CONTENTS

Preface xv
Contributors xvii

1 Low Power Multicore Processors for Embedded Systems 1

Fumio Arakawa

1.1 Multicore Chip with Highly Efficient Cores 1
1.2 SuperH™ RISC Engine Family (SH) Processor Cores 5
   1.2.1 History of SH Processor Cores 5
   1.2.2 Highly Efficient ISA 7
   1.2.3 Asymmetric In-Order Dual-Issue Superscalar
        Architecture 8
1.3 SH-X: A Highly Efficient CPU Core 9
   1.3.1 Microarchitecture Selections 11
   1.3.2 Improved Superpipeline Structure 12
   1.3.3 Branch Prediction and Out-of-Order Branch Issue 13
   1.3.4 Low Power Technologies 15
   1.3.5 Performance and Efficiency Evaluations 17
1.4 SH-X FPU: A Highly Efficient FPU 20
   1.4.1 FPU Architecture of SH Processors 21
   1.4.2 Implementation of SH-X FPU 24
   1.4.3 Performance Evaluations with 3D Graphics
        Benchmark 29
1.5 SH-X2: Frequency and Efficiency Enhanced Core 33
   1.5.1 Frequency Enhancement 33
   1.5.2 Low Power Technologies 34
1.6 SH-X3: Multicore Architecture Extension 34
   1.6.1 SH-X3 Core Specifications 34
   1.6.2 Symmetric and Asymmetric Multiprocessor Support 35
   1.6.3 Core Snoop Sequence Optimization 36
   1.6.4 Dynamic Power Management 39
   1.6.5 RP-1 Prototype Chip 40
      1.6.5.1 RP-1 Specifications 40
      1.6.5.2 Chip Integration and Evaluations 41
# CONTENTS

1.6.6 RP-2 Prototype Chip 43  
1.6.6.1 RP-2 Specifications 43  
1.6.6.2 Power Domain and Partial Power Off 43  
1.6.6.3 Synchronization Support Hardware 45  
1.6.6.4 Chip Integration and Evaluations 47  

1.7 SH-X4: ISA and Address Space Extension 47  
1.7.1 SH-X4 Core Specifications 48  
1.7.2 Efficient ISA Extension 49  
1.7.3 Address Space Extension 52  
1.7.4 Data Transfer Unit 53  
1.7.5 RP-X Prototype Chip 54  
1.7.5.1 RP-X Specifications 54  
1.7.5.2 Chip Integration and Evaluations 56  

References 57  

---  

## 2 Special-Purpose Hardware for Computational Biology 61  
*Siddharth Srinivasan*

2.1 Molecular Dynamics Simulations on Graphics Processing Units 62  
2.1.1 Molecular Mechanics Force Fields 63  
2.1.1.1 Bond Term 63  
2.1.1.2 Angle Term 63  
2.1.1.3 Torsion Term 63  
2.1.1.4 Van der Waal’s Term 64  
2.1.1.5 Coulomb Term 65  
2.1.2 Graphics Processing Units for MD Simulations 65  
2.1.2.1 GPU Architecture Case Study: NVIDIA Fermi 66  
2.1.2.2 Force Computation on GPUs 69  

2.2 Special-Purpose Hardware and Network Topologies for MD Simulations 72  
2.2.1 High-Throughput Interaction Subsystem 72  
2.2.1.1 Pairwise Point Interaction Modules 74  
2.2.1.2 Particle Distribution Network 74  
2.2.2 Hardware Description of the Flexible Subsystem 75  
2.2.3 Performance and Conclusions 77  

2.3 Quantum MC Applications on Field-Programmable Gate Arrays 77  
2.3.1 Energy Computation and WF Kernels 78  
2.3.2 Hardware Architecture 79  
2.3.2.1 Binning Scheme 79
2.3.3 PE and WF Computation Kernels 80

