



FPGA PROTOTYPING BY VERILOG EXAMPLES

Xilinx Spartan™-3 Version

Pong P. Chu
Cleveland State University



WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

This Page Intentionally Left Blank

FPGA PROTOTYPING BY VERILOG EXAMPLES

This Page Intentionally Left Blank

FPGA PROTOTYPING BY VERILOG EXAMPLES

Xilinx Spartan™-3 Version

Pong P. Chu
Cleveland State University



WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

Copyright © 2008 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

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 permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permission>.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic format. For information about Wiley products, visit our web site at www.wiley.com.

Library of Congress Cataloging-in-Publication Data:

Chu, Pong P., 1959—

FPGA prototyping by Verilog examples / Pong P. Chu.

p. cm.

Includes index.

ISBN 978-0-470-18532-2 (cloth)

1. Field programmable gate arrays—Design and construction. 2. Prototypes, Engineering. 3. Verilog (Computer hardware description language) I. Title.

TK7895.G36C484 2008

621.39'5—dc22

2008003732

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

In memory of my father, Chia Chi Chu

This Page Intentionally Left Blank

CONTENTS

Preface	xxi
Acknowledgments	xxvii

PART I BASIC DIGITAL CIRCUITS

1 Gate-level combinational circuit	1
1.1 Introduction	1
1.2 General description	2
1.3 Basic lexical elements and data types	3
1.3.1 Lexical elements	3
1.4 Data types	4
1.4.1 Four-value system	4
1.4.2 Data type groups	4
1.4.3 Number representation	5
1.4.4 Operators	5
1.5 Program skeleton	5
1.5.1 Port declaration	6
1.5.2 Program body	7
1.5.3 Signal declaration	7
1.5.4 Another example	8
1.6 Structural description	9
1.7 Testbench	12

1.8	Bibliographic notes	14
1.9	Suggested experiments	14
1.9.1	Code for gate-level greater-than circuit	14
1.9.2	Code for gate-level binary decoder	14
2	Overview of FPGA and EDA software	15
2.1	Introduction	15
2.2	FPGA	15
2.2.1	Overview of a general FPGA device	15
2.2.2	Overview of the Xilinx Spartan-3 devices	17
2.3	Overview of the Digilent S3 board	17
2.4	Development flow	19
2.5	Overview of the Xilinx ISE project navigator	21
2.6	Short tutorial on ISE project navigator	24
2.6.1	Create the design project and HDL codes	25
2.6.2	Create a testbench and perform the RTL simulation	26
2.6.3	Add a constraint file and synthesize and implement the code	26
2.6.4	Generate and download the configuration file to an FPGA device	29
2.7	Short tutorial on the ModelSim HDL simulator	31
2.8	Bibliographic notes	35
2.9	Suggested experiments	36
2.9.1	Gate-level greater-than circuit	36
2.9.2	Gate-level binary decoder	36
3	RT-level combinational circuit	39
3.1	Introduction	39
3.2	Operators	39
3.2.1	Arithmetic operators	41
3.2.2	Shift operators	41
3.2.3	Relational and equality operators	42
3.2.4	Bitwise, reduction, and logical operators	42
3.2.5	Concatenation and replication operators	43
3.2.6	Conditional operators	44
3.2.7	Operator precedence	44
3.2.8	Expression bit-length adjustment	45
3.2.9	Synthesis of z and x values	46
3.3	Always block for a combinational circuit	48
3.3.1	Basic syntax and behavior	48
3.3.2	Procedural assignment	49
3.3.3	Variable data types	49
3.3.4	Simple examples	49

