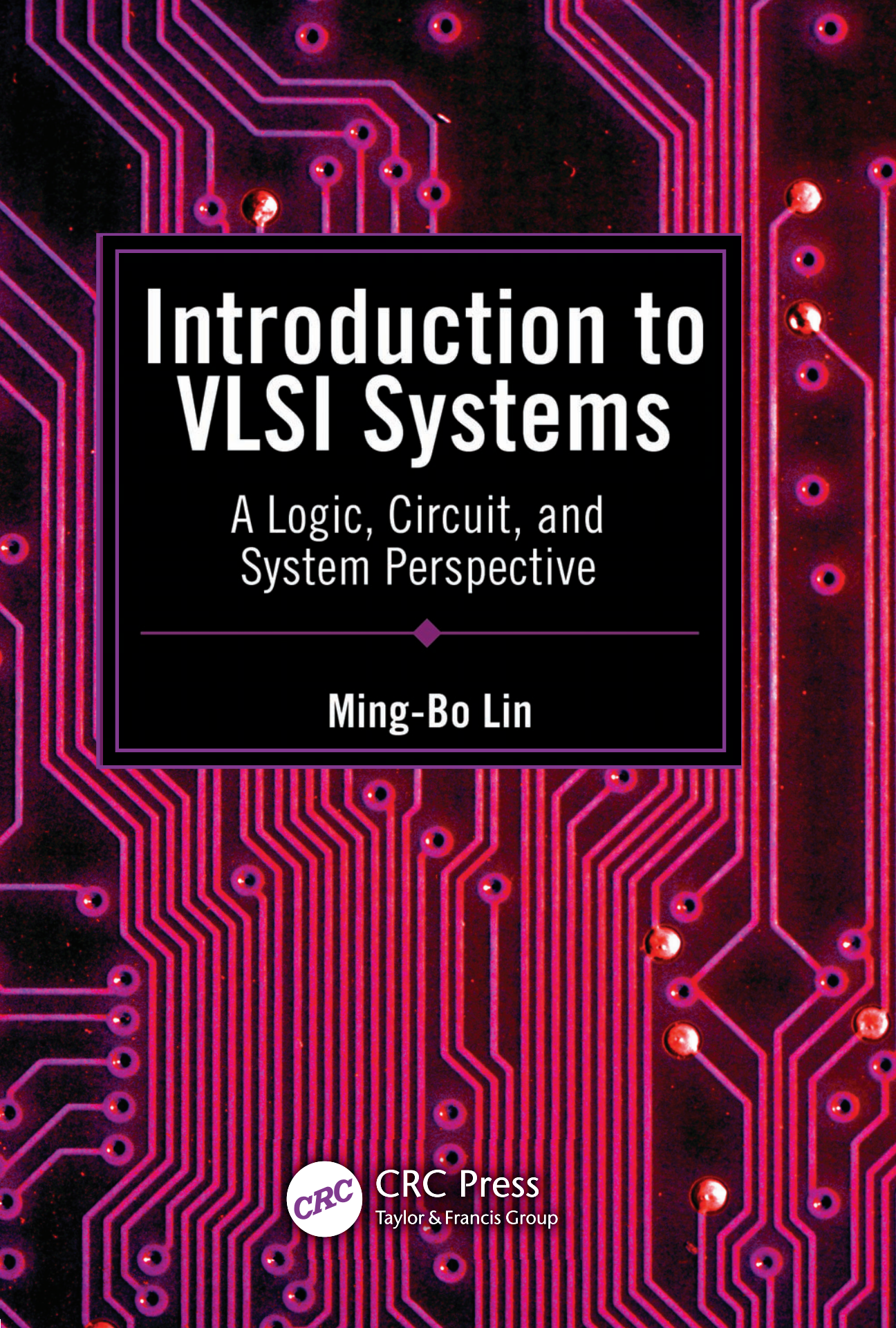

Introduction to VLSI Systems
A Logic, Circuit, and System Perspective
Ming-Bo Lin
CRC
CRC Press
Taylor & Francis Group

Introduction to VLSI Systems

A Logic, Circuit, and System Perspective

Introduction to VLSI Systems

A Logic, Circuit, and System Perspective

Ming-Bo Lin

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

To Alfred, Fanny, Alice, and Frank
and in memory of my parents

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 2011912

International Standard Book Number-13: 978-1-4398-9732-4 (eBook - PDF)

This page intentionally left blank

This page intentionally left blank

Contents

Preface

With the advance of semiconductors and the prosperous computer, consumer, as well as communication industries, the use of system-on-a-chip (SoC) has become an essential technique to reduce product cost. With this progress and continuous reduction of feature sizes, it becomes very important to understand the fundamentals of circuit and layout designs of very large-scale integration (VLSI) circuits due to the following reasons at least. First, addressing the harder problems requires fundamental understanding of circuit and layout design issues. Second, distinguished engineers can often develop their physical intuition to estimate the behavior of circuits rapidly without relying predominantly on computer-aided design (CAD) tools. This book addresses the need for teaching such a topic in terms of a logic, circuit, and system design perspective.

To achieve the above-mentioned goals, this book will focus on building an understanding of integrated circuits from the bottom up and pay much attention to both logic circuit and layout designs in addition to system designs. More precisely, this book has the following objectives. First, it is to familiarize the reader with the process of implementing a digital system as a full-custom integrated circuit. Second, it covers the principle of switch logic design and provides useful paradigms, which may apply to various static and dynamic logic families. Third, it concretely copes with the fabrication and layout designs of complementary metal-oxide-semiconductor (CMOS) VLSI. Hence, the reader will be able to design a VLSI system with the full-custom skill after reading this book. Fourth, it intends to cover the important issues of modern CMOS processes, including deep submicron devices, circuit optimization, interconnect modeling and optimization, signal integrity, power integrity, clocking and timing, power dissipation, and electrostatic discharge (ESD). As a consequence, the reader not only can comprehensively understand the features and limitations of modern VLSI technologies but also have enough background to adapt himself to this ever-changing field.

The contents of this book stem largely from the courses "*VLSI System Designs*" and "*Digital Integrated Circuits Analysis and Design,*" offered yearly at our campus over the past fifteen years. Both are elective courses for the undergraduate and first-year graduate. This book, *Introduction to VLSI: A Logic, Circuit, and System Perspective*, is intended to be useful both as a text for students and as a reference book for practicing engineers or a self-study book for readers. For classroom use, an abundance of examples are provided throughout the book for helping readers understand the basic features of full-custom VLSI and grasp the essentials of digital logic designs, digital circuit analysis and design, and system design issues as well. In addition, a lot of figures are

used to illustrate the theme concepts of the topics. Abundant review questions are also included in each chapter for helping readers test their understanding of the context.

Contents of This Book

The contents of this book are roughly partitioned into three parts. The first part covers Chapters 1 to 6 and focuses on the topics of hierarchical IC design, standard CMOS logic design, introductory physics of metal-oxide-semiconductor (MOS) transistors, device fabrication, physical layout, circuit simulation, and power dissipation as well as low-power design principles and techniques. The second part comprises Chapters 7 to 9 and deals with static logic, and dynamic logic, as well as sequential logic. The third part concentrates on system design issues and covers Chapters 10 to 16, and an appendix. This part mainly takes into account the datapath subsystem designs, memory modules, design methodologies and implementation options, interconnect, power distribution and clock designs, input/output modules and ESD protection networks, and testing as well as testable designs.

Chapter 1 introduces the features and capabilities, as well as the perspectives of VLSI systems. This chapter begins with the introduction of a brief history of integrated circuits, the challenges in VLSI design nowadays and in the near future. Then, it describes the VLSI design and fabrication, design principles and paradigms for CMOS logic circuits, as well as the design and implementation options of digital systems.

Chapter 2 deals with fundamental properties of semiconductors, characteristics of *pn* junctions, and features of MOS systems and MOS transistors. The topics of semiconductors include the differences between intrinsic and extrinsic semiconductors. The characteristics of *pn* junctions include basic structure, built-in potential, current equation, and junction capacitance. The basic operations of MOS transistors (MOSFETs) and their ideal current-voltage (I-V) characteristics are investigated in detail. In addition, the scaling theory of CMOS processes, nonideal features, threshold voltage effects, leakage current, short-channel I-V characteristics, temperature effects, and limitations of MOS transistors are also considered. Moreover, the simulation program with integrated circuit emphasis (SPICE) and related models for diodes and MOS transistors along with some examples are presented.

Chapter 3 addresses the basic manufacturing processes of semiconductor devices, including thermal oxidation, the doping process, photolithography, thin-film removal, and thin-film deposition, and their integration. In addition, the postfabrication processes, including the wafer test (or called the wafer probe), die separation, packaging, and the final test as well as the burn-in test, are discussed. Moreover, advanced packaging techniques, such as multichip module (MCM), and 3-D packaging, including system-in-package (SiP), system-on-package (SoP), and system-on-wafer (SoW), are described briefly.

Chapter 4 takes into account the interface between the fabrication process and circuit design, namely, layout design rules. The advanced layout considerations about signal integrity, manufacturability, and reliability in modern deep submicron (nanometer) processes are also considered. The latch-up problem inherent in CMOS processes is also considered in depth. After this, a widely used regular layout structure, the Euler path approach, is introduced with examples.

Chapter 5 concerns the resistance and capacitance parasitics of MOS transistors and their effects. To this end, we begin with the introduction of resistances and capacitances of MOS transistors and then consider the three common approaches to estimating the propagation delays, t_{pHL} and t_{pLH}. After this, we are concerned with cell delay and Elmore delay models. The path-delay optimization problems of logic chains composed of inverters or mixed-type logic gates follow. Finally, logical effect is

defined and its applications to the path-delay optimization problems are explored in great detail.

Chapter 6 presents the power dissipation and low-power designs. In this chapter, we first introduce the power dissipation of CMOS logic circuits and then focus on the low-power design principles and related issues. Finally, the techniques for designing power-manageable components and dynamic power management along with examples are dealt with in depth.

Chapter 7 investigates static logic circuits. For this purpose, we first consider CMOS inverters and its voltage transfer characteristics. NAND and NOR gates are then addressed. Finally, single-rail and dual-rail logic circuits along with typical examples are broadly examined. The logic design principles of these logic circuits are also investigated in depth.

