effective coding with vhdl

effective coding with vhdl is essential for designing reliable, maintainable, and efficient digital hardware systems. VHDL, or VHSIC Hardware Description Language, is widely used in the development of complex digital circuits, including FPGA and ASIC designs. Mastering effective coding with VHDL involves understanding its syntax, best practices, and design methodologies to optimize simulation, synthesis, and implementation. This article explores key techniques for writing clear and efficient VHDL code, strategies for debugging and testing, and approaches for improving readability and modularity. Additionally, it covers common pitfalls to avoid and practical tips for leveraging VHDL's powerful features. The following sections provide a comprehensive overview to help engineers and developers enhance their VHDL coding skills and produce high-quality hardware descriptions.

Understanding VHDL Fundamentals

Before diving into effective coding with VHDL, it is crucial to establish a solid understanding of VHDL fundamentals. VHDL is a strongly typed, concurrent hardware description language that allows designers to model electronic systems at various levels of abstraction. The language supports behavioral, structural, and dataflow modeling, making it versatile for different stages of hardware development.

VHDL Syntax and Structure

The syntax of VHDL is similar to Ada programming language, emphasizing strong typing and clear declaration of entities, architectures, signals, and processes. A typical VHDL design unit consists of an entity declaration that defines the interface and an architecture body that describes the internal behavior or structure. Understanding these building blocks is essential for writing effective VHDL code.

Data Types and Operators

VHDL includes a wide range of predefined data types such as std_logic, std_logic_vector, integer, and boolean. Effective coding with VHDL requires selecting appropriate data types to ensure precision and synthesis compatibility. Additionally, VHDL supports various operators for arithmetic, logical, and relational operations that must be used correctly to avoid synthesis mismatches.

Best Practices for Writing Efficient VHDL Code

Writing efficient VHDL code is a critical aspect of effective coding with VHDL. Efficient code not only improves simulation speed but also enhances the quality of the synthesized hardware. Following best practices helps maintain clarity, reduces errors, and facilitates easier maintenance and scalability.

Use Clear and Consistent Naming Conventions

Clear naming conventions for signals, variables, and entities improve code readability and reduce ambiguity. Using descriptive names that reflect the function or purpose of a component helps other engineers understand the design quickly.

Commenting and Documentation

Comprehensive comments and documentation are vital in complex VHDL projects. Comments should explain the intent of code sections, describe algorithms, and clarify non-obvious design decisions. This practice supports long-term maintenance and collaboration.

Leverage Concurrent and Sequential Statements Appropriately

VHDL allows concurrent execution of multiple processes, but sequential statements inside processes must be used judiciously. Effective coding with VHDL involves balancing these constructs to model hardware behavior accurately and efficiently.

Use Constants and Generics

Constants and generics promote code reuse and parameterization, enabling flexible designs that can be easily adapted to different configurations without rewriting code.

Modular Design and Code Organization

Modularity is a key principle in effective coding with VHDL, facilitating manageable, scalable, and reusable hardware descriptions. Organizing code into smaller, well-defined modules simplifies testing, debugging, and future modifications.

Entity and Architecture Separation

Separating interface definitions from implementation details by properly using entities and architectures enhances modularity and allows multiple architectures for a single entity to be developed and tested independently.

Hierarchical Design Approach

Adopting a hierarchical design methodology helps break complex systems into submodules or components. Each submodule can be developed and verified independently before integration, reducing complexity and improving reliability.

Use Packages for Reusable Components

VHDL packages enable grouping of related types, constants, functions, and procedures, promoting code reuse and consistency across multiple design units.

- Define common data types and subprograms in packages
- Import packages where needed using the 'use' clause
- Maintain packages to keep shared code organized

Simulation and Debugging Techniques

Simulation and debugging are integral parts of effective coding with VHDL, ensuring correct functionality before hardware implementation. Utilizing advanced simulation tools and systematic debugging approaches enhances design quality and reduces development time.

Writing Testbenches

Testbenches simulate the behavior of VHDL designs under various input conditions. Effective testbenches include stimulus generation, response monitoring, and result checking. They are essential for verifying logical correctness and timing behavior.

Using Assertions and Reports

Assertions provide runtime checks within VHDL code to validate assumptions and detect errors early during simulation. Reports complement assertions by providing informative messages that assist in debugging.

Waveform Analysis

Waveform viewers help visualize signal transitions and timing relationships, allowing designers to identify unexpected behavior or timing violations in their VHDL models.

Optimization Strategies for Synthesis

Effective coding with VHDL also demands attention to synthesis optimization to achieve efficient hardware utilization, performance, and power consumption. Understanding how synthesis tools interpret VHDL code is crucial for writing synthesizable and optimized designs.

Writing Synthesizable VHDL

Not all VHDL constructs are synthesizable. Designers must use synthesis-friendly coding styles, such as avoiding infinite loops, using clocked processes for sequential logic, and ensuring proper signal assignments.

Resource Sharing and Pipelining

Optimization techniques like resource sharing reduce hardware area by reusing functional units, while pipelining improves throughput and clock frequency by breaking combinational paths into stages.

Using Attributes and Pragmas

Many synthesis tools support attributes and pragmas that guide optimization, such as specifying timing constraints, resource allocation preferences, or preserving certain logic structures.

Common Pitfalls and How to Avoid Them

Awareness of common mistakes is essential for effective coding with VHDL. Avoiding these pitfalls can save significant debugging and rework time during development.

Mixing Signal and Variable Assignments

Confusing signals and variables, especially in processes, can lead to simulation and synthesis mismatches. Signals have scheduled updates, while variables update immediately, impacting behavior and timing.

Improper Use of Clocked Processes

Incorrectly coding clocked processes, such as missing edge detection or asynchronous resets, can cause synthesis errors or unintended hardware behavior.

Ignoring Timing Constraints

Failing to specify or meet timing constraints often results in designs that do not operate correctly at the target frequency. Proper timing analysis and constraint definition are vital.

Overcomplicating Designs

Writing unnecessarily complex code reduces readability and increases the chance of errors. Striving for simplicity and clarity enhances maintainability and optimization.

1. Use signals and variables appropriately and understand their differences
2. Verify clocking and reset logic thoroughly
3. Define and respect timing constraints in the synthesis tool
4. Keep designs modular and simple

Frequently Asked Questions

What are the best practices for writing effective VHDL code?

Best practices for writing effective VHDL code include using meaningful signal and entity names, consistent indentation and formatting, modular design with reusable components, thorough commenting, and adhering to coding standards to improve readability and maintainability.

How can I improve simulation performance when coding in VHDL?

To improve simulation performance, avoid unnecessary signal assignments, use appropriate data types, minimize the use of wait statements, and leverage synthesis directives to focus on critical parts of the design during simulation.

