

## Contents

**Preface** *xiii*

**About the Editors** *xvii*

<b>1</b>	<b>Basic Arithmetic Circuits</b>	<b>1</b>
	<i>Oscar Gustafsson and Lars Wanhammar</i>	
1.1	Introduction	1
1.2	Addition and Subtraction	1
1.2.1	Ripple-Carry Addition	2
1.2.2	Bit-Serial Addition and Subtraction	3
1.2.3	Digit-Serial Addition and Subtraction	4
1.3	Multiplication	4
1.3.1	Partial Product Generation	5
1.3.2	Avoiding Sign-Extension (the Baugh and Wooley Method)	6
1.3.3	Reducing the Number of Partial Products	6
1.3.4	Reducing the Number of Columns	8
1.3.5	Accumulation Structures	8
1.3.5.1	Add-and-Shift Accumulation	8
1.3.5.2	Array Accumulation	9
1.3.5.3	Tree Accumulation	9
1.3.5.4	Vector-Merging Adder	11
1.3.6	Serial/Parallel Multiplication	11
1.4	Sum-of-Products Circuits	15
1.4.1	SOP Computation	17
1.4.2	Linear-Phase FIR Filters	18
1.4.3	Polynomial Evaluation (Horner's Method)	18
1.4.4	Multiple-Wordlength SOP	18
1.5	Squaring	19
1.5.1	Parallel Squarers	19
1.5.2	Serial Squarers	21
1.5.3	Multiplication Through Squaring	23
1.6	Complex Multiplication	24

1.6.1	Complex Multiplication Using Real Multipliers	24
1.6.2	Lifting-Based Complex Multipliers	25
1.7	Special Functions	26
1.7.1	Square Root Computation	26
1.7.1.1	Digit-Wise Iterative Square Root Computation	27
1.7.1.2	Iterative Refinement Square Root Computation	27
1.7.2	Polynomial and Piecewise Polynomial Approximations	28
	References	30
<b>2</b>	<b>Shift-Add Circuits for Constant Multiplications</b>	<b>33</b>
	<i>Parmod Kumar Meher, C.-H. Chang, Oscar Gustafsson, A.P. Vinod, and M. Faust</i>	
2.1	Introduction	33
2.2	Representation of Constants	36
2.3	Single Constant Multiplication	40
2.3.1	Direct Simplification from a Given Number Representation	40
2.3.2	Simplification by Redundant Signed Digit Representation	41
2.3.3	Simplification by Adder Graph Approach	41
2.3.4	State of the Art in SCM	43
2.4	Algorithms for Multiple Constant Multiplications	43
2.4.1	MCM for FIR Digital Filter and Basic Considerations	43
2.4.2	The Adder Graph Approach	45
2.4.2.1	The Early Developments	45
2.4.2.2	$n$ -Dimensional Reduced Adder Graph (RAG- $n$ ) Algorithm	46
2.4.2.3	Maximum/Cumulative Benefit Heuristic Algorithm	47
2.4.2.4	Minimization of Logic Depth	47
2.4.2.5	Illustration and Comparison of AG Approaches	48
2.4.3	Common Subexpression Elimination Algorithms	49
2.4.3.1	Basic CSE Techniques in CSD Representation	51
2.4.3.2	Multidirectional Pattern Search and Elimination	52
2.4.3.3	CSE in Generalized Radix-2 SD Representation	53
2.4.3.4	The Contention Resolution Algorithm	54
2.4.3.5	Examples and Comparison of MCM Approaches	54
2.4.4	Difference Algorithms	56
2.4.5	Reconfigurable and Time-Multiplexed Multiple Constant Multiplications	56
2.5	Optimization Schemes and Optimal Algorithms	58
2.5.1	Optimal Subexpression Sharing	58
2.5.2	Representation Independent Formulations	60
2.6	Applications	62
2.6.1	Implementation of FIR Digital Filters and Filter Banks	62
2.6.2	Implementation of Sinusoidal and Other Linear Transforms	63