Chapter 8 considers the dynamic logic circuits. In this chapter, we first examine the basic dynamic logic circuits and the partially discharged hazards and its avoidance. After this, nonideal effects of dynamic logic are described in more detail. Finally, we address three classes of dynamic logic: single-rail dynamic logic, dual-rail dynamic logic, and clocked CMOS logic.

Chapter 9 is concerned with the sequential logic circuits. This chapter begins to introduce the fundamentals of sequential logic circuits, covering the sequential logic models, basic bistable devices, and metastable states and hazards, as well as arbiters. A variety of CMOS static and dynamic memory elements, including latches, flip-flops, and pulsed latches, are then considered in depth. The timing issues of systems based on flip-flops, latches, and pulsed latches are also dealt with in detail. Finally, pipeline systems are discussed.

Chapter 10 explores the basic components widely used in datapaths. These include basic combinational and sequential components. The former comprises decoders, encoders, multiplexers, demultiplexers, magnitude comparators, and shifters and the latter consists of registers and counters. In addition, arithmetic operations, including addition, subtraction, multiplication, and division, are also dealt with in depth. An arithmetic algorithm may usually be realized by using either a multiple-cycle or single-cycle structure. The shift, addition, multiplication, and division algorithms are used as examples to repeatedly manifest the essentials of these two structures.

Chapter 11 describes a broad variety of types of semiconductor memories. The semiconductor memory can be classified in terms of the type of data access and the information retention capability. According to the type of data access, semiconductor memory can be classified into serial access, content addressable, and random access. The random-access memory can be categorized into read/write memory and read-only memory. Read/write memory can be further subdivided into two types: static random access memory (SRAM) and dynamic random access memory (DRAM). According to the information retention capability, semiconductor memory may be classified into volatile and nonvolatile memory. The volatile memory includes static RAM and dynamic RAM while the nonvolatile memory contains ROM, ferroelectric RAM (FRAM), and magnetoresistance RAM (MRAM).

Chapter 12 is concerned with the design methodologies and implementation options of VLSI or digital systems. In this chapter, we begin to describe the related design approaches at both system and register-transfer (RT) levels. Design flows, including both RT and physical levels, and implementation options for digital systems are then addressed. Finally, a case study is given to illustrate how a real-world system can be designed and implemented with a variety of options, including the μP/DSP system, a

field-programmable device, and an application-specific integrated circuit (ASIC) with a cell library.

Chapter 13 copes with interconnect and its related issues. Interconnect in a VLSI or digital system mainly provides power delivery paths, clock delivery paths, and signal delivery paths. It plays an important role in any VLSI or digital system because it controls timing, power, noise, design functionality, and reliability. All of these closely related issues are discussed in detail.

Chapter 14 deals with power distribution and clock designs. The design issues of power distribution, power distribution networks, and decoupling capacitors and their related issues are presented in detail. The main goal of a clock system is to generate and distribute one or more clocks to all sequential devices or dynamic logic circuitry in the system with as little skew as possible. For this purpose, the clock system architecture, methods used to generate clocks, and clock distribution networks are focused on. The phase-locked and delay-locked loops are also investigated.

Chapter 15 addresses input/output (I/O) modules and electrostatic discharge (ESD) protection networks. The I/O modules generally include input and output buffers. They play important roles for communicating with the outside of a chip or a VLSI system. Associated with I/O buffers are ESD protection networks that are used to create current paths for discharging the static charge caused by ESD events in order to protect the core circuits from being damaged.

The final chapter (Chapter 16) explores the topics of testability and testable design. The goal of testing is to find any existing faults in a system or a circuit. This chapter first takes a look at VLSI testing and then examines fault models, test vector generation, and testable circuit design or design for testability. After this, the boundary scan standard (IEEE 1149.1), the system-level testing, such as SRAM, a core-based system, system-on-a-chip (SoC), and IEEE 1500 standard, are briefly dealt with.

The appendix surveys some synthesizable features with examples of Verilog hardware description language (Verilog HDL) and SystemVerilog. Through the use of these examples, the reader can readily describe their own hardware modules in Verilog HDL/SystemVerilog. In addition, basic design approaches of test benches are also dealt with concisely. Finally, the complete description of the start/stop timer described in Chapter 12 is presented in the context of Verilog HDL.

Use of This Book for Courses

The author has been using the contents of this book for the following two courses for many years, "*VLSI Systems Design*" (or "*An Introduction to VLSI*") and "*Digital Integrated Circuits Analysis and Design*." The objectives of the "VLSI Systems Design" course are to familiarize the student with an understanding of how to implement a digital system as a full-custom integrated circuit, to cover the principles of switch logic design and provide useful paradigms for CMOS logic circuits, to concretely cope with the fabrication and layout designs of CMOS VLSI, and to provide the student with the fundamental understanding of modern VLSI techniques. Based on these, the reader not only can comprehensively understand the features and limitations of modern VLSI technologies but also have enough background to adapt himself to this ever-changing field. In this course, the following sections are covered. The details can be referred to in lecture notes.

- Sections 1.1 to 1.4
- Sections 3.1 to 3.3
- Sections 4.1 to 4.4
- Sections 5.1 to 5.3 and 6.1

- Sections 7.1 (7.1.2 to 7.1.4), 7.2 and 7.3
- Sections 8.1 (8.1.2 to 8.1.3), 8.3 (8.3.1 to 8.3.3), 8.4 (8.4.1 and 8.4.2), and 8.5
- Sections 9.1 (9.1.1 to 9.1.3), 9.2 (9.2.1 to 9.2.3), 9.3 , and 9.4
- Sections 10.1 to 10.6
- Optional Sections 12.1 to 12.3
- Optional Sections 16.1 to 16.5

The essential aims of the "Digital Integrated Circuits Analysis and Design" course are to introduce the student to the important issues of modern CMOS processes, including deep submicron devices, circuit optimization, memory designs, interconnect modeling and optimization, low-power designs, signal integrity, power integrity, clocking and timing, power distribution, and electrostatic discharge (ESD). To reach these goals, the following sections are covered. The details can also be referred to in lecture notes.

- Sections 2.1 to 2.5
- Sections 7.1 to 7.3
- Sections 8.1 and 8.2
- Sections 11.1 to 11.6
- Sections 13.1 to 13.4
- Sections 14.1 to 14.3
- Sections 15.1 to 15.4
- Sections 6.1 to 6.4

Of course, instructors who adopt this book are encouraged to customize their own course outlines based on the contents of this book.

Supplements

The instructor's supplements, containing a solution manual and lecture notes in PowerPoint slides, are available for all instructors who adopt this book.

Acknowledgments

Most materials of this book have been taken from the courses ET5302 and ET5006 offered yearly at our campus over the past fifteen years. Many thanks are due to the students of these two courses, who suffered through many of the experimental class offerings based on the draft of this book. Valuable comments from the participants of both courses have helped in evolving the contents of this book and are greatly appreciated. Thanks are given to National Chip Implementation Center of National Applied Research Laboratories of Taiwan, R.O.C., for their support in VLSI education and related research in Taiwan over the past two decades. I also gratefully appreciate my mentor, Ben Chen, a cofounder of Chuan Hwa Book Ltd., for his invaluable support and continuous encouragement in my academic carrier over the past decades. I would like to extend special thanks to the people at CRC Press for their efforts in producing this book: in particular, Li-Ming Leong, Joselyn Banks-Kyle, and Jim McGovern. Finally, and most sincerely, I wish to thank my wife, Fanny, and my children, Alice and Frank, for their patience in enduring my absence from them during the writing of this book.

M. B. Lin
Taipei, Taiwan

This page intentionally left blank

1

Introduction

Although nowadays most application-specific integrated circuits (ASICs) are designed fully or partially based on a specific synthesis flow using hardware description languages (HDLs) and implemented by either field-programmable gate arrays (FPGAs) or cell libraries, it is increasingly important to understand the fundamentals of circuit and physical designs of very large-scale integration (VLSI) circuits at least due to the following reasons. First, addressing the harder problems requires fundamental understanding of circuit and physical design issues. Second, distinguished engineers can often develop their physical intuition to estimate the behavior of circuits rapidly without relying on computer-aided design (CAD) tools predominantly.

To achieve the above-mentioned objectives, this book will focus on building an understanding of integrated circuits from the bottom up and pay much attention to both physical layout and logic circuit designs in addition to system designs. After all, we always believe that the best way to learn VLSI design is by doing it.

To familiarize the reader with the process of implementing a digital system as a full-custom integrated circuit, in this tutorial chapter, we will address a brief history of integrated circuits, the challenges in VLSI design now and in the near future, the perspective on the VLSI design, the VLSI design and fabrication, principles and paradigms of complementary metal-oxide-semiconductor (CMOS) logic designs, as well as the design and implementation options of digital systems.

1.1 Introduction to VLSI

We first review in this section the history of VLSI technology in brief. Next, we introduce a silicon planar process on which modern CMOS processes are founded, and ultimate limitations of feature size. Then, we are concerned with the classification of VLSI circuits, the benefits of using VLSI circuits, the appropriate technologies for manufacturing VLSI circuits, and a brief introduction to scaling theory. Finally, design challenges in terms of deep-submicron (DSM) devices and wires are also dealt with in detail. Economics and future perspective of VLSI technology are explored briefly.

1.1.1 Introduction

In this subsection, we introduce the history of VLSI technology, a basic silicon planar process, and ultimate limitations of feature size.

1.1.1.1 A Brief History The history of VLSI can be dated back to the invention of transistors. In 1947, John Bardeen, Walter Brattain, and William Shockley (all at Bell laboratories, also known as Bell Labs) invented the first *point-contact transistor*. Due to this important contribution, they were awarded the Nobel Prize in Physics in 1956. After this invention, Bell Labs devoted themselves to develop *bipolar junction transistors* (BJTs). Their efforts founded the basis of modern BJTs. BJTs soon replaced vacuum tubes and became the mainstream of electronic systems due to the fact that they are more reliable and less noisy, and consume less power.

