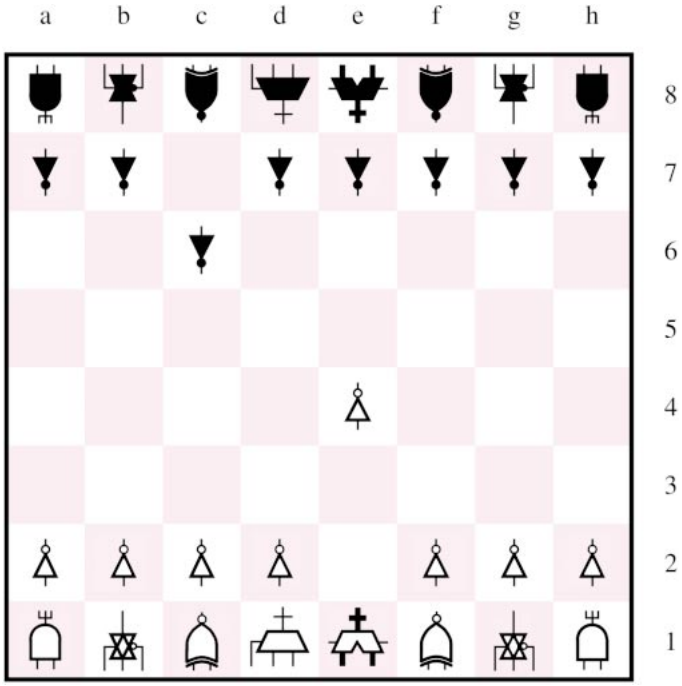
chapter

1

DESIGN CONCEPTS



1. e2–e4, c7–c6

This book is about logic circuits—the circuits from which computers are built. Proper understanding of logic circuits is vital for today's electrical and computer engineers. These circuits are the key ingredient of computers and are also used in many other applications. They are found in commonly used products, such as digital watches, various household appliances, CD players, and electronic games, as well as in large systems, such as the equipment for telephone and television networks.

The material in this book will introduce the reader to the many issues involved in the design of logic circuits. It explains the key ideas with simple examples and shows how complex circuits can be derived from elementary ones. We cover the classical theory used in the design of logic circuits in great depth because it provides the reader with an intuitive understanding of the nature of such circuits. But throughout the book we also illustrate the modern way of designing logic circuits, using sophisticated *computer aided design (CAD)* software tools. The CAD methodology adopted in the book is based on the industry-standard design language called VHDL. Design with VHDL is first introduced in Chapter 2, and usage of VHDL and CAD tools is an integral part of each chapter in the book.

Logic circuits are implemented electronically, using transistors on an integrated circuit chip. With modern technology it is possible to fabricate chips that contain tens of millions of transistors, as in the case of computer processors. The basic building blocks for such circuits are easy to understand, but there is nothing simple about a circuit that contains tens of millions of transistors. The complexity that comes with the large size of logic circuits can be handled successfully only by using highly organized design techniques. We introduce these techniques in this chapter, but first we briefly describe the hardware technology used to build logic circuits.

1.1 DIGITAL HARDWARE

Logic circuits are used to build computer hardware, as well as many other types of products. All such products are broadly classified as *digital hardware*. The reason that the name *digital* is used will become clear later in the book—it derives from the way in which information is represented in computers, as electronic signals that correspond to digits of information.

The technology used to build digital hardware has evolved dramatically over the past four decades. Until the 1960s logic circuits were constructed with bulky components, such as transistors and resistors that came as individual parts. The advent of integrated circuits made it possible to place a number of transistors, and thus an entire circuit, on a single chip. In the beginning these circuits had only a few transistors, but as the technology improved they became larger. Integrated circuit chips are manufactured on a silicon wafer, such as the one shown in Figure 1.1. The wafer is cut to produce the individual chips, which are then placed inside a special type of chip package. By 1970 it was possible to implement all circuitry needed to realize a microprocessor on a single chip. Although early microprocessors had modest computing capability by today's standards, they opened the door for the information processing revolution by providing the means for implementation of affordable personal computers. About 30 years ago Gordon Moore, chairman of Intel Corporation, observed that integrated circuit technology was progressing at an astounding rate, doubling the number of transistors that could be placed on a chip every 1.5 to 2 years.

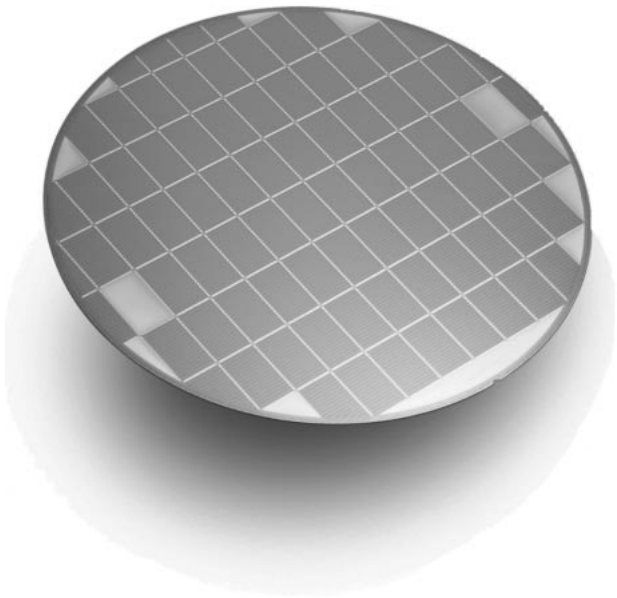


Figure 1.1 A silicon wafer (courtesy of Altera Corp.).

