

Brock J. LaMeres

Introduction to Logic Circuits & Logic Design with Verilog



Third Edition

 Springer

INTRODUCTION TO LOGIC CIRCUITS & LOGIC DESIGN WITH VERILOG

INTRODUCTION TO LOGIC CIRCUITS & LOGIC DESIGN WITH VERILOG

3RD EDITION

Brock J. LaMeres

 Springer

Brock J. LaMeres
Department of Electrical & Computer Engineering
Montana State University
Bozeman, MT, USA

ISBN 978-3-031-43945-2 ISBN 978-3-031-43946-9 (eBook)
<https://doi.org/10.1007/978-3-031-43946-9>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2017, 2019, 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Preface

The overall goal of this book is to fill a void that has appeared in the instruction of digital circuits over the past decade due to the rapid abstraction of system design. Up until the mid-1980s, digital circuits were designed using *classical* techniques. Classical techniques relied heavily on manual design practices for the synthesis, minimization, and interfacing of digital systems. Corresponding to this design style, academic textbooks were developed that taught classical digital design techniques. Around 1990, large-scale digital systems began being designed using hardware description languages (HDL) and automated synthesis tools. Broad-scale adoption of this *modern design* approach spread through the industry during this decade. Around 2000, hardware description languages and the modern digital design approach began to be taught in universities, mainly at the senior and graduate level. There were a variety of reasons that the modern digital design approach did not penetrate the lower levels of academia during this time. First, the design and simulation tools were difficult to use and overwhelmed freshman and sophomore students. Second, the ability to implement the designs in a laboratory setting was infeasible. The modern design tools at the time were targeted at custom-integrated circuits, which are cost and time prohibitive to implement in a university setting. Between 2000 and 2005, rapid advances in programmable logic and design tools allowed the modern digital design approach to be implemented in a university setting, even in lower-level courses. This allowed students to learn the modern design approach based on HDLs and prototype their designs in real hardware, mainly Field Programmable Gate Arrays (FPGAs). This spurred an abundance of textbooks to be authored teaching hardware description languages and higher levels of design abstraction. This trend has continued until today. While abstraction is a critical tool for engineering design, the rapid movement toward teaching only the modern digital design techniques has left a void for freshman- and sophomore-level courses in digital circuitry. Legacy textbooks that teach the classical design approach are outdated and do not contain sufficient coverage of HDLs to prepare the students for follow-on classes. Newer textbooks that teach the modern digital design approach move immediately into high-level behavioral modeling with minimal or no coverage of the underlying hardware used to implement the systems. As a result, students are not being provided the resources to understand the fundamental hardware theory that lies beneath the modern abstraction such as interfacing, gate level implementation, and technology optimization. Students moving too rapidly into high levels of abstraction have little understanding of what is going on when they click the “compile & synthesize” button of their design tool. This leads to graduates who can model a breadth of different systems in an HDL but have no depth into how the system is implemented in hardware. This becomes problematic when an issue arises in a real design and there is no foundational knowledge for the students to fall back on in order to debug the problem.

This new book addresses the lower-level foundational void by providing a comprehensive, bottom-up, coverage of digital systems. The book begins with a description of lower-level hardware including binary representations, gate-level implementation, interfacing, and simple combinational logic design. Only after a foundation has been laid in the underlying hardware theory is the Verilog language introduced. The Verilog introduction gives only the basic concepts of the language in order to model, simulate, and synthesize combinational logic. This allows the students to gain familiarity with the language and the modern design approach without getting overwhelmed by the full capability of the language. The book then covers sequential logic and finite state machines at the structural level. Once this secondary foundation has been laid, the remaining capabilities of Verilog are presented that allow sophisticated, synchronous systems to be modeled. An entire chapter is then dedicated to examples of sequential system modeling, which allows the students to learn by example. The second part of the textbook introduces the details of programmable logic, semiconductor memory, and arithmetic circuits. The book culminates with a discussion of computer system design, which incorporates all of the

knowledge gained in the previous chapters. Each component of a computer system is described with an accompanying Verilog implementation, all while continually reinforcing the underlying hardware beneath the HDL abstraction.