Ten years later after the invention of the transistor, Jack Kilby at Texas Instruments (TI) built the first integrated circuit, which explored the potential of the miniaturalization of building multiple transistors on a single piece of silicon. This work established the foundation of *transistor-transistor logic* (TTL) families, which were popular in the 1970s and 1980s. Some parts of them are still available and popular in use today although these circuits might not be still manufactured as their original design. Due to his invention of the integrated circuit, Jack Kilby was awarded the Nobel Prize in Physics in 2000.

Even though the *metal-oxide-semiconductor field-effect transistor* (MOSFET), or called the *metal-oxide-semiconductor* (MOS) transistor for short, was invented early before the BJT, it was not widely used until the 1970s. In 1925 and 1935, Julius Lilienfield (German) (US patent 1,745,175) and Oskar Heil (British patent 439,457), filed their patents separately. In 1963, Frank Wanlass at Fairchild built the first *complementary metal-oxide-semiconductor* (CMOS) logic gate with discrete components and demonstrated the ultra-low standby power feature of CMOS technology at the cost of two different types, n and p types, of MOS transistors. This circuit founded the CMOS realm nowadays.

1.1.1.2 Silicon Planar Processes With the advent of the *silicon planar process*, MOS integrated circuits become popular due to their low cost since each transistor occupies less area than a BJT and the manufacturing process is much simpler. The p-type MOS (pMOS) transistors were used at the onset of the MOS process but soon were replaced by n-type MOS (nMOS) transistors due to their poor performance, reliability, and yield. One advantage of nMOS logic circuits is that they need less area than their pMOS counterparts. This advantage was demonstrated by Intel in their development in nMOS technology with 1101, a 256-bit static *random-access memory* (RAM), and 4004, the first 4-bit microprocessor. Nevertheless, in the 1980s, the demand on *integration density* was rapidly increased; the high standby power dissipation of nMOS logic circuits severely limited the degree of integration. Consequently, the CMOS process emerged and rapidly took the place of the nMOS process as the mainstream of VLSI technology although the logic circuits in the CMOS process needed much more area than in the nMOS process. Now the CMOS process has become the most mature and popular technology for VLSI designs.

The essential feature of the silicon planar process is the capability of depositing selective dopant atoms on the specific regions on the surface of silicon to change or modify the electrical properties of these regions. To reach this, as illustrated in Figure 1.1 the following basic steps are performed: (1) an *oxide* (*silicon dioxide*, SiO_2) layer is formed on the surface of silicon, (2) the selective region of the oxide layer is removed, (3) desired dopant atoms are deposited on the surface of silicon and oxide layer, and (4) the dopant atoms on the selective region are diffused onto the silicon. By repeatedly using these four steps, the desired integrated circuit (IC) can be made.

The basic silicon planar process has been refined over and over again since the 1960s. The *feature size*, i.e., the allowed minimum line width or spacing between two lines for

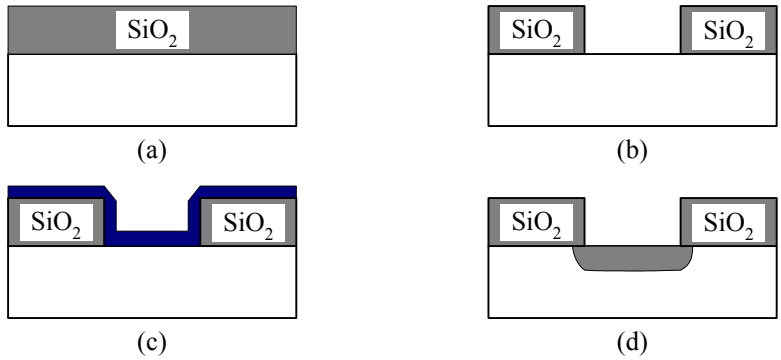


Figure 1.1: The essential steps of the silicon planar process: (a) Oxide formation; (b) selective oxide removal; (c) deposition of dopant atoms; (d) diffusion of dopant atoms.

a given process, of an IC has continued to be evolved at a rapid rate, from 8 μm in 1969 to 32 nm today (2010). With this evolution, the transistor count on a chip grows steeply, as depicted in Figure 1.2, which shows the evolution of Intel *microprocessors*. From the figure, we can see that the transistor count is almost doubled every two years. This exponential growth of transistor count per chip was first realized by Gorden Moore in 1965 and became the famous Moore's law, which states that *the transistor count per chip would double every 18 to 24 months.* Three years later, Robert Noyce and Gorden Moore founded Intel Corporation, which became the largest chip maker all over the world today.

Moore's law is actually driven by two major forces. First, the feature sizes are dramatically decreased with the refinement of equipment for integrated-circuit technology. Second, the die[1] area is increased to accommodate more transistors since the defect density of wafer has been reduced profoundly. Nowadays, dies with an area of 2×2 cm^2 are not uncommon. However, we will see that the die yield and the cost of each die are strongly dependent on the die area. To achieve a good die yield, the die area has to be controlled under a bound. The details of die yield and the more general topic, VLSI economics, will be discussed later in this section.

1.1.1.3 Ultimate Limitations of Feature Size The feature size cannot be reduced unlimitedly. At least two factors will prevent feature sizes from being decreased without limit. First of all, as the device size decreases, the statistical fluctuations in the number ($\approx \sqrt{n}$) will become an important factor to limit the performance of the circuit, not only for analog circuits but finally also for digital circuits. This will make the circuit design more difficult than before. Eventually, each device may contain only a few electrons and the entire concept of circuit design will be different from what we have been using today.

Another factor that limits the reduction of feature size indefinitely is the resolution and equipment of photolithography. Although resolution enhancement techniques such as *optical proximity correction* (OPC), *phase-shift masks*, *double patterning*, and *immersion photolithography* have enabled scaling to the present 45 nm and beyond, along the reduction of feature size, a more expensive photolithography equipment as well as the other manufacturing equipment are needed in the near future. The cost of

[1] A die usually means an integrated circuit on a wafer, while a chip denotes an integrated circuit that has been sliced from a wafer. Sometimes, they are used interchangeably.

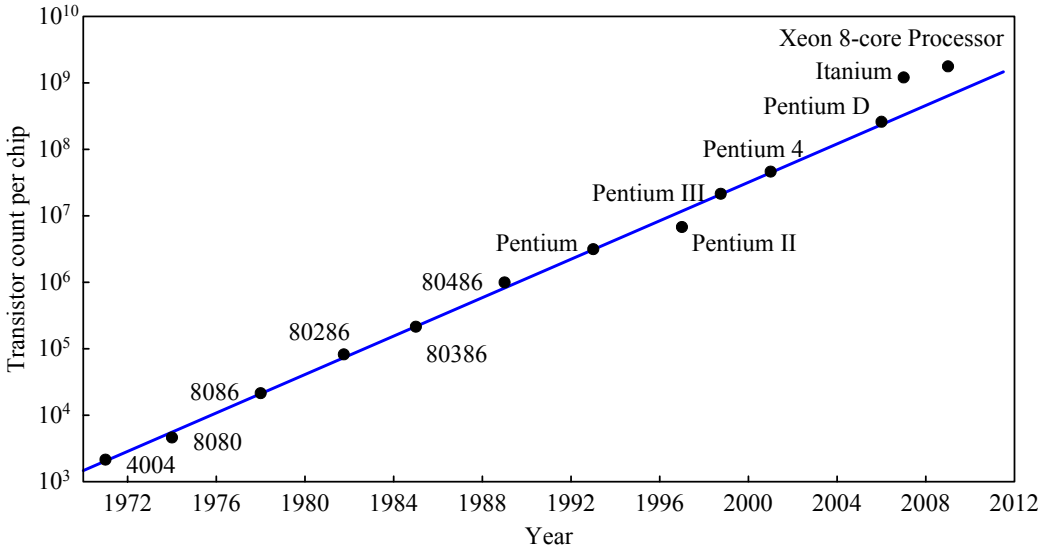


Figure 1.2: An illustration of Moore's law with the evolution of Intel microprocessors.

development and manufacturing equipment will likely limit the feature size available used to manufacture ICs.

Review Questions

Q1-1. Describe the essential steps of the silicon planar process.

Q1-2. Describe the meaning and implication of Moore's law.

Q1-3. What are the ultimate limitations of feature size?

1.1.2 Basic Features of VLSI Circuits

In this section, we are concerned with the meaning of VLSI circuits, the benefits of using VLSI circuits, the appropriate technologies for manufacturing VLSI circuits, and scaling theory.

1.1.2.1 Classification of VLSI Circuits Roughly speaking, the word "VLSI" means any integrated circuit containing a lot of components such as transistors, resistors, capacitors, and so on. More specifically, the *integrated circuits* (ICs) can be categorized into the following types in terms of the number of components that they contain.

- *Small-scale integration* (SSI) means an IC containing less than 10 gates.
- *Medium-scale integration* (MSI) means an IC containing up to 100 gates.
- *Large-scale integration* (LSI) means an IC containing up to 1000 gates.
- *Very large-scale integration* (VLSI) means an IC containing beyond 1000 gates.

Here the "gate" means a two-input basic gate such as an AND gate or a NAND gate. Another common metric to measure the complexity of an IC is the *transistor count*. Both *gate count* and transistor count in an FPGA device, a cell-based design, and a bipolar full-custom design, can be converted from one to another by using the following rule-of-thumb factors.

- *Field programmable gate array* (FPGA): In FPGA devices, due to some unavoidable overhead, 1 gate is approximately equivalent to 7 to 10 transistors, depending on the type of FPGA device under consideration.
- *CMOS cell-based design*: Since a basic two-input NAND gate consists of two pMOS and two nMOS transistors, one basic gate in cell-based design is often counted as four transistors.
- *Bipolar full-custom design*: In BJT logic, such as TTL, each basic gate roughly comprises ten basic components, including transistors and resistors.

Some popular VLSI circuit examples are as follows: microprocessors, *microcontrollers* (embedded systems), memory devices (static random access memory (SRAM), dynamic random access memory (DRAM), and *Flash memory*), various FPGA devices, and special-purpose processors, such as *digital signal processing* (DSP) and *graphics processing unit* (GPU) devices.

