RTL HARDWARE DESIGN USING VHDL

Coding for Efficiency, Portability, and Scalability

PONG P. CHU
RTL HARDWARE DESIGN USING VHDL
Coding for Efficiency, Portability, and Scalability

PONG P. CHU
Cleveland State University

WILEY-INTERSCIENCE
A JOHN WILEY & SONS, INC., PUBLICATION
This Page Intentionally Left Blank
This Page Intentionally Left Blank
RTL HARDWARE DESIGN USING VHDL
Coding for Efficiency, Portability, and Scalability

PONG P. CHU
Cleveland State University
To my parents Chia-Chi and Chi-Te, my wife Lee, and my daughter Patricia
This Page Intentionally Left Blank
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td></td>
<td>xix</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td></td>
<td>xxiii</td>
</tr>
<tr>
<td>1 Introduction to Digital System Design</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Device technologies</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Fabrication of an IC</td>
<td>2</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Classification of device technologies</td>
<td>2</td>
</tr>
<tr>
<td>1.2.3</td>
<td>Comparison of technologies</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td>System representation</td>
<td>8</td>
</tr>
<tr>
<td>1.4</td>
<td>Levels of Abstraction</td>
<td>9</td>
</tr>
<tr>
<td>1.4.1</td>
<td>Transistor-level abstraction</td>
<td>10</td>
</tr>
<tr>
<td>1.4.2</td>
<td>Gate-level abstraction</td>
<td>10</td>
</tr>
<tr>
<td>1.4.3</td>
<td>Register-transfer-level (RT-level) abstraction</td>
<td>11</td>
</tr>
<tr>
<td>1.4.4</td>
<td>Processor-level abstraction</td>
<td>12</td>
</tr>
<tr>
<td>1.5</td>
<td>Development tasks and EDA software</td>
<td>12</td>
</tr>
<tr>
<td>1.5.1</td>
<td>Synthesis</td>
<td>13</td>
</tr>
<tr>
<td>1.5.2</td>
<td>Physical design</td>
<td>14</td>
</tr>
<tr>
<td>1.5.3</td>
<td>Verification</td>
<td>14</td>
</tr>
<tr>
<td>1.5.4</td>
<td>Testing</td>
<td>16</td>
</tr>
<tr>
<td>1.5.5</td>
<td>EDA software and its limitations</td>
<td>16</td>
</tr>
</tbody>
</table>
# CONTENTS

1.6 Development flow  
1.6.1 Flow of a medium-sized design targeting FPGA  
1.6.2 Flow of a large design targeting FPGA  
1.6.3 Flow of a large design targeting ASIC  
1.7 Overview of the book  
1.7.1 Scope  
1.7.2 Goal  
1.8 Bibliographic notes  
Problems

## 2 Overview of Hardware Description Languages

2.1 Hardware description languages  
2.1.1 Limitations of traditional programming languages  
2.1.2 Use of an HDL program  
2.1.3 Design of a modern HDL  
2.1.4 VHDL  
2.2 Basic VHDL concept via an example  
2.2.1 General description  
2.2.2 Structural description  
2.2.3 Abstract behavioral description  
2.2.4 Testbench  
2.2.5 Configuration  
2.3 VHDL in development flow  
2.3.1 Scope of VHDL  
2.3.2 Coding for synthesis  
2.4 Bibliographic notes  
Problems

## 3 Basic Language Constructs of VHDL

3.1 Introduction  
3.2 Skeleton of a basic VHDL program  
3.2.1 Example of a VHDL program  
3.2.2 Entity declaration  
3.2.3 Architecture body  
3.2.4 Design unit and library  
3.2.5 Processing of VHDL code  
3.3 Lexical elements and program format  
3.3.1 Lexical elements  
3.3.2 VHDL program format  
3.4 Objects  
3.5 Data types and operators
## 5.3 Variable assignment statement

### 5.4 If statement

- **5.4.1 Syntax and examples**
- **5.4.2 Comparison to a conditional signal assignment statement**
- **5.4.3 Incomplete branch and incomplete signal assignment**
- **5.4.4 Conceptual implementation**
- **5.4.5 Cascading single- branched if statements**

### 5.5 Case statement

- **5.5.1 Syntax and examples**
- **5.5.2 Comparison to a selected signal assignment statement**
- **5.5.3 Incomplete signal assignment**
- **5.5.4 Conceptual implementation**

