

Digital System Designs and Practices

Using Verilog HDL
and FPGAs

Ming-Bo Lin

 WILEY

Companion Website



DIGITAL SYSTEM DESIGNS AND PRACTICES

Using Verilog HDL and FPGAs

Ming-Bo Lin

Department of Electronic Engineering

National Taiwan University of Science and Technology

Taipei, Taiwan



John Wiley & Sons (Asia) Pte Ltd

Copyright © 2008

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop, #02-01,
Singapore 129809

Visit our Home Page on www.wiley.com

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as expressly permitted by law, without either the prior written permission of the Publisher, or authorization through payment of the appropriate photocopy fee to the Copyright Clearance Center. Requests for permission should be addressed to the Publisher, John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop, #02-01, Singapore 129809, tel: 65-64632400, fax: 65-64646912, email: enquiry@wiley.com.sg

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The Publisher is not associated with any product or vendor mentioned in this book. All trademarks referred to in the text of this publication are the property of their respective owners.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Appendix A is reprinted with permission from IEEE Std 1364-2001, (Revision of IEEE Std 1364-1995), IEEE Standard Verilog[®] Hardware Description Language, Copyright 2001, by IEEE. The IEEE disclaims any responsibility or liability resulting from the placement and use in the described manner.

Other Wiley Editorial Offices

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

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

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons Canada Ltd, 6045 Freemont Blvd, Mississauga, ONT, L5R 4J3, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Cataloging-in-Publication Data

Lin, Ming-Bo.

Digital system designs and practices: using Verilog HDL and FPGAs / Ming-Bo Lin.
p. cm.

Includes bibliographical references and index.

ISBN 978-0-470-82323-1 (cloth)

1. Digital electronics. 2. Field programmable gate arrays. 3. Verilog (Computer hardware description language) I. Title.

TK7868.D5L535 2008

621.381-dc22

2008002506

ISBN 978-0-470-82323-1 (HB)

Typeset in 10.5/12pt Times by Thomson Digital, Noida, India.

Printed and bound in Singapore by Markono Print Media Pte Ltd, Singapore.

This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

DIGITAL SYSTEM DESIGNS AND PRACTICES

Using Verilog HDL and FPGAs

To Alfred, Fanny, Alice and Frank
and in memory of my parents

PREFACE

With the advance of the semiconductor and communication industries, the use of system-on-a-chip (SoC) has become an essential technique to decrease product costs. It has become increasingly important for electrical engineers to develop a good understanding of the key stages of hardware description language (HDL) design flow based on cell-based libraries or field-programmable gate array (FPGA) devices. This book addresses the need for teaching such a topic based on Verilog HDL and FPGAs.

The objective of this book, *Digital System Designs and Practices: Using Verilog HDL and FPGAs*, is intended to be useful both as a text for students and as a reference book for practicing engineers or a self-study book for readers. For class use, each chapter includes many worked problems and review questions for helping readers test their understanding of the context. In addition, throughout the book, an abundance of examples are provided for helping readers realize the basic features of Verilog HDL and grasp the essentials of digital system designs as well.

The contents of this book stem largely from the course *FPGA System Designs and Practices*, given at our campus over the past few years. This course is an 'undergraduate-elective' and first-year graduate course. This book is structured so that it can be used as a sequence of courses, such as *Hardware Description Language*, *FPGA System Designs and Practices*, *Digital System Designs*, *Advanced Digital System Designs*, and others.

CONTENTS OF THIS BOOK

The contents of this book can be roughly divided into four parts. The first part includes Chapters 1 to 7 and introduces the basic features and capabilities of Verilog HDL. This part can also be used as a reference for Verilog HDL. The second part covers Chapters 8 to 10 and contains basic combinational and sequential modules. In addition, the various options for implementing a digital system are discussed in detail. The third part consists of Chapters 11 to 13 and examines the three closely related topics: design, synthesis and verification. The last part considers the register-transfer level (RTL) and system level design examples and the techniques of testing and testable design. This part is composed of Chapters 14 to 16.