This phenomenon, informally known as *Moore's law*, continues to the present day. Thus in the early 1990s microprocessors could be manufactured with a few million transistors, and by the late 1990s it has become possible to fabricate chips that contain more than 10 million transistors.

Moore's law is expected to continue to hold true for at least the next decade. A consortium of integrated circuit manufacturers called the Semiconductor Industry Association (SIA) produces an estimate of how the technology is expected to evolve. Known as the *SIA Roadmap* [1], this estimate predicts the minimum size of a transistor that can be fabricated on an integrated circuit chip. The size of a transistor is measured by a parameter called its *gate length*, which we will discuss in Chapter 3. A sample of the SIA Roadmap is given in Table 1.1. In 1999 the minimum possible gate length that can be reliably manufactured is 0.14 μm. The first row of the table indicates that the minimum gate length is expected to reduce steadily to about 0.035 μm by the year 2012. The size of a transistor determines how many transistors can be placed in a given amount of chip area, with the current maximum being about 14 million transistors per cm^2. This number is expected to grow to 100 million transistors by the year 2012. The largest chip size is expected to be about 1300 mm^2 at that time; thus chips with up to 1.3 billion transistors will be possible! There is no doubt that this technology will have a huge impact on all aspects of people's lives.

The designer of digital hardware may be faced with designing logic circuits that can be implemented on a single chip or, more likely, designing circuits that involve a number of chips placed on a *printed circuit board (PCB)*. Frequently, some of the logic circuits can be realized in existing chips that are readily available. This situation simplifies the design task and shortens the time needed to develop the final product. Before we discuss the design

Table 1.1 A sample of the SIA Roadmap

	Year					
	1999	**2001**	**2003**	**2006**	**2009**	**2012**
Transistor gate length	0.14 μm	0.12 μm	0.10 μm	0.07 μm	0.05 μm	0.035 μm
Transistors per cm^2	14 million	16 million	24 million	40 million	64 million	100 million
Chip size	800 mm^2	850 mm^2	900 mm^2	1000 mm^2	1100 mm^2	1300 mm^2

process in more detail, we should introduce the different types of integrated circuit chips that may be used.

There exists a large variety of chips that implement various functions that are useful in the design of digital hardware. The chips range from very simple chips with low functionality to extremely complex chips. For example, a digital hardware product may require a microprocessor to perform some arithmetic operations, memory chips to provide storage capability, and interface chips that allow easy connection to input and output devices. Such chips are available from various vendors.

For most digital hardware products, it is also necessary to design and build some logic circuits from scratch. For implementing these circuits, three main types of chips may be used: standard chips, programmable logic devices, and custom chips. These are discussed next.

1.1.1 STANDARD CHIPS

Numerous chips are available that realize some commonly used logic circuits. We will refer to these as *standard chips*, because they usually conform to an agreed-upon standard in terms of functionality and physical configuration. Each standard chip contains a small amount of circuitry (usually involving fewer than 100 transistors) and performs a simple function. To build a logic circuit, the designer chooses the chips that perform whatever functions are needed and then defines how these chips should be interconnected to realize a larger logic circuit.

Standard chips were popular for building logic circuits until the early 1980s. However, as integrated circuit technology improved, it became inefficient to use valuable space on PCBs for chips with low functionality. Another drawback of standard chips is that the functionality of each chip is fixed and cannot be changed.

1.1.2 PROGRAMMABLE LOGIC DEVICES

In contrast to standard chips that have fixed functionality, it is possible to construct chips that contain circuitry that can be configured by the user to implement a wide range of different logic circuits. These chips have a very general structure and include a collec-

Figure 1.2 A field-programmable gate array chip (courtesy of Altera Corp.).

tion of *programmable switches* that allow the internal circuitry in the chip to be configured in many different ways. The designer can implement whatever functions are needed for a particular application by choosing an appropriate configuration of the switches. The switches are programmed by the end user, rather than when the chip is manufactured. Such chips are known as *programmable logic devices (PLDs)*. We will introduce them in Chapter 3.

Most types of PLDs can be programmed multiple times. This capability is advantageous because a designer who is developing a prototype of a product can program a PLD to perform some function, but later, when the prototype hardware is being tested, can make corrections by reprogramming the PLD. Reprogramming might be necessary, for instance, if a designed function is not quite as intended or if new functions are needed that were not contemplated in the original design.

