Chapter 1

Introduction

1.1 WHAT IS VHDL?

In 1980 the US government developed the Very High Speed Integrated Circuit (VHSIC) project to enhance the electronic design process, technology, and procurement, spawning development of many advanced integrated circuit (IC) process technologies. This was followed by the arrival of VHSIC Hardware Description Language (VHDL), which was endorsed by IEEE in 1986 in its attempt at standardization. By December of 1987 the IEEE 1076.1 standard for VHDL was approved and a VHDL Language Reference Manual (LRM) was published.

This book will be based primarily on the 1987 standard, since it is the foundation for most VHDL simulators and synthesis tools currently on the market. It is also the foundation for the VHDL Initiative Towards ASIC Libraries (VITAL) in detailed function and timing modeling of primitive components. However, the 1987 standard was updated in 1993 and must also be addressed. This is done in Chapter 14. VHDL'87 will refer to the 1987 standard and VHDL'93 will refer to the 1993 standard.

1.2 VHDL ADVANTAGES

VHDL offers the following advantages for digital design:

Standard: VHDL is an IEEE standard. Just like any standard (such as graphics X-window standard, bus communication interface standard, high-level programming languages, and so on), it reduces confusion and makes interfaces between tools, companies, and products easier. Any development to the standard would have better chances of lasting longer and have less chance of becoming obsolete due to incompatibility with others.

Government support: VHDL is a result of the VHSIC program; hence, it is clear that the US government supports the VHDL standard for electronic procurement. The Department of Defense (DOD) requires contractors to supply VHDL for all Application Specific Integrated Circuit (ASIC) designs.

Industry support: With the advent of more powerful and efficient VHDL tools has come the growing support of the electronic industry. Companies use VHDL tools not only with regard to defense contracts, but also for their commercial designs.

Portability: The same VHDL code can be simulated and used in many design tools and at different stages of the design process. This reduces dependency on a set of design tools whose limited capability may not be competitive in later markets. The VHDL standard also transforms design data much easier than a design database of a proprietary design tool.

Modeling capability: VHDL was developed to model all levels of designs, from electronic boxes to transistors. VHDL can accommodate behavioral constructs and mathematical routines that describe complex models, such as queuing networks and analog circuits. It allows use of multiple architectures and associates with the same design during various stages of the design process. As shown in Figure 1.1, VHDL can describe low-level transistors up to very large systems. This book will concentrate on using VHDL for the design of ASICs and digital blocks.

Reusability: Certain common designs can be described, verified, and modified slightly in VHDL for future use. This eliminates reading and marking changes to schematic pages, which is time consuming and subject to error. For example, a parameterized multiplier VHDL code can be reused easily by changing the width parameter so that the same VHDL code can do either 16 by 16 or 12 by 8 multiplication.

Technology and foundry independence: The functionality and behavior of the design can be described with VHDL and verified, making it foundry and technology independent. This frees the designer to proceed without having to wait for the foundry and technology to be selected.

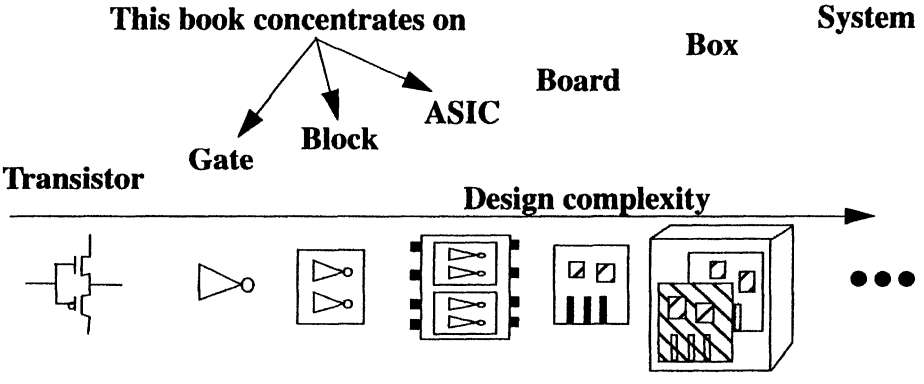


FIGURE 1.1 VHDL modeling capability.

