

Fully
revised
and
updated

Second Edition



EMBEDDED SYSTEMS

A CONTEMPORARY DESIGN TOOL

JAMES K. PECKOL



WILEY

Embedded Systems

Embedded Systems

A Contemporary Design Tool

Second Edition

James K. Peckol, Ph.D.

University of Washington

WILEY

This edition first published 2019
© 2019 John Wiley & Sons Ltd

Edition History

1st Edition:

Embedded system: a contemporary design tool / James K. Peckol.

ISBN 978-0-471-72180-2

(cloth)

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by law. Advice on how to obtain permission to reuse material from this title is available at <http://www.wiley.com/go/permissions>.

The right of James K. Peckol to be identified as the author of this work has been asserted in accordance with law.

Registered Offices

John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

Editorial Office

The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

Limit of Liability/Disclaimer of Warranty

While the publisher and authors have used their best efforts in preparing this work, they make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives, written sales materials or promotional statements for this work. The fact that an organization, website, or product is referred to in this work as a citation and/or potential source of further information does not mean that the publisher and authors endorse the information or services the organization, website, or product may provide or recommendations it may make. This work is sold with the understanding that the publisher is not engaged in rendering professional services. The advice and strategies contained herein may not be suitable for your situation. You should consult with a specialist where appropriate. Further, readers should be aware that websites listed in this work may have changed or disappeared between when this work was written and when it is read. Neither the publisher nor authors shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

Library of Congress Cataloging-in-Publication Data

Names: Peckol, James K., author.

Title: Embedded systems : A Contemporary Design Tool / James K. Peckol, Ph.D., University of Washington.

Description: 2nd edition. | Hoboken, NJ : John Wiley & Sons, Inc., 2019. |

Includes bibliographical references and index. |

Identifiers: LCCN 2018039258 (print) | LCCN 2018045459 (ebook) | ISBN

9781119457497 (Adobe PDF) | ISBN 9781119457558 (ePub) | ISBN 9781119457503

(hardcover)

Subjects: LCSH: Embedded computer systems. | Object-oriented methods

(Computer science)

Classification: LCC TK7895.E42 (ebook) | LCC TK7895.E42 P43 2019 (print) |

DDC 006.2/2--dc23

LC record available at <https://lccn.loc.gov/2018039258>

Cover Design: Wiley

Cover Image: © HYWARDS/iStock.com/Getty Images Plus

Set in 10/12 TimesLTStd by SPi Global, Chennai, India

Printed and bound by CPI Group (UK) Ltd, Croydon, CR0 4YY

10 9 8 7 6 5 4 3 2 1

*To my family: Near and Extended, Close and Distant,
Present and Departed, So Similar,
So Different, So Known, So Surprising ...
especially to our youngest brother Karl,
taken from us out of season during the last voyage
of the Edmund Fitzgerald.*

Contents

About the Author	xxxiii
Foreword	xxxv
Preface	xlix
Acknowledgment	lix
About the Companion Website	lxi

Part 1 Hardware and Software Infrastructure

1	The Hardware Side – Part 1: An Introduction	1
1.1	Introduction	1
1.2	The Hardware Side – Getting Started	3
1.3	The Core Level	3
1.3.1	The Microprocessor	6
1.3.2	The Microcomputer	7
1.3.3	The Microcontroller	7
1.3.4	The Digital Signal Processor	7
1.4	Representing Information	8
1.4.1	Word Size	9
1.5	Understanding Numbers	9
1.5.1	Resolution	10
1.5.2	Propagation of Error	11
1.5.2.1	Addition	11
1.5.2.2	Multiplication	12
1.6	Addresses	13
1.7	Instructions	14
1.8	Registers – A First Look	16
1.9	Embedded Systems – An Instruction Set View	18
1.9.1	Instruction Set – Instruction Types	18
1.9.2	Data Transfer Instructions	18
1.9.2.1	Addressing Modes	20
1.9.3	Execution Flow	26
1.9.3.1	Sequential Flow	26
1.9.3.2	Branch	27
1.9.3.3	If-else Construct	28
1.9.3.4	Loop	28
1.9.3.5	Procedure or Function Call	29
1.9.3.6	Arithmetic and Logic	32
1.10	Embedded Systems – A Register View	34
1.10.1	The Basic Register	35

1.10.2	Register Operations	35
1.10.2.1	Write to a Register	36
1.10.2.2	Read from a Register	36
1.11	Register Transfer Language	36
1.12	Register View of a Microprocessor	38
1.12.1	The Datapath	38
1.12.2	Processor Control	39
1.12.2.1	Fetch	39
1.12.2.2	Decode	40
1.12.2.3	Execute	40
1.12.2.4	Next	41
1.13	Summary	45
1.14	Review Questions	45
1.15	Thought Questions	46
1.16	Problems	47
2	The Hardware Side – Part 2: Combinational Logic – A Practical View	55
2.1	Introduction	55
2.2	A Look at Real-World Gates – Part 1: Signal Levels	56
2.2.1	Logic Levels	57
2.2.2	A First Look Inside the Logic Gate	59
2.2.3	Fan-In and Fan-Out	60
2.3	A Look at Real-World Gates – Part 2: Time	64
2.3.1	Rise and Fall Times	65
2.3.2	Propagation Delay	65
2.3.3	Race Conditions and Hazards	67
2.3.3.1	Static Hazard	67
2.3.3.2	Dynamic Hazard	69
2.4	A Look at Real-World Gates – Part 3: Signal Behavior in the Real World – the Legacy of Early Physicists	70
2.5	Look For the Guilty – A First Look at Signal Quality	71
2.5.1	Resistors	71
2.5.1.1	A Discrete Component First-Order Resistor Model	72
2.5.2	Capacitors	74
2.5.2.1	Discrete Component First-Order Capacitor Model	76
2.5.3	Inductors	78
2.5.3.1	Magnetic Field Lines – The First Principle	78
2.5.3.2	Magnetic Field Lines – The Second Principle	79
2.5.3.3	Magnetic Field Lines – The Third Principle	80
2.6	Inductance in Action	81
2.6.1	Wires and Conductors	82
2.7	Logic Circuit Models and Parasitic Components	84
2.7.1	First-Order RC Circuit Model	84
2.7.2	First-Order RL Circuit Model	86
2.7.3	Second-Order Series RLC Circuit Model	87
2.7.4	Tristate Drivers	89

