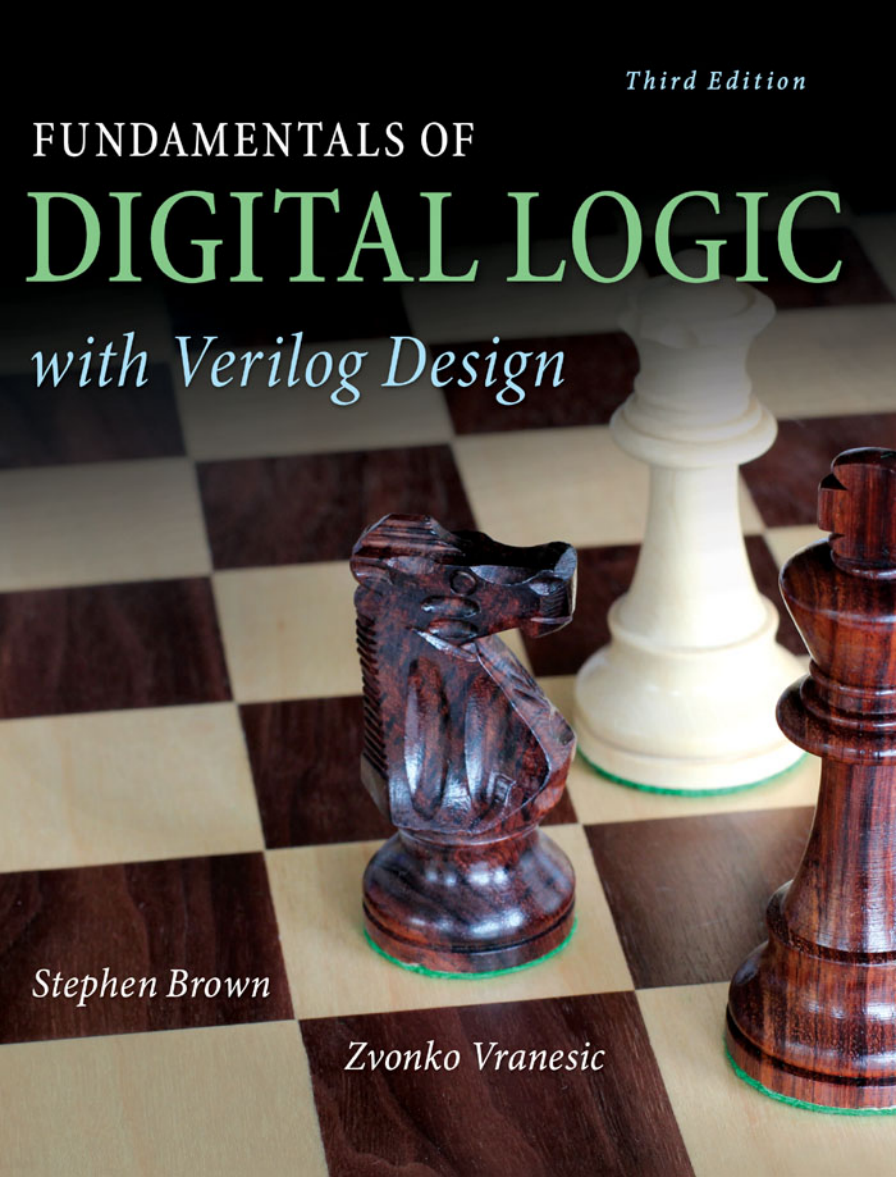
Third Edition
FUNDAMENTALS OF
DIGITAL LOGIC
with Verilog Design
Stephen Brown
Zvonko Vranesic

FUNDAMENTALS OF DIGITAL LOGIC WITH VERILOG DESIGN

THIRD EDITION

Stephen Brown and Zvonko Vranesic
Department of Electrical and Computer Engineering
University of Toronto

The McGraw-Hill Companies

FUNDAMENTALS OF DIGITAL LOGIC WITH VERILOG DESIGN, THIRD EDITION

Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020. Copyright © 2014 by The McGraw-Hill Companies, Inc. All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of The McGraw-Hill Companies, Inc., including, but not limited to, in any network or other electronic storage or transmission, or broadcast for distance learning.

Some ancillaries, including electronic and print components, may not be available to customers outside the United States.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 1 0 9 8 7 6 5 4 3

ISBN 978–0–07–338054–4
MHID 0–07–338054–7

Managing Director: *Thomas Timp*
Director: *Michael Lange*
Global Publisher: *Raghothaman Srinivasan*
Developmental Editor: *Vincent Bradshaw*
Marketing Manager: *Curt Reynolds*
Director, Content Production: *Terri Schiesl*
Senior Project Manager: *Melissa M. Leick*
Buyer: *Susan K. Culbertson*
Media Project Manager: *Prashanthi Nadipalli*
Cover Design: *Studio Montage, St. Louis, Missouri*
(USE) Cover Image: *Steven Brown and Zvonko Vranesic*
Compositor: *Techsetters, Inc.*
Typeface: *10/12 Times Roman*
Printer: *R. R. Donnelley, Crawfordsville, IN*

Library of Congress Cataloging-in-Publication Data

Brown, Stephen.
Fundamentals of digital logic with Verilog design / Stephen Brown and Zvonko Vranesic. — Third edition.
pages cm
ISBN 978–0–07–338054–4 (alk. paper)
1. Logic circuits—Design and construction—Data processing. 2. Verilog (Computer hardware description language). 3. Computer-aided design. I. Vranesic, Zvonko G. II. Title.

TK7868.L6B76 2014
621.39′2–dc23 2012042163

www.mhhe.com

To Susan and Anne

This page intentionally left blank

About the Authors

Stephen Brown received his B.A.Sc. degree in Electrical Engineering from the University of New Brunswick, Canada, and the M.A.Sc. and Ph.D. degrees in Electrical Engineering from the University of Toronto. He joined the University of Toronto faculty in 1992, where he is now a Professor in the Department of Electrical & Computer Engineering. He is also the Director of the worldwide University Program at Altera Corporation.

His research interests include field-programmable VLSI technology and computer architecture. He won the Canadian Natural Sciences and Engineering Research Council's 1992 Doctoral Prize for the best Ph.D. thesis in Canada and has published more than 100 scientific research papers.

He has won five awards for excellence in teaching electrical engineering, computer engineering, and computer science courses. He is a coauthor of two other books: *Fundamentals of Digital Logic with VHDL Design*, 3rd ed. and *Field-Programmable Gate Arrays*.

Zvonko Vranesic received his B.A.Sc., M.A.Sc., and Ph.D. degrees, all in Electrical Engineering, from the University of Toronto. From 1963–1965 he worked as a design engineer with the Northern Electric Co. Ltd. in Bramalea, Ontario. In 1968 he joined the University of Toronto, where he is now a Professor Emeritus in the Departments of Electrical & Computer Engineering and Computer Science. During the 1978–1979 academic year, he was a Senior Visitor at the University of Cambridge, England, and during 1984–1985 he was at the University of Paris, 6. From 1995 to 2000 he served as Chair of the Division of Engineering Science at the University of Toronto. He is also involved in research and development at the Altera Toronto Technology Center.

