Dedicated to our students, who motivate, challenge, and make us proud every day.
Contents

Preface xiii

Chapter 1 Introduction 1
  1.1 Dissecting the Title 2
    1.1.1 Design 2
    1.1.2 Logic Design 4
    1.1.3 Contemporary Logic Design 6
  1.2 A Brief History of Logic Design 8
  1.3 Computation 9
    1.3.1 Switches, Relays, and Circuits 10
    1.3.2 Transistors 12
    1.3.3 Digital Representations 13
    1.3.4 Encoding 14
  1.4 Examples 15
Chapter Review 25
Further Reading 27
Exercises 28

Chapter 2 Combinational Logic 33
  2.1 Outputs as a Function of Inputs 34
    2.1.1 Combinational Logic Defined 34
    2.1.2 Examples of Combinational Logic 34
  2.2 Laws and Theorems of Boolean Logic 37
    2.2.1 Axioms of Boolean Algebra 39
    2.2.2 Theorems of Boolean Algebra 42
    2.2.3 Duality and DeMorgan’s Law 45
  2.3 Realizing Boolean Formulas 46
    2.3.1 Logic Gates 46
    2.3.2 Logic Blocks and Hierarchy 49
    2.3.3 Time Behavior and Waveforms 50
    2.3.4 Minimizing the Number of Gates and Wires 52
    2.3.5 Case Study: 7-Segment Decoder 54
Chapter 3  Working with Combinational Logic  93

3.1 Two-Level Simplification  93
    3.1.1 Formalizing the Process of Boolean Minimization  99
    3.1.2 K-Maps Revisited: Five- and Six-Variable Functions  102

3.2 Automating Two-Level Simplification  103
    3.2.1 Quine-McCluskey Method  104
    3.2.2 Espresso Method  108
    3.2.3 Realizing S-o-P and P-o-S Logic Networks  111

3.3 Multilevel Simplification  114

3.4 Automating Multilevel Simplification  122
    3.4.1 Multilevel Logic Optimization Scripts  122
    3.4.2 Realizing Multilevel Logic Networks  126

3.5 Time Response in Combinational Networks  129
    3.5.1 Gate Delays  129
    3.5.2 Timing Waveforms  130
    3.5.3 Analysis of a Pulse-Shaper Circuit  131
    3.5.4 Hazards and Glitches  132
    3.5.5 Hazard Detection and Elimination in Two-Level Networks  133
    3.5.6 Static Hazards in Multilevel Networks  136
    3.5.7 Designing Static Hazard-Free Multilevel Circuits  137
    3.5.8 Dynamic Hazards  138
Contents

3.6 Hardware Description Languages 139
  3.6.1 Describing Structure 141
  3.6.2 Describing Behavior 141
  3.6.3 Delay 143
  3.6.4 Event-Driven Simulation 143

Chapter Review 146
Further Reading 146
Exercises 147

Chapter 4 Combinational Logic Technologies 155

4.1 History 156
  4.1.1 From Switches to Integrated Circuits 156
  4.1.2 Packaged Logic, Configurability, and Programmable Logic 158
  4.1.3 Technology Metrics 162

4.2 Basic Logic Components 164
  4.2.1 Fixed Logic 164
  4.2.2 Look-Up Tables 168
  4.2.3 Template-Based Logic 179

4.3 Two-Level and Multilevel Logic 196

4.4 Non-Gate Logic 205
  4.4.1 Tri-State Outputs 206
  4.4.2 Open-Collector Outputs and Wired Logic 210

Chapter Review 212
Further Reading 213
Exercises 213

Chapter 5 Case Studies in Combinational Logic Design 221

5.1 Design Procedure 222
5.2 A Simple Process Line-Control Problem 224
5.3 Telephone Keypad Decoder 227
5.4 Leap Year Calculation 231
5.5 Logic Function Unit 234
5.6 Adder Design 238
  5.6.1 Half Adder/Full Adder 238
  5.6.2 Carry-Lookahead Circuits 240
  5.6.3 Carry-Select Adder 244
  5.6.4 BCD Adder Design 245
Chapter 6  Sequential Logic Design  259