What techniques help in debugging VHDL code effectively?

Effective debugging techniques include using signal assertions, inserting testbenches with comprehensive test cases, utilizing waveform viewers, and employing VHDL’s report statements to track internal signal values during simulation.

How important is code modularity in VHDL and how can it be achieved?

Code modularity is crucial for scalability and reusability in VHDL. It can be achieved by designing small, well-defined components or entities, using packages for shared types and functions, and separating architecture and behavioral code cleanly.

What role do packages play in effective VHDL coding?

Packages in VHDL help organize and reuse code by grouping related type definitions, constants, functions, and procedures. This promotes code modularity, reduces redundancy, and makes maintenance easier.

How can I write synthesizable VHDL code that is also easy to maintain?

Writing synthesizable and maintainable VHDL code involves following synthesis guidelines (e.g., avoiding latches, using synchronous resets), maintaining clear structure, using constants and generics for flexibility, and documenting design intent through comments.

What are some common pitfalls to avoid when coding in VHDL?

Common pitfalls include mixing combinational and sequential logic improperly, neglecting proper reset logic, using improper signal assignments, not considering timing constraints, and writing overly complex monolithic code instead of modular designs.

How can generics enhance the flexibility of VHDL designs?

Generics allow parameterization of VHDL entities, enabling designers to create flexible and reusable modules that can be easily adapted to different data widths, timing, or configurations without rewriting the code.

Why is it important to separate testbench code from design code in VHDL?

Separating testbench code from design code ensures that the design remains clean and synthesizable, facilitates independent testing and verification, and allows reuse of the testbench for different design versions or configurations.

How do synchronous resets improve VHDL design reliability?

Synchronous resets ensure that reset conditions are aligned with the clock, reducing metastability issues and unpredictable behavior, which improves the reliability and predictability of the VHDL design during operation.

Additional Resources

1. *VHDL Programming by Example*
This book offers a hands-on approach to learning VHDL, making it ideal for both beginners and experienced designers. It covers fundamental concepts through practical examples, demonstrating how to write clear and efficient VHDL code. Readers will gain insights into simulation, synthesis, and testbench creation.

2. *RTL Coding Style Guidelines for VHDL*
Focusing on best practices, this book provides a comprehensive set of guidelines to write clean, readable, and maintainable VHDL code. It emphasizes coding styles that enhance design clarity and reduce errors during synthesis. The book is essential for engineers aiming to produce professional-grade RTL designs.

3. *Effective VHDL: Coding and Debugging Techniques*

This resource dives into advanced coding and debugging strategies to optimize VHDL development. It covers common pitfalls, optimization methods, and effective use of simulation tools. Readers learn to improve code reliability and performance through practical techniques.

4. *Design Patterns for VHDL Coding*
Introducing design patterns adapted for VHDL, this book helps designers reuse solutions for common hardware problems. It explains how to implement modular, scalable, and testable code structures. This approach leads to more efficient development cycles and better-designed hardware systems.

5. *VHDL for Designers: Principles and Practices*
Offering a balanced blend of theory and practice, this book guides readers through the principles of VHDL and their application in real-world projects. It covers synthesis, simulation, and verification with a focus on writing effective and efficient code. Examples reinforce the concepts for practical understanding.

6. *Mastering VHDL: From Basics to Advanced Coding Techniques*
This comprehensive guide takes readers from foundational VHDL concepts to sophisticated coding methodologies. It includes detailed explanations of language constructs, concurrent and sequential programming, and testbench development. The book is designed to help coders write robust and optimized hardware descriptions.

7. *VHDL Coding for Synthesis: Best Practices and Tools*
Focusing on synthesis-driven coding, this book offers strategies to ensure VHDL code is synthesizable and meets design constraints. It discusses tool-specific considerations and how to write code that translates efficiently into hardware. Readers benefit from insights into timing, resource utilization, and debugging synthesis issues.

8. *Testbench Techniques for VHDL Verification*
Verification is critical in hardware design, and this book specializes in creating effective testbenches using VHDL. It covers stimulus generation, response checking, and automated verification methods. The book enables designers to detect and fix errors early in the development process.

9. *Practical VHDL Coding: Tips and Tricks for Efficient Design*
Packed with practical advice, this book shares tips and tricks that improve VHDL coding efficiency and readability. It highlights common mistakes and how to avoid them, along with optimization techniques for performance gains. The book is a valuable resource for both novices and seasoned developers aiming to refine their coding style.

Effective Coding With Vhdl

Effective Coding with VHDL

Unlock the power of VHDL and design high-performance, reliable digital systems with ease! Are you struggling to write clean, efficient, and verifiable VHDL code? Do you find yourself overwhelmed by complex design challenges, debugging nightmares, and missed deadlines? Do you wish you could confidently tackle advanced VHDL features and create robust, reusable components? If so, then this book is your solution.

This comprehensive guide, "Mastering VHDL: From Beginner to Expert," will transform your VHDL coding skills, equipping you with the knowledge and practical techniques needed to succeed in digital design.

Contents:

Mastering VHDL: From Beginner to Expert

Introduction: Embarking on Your VHDL Journey

VHDL (VHSIC Hardware Description Language) is a powerful and widely used language for designing and verifying digital systems. It allows engineers to describe hardware at a high level of abstraction, enabling efficient design, simulation, and synthesis for FPGAs (Field-Programmable Gate Arrays) and ASICs (Application-Specific Integrated Circuits). This introduction sets the foundation for your VHDL learning journey. We'll cover the basics of VHDL's role in digital design, its advantages, and how it simplifies complex hardware development.

This section will also guide you through setting up your VHDL development environment. We'll explore popular Integrated Development Environments (IDEs) like ModelSim, Vivado, and others, providing step-by-step instructions for installation and configuration. This ensures you're ready to start writing and simulating your first VHDL code without encountering frustrating setup issues.

Chapter 1: VHDL Fundamentals: Building Blocks of Digital Design

This chapter lays the groundwork for your VHDL skills by introducing the core concepts:

1.1 Data Types: Representing Information

Understanding VHDL's data types is paramount. We'll explore fundamental types like `bit`, `bit_vector`, `integer`, `real`, `std_logic`, `std_logic_vector` and their applications. We'll discuss type declarations, type conversions, and the importance of choosing appropriate data types for efficient and accurate design. The implications of using `std_logic` versus `bit` will be discussed in detail, emphasizing the benefits of `std_logic` for handling high-impedance states and unknowns.

1.2 Operators: Manipulating Data

VHDL offers a rich set of operators for manipulating data. This section covers logical operators (AND, OR, XOR, NOT), arithmetic operators (+, -, , /, MOD, REM), relational operators (=, /=, <, >, <=, >=), and concatenation operators. We'll illustrate their usage through practical examples and explain the operator precedence rules to prevent common coding errors.

