Design and analysis of redundant binary booth multipliers

He, Ya Juan

2008

He, Y. J. (2008). Design and analysis of redundant binary booth multipliers. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/41412>

<https://doi.org/10.32657/10356/41412>

Design and Analysis of Redundant Binary Booth Multipliers

HE YAJUAN

School of Electrical & Electronic Engineering

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

2008

Acknowledgements

First and foremost, I am deeply indebted to my supervisor, Associate Professor Chang Chip Hong, for all of his invaluable guidance, continuous technical and personal support during my PhD candidature at Nanyang Technological University. His many years of research experience and unique sense in engineering research have led to a very effective and insightful guidance to my research work. In particular, I wish to thank him for teaching me the philosophies of research and intangible skills, which are the most important knowledge I have acquired in this research program. What I have learned from him will benefit me well beyond my graduation in my future career and personal life. Special thanks to Mrs. Chang for her continuous concern and kindness, from which I gained lots of wonderful memories. I really treasure the time spent with them.

I would like to thank Dr. Gu Jiangmin and Dr. Hossam A. H. Fahmy for their valuable discussions, suggestions and support in the research work. I would also like to thank the fellow members from the Chip's Family for the long run friendship, and invaluable help pertaining to my research. To the staffs and other students in the Center for Integrated Circuits and Systems (CICS) and the Center for High Performance Embedded Systems (CHiPES), I wish to convey my appreciation to all of them for their kind and friendly

assistance.

I would also like to express my gratitude to my previous supervisors and colleagues in the Institute of Microelectronics (IME). I would not have been here in Singapore if Dr. Xue Ping had not recruited me as a digital IC designer in 2001. I would like to thank Ms. Doreen Yeo Lee Guek and Mr. Wang Zhongjun particularly for their guidance in the early stage of my career.

I am grateful to my fiancé Li Qiang for his devotion, understanding, and patience. I would never know whether I am able to go through the tough time without his support. He is the most important source of inspiration, encouragement, and happiness.

My parents are always there for me and always have faith in me. It is never enough to say thanks to them. I dedicate this thesis to my parents with all my love.

Summary

Multiplication is a fundamental operation in most arithmetic computing systems. Over the last few decades, Redundant Binary (RB) number has emerged as a key internal format to speed up the partial product accumulation of tree-structured parallel multipliers due to its carry-free property and regularity in Very Large Scale Integrated (VLSI) implementation. In this thesis, the high performance energy-efficient multiplication operation has been investigated based on three key constituent components of the RB Booth multiplier architecture.

A new Redundant Binary to Normal Binary (RB-to-NB) conversion algorithm based on hybrid Carry-Lookahead/Carry-Select method has been proposed. The optimally designed carry-select adder sections are interleaved evenly in the mixed-radix carry-lookahead adder network to boost the performance of the reverse converter well above those designed based on a homogeneous type of carry-propagation adder. Towards this end, a 64-bit reverse converter circuit has been implemented in transistor level. The post-layout simulation results have indicated that the proposed converter circuit is capable of completing a 64-bit conversion in 829 ps and dissipating merely 5.84 mW from 1.8V at a data rate of 1GHz in a 0.18- μ m CMOS technology.

By fully exploiting the characteristics of the Booth encoded numbers, a high-speed energy-efficient RB multiplier architecture has been proposed based on the covalent redundant binary Booth encoding algorithm. The idea is to polarize the two adjacent Booth encoded digits to directly form an RB partial product for the ease of hard multiple generation and avoidance of correction vector. The synthesis results have shown that the RB multiplier based on the proposed algorithm outperforms its rivals in terms of speed and energy efficiency for the natural word lengths of computing from 8 bits to 64 bits.

With the study and evaluation on both existing and proposed new amalgamable modules for a number of RB multipliers, a structural and systematic approach has been proposed to design and analyze the RB high-performance multipliers. Twenty-one different $N \times N$ -bit RB multiplier architectures have been constructed with varying configurations of RB partial product generation, encoding, reduction and conversion methods. These multipliers have been implemented in gate-level VHDL with the same standard cell library and compared for various VLSI metrics such as area, delay, energy and energy efficiency. Based on the synthesis results for commonly used operand lengths, a large design space has been formulated from sensible topological combinations of different constituent modules of RB multiplier architecture for the exploration of the desirable performance characteristics.

Table of Contents

Acknowledgements	i
Summary	iii
Table of Contents	v
List of Figures	vii
List of Tables	ix
List of Acronyms and Abbreviations	xi
1 Introduction	1
1.1 Motivation	1
1.2 Research Objectives	6
1.3 Major Contributions	7
1.4 Organization of the Thesis	10
2 Digital Multiplier Architectures for Redundant Binary Arithmetic	12
2.1 Overview of Digital Multipliers	12
2.2 Redundant Binary Multiplier Architecture	17
2.2.1 Existing Booth Encoding Algorithms	18
2.2.2 Redundant Binary Adder	25
2.2.2.1 RBA with Sign-Magnitude Coding	28
2.2.2.2 RBA with Positive-Negative Coding	29

2.2.2.3	RBA with Positive-Negative-Complement Coding	31
2.2.3	Conversion Between RB and NB Numbers	32
2.3	Review of Existing RB Multipliers – Challenges and Opportunities	34
2.4	Experimental Methodology	39
2.4.1	Logical Effort	40
2.4.2	Transistor-Level Circuit Optimization and Simulation . .	45
2.4.3	Gate-Level Synthesis and Power Simulation	48
3	Hybrid Carry-Lookahead/Carry-Select Based RB-to-NB Converter	50
3.1	Introduction	50
3.2	Reconciliation of Coding Formats for Unanimity of RB-to-NB Conversion	53
3.3	Exploration on Hybrid CLA/CSL Architecture for RB-to-NB Conversion	57
3.3.1	Hybrid CLA/CSL Based Reverse Conversion Algorithm	57
3.3.2	Parallel-Prefix Carry-Lookahead with Uniform and non-Uniform Block Factors	60
3.4	Implementation of A 64-bit Reverse Converter	66
3.4.1	The Architecture of 64-bit Reverse Converter	66
3.4.2	Design Considerations: Modified Add-One CSL Scheme	70
3.5	Performance Evaluation	75
3.6	Summary	82
4	RB Multiplier with New Covalent Redundant Binary Booth Encoding	84
4.1	Introduction	84
4.2	Issues of Booth Encoding Algorithms for Redundant Binary Multiplication	86
4.2.1	Hard Multiple Problems Revisit	87
4.2.2	Negative Multiples and NB-to-RB Partial Products Conversion	87

TABLE OF CONTENTS

vii

4.2.3	Redundant Binary Booth Encoding (RBBE)	91
4.3	Covalent Redundant Binary Booth Encoding Algorithm	91
4.3.1	Radix-4 Covalent Redundant Binary Booth Encoding (CRBBE-2)	92
4.3.2	Radix-16 Covalent Redundant Binary Booth Encoding (CRBBE-4)	96
4.4	Circuit Design of Redundant Binary Multiplier	98
4.4.1	Circuit Design of CRBBE-4	99
4.4.2	CRBBE-4 Based RB Multiplier Architecture	101
4.5	Simulation Results	104
4.6	Summary	108
5	Energy Efficiency Evaluation of Redundant Binary Booth Multipliers	110
5.1	Introduction	110
5.2	Architectural Exploration on RB Multipliers	113
5.2.1	Taxonomy of Booth Encoders and Partial Product Generators (BEPPGs)	113
5.2.1.1	Normal Binary Booth- k Encoding (NBBE- k) . .	114
5.2.1.2	Redundant Binary Booth- k Encoding (RBBE- k)	115
5.2.2	One-Digit BEPPG Module	115
5.2.3	Qualitative Analysis of BEPPG on $N \times N$ -bit RB Multipliers	120
5.3	Coherent RB Coding Interface Components	122
5.3.1	One-Digit RB Adder Cells	122
5.3.2	Converters for Coherent RBA Interface	124
5.4	Performance Evaluation and Discussions	126
5.4.1	Configurations of RB Booth Multipliers	126
5.4.2	Numerical Simulation Results	129
5.4.3	Analyses and Discussions	130
5.4.3.1	Normal Binary Booth Encoding vs. Redundant Binary Booth Encoding	134
5.4.3.2	High-Radix Booth Encoding vs. Simple Booth Encoding	136

5.4.3.3	RB Coding Efficiency	139
5.5	Summary	141
6	Conclusions and Recommendations	143
6.1	Conclusions	143
6.2	Recommendations for Future Research	146
	Author's Publications	151
	Bibliography	153

List of Figures

2.1	Classification of digital multipliers.	13
2.2	Trichotomy of RB Booth multiplier architecture.	17
2.3	3M hard multiple generation and negation in partially redundant form [57].	23
2.4	RB adder for 5M hard multiple generation.	25
2.5	Circuit implementation of an RB full adder with sign-magnitude coding.	29
2.6	Circuit implementation of an RB full adder with positive-negative coding.	30
2.7	Circuit implementation of an RB full adder with positive-negative-complement coding.	31
2.8	An example of RB-to-NB conversion process.	33
2.9	Block diagram of carry generation in RB-to-NB converter [59]. .	34
2.10	FO4 delay illustration.	43
2.11	Transistor sizing optimization flowchart.	47
2.12	Transistor-level circuit simulation environment.	48
3.1	18-bit parallel-prefix carry generation with various block factors and block lengths of CSL.	64
3.2	Block diagram of the 64-bit reverse converter.	67
3.3	Block diagram of the modified 64-bit reverse converter.	68

3.4	Circuit implementation of G and P cells in the 5-stage CLA network.	69
3.5	CSL with a single RCA and an add-one circuit [101].	71
3.6	Modified add-one scheme.	72
3.7	6-bit CSL section with modified add-one scheme.	74
3.8	Full-custom layout of proposed 64-bit reverse converter.	81
4.1	Illustration of the correction vector generation on an 8×8 -bit multiplication with NBBE-2.	90
4.2	Radix-16 RBBE encoder and the partial product generator.	92
4.3	Radix-2 Booth encoded multiplier.	93
4.4	16×16 -bit RB multiplication with CRBBE-4.	98
4.5	Circuit implementation of CRBBE-4 encoder.	100
4.6	RB partial product generator of CRBBE-4.	101
4.7	Block diagram of 64×64 -bit RB multiplier architecture.	102
4.8	Schematic of RB full and half adders.	103
4.9	Comparison of normalized EDP of different Booth encoded RB multipliers.	108
5.1	Circuit implementations of BEPPG modules in NBBE.	117
5.2	Circuit implementations of BEPPG modules in RBBE.	118
5.3	Circuit implementation of RBA cells.	123
5.4	Three anterior converters used in RB multiplier design.	125
5.5	Posterior converter used in RB-to-NB conversion for PNC coding.	126
5.6	Scatter plot of area <i>vs.</i> worst-case delay and energy dissipation in natural logarithmic scale.	130
5.7	Normalized EDP of NBBE and RBBE multipliers.	136
5.8	Normalized EDP of high-radix and simple Booth multipliers.	138
	(a) PN coding	138
	(b) SM coding	138

LIST OF FIGURES

xi

5.9	Normalized EDP of all RB multipliers. The sizes of the multipliers from top left to bottom right are 8-bit, 16-bit, 24-bit, 32-bit, 48-bit and 64-bit.	140
-----	--	-----

List of Tables

2.1	Booth-1 Encoding	19
2.2	Booth-2 Encoding	20
2.3	Booth-3 Encoding	21
2.4	Booth-4 Encoding	22
2.5	RB Booth-3 Encoding	25
2.6	RB Booth-4 Encoding	26
2.7	Carry-Free Addition Rules for RBA	28
2.8	Sign-Magnitude Coding [58]	29
2.9	Positive-Negative Coding [59]	30
2.10	Positive-Negative-Complement Coding [13]	31
2.11	The Logical Effort and Parasitic Delay of Common Logic Gates	43
2.12	Key Definitions of Logical Effort	44
3.1	Comparison of Delay for Different Combinations of Block Factors of CLA and Block Lengths of CSL	77
3.2	Comparison of Transistor Count for Different Combinations of Block Factors of CLA and Block Lengths of CSL	77
3.3	Comparison of Area-Delay Product for Different Combinations of Block Factors of CLA and Block Lengths of CSL	78
3.4	Comparison of Delay for Different Converters	78
3.5	Comparison of Transistor Count for Different Converters	79
3.6	Comparison of Area-Delay Product for Different Converters	79

3.7	Comparisons of 64-bit Reverse Converters	80
3.8	Post-Layout Figure-of-Merit of Proposed 64-bit Reverse Converter	82
4.1	Permissible Duplet (d_{i+1}, d_i) in Radix-2 Booth Encoded Number	93
4.2	Polarization of (d_{i+1}, d_i) for Radix-4 CRBBE	94
4.3	Polarization of (d_{i+1}, d_i) for radix-16 CRBBE	97
4.4	Synthesis Results of Different Booth Encoded RB Multipliers . .	105
4.5	Energy-Delay Product of RB Multipliers	107
5.1	Delay and Unit Gate Number of One-Digit BEPPG Modules . .	119
5.2	Characteristics of $N \times N$ -bit RB Multiplier Architectures with Different BEPPGs	121
5.3	FO4 Delay and Complexity of RB Full and Half Adders	124
5.4	Configurations of RB Multipliers with Different Code Converters	127
5.5	Comparisons on Area of RB Multipliers	131
5.6	Comparisons on Worst-Case Delay of RB Multipliers	132
5.7	Comparisons on Energy Dissipation of RB Multipliers	133

List of Acronyms and Abbreviations

ADP	Area-Delay Product
BEPPG	Booth Encoder and Partial Product Generator
CLA	Carry-Lookahead Adder
CPA	Carry-Propagation Adder
CRBBE	Covalent Redundant Binary Booth Encoding
CSA	Carry-Save Adder
CSL	Carry-Select Adder
DC	Design Compiler
EDP	Energy-Delay Product
FO4	Fanout-of-4
LE	Logical Effort
LSB	Least Significant Bit
LSD	Least Significant Digit
MSB	Most Significant Bit
MSD	Most Significant Digit
MNBBE	Modified Normal Binary Booth Encoding
NB	Normal Binary
NBBE	Normal Binary Booth Encoding
PDP	Power-Delay Product
PN	Positive-Negative
PNC	Positive-Negative-Complement
PPG	Partial Product Generator
PRBBE	Partially Redundant Biased Booth Encoding
RB	Redundant Binary
RB-to-NB	Redundant Binary to Normal Binary
RBA	Redundant Binary Adder

RBBE	Redundant Binary Booth Encoding
RBFA	Redundant Binary Full Adder
RBHA	Redundant Binary Half Adder
RBPP	Redundant Binary Partial Product
RCA	Ripple-Carry Adder
SAIF	Switching Activity Interchange Format
SDR	Software Defined Radio
SM	Sign-Magnitude
TG	Transmission Gate
VLSI	Very Large Scale Integrated

Chapter 1

Introduction

1.1 Motivation

The digital age has fueled the enormous growth of the electronics industry with an interminable flow of cheaper, faster and more palatable digital signal processors. The ever more sophisticated VLSI circuits, in turn, have been stoked by the ubiquitous demand of various forms of signal processing requirements in a wide range of applications. Recently, the wireless technology has undergone an extensive research and development evolution, such as the emerging Software Defined Radio (SDR) [1, 2] and cognitive radio [3] that ensue the successful digital integrated circuit implementations of RF and analog front-end. Many high frequency analog functions have now been displaced by the high-performance arithmetic logic units. Supporting this progress is the infrastructure provided by the design tools, enhanced digital cell library and advanced semiconductor process technology. Comparing with analog integrated circuits, application specific digital signal processors are more robust to

noise and interference, which make them immune to the process and temperature variations. Meanwhile, digital designs benefit more from the shrinking process geometries and multiple metal layers for performance enhancements. They can also better leverage on the intellectual property reuse for faster time-to-market and cost reduction for large volume manufacturing.

The state-of-the-art digital signal processing applications play an important role in making the complex real-time algorithms for speech, audio, image processing, video, control and communication systems economically feasible [4–10]. Multiplication is one of the most commonly used arithmetic operators in these application specific data paths. Compared to many other arithmetic operations, multiplication is time consuming and power hungry. The critical paths dominated by digital multipliers often impose speed limits on the entire design. Therefore, there have been unending research interests and numerous publications on the design of high performance digital multipliers at different design abstraction levels [11–16].

A traditional important attribute of digital multipliers for most applications is the maximum operable speed. Today, design of high-speed multipliers remains a popular pursuit. However, as power consumption has become an increasingly important performance criterion in the design of digital systems, “using a design that is fast enough and consumes the least power” has emerged as a valued proposition compared to “using the fastest design” [17]. This design philosophy has resulted in a paradigm shift in the emphasis of arithmetic circuit performance from solely the fastest speed or the least power dissipation to the best Power-Delay Product (PDP). The PDP has the same meaning as the energy per operation, but the energy per operation is a preferred term when the only constraint for a design is the battery life. This

is because the energy per operation is a monotonic function of supply voltage and can be minimized as much as required simply by reducing the supply voltage. Very often, the supply voltage is fixed for the suggested nominal operation and each design has its own speed requirements for the slowest path, which is determined by the system's specification. Therefore when both constraints of speed and battery life are to be satisfied, the energy efficiency of the multiplier is improved by minimizing the product of energy and delay [18, 19]. As a result, Energy-Delay Product (EDP), rather than PDP serves as a new battleground for the contest of a variety of multiplier designs.

Most digital multiplier designs are based on the Normal Binary (NB) number representation [7, 20–28]. For signed number, a widely accepted interpretation of this term is the two's complement representation of the number. The current predominant multiplier architecture uses 3-to-2 counters and 4-to-2 compressors in a binary tree for parallel computation [6, 21–23, 28–31]. In the last two decades, most speed improvements in this architecture have been achieved via extreme circuit optimization and the use of advanced fabrication technology. The gain resulting from architectural innovation of NB multipliers is almost stagnant. It is conjectured that new insight of energy efficiency is unlikely to be derived from a matured architecture with an area-delay optimization outlook.

In view of this, alternative Redundant Binary (RB) number based architectures with the merit of carry propagation free accumulation are sought to speed up digital multiplication [11, 32–36]. The idea is to apply a simple signed digit representation as an internal format for the addition of multiple operands. The redundancy is exploited to speed up the addition of partial products, which is a crucial stage of the digital multiplier architecture. Fur-

thermore, the use of Redundant Binary Adders (RBAs) makes a more regular interconnection network and modular partial product summing tree structure. Advocators are optimistic that the nature of carry-free addition and structural regularity of RB multiplier architecture offers significant room for both power and latency reduction, although no rigorous experiment has been performed to prove the hypothesis.

There is no free lunch, though. Some overheads of existing RB multiplier architectures have seemed to play down their performance. One such factor involves the external communication overheads. Since the peripheral interfaces of most digital systems are still based on the NB number system, additional circuits are required to convert the NB partial products to RB numbers and vice versa. Although the forward conversion from the NB number to the RB number is a direct process and can be straightforwardly performed in constant time, additional compensation vector due to two's complement arithmetic and RB coding will be incurred in the formation of the RB Partial Products (RBPPs). If the accumulation network is of binary tree structure, it always favors the addition of a power-of-two number of partial products, which is very well suited to the natural word length of computing. The inclusion of extra vector offsets this optimality and increases the number of stages necessitated by the summing tree. This can actually hamper the performance and power consumption of RB multiplier in the application-specific data paths [37] and general purpose programs running on various computer architectures [38] that are operating with the power-of-two operand lengths. On the other hand, the conversion of the final RB sum to NB form is a more severe overhead, which has area-time complexity comparable to that of a Carry-Propagation Adder (CPA) of the same word length. The conversion needs to be performed at least once in

a multiplication using the RB number system, although this overhead can be reduced to be a very small fraction of the overall computation load in some signal processing applications, like digital filtering and correlation, where several multiplications and accumulations are performed before the final results are communicated through the peripheral.

To the best of our knowledge, RB multipliers enhanced for some specific applications have not been sufficiently explored by researchers in this community. Besides, the redundancy of the RB arithmetic can also be exploited to avoid the generation of correction vectors at the expense of increased complexity. This can be used advantageously to implement RB multipliers of scalable precisions. We believe that the potential and properties of RB arithmetic have yet to be fully evaluated and exploited in VLSI design. Given that redesigning a complex arithmetic operator to meet a given data path timing or a system level specification being a common sight of VLSI design, it is imperative to delve more into the figures of merit of RB multiplier architectures to aid the designer to make very lean performance tradeoffs. We quote Mead and Conway: “Perhaps the greatest challenge that VLSI presents to computer science is that of developing a theory of computation that accommodates a more general model of costs involved in computing” [39] as a general preface to this thesis. By circumventing some deficiencies identified in RB multiplier design, we hope that the research contributes a humble step to one of the long lasting subjects in computer arithmetic.

1.2 Research Objectives

The preliminary goal of this research is to explore the unaccustomed RB number for digital multiplier design. The performance stumbling blocks of the RB multiplier will be identified and investigated. New designs and ingenious architectures of RB multiplier and its constituent components will be proposed and characterized for critical VLSI metrics such as speed, energy per operation, and energy-delay products. The research will be approached with no prior assumption on the design abstraction level. This is to enable the most palatable design style at transistor level or modularity at gate level to be exploited to optimize newly developed submodules or configurations. This research project also aims at providing a new insight into the design tradeoff between speed and energy consumption among different families of Booth encoders and RB multipliers. An extensive analysis of different RB multiplier topologies for varying operand lengths is envisaged to provide a useful decision model in the early design phases of application-specific data paths, as their architectural optimizations are based on the knowledge of the arithmetic operators used.

The following specific problems have been targeted towards fulfilling the theme of this research.

1. To formulate the reverse conversion problem by investigating the properties of RB representation and its binary encoding schemes in order to devise new efficient architecture for RB to NB number conversion.
2. To overcome the hard multiple and negative multiple negation problems of high radix Booth encoding algorithms in order to design a high-speed and energy-efficient forward digit set converter for RB multiplier without

annihilating the structural regularity and modularity of its RB summing tree, especially for the power-of-two data word lengths in multimedia digital signal processing.

3. To explore the loci of different configurations of RB multipliers in the design space through comprehensive performance evaluations and structural analyses of different constituent modules and to deduce their overall implications on important but conflicting design constraints.

1.3 Major Contributions

Most of the work reported in this thesis has been focused on the architectural innovation of RB multipliers, with specific effort devoted to an extensive study of different RB multiplier configurations. It aims to provide better solutions in RB multiplier design, and the attempt has led to the following three major contributions and original results.

1. *Proposal of an efficient reverse converter for transforming the RB representation into NB domain*

A new RB-to-NB converter has been proposed for the communication of RB result through standard two's complement output interface. The converter has been implemented with a hybrid Carry-Lookahead Adder/Carry-Select Adder (CLA/CSL) method. The unique redundancy of RB coding has been utilized to formulate the reverse conversion problem with the carry recursion unrolled for efficient VLSI implementation. Mixed-radix carry generation trees for the CLA network has been explored. Logical Effort (LE) of both uniform and non-uniform block factor adder topologies have been analytically mod-

eled for different operand lengths in order to interleave different lengths of CSL with optimally matched delays in a carry prefix operator tree. The design features a CMOS circuit implementation of the ripple-carry adder chain optimized in a branch-based logic style to minimize the number of internal connections. A new add-one circuit has been proposed to further reduce the transistor count and its power consumption without speed penalty on the CSL section. The area-time ascendancy of our proposed reverse converter has been evinced by the total transistor count and the estimated delay time. A full-custom transistor-level implementation of a 64-bit converter circuit has been laid out with TSMC 0.18- μm CMOS process technology and a post-layout simulation has been performed to validate its performance.

2. Proposal of high-speed energy-efficient RB multiplier architectures based on new Covalent Redundant Binary Booth Encoding (CRBBE) algorithm

By exploiting the RB number system, a high-speed energy-efficient multiplier architecture has been proposed based on a new Booth Encoding algorithm. As the formation of a digit in the RBPP is analogous to the charge sharing of two oppositely charged atoms in a covalent bond, the proposed algorithm is named as the CRBBE algorithm. A polarization mapping is defined to combine two adjacent Booth encoded digits directly into an RBPP opposed to the conventional indirect grouping of two NB partial products after its generation. Consequently, the proposed method shares the same advantages of RB Booth encoder for the ease of generating the hard multiples and avoidance of correction vector. CRBBE generates the RBPPs more efficiently than the RB Booth encoder by consuming two RB digits for every RBPP it generated, which leads to less complex encoder and decoder for the same radix. The synthesis results show that the RB multiplier designed based on CRBBE algorithm out-

performs its rivals in terms of speed and energy efficiency for the power-of-two operand lengths.

3. A structural and systematic approach to the design and analysis of RB Booth multipliers

A structural and systematic approach has been proposed to design and analyze different RB multiplier architectures by decomposing them into several classes of constituent modules. The design considerations on each building modules, and their logic circuits have been qualitatively discussed and evaluated at a higher level of design abstraction. Altogether twenty-one different $N \times N$ -bit RB multiplier architectures have been constructed with various configurations of partial product encoding, generation and reduction to analyze their design tradeoffs in terms of area, delay and energy consumption. These RB multipliers have been implemented, simulated, analyzed and compared for various VLSI metrics with six commonly used operand lengths varying from 8 bits to 64 bits. The investigation has been carried out with a neutral standing using a consistent synthesis setup, and design guidelines have been drawn based on the appropriate figures of merit, such as energy per operation and energy-delay product. The deductions made are helpful to a system architect to select the most suitable multiplier topology with the desired characteristics.

The above contributions have led to the publications listed in the Author's Publications towards the end of the thesis.

1.4 Organization of the Thesis

The thesis is organized into six chapters. This chapter exposits the motivation and objectives of the research work. The achievements resulted from the work presented in this thesis are summarized into three main contributions in this chapter.

Chapter 2 presents the background information pertaining to the digital multiplier design. The base architecture of the RB multiplier focused in this research is outlined along with a comprehensive study of each of its building block. The design issues and challenges gathered from the literature survey are presented as a preamble to the more in-depth discussions of Chapters 3, 4, and 5. The last part of this chapter reviews the experimental methodologies, including the delay estimation method of LE, and the transistor-level and gate-level circuit optimization and simulation methodologies adopted in this research work.

The major contributions of this research are presented in Chapter 3 through Chapter 5. Chapter 3 focuses on the design of RB-to-NB converters. It shows the feasibility of adapting the reverse conversion algorithm to three different RB coding schemes before describing the proposed hybrid CLA/CSL based RB-to-NB converter architecture. Variants of circuit topologies for the parallel-prefix carry generation with uniform and non-uniform block factors are illustrated with the positive-negative RB coding scheme. An optimal implementation of a 64-bit reverse converter with the novel CSL circuit is detailed to elaborate the design concept. The performance evaluation using LE method of our proposed converter and the previous work are presented. The results are further corroborated with the pre-layout simulation results of two competitive

64-bit reverse converters implemented in 0.18- μm CMOS technology. Finally, the post-layout simulation results of the proposed converter are also reported.

Chapter 4 commences with an abridged description of the existing NB and RB Booth encoding algorithms and their overheads. A novel idea of CRBBE algorithm is then proposed. Radix-4 and Radix-16 CRBBE circuit designs and polarization mappings are illustrated to demonstrate their effective resolution of hard multiple problem and avoidance of error compensation vector. This is followed by the circuit implementation of a 64×64 -bit RB multiplier. The performance analysis of the RB multipliers based on the proposed CRBBE algorithm and their comparisons with other contenders are presented and discussed at the end of this chapter.

Chapter 5 presents a structural and systematic approach to the design and analysis of RB multipliers. A taxonomy of Booth Encoder and Partial Product Generator (BEPPG) is introduced for the ease of analysis. Different one-digit RBA cells and simple anterior and posterior converters of RBA summing tree for coherent RB coding interface are discussed. Twenty-one $N \times N$ -bit RB multiplier architectures are thus constructed from various configurations of partial product encoding, generation and reduction for design space exploration. The performance evaluation, discussion and concluding remarks on these competitive RB multiplier designs are provided at the end of this chapter.

Finally, Chapter 6 reviews the results achieved in this thesis and highlights the features and merits of the proposed methods. The pointers to several extended topics worthy of further research are also outlined.

Chapter 2

Digital Multiplier Architectures for Redundant Binary Arithmetic

2.1 Overview of Digital Multipliers

The VLSI implementation of digital multiplier is highly desirable for applications that involve a great deal of numerical computations. Generally, digital multipliers can be broadly classified into two categories based on their configurations. They are serial and parallel multipliers as shown in Figure 2.1. In a serial multiplier, the operands are input serially and hence its circuitry is relatively small and independent of the operand length. Therefore, the chip area and power consumption can be significantly minimized [40–42]. The drawback of serial multiplier is its severe speed limitation. Consequently, serial multipliers are usually used in applications where the IO is limited and bandwidth is ample. Though pipelining can be used to increase the throughput rate of serial multiplier [43], it is still far slower than its parallel counterpart.

Figure 2.1: Classification of digital multipliers.

In parallel multipliers, both operands are fed into the multiplier in a parallel mode [28, 44, 45]. The circuitry is more complex and occupies large silicon area [46, 47]. Depending on the structures of the primitive computing units of these parallel architectures, parallel multipliers are further classified into array and tree structured multipliers.

Array multipliers such as Braun multiplier [46] and Baugh-Wooley multiplier [20] have regular layout whereas Booth multiplier has fewer summands.