Written the Way It Is Taught

The organization of this book is designed to follow the way in which the material is actually learned. Topics are presented only once sufficient background has been provided by earlier chapters to fully understand the material. An example of this *learning-oriented* organization is how the Verilog language is broken into two chapters. Chapter 5 presents an introduction to Verilog and the basic constructs to model combinational logic. This is an ideal location to introduce the language because the reader has just learned about combinational logic theory in Chap. 4. This allows the student to begin gaining experience using the Verilog simulation tools on basic combinational logic circuits. The more advanced constructs of Verilog such as sequential modeling and test benches are presented in Chap. 8 only after a thorough background in sequential logic is presented in Chap. 7. Another example of this learning-oriented approach is how arithmetic circuits are not introduced until Chap. 12. While technically the arithmetic circuits in Chap. 12 are combinational logic circuits and could be presented in Chap. 4, the student does not have the necessary background in Chap. 4 to fully understand the operation of the arithmetic circuitry, so its introduction is postponed.

This incremental, *just-in-time* presentation of material allows the book to follow the way the material is actually taught in the classroom. This design also avoids the need for the instructor to assign sections that move back and forth through the text. This not only reduces course design effort for the instructor but allows the student to know where they are in the sequence of learning. At any point, the student should know the material in prior chapters and be moving toward understanding the material in subsequent ones.

An additional advantage of this book's organization is that it supports giving the student hands-on experience with digital circuitry for courses with an accompanying laboratory component. The flow is designed to support lab exercises that begin using discrete logic gates on a breadboard and then move into HDL-based designs implemented on off-the-shelf FPGA boards. Using this approach to a laboratory experience gives the student experience with the basic electrical operation of digital circuits, interfacing, and HDL-based designs.

Learning Outcomes

Each chapter begins with an explanation of its learning objective followed by a brief preview of the chapter topics. The specific learning outcomes are then presented for the chapter in the form of concise statements about the measurable knowledge and/or skills the student will be able to demonstrate by the end of the chapter. Each section addresses a single, specific learning outcome. This eases the process of assessment and gives specific details on student performance. There are over 1000 assessment tools in the form of exercise problems and concept check questions that are tied directly to specific learning outcomes for both formative and summative assessment.

Teaching by Example

With nearly 250 worked examples, concept checks for each section, 200+ supporting figures, and 1000+ assessment problems, students are provided with multiple ways to learn. Each topic is described in a clear, concise written form with accompanying figures as necessary. This is then followed by annotated worked examples that match the form of the exercise problems at the end of each chapter. Additionally, concept check questions are placed at the end of each section in the book to measure the

student's general understanding of the material using a concept inventory assessment style. These features provide the student multiple ways to learn the material and build an understanding of digital circuitry.

Course Design

The book can be used in multiple ways. The first is to use the book to cover two, semester-based college courses in digital logic. The first course in this sequence is an *introduction to logic circuits* and covers Chaps. 1, 2, 3, 4, 5, 6, and 7. This introductory course, which is found in nearly all accredited electrical and computer engineering programs, gives students a basic foundation in digital hardware and interfacing. Chapters 1, 2, 3, 4, 5, 6, and 7 only cover relevant topics in digital circuits to make room for a thorough introduction to Verilog. At the end of this course, students have a solid foundation in digital circuits and are able to design and simulate Verilog models of concurrent and hierarchical systems. The second course in this sequence covers *logic design* using Chaps. 8, 9, 10, 11, 12, 13, and 14. In this second course, students learn the advanced features of Verilog such as procedural assignments, sequential behavioral modeling, system tasks, and test benches. This provides the basis for building larger digital systems such as registers, finite state machines, and arithmetic circuits. Chapter 13 brings all of the concepts together through the design of a simple 8-bit computer system that can be simulated and implemented using many off-the-shelf FPGA boards. Chapter 14 introduces the relatively complex topic of floating-point systems.

This book can also be used in a more accelerated digital logic course that reaches a higher level of abstraction in a single semester. This is accomplished by skipping some chapters and moving quickly through others. In this use model, it is likely that Chap. 2 on numbers systems and Chap. 3 on digital circuits would be quickly referenced but not covered in detail. Chapters 4 and 7 could also be covered quickly in order to move rapidly into Verilog modeling without spending significant time looking at the underlying hardware implementation. This approach allows a higher level of abstraction to be taught but provides the student with the reference material so that they can delve in the details of the hardware implementation if interested.

All exercise and concept problems that do not involve a Verilog model are designed so that they can be implemented as a multiple choice or numeric entry question in a standard course management system. This allows the questions to be automatically graded. For the Verilog design questions, it is expected that the students will upload their Verilog source files and screenshots of their simulation waveforms to the course management system for manual grading by the instructor or teaching assistant.