2.7.4.1	Open Gate Inputs	90
2.8	Testing Combinational Circuits – Introduction and Philosophy	91
2.9	Modeling, Simulation, and Tools	92
2.10	Structural Faults	93
2.10.1	Stuck-at Faults	93
2.10.1.1	Stuck-at-Zero Faults	94
2.10.1.2	Stuck-at-One Faults	95
2.10.2	Open Circuit Faults	96
2.10.3	Bridging Faults	96
2.10.3.1	Nonfeedback Bridge Faults	97
2.10.3.2	Feedback Bridge Faults	99
2.11	Functional Faults	99
2.11.1	Hazards and Race Conditions	100
2.12	Summary	101
2.13	Review Questions	101
2.14	Thought Questions	102
2.15	Problems	104
3	The Hardware Side – Part 3: Storage Elements and Finite-State Machines – A Practical View	111
3.1	Introduction	111
3.2	The Concepts of State and Time	112
3.2.1	Time	112
3.2.2	State	112
3.2.3	State Changes	113
3.3	The State Diagram	113
3.4	Finite-State Machines – A Theoretical Model	114
3.5	Designing Finite-State Machines – Part 1: Registers	116
3.5.1	Storage Registers	116
3.5.2	Shift Registers	117
3.5.2.1	Shift Right Shift Register	117
3.5.2.2	Parallel In/Serial Out – Serial In/Parallel Out Left Shift Registers	121
3.5.3	Linear Feedback Shift Registers	122
3.6	Designing Finite-State Machines – Part 2: Counting and Dividing	124
3.6.1	Dividers	124
3.6.1.1	Divide by Two	124
3.6.2	Asynchronous Dividers and Counters	125
3.6.3	Synchronous Dividers and Counters	127
3.6.4	Johnson Counters	128
3.6.4.1	Two-Stage Johnson Counter	129
3.6.4.2	Three- or Greater Stage Johnson Counter	130
3.7	Practical Considerations – Part 1: Timing in Latches and Flip-Flops	131
3.7.1	Gated Latches	131
3.7.2	Flip-Flops	132
3.7.3	Propagation Delays	132
3.7.4	Timing Margin	133
3.8	Practical Considerations – Part 2: Clocks and Clock Distribution	135

3.8.1	The Source	135
3.8.1.1	Frequency	135
3.8.1.2	Precision and Stability	137
3.8.2	Designing a Clock System	137
3.8.2.1	Single-Phase Clocks	137
3.8.2.2	Multiple-Phase Clocks	138
3.8.2.3	More Than Four Phases	141
3.8.2.4	Multiple Clocks Versus Multiple Phases	141
3.8.2.5	Gating the Clock	141
3.9	Testing Sequential Circuits	141
3.9.1	The Finite-State Machine Model Revisited	142
3.9.2	Sequential Circuit Test – A First Look	142
3.9.2.1	Defining Homing and Transfer Sequences	145
3.9.3	Scan Design Techniques	148
3.9.4	Boundary Scan – Extending Scan-Path Techniques	149
3.10	Summary	152
3.11	Review Questions	152
3.12	Thought Questions	153
3.13	Problems	155
4	Memories and the Memory Subsystem	165
4.1	Introduction	165
4.2	Classifying Memory	166
4.3	A General Memory Interface	167
4.4	ROM Overview	168
4.4.1	Read Operation	169
4.5	Static RAM Overview	169
4.5.1	Write Operation	171
4.5.2	Read Operation	171
4.6	Dynamic RAM Overview	171
4.6.1	Read Operation	172
4.6.2	Write Operation	172
4.6.3	Refresh Operation	172
4.7	Chip Organization	173
4.8	Terminology	173
4.9	A Memory Interface in Detail	176
4.10	An SRAM Design	177
4.10.1	The Memory Array	177
4.10.2	Reading and Writing	179
4.10.3	Write	179
4.10.4	Read	179
4.11	A DRAM Design	180
4.11.1	DRAM Timing Analysis	181
4.11.1.1	Core Components	181
4.11.2	DRAM Refresh	182
4.12	The DRAM Memory Interface	183
4.12.1	Refresh Timing	183
4.12.2	Refresh Address	184

4.12.3	Refresh Arbitration	185
4.13	The Memory Map	188
4.14	Memory Subsystem Architecture	189
4.15	Basic Concepts of Caching	190
4.15.1	Locality of Reference	190
4.15.2	Cache System Architecture	192
4.16	Designing a Cache System	192
4.16.1	A High-Level Description	192
4.17	Caching – A Direct Mapped Implementation	193
4.18	Caching – An Associative Mapping Cache Implementation	196
4.19	Caching – A Block-Set Associative Mapping Cache Implementation	198
4.20	Dynamic Memory Allocation	199
4.20.1	Swapping	200
4.20.2	Overlays	200
4.20.3	Multiprogramming	201
4.20.3.1	Fixed	201
4.20.3.2	Variable Number	202
4.21	Testing Memories	202
4.21.1	RAM Memory	204
4.21.2	ROM Memory	206
4.22	Summary	208
4.23	Review Questions	208
4.24	Thought Questions	209
4.25	Problems	210
5	An Introduction to Software Modeling	215
5.1	Introduction	215
5.2	An Introduction to UML	216
5.3	UML Diagrams	217
5.4	Use Cases	218
5.4.1	Writing a Use Case	219
5.5	Class Diagrams	220
5.5.1	Class Relationships	221
5.5.1.1	Inheritance or Generalization	221
5.5.1.2	Interface	221
5.5.1.3	Containment	222
5.6	Dynamic Modeling with UML	223
5.7	Interaction Diagrams	223
5.7.1	Call and Return	224
5.7.2	Create and Destroy	224
5.7.3	Send	225
5.8	Sequence Diagrams	225
5.9	Fork and Join	226
5.10	Branch and Merge	227
5.11	Activity Diagram	228
5.12	State Chart Diagrams	228
5.12.1	Events	228
5.12.2	State Machines and State Chart Diagrams	229

