FPGA PROTOTYPING
BY VHDL EXAMPLES
FPGA PROTOTYPING
BY VHDL EXAMPLES
Xilinx Spartan™-3 Version

Pong P. Chu
Cleveland State University
To my parents, Chia-Chi and Chi-Te, my wife, Lee, and my daughter, Patricia
CONTENTS

Preface
Acknowledgments

PART I BASIC DIGITAL CIRCUITS

1 Gate-level combinational circuit
   1.1 Introduction
   1.2 General description
      1.2.1 Basic lexical rules
      1.2.2 Library and package
      1.2.3 Entity declaration
      1.2.4 Data type and operators
      1.2.5 Architecture body
      1.2.6 Code of a 2-bit comparator
   1.3 Structural description
   1.4 Testbench
   1.5 Bibliographic notes
   1.6 Suggested experiments
      1.6.1 Code for gate-level greater-than circuit
      1.6.2 Code for gate-level binary decoder

2 Overview of FPGA and EDA software
2.1 Introduction 11
2.2 FPGA 11
    2.2.1 Overview of a general FPGA device 11
    2.2.2 Overview of the Xilinx Spartan-3 devices 13
2.3 Overview of the Digilent S3 board 13
2.4 Development flow 15
2.5 Overview of the Xilinx ISE project navigator 17
2.6 Short tutorial on ISE project navigator 19
    2.6.1 Create the design project and HDL codes 21
    2.6.2 Create a testbench and perform the RTL simulation 22
    2.6.3 Add a constraint file and synthesize and implement the code 22
    2.6.4 Generate and download the configuration file to an FPGA device 24
2.7 Short tutorial on the ModelSim HDL simulator 27
2.8 Bibliographic notes 32
2.9 Suggested experiments 33
    2.9.1 Gate-level greater-than circuit 33
    2.9.2 Gate-level binary decoder 33

3 RT-level combinational circuit 35
3.1 Introduction 35
3.2 RT-level components 35
    3.2.1 Relational operators 37
    3.2.2 Arithmetic operators 37
    3.2.3 Other synthesis-related VHDL constructs 38
    3.2.4 Summary 40
3.3 Routing circuit with concurrent assignment statements 41
    3.3.1 Conditional signal assignment statement 41
    3.3.2 Selected signal assignment statement 44
3.4 Modeling with a process 46
    3.4.1 Process 46
    3.4.2 Sequential signal assignment statement 46
3.5 Routing circuit with if and case statements 47
    3.5.1 If statement 47
    3.5.2 Case statement 49
    3.5.3 Comparison to concurrent statements 50
    3.5.4 Unintended memory 52
3.6 Constants and generics 53
    3.6.1 Constants 53
    3.6.2 Generics 54
3.7 Design examples 56
    3.7.1 Hexadecimal digit to seven-segment LED decoder 56
    3.7.2 Sign-magnitude adder 59
### 4 Regular Sequential Circuit

4.1 Introduction  
4.1.1 D FF and register  
4.1.2 Synchronous system  
4.1.3 Code development  
4.2 HDL code of the FF and register  
4.2.1 D FF  
4.2.2 Register  
4.2.3 Register file  
4.2.4 Storage components in a Spartan-3 device\textsuperscript{\textregistered} Xilinx specific  
4.3 Simple design examples  
4.3.1 Shift register  
4.3.2 Binary counter and variant  
4.4 Testbench for sequential circuits  
4.5 Case study  
4.5.1 LED time-multiplexing circuit  
4.5.2 Stopwatch  
4.5.3 FIFO buffer  
4.6 Bibliographic notes  
4.7 Suggested experiments  
4.7.1 Programmable square wave generator  
4.7.2 PWM and LED dimmer  
4.7.3 Rotating square circuit  
4.7.4 Heartbeat circuit  
4.7.5 Rotating LED banner circuit  
4.7.6 Enhanced stopwatch  
4.7.7 Stack

### 5 FSM

5.1 Introduction
5.1 Mealy and Moore outputs
  5.1.1 Mealy and Moore outputs
  5.1.2 FSM representation

5.2 FSM code development

5.3 Design examples
  5.3.1 Rising-edge detector
  5.3.2 Debouncing circuit
  5.3.3 Testing circuit

5.4 Bibliographic notes

5.5 Suggested experiments
  5.5.1 Dual-edge detector
  5.5.2 Alternative debouncing circuit
  5.5.3 Parking lot occupancy counter

