The Designer’s Guide to VHDL
Third Edition
The Designer’s Guide
to VHDL
Third Edition

Peter J. Ashenden
EDA CONSULTANT, ASHENDEN DESIGNS PTY. LTD.
ADJUNCT ASSOCIATE PROFESSOR, ADELAIDE UNIVERSITY
To my wife Katrina
Contents

Preface xvi

1 Fundamental Concepts 1
1.1 Modeling Digital Systems 1
1.2 Domains and Levels of Modeling 3
    1.2.1 Modeling Example 3
1.3 Modeling Languages 7
1.4 VHDL Modeling Concepts 7
    1.4.1 Elements of Behavior 8
    1.4.2 Elements of Structure 10
    1.4.3 Mixed Structural and Behavioral Models 12
    1.4.4 Test Benches 13
    1.4.5 Analysis, Elaboration and Execution 14
1.5 Learning a New Language: Lexical Elements and Syntax 16
    1.5.1 Lexical Elements 17
        Comments 17
        Identifiers 19
        Reserved Words 20
        Special Symbols 22
        Numbers 22
        Characters 23
        Strings 23
        Bit Strings 24
    1.5.2 Syntax Descriptions 26
Exercises 29

2 Scalar Data Types and Operations 31
2.1 Constants and Variables 31
    2.1.1 Constant and Variable Declarations 31
    2.1.2 Variable Assignment 33
2.2 Scalar Types 34
    2.2.1 Type Declarations 34
    2.2.2 Integer Types 35
    2.2.3 Floating-Point Types 38
    2.2.4 Physical Types 39
        Time 42
    2.2.5 Enumeration Types 43
        Characters 44
        Booleans 46
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Case Study: A Pipelined Multiplier Accumulator</td>
<td>337</td>
</tr>
<tr>
<td>10.1</td>
<td>Algorithm Outline</td>
<td>337</td>
</tr>
<tr>
<td>10.2</td>
<td>A Behavioral Model</td>
<td>340</td>
</tr>
<tr>
<td>10.2.1</td>
<td>Testing the Behavioral Model</td>
<td>342</td>
</tr>
<tr>
<td>10.3</td>
<td>A Register-Transfer-Level Model</td>
<td>346</td>
</tr>
<tr>
<td>10.3.1</td>
<td>Testing the Register-Transfer-Level Model</td>
<td>350</td>
</tr>
<tr>
<td></td>
<td>Exercises</td>
<td>353</td>
</tr>
<tr>
<td>11</td>
<td>Aliases</td>
<td>355</td>
</tr>
<tr>
<td>11.1</td>
<td>Aliases for Data Objects</td>
<td>355</td>
</tr>
<tr>
<td>11.2</td>
<td>Aliases for Non-Data Items</td>
<td>360</td>
</tr>
<tr>
<td></td>
<td>Exercises</td>
<td>363</td>
</tr>
<tr>
<td>12</td>
<td>Generics</td>
<td>365</td>
</tr>
<tr>
<td>12.1</td>
<td>Generic Constants</td>
<td>365</td>
</tr>
<tr>
<td>12.2</td>
<td>Generic Types</td>
<td>372</td>
</tr>
<tr>
<td>12.3</td>
<td>Generic Lists in Packages</td>
<td>376</td>
</tr>
<tr>
<td>12.3.1</td>
<td>Local Packages</td>
<td>381</td>
</tr>
<tr>
<td>12.3.2</td>
<td>Abstract Data Types Using Packages</td>
<td>384</td>
</tr>
<tr>
<td>12.4</td>
<td>Generic Lists in Subprograms</td>
<td>389</td>
</tr>
<tr>
<td>12.5</td>
<td>Generic Subprograms</td>
<td>394</td>
</tr>
<tr>
<td>12.6</td>
<td>Generic Packages</td>
<td>407</td>
</tr>
<tr>
<td></td>
<td>Exercises</td>
<td>414</td>
</tr>
<tr>
<td>13</td>
<td>Components and Configurations</td>
<td>417</td>
</tr>
<tr>
<td>13.1</td>
<td>Components</td>
<td>417</td>
</tr>
<tr>
<td>13.1.1</td>
<td>Component Declarations</td>
<td>417</td>
</tr>
<tr>
<td>13.1.2</td>
<td>Component Instantiation</td>
<td>419</td>
</tr>
<tr>
<td>13.1.3</td>
<td>Packaging Components</td>
<td>420</td>
</tr>
<tr>
<td>13.2</td>
<td>Configuring Component Instances</td>
<td>422</td>
</tr>
<tr>
<td>13.2.1</td>
<td>Basic Configuration Declarations</td>
<td>422</td>
</tr>
<tr>
<td>13.2.2</td>
<td>Configuring Multiple Levels of Hierarchy</td>
<td>425</td>
</tr>
<tr>
<td>13.2.3</td>
<td>Direct Instantiation of Configured Entities</td>
<td>428</td>
</tr>
<tr>
<td>13.2.4</td>
<td>Generic and Port Maps in Configurations</td>
<td>429</td>
</tr>
<tr>
<td>13.2.5</td>
<td>Deferred Component Binding</td>
<td>435</td>
</tr>
<tr>
<td>13.3</td>
<td>Configuration Specifications</td>
<td>437</td>
</tr>
<tr>
<td>13.3.1</td>
<td>Incremental Binding</td>
<td>438</td>
</tr>
<tr>
<td></td>
<td>Exercises</td>
<td>444</td>
</tr>
<tr>
<td>14</td>
<td>Generate Statements</td>
<td>449</td>
</tr>
<tr>
<td>14.1</td>
<td>Generating Iterative Structures</td>
<td>449</td>
</tr>
<tr>
<td>14.2</td>
<td>Conditionally Generating Structures</td>
<td>455</td>
</tr>
<tr>
<td>14.2.1</td>
<td>Recursive Structures</td>
<td>462</td>
</tr>
<tr>
<td>14.3</td>
<td>Configuration of Generate Statements</td>
<td>465</td>
</tr>
<tr>
<td></td>
<td>Exercises</td>
<td>473</td>
</tr>
</tbody>
</table>
15 Access Types 479
15.1 Access Types 479
   15.1.1 Access Type Declarations and Allocators 479
   15.1.2 Assignment and Equality of Access Values 482
   15.1.3 Access Types for Records and Arrays 483
15.2 Linked Data Structures 486
   15.2.1 Deallocation and Storage Management 490
15.3 An Ordered-Dictionary ADT Using Access Types 491
Exercises 495