Recently, the term *ultra large-scale integration* (ULSI) is sometimes used to mean a single circuit capable of integrating billions of components. Nevertheless, we will not use this term in this book. Instead, we simply use the term VLSI to mean an IC with a high-degree integration of transistors and/or other components.

1.1.2.2 Benefits of Using VLSI Circuits Once we have classified the ICs, we are now in a position to consider the benefits of using VLSI circuits. In addition to the fact that integration may reduce manufacturing cost because virtually no manual assembly is required, integration significantly improves the design. This can be seen from the following consequences of integration. First, integration reduces parasitics, including capacitance, resistance, even inductance, and hence allows the resulting circuit to be operated at a higher speed. Second, integration reduces the power dissipation and hence generates less heat. A large amount of power dissipation of an IC is due to I/O circuits in which a lot of capacitors need to be charged and discharged when signals are switched. Most of these I/O circuits can be removed through proper integration. Third, the integrated system is physically smaller due to less chip area occupied than the original system. Thus, using VLSI technology to design an electronic system results in higher performance, consumes less power, and occupies less area. These factors reflect into the final product as smaller physical size, lower power dissipation, and reduced cost.

1.1.2.3 VLSI Technologies Many technologies may be used to design and implement an integrated circuit nowadays. The three most popular technologies are CMOS technology, BJT, and *gallium arsenide* (GaAs). Among these, the CMOS technology is the dominant one in the VLSI realm due to its attractive features of lowest power dissipation and highest integration density. The BJT has an inherent feature of higher operating frequency than its CMOS counterpart. Hence, BJTs are often used in *radio-frequency* (RF) applications. Modern CMOS processes may also embed the fabrication of BJTs as an extension, thereby leading to a hybrid process known as the *Bipolar-CMOS* (BiCMOS) process. To improve the performance of BJT, most modern BiCMOS processes also provide a new type of transistor, called a *silicon-germanium* (SiGe) transistor, to be used in special high-frequency applications such as RF transceivers, which need to operate up to 5 to 10 GHz. The third technology is GaAs, which is special for microwave applications needing an operating frequency up to 100 GHz.

There are two criteria that are used to evaluate whether a technology is capable of providing high-degree integration. These criteria include power dissipation and the

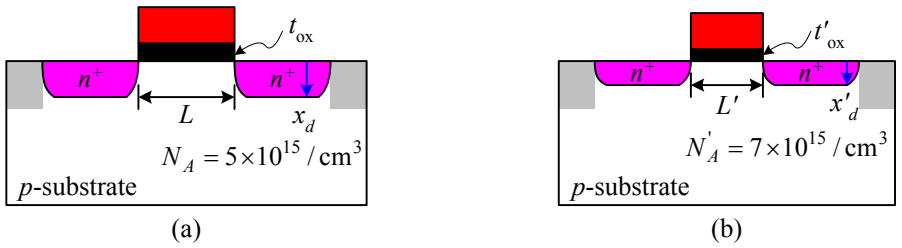


Figure 1.3: An illustration of the scaling principle (Not drawn in scale): (a) original device; (b) scaled-down device.

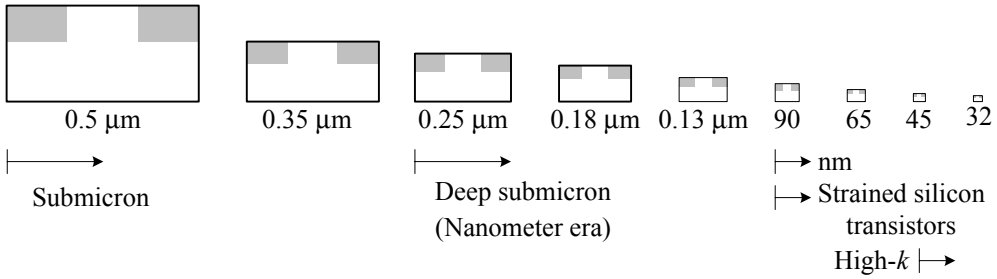


Figure 1.4: The feature-size trends of CMOS processes.

area of each transistor (or logic gate). For the time being, only CMOS technology can meet these two criteria at the same time and hence is widely used in industry to produce a broad variety of chips in the fields of communications, computers, consumer products, multimedia, and so on. However, the power dissipation problem is still the major challenge in designing such CMOS chips since the power dissipation of a single chip may reach up to 100 or 120 watts. Beyond this, the heat-removing mechanism may complicate the entire system design. Consequently, the available commercial devices often limit their power dissipation below this value through low-power design and/or a complex on- and/or off-chip *power management module*.

1.1.2.4 Scaling Theory By the driving force of refinement of feature sizes, Dennard proposed the constant-field scaling theory in 1974. Based on this theory, each dimension, x, y, and z, of a device is scaled down by a factor k, where $0 < k < 1$. In order to maintain the constant field in the device, all operating voltages require to be scaled down by the same factor of k and the charge density needs to be scaled up by a factor of $1/k$.

An illustration of the scaling principle is revealed in Figure 1.3. The channel length and width of the original device are scaled down by a factor k, namely, $L' = k \cdot L$ and $W' = k \cdot W$; the thickness of silicon dioxide (SiO_2) is also scaled down by the same factor, i.e., $t'_{ox} = k \cdot t_{ox}$. The same rule is applied to the maximum depletion depth x_d, that is, $x'_d = k \cdot x_d$.

Figure 1.4 shows the historical scaling records from 0.5 μm down to today, 32 nm. From the figure, we can see that the scaling factor k of each generation is about 0.7. This means that the chip area is decreased by a factor of two for each generation, thereby doubling the transistor count provided that the chip area remains the same.

The advantages of scaling down a device are as follows. First, the device density is increased by a factor of $1/k^2$. Second, the circuit delay is shortened by a factor of k. Third, the power dissipation per device is reduced by a factor of k^2. Consequently, the scaled-down device occupies less area, has higher performance, and consumes less power than the original one.

Review Questions

Q1-4. What is VLSI?

Q1-5. Explain the reasons why the CMOS technology has become the mainstream of the current VLSI realm.

Q1-6. What are the benefits of using VLSI circuits?

1.1.3 Design Issues of VLSI Circuits

A VLSI manufacturing process is called a *submicron* (SM) process when the feature size is below 1 μm, and a *deep submicron* (DSM) process when the feature size is roughly below 0.25 μm.[2] The corresponding devices made by these two processes are denoted SM devices and DSM devices, respectively. At present, DSM devices are popular in the design of a large-scale system because they provide a more economical way to integrate a much more complicated system into a single chip. The resulting chip is often referred to as a *system-on-a-chip* (SoC) device.

Even though DSM processes allow us to design a very complicated large-scale system, many design challenges indeed exist, in particular, when the feature sizes are beyond 0.13 μm. The associated design issues can be subdivided into two main classes: DSM devices and DSM interconnect.[3] In the following, we address each of these briefly.

1.1.3.1 Design Issues of DSM Devices The design issues of DSM devices include *thin-oxide (gate-oxide) tunneling/breakdown*, *gate leakage current*, *subthreshold current*, *velocity saturation*, *short-channel effects* on V_T, *hot-carrier effects*, and *drain-induced barrier lowering* (DIBL) effect.

The device features of typical DSM processes are summarized in Table 1.1. From the table, we can see that the thin-oxide (gate-oxide, i.e., silicon dioxide, SiO_2) thickness is reduced from 5.7 nm in a 0.25-μm process down to 1.65 nm in a 32-nm process. The side effects of this reduction are thin-oxide tunneling and breakdown. The thin-oxide tunneling may cause an extra gate leakage current. To avoid thin-oxide breakdown, the operating voltage applied to the gate has to be lowered. This means that the noise margins are reduced accordingly and the subthreshold current may no longer be ignored. To reduce the gate leakage current, high-k MOS transistors are widely employed starting from a 45-nm process. In high-k MOS transistors, a high-k dielectric is used to replace the gate oxide. Hence, the gate-dielectric thickness may be increased significantly, thereby reducing the gate leakage current dramatically. The actual gate-dielectric thickness depends on the relative permittivity of gate-dielectric material, referring to Section 3.4.1.2 for more details.

In addition, as the channel length of a device is reduced, velocity saturation, short-channel effects on V_T, and hot-carrier effects may no longer be ignored as in the case of a long-channel device. The electron and hole velocities in the channel or silicon bulk is proportional to the applied electric field when the electric field is below a critical

[2] Sometimes, a process with a feature size below 100 nm is referred to as a *nanometer* (nm) process.

[3] The wires linking together transistors, circuits, cells, modules, and systems are called interconnect.

Table 1.1: The device features of typical DSM processes.

Process	0.25 μm	0.18 μm	0.13 μm	90 nm	65 nm	45 nm	32 nm
	1998	1999	2000	2002	2006	2008	2010
t_{ox} (nm)	5.7	4.1	3.1	2.5	1.85	1.75	1.65
V_{DD} (V)	2.5	1.8	1.2	1.0	0.80	0.80	0.80
V_T (V)	0.55	0.4	0.35	0.35	0.32	0.32	0.32

Table 1.2: The metal1-wire features of typical DSM processes.

Process	0.25 μm	0.18 μm	0.13 μm	90 nm	65 nm	45 nm	32 nm
Thickness	0.61 μm	0.48 μm	0.40 μm	150 nm	170 nm	144 nm	95 nm
Width/space	0.3 μm	0.23 μm	0.17 μm	110 nm	105 nm	80 nm	56 nm
R/sq (m$\Omega/\square$)	44	56	68	112	100	118	178

value. However, these velocities will saturate at a value of about 8×10^6 cm/sec at 400 K, which is independent of the doping level and corresponds to an electric field with the strength of 6×10^4 V/cm for electrons and 2.4×10^5 V/cm for holes, respectively. When velocity saturation happens, the drain current of a MOS transistor will follow a linear rather than a quadratical relationship with applied gate-to-source voltage.