	5.12.2.1	UML State Chart Diagrams	230
	5.12.2.2	Transitions	230
	5.12.2.3	Guard Conditions	231
	5.12.2.4	Composite States	232
5.13		Dynamic Modeling with Structured Design Methods	233
	5.13.1	Brief Introduction to the Structured Design Philosophy	233
	5.13.2	Data and Control Flow Diagrams	234
	5.13.2.1	The Elements	234
5.14		Summary	237
5.15		Review Questions	237
5.16		Thought Questions	239
5.17		Problems	240
6		The Software Side – Part 1: The C Program	243
6.1		Introduction	243
6.2		Software and Its Manifestations	243
	6.2.1	Combining Hardware and Software	244
	6.2.2	High-Level Language	245
	6.2.3	Preprocessor	245
	6.2.4	Cross Compiler	246
	6.2.5	Assembler	246
	6.2.6	Linker and Loader	247
	6.2.7	Storing	248
6.3		An Embedded C Program	249
	6.3.1	A Program	249
	6.3.2	Developing Embedded Software	249
	6.3.2.1	Abstraction	250
6.4		C Building Blocks	250
	6.4.1	Fundamental Data – What’s in a Name?	250
	6.4.1.1	Identifiers in C	250
	6.4.2	Defining Variables – Giving Them a Name and a Value	251
	6.4.3	Defining Variables – Giving Them a Type, Scope, and Storage Class	252
	6.4.3.1	Type	252
	6.4.3.2	The const Qualifier	258
	6.4.3.3	Variable Names Revisited	259
	6.4.3.4	Type Conversions	259
	6.4.3.5	Scope	261
	6.4.3.6	Storage Class	263
6.5		C Program Structure	268
	6.5.1	Separate Compilation	268
	6.5.2	Translation Units	268
	6.5.3	Linking and Linkage	269
	6.5.3.1	Linking	269
	6.5.3.2	Linkage	270
	6.5.4	Where C Finds Functions	271
	6.5.5	Makefiles	271
	6.5.6	Standard and Custom Libraries	272

6.5.7	Debug and Release Builds	272
6.6	Summary	273
6.7	Review Questions	273
6.8	Thought Questions	274
6.9	Problems	275
7	The Software Side – Part 2: Pointers and Functions	279
7.1	Introduction	279
7.2	Bitwise Operators	280
7.2.1	Bit Manipulation Operations	280
7.2.2	Testing, Resetting, and Setting Bits	281
7.2.3	Arithmetic Operations	284
7.3	Pointer Variables and Memory Addresses	285
7.3.1	Getting Started	285
7.3.2	Simple Pointer Arithmetic	290
7.3.2.1	Pointer Comparison	293
7.3.3	Const Pointers	293
7.3.4	Generic and NULL Pointers	294
7.3.4.1	Generic Pointers	294
7.3.4.2	Null Pointers	295
7.4	The Function	296
7.4.1.1	Function Header	296
7.4.1.2	Function Name	296
7.4.1.3	Arguments or Parameter List	296
7.4.1.4	Return	296
7.4.1.5	The Function Body	297
7.4.2	Using a Function	297
7.4.3	Pass By Value	301
7.4.4	Pass By Reference	303
7.4.5	Function Name Scope	304
7.4.6	Function Prototypes	304
7.4.7	Nesting Functions	306
7.5	Pointers to Functions	306
7.6	Structures	310
7.6.1	The Struct	311
7.6.2	Initialization	313
7.6.3	Access	313
7.6.4	Operations	314
7.6.5	Structs as Data Members	314
7.6.5.1	Accessing Members	314
7.6.5.2	Initialization and Assignment	315
7.6.5.3	Functions	315
7.6.6	Pointers to Structs	318
7.6.6.1	Accessing Members	318
7.6.7	Passing Structs and Pointers to Structs	319
7.7	The Interrupt	320
7.7.1	The Interrupt Control Flow	320
7.7.2	The Interrupt Event	320

7.7.3	The Interrupt Service Routine – ISR	321
7.7.4	The Interrupt Vector Table	321
7.7.5	Control of the Interrupt	323
7.7.5.1	Enable–Disable	323
7.7.5.2	Recognizing an Interrupting Event	323
7.7.5.3	Interrupting and Masking an Interrupting Event	324
7.8	Summary	324
7.9	Review Questions	324
7.10	Thought Questions	326
7.11	Problems	327

Part 2 Developing the Foundation

8	Safety, Security, Reliability, and Robust Design	331
8.1	Introduction	331
8.2	Safety	333
8.3	Reliability	334
8.4	Faults, Errors, and Failures	336
8.5	Another Look at Reliability	337
8.6	Some Real-World Examples	337
8.6.1	Big Word ... Small Register	338
8.6.2	It’s My Turn – Not Yours	338
8.6.3	Where Do I Put My Stuff?	339
8.7	Single-Point and Common Mode Failure Model	340
8.8	Safe Specifications	340
8.9	Safe, Secure, and Robust Designs	341
8.9.1	Understanding System Requirements	341
8.9.2	Managing Essential Information	342
8.9.3	The Review Process	343
8.9.4	Bug Lists	344
8.9.5	Errors and Exceptions	344
8.9.6	Use the Available Tools	347
8.10	Safe and Robust Designs – The System	347
8.11	System Functional Level Considerations	347
8.11.1	Control and Alarm Subsystems	347
8.11.2	Memory and Bus Subsystems	348
8.11.3	Data Faults and the Communications Subsystem	349
8.11.4	Power and Reset Subsystems	349
8.11.5	Peripheral Device Subsystems	349
8.11.6	Clock Subsystem	349
8.12	System Architecture Level Considerations	350
8.12.1	Fail Operational ² /Fail Operational Capability	350
8.12.1.1	Same Design	351
8.12.1.2	Alternative Designs	351
8.12.2	Reduced Capability	352