2.3.3.1 Distance Computation Unit (DCU) 81
2.3.3.2 Calculate Function Unit and Accumulate Function Kernels 81

2.4 Conclusions and Future Directions 82
References 82

3 Embedded GPU Design 85
Byeong-Gyu Nam and Hoi-Jun Yoo

3.1 Introduction 85
3.2 System Architecture 86
3.3 Graphics Modules Design 88
  3.3.1 RISC Processor 88
  3.3.2 Geometry Processor 89
    3.3.2.1 Geometry Transformation 90
    3.3.2.2 Unified Multifunction Unit 90
    3.3.2.3 Vertex Cache 91
  3.3.3 Rendering Engine 92
3.4 System Power Management 95
  3.4.1 Multiple Power-Domain Management 95
  3.4.2 Power Management Unit 98
3.5 Implementation Results 99
  3.5.1 Chip Implementation 99
  3.5.2 Comparisons 100
3.6 Conclusion 102
References 105

4 Low-Cost VLSI Architecture for Random Block-Based Access of Pixels in Modern Image Sensors 107
Tareq Hasan Khan and Khan Wahid

4.1 Introduction 107
4.2 The DVP Interface 108
4.3 The iBRIDGE-BB Architecture 109
  4.3.1 Configuring the iBRIDGE-BB 110
  4.3.2 Operation of the iBRIDGE-BB 110
  4.3.3 Description of Internal Blocks 112
    4.3.3.1 Sensor Control 112
    4.3.3.2 I2C 112
    4.3.3.3 Memory Addressing and Control 114
    4.3.3.4 Random Access Memory (RAM) 114
    4.3.3.5 Column and Row Calculator 115
4.3.6 Physical Memory Address Generator 115
4.3.7 Clock Generator 116

4.4 Hardware Implementation 116
4.4.1 Verification in Field-Programmable Gate Array 116
4.4.2 Application in Image Compression 118
4.4.3 Application-Specific Integrated Circuit (ASIC) Synthesis and Performance Analysis 121

4.5 Conclusion 123
Acknowledgments 123
References 125

5 Embedded Computing Systems on FPGAs 127
Lesley Shannon

5.1 FPGA Architecture 128
5.2 FPGA Configuration Technology 129
5.2.1 Traditional SRAM-Based FPGAs 130
5.2.1.1 DPR 130
5.2.1.2 Challenges for SRAM-Based Embedded Computing System 131
5.2.1.3 Other SRAM-Based FPGAs 132
5.2.2 Flash-Based FPGAs 133
5.3 Software Support 133
5.3.1 Synthesis and Design Tools 134
5.3.2 OSs Support 135
5.4 Final Summary of Challenges and Opportunities for Embedded Computing Design on FPGAs 135
References 136

6 FPGA-Based Emulation Support for Design Space Exploration 139
Paolo Meloni, Simone Secchi, and Luigi Raffo

6.1 Introduction 139
6.2 State of the Art 140
6.2.1 FPGA-Only Emulation Techniques 141
6.2.2 FPGA-Based Cosimulation Techniques 142
6.2.3 FPGA-Based Emulation for DSE Purposes: A Limiting Factor 144
6.3 A Tool for Energy-Aware FPGA-Based Emulation: The MADNESS Project Experience 144
6.3.1 Models for Prospective ASIC Implementation 146
6.3.2 Performance Extraction 147
6.4 Enabling FPGA-Based DSE: Runtime-Reconfigurable Emulators 147
   6.4.1 Enabling Fast NoC Topology Selection 148
      6.4.1.1 WCT Definition Algorithm 148
      6.4.1.2 The Extended Topology Builder 151
      6.4.1.3 Hardware Support for Runtime Reconfiguration 152
      6.4.1.4 Software Support for Runtime Reconfiguration 153
   6.4.2 Enabling Fast ASIP Configuration Selection 154
      6.4.2.1 The Reference Design Flow 155
      6.4.2.2 The Extended Design Flow 156
      6.4.2.3 The WCC Synthesis Algorithm 157
      6.4.2.4 Hardware Support for Runtime Reconfiguration 158
      6.4.2.5 Software Support for Runtime Reconfiguration 161
6.5 Use Cases 161
   6.5.1 Hardware Overhead Due to Runtime Configurability 164
References 166