### 5.6 Simple for loop statement

- **5.6.1 Syntax**
- **5.6.2 Examples**
- **5.6.3 Conceptual implementation**

### 5.7 Synthesis of sequential statements

### 5.8 Synthesis guidelines

- **5.8.1 Guidelines for using sequential statements**
- **5.8.2 Guidelines for combinational circuits**

### 5.9 Bibliographic notes

### Problems

## 6 Synthesis Of VHDL Code

### 6.1 Fundamental limitations of EDA software

- **6.1.1 Computability**
- **6.1.2 Computation complexity**
- **6.1.3 Limitations of EDA software**

### 6.2 Realization of VHDL operators

- **6.2.1 Realization of logical operators**
- **6.2.2 Realization of relational operators**
- **6.2.3 Realization of addition operators**
- **6.2.4 Synthesis support for other operators**
- **6.2.5 Realization of an operator with constant operands**
- **6.2.6 An example implementation**

### 6.3 Realization of VHDL data types

- **6.3.1 Use of the `std_logic` data type**
- **6.3.2 Use and realization of the 'Z' value**
- **6.3.3 Use of the '-' value**

### 6.4 VHDL synthesis flow

- **6.4.1 RT-level synthesis**
- **6.4.2 Module generator**
6.4.3 Logic synthesis 142
6.4.4 Technology mapping 143
6.4.5 Effective use of synthesis software 148

6.5 Timing considerations 149
6.5.1 Propagation delay 150
6.5.2 Synthesis with timing constraints 154
6.5.3 Timing hazards 156
6.5.4 Delay-sensitive design and its dangers 158

6.6 Synthesis guidelines 160
6.7 Bibliographic notes 160
Problems 160

### 7 Combinational Circuit Design: Practice 163

7.1 Derivation of efficient HDL description 163
7.2 Operator sharing 164
  7.2.1 Sharing example 1 165
  7.2.2 Sharing example 2 166
  7.2.3 Sharing example 3 168
  7.2.4 Sharing example 4 169
  7.2.5 Summary 170
7.3 Functionality sharing 170
  7.3.1 Addition–subtraction circuit 171
  7.3.2 Signed–unsigned dual-mode comparator 173
  7.3.3 Difference circuit 175
  7.3.4 Full comparator 177
  7.3.5 Three-function barrel shifter 178
7.4 Layout-related circuits 180
  7.4.1 Reduced-xor circuit 181
  7.4.2 Reduced-xor-vector circuit 183
  7.4.3 Tree priority encoder 187
  7.4.4 Barrel shifter revisited 192
7.5 General circuits 196
  7.5.1 Gray code incrementor 196
  7.5.2 Programmable priority encoder 199
  7.5.3 Signed addition with status 201
  7.5.4 Combinational adder-based multiplier 203
  7.5.5 Hamming distance circuit 206
7.6 Synthesis guidelines 208
7.7 Bibliographic notes 208
Problems 208

### 8 Sequential Circuit Design: Principle 213
8.1 Overview of sequential circuits  
8.1.1 Sequential versus combinational circuits  
8.1.2 Basic memory elements  
8.1.3 Synchronous versus asynchronous circuits  
8.2 Synchronous circuits  
8.2.1 Basic model of a synchronous circuit  
8.2.2 Synchronous circuits and design automation  
8.2.3 Types of synchronous circuits  
8.3 Danger of synthesis that uses primitive gates  
8.4 Inference of basic memory elements  
8.4.1 D latch  
8.4.2 D FF  
8.4.3 Register  
8.4.4 RAM  
8.5 Simple design examples  
8.5.1 Other types of FFs  
8.5.2 Shift register  
8.5.3 Arbitrary-sequence counter  
8.5.4 Binary counter  
8.5.5 Decade counter  
8.5.6 Programmable mod-m counter  
8.6 Timing analysis of a synchronous sequential circuit  
8.6.1 Synchronized versus unsynchronized input  
8.6.2 Setup time violation and maximal clock rate  
8.6.3 Hold time violation  
8.6.4 Output-related timing considerations  
8.6.5 Input-related timing considerations  
8.7 Alternative one-segment coding style  
8.7.1 Examples of one-segment code  
8.7.2 Summary  
8.8 Use of variables in sequential circuit description  
8.9 Synthesis of sequential circuits  
8.10 Synthesis guidelines  
8.11 Bibliographic notes  
Problems  

