the filter bank channelizer of the Digital Advanced Mobile Phone Systems (D-AMPS). The decimation is moved to the left of the bandpass filters and the sampling rate chosen is 34.02MHz. The channel filter extracts 30 kHz D-AMPS channel from the input signal after downsampling by a factor of 350. The pass band and stop band edges are 30 and 30.5 kHz, respectively. The peak pass band ripple is chosen to be 0.1 dB. The filter stop band attenuation is -24dB and the number of taps is 200. FIR5 [VIN05b] is a channel filter generally employed in the receivers for the Personal Digital Cellular standard. The sampling rate of the wide band signal is 25.6MHz, which covers 1024 channels of 25.5 kHz spacing. FIR5 has a stop band attenuation of -30dB and the number of taps is 230.

First, we compare the logic complexity (LO) and logic depth (LD) of our proposed CRAH algorithm with the existing algorithms as shown in Table 6.3. N is the number of taps of the filter and l is the input wordlength. Harley, BHM, RAG- n , C1, Paško and NRSCSE in the table refer to the Harley's [HAR96], Modified Bull and Horrock's [DEM95], Reduced Adder Graph [DEM95], Logic depth minimized RAG- n (for short wordlength) or BHM (for long wordlength) [DEM02], Paško's [PAS99] and nonrecursive signed common subexpression elimination (NRSCSE) [MAR02] algorithms, respectively. For consistency, the number of logic operators presented in Table 6.3

includes the accumulator associated with each filter tap. This accounts for the slight deviation in some of the values reported in the literature, which exclude these accumulators.

Table 6.3 Experimental results comparison for CRAH

Filter	N	l	CSD		Hartley		BHM		RAG- n		C1		Paško		NRSCSE		CRAH	
			LO	LD	LO	LD	LO	LD	LO	LD	LO	LD	LO	LD	LO	LD	LO	LD
FIR1	25	8	35	3	33	3	30	3	30	-	30	3	30	3	30	3	30	3
FIR2	60	13	115	3	99	3	85	5	84	-	88	3	91	3	94	3	91	3
FIR3	121	14	265	4	204	4	172	8	171	-	172	8	179	5	188	4	176	5
FIR4	200	13	363	4	293	4	254	6	254	-	254	4	253	5	274	4	251	4
FIR5	230	13	394	4	323	4	282	6	281	-	282	4	282	5	304	4	281	4

Table 6.3 shows that CRAH provides a good balance in the tradeoff of logic complexity and logic depth. Comparing with the CSD-based algorithms, which are Hartley, NRSCSE and Pasko, CRAH provides the least logic complexity for all the filters being compared. Thanks to the guiding principle of CRAs that the logic depth is allowed to increase only when there is saving in adders, CRAH’s solutions achieve near optimal logic depth. Even when it is compared with NRSCSE, which employs only weight-two subexpressions and is expected to have the minimal logic depth, CRAH have the same logic depth in all filters except for FIR3, where it has one logic depth higher. Nevertheless, the logic complexity of CRAH remains lower than NRSCSE for all five filters. Comparing with the GD algorithms like BHM and C1, CRAH has the shortest logic depth. In three out of five filters, CRAH solutions possess the least number of adders. It is not surprising that the GD algorithms outperform CSD-based algorithms in terms of logic complexity in some cases due to its much larger search space. As opposed to CSD-based algorithms, GD algorithms are not restricted by the fixed number representations. In principle, they have an unlimited number of potential patterns from which better common subexpressions can be selected. The disadvantage is the statistics of the common subexpressions for the entire coefficient set is not known in advance before the coefficients are fully composed from the fundamentals, which accounts for their poorer

results in some cases. GD algorithms generally perform poorer in terms of logic depth. Even with control optimization on logic depth, as in C1, the partial sums grow with the composition of the coefficients. This logic depth constraint has somewhat limited the GD algorithms to achieve better results.

CRA exploiting inter-coefficient common subexpressions is also compared in Table 6.4. As mentioned in Section 6.1, the adder cost of repeated coefficients will be aggravated by the inter-coefficient subexpressions. As the size of the coefficient set increases, the probability of having repeated coefficients increases. The annihilation of the repetitive patterns by the inter-coefficient common subexpressions is more likely to offset the benefits it gained. To mitigate this effect, a restriction has been imposed on CRA that no inter-coefficient common subexpressions are searched or evaluated in repeated coefficients. Under this restriction, only half of the taps of the symmetric filter need to be considered. Table 6.4 presents the number of adders used in the solutions for the half filter generated by various algorithms. The logic depth is not compared in Table 6.4 since the MCM block of CRA’s solution is not fully combinational. The inter-coefficient common subexpressions either insert or reallocate registers in the CRA’s solution.

Table 6.4 Adder cost comparison for CRA

Filter	l	BHM	RAG- n	C1	Paško	CRA
FIR1	8	18	18	18	18	18
FIR2	13	55	54	58	61	53
FIR3	14	112	111	112	119	110
FIR4	13	154	154	154	150	150
FIR5	13	169	168	169	169	166
	15	198	196	198	200	194

In Table 6.4, FIR5 is compared in two different wordlengths to show the effect of wordlength on adder cost. It is clear from Table 6.4 that CRA yields the least number of logic operators for the benchmark filters. As CRAs are designed to deal with any signed

digit representation, it is believed that better results will be yielded by applying CRA to MSD. As exhaustive execution on all possible MSD coefficient sets to find the best solution is formidable, restricting the search space for optimal MSD representations for each coefficient to be processed by CRAs is an ongoing research.

Being a heuristic algorithm, CRAs does not guarantee the solutions it synthesized will have the lowest logic depth. However, the results obtained are asymptotically close to the minimal achievable logic complexity modeled and solved by the ILP algorithms [GUS02a, YUR99] for coefficients represented in CSD form. For the three examples given in [GUS02a], CRAH-2 gives exactly the same number of logic operators as [YUR99] as both optimization algorithms use only weight-2 common subexpressions, and CRAH produces the same results as [GUS02a]. In short, CRAs promise good overall performance in view of its achievable logic depth for solutions generated with commensurate optimality of logic complexity.

For more general comparisons, the performances of CRAH-2, CRAH, CRA-2, CRA, NRSCSE and Pasko algorithms were tested on randomly generated data. Every data set used in this test was taken from 50 sets of randomly generated coefficients. The word length of the coefficients is fixed at 11 because pragmatically, wordlengths of 11 to 14 are most frequently encountered in applications that use fixed point filters [LIM90]. The adder costs normalized by the cost of CSD direct implementation shown in Figure 6.12 give an insight into the relative hardware savings achievable by different algorithms. It is obvious that CRAs outperform the other algorithms being compared. The results show that CRA is capable of generating better solutions with lower number of logic operators especially for large filters. The advantage of CRA over CRAH is its effectiveness in harnessing the sharing of inter-coefficient common subexpressions. Although arguably most FIR filter coefficients are not randomly distributed, the random simulation results

are still indicative of the benefits of sharing inter-coefficient common subexpressions for general MCM problems that can be solved by CRA.