	8.12.2.1	Lightweight Redundancy	352
	8.12.2.2	Monitor Only	352
8.13		Busses – The Subsystem Interconnect	353
	8.13.1	The Star Configuration	353
	8.13.2	The Multidrop Bus Configuration	353
	8.13.3	The Ring Configuration	354
8.14		Data and Control Faults – Data Boundary Values	355
	8.14.1	Type Conformance	355
	8.14.2	Boundary Values	355
8.15		Data and Control Faults – The Communications Subsystem	356
	8.15.1	Damaged Data	356
	8.15.1.1	Detectability	356
	8.15.1.2	Extent	357
	8.15.1.3	Response	358
	8.15.2	Managing Damaged Data	358
	8.15.2.1	Parity	358
	8.15.2.2	Linear Codes	358
8.16		The Power Subsystem	367
	8.16.1	Full Operation	368
	8.16.2	Reduced Operation	368
	8.16.3	Backup Operation	369
8.17		Peripheral Devices – Built-in Self-Test (BIST)	370
	8.17.1	Self-Tests	370
	8.17.2	Busses	371
	8.17.3	ROM Memory	373
	8.17.4	RAM Memory	373
	8.17.4.1	Peripheral Devices	373
	8.17.4.2	What to Do If a Test Fails?	373
8.18		Failure Modes and Effects Analysis	374
8.19		Security – Look Behind You	376
8.20		Understanding the Problem – Looking at the System	376
8.21		Analyzing the Problem – Looking at Potential Vulnerabilities	377
8.22		Understanding the Problem – Looking at the Attacks	378
	8.22.1	Looking at the Software	378
	8.22.2	Looking at the Hardware	381
8.23		Dealing with the Problem – Protecting Against the Attacks	382
	8.23.1	Protecting the Software	382
	8.23.1.1	First Steps	382
	8.23.1.2	Second Steps	384
	8.23.1.3	Third Steps	385
	8.23.1.4	Fourth Steps	386
	8.23.2	Software Testing Tools	387
	8.23.3	Protecting the Hardware	388
	8.23.3.1	First Steps	388
	8.23.3.2	Second Steps	388
	8.23.3.3	Third Steps	390
	8.23.4	Protecting a Network Interface	390
	8.23.4.1	First Steps	391
	8.23.4.2	Second Steps	391

8.23.5	Firewall	395
8.24	Closure	396
8.25	Tomorrow	396
8.26	Summary	397
8.27	Review Questions	397
8.28	Thought Questions	398
8.29	Problems	399
9	Embedded Systems Design and Development – Hardware–Software Co-Design	403
9.1	Introduction	404
9.2	System Design and Development	405
9.2.1	Getting Ready – Start Thinking	406
9.2.2	Getting Started	407
9.3	Life-Cycle Models	408
9.3.1	The Waterfall Model	409
9.3.2	The V Cycle Model	410
9.3.3	The Spiral Model	411
9.3.4	Rapid Prototyping – Incremental	412
9.4	Problem Solving – Six Steps To Design	413
9.5	Hardware–Software Co-Design	414
9.5.1	The First Steps	415
9.5.2	Traditional Embedded Systems Development	415
9.6	History	417
9.6.1	Advantages of the Co-Design Methodology	417
9.7	Co-Design Process Overview	418
9.8	The Co-Design Process	419
9.9	Laying the Foundation	422
9.10	Identifying the Requirements	423
9.11	Formulating the Requirements Specification	425
9.11.1	The Environment	426
9.11.1.1	Characterizing External Entities	427
9.11.2	The System	427
9.11.2.1	Characterizing the System	429
9.12	The System Design Specification	437
9.12.1	The System	439
9.12.2	Quantifying the System	439
9.13	System Requirements Versus System Design Specifications	448
9.14	Executing the Hardware–Software Co-Design Process	449
9.15	Functional Decomposition	449
	Identifying the Functions	452
	Functional Decomposition	452
9.16	Partitioning and Mapping to an Architecture	454
9.16.1	Initial Thoughts	455
9.16.2	Coupling	457
9.16.3	Cohesion	458
9.16.4	A Few More Considerations	459
9.16.5	Approaches to Partitioning and Mapping	459

	9.16.5.1	The Approach	460
	9.16.5.2	The Method	460
	9.16.6	Evaluation of a Partition	465
9.17		Architectural Design	465
	9.17.1	Mapping Functions to Hardware	465
	9.17.2	Hardware and Software Specification and Design	466
9.18		Functional Model Versus Architectural Model	469
	9.18.1	The Functional Model	469
	9.18.2	The Architectural Model	470
	9.18.3	The Need for Both Models	470
9.19		Modeling Tools and Languages for Co-Design	470
	9.19.1	Why are We Modeling?	471
	9.19.2	What are We Modeling?	471
	9.19.3	Characterizing the Model	472
	9.19.4	Classes of MoCs	472
	9.19.4.1	Conceptual Model	472
	9.19.4.2	Analytic Model	472
	9.19.5	A Look at Some Models	473
	9.19.5.1	The Logic Circuit	474
	9.19.5.2	The Random Access Machine – RAM	474
	9.19.5.3	The Turing Machine	474
	9.19.5.4	The Pushdown Automaton Machine	475
	9.19.5.5	The Basic Finite State Machine	475
	9.19.5.6	Communicating Finite State Machines	476
	9.19.5.7	Extended FSM	476
	9.19.5.8	Co-Design FSM	477
	9.19.5.9	Program State Machines	478
	9.19.5.10	UML State Charts	479
	9.19.5.11	Petri Nets	479
	9.19.5.12	Kahn Process Network	480
	9.19.5.13	Control Flow – Data Flow – CDFG Graphs	480
9.20		Co-Synthesis	480
	9.20.1	Constraints	481
	9.20.2	Software Synthesis	482
	9.20.2.1	System Characterization	483
	9.20.2.2	Scheduling	483
	9.20.2.3	Synthesis Methods	483
9.21		Implementing The System	486
	9.21.1	Analyzing the System Design	486
	9.21.1.1	Static Analysis	486
	9.21.1.2	Dynamic Analysis	487
9.22		Co-Simulation	488
	9.22.1	Tools Supporting Modeling	489
	9.22.2	Types of Simulations	489
	9.22.3	Approaches	490
	9.22.3.1	Detailed Processor Model	490
	9.22.3.2	Cycle-Based Simulation – Bus Model	491
	9.22.3.3	Instruction Set Architecture – ISA Mode	491
	9.22.3.4	Compiled Model	491