5.6 Bibliographic notes

6 FSMD

6.1 Introduction
  6.1.1 Single RT operation
  6.1.2 ASMD chart
  6.1.3 Decision box with a register

6.2 Code development of an FSMD
  6.2.1 Debouncing circuit based on RT methodology
  6.2.2 Code with explicit data path components
  6.2.3 Code with implicit data path components
  6.2.4 Comparison
  6.2.5 Testing circuit

6.3 Design examples
  6.3.1 Fibonacci number circuit
  6.3.2 Division circuit
  6.3.3 Binary-to-BCD conversion circuit
  6.3.4 Period counter
  6.3.5 Accurate low-frequency counter

6.4 Bibliographic notes

6.5 Suggested experiments
  6.5.1 Alternative debouncing circuit
  6.5.2 BCD-to-binary conversion circuit
  6.5.3 Fibonacci circuit with BCD I/O: design approach 1
  6.5.4 Fibonacci circuit with BCD I/O: design approach 2
  6.5.5 Auto-scaled low-frequency counter
  6.5.6 Reaction timer
  6.5.7 Babbage difference engine emulation circuit

PART II I/O MODULES
7 UART

7.1 Introduction
7.2 UART receiving subsystem
  7.2.1 Oversampling procedure
  7.2.2 Baud rate generator
  7.2.3 UART receiver
  7.2.4 Interface circuit
7.3 UART transmitting subsystem
7.4 Overall UART system
  7.4.1 Complete UART core
  7.4.2 UART verification configuration
7.5 Customizing a UART
7.6 Bibliographic notes
7.7 Suggested experiments
  7.7.1 Full-featured UART
  7.7.2 UART with an automatic baud rate detection circuit
  7.7.3 UART with an automatic baud rate and parity detection circuit
  7.7.4 UART-controlled stopwatch
  7.7.5 UART-controlled rotating LED banner

8 PS2 Keyboard

8.1 Introduction
8.2 PS2 receiving subsystem
  8.2.1 Physical interface of a PS2 port
  8.2.2 Device-to-host communication protocol
  8.2.3 Design and code
8.3 PS2 keyboard scan code
  8.3.1 Overview of the scan code
  8.3.2 Scan code monitor circuit
8.4 PS2 keyboard interface circuit
  8.4.1 Basic design and HDL code
  8.4.2 Verification circuit
8.5 Bibliographic notes
8.6 Suggested experiments
  8.6.1 Alternative keyboard interface I
  8.6.2 Alternative keyboard interface II
  8.6.3 PS2 receiving subsystem with watchdog timer
  8.6.4 Keyboard-controlled stopwatch
  8.6.5 Keyboard-controlled rotating LED banner

9 PS2 Mouse

9.1 Introduction
9.2 PS2 receiving subsystem
  9.2.1 Physical interface of a PS2 port
  9.2.2 Device-to-host communication protocol
  9.2.3 Design and code
9.3 PS2 keyboard scan code
  9.3.1 Overview of the scan code
  9.3.2 Scan code monitor circuit
9.4 PS2 keyboard interface circuit
  9.4.1 Basic design and HDL code
  9.4.2 Verification circuit
9.5 Bibliographic notes
9.6 Suggested experiments
  9.6.1 Alternative keyboard interface I
  9.6.2 Alternative keyboard interface II
  9.6.3 PS2 receiving subsystem with watchdog timer
  9.6.4 Keyboard-controlled stopwatch
  9.6.5 Keyboard-controlled rotating LED banner

9.1 Introduction
9.2 PS2 receiving subsystem
  9.2.1 Physical interface of a PS2 port
  9.2.2 Device-to-host communication protocol
  9.2.3 Design and code
9.3 PS2 keyboard scan code
  9.3.1 Overview of the scan code
  9.3.2 Scan code monitor circuit
9.4 PS2 keyboard interface circuit
  9.4.1 Basic design and HDL code
  9.4.2 Verification circuit
9.5 Bibliographic notes
9.6 Suggested experiments
  9.6.1 Alternative keyboard interface I
  9.6.2 Alternative keyboard interface II
  9.6.3 PS2 receiving subsystem with watchdog timer
  9.6.4 Keyboard-controlled stopwatch
  9.6.5 Keyboard-controlled rotating LED banner

9.1 Introduction
9.2 PS2 receiving subsystem
  9.2.1 Physical interface of a PS2 port
  9.2.2 Device-to-host communication protocol
  9.2.3 Design and code
9.3 PS2 keyboard scan code
  9.3.1 Overview of the scan code
  9.3.2 Scan code monitor circuit
9.4 PS2 keyboard interface circuit
  9.4.1 Basic design and HDL code
  9.4.2 Verification circuit
9.5 Bibliographic notes
9.6 Suggested experiments
  9.6.1 Alternative keyboard interface I
  9.6.2 Alternative keyboard interface II
  9.6.3 PS2 receiving subsystem with watchdog timer
  9.6.4 Keyboard-controlled stopwatch
  9.6.5 Keyboard-controlled rotating LED banner