1.3 Signals and Variables: Storing and Communicating Data

The distinction between signals and variables is crucial in VHDL. Signals represent hardware connections and undergo a delta-delay during simulation, reflecting the propagation delay in real hardware. Variables, on the other hand, are used for internal calculations within a process and have immediate assignments. We’ll explain the difference with clear examples and demonstrate how to use them effectively.

1.4 Processes: Describing Sequential Logic

Processes are fundamental building blocks for describing sequential logic in VHDL. We'll explain different process sensitivity lists, including event-driven processes and wait statements. We'll also cover how processes are used to model flip-flops, counters, and other sequential elements. The concepts of concurrent and sequential statements will be clarified.

1.5 Combinational and Sequential Logic: Designing Basic Circuits

This section combines the knowledge acquired earlier to design basic combinational and sequential circuits. We'll implement simple circuits like adders, multiplexers, and counters using VHDL. The process of translating hardware schematics into VHDL code will be shown through step-by-step examples.

Chapter 2: Advanced VHDL Constructs: Enhancing Code Reusability and Efficiency

This chapter dives into the advanced features of VHDL that enable modularity, code reuse, and efficient design:

2.1 Generics: Parameterizing Components

Generics allow you to create parameterized components. This means you can instantiate the same component with different values without modifying the component's source code. This significantly improves code reusability. We'll provide examples of using generics to create flexible and adaptable components like parameterized adders and memories.

2.2 Functions and Procedures: Modularizing Functionality

Functions and procedures allow you to encapsulate code into reusable blocks. Functions return a value, while procedures perform actions without returning a value. This improves code readability and maintainability. We'll demonstrate how to write and use functions and procedures for common tasks in digital design.

2.3 Packages: Organizing Code

Packages provide a way to group related declarations, such as data types, constants, functions, and procedures. This improves code organization and facilitates code reuse across multiple projects. We'll discuss creating and using packages for common tasks and standard components.

2.4 Components: Creating Reusable Modules

Components allow you to encapsulate a piece of design into a reusable module. This supports hierarchical design, making complex designs easier to manage and maintain. We'll show how to create, instantiate, and connect components in a larger design.

Chapter 3: Testbenches and Verification: Ensuring Design Correctness

Thorough verification is crucial for successful digital design. This chapter covers writing effective testbenches to verify the functionality of VHDL designs:

3.1 Writing Effective Testbenches

This section details the creation of testbenches that rigorously test the functionality of your designs. We'll discuss various testing methodologies and how to write efficient testbenches to cover all possible scenarios.

3.2 Simulation and Debugging

This section covers simulation tools and techniques for debugging VHDL code. We'll demonstrate how to use simulators to detect errors and fix them effectively.

3.3 Coverage Analysis

We'll explore techniques for analyzing testbench coverage and identify areas requiring additional tests. This ensures thorough verification and higher confidence in the design's correctness.

Chapter 4: Design Patterns and Best Practices: Writing Clean and Efficient Code

This chapter focuses on writing readable, maintainable, and efficient VHDL code:

4.1 Coding Styles and Conventions

We’ll cover various coding styles and conventions commonly used in the industry to create consistent and understandable code. This section promotes code readability and collaboration.

4.2 Optimizing Code for Performance

Techniques to optimize VHDL code for faster execution and lower resource utilization will be discussed. We'll examine methods to reduce logic complexity and minimize resource usage in synthesis.

4.3 Best Practices for Reusability

This section covers guidelines for writing reusable and modular code, fostering efficient development and reduced redundancy. It builds upon the earlier sections on advanced constructs.

Chapter 5: Advanced Design Techniques: Tackling Complex Designs

This chapter delves into more advanced design techniques:

5.1 State Machines: Designing Complex Sequential Logic

This section focuses on designing complex sequential logic using state machines, a fundamental design pattern for managing complex behavior.

5.2 Pipelining: Improving Performance

We’ll discuss pipelining techniques to improve the performance of designs by overlapping execution stages.

5.3 Asynchronous Design: Handling Timing Challenges

We’ll cover strategies for designing and verifying asynchronous circuits which are less constrained by clock signals, and often used in certain high-speed applications.

Chapter 6: Synthesis and Implementation: Preparing for Hardware

This chapter bridges the gap between VHDL code and hardware implementation:

6.1 The Synthesis Process

A detailed look at how VHDL code is converted into hardware implementations for FPGAs and ASICs.

6.2 Optimizing Designs for Synthesis

We'll explore techniques to optimize designs for synthesis, maximizing performance and minimizing resource consumption.

6.3 Implementation in FPGAs and ASICs

We'll discuss the steps for implementing designs on FPGAs and ASICs, including constraints and timing analysis.

Conclusion: Your Continued VHDL Journey

This conclusion summarizes the key concepts covered and provides resources and direction for your continued learning. It also hints at future advancements and the ever-evolving landscape of VHDL and digital design.

FAQs

1. What is the difference between `std_logic` and `bit` in VHDL? `std_logic` offers a wider range of values, including high impedance and undefined states, making it better suited for modeling real-world hardware. `bit` is simpler but lacks this expressiveness.

2. How do I choose the right data type for my VHDL design? Consider the range of values needed, the required precision, and whether you need to handle undefined or high-impedance states.

3. What are the benefits of using generics in VHDL? Generics make your code more reusable and adaptable by allowing you to create parameterized components.

4. How do I write an effective testbench? Create a testbench that systematically tests all aspects of your design using various input combinations and checking for expected outputs.

5. What are some common VHDL coding mistakes to avoid? Common mistakes include incorrect signal assignments, confusing signals and variables, and improper use of wait statements.

6. What is the role of synthesis in VHDL design? Synthesis translates your VHDL code into a netlist, a description of the hardware implementation.

7. How do I optimize my VHDL code for performance? Optimize your code by minimizing logic depth, using efficient algorithms, and employing pipelining where appropriate.

8. What are some good resources for learning more about VHDL? Numerous online tutorials, books, and courses are available, along with vendor-specific documentation for your chosen FPGA or ASIC tools.

9. What are some common design patterns in VHDL? State machines, finite state machines, and pipelining are among the most commonly used design patterns in VHDL.

Related Articles:

1. VHDL for Beginners: A Step-by-Step Tutorial: This article provides a beginner-friendly introduction to VHDL, covering basic syntax and concepts.

2. Mastering VHDL Processes: Sequential Logic Design: A deep dive into VHDL processes and their use in designing sequential circuits.

3. Optimizing VHDL Code for FPGAs: Tips and Tricks: Practical advice on optimizing VHDL code for efficient implementation on FPGAs.