3.4	If statement	51
3.4.1	Syntax	51
3.4.2	Examples	52
3.5	Case statement	54
3.5.1	Syntax	54
3.5.2	Examples	54
3.5.3	The casez and casex statements	56
3.5.4	The full case and parallel case	56
3.6	Routing structure of conditional control constructs	57
3.6.1	Priority routing network	57
3.6.2	Multiplexing network	59
3.7	General coding guidelines for an always block	60
3.7.1	Common errors in combinational circuit codes	60
3.7.2	Guidelines	63
3.8	Parameter and constant	64
3.8.1	Constant	64
3.8.2	Parameter	65
3.8.3	Use of parameters in Verilog-1995	67
3.9	Design examples	67
3.9.1	Hexadecimal digit to seven-segment LED decoder	67
3.9.2	Sign-magnitude adder	71
3.9.3	Barrel shifter	73
3.9.4	Simplified floating-point adder	75
3.10	Bibliographic notes	80
3.11	Suggested experiments	80
3.11.1	Multifunction barrel shifter	80
3.11.2	Dual-priority encoder	80
3.11.3	BCD incrementor	81
3.11.4	Floating-point greater-than circuit	81
3.11.5	Floating-point and signed integer conversion circuit	81
3.11.6	Enhanced floating-point adder	81
4	Regular Sequential Circuit	83
4.1	Introduction	83
4.1.1	D FF and register	83
4.1.2	Synchronous system	84
4.1.3	Code development	85
4.2	HDL code of the FF and register	86
4.2.1	D FF	86
4.2.2	Register	89
4.2.3	Register file	90
4.2.4	Storage components in a Spartan-3 device ^{Xilinx specific}	91

4.3	Simple design examples	91
4.3.1	Shift register	91
4.3.2	Binary counter and variant	93
4.4	Testbench for sequential circuits	96
4.5	Case study	99
4.5.1	LED time-multiplexing circuit	99
4.5.2	Stopwatch	107
4.5.3	FIFO buffer	110
4.6	Bibliographic notes	115
4.7	Suggested experiments	115
4.7.1	Programmable square-wave generator	115
4.7.2	PWM and LED dimmer	115
4.7.3	Rotating square circuit	116
4.7.4	Heartbeat circuit	116
4.7.5	Rotating LED banner circuit	116
4.7.6	Enhanced stopwatch	116
4.7.7	Stack	117
5	FSM	119
5.1	Introduction	119
5.1.1	Mealy and Moore outputs	119
5.1.2	FSM representation	120
5.2	FSM code development	122
5.3	Design examples	125
5.3.1	Rising-edge detector	125
5.3.2	Debouncing circuit	130
5.3.3	Testing circuit	133
5.4	Bibliographic notes	135
5.5	Suggested experiments	135
5.5.1	Dual-edge detector	135
5.5.2	Alternative debouncing circuit	135
5.5.3	Parking lot occupancy counter	136
6	FSMD	139
6.1	Introduction	139
6.1.1	Single RT operation	139
6.1.2	ASMD chart	140
6.1.3	Decision box with a register	141
6.2	Code development of an FSMD	143
6.2.1	Debouncing circuit based on RT methodology	144
6.2.2	Code with explicit data path components	146

6.2.3	Code with implicit data path components	148
6.2.4	Comparison	150
6.2.5	Testing circuit	152
6.3	Design examples	153
6.3.1	Fibonacci number circuit	153
6.3.2	Division circuit	157
6.3.3	Binary-to-BCD conversion circuit	160
6.3.4	Period counter	164
6.3.5	Accurate low-frequency counter	167
6.4	Bibliographic notes	170
6.5	Suggested experiments	170
6.5.1	Alternative debouncing circuit	170
6.5.2	BCD-to-binary conversion circuit	171
6.5.3	Fibonacci circuit with BCD I/O: design approach 1	171
6.5.4	Fibonacci circuit with BCD I/O: design approach 2	171
6.5.5	Auto-scaled low-frequency counter	172
6.5.6	Reaction timer	172
6.5.7	Babbage difference engine emulation circuit	173
7	Selected Topics of Verilog	175
7.1	Blocking versus nonblocking assignment	175
7.1.1	Overview	175
7.1.2	Combinational circuit	177
7.1.3	Memory element	179
7.1.4	Sequential circuit with mixed blocking and nonblocking assignments	180
7.2	Alternative coding style for sequential circuit	182
7.2.1	Binary counter	182
7.2.2	FSM	185
7.2.3	FSMD	186
7.2.4	Summary	188
7.3	Use of the signed data type	188
7.3.1	Overview	188
7.3.2	Signed number in Verilog-1995	189
7.3.3	Signed number in Verilog-2001	190
7.4	Use of function in synthesis	191
7.4.1	Overview	191
7.4.2	Examples	192
7.5	Additional constructs for testbench development	193
7.5.1	Always block and initial block	194
7.5.2	Procedural statements	194
7.5.3	Timing control	196