9 Sequential Circuit Design: Practice  
9.1 Poor design practices and their remedies  
9.1.1 Misuse of asynchronous signals  
9.1.2 Misuse of gated clocks  
9.1.3 Misuse of derived clocks  
9.2 Counters
## 9.2 Counters and Registers

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2.1 Gray counter</td>
<td>265</td>
</tr>
<tr>
<td>9.2.2 Ring counter</td>
<td>266</td>
</tr>
<tr>
<td>9.2.3 LFSR (linear feedback shift register)</td>
<td>269</td>
</tr>
<tr>
<td>9.2.4 Decimal counter</td>
<td>272</td>
</tr>
<tr>
<td>9.2.5 Pulse width modulation circuit</td>
<td>275</td>
</tr>
</tbody>
</table>

## 9.3 Registers as temporary storage

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.3.1 Register file</td>
<td>276</td>
</tr>
<tr>
<td>9.3.2 Register-based synchronous FIFO buffer</td>
<td>279</td>
</tr>
<tr>
<td>9.3.3 Register-based content addressable memory</td>
<td>287</td>
</tr>
</tbody>
</table>

## 9.4 Pipelined design

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.4.1 Delay versus throughput</td>
<td>294</td>
</tr>
<tr>
<td>9.4.2 Overview on pipelined design</td>
<td>294</td>
</tr>
<tr>
<td>9.4.3 Adding pipeline to a combinational circuit</td>
<td>297</td>
</tr>
<tr>
<td>9.4.4 Synthesis of pipelined circuits and retiming</td>
<td>307</td>
</tr>
</tbody>
</table>

## 9.5 Synthesis guidelines

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>308</td>
</tr>
</tbody>
</table>

## 9.6 Bibliographic notes

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>309</td>
</tr>
</tbody>
</table>

## Problems

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>309</td>
</tr>
</tbody>
</table>

## 10 Finite State Machine: Principle and Practice

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1 Overview of FSMS</td>
<td>313</td>
</tr>
<tr>
<td>10.2 FSM representation</td>
<td>314</td>
</tr>
<tr>
<td>10.2.1 State diagram</td>
<td>315</td>
</tr>
<tr>
<td>10.2.2 ASM chart</td>
<td>317</td>
</tr>
<tr>
<td>10.3 Timing and performance of an FSM</td>
<td>321</td>
</tr>
<tr>
<td>10.3.1 Operation of a synchronous FSM</td>
<td>321</td>
</tr>
<tr>
<td>10.3.2 Performance of an FSM</td>
<td>324</td>
</tr>
<tr>
<td>10.3.3 Representative timing diagram</td>
<td>325</td>
</tr>
<tr>
<td>10.4 Moore machine versus Mealy machine</td>
<td>325</td>
</tr>
<tr>
<td>10.4.1 Edge detection circuit</td>
<td>326</td>
</tr>
<tr>
<td>10.4.2 Comparison of Moore output and Mealy output</td>
<td>328</td>
</tr>
<tr>
<td>10.5 VHDL description of an FSM</td>
<td>329</td>
</tr>
<tr>
<td>10.5.1 Multi-segment coding style</td>
<td>330</td>
</tr>
<tr>
<td>10.5.2 Two-segment coding style</td>
<td>333</td>
</tr>
<tr>
<td>10.5.3 Synchronous FSM initialization</td>
<td>335</td>
</tr>
<tr>
<td>10.5.4 One-segment coding style and its problem</td>
<td>336</td>
</tr>
<tr>
<td>10.5.5 Synthesis and optimization of FSM</td>
<td>337</td>
</tr>
<tr>
<td>10.6 State assignment</td>
<td>338</td>
</tr>
<tr>
<td>10.6.1 Overview of state assignment</td>
<td>338</td>
</tr>
<tr>
<td>10.6.2 State assignment in VHDL</td>
<td>339</td>
</tr>
<tr>
<td>10.6.3 Handling the unused states</td>
<td>341</td>
</tr>
<tr>
<td>10.7 Moore output buffering</td>
<td>342</td>
</tr>
<tr>
<td>10.7.1 Buffering by clever state assignment</td>
<td>342</td>
</tr>
</tbody>
</table>
10.7.2 Look-ahead output circuit for Moore output
10.8 FSM design examples
10.8.1 Edge detection circuit
10.8.2 Arbiter
10.8.3 DRAM strobe generation circuit
10.8.4 Manchester encoding circuit
10.8.5 FSM-based binary counter
10.9 Bibliographic notes
Problems