The Braun multiplier, invented by Braun Edward Louis in 1963 [46], is a relatively simple form of parallel multipliers. It is an intuitive paper-and-pencil method analogous to the way one would normally perform the multiplication by hand. Braun multiplier is also commonly known as the carry save array multiplier. This multiplier is well suited for multiplying two unsigned numbers. The iterative structure consists of an array of AND gates and adders

without any sequential logic or registers. The regular layout makes it ideal for VLSI and ASIC realization.

The Baugh-Wooley multiplier was designed by Bruce A. Wooley and Charles R. Baugh in 1973 [20]. This multiplier is actually an improved version of the Braun Multiplier as the hardware structures of both multipliers are very similar. However, Baugh-Wooley multiplier is able to operate with both the unsigned and signed numbers. It is conjectured that the invention of Baugh-Wooley multiplier has propelled the advent of computer arithmetic, because it is the first fast multiplier capable of performing both unsigned and signed multiplications in NB number system. An NB number is a weighted binary representation of an integer. The most frequently encountered NB number system today has signed numbers represented in two's complement form. Although Baugh-Wooley multiplier is time consuming and less efficient when dealing with large operands, it is nonetheless a good candidate even in today standard when the operands are less than 16 bits.

As early as 1951, A.D. Booth [48] introduced the Booth multiplier, which was also able to operate with both the unsigned and signed numbers. The Booth encoding algorithm provides a simple way to generate the product of two signed binary numbers by means of the Radix-2 arithmetic. The drawback of the original Booth's algorithm is that it becomes inefficient when there is a great number of isolated 1's in the operands. In 1961, MacSorley [49] proposed the modified Booth encoding algorithm, which is a parallel counterpart of the serial Booth encoding proposed specifically for the design and implementation of high-speed digital multiplier. For brevity, the modified Booth encoding is often referred to as Booth encoding in solid-state circuit. Soon after its introduction, the modified Booth encoding algorithm has evolved to a

ubiquitous algorithm in prevailing high-speed multipliers, especially for those that have to operate with large operands. Booth encoding has contributed significantly to the speedup and logic reduction on silicon implementation for two's complement multiplication.

For tree multipliers, the number of adder stages used for the addition of the partial products is minimized by arranging the adders in a binary tree form. Thus tree multipliers are faster than array multipliers. The first tree multiplier was introduced in 1964 by Wallace [45]. He suggested a notion of a Carry-Save Adder (CSA) tree as a way to efficiently and progressively reduce the multi-operand additions in the multiplication process to a final stage of two operand addition. The Wallace tree multiplier employs full and half adders to add up the partial products simultaneously in a parallel sequence. Later, Dadda [50] suggested an optimal compression scheme using different size counters (mainly 3-to-2 counters, which are full adders and 2-to-2 counters, which are half adders) and showed that different schemes of cell allocation, including the one introduced by Wallace, require different number of cells (counters). A natural VLSI layout of either Wallace or Dadda architecture is to distribute the cells such that the lengths of the interconnections are as short as possible. However, as the sum and carry signals need to be communicated to non-adjacent cells and propagated downwards across non-adjacent stages, the wiring and layout are irregular and more complicated than the array multipliers [12, 25, 29, 33, 51]. As these structures are not very regular to layout due to the asymmetric communication links of 3-to-2 counters within and across stages, 4-to-2 compressors (also known as 5-to-3 counter as it counts the number of '1' s in five binary inputs to produce a result of three binary bits) are suggested to facilitate the construction of a more regular binary tree. This ap-

proach was first introduced by Weinberger [52] in 1981 and improved by V. G. Oklobdzija and D. Villeger [27, 53] as a means to speed up the column compressions of the dot matrix representation of adder tree in parallel multipliers, since a 4-to-2 compressor can reduce four inputs of the same weight to two. It produces a more regular structure than one that is based on the use of 3-to-2 counters.

From an architectural perspective, the two basic operations in the multiplication algorithm, i.e., the generation of partial products and the accumulation of these partial products, are very crucial to its hardware performance. A retrospection of classic digital multiplier architectures reveal two lucid ways to speed up multiplication. One is to reduce the number of partial products; the other is to accelerate the accumulation process by minimizing its latency. Booth encoding algorithm has the advantage of reducing the number of partial products to be added, while the carry-saved adder tree approach using either 3-to-2 counters or 4-to-2 compressors speeds up the addition of partial products. When these two techniques are combined in a hybrid fashion, they can yield a multiplier that inherits the characteristics of traditional tree multiplier and Booth multiplier and is much faster than either one of them [54]. Today, this method is commonly used to realize high-speed two's complement multiplier since it is the fastest solution. It leads to the multiplication time proportional to the logarithm of the operand length [7]. Therefore, this research work will focus on the design and analysis of the equivalence of 4-to-2 compressor tree based Booth multiplier in RB regime.

Figure 2.2: Trichotomy of RB Booth multiplier architecture.

2.2 Redundant Binary Multiplier Architecture

Redundant binary representation is one of the signed digit representations first considered by Avizienis [55] in 1961 for fast parallel arithmetic. RB representation did not catch much attention until the early 1980's when Takagi *et al.* [56] proposed to apply this unconventional arithmetic to fast multiplication and Edamatsu *et al.* [5] implemented it in VLSI. There are at least two properties of RB arithmetic that make it a viable and potential substitute for the conventional NB multiplier: (1) the RBA can be configured to add any RB numbers free of carry propagation; (2) communications among RBAs within and across different layers of RBA tree are simpler than those of the full and half adders of CSA tree of NB multiplier. The use of RBA tree for the accumulation of partial products makes a highly modular and regular cell structure that can be easily

laid out on silicon. For ease of exposition, an RB multiplier is apportioned into three major building blocks. They are the BEPPG, the RBPP summing tree and the RB-to-NB converter. The anatomy of RB multiplier for functional analysis is shown in Figure 2.2. The BEPPG generates the RBPP according to the selected multiples. The RBA tree compresses multiple RBPPs to a single RB number. Finally, a reversed conversion is performed to output the result in NB format. Some known implementations of each of these building blocks will be reviewed in the following subsections.

2.2.1 Existing Booth Encoding Algorithms

In Booth multiplier, one of the two operands of multiplication is signed-digit encoded. The operand that is Booth encoded is called the multiplier and the other operand is called the multiplicand. The Booth encoding algorithm represents a simple and efficient way to reduce the number of summands required to produce the multiplication result. In radix- r Booth- k encoding ($r = 2^k$), a signed digit, d_i is generated from k adjacent multiplier bits, $b_{ki+k-1}b_{ki+k-2} \cdots b_{ki+1}b_{ki}$ and a borrow bit, b_{ki-1} as follows:

$$d_i = -2^{k-1}b_{ki+k-1} + \sum_{j=0}^{k-2} 2^j b_{ki+j} + b_{ki-1} \quad \text{for } i = 0, 1, \cdots, \left\lceil \frac{N}{k} \right\rceil - 1 \quad (2.1)$$

where k is a positive integer, $\lceil \alpha \rceil$ denotes the smallest integer value larger than or equal to α , N is the word length of the NB number B , and $b_{-1} = 0$.

When $k=1$, the Booth-1 digit d_i is converted from $b_i(b_{i-1})$ of the multiplier,

Table 2.1: Booth-1 Encoding

Normal Binary Bits		Normal Binary Bits	
$b_i(b_{i-1})$	Multiple	$b_i(b_{i-1})$	Multiple
0(0)	+0	1(0)	$-M$
0(0)	$+M$	1(0)	-0

B. The value of the encoded digit is given by:

$$d_i = -b_i + b_{i-1} \quad (2.2)$$

Table 2.1 shows the Booth-1 encoded digits and their corresponding binary bits with the overlapping bit in bracket. A multiple is the product of the Booth encoded digit, d_i and the multiplicand, M . From the multiples shown in Table 2.1, it is clear that Booth-1 encoding is almost useless in NB Booth multiplier. It does not help to reduce the number of partial products compared with a plain multiplier without any encoding.

In Booth-2 encoding, $k=2$, and every Booth-2 digit, d_i is mapped from the bits $b_{i+1}b_i(b_{i-1})$. Therefore, the value of the encoded digit d_i is given by:

$$d_i = -2b_{i+1} + b_i + b_{i-1} \quad (2.3)$$

In Booth-3 and Booth-4 encoding, the values of the encoded digits are expressed by (2.4) and (2.5), respectively.

$$d_i = -2^2b_{i+2} + 2b_{i+1} + b_i + b_{i-1} \quad (2.4)$$

$$d_i = -2^3b_{i+3} + 2^2b_{i+2} + 2b_{i+1} + b_i + b_{i-1} \quad (2.5)$$

Tables 2.2 to 2.4 list the multiples of Booth-2, Booth-3 and Booth-4 encoding, from all possible combinations of groups of 3, 4 and 5 multiplier bits, respectively.

Table 2.2: Booth-2 Encoding

Normal Binary Bits	Multiple	Normal Binary Bits	Multiple
$b_{2i+1}b_{2i}(b_{2i-1})$		$b_{2i+1}b_{2i}(b_{2i-1})$	
00(0)	+0	10(0)	$-2M$
00(1)	$+M$	10(1)	$-M$
01(0)	$+M$	11(0)	$-M$
01(1)	$+2M$	11(1)	-0

As the radix value, $r = 2k$ of the Booth- k (for positive integer k) encoded multiplier increases, the number of partial products decreases to $1/k$ of the original number. Intuitively, it is tempting to select as high as possible the radix of Booth encoding algorithm to encode the multiplier so as to reduce as many partial products as possible for the fastest multiplier. However, a close examination reveals that the number of multiples increases commensurately with the radix to $2k + 1$. Besides, the number of hard multiples, which are not the power-of-two factors of the multiplicand, also increases. These hard multiples are marked with '*' in Tables 2.3 and 2.4. For example, in Booth-3 encoding, there are two hard multiples, $\pm 3M$ out of a total of nine distinct multiples. While in Booth-4 encoding, there are eight hard multiples, which are $\pm 3M, \pm 5M, \pm 6M, \pm 7M$ out of seventeen distinct multiples. These hard

Table 2.3: Booth-3 Encoding

Normal Binary Bits	Multiple	Normal Binary Bits	Multiple
$b_{3i+2}b_{3i+1}b_{3i}(b_{3i-1})$		$b_{3i+2}b_{3i+1}b_{3i}(b_{3i-1})$	
000(0)	+0	100(0)	$-4M$
000(1)	$+M$	100(1)	$-3M^*$
001(0)	$+M$	101(0)	$-3M^*$
001(1)	$+2M$	101(1)	$-2M$
010(0)	$+2M$	110(0)	$-2M$
010(1)	$+3M^*$	110(1)	$-M$
011(0)	$+3M^*$	111(0)	$-M$
011(1)	$+4M$	111(1)	-0

* Hard multiples.

multiples cannot be obtained by simple shifting and/or complementation operations on the multiplicand. Additional time consuming CPAs are required to generate them. These CPAs increase the latency of the multiplier because the generation of partial products will not be accomplished until all these hard multiples are produced. Therefore, the advantage of Booth-3 and higher radix Booth encodings has been somewhat compromised due to the long delay and complex decoding logic required for the generation of hard multiples.

To speed up the generation of hard multiples in high-radix Booth encoding, a Partially Redundant Biased Booth Encoding (PRBBE) algorithm was proposed in [57]. Figure 2.3 depicts the generation and negation of the hard multiple, $3M$. It is generated in a partially redundant form by using a series of small

Table 2.4: Booth-4 Encoding

Normal Binary Bits	Multiple	Normal Binary Bits	Multiple
$b_{4i+3}b_{4i+2}b_{4i+1}b_{4i}(b_{4i-1})$		$b_{4i+3}b_{4i+2}b_{4i+1}b_{4i}(b_{4i-1})$	
0000(0)	+0	1000(0)	$-8M$
0000(1)	$+M$	1000(1)	$-7M^*$
0001(0)	$+M$	1001(0)	$-7M^*$
0001(1)	$+2M$	1001(1)	$-6M^*$
0010(0)	$+2M$	1010(0)	$-6M^*$
0010(1)	$+3M^*$	1010(1)	$-5M^*$
0011(0)	$+3M^*$	1011(0)	$-5M^*$
0011(1)	$+4M$	1011(1)	$-4M$
0100(0)	$+4M$	1100(0)	$-4M$
0100(1)	$+5M^*$	1100(1)	$-3M^*$
0101(0)	$+5M^*$	1101(0)	$-3M^*$
0101(1)	$+6M^*$	1101(1)	$-2M$
0110(0)	$+6M^*$	1110(0)	$-2M$
0110(1)	$+7M^*$	1110(1)	$-M$
0111(0)	$+7M^*$	1111(0)	$-M$
0111(1)	$+8M$	1111(1)	-0

* Hard multiples.

2.2 Redundant Binary Multiplier Architecture

23

Figure 2.3: $3M$ hard multiple generation and negation in partially redundant form [57].

length adders (4-bit). The carry bits of each small length adder is not propagated but brought forward to the partial product summing tree. However, when the $3M$ multiple is negated, both the sum and carry vectors need to be complemented and a '1' is added at their Least Significant Bit (LSB) positions. Therefore, the long strings of zeros between carries become strings of ones in the negative multiple. A properly selected biasing constant is introduced to revert the strings of ones back to strings of zeros. The '1's can be combined with the carry and sum bits to form a single compensation vector. The biasing constant of each such partial product introduces an extra compensation vector to the partial products summing tree.

The problem of generating hard multiples in high-radix Booth encoding was also addressed by Besli and Deshmukh [35, 36]. They noticed that some multiples can be obtained by subtracting one simple multiple from another, where a simple multiple refers to one that can be expressed as a power-of-two factor of the multiplicand. The partial products generated in this manner are in congruence with the format of the positive-negative RB coding. This RB coding format will be elaborated later in Section 2.2.2. This idea has led to a different Booth encoding logic, called the RB Booth Encoding (RBBE). Table 2.5 shows the RB Booth-3 encoding, where the original hard multiples $\pm 3M$ are replaced by $\pm(4M - M)$. Table 2.6 shows the RB Booth-4 encoding. Among the four hard multiples in the original Booth-4 encoding, $3M$, $6M$ and $7M$ are easily obtained by the subtraction of two simple multiples. The only exception is the hard multiple $5M$ (marked by '*' in Table 2.6), which cannot be generated in this manner. Therefore, additional hardware is necessary to generate the $5M$ multiple. A simple RB adder is suggested in [36] to add $4X$ and X , as shown in Figure 2.4. Fortunately, this RB addition is carry-free and it does not lie in

Figure 2.4: RB adder for $5M$ hard multiple generation.

the critical path of the RB BEPPG circuit.

Table 2.5: RB Booth-3 Encoding

Normal Binary Bits $b_{3i+2}b_{3i+1}b_{3i}(b_{3i-1})$	Multiple		Normal Binary Bits $b_{3i+2}b_{3i+1}b_{3i}(b_{3i-1})$	Multiple	
	$+M$	$-M$		$+M$	$-M$
0 0 0 (0)	0	0	1 0 0 (0)	0	$4M$
0 0 0 (1)	M	0	1 0 0 (1)	M	$4M$
0 0 1 (0)	M	0	1 0 1 (0)	M	$4M$
0 0 1 (1)	$2M$	0	1 0 1 (1)	0	$2M$
0 1 0 (0)	$2M$	0	1 1 0 (0)	0	$2M$
0 1 0 (1)	$4M$	M	1 1 0 (1)	0	M
0 1 1 (0)	$4M$	M	1 1 1 (0)	0	M
0 1 1 (1)	$4M$	0	1 1 1 (1)	0	0

2.2.2 Redundant Binary Adder

An RB number, in the context of this thesis, refers to a subset of a more generalized set of signed digit numbers [55]. It consists of digits from the set $\{\bar{1}, 0, 1\}$. By exploiting the redundancy of multi-bit binary representation of signed

Table 2.6: RB Booth-4 Encoding

Normal Binary Bits $b_{4i+3}b_{4i+2}b_{4i+1}b_{4i}(b_{4i-1})$	Multiple		Normal Binary Bits $b_{4i+3}b_{4i+2}b_{4i+1}b_{4i}(b_{4i-1})$	Multiple	
	$+M$	$-M$		$+M$	$-M$
0 0 0 0 (0)	0	0	1 0 0 0 (0)	0	$8M$
0 0 0 0 (1)	M	0	1 0 0 0 (1)	M	$8M$
0 0 0 1 (0)	M	0	1 0 0 1 (0)	M	$8M$
0 0 0 1 (1)	$2M$	0	1 0 0 1 (1)	$2M$	$8M$
0 0 1 0 (0)	$2M$	0	1 0 1 0 (0)	$2M$	$8M$
0 0 1 0 (1)	$4M$	M	1 0 1 0 (1)	0	$5M^*$
0 0 1 1 (0)	$4M$	M	1 0 1 1 (0)	0	$5M^*$
0 0 1 1 (1)	$4M$	0	1 0 1 1 (1)	0	$4M$
0 1 0 0 (0)	$4M$	0	1 1 0 0 (0)	0	$4M$
0 1 0 0 (1)	$5M^*$	0	1 1 0 0 (1)	M	$4M$
0 1 0 1 (0)	$5M^*$	0	1 1 0 1 (0)	M	$4M$
0 1 0 1 (1)	$8M$	$2M$	1 1 0 1 (1)	0	$2M$
0 1 1 0 (0)	$8M$	$2M$	1 1 1 0 (0)	0	$2M$
0 1 1 0 (1)	$8M$	M	1 1 1 0 (1)	0	M
0 1 1 1 (0)	$8M$	M	1 1 1 1 (0)	0	M
0 1 1 1 (1)	$8M$	0	1 1 1 1 (1)	0	0

* Hard multiples.

digit, an RBA is a unique and prerogative component in the RB multiplier design. It adds two RB digits to produce one RB digit in compliance with a set of

carry-free addition rules. These addition rules are designed to guarantee that the carry-out of an RBA is made independent of the actual carry-in. The compression ratio of an RBA is 2:1, which makes it behave like a 4-to-2 compressor used in the NB multiplier.

Table 2.7 illustrates all the nine possible combinations of input operands, a_i and b_i to an RBA. The RBA generates an intermediate sum s_i and an intermediate carry c_i before it outputs the final sum z_i . The carry-free addition rules for the RBA can be summarized as follows. Consider the i -th RBA that adds the i -th digits, a_i and b_i from two RB numbers. It receives h_{i-1} from the $(i-1)$ -th RBA, which is 0 if both inputs to the $(i-1)$ -th RBA are non-negative and 1 otherwise. This information is used advantageously by the RBA to generate an intermediate sum digit s_i and an intermediate carry digit c_i to avoid the propagation of carry. To eliminate the propagation of the possible carry-in of 1, an intermediate sum of $\bar{1}$ is generated and a carry-out of 1 is created to compensate for the required sum of 1. The final sum z_i is obtained by adding the current immediate sum s_i and the immediate carry c_{i-1} from the $(i-1)$ -th RBA. As the carry c_i is independent of c_{i-1} , the addition is carry free.

To implement RB arithmetic with standard logic elements, the RB number needs to be encoded into NB bit stream. According to the different mapping methods, there are three representative coding formats in RB number representation [58], [59] and [13]. Although the logic expressions of the RBA vary with the coding format used, they are essentially derived from the same adding rules. The underlying difference is the choice of appropriate intermediate control signals for the purpose of simplifying and optimizing the circuit in the selected coding format. In what follows, the design of RBA in each coding format will be discussed.

Table 2.7: Carry-Free Addition Rules for RBA

a_i	b_i	$a_{i-1}b_{i-1}$	h_{i-1}	c_i	s_i
0	0	don't care	any	0	0
$\bar{1}$	1			0	0
1	$\bar{1}$			0	0
0	$\bar{1}$	both are non-negative	0	0	$\bar{1}$
$\bar{1}$	0	otherwise	1	$\bar{1}$	1
0	1	both are non-negative	0	1	$\bar{1}$
1	0	otherwise	1	0	1
1	1	don't care	any	1	0
$\bar{1}$	$\bar{1}$			$\bar{1}$	0

2.2.2.1 RBA with Sign-Magnitude Coding

Table 2.8 shows the sign-magnitude coding for an RB digit: the bit on the right, d_i^a represents the magnitude of the signed digit, which is either '0' or '1', whereas the bit on the left, d_i^s indicates its sign, which is either '+' or '-'. This coding format is denoted as a dicit, (d_i^s, d_i^a) . The signed digit, D , can be expressed as:

$$D = (-1)^{d_i^s} \cdot d_i^a \quad (2.6)$$

As illustrated in [5, 58, 60], two intermediate signals, u_i and v_i , were introduced to make the realization of a simple circuit configuration compliant to the carry-free addition rule possible. Figure 2.5 shows the gate-level implementation of an RBA with sign-magnitude coding. It adds the two RB numbers

Table 2.8: Sign-Magnitude Coding [58]

Coding (d_i^s, d_i^a)	RB Digit D
(0, 0)	0
(0, 1)	1
(1, 0)	$\bar{0} \equiv 0$
(1, 1)	$\bar{1}$

Figure 2.5: Circuit implementation of an RB full adder with sign-magnitude coding.

expressed in the sign values, a_i^s and b_i^s , and the absolute values a_i^a and b_i^a to produce the sum bits, z_i^s and z_i^a . It is noted that, to simplify the circuit design, the input coding of (1,0) is refrained from feeding into the RBA directly and 0, 1 and $\bar{1}$ are completely specified by (0,0), (0,1) and (1,1), respectively.

2.2.2.2 RBA with Positive-Negative Coding

Another representation for RB number is the positive-negative coding as shown in Table 2.9. The value of the digit, D is equal to the subtraction of the

Table 2.9: Positive-Negative Coding [59]

Coding (d_i^+ , d_i^-)	RB Digit D
(0, 0)	0
(0, 1)	$\bar{1}$
(1, 0)	1
(1, 1)	0

Figure 2.6: Circuit implementation of an RB full adder with positive-negative coding.

right bit, d_i^- , from the left bit, d_i^+ , as indicated in (2.7) and Table 2.9.

$$D = d_i^+ - d_i^- \quad (2.7)$$

The RB full adder cell designed based on the positive-negative coding is first proposed in [59]. Its gate-level implementation is shown in Figure 2.6. It uses the intermediate signals, α_i and β_i to prevent continuous carry propagation by eliminating the collision of the sum and carry from the lower digit. Similarly, the inconsistent representation of zero, i.e., (1,1) needs to be removed from the input before the operands are fed into the RBA cell.

Table 2.10: Positive-Negative-Complement Coding [13]

Coding (d_i^+, d_i^-)	RB Digit D
(0, 0)	$\bar{1}$
(0, 1)	0
(1, 0)	0
(1, 1)	1

Figure 2.7: Circuit implementation of an RB full adder with positive-negative-complement coding.

2.2.2.3 RBA with Positive-Negative-Complement Coding

Table 2.10 shows the positive-negative-complement coding for an RB digit. The relationship between the values of digit D and its dibit, d_i^+ and d_i^- , is given by (2.8).

$$D = d_i^+ - \overline{d_i^-} \quad (2.8)$$

Figure 2.7 shows the gate-level implementation of an RBA cell with positive-negative-complement coding. Contrary to the previous two coding methods, there is no intermediate control signals required. c_j is the output carry signals, which can be calculated directly from the input signals so that the chain of carry propagation is limited to only one adder. Furthermore, due to the symmetry property of positive-negative-complement coding observed from Table 2.10, there is no preprocessing circuit required for each RB digit to avoid the inconsistent representations of '0' prior to its input into the RBA cell.

2.2.3 Conversion Between RB and NB Numbers

The use of RB number for digital multiplication is anomalous, or at least incompatible with the data transfer format through standard peripheral interfaces. The two input operands are, by de facto standard, assumed to be in two's complement form. Since the partial products generated by Booth encoding algorithm are NB numbers, to accumulate the NB partial products in an RBA tree, they must be converted to RBPPs using an NB-to-RB converter. To communicate the result to standard peripheral devices, the final product expressed in RB format also needs to be converted back to the NB number through an RB-to-NB converter.

The decimal value of an n -digit RB number, $F = (f_{n-1} f_{n-2} \cdots f_1 f_0)$ where $f_i \in \{0, 1, -1\}$, is given in (2.9). For an n -bit NB number $Z = (z_{n-1} z_{n-2} \cdots z_1 z_0)$ where $z_i \in \{0, 1\}$, its decimal value is given in (2.10). The conversion of an n -bit NB number representation to its RB number representation is simple and straightforward. It involves only the change of the Most Significant Bit (MSB). Thus, the time required by this conversion is independent of the operand

Figure 2.8: An example of RB-to-NB conversion process.

length. Therefore, the main overhead of RB multiplication process lies in the conversion of final partial product summation result from the RB form back to its NB representation.

$$F = \sum_{i=0}^{n-1} f_i \times 2^i \quad (2.9)$$

$$Z = -z_{n-1} \times 2^{n-1} + \sum_{i=0}^{n-2} z_i \times 2^i \quad (2.10)$$

A well known conversion process was illustrated by Hwang in [61]. In this conversion method, two NB numbers $T_1 = \sum_{t_i=1} t_i \cdot 2^i$ and $T_2 = \sum_{t_i=-1} (-t_i) \cdot 2^i$ are decomposed from an RB number, F in such a way that Z can be expressed as (2.11). A simple example is given in Figure 2.8 to illustrate this conversion process.

$$Z = T_1 - T_2 \quad (2.11)$$

The two's complement subtraction can be calculated as shown in (2.12). This implies that the reverse conversion can be performed directly by a two-operand CPA with a carry-in of '1' to the LSB.

$$Z = T_1 - T_2 = T_1 + \overline{T_2} + 1 \quad (2.12)$$

Figure 2.9: Block diagram of carry generation in RB-to-NB converter [59].

Traditionally, the RB-to-NB converter can be implemented in a straightforward way by a chain of serially connected full adders. Therefore, fast RB-to-NB conversion problem can be traced back to the origin of fast CPA logic [62]. A fast converter based on a carry-lookahead method was proposed in [63]. To simplify the carry generation logic, a new variable was defined to detect and signal carry propagation. In [64], a specialized carry propagation circuit was implemented with series Transmission Gates (TGs) to gain speed. A grouped carry-select method was proposed in [59] where the carry generation circuit was implemented with CSLs and grouped in such a way that the number of digits in the group increased by one progressively. The block diagram of carry generation in the conversion circuit is shown in Figure 2.9. This implementation is popular in subsequent RB multiplier designs [12, 14, 34, 35].

2.3 Review of Existing RB Multipliers – Challenges and Opportunities

Notwithstanding the carry-free addition property and regularity of RB adders, it has not intrigued as many new proposals of RB multiplier in its entirety as

envisioned. Over the last three decades, the number of RB multiplier proposals can be reviewed in three broad categories of architectures.

In the early 1980s, Takagi *et al.* [56] proposed a high-speed multiplication algorithm, which used RB representation internally. Based on this algorithm, the RB multiplier architecture presented in [51] was developed in three steps. Booth-2 encoding was applied in the first step to generate the partial products in NB representation. These partial products were added up pairwise in the second step, by means of a binary tree of RBAs. In the last step, the final product was converted into binary representation by means of a carry-lookahead adder. Based on this architecture, a 16-bit multiplier has been implemented on an LSI chip using a standard n-E/D MOS process [65]. It was the first silicon proved RB multiplier that demonstrated the speed competitiveness and performance repeatability in the digital multipliers of its time [33]. Later, enhanced performance RBA cell, as shown in Figure 2.5, was developed based on the sign-magnitude coding format [58]. Similar RB multiplier designs were then implemented with CMOS process in [5, 58, 60] to obtain faster CMOS multipliers with a reduced number of transistors and good layout regularity.

The noteworthy progress of RB arithmetic had certainly aroused the interest of computer architects and researchers to make further advancement in this field. In 1990s, a remarkably high-speed RB multiplier architecture was proposed by Makino *et al.* [59]. It was designed based on the positive-negative RB coding and it made two distinctions. One was the new design of RBA, as shown earlier in Figure 2.6. The other was the RB-to-NB converter, which was implemented efficiently with carry-select method as described in Section 2.2.3. This design was detailed further in [34] and [66]. A number of RB multipliers have been proposed thereafter based on the same architectural concept

[12, 14, 35, 36]. Among them, a conspicuous development came from Lee *et al.* [14]. This group of researchers proposed a radix-64 Booth encoding algorithm to emphatically improve the reduction rate of partial products. They defined 9 fundamental multiplying coefficients $\{0, 1, 2, 3, 4, 8, 16, 24, 32\}$ so that any of the 65 multiplying coefficients of Booth-6 encoder can be represented by an RB number made up of two fundamental multiplying coefficients. The idea was to reduce the number of multiplying coefficients needed to improve the reduction rate of partial products. In the mean time, Besli *et al.* [35, 36] proposed an RB Booth encoding algorithm to directly generate the partial products in RB format. This was perhaps the first successful resolution of hard multiple problem in high-radix Booth encoding algorithms by means of RB representation.

The third category of RB multiplier architectures was spearheaded by Kim *et al.* [13]. A novel RBA cell was proposed with positive-negative-complement coding as shown in Figure 2.7 [13, 67]. More importantly, this work presented a method for RB-to-NB conversion using a so-called equivalent bit conversion algorithm. It claimed to eliminate the need for carry propagation in the final conversion stage by taking the full advantage of RB multiplication process.