4. Advanced VHDL Techniques: State Machines and Pipelining: Exploration of state machines and pipelining techniques in VHDL.

5. VHDL Testbenches: Writing Effective Verification Strategies: A guide to creating comprehensive and effective testbenches for VHDL designs.

6. Understanding VHDL Data Types and Operators: Detailed explanation of different data types and operators in VHDL.

7. Introduction to VHDL Synthesis and Implementation: An overview of the synthesis and implementation process for VHDL designs.

8. Best Practices for VHDL Coding Style and Readability: Guidelines for writing clean, readable, and maintainable VHDL code.

9. Designing Asynchronous Circuits in VHDL: A detailed explanation of designing asynchronous circuits in VHDL, including techniques for managing timing issues and metastability.

effective coding with vhdl: Effective Coding with VHDL Ricardo Jasinski, 2016-05-27 A guide to applying software design principles and coding practices to VHDL to improve the readability, maintainability, and quality of VHDL code. This book addresses an often-neglected aspect of the creation of VHDL designs. A VHDL description is also source code, and VHDL designers can use the best practices of software development to write high-quality code and to organize it in a design. This book presents this unique set of skills, teaching VHDL designers of all experience levels how to apply the best design principles and coding practices from the software world to the world of hardware. The concepts introduced here will help readers write code that is easier to understand and more likely to be correct, with improved readability, maintainability, and overall quality. After a brief review of VHDL, the book presents fundamental design principles for writing code, discussing such topics as design, quality, architecture, modularity, abstraction, and hierarchy. Building on these concepts, the book then introduces and provides recommendations for each basic element of VHDL code, including statements, design units, types, data objects, and subprograms. The book covers naming data objects and functions, commenting the source code, and visually presenting the code on the screen. All recommendations are supported by detailed rationales. Finally, the book explores two uses of VHDL: synthesis and testbenches. It examines the key characteristics of code intended for synthesis (distinguishing it from code meant for simulation) and then demonstrates the design and implementation of testbenches with a series of examples that verify different kinds of models, including combinational, sequential, and FSM code. Examples from the book are also available on a companion website, enabling the reader to experiment with the complete source code.

effective coding with vhdl: RTL Hardware Design Using VHDL Pong P. Chu, 2006-04-20 The skills and guidance needed to master RTL hardware design This book teaches readers how to systematically design efficient, portable, and scalable Register Transfer Level (RTL) digital circuits using the VHDL hardware description language and synthesis software. Focusing on the module-level design, which is composed of functional units, routing circuit, and storage, the book illustrates the relationship between the VHDL constructs and the underlying hardware components, and shows how to develop codes that faithfully reflect the module-level design and can be synthesized into efficient gate-level implementation. Several unique features distinguish the book: * Coding style that shows a clear relationship between VHDL constructs and hardware components * Conceptual diagrams that illustrate the realization of VHDL codes * Emphasis on the code reuse * Practical examples that demonstrate and reinforce design concepts, procedures, and techniques * Two chapters on realizing sequential algorithms in hardware * Two chapters on scalable and parameterized designs and coding * One chapter covering the synchronization and interface between multiple clock domains Although the focus of the book is RTL synthesis, it also examines the synthesis task from the perspective of the overall development process. Readers learn good design practices and guidelines to ensure that an RTL design can accommodate future simulation, verification, and testing needs, and can be easily incorporated into a larger system or reused. Discussion is independent of technology and can be applied to both ASIC and FPGA devices. With a balanced presentation of fundamentals and practical examples, this is an excellent textbook for upper-level undergraduate or graduate courses in advanced digital logic. Engineers who need to make effective use of today's synthesis software and FPGA devices should also refer to this book.

effective coding with vhdl: VHDL Coding and Logic Synthesis with Synopsys Weng Fook Lee, 2000-08-22 This book provides the most up-to-date coverage using the Synopsys program in the design of integrated circuits. The incorporation of synthesis tools is the most popular new method of designing integrated circuits for higher speeds covering smaller surface areas.Synopsys is the dominant computer-aided circuit design program in the world. All of the major circuit manufacturers and ASIC design firms use Synopsys. In addition, Synopsys is used in teaching and laboratories at over 600 universities. - First practical guide to using synthesis with Synopsys - Synopsys is the #1 design program for IC design

effective coding with vhdl: The Student's Guide to VHDL Peter J. Ashenden, 2008-05-19 The Student's Guide to VHDL is a condensed edition of The Designer's Guide to VHDL, the most widely used textbook on VHDL for digital system modeling. The Student's Guide is targeted as a supplemental reference book for computer organization and digital design courses. Since publication of the first edition of The Student's Guide, the IEEE VHDL and related standards have been revised. The Designer's Guide has been revised to reflect the changes, so it is appropriate that The Student's Guide also be revised. In The Student's Guide to VHDL, 2nd Edition, we have included a design case study illustrating an FPGA-based design flow. The aim is to show how VHDL modeling fits into a design flow, starting from high-level design and proceeding through detailed design and verification, synthesis, FPGA place and route, and final timing verification. Inclusion of the case study helps to better serve the educational market. Currently, most college courses do not formally address the details of design flow. Students may be given informal guidance on how to proceed with lab projects. In many cases, it is left to students to work it out for themselves. The case study in The Student's Guide provides a reference design flow that can be adapted to a variety of lab projects.

effective coding with vhdl: Circuit Design with VHDL, third edition Volnei A. Pedroni, 2020-04-14 A completely updated and expanded comprehensive treatment of VHDL and its applications to the design and simulation of real, industry-standard circuits. This comprehensive treatment of VHDL and its applications to the design and simulation of real, industry-standard circuits has been completely updated and expanded for the third edition. New features include all VHDL-2008 constructs, an extensive review of digital circuits, RTL analysis, and an unequaled collection of VHDL examples and exercises. The book focuses on the use of VHDL rather than solely on the language, with an emphasis on design examples and laboratory exercises. The third edition begins with a detailed review of digital circuits (combinatorial, sequential, state machines, and FPGAs), thus providing a self-contained single reference for the teaching of digital circuit design with VHDL. In its coverage of VHDL-2008, it makes a clear distinction between VHDL for synthesis and VHDL for simulation. The text offers complete VHDL codes in examples as well as simulation results and comments. The significantly expanded examples and exercises include many not previously published, with multiple physical demonstrations meant to inspire and motivate students. The book is suitable for undergraduate and graduate students in VHDL and digital circuit design, and can be used as a professional reference for VHDL practitioners. It can also serve as a text for digital VLSI in-house or academic courses.