PLDs are available in a wide range of sizes. They can be used to realize much larger logic circuits than a typical standard chip can realize. Because of their size and the fact that they can be tailored to meet the requirements of a specific application, PLDs are widely used today. One of the most sophisticated types of PLD is known as a *field-programmable gate array (FPGA)*. FPGAs that contain more than 100 million transistors will soon be available [2,3]. A photograph of an FPGA chip that has 10 million transistors is shown in Figure 1.2. The chip consists of a large number of small logic circuit elements, which can be connected together using the programmable switches. The logic circuit elements are arranged in a regular two-dimensional structure.

1.1.3 CUSTOM-DESIGNED CHIPS

PLDs are available as off-the-shelf components that can be purchased from different suppliers. Because they are programmable, they can be used to implement most logic circuits found in digital hardware. However, PLDs also have a drawback in that the programmable switches consume valuable chip area and limit the speed of operation of implemented cir-

cuits. Thus in some cases PLDs may not meet the desired performance or cost objectives. In such situations it is possible to design a chip from scratch; namely, the logic circuitry that must be included on the chip is designed first and then an appropriate technology is chosen to implement the chip. Finally, the chip is manufactured by a company that has the fabrication facilities. This approach is known as *custom* or *semi-custom design*, and such chips are called *custom* or *semi-custom chips*. Such chips are intended for use in specific applications and are sometimes called *application-specific integrated circuits (ASICs)*.

The main advantage of a custom chip is that its design can be optimized for a specific task; hence it usually leads to better performance. It is possible to include a larger amount of logic circuitry in a custom chip than would be possible in other types of chips. The cost of producing such chips is high, but if they are used in a product that is sold in large quantities, then the cost per chip, amortized over the total number of chips fabricated, may be lower than the total cost of off-the-shelf chips that would be needed to implement the same function(s). Moreover, if a single chip can be used instead of multiple chips to achieve the same goal, then a smaller area is needed on a PCB that houses the chips in the final product. This results in a further reduction in cost.

A disadvantage of the custom-design approach is that manufacturing a custom chip often takes a considerable amount of time, on the order of months. In contrast, if a PLD can be used instead, then the chips are programmed by the end user and no manufacturing delays are involved.

1.2 THE DESIGN PROCESS

The availability of computer-based tools has greatly influenced the design process in a wide variety of design environments. For example, designing an automobile is similar in the general approach to designing a furnace or a computer. Certain steps in the development cycle must be performed if the final product is to meet the specified objectives. We will start by introducing a typical development cycle in the most general terms. Then we will focus on the particular aspects that pertain to the design of logic circuits.

The flowchart in Figure 1.3 depicts a typical development process. We assume that the process is to develop a product that meets certain expectations. The most obvious requirements are that the product must function properly, that it must meet an expected level of performance, and that its cost should not exceed a given target.

The process begins with the definition of product specifications. The essential features of the product are identified, and an acceptable method of evaluating the implemented features in the final product is established. The specifications must be tight enough to ensure that the developed product will meet the general expectations, but should not be unnecessarily constraining (that is, the specifications should not prevent design choices that may lead to unforeseen advantages).

From a complete set of specifications, it is necessary to define the general structure of an initial design of the product. This step is difficult to automate. It is usually performed by a human designer because there is no clear-cut strategy for developing a product's overall structure—it requires considerable design experience and intuition.

After the general structure is established, CAD tools are used to work out the details. Many types of CAD tools are available, ranging from those that help with the design