16 Files and Input/Output 499
16.1 Files 499
   16.1.1 File Declarations 499
   16.1.2 Reading from Files 501
   16.1.3 Writing to Files 504
   16.1.4 Files Declared in Subprograms 507
   16.1.5 Explicit Open and Close Operations 509
   16.1.6 File Parameters in Subprograms 512
   16.1.7 Portability of Files 514
16.2 The Package Textio 514
   16.2.1 Textio Read Operations 518
   16.2.2 Textio Write Operations 523
   16.2.3 Reading and Writing Other Types 527
   Standard Package Read and Write Operations 528
Exercises 530

17 Case Study: A Package for Memories 535
17.1 The Memories Package 535
17.2 Using the Memories Package 546
   17.2.1 Common Address and Data Conversions 551
Exercises 558

18 Test Bench and Verification Features 559
18.1 External Names 559
18.2 Force and Release Assignments 570
18.3 Embedded PSL in VHDL 575
Exercises 582

19 Shared Variables and Protected Types 585
19.1 Shared Variables and Mutual Exclusion 585
19.2 Uninstantiated Methods in Protected Types 597
Exercises 601

20 Attributes and Groups 603
20.1 Predefined Attributes 603
   20.1.1 Attributes of Scalar Types 603
20.1.2 Attributes of Array Types and Objects 604
20.1.3 Attributes Giving Types 605
20.1.4 Attributes of Signals 606
20.1.5 Attributes of Named Items 607
20.2 User-Defined Attributes 616
  20.2.1 Attribute Declarations 616
  20.2.2 Attribute Specifications 616
20.3 Groups 628
Exercises 630

21 Design for Synthesis 633
  21.1 Synthesizable Subsets 633
  21.2 Use of Data Types 634
    21.2.1 Scalar Types 635
    21.2.2 Composite and Other Types 636
  21.3 Interpretation of Standard Logic Values 637
  21.4 Modeling Combinational Logic 638
  21.5 Modeling Sequential Logic 641
    21.5.1 Modeling Edge-Triggered Logic 642
    21.5.2 Level-Sensitive Logic and Inferring Storage 650
    21.5.3 Modeling State Machines 652
  21.6 Modeling Memories 654
  21.7 Synthesis Attributes 658
  21.8 Metacomments 666
Exercises 667