	9.22.3.5	Hardware Model	492
	9.22.3.6	Master Slave Model	492
	9.22.3.7	Distributed Co-Simulation	493
	9.22.3.8	Heterogeneous Modeling – The Ptolemy Project	493
	9.22.3.9	Domains	494
	9.22.3.10	Classes of MoCs	494
9.23		Co-Verification	495
	9.23.1	Hardware–Software Co-Verification	495
	9.23.2	Tools Supporting Simulation and Verification	496
	9.23.2.1	Basic Capabilities	496
	9.23.2.2	Levels of Abstraction	496
9.24		Other Considerations	497
	9.24.1	Capitalization and Reuse	497
	9.24.1.1	Capitalization	497
	9.24.1.2	Reuse	497
	9.24.2	Requirements Traceability and Management	498
	9.24.2.1	Requirements Traceability	498
	9.24.2.2	Requirements Management	498
9.25		Archiving the Project	499
9.26		Summary	500
9.27		Review Questions	500
9.28		Thought Questions	502
9.29		Problems	503
10		Hardware Test and Debug	507
	10.1	Introduction	507
	10.2	Some Vocabulary	507
	10.3	Putting Together a Strategy	508
	10.4	Formulating a Plan	509
	10.5	Formalizing the Plan – Writing a Specification	511
	10.6	Executing the Plan – The Test Procedure and Test Cases	512
	10.7	Applying the Strategy – Egoless Design	513
	10.8	Applying the Strategy – Design Reviews	513
	10.9	Applying the Strategy – Module Debug and Test	514
	10.9.1	Black Box Tests	514
	10.9.2	White Box Tests	515
	10.9.3	Gray Box Tests	515
	10.10	Applying the Strategy – The First Steps	516
	10.10.1	The Parts	516
	10.10.2	Initial Tests and Measurements – Before Applying Power	517
	10.10.3	Initial Tests and Measurements – Immediately After Applying Power	518
	10.11	Applying the Strategy – Debugging and Testing	519
	10.11.1	The Reset System	519
	10.11.2	The Clocks and Timing	519
	10.11.3	The Inputs and Outputs	519
	10.11.4	Sudden Failure during Debugging	520
	10.12	Testing and Debugging Combinational Logic	521

10.13	Path Sensitizing	521
10.13.1	Single Variable–Single Path	522
10.13.1.1	Testing	522
10.13.1.2	Debugging	523
10.13.2	Single Variable–Two Paths	523
10.14	Masking and Untestable Faults	525
10.15	Single Variable–Multiple Paths	526
10.16	Bridge Faults	527
10.17	Debugging – Sequential Logic	529
10.18	Scan Design Testing	531
10.19	Boundary-Scan Testing	533
10.20	Memories and Memory Systems	535
10.21	Applying the Strategy – Subsystem and System Test	535
10.22	Applying the Strategy – Testing for Our Customer	536
10.22.1	Alpha and Beta Tests	536
10.22.2	Verification Tests	536
10.22.3	Validation Tests	536
10.22.4	Acceptance Tests	537
10.22.5	Production Tests	537
10.23	Self-Test	537
10.23.1	On Demand	537
10.23.2	In Background	537
10.24	Summary	538
10.25	Review Questions	538
10.26	Thought Questions	539
10.27	Problems	540

Part 3 Doing the Work

11	Real-Time Kernels and Operating Systems	541
11.1	Introduction	541
11.2	Tasks and Things	542
11.3	Programs and Processes	544
11.4	The CPU Is a Resource	544
11.4.1	Setting a Schedule	545
11.4.2	Changing Context	546
11.5	Threads – Lightweight and Heavyweight	547
11.5.1	A Single Thread	547
11.5.2	Multiple Threads	548
11.6	Sharing Resources	549
11.6.1	Memory Resource Management	549
11.6.1.1	System-Level Management	549
11.6.2	Process-Level Management	550
11.6.3	Reentrant Code	551
11.7	Foreground / Background Systems	551
11.8	The Operating System	551