When the channel length is comparable to the drain *depletion-layer* thickness, the device is referred to as a *short-channel device*. The *short-channel effect* (SCE) refers to the reduction in V_T of a short-channel device and might result from the combination of the following effects: *charge sharing*, DIBL, and *subsurface punchthrough*. Charge sharing refers to the fact that the charge needed to induce a channel is not totally supported by the gate voltage but instead may obtain some help from others. DIBL refers to the influence of drain voltage on the threshold voltage. Subsurface punchthrough refers to the influence of drain voltage on the source *pn*-junction electron barrier.

A *hot electron* is an electron with kinetic energy greater than its thermal energy. Due to the high enough electric field at the end of channel, electron-hole pairs may be generated in the space-charge region of the drain junction through *impact ionization* by hot electrons. These generated electron and hole carriers may have several effects. First, they may trap into the gate oxide and shift the threshold voltage V_T of the device gradually. Second, they may inject into the gate and become the gate current. Third, they may move into the substrate and become the substrate current. Fourth, they may cause the parasitic BJT to become forward-biased and cause the device to fail.

1.1.3.2 Design Issues of DSM Interconnect

The design issues of DSM interconnect arise from *RLC* parasitics and include *IR drop*, *RC delays*, *capacitive coupling*, *inductive coupling*, *Ldi/dt noise*, electromigration, and *antenna effects*. In what follows, we briefly describe each of these.

The wires in a VLSI chip function as conductors for carrying signals, power, and clocks. Due to the inherent resistance and capacitance of wires, each wire has its definite IR drop. The metal-wire features of typical DSM processes are summarized in Table 1.2. We can see from the table that the thickness and width of wires are reduced with the evolution of feature sizes. Thereby, the *sheet resistance*, i.e., resistance per square, of a wire is increased profoundly.

The wire resistance leads to the IR drop, which may deteriorate the performance of a logic circuit, even making the logic circuit malfunction. The IR drop becomes more

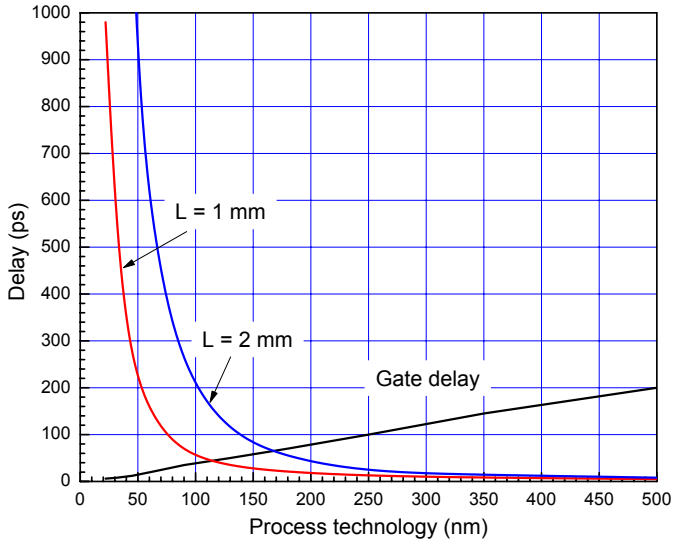


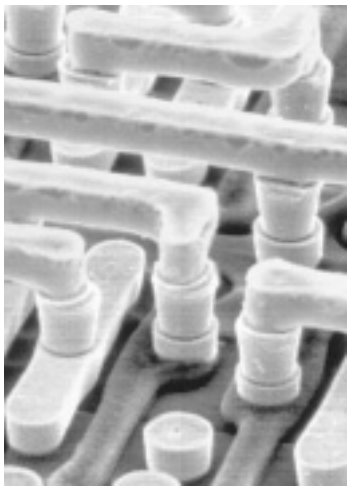
Figure 1.5: The gate versus wire delays in various processes.

noticeable in DSM processes due to the following reasons. First, the wire resistance increases with the decreasing feature sizes. Second, the currents passing through a wire increase with the reduction of supply voltage. Third, the power dissipation in modern DSM chips increases because more complex functions are to be executed by these chips. This implies that the current is increased accordingly. As a consequence, an important system design issue in designing a modern DSM chip is to reduce the IR drop in the chip.

The combination of wire capacitance and resistance leads to the RC delay, which may deteriorate the performance of logic circuits. With the reduction of feature sizes, the RC delay of wire is increased quadratically, but the gate delay is reduced significantly, with each generation having a factor of $k \approx 0.7$. Consequently, the signal delay through a wire could be easily longer than that through the gate driving it. As illustrated in Figure 1.5, for the given 1-mm wire, the interconnect delays are less than gate delays and can be ignored when the feature sizes are above 130 nm. However, as feature sizes are below 90 nm, the interconnect delays are much longer than gate delays, even dominating the delays of the entire system.

In addition, in DSM processes the spacing between two adjacent wires are narrowed, even much smaller than the thickness, as shown in Figure 1.6(a). An immediate consequence of this is that the capacitance between these two wires can no longer be overlooked and may result in a phenomenon called *capacitive coupling*, meaning that the signal in one wire is coupled into the other and may collapse the signal there. The capacitive coupling may cause a *signal-integrity problem*.

Besides capacitive coupling, the coupling effects may be caused by the wire inductance in DSM processes. *Faraday's law of induction* states that an induced voltage is produced in a conductor when it is emerged into a time-varying magnetic field produced by a nearby conductor carrying a current. This is called *inductive coupling*. Like capacitive coupling, inductive coupling may also make noises into adjacent wires. Nonetheless, unlike capacitive coupling being confined between two conductors, inductive coupling may affect a large area of circuits in an unpredictable way. This



(a)

Process (nm)	Metal layers
500	3 (Al)
350	4 (Al)
250	5 (Al)
180	6 (Al, low-k)
130	7/8/9 (Cu, low-k)
90	7/8/9 (Cu, low-k)
65	8/9 (Cu, low-k)
45	8/9 (Cu, low-k)
32	8/9 (Cu, low-k)

(b)

Figure 1.6: The revolution of wires of modern CMOS processes: (a) Four-layer aluminum with tungsten vias in a 0.35-μm technology (Courtesy of International Business Corporation. Unauthorized use not permitted); (b) scaling of metal layers in typical processes.

is because the inductive effect is formed in a closed loop and there may exist many possible *return paths* with different path lengths for a single signal path.

The induced voltage due to a time-varying current of a wire with self-inductance L can be expressed as Ldi/dt. The Ldi/dt effect often arises from the transitions of signals, especially clocks. Although the self-inductance of a wire is very small, the Ldi/dt effect could become pronounced if the current passing through it is changed rapidly. The Ldi/dt effect results in power supply spikes on the underlying circuit, leading to *Ldi/dt noises* and causing the *power-supply integrity problem* if they are large enough.

The *electromigration* is a phenomenon that electrons in a conductor migrates and dislodges the lattice of the conductor. This may result in breaking the conductor. In order to prevent this from occurring, the current density through a conductor must be controlled below a critical value determined by the conductor. The electromigration is more severely in aluminum than in copper. Hence, in practical processes using aluminum as conductors, the aluminum is usually alloyed with a small fraction of copper to reduce this effect.

A phenomenon that may cause a device to fail before the completion of a manufacture process is known as the *antenna effect*. The antenna effect is the phenomenon that charges are stored on metal wires during manufacturing. The charges stored on metal1 will cause the direct damage of gate oxide if the amount of charges is sufficient to produce a strong enough electric field. Consequently, it is necessary to limit the area of metal1 connected to polysilicon (or poly for short) directly.

1.1.3.3 Design Issues of VLSI Systems With the advance of CMOS processes, the feature sizes are reduced in a way much faster than what we can imagine. Along with this advance, many challenges in VLSI system designs follow immediately. These

challenges mainly cover the following: *power distribution network*, *power management*, and *clock distribution network*, as well as *design and test approach*.

A complete power distribution network needs to take into account the effects of IR drop, hot spots, Ldi/dt noises, ground bounce, and electromigration. In addition to IR drop and Ldi/dt noises, the *hot-spot problem* is essential in designing a VLSI system. The hot spots mean that temperatures of some small regions on a chip are curiously higher than the average value of the chip. These hot spots may deteriorate the performance of the chip, even causing the entire chip to fail eventually.

Careful analysis of power dissipation in all modules of a design is essential for keeping power dissipation within an acceptable limit. Since not all modules in a digital system need to be activated at all times, it is possible to only power the modules on demand at a time. Based on this, the power dissipation of a VLSI chip can be controlled under a desired bound. In fact, the power management has become an important issue for modern VLSI chip design. It may even determine whether the resulting product can be successful on the market or not.

The clock distribution network is another challenge in designing a VLSI system. Since in these systems a large area and a high operating frequency are required to carry out complex logic functions, the clock skew has to be controlled in a very narrow range. Otherwise, the systems may not function correctly. Hence, care must be taken in designing the clock distribution network. In addition, due to a large capacitance needs to be driven by the clock distribution network, the power dissipation of the clock distribution network may no longer be ignored and has to be coped with very carefully.

With the advent of high-degree integration of a VLSI chip, the increasing complexity of systems has made the related design much more difficult. An efficient and effective design approach is to use the divide-and-conquer paradigm to limit the number of components that can be handled at a time. However, combining various different modules into a desired system becomes more difficult with the increasing complexity of the system to be designed. Moreover, the testing for the combined system is more important and challenging.

Review Questions

Q1-7. Explain the meaning of Figure 1.5.

Q1-8. What is the hot-spot problem?

Q1-9. What are the design issues of DSM devices?

Q1-10. What are the design issues of DSM interconnect?

1.1.4 Economics of VLSI

The cost of an IC is roughly composed of two major factors: *fixed cost* and *variable cost*. The fixed cost, also referred to as the *nonrecurring engineering (NRE) cost*, is independent of the sales volume. It is mainly contributed by the cost from that a project is started until the first successful prototype is obtained. More precisely, the fixed cost covers direct and indirect costs. The direct cost includes the research and design (R&D) cost, manufacturing mask cost, as well as marketing and sales cost; the indirect cost comprises the investment of manufacturing equipments, the investment of CAD tools, building infrastructure cost, and so on. The variable cost is proportional to the product volume and is mainly the cost of manufacturing wafers, namely, *wafer price*, which is roughly in the range between 1,200 and 1,600 USD for a 300-mm wafer.