Chapter 1 introduces the features and capabilities of Verilog HDL and gives a tutorial example to illustrate how to use it to model a design at various levels of abstraction and in various modeling styles. In addition, we also demonstrate the use of Verilog HDL to verify a design after the description of a design is completed.

Chapter 2 deals with how to model a design in structural style. In this style, a module is described as a set of interconnected components, which include modules, user-defined primitives (UDPs), gate primitives and switch primitives. In this chapter,

we introduce the structural modeling at the gate and switch levels. The UDPs and modules are dealt with separately in Chapters 5 and 6.

Chapter 3 describes the essentials of dataflow modeling style. In this modeling style, the most basic statement is the continuous assignment, which consists of operators and operands in turn. The continuous assignment continuously drives a value onto a net and is usually used to model a combinational logic.

Chapter 4 is concerned with the behavioral modeling style, which provides users with the capability of modeling a design in a way like that of most high-level programming languages. In this modeling style, the most common statements include procedural assignments, selection statements and iterative (loop) statements. In addition, timing controls are also dealt with in detail.

Chapter 5 describes three additional behavioral ways provided by Verilog HDL which are widely used to model designs. These include tasks, functions and user-defined primitives (UDPs). Tasks and functions provide the ability to re-use the same piece of code from many places in a design and UDPs provide a means to model a design with a truth table. In addition, the predefined system tasks and functions are introduced. These system tasks and functions are useful when modeling, abstractly, a design in behavioral style or writing test benches for designs.

Chapter 6 discusses three closely related issues of hierarchical structural modeling. These include instantiations, `generate` statements and configurations. The instantiation is the mechanism through which the hierarchical structure is formed by modules being embedded into other modules. The `generate` statements can conditionally generate declarations and instantiations into a design. By using configurations, we may specify a new set of target libraries so as to change the mapping of a design without having to change the source description.

Chapter 7 deals with the additional features of Verilog HDL. These features include block constructs, procedural continuous assignments, `specify` blocks, timing checks and compiler directives.

Chapters 8 and 9 examine some basic combinational and sequential modules that are often used as basic building blocks to construct a complex design. In particular, these modules are the basic building blocks of a datapath when using the datapath and controller approach in a complex design.

Chapter 8 is concerned with the most commonly used combinational logic modules, which include encoders and decoders, multiplexers and demultiplexers, and magnitude comparators. In addition, a multiplexing-driven seven-segment light-emitting diode (LED) display system which combines the use of a decoder, as well as a multiplexer, is discussed in detail.

Chapter 9 examines several basic sequential modules that are widely used in digital systems. These include flip-flops, synchronizers, a switch-debouncing circuit, registers, data registers, register files, shift registers, counters (binary, BCD, Johnson), CRC generators and detectors, clock generators and pulse generators, as well as timing generators.

Chapter 10 describes various design options of digital systems. These options include application-specific integrated circuits (ASICs) and field-programmable devices. ASICs are devices that must be fabricated in IC foundries and can be designed with one of the following: full-custom, cell-based and gate-array-based approaches.

Field-programmable devices are the ones that can be personalized in laboratories and include programmable logic devices (PLDs), complex PLDs (CPLDs) and field-programmable gate arrays (FPGAs). In addition, the issues of interfacing two logic modules or devices with different logic levels and power-supply voltages are also dealt with in detail in this chapter.