7 FPGA Coprocessing Solution for Real-Time Protein Identification Using Tandem Mass Spectrometry 169
   Daniel Coca, István Bogdán, and Robert J. Beynon

7.1 Introduction 169
7.2 Protein Identification by Sequence Database Searching Using MS/MS Data 171
7.3 Reconfigurable Computing Platform 174
7.4 FPGA Implementation of the MS/MS Search Engine 176
   7.4.1 Protein Database Encoding 176
   7.4.2 Overview of the Database Search Engine 177
   7.4.3 Search Processor Architecture 178
   7.4.4 Performance 180
7.5 Summary 180
Acknowledgments 181
References 181

8 Real-Time Configurable Phase-Coherent Pipelines 185
   Robert L. Shuler, Jr., and David K. Rutishauser

8.1 Introduction and Purpose 185
   8.1.1 Efficiency of Pipelined Computation 185
   8.1.2 Direct Datapath (Systolic Array) 186
8.1.3 Custom Soft Processors 186
8.1.4 Implementation Framework (e.g., C to VHDL) 186
8.1.5 Multicore 187
8.1.6 Pipeline Data-Feeding Considerations 187
8.1.7 Purpose of Configurable Phase-Coherent Pipeline Approach 187

8.2 History and Related Methods 188
8.2.1 Issues in Tracking Data through Pipelines 188
8.2.2 Decentralized Tag-Based Control 189
8.2.3 Tags in Instruction Pipelines 189
8.2.4 Similar Techniques in Nonpipelined Applications 190
8.2.5 Development-Friendly Approach 190

8.3 Implementation Framework 191
8.3.1 Dynamically Configurable Pipeline 191
  8.3.1.1 Catching up with Synthesis of In-Line Operations 192
  8.3.1.2 Reconfiguration Example with Sparse Data Input 193
8.3.2 Phase Tag Control 195
  8.3.2.1 Tags 195
  8.3.2.2 Data Entity Record Types 195
  8.3.2.3 Tag Shells 196
  8.3.2.4 Latency Choices 197
  8.3.2.5 Example 197
  8.3.2.6 Strobes 198
  8.3.2.7 Tag Values 198
  8.3.2.8 Coding Overhead 198
  8.3.2.9 Reusing Functional Units 199
  8.3.2.10 Tag-Controlled Single-Adder Implementation 199
  8.3.2.11 Interference 200
8.3.3 Phase-Coherent Resource Allocation 200
  8.3.3.1 Determining the Reuse Interval 201
  8.3.3.2 Buffering Burst Data 201
  8.3.3.3 Allocation to Phases 202
  8.3.3.4 Harmonic Number 202
  8.3.3.5 Allocation Algorithm 202
  8.3.3.6 Considerations 203
  8.3.3.7 External Interface Units 204

8.4 Prototype Implementation 204
8.4.1 Coordinate Conversion and Regridding 205
8.4.2 Experimental Setup 206
8.4.3 Experimental Results 206
9 **Low Overhead Radiation Hardening Techniques for Embedded Architectures**  
*Sohan Purohit, Sai Rahul Chalamalasetti, and Martin Margala*

9.1 Introduction 211  
9.2 Recently Proposed SEU Tolerance Techniques 213  
9.2.1 Radiation Hardened Latch Design 214  
9.2.2 Radiation-Hardened Circuit Design Using Differential Cascode Voltage Swing Logic 215  
9.2.3 SEU Detection and Correction Using Decoupled Ground Bus 218  
9.3 Radiation-Hardened Reconfigurable Array with Instruction Rollback 223  
9.3.1 Overview of the MORA Architecture 223  
9.3.2 Single-Cycle Instruction Rollback 227  
9.3.3 MORA RC with Rollback Mechanism 230  
9.3.4 Impact of the Rollback Scheme on Throughput of the Architecture 232  
9.3.5 Comparison of Proposed Schemes with Competing SEU Hardening Schemes 234  
9.4 Conclusion 234  
References 236