Figure 6.13 Comparison of average adder cost

Due to the nature of the algorithms, the logic depth can be indirectly constrained on CRAs if desired. CRA- N will generate only common subexpressions of weight $w < N$ where N is a preset maximum weight limit. To the best of the author’s knowledge, there is only a few logic depth constrained common subexpression elimination algorithms [HAR96, MAR02] despite a few reported GD algorithms bear the theme of logic depth optimization [DEM02, PAR01]. Unlike the logic depth constrained CSE algorithm of [PAR01], the CRA- N incurs no overhead to the general CRA. Thus, it is of interest to scrutinize how the adder cost changes as MW (maximum weight allowed for precedence paths) increases. 50 randomly generated coefficient sets for each combination of the number of taps and coefficient word lengths were tested. The results are tabulated in Table 6.5. The number of coefficient sets that have the minimum number of logic operators from each N is recorded. The percentage of the least adder cost solutions obtained from each N is listed for every combination of taps and word length. For example, for the coefficient sets of tap = 40 and word length = 11, only 2% of the 50

randomly generated coefficient sets have the least adder cost when the maximum logic depth is constrained to two. This means that for this particular coefficient set, no better result can be achieved by allowing the logic depth to increase. There exist filter sets that exhibit a critical turning point at some N value beyond which the average adder cost will increase by relaxing the logic depth. This is an interesting phenomenon contrary to the conventional belief and the experimental results shown in [PAR01]. The number of coefficient sets that display this critical descend phenomenon (CDP) for each combination of N and l are accounted and its percentage is recorded in the last row labeled as %CDP of Table 6.5. The results of Table 6.5 show the following general trends. For the same word length, the larger the coefficient set, the higher the weight limit to obtain the least adder cost solution whereas for the same number of taps, the longer the word length, the higher the weight limit needed to achieve the best cost solution. The percentage CDP also increases as the order and precision of the filter increases. In Table 6.5, blank entry denotes zero percentage.

Table 6.5 Experimental results of CRAH- N with varying logic depth constraints

L	11				21				31			
N	20	40	60	80	20	40	60	80	20	40	60	80
$MW = 2$	14%	2%										
$MW = 3$	64%	58%	42%	36%	28%	18%	10%	24%	28%	14%	6%	18%
$MW = 4$	22%	40%	58%	64%	70%	58%	56%	46%	52%	56%	40%	38%
$MW = 5$					2%	24%	32%	30%	16%	30%	52%	44%
$MW = 6$							2%					
$MW = 7$									4%			
%CDP		4%	4%	2%	12%	6%	18%	18%	14%	10%	10%	24%

It should be noted that CRAH- N uses the same strategy when building higher weight subexpressions within the constraint of N . That is, higher-weight subexpressions are only created when it leads to a reduction in adder cost. From the %CDP of Table 6.5, it is conjectured that higher weight subexpressions may not always foster adder cost reduction, as one would expect. The problem may occur when the elimination of higher weight common subexpressions prevent the more beneficial elimination of some lower weight

common subexpressions whose number of occurrences is outweighing. Although the ways to obtain higher weight subexpressions vary from algorithm to algorithm, this local minimum exists, especially for the CSE algorithms that start the search from the highest-weight subexpressions.

To demonstrate the extent of redundancy that can be exploited from inter-coefficient common subexpressions (ICS) in comparison with intra-coefficient common subexpressions, statistic data are collected from 50 randomly generated coefficient sets of word length 11 and filter orders ranging from 10 to 80. The statistical results obtained from CRA-2 are shown in Table 6.6. The first row of Table 6.6 provides the filter order. The second row presents the average number of horizontal common subexpressions (HCS) appearing in the solutions for each filter order. The third row gives the average number of ICS, which includes vertical and oblique common subexpressions (VCS and OCS), appearing in the solutions for each filter order. The last row shows the percentages of ICS over all types of common subexpressions for each filter order. It should be noted that only weight-2 subexpressions are employed in CRA-2. The results show that the total number of VCS and OCS generated by CRA-2 is around half of the total number of common subexpressions.

Table 6.6 Statistic Percentages of VCS, OCS and HCS in CRA-2’s solutions

Filter Order	10	20	30	40	50	60	70	80
<i>HCS</i>	7.74	14.3	19.72	34.08	41.82	53.64	63.76	74.58
<i>ICS = VCS+OCS</i>	7.46	18.6	29.34	30.44	37.88	41.9	46.84	50.12
$\frac{ICS}{CS = HCS + ICS} \times 100\%$	49.1	56.5	59.8	47.2	47.5	43.86	42.35	40.16

Figure 6.14 presents the distribution of the heights of inter-coefficient common subexpressions (ICS) for coefficient sets of 80 taps and word length 11. The height and length of an inter-coefficient common subexpression have been defined earlier in Section 6.2 and illustrated in Figure 6.1. It is evident that most of the selected ICS have height

below 20. Although not plotted, it is also found that the distribution of ICS's lengths has a narrow spread centered at 0 and most of the ICSs have their lengths fall within the interval $[-2, 2]$. For coefficient sets of word length 11, the longest ICS is 8 bits. This information provides useful hints on how to lower the computation complexity of CRA. The frequency of sharing vertical and oblique common subexpressions reduces as the heights of these common subexpressions increase. Therefore, it makes good sense to limit the height of the inter-coefficient common subexpressions as the adder cost saving decreases rapidly with increasing height of these types of common subexpressions.

Figure 6.14 Distribution of the heights of ICS

The performance of CRAs is also valuated in terms of computation speed. The MATLAB programs of CRAH-2, CRAH, NRSCSE and Paško algorithms were run on a Pentium IV 1.9 GHz personal computer with 256 MBytes of system memory. Their average computation time in seconds for randomly generated filters with varying number of taps, and fixed word length ($= 11$) are compared in Figure 6.14. CRAH-2 and CRAH are chosen for this comparison because the two algorithms to which they are compared consider only horizontal common subexpressions. CRAH-2 uses the least CPU time in all the test cases. CRAH requires more computation time than Pasko algorithm on small

coefficient set. This is because for smaller coefficient sets, more time is spent on the contention path resolution than the search for the most common subexpressions at each iteration. As the coefficient set increases, the computation time of Paško exacerbates while that of the other algorithms remain relatively constant. Paško algorithm transforms the MCM block into a multiplier free linear transformation of dimension $m \times n$. More potential subexpressions are present with increasing values of m and n arising from the increase in filter taps, but the increase in the value of m also causes a drastic increase in computation effort. For CRAs, the number of iterations is dependent on the maximum order number of the subexpression list, which is limited by the word length. As the word length is fixed, the iteration time is relatively stable. That explains why the computational time increases only marginally as the coefficient set grows. CRAs generate the common subexpression list only once at the very beginning, which saves a lot of computational effort. This comparison shows that CRAs possess the attribute of good heuristic that achieves quality solutions for large problem size with well-bounded computational time.