Unfortunately, there are more myths than truths in the acclamations of success in RB multiplier realm. The most well known falsehood is the carry-free equivalent bit conversion algorithm proposed for the RB-to-NB conversion in [13]. The myth of carry-free reverse conversion had been shattered by a flaw in the truth table used in this algorithm. A carry chain in the conversion stage was erroneously neglected. The errors had been detected by several researchers [68, 69] and it was proven later that carry propagation is ineluctable in any multiplication process [69]. As a matter of fact, for most RB multipliers, the critical path includes the RB-to-NB conversion. In [70], a direct-conversion

scheme was also proposed without any carry propagation to minimize this critical path delay for parallel architectures. Despite the constant latency (i.e., it is independent of the word size) of this converter, carry propagation had been re-introduced into the revised addition rule. Therefore, the declaration was misleading as the original carry-free addition property had been completely abolished in the RBA tree of this multiplier.

From those unsuccessful attempts, we can conclude affirmatively that the parallel transformation from any redundant number representation to NB number without incurring some degree of carry propagation is impossible. Some endeavor is required to optimize the trade-off between carry propagation and conversion efficiency. In RB multiplier, maintaining the carry-free addition property in the RBA tree is preferred to annihilating this property to improve the reverse conversion efficiency. With the carry-free RBA tree, the carry propagation is bound to be imposed on the final RB-to-NB conversion stage. The key point is that the carry-propagation delay occurs only once, at the very end, rather than in each addition step. Therefore, fast and efficient RB-to-NB converter, which is the performance bottleneck in the entire RB multiplier, is an optimization target of our research detailed in Chapter 3.

The BEPPG stage is yet another crucial stage in the trichotomy of RB multiplier architecture. Since negation in two's complement arithmetic requires carry propagation addition, negative partial product is more efficiently generated by a bit inversion of the multiplicand followed by an insertion of a '1' at its LSB position in the partial product summing tree. Therefore, one additional partial product row is generated to complete the two's complement negation of partial products for negative multiples. For example, Booth-2 encoding generates 5 instead of 4 partial products for an 8×8 -bit multiplication.

The additional delay required to add an extra partial product row critically slows down short operand length multiplier due to the relatively fewer number of adder stages in its partial product summing tree. This is the case especially for the power-of-two operand lengths, which are the most commonly encountered word lengths for application-specific data paths and general computing benchmarks [37, 38, 71–74]. Therefore, new RB multiplier architecture with BEPPGs that eradicates the overhead of negation, especially for power-of-two operand lengths, is another perspective that has not been delved into adequately. This topic will be investigated in Chapter 4.

Owing to the absence of hard multiple, Booth-2 encoding is attractive in digital multiplier design. In [14], Booth-6 (radix-64) encoding was claimed to be optimal for RB multiplier design. The claim was substantiated by the performance ascendancy of their proposed RB multiplier over other RB multipliers that used different radix Booth encoders. However, the comparisons were made based on the published experimental results targeted on different process technologies. From their experimental results, it is evident that the critical delay of Booth encoding and partial product generation of their scheme contributed to almost 41% of the total delay time, which was much higher than the 26% reported in [34]. Furthermore, a closer examination also reveals that the proposed Booth-6 encoder circuit, designed and optimized at transistor level, is actually Booth-3 encoder in disguise.

All in all, we observe a lack of systematic analysis of the fabrics of RB multiplier circuits. With better understanding of their performance tradeoffs, particularly in terms of energy dissipation and delay, it will be less tedious to tailor the RB multiplier design to different application specific constraints. Legitimate amalgamation of existing and newly proposed modules will be analyzed

in Chapter 5. The consolidation of these results will facilitate fruitful exploration of RB multiplier architectures with more desirable performance characteristics.

2.4 Experimental Methodology

Full-custom and semi-custom implementations are two design options for the study of RB multiplier circuits. Full-custom implementation is good for the design of new dedicated cells or unique functional blocks that could capitalize on different CMOS logic styles and device scaling at transistor level to maximize the performance of the circuit. However, full custom circuits are laborious to design and its portability from one technology to another is not assured. In addition, the time consuming transistor-level circuit simulation makes it difficult to fairly evaluate the circuits with a large number of inputs due to the curse of dimensionality. The accuracy of performance analysis, such as average power consumption, is stochastically dependent on the inputs. Consequently, a fair comparison of transistor-level circuits designed with different optimization agenda is not always possible. More often than not, different designs of the same function are compared based on the reported results or by empirical simulations using a transistor sizing technique best understood by the designer. In Chapter 3, we exploit the advantage of low-power pass transistor logic and circuit techniques at transistor level to design a new specialized RB-to-NB converter cell. Although onerous, it provides a good insight into the effect of meticulous device scaling and layout optimization in full-custom implementation at this level of complexity.

To take advantage of module generations for arithmetic circuits with different operand lengths and to expedite their simulations, design modeling at a higher level of abstraction using VHDL and semi-custom implementation are preferred. Design at gate level uses pre-designed logic components from proven standard-cell library. It facilitates a more extensive performance analysis and comparison of multipliers of different operand lengths using the same set of synthesis and optimization tools. Therefore, the circuits of Chapters 4 and 5 are designed, synthesized and simulated at gate level using a standard-cell semi-custom design flow.

This section provides some preliminaries pertaining to the optimization methods and simulation environments adopted for the design and simulation of components (at transistor level) and circuits (at gate level) of RB multipliers studied in this thesis. This arrangement is to minimize the reiteration of the same supplementary content in later chapters.

2.4.1 Logical Effort

Circuit topology affects performance. However, the performance of a chosen circuit topology could only be evaluated after the design is completed. In this regard, a consistent and accurate estimation model is a salvage to design effort as it can be used to analyze the performance before a design is implemented. The performance of a digital circuit is normally measured by its critical path delay, which is the worst-case delay of a circuit from any input transition to the latest output transition over all possible input patterns. As pointed out in [75], performance evaluation and comparison of complex gates based on unit gate delay model in CMOS digital circuits can be misleading because the delays

are largely influenced by their circuit topology (fan-in) and loading (fan-out). To provide a fast and consistent means to estimate the critical path speed, we have adopted the LE model.

LE is based on the first order linear RC delay model and it provides a convenient shorthand for more realistic speed estimation of CMOS digital circuits [76]. This technique accounts for the fact that the speed of a given logic circuit is dependent on both its fan-in and fan-out. LE makes technology-independent comparison of different architectures for the same function implemented in different process technologies feasible by normalizing the speed to that of a minimum-sized inverter. The method can capture reasonably well the effect of transistor sizings according to the critical paths of the corresponding circuit architectures and their parasitics. The accuracy of LE estimation has been attested by reliable circuit simulation tool HSPICE by many researchers [77–79].

This sub-section briefly presents the fundamental formulae of LE from [76] for an appreciation of our adoption of this methodology.

The delay of a logic block, d , in LE is given by (2.13).

$$d = gh + p \quad (2.13)$$

The meanings of each variable used in (2.13) are explained as follows:

LE, g is the total gate capacitance of a logic gate relative to that of a minimum-sized inverter. It characterizes the influence of a logic gate's structure to its current drive to an output load.

Electrical effort, h is the ratio of the output capacitance of a gate to its input capacitance. It describes the influence of the load of a logic gate to its perfor-

mance. It also indicates how sizing of transistors determines the load-driving capability.

Parasitic delay, p is the total diffusion capacitance on the output node of a gate relative to that of a minimum-sized inverter. It captures the intrinsic delay of the gate due to its own internal capacitance.

Equation (2.13) is used to estimate the delay of a single gate. It models the delay contributed independently by the LE g , electrical effort h , and parasitic delay p . The delay of a gate is the sum of the parasitic delay and the effort delay, which is the product of g and h . This delay is expressed in terms of a basic delay unit denoted by τ . The absolute value of τ , in ns depends on the process technology, but the unitless delay expressed in τ is consistent across a broad range of process technologies.

Figure 2.10 shows an example of the delay estimation by LE for an inverter driving four identical inverters. Since each inverter is identical, $C_{out} = 4C_{in}$, so $h = 4$. The LE g of an inverter is one by definition, and the typical parasitic delay p of an inverter is also one. According to (2.13), $d = gh + p = 1 \times 4 + 1 = 5$. This circuit delay is known as the Fanout-of-4 (FO4) delay. FO4 delay is useful in expressing delay in a process-independent way since most designers know the delay of an FO4 inverter in their process. Therefore it can be used conveniently to predict how a circuit performance will scale when it is ported to other processes.

Table 2.11 summarizes the LE and parasitic delay of commonly used logic gates. The delay of a path is the sum of the delays of all gates along the path. The principle expressions of LE are summarized in Table 2.12. It involves the branching effort, b , which is defined as the ratio of the total capacitive load on

Figure 2.10: FO4 delay illustration.

Table 2.11: The Logical Effort and Parasitic Delay of Common Logic Gates

Logical Gate		Logical Effort g	Parasitic Delay p
Inverter		1	1
NAND	2-input	$4/3$	2
	3-input	$5/3$	3
	4-input	$6/3$	4
NOR	2-input	$5/3$	2
	3-input	$7/3$	3
	4-input	$9/3$	4
XOR	2-input	4	4

one logic gate's output to the gate capacitance of the next gate on the path in examination. If the path does not branch, the branching effort is one. The path branching effort B is the product of the branching effort at each of the stages along the path, as indicated in Table 2.12.

The total path effort reflects the complexity of a path, taking into account the LE and load of each gate along the path. If the total path effort F is determined and the path has N stages, the path effort delay D_F is minimized when

Table 2.12: Key Definitions of Logical Effort

Term	Stage Expression	Path Expression
Logical Effort	g (refer to Table 2.11)	$G = \prod g_i$
Electrical Effort	$h = \frac{c_{out}}{c_{in}}$	$H = \frac{C_{out}(\text{path})}{C_{in}(\text{path})}$
Branching Effort	N.A.	$B = \prod b_i$
Effort	$f = g \cdot h$	$F = G \cdot B \cdot H$
Effort Delay	f	$D_f = \sum f_i$
Number of Stages	1	N
Parasitic Delay	p (refer to Table 2.11)	$P = \sum p_i$
Delay	$d = f + p$	$D = D_F + P$

each stage bears the same effort, f , which is given as:

$$f = g_i h_i = F^{\frac{1}{N}} \quad (2.14)$$

In such a case, the path delay, D , will be equal to:

$$D = N \cdot F^{\frac{1}{N}} + P = N \cdot f + P \quad (2.15)$$

Suppose the input load of a path is C_1 , to account for the large fan-out at the input of the path, it is normally assumed that the circuit drives a copy of itself. In this case, the output load of the path $C_{L,N}$ is the total input load of

the path that includes C_1 and its fan-outs. The electrical effort, H , of the path can be considered as $C_1/C_1 = 1$, and the branching effort of the last stage will become $C_{L,N}/C_1$.

2.4.2 Transistor-Level Circuit Optimization and Simulation

As shown in [80], the transistor sizing for optimal performance is technology dependent. The scaling operations are carried out in iterations transistor by transistor. To provide a good tradeoff between the somewhat conflicting power and delay performances, the goal of optimization is to minimize the product of the worst-case delay and the average power consumption. Suppose a circuit for optimization is composed of N transistors, labeled from T_1 to T_N and they are initialized with reasonable sizes at the outset. For a certain technology the channel lengths of all transistors are fixed at the minimal feature size, so the only variable to be optimized is the channel width of each transistor. The first optimization run is begun with varying the channel width of T_1 in $2m + 1$ steps and a step size of ψ to probe the circuit performance. In other words, the different channel widths of T_1 simulated are $l_{1,0} - m\psi$, $l_{1,0} - (m - 1)\psi, \dots, l_{1,0}, \dots, l_{1,0} + (m - 1)\psi, l_{1,0} + m\psi$, where $l_{1,0}$ is the initial size of T_1 . The probing sizes for T_1 are formally expressed as

$$l_{1,i} = l_{1,0} + i\psi \quad \text{for } i = -m, -m + 1, \dots, 0, \dots, m - 1, m \quad (2.16)$$

During this run, the sizes of all other transistors remain unchanged.

Suppose that the j -th channel width $l_{1,j}$ of T_1 provides the circuit with the lowest PDP through the simulation. We update T_1 with $l_{1,j}$ and carry on with

the second run for T_2 . After the second run, T_2 will be updated with its best channel width. The process goes on until the last transistor T_N is updated. An iteration is said to be completed when all the transistors have been updated. However, one iteration is not sufficient for the optimization because when a new transistor is sized in the current run, the other transistor sizes updated in the previous run may no longer maintain its optimality. Therefore, more iterations beginning with T_1 are needed. The iteration process stops when the performance difference in two successive iterations is smaller than a given error ε . Let Θ_{i-1} and Θ_i be the optimized PDP at the end of the $(i-1)$ -th and i -th iterations, respectively. The termination criterion is given by:

$$\frac{\Theta_{i-1} - \Theta_i}{\Theta_i} \leq \varepsilon \quad (2.17)$$

Figure 2.11 indicates the flow chart of transistor sizing procedure. In order to obtain enough coverage so that the optimal or quasi-optimal operating point would fall into the search region, and to allow for fine calibration, the resolution of the sizing step ψ may be made variable. Large step size is used at the first few iterations and smaller step size is used for the remaining iterations.

The transistor-level simulations mentioned in this thesis are performed by HSPICE [81] based on the TSMC 0.18- μm CMOS process model. Figure 2.12 illustrates the simulation setup environment. All measurements were taken with each input signal pulse-shaped by a driver consisting of two inverters in series, and each output node driving two unit sized inverter load.

The delay is measured from the earliest input signal reaching 50% of the supply voltage to the latest output signal reaching 50% of the supply voltage for each input cycle. The worst-case delay is the largest delay among all input

Figure 2.11: Transistor sizing optimization flowchart.

Figure 2.12: Transistor-level circuit simulation environment.

data. For each simulation, HSPICE measures the power consumed by a circuit for a given set of inputs. Since the power dissipation is a strong function of the inputs, the circuit under test is presented with thousands of independent, pseudorandom input stimuli.

2.4.3 Gate-Level Synthesis and Power Simulation

A standard cell library is a collection of low level logic functions, which are realized as fixed height, variable width full-custom cells. A typical standard-cell library contains two main components:

1. Timing Abstract: This provides functional definitions, timing, power, and noise information for each cell.
2. Layout Abstract: This contains reduced information about the cell layouts, which is sufficient for automated “Place and Route” tools.

The circuit design using standard-cell library is described at gate level in VHDL. They are synthesized and mapped to Artisan TSMC 0.18- μm CMOS standard-cell library [82] using the Synopsys Design Compiler (DC) [83] with a specified wire load model. All simulations are carried out at supply voltage

of 1.8V and a room temperature of 25°C. A standard buffer of strength 2X is used for both input drive and output load. The option for logic structuring was turned off to prevent the tool from changing the structure of the unit cells.

The average power consumptions are simulated by Synopsys Power Compiler [83] with Monte Carlo statistical method [17, 84–88]. This method is widely used for simulating the behavior of stochastic computations. Its use of randomness and the repetitive nature of the process are analogous to the activities conducted at a casino. With this method, power is calculated till a level of confidence is reached with a tolerable error. The advantage is that it quantifies the accuracy of the average power obtained for a set of random input vectors that are used to simulate the circuit to determine the switching activity. The switching activity in Switching Activity Interchange Format (SAIF) was annotated from random input vectors for running a power analysis. Since the inputs are independent, power can be approximated to be normally distributed [85]. Hence the mean power dissipation of the circuit is given by

$$P = \mu + t_{\alpha} \cdot \left(\frac{\sigma}{\sqrt{S}} \right) \quad (2.18)$$

where μ is the sample mean, σ is the standard deviation, S is the sample size and t_{α} is t -distribution value that corresponds to a confidence level of α .

The term $t_{\alpha} \cdot \left(\frac{\sigma}{\sqrt{S}} \right)$ is the error that is associated with the simulation. So after plotting the values, if an error ε is assumed, the factor t_{α} can be determined. This t_{α} can be used to determine the confidence level from the statistical tables. By this means, we can convincingly claim that with a confidence level α , an average power P obtained by the simulation has an error bound, ε .

Chapter 3

Hybrid Carry-Lookahead/Carry-Select Based RB-to-NB Converter

3.1 Introduction

A typical multiplier produces an output of the same format as its input operands. As the accustomed bus architectures of standard peripheral devices are still based on the NB number representation, an additional reverse conversion step is indispensable in the final stage of the RB multiplier to convert the summation result in RB form back to NB domain. Unfortunately, this stage appears to be the performance bottleneck of the entire RB multiplier architecture.

As illustrated in Section 2.2.3, techniques endeavor to circumvent the RB-to-NB conversion problem are affiliated to the family of CPAs. Several fast parallel adder architectures for the reverse conversion process have been proposed in [34, 63, 64], each of which is tailored to a specific RB coding scheme. Different coding methods offer subtle optimization variants, giving rise to ded-

icated circuit implementations consequently. Since the conversions into and out of RB regime reciprocate the intermediate RB arithmetic, we ask if there is a way to unify the conversion algorithm so that an efficient architecture can be devised to adapt to all three coding methods presented in Chapter 2. Instead of reconciling the conversion to suit the coding format, the reverse is proposed in this chapter. By freeing the restriction of coding format, the same efficient RB-to-NB converter can then be used with different front end amalgamations in the trichotomy of RB multiplier. As will be seen in Chapter 5, the RB-to-NB converter architecture developed in this chapter has helped to spawn several new configurations of RB multiplier and eased the exploration of design features of many RB multiplier topologies.

Among the parallel adders, hybrid carry-lookahead and carry-select adder has been widely accepted as the most efficient adder of good overall area, delay and power consumption performances. It has been employed for the design of various fast adders in the NB regime [75, 89, 90]. Motivated by a general architecture recently proposed by Wang *et al.* [91], this chapter is dedicated to the design and development of an efficient RB-to-NB converter by incorporating such fast addition technique with new circuit design strategies.

The reverse conversion problem is formulated by exploiting the characteristics of RB coding to unroll the carry recursion for efficient hybrid CLA/CSL implementation. The redundancy of the binary subtraction encoding of signed digit is used to simplify the first stage of the carry generation network as well as the constituent Ripple-Carry Adder (RCA) of CSL section. Furthermore, the mixed-radix carry generation trees for the CLA network are explored. The LE of both uniform and non-uniform block factor adder topologies are analytically modeled for different operand lengths, with the interleaving of different

CSL section lengths. The carries are generated based on the Kogge-Stone prefix operator tree [92] by virtue of its fast speed and modularity. For a fair performance benchmarking of our proposed reverse converter architecture, different reverse converters of varying operand lengths are optimized using the same recursive sizing procedure according to the critical paths estimated from their respective architectures. The area-time ascendancy of our proposed reverse converter is evinced by the total transistor count and the LE delay estimation. Towards this end, a 64-bit converter circuit has been implemented in transistor level to validate its performance.

The remainder of this chapter is organized as follows: Section 3.2 shows the feasibility of adapting reverse conversion algorithm to three different RB coding schemes. Section 3.3 describes the proposed hybrid CLA/CSL based RB-to-NB converter architecture and its variants of circuit topologies for the parallel-prefix carry generation with uniform and non-uniform block factors. An optimal implementation of a 64-bit reverse converter with the novel CSL circuit is detailed in Section 3.4 to elaborate the design concept. The performance evaluation of our converter and previous work using LE method is presented in Section 3.5, along with the pre-layout HSPICE simulation results of two competitive 64-bit reverse converters implemented in 0.18- μm CMOS technology. The post-layout simulation results of our proposed converter are also reported. The chapter is closed with a summary in Section 3.6. A part of the work in Section 3.4.2 has been presented at the 2005 IEEE International Symposium on Circuits and Systems [93]. The work in Section 3.4 has been presented at the 2006 International Symposium on Circuits and Systems [94]. A revised manuscript that contains a large portion of the work presented in this chapter is currently being reviewed as a regular paper in the *IEEE Transac-*

tions on Circuits and Systems–I.

3.2 Reconciliation of Coding Formats for Unanimity of RB-to-NB Conversion

As indicated in Section 2.2.3, the reverse conversion can be performed by a two-operand CPA with a carry-in of ‘1’ to the LSB. According to (2.12), if $F = (F^+, F^-)$ represents the final RBPP with positive-negative coding, the NB result Z can be expressed as:

$$Z = F^+ - F^- = F^+ + \overline{F^-} + 1 \quad (3.1)$$

where F^+ and F^- are akin to T_1 and T_2 of (2.12), respectively.

Let (f_i^+, f_i^-) denote a digit of the final RBPP (F^+, F^-) , and c_{i-1} be the carry-in from the next lower order digit, then the sum output of the partial products z_i , and the carry-out signal c_i , can be derived from (3.1) as follows:

$$\begin{cases} z_i = f_i^+ \oplus \overline{f_i^-} \oplus c_{i-1} \\ c_i = f_i^+ \cdot \overline{f_i^-} + f_i^+ \cdot c_{i-1} + \overline{f_i^-} \cdot c_{i-1}, \quad c_{-1} = 1 \end{cases} \quad (3.2)$$

where $i = 0, 1, \dots, N - 1$, and N is the number of digits of the final RBPP to be converted to NB number. c_{-1} represents the first carry-in to the Least Significant Digit (LSD) of the RB number.

We make use of the fact that in RB multipliers: the binary pair representing each RB digit can never become ‘1’ simultaneously. This is because (1, 1) has been converted to (0, 0) before the RBA tree stage to eliminate the inconsistent

54 3.2 Reconciliation of Coding Formats for Unanimity of RB-to-NB Conversion

representations of '0' in order to simplify the design of RBA cell as described in [34]. The inherent redundancy of this coding format gives rise to the following simplifications for the generation of carry-generate bit g_i , carry-propagate bit p_i , and the half-sum bit d_i .

$$\begin{cases} g_i = f_i^+ \cdot \overline{f_i^-} = f_i^+ \\ p_i = f_i^+ + \overline{f_i^-} = \overline{f_i^-} \\ d_i = f_i^+ \oplus \overline{f_i^-} = \overline{f_i^+ + f_i^-} \end{cases} \quad (3.3)$$

With (3.3), (3.2) can be easily rewritten as follows:

$$\begin{cases} z_i = d_i \oplus c_{i-1} = (\overline{f_i^+ + f_i^-}) \oplus c_{i-1} \\ c_i = g_i + p_i \cdot c_{i-1} = f_i^+ + \overline{f_i^-} \cdot c_{i-1}, \quad c_{-1} = 1 \end{cases} \quad (3.4)$$

Similarly, according to (2.12), if $F = (F^+, F^-)$ represents the RB result with positive-negative-complement coding, the NB result Z can be expressed as (3.5) and consequently, the sum bit z_i and carry-out signal c_i can be derived in (3.6).

$$Z = F^+ - \overline{F^-} = F^+ + F^- + 1 \quad (3.5)$$

$$\begin{cases} z_i = f_i^+ \oplus f_i^- \oplus c_{i-1} \\ c_i = f_i^+ \cdot f_i^- + f_i^+ \cdot c_{i-1} + f_i^- \cdot c_{i-1}, \quad c_{-1} = 1 \end{cases} \quad (3.6)$$

In (3.5), F^+ is akin to T_1 and $\overline{F^-}$ is akin to T_2 of (2.12).

As mentioned in Section 2.2.2.3, due to the symmetry property of positive-negative-complement coding, the redundant representation of '0' in the partial product digits can be tolerated prior to its input into the RBA cell. However, to keep the conversion algorithm concise and consistent in each coding method,

3.2 Reconciliation of Coding Formats for Unanimity of RB-to-NB Conversion 55

the inconsistent representations of '0' should be eliminated, i.e., the (1, 0) case should be converted to (0, 1), at the end of the RBA summing tree. Therefore, the carry-generate bit g_i , carry-propagate bit p_i , and the half-sum bit d_i can be determined as follows:

$$\begin{cases} g_i = f_i^+ \cdot f_i^- = f_i^+ \\ p_i = f_i^+ + f_i^- = f_i^- \\ d_i = f_i^+ \oplus f_i^- \end{cases} \quad (3.7)$$

With (3.7), the carry-out signal in (3.6) can be rewritten as:

$$c_i = f_i^+ + f_i^- \cdot c_{i-1}, \quad c_{-1} = 1 \quad (3.8)$$

The carry generation of (3.8) is similar to that of (3.4). This is because the positive-negative coding and the positive-negative-complement coding are defined as the difference of two NB numbers akin to T_1 and T_2 of (2.12). In order to unify the reverse conversion of RB number, the sign-magnitude coded RB number needs to be expressed as a difference of two NB numbers as well.

If $F = (F^s, F^a)$ represents the final RBPP in sign-magnitude coding, the corresponding NB number Z can be derived as:

$$Z = \overline{F^s} \cdot F^a - F^s \cdot F^a = \overline{F^s} \cdot F^a + \overline{F^s \cdot F^a} + 1 \quad (3.9)$$

where $(\overline{F^s} \cdot F^a)$ is akin to T_1 and $(F^s \cdot F^a)$ is akin to T_2 of (2.12)

Let (f_i^s, f_i^a) denote each digit in (F^s, F^a) , and c_{i-1} be the carry-in from the next lower order digit, then the sum output of the partial products z_i , and the

carry-out signal c_i in the conversion, can be derived as follows:

$$\begin{cases} z_i = (\overline{f_i^s} \cdot f_i^a) \oplus (\overline{f_i^s} \cdot \overline{f_i^a}) \oplus c_{i-1} \\ c_i = (\overline{f_i^s} \cdot f_i^a) \cdot (\overline{f_i^s} \cdot \overline{f_i^a}) + (\overline{f_i^s} \cdot f_i^a) \cdot c_{i-1} + (\overline{f_i^s} \cdot \overline{f_i^a}) \cdot c_{i-1}, \quad c_{-1} = 1 \end{cases} \quad (3.10)$$

Since (1, 0) has been converted to (0, 0) before the RBA summing tree stage to eliminate the inconsistent representations of '0' as described in Section 2.2.2.1, the expression of g_i , p_i and d_i signals can be derived as:

$$\begin{cases} g_i = (\overline{f_i^s} \cdot f_i^a) \cdot (\overline{f_i^s} \cdot \overline{f_i^a}) = \overline{f_i^s} \cdot f_i^a \\ p_i = (\overline{f_i^s} \cdot f_i^a) + (\overline{f_i^s} \cdot \overline{f_i^a}) = \overline{f_i^s} \\ d_i = (\overline{f_i^s} \cdot f_i^a) \oplus (\overline{f_i^s} \cdot \overline{f_i^a}) = \overline{f_i^a} \end{cases} \quad (3.11)$$

With (3.11), the sum and carry-out bits in (3.10) can be simplified to:

$$\begin{cases} z_i = f_i^a \oplus \overline{c_{i-1}} \\ \overline{c_i} = f_i^s + \overline{f_i^a} \cdot \overline{c_{i-1}}, \quad c_{-1} = 1 \end{cases} \quad (3.12)$$

From (3.4), (3.8) and (3.12), it is obvious that the reverse conversion of different RB coded numbers shares the same logical structure. This is important because a unanimity of carry generation makes the conversion algorithm unified for all three coding methods without incurring much overhead.

In what follows, a new architectural translation of a reverse conversion algorithm is elaborated based on the positive-negative coding format. This coding format has been used pervasively in the design of RB multipliers [34–36, 64]. Therefore, it is chosen to exposit the proposed RB-to-NB converter for the convenience of performance benchmarking. It should be noted, however,

3.3 Exploration on Hybrid CLA/CSL Architecture for RB-to-NB Conversion 57

that using the above coding reconciliation, the proposed architecture can be easily developed for the other two coding formats as well.

3.3 Exploration on Hybrid CLA/CSL Architecture for RB-to-NB Conversion

Many adder types in NB regime exist and each has its own advantages and disadvantages. By far, the most comprehensive study from VLSI perspective of parallel adders has been provided in [17]. Among the two operand parallel adders, CLA with ELM [95] and B&K [96] adders being special variants, is widely known as the fastest adder with huge hardware cost, RCA consumes least chip area but has the longest delay time, and CSL is intermediate in performance between speed and area. To speed up CSL computation, CLA architecture is used to generate the select signals of CSL, leading to the hybrid adder of CLA/CSL [75, 89–91]. In this section, we will further explore the parallel and unidirectional generation of carry-select signals by leveraging on the dedicated RB number encoding for the RB-to-NB reverse conversion algorithm.

3.3.1 Hybrid CLA/CSL Based Reverse Conversion Algorithm

In a hybrid CLA/CSL circuit, the selected carry signals and sum bits are generated simultaneously by the cooperative execution of CLA and CSL networks. The selected carries are generated by a CLA tree without back propagation. The number of carry outputs to be generated is significantly reduced with reg-

ular interleaves of CSL sections. The sum bits are computed in sections by the CSL. Conventionally, each section of CSL is implemented with dual RCA blocks with the constant carry-in of 0 and 1, respectively. Due to the anticipatory parallel computation, once the carry signal for the local section is generated by the carry-lookahead network, the corresponding carry-select adders will choose the correct sum and produce the output directly.