11 Register Transfer Methodology: Principle

11.1 Introduction
11.1.1 Algorithm
11.1.2 Structural data flow implementation
11.1.3 Register transfer methodology
11.2 Overview of FSMD
11.2.1 Basic RT operation
11.2.2 Multiple RT operations and data path
11.2.3 FSM as the control path
11.2.4 ASMD chart
11.2.5 Basic FSMD block diagram
11.3 FSMD design of a repetitive-addition multiplier
11.3.1 Converting an algorithm to an ASMD chart
11.3.2 Construction of the FSMD
11.3.3 Multi-segment VHDL description of an FSMD
11.3.4 Use of a register value in a decision box
11.3.5 Four- and two-segment VHDL descriptions of FSMD
11.3.6 One-segment coding style and its deficiency
11.4 Alternative design of a repetitive-addition multiplier
11.4.1 Resource sharing via FSMD
11.4.2 Mealy-controlled RT operations
11.5 Timing and performance analysis of FSMD
11.5.1 Maximal clock rate
11.5.2 Performance analysis
11.6 Sequential add-and-shift multiplier
11.6.1 Initial design
11.6.2 Refined design
11.6.3 Comparison of three ASMD designs
11.7 Synthesis of FSMD
11.8 Synthesis guidelines
11.9 Bibliographic notes
Problems
12 Register Transfer Methodology: Practice

12.1 Introduction
12.2 One-shot pulse generator
   12.2.1 FSM implementation
   12.2.2 Regular sequential circuit implementation
   12.2.3 Implementation using RT methodology
   12.2.4 Comparison
12.3 SRAM controller
   12.3.1 Overview of SRAM
   12.3.2 Block diagram of an SRAM controller
   12.3.3 Control path of an SRAM controller
12.4 GCD circuit
12.5 UART receiver
12.6 Square-root approximation circuit
12.7 High-level synthesis
12.8 Bibliographic notes
   Problems

13 Hierarchical Design in VHDL

13.1 Introduction
   13.1.1 Benefits of hierarchical design
   13.1.2 VHDL constructs for hierarchical design
13.2 Components
   13.2.1 Component declaration
   13.2.2 Component instantiation
   13.2.3 Caveats in component instantiation
13.3 Generics
13.4 Configuration
   13.4.1 Introduction
   13.4.2 Configuration declaration
   13.4.3 Configuration specification
   13.4.4 Component instantiation and configuration in VHDL 93
13.5 Other supporting constructs for a large system
   13.5.1 Library
   13.5.2 Subprogram
   13.5.3 Package
13.6 Partition
   13.6.1 Physical partition
   13.6.2 Logical partition
13.7 Synthesis guidelines
13.8 Bibliographic notes
15.2 Data types for two-dimensional signals
   15.2.1 Genuine two-dimensional data type
   15.2.2 Array-of-arrays data type
   15.2.3 Emulated two-dimensional array
   15.2.4 Example
   15.2.5 Summary
15.3 Commonly used intermediate-sized RT-level components
   15.3.1 Reduced-xor circuit
   15.3.2 Binary decoder
   15.3.3 Multiplexer
   15.3.4 Binary encoder
   15.3.5 Barrel shifter
15.4 More sophisticated examples
   15.4.1 Reduced-xor-vector circuit
   15.4.2 Multiplier
   15.4.3 Parameterized LFSR
   15.4.4 Priority encoder
   15.4.5 FIFO buffer
15.5 Synthesis of parameterized modules
15.6 Synthesis guidelines
15.7 Bibliographic notes
Problems

16 Clock and Synchronization: Principle and Practice
16.1 Overview of a clock distribution network
   16.1.1 Physical implementation of a clock distribution network
   16.1.2 Clock skew and its impact on synchronous design
16.2 Timing analysis with clock skew
   16.2.1 Effect on setup time and maximal clock rate
   16.2.2 Effect on hold time constraint
16.3 Overview of a multiple-clock system
   16.3.1 System with derived clock signals
   16.3.2 GALS system
16.4 Metastability and synchronization failure
   16.4.1 Nature of metastability
   16.4.2 Analysis of MTBF($T_r$)
   16.4.3 Unique characteristics of MTBF($T_r$)
16.5 Basic synchronizer
   16.5.1 The danger of no synchronizer
   16.5.2 One-FF synchronizer and its deficiency
   16.5.3 Two-FF synchronizer
   16.5.4 Three-FF synchronizer
CONTENTS