11.9	The Real-Time Operating System (RTOS)	553
11.10	Operating System Architecture	553
11.11	Tasks and Task Control Blocks	555
11.11.1	The Task	555
11.11.2	The Task Control Block	555
11.11.3	A Simple Kernel	557
11.11.4	Interrupts Revisited	560
11.12	Memory Management Revisited	564
11.12.1	Duplicate Hardware Context	565
11.12.2	Task Control Blocks	567
11.12.3	Stacks	567
11.12.3.1	Runtime Stack	568
11.12.3.2	Application Stacks	569
11.12.3.3	Multiprocessing Stacks	569
11.13	Summary	570
11.14	Review Questions	570
11.15	Thought Questions	571
11.16	Problems	571
12	Tasks and Task Management	573
12.1	Introduction	573
12.2	Time, Time-Based Systems, and Reactive Systems	574
12.2.1	Time	574
12.2.2	Reactive and Time-Based Systems	574
12.3	Task Scheduling	577
12.3.1	CPU Utilization	577
12.3.2	Scheduling Decisions	578
12.3.3	Scheduling Criteria	578
12.3.3.1	Priority	578
12.3.3.2	Turnaround Time	579
12.3.3.3	Throughput	579
12.3.3.4	Waiting Time	580
12.3.3.5	Response Time	580
12.4	Scheduling Algorithms	580
12.4.1	Asynchronous Interrupt Event Driven	580
12.4.2	Polled and Polled with a Timing Element	581
12.4.3	State Based	581
12.4.4	Synchronous Interrupt Event Driven	582
12.4.5	Combined Interrupt Event Driven	582
12.4.6	Foreground–Background	582
12.4.7	Time-Shared Systems	583
12.4.7.1	First-Come First-Served	583
12.4.7.2	Shortest Job First	583
12.4.7.3	Round Robin	583
12.4.8	Priority Schedule	583
12.4.8.1	Rate-Monotonic	583
12.4.8.2	Earliest Deadline	584
12.4.8.3	Least Laxity	585

	12.4.8.4	Maximum Urgency	585
12.5		Real-Time Scheduling Considerations	586
12.6		Algorithm Evaluation	586
	12.6.1	Deterministic Modeling	586
	12.6.2	Queuing Models	588
	12.6.3	Simulation	589
	12.6.4	Implementation	589
12.7		Tasks, Threads, and Communication	589
	12.7.1	Getting Started	589
	12.7.2	Intertask / Interthread Communication	589
	12.7.3	Shared Variables	590
	12.7.3.1	Global Variables	590
	12.7.3.2	Shared Buffer	591
	12.7.3.3	Shared Double Buffer – Ping-Pong Buffer	591
	12.7.3.4	Ring Buffer	593
	12.7.3.5	Mailbox	593
	12.7.4	Messages	594
	12.7.4.1	Communication	595
	12.7.4.2	Buffering	599
12.8		Task Cooperation, Synchronization, and Sharing	599
	12.8.1	Critical Sections and Synchronization	600
	12.8.2	Flags	604
	12.8.3	Token Passing	606
	12.8.4	Interrupts	606
	12.8.5	Semaphores	607
	12.8.6	Process Synchronization	609
	12.8.7	Spin Lock and Busy Waiting	609
	12.8.8	Counting Semaphores	609
12.9		Talking and Sharing in Space	610
	12.9.1	The Bounded Buffer Problem	610
	12.9.2	The Readers and Writers Problem	612
12.10		Monitors	614
	12.10.1	Condition Variables	615
	12.10.2	Bounded Buffer Problem with Monitor	616
12.11		Starvation	618
12.12		Deadlocks	618
12.13		Summary	618
12.14		Review Questions	618
12.15		Thought Questions	619
12.16		Problems	620
13		Deadlocks	625
	13.1	Introduction	625
	13.2	Sharing Resources	625
	13.3	System Model	626
	13.4	Deadlock Model	627
	13.4.1	Necessary Conditions	628
	13.5	A Graph Theoretic Tool – The Resource Allocation Graph	628

13.6	Handling Deadlocks	631
13.7	Deadlock Prevention	631
	13.7.1 Mutual Exclusion	631
	13.7.2 Hold and Wait	632
	13.7.3 No Preemption	632
	13.7.4 Circular Wait	632
13.8	Deadlock Avoidance	633
	13.8.1 Algorithms Based on the Resource Allocation Graph	634
	13.8.2 Banker's Algorithm and Safe States	635
13.9	Deadlock Detection	638
	13.9.1 Detection in a Single-Instance Environment	638
	13.9.2 Deadlock Recovery	638
	13.9.2.1 Task Termination	639
	13.9.2.2 Resource Preemption	639
13.10	Summary	640
13.11	Review Questions	640
13.12	Thought Questions	641
13.13	Problems	641
14	Performance Analysis and Optimization	645
14.1	Introduction	645
14.2	Getting Started	646
14.3	Performance or Efficiency Measures	646
	14.3.1 Introduction	646
	14.3.2 The System	648
	14.3.3 Some Limitations	648
14.4	Complexity Analysis – A High-Level Measure	649
14.5	The Methodology	651
	14.5.1 A Simple Experiment	651
	14.5.2 Working with Big Numbers	652
	14.5.3 Asymptotic Complexity	652
14.6	Comparing Algorithms	652
	14.6.1 Big-O Notation	653
	14.6.2 Big-O Arithmetic	654
14.7	Analyzing Code	655
	14.7.1 Constant Time Statements	655
	14.7.2 Looping Constructs	656
	14.7.2.1 For Loops	656
	14.7.2.2 While Loops	657
	14.7.3 Sequences of Statements	657
	14.7.4 Conditional Statements	658
	14.7.5 Function Calls	658
14.8	Analyzing Algorithms	659
	14.8.1 Analyzing Search	659
	14.8.1.1 Linear Search	659
	14.8.1.2 Binary Search	660
	14.8.2 Analyzing Sort	660
	14.8.2.1 Selection Sort	660