The next three chapters consider three closely related issues: design, synthesis and verification. Chapter 11 introduces two useful techniques by which a system can be designed. These techniques include the finite-state machine (FSM) and register-transfer level (RTL) design approaches. The former may be described by using a state diagram or an algorithmic state machine (ASM) chart; the latter may be described by an ASM chart or by using the datapath and controller (DP+CU) paradigm. For a simple system, a three-step paradigm introduced in the chapter may be used to derive the datapath and controller of a design from its ASM chart. For complex systems, their datapaths and controllers are often derived from specifications in a state-of-the-art manner. An example of displaying four-digit data on a commercial dot-matrix liquid-crystal display (LCD) module is used to illustrate this approach. In this chapter, we also emphasize the concept that a hardware algorithm can usually be realized by using either a multiple-cycle or a single-cycle structure. The choice is based on the tradeoff among area (hardware cost), performance (operating frequency or propagation delay) and power consumption.

Chapter 12 is concerned with the principles of logic synthesis and the general architecture of synthesis tools. The function of logic synthesis is to transform an RTL representation into gate-level netlists. In order to make good use of synthesis tools, we need to provide the design environment and design constraints along with an RTL code and technology library. Moreover, we give some guidelines about how to write a good Verilog HDL code such that it can be accepted by most logic synthesis tools and can achieve the best compile times and synthesis results. These guidelines also include clock signals, reset signals and how to partition a design.

Verification is a necessary process that makes sure a design can meet its specifications both in function and timing. Chapter 13 deals with this issue in more detail and gives a comprehensive example based on FPGA design flow to illustrate how to enter, synthesize, implement and configure the underlying FPGA device of a design. Along with the design flow, static timing analyses are also given and explained. In addition, design verification through dynamic timing simulations, incorporating the delays of logic elements and interconnect, is introduced.

The next two chapters are concerned with more complex modules. Chapter 14 examines many frequently used arithmetic modules, including addition, multiplication, division, ALU, shift and two digital-signal processing (DSP) filters as well. Along with these arithmetic operations and their algorithms, we also re-emphasize the concept that a hardware algorithm can often be realized by using either a multiple-cycle or a single-cycle structure.

Chapter 15 describes the design of a small μC system, which is the most complex design example in the book. This system includes a general-purpose input and output (GPIO), timers and a universal asynchronous receiver and transmitter (UART) being connected by a system bus composed of an address bus and a data bus, as well as a control bus. The 16-bit CPU provides 27 instructions and 7 addressing modes.

The final chapter is concerned with the topic of testability and testable design. Testing is the only way to ensure that a system or a circuit may function properly. The goal of testing is to find any existing faults in a system or a circuit. In this chapter, we examine fault models, test vector generations, testable circuit design or design for testability. In addition, system-level testing, such as SRAM, a core-based system and system-on-a-chip (SoC), are also briefly dealt with.

Appendix A contains a complete syntax reference of Verilog HDL, including the keywords and formal definition of the Verilog-2001 standard in Backus-Naur Form (BNF).

SUPPLEMENTS

Two important and useful supplements are available for this book at the following URL: www.wiley.com/go/mblin. The first is student supplements, including source files of Verilog HDL examples in the book and the pdf files of lecture notes. The second is the instructor's supplements, containing figures, a solution manual and lecture notes in power-point files, in addition to the student supplements.

STUDENT PROJECTS

Many end-of-chapter problems may be assigned as student projects, in particular, the problems of Chapters 11, 14 and 15. Of course, many other chapters may also contain problems that may be used for the same purpose.

ACKNOWLEDGMENTS

Most material of this book has been taken from the course ET5009 offered at the National Taiwan University of Science and Technology over the past few years. My thanks go to the students of this course, who suffered through many of the experimental class offerings based on the draft of this book. Valuable comments from the participants of the course have helped in evolving the contents of this book and are greatly appreciated. Thanks to my mentor, Ben Chen, who is also a cofounder of the Chuan Hwa Book Company, who brought me into this colorful digital world about thirty years ago. In addition, he has also kindly allowed me to freely use figures, tables and even parts of material from my earlier Chinese Books, published by the Chuan Hwa Book Company, in this current book. Without this permission, it would have needed much more time to prepare the manuscript of this book. Finally but not least, I would like to thank my children, Alice and Frank, and my wife, Fanny, for their patience in enduring my absence from them during the writing of this book. I am also grateful to the publisher's staff for their support, encouragement and willingness to give prompt assistance during this book project.