From (3.4), by unrolling the c_i recursion, we obtain the following equations:

$$\begin{aligned}
 c_0 &= f_0^+ + \overline{f_0^-} \cdot c_{-1} = \overline{f_0^-} \\
 c_1 &= f_1^+ + \overline{f_1^-} \cdot c_0 = f_1^+ + \overline{f_1^-} \cdot \overline{f_0^-} \\
 &\vdots \\
 c_i &= f_i^+ + \overline{f_i^-} \cdot f_{i-1}^+ + \dots + \overline{f_i^-} \cdot \overline{f_{i-1}^-} \cdots \overline{f_2^-} \cdot \overline{f_1^-} \cdot \overline{f_0^-} \\
 &= \overline{f_i^-} \cdot \{f_i^+ + f_{i-1}^+ + \dots + \overline{f_{i-1}^-} \cdots \overline{f_{i-r}^-} \cdot f_{i-r-1}^+ + \dots + \overline{f_{i-1}^-} \cdots \overline{f_1^-} \cdot \overline{f_0^-}\} \\
 &= \overline{f_i^-} \cdot H_i
 \end{aligned} \tag{3.13}$$

where $H_i = f_i^+ + f_{i-1}^+ + \bigvee_{j=1}^{i-2} \left(\bigwedge_{r=j+1}^{i-1} \overline{f_r^-} \right) \cdot f_j^+ + \bigwedge_{r=0}^{i-1} \overline{f_r^-}$ and the operators $\vee$ and $\wedge$ denote the Boolean sum and product, respectively.

Based on (3.13), H_i can be expanded by iteratively unrolling the recursion as follows:

$$\begin{aligned}
 H_i &= G_k + P_k \cdot H_{i-1} = G_k + P_k \cdot G_{k-1} + P_k \cdot P_{k-1} \cdot H_{i-2} \\
 &= G_k + P_k \cdot G_{k-1} + \dots + P_k \cdots P_{k-r} \cdot G_{k-r-1} + \dots + P_k \cdots P_2 \cdot G_1
 \end{aligned} \tag{3.14}$$

where $P_k = \overline{f_{i-1}^-} \cdot \overline{f_{i-2}^-} \cdot \overline{f_{i-3}^-}$, $G_k = f_i^+ + f_{i-1}^+ + \overline{f_{i-1}^-} \cdot f_{i-2}^+$, $G_1 = f_2^+ + f_1^+ + \overline{f_1^-} \cdot \overline{f_0^-}$.

For ease of exposition, a block factor of three is assumed in the above

derivation, i.e., P_k and G_k are Boolean product and sum of three terms, respectively. The decomposition of H_{i-1} in the derivation of (3.14) shows that only some instead of all of the carries need to be generated. The integer index, i , signifies the bit position of the selected carry signal to be generated from the carry-lookahead unit. The number of carry generation units is dependent on the operand length. The integer index, k is used to uniquely identify each carry generation unit. If N is the length of the RB operand to be converted to two's complement number, then the range of all integer values of k , and the positions of all the carry signals, i can be determined as follows:

$$1 \leq k \leq \lfloor N/3 \rfloor, \quad i = 3 \cdot k - 1 \quad (3.15)$$

where $\lfloor a \rfloor$ denotes the largest integer value not exceeding a .

More levels of decomposition are also possible following the similar approach of derivation according to the operand length N . H_i , and hence c_i , can be generated in a hierarchy of homogeneous carry-lookahead units. These signals are the inputs to the CSL sections at the leaves of the lookahead tree. Such conversion algorithm provides similar advantageous structure as parallel-prefix Ling's carry generation algorithm [97]. Instead of generating all the carry propagation signals like traditional parallel prefix adder, our hybrid CLA/CSL conversion method can make use of pseudo-carries to create the selected carry propagation signals.

In the above derivation, the block factor is assumed to be uniform and equal to 3. It is noted that in hybrid CLA/CSL configuration, the choice of carries in the carry network varies with the block factor of CLA and the block length of CSL. It also affects the internal load distribution of the lookahead logic and

the depth of the carry tree. For a fixed word length of RB operand, more than one solutions are available to implement the hybrid CLA/CSL based reverse converter. Typically, the block factor and block length are chosen to equalize the critical carry generation chain and the carry-select chain. The timings of these two chains are highly dependent on the logic styles and fan-out factor per gate for a given technology of implementation. In what follows, we will probe on this structural optimization with reference to CMOS circuit and branch-based logic design style [98], whereby the logic cells involved are made of parallel branches, each with a limited number of serially connected transistors.

3.3.2 Parallel-Prefix Carry-Lookahead with Uniform and non-Uniform Block Factors

Motivated by the design space of hybrid CLA/CSL architectures for RB-to-NB reverse conversion, our aim in this section is to find an optimal point to combine the CLA and CSL sections for better performance by analyzing the consequential carry generation schemes with uniform and non-uniform block factors. The block factor refers to the number of binary terms in the Boolean product, P_i and Boolean sum, G_i in the generalized expression of (3.14). For layout regularity and balanced multiplexer load, it makes good sense to interleave the carry-select signals generated by the CLA network evenly to all CSL sections. While keeping the block lengths of CSL sections identical, the block factors at different stages of the carry generation network can vary to minimize the difference between the arrival time of carry-select signal and the critical delay of RCA in the CSL section.

For an N -bit RB operand, let l indicate the depth of the carry generation

tree, i.e., the maximum number of lookahead cells traversed from any input to the final carry generation unit. Further, let b_i denote the block factor of the cells at stage i where $i = 1, 2, \dots, l$. When non-uniform block factor is used, it is important to use an interconnection structure that is regular to ease its implementation. The Kogge-Stone liked tree [92] is adopted for our study as the fan-out of each cell throughout the network can be made fairly constant, especially for those that lie on the critical paths. To account for the number of cells with varying fan-ins and the carries they generated for a given operand length, a transitive stage, $t \in (1, l]$ is defined as a stage in the carry generation network where the outputs of all cells in this stage are separated by exactly the block length, m of the CSL section, i.e., $m = \prod_{i=1}^t b_i$.

Then the positions of carries, $D(i, j)$ generated by the j -th cell located at the i -th stage of the carry generation network with block factor, b_i for $i = 1, 2, \dots, t$ can be determined by the following equation:

$$D(i, j) = j \prod_{r=1}^i b_r - 1 \quad (3.16)$$

$$\forall i = 1, 2, \dots, t \quad \text{and}$$

$$j = 1, 2, \dots, \left\lfloor \frac{N}{\prod_{r=1}^i b_r} \right\rfloor$$

For stages beyond t , the positions of the carries generated are given by the

following equation:

$$D(i, j) = \left(j + \prod_{r=t+1}^i b_r \right) \prod_{r=1}^t b_r - 1 \quad (3.17)$$

$$\forall i = t+1, t+2, \dots, l \quad \text{and}$$

$$j = 1, 2, \dots, \left\lfloor \frac{N - \prod_{r=1}^i b_r}{\prod_{r=1}^t b_r} \right\rfloor$$

In (3.16) and (3.17), j is the index of the lookahead cell enumerated from the right of the tree. The number of stages, l of the carry network is bounded by:

$$\lceil \log_{b_{\max}} N \rceil \leq l \leq \lceil \log_{b_{\min}} N \rceil \quad (3.18)$$

where $b_{\max}$ and $b_{\min}$ are the maximum and minimum block factors, respectively.

All cells in the carry generation network can be built using a single complex gate in static CMOS logic design style since CMOS is the most interesting implementation in terms of its trade-off between power and delay performances, and it offers high noise margins and robustness to device and voltage scalings [99]. However, it should be noted that in practice, complex CMOS gates are used for a maximum fan-in of 5 to 6 [100]. To avoid chaining many P-channel devices in series for low-voltage and deep submicron technology, we restrict the maximum number of MOSFET in series from V_{DD} to ground to no more than 6, i.e., $b_i \leq 3$ for all i to investigate the variants of mixed binary and ternary radix carry-lookahead tree using the prefix-notation [92, 96].

In prefix notation, the prefix operator, denoted by ‘ $\bullet$ ’ operates on pairs of binary generate and propagate signals. The initial generate and propagate pairs

are represented by (g_i, p_i) . From (3.3), the setup time of individual generate and propagate signals for prefix addition have been substantially reduced due to the exploitation of the redundancy of RB coding. The binary pair, $(G_{i:j}, P_{i:j})$ is used to denote the group generate and propagate terms produced from bits $i, i-1, \dots, j+1$ to j . From [90], we have:

$$\begin{aligned} (G_{i:j}, P_{i:j}) &= (g_i, p_i) \bullet (g_{i-1}, p_{i-1}) \bullet \dots \bullet (g_{j+1}, p_{j+1}) \bullet (g_j, p_j) \quad \text{for } i > j \\ (g_i, p_i) \bullet (g_{i-1}, p_{i-1}) &= (g_i + p_i g_{i-1}, p_i p_{i-1}) \quad (b = 2) \\ (g_i, p_i) \bullet (g_{i-1}, p_{i-1}) \bullet (g_{i-2}, p_{i-2}) &= (g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2}, p_i p_{i-1} p_{i-2}) \quad (b = 3) \end{aligned} \quad (3.19)$$

As prefix operator is idempotent, $(G_{i:j}, P_{i:j})$ can be derived by the association of two overlapping terms as follows [90]:

$$(G_{i:j}, P_{i:j}) = (G_{i:n}, P_{j:n}) \bullet (G_{m:j}, P_{m:j}) \quad \text{for } i > m \geq n > j \quad (3.20)$$

Figure 3.1 illustrates the possible implementations of an 18-bit parallel-prefix carry computation with the uniform block factor (fixed radix) of 2 or 3, and non-uniform block factor (mixed radix) of 2 and 3. In this figure, the solid node, $\bullet$, represents the prefix operator and the white hollow node, $\circ$, represents flow-through node or buffer. Figures 3.1(a)-(c) depict several carry generation schemes for different block lengths of CSL, m of 2-, 4- and 8-bits using a fixed block factor of $b = 2$. Similarly, Figures 3.1(d) and 3.1(e) show two different schemes with $m = 3$ and 9 using a fixed block factor of 3. Various combinations of m and b for mixed radix schemes are illustrated in Figures 3.1(f)-(l). It is worth noted that in the mixed radix schemes, the block factor is uniform within the same stage but non-uniform across stages. Completely mixed-radix

Figure 3.1: 18-bit parallel-prefix carry generation with various block factors and block lengths of CSL.

carry generation trees are analytically obscure to unify with the CSL sections for hybrid CLA/CSL due to the immense number of complex formations that could be enumerated. The use of mixed radix cells within the same stage can easily annihilate the regularity and unnecessarily complicate circuit and layout optimizations. Therefore, it will only be considered on a sparse number of leave cells after the optimal regular mixed radix tree has been established and provided its use will reduce the depth of the tree with minimal impediment on structural regularity.

The speed of hybrid CLA/CSL architecture is minimized when the critical delay of CLA network is commensurate with that of the CSL sections. Therefore, for speed optimization, the block factor for the CLA complex gates at each stage of the hierarchy can be optimized to tailor to the optimal block length of CSL based on the implementations of the RCA and multiplexer logics. The RB coding has been beneficially exploited in the RCA for the CSL circuit. Transistor-level circuit design techniques have been applied to devise a new area and power efficient add-one circuit for the carry-select adder. The details of the novel CSL circuit for the hybrid CLA/CSL reverse converter is best demonstrated with a 64-bit RB-to-NB converter to be presented in the next section. The critical paths along the CLA network and the CSL section can be evaluated with a good technology-independent LE model detailed in Section 2.4.1 of Chapter 2. The LE model can account for the delays due to the circuit topology and the loading at transistor-level implementation independent of the process factor. This helps to determine an optimized mixed-radix CLA structure for a given operand length. That will be carried out in Section 3.5.

3.4 Implementation of A 64-bit Reverse Converter

3.4.1 The Architecture of 64-bit Reverse Converter

In this section, a 64-bit RB-to-NB reverse converter is implemented with our proposed CLA/CSL based architecture. In principle, the block length of CSL can take any discrete value for any combination of the product of b_1 through b_t at the transitive stage. In our implementation, the fan-in and fan-out of every circuit node have been limited to no more than 3 to avoid chaining more than 3 P-channel/N-channel devices in series. Therefore, the length of CSL section will be a power of 2 or 3 or a multiple of 6 since the value of b_i is confined to 2 or 3. We have selected an optimal block length of CSL, $m = 6$ to match a mixed radix carry generation network from among the plausible topologies presented in Section 3.3 based on an objective evaluation through a LE model. The model of evaluation and the results will be discussed in details in the next section.

For CSL sections of constant block length, $m = 6$, the implementation of a 64-bit reverse converter is not unique as the depth of the carry generation tree can be either 5 or 6 according to the block factors selected for each stage. An efficient implementation of a 64-bit reverse converter architecture with $m = 6$ is shown in the block diagram of Figure 3.2. The numerical indices, i and j , are used to identify the stage numbers and the positions of the CLA cells at each stage, respectively.

From Figure 3.2, the regular carry generation network of the hybrid CLA/CSL-based 64-bit reverse converter is optimized with $b_1 = 3, b_2 = b_3 = b_4 = b_5 = b_6 = 2$. For $m = 6$, the transitive stage occurs at stage 2 but b_1 instead of b_2

3.4 Implementation of A 64-bit Reverse Converter

67

Figure 3.2: Block diagram of the 64-bit reverse converter.

is selected to be of radix-3 to take advantage of the simplification of the carry equation (3.14) with direct RB input in the first stage. From (3.16) and (3.17), the ten carry outputs generated by the carry-lookahead network are, C_5 , C_{11} , C_{17} , C_{23} , C_{29} , C_{35} , C_{41} , C_{47} , C_{53} and C_{59} , each of which is to be fed into one of the ten CSL sections excluding the first section. The first section has a constant input, which can be implemented directly as a modified RCA without the add-one circuit. For the end section, a 4-bit CSL suffices since the total number of sum bits is 64. From the topological diagram, it is observed that the two cells, H_{53} and H_{59} that generate the carries, C_{53} and C_{59} at the last stage can be ganged with their preceding cells in Stage 5 to save one stage. The two radix-3 prefix operators so formed have the additional inputs stemmed from the first two buffer nodes at Stage 4. Therefore, none of the operators has a fan-out that exceeds three, and there is no change in the critical path except a slight perturbation to the wiring. Figure 3.3 shows the block diagram of the final 5-stage reverse converter.

Let $P_j^{(i)}$ and $G_j^{(i)}$ represent the carry propagate and generate signals pro-

Figure 3.3: Block diagram of the modified 64-bit reverse converter.

duced by the j -th prefix operator at the i -th stage. Then, using the expansion of (3.14), the mixed radix lookahead cells of the CLA network are constructed with the following logical expressions after appropriate re-substitution of terms:

$$\begin{aligned}
 H_5 &= G_2^{(1)} + P_2^{(1)} \cdot G_1^{(1)} \\
 H_{11} &= G_2^{(2)} + P_2^{(2)} \cdot G_1^{(2)} \\
 H_{17} &= G_2^{(3)} + P_2^{(3)} \cdot G_1^{(2)} \\
 H_{23} &= G_3^{(3)} + P_3^{(3)} \cdot G_1^{(3)} \\
 H_{29} &= G_3^{(4)} + P_3^{(4)} \cdot G_1^{(2)} \\
 H_{35} &= G_4^{(4)} + P_4^{(4)} \cdot G_1^{(3)} \\
 H_{41} &= G_5^{(4)} + P_5^{(4)} \cdot G_1^{(4)} \\
 H_{47} &= G_6^{(4)} + P_6^{(4)} \cdot G_2^{(4)} \\
 H_{53} &= G_5^{(5)} + P_5^{(5)} \cdot G_1^{(2)} \\
 &= G_7^{(4)} + P_7^{(4)} \cdot G_3^{(4)} + P_7^{(4)} \cdot P_3^{(4)} \cdot G_1^{(2)} \\
 H_{59} &= G_6^{(5)} + P_6^{(5)} \cdot G_1^{(3)} \\
 &= G_8^{(4)} + P_8^{(4)} \cdot G_4^{(4)} + P_8^{(4)} \cdot P_4^{(4)} \cdot G_1^{(3)}
 \end{aligned} \tag{3.21}$$

3.4 Implementation of A 64-bit Reverse Converter

69

Figure 3.4: Circuit implementation of G and P cells in the 5-stage CLA network.

Only seven different types of cells are required for the carry generation network. These schematics are shown in Figure 3.4. In Figure 3.4, type A and B cells are used to generate the G signals in the first stage and P signals for all radix-3 prefix operators, respectively. Type C and D cells are NAND and NOR gate used to generate the P signals for all radix-2 prefix operators. Type E and F cells are complex gates used to generate the G signals for all radix-2 prefix operators. Type G cell is a complex gate used merely to generate H_{53} and H_{59} at Stage 5. It should be noted that all the P and G outputs alternate in polarities in the odd and even stages to unify the cells used. This unification not only simplifies and modularizes the circuit of the carry network but also

reduces its delay.

3.4.2 Design Considerations: Modified Add-One CSL Scheme

Among the myriad of arithmetic circuit design techniques, CSL has emerged as an eminent approach to address the area-time trade-off of CPA design. It exhibits the advantage of logarithmic gate depth as in any structure of the distant-carry adder family. When it is used together with CLA in the proposed RB-to-NB reverse converter, higher speed can be achieved at the expense of increased hardware cost. Our approach to hybrid CLA/CSL design differs from others [17, 89, 98] in that we combine the logic structure with circuit technique to minimize the number of transistors used in CSL section without degrading its performance. Besides, we have fully tapped on the redundancy of RB coding to halve the logics in the mandatory copy of the RCA of each CSL section, which further speeds up its sum and carry generation.

From (3.4), two separate carry chains with block carry-in of 0 and 1 can be derived for the CSL network. The benefit of having two carry chains is the saving of certain hardware resources. However, it also has the potential to increase the overall latency of the hybrid CLA/CSL circuit. A possible solution to improve the speed is to use two copies of RCA blocks and select the correct sum value according to the final block carry-in signals. Although it improves the speed significantly, the amount of transistors used is also sizable. To circumvent this problem, an add-one circuit was proposed by Chang *et al.* in [101]. As opposed to using dual RCAs in the conventional CSL, the architecture of contemporary CSL adder comprises a single RCA, a first zero detection and

selective complement add-one circuit, and a carry-select multiplexer circuit as shown in Figure 3.5 [101]. It achieves a 29.2% area reduction at the expense of 5.9% speed penalty for a 64-bit CSL over the conventional dual RCA design. The circuit was further modified by Kim *et al.* [102] to achieve even better performance. Unfortunately, an omission of a multiplexer in the MSB position of the add-one block was found in the design depicted in the circuit architecture schematic of [102].

Figure 3.5: CSL with a single RCA and an add-one circuit [101].

We improve the add-one CSL circuit to further minimize the transistor count without speed penalty at all. Since the add-one circuit is in essence, based on a first zero detection logic, it generates Z^1 by inverting each bit in Z^0 starting from the LSB until the first zero is encountered, where Z^0 and Z^1 are the sum outputs of the two copies of RCA with block carry-in 0 and 1, respectively. Figure 3.6 depicts the proposed add-one circuit using buffers with only one inverter and we have proved that the add-one circuit with single inverter

buffers performs exactly the same function as that shown in Figure 3.5 [93].

Figure 3.6: Modified add-one scheme.

In Figure 3.5, the complements of the sum bits are generated from the internal nodes of PMOS-NMOS chain. Before the first zero bit is detected, each PMOS and NMOS pair functions as an inverter. Once the first zero bit has occurred, the PMOS and NMOS pair acts as a multiplexer, which selects either V_{DD} or Z_{i-1}^* as described by:

$$Z_i^* = Z_i^0 \cdot Z_{i-1}^* + \overline{Z_i^0} \quad (3.22)$$

To select the correct sum,

$$Z_i = Z_i^1 \cdot C_{in} + Z_i^0 \cdot \overline{C_{in}} = (Z_i^0 \odot Z_{i-1}^*) \cdot C_{in} + Z_i^0 \cdot \overline{C_{in}} \quad (3.23)$$

In our proposed add-one circuit of Figure 3.6, there is no change in the output logic before the buffer is inserted. From (3.22) and (3.23), we have:

$$\begin{aligned} \overline{Z_i^*} &= \overline{Z_i^0 \cdot Z_{i-1}^* + \overline{Z_i^0}} = \overline{Z_i^0} \cdot \overline{Z_{i-1}^*} \cdot Z_i^0 \\ Z_x &= \overline{\overline{Z_i^0} \cdot \overline{Z_{i-1}^*} + \overline{Z_i^0} \cdot 0} = Z_i^0 \cdot \overline{Z_{i-1}^*} = \overline{Z_i^*} \end{aligned} \quad (3.24)$$

Similarly,

$$\begin{aligned} Z_y &= Z_{i+1}^0 \cdot Z_x = Z_{i+1}^0 \cdot \overline{Z_i^*} = \overline{Z_{i+1}^*} \\ Z_p &= Z_{i+2}^0 \cdot \overline{Z_y} + \overline{Z_{i+2}^0} = Z_{i+2}^0 \cdot Z_{i+1}^* + \overline{Z_{i+2}^0} = Z_{i+2}^* \end{aligned} \quad (3.25)$$

This verifies that our proposed add-one circuit is functionally equivalent to the circuit of Figure 3.5, but the total number of inverters has been reduced by half. There is no speed penalty since the block carry-in signals are generated in parallel by the CLA network rather than locally generated from the CSL sections sequentially. In fact, it is envisioned that with the shorter chain and potential reduction of internal signal toggling, power dissipation will be lowered. The internal carry chain of RCA can also be shortened by leveraging on the special property of RB numbers from (3.3). This is elaborated as follows.

From the original input f_i^+ and f_i^- of the RB number to be converted, the internal carry signal of an RCA can be simplified as follows:

$$\begin{aligned} c_0^0 &= f_0^+ \\ \overline{c_1^0} &= \overline{f_1^+ + f_1^- \cdot c_0^0} = \overline{f_1^-} \cdot \overline{(f_1^+ + c_0^0)} \\ c_2^0 &= \overline{f_2^- + f_2^+ \cdot \overline{c_1^0}} = \overline{f_2^+} \cdot \overline{(f_2^- + \overline{c_1^0})} \\ &\vdots \end{aligned} \quad (3.26)$$

Therefore, the carry generation chain in the proposed RCA can be readily implemented in branch-based logic style to minimize the number of internal connections [98]. Branch-based circuits possess high noise margins and robustness to voltage and device scaling reminiscent of classical static CMOS

design style. From (3.3), by exploiting the RB coding, both the sum bits and its complement can be generated simultaneously using the new XOR/XNOR circuit [103] proposed by our research group to enhance their driving capability. Comparing with the full adder of Figure 3.5, which requires two XOR gates for the sum generation, the propagation delay has been reduced. With the co-generation of complementary sum bits, the threshold voltage (V_{th}) drop problem of pass transistors [104] in the add-one circuit of Figure 3.5 can also be overcome by replacing them with TGs.

Figure 3.7: 6-bit CSL section with modified add-one scheme.

The modified 6-bit ripple-carry chain and the new add-one circuit are integrated into a 6-bit CSL as shown in Figure 3.7. Identical circuit topology for the odd and even carry generation cells are used to implement the carry signals with alternating polarities of inputs and outputs in the modified ripple-carry chain. At the bottom, the final sum of CSL can be generated from the add-one circuit by using only a group of NAND gates and multiplexers. It realizes the following logic equation derived from (3.23).

$$\begin{aligned} Z_i &= Z_i^0 \cdot Z_{i-1}^* \cdot C_{in} + \overline{Z_i^0} \cdot \overline{Z_{i-1}^*} \cdot C_{in} + Z_i^0 \cdot \overline{C_{in}} \\ &= Z_i^0 \cdot (\overline{Z_{i-1}^*} \cdot C_{in}) + \overline{Z_i^0} \cdot (Z_{i-1}^* \cdot C_{in}) \end{aligned} \quad (3.27)$$

The output, Z_i is selected from the two data input signals, Z_i^0 and its complement. The select signal $\overline{Z_{i-1}^*} \cdot C_{in}$ is generated by an NAND gate from the carry-in, C_{in} and the complement of Z_{i-1}^* . Thus, we can eliminate one inverter in each buffer from the corresponding block of Figure 3.5 without violating its functionality. The NAND gates also function as buffers to improve the driving capability.

3.5 Performance Evaluation

This section evaluates the performance of our newly proposed converter architecture and compares it with three competitive converters [34, 63, 64]. According to Section 3.2, a fixed size operand can have several ramifications of architecture depending on the block factors and block lengths of the CSL sections chosen for the implementation. Therefore, we are interested to find an optimum realization for a given operand size from among the feasible solu-

tions of our base converter architecture.

As introduced in Section 2.4.1, LE method provides a fast and consistent means to evaluate the potential performance of a digital circuit [76]. With the LE model, we analyze the critical path delays of our proposed reverse converter for operand lengths of 8, 16, 32, 64 and 128 with various block factors b of CLA and block lengths m of CSL. The results are shown in Table 3.1. The delay time in Table 3.1 is normalized and expressed in terms of the delay of FO4 inverter. For certain operand length, there exist more than one implementations correspond to some block factor and block length of CSL. It should be noted, however, that only the fastest solution is presented in Table 3.1. Since the transistor count of a circuit has an indirect correlation to the VLSI area, the number of transistors in each circuit is accounted and summarized in Table 3.2. The Area-Delay Products (ADP) are also provided in Table 3.3, where the area is measured in terms of transistor count and the ADP values are normalized by the ADP value of the case of $b \in \{2, 3\}$ and $m = 6$, and the obtained ratios are expressed in percentage. From these results, the most optimum reverse converter with the minimum ADP for each operand length is selected to compare against other fast reverse converters.

For a fair comparison, the contender circuits, CONV1 [34], CONV2 [63] and CONV3 [64] were replicated as reported in the literature. For high speed operation, CONV1 is optimized using CSL with minimized critical path as indicated in [34]. CONV2 is a simplified CLA, which uses inverters instead of complex CMOS circuits to produce the “generate signal G ” and the “propagate signal P ” according to the consideration presented in [63]. CONV3 uses series TGs for the carry propagation circuit. Hence buffers are inserted in every two TGs to prevent the signal from decaying.

Table 3.1: Comparison of Delay for Different Combinations of Block Factors of CLA and Block Lengths of CSL

Word Length	Delay (FO4)										
	$b=2$			$b=3$		$b \in \{2,3\}$					
	$m=2$	$m=4$	$m=8$	$m=3$	$m=9$	$m=2$	$m=3$	$m=4$	$m=6$	$m=8$	$m=9$
8	4.60	4.17	—	4.58	—	4.64	4.58	4.17	6.11	—	—
16	6.35	6.20	8.23	6.48	8.57	6.49	5.57	5.59	6.89	8.23	8.57
32	7.51	7.46	8.23	7.44	9.24	7.71	7.25	7.07	7.09	8.23	9.24
64	8.63	8.59	8.51	8.87	9.92	8.92	8.66	8.43	8.34	8.48	9.46
128	9.83	9.79	9.66	10.83	10.32	10.45	9.84	9.70	9.62	9.64	9.81

Table 3.2: Comparison of Transistor Count for Different Combinations of Block Factors of CLA and Block Lengths of CSL

Word Length	Number of Transistors										
	$b=2$			$b=3$		$b \in \{2,3\}$					
	$m=2$	$m=4$	$m=8$	$m=3$	$m=9$	$m=2$	$m=3$	$m=4$	$m=6$	$m=8$	$m=9$
8	180	178	—	149	—	178	149	178	154	—	—
16	468	476	416	496	400	448	500	466	456	416	400
32	1142	1126	1070	1121	1064	1138	1216	1108	1120	1062	1064
64	2664	2520	2502	2594	2274	2732	2579	2472	2486	2480	2478
128	6056	5482	5270	5539	4972	6503	5658	5238	5140	5094	5196

The delays of different reverse converters based on the proven technology-independent LE model for various operand lengths are tabulated in Table 3.4. The number of transistors required by each converter is shown in Table 3.5. The ADP are also provided in Table 3.6 to compare the combined criterion of cost and performance of different reverse converters. In Table 3.6, the ADP cost of CONV3 is used as a reference to normalize the ADP cost of all other converters.

Table 3.3: Comparison of Area-Delay Product for Different Combinations of Block Factors of CLA and Block Lengths of CSL

Word Length	Area-Delay Product (%)										
	$b=2$			$b=3$		$b \in \{2,3\}$					
	$m=2$	$m=4$	$m=8$	$m=3$	$m=9$	$m=2$	$m=3$	$m=4$	$m=6$	$m=8$	$m=9$
8	88.0	78.9	—	72.5	—	87.8	72.5	78.9	100	—	—
16	94.6	93.9	109.0	102.3	109.1	92.5	88.6	82.9	100	109.0	109.1
32	108.0	105.8	110.9	105.0	123.8	110.5	111.0	98.7	100	110.1	123.8
64	110.9	104.4	102.7	111.0	108.8	117.5	107.7	100.5	100	101.4	113.1
128	120.4	108.5	103.0	121.3	103.8	137.4	112.6	102.8	100	99.3	103.1

Table 3.4: Comparison of Delay for Different Converters

Word Length	Delay (FO4)			
	This Work	CONV1 [34]	CONV2 [63]	CONV3 [64]
8	4.17	6.37	6.21	10.00
16	5.59	8.21	8.58	10.95
32	7.07	10.00	10.26	14.74
64	8.34	12.74	13.58	15.47
128	9.64	15.93	17.05	19.71

From the above tables, it is evident that our newly proposed reverse converter has the minimum ADP for any operand length in comparison with other converters. More importantly, the delay of our proposed converter does not escalate with the increase of operand length as badly as other converters. As the word length increases, the improvement in the combined area-time performance becomes more prominent.