His current research interests include computer architecture and field-programmable VLSI technology.

He is a coauthor of four other books: *Computer Organization and Embedded Systems*, 6th ed.; *Fundamentals of Digital Logic with VHDL Design*, 3rd ed.; *Microcomputer Structures*; and *Field-Programmable Gate Arrays*. In 1990, he received the Wighton Fellowship for "innovative and distinctive contributions to undergraduate laboratory instruction." In 2004, he received the Faculty Teaching Award from the Faculty of Applied Science and Engineering at the University of Toronto.

He has represented Canada in numerous chess competitions. He holds the title of International Master.

PREFACE

This book is intended for an introductory course in digital logic design, which is a basic course in most electrical and computer engineering programs. A successful designer of digital logic circuits needs a good understanding of basic concepts and a firm grasp of the modern design approach that relies on computer-aided design (CAD) tools.

The main goals of the book are (1) to teach students the fundamental concepts in classical manual digital design and (2) illustrate clearly the way in which digital circuits are designed today, using CAD tools. Even though modern designers no longer use manual techniques, except in rare circumstances, our motivation for teaching such techniques is to give students an intuitive feeling for how digital circuits operate. Also, the manual techniques provide an illustration of the types of manipulations performed by CAD tools, giving students an appreciation of the benefits provided by design automation. Throughout the book, basic concepts are introduced by way of examples that involve simple circuit designs, which we perform using both manual techniques and modern CAD-tool-based methods. Having established the basic concepts, more complex examples are then provided, using the CAD tools. Thus our emphasis is on modern design methodology to illustrate how digital design is carried out in practice today.

TECHNOLOGY

The book discusses modern digital circuit implementation technologies. The emphasis is on programmable logic devices (PLDs), which is the most appropriate technology for use in a textbook for two reasons. First, PLDs are widely used in practice and are suitable for almost all types of digital circuit designs. In fact, students are more likely to be involved in PLD-based designs at some point in their careers than in any other technology. Second, circuits are implemented in PLDs by end-user programming. Therefore, students can be provided with an opportunity, in a laboratory setting, to implement the book's design examples in actual chips. Students can also simulate the behavior of their designed circuits on their own computers. We use the two most popular types of PLDs for targeting of designs: complex programmable logic devices (CPLDs) and field-programmable gate arrays (FPGAs).

We emphasize the use of a hardware description language in specifying the logic circuits, because the HDL-based approach is the most efficient design method to use in practice. We describe in detail the IEEE Standard Verilog HDL language and use it extensively in examples.

SCOPE OF THE BOOK

This edition of the book has been extensively restructured. All of the material that should be covered in a one-semester course is now included in Chapters 1 to 6. More advanced material is presented in Chapters 7 to 11.

Chapter 1 provides a general introduction to the process of designing digital systems. It discusses the key steps in the design process and explains how CAD tools can be used to automate many of the required tasks. It also introduces the representation of digital information.

Chapter 2 introduces the logic circuits. It shows how Boolean algebra is used to represent such circuits. It introduces the concepts of logic circuit synthesis and optimization, and shows how logic gates are used to implement simple circuits. It also gives the reader a first glimpse at Verilog, as an example of a hardware description language that may be used to specify the logic circuits.

Chapter 3 concentrates on circuits that perform arithmetic operations. It discusses numbers and shows how they can be manipulated using logic circuits. This chapter illustrates how Verilog can be used to specify the desired functionality and how CAD tools provide a mechanism for developing the required circuits.

Chapter 4 presents combinational circuits that are used as building blocks. It includes the encoder, decoder, and multiplexer circuits. These circuits are very convenient for illustrating the application of many Verilog constructs, giving the reader an opportunity to discover more advanced features of Verilog.

Storage elements are introduced in Chapter 5. The use of flip-flops to realize regular structures, such as shift registers and counters, is discussed. Verilog-specified designs of these structures are included.

Chapter 6 gives a detailed presentation of synchronous sequential circuits (finite state machines). It explains the behavior of these circuits and develops practical design techniques for both manual and automated design.

Chapter 7 is a discussion of a number of practical issues that arise in the design of real systems. It highlights problems often encountered in practice and indicates how they can be overcome. Examples of larger circuits illustrate a hierarchical approach in designing digital systems. Complete Verilog code for these circuits is presented.

Chapter 8 deals with more advanced techniques for optimized implementation of logic functions. It presents algorithmic techniques for optimization. It also explains how logic functions can be specified using a cubical representation as well as using binary decision diagrams.

Asynchronous sequential circuits are discussed in Chapter 9. While this treatment is not exhaustive, it provides a good indication of the main characteristics of such circuits. Even though the asynchronous circuits are not used extensively in practice, they provide an excellent vehicle for gaining a deeper understanding of the operation of digital circuits in general. They illustrate the consequences of propagation delays and race conditions that may be inherent in the structure of a circuit.

Chapter 10 presents a complete CAD flow that the designer experiences when designing, implementing, and testing a digital circuit.

Chapter 11 introduces the topic of testing. A designer of logic circuits has to be aware of the need to test circuits and should be conversant with at least the most basic aspects of testing.

Appendix A provides a complete summary of Verilog features. Although use of Verilog is integrated throughout the book, this appendix provides a convenient reference that the reader can consult from time to time when writing Verilog code.

The electronic aspects of digital circuits are presented in Appendix B. This appendix shows how the basic gates are built using transistors and presents various factors that affect circuit performance. The emphasis is on the latest technologies, with particular focus on CMOS technology and programmable logic devices.

WHAT CAN BE COVERED IN A COURSE

Much of the material in the book can be covered in 2 one-quarter courses. A good coverage of the most important material can be achieved in a single one-semester, or even a one-quarter course. This is possible only if the instructor does not spend too much time teaching the intricacies of Verilog and CAD tools. To make this approach possible, we organized the Verilog material in a modular style that is conducive to self-study. Our experience in teaching different classes of students at the University of Toronto shows that the instructor may spend only three to four lecture hours on Verilog, describing how the code should be structured, including the use of design hierarchy, using scalar and vector variables, and on the style of code needed to specify sequential circuits. The Verilog examples given in the book are largely self-explanatory, and students can understand them easily.

The book is also suitable for a course in logic design that does not include exposure to Verilog. However, some knowledge of Verilog, even at a rudimentary level, is beneficial to the students, and it is a great preparation for a job as a design engineer.

One-Semester Course

The following material should be covered in lectures:

- Chapter 1—all sections.
- Chapter 2—all sections.
- Chapter 3—Sections 3.1 to 3.5.
- Chapter 4—all sections.
- Chapter 5—all sections.
- Chapter 6—all sections.

One-Quarter Course

In a one-quarter course the following material can be covered:

- Chapter 1—all sections.
- Chapter 2—all sections.

- Chapter 3—Sections 3.1 to 3.3 and Section 3.5.
- Chapter 4—all sections.
- Chapter 5—all sections.
- Chapter 6—Sections 6.1 to 6.4.

