


Brock J. LaMeres

Introduction to Logic Circuits & Logic Design with VHDL

Second Edition

 Springer

INTRODUCTION TO LOGIC CIRCUITS & LOGIC DESIGN WITH VHDL

INTRODUCTION TO LOGIC CIRCUITS & LOGIC DESIGN WITH VHDL

2ND EDITION

Brock J. LaMeres

 Springer

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

ISBN 978-3-030-12488-5 ISBN 978-3-030-12489-2 (eBook)
<https://doi.org/10.1007/978-3-030-12489-2>

Library of Congress Control Number: 2019932161

© Springer Nature Switzerland AG 2017, 2019

This work is subject to copyright. All rights are reserved 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, express 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.

Cover credit: ID 98722794 © Vladimir Timofeev | Dreamstime.com

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

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 and 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 book addresses the lower-level foundational void by providing a comprehensive, bottoms-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 VHDL language introduced. The VHDL 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 component level. Once this secondary foundation has been laid, the remaining capabilities of VHDL 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 VHDL 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 VHDL language is broken into two chapters. Chapter 5 presents an introduction to VHDL 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 VHDL simulation tools on basic combinational logic circuits. The more advanced constructs of VHDL 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 VHDL. At the end of this course, students have a solid foundation in digital circuits and are able to design and simulate VHDL models of concurrent and hierarchical systems. The second course in this sequence covers *logic design* using Chaps. 8, 9, 10, 11, 12, and 13. In this second course, students learn the advanced features of VHDL such as packages, sequential behavioral modeling, 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.

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 number 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 VHDL 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 VHDL 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 VHDL design questions, it is expected that the students will upload their VHDL 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 VHDL test benches for all problems. Additional resources are made available as demand grows. The *YouTube* videos cover every section in the book and 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 Second Edition

The most common request from adopters of the first edition of this book was more assessment problems and accompanying videos. As a result, the second edition now contains over 1000 assessment questions, and YouTube videos have been created for every section of the book. Additionally, more worked examples have been added so that every section has abundant examples of how to apply the content to designing and analyzing digital circuits.

Bozeman, MT, USA

Brock J. LaMeres

Acknowledgments

Endless thanks are given to my beautiful wife and soul mate, JoAnn. Every day is a gift with you by my side.

Contents

| | |
|--|----|
| 1: INTRODUCTION: ANALOG VS. 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 | 24 |
| 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> | 50 |
| 3.2.2 <i>Output DC Specifications</i> | 51 |
| 3.2.3 <i>Input DC Specifications</i> | 52 |
| 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 |
| 4.1.4 | <i>Functionally Complete Operation Sets</i> | 110 |
| 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> | 118 |
| 4.3.3 | <i>Canonical Product of Sums (POS)</i> | 119 |
| 4.3.4 | <i>The Maxterm List (Π)</i> | 122 |
| 4.3.5 | <i>Minterm and Maxterm List Equivalence</i> | 123 |
| 4.4 | LOGIC MINIMIZATION | 125 |
| 4.4.1 | <i>Algebraic Minimization</i> | 125 |
| 4.4.2 | <i>Minimization Using Karnaugh Maps</i> | 126 |
| 4.4.3 | <i>Don't Cares</i> | 139 |
| 4.4.4 | <i>Using XOR Gates</i> | 139 |
| 4.5 | TIMING HAZARDS AND GLITCHES | 142 |
| 5: | VHDL (PART 1) | 155 |
| 5.1 | HISTORY OF HARDWARE DESCRIPTION LANGUAGES | 156 |
| 5.2 | HDL ABSTRACTION | 159 |
| 5.3 | THE MODERN DIGITAL DESIGN FLOW | 162 |
| 5.4 | VHDL CONSTRUCTS | 165 |
| 5.4.1 | <i>Data Types</i> | 166 |
| 5.4.2 | <i>Libraries and Packages</i> | 168 |
| 5.4.3 | <i>The Entity</i> | 168 |
| 5.4.4 | <i>The Architecture</i> | 169 |
| 5.5 | MODELING CONCURRENT FUNCTIONALITY IN VHDL | 171 |
| 5.5.1 | <i>VHDL Operators</i> | 171 |
| 5.5.2 | <i>Concurrent Signal Assignments</i> | 174 |