9.1 Introduction
9.2 PS2 receiving subsystem
  9.2.1 Physical interface of a PS2 port
  9.2.2 Device-to-host communication protocol
  9.2.3 Design and code
9.3 PS2 keyboard scan code
  9.3.1 Overview of the scan code
  9.3.2 Scan code monitor circuit
9.4 PS2 keyboard interface circuit
  9.4.1 Basic design and HDL code
  9.4.2 Verification circuit
9.5 Bibliographic notes
9.6 Suggested experiments
  9.6.1 Alternative keyboard interface I
  9.6.2 Alternative keyboard interface II
  9.6.3 PS2 receiving subsystem with watchdog timer
  9.6.4 Keyboard-controlled stopwatch
  9.6.5 Keyboard-controlled rotating LED banner

9.1 Introduction
9.2 PS2 receiving subsystem
  9.2.1 Physical interface of a PS2 port
  9.2.2 Device-to-host communication protocol
  9.2.3 Design and code
9.3 PS2 keyboard scan code
  9.3.1 Overview of the scan code
  9.3.2 Scan code monitor circuit
9.4 PS2 keyboard interface circuit
  9.4.1 Basic design and HDL code
  9.4.2 Verification circuit
9.5 Bibliographic notes
9.6 Suggested experiments
  9.6.1 Alternative keyboard interface I
  9.6.2 Alternative keyboard interface II
  9.6.3 PS2 receiving subsystem with watchdog timer
  9.6.4 Keyboard-controlled stopwatch
  9.6.5 Keyboard-controlled rotating LED banner
9.1 Introduction  199
9.2 PS2 mouse protocol  200
  9.2.1 Basic operation  200
  9.2.2 Basic initialization procedure  200
9.3 PS2 transmitting subsystem  201
  9.3.1 Host-to-PS2-device communication protocol  201
  9.3.2 Design and code  202
9.4 Bidirectional PS2 interface  206
  9.4.1 Basic design and code  206
  9.4.2 Verification circuit  208
9.5 PS2 mouse interface  210
  9.5.1 Basic design  210
  9.5.2 Testing circuit  212
9.6 Bibliographic notes  214
9.7 Suggested experiments  214
  9.7.1 Keyboard control circuit  214
  9.7.2 Enhanced mouse interface  214
  9.7.3 Mouse-controlled seven-segment LED display  214

10 External SRAM  215
10.1 Introduction  215
10.2 Specification of the IS61LV25616AL SRAM  216
  10.2.1 Block diagram and I/O signals  216
  10.2.2 Timing parameters  216
10.3 Basic memory controller  220
  10.3.1 Block diagram  220
  10.3.2 Timing requirement  221
  10.3.3 Register file versus SRAM  222
10.4 A safe design  222
  10.4.1 ASMD chart  222
  10.4.2 Timing analysis  223
  10.4.3 HDL implementation  224
  10.4.4 Basic testing circuit  226
  10.4.5 Comprehensive SRAM testing circuit  228
10.5 More aggressive design  233
  10.5.1 Timing issues  233
  10.5.2 Alternative design I  234
  10.5.3 Alternative design II  236
  10.5.4 Alternative design III  237
  10.5.5 Advanced FPGA features\textit{Xilinx specific}  237
10.6 Bibliographic notes  240
10.7 Suggested experiments  240
10.7.1 Memory with a 512K-by-16 configuration 240
10.7.2 Memory with a 1M-by-8 configuration 240
10.7.3 Memory with an 8M-by-1 configuration 240
10.7.4 Expanded memory testing circuit 241
10.7.5 Memory controller and testing circuit for alternative design I 241
10.7.6 Memory controller and testing circuit for alternative design II 241
10.7.7 Memory controller and testing circuit for alternative design III 241
10.7.8 Memory controller with DCM 241
10.7.9 High-performance memory controller 241