VERILOG

Verilog is a complex language, which some instructors feel is too hard for beginning students to grasp. We fully appreciate this issue and have attempted to solve it. It is not necessary to introduce the entire Verilog language. In the book we present the important Verilog constructs that are useful for the design and synthesis of logic circuits. Many other language constructs, such as those that have meaning only when using the language for simulation purposes, are omitted. The Verilog material is introduced gradually, with more advanced features being presented only at points where their use can be demonstrated in the design of relevant circuits.

The book includes more than 120 examples of Verilog code. These examples illustrate how Verilog is used to describe a wide range of logic circuits, from those that contain only a few gates to those that represent digital systems such as a simple processor.

All of the examples of Verilog code presented in the book are provided on the Authors' website at

www.eecg.toronto.edu/~brown/Verilog_3e

SOLVED PROBLEMS

The chapters include examples of solved problems. They show how typical homework problems may be solved.

HOMEWORK PROBLEMS

More than 400 homework problems are provided in the book. Answers to selected problems are given at the back of the book. Solutions to all problems are available to instructors in the *Solutions Manual* that accompanies the book.

POWERPOINT SLIDES AND SOLUTIONS MANUAL

PowerPoint slides that contain all of the figures in the book are available on the Authors' website. Instructors can request access to these slides, as well as access to the Solutions Manual for the book, at:

www.mhhe.com/brownvranesic

CAD Tools

Modern digital systems are quite large. They contain complex logic circuits that would be difficult to design without using good CAD tools. Our treatment of Verilog should enable the reader to develop Verilog code that specifies logic circuits of varying degrees of complexity. To gain proper appreciation of the design process, it is highly beneficial to implement the designs using commercially-available CAD tools. Some excellent CAD tools are available free of charge. For example, the Altera Corporation has its Quartus II CAD software, which is widely used for implementing designs in programmable logic devices such as FPGAs. The Web Edition of the Quartus II software can be downloaded from Altera's website and used free of charge, without the need to obtain a license. In previous editions of this book a set of tutorials for using the Quartus II software was provided in the appendices. Those tutorials can now be found on the Authors' website. Another set of useful tutorials about Quartus II can be found on Altera's University Program website, which is located at www.altera.com/education/univ.

Acknowledgments

We wish to express our thanks to the people who have helped during the preparation of the book. Dan Vranesic produced a substantial amount of artwork. He and Deshanand Singh also helped with the preparation of the solutions manual. Tom Czajkowski helped in checking the answers to some problems. The reviewers, William Barnes, New Jersey Institute of Technology; Thomas Bradicich, North Carolina State University; James Clark, McGill University; Stephen DeWeerth, Georgia Institute of Technology; Sander Eller, Cal Poly Pomona; Clay Gloster, Jr., North Carolina State University (Raleigh); Carl Hamacher, Queen's University; Vincent Heuring, University of Colorado; Yu Hen Hu, University of Wisconsin; Wei-Ming Lin, University of Texas (San Antonio); Wayne Loucks, University of Waterloo; Kartik Mohanram, Rice University; Jane Morehead, Mississippi State University; Chris Myers, University of Utah; Vojin Oklobdzija, University of California (Davis); James Palmer, Rochester Institute of Technology; Gandhi Puvvada, University of Southern California; Teodoro Robles, Milwaukee School of Engineering; Tatyana Roziner, Boston University; Rob Rutenbar, Carnegie Mellon University; Eric Schwartz, University of Florida; Wen-Tsong Shiue, Oregon State University; Peter Simko, Miami University; Scott Smith, University of Missouri (Rolla); Arun Somani, Iowa State University; Bernard Svihel, University of Texas (Arlington); and Zeljko Zilic, McGill University provided constructive criticism and made numerous suggestions for improvements.

The support of McGraw-Hill people has been exemplary. We truly appreciate the help of Raghu Srinivasan, Vincent Bradshaw, Darlene Schueller, Curt Reynolds, and Michael Lange. We are also grateful for the excellent support in the typesetting of the book that has been provided by Techsetters, Inc.

Stephen Brown and Zvonko Vranesic

CONTENTS

Chapter 1

INTRODUCTION 1

1.1 Digital Hardware 2
- 1.1.1 Standard Chips 4
- 1.1.2 Programmable Logic Devices 5
- 1.1.3 Custom-Designed Chips 5

1.2 The Design Process 6
1.3 Structure of a Computer 8
1.4 Logic Circuit Design in This Book 8
1.5 Digital Representation of Information 11
- 1.5.1 Binary Numbers 12
- 1.5.2 Conversion between Decimal and Binary Systems 13
- 1.5.3 ASCII Character Code 14
- 1.5.4 Digital and Analog Information 16

1.6 Theory and Practice 16
Problems 18
References 19

Chapter 2

INTRODUCTION TO LOGIC CIRCUITS 21

2.1 Variables and Functions 22
2.2 Inversion 25
2.3 Truth Tables 26
2.4 Logic Gates and Networks 27
- 2.4.1 Analysis of a Logic Network 29

2.5 Boolean Algebra 33
- 2.5.1 The Venn Diagram 37
- 2.5.2 Notation and Terminology 42
- 2.5.3 Precedence of Operations 43

2.6 Synthesis Using AND, OR, and NOT Gates 43
- 2.6.1 Sum-of-Products and Product-of-Sums Forms 48

2.7 NAND and NOR Logic Networks 54
2.8 Design Examples 59
- 2.8.1 Three-Way Light Control 59
- 2.8.2 Multiplexer Circuit 60
- 2.8.3 Number Display 63

2.9 Introduction to CAD Tools 64
- 2.9.1 Design Entry 64
- 2.9.2 Logic Synthesis 66
- 2.9.3 Functional Simulation 67
- 2.9.4 Physical Design 67
- 2.9.5 Timing Simulation 67
- 2.9.6 Circuit Implementation 68
- 2.9.7 Complete Design Flow 68

2.10 Introduction to Verilog 68
- 2.10.1 Structural Specification of Logic Circuits 70
- 2.10.2 Behavioral Specification of Logic Circuits 72
- 2.10.3 Hierarchical Verilog Code 76
- 2.10.4 How *NOT* to Write Verilog Code 78

2.11 Minimization and Karnaugh Maps 78
2.12 Strategy for Minimization 87
- 2.12.1 Terminology 87
- 2.12.2 Minimization Procedure 89

2.13 Minimization of Product-of-Sums Forms 91
2.14 Incompletely Specified Functions 94
2.15 Multiple-Output Circuits 96
2.16 Concluding Remarks 101
2.17 Examples of Solved Problems 101
Problems 111
References 120

Chapter 3

NUMBER REPRESENTATION AND ARITHMETIC CIRCUITS 121

3.1 Positional Number Representation 122
- 3.1.1 Unsigned Integers 122
- 3.1.2 Octal and Hexadecimal Representations 123

3.2 Addition of Unsigned Numbers 125
- 3.2.1 Decomposed Full-Adder 129
- 3.2.2 Ripple-Carry Adder 129
- 3.2.3 Design Example 130