22 Case Study: System Design Using the Gumnut Core 669
  22.1 Overview of the Gumnut 669
    22.1.1 Instruction Set Architecture 669
    22.1.2 External Interface 674
      The Gumnut Entity Declaration 676
      Instruction and Data Memories 677
  22.2 A Behavioral Model 681
    22.2.1 The Gumnut Definitions Package 681
    22.2.2 The Gumnut Behavioral Architecture Body 687
      Overview of the Interpreter 690
      Resetting the Interpreter 691
      Acknowledging an Interrupt 691
      Fetching an Instruction 692
      Performing an Arithmetic/Logical Operation 693
      Performing a Shift Operation 694
      Performing a Memory-I/O Instruction 695
      Performing a Branch Instruction 697
      Performing a Jump Instruction 697
      Performing a Miscellaneous Instruction 698
    22.2.3 Verifying the Behavioral Model 699
  22.3 A Register-Transfer-Level Model 704
22.3.1 The Architecture Body 706
22.3.2 Verifying the RTL Model 720
22.4 A Digital Alarm Clock 721
  22.4.1 System Design 722
  22.4.2 Synthesizing and Implementing the Alarm Clock 729
Exercises 731

23 Miscellaneous Topics 733
  23.1 Guards and Blocks 733
    23.1.1 Guarded Signals and Disconnection 733
       The Driving Attribute 737
       Guarded Ports 738
       Guarded Signal Parameters 739
    23.1.2 Blocks and Guarded Signal Assignment 739
       Explicit Guard Signals 742
       Disconnection Specifications 743
    23.1.3 Using Blocks for Structural Modularity 744
       External Names and Blocks 747
       Generics and Ports in Blocks 748
       Configuring Designs with Blocks 748
  23.2 IP Encryption 750
    23.2.1 Key Exchange 769
  23.3 VHDL Procedural Interface (VHPI) 770
    23.3.1 Direct Binding 771
    23.3.2 Tabular Registration and Indirect Binding 773
    23.3.3 Registration of Applications and Libraries 775
  23.4 Postponed Processes 776
  23.5 Conversion Functions in Association Lists 779
  23.6 Linkage Ports 785
Exercises 786

A Standard Packages 793
  A.1 The Predefined Package standard 793
  A.2 The Predefined Package env 797
  A.3 The Predefined Package textio 797
  A.4 Standard VHDL Mathematical Packages 799
    A.4.1 The math_real Package 799
    A.4.2 The math_complex Package 801
  A.5 The std_logic_1164 Multivalue Logic System Package 802
  A.6 Standard Integer Numeric Packages 806
    A.6.1 The numeric_bit Package 806
    A.6.2 The numeric_std Package 812
    A.6.3 The numeric_bit_unsigned Package 813
    A.6.4 The numeric_std_unsigned Package 815
  A.7 Standard Fixed-Point Packages 816
    A.7.1 The fixed_float_types Package 816
    A.7.2 The fixed_generic_pkg Package 816
A.7.3 The fixed_pkg Package 829
A.8 Standard Floating-Point Packages 829
   A.8.1 The float_generic_pkg Package 829
   A.8.2 The float_pkg Package 840

B VHDL Syntax 841
   B.1 Design File 843
   B.2 Library Unit Declarations 843
   B.3 Declarations and Specifications 845
   B.4 Type Definitions 848
   B.5 Concurrent Statements 850
   B.6 Sequential Statements 852
   B.7 Interfaces and Associations 855
   B.8 Expressions and Names 856

C Answers to Exercises 859

References 889

Index 891
Preface

VHDL is a language for describing digital electronic systems. It arose out of the United States government’s Very High Speed Integrated Circuits (VHSIC) program. In the course of this program, it became clear that there was a need for a standard language for describing the structure and function of integrated circuits (ICs). Hence the VHSIC Hardware Description Language (VHDL) was developed. It was subsequently developed further under the auspices of the Institute of Electrical and Electronic Engineers (IEEE) and adopted in the form of the IEEE Standard 1076, Standard VHDL Language Reference Manual, in 1987. This first standard version of the language is often referred to as VHDL-87.