To validate and reinforce the results estimated by the LE model, two 64-

Table 3.5: Comparison of Transistor Count for Different Converters

Word Length	Number of Transistors			
	This Work	CONV1 [34]	CONV2 [63]	CONV3 [64]
8	178	264	228	262
16	466	536	564	604
32	1108	1048	1268	1312
64	2486	1840	2676	2936
128	5094	3906	5874	6430

Table 3.6: Comparison of Area-Delay Product for Different Converters

Word Length	Area-Delay Product (%)			
	This Work	CONV1 [34]	CONV2 [63]	CONV3 [64]
8	28.3	64.2	54.0	100
16	39.4	66.5	73.2	100
32	40.5	54.2	67.2	100
64	45.6	51.6	80.0	100
128	38.7	49.1	79.0	100

bit reverse converters have been implemented. One is our proposed converter and the other is CONV1 since it is the most competitive one according to its performance evaluated earlier in Tables 3.4 – 3.6. Besides the worst-case delay, the converters are also simulated for their average power consumptions. A simulation environment realistic to the actual circuit operation conditions, where the cell has both driving and driven circuit, has been set up as discussed in Section 2.4.2. All the 128 bit inputs are loaded from the input buffers before they are fed into the 64-bit converter circuits and the 64 bit outputs are also

Table 3.7: Comparisons of 64-bit Reverse Converters

64-bit Reverse Converter	Delay (ps)	Power (mW)	PDP (pJ)
This Work	636	0.38	0.242
CONV1 [34]	979	0.64	0.627

loaded to the buffers after they are exported. For a fair comparison, each circuit is optimized in speed first for the critical path estimated from the architecture. The optimization process for PDP outlined in Section 2.4.2 is then carried out recursively for the whole converter circuit until all transistor sizes converged [104]. All the circuits are simulated using HSPICE [81] based on the TSMC 0.18- μm CMOS process model. For each simulation, HSPICE generates an average power consumption value. As the dynamic power dissipation increases linearly with frequency and quadratic with supply voltage, both circuits are simulated at the same data rate of 100MHz and the same supply voltage of 1.8V with 4096 randomly generated input data. Comparison of these two converters in terms of the worst-case delay, average power dissipation and their product are listed in Table 3.7.

From Table 3.7, our proposed 64-bit reverse converter outperforms CONV1. It runs 1.5 times faster than CONV1 and consumes 40% less power. Furthermore, this simulation result is highly correlated to the relative performance difference between our converter and CONV1 in Table 3.1 for the 64-bit word length. The gate delay of an FO4 inverter for TSMC 0.18- μm CMOS process technology at 1.8V is simulated to be 70ps. Therefore, the deviation between HSPICE pre-layout simulation and LE estimation is less than 10%. This validates the legitimacy of the rapid performance evaluation based on the LE

Figure 3.8: Full-custom layout of proposed 64-bit reverse converter.

model.

A full-custom layout of the proposed 64-bit reverse converter circuit was carried out using the TSMC 0.18- μm CMOS process, which features six metal and one poly layers. The layout pattern of the converter is shown in Figure 3.8 and the post-layout simulation results are summarized in Table 3.8. Table 3.8 presents a more accurate delay and power consumption evaluation of our proposed converter as the parasitics attributed to wires have been back annotated for the post-layout simulation.

Table 3.8: Post-Layout Figure-of-Merit of Proposed 64-bit Reverse Converter

Area (mm ²)		0.08		
Supply Voltage (V)		3.3	1.8	1.1
Delay Time (ps)		598	829	1618
Average Power	50MHz	0.79	0.239	0.089
	100MHz	1.67	0.492	0.187
Dissipation (mW)	500MHz	8.85	2.61	0.959
	1GHz	19.3	5.84	—

3.6 Summary

Despite carry-free addition can be achieved for RB multiplier in the partial product accumulation process, it has been well-acknowledged that absolutely carry-free RB multiplier is impossible in practice. The performance bottleneck lies in the ineluctable carry propagation in the redundant number to NB number conversion process. In this chapter, we have shown that the inherent redundancy of RB coding can be fully exploited to simplify and speed up the reverse conversion through an elegant amalgamation of mixed-radix carry-lookahead network and novel carry-select adder. A hybrid CLA/CSL adder realization is well suited to the proposed formulation of reverse conversion problem. The carries of the CLA network are selected to equalize the critical path of the optimally designed CSL sections for a given operand length. The carry generation network is implemented with several heterogeneous CMOS basic cells, and the CSL section is simplified without jeopardizing the critical path delay by making use of the group carry-in signals generated by the multi-

level CLA network. To further reduce the cost of implementing the carry-select adder, the ripple-carry adder chain is modified and incorporated with a new add-one circuit. We have shown by means of LE technique that our proposed reverse converter outperforms three other competitive converters in terms of latency, transistor count and their ADP for operand lengths vary from 8 bits to 128 bits. The speed improvement over other converters is more prominent with increased operand length. The HSPICE simulation results of 64-bit transistor-level implementations of our proposed converter and the best contender obtained from the LE model proved the superiority of our proposed converter.

Chapter 4

RB Multiplier with New Covalent Redundant Binary Booth Encoding

4.1 Introduction

Besides the back end circuit of RB-to-NB converter discussed in Chapter 3, the front end design plays a very crucial role in the performance and cost of the RB multiplier as well. The design of Booth encoder and RB Partial Product Generator (PPG) influences the efficiency of the RB partial product generation. The number of RBPPs that can be saved by this stage impacts the cost, performance and power consumption of the RB summing tree and the multiplier as a whole. Although the number of partial products can be reduced with high-radix Booth encoder, the number of hard multiples that are expensive to generate also increases simultaneously [7]. In conventional RB multiplier design, modified Booth encoding algorithm in an NB regime is employed to reduce the number of partial products and then pairs of NB partial products

are encoded to form RBPPs. In this process, an additional constant binary vector is introduced to compensate for the aggregate errors resulting from both the RB coding and Booth encoding [13, 34, 60]. This correction vector incurs hardware overhead in the RB summing tree and, to a certain extent, offsets the regular structure of RB summing tree and increases its switching activities. To overcome the hard multiple and correction vector problems, an RB Booth encoder was proposed in [35, 36]. This chapter introduces yet another RB Booth encoder. Its unique RBPP generation method produces a more efficient RB multiplier architecture than that developed from RBBE.

As 8-, 16-, 32- and 64-bit operands are pervasively used in application-specific data paths [37, 38, 72, 74] and thousands of general purpose programs running in all architectures of computers [71, 73], this chapter focuses on power-of-two word length RB multipliers to exploit the binary logarithmic partial product reduction rate of the RBA summing tree. By scrutinizing the overheads of existing Booth encoding algorithms, a new CRBBE is proposed [15]. Our proposed method overcomes the hard multiple generation problem of NB Booth encoders without incurring any correction vector. Compared to the RB Booth encoder in [36], CRBBE generates the RBPPs more efficiently by consuming two RB digits for every RBPP it generated. Consequently, our encoder and decoder are less complex for the same radix. Since many emerging digital signal processing and multi-media applications that require fast digital multiplications are now migrated into portable devices, energy dissipation is becoming a criterion as important as delay in the design of efficient digital multipliers. When both constraints of throughput and battery life need to be satisfied, the energy efficiency of the multiplier is improved with a lower energy-delay product [18, 19]. Therefore, in the experiment results, we demon-

strate that the proposed Booth encoder and decoder make energy-efficient RB multipliers for power-of-two operand lengths by effectively eliminating the hardware overhead of baseline Booth encoder.

The remainder of this chapter is organized as follows. The existing conventional NB and RB Booth encoding algorithms and their overheads are briefly described in Section 4.2. Section 4.3 presents the proposed CRBBE algorithm. This is followed by the circuit implementation of RB multiplier in Section 4.4 to elaborate the design concept. The performance analysis of the proposed RB multiplier and the comparisons with other contenders are presented in Section 4.5. Section 4.6 summarizes this chapter. A part of the work in Section 4.3 has been presented at the 2005 International Symposium on Circuits and Systems [15]. A large portion of the work presented in this chapter has been submitted for review as a regular paper in the *IEEE Transactions on Computers*.

4.2 Issues of Booth Encoding Algorithms for Redundant Binary Multiplication

In fast digital multiplier designs, modified Booth encoding algorithm is an efficient way to reduce the number of partial products by grouping consecutive multiplier bits to form signed multiples [49]. In this section, two major issues on using the modified Booth encoding algorithm for RB multiplication and some existing solutions are recalled.

4.2.1 Hard Multiple Problems Revisit

Normal Binary Booth Encoding (NBBE) refers to the application of modified Booth encoding [49] to NB number. As mentioned in Section 2.2.1, in radix- r Booth- k encoding ($r = 2^k$), as the radix number increases, the number of encoded Booth digits and hence the number of partial products are reduced to $1/k$. However, as the number of multiples increases with the radix to $2^k + 1$, the number of hard multiples also increases simultaneously [7, 105]. As indicated in radix-8 Booth encoding, the multiplier is partitioned into 4-bit groups with an overlapping borrow bit between two adjacent groups. Each group is encoded in parallel to generate a select signal from the set $\{\pm 4M, \pm 3M, \pm 2M, \pm M, 0\}$ according to Table 2.3. Here nM refers to the select signal for the partial product nX , where X is the multiplicand. The partial product, $3X$ is a hard multiple, which can only be obtained by adding X and $2X$. A CPA is needed to generate the partial product $3X$ from the multiplicand, X . The existence of hard multiples increases the latency of the multiplier as a whole because the generation of the partial products will not be accomplished until all these hard multiples are produced. Therefore, Booth encoding of radix-8 and above are rarely used because of the criticality of generating the hard multiples and the complexity of the decoding logic.

4.2.2 Negative Multiples and NB-to-RB Partial Products

Conversion

Since negation in two's complement arithmetic requires carry propagation addition, negative partial products are more efficiently generated by bit inver-

88 4.2 Issues of Booth Encoding Algorithms for Redundant Binary Multiplication

sion of the multiplicand followed by the insertion of a '1' at its LSB position. Therefore, one additional partial product row is generated in the partial product summing tree to complete the NB negation of partial products for negative multiples.

Furthermore, to accumulate the NB partial products in RBA tree, the NB partial products generated by NBBE must be converted to RBPPs. An NB number can be encoded into RB representation using either sign-magnitude, positive-negative or positive-negative-complement codings. For convenience, this chapter adopts the positive-negative-complement coding to illustrate the issue but the discussion is also valid for the other codings.

In RB multiplication, the summation of two n -bit NB partial products, $A = (a_{n-1}a_{n-2} \dots a_0)_2$ and $B = (b_{n-1}b_{n-2} \dots b_0)_2$ can be combined into a single n -digit RB number, R , by:

$$R = A + B = A - (-B) \quad (4.1)$$

Since $-B = \overline{B} + 1$, substituting it into (4.1) gives:

$$\begin{aligned} R &= A - (\overline{B} + 1) = A - \overline{B} - 1 \\ &= \left(-2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i \right) - \left(-2^{n-1}\overline{b_{n-1}} + \sum_{i=0}^{n-2} 2^i \overline{b_i} \right) - 1 \\ &= -2^{n-1}(a_{n-1} - \overline{b_{n-1}}) + \sum_{i=0}^{n-2} 2^i (a_i - \overline{b_i}) - 1 \end{aligned} \quad (4.2)$$

According to Table 2.10, an RB number r can be encoded using positive-

4.2 Issues of Booth Encoding Algorithms for Redundant Binary Multiplication 89

negative-complement coding with r^+ and r^- , by:

$$r = (r^+, r^-) = r^+ - \overline{r^-} \quad (4.3)$$

where $r^+, r^- \in \{0, 1\}$, and $r \in \{0, 1, \bar{1}\}$.

Therefore, the term $(a_i - \bar{b}_i)$ in (4.2) can be encoded as $r_i = (a_i, b_i)$. To eliminate the hardware required for sign extension, the Most Significant Digit (MSD) term can be simply negated as $-(a_{n-1}, b_{n-1})$. From (4.3), it is noted that

$$-(r^+, r^-) = -r^+ + \overline{r^-} = (\overline{r^-}, \overline{r^+}) \quad (4.4)$$

Since the positive-negative-complement coding is symmetric, r^+ and r^- is commutative and $(\overline{r^-}, \overline{r^+}) \equiv (\overline{r^+}, \overline{r^-})$. Therefore, R can be coded as follows:

$$R = (\overline{a_{n-1}}, \overline{b_{n-1}})(a_{n-2}, b_{n-2}) \dots (a_1, b_1)(a_0, b_0) + (0, 0) \quad (4.5)$$

This method of generating an RBPP from two adjacent NB partial products is adopted in the RB multiplier of [13, 34]. From (4.5), it is clear that every RBPP row so composed requires one correction constant, $(0, 0) \equiv \bar{1}$, to be added by an RBA at its LSB position. All the correction constants generated from RBPP encoding, together with those constants from negative multiples, can be accumulated to form a new RBPP, called the RB correction vector.

Figure 4.1 exemplifies the procedure of correction vector generation. It can be seen that NBBE-2 (radix-4 NB Booth encoding) generates three instead of two RBPPs for an 8×8 -bit multiplication. Owing to the absence of hard multiple, NBBE-2 is attractive especially for short operand length multiplication.

Figure 4.1: Illustration of the correction vector generation on an 8×8 -bit multiplication with NBBE-2.

However, the additional delay required to add an extra partial product row critically slows down short operand length multiplier due to the relatively fewer number of adder stages in its partial product summing tree.

Therefore, the RB correction vector incurs additional hardware for its accumulation. It can even increase the number of stages of the summing tree, if the word length of the multiplier is 2^n , such as the 8-bit and 16-bit multipliers in application-specific data paths of multimedia and wireless applications [37, 74], and the multipliers for single extended and double extended floating point numbers, whose effective mantissa are 32 and 64 bits, respectively [73]. Consequently, the power dissipation and worst-case delay are also degraded by the inclusion of these correction vectors.

4.2.3 Redundant Binary Booth Encoding (RBBE)

In [36], a method was proposed to obtain the hard multiples from the differences of two simple power-of-two multiples. As introduced in Section 2.2.1, the partial products generated in this way conform to the format of positive-negative RB coding. The advantage of this method is the correction vector due to the NB arithmetic and RB coding has been completely eliminated. Compared to NBBE, the ease of generating the hard multiples by RBBE, to a certain extent has been offset by its complex circuitry. High-radix RBBE requires high fan-in gates in the PPG circuit (see Figure 4.2). Since the circuit for each digit of the RBPP will be duplicated in a large number, the overhead of high fan-in gates is more prominent in long operand length multiplier. Besides, as only one Booth encoded digit is consumed for one RBPP, half of the binary bits representing an RBPP generated from a simple power-of-two multiple in the RBBE circuit are filled with '0's, which is rather inefficient.

4.3 Covalent Redundant Binary Booth Encoding Algorithm

To overcome the shortcoming of existing Booth encoding algorithms, we propose a new Booth encoding algorithm to simplify the generation of hard multiples and reduce the number of RBPPs without introducing any form of correction vector that can aggravate the critical path of partial product accumulation tree. The proposed algorithm binds two adjacent modified Booth encoders to compose an RBPP by exploiting the encoding of RB number. The common bit of the two adjacent Booth encoders is used as an enabler for the polarization of

Figure 4.2: Radix-16 RBBE encoder and the partial product generator.

two equally weighted partial product bits. As the formation of an RBPP digit is analogous to the charge sharing of two oppositely charged atoms in a covalent bond, we name the proposed algorithm the Covalent Redundant Binary Booth Encoding (CRBBE).

4.3.1 Radix-4 Covalent Redundant Binary Booth Encoding (CRBBE-2)

Figure 4.3 shows the simplest radix-2 Booth encoded multiplier. From (2.1), the signed digit, $d_i = -y_i + y_{i-1}$ is encoded from $y_i(y_{i-1})$, where the borrow bit is in bracket. Since the borrow bit from which d_{i+1} is encoded is the MSB of the binary bits from which d_i is encoded, not all combinations of two digits

Figure 4.3: Radix-2 Booth encoded multiplier.

Table 4.1: Permissible Duplet (d_{i+1}, d_i) in Radix-2 Booth Encoded Number

$d_{i+1} = 1$	$d_{i+1} = 0$	$d_{i+1} = \bar{0}$	$d_{i+1} = \bar{1}$
$(1, \bar{0})$	$(0, 1)$	$(\bar{0}, \bar{0})$	$(\bar{1}, 1)$
$(1, \bar{1})$	$(0, 0)$	$(\bar{0}, \bar{1})$	$(\bar{1}, 0)$

from $\{0, \bar{0}, 1, \bar{1}\}$ are permissible for any pair of contiguous digits in an encoded number. The following properties are observed.

Property 1: No two consecutive non-zero digits are of the same sign, i.e., $d_{i+1} \times d_i = -1, i \in [0, N-2]$, where d_{i+1} and d_i are two adjacent non-zero digits and N is the word length of radix-2 Booth encoded number.

Property 1 implies that the signs of the nonzero digits alternate in the encoded multiplier.

Property 2: Any zero between a leading 1 and a trailing $\bar{1}$ is a negative zero, $\bar{0}$.

Table 4.1 shows all permissible combinations of two contiguous encoded digits d_{i+1} and d_i , which are grouped into four categories based on the left digit d_{i+1} .

From the analysis of Section 4.2, it is evident that if two adjacent NBBEs always generate signed digits of opposite polarities, their corresponding NB partial products can be directly combined to form a single positive-negative-

Table 4.2: Polarization of (d_{i+1}, d_i) for Radix-4 CRBBE

$d_{i+1} = 1$	$d_{i+1} = 0$	$d_{i+1} = \bar{0}$	$d_{i+1} = \bar{1}$
$(1, 0) = (1, \bar{0})$	$(0, 1) = (\bar{0}, 1)$	$(\bar{0}, \bar{0}) = (0, \bar{0})$	$(\bar{1}, 1) = (\bar{1}, 1)$
$(1, \bar{1}) = (1, \bar{1})$	$(0, 0) = (\bar{0}, 0)$	$(\bar{0}, \bar{1}) = (0, \bar{1})$	$(\bar{1}, 0) = (\bar{1}, 0)$

complement coded RBPP without any correction vector. This is only possible if contiguous digits of the Booth encoded multiplier alternate in signs. The duplets in the middle two columns of Table 4.1 obviously do not fulfill this criterion.

Since the signed digit representation of a number is not canonic and the neutral polarity zero can be expressed in both positive and negative forms, we can map all possible duplets in Table 4.1 to (p_l^+, p_l^-) , such that one digit of the pair is positive and the other digit is negative without changing the compound multiple coefficient, c_l .

$$c_l = 2d_{i+1} + d_i = 2p_l^+ + p_l^- \quad (4.6)$$

where $p_l^+, p_l^- \in \{\pm 0, \pm 1\}$ and $\text{sign}(p_l^+) \neq \text{sign}(p_l^-)$. $l = 0, 1, \dots, \lceil \frac{N}{2} \rceil - 1$ and $i = 2l$.

The multiple $c_l X$ is an RBPP composed from the two adjacent NB partial products, $2d_{i+1}X$ and $d_i X$. For ease of exposition, the digit pair, (p_l^+, p_l^-) is called a dipole and the mapping $\theta : (d_{i+1}, d_i) \rightarrow (p_l^+, p_l^-)$ is called polarization. For example, in the second column of Table 4.1, $(d_{i+1}, d_i) = (0, 1)$ can be mapped to a dipole of either $(1, \bar{1})$ or $(\bar{0}, 1)$. Table 4.2 shows all the dipoles. The dipole allows an RBPP $c_l X$ to be composed from the difference of two multiples in the

PPG. Due to the symmetry, for every positive-negative dipole in the shaded column of Table 4.2, there is always a corresponding negative-positive dipole in the unshaded column with their coefficients, c_l of (4.6) differ only in sign. This property can be used to reduce the hardware for the CRBBE circuit so that only one selector logic of each distinct signed multiple magnitude needs to be generated. The positive-negative-complement encoded RBPP corresponding to the dipole, (p_l^+, p_l^-) is denoted by $(pp_{l,j}^+, \overline{pp_{l,j}^-})$.

$$(pp_{l,j}^+, \overline{pp_{l,j}^-}) = pp_{l,j}^+ - pp_{l,j}^- \equiv (2p_l^+ + p_l^-) \cdot X_j \quad (4.7)$$

where $pp_{l,j}^+, pp_{l,j}^- \in \{0, 1\}$, and the subscripts, l and j are the indices of the multiplier and the multiplicand bit, respectively.

A multiple in the unshaded column can be generated from its corresponding multiple in the shaded column by simply swapping the values of $pp_{l,j}^+$ and $pp_{l,j}^-$ without generating any correction vector.

Radix-4 CRBBE produces $\lceil \frac{N}{2} \rceil$ RBPPs without the correction vector problem of NBBE and yet the selector logic and RB PPG circuit is simpler than RBBE of the same radix. It is interesting to note that radix-8 CRBBE can be created from binding two heterogeneous Booth encoders. The encoded digits from a radix-2 and a radix-4 NBBE can be ‘polarized’ to avoid the generation of all the hard multiples of radix-8. With a simple tweak, CRBBE can be easily extended to radix-16 to achieve even higher RBPP reduction rate without having its critical path aggravated by any hard multiple. This will be illustrated in the next subsection.

4.3.2 Radix-16 Covalent Redundant Binary Booth Encoding (CRBBE-4)

From (2.1), two contiguous radix-4 NBBE encoded digits, d_{i+1} and d_i share a common bit, $y_{k(i+1)-1}$ from the multiplier and it exhibits the following property:

Property 3: if the LSB $y_{k(i+1)-1}$ that encodes d_{i+1} is 0, d_i is non-negative. Otherwise, if $y_{k(i+1)-1} = 1$, d_i is non-positive.

The above property is actually a generalization of Property 2. It indicates that, irrespective of the radix of Booth encoding, only restricted combinations of contiguous digit pairs from the set $\{\pm 0, \pm 1, \dots, \pm(r/2)\}$ are permissible in an encoded number.

With this restriction on the legitimacy of encoded digits, two contiguous digits, $d_{i+1}d_i$, of radix-4 Booth encoding can be mapped from three contiguous bits $y_{2i+3}y_{2i+2}y_{2i+1}$ and $y_{2i+1}y_{2i}y_{2i-1}$ of the multiplier as shown in Table 4.3, where $i = 0, 1, \dots, \lceil \frac{N}{2} \rceil - 1$. In Table 4.3, all possible duplets are mapped to the dipoles, (p_l^+, p_l^-) for $l = 0, 1, \dots, \lceil \frac{N}{4} \rceil - 1$ such that

$$c_l = 4d_{i+1} + d_i = 4p_l^+ + p_l^- \quad (4.8)$$

where the multiple $c_l X$ is an RBPP composed from the two adjacent NB partial products, $4d_{i+1}X$ and d_iX , and $i = 2l$.

The positive-negative dipoles are listed in the shaded columns of Table 4.3 while their negative-positive counterparts appear in the unshaded columns. The only exception is when $(d_{i+1}, d_i) = (1, 1)$ and $(\bar{1}, \bar{1})$. These two cases cor-

Table 4.3: Polarization of (d_{i+1}, d_i) for radix-16 CRBBE

$d_{i+1} = 2$	$d_{i+1} = 1$	$d_{i+1} = 0$	$d_{i+1} = \bar{0}$	$d_{i+1} = \bar{1}$	$d_{i+1} = \bar{2}$
$(2, \bar{0}) = (2, \bar{0})$	$(1, 2) = (2, \bar{2})$	$(0, 2) = (\bar{0}, 2)$	$(\bar{0}, \bar{0}) = (0, 0)$	$(\bar{1}, 2) = (\bar{1}, 2)$	$(\bar{2}, 2) = (\bar{2}, 2)$
$(2, \bar{1}) = (2, \bar{1})$	$(1, 1)^*$	$(0, 1) = (\bar{0}, 1)$	$(\bar{0}, \bar{1}) = (0, \bar{1})$	$(\bar{1}, 1) = (\bar{1}, 1)$	$(\bar{2}, 1) = (\bar{2}, 1)$
$(2, \bar{2}) = (2, \bar{2})$	$(1, 0) = (1, \bar{0})$	$(0, 0) = (\bar{0}, 0)$	$(\bar{0}, \bar{2}) = (0, \bar{2})$	$(\bar{1}, 0) = (\bar{1}, 0)$	$(\bar{2}, 0) = (\bar{2}, 0)$
	$(1, \bar{0}) = (1, \bar{0})$			$(\bar{1}, \bar{0}) = (\bar{1}, \bar{0})$	
	$(1, \bar{1}) = (1, \bar{1})$			$(\bar{1}, \bar{1})^*$	
	$(1, \bar{2}) = (1, \bar{2})$			$(\bar{1}, \bar{2}) = (\bar{2}, 2)$	

* Hard multiples.

respond to the special hard multiples, $\pm 5X$, which are marked with “*” in Table 4.3. This hard multiple can be generated using the dedicated carry-free RBA of [36]. It turns out that this RBA does not lie in the critical path of the CRBBE encoder. The RBPP, $(pp_{l,j}^+, pp_{l,j}^-)$ generated by the dipole (p_l^+, p_l^-) is expressed as follows:

$$(pp_{l,j}^+, \overline{pp_{l,j}^-}) = pp_{l,j}^+ - pp_{l,j}^- = (4p_l^+ + p_l^-) \cdot X_j \quad (4.9)$$

A detailed work-out example for a 16×16 -bit multiplication based on CRBBE-4 algorithm is shown in Figure 4.4. The generation of the hard multiple, $5X$, by an RBA is shown at the top of the figure. Except the hard multiple $(1, 1)$, the three dipoles, $(2, \bar{2})$, $(\bar{1}, 1)$ and $(\bar{0}, 1)$ are used to generate the RBPPs $6X$, $-3X$, and $1X$, respectively.

The RBPPs reduction rate of radix-16 CRBBE is $\frac{1}{4}$. Higher radix CRBBE algorithm can be similarly derived without introducing additional row of correction vector. Although most hard multiples for radix-32 can be more readily

$$\begin{array}{l}
 \boxed{5X = \begin{array}{l} 0011111101111100 \\ 0001101110111110 \end{array} = 00100100\bar{1}10000\bar{1}0} \\
 \text{Multiplicand : } 0000011100100110 = 1830_{10} \\
 \text{Multiplier : } \times \begin{array}{l} \textcircled{0110010011010001} \\ \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ (2, \bar{2}) \quad (1, 1) \quad (\bar{1}, 1) \quad (\bar{0}, 1) \quad (p_l^+, p_l^-) \end{array} = 25809_{10} \\
 \hline
 \begin{array}{r}
 0000011100100110 \} \dots 1X \\
 0000000000000000 \\
 0000011100100110 \} \dots -3X \\
 0001110010011000 \\
 0011111101111100 \} \dots 5X \\
 0001101110111110 \\
 + 0011100100110000 \} \dots 6X \\
 0000111001001100 \\
 \hline
 0000011100100110 \\
 000\bar{1}\bar{1}011\bar{1}01\bar{1}\bar{1}110 \\
 00100100\bar{1}10000\bar{1}0 \\
 + 00110\bar{1}\bar{1}10\bar{1}11\bar{1}100 \\
 \hline
 000000110\bar{1}01001\bar{1}\bar{1}011\bar{1}11000000110 \\
 \downarrow \\
 00000010110100001010111000000110 = 47230470_{10}
 \end{array}
 \end{array}$$

Figure 4.4: 16×16-bit RB multiplication with CRBBE-4.

resolved than NBBE of the same radix, there exist some hard multiples which can not be generated as efficiently as the $5X$ multiple in this manner. Thus, CRBBE algorithm with $k \geq 5$ will not be pursued in this chapter.

4.4 Circuit Design of Redundant Binary Multiplier

This section presents the circuit design of CRBBE-4 and exemplifies its use in a 64×64-bit RB multiplier.

4.4.1 Circuit Design of CRBBE-4

There are 16 slices of CRBBE circuit for a 64×64 -bit RB multiplier. Figure 4.5 shows the l -th radix-16 CRBBE circuit for generating the control signal $c_l M_l$. It is composed of two adjacent radix-4 Booth encoders. Its gate-level implementation is shown in Figure 4.5(a), where the sign and magnitude of the radix-4 Booth encoded digit d_i are represented with three binary bits, sgn_i , $m_i^{(2)}$ and $m_i^{(1)}$ as follows:

$$d_i = (-1)^{sgn_i} (m_i^{(2)} + m_i^{(1)}) \quad (4.10)$$

The indices i and l are related by $i = 2l$. The lower encoder takes three consecutive bits $y_{2i+1}y_{2i}y_{2i-1} \equiv y_{4l+1}y_{4l}y_{4l-1}$ from the multiplier to generate the magnitude bits, $m_{2l}^{(2)}$ and $m_{2l}^{(1)}$ of d_i . Its sign bit, $sgn_i = y_{4l+1}$. The upper encoder takes the binary bits $y_{2i+3}y_{2i+2}y_{2i+1} \equiv y_{4l+3}y_{4l+2}y_{4l+1}$, and generates the magnitude bits $m_{2l+1}^{(2)}$ and $m_{2l+1}^{(1)}$ of d_{i+1} . Its sign bit, $sgn_{i+1} = y_{4l+3}$. All these output signals are mapped by the polarization circuit as shown in Figure 4.5(b). The control signals, $c_l M_l$ it generated are used to select the RBPP corresponds to the multiples, $c_l X$.