Instructor Resources

Instructors adopting this book can access a growing collection of supplementary learning resources including YouTube videos created by the author, a solutions manual, a laboratory manual, and Verilog test benches for all problems. Additional resources are made available as demand grows. The growing library of YouTube videos can provide supplementary learning materials for students or facilitate fully online or flipped delivery of this material. The videos are found at https://www.youtube.com/c/DigitalLogicProgramming_LaMeres. The solutions manual contains a graphic-rich description of select exercise problems. A complementary lab manual has also been developed to provide additional learning activities based on both the 74HC discrete logic family and an off-the-shelf FPGA board. This manual is provided separately from the book in order to support the ever-changing technology options available for laboratory exercises.

What's New in the Third Edition

The third edition adds a new chapter on floating-point numbers. By popular demand, this chapter was added to provide a more comprehensive understanding of modern computers. The chapter provides a low-level explanation of floating-point numbers including formation, standardization, conversions, and arithmetic operations. It then moves into Verilog modeling of basic arithmetic operations using a manual implementation approach. An additional 35 assessment problems and 4 concept checks are included in Chap. 14.

Bozeman, MT, USA

Brock J. LaMeres

Acknowledgments

For my mom (Kathleen). Thank you for the life you have given me. Your love and support have always been a constant that I can count on in good times and bad. Your positivity is contagious and touches everyone around you. You have blessed me with your patience and sense of humor, both of which have made this life much more enjoyable. I love you.

Contents

1: INTRODUCTION: ANALOG VERSUS DIGITAL	1
1.1 DIFFERENCES BETWEEN ANALOG AND DIGITAL SYSTEMS	1
1.2 ADVANTAGES OF DIGITAL SYSTEMS OVER ANALOG SYSTEMS	3
2: NUMBER SYSTEMS	7
2.1 POSITIONAL NUMBER SYSTEMS	7
2.1.1 <i>Generic Structure</i>	8
2.1.2 <i>Decimal Number System (Base 10)</i>	9
2.1.3 <i>Binary Number System (Base 2)</i>	9
2.1.4 <i>Octal Number System (Base 8)</i>	10
2.1.5 <i>Hexadecimal Number System (Base 16)</i>	10
2.2 BASE CONVERSION	11
2.2.1 <i>Converting to Decimal</i>	11
2.2.2 <i>Converting from Decimal</i>	14
2.2.3 <i>Converting Between 2^n Bases</i>	18
2.3 BINARY ARITHMETIC	22
2.3.1 <i>Addition (Carries)</i>	22
2.3.2 <i>Subtraction (Borrows)</i>	23
2.4 UNSIGNED AND SIGNED NUMBERS	25
2.4.1 <i>Unsigned Numbers</i>	25
2.4.2 <i>Signed Numbers</i>	26
3: DIGITAL CIRCUITRY AND INTERFACING	43
3.1 BASIC GATES	43
3.1.1 <i>Describing the Operation of a Logic Circuit</i>	43
3.1.2 <i>The Buffer</i>	45
3.1.3 <i>The Inverter</i>	46
3.1.4 <i>The AND Gate</i>	46
3.1.5 <i>The NAND Gate</i>	47
3.1.6 <i>The OR Gate</i>	47
3.1.7 <i>The NOR Gate</i>	47
3.1.8 <i>The XOR Gate</i>	48
3.1.9 <i>The XNOR Gate</i>	49
3.2 DIGITAL CIRCUIT OPERATION	50
3.2.1 <i>Logic Levels</i>	51
3.2.2 <i>Output DC Specifications</i>	51
3.2.3 <i>Input DC Specifications</i>	53
3.2.4 <i>Noise Margins</i>	53
3.2.5 <i>Power Supplies</i>	54
3.2.6 <i>Switching Characteristics</i>	56
3.2.7 <i>Data Sheets</i>	57