7.5.4	Delay control	196
7.5.5	Event control	197
7.5.6	Wait statement	197
7.5.7	Timescale directive	197
7.5.8	System functions and tasks	198
7.5.9	User-defined functions and tasks	202
7.5.10	Example of a comprehensive testbench	204
7.6	Bibliographic notes	210
7.7	Suggested experiments	210
7.7.1	Shift register with blocking and nonblocking assignments	210
7.7.2	Alternative coding style for BCD counter	211
7.7.3	Alternative coding style for FIFO buffer	211
7.7.4	Alternative coding style for Fibonacci circuit	211
7.7.5	Dual-mode comparator	211
7.7.6	Enhanced binary counter monitor	212
7.7.7	Testbench for FIFO buffer	212
PART II I/O MODULES		
8	UART	215
8.1	Introduction	215
8.2	UART receiving subsystem	216
8.2.1	Oversampling procedure	216
8.2.2	Baud rate generator	217
8.2.3	UART receiver	217
8.2.4	Interface circuit	220
8.3	UART transmitting subsystem	223
8.4	Overall UART system	226
8.4.1	Complete UART core	226
8.4.2	UART verification configuration	228
8.5	Customizing a UART	230
8.6	Bibliographic notes	232
8.7	Suggested experiments	232
8.7.1	Full-featured UART	232
8.7.2	UART with an automatic baud rate detection circuit	233
8.7.3	UART with an automatic baud rate and parity detection circuit	233
8.7.4	UART-controlled stopwatch	233
8.7.5	UART-controlled rotating LED banner	234
9	PS2 Keyboard	235
9.1	Introduction	235
9.2	PS2 receiving subsystem	236

9.2.1	Physical interface of a PS2 port	236
9.2.2	Device-to-host communication protocol	236
9.2.3	Design and code	236
9.3	PS2 keyboard scan code	240
9.3.1	Overview of the scan code	240
9.3.2	Scan code monitor circuit	241
9.4	PS2 keyboard interface circuit	244
9.4.1	Basic design and HDL code	244
9.4.2	Verification circuit	246
9.5	Bibliographic notes	248
9.6	Suggested experiments	248
9.6.1	Alternative keyboard interface I	248
9.6.2	Alternative keyboard interface II	249
9.6.3	PS2 receiving subsystem with watchdog timer	249
9.6.4	Keyboard-controlled stopwatch	249
9.6.5	Keyboard-controlled rotating LED banner	249
10	PS2 Mouse	251
10.1	Introduction	251
10.2	PS2 mouse protocol	252
10.2.1	Basic operation	252
10.2.2	Basic initialization procedure	252
10.3	PS2 transmitting subsystem	253
10.3.1	Host-to-PS2-device communication protocol	253
10.3.2	Design and code	254
10.4	Bidirectional PS2 interface	259
10.4.1	Basic design and code	259
10.4.2	Verification circuit	260
10.5	PS2 mouse interface	263
10.5.1	Basic design	263
10.5.2	Testing circuit	265
10.6	Bibliographic notes	266
10.7	Suggested experiments	266
10.7.1	Keyboard control circuit	267
10.7.2	Enhanced mouse interface	267
10.7.3	Mouse-controlled seven-segment LED display	267
11	External SRAM	269
11.1	Introduction	269
11.2	Specification of the IS61LV25616AL SRAM	270
11.2.1	Block diagram and I/O signals	270

11.2.2	Timing parameters	270
11.3	Basic memory controller	274
11.3.1	Block diagram	274
11.3.2	Timing requirement	275
11.3.3	Register file versus SRAM	276
11.4	A safe design	276
11.4.1	ASMD chart	276
11.4.2	Timing analysis	277
11.4.3	HDL implementation	278
11.4.4	Basic testing circuit	281
11.4.5	Comprehensive SRAM testing circuit	283
11.5	More aggressive design	288
11.5.1	Timing issues	288
11.5.2	Alternative design I	288
11.5.3	Alternative design II	290
11.5.4	Alternative design III	291
11.5.5	Advanced FPGA features ^{Xilinx specific}	293
11.6	Bibliographic notes	294
11.7	Suggested experiments	294
11.7.1	Memory with a 512K-by-16 configuration	294
11.7.2	Memory with a 1M-by-8 configuration	295
11.7.3	Memory with an 8M-by-1 configuration	295
11.7.4	Expanded memory testing circuit	295
11.7.5	Memory controller and testing circuit for alternative design I	295
11.7.6	Memory controller and testing circuit for alternative design II	295
11.7.7	Memory controller and testing circuit for alternative design III	295
11.7.8	Memory controller with DCM	295
11.7.9	High-performance memory controller	296
12	Xilinx Spartan-3 Specific Memory	297
12.1	Introduction	297
12.2	Embedded memory of Spartan-3 device	297
12.2.1	Overview	297
12.2.2	Comparison	298
12.3	Method to incorporate memory modules	298
12.3.1	Memory module via HDL component instantiation	299
12.3.2	Memory module via Core Generator	299
12.3.3	Memory module via HDL inference	300
12.4	HDL templates for memory inference	300
12.4.1	Single-port RAM	300
12.4.2	Dual-port RAM	303
12.4.3	ROM	305