*Ming-Bo Lin
Taipei, Taiwan*

CONTENTS

<i>PREFACE</i>	xxi
CHAPTER 1 <i>INTRODUCTION</i>	1
1.1 Introduction	1
1.1.1 Popularity of Verilog HDL	1
1.1.2 Simple Examples of Verilog HDL	2
1.1.3 HDL-Based Design	4
1.2 Introduction to Verilog	6
1.2.1 Module Concept	6
1.2.2 Lexical Conventions	7
1.2.3 Value Set	8
1.2.4 Constants	8
1.2.5 Data Types	10
1.2.6 Primitives	11
1.2.7 Attributes	12
1.3 Module Modeling Styles	13
1.3.1 Modules	13
1.3.2 Structural Modeling	16
1.3.3 Dataflow Modeling	17
1.3.4 Behavioral Modeling	19
1.3.5 Mixed-Style Modeling	20
1.4 Simulation	21
1.4.1 Basic Simulation Constructs	21
1.4.2 Related Compiler Directive and System Tasks	22
1.4.3 A Tutorial Example	24
Summary	27
References	28
Problems	28
CHAPTER 2 <i>STRUCTURAL MODELING</i>	31
2.1 Gate-Level Modeling	31
2.1.1 Gate Primitives	32

2.1.2 Tristate Buffers	39
2.1.3 Wired Logic	41
2.2 Gate Delays	44
2.2.1 Delay Models	44
2.2.2 Delay Specifications	46
2.3 Hazards	47
2.3.1 Static Hazards	48
2.3.2 Dynamic Hazards	50
2.4 Switch-Level Modeling	52
2.4.1 MOS Switches	52
2.4.2 CMOS Switch	56
2.4.3 Bidirectional Switches	58
2.4.4 Delay Specifications	59
2.4.5 Signal Strength	60
2.4.6 trireg Net	62
Summary	65
References	66
Problems	66
CHAPTER 3 DATAFLOW MODELING	69
3.1 Dataflow Modeling	69
3.1.1 Continuous Assignment	69
3.1.2 Expressions	70
3.1.3 Delays	72
3.2 Operands	74
3.2.1 Constants	74
3.2.2 Data Types	78
3.2.3 Bit-Select and Part-Select	80
3.2.4 Array and Memory Elements	82
3.3 Operators	84
3.3.1 Bit-wise Operators	84
3.3.2 Arithmetic Operators	86
3.3.3 Concatenation and Replication Operators	89
3.3.4 Reduction Operators	91
3.3.5 Logical Operators	93
3.3.6 Relational Operators	93
3.3.7 Equality Operators	94

3.3.8 Shift Operators	96
3.3.9 Conditional Operator	97
Summary	99
References	100
Problems	100
CHAPTER 4 BEHAVIORAL MODELING	103
4.1 Procedural Constructs	103
4.1.1 initial Block	104
4.1.2 always Block	106
4.2 Procedural Assignments	107
4.2.1 Procedural Assignments	107
4.2.2 Blocking Assignments	108
4.2.3 Nonblocking Assignments	110
4.2.4 Blocking versus Nonblocking Assignments	113
4.3 Timing Control	117
4.3.1 Delay Timing Control	117
4.3.2 Event Timing Control	118
4.4 Selection Statements	125
4.4.1 if-else Statement	125
4.4.2 case Statement	127
4.4.3 casex and casez Statements	132
4.5 Iterative (Loop) Statements	133
4.5.1 while Loop Statement	133
4.5.2 for Loop Statement	135
4.5.3 repeat Loop Statement	137
4.5.4 forever Loop Statement	138
Summary	139
References	140
Problems	141
CHAPTER 5 TASKS, FUNCTIONS AND UDPS	145
5.1 Tasks	145
5.1.1 Task Definition and Call	145
5.1.2 Types of Tasks	149
5.2 Functions	152
5.2.1 Function Definition and Call	152