3.3	LOGIC FAMILIES	62
3.3.1	<i>Complementary Metal Oxide Semiconductors (CMOS)</i>	62
3.3.2	<i>Transistor-Transistor Logic (TTL)</i>	71
3.3.3	<i>The 7400 Series Logic Families</i>	73
3.4	DRIVING LOADS	77
3.4.1	<i>Driving Other Gates</i>	77
3.4.2	<i>Driving Resistive Loads</i>	79
3.4.3	<i>Driving LEDs</i>	81
4:	COMBINATIONAL LOGIC DESIGN	93
4.1	BOOLEAN ALGEBRA	93
4.1.1	<i>Operations</i>	94
4.1.2	<i>Axioms</i>	94
4.1.3	<i>Theorems</i>	95
4.2	COMBINATIONAL LOGIC ANALYSIS	111
4.2.1	<i>Finding the Logic Expression from a Logic Diagram</i>	111
4.2.2	<i>Finding the Truth Table from a Logic Diagram</i>	112
4.2.3	<i>Timing Analysis of a Combinational Logic Circuit</i>	113
4.3	COMBINATIONAL LOGIC SYNTHESIS	115
4.3.1	<i>Canonical Sum of Products</i>	115
4.3.2	<i>The Minterm List (Σ)</i>	116
4.3.3	<i>Canonical Product of Sums (POS)</i>	118
4.3.4	<i>The Maxterm List (Π)</i>	120
4.3.5	<i>Minterm and Maxterm List Equivalence</i>	122
4.4	LOGIC MINIMIZATION	124
4.4.1	<i>Algebraic Minimization</i>	124
4.4.2	<i>Minimization Using Karnaugh Maps</i>	125
4.4.3	<i>Don't Cares</i>	137
4.4.4	<i>Using XOR Gates</i>	138
4.5	TIMING HAZARDS AND GLITCHES	141
5:	VERILOG (PART 1)	153
5.1	HISTORY OF HARDWARE DESCRIPTION LANGUAGES	154
5.2	HDL ABSTRACTION	157
5.3	THE MODERN DIGITAL DESIGN FLOW	160
5.4	VERILOG CONSTRUCTS	163
5.4.1	<i>Data Types</i>	164
5.4.2	<i>The Module</i>	168
5.4.3	<i>Verilog Operators</i>	171
5.5	MODELING CONCURRENT FUNCTIONALITY IN VERILOG	176
5.5.1	<i>Continuous Assignment</i>	176
5.5.2	<i>Continuous Assignment with Logical Operators</i>	176
5.5.3	<i>Continuous Assignment with Conditional Operators</i>	177
5.5.4	<i>Continuous Assignment with Delay</i>	179

5.6	STRUCTURAL DESIGN AND HIERARCHY	182
5.6.1	<i>Lower-Level Module Instantiation</i>	182
5.6.2	<i>Gate Level Primitives</i>	184
5.6.3	<i>User-Defined Primitives</i>	185
5.6.4	<i>Adding Delay to Primitives</i>	186
5.7	OVERVIEW OF SIMULATION TEST BENCHES	187
6:	MSI LOGIC	193
6.1	DECODERS	193
6.1.1	<i>Example: One-Hot Decoder</i>	194
6.1.2	<i>Example: 7-Segment Display Decoder</i>	196
6.2	ENCODERS	200
6.2.1	<i>Example: One-Hot Binary Encoder</i>	200
6.3	MULTIPLEXERS	202
6.4	DEMULTIPLEXERS	205
7:	SEQUENTIAL LOGIC DESIGN	211
7.1	SEQUENTIAL LOGIC STORAGE DEVICES	211
7.1.1	<i>The Cross-Coupled Inverter Pair</i>	211
7.1.2	<i>Metastability</i>	212
7.1.3	<i>The SR Latch</i>	214
7.1.4	<i>The S'R' Latch</i>	217
7.1.5	<i>SR Latch with Enable</i>	219
7.1.6	<i>The D-Latch</i>	222
7.1.7	<i>The D-Flip-Flop</i>	223
7.2	SEQUENTIAL LOGIC TIMING CONSIDERATIONS	227
7.3	COMMON CIRCUITS BASED ON SEQUENTIAL STORAGE DEVICES	228
7.3.1	<i>Toggle Flop Clock Divider</i>	228
7.3.2	<i>Ripple Counter</i>	229
7.3.3	<i>Switch Debouncing</i>	230
7.3.4	<i>Shift Registers</i>	234
7.4	FINITE-STATE MACHINES	236
7.4.1	<i>Describing the Functionality of an FSM</i>	236
7.4.2	<i>Logic Synthesis for an FSM</i>	238
7.4.3	<i>FSM Design Process Overview</i>	245
7.4.4	<i>FSM Design Examples</i>	246
7.5	COUNTERS	254
7.5.1	<i>2-Bit Binary Up Counter</i>	254
7.5.2	<i>2-Bit Binary Up/Down Counter</i>	256
7.5.3	<i>2-Bit Gray Code Up Counter</i>	258
7.5.4	<i>2-Bit Gray Code Up/Down Counter</i>	260
7.5.5	<i>3-Bit One-Hot Up Counter</i>	262
7.5.6	<i>3-Bit One-Hot Up/Down Counter</i>	264
7.6	FINITE-STATE MACHINE'S RESET CONDITION	267