From the above discussion, the cost per IC can be expressed as follows.

$$\text{Cost per IC} = \text{Variable cost of IC} + \frac{\text{Fixed cost}}{\text{Volume}} \tag{1.1}$$

The variable cost per IC can be formulated as the following equation.

$$\text{Variable cost of IC} = \frac{\text{Cost of die} + \text{Cost of testing die} + \text{Cost of packaging and final test}}{\text{Final test yield} \times \text{Dies per wafer}} \tag{1.2}$$

The cost of a die is the wafer price divided by the number of good dies and can be represented as the following formula.

$$\text{Cost of die} = \frac{\text{Wafer price}}{\text{Dies per wafer} \times \text{Die yield}} \tag{1.3}$$

The number of dies in a wafer, excluding fragmented dies on the boundary, can be approximated by the following equation.

$$\text{Dies per wafer} = \frac{3}{4}\frac{d^2}{A} - \frac{1}{2\sqrt{A}}d \tag{1.4}$$

where d is the diameter of the wafer and A is the area of square dies. The derivation of this equation is left to the reader as an exercise.

The die yield can be estimated by the following widely used function.

$$\text{Die yield} = \left(1 + \frac{D_0 A}{\alpha}\right)^{-\alpha} \tag{1.5}$$

where D_0 is the defect density, i.e., the defects per unit area, in defects/cm^2, and α is a measure of manufacturing complexity. The typical values of D_0 and α are 0.3 to 1.3 and 4.0, respectively. From this equation, it is clear that the die yield is inversely proportional to the die area.

The following two examples exemplify the above concepts about the cost of an IC. In these two examples, we intend to ignore the fixed cost and only take into account the wafer price when calculating die cost.

Example 1-1: (Die area is 1 cm^2.) Assume that the diameter of the wafer is 30 cm and the die area is 1 cm^2. The defect density D_0 is 0.6 defects/cm^2 and the manufacturing complexity α is 4. Supposing that the wafer price is 1,500 USD, calculate the cost of each die without considering the fixed cost.

Solution: The number of dies per wafer can be estimated by using Equation (1.4).

$$\begin{aligned}\text{Dies per wafer} &= \frac{3}{4}\frac{d^2}{A} - \frac{1}{2\sqrt{A}}d \\ &= \frac{3}{4}\left(\frac{30^2}{1}\right) - \frac{1}{2\sqrt{1}}30 = 660\end{aligned}$$

The die yield is obtained from Equation (1.5) and is as follows.

$$\begin{aligned}\text{Die yield} &= \left(1 + \frac{D_0 A}{\alpha}\right)^{-\alpha} \\ &= \left(1 + \frac{0.6 \times 1}{4}\right)^{-4} = 0.57\end{aligned}$$

The cost of each die is calculated as follows by using Equation (1.3).

$$\begin{aligned}\text{Cost of die} &= \frac{\text{Wafer price}}{\text{Dies per wafer}\times\text{Die yield}} \\ &= \frac{1,500}{660\times 0.57} = 3.98 \text{ (USD)}\end{aligned}$$

Example 1-2: (Die area is 2.5 mm × 2.5 mm.) Assume that the diameter of the wafer is 30 cm and the die area is 2.5 mm × 2.5 mm. The defect density D_0 is 0.6 defects/cm^2 and the manufacturing complexity α is 4. The wafer price is 1,500 USD. Calculate the cost of each die without considering the fixed cost.

Solution: The number of dies per wafer can be estimated by using Equation (1.4).

$$\begin{aligned}\text{Dies per wafer} &= \frac{3}{4}\frac{d^2}{A} - \frac{1}{2\sqrt{A}}d \\ &= \frac{3}{4}\left(\frac{30^2}{0.0625}\right) - \frac{1}{2\sqrt{0.0625}}30 = 10,740\end{aligned}$$

The die yield is obtained from Equation (1.5) and is as follows.

$$\begin{aligned}\text{Die yield} &= \left(1+\frac{D_0 A}{\alpha}\right)^{-\alpha} \\ &= \left(1+\frac{0.6\times 0.0625}{4}\right)^{-4} = 0.96\end{aligned}$$

The cost of each die is calculated as follows by using Equation (1.3).

$$\begin{aligned}\text{Cost of die} &= \frac{\text{Wafer price}}{\text{Dies per wafer}\times\text{Die yield}} \\ &= \frac{1,500}{10,740\times 0.96} = 0.15 \text{ (USD)}\end{aligned}$$

From the above two examples, it is apparent that the die yield is a strong function of die area, namely, inversely proportional to the die area. Thus, in practical applications, it is necessary to limit the die area in order to control the die yield and hence the die cost within an acceptable value. Note that in practical chip design projects, the die area, power dissipation, and performance, are usually the three major factors to be considered and traded off.

Review Questions

Q1-11. What are the two main factors determining the cost of an IC?

Q1-12. Describe the meaning of the nonrecurring engineering (NRE) cost.

Q1-13. What is the major factor of variable cost of an IC?

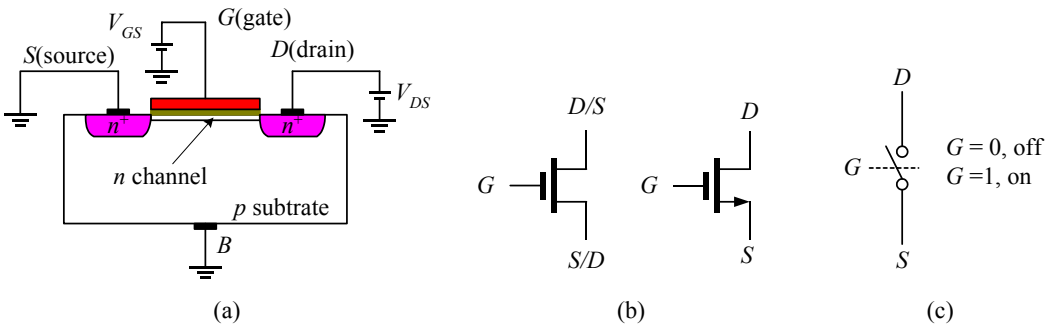


Figure 1.7: The (a) physical structure, (b) circuit symbols, and (c) switch circuit model of an nMOS transistor.

1.2 MOS Transistors as Switches

In this section, we begin to consider basic operations of both nMOS and pMOS transistors and regard these two types of transistors as switches, known as *nMOS switches* and *pMOS switches*, respectively. Then, we consider the combining effects of both nMOS and pMOS transistors as a combined switch, called a *transmission gate* (TG) or a *CMOS switch*. The features of these three types of switches used in logic circuits are also discussed. Finally, we deal with the basic design principles of *switch logic*. Such logic is also referred to as *steering logic* because the data inputs are routed directly to outputs under the control of some specific signals.

1.2.1 nMOS Transistors

The physical structure of an nMOS transistor is basically composed of a metal-oxide-silicon (MOS) system and two n^+ regions on the surface of a p-type silicon substrate, as depicted in Figure 1.7(a). The MOS system is a sandwich structure where a dielectric (an insulator) is inserted between a metal or a polysilicon and a p-type substrate. The metal or polysilicon is called the *gate*. The two n^+ regions on the surface of the substrate are referred to as *drain* and *source*, respectively.

The operation of an nMOS transistor can be illustrated by Figure 1.7(a). When a large enough positive voltage V_{GS} is applied to the gate (electrode), electrons are attracted toward the silicon surface from the p-type substrate due to a positive electric field being built on the silicon surface by the gate voltage. These electrons form a channel between the drain and source. The minimum voltage V_{GS} inducing the channel is defined as the *threshold voltage*, denoted V_{Tn}, of the nMOS transistor. The value of V_{Tn} ranges from 0.3 V to 0.7 V for the present submicron and deep-submicron processes, depending on a particular process of interest.

For digital applications, an nMOS transistor can be thought of as a simple switch element. The switch is turned on when the gate voltage is greater than or equal to its threshold voltage and turned off otherwise. Due to the symmetric structure of an nMOS transistor, either of n^+ regions can be used as the source or drain, depending on how the operating voltage is applied. One with more positive voltage is the drain and the other is the source because the carriers on the nMOS transistor are electrons.

Figure 1.7(b) shows the circuit symbols that are often used in circuit designs. The one with an explicit arrow associated with the source electrode is often used in analog

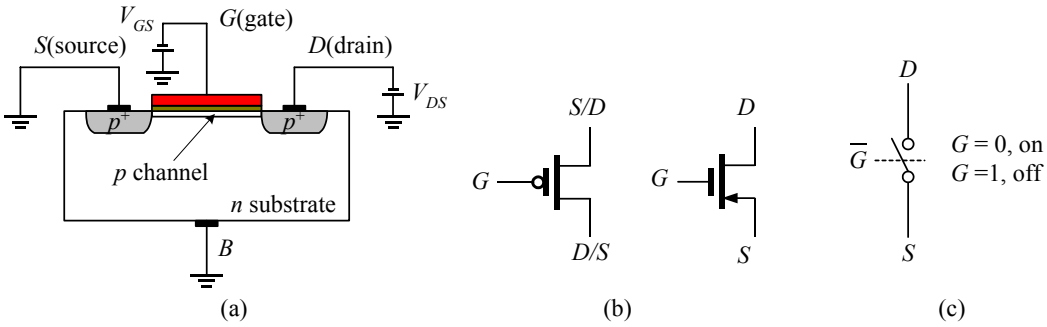


Figure 1.8: The (a) physical structure, (b) circuit symbols, and (c) switch circuit model of a pMOS transistor.

applications, where the roles of source and drain are fixed. The other without an explicit arrow is often used in digital applications because the roles of the drain and source will be dynamically determined by the actual operating conditions of the circuit. The switch circuit model is depicted in Figure 1.7(c).

1.2.2 pMOS Transistors

Likewise, the physical structure of a pMOS transistor comprises a MOS system and two p^+ regions on the surface of an n-type silicon substrate, as depicted in Figure 1.8(a). The MOS system is a sandwich structure where a dielectric (an insulator) is inserted between a metal or polysilicon and an n-type substrate. The metal or polysilicon is called a gate. The two p^+ regions on the surface of substrate are referred to as the drain and source, respectively.

