
CONTENTS

List of Figures	xv
Foreword by Clayton Christensen	xxi
Foreword by John Hennessy	xxv
Author's Preface	xxvii
Acknowledgments	xxxii
Chapter 1 The Case for a New SOC Design Methodology	1
1.1 The Age of Megagate SOC's	1
1.1.1 Moore's Law Means Opportunity and Crisis	3
1.1.2 Roadblock 1: Building the Wrong Chip	4
1.1.3 Roadblock 2: Building the Chip Wrong	5
1.2 The Fundamental Trends of SOC Design	6
1.2.1 A New SOC for Every System is a Bad Idea	6
1.2.2 SOC Design Reform: Lower Design Cost and Greater Design Flexibility	7
1.2.3 Concurrency	8
1.2.4 Programmability	10
1.2.5 Programmability Versus Efficiency	10
1.2.6 The Key to SOC Design Success: Domain-Specific Flexibility	14
1.3 What's Wrong with Today's Approach to SOC Design?	15

1.3.1	What's Wrong with Traditional Processors?	16
1.3.2	What's Wrong with Traditional SOC Methodology?	18
1.4	Preview: An Improved Design Methodology for SOC Design	21
1.4.1	The SOC Design Flow	21
1.4.2	Configurable Processor as Building Block	22
1.4.3	A Trivial Example	24
1.4.4	Results of Application-Specific Processor Configuration	25
1.4.5	The Processor as SOC Building Block	26
1.4.6	Solving the System Design Problem	27
1.4.7	Implications of Improved SOC Methodology	32
1.4.8	The Transition to Processor-based SOC Design	33
1.5	Further Reading	34
Chapter 2	SOC Design Today	36
2.1	Hardware System Structure	36
2.1.1	How Is RTL Used Today?	37
2.1.2	Control, Data Path, and Memory	38
2.1.3	Hardware Trends	40
2.2	Software Structure	41
2.2.1	Software Trends	44
2.3	Current SOC Design Flow	44
2.4	The Impact of Semiconductor Economics	48
2.5	Six Major Issues in SOC Design	50
2.5.1	Changing Market Needs	50
2.5.2	Inadequate Product Volume and Longevity	50
2.5.3	Inflexibility in the Semiconductor Supply Chain	51
2.5.4	Inadequate Performance, Efficiency, and Cost	52
2.5.5	Risk, Cost, and Delay in Design and Verification	53
2.5.6	Inadequate Coordination Between Hardware and Software Teams	54
2.5.7	Solving the Six Problems	54
2.6	Further Reading	55
Chapter 3	A New Look at SOC Design	56
3.1.1	The Basics of Processor-Centric SOC Architecture	57
3.1.2	Processor Generation	57
3.2	Accelerating Processors for Traditional Software Tasks	62
3.2.1	The Evolution of Generic Processors	62
3.2.2	Explaining Configurability and Extensibility	64

3.2.3	Processor Extensibility	64
3.2.4	Designer-Defined Instruction Sets	66
3.2.5	Memory Systems and Configurability	67
3.2.6	The Origins of Configurable Processors	69
3.3	Example: Tensilica Xtensa Processors for EEMBC Benchmarks	70
3.3.1	EEMBC Consumer Benchmarks	71
3.3.2	Telecommunications	72
3.3.3	EEMBC Networking Benchmarks	73
3.3.4	The Processor as RTL Alternative	76
3.4	System Design with Multiple Processors	78
3.4.1	Available Concurrency	79
3.4.2	Parallelism and Power	80
3.4.3	A Pragmatic View of Multiple Processor Design Methodology	81
3.4.4	Forms of Partitioning	82
3.4.5	Processor Interface and Interconnect	84
3.4.6	Communications between Tasks	89
3.5	New Essentials of SOC Design Methodology	89
3.5.1	SOC Design Flow	90
3.5.2	The Essential Phases of the New Flow	92
3.6	Addressing the Six Problems	93
3.6.1	Make the SOC More Programmable	94
3.6.2	Build an Optimized Platform to Aggregate Volume	94
3.6.3	Use Portable IP Foundations for Supply Leverage	95
3.6.4	Optimize Processors for Performance and Efficiency	95
3.6.5	Replace Hard-wired Design with Tuned Processors	95
3.6.6	Unify Hardware and Software with Processor-Centric SOC Methodology	96
3.6.7	Complex SOC and the Six Problems	96
3.7	Further Reading	96
Chapter 4	System-Level Design of Complex SOC	99
4.1	Complex SOC System Architecture Opportunities	101
4.1.1	The Basic Process of Parallel Design	102
4.1.2	The SOC as a Network of Interacting Components	103
4.1.3	Impact of Silicon Scaling on System Partitioning	105
4.1.4	Why Multiple Processors	106
4.1.5	Types of Concurrency and System Architecture	107