	14.8.2.2 Quick Sort	661
14.9	Analyzing Data Structures	661
	14.9.1 Array	661
	14.9.2 Linked List	662
14.10	Instructions in Detail	662
	14.10.1 Getting Started	663
	14.10.2 Flow of Control	663
	14.10.2.1 Sequential	664
	14.10.2.2 Branch	664
	14.10.2.3 Loop	665
	14.10.2.4 Function Call	665
	14.10.3 Analyzing the Flow of Control – Two Views	666
	14.10.3.1 Sequential Flow	666
	14.10.3.2 Branch	666
	14.10.3.3 Loop	667
	14.10.3.4 Function Call	668
	14.10.3.5 Co-routine	670
	14.10.3.6 Interrupt Call	671
14.11	Time, etc. – A More Detailed Look	671
	14.11.1 Metrics	672
14.12	Response Time	672
	14.12.1 Polled Loops	673
	14.12.2 Co-routine	674
	14.12.3 Interrupt-Driven Environment	674
	14.12.3.1 Preemptive Schedule	674
	14.12.3.2 Nonpreemptive Schedule	675
	14.12.4 Meeting Real-Time Constraints	675
	14.12.4.1 Deadline Monotonic Analysis	676
	14.12.4.2 Vocabulary	676
	14.12.4.3 Analysis	677
	14.12.4.4 Priority Ceiling Protocol	678
14.13	Time Loading	679
	14.13.1 Instruction Counting	680
	14.13.2 Simulation	680
	14.13.2.1 Models	680
	14.13.3 Timers	681
	14.13.4 Instrumentation	681
14.14	Memory Loading	682
	14.14.1 Memory Map	682
	14.14.2 Designing a Memory Map	683
	14.14.2.1 Instruction/Firmware Area	683
	14.14.2.2 RAM Area	683
	14.14.2.3 Stack Area	684
14.15	Evaluating Performance	684
	14.15.1 Early Stages	685
	14.15.2 Mid-stages	685
	14.15.3 Later Stages	685
14.16	Thoughts on Performance Optimization	685
	14.16.1 Questions to Ask	685

14.17	Performance Optimization	686
14.17.1	Common Mistakes	686
14.18	Tricks of the Trade	687
14.19	Hardware Accelerators	692
14.20	Introduction – Target Low Power	693
14.21	Low Power – A High-Level View	693
14.21.1	Zero Power Consumption	694
14.21.2	Static Power Consumption	694
14.21.2.1	Sources of Static Power Consumption	694
14.21.2.2	Addressing Static Power Consumption	695
14.21.3	Dynamic Power Consumption	696
14.21.3.1	Sources of Dynamic Power Consumption – Hardware	697
14.21.3.2	Sources of Dynamic Power Consumption – Software	701
14.22	Addressing Dynamic Power Consumption – Hardware	702
14.22.1	Power Management Schemes – Hardware	703
14.22.2	Advanced Configuration and Power Interface (ACPI)	704
14.22.3	Dynamic Voltage and Frequency Scaling	705
14.23	Addressing Dynamic Power Consumption – Software	706
14.23.1	Measuring Power Consumption	707
14.23.2	Caches and Performance	708
14.24	Trade-Offs	709
14.25	Summary	709
14.26	Review Questions	710
14.27	Thought Questions	710
14.28	Problems	711

Part 4 Developing the Foundation

15	Working Outside of the Processor I: A Model of Interprocess Communication	715
15.1	Communication and Synchronization with the Outside World	715
15.2	First Steps: Understanding the Problem	716
15.3	Interprocess Interaction Revisited	717
15.4	The Model	719
15.4.1	Information	719
15.4.2	Places	720
15.4.3	Control and Synchronization	720
15.4.4	Transport	721
15.5	Exploring the Model	722
15.5.1	The Transport Mechanism	722
15.5.1.1	The Interconnection Topology	722
15.5.1.2	Star	723
15.5.1.3	Ring	724
15.5.2	Control and Synchronization	727

15.5.3	Information Flow	727
15.5.3.1	Direction of Flow	727
15.5.3.2	Magnitude of the Flow	728
15.5.3.3	I/O Timing	729
15.5.3.4	Software Device Drivers	729
15.5.4	Places	729
15.6	Summary	730
15.7	Review Questions	730
15.8	Thought Questions	730
16	Working Outside of the Processor I: Refining the Model of Interprocess Communication	733
16.1	Communication and Synchronization with the Outside World	733
16.2	The Local Device Model	734
16.2.1	Control, Synchronization, and Places	735
16.2.1.1	A Serial Model	735
16.2.1.2	A Parallel Model	736
16.2.2	Information – Data	737
16.2.3	Transport	737
16.3	Implementing the Local Device Model – A First Step	737
16.3.1	An Overview	737
16.3.1.1	Main Memory Address Space	738
16.3.1.2	I/O Ports	738
16.3.1.3	Peripheral Processor	739
16.3.2	Main Memory Address Space – Memory-Mapped I/O	739
16.3.2.1	Address Bus	740
16.3.2.2	Data Bus	740
16.3.2.3	Control Bus	740
16.3.2.4	Read	741
16.3.2.5	Write	741
16.3.2.6	Bidirectional Bus	742
16.3.3	I/O Ports – Program-Controlled I/O	744
16.3.4	The Peripheral Processor	744
16.4	Implementing the Local Device Model – A Second Step	745
16.4.1	Information Interchange – An Event	745
16.4.2	Information Interchange – A Shared Variable	746
16.4.3	Information Interchange – A Message	746
16.5	Implementing an Event-Driven Exchange – Interrupts and Polling	747
16.5.1	Polling	747
16.5.2	Interrupts	749
16.5.2.1	Single Interrupt Line with Single Device	749
16.5.2.2	Single Interrupt Line with Multiple Devices	749
16.5.2.3	Multiple Interrupt Lines	753
16.5.3	Masking Interrupts	754
16.6	A Message	755
16.6.1	Asynchronous Information Exchange	756
16.6.1.1	Strobes	756
16.6.1.2	Strobe with Acknowledge	757