12.5	Bibliographic notes	307
12.6	Suggested experiments	307
12.6.1	Block-RAM-based FIFO	307
12.6.2	Block-RAM-based stack	307
12.6.3	ROM-based sign-magnitude adder	307
12.6.4	ROM-based $\sin(x)$ function	308
12.6.5	ROM-based $\sin(x)$ and $\cos(x)$ functions	308
13	VGA controller I: graphic	309
13.1	Introduction	309
13.1.1	Basic operation of a CRT	309
13.1.2	VGA port of the S3 board	311
13.1.3	Video controller	311
13.2	VGA synchronization	312
13.2.1	Horizontal synchronization	312
13.2.2	Vertical synchronization	314
13.2.3	Timing calculation of VGA synchronization signals	315
13.2.4	HDL implementation	315
13.2.5	Testing circuit	318
13.3	Overview of the pixel generation circuit	319
13.4	Graphic generation with an object-mapped scheme	319
13.4.1	Rectangular objects	320
13.4.2	Non-rectangular object	325
13.4.3	Animated object	326
13.5	Graphic generation with a bit-mapped scheme	332
13.5.1	Dual-port RAM implementation	332
13.5.2	Single-port RAM implementation	337
13.6	Bibliographic notes	337
13.7	Suggested experiments	337
13.7.1	VGA test pattern generator	337
13.7.2	SVGA mode synchronization circuit	338
13.7.3	Visible screen adjustment circuit	338
13.7.4	Ball-in-a-box circuit	338
13.7.5	Two-balls-in-a-box circuit	339
13.7.6	Two-player pong game	339
13.7.7	Breakout game	339
13.7.8	Full-screen dot trace	339
13.7.9	Mouse pointer circuit	340
13.7.10	Small-screen mouse scribble circuit	340
13.7.11	Full-screen mouse scribble circuit	340
14	VGA controller II: text	341

14.1	Introduction	341
14.2	Text generation	341
14.2.1	Character as a tile	341
14.2.2	Font ROM	342
14.2.3	Basic text generation circuit	344
14.2.4	Font display circuit	345
14.2.5	Font scaling	347
14.3	Full-screen text display	348
14.4	The complete pong game	352
14.4.1	Text subsystem	352
14.4.2	Modified graphic subsystem	358
14.4.3	Auxiliary counters	359
14.4.4	Top-level system	361
14.5	Bibliographic notes	366
14.6	Suggested experiments	366
14.6.1	Rotating banner	366
14.6.2	Underline for the cursor	366
14.6.3	Dual-mode text display	366
14.6.4	Keyboard text entry	366
14.6.5	UART terminal	366
14.6.6	Square-wave display	367
14.6.7	Simple four-trace logic analyzer	367
14.6.8	Complete two-player pong game	368
14.6.9	Complete breakout game	368

PART III PICOBLAZE MICROCONTROLLER^{XILINX SPECIFIC}

15	PicoBlaze Overview	371
15.1	Introduction	371
15.2	Customized hardware and customized software	372
15.2.1	From special-purpose FSMD to general-purpose microcontroller	372
15.2.2	Application of microcontroller	374
15.3	Overview of PicoBlaze	374
15.3.1	Basic organization	374
15.3.2	Top-level HDL modules	376
15.4	Development flow	377
15.5	Instruction set	377
15.5.1	Programming model	379
15.5.2	Instruction format	379
15.5.3	Logical instructions	380
15.5.4	Arithmetic instructions	381
15.5.5	Compare and test instructions	382