7.7 SEQUENTIAL LOGIC ANALYSIS	268
7.7.1 Finding the State Equations and Output Logic Expressions of an FSM	268
7.7.2 Finding the State Transition Table of an FSM	269
7.7.3 Finding the State Diagram of an FSM	270
7.7.4 Determining the Maximum Clock Frequency of an FSM	271
8: VERILOG (PART 2)	287
8.1 PROCEDURAL ASSIGNMENT	287
8.1.1 Procedural Blocks	287
8.1.2 Procedural Statements	290
8.1.3 Statement Groups	295
8.1.4 Local Variables	295
8.2 CONDITIONAL PROGRAMMING CONSTRUCTS	296
8.2.1 if-else Statements	296
8.2.2 case Statements	297
8.2.3 casez and casex Statements	299
8.2.4 forever Loops	299
8.2.5 while Loops	299
8.2.6 repeat Loops	300
8.2.7 for Loops	300
8.2.8 disable	301
8.3 SYSTEM TASKS	302
8.3.1 Text Output	302
8.3.2 File Input/Output	303
8.3.3 Simulation Control and Monitoring	305
8.4 TEST BENCHES	306
8.4.1 Common Stimulus Generation Techniques	307
8.4.2 Printing Results to the Simulator Transcript	308
8.4.3 Automatic Result Checking	309
8.4.4 Using Loops to Generate Stimulus	311
8.4.5 Using External Files in Test Benches	312
9: BEHAVIORAL MODELING OF SEQUENTIAL LOGIC	319
9.1 MODELING SEQUENTIAL STORAGE DEVICES IN VERILOG	319
9.1.1 D-Latch	319
9.1.2 D-Flip-Flop	320
9.1.3 D-Flip-Flop with Asynchronous Reset	320
9.1.4 D-Flip-Flop with Asynchronous Reset and Preset	321
9.1.5 D-Flip-Flop with Synchronous Enable	322
9.2 MODELING FINITE STATE MACHINES IN VERILOG	323
9.2.1 Modeling the States	325
9.2.2 The State Memory Block	325
9.2.3 The Next-State Logic Block	325
9.2.4 The Output Logic Block	326
9.2.5 Changing the State Encoding Approach	328

9.3	FSM DESIGN EXAMPLES IN VERILOG	329
9.3.1	<i>Serial Bit Sequence Detector in Verilog</i>	329
9.3.2	<i>Vending Machine Controller in Verilog</i>	331
9.3.3	<i>2-Bit, Binary Up/Down Counter in Verilog</i>	333
9.4	MODELING COUNTERS IN VERILOG	335
9.4.1	<i>Counters in Verilog Using a Single Procedural Block</i>	335
9.4.2	<i>Counters with Range Checking</i>	336
9.4.3	<i>Counters with Enables in Verilog</i>	336
9.4.4	<i>Counters with Loads</i>	337
9.5	RTL MODELING	338
9.5.1	<i>Modeling Registers in Verilog</i>	338
9.5.2	<i>Registers as Agents on a Data Bus</i>	339
9.5.3	<i>Shift Registers in Verilog</i>	341
10:	MEMORY	347
10.1	MEMORY ARCHITECTURE AND TERMINOLOGY	347
10.1.1	<i>Memory Map Model</i>	347
10.1.2	<i>Volatile vs. Nonvolatile Memory</i>	348
10.1.3	<i>Read Only vs. Read/Write Memory</i>	348
10.1.4	<i>Random Access vs. Sequential Access</i>	348
10.2	NONVOLATILE MEMORY TECHNOLOGY	349
10.2.1	<i>ROM Architecture</i>	349
10.2.2	<i>Mask Read Only Memory</i>	352
10.2.3	<i>Programmable Read Only Memory</i>	353
10.2.4	<i>Erasable Programmable Read Only Memory</i>	354
10.2.5	<i>Electrically Erasable Programmable Read Only Memory</i>	356
10.2.6	<i>FLASH Memory</i>	357
10.3	VOLATILE MEMORY TECHNOLOGY	358
10.3.1	<i>Static Random-Access Memory</i>	358
10.3.2	<i>Dynamic Random-Access Memory</i>	361
10.4	MODELING MEMORY WITH VERILOG	368
10.4.1	<i>Read Only Memory in Verilog</i>	368
10.4.2	<i>Read/Write Memory in Verilog</i>	369
11:	PROGRAMMABLE LOGIC	375
11.1	PROGRAMMABLE ARRAYS	375
11.1.1	<i>Programmable Logic Array (PLA)</i>	375
11.1.2	<i>Programmable Array Logic (PAL)</i>	376
11.1.3	<i>Generic Array Logic (GAL)</i>	377
11.1.4	<i>Hard Array Logic (HAL)</i>	378
11.1.5	<i>Complex Programmable Logic Devices (CPLD)</i>	378
11.2	FIELD PROGRAMMABLE GATE ARRAYS (FPGAs)	379
11.2.1	<i>Configurable Logic Block (or Logic Element)</i>	380
11.2.2	<i>Look-Up Tables (LUTs)</i>	381
11.2.3	<i>Programmable Interconnect Points (PIPs)</i>	384