3.3 Signed Numbers 132
3.3.1 Negative Numbers 133
3.3.2 Addition and Subtraction 135
3.3.3 Adder and Subtractor Unit 138
3.3.4 Radix-Complement Schemes* 139
3.3.5 Arithmetic Overflow 143
3.3.6 Performance Issues 145
3.4 Fast Adders 145
3.4.1 Carry-Lookahead Adder 146
3.5 Design of Arithmetic Circuits Using CAD Tools 151
3.5.1 Design of Arithmetic Circuits Using Schematic Capture 151
3.5.2 Design of Arithmetic Circuits Using Verilog 152
3.5.3 Using Vectored Signals 155
3.5.4 Using a Generic Specification 156
3.5.5 Nets and Variables in Verilog 158
3.5.6 Arithmetic Assignment Statements 159
3.5.7 Module Hierarchy in Verilog Code 163
3.5.8 Representation of Numbers in Verilog Code 166
3.6 Multiplication 167
3.6.1 Array Multiplier for Unsigned Numbers 167
3.6.2 Multiplication of Signed Numbers 169
3.7 Other Number Representations 170
3.7.1 Fixed-Point Numbers 170
3.7.2 Floating-Point Numbers 172
3.7.3 Binary-Coded-Decimal Representation 174
3.8 Examples of Solved Problems 178
Problems 184
References 188

Chapter 4
COMBINATIONAL-CIRCUIT BUILDING BLOCKS 189

4.1 Multiplexers 190
4.1.1 Synthesis of Logic Functions Using Multiplexers 193
4.1.2 Multiplexer Synthesis Using Shannon's Expansion 196
4.2 Decoders 201
4.2.1 Demultiplexers 203
4.3 Encoders 205
4.3.1 Binary Encoders 205
4.3.2 Priority Encoders 205
4.4 Code Converters 208
4.5 Arithmetic Comparison Circuits 208
4.6 Verilog for Combinational Circuits 210
4.6.1 The Conditional Operator 210
4.6.2 The If-Else Statement 212
4.6.3 The Case Statement 215
4.6.4 The For Loop 221
4.6.5 Verilog Operators 223
4.6.6 The Generate Construct 228
4.6.7 Tasks and Functions 229
4.7 Concluding Remarks 232
4.8 Examples of Solved Problems 233
Problems 243
References 246

Chapter 5
FLIP-FLOPS, REGISTERS, AND COUNTERS 247

5.1 Basic Latch 249
5.2 Gated SR Latch 251
5.2.1 Gated SR Latch with NAND Gates 253
5.3 Gated D Latch 253
5.3.1 Effects of Propagation Delays 255
5.4 Edge-Triggered D Flip-Flops 256
5.4.1 Master-Slave D Flip-Flop 256
5.4.2 Other Types of Edge-Triggered D Flip-Flops 258
5.4.3 D Flip-Flops with Clear and Preset 260
5.4.4 Flip-Flop Timing Parameters 263
5.5 T Flip-Flop 263
5.6 JK Flip-Flop 264
5.7 Summary of Terminology 266
5.8 Registers 267
5.8.1 Shift Register 267
5.8.2 Parallel-Access Shift Register 267
5.9 Counters 269
5.9.1 Asynchronous Counters 269
5.9.2 Synchronous Counters 272
5.9.3 Counters with Parallel Load 276
5.10 Reset Synchronization 278
5.11 Other Types of Counters 280
5.11.1 BCD Counter 280
5.11.2 Ring Counter 280

5.11.3 Johnson Counter 283
5.11.4 Remarks on Counter Design 283
5.12 Using Storage Elements with CAD Tools 284
5.12.1 Including Storage Elements in Schematics 284
5.12.2 Using Verilog Constructs for Storage Elements 285
5.12.3 Blocking and Non-Blocking Assignments 288
5.12.4 Non-Blocking Assignments for Combinational Circuits 293
5.12.5 Flip-Flops with Clear Capability 293
5.13 Using Verilog Constructs for Registers and Counters 295
5.13.1 Flip-Flops and Registers with Enable Inputs 300
5.13.2 Shift Registers with Enable Inputs 302
5.14 Design Example 302
5.14.1 Reaction Timer 302
5.14.2 Register Transfer Level (RTL) Code 309
5.15 Timing Analysis of Flip-flop Circuits 310
5.15.1 Timing Analysis with Clock Skew 312
5.16 Concluding Remarks 314
5.17 Examples of Solved Problems 315
Problems 321
References 329

Chapter 6
SYNCHRONOUS SEQUENTIAL CIRCUITS 331

6.1 Basic Design Steps 333
6.1.1 State Diagram 333
6.1.2 State Table 335
6.1.3 State Assignment 336
6.1.4 Choice of Flip-Flops and Derivation of Next-State and Output Expressions 337
6.1.5 Timing Diagram 339
6.1.6 Summary of Design Steps 340
6.2 State-Assignment Problem 344
6.2.1 One-Hot Encoding 347
6.3 Mealy State Model 349
6.4 Design of Finite State Machines Using CAD Tools 354
6.4.1 Verilog Code for Moore-Type FSMs 355
6.4.2 Synthesis of Verilog Code 356
6.4.3 Simulating and Testing the Circuit 358
6.4.4 Alternative Styles of Verilog Code 359
6.4.5 Summary of Design Steps When Using CAD Tools 360
6.4.6 Specifying the State Assignment in Verilog Code 361
6.4.7 Specification of Mealy FSMs Using Verilog 363
6.5 Serial Adder Example 363
6.5.1 Mealy-Type FSM for Serial Adder 364
6.5.2 Moore-Type FSM for Serial Adder 367
6.5.3 Verilog Code for the Serial Adder 370
6.6 State Minimization 372
6.6.1 Partitioning Minimization Procedure 374
6.6.2 Incompletely Specified FSMs 381
6.7 Design of a Counter Using the Sequential Circuit Approach 383
6.7.1 State Diagram and State Table for a Modulo-8 Counter 383
6.7.2 State Assignment 384
6.7.3 Implementation Using D-Type Flip-Flops 385
6.7.4 Implementation Using JK-Type Flip-Flops 386
6.7.5 Example—A Different Counter 390
6.8 FSM as an Arbiter Circuit 393
6.9 Analysis of Synchronous Sequential Circuits 397
6.10 Algorithmic State Machine (ASM) Charts 401
6.11 Formal Model for Sequential Circuits 405
6.12 Concluding Remarks 407
6.13 Examples of Solved Problems 407
Problems 416
References 420

Chapter 7
DIGITAL SYSTEM DESIGN 421

7.1 Bus Structure 422
7.1.1 Using Tri-State Drivers to Implement a Bus 422
7.1.2 Using Multiplexers to Implement a Bus 424
7.1.3 Verilog Code for Specification of Bus Structures 426
7.2 Simple Processor 429

7.3 A Bit-Counting Circuit 441
7.4 Shift-and-Add Multiplier 446
7.5 Divider 455
7.6 Arithmetic Mean 466
7.7 Sort Operation 470
7.8 Clock Synchronization and Timing Issues 478
7.8.1 Clock Distribution 478
7.8.2 Flip-Flop Timing Parameters 481
7.8.3 Asynchronous Inputs to Flip-Flops 482
7.8.4 Switch Debouncing 483
7.9 Concluding Remarks 485
Problems 485
References 489