The polarization circuit performs the mapping, $\theta : (d_{i+1}, d_i) \rightarrow (p_l^+, p_l^-)$. The control signals, $1M_l$, $2M_l$, $4M_l$ and $8M_l$ are computed as follows:

$$1M_l = m_{2l}^{(1)} \cdot \overline{5M_l} \quad (4.11)$$

$$2M_l = m_{2l}^{(2)} \quad (4.12)$$

$$4M_l = \overline{(m_{2l+1}^{(1)} \cdot m_{2l}^{(2)} \cdot (sgn_{2l+1} \odot sgn_{2l}) \odot m_{2l+1}^{(1)})} \cdot \overline{5M_l} \quad (4.13)$$

$$8M_l = \overline{m_{2l+1}^{(1)} \cdot m_{2l}^{(2)} \cdot (sgn_{2l+1} \odot sgn_{2l}) \odot m_{2l+1}^{(2)}} \quad (4.14)$$

Figure 4.5: Circuit implementation of CRBBE-4 encoder.

The special $5M_l$ multiple is generated by:

$$5M_l = (sgn_{2l+1} \odot sgn_{2l}) \cdot m_{2l+1}^{(1)} \cdot m_{2l}^{(1)} \quad (4.15)$$

The control flag, $swap$ is used to exchange pp_l^+ and pp_l^- in the PPG to negate the selected RBPP. When d_{i+1} is zero, the sign bit of d_{i+1} is complemented before it is used as an active high swap flag to the RB PPG. Otherwise, the original sign of d_{i+1} is used as the swap flag. Therefore, the $swap$ signal can be generated by:

$$swap_l = \overline{(m_{i+1}^{(1)} + m_{i+1}^{(2)})} \oplus sgn_{i+1} = \overline{(m_{2l+1}^{(1)} + m_{2l+1}^{(2)})} \oplus sgn_{2l+1} \quad (4.16)$$

Figure 4.6 shows a slice of the RB PPG circuit for the generation of the j -th digit of the l -th RBPP, $pp_{l,j}^+$ and $pp_{l,j}^-$. Comparing with Figure 4.2, the RB PPG circuit of CRBBE-4 is less complex.

Figure 4.6: RB partial product generator of CRBBE-4.

4.4.2 CRBBE-4 Based RB Multiplier Architecture

Figure 4.7 shows the block diagram of a 64×64 -bit CRBBE multiplier, which consists of three stages, Booth encoder and RB PPG, RBA summing tree and RB-to-NB converter.

In the first stage, 16 slices of CRBBE encoders are used to generate the control signals from the multiplier. The $5X$ hard multiple is generated by the RBA and the multiplicand bits are shifted and selected into 16 rows of RBPPs in 16 slices of RB PPG.

In the second stage, a 4-stage RBA summing tree is used to sum 16 RBPPs. Only the multi-digit RBA blocks, annotated with the number of RB partial product digits input to each block, are shown in Figure 4.7. Each RBA block contains 64 RB Full Adder (RBFA) cells and a varying number of RB Half Adder (RBHA) cells depending on where they are located. The RBA block in

Figure 4.7: Block diagram of 64x64-bit RB multiplier architecture.

the i -th level, designated RBA_i ($i = 1$ to 4) contains 2^{i+1} RBHA cells in its MSD positions. The RBFA and RBHA cells modified from [13] are shown in Figure 4.8. According to (4.9), due to the positive-negative-complement coding, the second binary bit, $pp_{i,j}^-$ of the RBPP generated from CRBBE and RB PPG circuit should be inverted before it is input to the RBA. In [34], a preprocessing circuit is needed for each RB digit to avoid the inconsistent representations of '0' prior to the RBA summing tree stage. An important benefit of the coding format adopted in this design is that these preprocessing circuits can be completely eliminated due to its symmetry. The issues of anterior and poste-

Figure 4.8: Schematic of RB full and half adders.

rior interface converters of RBA summing tree for different RB coding schemes will be further elaborated in Chapter 5.

An RB-to-NB converter converts the final accumulation result to NB representation. Due to the unequal delay profile of the final RB result bits, the reverse conversion can be carried out in uneven groups of consecutive digits according to their arrival time. Groups of 4, 4, 8, 16 and 96 digits from the LSD position are evaluated concurrently. The first three groups of 4, 4 and 8 digits can be independently converted with ripple-carry adders to reduce the circuit complexity. The carry generation of next group of 16 digits can be evaluated with a carry-lookahead adder as they do not depend on the final summation results in the RBA summing tree stage. Therefore, the conversion speed of RB-to-NB stage depends solely on the conversion time of the most significant 96 digit group. This group is converted with a hybrid CLA/CSL as discussed in Chapter 3.

4.5 Simulation Results

This section evaluates the overall performance of the proposed radix-16 covalent redundant binary Booth encoding (CRBBE-4) multiplier. The results are compared with RB multipliers designed with radix-4, radix-8, radix-16 NB Booth Encoding (NBBE-2, NBBE-3, NBBE-4) [7], radix-8 Partially Redundant Biased Booth Encoding (PRBBE-3) [57], and radix-16 Redundant Binary Booth Encoding (RBBE-4) [36]. The Booth encoder and PPG stage in each contender multiplier is replicated as reported in literature. Meanwhile, the same RBA summing tree and RB-to-NB converter circuits are used for all multipliers.

Each design is described at gate level in VHDL. The functionalities of the algorithms are verified by ModelSim [106] for randomly generated input patterns. The designs are synthesized and mapped to Artisan TSMC 0.18- μm standard-cell library [82] using the Synopsys Design Compiler [83] with a nominal wire load model. The simulation environment is setup as described in Section 2.4.3, which is supplied with 1.8V at 25°C room temperature. Each design is optimized for speed to their minimum achievable delay. The average power consumptions are simulated by Synopsys Power Compiler [107] with back annotated switching activity files generated from random input vectors to each design. The Monte Carlo statistical model [85] (see Chapter 2, Section 2.4.3) is adopted to obtain the mean power dissipation of each design with more than 99.9% confidence level that the error is bounded below 3%. The energy per operation of each design is obtained by dividing the average power dissipation by the input rate of the test vectors, which is the maximum frequency that each individual multiplier is capable to function.

Table 4.4 summarizes the worst-case delay and energy dissipation of the

Table 4.4: Synthesis Results of Different Booth Encoded RB Multipliers

RB Multipliers	Delay (ns)				Energy Dissipation (pJ)			
	8b	16b	32b	64b	8b	16b	32b	64b
CRBBE-4	1.588	2.159	2.969	3.938	5.084	16.25	58.37	237.02
<i>RBBE-4</i>	1.712	2.396	3.246	4.295	5.646	19.07	68.03	259.87
<i>NBBE-2</i>	1.809	2.468	3.286	4.297	4.916	15.52	55.73	229.78
<i>NBBE-3</i>	2.143	2.743	3.511	4.672	5.480	16.54	53.51	205.19
<i>NBBE-4</i>	2.314	3.057	4.025	4.976	6.584	19.13	59.11	222.84
<i>PRBBE-3</i>	1.885	2.502	3.301	4.419	5.544	17.02	56.63	220.96

RB multipliers. The proposed CRBBE-4 multiplier is the fastest design for all power-of-two operand lengths. On average, it is 8.50%, 10.68%, 19.58%, 26.96%, and 12.60% faster than RBBE-4, NBBE-2, NBBE-3, NBBE-4 and PRBBE-3, respectively. Among the Booth multipliers, NBBE-2, RBBE-4 and CRBBE-4 have the same reduction rate of $\frac{1}{4}$ based on the number of RBPPs. Due to the correction vector, the speed of NBBE-2 multiplier is degraded by the additional stage in the partial product summation network. Compared to CRBBE-4, NBBE-3, PRBBE-3 and NBBE-4 are able to reduce more RBPPs, making the effect of correction vector negligible. However, these higher radix Booth multipliers suffer from a more severe hard multiple problem due to the inevitable carry propagation in generating the hard multiples.

The RB multiplier with NBBE-2 dissipates the least energy in 8-bit and 16-bit multiplications due to the absence of hard multiple and its simplest Booth encoder and PPG circuits. For larger operand length of 32 and 64 bits, NBBE-

3 consumes the least energy among all NBBE multipliers in view of a better trade-off between the complexity of the RBA summing tree and the number of CPAs required for the generation of hard multiples. For lower operand lengths, the energy dissipation of the proposed CRBBE-4 is very close to that of NBBE-2. Despite the complexity of Booth encoder and PPG is lower for NBBE-2, its RBAs in the summing tree outnumber that of CRBBE-4, which accounts for the reduced ascendancy in energy dissipation. This is because the number of RBA stages of NBBE-2 is comparatively larger than that of CRBBE-4 due to its extra correction vector. The complexity of hard multiple generation and increase in partial product compensation terms of NBBE-3, NBBE-4 and PRBBE-3 cause higher switching activities in the 8-bit and 16-bit multipliers. As the operand length increases to 64 bits, the energy consumption margin due to these overheads reduces and the switching activities become dominated by the complexity of the RBA summing tree. CRBBE-4 dissipates less energy than RBBE-4 for all word lengths. Both RBBE-4 and CRBBE-4 have no hard multiple and correction vector issues, but the PPG of CRBBE-4 is much simpler than that of RBBE-4. The better energy dissipation of CRBBE-4 over RBBE-4 is primarily due to its power reduction over a large number of PPGs.

For the same rate of partial product reduction, it is interesting to note that with small length adders and additional compensation vector, PRBBE-3 multiplier achieves higher speed than NBBE-3 with a penalty of more energy dissipation. Therefore, if both throughput and battery life need to be optimized simultaneously, the energy per operation has to be minimized in the same time as the delay. The EDP is a better metric than the energy per operation for benchmarking the energy efficiency of a circuit [18, 19]. This metric makes the evaluation less sensitive to the reduction of either energy or delay by simply

Table 4.5: Energy-Delay Product of RB Multipliers

RB Multipliers	EDP (fJ/MHz)			
	8b	16b	32b	64b
CRBBE-4	8.073	35.08	173.30	933.38
<i>RBBE-4</i>	9.666	45.69	220.83	1,116.14
<i>NBBE-2</i>	8.893	38.30	183.13	987.36
<i>NBBE-3</i>	11.744	45.37	187.87	958.65
<i>NBBE-4</i>	15.235	58.48	237.92	1,108.85
<i>PRBBE-3</i>	10.450	42.58	186.94	976.42

changing the supply voltage than optimizing circuit topology. The EDP of all multipliers being compared are tabulated in Table 4.5. For ease of comparison, the bar chart of normalized EDP is plotted in Figure 4.9, where the EDP for each operand length is normalized so that the multiplier with the largest EDP has an EDP of one. The results show that our proposed CRBBE-4 multiplier is most energy efficient. It exhibits at least 9.22%, 8.41%, 5.37%, 2.64% less EDP than the most competitive multipliers for word lengths of 8, 16, 32 and 64 bits, respectively. Among the radix-16 Booth encoded RB multipliers, the EDPs of our proposed CRBBE-4 multipliers are at least 16.48%, 23.22%, 21.52% and 15.82% lower for 8-, 16-, 32- and 64-bit operands, respectively.

Figure 4.9: Comparison of normalized EDP of different Booth encoded RB multipliers.

4.6 Summary

The use of RB arithmetic in the design of high-speed digital multiplier is beneficial due to its high modularity and carry-free addition. To reduce the number of partial products, high-radix modified Booth encoding algorithm is desired. However, its use is hampered by the complexity of generating the hard multiples and the overheads resulting from negative multiples and NB to RB number conversion. In this chapter, an energy-efficient RB multiplier based on a new covalent RB Booth encoding is presented. The idea is to polarize the two adjacent Booth encoded digits to directly convert a NB partial product to RBPP without incurring any correction vector. The proposed method fully exploits the characteristics of positive-negative-complement coding of RB number to directly generate an RBPP from two adjacent Booth encoded digits. Consequently, it shares the same advantages of RB Booth encoder for the ease of

generating hard multiples and avoidance of error correction vector, the two problems that are confronted by RB multiplier with NB Booth encoding. The synthesis results show that the RB multiplier based on CRBBE algorithm outperforms its rivals in terms of speed and energy efficiency for power-of-two operand lengths.

Some interesting phenomena are observed in the experimental results of Section 4.5. In the next chapter, further analysis will be carried out on many more different configurations of RB multipliers for several commonly used operand lengths, which include two operand lengths that are not power-of-two.

Chapter 5

Energy Efficiency Evaluation of Redundant Binary Booth Multipliers

5.1 Introduction

RB representation possesses some figures of merits as an internal format in emerging digital multiplier design due to its carry-free property and simplification on sign extension problem. Being a non classical representation of computer arithmetic, its worthiness of fulfilling the desirable VLSI goals of high performance, low power and small footprint in digital multiplier design has yet to be acclaimed. From the literature survey of Section 2.3, a number of RB multipliers proposed recently have found to be ambiguously constructed and the performances are of controversial veracity [13, 68, 69, 108]. We believe a structural approach such as that [109] used in analyzing the performance

of one-bit CMOS full adder cells could help to provide a good insight into the trade-off and limitation of RB arithmetic, and eradicate some myths of RB arithmetic on digital multiplier design.

From the literature survey of Chapter 2, the trajectory of an RB multiplier in the area-time space is a strong function of the ways the partial products are generated and how they are encoded. The aim of this chapter is to present a systematic analysis of many potential compositions of the fabrics that made up different RB multiplier circuits. The fabrics are characterized by the radix and type of Booth encoders and decoders, as well as the coding format used for the RBPP representation, addition and conversion. A multitude of algorithm-to-architecture translations exist for each building block but not all of them are compatible. What has been lacking at present is the understanding of the extent of influences on different VLSI performance factors of one module to its concomitant module upon their integration. With this motive as genesis, the existing and proposed new modules from each building block that have potential to form a high-performance RB multiplier are independently studied and evaluated. The advantage of this anatomy is that it facilitates speciation of RB multipliers from sensible topological combinations of modules. Altogether twenty-one different $N \times N$ -bit RB multiplier architectures are constructed with varying configurations of partial product encoding, generation and reduction to explore the design space.

Due to the pervasion of mobile communication systems, and the severe constriction in the space and weight of portable electronics, power or more specifically, the energy per operation has been an ineluctable evaluation metric in VLSI design. As power or energy consumption is a monotonic function of supply voltage, it can in principle be reduced as much as possible by re-

ducing the supply voltage. This strategy is not compatible with increasing throughput rate. Technology and manufacturing yield also pose a limit on the supply voltage reduction. Nevertheless, speed remains an important attribute of digital multiplier design as multiplications have found to be the bottleneck in the data paths of many real-time digital signal processing benchmarks. Due to the fact that the fastest strategies are not always the ones that consume the most power, the designer might sometimes prefer to “using a design that is fast enough and consumes the least power” than “using the fastest design” [17]. Therefore, apart from aiding a designer in selecting an RB multiplier architecture for a given word length with the delay and power characteristic, this chapter also provides the energy-delay product evaluation for design tradeoffs in power saving and performance enhancement. With a myriad of RB multiplier designs at the disposal of a computer architect, it helps to provide a better understanding of different dovetailings of architectural constructs and their implications on important but conflicting design constraints. The intriguing augmentation and restriction between different architectural modules of Booth encoders and RB arithmetic coding inferred from this comparative study are instrumental to the innovation of RB multiplier designs.

The remainder of this chapter is organized as follows. Section 5.2 provides a taxonomic designation of BEPPG for ease of analysis. Several one-digit BEPPG modules are qualitatively analyzed before their influences on the area-time performance of $N \times N$ -bit RB multiplier architectures are discussed based on the FO4 delay and number of unit gates. Section 5.3 presents the coherent RB coding interface components which include the one-digit RBA cell and some simple anterior and posterior converters of RBA summing tree. In Section 5.4, twenty-one $N \times N$ -bit RB multiplier architectures are constructed with

varying configurations of partial product reduction and RB coding methods for design space exploration. The performance evaluation and discussion on these designs are also presented. Finally, the concluding remarks from these analyses are provided in Section 5.5.

5.2 Architectural Exploration on RB Multipliers

5.2.1 Taxonomy of Booth Encoders and Partial Product Generators (BEPPGs)

Being the front end circuits of RB multiplier design, the Booth encoder and the PPG contribute critically to the performance and cost of the multiplier as a whole. How efficient the RBPPs are generated affects the area-delay-power trade-off of subsequent summation network. A Booth encoder can be deemed as a digit-set converter as each slice of it converts a string of binary bits to a signed digit. The choice of a good digit-set converter for a given operand length is prerogative in that once it is fixed, the RB multiplier design loses a great deal of mobility on the speed-size optimization space. This subsection focuses on various configurations of the BEPPG modules based on the existing Booth encoding algorithms, which have been discussed extensively in Section 2.2.1.

For the convenience of analysis, we make a dichotomy of the Booth encoding algorithms according to the way in which the partial products are generated. Those Booth encoding methods that generate the partial products in NB format are classified as the NBBE and those others that generate the RBPPs

directly are classified as RBBE. The partial product generator is also known as the Booth decoder. Since the Booth encoder and decoder are normally dovetailed as a single entity, for brevity, the abbreviations of NBBE and RBBE are also used for the dovetailed BEPPG with no ambiguity.

5.2.1.1 Normal Binary Booth- k Encoding (NBBE- k)

In NB Booth- k algorithm (k is a positive integer), a Booth-encoded digit is generated from $k+1$ consecutive bits of a NB number. As illustrated in (2.1), the digit-set conversion process entails no carry propagation when $k \leq 2$. This is referred to as the simple Booth encoding, as opposed to the high-radix Booth encoding. In simple NB Booth encoding, NBBE-1 is obsolete as it has zero partial product reduction. Therefore, the only useful one left is NBBE-2, which is widely used in high-speed digital multipliers to halve the number of NB partial products. To minimize the delay time and eliminate the glitches associated with the Booth multiplier, a modified NB Booth encoding was proposed in [110] and [111]. Compared with NBBE-2, the Modified NBBE-2 (MNBBE-2) saves one gate delay in the path of Booth encoder with the penalty of an increased number of gates used in the PPG.

With the radix value increases to $k > 2$, hard multiples emerge and mandate carry propagation additions, which complicate the realization of high-radix Booth encoders and their PPGs. Although the number of partial products in the summation network can be proportionally reduced by increasing the radix of NBBE, there is a limit over which the advantage of high partial product reduction rate is offset by the sophistication of generating the hard multiples and the decoding logic.

Based on our classification, the PRBBE algorithm reviewed in Section 2.2.1 falls under the high-radix NBBE category. This is because the partial products generated from PRBBE are in NB format.

5.2.1.2 Redundant Binary Booth- k Encoding (RBBE- k)

In RBBE scheme, most of the multiples can be expressed as a difference of two simple power-of-two multiples. The partial products that are generated conform to the format of positive-negative RB coding. This encoding method has eliminated the correction vector in the RB summing tree due to two's complement arithmetic and RB coding. A representative of RBBE is that of [36]. However, as only one Booth encoded digit is consumed for one RBPP, half of the binary bits representing the RBPP generated from the simple multiple in RBBE are filled with '0's, which is rather inefficient.

A derivative of RBBE scheme is the new CRBBE presented in Chapter 4. It binds two adjacent modified Booth encoders to compose an RBPP. It shares the same advantages of RBBE for the ease of generating hard multiples and avoidance of error compensation vector, the two problems associated with NBBE.

5.2.2 One-Digit BEPPG Module

To avoid superfluous simulation data from obscuring a meaningful analysis, we omit the less competitive parametric modules and focus our evaluation on those representative and heterogeneous rivals. In consideration of the severity of hard multiple problem, it is reasonable to stop at radix-16 ($k \leq 4$) for high-radix NBBE and RBBE. For the meditative PRBBE, we consider the most

appealing PRBBE-3 for the analysis based on the recommendation of [57].

Under this premise, there are altogether seven competitive BEPPG modules proposed in recent RB multiplier designs. Figure 5.1 and Figure 5.2 illustrate the gate-level implementations of one slice of these BEPPG modules in NBBE and RBBE, respectively. For each BEPPG slice, a potential critical path is highlighted.

Apart from the difference in the generation of multiples, PRBBE-3 has exactly the same BEPPG as NBBE-3 [57]. Therefore, it can be demonstrated with the same schematic as NBBE-3 in Figure 5.1. Similarly, as far as only the encoder logic is concerned, NBBE-4 is equivalent to RBBE-4. They are differentiated by the PPGs. The CRBBE-4 circuit described in the previous chapter is implemented as shown in Figure 5.2, by abutting two Booth-2 encoders with an auxiliary polarization circuit.

An abridged characterization of the area-time requirement to generate one digit of RBPP is performed for each type of BEPPG modules. It should be noted that since the partial product generated by each slice of NBBE module is in NB form, two slices of NBBE based modules are required to generate one digit of RBPP. The delay of each module is evaluated on the critical path and expressed in terms of the FO4 delay in a CMOS 0.18- μm process model, and the number of unit gates (a unit gate is equivalent to a two-input NAND gate) of the Booth encoder and the partial product generator are separately accounted for the area complexity. The characterization is shown in Table 5.1.

From Table 5.1, MNBBE-2 has the shortest delay time and NBBE-2 is the most compact design to generate one digit of RBPP. For the same partial product reduction rate, RBBE-4 and CRBBE-4 are slower and more complex com-

Figure 5.1: Circuit implementations of BEPPG modules in NBBE.

Figure 5.2: Circuit implementations of BEPPG modules in RBBE.

Table 5.1: Delay and Unit Gate Number of One-Digit BEPPG Modules

BEPPG	Delay (FO4)	No. of Unit Gate	
		BE	PPG
NBBE-2	6.208	14	12
MNBBE-2	4.952	18	22
NBBE-3	7.168	34	20
NBBE-4	8.456	66	36
PRBBE-3	7.168	34	20
RBBE-4	9.002	33	28
CRBBE-4	7.212	26	16

paring with the above two modules. As this evaluation is made at digit level regardless of the type of RBPP generated, the delay and complexity of the CPA required to generate the hard multiple have not been apportioned. Therefore, although NBBE-3 and PRBBE-3 generate the hard multiples differently, they exhibit the same performance in Table 5.1. Furthermore, high-radix Booth encoding modules, NBBE-3 and NBBE-4 are obviously inferior to the simple Booth encoding module in standalone comparison. However, due to the different partial product reduction rate, the landscape of RB multiplier employing these BEPPG modules might change as the length of the operand varies.

5.2.3 Qualitative Analysis of BEPPG on $N \times N$ -bit RB Multipliers

Let N be the operand length of the RB multiplier, with Booth- k encoding, the number of Booth encoders, n_{BE} , and PPGs n_{PPG} , can be calculated as indicated in (5.1) and (5.2), respectively.

$$n_{BE} = \left\lceil \frac{N}{k} \right\rceil \quad (5.1)$$

$$n_{PPG} = (N + k - 1) \cdot \left\lceil \frac{N}{k} \right\rceil \quad (5.2)$$

Therefore, the total number of RBPPs in the summation network can be derived as in (5.3) and (5.4), for NBBE- k and RBBE- k , respectively.

$$n_{PP-NBBE} = \left\lceil \frac{N}{2k} \right\rceil + 1 \quad (5.3)$$

$$n_{PP-RBBE} = \left\lceil \frac{N}{k} \right\rceil \quad (5.4)$$

From (5.1) to (5.4), the number of Booth encoders and PPGs are the same for the same radix of NBBE and RBBE algorithms, but the number of RBPPs generated from NBBE is around half of that generated from RBBE. Therefore, NBBE- k has approximately the same reduction rate of RBPPs as RBBE- $2k$. On the other hand, by comparing (5.3) and (5.4), it is noted that the correction vector needed by NBBE for the RB coding and the partial product negation has been eliminated in RBBE. If the bit length of the multiplier is exactly $2^{n+1} \cdot k$, the extra vector required by the NBBE multiplier will cost not only additional hardware and more power consumption for its accumulation, but also extra

Table 5.2: Characteristics of $N \times N$ -bit RB Multiplier Architectures with Different BEPPGs

Multiplier	No. of BE	No. of PPG	No. of RBPP	No. of RBA Stage	CPA Incurred	Correction Vector Incurred
NBBE-2	$\left\lceil \frac{N}{2} \right\rceil$	$(N+1) \cdot \left\lceil \frac{N}{2} \right\rceil$	$\left\lceil \frac{N}{4} \right\rceil + 1$	$\left\lceil \log_2 \left(\left\lceil \frac{N}{4} \right\rceil + 1 \right) \right\rceil$	N	Y
MNBBE-2	$\left\lceil \frac{N}{2} \right\rceil$	$(N+1) \cdot \left\lceil \frac{N}{2} \right\rceil$	$\left\lceil \frac{N}{4} \right\rceil + 1$	$\left\lceil \log_2 \left(\left\lceil \frac{N}{4} \right\rceil + 1 \right) \right\rceil$	N	Y
NBBE-3	$\left\lceil \frac{N}{3} \right\rceil$	$(N+2) \cdot \left\lceil \frac{N}{3} \right\rceil$	$\left\lceil \frac{N}{6} \right\rceil + 1$	$\left\lceil \log_2 \left(\left\lceil \frac{N}{6} \right\rceil + 1 \right) \right\rceil$	Y	Y
NBBE-4	$\left\lceil \frac{N}{4} \right\rceil$	$(N+3) \cdot \left\lceil \frac{N}{4} \right\rceil$	$\left\lceil \frac{N}{8} \right\rceil + 1$	$\left\lceil \log_2 \left(\left\lceil \frac{N}{8} \right\rceil + 1 \right) \right\rceil$	Y	Y
PRBBE-3	$\left\lceil \frac{N}{3} \right\rceil$	$(N+2) \cdot \left\lceil \frac{N}{3} \right\rceil$	$\left\lceil \frac{N+1}{6} \right\rceil + 1$	$\left\lceil \log_2 \left(\left\lceil \frac{N+1}{6} \right\rceil + 1 \right) \right\rceil$	Y	Y
RBBE-4	$\left\lceil \frac{N}{4} \right\rceil$	$(N+3) \cdot \left\lceil \frac{N}{4} \right\rceil$	$\left\lceil \frac{N}{4} \right\rceil$	$\left\lceil \log_2 \left\lceil \frac{N}{4} \right\rceil \right\rceil$	N	N
CRBBE-4	$\left\lceil \frac{N}{4} \right\rceil$	$(N+3) \cdot \left\lceil \frac{N}{4} \right\rceil$	$\left\lceil \frac{N}{4} \right\rceil$	$\left\lceil \log_2 \left\lceil \frac{N}{4} \right\rceil \right\rceil$	N	N

delay in generating the final product.

Table 5.2 summarizes the characteristics of seven $N \times N$ -bit multipliers employing different BEPPG modules of Figure 5.1 and Figure 5.2. It lists the quantity of various forms of resources, including the number of Booth encoders, the number of PPGs, the number of RBPPs, and the number of stages of RBA summing tree. It also indicates whether or not the correction vector and the CPA are required for each RB multiplier.

From Table 5.2, it can be seen that the RB multiplier architecture employing NBBE-4 has the largest partial product reduction rate. Due to the least number of RBPPs, it may also have the smallest number of stages in the RBA summing tree. However, this advantage is mediated by the requirement of CPAs for the generation of hard multiples. NBBE-3 and PRBBE-3 also face the same problem

although only one hard multiple is needed to be generated. The RB multiplier architectures employing NBBE-2 and RBBE-4 have exactly the same feature as those with MNBBE-2 and CRBBE-4, respectively.

5.3 Coherent RB Coding Interface Components

5.3.1 One-Digit RB Adder Cells

The RBPP summing tree is the cornerstone of RB multiplier. The key component that appears abundantly in RB summing tree is the RBFA cell. In Section 2.2.2, three representative RBA cells designed based on different coding methods [13, 34, 58] have been introduced. To make the abbreviation meaningful, from this point onwards, the RBA cells designed with the sign-magnitude, positive-negative and positive-negative-complement codings are abbreviated as RBA_SM, RBA_PN and RBA_PNC, respectively. These RBA cells are evaluated here to manifest the effect of coding on the performance of RBA. To make this section self-contained, their gate-level circuit implementations are reproduced in Figure 5.3. In addition, the corresponding half adder cells are also developed to simplify the design of RBPP summing tree. These RB half adders are useful for the summation of an RB variable and an RB constant in some corner cells of the RB summing tree. The respective RB half adders are prefixed with RBHA to differentiate them from the RB full adders.

The FO4 delay with CMOS 0.18- μm process model and the number of unit gates of RB full and half adders for SM, PN and PNC codings are summarized in Table 5.3.

5.3 Coherent RB Coding Interface Components

123

Figure 5.3: Circuit implementation of RBA cells.

Table 5.3: FO4 Delay and Complexity of RB Full and Half Adders

RBA Cell	Delay (FO4)	No. of Unit Gate
RBFA_SM	3.824	15
RBHA_SM	2.924	7
RBFA_PN	3.456	17
RBHA_PN	2.822	8
RBFA_PNC	3.740	21
RBHA_PNC	2.606	8

Since the use of RBHA and RBFA is mutually exclusive and the critical path in the RB summing tree is dominated by the number of RBFA, the delay of RBHA is of less significant in this comparison. This result indicates that RBA_PN is the fastest adder among the three coding schemes with moderate adder complexity. SM coding leads to the least complexity RB full and half adder cells, but these adder cells are also the slowest. RBA_PNC has an intermediate speed but uses the most number of gates.