11.2.4	<i>Input/Output Block (IOBs)</i>	385
11.2.5	<i>Configuration Memory</i>	386
12:	ARITHMETIC CIRCUITS	389
12.1	ADDITION	389
12.1.1	<i>Half Adders</i>	389
12.1.2	<i>Full Adders</i>	390
12.1.3	<i>Ripple Carry Adder</i>	392
12.1.4	<i>Carry Look Ahead Adder</i>	394
12.1.5	<i>Adders in Verilog</i>	397
12.2	SUBTRACTION	402
12.3	MULTIPLICATION	405
12.3.1	<i>Unsigned Multiplication</i>	405
12.3.2	<i>A Simple Circuit to Multiply by Powers of Two</i>	408
12.3.3	<i>Signed Multiplication</i>	408
12.4	DIVISION	411
12.4.1	<i>Unsigned Division</i>	411
12.4.2	<i>A Simple Circuit to Divide by Powers of Two</i>	414
12.4.3	<i>Signed Division</i>	415
13:	COMPUTER SYSTEM DESIGN	419
13.1	COMPUTER HARDWARE	419
13.1.1	<i>Program Memory</i>	420
13.1.2	<i>Data Memory</i>	420
13.1.3	<i>Input/Output Ports</i>	420
13.1.4	<i>Central Processing Unit</i>	421
13.1.5	<i>A Memory-Mapped System</i>	422
13.2	COMPUTER SOFTWARE	424
13.2.1	<i>Opcodes and Operands</i>	425
13.2.2	<i>Addressing Modes</i>	425
13.2.3	<i>Classes of Instructions</i>	426
13.3	COMPUTER IMPLEMENTATION: AN 8-BIT COMPUTER EXAMPLE	433
13.3.1	<i>Top-Level Block Diagram</i>	433
13.3.2	<i>Instruction Set Design</i>	434
13.3.3	<i>Memory System Implementation</i>	435
13.3.4	<i>CPU Implementation</i>	439
13.4	ARCHITECTURE CONSIDERATIONS	460
13.4.1	<i>Von Neumann Architecture</i>	460
13.4.2	<i>Harvard Architecture</i>	460
14:	FLOATING-POINT SYSTEMS	465
14.1	OVERVIEW OF FLOATING-POINT NUMBERS	465
14.1.1	<i>Limitations of Fixed-Point Numbers</i>	465
14.1.2	<i>The Anatomy of a Floating-Point Number</i>	466
14.1.3	<i>The IEEE 754 Standard</i>	467

14.1.4 Single-Precision Floating-Point Representation (32-Bit)	467
14.1.5 Double-Precision Floating-Point Representation (64-Bit)	471
14.1.6 IEEE 754 Special Values	474
14.1.7 IEEE 754 Rounding Types	476
14.1.8 Other Capabilities of the IEEE 754 Standard	477
14.2 IEEE 754 BASE CONVERSIONS	478
14.2.1 Converting from Decimal into IEEE 754 Single-Precision Numbers	478
14.2.2 Converting from IEEE 754 Single-Precision Numbers into Decimal	481
14.3 FLOATING-POINT ARITHMETIC	482
14.3.1 Addition and Subtraction of IEEE 754 Numbers	482
14.3.2 Multiplication and Division of IEEE 754 Numbers	490
14.4 FLOATING-POINT MODELING IN VERILOG	496
14.4.1 Modeling Floating-Point Addition in Verilog	496
14.4.2 Modeling Floating-Point Subtraction in Verilog	501
14.4.3 Modeling Floating-Point Multiplication in Verilog	505
14.4.4 Modeling Floating-Point Division in Verilog	508
APPENDIX A: LIST OF WORKED EXAMPLES	515
APPENDIX B: CONCEPT CHECK SOLUTIONS	521
INDEX	523