| | | |
|-----------|---|-----|
| 5.5.3 | <i>Concurrent Signal Assignments with Logical Operators</i> | 174 |
| 5.5.4 | <i>Conditional Signal Assignments</i> | 175 |
| 5.5.5 | <i>Selected Signal Assignments</i> | 177 |
| 5.5.6 | <i>Delayed Signal Assignments</i> | 178 |
| 5.6 | STRUCTURAL DESIGN USING COMPONENTS | 181 |
| 5.6.1 | <i>Component Instantiation</i> | 181 |
| 5.7 | OVERVIEW OF SIMULATION TEST BENCHES | 183 |
| 6: | MSI LOGIC | 191 |
| 6.1 | DECODERS | 191 |
| 6.1.1 | <i>Example: One-Hot Decoder</i> | 191 |
| 6.1.2 | <i>Example: 7-Segment Display Decoder</i> | 194 |
| 6.2 | ENCODERS | 199 |
| 6.2.1 | <i>Example: One-Hot Binary Encoder</i> | 199 |
| 6.3 | MULTIPLEXERS | 201 |
| 6.4 | DEMULTIPLEXERS | 203 |
| 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> | 220 |
| 7.1.6 | <i>The D-Latch</i> | 221 |
| 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 a FSM</i> | 236 |
| 7.4.2 | <i>Logic Synthesis for a 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 | 253 |
| 7.5.1 | <i>2-Bit Binary Up Counter</i> | 253 |
| 7.5.2 | <i>2-Bit Binary Up/Down Counter</i> | 255 |
| 7.5.3 | <i>2-Bit Gray Code Up Counter</i> | 256 |
| 7.5.4 | <i>2-Bit Gray Code Up/Down Counter</i> | 258 |
| 7.5.5 | <i>3-Bit One-Hot Up Counter</i> | 260 |
| 7.5.6 | <i>3-Bit One-Hot Up/Down Counter</i> | 262 |
| 7.6 | FINITE STATE MACHINE'S RESET CONDITION | 266 |

| | |
|---|------------|
| 7.7 SEQUENTIAL LOGIC ANALYSIS | 267 |
| 7.7.1 Finding the State Equations and Output Logic Expressions of a FSM | 267 |
| 7.7.2 Finding the State Transition Table of a FSM | 268 |
| 7.7.3 Finding the State Diagram of a FSM | 269 |
| 7.7.4 Determining the Maximum Clock Frequency of a FSM | 270 |
| 8: VHDL (PART 2) | 285 |
| 8.1 THE PROCESS | 285 |
| 8.1.1 Sensitivity List | 285 |
| 8.1.2 The Wait Statement | 286 |
| 8.1.3 Sequential Signal Assignments | 287 |
| 8.1.4 Variables | 289 |
| 8.2 CONDITIONAL PROGRAMMING CONSTRUCTS | 290 |
| 8.2.1 If/Then Statements | 290 |
| 8.2.2 Case Statements | 292 |
| 8.2.3 Infinite Loops | 293 |
| 8.2.4 While Loops | 295 |
| 8.2.5 For Loops | 295 |
| 8.3 SIGNAL ATTRIBUTES | 296 |
| 8.4 TEST BENCHES | 298 |
| 8.4.1 Report Statement | 299 |
| 8.4.2 Assert Statement | 300 |
| 8.5 PACKAGES | 302 |
| 8.5.1 STD_LOGIC_1164 | 302 |
| 8.5.2 NUMERIC_STD | 306 |
| 8.5.3 NUMERIC_STD_UNSIGNED | 308 |
| 8.5.4 NUMERIC_BIT | 308 |
| 8.5.5 NUMERIC_BIT_UNSIGNED | 309 |
| 8.5.6 MATH_REAL | 309 |
| 8.5.7 MATH_COMPLEX | 311 |
| 8.5.8 TEXTIO and STD_LOGIC_TEXTIO | 311 |
| 8.5.9 Legacy Packages (STD_LOGIC_ARITH/UNSIGNED/SIGNED) | 322 |
| 9: BEHAVIORAL MODELING OF SEQUENTIAL LOGIC | 329 |
| 9.1 MODELING SEQUENTIAL STORAGE DEVICES IN VHDL | 329 |
| 9.1.1 D-Latch | 329 |
| 9.1.2 D-Flip-Flop | 330 |
| 9.1.3 D-Flip-Flop with Asynchronous Reset | 330 |
| 9.1.4 D-Flip-Flop with Asynchronous Reset and Preset | 331 |
| 9.1.5 D-Flip-Flop with Synchronous Enable | 332 |
| 9.2 MODELING FINITE STATE MACHINES IN VHDL | 334 |
| 9.2.1 Modeling the States with User-Defined, Enumerated Data Types | 335 |
| 9.2.2 The State Memory Process | 335 |
| 9.2.3 The Next State Logic Process | 336 |
| 9.2.4 The Output Logic Process | 336 |
| 9.2.5 Explicitly Defining State Codes with Subtypes | 338 |