6.1  Sequential Logic Elements  260
       6.1.1  Simple Circuits with Feedback  260
       6.1.2  Basic Latches  265
       6.1.3  Clocks  267
       6.1.4  Combining Latches  268
       6.1.5  Master–Slave Latches and Edge-Triggered Flip-Flops  270
       6.1.6  Timing Definitions  273

6.2  Timing Methodologies  278
       6.2.1  Cascaded Flip-Flops and Setup/Hold/Propagation  279
       6.2.2  Clock Skew  281
       6.2.3  Asynchronous Inputs  282
       6.2.4  Metastability and Synchronizer Failure  284
       6.2.5  Self-timed and Speed-independent Circuits  285

6.3  Registers  289
       6.3.1  Storage Registers  289
       6.3.2  Shift Registers  291

6.4  Hardware Description Languages  295

Chapter Review  298
Further Reading  299
Exercises  300

Chapter 7  Finite State Machines  307

7.1  Counters  308
       7.1.1  Counter Design Procedure  310
       7.1.2  Counters with More Complex Sequencing  312
       7.1.3  Self-Starting Counters  314
       7.1.4  Counter Reset  316
       7.1.5  Counter Variations  316
7.2 The Concept of the State Machine  321
  7.2.1 Odd or Even Parity Checker  321
  7.2.2 Timing in State Machines  324
7.3 Basic FSM Design Approach  326
  7.3.1 Finite State Machine Design Procedure  327
  7.3.2 Moore and Mealy Machines  332
  7.3.3 State Diagram Representation  333
  7.3.4 Comparison of the Two Machine Types  334
7.4 Motivation for Optimization  339
  7.4.1 Two State Diagrams, Same I/O Behavior  339
  7.4.2 Advantages of Minimum States  340
  7.4.3 State, Input, and Output Encoding  341
  7.4.4 Factoring State Machines  342
  7.4.5 A Traffic-Light Controller  342
Chapter Review  346
Further Reading  347
Exercises  348

Chapter 8 Working with Finite State Machines  355
8.1 State Minimization/Reduction  356
  8.1.1 Row-Matching Method  356
  8.1.2 Implication Chart Method  360
  8.1.3 Equivalent States in the Presence of Don't Cares  365
  8.1.4 When State Minimization Doesn't Help  366
8.2 State Assignment  367
  8.2.1 Sequential Encoding  369
  8.2.2 Random Encoding  370
  8.2.3 One-Hot Encoding  371
  8.2.4 Output-Oriented Encoding  372
  8.2.5 Heuristic Methods  374
8.3 Finite State Machine Partitioning  380
  8.3.1 Finite State Machine Partitioning by Introducing Idle States  381
8.4 Hardware Description Languages  386
Chapter Review  391
Further Reading  392
Exercises  392
## Contents

Epilogue  509  

Appendix A  Number Systems  511

A.1 Positional Number Notation  511
   A.1.1 Decimal Numbers  511
   A.1.2 Binary, Octal, and Hexadecimal Numbers  512

A.2 Conversion Between Binary, Octal, and Hexadecimal Systems  513
   A.2.1 Conversion from Binary to Octal or Hexadecimal  513
   A.2.2 Conversion from Octal to Hexadecimal and Vice Versa  514
   A.2.3 Conversion from Base 10 to Base 2: Successive Division  514

A.3 Binary Arithmetic Operations  516
   A.3.1 Addition in Positional Notation  516
   A.3.2 Subtraction in Positional Notation  518

A.4 Representation of Negative Numbers  520
   A.4.1 Sign and Magnitude  520
   A.4.2 Ones-Complement Numbers  521
   A.4.3 Twos-Complement Numbers  522
   A.4.4 Addition and Subtraction of Numbers  524
   A.4.5 Overflow Conditions  526