15.5.6 Shift and rotate instructions	383
15.5.7 Data movement instructions	384
15.5.8 Program flow control instructions	386
15.5.9 Interrupt related instructions	389
15.6 Assembler directives	390
15.6.1 The KCPSM3 directives	390
15.6.2 The PBlazeIDE directives	390
15.7 Bibliographic notes	391
16 PicoBlaze Assembly Code Development	393
16.1 Introduction	393
16.2 Useful code segments	393
16.2.1 KCPSM3 conventions	393
16.2.2 Bit manipulation	394
16.2.3 Multiple-byte manipulation	395
16.2.4 Control structure	396
16.3 Subroutine development	398
16.4 Program development	399
16.4.1 Demonstration example	400
16.4.2 Program documentation	404
16.5 Processing of the assembly code	406
16.5.1 Compiling with KCSPM3	406
16.5.2 Simulation by PBlazeIDE	407
16.5.3 Reloading code via the JTAG port	410
16.5.4 Compiling by PBlazeIDE	410
16.6 Syntheses with PicoBlaze	411
16.7 Bibliographic notes	412
16.8 Suggested experiments	412
16.8.1 Signed multiplication	412
16.8.2 Multi-byte multiplication	412
16.8.3 Barrel shift function	413
16.8.4 Reverse function	413
16.8.5 Binary-to-BCD conversion	413
16.8.6 BCD-to-binary conversion	413
16.8.7 Heartbeat circuit	413
16.8.8 Rotating LED circuit	413
16.8.9 Discrete LED dimmer	413
17 PicoBlaze I/O Interface	415
17.1 Introduction	415
17.2 Output port	416

17.2.1	Output instruction and timing	416
17.2.2	Output interface	417
17.3	Input port	418
17.3.1	Input instruction and timing	418
17.3.2	Input interface	419
17.4	Square program with a switch and seven-segment LED display interface	421
17.4.1	Output interface	421
17.4.2	Input interface	422
17.4.3	Assembly code development	424
17.4.4	HDL code development	431
17.5	Square program with a combinational multiplier and UART console	434
17.5.1	Multiplier interface	434
17.5.2	UART interface	435
17.5.3	Assembly code development	436
17.5.4	HDL code development	446
17.6	Bibliographic notes	449
17.7	Suggested experiments	449
17.7.1	Low-frequency counter I	449
17.7.2	Low-frequency counter II	449
17.7.3	Auto-scaled low-frequency counter	449
17.7.4	Basic reaction timer with a software timer	449
17.7.5	Basic reaction timer with a hardware timer	450
17.7.6	Enhanced reaction timer	450
17.7.7	Small-screen mouse scribble circuit	450
17.7.8	Full-screen mouse scribble circuit	450
17.7.9	Enhanced rotating banner	450
17.7.10	Pong game	450
17.7.11	Text editor	451
18	PicoBlaze Interrupt Interface	453
18.1	Introduction	453
18.2	Interrupt handling in PicoBlaze	453
18.2.1	Software processing	454
18.2.2	Timing	455
18.3	External interface	456
18.3.1	Single interrupt request	456
18.3.2	Multiple interrupt requests	456
18.4	Software development considerations	457
18.4.1	Interrupt as an alternative scheduling scheme	457
18.4.2	Development of an interrupt service routine	458
18.5	Design example	458
18.5.1	Interrupt interface	458

18.5.2	Interrupt service routine development	459
18.5.3	Assembly code development	459
18.5.4	HDL code development	461
18.6	Bibliographic notes	464
18.7	Suggested experiments	464
18.7.1	Alternative timer interrupt service routine	464
18.7.2	Programmable timer	464
18.7.3	Set-button interrupt service routine	465
18.7.4	Interrupt interface with two requests	465
18.7.5	Four-request interrupt controller	465
Appendix A: Sample Verilog templates		467
A.1	Numbers and operators	467
A.1.1	Sized and unsized numbers	467
A.1.2	Operators	468
A.2	General Verilog constructs	469
A.2.1	Overall code structure	469
A.2.2	Component instantiation	470
A.3	Routing with conditional operator and if and case statements	470
A.3.1	Conditional operator and if statement	470
A.3.2	Case statement	471
A.4	Combinational circuit using an always block	472
A.4.1	Always block without default output assignment	472
A.4.2	Always block with default output assignment	472
A.5	Memory Components	473
A.5.1	Register template	473
A.5.2	Register file	474
A.6	Regular sequential circuits	474
A.7	FSM	476
A.8	FSMD	478
A.9	S3 board constraint file (s3.ucf)	480
References		485
Topic Index		487