2.6.3	Other Applications	63
2.7	Pitfalls and Scope for Future Work	64
2.7.1	Selection of Figure of Merit	64
2.7.2	Benchmark Suites for Algorithm Evaluation	65
2.7.3	FPGA-Oriented Design of Algorithms and Architectures	65
2.8	Conclusions	66
	References	67
<b>3</b>	<b>DA-Based Circuits for Inner-Product Computation</b>	<b>77</b>
	<i>Mahesh Mehendale, Mohit Sharma, and Pramod Kumar Meher</i>	
3.1	Introduction	77
3.2	Mathematical Foundation and Concepts	78
3.3	Techniques for Area Optimization of DA-Based Implementations	81
3.3.1	Offset Binary Coding	81
3.3.2	Adder-Based DA	85
3.3.3	Coefficient Partitioning	85
3.3.4	Exploiting Coefficient Symmetry	87
3.3.5	LUT Implementation Optimization	88
3.3.6	Adder-Based DA Implementation with Single Adder	90
3.3.7	LUT Optimization for Fixed Coefficients	90
3.3.8	Inner-Product with Data and Coefficients Represented as Complex Numbers	92
3.4	Techniques for Performance Optimization of DA-Based Implementations	93
3.4.1	Two-Bits-at-a-Time (2-BAAT) Access	93
3.4.2	Coefficient Distribution over Data	93
3.4.3	RNS-Based Implementation	95
3.5	Techniques for Low Power and Reconfigurable Realization of DA-Based Implementations	98
3.5.1	Adder-Based DA with Fixed Coefficients	99
3.5.2	Eliminating Redundant LUT Accesses and Additions	100
3.5.3	Using Zero-Detection to Reduce LUT Accesses and Additions	102
3.5.4	Nega-Binary Coding for Reducing Input Toggles and LUT Look-Ups	103
3.5.5	Accuracy versus Power Tradeoff	107
3.5.6	Reconfigurable DA-Based Implementations	108
3.6	Conclusion	108
	References	109
<b>4</b>	<b>Table-Based Circuits for DSP Applications</b>	<b>113</b>
	<i>Pramod Kumar Meher and Shen-Fu Hsiao</i>	
4.1	Introduction	113
4.2	LUT Design for Implementation of Boolean Function	115

4.3	Lookup Table Design for Constant Multiplication	117
4.3.1	Lookup Table Optimizations for Constant Multiplication	117
4.3.1.1	Antisymmetric Product Coding for LUT Optimization	119
4.3.1.2	Odd Multiple-Storage for LUT Optimization	119
4.3.2	Implementation of LUT-Multiplier using APC for $L = 5$	122
4.3.3	Implementation of Optimized LUT using OMS Technique	123
4.3.4	Optimized LUT Design for Signed and Unsigned Operands	124
4.3.5	Input Operand Decomposition for Large Input Width	126
4.4	Evaluation of Elementary Arithmetic Functions	127
4.4.1	Piecewise Polynomial Approximation (PPA) Approach for Function Evaluation	128
4.4.1.1	Accuracy of PPA Method	129
4.4.1.2	Input Interval Partitioning for PPA	130
4.4.2	Table-Addition (TA) Approach for Function Evaluation	131
4.4.2.1	Bipartite Table-Addition Approach for Function Evaluation	131
4.4.2.2	Accuracy of TA Method	132
4.4.2.3	Multipartite Table-Addition Approach for Function Evaluation	134
4.5	Applications	134
4.5.1	LUT-Based Implementation of Cyclic Convolution and Orthogonal Transforms	135
4.5.2	LUT-Based Evaluation of Reciprocals and Division Operation	136
4.5.2.1	Mathematical Formulation	136
4.5.2.2	LUT-Based Structure for Evaluation of Reciprocal	137
4.5.2.3	Reduction of LUT Size	137
4.5.3	LUT-Based Design for Evaluation of Sigmoid Function	138
4.5.3.1	LUT Optimization Strategy using Properties of Sigmoid Function	139
4.5.3.2	Design Example for $\epsilon = 0.2$	141
4.5.3.3	Implementation of Design Example for $\epsilon = 0.2$	142
4.6	Summary	143
	References	145
<b>5</b>	<b>CORDIC Circuits</b>	149
	<i>Pramod Kumar Meher, Javier Valls, Tso-Bing Juang, K. Sridharan, and Koushik Maharatna</i>	
5.1	Introduction	149
5.2	Basic CORDIC Techniques	151
5.2.1	The CORDIC Algorithm	151
5.2.1.1	Iterative Decomposition of Angle of Rotation	152
5.2.1.2	Avoidance of Scaling	153