11 Xilinx Spartan-3 Specific Memory 243

11.1 Introduction 243
11.2 Embedded memory of Spartan-3 device 243
  11.2.1 Overview 243
  11.2.2 Comparison 244
11.3 Method to incorporate memory modules 244
  11.3.1 Memory module via HDL component instantiation 245
  11.3.2 Memory module via Core Generator 245
  11.3.3 Memory module via HDL inference 246
11.4 HDL templates for memory inference 246
  11.4.1 Single-port RAM 246
  11.4.2 Dual-port RAM 249
  11.4.3 ROM 251
11.5 Bibliographic notes 254
11.6 Suggested experiments 254
  11.6.1 Block-RAM-based FIFO 254
  11.6.2 Block-RAM-based stack 254
  11.6.3 ROM-based sign-magnitude adder 255
  11.6.4 ROM based \( \sin(x) \) function 255
  11.6.5 ROM-based \( \sin(x) \) and \( \cos(x) \) functions 255

12 VGA controller I: graphic 257

12.1 Introduction 257
  12.1.1 Basic operation of a CRT 257
  12.1.2 VGA port of the S3 board 259
  12.1.3 Video controller 259
12.2 VGA synchronization 260
  12.2.1 Horizontal synchronization 260
  12.2.2 Vertical synchronization 262
  12.2.3 Timing calculation of VGA synchronization signals 263
  12.2.4 HDL implementation 263
CONTENTS

12.2.5 Testing circuit 266
12.3 Overview of the pixel generation circuit 267
12.4 Graphic generation with an object-mapped scheme 268
12.4.1 Rectangular objects 269
12.4.2 Non-rectangular object 273
12.4.3 Animated object 275
12.5 Graphic generation with a bit-mapped scheme 282
12.5.1 Dual-port RAM implementation 282
12.5.2 Single-port RAM implementation 287
12.6 Bibliographic notes 287
12.7 Suggested experiments 287
12.7.1 VGA test pattern generator 287
12.7.2 SVGA mode synchronization circuit 288
12.7.3 Visible screen adjustment circuit 288
12.7.4 Ball-in-a-box circuit 288
12.7.5 Two-balls-in-a-box circuit 289
12.7.6 Two-player pong game 289
12.7.7 Breakout game 289
12.7.8 Full-screen dot trace 289
12.7.9 Mouse pointer circuit 290
12.7.10 Small-screen mouse scribble circuit 290
12.7.11 Full-screen mouse scribble circuit 290

13 VGA controller II: text 291

13.1 Introduction 291
13.2 Text generation 291
13.2.1 Character as a tile 291
13.2.2 Font ROM 292
13.2.3 Basic text generation circuit 294
13.2.4 Font display circuit 295
13.2.5 Font scaling 297
13.3 Full-screen text display 298
13.4 The complete pong game 302
13.4.1 Text subsystem 302
13.4.2 Modified graphic subsystem 309
13.4.3 Auxiliary counters 310
13.4.4 Top-level system 312
13.5 Bibliographic notes 317
13.6 Suggested experiments 317
13.6.1 Rotating banner 317
13.6.2 Underline for the cursor 317
13.6.3 Dual-mode text display 317
PART III PICOBLAZE MICROCONTROLLER

14 PicoBlaze Overview

14.1 Introduction

14.2 Customized hardware and customized software

14.2.1 From special-purpose FSMD to general-purpose microcontroller

14.2.2 Application of microcontroller

14.3 Overview of PicoBlaze

14.3.1 Basic organization

14.3.2 Top-level HDL modules

14.4 Development flow

14.5 Instruction set

14.5.1 Programming model

14.5.2 Instruction format

14.5.3 Logical instructions

14.5.4 Arithmetic instructions

14.5.5 Compare and test instructions

14.5.6 Shift and rotate instructions

14.5.7 Data movement instructions

14.5.8 Program flow control instructions

14.5.9 Interrupt related instructions

14.6 Assembler directives

14.6.1 The KCPSM3 directives

14.6.2 The PBlazeIDE directives

14.7 Bibliographic notes

15 PicoBlaze Assembly Code Development

15.1 Introduction

15.2 Useful code segments

15.2.1 KCPSM3 conventions

15.2.2 Bit manipulation

15.2.3 Multiple-byte manipulation

15.2.4 Control structure

15.3 Subroutine development

15.4 Program development
## CONTENTS

15.4.1 Demonstration example 352
15.4.2 Program documentation 356
15.5 Processing of the assembly code 358
15.5.1 Compiling with KCSPM3 358
15.5.2 Simulation by PBlazeIDE 359
15.5.3 Reloading code via the JTAG port 362
15.5.4 Compiling by PBlazeIDE 362
15.6 Syntheses with PicoBlaze 363
15.7 Bibliographic notes 364
15.8 Suggested experiments 365
15.8.1 Signed multiplication 365
15.8.2 Multi-byte multiplication 365
15.8.3 Barrel shift function 365
15.8.4 Reverse function 365
15.8.5 Binary-to-BCD conversion 365
15.8.6 BCD-to-binary conversion 365
15.8.7 Heartbeat circuit 365
15.8.8 Rotating LED circuit 366
15.8.9 Discrete LED dimmer 366