Figure 6.15 Comparison on computation times.

6.6 Summary

In this chapter, the contention resolution algorithm of Chapter 5 is generalized to handle horizontal, vertical and oblique common subexpressions of any weight. Derivatives of contention algorithms (CRAs) are proposed to trade the mutually exclusive optimal design goals of logic depth and logic complexity versus the versatility of two-dimensional common subexpressions. Sensible logic depth optimization has been made possible by constraining the length of the precedence paths as the programs evolve. Among the variants of the proposed algorithm, CRAH-2 yields the shortest logic depth architecture with substantially reduced number of logic operators. Experimental results reveal that when the highest weight of the coefficient set is around three (a pragmatic criterion in filter coefficients synthesized for some audio applications), CRAH-2 gives the best results in both logic depth and logic complexity. The most versatile variant of CRA has no constraint on the weights of the common subexpressions and it searches for all horizontal, vertical and oblique subexpressions. It leads to a significant reduction in the number of arithmetic operations required to implement the MCM block. Evidence of the profitability of inter-coefficient common subexpression extraction has been demonstrated statistically and through testing on practical filters. CRA always achieves either the lowest or close to the lowest adder cost. When its generated solutions are compared with the solutions generated by other algorithms of comparable adder cost, the logic depth of the solutions produced by CRA is much lower. This logic depth driven synthesis option is important as lowering the logic depth improves the throughput rate and minimizes spurious switching activities through the critical path, which is important for some applications. Comparison with related work based on randomly generated filter coefficients for large MCM block shows that the logic depth driven algorithm produces low cost circuits with shorter critical path. Moreover, the proposed algorithms are also computationally efficient. The runtime of CRAH-2 is comparatively lower than NRSCSE and the computation efficiency of CRAH is significantly higher than the Paško algorithm.

as the problem dimension grows. The experimental results with benchmark filters and randomly generated coefficient sets are analyzed and compared with a number of well known common subexpression elimination methods to demonstrate the effectiveness and efficiency of the proposed approach.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Multiple Constants Multiplication (MCM) is a high-level computation model that is applicable to efficient hardware translation of ubiquitous digital signal processing functions like correlation, convolution, filtering and linear transformation in wireless and multi-media applications. Fully software driven DSP solutions to MCM problems are unlikely to process the challenging broadband signals with limited power budget. Thus, the fast evolving software defined radio and contemporary communication systems include customized application specific integrated circuit (ASIC) modules tailor specifically for wave shaping functions like correlators and high-speed filters [GUN99]. It has also been shown that filter banks in many multirate audio and image processing applications can be implemented as parallel MCM blocks using the Farrow structures [FAR88, GUN99]. These ASICs are designed for low die area, high throughput rate and low power. The throughput rate and power consumption requirements, in particularly exert a limit on the logic depth of the MCM block for a given supply voltage and cell library. Field Programmable Gate Array (FPGA) is yet another viable alternative to general-purpose processors for MCM solution with copious architectural mapping options. The work developed in this thesis is dedicated to FIR filter generation technology imperative to address the crucial design constraints of reuse and short turnaround time for cost effective ASIC and FPGA implementations. The contributions

can be capsuled in two major areas: the synthesis of low-complexity filter coefficients from the frequency response specification, and the optimization of logic depth and logic complexity of hardware implementations from a set of constant coefficients. To embark on these objectives, the CSD representation of the coefficients is explored.

The CSD representation plays a major role in the design of digital filters. A good understanding of the specific properties exhibited by these numbers and their appropriate exploitation helps to improve the quality and efficiency of existing algorithms, or even leads to new concepts and new algorithms. In this study, a regularly structured Hamming Weight Pyramid (HWP) have been proposed to assist in exploring the interesting properties of CSD numbers and the essential relationship between the natural numbers and their hamming weights in CSD and MSD representation. This formulation has provided motivation to develop a simple and efficient method of converting decimal number to its CSD form, and a more intuitive and succinct structure for representing all CSD numbers of a given word length. It is demonstrated that HWP can be employed to calculate the hamming weight in place of the complicated cost metric used by the RAG- n algorithm for implementing digital filters with minimal logic complexity. It is envisaged that other applications that employ CSD can be benefited from further exploitation of the derived properties of HWP, for example, ternary value finite automata [HAS96] and filter coefficient synthesis [CHE99] can be spawn from locating the constant coefficients and through direct manipulation on the proposed HWP.

For the work on filter coefficient synthesis, an efficient algorithm has been proposed for the maximal sharing of common subexpressions in the coefficients synthesized to meet the desired magnitude response specification. The proposed algorithm considers the reuse of the two most frequently used and most profitable subexpressions, 101 and $10\bar{1}$. The multiple-constraint problem of maximizing the sharing of the above two subexpressions

and minimizing the spare SPT terms is efficiently tackled in two heuristic search stages in an elegant and logical manner. With the help of a modified Hamming Weight Pyramid (CS-HWP), a minimal CSD coefficient set with the minimal NPRM in the continuum of the rounded CSD coefficient set is found in the first stage. The search has been highly efficient as it is confined to only worthy candidate coefficients with equal or smaller number of CSPT (see definition in Chapter 4) terms as the initial set. Depending on the wordlength of the coefficients, and the mismatch between the desired response and that obtained in the first stage, bivariate quantization steps in terms of discrete SPT terms are iteratively added in the second stage to compensate for the quantization errors until the desired normalized peak ripple magnitude (NPRM) has been met. It is shown, by means of benchmark design examples, that the proposed algorithm is indeed capable of synthesizing transposed direct form FIR filters with the least adder cost in both the CSA-tree and conventional adder based architecture among the existing filter synthesis algorithms. As it is a local search algorithm and the search range is well defined, the proposed algorithm finds a set of filter coefficients that meets the filter specification in a reasonable time.

