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 comparability*.

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 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" and the materials are included to help readers understand the older Verilog codes.

FYI

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