Welcome to the book!

The advance of very large-scale integration (VLSI) process technologies has made “system on a chip” feasible for very complex systems. This is partly responsible for the market demand for much shorter design cycles even though design complexity has continued to increase.

There are many design and VHDL code examples, simulation waveforms, and synthesized schematics shown in this book to illustrate their correspondence. VHDL code can be downloaded from the Internet for exercises. All examples have been verified with VHDL simulation and synthesis tools to ensure high fidelity of the VHDL code. All VHDL codes are listed line by line for reference and discussion in the text. Each VHDL example is complete, no fragments so that the complete picture is clear to you. Design techniques of VHDL coding, verification, and synthesis that implement the design concepts and principles are illustrated with actual examples throughout the book.

In the next section, an integrated design process and methodology is introduced to serve as our base line design flow and approach.

1.1 INTEGRATED DESIGN PROCESS AND METHODOLOGY

Figure 1-1 shows a flow chart that describes an integrated VHDL design process and methodology. The design is first described in VHDL. VHDL coding techniques will be applied to model the design efficiently and effectively. A preliminary synthesis for each VHDL code is done to check for obvious design errors such as unintended latches and insufficient sensitivity list before the design is simulated. This also ensures that all VHDL code can be synthesized. This process is discussed and illustrated in Chapter 2 as the VHDL design process for a block. A test bench is developed to facilitate the design verification. Test bench examples and verification techniques are discussed throughout the book for various designs. A more detailed synthesis can be started (in parallel) with the test bench development) to calibrate design and timing constraints. The design architecture may require a change, based on the timing result. The complete design can be verified with the test bench. There will be iterations
among the VHDL coding, synthesis, test bench, and simulation. After the design is free of design errors, the complete and detailed synthesis is done. A netlist can be generated for the layout tool to do the layout. The layout may be performed with Field Programmable Gate Array (FPGA), gate array, or standard cell layout tools. The layout tool can generate another VHDL netlist with the timing delay file in Standard Delay Format (SDF), which can be used for simulation with the test bench. Using the simulation results, test vectors can be generated for the foundry to test the fabricated ASIC design. A FPGA design does not need the test vector. The layout can be performed in-house or by a vendor. For the ASIC design, the layout tool can generate a GDSII file to describe the masks for fabrication.

The synthesis tool can also generate a VHDL netlist for postsynthesis and pre-layout simulation. This is not common. The postlayout simulation is preferred since the timing generated from the layout tools is much closer to the actual hardware. Note also that the test bench should be used for both functional simulation and postlayout timing simulation without the need to change.

In this book, many examples are used to illustrate all the steps of the process described in Figure 1-1.

![VHDL design process and methodology.](image)

**1.2 BOOK OVERVIEW**

Chapter 2 describes how to write VHDL to model basic digital circuit primitives or gates. Flip-flops, latches, and three-state buffers inference is illustrated with examples. VHDL synthesis rules are presented to provide guidelines what circuits will be synthesized by synthesis tools.
Chapter 3 presents the VHDL simulation and synthesis design environments. Synopsys simulation and synthesis design environments and design tools are introduced. Mentor QuickVHDL simulation environment is also discussed. A typical design process for a block is discussed to improve the debugging process.

Chapter 4 presents VHDL modeling, synthesis, and verification of several basic combination circuits such as selector, encoder, code converter, equality checker, and comparators. Each circuit is presented with different VHDL models so that their differences and trade-offs are discussed.

Chapter 5 concentrates on several binary arithmetic circuits. Half adder, full adder, ripple adder, carry look ahead adder, count one, leading zero, and barrel shifter are presented with VHDL modeling, synthesis, and test bench.

Chapter 6 discusses sequential circuits such as counters, shift registers, parallel to serial converter, and serial to parallel converter with VHDL modeling, synthesis, and test bench.

Chapter 7 presents a framework to organize registers in the design. Registers are categorized and discussed. Partition, synthesis, and verification strategies of registers are discussed. VHDL modeling, synthesis, and verification are illustrated.

Chapter 8 is dedicated to clock- and reset-related circuits. The synchronization between different clock domains is discussed. Clock tree generation, clock delay, and clock skew are presented and discussed with timing diagrams and VHDL code. The issues of gated clock and clock divider are also introduced.

Chapter 9 presents examples of dual-port RAM, synchronous and asynchronous FIFO, and dynamic RAM VHDL models. These blocks are commonly used as custom drop-in macros. They are used to interact with the rest of the design so that the complete design can be verified.

Chapter 10 illustrates the complete semicustom ASIC design process of a Finite Impulse Response ASIC design through the steps of design description, VHDL coding, functional verification, synthesis, layout, back-annotated timing verification.

Chapter 11 discusses the concept of a microprogram controller. The design of a AMD AM2910 is presented through the gate array design process from VHDL coding to postlayout back-annotated timing verification. Test vector generation is also illustrated.

Chapter 12 discusses the principles of error and correcting Hamming codes. An actual TI EDAC integrated circuit is used as an example to design from VHDL code to all steps of FPGA design process.

Chapter 13 presents the concepts of binary fixed-point multiplication algorithms such as Booth-Wallace multiplier. The VHDL coding, synthesis, and verification are presented.

Chapter 14 discusses the concepts of binary fixed-point division algorithms. VHDL coding, synthesis, and verification are presented.

Chapter 15 discusses the floating-point number representation. Floating-point addition and multiplication algorithms are discussed and implemented with VHDL codes. They are verified and synthesized.