effective coding with vhdl: *VHDL: Programming by Example* Douglas L. Perry, 2002-06-02 * Teaches VHDL by example * Includes tools for simulation and synthesis * CD-ROM containing Code/Design examples and a working demo of ModelSIM

effective coding with vhdl: The Designer's Guide to VHDL Peter J. Ashenden, 2002 CD-ROM contains: Access to an introductory version of a graphical VHDL simulator/debugger from FTL Systems -- Code for examples and case studies.

effective coding with vhdl: Vhdl by Example Blaine C. Readler, 2014-05-28 A practical primer for the student and practicing engineer already familiar with the basics of digital design, the reference develops a working grasp of the VHLD hardware description language step-by-step using easy-to-understand examples. Starting with a simple but workable design sample, increasingly more complex fundamentals of the language are introduced until all core features of VHDL are brought to light. Included in the coverage are state machines, modular design, FPGA-based memories, clock management, specialized I/O, and an introduction to techniques of simulation. The goal is to prepare the reader to design real-world FPGA solutions. All the sample code used in the book is available online. What Strunk and White did for the English language with The Elements of Style, VHDL BY EXAMPLE does for FPGA design.

effective coding with vhdl: *VHDL Coding Styles and Methodologies* Ben Cohen, 2012-12-06 VHDL Coding Styles and Methodologies was originally written as a teaching tool for a VHDL training course. The author began writing the book because he could not find a practical and easy to read book that gave in depth coverage of both, the language and coding methodologies. This book is intended for: 1. College students. It is organized in 13 chapters, each covering a separate aspect of the language, with complete examples. All VHDL code described in the book is on a companion 3.5 PC disk. Students can compile and simulate the examples to get a greater understanding of the language. Each chapter includes a series of exercises to reinforce the concepts. 2. Engineers. It is written by an aerospace engineer who has 26 years of hardware, software, computer architecture and simulation experience. It covers practical applications ofVHDL with coding styles and methodologies that represent what is current in the industry. VHDL synthesizable constructs are identified. Guidelines for testbench designs are provided. Also included is a project for the design of a synthesizable Universal Asynchronous Receiver Transmitter (UART), and a testbench to verify proper operation of the UART in a realistic environment, with CPU interfaces and transmission line jitter. An introduction to VHDL Initiative Toward ASIC Libraries (VITAL) is also provided. The book emphasizes VHDL 1987 standard but provides guidelines for features implemented in VHDL 1993.

effective coding with vhdl: High-level Synthesis Michael Fingeroff, 2010 Are you an RTL or system designer that is currently using, moving, or planning to move to an HLS design environment? Finally, a comprehensive guide for designing hardware using C++ is here. Michael Fingeroff's High-Level Synthesis Blue Book presents the most effective C++ synthesis coding style for achieving high quality RTL. Master a totally new design methodology for coding increasingly complex designs! This book provides a step-by-step approach to using C++ as a hardware design language, including an introduction to the basics of HLS using concepts familiar to RTL designers. Each chapter provides easy-to-understand C++ examples, along with hardware and timing diagrams where appropriate. The book progresses from simple concepts such as sequential logic design to more complicated topics such as memory architecture and hierarchical sub-system design. Later chapters bring together many of the earlier HLS design concepts through their application in simplified design examples. These examples illustrate the fundamental principles behind C++ hardware design, which will translate to much larger designs. Although this book focuses primarily on C and C++ to present the basics of C++ synthesis, all of the concepts are equally applicable to SystemC when describing the core algorithmic part of a design. On completion of this book, readers should be well on their way to becoming experts in high-level synthesis.

effective coding with vhdl: FPGA Prototyping by VHDL Examples Pong P. Chu, 2011-09-20 This book uses a learn by doing approach to introduce the concepts and techniques of VHDL and FPGA to designers through a series of hands-on experiments. FPGA Prototyping by VHDL Examples provides a collection of clear, easy-to-follow templates for quick code development; a large number of practical examples to illustrate and reinforce the concepts and design techniques; realistic projects that can be implemented and tested on a Xilinx prototyping board; and a thorough exploration of the Xilinx PicoBlaze soft-core microcontroller.

effective coding with vhdl: *The VHDL Cookbook* Peter J. Ashenden, 1990 A quick introduction to VHDL.

effective coding with vhdl: VHDL: A Logic Synthesis Approach D. Naylor, S. Jones, 1997-07-31 This book is structured in a practical, example-driven, manner. The use of VHDL for constructing logic synthesisers is one of the aims of the book; the second is the application of the tools to the design process. Worked examples, questions and answers are provided together with do and don'ts of good practice. An appendix on logic design the source code are available free of charge over the Internet.

effective coding with vhdl: *VHDL Starter's Guide* Sudhakar Yalamanchili, 1998 VHDL Starter's Guide has been written for the student and practitioner alike as a clear and concise tutorial on VHDL (VHSIC Hardware Description Language). It provides a hands-on, step-by-step introduction to learning VHDL as an applied language to be used in the design and testing of digital logic networks. Command syntax and structure are emphasized, and the writing is based on many examples of real-world logic circuits.

effective coding with vhdl: Beautiful Code Greg Wilson, Andy Oram, 2007-06-26 How do the experts solve difficult problems in software development? In this unique and insightful book, leading computer scientists offer case studies that reveal how they found unusual, carefully designed solutions to high-profile projects. You will be able to look over the shoulder of major coding and design experts to see problems through their eyes. This is not simply another design patterns book, or another software engineering treatise on the right and wrong way to do things. The authors think aloud as they work through their project's architecture, the tradeoffs made in its construction, and when it was important to break rules. This book contains 33 chapters contributed by Brian Kernighan, KarlFogel, Jon Bentley, Tim Bray, Elliotte Rusty Harold, Michael Feathers,Alberto Savoia, Charles Petzold, Douglas Crockford, Henry S. Warren,Jr., Ashish Gulhati, Lincoln Stein, Jim Kent, Jack Dongarra and PiotrLuszczek, Adam Kolawa, Greg Kroah-Hartman, Diomidis Spinellis, AndrewKuchling, Travis E. Oliphant, Ronald Mak, Rogerio Atem de Carvalho andRafael Monnerat, Bryan Cantrill, Jeff Dean and Sanjay Ghemawat, SimonPeyton Jones, Kent Dybvig, William Otte and Douglas C. Schmidt, AndrewPatzer, Andreas Zeller, Yukihiro Matsumoto, Arun Mehta, TV Raman,Laura Wingerd and Christopher Seiwald, and Brian Hayes. Beautiful Code is an opportunity for master coders to tell their story. All author royalties will be donated to Amnesty International.