### 16 PicoBlaze I/O Interface

16.1 Introduction 367
16.2 Output port
- 16.2.1 Output instruction and timing 368
- 16.2.2 Output interface 369
16.3 Input port
- 16.3.1 Input instruction and timing 371
- 16.3.2 Input interface 371
16.4 Square program with a switch and seven-segment LED display interface
- 16.4.1 Output interface 374
- 16.4.2 Input interface 375
- 16.4.3 Assembly code development 376
- 16.4.4 VHDL code development 384
16.5 Square program with a combinational multiplier and UART console 386
- 16.5.1 Multiplier interface 387
- 16.5.2 UART interface 387
- 16.5.3 Assembly code development 389
- 16.5.4 VHDL code development 398
16.6 Bibliographic notes 402
16.7 Suggested experiments
- 16.7.1 Low-frequency counter I 402
- 16.7.2 Low-frequency counter II 402
16.7.3 Auto-scaled low-frequency counter  
16.7.4 Basic reaction timer with a software timer  
16.7.5 Basic reaction timer with a hardware timer  
16.7.6 Enhanced reaction timer  
16.7.7 Small-screen mouse scribble circuit  
16.7.8 Full-screen mouse scribble circuit  
16.7.9 Enhanced rotating banner  
16.7.10 Pong game  
16.7.11 Text editor

17 PicoBlaze Interrupt Interface  
17.1 Introduction  
17.2 Interrupt handling in PicoBlaze  
17.2.1 Software processing  
17.2.2 Timing  
17.3 External interface  
17.3.1 Single interrupt request  
17.3.2 Multiple interrupt requests  
17.4 Software development considerations  
17.4.1 Interrupt as an alternative scheduling scheme  
17.4.2 Development of an interrupt service routine  
17.5 Design example  
17.5.1 Interrupt interface  
17.5.2 Interrupt service routine development  
17.5.3 Assembly code development  
17.5.4 VHDL code development  
17.6 Bibliographic notes  
17.7 Suggested experiments  
17.7.1 Alternative timer interrupt service routine  
17.7.2 Programmable timer  
17.7.3 Set-button interrupt service routine  
17.7.4 Interrupt interface with two requests  
17.7.5 Four-request interrupt controller

Appendix A: Sample VHDL templates  
A.1 General VHDL constructs  
A.1.1 Overall code structure  
A.1.2 Component instantiation  
A.2 Combinational circuits  
A.2.1 Arithmetic operations  
A.2.2 Fixed-amount shift operations
A.2.3 Routing with concurrent statements
A.2.4 Routing with if and case statements
A.2.5 Combinational circuit using process
A.3 Memory Components
  A.3.1 Register template
  A.3.2 Register file
A.4 Regular sequential circuits
A.5 FSM
A.6 FSMD
A.7 S3 board constraint file (s3.ucf)

References

Topic Index
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 is limited to 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 VHDL, and a separate book with a similar title uses
Verilog. 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 com-
parability.

**Audience and perquisites** 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 is 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 need. 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 in other boards as well. The applicability of the HDL codes is summarized below.

- **Spartan-3 Starter 3 (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 in the S3 board. There are two differences. First, the “color depth” of its VGA interface is expanded from 3 bits to 8 bits. The the output of the VGA interface circuits discussed in Chapters 12 and 13 needs to be modified accordingly. Second, it contains a more sophisticated external memory device. Although the device can be configured as an asynchronous SRAM, the timing characteristics is different from that of the S3 board’s memory device, and thus the HDL codes for the memory controller in Chapter 10 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 7, 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 10 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.

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 7 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 8 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 9 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 10 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 11 discusses the inference and application of Spartan-3 device-specific components. The focus is on the FPGA’s internal memory blocks and the digital clock management (DCM) circuit.

Chapter 12 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 interface. The monitor is connected to the prototyping board’s VGA port.

Chapter 13 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 14 provides an overview of the organization and instruction set of PicoBlaze.

Chapter 15 introduces the basic assembly programming and provides an overview of the development process.

Chapter 16 discusses PicoBlaze’s I/O feature and illustrates the procedure to derive customized circuits to interface other I/O peripherals.

Chapter 17 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**

While the examples of this 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 device-independent and software-neutral as much 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, such as the one shown on the outer edge, to indicate that the discussion in the 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.

**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 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
October 2007
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, and 6.5 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 8.3 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