5.2.2	Types of Functions	154
5.2.3	Constant Functions	155
5.2.4	Sharing Tasks and Functions	156
5.3	System Tasks and Functions	158
5.3.1	Simulation-Related System Tasks	159
5.3.2	File I/O System Tasks	163
5.3.3	String Formatting System Tasks	169
5.3.4	Conversion System Functions	171
5.3.5	Probability Distribution System Functions	172
5.3.6	Stochastic Analysis System Tasks	173
5.3.7	Command Line Arguments	174
5.4	User-Defined Primitives	176
5.4.1	UDP Basics	176
5.4.2	Combinational UDP	178
5.4.3	Sequential UDP	180
	Summary	185
	References	186
	Problems	186
 CHAPTER 6 <i>HIERARCHICAL STRUCTURAL MODELING</i>		189
6.1	Module	189
6.1.1	Module Definition	190
6.1.2	Parameters	192
6.1.3	Module Instantiation	194
6.1.4	Module Parameter Values	196
6.1.5	Hierarchical Names	200
6.2	generate Statement	201
6.2.1	Generate-Loop Statement	202
6.2.2	Generate-Conditional Statement	204
6.2.3	Generate-Case Statement	210
6.3	Configurations	212
6.3.1	Library	212
6.3.2	Basic Configuration Elements	213
	Summary	221
	References	222
	Problems	222

CHAPTER 7	ADVANCED MODELING TECHNIQUES	225
7.1	Sequential and Parallel Blocks	225
7.1.1	Sequential Blocks	226
7.1.2	Parallel Blocks	227
7.1.3	Special Features of Blocks	228
7.1.4	The <code>disable</code> Statement	230
7.2	Procedural Continuous Assignments	231
7.2.1	<code>assign</code> and <code>deassign</code> Statements	232
7.2.2	<code>force</code> and <code>release</code> Statements	233
7.3	Delay Models and Timing Checks	235
7.3.1	Delay Models	235
7.3.2	Specify Blocks	238
7.3.3	Timing Checks	247
7.4	Compiler Directives	259
7.4.1	<code>'define</code> and <code>'undef</code> Compiler Directives	260
7.4.2	<code>'include</code> Compiler Directive	261
7.4.3	<code>'ifdef</code> , <code>'else</code> , <code>'elsif</code> , <code>'endif</code> and <code>'ifndef</code> Compiler Directives	261
7.4.4	<code>'timescale</code> Compiler Directive	262
7.4.5	Miscellaneous Compiler Directives	263
	Summary	265
	References	266
	Problems	266
CHAPTER 8	COMBINATIONAL LOGIC MODULES	273
8.1	Decoders	273
8.1.1	Decoders	274
8.1.2	Expansion of Decoders	277
8.2	Encoders	278
8.2.1	Encoders	278
8.2.2	Priority Encoders	280
8.3	Multiplexers	283
8.3.1	Multiplexers	283
8.3.2	Expansion of Multiplexers	287
8.4	Demultiplexers	288
8.4.1	Demultiplexers	288
8.4.2	Expansion of Demultiplexers	292