effective coding with vhdl: Computer Organization and Design RISC-V Edition David A. Patterson, John L. Hennessy, 2017-05-12 The new RISC-V Edition of Computer Organization and Design features the RISC-V open source instruction set architecture, the first open source architecture designed to be used in modern computing environments such as cloud computing, mobile devices, and other embedded systems. With the post-PC era now upon us, Computer Organization and Design moves forward to explore this generational change with examples, exercises, and material highlighting the emergence of mobile computing and the Cloud. Updated content featuring tablet computers, Cloud infrastructure, and the x86 (cloud computing) and ARM (mobile computing devices) architectures is included. An online companion Web site provides advanced content for further study, appendices, glossary, references, and recommended reading. - Features RISC-V, the first such architecture designed to be used in modern computing environments, such as cloud computing, mobile devices, and other embedded systems - Includes relevant examples, exercises, and material highlighting the emergence of mobile computing and the cloud

effective coding with vhdl: Principles of Verifiable RTL Design Lionel Bening, Harry D. Foster, 2001-05-31 The first edition of Principles of Verifiable RTL Design offered a common sense method for simplifying and unifying assertion specification by creating a set of predefined specification modules that could be instantiated within the designer's RTL. Since the release of the first edition, an entire industry-wide initiative for assertion specification has emerged based on ideas presented in the first edition. This initiative, known as the Open Verification Library Initiative (www.verificationlib.org), provides an assertion interface standard that enables the design engineer to capture many interesting properties of the design and precludes the need to introduce new HDL constructs (i.e., extensions to Verilog are not required). Furthermore, this standard enables the design engineer to `specify once,' then target the same RTL assertion specification over multiple verification processes, such as traditional simulation, semi-formal and formal verification tools. The Open Verification Library Initiative is an empowering technology that will benefit design and verification engineers while providing unity to the EDA community (e.g., providers of testbench generation tools, traditional simulators, commercial assertion checking support tools, symbolic simulation, and semi-formal and formal verification tools). The second edition of Principles of Verifiable RTL Design expands the discussion of assertion specification by including a new chapter entitled `Coverage, Events and Assertions'. All assertions exampled are aligned with the Open Verification Library Initiative proposed standard. Furthermore, the second edition provides expanded discussions on the following topics: start-up verification; the place for 4-state simulation; race conditions; RTL-style-synthesizable RTL (unambiguous mapping to gates); more `bad stuff'. The goal of the second edition is to keep the topic current. Principles of Verifiable RTL Design, A Functional Coding Style Supporting Verification Processes, Second Edition tells you how you can write Verilog to describe chip designs at the RTL level in a manner that cooperates with verification processes. This cooperation can return an order of magnitude improvement in performance and capacity from tools such as simulation and equivalence checkers. It reduces the labor costs of coverage and formal model checking by facilitating communication between the design engineer and the verification engineer. It also orients the RTL style to provide more useful results from the overall verification process.

effective coding with vhdl: *HDL Chip Design* Douglas J. Smith, 1996

effective coding with vhdl: FPGA Prototyping by Verilog Examples Pong P. Chu, 2011-09-20 FPGA Prototyping Using Verilog Examples will provide you with a hands-on introduction to Verilog synthesis and FPGA programming through a “learn by doing” approach. By following the clear, easy-to-understand templates for code development and the numerous practical examples, you can quickly develop and simulate a sophisticated digital circuit, realize it on a prototyping device, and verify the operation of its physical implementation. This introductory text that will provide you with a solid foundation, instill confidence with rigorous examples for complex systems and prepare you for future development tasks.

effective coding with vhdl: *Digital Logic Design Using Verilog* Vaibhbav Taraate, 2016-05-17 This book is designed to serve as a hands-on professional reference with additional utility as a textbook for upper undergraduate and some graduate courses in digital logic design. This book is organized in such a way that that it can describe a number of RTL design scenarios, from simple to complex. The book constructs the logic design story from the fundamentals of logic design to advanced RTL design concepts. Keeping in view the importance of miniaturization today, the book gives practical information on the issues with ASIC RTL design and how to overcome these concerns. It clearly explains how to write an efficient RTL code and how to improve design performance. The book also describes advanced RTL design concepts such as low-power design, multiple clock-domain design, and SOC-based design. The practical orientation of the book makes it ideal for training programs for practicing design engineers and for short-term vocational programs. The contents of the book will also make it a useful read for students and hobbyists.

effective coding with vhdl: FPGA Prototyping by VHDL Examples Pong P. Chu, 2017-10-23 A hands-on introduction to FPGA prototyping and SoC design This Second Edition of the popular book follows the same "learning-by-doing" approach to teach the fundamentals and practices of VHDL synthesis and FPGA prototyping. It uses a coherent series of examples to demonstrate the process to develop sophisticated digital circuits and IP (intellectual property) cores, integrate them into an SoC (system on a chip) framework, realize the system on an FPGA prototyping board, and verify the hardware and software operation. The examples start with simple gate-level circuits, progress gradually through the RT (register transfer) level modules, and lead to a functional embedded system with custom I/O peripherals and hardware accelerators. Although it is an introductory text, the examples are developed in a rigorous manner, and the derivations follow strict design guidelines and coding practices used for large, complex digital systems. The new edition is completely updated. It presents the hardware design in the SoC context and introduces the hardware-software co-design concept. Instead of treating examples as isolated entities, the book integrates them into a single coherent SoC platform that allows readers to explore both hardware and software "programmability" and develop complex and interesting embedded system projects. The revised edition: Adds four general-purpose IP cores, which are multi-channel PWM (pulse width modulation) controller, I2C controller, SPI controller, and XADC (Xilinx analog-to-digital converter) controller. Introduces a music synthesizer constructed with a DDFS (direct digital frequency synthesis) module and an ADSR (attack-decay-sustain-release) envelop generator. Expands the original video controller into a complete stream-based video subsystem that incorporates a video synchronization circuit, a test pattern generator, an OSD (on-screen display) controller, a sprite generator, and a frame buffer. Introduces basic concepts of software-hardware co-design with Xilinx MicroBlaze MCS soft-core processor. Provides an overview of bus interconnect and interface circuit. Introduces basic embedded system software development. Suggests additional modules and peripherals for interesting and challenging projects. The FPGA Prototyping by VHDL Examples, Second Edition makes a natural companion text for introductory and advanced digital design courses and embedded system course. It also serves as an ideal self-teaching guide for practicing engineers who wish to learn more about this emerging area of interest.

