

*RAPID PROTOTYPING
OF DIGITAL SYSTEMS
SOPC EDITION*

RAPID PROTOTYPING OF DIGITAL SYSTEMS SOPC EDITION

JAMES O. HAMBLÉN

**SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
GEORGIA INSTITUTE OF TECHNOLOGY**

TYSON S. HALL

**SCHOOL OF COMPUTING
SOUTHERN ADVENTIST UNIVERSITY**

MICHAEL D. FURMAN

**DEPARTMENT OF ENGINEERING
CAMBRIDGE UNIVERSITY**

James O. Hamblen
Georgia Institute of Technology
Atlanta, GA

Tyson S. Hall
Southern Adventist University
Collegedale, TN

Michael D. Furman
University of Florida
Gainesville, FL

Library of Congress Control Number: 2007934543

ISBN 978-0-387-72670-0

e-ISBN 978-0-387-72671-7

Printed on acid-free paper.

© 2008 Springer Science+Business Media, LLC

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden. The use in this publication of trade names, trademarks, service marks and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs

Springer Science+Business Media, LLC or the author(s) make no warranty or representation, either express or implied, with respect to this DVD or book, including their quality, merchantability, or fitness for a particular purpose. In no event will Springer Science+Business Media, LLC or the author(s) be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the disc or book, even if Springer Science+Business Media, LLC or the author(s) has been advised of the possibility of such damages.

Cover artwork based on FPGA image courtesy of Altera. Chip Images ©1995-2004 courtesy of Michael Davidson, Florida State University, <http://micro.magnet.fsu.edu/chipshots>. Altera, Byteblaster*, Cyclone, MAX, APEX, ACEX and QUARTUS are registered trademarks of Altera Corporation. XC4000 and Virtex are registered trademarks of Xilinx, Inc. MIPS is a registered trademark of MIPS Technologies, Inc. Plexiglas is a registered trademark of Rohn and Hass Company. This publication includes images from Corel Draw which are protected by the copyright laws of the U.S., Canada and elsewhere. Used under license.

9 8 7 6 5 4 3 2 1

springer.com

RAPID PROTOTYPING OF DIGITAL SYSTEMS SOPC EDITION

Table of Contents

1	<i>Tutorial I: The 15 Minute Design</i>	2
1.1	Design Entry using the Graphic Editor	9
1.2	Compiling the Design	16
1.3	Simulation of the Design	17
1.4	Testing Your Design on an FPGA Board	18
1.5	Downloading Your Design to the DE1 Board	19
1.6	Downloading Your Design to the DE2 Board	22
1.7	Downloading Your Design to the UP3 Board	25
1.8	Downloading Your Design to the UP2 or UP1 Board	27
1.9	The 10 Minute VHDL Entry Tutorial	29
1.10	Compiling the VHDL Design	32
1.11	The 10 Minute Verilog Entry Tutorial	34
1.12	Compiling the Verilog Design	36
1.13	Timing Analysis	38
1.14	The Floorplan Editor	39
1.15	Symbols and Hierarchy	40
1.16	Functional Simulation	41
1.17	Laboratory Exercises	42
2	<i>FPGA Development Board Hardware and I/O Features</i>	46
2.1	FPGA and External Hardware Features	47
2.2	The FPGA Board's Memory Features	48
2.3	The FPGA Board's I/O Features	49
2.4	Obtaining an FPGA Development Board and Cables	53
3	<i>Programmable Logic Technology</i>	56
3.1	CPLDs and FPGAs	59
3.2	Altera MAX 7000S Architecture – A Product Term CPLD Device	60