| | | |
|------------|---|------------|
| 9.3 | FSM DESIGN EXAMPLES IN VHDL | 339 |
| 9.3.1 | <i>Serial Bit Sequence Detector in VHDL</i> | 339 |
| 9.3.2 | <i>Vending Machine Controller in VHDL</i> | 341 |
| 9.3.3 | <i>2-Bit, Binary Up/Down Counter in VHDL</i> | 343 |
| 9.4 | MODELING COUNTERS IN VHDL | 345 |
| 9.4.1 | <i>Counters in VHDL Using the Type UNSIGNED</i> | 345 |
| 9.4.2 | <i>Counters in VHDL Using the Type INTEGER</i> | 346 |
| 9.4.3 | <i>Counters in VHDL Using the Type STD_LOGIC_VECTOR</i> | 347 |
| 9.4.4 | <i>Counters with Enables in VHDL</i> | 349 |
| 9.4.5 | <i>Counters with Loads</i> | 350 |
| 9.5 | RTL MODELING | 352 |
| 9.5.1 | <i>Modeling Registers in VHDL</i> | 352 |
| 9.5.2 | <i>Shift Registers in VHDL</i> | 353 |
| 9.5.3 | <i>Registers as Agents on a Data Bus</i> | 354 |
| 10: | MEMORY | 361 |
| 10.1 | MEMORY ARCHITECTURE AND TERMINOLOGY | 361 |
| 10.1.1 | <i>Memory Map Model</i> | 361 |
| 10.1.2 | <i>Volatile Versus Non-volatile Memory</i> | 362 |
| 10.1.3 | <i>Read-Only Versus Read/Write Memory</i> | 362 |
| 10.1.4 | <i>Random Access Versus Sequential Access</i> | 362 |
| 10.2 | NON-VOLATILE MEMORY TECHNOLOGY | 363 |
| 10.2.1 | <i>ROM Architecture</i> | 363 |
| 10.2.2 | <i>Mask Read-Only Memory (MROM)</i> | 366 |
| 10.2.3 | <i>Programmable Read-Only Memory (PROM)</i> | 367 |
| 10.2.4 | <i>Erasable Programmable Read-Only Memory (EPROM)</i> | 368 |
| 10.2.5 | <i>Electrically Erasable Programmable Read-Only Memory (EEPROM)</i> | 370 |
| 10.2.6 | <i>FLASH Memory</i> | 371 |
| 10.3 | VOLATILE MEMORY TECHNOLOGY | 371 |
| 10.3.1 | <i>Static Random Access Memory (SRAM)</i> | 372 |
| 10.3.2 | <i>Dynamic Random Access Memory (DRAM)</i> | 375 |
| 10.4 | MODELING MEMORY WITH VHDL | 382 |
| 10.4.1 | <i>Read-Only Memory in VHDL</i> | 382 |
| 10.4.2 | <i>Read/Write Memory in VHDL</i> | 384 |
| 11: | PROGRAMMABLE LOGIC | 393 |
| 11.1 | PROGRAMMABLE ARRAYS | 393 |
| 11.1.1 | <i>Programmable Logic Array (PLA)</i> | 393 |
| 11.1.2 | <i>Programmable Array Logic (PAL)</i> | 394 |
| 11.1.3 | <i>Generic Array Logic (GAL)</i> | 395 |
| 11.1.4 | <i>Hard Array Logic (HAL)</i> | 396 |
| 11.1.5 | <i>Complex Programmable Logic Devices (CPLDs)</i> | 396 |
| 11.2 | FIELD-PROGRAMMABLE GATE ARRAYS (FPGAs) | 397 |
| 11.2.1 | <i>Configurable Logic Block (or Logic Element)</i> | 398 |
| 11.2.2 | <i>Look-Up Tables (LUTs)</i> | 399 |
| 11.2.3 | <i>Programmable Interconnect Points (PIPs)</i> | 402 |