5.2.2	Generalization of the CORDIC Algorithm	154
5.2.3	Multidimensional CORDIC	155
5.3	Advanced CORDIC Algorithms and Architectures	156
5.3.1	High-Radix CORDIC Algorithm	157
5.3.2	Angle Recoding Methods	158
5.3.2.1	Elementary Angle Set Recoding	158
5.3.2.2	Extended Elementary Angle Set Recoding	159
5.3.2.3	Parallel Angle Recoding	159
5.3.3	Hybrid or Coarse-Fine Rotation CORDIC	161
5.3.3.1	Coarse-Fine Angular Decomposition	161
5.3.3.2	Implementation of Hybrid CORDIC	162
5.3.3.3	Shift-Add Implementation of Coarse Rotation	163
5.3.3.4	Parallel CORDIC-Based on Coarse-Fine Decomposition	164
5.3.4	Redundant Number-Based CORDIC Implementation	164
5.3.5	Pipelined CORDIC Architecture	166
5.3.6	Differential CORDIC Algorithm	167
5.4	Scaling, Quantization, and Accuracy Issues	168
5.4.1	Implementation of Mixed-Scaling Rotation	168
5.4.2	Low-Complexity Scaling	169
5.4.3	Quantization and Numerical Accuracy	170
5.4.4	Area–Delay–Accuracy Trade-off	170
5.5	Applications of CORDIC	172
5.5.1	Matrix Computation	172
5.5.1.1	QR Decomposition	172
5.5.1.2	Singular Value Decomposition and Eigenvalue Estimation	173
5.5.2	Signal Processing and Image Processing Applications	173
5.5.3	Applications to Communication	174
5.5.3.1	Direct Digital Synthesis	175
5.5.3.2	Analog and Digital Modulation	175
5.5.3.3	Other Communication Applications	176
5.5.4	Applications of CORDIC to Robotics and Graphics	176
5.5.4.1	Direct Kinematics Solution (DKS) for Serial Robot Manipulators	176
5.5.4.2	Inverse Kinematics for Robot Manipulators	177
5.5.4.3	CORDIC for Other Robotics Applications	177
5.5.4.4	CORDIC for 3D Graphics	177
5.6	Conclusions	178
	References	178
<b>6</b>	<b>RNS-Based Arithmetic Circuits and Applications</b>	<b>186</b>
	<i>P.V. Ananda Mohan</i>	
6.1	Introduction	186