A.5 BCD Number Representation  527

Appendix Review  528
Exercises  529

Appendix B  Basic Electronics  533

B.1 Basic Electricity  533
   B.1.1 Terminology  533
   B.1.2 Fundamental Quantities and Laws  534

B.2 Logic Gates from Resistors and Diodes  536
   B.2.1 Voltage Dividers  536
   B.2.2 Diode Logic  537

B.3 Bipolar-Transistor Logic  538
   B.3.1 Basic Bipolar-Transistor Logic  539
   B.3.2 Diode-Transistor Logic  539
   B.3.3 Transistor-Transistor Logic  541
   B.3.4 TTL Circuits and Noise Margin  542

B.4 MOS Transistors  543
   B.4.1 Voltage-Controlled Switches  543
   B.4.2 Logic Gates from MOS Switches  544
Appendix C Flip-Flop Types 566

C.1 Flip-Flop Components 566
C.2 Realizing Circuits with Different Kinds of Flip-Flops 568
  C.2.1 Conversion of One Flip-Flop Type to Another 568
C.3 Shift Registers and Counters 570
  C.3.1 Implementation with R-S Flip-Flops 572
  C.3.2 Implementation with J-K Flip-Flops 574
  C.3.3 Implementation with T Flip-Flops 576
  C.3.4 Implementation with D Flip-Flops 576
  C.3.5 Comparison and Summary 578
Appendix Review 579
Exercises 579

Index 581
Preface

A Second Edition

In the decade since the first edition of this book was published, the technologies of digital design have continued to evolve. The evolution has run along two closely related tracks: the underlying physical technology and the software tools that facilitate the application of the new devices. The trends identified in the first edition have continued stronger than ever and promise to continue for some time to come. Specifically, programmable logic has become virtually the norm for digital designers and the art of digital design now absolutely requires the software skills to deal with hardware description languages.

No longer do we see the familiar yellow cover of the TTL Data Book on every designer’s bookshelf. In fact, for many application areas, even small programmable logic devices (PLDs), the mainstays of the 1970s and early 1980s, are rapidly disappearing. The burgeoning market for smaller, lower power, and more portable devices has driven high levels of integration into almost every product. This also has changed the nature of optimization; the focus is now on what goes into each chip rather than on the collection of individual gates needed to realize the design. The optimizations of today are more and more often made at the architecture level rather than in the switches.

Hardware designers now spend the majority of their time dealing with software. Specifically, the tools needed to efficiently map digital designs onto the emerging programmable devices that are growing ever more sophisticated. They capture their design specifications in software with language appropriate for describing the parallelism of hardware; they use software tools to simulate their designs and then to synthesize it into the implementation technology of choice. Design time is reduced radically as market pressures require products to be introduced quickly, at the right price and performance.

Although the evergrowing complexity of designs necessitates more powerful abstractions, the fundamentals haven’t changed. In fact, the contemporary digital designer must have a broader understanding of the discipline of computation than ever before, including both hardware and software. In this second edition, we provide this broader perspective.

Changes from the First Edition

There are many changes from the first edition. These were motivated by four concerns. First, we updated the hardware technologies
discussed in the book. Second, we added a more complete, if nevertheless introductory, treatment of the software tools that are now so commonplace in the designer's tool kit. Third, we responded to the comments and suggestions received over the years by the many faculty and practitioners who have used the book. Finally, we rationalized the organization of the text so that concepts, technologies, tools, and practical matters were more clearly defined.

New Introduction

The introduction has been changed from one that focused on the process of design to one that introduces the concepts of computation, encoding, sequencing, and instruction interpretation. This sets out a better road map to the rest of the book and provides a rationale for its organization. Rather than discussing the design process in the abstract, we now include many more case studies to help the student gain that understanding by seeing the process in action.

Repartitioning of Material

Each of the two major sections on combinational and sequential logic was divided into a set of chapters. These first cover the fundamental concepts, then describe the principles of manipulating the logic into different forms, followed by a discussion of the optimizations and tools that are available, and concludes with an overview of the technologies available to build logic circuits. Each is capped by a set of comprehensive design case studies that make each of the issues concrete.