effective coding with vhdl: Design Recipes for FPGAs: Using Verilog and VHDL Peter Wilson, 2011-02-24 Design Recipes for FPGAs: Using Verilog and VHDL provides a rich toolbox of design techniques and templates to solve practical, every-day problems using FPGAs. Using a modular structure, the book gives 'easy-to-find' design techniques and templates at all levels, together with functional code. Written in an informal and 'easy-to-grasp' style, it goes beyond the principles of FPGA s and hardware description languages to actually demonstrate how specific designs can be synthesized, simulated and downloaded onto an FPGA. This book's 'easy-to-find' structure begins with a design application to demonstrate the key building blocks of FPGA design and how to connect them, enabling the experienced FPGA designer to quickly select the right design for their application, while providing the less experienced a 'road map' to solving their specific design problem. The book also provides advanced techniques to create 'real world' designs that fit the device required and which are fast and reliable to implement. This text will appeal to FPGA designers of all levels of experience. It is also an ideal resource for embedded system development engineers, hardware and software engineers, and undergraduates and postgraduates studying an embedded system which focuses on FPGA design. - A rich toolbox of practical FGPA design techniques at an engineer's finger tips - Easy-to-find structure that allows the engineer to quickly locate the information to solve their FGPA design problem, and obtain the level of detail and understanding needed

effective coding with vhdl: Practical FPGA Programming in C David Pellerin, Scott Thibault, 2005 FPGA brings high performance applications to market quickly – this book covers the many emerging platforms in a proven, effective manner.

effective coding with vhdl: *100 Power Tips for FPGA Designers* ,

effective coding with vhdl: Writing Testbenches: Functional Verification of HDL Models Janick Bergeron, 2012-10-21 mental improvements during the same period. What is clearly needed in verification techniques and technology is the equivalent of a synthesis productivity breakthrough. In the second edition of Writing Testbenches, Bergeron raises the verification level of abstraction by introducing coverage-driven constrained-random transaction-level self-checking testbenches all made possible through the introduction of hardware verification languages (HVLs), such as e from Verisity and OpenVera from Synopsys. The state-of-art methodologies described in Writing Test benches will contribute greatly to the much-needed equivalent of a synthesis breakthrough in verification productivity. I not only highly recommend this book, but also I think it should be required reading by anyone involved in design and verification of today's ASIC, SoCs and systems. Harry Foster Chief Architect Verplex Systems, Inc. xviii Writing Testbenches: Functional Verification of HDL Models PREFACE If you survey hardware design groups, you will learn that between 60% and 80% of their effort is now dedicated to verification.

effective coding with vhdl: VHDL-2008 Peter J. Ashenden, Jim Lewis, 2007-11-26 VHDL-2008: Just the New Stuff, as its title says, introduces the new features added to the latest revision of the IEEE standard for the VHDL hardware description language. Written by the Chair and Technical Editor of the IEEE working group, the book is an authoritative guide to how the new features work and how to use them to improve design productivity. It will be invaluable for early adopters of the new language version, for tool implementers, and for those just curious about where VHDL is headed.* First in the market describing the new features of VHDL 2008;* Just the new features, so existing users and implementers can focus on what's new; * Helps readers to learn the new features soon, rather than waiting for new editions of complete VHDL reference books. * Authoritative, written by experts in the area; * Tutorial style, making it more accessible than the VHDL Standard Language Reference Manual.

effective coding with vhdl: Reuse Methodology Manual for System-on-a-chip Designs Michael Keating, Pierre Bricaud, 1999 Silicon technology now allows us to build chips consisting of tens of millions of transistors. This technology not only promises new levels of system integration onto a single chip, but also presents significant challenges to the chip designer. As a result, many ASIC developers and silicon vendors are re-examining their design methodologies, searching for ways to make effective use of the huge numbers of gates now available. Design reuse -- the use of pre-designed and pre-verified cores -- is the most promising opportunity to bridge the gap between available gate-count and designer productivity. Reuse Methodology Manual for System-On-A-Chip Designs, Second Edition outlines an effective methodology for creating reusable designs for use in a System-on-a-Chip (SoC) design methodology. Silicon and tool technologies move so quickly that no singlemethodology can provide a permanent solution to this highly dynamic problem. Instead, this manual is an attempt to capture and incrementally improve on current best practices in the industry, and to give a coherent, integrated view of the design process.

effective coding with vhdl: Digital Design Mohammad A. Karim, Xinghao Chen, 2017-12-19 In today’s digital design environment, engineers must achieve quick turn-around time with ready accesses to circuit synthesis and simulation applications. This type of productivity relies on the principles and practices of computer aided design (CAD). Digital Design: Basic Concepts and Principles addresses the many challenging issues critical to today’s digital design practices such as hazards and logic minimization, finite-state-machine synthesis, cycles and races, and testability theories while providing hands-on experience using one of the industry’s most popular design application, Xilinx Web PACKTM. The authors begin by discussing conventional and unconventional number systems, binary coding theories, and arithmetic as well as logic functions and Boolean algebra. Building upon classic theories of digital systems, the book illustrates the importance of logic minimization using the Karnaugh map technique. It continues by discussing implementation options and examining the pros and cons of each method in addition to an assessment of tradeoffs that often accompany design practices. The book also covers testability, emphasizing that a good digital design must be easy to verify and test with the lowest cost possible. Throughout the text, the authors analyze combinational and sequential logic elements and illustrate the designs of these components in structural, hierarchical, and behavior VHDL descriptions. Coveringfundamentals and best practices, Digital Design: Basic Concepts and Principles provides you with critical knowledge of how each digital component ties together to form a system and develops the skills you need to design and simulate these digital components using modern CAD software.

effective coding with vhdl: *Digital Logic and Microprocessor Design with VHDL* Enoch O. Hwang, 2006 This book will teach students how to design digital logic circuits, specifically combinational and sequential circuits. Students will learn how to put these two types of circuits together to form dedicated and general-purpose microprocessors. This book is unique in that it combines the use of logic principles and the building of individual components to create data paths and control units, and finally the building of real dedicated custom microprocessors and general-purpose microprocessors. After understanding the material in the book, students will be able to design simple microprocessors and implement them in real hardware.

effective coding with vhdl: *The Complete Verilog Book* Vivek Sagdeo, 2007-05-08 The Verilog hardware description language (HDL) provides the ability to describe digital and analog systems. This ability spans the range from descriptions that express conceptual and architectural design to detailed descriptions of implementations in gates and transistors. Verilog was developed originally at Gateway Design Automation Corporation during the mid-eighties. Tools to verify designs expressed in Verilog were implemented at the same time and marketed. Now Verilog is an open standard of IEEE with the number 1364. Verilog HDL is now used universally for digital designs in ASIC, FPGA, microprocessor, DSP and many other kinds of design-centers and is supported by most of the EDA companies. The research and education that is conducted in many universities is also using Verilog. This book introduces the Verilog hardware description language and describes it in a comprehensive manner. Verilog HDL was originally developed and specified with the intent of use with a simulator. Semantics of the language had not been fully described until now. In this book, each feature of the language is described using semantic introduction, syntax and examples. Chapter 4 leads to the full semantics of the language by providing definitions of terms, and explaining data structures and algorithms. The book is written with the approach that Verilog is not only a simulation or synthesis language, or a formal method of describing design, but a complete language addressing all of these aspects. This book covers many aspects of Verilog HDL that are essential parts of any design process.