Chapter 8

OPTIMIZED IMPLEMENTATION OF LOGIC FUNCTIONS 491

8.1 Multilevel Synthesis 492
8.1.1 Factoring 493
8.1.2 Functional Decomposition 496
8.1.3 Multilevel NAND and NOR Circuits 502
8.2 Analysis of Multilevel Circuits 504
8.3 Alternative Representations of Logic Functions 510
8.3.1 Cubical Representation 510
8.3.2 Binary Decision Diagrams 514
8.4 Optimization Techniques Based on Cubical Representation 520
8.4.1 A Tabular Method for Minimization 521
8.4.2 A Cubical Technique for Minimization 529
8.4.3 Practical Considerations 536
8.5 Concluding Remarks 537
8.6 Examples of Solved Problems 537
Problems 546
References 549

Chapter 9

ASYNCHRONOUS SEQUENTIAL CIRCUITS 551

9.1 Asynchronous Behavior 552
9.2 Analysis of Asynchronous Circuits 556
9.3 Synthesis of Asynchronous Circuits 564
9.4 State Reduction 577
9.5 State Assignment 592
9.5.1 Transition Diagram 595
9.5.2 Exploiting Unspecified Next-State Entries 598
9.5.3 State Assignment Using Additional State Variables 602
9.5.4 One-Hot State Assignment 607
9.6 Hazards 608
9.6.1 Static Hazards 609
9.6.2 Dynamic Hazards 613
9.6.3 Significance of Hazards 614
9.7 A Complete Design Example 616
9.7.1 The Vending-Machine Controller 616
9.8 Concluding Remarks 621
9.9 Examples of Solved Problems 623
Problems 631
References 635

Chapter 10

COMPUTER AIDED DESIGN TOOLS 637

10.1 Synthesis 638
10.1.1 Netlist Generation 638
10.1.2 Gate Optimization 638
10.1.3 Technology Mapping 640
10.2 Physical Design 644
10.2.1 Placement 646
10.2.2 Routing 647
10.2.3 Static Timing Analysis 648
10.3 Concluding Remarks 650
References 651

Chapter 11

TESTING OF LOGIC CIRCUITS 653

11.1 Fault Model 654
11.1.1 Stuck-at Model 654
11.1.2 Single and Multiple Faults 655
11.1.3 CMOS Circuits 655
11.2 Complexity of a Test Set 655
11.3 Path Sensitizing 657
11.3.1 Detection of a Specific Fault 659
11.4 Circuits with Tree Structure 661
11.5 Random Tests 662

11.6 Testing of Sequential Circuits 665
11.6.1 Design for Testability 665
11.7 Built-in Self-Test 669
11.7.1 Built-in Logic Block Observer 673
11.7.2 Signature Analysis 675
11.7.3 Boundary Scan 676
11.8 Printed Circuit Boards 676
11.8.1 Testing of PCBs 678
11.8.2 Instrumentation 679
11.9 Concluding Remarks 680
Problems 680
References 683

Appendix A
VERILOG REFERENCE 685

A.1 Documentation in Verilog Code 686
A.2 White Space 686
A.3 Signals in Verilog Code 686
A.4 Identifier Names 687
A.5 Signal Values, Numbers, and Parameters 687
A.5.1 Parameters 688
A.6 Net and Variable Types 688
A.6.1 Nets 688
A.6.2 Variables 689
A.6.3 Memories 690
A.7 Operators 690
A.8 Verilog Module 692
A.9 Gate Instantiations 694
A.10 Concurrent Statements 696
A.10.1 Continuous Assignments 696
A.10.2 Using Parameters 697
A.11 Procedural Statements 698
A.11.1 Always and Initial Blocks 698
A.11.2 The If-Else Statement 700
A.11.3 Statement Ordering 701
A.11.4 The Case Statement 702
A.11.5 Casez and Casex Statements 703
A.11.6 Loop Statements 704
A.11.7 Blocking versus Non-blocking Assignments for Combinational Circuits 708
A.12 Using Subcircuits 709
A.12.1 Subcircuit Parameters 710
A.12.2 The Generate Capability 712
A.13 Functions and Tasks 713
A.14 Sequential Circuits 716
A.14.1 A Gated D Latch 717
A.14.2 D Flip-Flop 717
A.14.3 Flip-Flops with Reset 718
A.14.4 Registers 718
A.14.5 Shift Registers 720
A.14.6 Counters 721
A.14.7 An Example of a Sequential Circuit 722
A.14.8 Moore-Type Finite State Machines 723
A.14.9 Mealy-Type Finite State Machines 724
A.15 Guidelines for Writing Verilog Code 725
A.16 Concluding Remarks 731
References 731

Appendix B
IMPLEMENTATION TECHNOLOGY 733

B.1 Transistor Switches 734
B.2 NMOS Logic Gates 736
B.3 CMOS Logic Gates 739
B.3.1 Speed of Logic Gate Circuits 746
B.4 Negative Logic System 747
B.5 Standard Chips 749
B.5.1 7400-Series Standard Chips 749
B.6 Programmable Logic Devices 753
B.6.1 Programmable Logic Array (PLA) 754
B.6.2 Programmable Array Logic (PAL) 757
B.6.3 Programming of PLAs and PALs 759
B.6.4 Complex Programmable Logic Devices (CPLDs) 761
B.6.5 Field-Programmable Gate Arrays 764
B.7 Custom Chips, Standard Cells, and Gate Arrays 769
B.8 Practical Aspects 771
B.8.1 MOSFET Fabrication and Behavior 771
B.8.2 MOSFET On-Resistance 775
B.8.3 Voltage Levels in Logic Gates 776
B.8.4 Noise Margin 778
B.8.5 Dynamic Operation of Logic Gates 779
B.8.6 Power Dissipation in Logic Gates 782
B.8.7 Passing 1s and 0s Through Transistor Switches 784
B.8.8 Transmission Gates 786
B.8.9 Fan-in and Fan-out in Logic Gates 788
B.8.10 Tri-state Drivers 792

B.9 Static Random Access Memory (SRAM) 794
B.9.1 SRAM Blocks in PLDs 797
B.10 Implementation Details for SPLDs, CPLDs, and FPGAs 797
B.10.1 Implementation in FPGAs 804
B.11 Concluding Remarks 806
B.12 Examples of Solved Problems 807
Problems 814
References 823

ANSWERS 825

INDEX 839

chapter

1

INTRODUCTION

CHAPTER OBJECTIVES

In this chapter you will be introduced to:

- Digital hardware components
- An overview of the design process
- Binary numbers
- Digital representation of information

This book is about logic circuits—the circuits from which computers are built. Proper understanding of logic circuits is vital for today's electrical and computer engineers. These circuits are the key ingredient of computers and are also used in many other applications. They are found in commonly-used products like music and video players, electronic games, digital watches, cameras, televisions, printers, and many household appliances, as well as in large systems, such as telephone networks, Internet equipment, television broadcast equipment, industrial control units, and medical instruments. In short, logic circuits are an important part of almost all modern products.

