Check for updates

Chapter 1: The Modern Digital Design Flow

The purpose of hardware description languages is to describe digital circuitry using a text-based language. HDLs provide a means to describe large digital systems without the need for schematics, which can become impractical in very large designs. HDLs have evolved to support logic simulation at different levels of abstraction. This provides designers the ability to begin designing and verifying functionality of large systems at a high level of abstraction and postpone the details of the circuit implementation until later in the design cycle. This enables a top-down design approach that is scalable across different logic families. HDLs have also evolved to support automated *synthesis*, which allows the CAD tools to take a functional description of a system (e.g., a truth table) and automatically create the gate-level circuitry to be implemented in real hardware. This allows designers to focus their attention on designing the behavior of a system and not spend as much time performing the formal logic synthesis steps as in the classical digital design approach.

There are two dominant hardware description languages in use today. They are VHDL and Verilog. VHDL stands for *very high speed integrated circuit hardware description language*. Verilog is not an acronym but rather a trade name. The use of these two HDLs is split nearly equally within the digital design industry. Once one language is learned, it is simple to learn the other language, so the choice of the HDL to learn first is somewhat arbitrary. In this text we will use Verilog to learn the concepts of an HDL. Verilog is more lenient on its typecasting than VHDL, so it is a good platform for beginners as systems can be designed with less formality. The goal of this chapter is to provide the background and context of the modern digital design flow using an HDL-based approach.

Learning Outcomes—After completing this chapter, you will be able to:

1.1 Describe the role of hardware description languages in modern digital design.
1.2 Describe the fundamentals of design abstraction in modern digital design.
1.3 Describe the modern digital design flow based on hardware description languages.

1.1 History of Hardware Description Languages

The invention of the integrated circuit is most commonly credited to two individuals who filed patents on different variations of the same basic concept within 6 months of each other in 1959. Jack Kilby filed the first patent on the integrated circuit in February of 1959 titled “Miniaturized Electronic Circuits” while working for *Texas Instruments*. Robert Noyce was the second to file a patent on the integrated circuit in July of 1959 titled “Semiconductor Device and Lead Structure” while at a company he cofounded called *Fairchild Semiconductor*. Kilby went on to win the Nobel Prize in Physics in 2000 for his invention, while Noyce went on to cofound *Intel Corporation* in 1968 with Gordon Moore. In 1971, Intel introduced the first single-chip microprocessor using integrated circuit technology, the *Intel 4004*. This microprocessor IC contained 2300 transistors. This series of inventions launched the semiconductor industry, which was the driving force behind the growth of Silicon Valley and led to 40 years of unprecedented advancement in technology that has impacted every aspect of the modern world.

Gordon Moore, cofounder of Intel, predicted in 1965 that the number of transistors on an integrated circuit would double every 2 years. This prediction, now known as *Moore’s Law*, has held true since the invention of the integrated circuit. As the number of transistors on an integrated circuit grew, so did the size of the design and the functionality that could be implemented. Once the first microprocessor was

© Springer Nature Switzerland AG 2019

B. J. LaMeres, *Quick Start Guide to Verilog*, https://doi.org/10.1007/978-3-030-10552-5_1

invented in 1971, the capability of CAD tools increased rapidly enabling larger designs to be accomplished. These larger designs, including newer microprocessors, enabled the CAD tools to become even more sophisticated and, in turn, yield even larger designs. The rapid expansion of electronic systems based on digital integrated circuits required that different manufacturers needed to produce designs that were compatible with each other. The adoption of logic family standards helped manufacturers ensure their parts would be compatible with other manufacturers at the physical layer (e.g., voltage and current); however, one challenge that was encountered by the industry was a way to document the complex behavior of larger systems. The use of schematics to document large digital designs became too cumbersome and difficult to understand by anyone besides the designer. Word descriptions of the behavior were easier to understand, but even this form of documentation became too voluminous to be effective for the size of designs that were emerging.