The operation of a pMOS transistor can be illustrated by Figure 1.8(a). When a large negative voltage V_{GS} is applied to the gate (electrode), holes are attracted toward the silicon surface from the n-type substrate due to a negative electric field being built on the silicon surface by the gate voltage. These holes form a channel between the drain and source. The minimum voltage $|V_{GS}|$ inducing the channel is defined as the threshold voltage, denoted V_{Tp}, of the pMOS transistor. The value of V_{Tp} ranges from -0.3 V to -0.7 V for the present submicron and deep-submicron processes, depending on a particular process of interest.

Like an nMOS transistor, a pMOS transistor can be regarded as a simple switch element for digital applications. The switch is turned on when the gate voltage is less than or equal to its threshold voltage and turned off otherwise. Due to the symmetric structure of a pMOS transistor, either of p^+ regions can be used as the source or drain, depending on how the operating voltage is applied. One with more positive voltage is the source and the other is the drain since the carriers on the pMOS transistor are holes.

Figure 1.8(b) shows the circuit symbols that are often used in circuit designs. The symbol convention of pMOS transistors is exactly the same as that of nMOS transistors. The one with an explicit arrow associated with the source electrode is often used in analog applications in which the roles of source and drain are fixed. The other without an explicit arrow but with a circle at the gate is often used in digital applications because the roles of drain and source will be dynamically determined by the actual operating conditions of the circuit. The circle is used to distinguish it from the nMOS

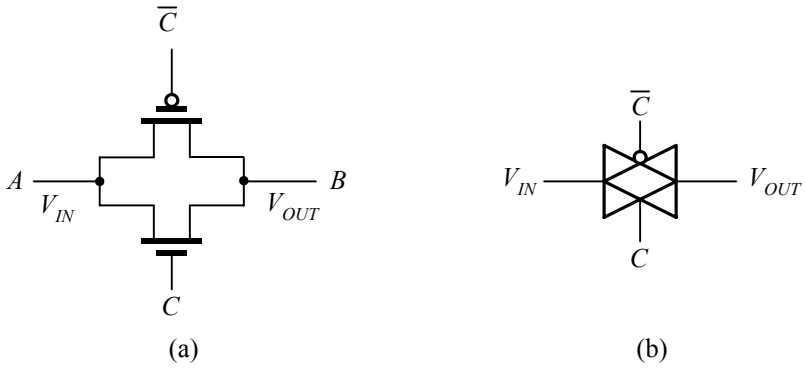


Figure 1.9: The (a) circuit structure and (b) logic symbol of a TG switch.

transistor and to indicate that the pMOS transistor is at active-low enable. The switch circuit model is depicted in Figure 1.8(c).

1.2.3 CMOS Transmission Gates

Since a voltage of magnitude V_{Tn} between the gate and source of an nMOS transistor is required to turn on the transistor, the maximum output voltage of an nMOS switch is equal to $V_{DD} - V_{Tn}$, provided that V_{DD} is applied to both gate and drain electrodes. Similarly, the minimum output voltage of a pMOS switch is equal to $|V_{Tp}|$, provided that 0 V is applied to both gate and drain electrodes. The above two statements can be restated in terms of information transfer by letting 0 V represent logic 0 and V_{DD} denote logic 1 as follows. The nMOS transistor can pass 0 perfectly but cannot pass 1 without degradation; the pMOS transistor can pass 1 perfectly but cannot pass 0 without degradation.

The aforementioned shortcomings of nMOS and pMOS transistors may be overcome by combining an nMOS transistor with a pMOS transistor as a parallel-connected switch, referred to as a transmission gate (TG) or a CMOS switch, as shown in Figure 1.9. Since both nMOS and pMOS transistors are connected in parallel, the imperfect feature of one transistor will be made up by the other. Figure 1.9(a) shows the circuit structure of a TG switch and Figure 1.9(b) shows the logic symbol often used in logic diagrams.

Even though using TG switches may overcome the degradation of information passing through them, each TG switch needs two transistors, one nMOS and one pMOS. This means that the use of TG switches needs more area than the use of nMOS switches or pMOS switches alone. In practice, for area-limited applications the use of nMOS transistors is much more preferable to pMOS transistors since the electron mobility is much greater than hole mobility. Hence, nMOS transistors perform much better than pMOS transistors.

Review Questions

Q1-14. Describe the operation of nMOS transistors.

Q1-15. Describe the operation of pMOS transistors.

Q1-16. Describe the operation of CMOS switches.

Q1-17. What is the drawback of an nMOS transistor when used as a switch?

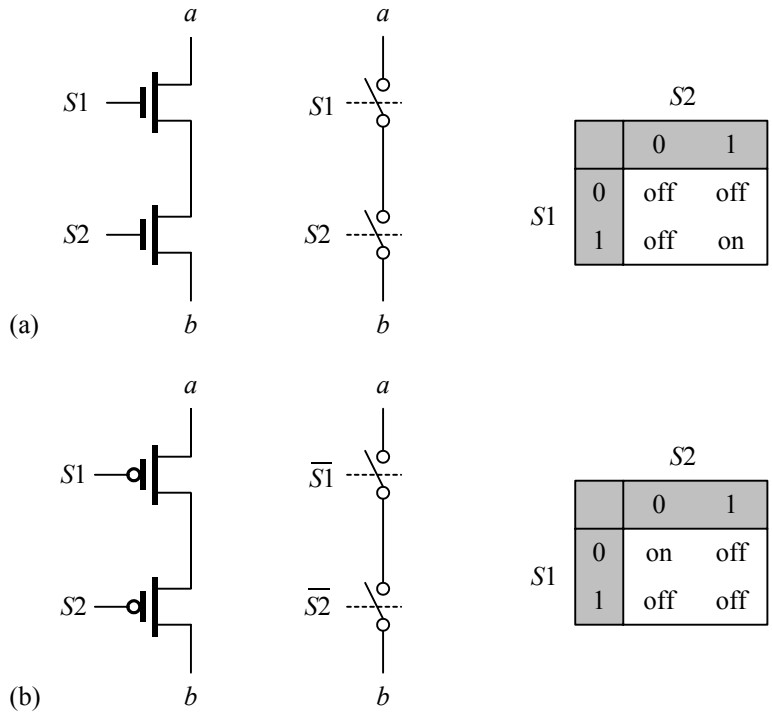


Figure 1.10: The operations of series switches: (a) using nMOS switches; (b) using pMOS switches.

Q1-18. What is the drawback of a pMOS transistor when used as a switch?

1.2.4 Simple Switch Logic Design

As introduced, any of three switches, nMOS, pMOS, and TG, may be used as a switch to control the close (on) or open (off) status of two points. Based on a proper combination of these switches, a switch logic circuit can be constructed. In the following, we begin with the discussion of compound switches and then introduce a systematic design methodology for constructing a switch logic circuit from a given switching function.

1.2.4.1 Compound Switches For many applications, we often combine two or more switches in a serial, parallel, or combined fashion to form a compound switch. For instance, the case of two switches being connected in series to form a compound switch is shown in Figure 1.10. The operation of the resulting switch is controlled by two control signals: $S1$ and $S2$. The compound switch is turned on only when both control signals $S1$ and $S2$ are asserted and remains in an off state otherwise.

Recall that to activate an nMOS switch we need to apply a high-level voltage to its gate and to activate a pMOS switch we need to apply a low-level voltage to its gate. As a result, the compound nMOS switch shown in Figure 1.10(a) is turned on only when both control signals $S1$ and $S2$ are at high-level voltages (usually V_{DD}) and remains in an off state in all other combinations of control signals. The compound pMOS switch depicted in Figure 1.10(b) is turned on only when both control signals

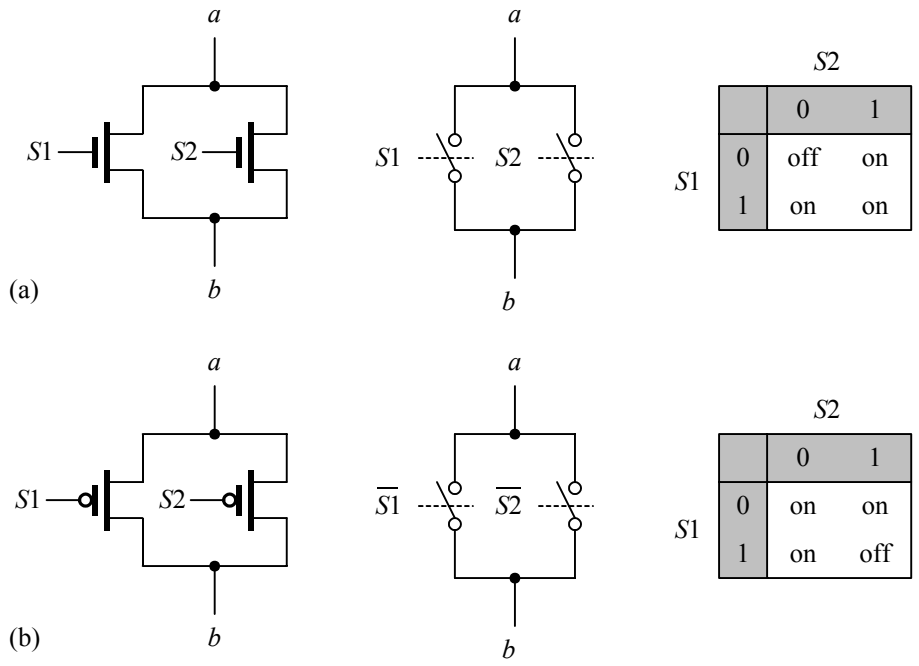


Figure 1.11: The operations of parallel switches: (a) using nMOS switches; (b) using pMOS switches.

$S1$ and $S2$ are at low-level voltages (usually at the ground level) and remains in an off state in all other combinations of control signals.

Figure 1.11 shows the case of two switches being connected in parallel to form a compound switch. The operation of the resulting switch is controlled by two control signals: $S1$ and $S2$. The compound switch is turned on whenever either switch is on. Therefore, the compound switch is turned off only if both control signals $S1$ and $S2$ are deasserted and remains in an on state otherwise.