	16.6.1.3	Full Handshake	757
	16.6.1.4	Resynchronization	758
	16.6.1.5	Analysis	759
	16.6.2	Synchronous Information Exchange	759
	16.6.2.1	Bit Synchronization	759
16.7		The Remote Device Model	762
	16.7.1	Places and Information	764
	16.7.2	Control and Synchronization	764
	16.7.3	Transport	764
16.8		Implementing the Remote Device Model – A First Step	764
	16.8.1	The OSI and TCP/IP Protocol Stacks	765
	16.8.1.1	OSI – Physical Layer	767
	16.8.1.2	OSI – Data Link Layer	767
	16.8.1.3	TCP/IP – Host to Network	767
	16.8.1.4	OSI – Network Layer	767
	16.8.1.5	TCP/IP – Internet Layer	767
	16.8.1.6	OSI – Transport Layer	767
	16.8.1.7	TCP/IP – Transport Layer	768
	16.8.1.8	OSI – Session Layer	768
	16.8.1.9	OSI – Presentation Layer	768
	16.8.1.10	OSI – Application Layer	768
	16.8.1.11	TCP/IP – Application Layer	768
	16.8.2	The Models	769
	16.8.2.1	The Client–Server Model	769
	16.8.2.2	The Peer-to-Peer Model	769
	16.8.2.3	The Group Multicast Model	770
16.9		Implementing the Remote Device Model – A Second Step	771
	16.9.1	The Messages	771
	16.9.2	The Message Structure	771
	16.9.3	Message Control and Synchronization	772
	16.9.3.1	The Header	772
	16.9.3.2	The Transfer Scheme	772
16.10		Working with Remote Tasks	773
	16.10.1	Preliminary Thoughts on Working with Remote Tasks	773
	16.10.1.1	Local vs. Remote Addresses and Data	774
	16.10.1.2	Repeated Task Execution	774
	16.10.1.3	Node Failure, Link Failure, Message Loss	775
	16.10.2	Procedures and Remote Procedures	775
	16.10.2.1	Calling a Remote Procedure – RPC Semantics	775
	16.10.2.2	The Transport	777
	16.10.2.3	Message Source and Destination	777
	16.10.2.4	The Protocol	778
	16.10.2.5	RPC Interface Definition	779
	16.10.3	Node Failure, Link Failure, Message Loss	779
	16.10.3.1	Node Failure and Loss	779
	16.10.3.2	Message Loss	780
16.11		Group Multicast Revisited	781
	16.11.1	Atomic Multicast	781
	16.11.2	Reliable Multicast	781

16.12	Connecting to Distributed Processes – Pipes, Sockets, and Streams	782
16.12.1	Pipes	782
16.12.2	Sockets	782
16.12.3	Stream Communication	784
16.13	Summary	784
16.14	Review Questions	784
16.15	Thought Questions	785
16.16	Problems	786
17	Working Outside of the Processor II: Interfacing to Local Devices	789
17.1	Shared Variable I/O – Interfacing to Peripheral Devices	789
17.2	The Shared Variable Exchange	790
17.3	Generating Analog Signals	790
17.3.1	Binary Weighted Digital-to-Analog Converter	791
17.3.2	R/2R Ladder Digital-to-Analog Converter	794
17.4	Common Measurements	796
17.4.1	Voltage	796
17.4.2	Current	796
17.4.3	Resistance	797
17.5	Measuring Voltage	797
17.5.1	Dual Slope Analog-to-Digital Conversion	797
17.5.2	Successive Approximation Analog-to-Digital Conversion	801
17.5.2.1	Sample and Hold	803
17.5.3	VCO Analog-to-Digital Conversion	804
17.6	Measuring Resistance	806
17.7	Measuring Current	808
17.8	Measuring Temperature	808
17.8.1	Sensors	809
17.8.2	Making the Measurement	809
17.8.3	Working with Nonlinear Devices	810
17.9	Generating Digital Signals	812
17.9.1	Motors and Motor Control	812
17.9.2	DC Motors	812
17.9.3	Servo Motors	814
17.9.4	Stepper Motors	814
17.10	Controlling DC and Servo Motors	815
17.10.1	DC Motors	815
17.10.2	Servo Motors	816
17.10.3	Controlling Stepper Motors	817
17.10.4	Motor Drive Circuitry	819
17.10.5	Motor Drive Noise	821
17.11	LEDs and LED Displays	821
17.11.1	Individual LEDs	821
17.11.2	Multi-LED Displays	822
17.12	Measuring Digital Signals	824
17.12.1	The Approach	824
17.12.2	Working with Asynchronous Signals	825
17.12.3	Buffering Input Signals	826

17.12.4	Inside vs. Outside	827
17.12.5	Measuring Frequency and Time Interval	827
17.12.5.1	Measuring the Period – An Internal Implementation	827
17.12.5.2	Measuring the Period – An External Implementation	828
17.12.5.3	Counting for a Known Interval – An Internal Implementation	829
17.12.5.4	Counting for a Known Interval – An External Implementation	830
17.13	Summary	831
17.14	Review Questions	831
17.15	Thought Questions	832
17.16	Problems	832
18	Working Outside of the Processor III: Interfacing to Remote Devices	837
18.1	Common Network-Based I/O Architectures	837
18.2	Network-Based Systems	838
18.3	RS-232/EIA-232 – Asynchronous Serial Communication	838
18.3.1	Introduction	838
18.3.2	The EIA-232 Standard	839
18.3.2.1	What It Is ...	840
18.3.2.2	What they Think It Is ...	840
18.3.3	EIA-232 Addressing	842
18.3.4	Asynchronous Serial Communication	842
18.3.5	Configuring the Interface	842
18.3.6	Data Recovery and Timing	843
18.3.7	EIA-232 Interface Signals	844
18.3.8	An Implementation	845
18.4	The Universal Serial Bus – Synchronous Serial Communication	845
18.4.1	Background	846
18.4.2	The Universal Serial Bus Architecture	846
18.4.3	The Universal Serial Bus Protocol	847
18.4.4	USB Devices	848
18.4.5	Transfer Types	848
18.4.6	Device Descriptors	849
18.4.7	Network Configuration	850
18.4.8	USB Transactions	851
18.4.9	USB Interface Signals	852
18.4.10	The Physical Environment	853
18.4.10.1	Low-Speed Cables	853
18.4.10.2	High-Speed Cables	853
18.4.10.3	Cables and Cable Power	853
18.4.11	Detecting Device Attachment and Speed	854
18.4.12	Differential Pair Signaling	855
18.4.13	Implementation	855
18.5	I ² C – A Local Area Network	855
18.5.1	The Architecture	855