To keep up with advances in semiconductor technology, design complexity and design methodology have to be improved. The traditional gate-level schematic capture approach cannot meet the shortened design cycle and market window. A higher level of abstraction to capture designs is not an option, but a requirement. For this reason Hardware Description Languages (HDL) such as Verilog and VHDL were developed. Synthesis tools take these HDLs and automatically translate them into schematics, thereby enabling designers to manage more complex and bigger designs in shorter design schedules.

VHDL stands for VHSIC (Very High Speed Integrated Circuits) Hardware Description Language. Designers have interpreted VHDL as meaning Very Hard Description Language (VHDL). The major goal of this book is to define VHDL as Very Handy Description Language for designing and modeling digital circuits.

This book is the result of the author’s practical experience in both design and teaching. Many of the design techniques and design considerations illustrated throughout the chapters are examples of real designs. The author’s teaching experience has led to a step-by-step presentation which addresses common mistakes and hard-to-understand concepts in a way that eases learning.

VHDL is introduced with practical design examples, simulation waveforms, and schematics so that readers can better understand their correspondence and relationships. Unique features of the book include the following:

1. There are more than 115 complete examples of 6200 lines of VHDL code in the book. Every line of VHDL code is analyzed, and most of them are simulated and synthesized with simulation waveforms and schematics shown. VHDL codes, simulation waveforms, and schematics are shown together, allowing readers to grasp the concepts easier and faster.
2. Every line of VHDL code has an associated line number for easy reference and discussion. The font used for the VHDL code portion is “Courier” because each character of this font occupies the same width, which allows them to line up vertically. This is close to how an ASCII file appears when readers type in their VHDL code with any text editor.
3. The VHDL code examples are carefully designed to illustrate various VHDL constructs, features, practical design considerations, and techniques. The examples are complete so that readers can assimilate overall ideas more easily.
4. Challenging exercises are provided at the end of each chapter so that readers can put into practice the ideas and information offered in the chapter.
5. A complete design project from concept to final timing verification is provided to demonstrate the entire design process.
6. All VHDL syntax and constructs are discussed with complete examples.
7. Practical design techniques for finite state machines, synthesis processes, and test benches are discussed with complete examples.
8. VHDL’93 important updates are discussed with complete examples.

BOOK ORGANIZATION

This book is divided into 14 chapters and 3 appendixes. All VHDL constructs are discussed in Chapters 1 through 7 and Chapter 10. Chapters 8 through 13 are dedicated to practical design examples. Chapter 14 summarizes VHDL’93 updates. Appendix A is a quick VHDL syntax reference with a complete example. Appendix B summarizes the declarations at each region. Appendix C is a VHDL’93 grammar and syntax reference. The details of each chapter are as follows:

Chapter 1 gives a quick overview of VHDL history and usage. The synthesis terminology is defined and illustrated with a small example. The comparisons of the traditional schematic design approach and the top-down design approach with VHDL and synthesis are discussed.

Chapter 2 summarizes VHDL basic elements such as lexical elements, identifiers, reserved words, operators, packages, entities, architectures, objects, types, and attributes.

Chapter 3 introduces the VHDL simulation concepts of the process concurrency, event driven simulation, delta delay, process sensitivity list, and signal. It also gives an overview where sequential statements and concurrent statements can appear.

Chapter 4 discusses all sequential statements and provides complete examples and applications.

Chapter 5 discusses all VHDL concurrent statements with complete examples and applications.

Chapter 6 presents VHDL subprograms and packages. The concepts of overloading and bus resolution functions are discussed. Practical examples are provided.

Chapter 7 discusses VHDL libraries, design units, and configurations. Complete examples are provided.

Chapter 8 focuses on the important issues of writing VHDL for synthesis. Many examples are presented to infer latched, flipflops, tristate buffers, and combinational circuits. The synthesis process is discussed and illustrated with an actual example from VHDL coding to simulating with the backannotated timing delay.

Chapter 9 illustrates writing VHDL for finite state machines. Complete examples, simulations, and synthesized schematics are provided.