5.3.2 Converters for Coherent RBA Interface

To fuse the heterogeneous fabrics designed with different coding formats into an RB multiplier, some simple converters are needed before and/or after the RBA summing tree. Although Booth encoding itself can be seen as a digit-set converter, its purpose is to reduce the number of stages required in the RB summing tree. Not all Booth encoding schemes discussed in Section 5.2.2 prepare the partial products in a form ready for consumption by the RBA. Some sim-

Figure 5.4: Three anterior converters used in RB multiplier design.

ple converters are required to convert the NB partial products to RBPPs prior to the RBA summing tree stage. Figure 5.4 shows three different one-digit anterior converters. Provision has also been made to eliminate the ambiguity of dual representations of '0' in these converters. Figure 5.4(a) illustrates the one-digit converter used in the conversion of NB partial products to RBPPs in order to add them with the RBA_SM summing tree for the NBBE based RB multipliers. Since the RBPPs generated directly from the RBBE algorithm assume the PN coding format, the anterior converter is needed to adapt them to the RBAs designed for other coding methods. Figure 5.4(b) depicts such a converter used to prepare the RBPPs generated by RBBE for the reduction by the RBA_SM summing tree. Figure 5.4(c) shows another converter used to adapt the RBPPs to RBA_PN summing tree for both NBBE and RBBE based RB multipliers. It should be noted that NB-to-RB converters are also necessary for RBA_PNC addition. However, each of them can be reduced to a simple inverter as indicated in Section 4.4.2 and absorbed into the receiving RBA cell.

Another kind of converter needed for the fusion of heterogenous coding formats appears in the final stage of RB multiplier. This simple converter circuit is referred to as a posterior converter, which is used to adapt an RB-to-NB reverse converter circuit to the RB input of any coding format. The reverse conversion algorithm can be unified for all three coding methods as discussed

Figure 5.5: Posterior converter used in RB-to-NB conversion for PNC coding.

in Section 3.2. The unanimity of carry generation using the same logical structure has already been taken care by the above forward converters for the SM and PN coding schemes. The redundant mappings have been removed prior to the RBA summing tree stage to simplify the RBA cell design. For the case of PNC, due to the coding symmetry, there is no need to eradicate the dual representations of zero before the RBA summing tree. The resolution of redundant mapping can be deferred until the RB-to-NB converter stage. Figure 5.5 illustrates the coherent converter, which is used only for the PNC coding in order to unify the reverse conversion algorithm.

5.4 Performance Evaluation and Discussions

5.4.1 Configurations of RB Booth Multipliers

Based on the characteristics of the fabrics presented in Sections 5.2 and 5.3, different configurations of RB Booth multipliers are delineated according to the types of RBA cells and converters in Table 5.4. The logic equations of one-digit anterior and posterior converters in NBBE and RBBE based multiplier architectures are also listed in the table where applicable. Any efficient parallel adder architectures can be employed to improvise the RB-to-NB converter. The RB-to-NB converters implemented for the three RB coding schemes, SM, PN and

Table 5.4: Configurations of RB Multipliers with Different Code Converters

RB Multiplier Architecture	Anterior Converter	RBA Cell	Posterior Converter	RB-to-NB Converter
NBBE	$\begin{cases} f_i^{s'} = \overline{f_i^s} + f_i^a \\ f_i^{a'} = f_i^s \odot f_i^a \end{cases}$	RBA_SM	N.A.	CONV_SM
RBBE	$\begin{cases} f_i^{s'} = \overline{f_i^s} \cdot f_i^a \\ f_i^{a'} = f_i^s \odot f_i^a \end{cases}$			
NBBE & RBBE	$\begin{cases} f_i^{+'} = \overline{f_i^+} \cdot f_i^- \\ f_i^{-'} = f_i^+ \cdot \overline{f_i^-} \end{cases}$	RBA_PN	N.A.	CONV_PN
NBBE & RBBE	N. A.	RBA_PNC	$\begin{cases} f_i^{+'} = f_i^+ \cdot f_i^- \\ f_i^{-'} = f_i^+ + f_i^- \end{cases}$	CONV_PNC

PNC are abbreviated as CONV_SM, CONV_PN, and CONV_PNC, respectively.

If PNC coding is used for the RB multiplier design, the anterior converters can be saved, but posterior converters are introduced to remove the representation redundancy. From the logic equations, the delays of the anterior and posterior converters are about the same. The anterior converters for RBA_SM are slightly slower due to the longer XOR or XNOR gate delay. All converters perform in constant gate-delay time independent of the word length. The difference comes from the number of the converter circuits required. A posterior converter is required for each digit in only the final sum of the RBA summing tree whereas an anterior converter is needed for every digit of the RBPPs before the RBA summing tree. The anterior converter circuits certainly outnumber the posterior converter circuits. From this point of view, PNC coding seems to be more efficient than the other two codings. Furthermore, for RB-to-NB converters, according to (3.4), (3.8) and (3.12), the delay time of these

three converters in CMOS implementation is comparable, while CONV_SM is slightly simpler.

The coding efficiency can be qualitatively analyzed in each stage as discussed. However, when different BEPPG modules are amalgamated with RBA summing tree using different RB coding methods, the efficiency of the RB multiplier design due to different coding methods cannot be easily ascertained. There are bewildering design options considering the number of modules substitutable in each stage of the RB Booth multiplier architecture. Every module has some intriguing merits of its own. When the modules augment each other, it makes the configuration more competitive under certain operand length. The findings are best corroborated by the synthesis results.

To date, no systematic analysis of important VLSI metrics has been made in the literature for different RB multiplier topologies by applying a uniform simulation and comparison strategy. In this section, a variety of RB multiplier topologies derived from several intriguing Booth encoding methods and three main RB coding schemes are implemented, synthesized and compared for speed, power consumption and energy-delay products. Altogether twenty-one different RB multipliers are built from various designs of each module of Table 5.4. The RB-to-NB converters of all these multipliers are designed with the same hybrid carry-lookahead/carry-select conversion algorithm proposed in Chapter 3. The following convention is adopted for the nomenclature of RB multipliers. Each multiplier is denoted by a prefix of its BEPPG module name indicated in Section 5.2.2 and a postfix of the designated coding format. Among these 21 RB multiplier configurations, 15 designs are presented for the first time.

5.4.2 Numerical Simulation Results

This subsection presents the simulation results of RB multipliers for six commonly used operand lengths from 8 to 64 bits to extrapolate the performance trajectory of each multiplier as it scales. Each design is structurally described at gate level using VHDL. The designs are functionally verified by ModelSim [106] for randomly generated input patterns before they are synthesized and mapped to Artisan TSMC 0.18- μm standard cell library [82] using the Synopsys Design Compiler [83] with a nominal wire load model. The gate-level simulation is performed using the environment described in Section 2.4.3. The mean power dissipation for each RB multiplier is calculated with the Monte Carlo statistical model with more than 99.9% confidence level that the error is bounded below 3%.

Since multiplication is often the speed-limiting elements in application, optimization in terms of speed is pursued by the synthesis tool. Table 5.5 shows the area results of the synthesis and it is indicative of the relative complexity of the RB multipliers in comparison when area optimality is traded for speed. Table 5.6 lists the worst-case delays of different sizes of RB multiplier configurations. The power consumption is also simulated based on the maximum input rate that each individual multiplier is able to function. Therefore, the energy per operation of each design can be obtained by multiplying the average power consumption with the worst-case delay. These results are summarized in Table 5.7. The area-time-energy trade offs are illuminated in a three-dimensional scatter plot of Figure 5.6, where the abscissas are the natural logarithm of energy dissipation in pJ and worse case delay in ns, and the ordinate is the natural logarithm of area in μm^2 . Different shapes and symbols are used

Figure 5.6: Scatter plot of area *vs.* worst-case delay and energy dissipation in natural logarithmic scale.

to denote different RB multiplier configurations. The shapes and symbols are colored in blue, red and green to indicate the RB coding schemes of SM, PN and PNC, respectively.

5.4.3 Analyses and Discussions

The voluminous amount of data makes the analysis difficult due to the intricate correlation between different contributing factors. In this subsection, the results are discussed in three perspectives. First, the Booth encoder and decoder complexity of two different classes of Booth multipliers and the effect of extra correction vector as the size of the multiplier changes. Second, the ad-

Table 5.5: Comparisons on Area of RB Multipliers

S/N	RB Multiplier Architecture	Area(μm^2)					
		8×8-b	16×16-b	24×24-b	32×32-b	48×48-b	64×64-b
1	NBBE-2.SM	18,939	51,825	97,174	164,320	359,613	629,400
2	MNBBE-2.SM	20,065	54,996	107,508	182,261	400,570	703,345
3	NBBE-3.SM	22,055	57,292	97,144	148,954	295,150	505,769
4	NBBE-4.SM	22,026	59,099	99,009	155,450	319,171	550,086
5	PRBBE-3.SM	22,705	59,644	102,981	164,645	344,313	584,378
6	RBBE-4.SM	23,616	65,721	119,381	211,051	444,232	716,549
7	CRBBE-4.SM	19,059	54,782	102,656	168,407	359,431	646,877
8	NBBE-2.PN	21,142	57,073	102,125	169,905	367,058	646,794
9	MNBBE-2.PN	22,076	60,574	112,995	186,667	408,234	755,796
10	NBBE-3.PN	23,105	58,932	104,335	153,411	300,463	536,930
11	NBBE-4.PN	24,158	63,919	106,939	162,929	320,321	580,345
12	PRBBE-3.PN	24,253	63,528	107,564	171,882	346,699	603,792
13	RBBE-4.PN	26,882	69,919	126,832	229,283	460,102	761,062
14	CRBBE-4.PN	21,295	59,034	110,154	177,137	376,428	669,862
15	NBBE-2.PNC	19,536	56,537	106,098	175,451	376,892	659,308
16	MNBBE-2.PNC	21,152	58,038	115,946	186,179	411,403	752,977
17	NBBE-3.PNC	22,513	60,038	101,508	158,051	315,466	555,377
18	NBBE-4.PNC	24,534	63,289	104,514	166,072	330,264	586,514
19	PRBBE-3.PNC	23,152	62,554	108,320	174,147	357,295	621,987
20	RBBE-4.PNC	24,988	69,011	130,045	242,481	471,647	769,544
21	CRBBE-4.PNC	20,817	57,829	109,324	176,254	388,313	691,969

Table 5.6: Comparisons on Worst-Case Delay of RB Multipliers

S/N	RB Multiplier Architecture	Delay(ns)					
		8×8-b	16×16-b	24×24-b	32×32-b	48×48-b	64×64-b
1	NBBE-2_SM	1.906	2.581	3.011	3.405	3.856	4.427
2	MNBBE-2_SM	1.766	2.401	2.795	3.194	3.679	4.209
3	NBBE-3_SM	2.158	2.799	3.329	3.628	4.329	4.702
4	NBBE-4_SM	2.433	3.192	3.750	4.109	4.756	5.156
5	PRBBE-3_SM	1.985	2.629	3.156	3.487	4.078	4.582
6	RBBE-4_SM	1.823	2.518	3.050	3.429	4.039	4.453
7	CRBBE-4_SM	1.675	2.269	2.863	3.167	3.738	4.181
8	NBBE-2_PN	1.787	2.358	2.723	3.064	3.562	4.011
9	MNBBE-2_PN	1.647	2.219	2.591	2.923	3.392	3.870
10	NBBE-3_PN	2.131	2.708	3.154	3.433	4.002	4.391
11	NBBE-4_PN	2.303	3.019	3.469	3.872	4.418	4.819
12	PRBBE-3_PN	1.802	2.401	2.829	3.202	3.735	4.249
13	RBBE-4_PN	1.741	2.351	2.828	3.197	3.786	4.184
14	CRBBE-4_PN	1.629	2.193	2.651	2.892	3.454	3.812
15	NBBE-2_PNC	1.809	2.468	2.834	3.286	3.684	4.297
16	MNBBE-2_PNC	1.677	2.236	2.638	3.034	3.531	4.054
17	NBBE-3_PNC	2.143	2.743	3.232	3.511	4.188	4.672
18	NBBE-4_PNC	2.314	3.057	3.576	4.025	4.595	4.976
19	PRBBE-3_PNC	1.885	2.502	2.925	3.301	3.915	4.419
20	RBBE-4_PNC	1.712	2.396	2.865	3.246	3.898	4.295
21	CRBBE-4_PNC	1.588	2.159	2.652	2.969	3.579	3.938

Table 5.7: Comparisons on Energy Dissipation of RB Multipliers

S/N	RB Multiplier Architecture	Energy Dissipation (pJ)					
		8×8-b	16×16-b	24×24-b	32×32-b	48×48-b	64×64-b
1	NBBE-2_SM	4.789	14.82	29.98	53.78	122.05	223.42
2	MNBBE-2_SM	5.484	16.83	35.01	61.64	140.07	249.65
3	NBBE-3_SM	5.427	15.94	30.19	50.29	106.16	196.62
4	NBBE-4_SM	6.133	18.68	34.26	55.93	117.77	210.93
5	PRBBE-3_SM	5.322	16.57	31.99	54.92	120.04	210.29
6	RBBE-4_SM	5.684	18.99	35.04	62.96	142.33	252.54
7	CRBBE-4_SM	5.093	16.03	32.93	56.57	130.18	228.24
8	NBBE-2_PN	4.931	16.04	31.95	57.66	127.11	233.41
9	MNBBE-2_PN	5.651	17.85	36.56	63.98	144.19	265.54
10	NBBE-3_PN	5.451	16.76	32.36	54.92	114.07	208.03
11	NBBE-4_PN	6.496	19.99	37.62	60.07	127.12	228.48
12	PRBBE-3_PN	5.552	17.43	33.75	58.06	124.15	224.97
13	RBBE-4_PN	5.661	18.84	36.49	66.09	147.93	264.06
14	CRBBE-4_PN	5.135	16.25	33.46	59.08	132.80	238.56
15	NBBE-2_PNC	4.916	15.52	31.51	55.73	123.95	229.78
16	MNBBE-2_PNC	5.607	17.73	37.49	62.78	143.21	258.92
17	NBBE-3_PNC	5.480	16.54	31.82	53.51	111.13	205.19
18	NBBE-4_PNC	6.584	19.13	35.96	59.11	123.64	222.84
19	PRBBE-3_PNC	5.544	17.02	33.98	56.63	123.05	220.96
20	RBBE-4_PNC	5.646	19.07	37.11	68.03	145.78	259.87
21	CRBBE-4_PNC	5.084	16.25	34.07	58.37	131.75	237.02

versity of hard multiples as the radix of the Booth multiplier increases. Third, the impact RB coding method has on the overall performance of the multiplier. Since the coding efficiency analysis has been decoupled in the first two discussions, only the results of RB multipliers with PN coding are presented for analysis in Subsections 5.4.3.1 and 5.4.3.2. The exceptions that deviate from the general extrapolation are singled out for separate discussion in these subsections.

5.4.3.1 Normal Binary Booth Encoding vs. Redundant Binary Booth Encoding

As discussed in Section 5.2, Booth encoding is classified as NBBE and RBBE depending on the way their RBPPs are generated. For the same radix number, the partial product reduction rate of NBBE is double that of RBBE. To account for effects due to the different types of Booth encoders and decoders, a reasonable and meaningful comparison shall be based on the same RBPP reduction rate. Therefore, two NBBE multipliers: NBBE-2 and MNBBE-2, and two RBBE multipliers: RBBE-4 and CRBBE-4 with the same reduction rate of $1/4$ have been selected for this discussion.

From Table 5.6, it is found that the CRBBE-4 multiplier is the fastest design for all the power-of-two operand lengths. For these operand lengths, CRBBE-4 multiplier executes on average 6.60%, 1.21%, and 7.90% faster than NBBE-2, MNBBE-2, and RBBE-4 multipliers, respectively. Due to the existence of correction vector, the speed of NBBE multiplier is degraded by the additional stage in the partial product summation network. For 24-bit and 48-bit multipliers, MNBBE-2 multiplier executes on average 4.81%, 9.39% and 2.03% faster

than NBBE-2, RBBE-4 and CRBBE-4 multipliers, respectively. This is because when the operand lengths are not power-of-two, the extra correction vector contributes little or no effect to the critical path delay.

From Table 5.7, it is evident that NBBE-2 multiplier always consumes the least energy. It saves about 11.55%, 13.11% and 3.10% energy comparing with MNBBE-2, RBBE-4 and CRBBE-4 multipliers, respectively. A closed examination of the break down of our power analysis results reveals that although MNBBE-2 multiplier consumes the least switching power, it consumes larger cell internal power, which can probably be imputed to its larger gate internal capacitance. CRBBE-4 multiplier is secondary in energy and its energy consumption approximates that of NBBE-2 multiplier. Despite having a lower complexity of Booth encoder and PPG, the RBAs in the RBPP summing tree of NBBE-2 multiplier outnumber those of CRBBE-4 multiplier, which accounts for the reduced ascendancy in energy dissipation. RBBE-4 multiplier presents lower speed and dissipates more energy than CRBBE-4 multiplier for all word lengths. This is primarily due to its less efficient encoder and much more complicated PPG.

If both speed and energy consumption are pursued simultaneously, the combined effect of energy efficiency is best benchmarked using the EDP metric. Figure 5.7 shows the EDP of RB multipliers of these four multipliers. The EDP for each operand length is normalized so that the multiplier with the largest EDP has an EDP of one. The results show that CRBBE-4 multiplier is most energy efficient for the power-of-two operand lengths, and NBBE-2 multiplier tops all multipliers for operand lengths that are not power-of-two.

Similar trends of delay, energy and EDPs are also observed for the same

Figure 5.7: Normalized EDP of NBBE and RBBE multipliers.

four multiplier architectures with SM and PNC codings except that the extent of performance difference in each case varies somewhat.

5.4.3.2 High-Radix Booth Encoding vs. Simple Booth Encoding

As indicated in Section 5.2, the existence of hard multiples is a major issue of high-radix Booth encoding schemes. To assess the significance of hard multiples, the high-radix Booth encoding schemes, NBBE-3, PRBBE-3 and NBBE-4 are highlighted for comparison with the simple Booth encoding, NBBE-2.

From Table 5.6, it is conspicuous that high-radix Booth multipliers are slower in this group of RB multipliers. On average, NBBE-2 multiplier outperforms NBBE-3, NBBE-4 and PRBBE-3 multipliers in speed by 12.19%, 20.47% and 3.49%, respectively. The delay time aggravates as the radix number increases in the high-radix Booth multipliers. This shows that the generation of

hard multiples is indeed a major performance stumbling block of these multipliers.

As observed from Table 5.7, among the three high-radix Booth multipliers, NBBE-3 multiplier consumes the least energy in view of a better trade-off between the complexity of the RBA summing tree and the number of CPAs required for their hard multiple generations. The energy saving of NBBE-2 multiplier is not prominent and it diminishes gradually as the operand length increases. It exhibits 9.54%, 4.27%, 1.27% lower energy dissipation than NBBE-3 for 8-bit, 16-bit and 24-bit multipliers, respectively. When the word length increases to 32, 48 and 64 bits, it begins to consume respectively, 4.75%, 10.26%, 10.87% more energy than NBBE-3 multiplier. This can be explained as follows. Comparing with NBBE-2 multiplier, NBBE-3 multiplier has more complex Booth encoder and selector logics, as well as high overhead of hard multiple generation. When the size of the multiplier is small, excessive energy are dissipated in these logic circuits. As the word length of the multiplier increases, more RBPPs can be reduced by NBBE-3 and the energy reduction in the RBA summing tree offsets these logic overheads.

For the same rate of partial product reduction, it is interesting to note that with small length adders and additional compensation vector, PRBBE-3 multiplier achieves higher speed than NBBE-3 multiplier with a penalty of more energy dissipation. Figure 5.8(a) shows the normalized EDP of these four RB multipliers graphically. It indicates that NBBE-2 multiplier is the most energy-efficient design from 8 bits to 48 bits but the efficiency decreases gradually and it loses out to NBBE-3 multiplier when the operand length increases to 64 bits.

These four multiplier architectures with PNC coding follow similar trends

Figure 5.8: Normalized EDP of high-radix and simple Booth multipliers.

in delay, energy and EDP comparisons. Therefore, the above analysis is also valid for PNC coding. However, the EDP rankings of the same four multipliers designed with SM coding have some subtle differences as illustrated in Figure 5.8(b). It is noted that NBBE-3 multiplier becomes advantageous from 32 bits onwards instead of 64 bits. The EDP gaps between NBBE-3 and PRBBE-3 shown in Figure 5.8(b) also display a different trend from that shown in Figure 5.8(a). This has led to the following investigation on the coding efficiency.

5.4.3.3 RB Coding Efficiency

It can be observed from Table 5.6 that most of the RB multipliers implemented with PN coding are faster than their counterparts implemented with the other two codings. So generally speaking, designs with positive-negative coding possess higher speed. This is probably because of the RBA cell, RBA_PN is the fastest among the three RBA cells. On the other hand, as noted in many cases of Table 5.7, SM coding produces the multiplier designs with the least energy consumption. This is also consistent with the earlier qualitative analysis, which indicates that RB multiplier implemented with RBA_SM and CONV_SM has the least logic complexity. To further investigate the RB coding efficiency, the normalized EDP values for all RB multipliers are consolidated in Figure 5.9, with one chart for each word length from 8 bits to 64 bits.

From these results, we have the following insights:

1. It is difficult to make a conclusive inference on coding efficiency, but some RB coding schemes are found to have a flair for certain Booth multiplier architectures. In most situations, NBBE-2, MNBBE-2 and PRBBE-3 multipliers perform better with PN coding; NBBE-3 and NBBE-4 multi-

Figure 5.9: Normalized EDP of all RB multipliers. The sizes of the multipliers from top left to bottom right are 8-bit, 16-bit, 24-bit, 32-bit, 48-bit and 64-bit.

pliers are more efficient with SM coding. CRBBE-4 and RBBE-4 multiplier is more energy efficient with PNC coding only when the operand length is small and the advantage tends towards PN coding when the operand length becomes larger.

2. For power-of-two operand lengths, CRBBE-4.PN multiplier achieves the smallest EDP. This is because its fast speed transcends the somewhat higher energy dissipation in the RBA summing tree. However, this ascendancy in EDP becomes less prominent when the word length increases.

This is possibly caused by its relatively complex Booth encoder and decoder logics in the partial product generation, comparing with those of NBBE multiplier.

3. For operand lengths that are not power-of-two, NBBE-2_PN and NBBE-3_SM multipliers outperform other 24-bit and 48-bit multipliers, respectively. This empirical conclusion is also consistent with the qualitative analysis made pertaining to the issues of high-radix and simple Booth encoding methods.

5.5 Summary

In this Chapter, high performance Booth multiplier based on RB number representation has been investigated by dissecting its key constituent modules. The design considerations on several building modules and their logic circuits have been qualitatively discussed at a higher level of abstraction to highlight the potential performance trade-off for further empirical study. The unification of the reverse converter proposed in the preceding chapter and the coherent anterior and posterior interfacing logics make harmonious composition of RB multipliers from heterogeneously encoded modules possible. Upon ruling out incompatible and uncompetitive architectural options, twenty-one different configurations (most of them are novel circuit configurations not explicitly reported in literature) of $N \times N$ -bit RB multiplier architectures have been constructed from combinations of various designs of each module. These RB multipliers have been implemented, simulated, analyzed and compared for different scales of operand lengths from $N = 8$ to 64. The investigation has been

carried out with a neutral standing using a consistent synthesis setup and an appropriate figure of merit. Based on the simulation results, design guidelines have been deduced to help an architect to select the most suitable topology with the desired characteristics. To summarize, high-radix Booth multiplier is not suitable for speed-dominated design, but it remains an attractive choice for low power applications with large dynamic range. Covalent RB Booth encoding is recommended for power-of-two operand lengths for its high speed and low energy-delay product especially for digital multimedia applications where 8-bit and 16-bit multiplications are ubiquitous. We have also shown that the advantages of some topologies can be undermined by the types of RB coding format used. In general, sign-magnitude coding is more likely to produce lower power designs for the same Booth multiplier architecture, while positive-negative coding tends to yield higher speed designs.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

Most of the research in digital multipliers in the last few decades has focused on reducing the delay of RBPP accumulation. In the era of pervasive computing, however, the emphasis of VLSI design is on both high speed and low power operation. This thesis has presented several new insights into the high-speed and energy-efficient RB multipliers. The RB multiplier architecture is trichotomized into a BEPPG module, an RBA summing tree, and an RB-to-NB converter. Advances in the architectural innovation of the BEPPG module and the RB-to-NB converter have been made over previous RB multiplier architectures. The improvement measured in terms of the energy-delay product implies that the composite criterion of processing speed and energy dissipation can not be simply achieved by supply voltage tuning. Independent studies and evaluations have been performed on the existing and proposed modules in each building block for better design space exploration. A structural ap-

proach has also been proposed to analyze the performance of $N \times N$ -bit RB multiplier constructed with a conglomerate of RBPP generation, encoding, reduction and conversion methods. Based on the analysis, the RB multiplier design space can be further enlarged through the informed decisions of the relative merits and tradeoffs of these architectural options.

To streamline the RB-to-NB converter design for the study of high-speed RB multiplier architectures, a new reverse conversion algorithm based on hybrid CLA/CSL method has been proposed to fully exploit the redundancy of RB coding for VLSI efficient implementation. The hierarchical expansion of the carry equation for the reverse conversion algorithm creates a regular multi-level structure. For a given RB operand length, an assortment of fast and regular CLA networks with non-uniform block factors has been explored. The evaluation has been made in conjunction with various block lengths of the CSL sections to find an optimal topology for the fastest reverse converter with low area cost. A highly optimized ripple-carry adder chain and an ingenious add-one circuit have also been proposed for the CSL circuit to lower its transistor count at no speed penalty. The LE characterization, which captures the effect of circuit's fan-in, fan-out and transistor sizing on performance, has been applied to analyze and model the speed of variants of carry generation network and different lengths of CSL sections for different operand lengths. The superiority of our proposed converter has been demonstrated by the HSPICE simulation results of the 64-bit transistor-level implementations of proposed converter compared against the same implementation of the fastest contender estimated from the LE model.

By exploiting the RB system and existing Booth encoding algorithms, an energy-efficient RB multiplier based on a new CRBBE algorithm has been pro-

posed. The proposed method fully exploits the characteristics of the Booth encoded numbers to overcome the two problems that are confronted by RB multiplier with NBBE. Consequently, it shares the same advantages of RB Booth encoding (RBBE), which facilitates the hard multiples generation and achieves a compatible reduction of RBPPs without inducing any correction vector. As the CRBBE algorithm generates the RBPPs more efficiently by consuming two RB digits for every RBPP it generated, the proposed encoder and decoder are less complex compared to the RBBE algorithm for the same radix. The detailed gate-level simulations results further indicated that the RB multiplier based on CRBBE-4 outperformed its rivals in terms of speed and energy efficiency for the power-of-two operand lengths ranging from 8 bits to 64 bits.

Finally, a structural and systematic approach has been proposed to design and analyze the RB multiplier architectures. Existing and proposed new constituent modules of high-performance RB multipliers have been qualitatively analyzed. Coherent logics for the harmonious amalgamation of different RB encoded circuits of each modular stage have been suggested. Altogether twenty-one different configurations of $N \times N$ -bit RB multiplier architectures have been implemented for commonly used operand lengths varying from 8 bits to 64 bits, including those that are not power-of-two. Most of these RB multiplier configurations are novel and their performance have not been explicitly studied in the literature. These multipliers have been synthesized with the same standard cell library and compared for the various VLSI metrics to explore a diversified design space from sensible topological combinations of different core functional modules. To summarize, high-radix Booth encoding algorithms are not as attractive as they used to be perceived in high-speed multiplier design. However, they remain attractive for low-power applications

with large dynamic range, especially for radix-8 NB Booth encoding since it presents a better trade-off between the complexity of the RBA summing tree and the number of CPAs required for their hard multiple generations. Covalent RB Booth encoding is recommended for the power-of-two operand lengths due to its high speed and low energy-delay product. Furthermore, it has been shown that the performances of certain topologies can be moderated by the types of RB coding format used. In general, sign-magnitude RB coding is more likely to produce lower power designs for the same Booth multiplier architecture, while positive-negative coding tends to yield higher speed designs.

In summary, the objectives set forth in this thesis on the design and analyses of RB Booth multipliers have been met. Apart from the new modules proposed in individual building block, the study shall pave the way to the advancement of RB multipliers and revitalize the applications of RB arithmetic.

6.2 Recommendations for Future Research

As usual, no research will be completed, since a new discovery naturally triggers the pursuit of the new frontier and dimension it projected. Based on the research presented in this thesis, several relevant topics and directions worthy of further exploration have been identified. Some of these areas are presently being investigated by other members of our research group.

As technology scaling continues to advance with shrinking feature sizes, the ratio of wire performance to the gate performance will keep increasing. Therefore, further evaluations on the RB multiplier architectures mentioned in Chapter 5 can be made more accurate with the parasitics extracted from the

layout of each design. However, this evaluation process itself introduces additional biases to the results due to the effect of layout optimization and an added dimension of tradeoff analysis beyond the logic elements of interest, which is the main scope of our current study. As it stands, the trends of the relative merits of different architectures are not significantly affected by different wire load models experimented. The presented approach can still be conceived as a promising way to scrutinize the structural features of RB multipliers. The analytical concept could be extended in future to investigate the effect of technology nodes, transistor-level optimization, floor planning, layout strategies and other custom design issues, by means of a customized granular module generator. New basis of analytical setup and figure of merit need to be established in order to realistically compare the interconnect and noise issues in advanced deep sub-micron process technology. The simulations are best performed and validated by creating a module generator using the 65-nm standard cell library, which is not presently available in our group. The intrinsic properties gained from the simulated performances at gate level are helpful pointers to develop this sophisticated platform for design and analysis in future.