In 1983, the US Department of Defense (DoD) sponsored a program to create a means to document the behavior of digital systems that could be used across all of its suppliers. This program was motivated by a lack of adequate documentation for the functionality of application specific integrated circuits (ASICs) that were being supplied to the DoD. This lack of documentation was becoming a critical issue as ASICs would come to the end of their life cycle and need to be replaced. With the lack of a standardized documentation approach, suppliers had difficulty reproducing equivalent parts to those that had become obsolete. The DoD contracted three companies (Texas Instruments, IBM, and Intermetrics) to develop a standardized documentation tool that provided detailed information about both the interface (i.e., inputs and outputs) and the behavior of digital systems. The new tool was to be implemented in a format similar to a programming language. Due to the nature of this type of language-based tool, it was a natural extension of the original project scope to include the ability to *simulate* the behavior of a digital system. The simulation capability was desired to span multiple levels of abstraction to provide maximum flexibility. In 1985, the first version of this tool, called VHDL, was released. In order to gain widespread adoption and ensure consistency of use across the industry, VHDL was turned over to the *Institute of Electrical and Electronic Engineers* (IEEE) for standardization. IEEE is a professional association that defines a broad range of open technology standards. In 1987, IEEE released the first industry standard version of VHDL. The release was titled IEEE 1076-1987. Feedback from the initial version resulted in a major revision of the standard in 1993 titled IEEE 1076-1993. While many minor revisions have been made to the 1993 release, the 1076-1993 standard contains the vast majority of VHDL functionality in use today. The most recent VHDL standard is IEEE 1076-2008.

Also in 1983, the Verilog HDL was developed by *Automated Integrated Design Systems* as a logic simulation language. The development of Verilog took place completely independent from the VHDL project. Automated Integrated Design Systems (renamed *Gateway Design Automation* in 1985) was acquired by CAD tool vendor *Cadence Design Systems* in 1990. In response to the popularity of Verilog's intuitive programming and superior simulation support, and also to stay competitive with the emerging VHDL standard, Cadence made the Verilog HDL open to the public. IEEE once again developed the open standard for this HDL, and in 1995 released the Verilog standard titled IEEE 1364-1995. This release has undergone numerous revisions with the most significant occurring in 2001. It is common to refer to the major releases as "Verilog 1995" and "Verilog 2001" instead of their official standard numbers.

The development of CAD tools to accomplish automated logic synthesis can be dated back to the 1970s when IBM began developing a series of practical synthesis engines that were used in the design of their mainframe computers; however, the main advancement in logic synthesis came with the founding of a company called *Synopsis* in 1986. Synopsis was the first company to focus on logic synthesis directly from HDLs. This was a major contribution because designers were already using HDLs to describe and simulate their digital systems, and now logic synthesis became integrated in the same design flow. Due to the complexity of synthesizing highly abstract functional descriptions, only lower levels of abstraction that were thoroughly elaborated were initially able to be synthesized. As CAD tool

capability evolved, synthesis of higher levels of abstraction became possible, but even today not all functionality that can be described in an HDL can be synthesized.

The history of HDLs, their standardization, and the creation of the associated logic synthesis tools is key to understanding the use and limitations of HDLs. HDLs were originally designed for documentation and behavioral simulation. Logic synthesis tools were developed independently and modified later to work with HDLs. This history provides some background into the most common pitfalls that beginning digital designers encounter, that being that mostly any type of behavior can be described and simulated in an HDL, but only a subset of well-described functionality can be synthesized. Beginning digital designers are often plagued by issues related to designs that simulate perfectly but that will not synthesize correctly. In this book, an effort is made to introduce Verilog at a level that provides a reasonable amount of abstraction while preserving the ability to be synthesized. Figure 1.1 shows a timeline of some of the major technology milestones that have occurred in the past 150 years in the field of digital logic and HDLs.