3.3	Altera Cyclone Architecture – A Look-Up Table FPGA Device	62
3.4	Xilinx 4000 Architecture – A Look-Up Table FPGA Device	65
3.5	Computer Aided Design Tools for Programmable Logic	67
3.6	Next Generation FPGA CAD tools	68
3.7	Applications of FPGAs	69
3.8	Features of New Generation FPGAs	69
3.9	For additional information	70
3.10	Laboratory Exercises	71
4	<i>Tutorial II: Sequential Design and Hierarchy</i>	74
4.1	Install the Tutorial Files and FPGAcCore Library for your board	74
4.2	Open the tutor2 Schematic	75
4.3	Browse the Hierarchy	76
4.4	Using Buses in a Schematic	78
4.5	Testing the Pushbutton Counter and Displays	79
4.6	Testing the Initial Design on the Board	80
4.7	Fixing the Switch Contact Bounce Problem	81
4.8	Testing the Modified Design on the FPGA Board	82
4.9	Laboratory Exercises	83
5	<i>FPGAcCore Library Functions</i>	88
5.1	FPGAcCore LCD_Display: LCD Panel Character Display	90
5.2	FPGAcCore DEC_7SEG: Hex to Seven-segment Decoder	92
5.3	FPGAcCore Debounce: Pushbutton Debounce	94
5.4	FPGAcCore OnePulse: Pushbutton Single Pulse	95
5.5	FPGAcCore Clk_Div: Clock Divider	96
5.6	FPGAcCore VGA_Sync: VGA Video Sync Generation	97
5.7	FPGAcCore Char_ROM: Character Generation ROM	99
5.8	FPGAcCore Keyboard: Read Keyboard Scan Code	100
5.9	FPGAcCore Mouse: Mouse Cursor	102
5.10	For additional information	103
6	<i>Using VHDL for Synthesis of Digital Hardware</i>	106
6.1	VHDL Data Types	106
6.2	VHDL Operators	107
6.3	VHDL Based Synthesis of Digital Hardware	108
6.4	VHDL Synthesis Models of Gate Networks	108

6.5	VHDL Synthesis Model of a Seven-segment LED Decoder	109
6.6	VHDL Synthesis Model of a Multiplexer	111
6.7	VHDL Synthesis Model of Tri-State Output	112
6.8	VHDL Synthesis Models of Flip-flops and Registers	112
6.9	Accidental Synthesis of Inferred Latches	114
6.10	VHDL Synthesis Model of a Counter	114
6.11	VHDL Synthesis Model of a State Machine	115
6.12	VHDL Synthesis Model of an ALU with an Adder/Subtractor and a Shifter	117
6.13	VHDL Synthesis of Multiply and Divide Hardware	118
6.14	VHDL Synthesis Models for Memory	119
6.15	Hierarchy in VHDL Synthesis Models	123
6.16	Using a Testbench for Verification	125
6.17	For additional information	126
6.18	Laboratory Exercises	126
7	<i>Using Verilog for Synthesis of Digital Hardware</i>	130
7.1	Verilog Data Types	130
7.2	Verilog Based Synthesis of Digital Hardware	130
7.3	Verilog Operators	131
7.4	Verilog Synthesis Models of Gate Networks	132
7.5	Verilog Synthesis Model of a Seven-segment LED Decoder	132
7.6	Verilog Synthesis Model of a Multiplexer	133
7.7	Verilog Synthesis Model of Tri-State Output	134
7.8	Verilog Synthesis Models of Flip-flops and Registers	135
7.9	Accidental Synthesis of Inferred Latches	136
7.10	Verilog Synthesis Model of a Counter	136
7.11	Verilog Synthesis Model of a State Machine	137
7.12	Verilog Synthesis Model of an ALU with an Adder/Subtractor and a Shifter	138
7.13	Verilog Synthesis of Multiply and Divide Hardware	139
7.14	Verilog Synthesis Models for Memory	140
7.15	Hierarchy in Verilog Synthesis Models	143
7.16	For additional information	144
7.17	Laboratory Exercises	144
8	<i>State Machine Design: The Electric Train Controller</i>	148
8.1	The Train Control Problem	148

8.2	Train Direction Outputs (DA1-DA0, and DB1-DB0)	149
8.3	Switch Direction Outputs (SW1, SW2, and SW3)	150
8.4	Train Sensor Input Signals (S1, S2, S3, S4, and S5)	150
8.5	An Example Controller Design	151
8.6	VHDL Based Example Controller Design	154
8.7	Verilog Based Example Controller Design	157
8.8	Automatically Generating a State Diagram of a Design	160
8.9	Simulation Vector file for State Machine Simulation	161
8.10	Running the Train Control Simulation	162
8.11	Running the Video Train System (After Successful Simulation)	162
8.12	A Hardware Implementation of the Train System Layout	164
8.13	Laboratory Exercises	166
9	<i>A Simple Computer Design: The μP 3</i>	170
9.1	Computer Programs and Instructions	171
9.2	The Processor Fetch, Decode and Execute Cycle	172
9.3	VHDL Model of the μ P 3	179
9.4	Verilog Model of the μ P 3	182
9.5	Automatically Generating a State Diagram of the μ P3	186
9.6	Simulation of the μ P3 Computer	187
9.7	Laboratory Exercises	188
10	<i>VGA Video Display Generation using FPGAs</i>	192
10.1	Video Display Technology	192
10.2	Video Refresh	192
10.3	Using an FPGA for VGA Video Signal Generation	195
10.4	A VHDL Sync Generation Example: FPGAcore VGA_SYNC	196
10.5	Final Output Register for Video Signals	198
10.6	Required Pin Assignments for Video Output	198
10.7	Video Examples	199
10.8	A Character Based Video Design	200
10.9	Character Selection and Fonts	200
10.10	VHDL Character Display Design Examples	203
10.11	A Graphics Memory Design Example	206
10.12	Video Data Compression	207
10.13	Video Color Mixing using Dithering	207