Like all IEEE standards, the VHDL standard is subject to review from time to time. Comments and suggestions from users of the 1987 standard were analyzed by the IEEE working group responsible for VHDL, and in 1992 a revised version of the standard was proposed. This was eventually adopted in 1993, giving us VHDL-93. A second round of revision of the standard was started in 1998. That process was completed in 2001, giving us VHDL-2002. After that, further development took place in the IEEE working group and in a technical committee of an organization, Accellera, whose charter is to promote standards for electronics design. These efforts led to the current version of the language, VHDL-2008, described in this book.

VHDL is designed to fill a number of needs in the design process. First, it allows description of the structure of a system, that is, how it is decomposed into subsystems and how those subsystems are interconnected. Second, it allows the specification of the function of a system using familiar programming language forms. Third, as a result, it allows the design of a system to be simulated before being manufactured, so that designers can quickly compare alternatives and test for correctness without the delay and expense of hardware prototyping. Fourth, it allows the detailed structure of a design to be synthesized from a more abstract specification, allowing designers to concentrate on more strategic design decisions and reducing time to market.

This book presents a structured guide to the modeling facilities offered by the VHDL language, showing how they can be used for the design of digital systems. The book does not purport to teach digital design, since that topic is large enough by itself to warrant several textbooks covering its various aspects. Instead, the book assumes that the reader has at least a basic grasp of digital design concepts, such as might be gained from a first course in digital design in an engineering degree program. Some exposure to computer programming and to concepts of computer organization will also be beneficial. This book is suitable for use in a course in digital or computer design and will also serve practicing engineers who need to acquire VHDL fluency as part of their changing job requirements.

One pervasive theme running through the presentation in this book is that modeling a system using a hardware description language is essentially a software design exercise. This implies that good software engineering practice should be applied. Hence the treatment in this book draws directly from experience in software engineering. There are nu-
merous hints and techniques from small-scale and large-scale software engineering presented throughout the book, with the sincere intention that they might be of use to readers.

I am particularly pleased to be able to include this book in the Morgan Kaufmann Series in Systems on Silicon. Modeling for simulation and synthesis is a vital part of a design methodology for large-scale systems. VHDL allows models to be expressed at a range of levels of abstraction, from gate-level up to algorithmic and architectural levels. It will continue to play an important role in the design of silicon-based systems for some time to come.

Structure of the Book

The Designer's Guide to VHDL is organized so that it can be read linearly from front to back. This path offers a graduated development, with each chapter building on ideas introduced in the preceding chapters. Each chapter introduces a number of related concepts or language facilities and illustrates each one with examples. Scattered throughout the book are three case studies, which bring together preceding material in the form of extended worked examples.

Chapter 1 introduces the idea of a hardware description language and outlines the reasons for its use and the benefits that ensue. It then proceeds to introduce the basic concepts underlying VHDL, so that they can serve as a basis for examples in subsequent chapters. The next three chapters cover the aspects of VHDL that are most like conventional programming languages. These may be used to describe the behavior of a system in algorithmic terms. Chapter 2 explains the basic type system of the language and introduces the scalar data types. Chapter 3 describes the sequential control structures, and Chapter 4 covers composite data structures used to represent collections of data elements. In Chapter 5, the main facilities of VHDL used for modeling hardware are covered in detail. These include facilities for modeling the basic behavioral elements in a design, the signals that interconnect them and the hierarchical structure of the design.

The next group of chapters extends this basic set of facilities with language features that make modeling of large systems more tractable. Chapter 6 introduces procedures and functions, which can be used to encapsulate behavioral aspects of a design. Chapter 7 introduces the package as a means of collecting together related parts of a design or of creating modules that can be reused in a number of designs. Chapter 8 deals with the important topic of resolved signals, and Chapter 9 describes a number of predefined and standard packages for use in VHDL designs. The combination of facilities described in these early chapters is sufficient for many modeling tasks, so Chapter 10 brings them together in the first case study, in which a multiplier/accumulator circuit is designed.

The third group of chapters covers advanced modeling features in VHDL. Chapter 11 covers aliases as a way of managing the large number of names that arise in a large model. Chapter 12 describes generics as a means of parameterizing the behavior and structure of a design and enhancing the resusability of designs. This leads to a discussion of abstract data types as a means of managing the complexity associated with large designs. Chapter 13 deals with the topics of component instantiation and configuration. These features are important in large real-world models, but they can be difficult to understand. Hence this book introduces structural modeling through the mechanism of direct instantiation in ear-
lier chapters and leaves the more general case of component instantiation and configuration until this later chapter. In Chapter 14, generated regular structures are covered.