effective coding with vhdl: Digital System Design with FPGA: Implementation Using Verilog and VHDL Cem Unsalan, Bora Tar, 2017-07-14 Master FPGA digital system design and implementation with Verilog and VHDL This practical guide explores the development and deployment of FPGA-based digital systems using the two most popular hardware description languages, Verilog and VHDL. Written by a pair of digital circuit design experts, the book offers a solid grounding in FPGA principles, practices, and applications and provides an overview of more complex topics. Important concepts are demonstrated through real-world examples, ready-to-run code, and inexpensive start-to-finish projects for both the Basys and Arty boards. Digital System Design with FPGA: Implementation Using Verilog and VHDL covers: • Field programmable gate array fundamentals • Basys and Arty FPGA boards • The Vivado design suite • Verilog and VHDL • Data types and operators • Combinational circuits and circuit blocks • Data storage elements and sequential circuits • Soft-core microcontroller and digital interfacing • Advanced FPGA applications • The future of FPGA

effective coding with vhdl: Digital Systems Design Using VHDL Lizy Kurian John, Charles Roth, 2017-01-01

effective coding with vhdl: The Design Warrior's Guide to FPGAs Clive Maxfield, 2004-06-16 Field Programmable Gate Arrays (FPGAs) are devices that provide a fast, low-cost way for embedded system designers to customize products and deliver new versions with upgraded features, because they can handle very complicated functions, and be reconfigured an infinite number of times. In addition to introducing the various architectural features available in the latest generation of FPGAs, The Design Warrior's Guide to FPGAs also covers different design tools and flows.This book covers information ranging from schematic-driven entry, through traditional HDL/RTL-based simulation and logic synthesis, all the way up to the current state-of-the-art in pure C/C++ design capture and synthesis technology. Also discussed are specialist areas such as mixed hardward/software and DSP-based design flows, along with innovative new devices such as field programmable node arrays (FPNAs). Clive Max Maxfield is a bestselling author and engineer with a large following in the electronic design automation (EDA)and embedded systems industry. In this comprehensive book, he covers all the issues of interest to designers working with, or contemplating a move to, FPGAs in their product designs. While other books cover fragments of FPGA technology or applications this is the first to focus exclusively and comprehensively on FPGA use for embedded systems. - First book to focus exclusively and comprehensively on FPGA use in embedded designs - World-renowned best-selling author - Will help engineers get familiar and succeed with this new technology by providing much-needed advice on choosing the right FPGA for any design project

effective coding with vhdl: *Abstraction Mechanisms and Language Design* Paul N. Hilfinger, 1983 The research reported in this book advances the art of designing programming languages. It sets forth some design principles for abstraction mechanisms and demonstrates their power by showing how they led to improvements in the design of Ada, a new language devised for the Department of Defense and one that will be wisely used in DOD-related projects. The author was a full-time consultant for the preliminary Ada language design and his suggestions for revising it were remarkable in that of the many revisions proposed, his were among the few that proved effective. The book also considers the likely outcome of even more substantial changes to the current version of Ada. In focusing on the effects of abstracting or generalizing the definitional mechanisms of a particular language, the author notes that these effects are manifested in two ways: in the complexity and expressive power of the language, and in the expected performance of programs written in it. He concludes, with regard to the first effect, that proper design of the abstraction facilities of a language can simpify it and icrease its utility to programmers, and wiht regard to the second, that abstraction mechanisms need not involve.

effective coding with vhdl: Electric Discharge Lamps John F. Waymouth, 1971 This book brings together an extraordinary amount of data on all the major types of electric discharge lamps which are now in commercial use.

effective coding with vhdl: Digital Systems Design With Vhdl And Synthesis: An Integrated Approach Chang, 2007-01-10 This book presents an integrated approach to digital design principles, processes, and implementations to help the reader design increasingly complex systems within shorter design cycles. It also introduces digital design concepts, VHDL coding, VHDL simulation, synthesis commands, and strategies together. · VHDL and Digital Circuit Primitives· VHDL Simulation and Synthesis Environment and Design Process· Basic Combinational Circuits· Basic Binary Arithmetic Circuits· Basic Sequential Circuits· Registers· Clock and Reset Circuits· Dual-Port RAM, FIFO, and DRAM Modeling· A Design Case Study: Finite Impulse Response Filter ASIC Design· A Design Case Study: A Microprogram Controller Design· Error Detection and Correction· Fixed-Point Multiplication· Fixed-Point Division· Floating-Point Arithmetic

effective coding with vhdl: VHDL for Programmable Logic Kevin Skahill, 1996

effective coding with vhdl: A VHDL Primer Jayaram Bhasker, 1995 This book details molecular methodologies used in identifying a disease gene, from the initial stage of study design to the next stage of preliminary locus identification, and ending with stages involved in target characterization and validation.

effective coding with vhdl: Embedded Microprocessor System Design using FPGAs Uwe

Meyer-Baese, 2021-04-16 This textbook for courses in Embedded Systems introduces students to necessary concepts, through a hands-on approach. It gives a great introduction to FPGA-based microprocessor system design using state-of-the-art boards, tools, and microprocessors from Altera/Intel® and Xilinx®. HDL-based designs (soft-core), parameterized cores (Nios II and MicroBlaze), and ARM Cortex-A9 design are discussed, compared and explored using many hand-on designs projects. Custom IP for HDMI coder, Floating-point operations, and FFT bit-swap are developed, implemented, tested and speed-up is measured. Downloadable files include all design examples such as basic processor synthesizable code for Xilinx and Altera tools for PicoBlaze, MicroBlaze, Nios II and ARMv7 architectures in VHDL and Verilog code, as well as the custom IP projects. Each Chapter has a substantial number of short quiz questions, exercises, and challenging projects. Explains soft, parameterized, and hard core systems design tradeoffs; Demonstrates design of popular KCPSM6 8 Bit microprocessor step-by-step; Discusses the 32 Bit ARM Cortex-A9 and a basic processor is synthesized; Covers design flows for both FPGA Market leaders Nios II Altera/Intel and MicroBlaze Xilinx system; Describes Compiler-Compiler Tool development; Includes a substantial number of Homework's and FPGA exercises and design projects in each chapter.

effective coding with vhdl: Advanced Chip Design Kishore Mishra, 2013 The book is intended for digital and system design engineers with emphasis on design and system architecture. The book is broadly divided into two sections - chapters 1 through 10, focusing on the digital design aspects and chapters 11 through 20, focusing on the system aspects of chip design. It comes with real-world examples in Verilog and introduction to SystemVerilog Assertions (SVA).