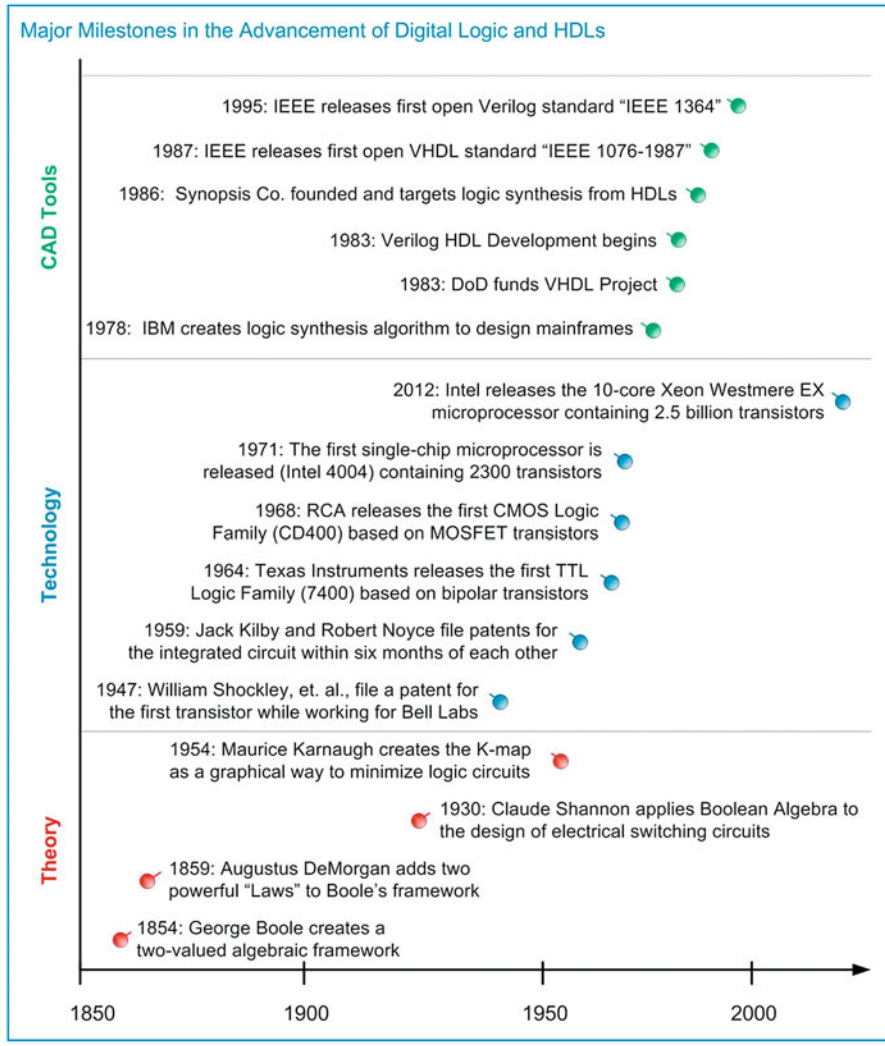

Fig. 1.1
Major milestones in the advancement of digital logic and HDLs

Concept Check

CC1.1 Why does Verilog support modeling techniques that *aren't* synthesizable?

(A) There wasn't enough funding available to develop synthesis capability as it all went to the VHDL project.

(B) At the time Verilog was created, synthesis was deemed too difficult to implement.

(C) To allow Verilog to be used as a generic programming language.

(D) Verilog needs to support all steps in the modern digital design flow, some of which are unsynthesizable such as test pattern generation and timing verification.

1.2 HDL Abstraction

HDLs were originally defined to be able to model behavior at multiple levels of abstraction. Abstraction is an important concept in engineering design because it allows us to specify how systems will operate without getting consumed prematurely with implementation details. Also, by removing the details of the lower-level implementation, simulations can be conducted in reasonable amounts of time to model the higher-level functionality. If a full computer system was simulated using detailed models for every MOSFET, it would take an impracticable amount of time to complete. Figure 1.2 shows a graphical depiction of the different layers of abstraction in digital system design.