10.14	VHDL Graphics Display Design Example	208
10.15	Higher Video Resolution and Faster Refresh Rates	209
10.16	Laboratory Exercises	210
11	<i>Interfacing to the PS/2 Keyboard and Mouse</i>	214
11.1	PS/2 Port Connections	214
11.2	Keyboard Scan Codes	215
11.3	Make and Break Codes	215
11.4	The PS/2 Serial Data Transmission Protocol	216
11.5	Scan Code Set 2 for the PS/2 Keyboard	218
11.6	The Keyboard FPGACore	220
11.7	A Design Example Using the Keyboard FPGACore	223
11.8	Interfacing to the PS/2 Mouse	224
11.9	The Mouse FPGACore	226
11.10	Mouse Initialization	226
11.11	Mouse Data Packet Processing	227
11.12	An Example Design Using the Mouse FPGACore	228
11.13	For Additional Information	229
11.14	Laboratory Exercises	229
12	<i>Legacy Digital I/O Interfacing Standards</i>	232
12.1	Parallel I/O Interface	232
12.2	RS-232C Serial I/O Interface	233
12.3	SPI Bus Interface	235
12.4	I ² C Bus Interface	237
12.5	For Additional Information	239
12.6	Laboratory Exercises	239
13	<i>FPGA Robotics Projects</i>	242
13.1	The FPGA-bot Design	242
13.2	FPGA-bot Servo Drive Motors	242
13.3	Modifying the Servos to make Drive Motors	243
13.4	VHDL Servo Driver Code for the FPGA-bot	244
13.5	Low-cost Sensors for an FPGA Robot Project	246
13.6	Assembly of the FPGA-bot Body	259
13.7	I/O Connections to the board's Expansion Headers	266
13.8	Robot Projects Based on R/C Toys, Models, and Robot Kits	267

13.9	For Additional Information _____	275
13.10	Laboratory Exercises _____	277
14	<i>A RISC Design: Synthesis of the MIPS Processor Core</i> _____	284
14.1	The MIPS Instruction Set and Processor _____	284
14.2	Using VHDL to Synthesize the MIPS Processor Core _____	287
14.3	The Top-Level Module _____	288
14.4	The Control Unit _____	291
14.5	The Instruction Fetch Stage _____	293
14.6	The Decode Stage _____	296
14.7	The Execute Stage _____	298
14.8	The Data Memory Stage _____	300
14.9	Simulation of the MIPS Design _____	301
14.10	MIPS Hardware Implementation on the FPGA Board _____	302
14.11	For Additional Information _____	303
14.12	Laboratory Exercises _____	304
15	<i>Introducing System-on-a-Programmable-Chip</i> _____	310
15.1	Processor Cores _____	310
15.2	SOPC Design Flow _____	311
15.3	Initializing Memory _____	313
15.4	SOPC Design versus Traditional Design Modalities _____	315
15.5	An Example SOPC Design _____	316
15.6	Hardware/Software Design Alternatives _____	317
15.7	For additional information _____	317
15.8	Laboratory Exercises _____	318
16	<i>Tutorial III: Nios II Processor Software Development</i> _____	322
16.1	Install the DE board files _____	322
16.2	Starting a Nios II Software Project _____	322
16.3	The Nios II IDE Software _____	324
16.4	Generating the Nios II System Library _____	325
16.5	Software Design with Nios II Peripherals _____	326
16.6	Starting Software Design – main() _____	329
16.7	Downloading the Nios II Hardware and Software Projects _____	330
16.8	Executing the Software _____	331
16.9	Starting Software Design for a Peripheral Test Program _____	331