Documentation: VHDL is a design description language which allows documentation to be located in a single place by embedding it in the code. The combining of comments and the code that actually dictates what the design should do reduces the ambiguity between specification and implementation.

New design methodology: Using VHDL and synthesis creates a new methodology that increases the design productivity, shortens the design cycle, and lowers costs. It amounts to a revolution comparable to that introduced by the automatic semi-custom layout synthesis tools of the last few years.

1.3 WHAT IS LOGIC SYNTHESIS?

Synthesis, in the domain of digital design, is a process of translation and optimization. For example, layout synthesis is a process of taking a design netlist and translating it into a form of data that facilitates placement and routing, resulting in optimizing timing and/or chip size. Logic synthesis, on the other hand, is the process of taking a form of input (VHDL), translating it into a form (Boolean equations and synthesis tool specific), and then optimizing in terms of propagation delay and/or area. For example, Figure 1.2 shows the schematic view of the synthesis tool specific internal form translated from a VHDL code. Note that there are some blocks to generate sequential circuits, such as flipflops, and some blocks to generate combinational circuits.

After the VHDL code is translated into an internal form, the optimization process can be performed based on constraints such as speed, area, power, and so on. Figure 1.3 depicts the schematic, after the circuit is optimized.

1.4 NEW DESIGN METHODOLOGY

Introducing VHDL and synthesis enables the design community to explore a new design methodology. The traditional design approach, as shown in Figure 1.4a, starts with drawing schematics and then performs functional and timing simulation based on the same schematic. If there is any design error, the process iterates back to update schematics. After the layout, functions and back annotated timing are verified again with the same schematics.

The VHDL-based design approach is illustrated in Figure 1.4b. The design is functionally described with VHDL. VHDL simulation is used to verify the functionality of the design. In general, modifying VHDL source code is much faster than changing schematics. This allows designers to make faster functionally correct designs, to explore more architecture trade-offs, and to have more impact on the designs. After the functions match the requirements, the VHDL code is synthesized to generate schematics (or equivalent netlists). The netlist can be used to layout the circuit and to verify the timing requirement (both before or after the layout). The design changes can be made by modifying VHDL code or changing the constraints (timing, area, and so on) in the synthesis. This new design approach and methodology has improved the design process by shortening design time, reducing the number of design iterations, and increasing the design complexity that designers can manage.