In application specific data paths, each multiplier is generally designed for a fixed operand length determined by the range estimation. Rarely, will a designer take an already existing multiplier and use it directly for higher or lower operand lengths in another application or data path. Redesigning multipliers for a given operand length has been a standard practice to meet a system's specification. However, with the advent of intelligent multimedia and embedded systems, the quest for operator scalability has been intensified by the ever evolving wireless communication standards and data streaming proto-

cols. The penalty for achieving a higher versatility and reusability of arithmetic operation has been well accepted provided that the critical performance metrics are also appropriately scaled to suit the operand length. If area is not a premium, an $N \times N$ -bit RB multiplier can be designed to cater for the maximum operand length of a range of standard applications, and reconfigured or parameterized to perform the multiplication of less than N bits. For example, in an adaptive filter, the precision of the inputs and the coefficients may change dynamically to suit the resolution and attenuation characteristic of the front-end circuitry of different wireless communication standards. Since our proposed CRBBE algorithm presented in Chapter 4 generates the partial product in an efficient way without any additional correction vector, it is a viable candidate for implementing scalable integer multiplier based on RB arithmetic. An important criterion for scalability is the ease of composing a larger word length multiplier from several smaller word length multipliers. If it is implemented with other RB multipliers, the extension of bit length is hampered by the co-generated correction vectors. If it is implemented with NB multipliers, the processing of signed number will use more multiplexers and connecting wires to detect the boundary of sign extension, and select and route the partial products. Therefore, CRBBE multiplier possesses the promising features for minimizing the overhead and connectivity of scalable multiplier design. A good characterization of the power-delay locus will enable a better match of appropriate scalable multiplier over a range of application profiles. The study of Chapter 5 can be extended to the case of configurable and/or scalable RB multipliers.

The third potential research direction is in the high-radix multi-operand addition. One major merit of redundant number system stems from its carry-free

addition property. The carry-free addition is made possible by a special set of adding rules. Presently, the carry-free adding rules are developed for radix-2 RBA cell. A natural propellant is to generalize the adding rules and optimize them for higher radix RB adders. The question is: can better speed and lower complexity design be accomplished with the revised carry-free adding rules to achieve more than 2:1 RBPP reduction rate? It is acknowledged that the RBA behaves like a 4-to-2 compressor in the carry save addition of NB partial products since both of them can reduce four inputs of the same weight to two. A 4-to-2 compressor is an extension of a 3-to-2 counter to speed up the column compression of the dot matrix representation of NB adder tree. Higher order compressor families, like 6-to-2 and 9-to-2 compressors have also been proposed but they are not as successful in either simplifying or accelerating the NB partial product accumulation. In NB multipliers, these high order counters and compressors avoid the propagation of carries by saving them for the successive stages. The number of inputs and intermediate outputs becomes awkwardly irregular, which results in massive and long lateral communication wirings both within and across stages of the carry-saved adder tree. Due to the simpler lateral and cross-stage interconnections and good regularity of RBA tree, higher radix signed digit adders have potential to reduce the RBPPs more efficiently than the corresponding higher order compressors in NB partial product summing tree.

Author's Publications

Journal Papers

1. Yajuan He and Chip-Hong Chang, "A Power-Delay Efficient Hybrid Carry-Lookahead/Carry-Select Based Redundant Binary to Two's Complement Converter," *IEEE Transactions on Circuits and Systems–I: Regular Papers*, vol. 55, no. 1, pp. 336-346, Feb. 2008.
2. Yajuan He and Chip-Hong Chang, "A New Redundant Binary Booth Encoding for Fast 2^n -bit Multiplier Design," *IEEE Transactions on Circuits and Systems–I: Regular Papers*, submitted for review as a regular paper.
3. Yajuan He and Chip-Hong Chang, "A New Insight into Redundant Binary Booth Multipliers: Architectural Exploration and Energy Efficiency Evaluation," *IET Circuits, Devices and Systems*, submitted for review as a regular paper.

Conference Papers

1. Chip-Hong Chang, Yajuan He, and Jiangmin Gu, "An alternative scheme of redundant binary multiplier," in *Proc. 2004 IEEE Asia-Pacific Conference*

- on Circuits and Systems (APCCAS)*, Tainan, Taiwan, R.O.C., Dec. 6–9, 2004, pp. 33–36.
2. Yajuan He, Chip-Hong Chang, and Jiangmin Gu, “An area efficient 64-bit square root carry-select adder for low power applications,” in *Proc. 2005 IEEE International Symposium on Circuits and Systems (ISCAS)*, Kobe, Japan, May 23–26, 2005, vol. 4, pp. 4082–4085. (Receipt of Student Paper Contest for Travel Support)
 3. Yajuan He, Chip-Hong Chang, Jiangmin Gu, and Hossam A. H. Fahmay, “A novel covalent redundant binary Booth encoder,” in *Proc. 2005 IEEE International Symposium on Circuits and Systems (ISCAS)*, Kobe, Japan, May 23–26, 2005, vol. 1, pp. 69–72.
 4. Yajuan He and Chip-Hong Chang, “A Low-power High-speed RB-to-NB Converter for Fast Redundant Binary Multiplier,” in *Proc. 2006 IEEE International Symposium on Circuits and Systems (ISCAS)*, Kos, Greece, May 21–24, 2006, pp. 2405–2408.

Bibliography

- [1] R. Bagheri, A. Mirzaei, M. Heidari, S. Chehrazi, M. Lee, M. Mikhemar, W. Tang, and A. Abidi, "Software-defined radio receiver: dream to reality," *IEEE Communications Magazine*, vol. 44, no. 8, pp. 111 – 118, Aug. 2006.
- [2] A. A. Abidi, "The path to the software-defined radio receiver," *IEEE J. Solid-State Circuits*, vol. 42, no. 5, pp. 954–966, May 2007.
- [3] B. Krenik, "Cellular handset evolution - convergence of high-speed data services," in *2004 IEEE Radio Frequency Integrated Circuits (RFIC) Symposium*, Jun. 2004, p. 6.
- [4] H.-C. Chow and I.-C. Wey, "A 3.3V 1 GHz high speed pipelined Booth multiplier," in *Proc. 2002 IEEE Int. Symp. Circuits Syst. (ISCAS'2002)*, vol. 1, Arizona, USA, May 2002, pp. 457–460.
- [5] H. Edamatsu, T. Taniguchi, T. Nishiyama, and S. Kuninobu, "A 33 MFLOPS floating point processor using redundant binary representation," in *1988 IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, USA, Feb. 1988, pp. 152–153, 342–343.
- [6] J. Gu, C. H. Chang, and K. S. Yeo, "Algorithm and architecture of a high density, low power scalar product macrocell," *IEE Proceedings Computers and Digital Techniques*, vol. 151, no. 2, pp. 161–172, Mar. 2004.
- [7] B. Parhami, *Computer Arithmetic Algorithms And Hardware Designs*. New York: Oxford University Press, 2000.
- [8] J. M. Rabaey, *Digital Integrated Circuits - A design perspective*. Prentice Hall Press, 2001.

- [9] M. Tonomura, "High-speed digital circuit of discrete cosine transform," *IEICE Trans. Fundamentals*, vol. E78-A, no. 8, pp. 1342–1350, Aug. 1995.
- [10] Z. Yu, M. L. Yu, K. Azader, and A. N. Willson, Jr., "A low power adaptive filter using dynamic reduced 2's-complement representation," in *Proc. 2002 IEEE Custom Integrated Circuit Conf. (CICC'2002)*, Orlando, FL, May 2002, pp. 141–144.
- [11] H. Sakamoto, H. Ochi, K. Uda, K. Taki, B.-Y. Lee, and T. Tsuda, "A 16-bit redundant binary multiplier using low-power pass-transistor logic SPL," in *Proc. ASP-DAC 2000 Asia South Pacific Design Automation Conference*, vol. 1, Yokohama, Japan, Jan. 2000, pp. 33–34.
- [12] N. Itoh, Y. Naemura, H. Makino, Y. Nakase, T. Yoshihara, and Y. Horiba, "A 600-MHz 54×54 -bit multiplier with rectangular-styled Wallace tree," *IEEE J. Solid-State Circuits*, vol. 36, no. 2, pp. 249–257, 2001.
- [13] Y. Kim, B.-S. Song, J. Grosspietsch, and S. F. Gillig, "A carry-free $54b \times 54b$ multiplier using equivalent bit conversion algorithm," *IEEE J. Solid-State Circuits*, vol. 36, no. 10, pp. 1538–1545, Oct. 2001.
- [14] S.-H. Lee, S.-J. Bae, and H.-J. Park, "A compact radix-64 54×54 CMOS redundant binary parallel multiplier," *IEICE Trans. Electron.*, vol. E85-C, no. 6, pp. 1342–1350, Jun. 2002.
- [15] Y. He, C. H. Chang, J. Gu, and H. A. H. Fahmy, "A novel covalent redundant binary Booth encoder," in *Proc. 2005 IEEE Int. Symp. Circuits Syst. (ISCAS'2005)*, vol. 1, Kobe, Japan, May 2005, pp. 69–72.
- [16] J.-Y. Kang and J.-L. Gaudiot, "A simple high-speed multiplier design," *IEEE Trans. Computers*, vol. 55, no. 10, pp. 1253–1258, Oct. 2006.
- [17] C. Nagendra, M. J. Irwin, and R. M. Owens, "Area-time-power tradeoffs in parallel adders," *IEEE Trans. Circuits Syst.-II: Analog and Digital Signal Processing*, vol. 43, no. 10, pp. 689–702, Oct. 1996.
- [18] R. Gonzalez and M. Horowitz, "Energy dissipation in general purpose microprocessors," *IEEE J. Solid-State Circuits*, vol. 31, no. 9, pp. 1277–1284, Sept. 1996.

- [19] R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and threshold voltage scaling for low power CMOS," *IEEE J. Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, Aug. 1997.
- [20] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. Computers*, vol. 22, no. 12, pp. 1045–1047, 1973.
- [21] P. Bonatto and V. Oklobdzija, "Evaluation of Booth's algorithm for implementation in parallel multipliers," in *Proc. 29th IEEE Asilomar Conf. Signals, Syst., Computers (ACSSC)*, vol. 1, Pacific Grove, CA, USA, Nov. 1996, pp. 608–610.
- [22] Y. Hagihara, S. Inui, A. Yoshikawa, S. Nakazato, S. Iriki, R. Ikeda, Y. Shibue, T. Inaba, M. Kagamihara, and M. Yamashina, "A 2.7-ns 0.25- μ m CMOS 54 $\times$ 54-b multiplier," in *1998 IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, vol. 41, Feb. 1998, pp. 296–297.
- [23] S. F. Hsiao, M. R. Jiang, and J. S. Yeh, "Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers," *Electron. Lett.*, vol. 34, no. 4, pp. 341–343, 1998.
- [24] K.-Y. Khoo, Z. Yu, and A. N. Willson, Jr., "Improved-Booth encoding for low-power multipliers," in *Proc. 1999 IEEE Int. Symp. Circuits Syst. (ISCAS'1999)*, vol. 1, San Diego, CA, USA, 1999, pp. 62–65.
- [25] M. Nagamatsu, S. Tanaka, J. Mori, T. Noguchi, and K. Hatanaka, "A 15-ns 32 $\times$ 32-b CMOS multiplier with an improved parallel structure," *IEEE J. Solid-State Circuits*, vol. 25, no. 2, pp. 494–497, Apr. 1990.
- [26] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome, "A 4.4 ns CMOS 54 $\times$ 54-b multiplier using pass-transistor multiplexer," *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 251–257, Mar. 1995.
- [27] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," *IEEE Trans. Computers*, vol. 45, no. 3, pp. 294–305, Mar. 1996.

- [28] Z. Wang, G. Jullien, and W. C. Miller, "A new design technique for column compression multipliers," *IEEE Trans. Computers*, vol. 44, no. 8, pp. 962–970, Aug. 1995.
- [29] M. Margala and N. G. Durdle, "Low-power low-voltage 4-2 compressors for VLSI applications," in *Proc. IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, Como, Italy, 1999, pp. 84–90.
- [30] K. Prasad and K. K. Parhi, "Low-power 4-2 and 5-2 compressors," in *Proc. 35th IEEE Asilomar Conf. Signals, Syst., Computers (ACSSC)*, vol. 1, Pacific Grove, CA, USA, Nov. 2001, pp. 129–133.
- [31] D. Radhakrishnan and A. Preethy, "Low power CMOS pass logic 4-2 compressor for high-speed multiplication," in *Proc. 43th IEEE Midwest Symp. Circuits Syst. (MWSCAS'2000)*, vol. 3, Lansing MI, Aug. 2000, pp. 1296–1298.
- [32] K.-W. Shin and B.-S. Song, "A complex multiplier architecture based on redundant binary arithmetic," in *Proc. 1997 IEEE Int. Symp. Circuits Syst. (ISCAS'1997)*, vol. 3, Hong Kong, China, Jun. 1997, pp. 1944–1947.
- [33] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "A high-speed multiplier using a redundant binary adder tree," *IEEE J. Solid-State Circuits*, vol. 22, no. 1, pp. 28–34, Feb. 1987.
- [34] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Mashiko, "An 8.8-ns 54×54 -bit multiplier with high speed redundant binary architecture," *IEEE J. Solid-State Circuits*, vol. 31, no. 6, pp. 773–783, Jun. 1996.
- [35] N. Besli and R. Deshmukh, "A 54×54 bit multiplier with a new redundant binary Booth's encoding," in *Proc. IEEE Conf. CCECE*, vol. 2, May 2002, pp. 597–602.
- [36] N. Besli and R. G. Deshmukh, "A novel redundant binary signed-digit (RBSD) Booth's encoding," in *Proc. IEEE Southeast Conference*, Columbia, South Carolina, USA, Apr. 2002, pp. 426–431.

- [37] O. T.-C. Chen, L.-H. Chen, N.-W. Lin, and C.-C. Chen, "Application-specific data path for highly efficient computation of multistandard video codecs," *IEEE Trans. Circuits Syst. Video Techno.*, vol. 17, no. 1, pp. 26–42, Jan. 2007.
- [38] S. Perri, P. Corsonello, and G. Cocorullo, "A 64-bit reconfigurable adder for low power media processing," *Electron. Lett.*, vol. 38, no. 9, pp. 397–399, Apr. 2002.
- [39] C. Mead and L. A. Conway, *Introduction to VLSI Systems*. Reading, MA: Addison-Wesley, 1980.
- [40] K. Z. Pekmestzi, P. Kalivas, N. Moshopoulos, and J. Sifnaios, "Complex constant number serial multipliers," in *Proc. IEE Circuits, Devices and Systems*, vol. 150, no. 5, Oct. 2003, pp. 405–410.
- [41] S. Lu and J. Kenney, "Design of most-significant-bit-first serial multiplier," *Electronics Letters*, vol. 31, no. 14, pp. 1133–1135, Jul. 1995.
- [42] Y. Chang, J. H. Satyanarayana, and K. K. Parhi, "Low-power digit-serial multipliers," in *Proc. 1997 IEEE Int. Symp. Circuits Syst. (ISCAS'1997)*, vol. 3, Hong Kong, China, Jun. 1997, pp. 2164–2167.
- [43] L. Fanucci and M. Forliti, "Interlaced diagonal-wise pipelined serial multiplier," *Electronics Letters*, vol. 36, no. 21, pp. 1824–1825, Oct. 2000.
- [44] M. Mehta, V. Parmar, and E. E. Swartzlander, Jr., "High-speed multiplier design using multi-input counter and compressor circuits," in *Proc. 10th IEEE Symp. Computer Arithmetic*, Jun. 1991, pp. 43–50.
- [45] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Computers*, vol. 13, no. 2, pp. 14–17, 1964.
- [46] E. L. Braun, *Digital Computer Design, Logic Circuitry, Synthesis*. New York: Academic Press, 1963.
- [47] P. J. Song and G. D. Micheli, "Circuit and architecture trade-off for high-speed multiplication," *IEEE J. Solid-State Circuits*, vol. 26, pp. 1184–1198, Apr. 1991.

- [48] A. D. Booth, "A signed binary multiplication technique," *Quarterly J. of Mechanics and Applied Maths.*, vol. 4, pp. 236–240, Jun. 1951.
- [49] O. L. MacSorley, "High-speed arithmetic in binary computers," *IRE Proceedings*, vol. 49, pp. 67–91, Jan. 1961.
- [50] L. Dadda, "Some schemes for parallel multiplier," *Alta Frequenza*, vol. 34, pp. 349–356, May 1965.
- [51] N. Takagi, H. Yasuura, and S. Yajima, "High-speed VLSI multiplication algorithm with a redundant binary addition tree," *IEEE Trans. Computers*, vol. C-34, no. 9, pp. 789–796, Sept. 1985.
- [52] A. Weinberger, "4-2 carry-save adder module," *IBM Tech. Disclosure Bulletin*, vol. 23, Jan. 1981.
- [53] D. Villeger and V. G. Oklobdzija, "Analysis of Booth encoding efficiency in parallel multipliers using compressors for reduction of partial products," in *Proc. 27th IEEE Asilomar Conf. Signals, Syst., Computers (ACSSC)*, vol. 1, Pacific Grove, CA, USA, Nov. 1993, pp. 781–784.
- [54] B. Millar, P. E. Madrid, and E. E. Swartzlander, Jr., "A fast hybrid multiplier combining Booth and Wallace/Dadda algorithms," in *Proc. 35th IEEE Midwest Symp. Circuits Syst. (MWSCAS'1992)*, vol. 1, Washington DC, Aug. 1992, pp. 158–165.
- [55] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Computers*, vol. EC-10, pp. 389–400, Sept. 1961.
- [56] N. Takagi, "A high-speed multiplier with a regular cellular array structure using redundant binary representation," Yajima Lab., Dep. Inform. Sci., Kyoto Univ, Kyoto, Japan, Tech. Rep. R82-14, Jun. 1982.
- [57] G. W. Bewick, "Fast multiplication: algorithms and implementation," Ph.D. dissertation, Stanford University, Feb. 1994.
- [58] S. Kuninobu, T. Nishiyama, H. Edamatsu, T. Taniguchi, and N. Takagi, "Design of high-speed MOS multiplier and divider using redundant bi-

- nary representation," in *Proc. 8th IEEE Symp. Computer Arithmetic*, May 1987, pp. 80–86.
- [59] H. Makino, Y. Nakase, and H. Shinohara, "An 8.8-ns 54×54-bit multiplier using new redundant binary architecture," in *1993 IEEE Int. Conf. Computer Design (ICCD'1993)*, Cambridge, MA, Oct. 1993, pp. 202–205.
- [60] S. Kuninobu, T. Nishiyama, and T. Taniguchi, "High speed MOS multiplier and divider using redundant binary representation and their implementation in a microprocessor," *IEICE Trans. Electron.*, vol. E76-C, no. 3, pp. 436–445, Mar. 1993.
- [61] K. Hwang, *Computer Arithmetic, Principles, Architecture, and Design*. New York: Wiley, 1979.
- [62] G. M. Blair, "The equivalence of twos-complement addition and the conversion of redundant-binary to twos-complement numbers," *IEEE Trans. Circuits Syst.-I: Regular Papers*, vol. 45, pp. 669–671, Jun. 1998.
- [63] S. M. Yen, C. S. Lai, C. H. Chen, and J. Y. Lee, "An efficient redundant-binary number to binary number converter," *IEEE J. Solid-State Circuits*, vol. 27, no. 1, pp. 109–112, Jan. 1992.
- [64] H. R. Srinivas and K. K. Parhi, "A fast VLSI adder architecture," *IEEE J. Solid-State Circuits*, vol. 27, no. 5, pp. 761–767, May 1992.
- [65] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, "High speed multiplier LSI using a redundant binary adder tree," in *1984 IEEE Int. Conf. Computer Design (ICCD'1984)*, Oct. 1984.
- [66] H. Makino, H. Suzuki, H. Morinaka, Y. Nakase, K. Mashiko, and T. Sumi, "A 286 MHz 64-b floating point multiplier with enhanced CG operation," *IEEE J. Solid-State Circuits*, vol. 31, no. 4, pp. 504–513, Apr. 1996.
- [67] K.-W. Shin, B.-S. Song, and K. Bacrania, "A 200-MHz complex number multiplier using redundant binary arithmetic," *IEEE J. Solid-State Circuits*, vol. 33, no. 6, pp. 1538–1545, Jun. 1998.

- [68] M. D. Ercegovac and T. Lang, "Comments on 'a carry-free $54b \times 54b$ multiplier using equivalent bit conversion algorithm'," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 160–161, Jan. 2003.
- [69] W. Rulling, "A remark on carry-free binary multiplication," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 159–160, Jan. 2003.
- [70] I. Choo and R. G. Deshmukh, "A novel conversion scheme from a redundant binary number to two's complement binary number for parallel architectures," in *Proc. IEEE Southeast Conf.*, vol. 2, Clemson, South Carolina, USA, Apr. 2001, pp. 196–201.
- [71] N. Slingerland and A. J. Smith, "Measuring the performance of multimedia instruction sets," *IEEE Trans. Computers*, vol. 51, no. 11, pp. 1317–1332, 2002.
- [72] A. Beaumont-Smith, J. Tsimbinos, C. C. Lim, and W. Marwood, "A VLSI chip implementation of an A/D converter error table compensator," *Computer Standard & Interfaces*, vol. 23, pp. 111–122, 2001.
- [73] C. Shi, W. Wang, L. Zhou, L. Gao, P. Liu, and Q. Yao, "32b RISC/DSP media processor: MediaDSP3201," in *SPIE Embedded Processors for Multimedia and Communications II*, vol. 5683, San Jose, USA, Mar. 2005, pp. 43–52.
- [74] M. Katona, A. Pizurica, N. Teslic, V. Kovacevic, and W. Philips, "A real-time wavelet-domain video denoising implementation in FPGA," *EURASIP J. Embedded Syst.*, pp. 1–12, 2006.
- [75] N. Quach and M. J. Flynn, "High speed addition in CMOS," *IEEE Trans. Computers*, vol. 41, no. 12, pp. 1612–1615, Dec. 1992.
- [76] I. Sutherland, R. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann, 1999.
- [77] H. Q. Dao and V. Oklobdzija, "Application of logical effort on delay analysis of 64-bit static carry-lookahead adder," in *Proc. 35th IEEE Asilomar Conf. Signals, Syst., Computers (ACSSC)*, vol. 2, Pacific Grove, CA, USA, Nov. 2001, pp. 1322–1324.

- [78] D. Harris and I. Sutherland, "Logical effort of carry propagate adders," in *Proc. 37th IEEE Asilomar Conf. Signals, Syst., Computers (ACSSC)*, vol. 1, Pacific Grove, CA, USA, Nov. 2003, pp. 873–878.
- [79] V. G. Oklobdzija, B. R. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energy-delay estimation technique for high-performance microprocessor VLSI adders," in *Proc. 16th IEEE Symp. Computer Arithmetic (ARITH)*, Santiago de Compostela, Spain, Jun. 2003, pp. 272–279.
- [80] M. Sayed and W. Badawy, "Performance analysis of single bit full adder cells using 0.18, 0.25 and 0.35 μm CMOS technologies," in *Proc. 2002 IEEE Int. Symp. Circuits Syst. (ISCAS'2002)*, Scottsdale, Arizona, USA, May 2002, pp. 559–562.
- [81] *Star-Hspice Manual Release*, Synopsys, Inc., 2004.
- [82] *TSMC 0.18 μm Process 1.8-Volt SAGE-XTM Standard Cell Library Databook*, Artisan Components, Inc., Oct. 2001.
- [83] *Design Compiler User Guide*, Synopsys, Inc. 2003.
- [84] A. Hald, *Statistical Theory with Engineering Applications*. New York: Wiley, 1952.
- [85] R. Burch, F. N. Najm, P. Yang, and T. N. Trick, "A Monte Carlo approach for power estimation," *IEEE Trans. VLSI Syst.*, vol. 1, no. 1, pp. 63–71, Mar. 1993.
- [86] C. Nagendra, R. M. Owens, and M. J. Irwin, "Power-delay characteristics of CMOS adders," *IEEE Trans. VLSI Syst.*, vol. 2, no. 3, pp. 377–381, Sept. 1994.
- [87] M. Xakellis and F. Najm, "Statistical estimation of the switching activity in digital circuits," in *Proc. 31st IEEE Design Automation Conf.*, Oct. 1994, pp. 728–733.
- [88] C.-S. Ding, C.-T. Hsieh, and M. Pedram, "Improving the efficiency of Monte Carlo power estimation," *IEEE Trans. VLSI Syst.*, vol. 8, no. 5, pp. 584–593, Oct. 2000.

- [89] T. Lynch and E. E. Swartzlander, Jr., "A spanning tree carry lookahead adder," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 931–939, Aug. 1992.
- [90] V. Kantabutra, "A recursive carry-lookahead/carry-select hybrid adder," *IEEE Trans. Computers*, vol. 42, no. 12, pp. 1495–1499, Dec. 1993.
- [91] Y. Wang, C. Pai, and X. Song, "The design of hybrid carry-lookahead/carry-select adders," *IEEE Trans. Circuits Syst.–II: Analog and Digital Signal Processing*, vol. 49, no. 1, pp. 16–24, Jan. 2002.
- [92] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Computers*, vol. 22, no. 8, pp. 786–793, Aug. 1973.
- [93] Y. He, C. H. Chang, and J. Gu, "An area efficient 64-bit square root carry-select adder for low power applications," in *Proc. 2005 IEEE Int. Symp. Circuits Syst. (ISCAS'2005)*, vol. 4, Kobe, Japan, May 2005, pp. 4082–4085.
- [94] Y. He and C. H. Chang, "A low-power high-speed RB-to-NB converter for fast redundant binary multiplier," in *Proc. 2006 IEEE Int. Symp. Circuits Syst. (ISCAS'2006)*, Kos, Greece, May 2006, pp. 2405–2408.
- [95] T. P. Kelliher, R. M. Owens, M. J. Irwin, and T.-T. Hwang, "ELM – a fast addition algorithm discovered by a program," *IBM J. Research Development*, vol. 41, no. 9, pp. 1181–1184, Sep. 1992.
- [96] R. P. Brent and H. T. Kung, "A regular layout for parallel adders," *IEEE Trans. Computers*, vol. C-31, pp. 260–264, Mar. 1982.
- [97] H. Ling, "High speed binary adder," *IBM J. Research Development*, pp. 156–166, May 1981.
- [98] A. Neve, H. Schettler, T. Ludwig, and D. Flandre, "Power-delay product minimization in high-performance 64-bit carry-select adders," *IEEE Trans. VLSI Syst.*, vol. 12, no. 3, pp. 235–244, Mar. 2004.
- [99] M. Alioto and G. Palumbo, "Analysis and comparison on full adder block in submicron technology," *IEEE Trans. VLSI Syst.*, vol. 10, no. 6, pp. 806–823, Dec. 2002.

- [100] A. Bellaouar and M. I. Elmasry, *Low-Power Digital VLSI Design: Circuits and Systems*. Kluwer Academic Publishers, 1995.
- [101] T. Y. Chang and M. J. Hsiao, "Carry-select adder using single ripple-carry adder," *Electronics Letters*, vol. 34, no. 22, pp. 2101–2103, Oct. 1998.
- [102] Y. Kim and L. S. Kim, "64-bit carry-select adder with reduced area," *Electronics Letters*, vol. 37, no. 10, pp. 614–615, May 2001.
- [103] C. H. Chang, J. Gu, and M. Zhang, "Ultra low voltage, low power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," *IEEE Trans. Circuits Syst.–I: Regular Papers*, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.
- [104] —, "A review of 0.18- μ m full adder performances for tree structured arithmetic circuits," *IEEE Trans. VLSI Syst.*, vol. 13, no. 6, pp. 686–695, Jun. 2005.
- [105] W. L. Gallagher and E. E. Swartzlander, Jr., "High radix Booth multipliers using reduced area adder trees," in *Proc. 28th IEEE Asilomar Conf. Signals, Syst., Computers (ACSSC)*, vol. 1, Pacific Grove, CA, USA, Nov. 1994, pp. 545–549.
- [106] *ModelSim User's Manual*, Mentor Graphics, Inc., 2004.
- [107] *Power Compiler Reference Manual*, Synopsys, Inc., Jun. 2004.
- [108] Y. Kim, B.-S. Song, J. Grosspietsch, and S. F. Gillig, "Correction to 'a carry-free $54b \times 54b$ multiplier using equivalent bit conversion algorithm'," *IEEE J. Solid-State Circuits*, vol. 38, no. 1, p. 159, Jan. 2003.
- [109] A. Shams, T. Darwish, and M. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells," *IEEE Trans. VLSI Syst.*, vol. 10, pp. 20–29, 1 2002.
- [110] R. Fried, "Minimizing energy dissipation in high-speed multipliers," in *Proc. 1997 IEEE Int. Symp. Low Power Electronics and Design*, 1997, pp. 214–219.
- [111] W.-C. Yeh and C.-W. Jen, "High-speed Booth encoded parallel multiplier design," *IEEE Trans. Computers*, vol. 49, no. 7, pp. 692–701, Jul. 2000.