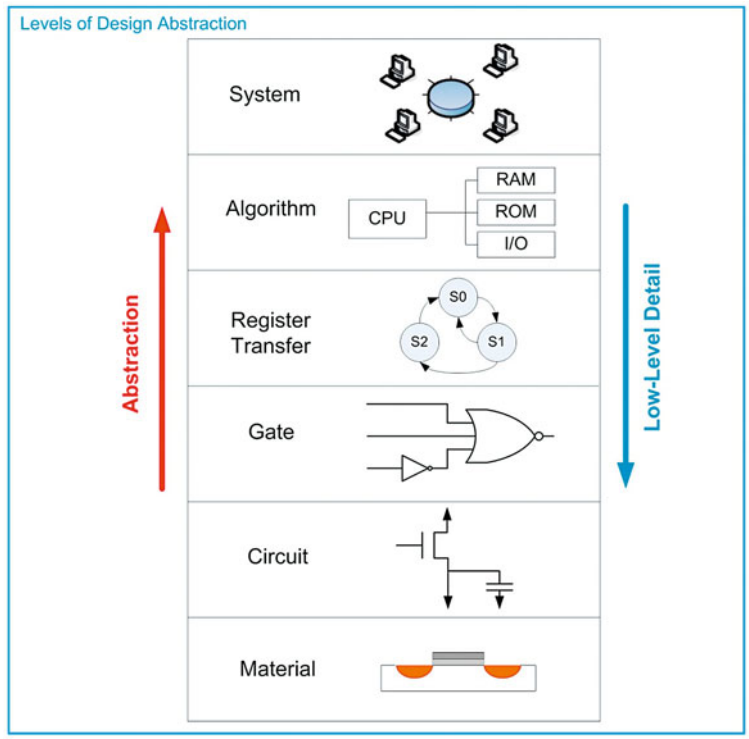

Fig. 1.2
Levels of design abstraction

The highest level of abstraction is the *system level*. At this level, behavior of a system is described by stating a set of broad specifications. An example of a design at this level is a specification such as "the computer system will perform 10 Tera Floating Point Operations per Second (10 TFLOPS) on double precision data and consume no more than 100 W of power." Notice that these specifications do not dictate the lower-level details such as the type of logic family or the type of computer architecture to use. One level down from the system level is the *algorithmic level*. At this level, the specifications begin to be broken down into subsystems, each with an associated behavior that will accomplish a part of the primary task. At this level, the example computer specifications might be broken down into subsystems such as a central processing unit (CPU) to perform the computation and random-access memory (RAM) to hold the inputs and outputs of the computation. One level down from the algorithmic level is the *register transfer level* (*RTL*). At this level, the details of how data is moved between and within subsystems are described in addition to how the data is manipulated based on system inputs. One level down from the RTL level is the *gate level*. At this level, the design is described using basic gates and registers (or storage elements). The gate level is essentially a schematic (either graphically or text-based) that contains the components and connections that will implement the functionality from the above levels of abstraction. One level down from the gate level is the *circuit level*. The circuit level describes the operation of the basic gates and registers using transistors, wires, and other electrical components such as resistors and capacitors. Finally, the lowest level of design abstraction is the *material level*. This level describes how different materials are combined and shaped in order to implement the transistors, devices, and wires from the circuit level.

HDLs are designed to model behavior at all of these levels with the exception of the material level. While there is some capability to model circuit level behavior such as MOSFETs as ideal switches and pull-up/pull-down resistors, HDLs are not typically used at the circuit level. Another graphical depiction of design abstraction is known as the **Gajski and Kuhn's Y-chart**. A Y-chart depicts abstraction across three different design domains: behavioral, structural, and physical. Each of these design domains contains levels of abstraction (i.e., system, algorithm, RTL, gate, and circuit). An example Y-chart is shown in Fig. 1.3.