For any given set of filter coefficients, its implementation can be effectively optimized by common subexpression elimination (CSE). In this area of research, several contention resolution algorithms have been proposed to circumvent the local minimum problem encountered by many heuristic CSE algorithms. A succinct admissibility graph representation of the coefficient set has been developed and employed as an efficient data structure for the proposed algorithms which is unique in its fundamental concept of operations. The benefit of the proposed contention resolution algorithms is derived from their ability to appraise and substitute the chosen subexpressions when better alternatives emerge. Although reduction in the number of logic operators is used directly as the criterion of the appraisal, logic depth optimization has been acquired as an important

advantage by constraining the weights of the selected common subexpressions as the algorithms evolve. With the core technique of contention resolution, derivatives have been devised to permit sharing of compositive types of common subexpression for implementing different filter structures. Among the proposed algorithms, CRAH-2 yields the shortest logic depth architecture with substantially reduced number of logic operators. Experimental results reveal that when the highest-weight of the coefficient set is around three (a pragmatic criterion in filter coefficients synthesized for some audio applications), CRAH-2 gives the best results in both logic depth and logic complexity. The most versatile variant of CRA has no constraint on the weights of the common subexpressions and it searches for all horizontal, vertical and oblique subexpressions. It leads to a significant reduction in the number of arithmetic operations required to implement the MCM solutions. Evidence of the profitability of inter-coefficient common subexpression extraction has been demonstrated statistically and through testing on practical filters. CRA always achieves either the lowest or close to the lowest adder cost. When its generated solutions are compared with the solutions generated by other algorithms of comparable adder cost, the logic depth of the solutions produced by CRA is much lower. This logic depth driven synthesis option is important as lowering the logic depth improves the throughput rate and minimizes spurious switching activities through the critical path. Comparison with related work based on randomly generated filter coefficients for large MCM block shows that these logic depth driven algorithm produces low cost circuits with shorter critical path. Moreover, the algorithms are also computationally efficient. The runtime of CRAH-2 is comparatively lower than the classic NRSCSE [PEI02] and the computation efficiency of CRAH is significantly higher than Paško's algorithm [PAS99] as the problem dimension grows.

In summary, the objectives set forth in this thesis on the design and optimization of digital filters have been met and the quality of the solutions and performances of the contributed algorithms are attested by thorough computer experiments.

7.2 Suggestions for Further Work

Based on the work presented in this thesis, several potential relevant topics for further work are suggested. Some of these areas are presently being investigated by other members of the same research group.

7.2.1 Applications of MSD and other SD representations in filter design

The study of synthesis and optimization of digital filter in this thesis is primarily based on CSD representation. As mentioned in Chapter 2, MSD representation provides the same minimum hamming weight as CSD representation but with heterogeneous formats for a given number. The polymorphisms of MSD are opportune for betterment of cost of subexpression sharings, but the problems of their generation and selection are also an impediment to the development of efficient algorithm. So far, elegant formulation of different MSD representations for a given number is lacking and the MSD representations are usually generated based on an exhaustive search [DEM04, GUS02, PAR02]. For the coefficient synthesis problem from amplitude response specification, to the best of the author's knowledge, there is no algorithm that truly exploit the polymorphism of MSD coefficients. This could be attributed to the fact that majority of the coefficient synthesis algorithms do not consider common subexpression sharing, and as a result, there is no apparent benefit to replace CSD by MSD since both of them have the same hamming weight. This is no true for the CSPT terms rather than SPT terms based optimization problem recast in Chapter 4. The extension of the proposed algorithm to MSD will certainly evoke savings of adders due to the potential of having more

common subexpression sharings. The difficulty to qualify a sensitivity measure of the fluctuations of amplitude response to each filter coefficient adds new challenges to the design of good heuristic synthesis methods as the demand of computational efficiency is in par with that for the conversion algorithm for MSD and hitherto undefined criterion for what constitutes the good MSD form. Some insight may be obtained from the pros and cons of MSD involvement in optimization from a handful of CSE algorithms that use MSD [PAR01, YAO04]. Involving MSD in filter design necessitates the study of MSD representation and its conversion method. Solutions obtained with MSD are expected to have better quality than those with CSD, as observed from the reports of existing MSD based CSE algorithms [PAR01, YAO04].

Besides, there are also proposals that postulate the relaxation of hamming weights to non-minimal representation can increase the chance of common subexpression sharings and therefore lead to higher reduction of adder cost [DEM05]. This thought has generated a surge of interest to explore other signed digit representations in filter design. The basic idea of optimization remains the same and the challenges are still very much falls into the conversion and selection criteria of the SD representations. The solutions to generation and application of MSD and other SD representations can be classified to two categories: (1) to exhaustively obtain all fulfilling MSD or SD in order to reach a decision according to object function, as in some present CSE and GD algorithms [PAR01, DEM05]. This method is obviously simple but will not be practical for big problem due to the computational complexity. (2) To keep the format of number representation fluid in the optimization process and only 'hard code' the number by converting it to a fixed representation at the critical point, as proposed in some algorithms [YAO04]. This method is not the most optimum but comparing with the former method, it is more efficient. It is envisage that optimality of present CSD-based algorithms have reached a

stagnant, MSD and SD-based methods will form a new trend of research in future and promises better solutions in practical large filter design.

7.2.2 Minimization on switching activity and adder complexity

As low power design becomes more desirable in integrated circuits, further optimizations can be made on switching activity and adder complexity. In digital CMOS circuits (for minimum feature size above 100nm), the total power consumption is dominated by the dynamic power, which can be approximated by [DEM04]

$$P_s = \frac{1}{2} V_{DD}^2 f_c C_L \alpha \quad (0.1)$$

where V_{DD} is the supply voltage, f_c is the clock frequency, C_L is the load capacitance and α is the switching activity, which is the probability of a 0 to 1 or 1 to 0 transition in a clock period. Reducing the logic depth of a circuit will reduce the length of paths via which glitches can propagate, thereby reducing α [DEM02, PAR01]. A method for computing the switching activity in bit-serial constant-coefficient multipliers has been proposed by Johansson, et. al. [JOH04]. They also mentioned that it is possible to reduce the switching activity by selecting the best structure of multiplier through the lookup tables generated for the switching activities of different structures. The use of these methods are still limited in filter design, but they show the direction of more dependable approaches to the estimation and hence reduction of power consumption in digital filter synthesis and optimization problem.

As in many other CSE algorithms, we assume all adders are of the same types and cost and shifters are virtually free irrespective of the amount of shifts. In actual implementation, the complexity of adders available in the ASIC library may have non negligible effect in terms of the overall power and critical path delay of the solution. Both our proposed CRA algorithms and many other CSE algorithms have mitigated the

discrepancy by biasing towards the shorter common subexpressions when two or more common subexpressions of same saving in adder cost are considered in order to reduce the adder length in the implementation. A fine-grain cost component may be optimized by considering the number of full adders used instead of the number of adders to account for the complexity of different types of adder used. In [VIN05], Vinod and Lai have proposed an efficient set partitioning method to reduce the adder length by extracting the common shift. They used the full adder cost as the logic complexity measure. In [CHE99, HAW96, YE04], optimization of FIR filter with carry-saved adders have been considered. We believe the adder length and different types of adder available in the cell library may be added as a new dimension in digital filter design. The optimization problem will then take dramatically different constraints and objective function but the performance will be better correlated with the resources available for the targeted implementation.