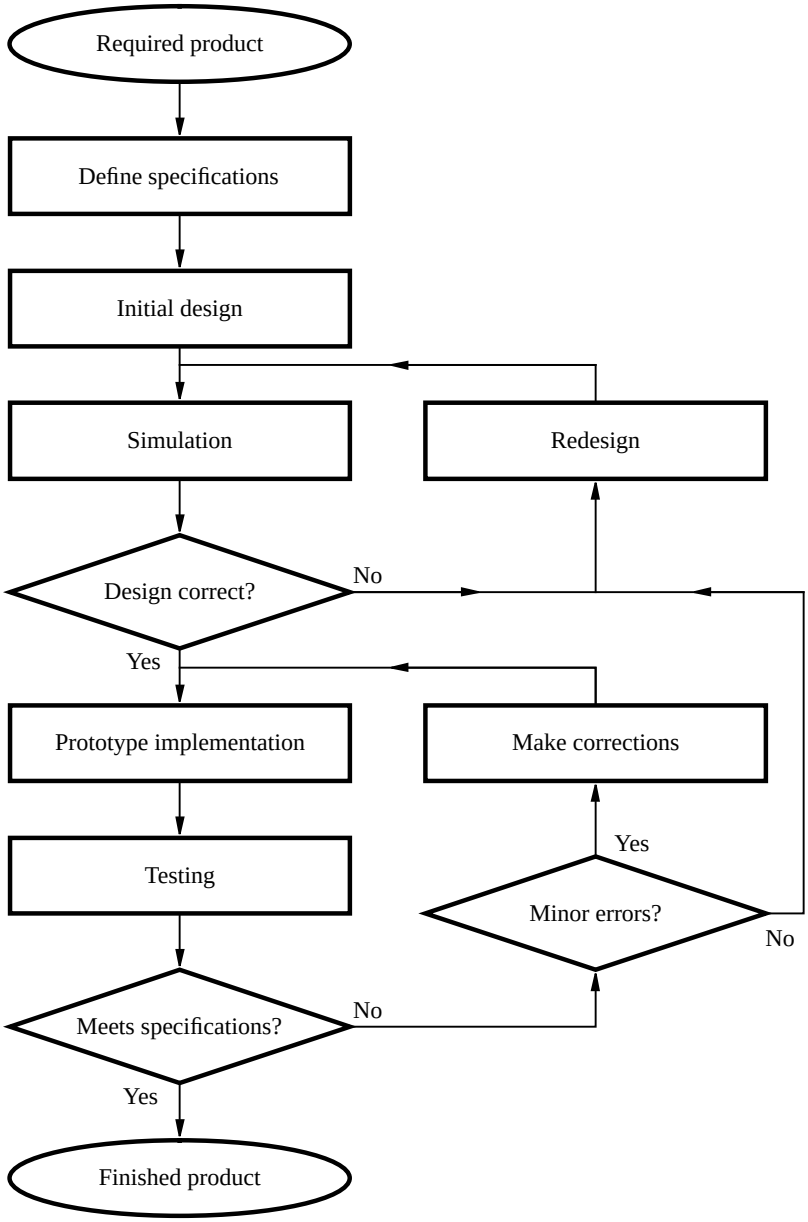


Figure 1.3 The development process.

of individual parts of the system to those that allow the entire system's structure to be represented in a computer. When the initial design is finished, the results must be verified against the original specifications. Traditionally, before the advent of CAD tools, this step involved constructing a physical model of the designed product, usually including just the key parts. Today it is seldom necessary to build a physical model. CAD tools enable

designers to simulate the behavior of incredibly complex products, and such simulations are used to determine whether the obtained design meets the required specifications. If errors are found, then appropriate changes are made and the verification of the new design is repeated through simulation. Although some design flaws may escape detection via simulation, usually all but the most subtle problems are discovered in this way.

When the simulation indicates that the design is correct, a complete physical prototype of the product is constructed. The prototype is thoroughly tested for conformance with the specifications. Any errors revealed in the testing must be fixed. The errors may be minor, and often they can be eliminated by making small corrections directly on the prototype of the product. In case of large errors, it is necessary to redesign the product and repeat the steps explained above. When the prototype passes all the tests, then the product is deemed to be successfully designed and it can go into production.

1.3 DESIGN OF DIGITAL HARDWARE

Our previous discussion of the development process is relevant in a most general way. The steps outlined in Figure 1.3 are fully applicable in the development of digital hardware. Before we discuss the complete sequence of steps in this development environment, we should emphasize the iterative nature of the design process.

1.3.1 BASIC DESIGN LOOP

Any design process comprises a basic sequence of tasks that are performed in various situations. This sequence is presented in Figure 1.4. Assuming that we have an initial concept about what should be achieved in the design process, the first step is to generate an initial design. This step often requires a lot of manual effort because most designs have some specific goals that can be reached only through the designer's knowledge, skill, and intuition. The next step is the simulation of the design at hand. There exist excellent CAD tools to assist in this step. To carry out the simulation successfully, it is necessary to have adequate input conditions that can be applied to the design that is being simulated and later to the final product that has to be tested. Applying these input conditions, the simulator tries to verify that the designed product will perform as required under the original product specifications. If the simulation reveals some errors, then the design must be changed to overcome the problems. The redesigned version is again simulated to determine whether the errors have disappeared. This loop is repeated until the simulation indicates a successful design. A prudent designer expends considerable effort to remedy errors during simulation because errors are typically much harder to fix if they are discovered late in the design process. Even so, some errors may not be detected during simulation, in which case they have to be dealt with in later stages of the development cycle.