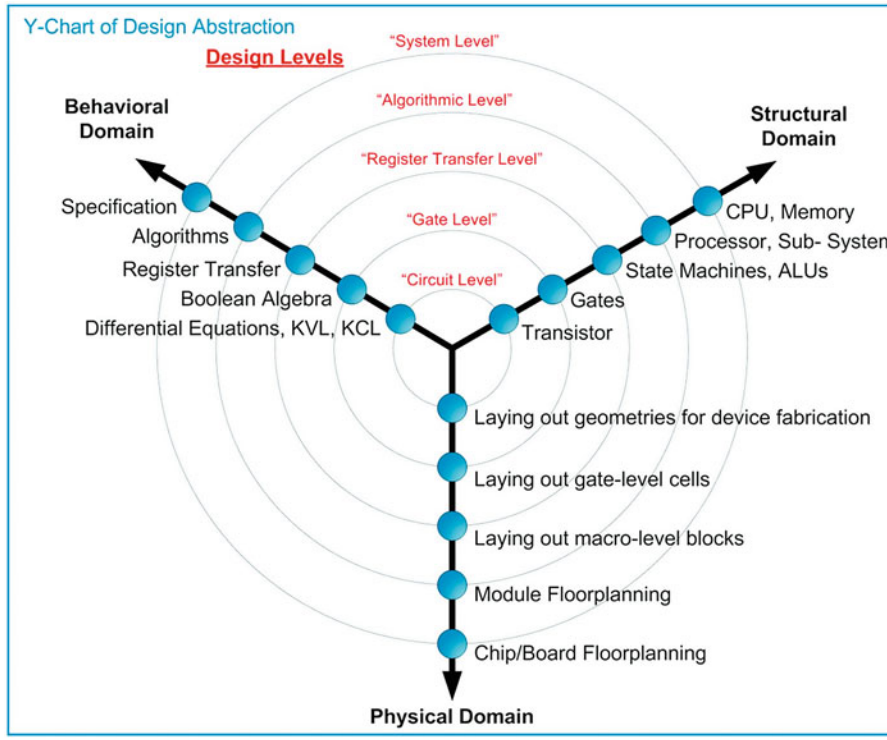

Fig. 1.3
Y-chart of design abstraction

A Y-chart also depicts how the abstraction levels of different design domains are related to each other. A top-down design flow can be visualized in a Y-chart by spiraling inward in a clockwise direction. Moving from the behavioral domain to the structural domain is the process of *synthesis*. Whenever synthesis is performed, the resulting system should be compared with the prior behavioral description. This checking is called *verification*. The process of creating the physical circuitry corresponding to the structural description is called *implementation.* The spiral continues down through the levels of abstraction until the design is implemented at a level that the geometries representing circuit elements (transistors, wires, etc.) are ready to be fabricated in silicon. Figure 1.4 shows the top-down design process depicted as an inward spiral on the Y-chart.

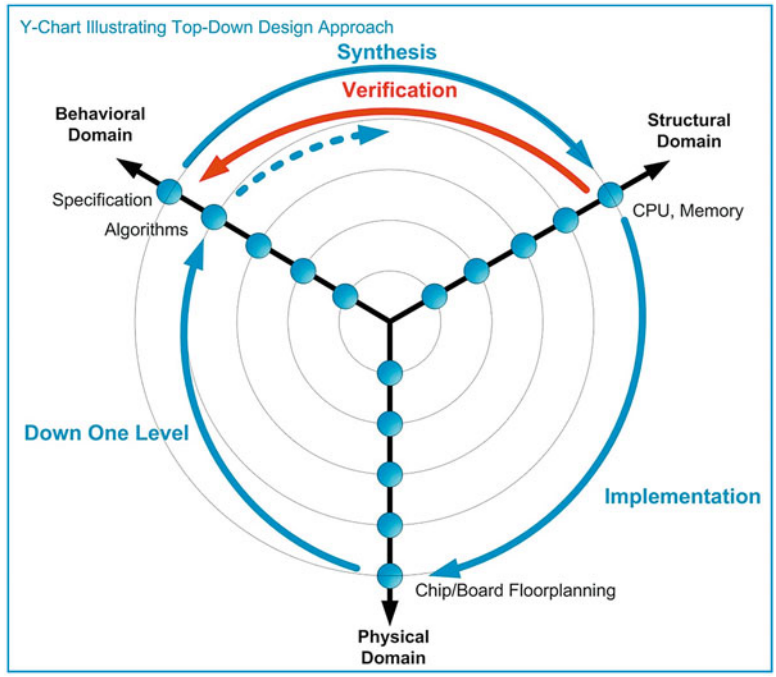

Fig. 1.4
Y-chart illustrating top-down design approach

The Y-chart represents a formal approach for large digital systems. For large systems that are designed by teams of engineers, it is critical that a formal, top-down design process is followed to eliminate potentially costly design errors as the implementation is carried out at lower levels of abstraction.

Concept Check

CC1.2 Why is abstraction an essential part of engineering design?

(A) Without abstraction all schematics would be drawn at the transistor level.

(B) Abstraction allows computer programs to aid in the design process.

(C) Abstraction allows the details of the implementation to be hidden while the higher-level systems are designed. Without abstraction, the details of the implementation would overwhelm the designer.

(D) Abstraction allows analog circuit designers to include digital blocks in their systems.

1.3 The Modern Digital Design Flow

When performing a smaller design or the design of fully contained subsystems, the process can be broken down into individual steps. These steps are shown in Fig. 1.5. This process is given generically and applies to both *classical* and *modern* digital design. The distinction between classical and modern is that modern digital design uses HDLs and automated CAD tools for simulation, synthesis, place and route, and verification.