| | |
|---|------------|
| 11.2.4 <i>Input/Output Blocks (IOBs)</i> | 403 |
| 11.2.5 <i>Configuration Memory</i> | 404 |
| 12: ARITHMETIC CIRCUITS | 407 |
| 12.1 ADDITION | 407 |
| 12.1.1 <i>Half Adders</i> | 407 |
| 12.1.2 <i>Full Adders</i> | 408 |
| 12.1.3 <i>Ripple Carry Adder (RCA)</i> | 410 |
| 12.1.4 <i>Carry Look Ahead Adder (CLA)</i> | 412 |
| 12.1.5 <i>Adders in VHDL</i> | 415 |
| 12.2 SUBTRACTION | 421 |
| 12.3 MULTIPLICATION | 424 |
| 12.3.1 <i>Unsigned Multiplication</i> | 424 |
| 12.3.2 <i>A Simple Circuit to Multiply by Powers of Two</i> | 427 |
| 12.3.3 <i>Signed Multiplication</i> | 428 |
| 12.4 DIVISION | 430 |
| 12.4.1 <i>Unsigned Division</i> | 430 |
| 12.4.2 <i>A Simple Circuit to Divide by Powers of Two</i> | 433 |
| 12.4.3 <i>Signed Division</i> | 434 |
| 13: COMPUTER SYSTEM DESIGN | 439 |
| 13.1 COMPUTER HARDWARE | 439 |
| 13.1.1 <i>Program Memory</i> | 440 |
| 13.1.2 <i>Data Memory</i> | 440 |
| 13.1.3 <i>Input/Output Ports</i> | 440 |
| 13.1.4 <i>Central Processing Unit</i> | 441 |
| 13.1.5 <i>A Memory Mapped System</i> | 442 |
| 13.2 COMPUTER SOFTWARE | 444 |
| 13.2.1 <i>Opcodes and Operands</i> | 445 |
| 13.2.2 <i>Addressing Modes</i> | 445 |
| 13.2.3 <i>Classes of Instructions</i> | 446 |
| 13.3 COMPUTER IMPLEMENTATION: AN 8-BIT COMPUTER EXAMPLE | 453 |
| 13.3.1 <i>Top Level Block Diagram</i> | 453 |
| 13.3.2 <i>Instruction Set Design</i> | 454 |
| 13.3.3 <i>Memory System Implementation</i> | 455 |
| 13.3.4 <i>CPU Implementation</i> | 460 |
| 13.4 ARCHITECTURE CONSIDERATIONS | 480 |
| 13.4.1 <i>Von Neumann Architecture</i> | 480 |
| 13.4.2 <i>Harvard Architecture</i> | 480 |
| APPENDIX A: LIST OF WORKED EXAMPLES | 487 |
| APPENDIX B: CONCEPT CHECK SOLUTIONS | 493 |
| INDEX | 495 |