16.5.5 Proper use of a synchronizer 621

16.6 Single enable signal crossing clock domains 623
  16.6.1 Edge detection scheme 623
  16.6.2 Level-alternation scheme 627

16.7 Handshaking protocol 630
  16.7.1 Four-phase handshaking protocol 630
  16.7.2 Two-phase handshaking protocol 637

16.8 Data transfer crossing clock domains 639
  16.8.1 Four-phase handshaking protocol data transfer 641
  16.8.2 Two-phase handshaking data transfer 650
  16.8.3 One-phase data transfer 651

16.9 Data transfer via a memory buffer 652
  16.9.1 FIFO buffer 652
  16.9.2 Shared memory 660

16.10 Synthesis of a multiple-clock system 661

16.11 Synthesis guidelines 662
  16.11.1 Guidelines for general use of a clock 662
  16.11.2 Guidelines for a synchronizer 662
  16.11.3 Guidelines for an interface between clock domains 662

16.12 Bibliographic notes 663

Problems 663

References 665

Topic Index 667
With the maturity and availability of hardware description language (HDL) and synthesis software, using them to design custom digital hardware has become a mainstream practice. Because of the resemblance of an HDL code to a traditional program (such as a C program), some users believe incorrectly that designing hardware in HDL involves simply writing syntactically correct software code, and assume that the synthesis software can automatically derive the physical hardware. Unfortunately, synthesis software can only perform transformation and local optimization, and cannot convert a poor description into an efficient implementation. Without an understanding of the hardware architecture, the HDL code frequently leads to unnecessarily complex hardware, or may not even be synthesizable.

This book provides in-depth coverage on the systematical development and synthesis of efficient, portable and scalable register-transfer-level (RT-level) digital circuits using the VHDL hardware description language. RT-level design uses intermediate-sized components, such as adders, comparators, multiplexers and registers, to construct a digital system. It is the level that is most suitable and effective for today’s synthesis software.

RT-level design and VHDL are two somewhat independent subjects. VHDL code is simply one of the methods to describe a hardware design. The same design can also be described by a schematic or code in other HDLs. VHDL and synthesis software will not lead automatically to a better or worse design. However, they can shield designers from low-level details and allow them to explore and research better architectures.

The emphasis of the book is on hardware rather than language. Instead of treating synthesis software as a mysterious black box and listing “recipe-like” codes, we explain the relationship between the VHDL constructs and the underlying hardware structure and illustrate how to explore the design space and develop codes that can be synthesized into efficient cell-level implementation. The discussion is independent of technology and can
be applied to both ASIC and FPGA devices. The VHDL codes listed in the book largely follow the IEEE 1076.6 RTL synthesis standard and can be accepted by most synthesis software. Most codes can be synthesized without modification by the free “demo-version” synthesis software provided by FPGA vendors.

Scope The book focuses primarily on the design and synthesis of RT-level circuits. A subset of VHDL is used to describe the design. The book is not intended to be a comprehensive ASIC or FPGA book. All other issues, such as device architecture, placement and routing, simulation and testing, are discussed exclusively from the context of RT-level design.

Unique features The book is a hardware design text. VHDL and synthesis software are used as tools to realize the intended design. Several unique features distinguish the book:

- Suggest a coding style that shows a clear relationship between VHDL constructs and hardware components.
- Use easy-to-understand conceptual diagrams, rather than cell-level netlists, to explain the realization of VHDL codes.
- Emphasize the reuse aspect of the codes throughout the book.
- Consider RT-level design as an integral part of the overall development process and introduce good design practices and guidelines to ensure that an RT-level description can accommodate future simulation, verification and testing needs.
- Make the design “technology neutral” so that the developed VHDL code can be applied to both ASIC and FPGA devices.
- Follow the IEEE 1076.6 RTL synthesis standard to make the codes independent of synthesis software.
- Provide a set of synthesis guidelines at the end of each chapter.
- Contain a large number of non-trivial, practical examples to illustrate and reinforce the design concepts, procedures and techniques.
- Include two chapters on realizing sequential algorithms in hardware (known as “register transfer methodology”) and on designing control path and data path.
- Include two chapters on the scalable and parameterized designs and coding.
- Include a chapter on the synchronization and interface between multiple clock domains.