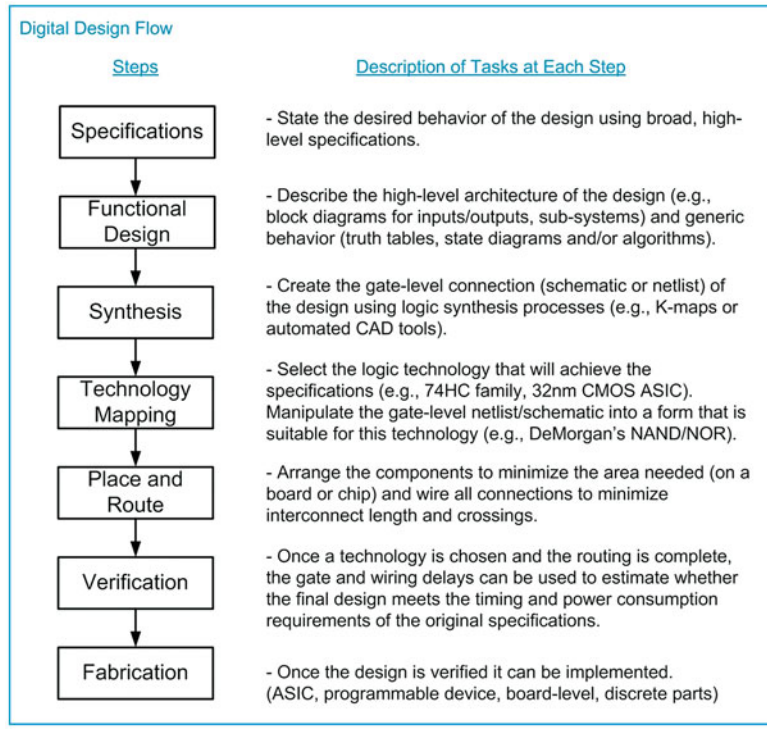

Fig. 1.5
Generic digital design flow

This generic design process flow can be used across classical and modern digital design, although modern digital design allows additional verification at each step using automated CAD tools. Figure 1.6 shows how this flow is used in the classical design approach of a combinational logic circuit.