This Page Intentionally Left Blank

PREFACE

HDL (hardware description language) and *FPGA* (field-programmable gate array) devices allow designers to quickly develop and simulate a sophisticated digital circuit, realize it on a prototyping device, and verify operation of the physical implementation. As these technologies mature, they have become mainstream practice. We can now use a PC and an inexpensive FPGA prototyping board to construct a complex and sophisticated digital system. This book uses a “learning by doing” approach and illustrates the FPGA and HDL development and design process by a series of examples. A wide range of examples is included, from a simple gate-level circuit to an embedded system with an 8-bit soft-core microcontroller and customized I/O peripherals. All examples can be synthesized and physically tested on a prototyping board.

Focus and audience

Focus The main focus of this book is on the effective derivation of hardware, not the syntax of HDL. Instead of explaining every language construct, the book focuses on a small synthesizable subset and uses about a dozen code templates to provide the skeletons of various types of circuits. These templates are general and can easily be integrated to construct a large, complex system. Although this approach limits the “freedom” of syntactic expression, it will not prevent us from developing innovative hardware architecture. Because of the generality and flexibility of HDL, the same circuit can usually be described by a wide variety of language constructs and coding styles. Many of these codes are intended for modeling. They may lead to unnecessarily complex hardware implementation and sometimes cannot be synthesized at all. The template approach actually forces us to think more about hardware and develop a good coding practice for synthesis. Since we are

more interested in hardware, it is more beneficial to spend time on developing 10 different hardware architectures with the same code template rather than describing the same circuit with 10 different versions of codes.

There are two popular HDLs, *VHDL* and *Verilog*. Both languages are used widely and are IEEE standards. This book uses Verilog, and a separate book with a similar title uses VHDL. Despite the drastic syntactic differences in the two languages, their capabilities are very similar, particularly for our purposes. After we comprehend the design practice and coding methodology in one language, learning the other language is rather straightforward.

Although the book is intended for beginning designers, the examples follow strict design guidelines and prepare readers for future endeavors. The coding and design practice is “forward compatible,” which means that:

- The same practice can be applied to large design in the future.
- The same practice can aid other system development tasks, including simulation, timing analysis, verification, and testing.
- The same practice can be applied to ASIC technology and different types of FPGA devices.
- The code can be accepted by synthesis software from different vendors.

In summary, the book is a hands-on, hardware-centric text that involves *minimal HDL overhead* and follows good design and coding practice to achieve *maximal forward compatibility*.

Audience and prerequisites The book contains three major parts: basic digital circuits, peripheral modules, and embedded microcontroller. The intended audience is students in an introductory or advanced digital system design course as well as practicing engineers who wish to learn FPGA- and HDL-based development. For the materials in the first two parts, readers need to have a basic knowledge of digital systems, usually a required course in electrical engineering and computer engineering curricula. For the materials in the third part, prior exposure to assembly language programming will be helpful.

Logistics

Although a major goal of this book is to teach readers to develop software-independent and device-neutral HDL codes, we have to choose a software package and a prototyping board to synthesize and implement the design examples. The synthesis software and FPGA devices from Xilinx, a leading manufacture in this area, are used in the book.

Software The synthesis software used in the book is the Web version of the Xilinx *ISE* package. The functionality of this version is similar to that of the full version but supports only a limited number of devices. Most introductory development boards use FPGA devices from the inexpensive Spartan-3 family. Since the Web version supports the Spartan-3 device, it fits our needs. The simulation software used in the book is the starter version of Mentor Graphics’ *ModelSim XE III* package. It is a customized edition of *ModelSim*. Both software packages are free and can be downloaded from Xilinx’s Web site.