7.2.3 Development of EDA framework

To date, some of the algorithms developed in this thesis have been incorporated into a rudimentary automated and algorithm independent platform - 'sys-FIR' developed by other researchers in the same group [UDI05] to expedite the process of filter implementation. This tool was designed based on a unified framework of transposed direct form structure and was used to generate the RTL codes for several practical and application specific filters. They were then subjected through an ASIC flow using the same standard cell library. This evaluation and analysis of the VLSI area, delay and power provides a good insight into the solutions of existing algorithms including the proposed CRAs. The main objective behind the development of sys-FIR is to empower the designer with more control over the pre-synthesis hardware optimization of the filter. Apart from the coefficient file and the input signal length, the following parameters can be manipulated by the user to get hardware efficient RTLs before tool chain synthesis begins: symmetric or asymmetric filter, types of adder, different CLAs, and fan-out factor.

The available options have been rather restrictive at present, which can be enriched in future. Besides, a good graphical user interface to allow easy plug-and-play of new optimization methods will be developed so that methods for optimizing VLSI metrics by incorporating heterogeneous adder types in a filter can be readily studied.

The vision is to develop a unified EDA environment to automate silicon compiled solutions for generic linear-phase FIR filters based on primitive cell library of full and half adders, multiplexers and flip-flops. By accepting the desired filter response as design specifications, and throughput, silicon area and power dissipation as design constraints, the EDA tools provide an integrated environment for the developed design methodologies to yield several viable solutions. Based on the extracted parameters of the fully characterized primitive cells of the targeted fabrication technology, the output of the tool will be a series of files containing the synthesizable HDL codes describing the hardware structure of the designed filter.

Author's Publication

Journal Papers

- [1] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, "Modified reduced adder graph algorithm for multiplierless FIR filters," *Electronics Letters*, vol. 41, no. 6, pp. 302-303, Mar. 2005.
- [2] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, "Contention resolution algorithms for common subexpression elimination in digital filter design," *IEEE Transactions on Circuits and Systems II - Express Briefs*, vol. 52, no. 10, pp. 695-700, Oct. 2005.
- [3] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, "Hamming Weight Pyramid - A New Insight into Canonical Signed Digit Representation and Its Applications," accepted by *Journal of Computer Systems Science & Engineering*, Sep. 2006.
- [4] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, "Contention Resolution – A New Approach to Versatile Subexpressions Sharing in Multiple Constant Multiplications," submitted to *IEEE Transactions on Circuits and Systems I*, May, 2005. (revised and resubmitted Jan. 2007)
- [5] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, "Design of Low-Complexity FIR Filters Based on Signed-Powers-of-Two Coefficients with Reusable Common Subexpressions," accepted by *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Jan. 2007.

Conference Papers

- [1] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, "A new contention resolution algorithm for the design of minimal logic depth multiplierless filters," in *Proc. IEEE Int. Symp. on Circuits and Systems*, (ISCAS-2004), Vancouver, Canada, pp. 481-484, May 2004.
- [2] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, "HWP: a new insight into canonical signed digit," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS-2004)*, Vancouver, Canada, pp. 201-204, May 2004.
- [3] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, "Efficient algorithms for common subexpression elimination in digital filter design," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, Montreal, Canada, pp. 481-484, May 2004.
- [4] **Fei Xu**, Jiajia Chen, Chip-Hong Chang and Ching-Chuen Jong, "A modified reduced adder graph algorithm for multiplier block minimization in digital filters," in *Proc. IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS 2004)*, Tainan, Taiwan, pp. 705-708, Dec. 2004.
- [5] Sharma Udit, **Fei Xu**, Chip-Hong Chang, Ching-Chuen Jong and Kiat-Seng Yeo, "Sys-fir: a compiler for evaluating VLSI performance metrics of reduced adder cost FIR filters," in *Proc. IEEE Int. Symp. on Low-power and High-speed Chips (COOL Chips VIII)*, pp. 339-346, Yokohama, Japan, Apr. 2005.
- [6] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, " I^2 CRA: contention resolution algorithm for intra- and inter-coefficient common subexpression elimination," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS-2005)*, pp. 1823 – 1826, Kobe, Japan, May 2005.

- [7] **Fei Xu**, Chip-Hong Chang and Ching-Chuen Jong, "A new integrated approach to the design of low-complexity FIR filters," in *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS-2006)*, pp. 21-24, Kos Island, Greece, May 2006.

Bibliography

- [AKO97] D. A. Akopian, O. Vainio, S. S. Agaian, and J. T. Astola, "SBNR processor for stack filters," *IEEE Transactions on Circuits and Systems II*, vol. 44, pp. 197-208, 1997.
- [ARN93] S. Arno and F. S. Wheeler, "Signed digit representations of minimal Hamming weight," *IEEE Transactions on Computers*, vol. 42, pp. 1007-1010, 1993.
- [AIT93] D. Ait-Boudaoud and R. Cemes, "Modified sensitivity criterion for the design of powers-of-two FIR filters," *Electronics Letters*, vol. 29, pp. 1467-1469, Aug., 1993.
- [AVI61] A. Avizienis, "Signed-digit representation for fast parallel arithmetic," *IRE Transactions on Electronic Computer*, vol. 10, pp. 389-400, 1961.
- [BEN92] N. Benvenuto, M. Marchesi, and A. Uncini, "Applications of simulated annealing for the design of special digital filters," *IEEE Transactions on Signal Processing*, vol. 40, pp. 323-332, Feb., 1992.
- [BUL91] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proc. - G*, vol. 138, no. 3, pp. 401-412, Jun., 1991.
- [BUS01] F. Y. Busaba, C. A. Krygowski, W. H. Li, E. M. Schwarz, and S. R. Carlough, "The IBM z900 decimal arithmetic unit," in *Conf. Record of the Thirty-Fifth Asilomara on Signals, Systems and Computers*, vol. 2, pp. 1335-1339, 2001.
- [CEM93] R. Cemes and D. Ait-Boudaoud, "Genetic approach to design of multiplierless FIR filters," *Electronics Letters*, vol. 29, pp. 2090-2091, Nov., 1993.
- [CHE99] 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 Transactions on Circuits and Systems II*, vol. 46, no. 1, pp. 29-39, Jan., 1999.
- [CLA73] W. Clark and J. Liang, "On arithmetic weight for a general radix representation of integers (Corresp.)," *IEEE Transactions on Information Theory*, vol. 19, pp. 823-826, 1973.