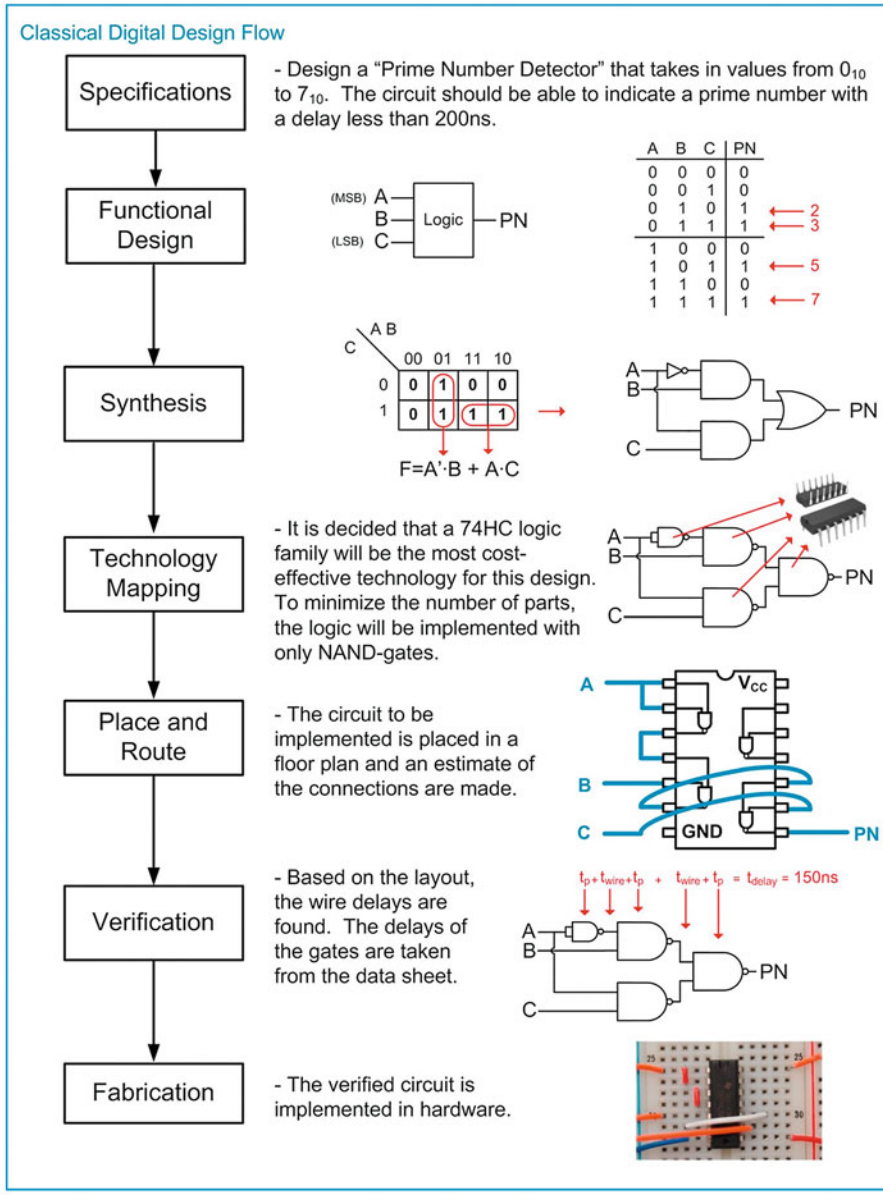

Fig. 1.6
Classical digital design flow

The modern design flow based on HDLs includes the ability to simulate functionality at each step of the process. Functional simulations can be performed on the initial behavioral description of the system. At each step of the design process the functionality is described in more detail, ultimately moving toward the fabrication step. At each level, the detailed information can be included in the simulation to verify that the functionality is still correct and that the design is still meeting the original specifications. Figure 1.7 shows the modern digital design flow with the inclusion of simulation capability at each step.

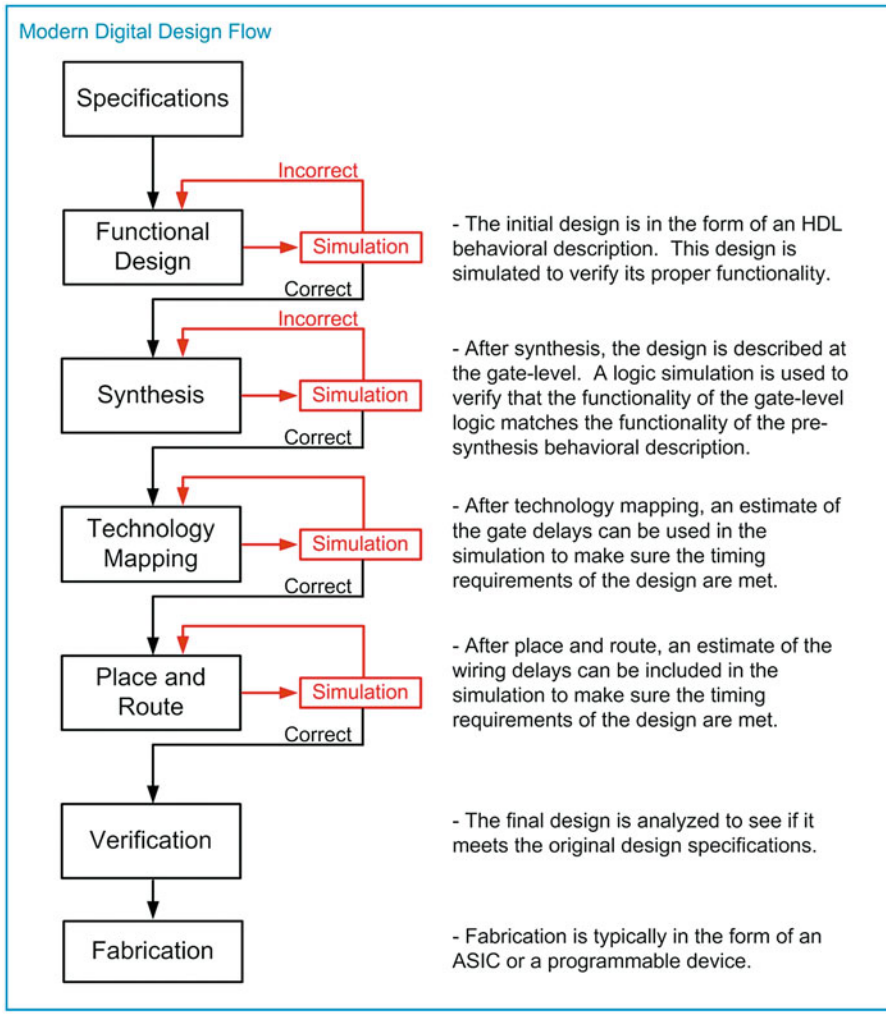

Fig. 1.7
Modern digital design flow

Concept Check

CC1.3 Why did digital designs move from schematic entry to text-based HDLs?