The material in this book will introduce the reader to the many issues involved in the design of logic circuits. It explains the key ideas with simple examples and shows how complex circuits can be derived from elementary ones. We cover the classical theory used in the design of logic circuits because it provides the reader with an intuitive understanding of the nature of such circuits. But, throughout the book, we also illustrate the modern way of designing logic circuits using sophisticated *computer aided design (CAD)* software tools. The CAD methodology adopted in the book is based on the industry-standard design language called the *Verilog hardware description language*. Design with Verilog is first introduced in Chapter 2, and usage of Verilog and CAD tools is an integral part of each chapter in the book.

Logic circuits are implemented electronically, using transistors on an integrated circuit chip. Commonly available chips that use modern technology may contain more than a billion transistors, as in the case of some computer processors. The basic building blocks for such circuits are easy to understand, but there is nothing simple about a circuit that contains billions of transistors. The complexity that comes with large circuits can be handled successfully only by using highly-organized design techniques. We introduce these techniques in this chapter, but first we briefly describe the hardware technology used to build logic circuits.

1.1 DIGITAL HARDWARE

Logic circuits are used to build computer hardware, as well as many other types of products. All such products are broadly classified as *digital hardware*. The reason that the name *digital* is used will be explained in Section 1.5—it derives from the way in which information is represented in computers, as electronic signals that correspond to digits of information.

The technology used to build digital hardware has evolved dramatically over the past few decades. Until the 1960s logic circuits were constructed with bulky components, such as transistors and resistors that came as individual parts. The advent of integrated circuits made it possible to place a number of transistors, and thus an entire circuit, on a single chip. In the beginning these circuits had only a few transistors, but as the technology improved they became more complex. Integrated circuit chips are manufactured on a silicon wafer, such as the one shown in Figure 1.1. The wafer is cut to produce the individual chips, which are then placed inside a special type of chip package. By 1970 it was possible to implement all circuitry needed to realize a microprocessor on a single chip. Although early microprocessors had modest computing capability by today's standards, they opened the door for the information processing revolution by providing the means for implementation of affordable personal computers.

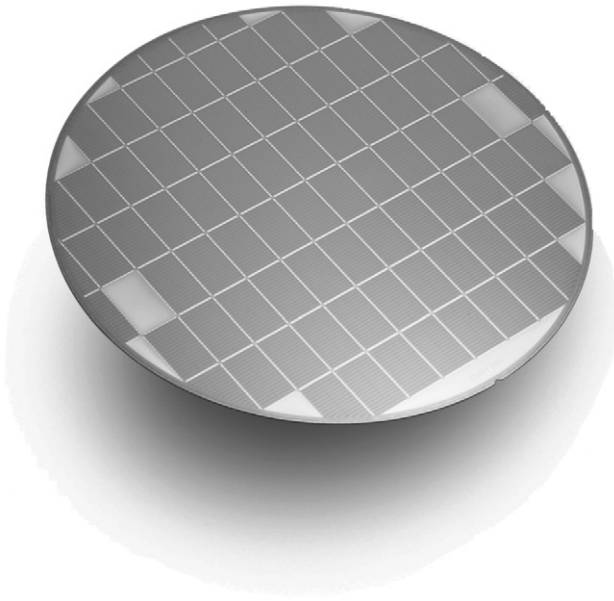

Figure 1.1 A silicon wafer (courtesy of Altera Corp.).

About 30 years ago Gordon Moore, chairman of Intel Corporation, observed that integrated circuit technology was progressing at an astounding rate, approximately doubling the number of transistors that could be placed on a chip every two years. This phenomenon, informally known as *Moore's law*, continues to the present day. Thus in the early 1990s microprocessors could be manufactured with a few million transistors, and by the late 1990s it became possible to fabricate chips that had tens of millions of transistors. Presently, chips can be manufactured containing billions of transistors.

Moore's law is expected to continue to hold true for a number of years. A consortium of integrated circuit associations produces a forecast of how the technology is expected to evolve. Known as the *International Technology Roadmap for Semiconductors (ITRS)* [1], this forecast discusses many aspects of technology, including the maximum number of transistors that can be manufactured on a single chip. A sample of data from the ITRS is given in Figure 1.2. It shows that chips with about 10 million transistors could be successfully manufactured in 1995, and this number has steadily increased, leading to today's chips with over a billion transistors. The roadmap predicts that chips with as many as 100 billion transistors will be possible by the year 2022. There is no doubt that this technology will have a huge impact on all aspects of people's lives.

The designer of digital hardware may be faced with designing logic circuits that can be implemented on a single chip or designing circuits that involve a number of chips placed on a *printed circuit board (PCB)*. Frequently, some of the logic circuits can be realized

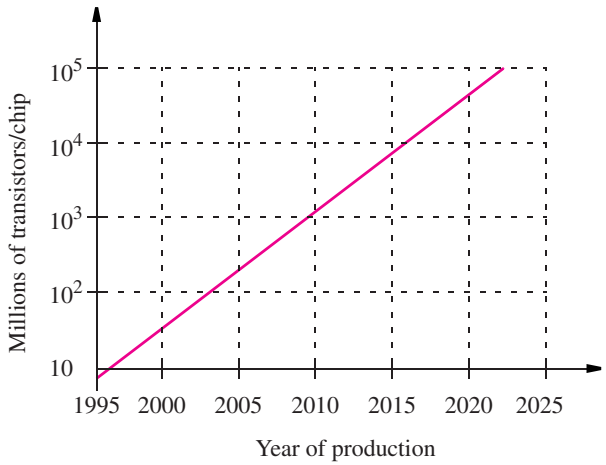

Figure 1.2 An estimate of the maximum number of transistors per chip over time.

in existing chips that are readily available. This situation simplifies the design task and shortens the time needed to develop the final product. Before we discuss the design process in detail, we should introduce the different types of integrated circuit chips that may be used.

There exists a large variety of chips that implement various functions that are useful in the design of digital hardware. The chips range from simple ones with low functionality to extremely complex chips. For example, a digital hardware product may require a microprocessor to perform some arithmetic operations, memory chips to provide storage capability, and interface chips that allow easy connection to input and output devices. Such chips are available from various vendors.

For many digital hardware products, it is also necessary to design and build some logic circuits from scratch. For implementing these circuits, three main types of chips may be used: standard chips, programmable logic devices, and custom chips. These are discussed next.

1.1.1 STANDARD CHIPS

Numerous chips are available that realize some commonly-used logic circuits. We will refer to these as *standard chips*, because they usually conform to an agreed-upon standard in terms of functionality and physical configuration. Each standard chip contains a small amount of circuitry (usually involving fewer than 100 transistors) and performs a simple function. To build a logic circuit, the designer chooses the chips that perform whatever functions are needed and then defines how these chips should be interconnected to realize a larger logic circuit.

Standard chips were popular for building logic circuits until the early 1980s. However, as integrated circuit technology improved, it became inefficient to use valuable space on PCBs for chips with low functionality. Another drawback of standard chips is that the functionality of each chip is fixed and cannot be changed.

1.1.2 PROGRAMMABLE LOGIC DEVICES

In contrast to standard chips that have fixed functionality, it is possible to construct chips that contain circuitry which can be configured by the user to implement a wide range of different logic circuits. These chips have a very general structure and include a collection of *programmable switches* that allow the internal circuitry in the chip to be configured in many different ways. The designer can implement whatever functions are required for a particular application by setting the programmable switches as needed. The switches are programmed by the end user, rather than when the chip is manufactured. Such chips are known as *programmable logic devices (PLDs)*.