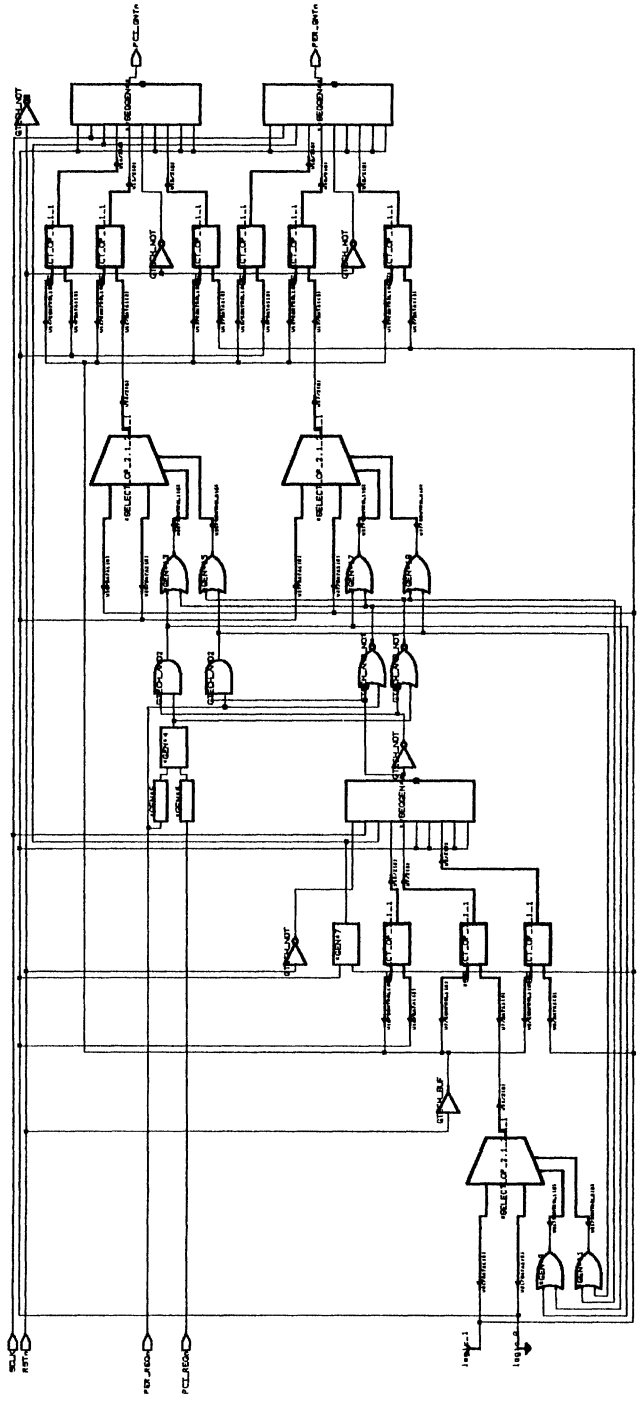


FIGURE 1.2 Translated VHDL into an internal form.

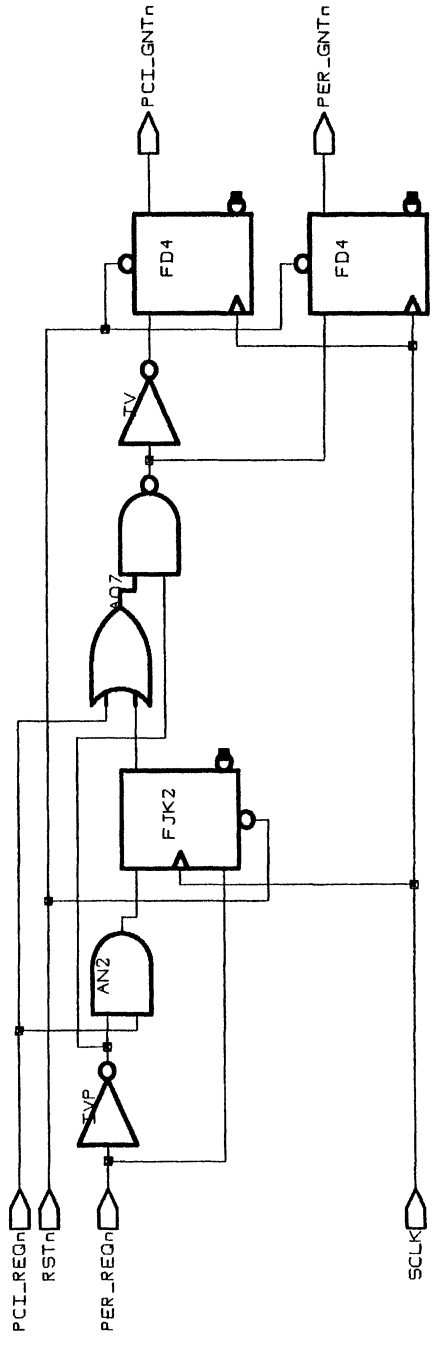
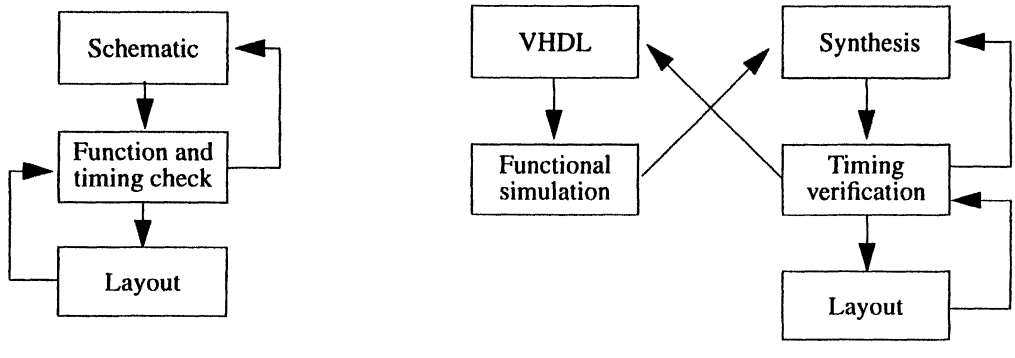