-
- [DAS96] S. K. Das and M. C. Pinotti, "Fast VLSI circuits for CSD coding and GNAF coding," *Electronics Letters*, vol. 32, pp. 632-634, 1996.
 - [DAV02] T. N. Davidson, Z. Q. Luo, and J. F. Sturm, "Linear matrix inequality formulation of spectral mask constraints with applications to FIR filter design," *IEEE Transactions on Signal Processing*, vol. 50, iss. 11, pp. 2702-2715, Nov., 2002.
 - [DEM94] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proc. on Circuits, Devices and Systems*, vol. 141, pp. 407-413, 1994.
 - [DEM95] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Transactions on Circuits and Systems II*, vol. 42, no. 9, pp. 569-577, Sep., 1995.
 - [DEM00] A. G. Dempster and N. P. Murphy, "Efficient interpolators and filter banks using multiplier blocks," *IEEE Transactions on Signal Processing*, vol. 48, no. 1, pp. 257-261, Jan., 2000.
 - [DEM02] A. G. Dempster, S. S. Demirsoy, and I. Kale, "Designing multiplier blocks with low logic depth," in *Proc. IEEE Int. Symp. on Circuits and Systems*, Phoenix, Arizona, vol. 5, pp. 773-776, May, 2002.
 - [DEM04a] A. G. Dempster and M. D. Macleod, "Generation of signed-digit representations for integer multiplication," *IEEE Signal Processing Letters*, vol. 11, pp. 663-665, 2004.
 - [DEM04b] A. G. Dempster and M. D. Macleod, "Digital filter design using subexpression elimination and signed-digit representations," in *Proc. IEEE Int. Symp. on Circuits and Systems*, vol. 3, pp. 169-172, May, 2004.
 - [DEM04c] A. G. Dempster, M. D. Macleod, and O. Gustafsson, "Comparison of graphical and subexpression methods for design of efficient multipliers," in *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, vol. 3, pp. 169-172, May, 2004.
 - [DEM05] A. G. Dempster and M. D. Macleod, "Multiplication by two integers using the minimum number of adders," in *Proc. IEEE Int. Symp. on Circuits and Systems*, vol. 2, pp. 1814-1817, May, 2005.

-
- [FAR00] F. A. Farag, C. Galup-Montoro, and M. C. Schneider, "Digitally programmable switched-current FIR filter for low-voltage applications," *IEEE Journal of Solid-State Circuits*, vol. 35, iss. 4, pp. 637-641, Apr., 2000.
 - [FAR88] W. Farrow, "A continuously variable digital delay element," in *Proc. Int. Symp. on Circuits and Systems*, Espoo, Finland, vol. 3, pp. 2641-2645, 1988.
 - [GAR65] L. H. Garner, "Number Systems and Arithmetic," *Advances in Computers*, vol. 6, pp. 131-194, 1965.
 - [GUN99] J. E. Gunn, K. S. Baron and W. Ruczczyk, "A low-power DSP core-based software radio architecture," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 574-590, Apr. 1999.
 - [GUS01a] O. Gustafsson, H. Johansson, and L. Wanhammar, "An MILP approach for the design of linear-phase FIR filters with minimum number of signed-power-of-two terms," in *Proc. European Conf. Circuit Theory Design*, Espoo, Finland, Aug., 2001.
 - [GUS01b] O. Gustafsson, H. Ohlsson, and L. Wanhammar, "Minimum-adder integer multipliers using carry-save adders," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol.2, pp. 709-712, 2001.
 - [GUS02a] O. Gustafsson and L. Wanhammar, "ILP modeling of the common subexpression sharing problem," in *Proc. IEEE Int. Conf. on Electronics Circuits and Systems*, Dubrovnik, Croatia, vol. 3, pp. 1171-1174, Sep., 2002.
 - [GUS02b] O. Gustafsson, A. G. Dempster, and L. Wanhammar, "Extended results for minimum-adder constant integer multipliers," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol.1, pp. I-73-I-76, 2002.
 - [GUS02c] O. Gustafsson and L. Wanhammar, "Design of linear-phase FIR filters combining subexpression sharing with MILP", in *Proc. IEEE Midwest Symp. on Circuits and Systems*, Tulsa, OK, vol. 3, pp. 9-12, Aug., 2002.
 - [GUS03] O. Gustafsson, Contributions to Low-Complexity Digital Filters. *PhD Dissertation*, Linköping University, Linköping, Sweden, 2003.
 - [GUS04] O. Gustafsson, A. G. Dempster, and L. Wanhammar, "Multiplier blocks using carry-save adders," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol.2, pp. 473-476, 2002.

-
- [HAR96] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Transactions on Circuits and Syst.- II*, vol. 43, no. 10, pp. 677-688, Oct. 1996.
 - [HAS96] R. Hashemian, "A new method for conversion of a 2's complement to canonic signed digit number system and its representation," in *Conf. Record of Thirtieth Asilomar on Signals, Systems and Computers*, vol. 2, pp. 904-907, 1996.
 - [HAW96] R. A. Hawley, B. C. Wong, T. J. Lin, J. Laskowski and H. Samueli, "Design techniques for silicon compiler implementations of high-speed FIR digital filters," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 5, pp. 656-666, May 1996.
 - [HER01] J. P. Heron, R. Woods, S. Sezer and R. H. Turner, "Development of a run-time reconfiguration system with low reconfiguration overhead," *Journal of VLSI Signal Processing*, vol. 28, pp. 97-113, 2001.
 - [HER73] O. Herrmann, L. R. Rabiner, and D. S. K. Chan, "Practical Design Rules for Optimum Finite Impulse Response Lowpass Digital Filters," *Journal of Bell Systems Technology*, vol. 52, pp. 769-799, Jul.-Aug., 1973.
 - [HER95] A. Herrfeld and S. Hentschke, "Look-ahead circuit for CSD-code carry determination," *Electronics Letters*, vol. 31, pp. 434-435, 1995.
 - [HEW00] R. M. Hewlitt, E. S. Swartzlantler, "Canonical signed digit representation for FIR digital filters," in *Proc. IEEE Workshop on Signal Processing Systems*, pp. 416-426, 2000.
 - [HWA79] K. Hwang, *Computer Arithmetic: Principles, Architecture, and Design*. New York: John Wiley & Sons, 1979.
 - [ICH00] K. Ichige, M. Iwaki, and R. I shii, "Accurate estimation of minimum filter length for optimum FIR digital filters," *IEEE Transactions on Circuits Systems II*, vol. 47, no. 10, pp. 1008-1016, Oct. 2000.
 - [IEEE79] *Programs for Digital Signal Processing*. Algorithm 5.1, IEEE Press, New York, 1979.
 - [IEEE04] IEEE Standard Committee 2004: Some proposals for revising ANSI/IEEE Std 754-1985. <http://754r.ucbtest.org/>