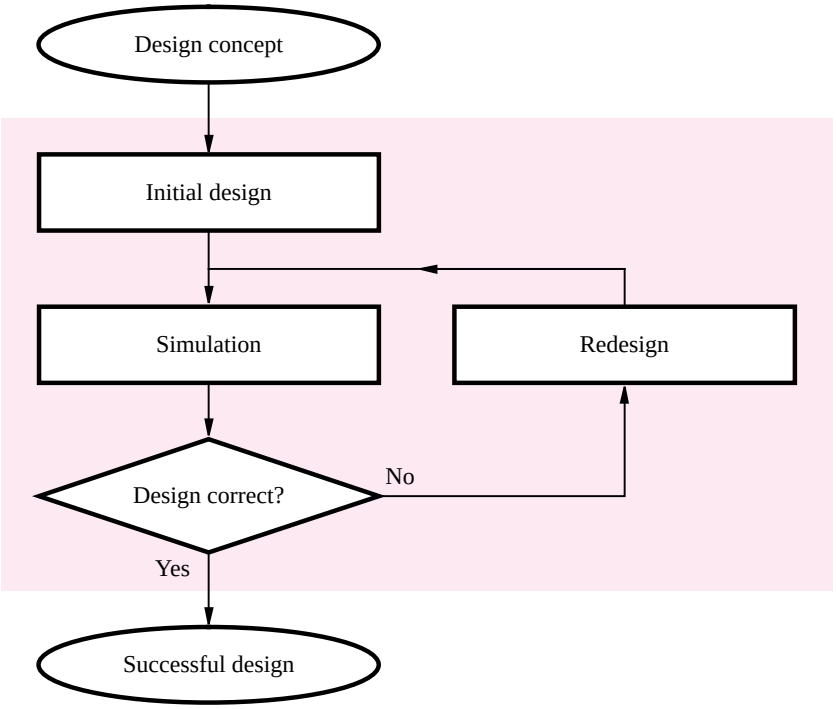


Figure 1.4 The basic design loop.

1.3.2 DESIGN OF A DIGITAL HARDWARE UNIT

Digital hardware products usually involve one or more PCBs that contain many chips and other components. Development of such products starts with the definition of the overall structure. Then the required integrated circuit chips are selected, and the PCBs that house and connect the chips together are designed. If the selected chips include PLDs or custom chips, then these chips must be designed before the PCB-level design is undertaken. Since the complexity of circuits implemented on individual chips and on the circuit boards is usually very high, it is essential to make use of good CAD tools.

An example of a PCB is given in Figure 1.5. The PCB is a part of a large computer system designed at the University of Toronto. This computer, called *NUMAchine* [4,5], is a *multiprocessor*, which means that it contains many processors that can be used together to work on a particular task. The PCB in the figure contains one processor chip and various memory and support chips. Complex logic circuits are needed to form the interface between the processor and the rest of the system. A number of PLDs are used to implement these logic circuits.

To illustrate the complete development cycle in more detail, we will consider the steps needed to produce a digital hardware unit that can be implemented on a PCB. This hardware

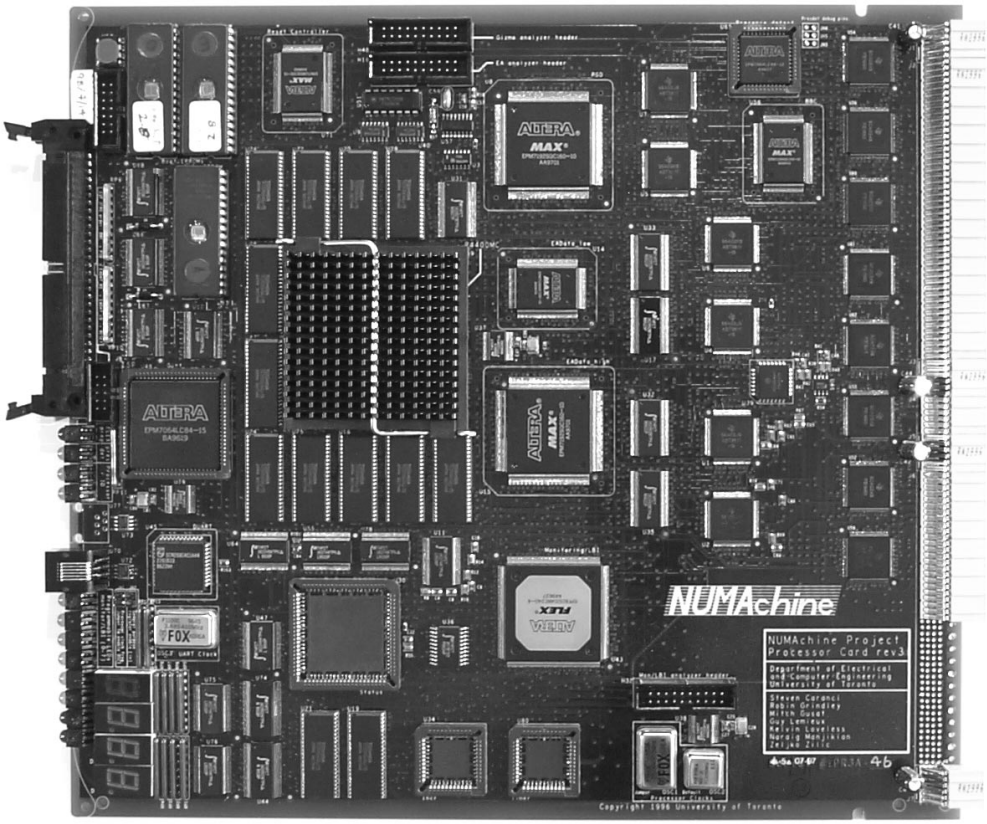


Figure 1.5 A printed circuit board.

could be viewed as a very complex logic circuit that performs the functions defined by the product specifications. Figure 1.6 shows the design flow, assuming that we have a design concept that defines the expected behavior and characteristics of this large circuit.