16.10	Handling Interrupts	334
16.11	Accessing Parallel I/O Peripherals	335
16.12	Communicating with the LCD Display (<i>DE2 only</i>)	336
16.13	Testing SRAM	339
16.14	Testing Flash Memory	340
16.15	Testing SDRAM	341
16.16	Downloading the Nios II Hardware and Software Projects	346
16.17	Executing the Software	347
16.18	For additional information	347
16.19	Laboratory Exercises	348
17	<i>Tutorial IV: Nios II Processor Hardware Design</i>	352
17.1	Install the DE board files	352
17.2	Creating a New Project	352
17.3	Starting SOPC Builder	353
17.4	Adding a Nios II Processor	355
17.5	Adding UART Peripherals	358
17.6	Adding an Interval Timer Peripheral	359
17.7	Adding Parallel I/O Components	360
17.8	Adding an SRAM Memory Controller	361
17.9	Adding an SDRAM Memory Controller	362
17.10	Adding the LCD Module (<i>DE2 Board Only</i>)	362
17.11	Adding an External Bus	363
17.12	Adding Components to the External Bus	364
17.13	Global Processor Settings	364
17.14	Finalizing the Nios II Processor	365
17.15	Add the Processor Symbol to the Top-Level Schematic	366
17.16	Create a Phase-Locked Loop Component	367
17.17	Complete the Top-Level Schematic	368
17.18	Design Compilation	368
17.19	Testing the Nios II Project	369
17.20	For additional information	370
17.21	Laboratory Exercises	370
18	<i>Operating System Support for SOPC Design</i>	374
18.1	Nios II OS Support	376

18.2	eCos	377
18.3	μ C/OS-II	378
18.4	μ Clinux	379
18.5	Implementing the μ Clinux on the DE Board	380
18.6	Hardware Design for μ Clinux Support	380
18.7	Configuring the DE Board	382
18.8	Exploring μ Clinux on the DE Board	385
18.9	PS/2 Device Support in μ Clinux	386
18.10	Video Display in μ Clinux	386
18.11	USB Devices in μ Clinux (<i>DE2 Board Only</i>)	387
18.12	Network Communication in μ Clinux (<i>DE2 Board Only</i>)	387
18.13	For additional information	388
18.14	Laboratory Exercises	388
<hr/>		
<i>Appendix A: Generation of Pseudo Random Binary Sequences</i>		<i>391</i>
<i>Appendix B: Quartus II Design and Data File Extensions</i>		<i>393</i>
<i>Appendix C: Common FPGA Pin Assignments</i>		<i>394</i>
<i>Appendix D: ASCII Character Code</i>		<i>396</i>
<i>Appendix E: Common I/O Connector Pin Assignments</i>		<i>397</i>
<i>Glossary</i>		<i>399</i>
<i>Index</i>		<i>407</i>
<i>About the Accompanying DVD</i>		<i>411</i>

PREFACE

Changes to the SOPC Edition

Rapid Prototyping of Digital Systems provides an exciting and challenging laboratory component for undergraduate digital logic and computer design courses using FPGAs and CAD tools for simulation and hardware implementation. The more advanced topics and exercises also make this text useful for upper level courses in digital logic, programmable logic, and embedded systems. The SOPC edition includes Altera's new Quartus II CAD tool and includes laboratory projects for Altera's DE2 and the new DE1 FPGA boards. Student laboratory projects provided on the book's DVD include video graphics and text, mouse and keyboard input, and several computer designs.

Rapid Prototyping of Digital Systems includes four tutorials on the Altera Quartus II and Nios II tool environment, an overview of programmable logic, and IP cores with several easy-to-use input and output functions. These features were developed to help students get started quickly. Early design examples use schematic capture and IP cores developed for the Altera UP and DE FPGA boards. VHDL is used for more complex designs after a short introduction to VHDL-based synthesis. Verilog is also now supported as an option for the student projects.

New chapters in this edition provide an overview of System-On-a-Programmable Chip (SOPC) technology and SOPC design examples for the DE1 & 2 boards using Altera's new Nios II Processor hardware, the C software development tools, an overview of OS support for SOPC, and the uClinux operating system. A full set of Altera's FPGA CAD tools is included on the book's DVD.