4.1.6	Latency, Bandwidth and Communications Structure	110
4.1.7	Reliability and Scalability in SOC Communications Architecture	117
4.1.8	Communications Programming Flexibility	118
4.1.9	Early vs. Late-Binding of Interaction Mechanisms	120
4.2	Major Decisions in Processor-Centric SOC Organization	122
4.2.1	The Starting Point: Essential Interfaces and Computation	123
4.2.2	Parallelizing a Task	126
4.2.3	Assigning Tasks to Processors	128
4.2.4	Choosing the Right Communications Structure	133
4.3	Communication Design = Software Mode + Hardware Interconnect	138
4.3.1	Software Communication Modes	138
4.3.2	Message Passing	139
4.3.3	Shared Memory	141
4.3.4	Device Driver	144
4.4	Hardware Interconnect Mechanisms	145
4.4.1	Buses	146
4.4.2	Direct Connect Ports	150
4.4.3	Data Queues	152
4.4.4	Time-Multiplexed Processor	155
4.5	Performance-Driven Communication Design	155
4.5.2	System Modeling Languages	157
4.5.3	System Modeling Example: XTMP	159
4.5.4	Balancing Computation and Communications	165
4.6	The SOC Design Flow	166
4.6.1	Recommended Design Flow	166
4.6.2	Shifts in SOC Design Methodology	168
4.7	Non-Processor Building Blocks in Complex SOC	170
4.7.1	Memories	170
4.7.2	I/O Peripherals	171
4.7.3	Hardwired Logic Blocks	173
4.8	Implications of Processor-Centric SOC Architecture	174
4.9	Further Reading	176
Chapter 5	Configurable Processors: A Software View	178
5.1	Processor Hardware/Software Cogenesis	180

5.1.1	Applications, Programming Languages, and Processor Architecture	180
5.1.2	A Quick Example: Pixel Blending	182
5.2	The Process of Instruction Definition and Application Tuning	184
5.2.1	Profiling and Performance	184
5.2.2	New Instructions for Performance and Efficiency	190
5.3	The Basics of Instruction Extension	190
5.3.1	Instruction Extension Methods	193
5.3.2	Upgrading the Application	198
5.3.3	The Tradeoff between Instruction-Set Performance and Generality	201
5.3.4	Operation Fusion	202
5.3.5	Compound Operations	209
5.3.6	SIMD Instructions	211
5.4	The Programmer's Model	214
5.4.1	The Base User Instruction Set	214
5.4.2	The Application-Specific Instruction Set	218
5.4.3	The System-Programming Instruction Set	219
5.5	Processor Performance Factors	221
5.5.1	The Software Development Environment	226
5.5.2	The Software Runtime Environment	233
5.5.3	Processor Generation Flow	236
5.6	Example: Tuning a Large Task	236
5.7	Memory-System Tuning	243
5.7.1	Basic Memory-System Strategy	243
5.7.2	Detailed Memory-System Tuning	244
5.7.3	Aggregate Memory System Performance	245
5.7.4	Inner-Loop Data-Reference Tuning	248
5.8	Long Instruction Words	253
5.8.1	Code Size and Long Instructions	255
5.8.2	Long Instruction Words and Automatic Processor Generation	257
5.9	Fully Automatic Instruction-Set Extension	260
5.10	Further Reading	266
Chapter 6	Configurable Processors: A Hardware View	267
6.1	Application Acceleration: A Common Problem	268
6.2	Introduction to Pipelines and Processors	271