An orderly way of dealing with the complexity involved is to partition the circuit into smaller blocks and then to design each block separately. Breaking down a large task into more manageable smaller parts is known as the divide-and-conquer approach. The design of each block follows the procedure outlined in Figure 1.4. The circuitry in each block is defined, and the chips needed to implement it are chosen. The operation of this circuitry is simulated, and any necessary corrections are made.

Having successfully designed all blocks, the interconnection between the blocks must be defined, which effectively combines these blocks into a single large circuit. Now it is necessary to simulate this complete circuit and correct any errors. Depending on the errors encountered, it may be necessary to go back to the previous steps as indicated by the paths A, B, and C in the flowchart. Some errors may be caused by incorrect connections

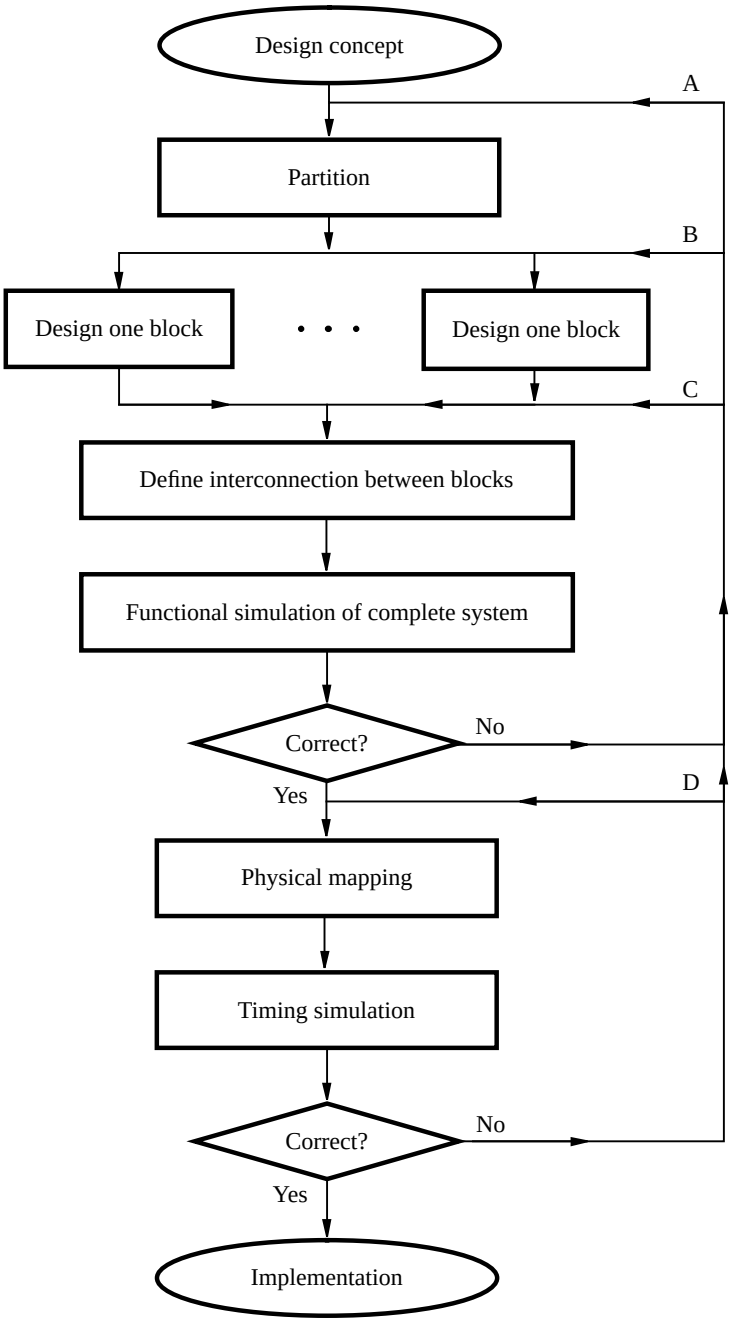


Figure 1.6 Design flow for logic circuits.

between the blocks, in which case these connections have to be redefined, following path C. Some blocks may not have been designed correctly, in which case path B is followed and the erroneous blocks are redesigned. Another possibility is that the very first step of partitioning the overall large circuit into blocks was not done well, in which case path A is followed. This may happen, for example, if none of the blocks implement some functionality needed in the complete circuit.

Successful completion of functional simulation suggests that the designed circuit will correctly perform all of its functions. The next step is to decide how to realize this circuit on a PCB. The physical location of each chip on the board has to be determined, and the wiring pattern needed to make connections between the chips has to be defined. We refer to this step as the *physical design* of the PCB. CAD tools are relied on heavily to perform this task automatically.