Intended Audience

This text is intended to provide an exciting and challenging laboratory component for an undergraduate digital logic design class. The more advanced topics and exercises are also appropriate for consideration at schools that have an upper level course in digital logic or programmable logic. There are a number of excellent texts on digital logic design. For the most part, these texts do not include or fully integrate modern CAD tools, logic simulation, logic synthesis using hardware description languages, design hierarchy, current generation field programmable gate array (FPGA) technology and SOPC design. The goal of this text is to introduce these topics in the laboratory portion of the course. Even student laboratory projects can now implement entire digital and computer systems with hundreds of thousands of gates.

Over the past eight years, we have developed a number of interesting and challenging laboratory projects involving serial communications, state machines with video output, video games and graphics, simple computers, keyboard and mouse interfaces, robotics, and pipelined RISC processor cores.

Source files and additional example files are available on the DVD for all designs presented in the text. The student version of the PC based CAD tool on the DVD can be freely distributed to students. Students can purchase their own FPGA board for little more than the price of a contemporary textbook. As an alternative, a few of the low-cost FPGA boards can be shared among students in a laboratory. Course instructors should contact the Altera University Program for detailed information on obtaining full versions of the CAD tools for laboratory PCs and educational FPGA boards for student laboratories.

Topic Selection and Organization

Chapter 1 is a short CAD tool tutorial that covers design entry, simulation, and hardware implementation using an FPGA. The majority of students can enter the design, simulate, and have the design successfully running on the FPGA board in less than thirty minutes. After working through the tutorial and becoming familiar with the process, similar designs can be accomplished in less than 10 minutes.

Chapter 2 provides an overview of the various FPGA development boards. The features of each board are briefly described. Several tables listing pin connections of various I/O devices serve as an essential reference whenever a hardware design is implemented on the DE1, DE2, UP3, or UP 2 FPGA boards.

Chapter 3 is an introduction to programmable logic technology. The capabilities and internal architectures of the most popular CPLDs and FPGAs are described. These include the Cyclone FPGA used on the FPGA board, and the Xilinx 4000 family FPGAs.

Chapter 4 is a short CAD tool tutorial that serves as both a hierarchical and sequential design example. A counter is clocked by a pushbutton and the output is displayed in the seven-segment LEDs. The design is downloaded to the FPGA board and some real world timing issues arising from switch contact bounce are resolved. It uses several functions from the FPGAcore library which greatly simplify use of the FPGA's input and output capabilities.

Chapter 5 describes the available FPGAcore library I/O functions. The I/O devices include switches, the LCD, a decoder for seven segment LEDs, a multiple output clock divider, VGA output, keyboard input, and mouse input.

Chapter 6 is an introduction to the use of VHDL for the synthesis of digital hardware. Rather than a lengthy description of syntax details, models of the commonly used digital hardware devices are developed and presented. Most VHDL textbooks use models developed only for simulation and frequently use language features not supported in synthesis tools. Our easy to understand synthesis examples were developed and tested on FPGAs using the Altera CAD tools.

Chapter 7 is an introduction to the use of Verilog for the synthesis of digital hardware. The same hardware designs as Chapter 6 as modeled in Verilog. It is optional, but is included for those who would like an introduction to Verilog.

Chapter 8 is a state machine design example. The state machine controls a virtual electric train simulation with video output generated directly by the FPGA. Using track sensor input, students must control two trains and three

track switches to avoid collisions. An actual model train layout can also be built using the new digital DCC trains interfaced to an FPGA board.

Chapter 9 develops a model of a simple computer. The fetch, decode, and execute cycle is introduced and a brief model of the computer is developed using VHDL. A short assembly language program can be entered in the FPGA's internal memory and executed in the simulator.

Chapter 10 describes how to design an FPGA-based digital system to output VGA video. Numerous design examples are presented containing video with both text and graphics. Fundamental design issues in writing simple video games and graphics using an FPGA board are examined.

Chapter 11 describes the PS/2 keyboard and mouse operation and presents interface examples for integrating designs on an FPGA board. Keyboard scan code tables, mouse data packets, commands, status codes, and the serial communications protocol are included. VHDL code for a keyboard and mouse interface is also presented.

Chapter 12 describes several of the common I/O standards that are likely to be encountered in FPGA systems. Parallel, RS232 serial, SPI, and I²C standards and interfacing are discussed.