The fourth group of chapters covers language facilities generally used for system-level modeling. Chapter 15 introduces the notion of access types (or pointers) and uses them to develop linked data structures. The topic of abstract data types is revisited in the context of container data types. Chapter 16 covers the language facilities for input and output using files, including binary files and text files. Chapter 17 is a case study in which a package for designing memories is developed. The package draws upon features described in the third and fourth groups of chapters.

In the fifth group of chapters, we introduce language features for advanced design and verification. Chapter 18 deals with features for test bench support and verification. It describes how specifications written in the IEEE standard Property Specification Language (PSL) can be embedded in VHDL models. Chapter 19 covers protected types and their use as a means of concurrency control. Chapter 20 describes how we can annotate items in a design with attributes to specify information to be used by design automation tools. This leads into Chapter 21, which covers guidelines for writing synthesizable models. This group of chapters is drawn together in a further case study, Chapter 22, showing development of a synthesizable processor core and its use in a small embedded system, a digital alarm clock.

The final chapter, Chapter 23, is a miscellany of advanced topics not covered in the previous chapters. It includes a discussion of blocks and guarded signals, which are not as widely used in modern designs as previously. Nonetheless, we describe them here for completeness. The chapter also covers use of features for encrypting the source text of models as a means of protecting intellectual property (IP), and use of features of the VHDL Procedureall Interface (VHPI) for incorporating models and applications written in non-VHDL programming languages.

Each chapter in the book is followed by a set of exercises designed to help the reader develop understanding of the material. Where an exercise relates to a particular topic described in the chapter, the section number is included in square brackets. An approximate “difficulty” rating is also provided, expressed using the following symbols:

- **➊** quiz-style exercise, testing basic understanding
- **➋** basic modeling exercise—10 minutes to half an hour effort
- **➌** advanced modeling exercise—one half to two hours effort
- **➍** modeling project—half a day or more effort

Answers for the first category of exercises are provided in Appendix C. The remaining categories involve developing VHDL models. Readers are encouraged to test correctness of their models by running them on a VHDL simulator. This is a much more effective learning exercise than comparing paper models with paper solutions.

**Changes in the Second and Third Editions**

The first edition of this book was published in 1995, just as VHDL-93 was gaining acceptance. The second edition was updated to reflect the changes in VHDL-2002. Many of the
changes in the language standard corrected ambiguities in the previous standard that caused incompatibility between VHDL tools from different vendors. There were also changes that enhanced the usability of the language. The text and examples in the second edition were revised where necessary to reflect the changes in the language. Furthermore, following publication of the first edition, a number of VHDL-related standards were published and gained widespread acceptance. The second edition added descriptions of the IEEE 1076.3 synthesis and IEEE 1076.2 math packages, and was revised to cover the IEEE 1076.6 Synthesis Interoperability Standard.

The latest revision of the language, VHDL-2008, adds a number of significant new language features, making this edition of The Designer’s Guide to VHDL significantly bigger than its predecessors. VHDL-2008 also specifies numerous minor new features and changes to existing features to enhance the usability of the language. This edition integrates descriptions of all of the new and revised features into the text. The differences between the various versions are highlighted in call-outs within the text, headed with “VHDL-2002,” “VHDL-93,” or “VHDL-87,” as appropriate. In addition, some of the material has been removed or rearranged. The case study on a package for arithmetic on bit-vector operands has been deleted because the standard numeric packages have now become widespread. The first case study in this book is a revised version of the MAC case study in previous editions, and shows how the standard packages can be used. The chapter on blocks and guarded signals has been contracted and moved to a section in the last chapter, since the features are now little used in practice. There is a greater emphasis on synthesis in this edition. What was an appendix on the topic in previous editions has been substantially revised and promoted to full chapter status. The large case study showing development of a 32-bit processor model has been revised to show a smaller synthesizable model of an 8-bit microcontroller core and its use in an embedded system. This is much more relevant, both for educational purposes and professional practice. Finally, this edition includes a listing of all of the VHDL standard packages as an appendix for reference.

**Resources for Help and Information**

Although this book attempts to be comprehensive in its coverage of VHDL, there will no doubt be questions that it does not answer. For these, the reader will need to seek other resources. A valuable source of experience and advice, often overlooked, is one’s colleagues, either at the workplace or in user groups. User groups generally hold regular meetings that either formally or informally include a time for questions and answers. Many also run e-mail lists and on-line discussion groups for problem solving.