Chapter 10 discusses more behavioral modeling of file text I/O, ROM, RAM, pad models, guarded block, guarded signal, disconnect, and null constructs.

Chapter 11 presents a small design case with techniques for implementing test benches.

Chapter 12 illustrates the design of an ALU (arithmetic logic unit) with VHDL code, simulation, and synthesis.
Chapter 13 describes a design case study, which allows the reader to go through the complete design process from the design concept, VHDL implementation, verification, test bench, layout, and post-layout verification.

Chapter 14 summarizes the important updates of VHDL'93. Many examples and actual simulations are provided.

**AUTHOR'S NOTE**

Readers who have more software than hardware experience usually have no problem writing another programming language. Their challenges are usually found in the following areas:

1. Process concurrency
2. Sequential statements regions and concurrent statements regions
3. Hardware implication—schematic and simulation waveform
4. Timing concepts
5. Mixing other program languages with VHDL, especially with regard to reserved words and delimiters

Readers who have more hardware than software experience find their challenges in the following areas:

1. VHDL syntax
2. Writing VHDL to imply hardware
3. Sequential statements regions and concurrent statements regions
4. Taking advantages of VHDL constructs
5. Software concepts such as subprograms, packages, libraries, and configurations.

In this book, the VHDL syntax is introduced with BNF (Backus Normal Form). BNF is referred as the production rule and has the format: `left_hand_side ::= right_hand_side`. The symbol `::=` (two semicolons and one equal sign) means “can be replaced.” Thus, `left_hand_side ::= right_hand_side` can be read as “the `left_hand_side` can be replaced with the `right_hand_side`.” The following are special notations used in the `right_hand_side`:

- `[ ... ]` Anything inside the square bracket is optional. It can appear as 0 or 1 time.
- `{ ... }` Anything inside the bracket can be repeated, 0 or any number of times.
- `... I ...` The vertical bar indicates the item to the left or right of the vertical bar can be used.
- `if` VHDL reserved words are printed in **bold face**.
- `___` Underscored portions indicate the VHDL’93 updates.
  Any other special character such as comma (,) and symbol (:=) would be part of the actual text in the syntax.

For example, the if statement can be defined as follows:
if_statement ::=  
   [ if_label : ] if condition then  
   sequence_of_statements  
   [ elsif condition then  
   sequence_of_statements ]  
   [ else  
   sequence_of_statements ]  
   end if [ if_label ];

The preceding if statement BNF shows reserved words if, then, else, elsif, end. It can have an optional label (with the colon :), the if condition then can appear only one time, followed by sequence of statements (that can be further replaced), the elsif clause can be repeated 0 or any number of times since it is inside the { } bracket, the else clause can appear 0 or 1 times since it is inside the [ ] square bracket. It is closed with the reserved words end if, followed by the optional label (the same label as in the beginning with no colon :), and ended with the semicolon ;.

The next BNF example shows that an instantiation list can have one or more labels (separated by commas), or the reserved word others, or the reserved word all, but not combinations of any two items from the three choices separated by the vertical bars.

instantiation_list ::= instantiation_label { , instantiation_label } I others I all

ACKNOWLEDGMENTS

This book would have been impossible without the following people: Ron Fernandes edited the entire first manuscript. Bill Cormier, Erik Oson, and Carl Wagner of Synopsys supported and provided technical assistance with Synopsys tools. Dear friends Hanchi and Clare Lin, Pinchung and Debra Chen, Johnny and Sunkoo To, Scott and Amy Weaver, Tony and Mayda Eng continuously encouraged me when I did not see the end of this adventure. John Cooley provided test vectors used in Chapter 11. My wife Tsai-Wei, and children Alan, Steven, and Jocelyn were patient and sacrificed many of our weekends.

The IEEE Computer Society Press editorial team, Bill Sanders, Mohamed Fayad, Cheryl Smith, and Lisa O’Conner contributed many valuable comments. Additionally, Joni Harlan from Bookmark Media did a skillful and thorough editing of the manuscript.

Ross Barta, Gary Nelson, Warren Snapp, and Mike White gave management support.

The author deeply appreciates all of these people.