Chapter 13 develops a design for an adaptable mobile robot using an FPGA board as the controller. Servo motors and several sensor technologies for a low cost mobile robot are described. A sample servo driver design is presented. Commercially available parts to construct the robot described can be obtained for as little as \$60. Several robots can be built for use in the laboratory. Students with their own FPGA board may choose to build their own robot following the detailed instructions found in section 13.6.

Chapter 14 describes a single clock cycle model of the MIPS RISC processor based on the hardware implementation presented in the widely used Patterson and Hennessy textbook, *Computer Organization and Design the Hardware/Software Interface*. Laboratory exercises that add new instructions, features, and pipelining are included at the end of the chapter.

Chapters 15, 16, and 17 introduce students to SOPC design using the Nios II RISC processor core. Chapter 15 is an overview of the SOPC design approach. Chapter 16 contains a tutorial for the Nios II IDE software development tool and examples using the Nios II C/C++ compiler. Chapter 17 contains a tutorial on the processor core hardware configuration tool, SOPC builder. A DE2, DE1, or FPGA board is required for this new material since it is not supported on the UP2 or UP1's smaller FPGA.

Chapter 18 is new to the fourth edition and introduces students to a Linux based Real-Time Operating System (RTOS). A tutorial shows how the μ Clinux OS can be ported to the DE2 and DE1 FPGA boards.

We anticipate that some schools will still choose to begin with TTL designs on a small protoboard for the first few labs. The first chapter can be started at this time since only OR and NOT logic functions are used to introduce the CAD tool environment. The CAD tool can also be used for simulation of TTL labs, since a TTL parts library is included.

Even though VHDL and Verilog are complex languages, we have found after several years of experimentation that students can write HDL models to synthesize hardware designs after a short overview with a few basic hardware design examples. The use of HDL templates and online help files in the CAD tool make this process easier. After the initial experience with HDL synthesis, students dislike the use of schematic capture on larger designs since it can be time consuming. Experience in industry has been much the same since large productivity gains have been achieved using HDL based synthesis tools for FPGAs and Application Specific Integrated Circuits (ASICs).

Most digital logic classes include a simple computer design such as the one presented in Chapter 9 or a RISC processor such as the one presented in Chapter 14. If this is not covered in the first digital logic course, it could be used as a lab component for a subsequent computer architecture class.

A typical quarter or semester length course could not cover all of the topics presented. The material in Chapters 7 through 17 can be used on a selective basis. The keyboard and mouse are supported by FPGAcore library functions, and the material presented in Chapter 11 is not required to use these library functions for keyboard or mouse input. A DE1, DE2, or FPGA board is required for the SOPC Nios designs in Chapters 16 and 17.

A video game based on the material in Chapter 10 can serve as the basis for a final design project. We use robots with sensors from Chapter 13 that are controlled by the simple computer in Chapter 9. Students really enjoy working with the robot, and it presents almost infinite possibilities for an exciting design competition. More advanced classes might want to develop projects based on the Nios II processor reference design in Chapter 16 and 17 using C/C++ code or use the uClinux material in Chapter 18 to develop more complex application programs for embedded devices.

Software and Hardware Packages

We recommend the use of the new 7.1 SP1 web version of Quartus II FPGA CAD included with this book; all exercises were tested using this version. FPGA boards are available from the Altera University Program at special student pricing. Although boards can be easily shared among several students in a lab setting, pricing makes it possible for students who would like to purchase their own to do so.

Details and suggestions for additional cables that may be required for a laboratory setup can be found in Section 2.4. Source files for all designs presented in the text are available on the DVD.

Additional Web Material and Resources

There is a web site for the text with additional course materials, slides, text errata, and software updates at:

<http://www.ece.gatech.edu/users/hamblen/book/book4e.htm>

Acknowledgments

Over three thousand students and several hundred teaching assistants have contributed to this work during the past eight years. In particular, we would like to acknowledge Doug McAlister, Michael Sugg, Jurgen Vogel, Greg Ruhl, Eric Van Heest, Mitch Kispet, Evan Anderson, Zachary Folkerts, and Nick Clark for their help in testing and developing several of the laboratory assignments and tools. Stephen Brown, Mike Phipps, Joe Hanson, Tawfiq Mossadak, and Eric Shiflet at Altera provided software, hardware, helpful advice, and encouragement.