-
- [IQB93] Z. Iqbal, M. Potkonjak, S. Dey, and A. Parker, "Critical path minimization using retiming and algebraic speed-up," in *ACM/IEEE Design Automation Conf.*, pp. 573-577, 1993.
 - [JAI91] R. Jain, P. T. Yang and T. Yoshino, "FIRGEN: A computer-aided design system for high performance FIR filter intergrated circuits," *IEEE Transactions On Signal Processing*, vol. 39, no. 7, pp. 1655-1668, Jul. 1991.
 - [JAN02] Y. H. Jang and S. J. Yang, "Low-power CSD linear phase FIR filter structure using vertical common sub-expression," *Electronics Letters*, vol. 38, no. 15, pp. 777-779, Jul. 2002.
 - [KAI66] J. F. Kaiser, *Digital filters: systems analysis by Digital computer*, John. Wiley & Sons, 1966.
 - [KAI68] J. Kaiser, "The digital filter and speech communication," *IEEE Transactions on Audio and Electroacoustics*, vol. 16, iss. 2, pp. 180-183. Jun., 1968.
 - [KAI74] J. F. Kaiser, "Nonrecursive digital filter design using 10-sinh window function," in *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 20-23, Apr. 1974.
 - [KAN01] H. J. Kang and I. C. Park, "FIR filter synthesis algorithms for minimizing the delay and the number of adders," *IEEE Transactions. on Circuits and Systems II*, vol. 48, no. 8, pp. 770-777, Aug. 2001.
 - [KHO96] K.-Y. Khoo, A. Kwentus, and A. N. Willson, Jr., "A programmable FIR digital filter using CSD coefficients," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 869-874, 1996.
 - [KOD81] D. Kodek and K. Steiglitz, "Comparison of optimal and local search methods for designing finite wordlength FIR digital filters," *IEEE Transactions on Circuits and Systems II*, vol. CAS-28, pp. 28-32, Jan. 1981.
 - [KOL00] B. Kolman, R. C. Busby and S. Ross, *Discrete Mathematical Structures*. New Jersey: Prentice Hall, 2000.
 - [KWE97] A. Y. Kwentus, Z. Jiang, and A. N. Willson, Jr., "Application of filter sharpening to cascaded integrator-comb decimation filters," *IEEE Transactions on Signal Processing*, vol. 45, pp. 457-467, 1997.

-
- [LEE93] J. H. Lee, C. K. Chen, and Y. C. Lim, "Design of discrete coefficient FIR digital filters with arbitrary amplitude and phase responses," *IEEE Transactions on Circuits and Systems II*, vol. 40, pp. 444-448, Jul., 1993.
 - [LEI90] S.-M. Lei and K. Yao, "A class of systolizable IIR digital filters and its design for proper scaling and minimum output roundoff noise," *IEEE Transactions on Circuits and Systems*, vol. 37, iss. 10, pp. 1217-1230, Oct., 1990.
 - [LI93] D. Li, J. Song, and Y. C. Lim, "A polynomial-time algorithm for designing digital filters with power-of-two coefficients," in *Proc. IEEE Int. Symp. on Circuits and Systems*, vol. 1, pp. 84-87, Chicago, May 1993.
 - [LI05] Xun Li, Xingzhong Chen, and M. Qasmi, "A broad-band digital filtering approach for time-domain Simulation of pulse propagation in optical fiber," *Journal of Lightwave Technology*, vol. 23, iss. 2, pp. 864-875, Feb., 2005.
 - [LI96] J. Li and P. Stoica, "An adaptive filtering approach to spectral estimation and SAR imaging," *IEEE Transactions on Signal Processing*, vol. 44, iss. 6, pp. 1469 – 1484, Jun., 1996.
 - [LIM83] Y. C. Lim and S. R. Parker, "Discrete coefficient FIR digital filter design based upon an LMS criteria," *IEEE Transactions on Circuits and Systems*, vol. CAS-30, pp. 723–739, Oct., 1983.
 - [LIM90] Y. C. Lim, "Design of discrete-coefficient-value linear phase FIR filters with optimum normalized peak ripple magnitude," *IEEE Transactions on Circuits and Systems*, vol. 37, no. 12, pp. 1480-1486, Dec., 1990.
 - [LIM91] Y. C. Lim, J. B. Evans, and B. Liu, "Decomposition of binary integers into signed power-of-two terms," *IEEE Transactions on Circuits and Systems*, vol. 38, pp. 667-672, 1991.
 - [LIU01] M. Liu, C. Chen, D. Shin, C. Lin and S. Jou, "Low-power multiplierless FIR filter synthesizer based on CSD code," in *Proc. IEEE Int. Symp. on Circuits and Systems*, vol. 4, pp. 666 –669, May 2001.
 - [MAC04] M. D. Macleod and A. G. Dempster, "Common subexpression elimination algorithm for low-cost multiplierless implementation of matrix multipliers," *Electronics Letters*, vol. 40, iss. 11, pp. 651-652, May 2004.

-
- [MAR02] M. Martinez-Peiró, E. I. Boemo, and L. Wanhammar, "Design of high-Speed multiplierless filters using a nonrecursive signed common subexpression algorithm," *IEEE Transactions on Circuit and Systems*, vol. 49, no. 3, pp. 196-203, Mar. 2002.
 - [OPP89] A. P. Oppenheim and R. W. Schaffer, *Discrete-Time Signal Processing*, Prentice-Hall, 1989.
 - [PAR00] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, 2000.
 - [PAR01] I. C. Park and H. J. Kang, "Digital filter synthesis based on minimal signed digit representation," in *Proc. IEEE Design Automation Conf.*, Las Vegas, Nevada, pp. 468-473, Jun. 2001.
 - [PAR02] I. C. Park and H. J. Kang, "Digital filter synthesis based on an algorithm to generate all minimal signed digit representations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, pp. 1525-1529, 2002.
 - [PAS99] R. Paško, P. Schaumont, V. Derudder, S. Vernalde and D. Ďuračková, "A new algorithm for elimination of common subexpressions," *IEEE Transactions on Computer-Aided Design*, vol. 18, no. 1, pp. 58-68, Jan. 1999.
 - [PEL04] C. Pelard, E. Gebara, A. J. Kim, M. G. Vrazel, F. Bien, Y. Hur, M. Moonkyun, S. Chandramouli, C. Chun, S. Bajekal, S. E. Ralph, B. Schmukler, V. M. Hietala, and J. Laskar, "Realization of multigigabit channel equalization and crosstalk cancellation integrated circuits," *IEEE Journal of Solid-State Circuits*, vol. 39, iss. 10, pp. 1659 - 1670 Oct., 2004.
 - [PER05] P. Persson, S. Nordebo, and I. Claesson, "Design of discrete coefficient FIR filters by a fast entropy-directed deterministic annealing algorithm," *IEEE Transactions on Signal Processing*, vol. 53, pp. 1006-1014, Mar., 2005.
 - [POT96] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Transactions on Computer-Aided Design*, vol. 15, no. 2, pp. 151-165, Feb. 1996.