Book organization The book is basically divided into three major parts. The first part, Chapters 1 to 6, provides a comprehensive overview of VHDL and the synthesis process, and examines the hardware implementation of basic VHDL language constructs. The second part, Chapters 7 to 12, covers the core of the RT-level design, including combinational circuits, “regular” sequential circuits, finite state machine and circuits designed by register transfer methodology. The third part, Chapters 13 to 16, covers the system issues, including the hierarchy, parameterized and scalable design, and interface between clock domains. More detailed descriptions of the chapters follow.

- Chapter 1 presents a “big picture” of digital system design, including an overview on device technologies, system representation, development flow and software tools.
- Chapter 2 provides an overview on the design, usage and capability of a hardware description language. A series of simple codes is used to introduce the basic modeling concepts of VHDL.
- Chapter 3 provides an overview of the basic language constructs of VHDL, including lexical elements, objects, data types and operators. Because VHDL is a strongly typed language, the data types and operators are discussed in more detail.
Chapter 4 covers the syntax, usage and implementation of concurrent signal assignment statements of VHDL. It shows how to realize these constructs by multiplexing and priority routing networks.

Chapter 5 examines the syntax, usage and implementation of sequential statements of VHDL. It shows the realization of the sequential statements and discusses the caveats of using these statements.

Chapter 6 explains the realization of VHDL operators and data types, provides an in-depth overview on the synthesis process and discusses the timing issue involved in synthesis.

Chapter 7 covers the construction and VHDL description of more sophisticated combinational circuits. Examples show how to transform conceptual ideas into hardware, and illustrate resource-sharing and circuit-shaping techniques to reduce circuit size and increase performance.

Chapter 8 introduces the synchronous design methodology and the construction and coding of synchronous sequential circuits. Basic "regular" sequential circuits, such as counters and shift registers, in which state transitions exhibit a regular pattern, are examined.

Chapter 9 explores more sophisticated regular sequential circuits. The design examples show the implementation of a variety of counters, the use of registers as fast, temporary storage, and the construction of pipelined combinational circuits.

Chapter 10 covers finite state machine (FSM), which is a sequential circuit with "random" transition patterns. The representation, timing and implementation issues of FSMs are studied with an emphasis on its use as the control circuit for a large, complex system.

Chapter 11 introduces the register transfer methodology, which describes system operation by a sequence of data transfers and manipulations among registers, and demonstrates the construction of the data path (a regular sequential circuit) and the control path (an FSM) used in this methodology.

Chapter 12 uses a variety of design examples to illustrate how the register transfer methodology can be used in various types of problems and to highlight the design procedure and relevant issues.

Chapter 13 features the design hierarchy, in which a system is gradually divided into smaller parts. Mechanisms and language constructs of VHDL used to specify and configure a hierarchy are examined.

Chapter 14 introduces parameterized design, in which the width and functionality of a circuit are specified by explicit parameters. Simple examples illustrate the mechanisms used to pass and infer parameters and the language constructs used to describe the replicated structures.

Chapter 15 provides more sophisticated parameterized design examples. The main focus is on the derivation of efficient parameterized RT-level modules that can be used as building blocks of larger systems.

Chapter 16 covers the effect of a non-ideal clock signal and discusses the synchronization of an asynchronous signal and the interface between two independent clock domains.

Audience The intended audience for the book is students in advanced digital system design course and practicing engineers who wish to sharpen their design skills or to learn the effective use of today's synthesis software. Readers need to have basic knowledge of digital systems. The material is normally covered in an introductory digital design course,
which is a standard part in all electrical engineering and computer engineering curricula. No prior experience on HDL or synthesis is needed.

Verilog is another popular HDL. Since the book emphasizes hardware and methodology rather than language constructs, readers with prior Verilog experience can easily follow the discussion and learn VHDL along the way. Most VHDL codes can easily be translated into the Verilog language.

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

- Errata.
- Summary of coding guidelines.
- Code listing.
- Links to demo-version synthesis software.
- Links to some referenced materials.
- Frequently asked questions (FAQ) on RTL synthesis.
- Lecture slides for instructors.

**Errata** The book is "self-prepared," which means the author has prepared all materials, including the illustrations, tables, code listing, indexing and formatting, by himself. As the 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 2006*
ACKNOWLEDGMENTS

The author would like to express his gratitude to Professor George L. Kramerich for his encouragement and help during the course of this project. The work was partially supported by educational material development grant 0126752 from the National Science Foundation and a Teaching Enhancement grant from Cleveland State University.

P. P. Chu