More Emphasis on Programmable Logic

We have added new material on the latest programmable logic technologies that have quickly become the dominant style for realizing digital designs. We do not attempt to provide all the information needed to work with any one technology. Those used will vary dramatically from institution to institution. Therefore, the book needs to be supplemented with a laboratory guide that covers the specifics of a particular installation. In this text, we focused on the underlying concepts. We expect laboratory guides to be available in the form of web-based materials that can be easily customized to the variety already out there and updated as new technologies emerge.

Inclusion of Hardware Description Languages

HDLs are now given a more central role to reflect their total acceptance by the design community over the past 10 years. We describe only the basics of one of the dominant languages, namely Verilog, focusing on describing behavior, as well as covering the basics of HDL simulation models. We highlight the power of the languages in making designs more parameterizable and customizable and designers more efficient.
New Design Case Studies

Nothing helps students learn design as much as designing for themselves. The next best thing is to provide a large collection of examples where the intuitions and rules of thumb are discussed explicitly. The hope is that this will help bootstrap new digital designers into the world of practical applications rather than the drill problems with which were the norm in simpler times. There are many new and extensive design examples sprinkled throughout the text and in two large chapters focusing on combinational and sequential logic.

Elimination of Chapters on Datapath, Control, and Register-Transfer

We decided to remove the last two chapters of the first edition, that focused on datapath and register-transfer design, and a simple processor as an in-depth design case study of the interaction of control and datapath. While these topics are without a doubt important, on reflection we felt they are better left for a more extended study of digital design than could be included within the page limit of this edition. Instead, we chose more intensive coverage of programmable logic and HDLs, with extensive but smaller design examples spread throughout the text. We plan to make supplementary materials on the eliminated topics available on the web.

Navigating the Book

The book is organized into 10 chapters and three appendices. Chapter 1 is an overall introduction to the field. Chapters 2 through 5 cover combinational logic. Chapters 6 through 10 cover sequential logic. The three appendices provide some potentially useful background material that may have been part of other courses in a computer or electrical engineering curriculum.

Chapter 1 is an ambitious attempt to introduce many of the concepts of digital design through a short history of the evolution of digital hardware and two simple examples. Many may find that it introduces too many concepts too quickly for students to grasp their importance. However, this was not the intent. We fully expect students to be somewhat overwhelmed by the number of new concepts that come up in the discussion of the example. The purpose of the chapter is to provide an aerial view of the field so that students find it easier to see how the pieces they will see, in much greater detail and depth in later chapters, fit together coherently. It is certainly possible to replace this chapter with a more traditional introduction.

The next four chapters lay out the concepts of combinational logic design, closing with a set of comprehensive examples.

Chapter 2 covers the basics of combinational logic from simple gates to their time behavior. It lays out the concepts of two-level and multilevel logic and motivates us as to why we would want to simplify logic. Some of the basic machinery for manipulating logic is presented with an emphasis on pencil-and-paper methods.
Chapter 3 delves into methods for working with combinational logic. It begins by describing the algorithms inside of today’s CAD tools and ends with an overview of hardware description languages and uses Verilog to demonstrate key elements. Included is a discussion of the discrete simulation concepts that help to clarify the language constructs. There is probably not enough detail to make this book the sole resource for laboratory work with CAD tools. We wanted to keep the book focused on key concepts rather than on details of particular tools. It will need to be supplemented with appropriate manuals for the particular tools students will find in their own laboratories. This chapter also covers timing issues in more detail, including hazards and hazard-elimination strategies.

Chapter 4 presents the full range of implementation technologies available to the logic designer for combinational logic. It is paired with Chapter 9 that does the same for sequential logic. Chapter 4 starts with basic logic gates (as in the traditional TTL-based courses), but quickly progresses to programmable logic (PLDs and two-level forms) and then to field-programmable gate arrays. We also discuss other types of logic constructs such as tri-state and open-collector logic. Basic electronics to support this discussion are in Appendix B.