FPGA prototyping board This book is prepared to be used with several entry-level FPGA prototyping boards manufactured by Digilent Inc., including the *Spartan-3 Starter*, *Nexys-2*, and *Basys* boards, all of which contain a Spartan-3/3E FPGA device and have

similar I/O peripherals. The design examples in the book are based on the Spartan-3 Starter board (or simply the *S3 board*), but most of them can be used directly on other boards as well. The applicability of the HDL codes is summarized below.

- **Spartan-3 Starter (S3) board.** The S3 board contains all the peripherals and no additional accessory module is needed. All HDL codes and discussions can be applied to this board directly.
- **Nexys-2 board.** The Nexys-2 board is a newer board, which contains a larger FPGA device and a larger memory chip. Its peripherals are similar to those on the S3 board. There are two differences. First, the “color depth” of its VGA interface is expanded from 3 bits to 8 bits. Thus, the output of the VGA interface circuits discussed in Chapters 13 and 14 needs to be modified accordingly. Second, the Nexys-2 board contains a more sophisticated external memory device. Although the device can be configured as an asynchronous SRAM, the timing characteristics are different from those of the S3 board’s memory device, and thus the HDL codes for the memory controller in Chapter 11 cannot be used directly. However, the same design principle can be applied to construct a new controller.
- **Basys board.** The Basys board is a simpler board. It lacks the RS-232 connector. To implement the UART module and the serial interface discussed in Chapter 8, we need Digilent’s *RS-232 converter peripheral module*. The Basys board has no external memory devices, and thus the discussion of the memory controller in Chapter 11 is not applicable.
- **Other FPGA boards.** Most peripherals discussed in this book are de facto industrial standards, and the corresponding HDL codes can be used as long as a board provides proper analog interface circuits and connectors. Except for the Xilinx-specific portions, the codes can be applied to the boards based on the FPGA devices from other manufacturers as well.

PC Accessories The design examples include interfaces to several PC peripheral devices. A keyboard, a mouse, and a VGA monitor are required for the respective modules, and a “straight-through” serial cable (the most commonly used type) is required for the UART module. These accessories are widely available and can probably be obtained from an old PC.

Book organization

The book is divided into three major parts. Part I introduces the elementary HDL constructs and their hardware counterparts, and demonstrates the construction of a basic digital circuit with these constructs. It consists of six chapters:

- Chapter 1 describes the skeleton of an HDL program, basic language syntax, and logical operators. Gate-level combinational circuits are derived with these language constructs.
- Chapter 2 provides an overview of an FPGA device, prototyping board, and development flow. The development process is demonstrated by a tutorial on Xilinx ISE synthesis software and a tutorial on Mentor Graphics ModelSim simulation software.
- Chapter 3 introduces HDL’s relational and arithmetic operators and routing constructs. These correspond to medium-sized components, such as comparators, adders, and multiplexers. Module-level combinational circuits are derived with these language constructs.

- Chapter 4 covers the codes for memory elements and the construction of “regular” sequential circuits, such as counters and shift registers, in which the state transitions exhibit a regular pattern.
- Chapter 5 discusses the construction of a finite state machine (FSM), which is a sequential circuit whose state transitions do not exhibit a simple, regular pattern.
- Chapter 6 presents the construction of an FSM with data path (FSMD). The FSMD is used to implement register transfer (RT) methodology, in which the system operation is described by data transfers and manipulations among registers.
- Chapter 7 discusses several more advanced topics on language constructs and coding techniques and introduces the development of more sophisticated testbenches. This chapter can be skipped without affecting the remaining chapters.

Part II applies the techniques from Part I to design an array of peripheral modules for the prototyping board. Each chapter covers the development, implementation, and verification of an individual peripheral. These modules can be incorporated to a larger project. Part II consists of seven chapters:

- Chapter 8 discusses the design of a universal asynchronous receiver and transmitter (UART), which provides a serial link to receive and transmit data via the prototyping board’s RS-232 port.
- Chapter 9 covers the design of a keyboard interface, which reads scan code from a keyboard. The keyboard is connected via the prototyping board’s PS2 port.
- Chapter 10 covers the design of a mouse interface, which obtains the button and movement information from a mouse. The mouse is also connected via the prototyping board’s PS2 port.
- Chapter 11 discusses the implementation and timing issues of a memory controller. The controller is used to read data from and write data to the two static random access memory (SRAM) devices on the S3 board.
- Chapter 12 discusses the inference and application of Spartan-3 device-specific components. The focus is on the FPGA’s internal memory blocks.
- Chapter 13 presents the design and implementation of a video controller. The discussion covers the generation of video synchronization signals and shows the construction of simple bit- and object-mapped graphical interfaces. The monitor is connected to the prototyping board’s VGA port.
- Chapter 14 continues development of the video controller. The discussion illustrates the construction of text interface and general tile-mapped scheme.