Once the placement of chips and the actual wire connections on the PCB have been established, it is desirable to see how this physical layout will affect the performance of the circuit on the finished board. It is reasonable to assume that if the previous functional simulation indicated that all functions will be performed correctly, then the CAD tools used in the physical design step will ensure that the required functional behavior will not be corrupted by placing the chips on the board and wiring them together to realize the final circuit. However, even though the functional behavior may be correct, the realized circuit may operate more slowly than desired and thus lead to inadequate performance. This condition occurs because the physical wiring on the PCB involves metal traces that present resistance and capacitance to electrical signals and thus may have a significant impact on the speed of operation. To distinguish between simulation that considers only the functionality of the circuit and simulation that also considers timing behavior, it is customary to use the terms *functional simulation* and *timing simulation*. A timing simulation may reveal potential performance problems, which can then be corrected by using the CAD tools to make changes in the physical design of the PCB.

Having completed the design process, the designed circuit is ready for physical implementation. The steps needed to implement a prototype board are indicated in Figure 1.7. A first version of the board is built and tested. Most minor errors that are detected can usually be corrected by making changes directly on the prototype board. This may involve changes in wiring or perhaps reprogramming some PLDs. Larger problems require a more substantial redesign. Depending on the nature of the problem, the designer may have to return to any of the points A, B, C, or D in the design process of Figure 1.6.

We have described the development process where the final circuit is implemented using many chips on a PCB. The material presented in this book is directly applicable to this type of design problem. However, for practical reasons the design examples that appear in the book are relatively small and can be realized in a single integrated circuit, either a custom-designed chip or a PLD. All the steps in Figure 1.6 are relevant in this case as well, with the understanding that the circuit blocks to be designed are on a smaller scale.

1.4 LOGIC CIRCUIT DESIGN IN THIS BOOK

In this book we use PLDs extensively to illustrate many aspects of logic circuit design. We selected this technology because it is widely used in real digital hardware products

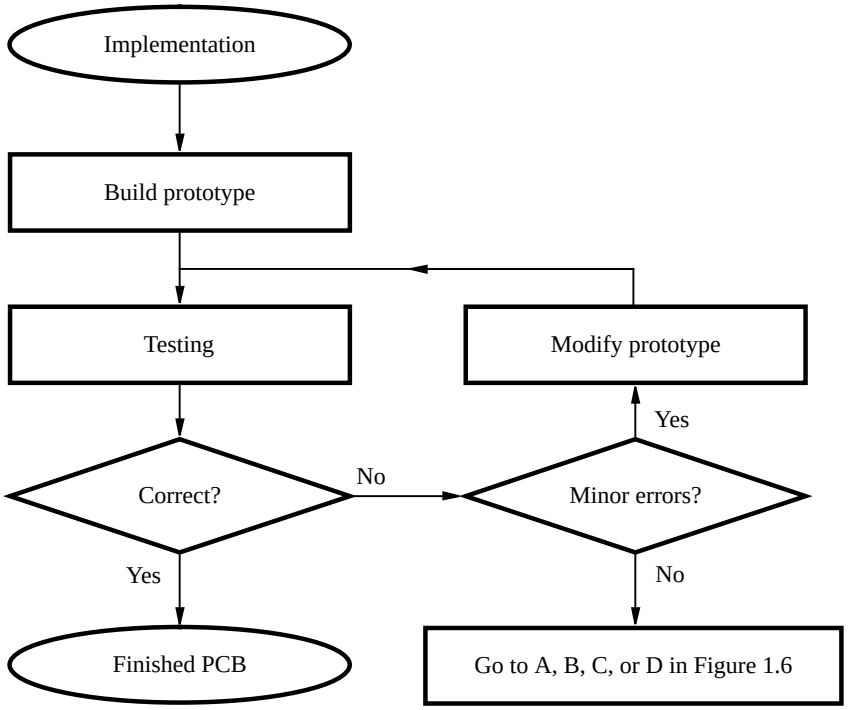


Figure 1.7 Completion of PCB development.

and because the chips are user programmable. PLD technology is particularly well suited for educational purposes because many readers have access to facilities for programming PLDs, which enables the reader to actually implement the sample circuits. To illustrate practical design issues, in this book we use two types of PLDs—they are the two types of devices that are widely used in digital hardware products today. One type is known as *complex programmable logic devices* (CPLDs) and the other as *field-programmable gate arrays* (FPGAs). These chips are introduced in Chapter 3.

We will illustrate the automated design of logic circuits using a sophisticated CAD system from Altera Corporation, one of the world's leading suppliers of PLDs. The system is called *MAX+plusII*. This industrial-quality software supports all phases of the design cycle and is powerful and easy to use. To allow the reader to obtain hands-on experience with the CAD tools, a CD-ROM containing the MAX+plusII software accompanies the book. The software is easily installed on a suitable personal computer, and we provide a sequence of complete step-by-step tutorials to illustrate the proper use of the CAD tools in concert with the book.