6.2	Modulo Addition and Subtraction	189
6.2.1	Modulo $(2^n - 1)$ Adders	189
6.2.2	Modulo $(2^n + 1)$ Adders	191
6.3	Modulo Multiplication and Modulo Squaring	193
6.3.1	Multipliers for General Moduli	194
6.3.2	Multipliers mod $(2^n - 1)$	195
6.3.3	Multipliers mod $(2^n + 1)$	196
6.3.4	Modulo Squarers	199
6.4	Forward (binary to RNS) Conversion	200
6.5	RNS to Binary Conversion	203
6.5.1	CRT-Based RNS to Binary Conversion	203
6.5.2	Mixed Radix Conversion	206
6.5.3	RNS to Binary conversion using New CRT	207
6.5.4	RNS to Binary conversion using Core Function	208
6.6	Scaling and Base Extension	210
6.7	Magnitude Comparison and Sign Detection	213
6.8	Error Correction and Detection	214
6.9	Applications of RNS	216
6.9.1	FIR Filters	216
6.9.2	RNS in Cryptography	218
6.9.2.1	Montgomery Modular Multiplication	218
6.9.2.2	Elliptic Curve Cryptography using RNS	221
6.9.2.3	Pairing processors using RNS	223
6.9.3	RNS in Digital Communication Systems	225
	References	226
<b>7</b>	<b>Logarithmic Number System</b>	<b>237</b>
	<i>Vassilis Paliouras and Thanos Stouraitis</i>	
7.1	Introduction	237
7.1.1	The Logarithmic Number System	237
7.1.2	Organization of the Chapter	237
7.2	Basics of LNS Representation	238
7.2.1	LNS and Equivalence to Linear Representation	238
7.3	Fundamental Arithmetic Operations	240
7.3.1	Multiplication, Division, Roots, and Powers	240
7.3.2	Addition and Subtraction	241
7.3.2.1	Direct Techniques and Approximations	242
7.3.2.2	Cancellation of Singularities	242
7.4	Forward and Inverse Conversion	249
7.5	Complex Arithmetic in LNS	250
7.6	LNS Processors	251
7.6.1	A VLIW LNS Processor	252
7.6.1.1	The Architecture	253

7.6.1.2	The Arithmetic Logic Units	253
7.6.1.3	The Register Set	256
7.6.1.4	Memory Organization	256
7.6.1.5	Applications and the Proposed Architecture	257
7.6.1.6	Performance and Comparisons	257
7.7	LNS for Low-Power Dissipation	257
7.7.1	Impact of LNS Encoding on Signal Activity	258
7.7.2	Power Dissipation and LNS Architecture	261
7.8	Applications	265
7.8.1	Signal Processing and Communications	265
7.8.2	Video Processing	267
7.8.3	Graphics	268
7.9	Conclusions	268
	References	269
<b>8</b>	<b>Redundant Number System-Based Arithmetic Circuits</b>	<b>273</b>
	<i>G. Jaberipur</i>	
8.1	Introduction	273
8.1.1	Introductory Definitions and Examples	274
8.1.1.1	Posibits and Negabits	275
8.2	Fundamentals of Redundant Number Systems	278
8.2.1	Redundant Digit Sets	278
8.3	Redundant Number Systems	280
8.3.1	Constant Time Addition	281
8.3.2	Carry-Save Addition	283
8.3.3	Borrow Free Subtraction	285
8.4	Basic Arithmetic Circuits for Redundant Number Systems	287
8.4.1	Circuit Realization of Carry-Free Adders	287
8.4.2	Fast Maximally Redundant Carry-Free Adders	288
8.4.3	Carry-Free Addition of Symmetric Maximally Redundant Numbers	290
8.4.4	Addition and Subtraction of Stored-Carry Encoded Redundant Operands	292
8.4.4.1	Stored Unibit Carry Free Adders	294
8.4.4.2	Stored Unibit Subtraction	296
8.5	Binary to Redundant Conversion and the Reverse	297
8.5.1	Binary to MRSD Conversion and the Reverse	297
8.5.2	Binary to Stored Unibit Conversion and the Reverse	298
8.6	Special Arithmetic Circuits for Redundant Number Systems	299
8.6.1	Radix- $2^h$ MRSD Arithmetic Shifts	299
8.6.2	Stored Unibit Arithmetic Shifts	300

- 8.6.3 Apparent Overflow 303
- 8.7 Applications 303
  - 8.7.1 Redundant Representation of Partial Products 305
  - 8.7.2 Recoding the Multiplier to a Redundant Representation 305
  - 8.7.3 Use of Redundant Number Systems in Digit Recurrence Algorithms 306
  - 8.7.4 Transcendental Functions and Redundant Number Systems 306
  - 8.7.5 RDNS and Fused Multiply–Add Operation 307
  - 8.7.6 RDNS and Floating Point Arithmetic 307
  - 8.7.7 RDNS and RNS Arithmetic 308
- 8.8 Summary and Further Reading 308
  - References 309
  
- Index 313**