Accellera is one of a number of organizations that sponsors the EDA Industry Working Groups Web server (www.eda.org). The server has links to Web pages and repositories of several VHDL standards groups and user groups.

Readers who have access to the Usenet electronic news network will find the news group comp.lang.vhdl a valuable resource. This discussion group is a source of announcements, sample models, questions and answers and useful software. Participants include VHDL users and people actively involved in the language standard working group and in VHDL tool development. The “frequently asked questions” (FAQ) file for this group is a mine of useful pointers to books, products and other information. It is archived at www.eda.org.
One resource that must be mentioned is IEEE Standard 1076, IEEE Standard VHDL Language Reference Manual, sometimes referred to as the “VHDL Bible.” It is the authoritative source of information about VHDL. However, since it is a definitional document, not a tutorial, it is written in a complex legalistic style. This makes it very difficult to use to answer the usual questions that arise when writing VHDL models. It should only be used once you are somewhat familiar with VHDL. It can be ordered from the IEEE at standards.ieee.org.

This book contains numerous examples of VHDL models that may also serve as a resource for resolving questions. The VHDL source code for these examples and the case studies, as well as other related information, is available on the companion website for the book at books.elsevier.com/companions/9780120887859.

Although I have been careful to avoid errors in the example code, there are no doubt some that I have missed. I would be pleased to hear about them, so that I can correct them in the on-line code and in future printings of this book. Errata and general comments can be e-mailed to me at vhdl-book@ashenden.com.au.

Acknowledgments

The seeds for this book go back to 1990 when I developed a brief set of notes, The VHDL Cookbook, for my computer architecture class at the University of Adelaide. At the time, there were few books on VHDL available, so I made my booklet available for on-line access. News of its availability spread quickly around the world, and within days, my e-mail in-box was bursting. At the time of writing this, nearly 20 years later, I still regularly receive messages about the Cookbook. Many of the respondents urged me to write a full textbook version. With that encouragement, I embarked upon the exercise that led to the first edition of The Designer’s Guide to VHDL. Two years after publication of The Designer’s Guide, the need for a book specifically for students became evident. That led to publication of the first edition of The Student’s Guide to VHDL. I am grateful to the many engineers, students and teachers around the world who gave me the impetus to write these books and who made them such a success. I hope this new edition will continue to meet the need for a comprehensive guide to VHDL.

In the previous editions of The Designer’s Guide and The Student’s Guide, I had the opportunity to extend thanks to the many people who assisted in development of the books. They included my colleagues at the University of Adelaide; my research collaborators, Phil Wilsey at the University of Cincinnati and Perry Alexander at the University of Kansas; the staff at Morgan Kaufmann Publishers, including, in particular, Denise Penrose; the reviewers of the manuscript for the first edition, namely, Poras Balsara of the University of Texas, Paul Menchini of Menchini & Associates, David Pitts of GTE Labs and the University of Lowell and Philip Wilsey of the University of Cincinnati; David Bishop for his contribution to the material on synthesis in the first edition of The Designer’s Guide; and Mentor Graphics Corporation, for use of their ModelSim simulator to check the example models. I remain grateful to all of these people and organizations for their valuable contributions to the earlier editions and to this edition.

For the current edition, I would also like to thank Jim Lewis, who collaborated on a recent book, VHDL-2008: Just the New Stuff. Much of the material from that book has found its way into this book in one form or another. Thanks also to Mentor Graphics Cor-
poration for continued use of the ModelSim simulator to check the example code. I con-
tinue to enjoy an excellent working relationship with the staff at Morgan Kaufmann
Publishers and their parent company, Elsevier. Thanks to Chuck Glaser, Senior Acquisi-
tions Editor, for his support in the continued development of these VHDL books; to Dawn-
marie Simpson, Senior Project Manager in the Production Department, for her meticulous
attention to detail; and to Denise Penrose, Publisher, for her longstanding support of my
writing endeavors.

The previous editions of *The Designer’s Guide to VHDL* were dedicated to my wife
Katrina. As I said in the first edition preface, I used to think that authors dedicating their
books to their partners was somewhat contrived, but that Katrina’s understanding, encour-
agement and support taught me otherwise. I remain deeply grateful for her continued sup-
port and am honored to also dedicate this third edition to her.