-
- [PRO96] J. G. Proakis and D. G. Manolakis, *Digital signal processing: principles, algorithms, and applications*, Prentice-Hall, 1996.
 - [REI60] R. W. Reitwiesner, "Binary arithmetic," in *Advances in Computers*, New York: Academic Press, vol. 1, pp. 231-308, 1960.
 - [REU89] P. Reutz, "The architectures and design of a 20-MHz real-time DSP chip set," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 338-348, Apr., 1989.
 - [ROS04] V. S. Rosa, E. Costa, J. C. Monteiro, and S. Bampi, "An improved synthesis method for low power hardwired FIR filters," in *17th Symp. Integrated Circuits and Systems Design*, pp. 237-241, 2004.
 - [SAM89] H. Samuelli, "An improved search algorithm for the design of multiplier-less FIR filters with powers-of-two coefficients," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 7, pp. 1044-1047, Jul. 1989.
 - [SAM90] H. Samuelli, "The design of multiplierless digital data transmission filters with powers-of-two coefficients," in *Proc. IEEE Int. Telecommunications Symp.*, Rio de Janeiro, Brazil, pp. 425-429, Sep. 1990.
 - [SCH93] J. D. Schaffer and L. J. Eshelman, "Designing multiplierless digital filters using genetic algorithms," in *Proc. 5th Int. Conf. Genetic Algorithms*, Urbana-Champaign, IL, pp. 439-444, Jul. 1993.
 - [SCH02] E. M. Schwarz, M. A. Check, C. L. K. Shum, T. Koehler, S. B. Swaney, J. D. MacDougall, and C. A. Krygowski, "The microarchitecture of the IBM eServer z900 processor," *IBM Journal of Research and Development*, vol. 46, pp. 381-394, 2002.
 - [SHA91] H. Shaffeu, M. M. Jones, H. D. Griffiths, and J. T. Taylor, "Improved design procedure for multiplierless FIR digital filters," *Electronics Letters*, vol. 27, pp. 1142-1144, Jun. 20, 1991.
 - [SUZ85] M. Suzuki, N. Miki, and N. Nagai, "New wave digital filters for basic reactance sections," *IEEE Transactions on Circuits and Systems*, vol. 32, iss. 4, pp. 337 – 348, Apr., 1985.
 - [UDI05] S. Udit, F. Xu, C. H. Chang, C. C. Jong, and K. S. Yeo, "Sys-fir: A compiler for evaluating VLSI performance metrics of reduced adder cost FIR filters,"

- in *Proc. IEEE Int. Symp. on low-power and high-speed Chips*, pp. 339-346, Yokohama, Japan, Apr., 2005.
- [VIN03] A. P. Vinod, E. M. Lai, A. B. Premkuntar, and C. T. Lau, "FIR filter implementation by efficient sharing of horizontal and vertical common subexpressions," *Electronics Letters*, vol. 39, no. 2, pp. 251-253, Jan., 2003.
- [VIN05a] A. P. Vinod, and E. M. Lai, "Low power and high-speed implementation of FIR filters for software defined radio receivers," *IEEE Transactions on Wireless Communications*, vol. 5, iss. 7, pp. 1669-1675, Jul., 2006.
- [VIN05b] A. P. Vinod, and E. M. Lai, "On the implementation of efficient channel filters for wideband receivers by optimizing common subexpression elimination method," *IEEE Transactions on Computer-Aided Design*, vol. 24, no. 2, pp. 295-304, Feb., 2005.
- [XU04a] F. Xu, C. H. Chang and C. C. Jong, "A modified reduced adder graph algorithm for multiplier block minimization in digital filters," in *Proc. IEEE Asia-Pacific Conference on Circuits and Systems*, pp. 33-36, Dec., 2004.
- [XU04b] F. Xu, C. H. Chang and C. C. Jong, "A new contention resolution algorithm for the design of minimal logic depth multiplierless filters," in *Proc. IEEE Int. Symp. Circuits and Systems*, Vancouver, Canada, Vol III, pp. 481-484, May, 2004.
- [XU04c] F. Xu, C. H. Chang and C. C. Jong, "Efficient Algorithms for Common Subexpression elimination in Digital Filter Design," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Montreal, Canada, pp. 481-484, May, 2004.
- [XU04d] F. Xu, C. H. Chang and C. C. Jong, "HWP: a new insight into canonical signed digit," in *Proc. IEEE Int. Symp. on Circuits and Systems*, Vancouver, Canada, pp. 201-204, May, 2004.
- [XU05a] F. Xu, C. H. Chang and C. C. Jong "I²CRA: Contention Resolution Algorithm for Intra- and Inter-Coefficient Common Subexpression Elimination," in *Proc. IEEE Int. Symp. on Circuits and Systems*, pp. 1823 – 1826, Kobe, Japan, May 2005.

-
- [XU05b] F. Xu, C. H. Chang and C. C. Jong, "Contention resolution algorithms for common subexpression elimination in digital filter design," *IEEE Transactions on Circuits and Systems-II Express Brief*, vol. 52, no. 10, pp. 695-700, Oct., 2005.
 - [XU05c] F. Xu, C. H. Chang and C. C. Jong, "Modified reduced adder graph algorithm for multiplierless FIR filters," *Electronics Letters*, vol. 41, iss. 6, pp. 302 – 303, 2005.
 - [YAO01] C. Y. Yao, "A Study of SPT-term distribution of CSD numbers and its application for designing fixed-point linear phase FIR," in *Proc. IEEE Int. Symp. on Circuits and Systems*, vol.2, pp. 301 -304, 2001.
 - [YAO04] C. Y. Yao, H. H. Chen, T. F. Lin, C. J. Chien and C. T. Hsu, " A novel common-subexpression-elimination method for synthesizing fixed-point FIR filters," *IEEE Trans on Circuits and Systems I*, vol. 51, no. 11, pp. 2215-2221, Nov., 2004.
 - [YE03] Z. Ye and C. H. Chang, "A hybrid CSA tree for merged arithmetic architecture of FIR filter," in *Proc. 3rd IEEE/EURASIP Int. Symp. on Image and Signal Processing and Analysis*, Rome, Italy, September 2003, pp. 449-453.
 - [YE04] Z. Ye, R. K. Satzoda, U. Sharma, N. Nazimudeen and C. H. Chang, "Performance evaluation of direct form FIR filter with merged arithmetic architecture," in *Proc. 2nd IEEE Int. Workshop on Electronic Design, Test and Applications*, Perth, Australia, January 2004, pp. 407-409.
 - [YLI01] J. Yli-Kaakinen and T. Saramaki, "A systematic algorithm for the design of multiplierless FIR filters," in *Proc. IEEE Int. Symp. on Circuits and Systems*, vol. 2, pp.185-188, Geneva, Switzerland, 2001.
 - [YOS90] T. Yoshiono, et al. "A 100-MHz 64-tap FIR digital filter in 0.8- μ m BiCMOS gate array," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 6, pp. 1494-1501, Dec. 1990.
 - [YUR99] A. Yurdakul and G. Dundar, "Multiplierless realization of linear DSP transforms by using common two-term expressions," *Journal of VLSI Signal Processing*, vol. 22, pp. 163-172, Sep. 1999.