PLDs are available in a wide range of sizes, and can be used to implement very large logic circuits. The most commonly-used type of PLD is known as a *field-programmable gate array (FPGA)*. The largest FPGAs contain billions of transistors [2, 3], and support the implementation of complex digital systems. An FPGA consists of a large number of small logic circuit elements, which can be connected together by using programmable switches in the FPGA. Because of their high capacity, and their capability to be tailored to meet the requirements of a specific application, FPGAs are widely used today.

1.1.3 CUSTOM-DESIGNED CHIPS

FPGAs are available as off-the-shelf components that can be purchased from different suppliers. Because they are programmable, they can be used to implement most logic circuits found in digital hardware. However, they also have a drawback in that the programmable switches consume valuable chip area and limit the speed of operation of implemented circuits. Thus in some cases FPGAs may not meet the desired performance or cost objectives. In such situations it is possible to design a chip from scratch; namely, the logic circuitry that must be included on the chip is designed first and then the chip is manufactured by a company that has the fabrication facilities. This approach is known as *custom* or *semi-custom design*, and such chips are often called *application-specific integrated circuits (ASICs)*.

The main advantage of a custom chip is that its design can be optimized for a specific task; hence it usually leads to better performance. It is possible to include a larger amount of logic circuitry in a custom chip than would be possible in other types of chips. The cost of producing such chips is high, but if they are used in a product that is sold in large quantities, then the cost per chip, amortized over the total number of chips fabricated, may be lower than the total cost of off-the-shelf chips that would be needed to implement the same function(s). Moreover, if a single chip can be used instead of multiple chips to achieve the same goal, then a smaller area is needed on a PCB that houses the chips in the final product. This results in a further reduction in cost.

A disadvantage of the custom-design approach is that manufacturing a custom chip often takes a considerable amount of time, on the order of months. In contrast, if an FPGA can be used instead, then the chips are programmed by the end user and no manufacturing delays are involved.

1.2 THE DESIGN PROCESS

The availability of computer-based tools has greatly influenced the design process in a wide variety of environments. For example, designing an automobile is similar in the general approach to designing a furnace or a computer. Certain steps in the development cycle must be performed if the final product is to meet the specified objectives.

The flowchart in Figure 1.3 depicts a typical development process. We assume that the process is to develop a product that meets certain expectations. The most obvious requirements are that the product must function properly, that it must meet an expected level of performance, and that its cost should not exceed a given target.

The process begins with the definition of product specifications. The essential features of the product are identified, and an acceptable method of evaluating the implemented features in the final product is established. The specifications must be tight enough to ensure that the developed product will meet the general expectations, but should not be unnecessarily constraining (that is, the specifications should not prevent design choices that may lead to unforeseen advantages).

From a complete set of specifications, it is necessary to define the general structure of an initial design of the product. This step is difficult to automate. It is usually performed by a human designer because there is no clear-cut strategy for developing a product's overall structure—it requires considerable design experience and intuition.

After the general structure is established, CAD tools are used to work out the details. Many types of CAD tools are available, ranging from those that help with the design of individual parts of the system to those that allow the entire system's structure to be represented in a computer. When the initial design is finished, the results must be verified against the original specifications. Traditionally, before the advent of CAD tools, this step involved constructing a physical model of the designed product, usually including just the key parts. Today it is seldom necessary to build a physical model. CAD tools enable designers to simulate the behavior of incredibly complex products, and such simulations are used to determine whether the obtained design meets the required specifications. If errors are found, then appropriate changes are made and the verification of the new design is repeated through simulation. Although some design flaws may escape detection via simulation, usually all but the most subtle problems are discovered in this way.

When the simulation indicates that the design is correct, a complete physical prototype of the product is constructed. The prototype is thoroughly tested for conformance with the specifications. Any errors revealed in the testing must be fixed. The errors may be minor, and often they can be eliminated by making small corrections directly on the prototype of the product. In case of large errors, it is necessary to redesign the product and repeat the steps explained above. When the prototype passes all the tests, then the product is deemed to be successfully designed and it can go into production.

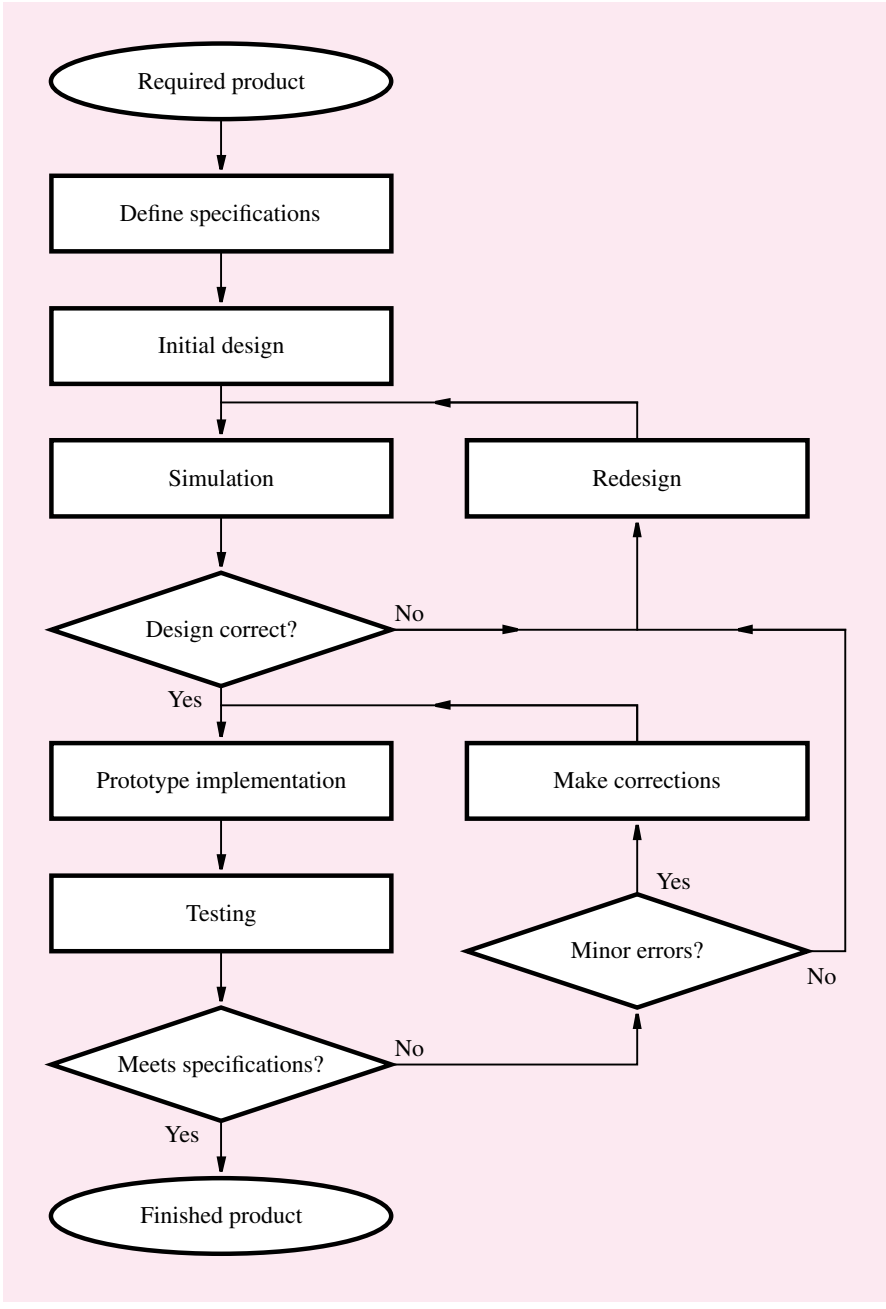

Figure 1.3 The development process.