Chapter 5 culminates the combinational logic section of the book with seven examples of increasing complexity. We emphasize problem solving from the initial specification and have provided considerable discussion of how to transform an initial informal description of the problem into precise logical statements while keeping track of the assumptions that are being made. Our goal in this chapter is to show the range of logic design and how to judge design tradeoffs and take advantage of optimization opportunities.

The remaining five chapters do the same for sequential logic what the Chapters 2 through 5 did for combinational logic.

Chapter 6 begins this section by introducing the idea of circuits with feedback and how they can be analyzed. We develop the basic elements of sequential logic, latches, and flip-flops by recapitulating their evolution. This is coupled with a discussion of the timing methodologies that make it practical to build large sequential logic systems. These methodologies are illustrated with simple sequential systems of shift registers. The chapter concludes with a continuation of the exposition of hardware description languages started in Chapter 3 and now extended to basic sequential logic elements.

Chapter 7 covers the central concept of finite state machines. It begins by using counters as a simple form of FSM and then moves on to the basic Moore and Mealy models for organizing sequential behavior. Like Chapter 2, it concludes by motivating the various optimization opportunities.

Chapter 8 extends the basic ideas of Chapter 7 and expands on the details of FSM optimization by treating state minimization, state encoding, and FSM partitioning, in turn. Each of these is illustrated with examples that highlight the tradeoffs at each stage of optimization. An
additional section at the end of the chapter provides some guidelines for structuring FSM descriptions in HDLs.

Chapter 9 concludes the discussion of implementation technologies. It recapitulates all the technologies used for combinational logic introduced in Chapter 4 but focusing on their sequential logic elements.

Chapter 10 is a large chapter with six comprehensive design examples that bring to practice all the concepts in the text. It begins with the sequential logic example from Chapter 1, now discussed in full detail, to tie back to the start of the text and ends with the serial transmission of characters from a keypad to display. The latter examples focus on the partitioning of design problems into communicating pieces along two dimensions: parallel state machines and partitioning into data-path and control.

The three appendices cover number systems, basic electronics, and flip-flop types. The first two are likely to cover concepts that students are likely to have already seen in mathematics, physics, electrical engineering, or computer science introductory courses. They are not intended to be extensive treatment of these topics but only provide the background most directly connected to the main topics of this text. The appendix on flip-flop types is provided for historical completeness.

The Complete Teaching Package

The material in this book easily fills a quarter-long course and, therefore, be comfortably covered in a semester-long course. In fact, it is likely that supplemental topics, governed by the place in the curriculum the semester-long course occupies, can and should be included. These could be: more in-depth discussion of CAD algorithms including their data structures, efficiency, and implementation; further discussion of design tradeoffs in a particular implementation technology such as FPGAs; a larger design problem that can serve as a term project to highlight issues of scale and debugging; and topics from computer organization emphasizing partitioning into data-path and control and optimized structure for both. Of course, individual instructors may also find that re-ordering some of the material makes more sense in their environments. For example, it is certainly possible to proceed by following the two sections in parallel rather than serially. Chapter 2 plus 6 and 7 can be paired, followed by 3 and 8, then 4 and 9, with the larger design examples of 5 and 10 together at the end. Many topics can also be skipped altogether. For example, CAD tools and their algorithms may be relegated to another course. Similarly, HDLs do not need to be included if the design environment focuses on schematic-level design. In the technology dimension, FPGAs can be skipped as they may be included in a later course on more advanced design methods. Our goal in organizing the book was to make it easier to make these customizations.
Finally, we are making a wealth of supplementary material available to course instructors and students. Our publisher’s web site includes:

• A set of CAD tools that supports all the concepts presented in this text;
• A comprehensive set of lecture slides;
• Samples of possible laboratory assignments and projects, and solutions to all the problems in the text; and
• Supplementary material on computer organization for those that include that material in their introductory logic design classes.

We hope you will agree with us that this second edition is a worthy successor to the first.

Randy H. Katz
Gaetano Borriello