For educational purposes Altera provides a laboratory development PCB, which is called the UP-1 board. This PCB, shown in Figure 1.8, contains both a CPLD and an FPGA chip and has an interface for connecting the board to a personal computer. Logic circuits can be designed on the computer using MAX+plusII and then *downloaded* into the PLDs, thus realizing the designed circuit. The reader is encouraged to obtain the board from Altera,

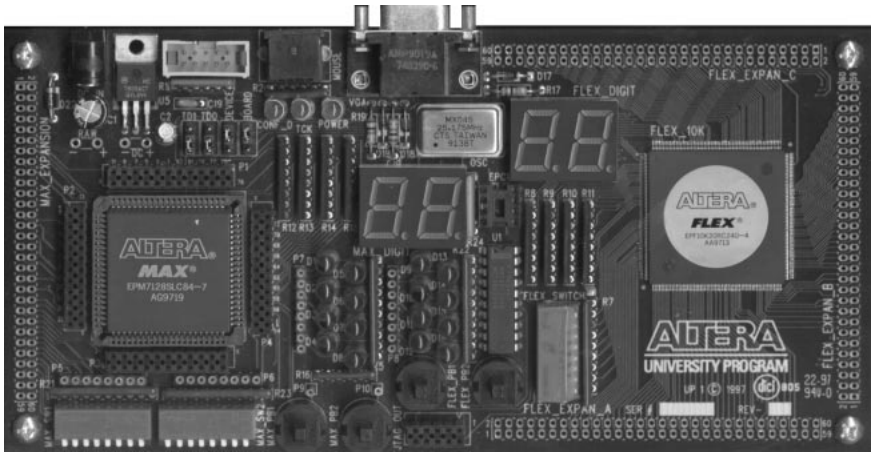


Figure 1.8 The Altera UP-1 laboratory development board.

which can be done by accessing the University Program part of Altera's Web site. All the examples of logic circuits presented in this book can be implemented on the UP-1 board.

1.5 THEORY AND PRACTICE

Modern design of logic circuits depends heavily on CAD tools, but the discipline of logic design evolved long before CAD tools were invented. This chronology is quite obvious because the very first computers were built with logic circuits, and there certainly were no computers available on which to design them!

Numerous manual design techniques have been developed to deal with logic circuits. Boolean algebra, which we will introduce in Chapter 2, was adopted as a mathematical means for representing such circuits. An enormous amount of "theory" was developed, showing how certain design issues may be treated. To be successful, a designer had to apply this knowledge in practice.

CAD tools not only made it possible to design incredibly complex circuits but also made the design work much simpler in general. They perform many tasks automatically, which may suggest that today's designer need not understand the theoretical concepts used in the tasks performed by CAD tools. An obvious question would then be, Why should one study the theory that is no longer needed for manual design? Why not simply learn how to use the CAD tools?

There are three big reasons for learning the relevant theory. First, although the CAD tools perform the automatic tasks of optimizing a logic circuit to meet particular design objectives, the designer has to give the original description of the logic circuit. If the designer specifies a circuit that has inherently bad properties, then the final circuit will also be of poor quality. Second, the algebraic rules and theorems for design and manipulation

of logic circuits are directly implemented in today's CAD tools. It is not possible for a user of the tools to understand what the tools do without grasping the underlying theory. Third, CAD tools offer many optional processing steps that a user can invoke when working on a design. The designer chooses which options to use by examining the resulting circuit produced by the CAD tools and deciding whether it meets the required objectives. The only way that the designer can know whether or not to apply a particular option in a given situation is to know what the CAD tools will do if that option is invoked—again, this implies that the designer must be familiar with the underlying theory. We discuss the classical logic circuit theory extensively in this book, because it is not possible to become an effective logic circuit designer without understanding the fundamental concepts.

On a final note, there is another good reason to learn some logic circuit theory even if it were not required for CAD tools. Simply put, it is interesting and intellectually challenging. In the modern world filled with sophisticated automatic machinery, it is tempting to rely on tools as a substitute for thinking. However, in logic circuit design, as in any type of design process, computer-based tools are not a substitute for human intuition and innovation. Computer-based tools can produce good digital hardware designs only when employed by a designer who thoroughly understands the nature of logic circuits.

REFERENCES

1. Semiconductor Industry Association, "National Technology Roadmap for Semiconductors," http://www.semichips.org/
2. Altera Corporation, "APEX 20K Advance Information Brief," http://www.altera.com
3. Xilinx Corporation, "Virtex Field Programmable Gate Arrays," http://www.xilinx.com
4. S. Brown, N. Manjikian, Z. Vranesic, S. Caranci, A. Grbic, R. Grindley, M. Gusat, K. Loveless, Z. Zilic, and S. Srbljic, "Experience in Designing a Large-Scale Multiprocessor Using Field-Programmable Devices and Advanced CAD Tools," 33rd IEEE Design Automation Conference, Las Vegas, June 1996.
5. A. Grbic, S. Brown, S. Caranci, R. Grindley, M. Gusat, G. Lemieux, K. Loveless, N. Manjikian, S. Srbljic, M. Stumm, Z. Vranesic, and Z. Zilic, " The Design and Implementation of the NUMAchine Multiprocessor," IEEE Design Automation Conference, San Francisco, June 1998.