FIGURE 1.3 Optimized schematic.

1.5 BOOK OVERVIEW

The book has 14 chapters. This chapter gives a quick overview of VHDL history and usage. Chapter 2 summarizes VHDL basic language elements. Chapter 3 introduces the VHDL simulation concepts. Chapters 4 and 5 discuss VHDL sequential and concurrent statements respectively. Chapter 6 presents VHDL subprograms and packages. Chapter 7 discusses VHDL library and design unit concepts and configuration. Chapter 8 points out important issues of writing VHDL for synthesis. Chapter 9 illustrates writing VHDL for finite state machines. Chapter 10 discusses more behavioral modeling of file text I/O, ROM, RAM, pad models, guarded block, guarded signal, disconnect, and null constructs. Chapter 11 presents a small design case with test bench design techniques. Chapter 12 discusses an ALU design example. Chapter 13 describes a design case study that goes through the complete design process from the design concept, VHDL implementation, verification, test bench, layout, and post-layout verification. Chapter 14 summarizes VHDL'93 important updates.

In this book, many design examples with VHDL are used for simulation and synthesis illustrations. The original VHDL code is imported into the book with no modification so that readers can see the entire VHDL code (not just part of it). Every line of VHDL code has an associated line number for easy reference and discussion. The font used for the VHDL code portion is "Courier"—each character occupies the same width and characters line up vertically. This would be close to how an ASCII file appears when readers type in their VHDL code with any text editor. The schematics are generated with Synopsys synthesis tools and then imported. Note that the schematic can be different from what you may implement manually from your synthesis output. This is due to different technology component libraries and constraints that are used in the synthesis process. However, the synthesized schematic should represent and correspond functionally with the VHDL code. Three purposes are served by including the entire VHDL code, simulation, and synthesized schematic: (1) Typing



(a) Traditional schematic design approach (b) VHDL-based design approach

FIGURE 1.4 Design process comparison.

errors in the VHDL code are avoided; (2) readers can use the code as examples for reference, exercises, and to build upon them; (3) readers will see the correspondence among VHDL code, simulation, and synthesis results.

1.6 EXERCISES

1.1 What do VHSIC, VHDL, and VITAL stand for?

1.2 What advantages can we expect when using VHDL in digital designs? Do you agree on the advantages discussed in the text?

1.3 What is synthesis? Have you seen the terminology of "layout" synthesis? Can we say that C language compiler is a synthesis tool for C programming language?

1.4 In software design, the trend has been moving from writing machine dependent assembly languages to writing machine independent standard C high-level programming language. In hardware design, library dependent schematics can be created manually or library and technology independent VHDL code can be written to describe equivalent functions. Are there any similarities between software and hardware design? Do you agree that more and more digital designs will be using VHDL and synthesis just like the trend in software design?