1.3 STRUCTURE OF A COMPUTER

To understand the role that logic circuits play in digital systems, consider the structure of a typical computer, as illustrated in Figure 1.4*a*. The computer case houses a number of printed circuit boards (PCBs), a power supply, and (not shown in the figure) storage units, like a hard disk and DVD or CD-ROM drives. Each unit is plugged into a main PCB, called the *motherboard*. As indicated on the bottom of the figure, the motherboard holds several integrated circuit chips, and it provides slots for connecting other PCBs, such as audio, video, and network boards.

Figure 1.4*b* illustrates the structure of an integrated circuit chip. The chip comprises a number of subcircuits, which are interconnected to build the complete circuit. Examples of subcircuits are those that perform arithmetic operations, store data, or control the flow of data. Each of these subcircuits is a logic circuit. As shown in the middle of the figure, a logic circuit comprises a network of connected *logic gates*. Each logic gate performs a very simple function, and more complex operations are realized by connecting gates together. Logic gates are built with transistors, which in turn are implemented by fabricating various layers of material on a silicon chip.

This book is primarily concerned with the center portion of Figure 1.4*b*—the design of logic circuits. We explain how to design circuits that perform important functions, such as adding, subtracting, or multiplying numbers, counting, storing data, and controlling the processing of information. We show how the behavior of such circuits is specified, how the circuits are designed for minimum cost or maximum speed of operation, and how the circuits can be tested to ensure correct operation. We also briefly explain how transistors operate, and how they are built on silicon chips.

1.4 LOGIC CIRCUIT DESIGN IN THIS BOOK

In this book we use a modern design approach based on the Verilog hardware description language and CAD tools to illustrate many aspects of logic circuit design. We selected this technology because it is widely used in industry and because it enables the readers to implement their designs in FPGA chips, as discussed below. This technology is particularly well-suited for educational purposes because many readers have access to facilities for using CAD tools and programming FPGA devices.

To gain practical experience and a deeper understanding of logic circuits, we advise the reader to implement the examples in this book using CAD software. Most of the major vendors of CAD systems provide their software at no cost to university students for educational use. Some examples are Altera, Cadence, Mentor Graphics, Synopsys, and Xilinx. The CAD systems offered by any of these companies can be used equally well with this book. Two CAD systems that are particularly well-suited for use with this book are the Quartus II software from Altera and the ISE software from Xilinx. Both of these CAD systems support all phases of the design cycle for logic circuits and are powerful and easy to use. The reader is encouraged to visit the website for these companies, where

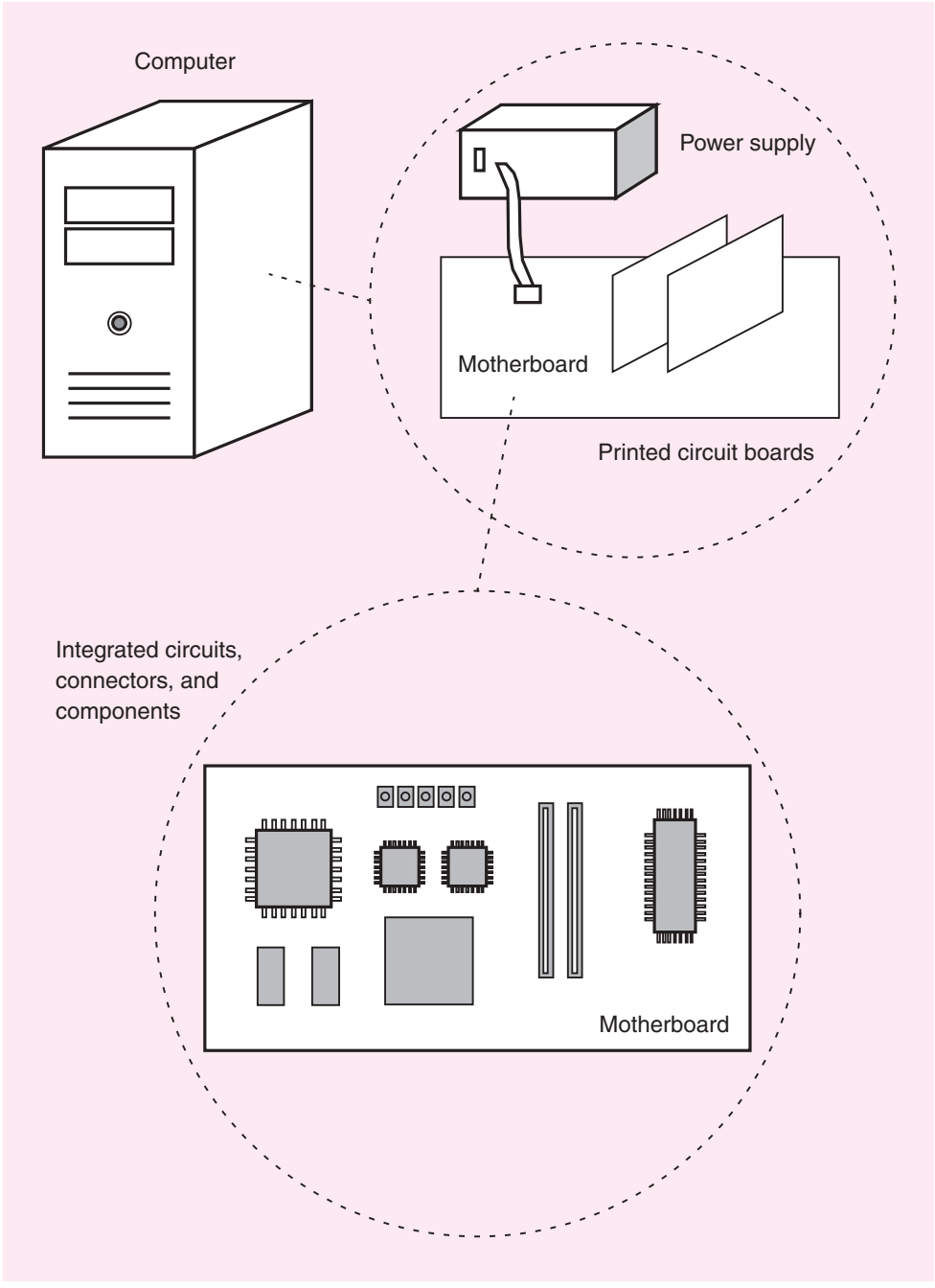

Figure 1.4 A digital hardware system (Part *a*).