6.2.1	Pipelining Fundamentals	272
6.2.2	RISC Pipeline Basics	272
6.2.3	Pipelines for Extended Instruction-Set Implementation	274
6.2.4	Guarantee of Correctness in Processor Hardware Extension	275
6.3	Hardware Blocks to Processors	276
6.3.1	The Basic Transformation of Hardware into Instructions	277
6.3.2	One Primitive Operation per Instruction	280
6.3.3	Multiple Independent Operations per Instruction	285
6.3.4	Pipelined Instruction	288
6.3.5	Tradeoffs in Mapping Hardware Functions to Processor Instructions	290
6.4	Moving from Hardwired Engines to Processors	291
6.4.1	Translating Finite-State Machines to Software	292
6.4.2	Designing Application-Specific Processors for Flexibility	296
6.4.3	Moving from Microcoded Engines to Processors	298
6.4.4	Microcode Data Paths	302
6.4.5	Encoding Operations	304
6.4.6	Microprograms	307
6.5	Designing the Processor Interface	308
6.5.1	Memory-Mapped RAM	311
6.5.2	Memory-Mapped Queues and Registers	314
6.5.3	Wire-Based Input and Output	319
6.6	A Short Example: ATM Packet Segmentation and Reassembly	323
6.7	Novel Roles for Processors in Hardware Replacement	328
6.7.1	The Deeply Buried Task Engine	328
6.7.2	Designing with Spare Processors	330
6.7.3	The System-Monitor Processor	332
6.8	Processors, Hardware Implementation, and Verification Flow	333
6.8.1	Hardware Flow	333
6.8.2	Verification Flow	336
6.9	Progress in Hardware Abstraction	339
6.10	Further Reading	340
Chapter 7	Advanced Topics in SOC Design	342
7.1	Pipelining for Processor Performance	342
7.2	Inside Processor Pipeline Stalls	346
7.2.2	Pipelines and Exceptions	350
7.2.3	Alternative Pipelining for Complex Instructions	352

7.3	Optimizing Processors to Match Hardware	354
7.3.1	Overcoming Differences in Branch Architecture	354
7.3.2	Overcoming Limitations in Memory Access	359
7.4	Multiple Processor Debug and Trace	361
7.4.1	MP Debug	361
7.4.2	MP Trace	363
7.5	Issues in Memory Systems	365
7.5.1	Pipelining with Multiple Memory Ports	365
7.5.2	Memory Alignment in SIMD Instruction Sets	366
7.5.3	Synchronization Mechanisms for Shared Memory	367
7.5.4	Instruction ROM	372
7.6	Optimizing Power Dissipation in Extensible Processors	372
7.6.1	Core Power	373
7.6.2	Impact of Extensibility on Performance	374
7.6.3	Memory Power	375
7.6.4	Cache Power Dissipation Guide	377
7.7	Essentials of TIE	377
7.7.1	TIE Operations	380
7.7.2	TIE States and Register Files	382
7.7.3	External TIE Ports and Queues	383
7.7.4	TIE Constants	385
7.7.5	TIE Function Scheduling (use and def)	387
7.7.6	Using Built-in Registers, Interfaces, and Functions with TIE	387
7.7.7	Shared and Iterative TIE Functions	390
7.7.8	Multi-Slot Instructions	391
7.8	Further Reading	392
Chapter 8	The Future of SOC Design: The Sea of Processors	394
8.1.1	What's Happening to SOC Design?	395
8.1.2	SOC and ROI	396
8.1.3	The Designer's Dilemma	398
8.1.4	The Limitations of General-Purpose Processors	399
8.1.5	The New Processor	401
8.1.6	What Makes These Processors Different?	403
8.1.7	The SOC Design Transition	404
8.2	Why Is Software Programmability So Central?	407
8.3	Looking into the Future of SOC	410
8.4	Processor Scaling Model	410

8.4.1	Summary of Model Assumptions	417
8.5	Future Applications of Complex SOC	419
8.6	The Future of the Complex SOC Design Process	421
8.7	The Future of the Industry	427
8.8	The Disruptive-Technology View	431
8.9	The Long View	436
8.10	Further Reading	437
	Index	439