10 **Hybrid Partially Adaptive Fault-Tolerant Routing for 3D Networks-on-Chip**  
*Sudeep Pasricha and Yong Zou*

10.1 Introduction 239  
10.2 Related Work 240  
10.3 Proposed 4NP-First Routing Scheme 242  
10.3.1 3D Turn Models 242  
10.3.2 4NP-First Overview 243  
10.3.3 Turn Restriction Checks 244  
10.3.4 Prioritized Valid Path Selection 246  
10.3.4.1 Prioritized Valid Path Selection for Case 1 246  
10.3.4.2 Prioritized Valid Path Selection for Case 2 246  
10.3.5 4NP-First Router Implementation 248  
10.4 Experiments 250  
10.4.1 Experimental Setup 250  
10.4.2 Comparison with Existing FT Routing Schemes 251
10.5 Conclusion 255
References 256

11 Interoperability in Electronic Systems 259
Andrew Leone

11.1 Interoperability 259
11.2 The Basis for Interoperability: The OSI Model 261
11.3 Hardware 263
11.4 Firmware 266
11.5 Partitioning the System 268
11.6 Examples of Interoperable Systems 270

12 Software Modeling Approaches for Presilicon System Performance Analysis 273
Kenneth J. Schultz and Frederic Risacher

12.1 Introduction 273
12.2 Methodologies 275
   12.2.1 High-Level Software Description 276
   12.2.2 Transaction Trace File 278
   12.2.3 Stochastic Traffic Generator 282
12.3 Results 283
   12.3.1 Audio Power Consumption: High-Level Software Description 284
   12.3.2 Cache Analysis: Transaction Trace File 285
   12.3.3 GPU Traffic Characterization: Stochastic Traffic Generator 287
12.4 Conclusion 288
References 288

13 Advanced Encryption Standard (AES) Implementation in Embedded Systems 291
Issam Hammad, Kamal El-Sankary, and Ezz El-Masry

13.1 Introduction 291
13.2 Finite Field 292
   13.2.1 Addition in Finite Field 293
   13.2.2 Multiplication in Finite Field 293
13.3 The AES 293
   13.3.1 Shift Rows/Inverse Shift Rows 295
   13.3.2 Byte Substitution and Inverse Byte Substitution 295
      13.3.2.1 Multiplicative Inverse Calculation 296
      13.3.2.2 Affine Transformation 297
   13.3.3 Mix Columns/Inverse Mix Columns Steps 298
   13.3.4 Key Expansion and Add Round Key Step 299
13.4 Hardware Implementations for AES  
13.4.1 Composite Field Arithmetic S-BOX  
13.4.2 Very High Speed AES Design  
13.5 High-Speed AES Encryptor with Efficient Merging Techniques  
13.5.1 The Integrated-BOX  
13.5.2 Key Expansion Unit  
13.5.3 The AES Encryptor with the Merging Technique  
13.5.4 Results and Comparison  
14 Reconfigurable Architecture for Cryptography over Binary Finite Fields  
Samuel Antão, Ricardo Chaves, and Leonel Sousa  
14.1 Introduction  
14.2 Background  
14.2.1 Elliptic Curve Cryptography  
14.2.1.1 Finite Field Arithmetic  
14.2.1.2 Elliptic Curve Arithmetic  
14.2.2 Advanced Encryption Standard  
14.2.3 Random Number Generators  
14.3 Reconfigurable Processor  
14.3.1 Processing Unit for Elliptic Curve Cryptography  
14.3.1.1 Storage Logic  
14.3.1.2 Addition Logic  
14.3.1.3 Multiplication Logic  
14.3.1.4 Reduction and Squaring Logic  
14.3.2 Processing Unit for the AES  
14.3.3 Random Number Generator  
14.3.4 Microinstructions and Access Arbiter  
14.4 Results  
14.4.1 Individual Components  
14.4.2 Complete Processor Evaluation  
14.5 Conclusions  
References