(A) HDL models could be much larger by describing functionality in text similar to traditional programming language.

(B) Schematics required sophisticated graphics hardware to display correctly.

(C) Schematics symbols became too small as designs became larger.

(D) Text was easier to understand by a broader range of engineers.

Summary

- The modern digital design flow relies on computer-aided engineering (CAE) and computer-aided design (CAD) tools to manage the size and complexity of today's digital designs.
- Hardware description languages (HDLs) allow the functionality of digital systems to be entered using text. VHDL and Verilog are the two most common HDLs in use today.
- Verilog was originally created to support functional simulation of text-based designs.
- The ability to automatically synthesize a logic circuit from a Verilog behavioral description became possible approximately 10 years after the original definition of Verilog. As such, only a subset of the behavioral modeling techniques in Verilog can be automatically synthesized.
- HDLs can model digital systems at different levels of design abstraction. These include the *system*, *algorithmic*, *RTL*, *gate*, and *circuit* levels. Designing at a higher level of abstraction allows more complex systems to be modeled without worrying about the details of the implementation.

Exercise Problems

Section 1.1: History of HDLs

1.1.1 What was the original purpose of Verilog?

1.1.2 Can all of the functionality that can be described in Verilog be simulated?

1.1.3 Can all of the functionality that can be described in Verilog be synthesized?

Section 1.2: HDL Abstraction

1.2.1 Give the level of design abstraction that the following statement relates to: *if there is ever an error in the system, it should return to the reset state.*

1.2.2 Give the level of design abstraction that the following statement relates to: *once the design is implemented in a sum of products form, DeMorgan's Theorem will be used to convert it to a NAND-gate only implementation*.

1.2.3 Give the level of design abstraction that the following statement relates to: *the design will be broken down into two subsystems, one that will handle data collection and the other that will control data flow*.

1.2.4 Give the level of design abstraction that the following statement relates to: *the interconnect on the IC should be changed from aluminum to copper to achieve the performance needed in this design*.

1.2.5 Give the level of design abstraction that the following statement relates to: *the MOSFETs need to be able to drive at least eight other loads in this design.*

1.2.6 Give the level of design abstraction that the following statement relates to: *this system will contain 1 host computer and support up to 1000 client computers.*

1.2.7 Give the design domain that the following activity relates to: *drawing the physical layout of the CPU will require 6 months of engineering time*.

1.2.8 Give the design domain that the following activity relates to: *the CPU will be connected to four banks of memory.*

1.2.9 Give the design domain that the following activity relates to: *the fan-in specifications for this logic family require excessive logic circuitry to be used*.

1.2.10 Give the design domain that the following activity relates to: *the performance specifications for this system require 1 TFLOP at <5 W.*

Section 1.3: The Modern Digital Design Flow

1.3.1 Which step in the modern digital design flow does the following statement relate to: *a CAD tool will convert the behavioral model into a gate-level description of functionality.*

1.3.2 Which step in the modern digital design flow does the following statement relate to: *after realistic gate and wiring delays are determined, one last simulation should be performed to make sure the design meets the original timing requirements.*

1.3.3 Which step in the modern digital design flow does the following statement relate to: *if the memory is distributed around the perimeter of the CPU, the wiring density will be minimized.*

1.3.4 Which step in the modern digital design flow does the following statement relate to: *the design meets all requirements, so now I'm building the hardware that will be shipped.*

1.3.5 Which step in the modern digital design flow does the following statement relate to: *the system will be broken down into three subsystems with the following behaviors.*

1.3.6 Which step in the modern digital design flow does the following statement relate to: *this system needs to have 10 Gbytes of memory.*

1.3.7 Which step in the modern digital design flow does the following statement relate to: *to meet the power requirements, the gates will be implemented in the 74HC logic family.*