Part III introduces an FPGA-based soft-core microcontroller, known as *PicoBlaze*, and demonstrates the integration of a general-purpose processor and customized circuit. It includes four chapters:

- Chapter 15 provides an overview of the organization and instruction set of PicoBlaze.
- Chapter 16 introduces the basic assembly programming and provides an overview of the development process.
- Chapter 17 discusses PicoBlaze’s I/O feature and illustrates the procedure to derive customized circuits to interface other I/O peripherals.
- Chapter 18 discusses PicoBlaze’s interrupt capability and demonstrates the construction of a customized interrupt-handling circuit.

In addition to regular chapters, the appendix summarizes and lists all code templates.

Special marks^{Xilinx specific} We use two special paragraph marks in the book: one for a *Xilinx-specific feature* and one for *Verilog-1995 constructs*. While the examples

described in the book are implemented on a Xilinx-based prototyping board and the codes are synthesized by Xilinx ISE software, we try to make the HDL codes as device independent and software neutral as possible. Most discussions and codes can be applied to different target devices and different synthesis software as well. However, certain codes or device features are unique to Xilinx ISE software or Spartan-3 FPGA devices. We use the *Xilinx* *specific* superscript, as in the heading of this section, to indicate that the discussion in the corresponding section or chapter is unique to Xilinx.

Similarly, we use marginal notes, as shown on the outer edge, to indicate that the discussion in a paragraph is unique to Xilinx. This note indicates that the code or design is no longer portable and needs to be revised when a different software package or target device is used. **Xilinx specific**

The Verilog language was first ratified in 1995 (referred to as Verilog-1995) and then revised in 2001 (referred to as Verilog-2001). Many useful enhancements are added in the revised version. We use Verilog-2001 in this book. If a language construct differs in the two versions, we describe the old syntax briefly in a separate paragraph and use a marginal note, as shown on the outer edge, for this type of discussion. It indicates “for your information” **FYI** and the materials are included to help readers understand the older Verilog codes.

Instructional use

The book can be a good companion text for an introductory digital systems course or an advanced project-oriented course. In an introductory digital systems course, the book supplies the lab portion of the curriculum. The chapters in Part I basically follow the sequence of a typical curriculum and can be presented along with regular lectures. One or two peripheral modules can be selected as case studies, and corresponding experiments can be used as term projects.

In an advanced project-oriented course, the book provides a base for independent projects. The materials in Part I should be treated as an overview or refresher, which provides a general background on HDL, synthesis, and FPGA boards. Some modules in Part II can be used to demonstrate the design of more complex circuits. These modules can also be considered as building blocks (i.e., IPs) or subsystems to be integrated into final projects. The PicoBlaze microcontroller discussed in Part III can be used as a general-purpose processor if an embedded-system type of project is desired.

Companion Web site

An accompanying Web site (http://academic.csuohio.edu/chu_p/rtl) provides additional information, including the following materials:

- Errata
- Code templates
- HDL code listing and relevant files
- Links to synthesis and simulation software
- Links to referenced materials
- Additional project ideas

Errata The book is self-prepared, which means that the author has produced all aspects of the text, including illustrations, tables, code listings, indexing, and formatting. As errors

are always bound to happen, the accompanying Web site provides an updated errata sheet and a place to report errors.

P. P. CHU

Cleveland, Ohio

January 2008

ACKNOWLEDGMENTS

The author would like to express his gratitude to Professor George L. Kramerich for his encouragement and help.

The author also thanks John Wiley & Sons, Inc. for giving permission to use Figures 3.1, 3.2, 4.2, 4.10, 4.11, 6.5, and 7.2 from my text *RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability*, and Xilinx, Inc. for giving permission to use Figures 2.3 and 9.3 from the *Spartan-3 Starter Kit Board User Guide*.

All trademarks used or referred to in this book are the property of their respective owners.

P. P. Chu

This Page Intentionally Left Blank