8.5 Magnitude Comparators	293
8.5.1 Magnitude Comparators	294
8.5.2 Cascadable Magnitude Comparators	294
8.6 A Case Study: Seven-Segment LED Display	296
8.6.1 Seven-Segment LED Display	296
8.6.2 Multiplexing-Driven Seven-Segment LED Display	299
Summary	303
References	304
Problems	304
CHAPTER 9 SEQUENTIAL LOGIC MODULES	307
9.1 Flip-Flops	307
9.1.1 Flip-Flops	308
9.1.2 Metastable State	312
9.1.3 Synchronizers	314
9.1.4 A Switch-Debouncing Circuit	319
9.2 Memory Elements	321
9.2.1 Registers	321
9.2.2 Register Files	323
9.2.3 Synchronous RAM	324
9.2.4 Asynchronous RAM	325
9.3 Shift Registers	332
9.3.1 Shift Registers	332
9.3.2 Universal Shift Registers	334
9.4 Counters	338
9.4.1 Ripple Counters	338
9.4.2 Synchronous Counters	340
9.5 Sequence Generators	345
9.5.1 PR-Sequence Generators	345
9.5.2 CRC Generator/Detectors	349
9.5.3 Ring Counters	353
9.5.4 Johnson Counters	354
9.6 Timing Generators	356
9.6.1 Multiphase Clock Generators	356
9.6.2 Digital Monostable Circuits	358
Summary	360
References	361
Problems	362

CHAPTER 10	<i>DESIGN OPTIONS OF DIGITAL SYSTEMS</i>	367
10.1	Design Options of Digital Systems	368
10.1.1	Hierarchical System Design	368
10.1.2	Design Options of Digital Systems	370
10.1.3	ASIC Designs	373
10.1.4	Design with Field-Programmable Devices	378
10.2	PLD Modeling	382
10.2.1	ROM	383
10.2.2	PLA	385
10.2.3	PAL	387
10.2.4	PLA Modeling	391
10.3	CPLD	396
10.3.1	XC9500 Family	397
10.3.2	MAX7000 Family	401
10.4	FPGA	406
10.4.1	Xilinx FPGA Devices	406
10.4.2	Altera FPGA Devices	413
10.5	Practical Issues	418
10.5.1	I/O Standards	419
10.5.2	Voltage Tolerance	420
	Summary	422
	References	423
	Problems	424
CHAPTER 11	<i>SYSTEM DESIGN METHODOLOGY</i>	427
11.1	Finite-State Machine	427
11.1.1	Types of Sequential Circuits	428
11.1.2	FSM Modeling Styles	429
11.1.3	Implicit versus Explicit FSM	435
11.2	RTL Design	438
11.2.1	ASM Chart	438
11.2.2	ASM Modeling Styles	441
11.2.3	Datapath and Controller Design	449
11.3	RTL Implementation Options	464
11.3.1	Single-Cycle Structure	464
11.3.2	Multiple-Cycle Structure	465
11.3.3	Pipeline Structure	466
11.3.4	FSM versus Iterative Logic	469

11.4 A Case Study: Liquid-Crystal Displays	477
11.4.1 Principles of LCDs	477
11.4.2 Commercial Dot-Matrix LCD Modules	479
11.4.3 Datapath Design	484
11.4.4 Controller Design	487
Summary	495
References	496
Problems	497
CHAPTER 12 <i>SYNTHESIS</i>	501
12.1 Design Flow of ASICs and FPGA-Based Systems	501
12.1.1 The General Design Flow	502
12.1.2 Timing-Driven Placement	505
12.2 Design Environment and Constraints	508
12.2.1 Design Environment	509
12.2.2 Design Constraints	510
12.2.3 Optimization	511
12.3 Logic Synthesis	512
12.3.1 Architecture of Logic Synthesizers	512
12.3.2 Two-Level Logic Synthesis	515
12.3.3 Multilevel Logic Synthesis	517
12.3.4 Technology-Dependent Synthesis	522
12.4 Language Structure Synthesis	524
12.4.1 Synthesis of Assignment Statements	524
12.4.2 Synthesis of Selection Statements	525
12.4.3 Delay Values	527
12.4.4 Synthesis of Positive and Negative Signals	529
12.4.5 Synthesis of Loop Statements	530
12.4.6 Memory and Register Files	533
12.5 Coding Guidelines	533
12.5.1 Guidelines for Clocks	534
12.5.2 Guidelines for Resets	535
12.5.3 Partitioning for Synthesis	536
Summary	538
References	539
Problems	539