In Figure 1.11(a), the compound nMOS switch is turned on whenever one control signal of $S1$ and $S2$ is at a high-level voltage (usually V_{DD}) and remains in an off state only when both control signals are at the ground level. In Figure 1.11(b), the compound pMOS switch is turned on whenever one control signal of $S1$ and $S2$ is at the ground level and remains in an off state only when both control signals $S1$ and $S2$ are at high-level voltages.

1.2.4.2 The f/$\bar{f}$ Paradigm A simple logic pattern utilizing the aforementioned features of both nMOS and pMOS switches to implement a switching function is referred to as the *$f/\bar{f}$ paradigm*, fully CMOS (FCMOS) logic, or CMOS logic for short. The rationale behind the $f/\bar{f}$ paradigm is on the observation that pMOS switches are ideal switches for transferring logic-1 signals whereas nMOS switches are ideal switches for transferring logic-0 signals. Consequently, we apply pMOS switches to implement the true switching function f while applying nMOS switches to implement the complementary switching function $\bar{f}$.

The block diagram of the $f/\bar{f}$ paradigm is shown in Figure 1.12, where two logic blocks are referred to as the f block and $\bar{f}$ block, respectively. The f block realizes

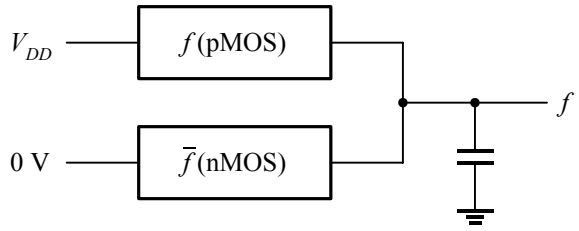


Figure 1.12: The block diagram of the $f/\bar{f}$ paradigm.

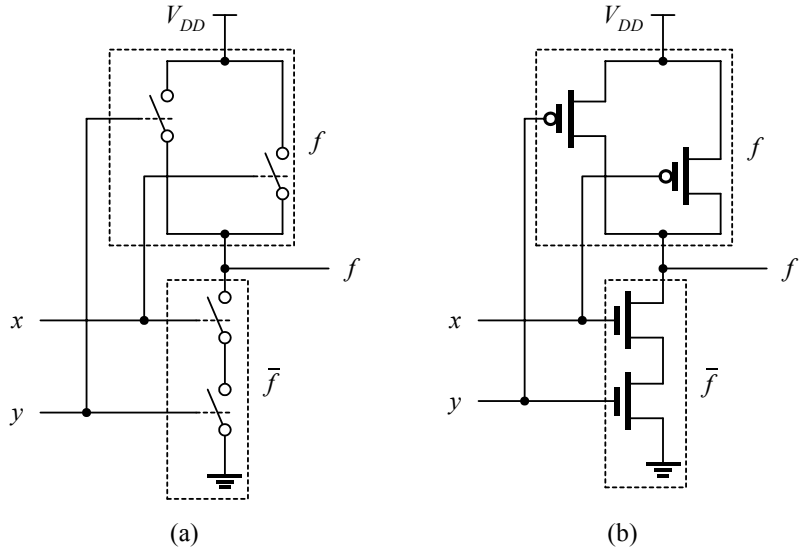


Figure 1.13: The (a) switch logic circuit and (b) CMOS logic circuit of a two-input NAND gate.

the true switching function and connects the output to V_{DD} (namely logic 1) when it is on while the $\bar{f}$ block realizes the complementary switching function and connects the output to ground (namely logic 0) when it is on. The capacitor at the output node denotes the parasitic capacitance inherent in the output node.

In what follows, we give two examples to illustrate how to design practical switch logic circuits using the $f/\bar{f}$ paradigm. The first example is to design a two-input NAND gate.

Example 1-3: (A two-input NAND gate.) The switching function of a two-input NAND gate is $f(x, y) = \overline{x \cdot y}$. Use the $f/\bar{f}$ paradigm to design and realize it.

Solution: By DeMorgan's law, $f(x, y) = \overline{x \cdot y} = \bar{x}+\bar{y}$. Hence, the f block is realized by two parallel pMOS transistors. The $\bar{f}$ block is realized by two series nMOS transistors because the complementary function of f is $\bar{f}(x, y) = x \cdot y$. The resulting switch and CMOS logic circuits are shown in Figures 1.13(a) and (b), respectively.

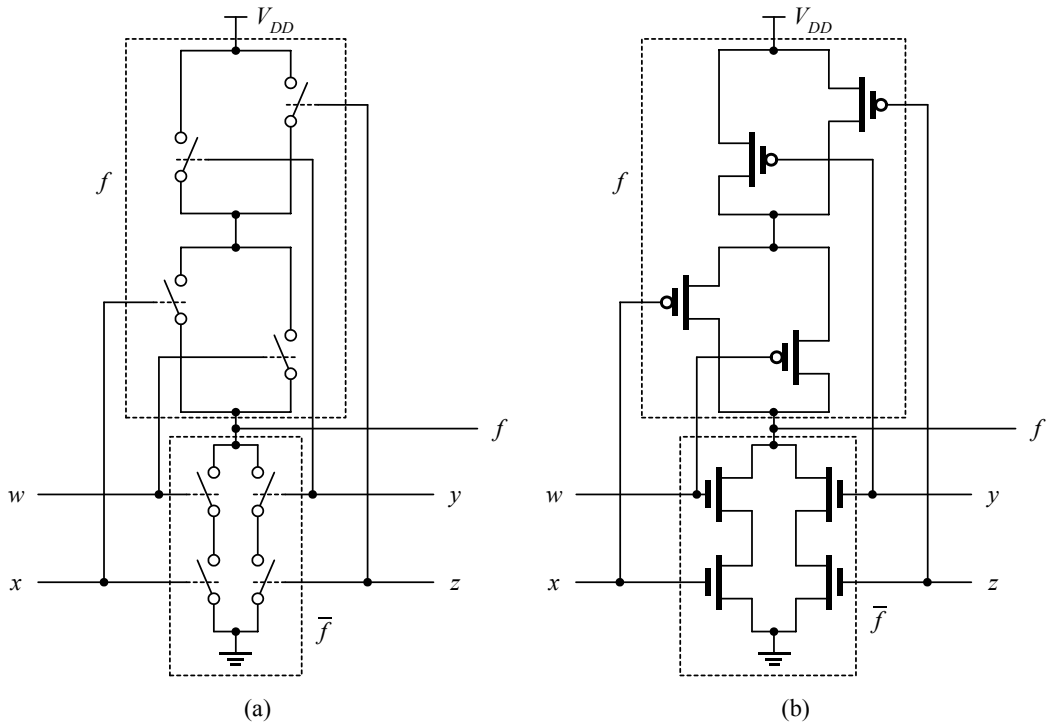


Figure 1.14: The (a) switch logic circuit and (b) CMOS logic circuit of an AOI gate.

It is evident from Figure 1.13 that both functions of f and $\bar{f}$ blocks are dual with each other. The following example further demonstrates how a more complex logic circuit known as an *AND-OR-Inverter* (AOI) gate can be designed using the $f/\bar{f}$ paradigm and realized with both types of nMOS and pMOS switches.

Example 1-4: (An AOI gate.) Realize the following switching function $f(w, x, y, z) = \overline{w \cdot x + y \cdot z}$ using the $f/\bar{f}$ paradigm.

Solution: By DeMorgan's law, $f(w, x, y, z) = \overline{w \cdot x} \cdot \overline{y \cdot z} = (\bar{w} + \bar{x}) \cdot (\bar{y} + \bar{z})$. Hence, the f block is realized by two pairs of pMOS transistors connected in parallel and then these two pairs are connected in series. The $\bar{f}$ block is realized by two pairs of nMOS transistors connected in series and then these two pairs are connected in parallel because the complementary function of f is $\bar{f}(w, x, y, z) = w \cdot x + y \cdot z$. The resulting switch and CMOS logic circuits are shown in Figures 1.14(a) and (b), respectively.

Review Questions

Q1-19. Using the $f/\bar{f}$ paradigm, design a CMOS two-input NOR gate.
Q1-20. Using the $f/\bar{f}$ paradigm, design a CMOS three-input NAND gate.
Q1-21. Using the $f/\bar{f}$ paradigm, design a CMOS three-input NOR gate.

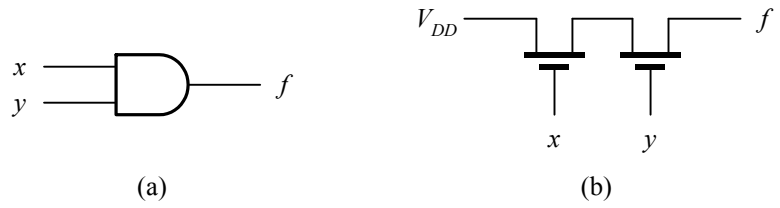


Figure 1.15: The difference between (a) gate logic circuit and (b) switch logic circuit.

1.2.5 Principles of CMOS Logic Design

The basic design principles of CMOS logic can be illustrated by exploring the difference between gate logic and switch logic circuits depicted in Figures 1.15(a) and (b), respectively. In Figure 1.15(a), the output f is 1 if both inputs x and y are 1, and is 0 otherwise; in Figure 1.15(b), the output f is 1 if both inputs x and y are 1, and is undefined otherwise. Consequently, the switch logic circuit shown in Figure 1.15(b) cannot realize the function of Figure 1.15(a) exactly because it only performs the function when both x and y are 1 but nothing else.

1.2.5.1 Two Fundamental Rules A general logic circuit has two definite values: logic 0 (ground) and logic 1 (V_{DD}). In order for a switch logic circuit to specify a definite value for every possible combination of input variables, the following two rules should be followed so as to correctly and completely realize a switching function using CMOS switches.

- **Rule 1 (node-value rule):** The signal summing point (such as f) must always be connected to 0 (ground) or 1 (V_{DD}) at any time.
- **Rule 2 (node-conflict-free rule):** The signal summing point (such as f) must never be connected to 0 (ground) and 1 (V_{DD}) at the same time.

Any logic circuit must always follow **Rule 1** in order to work correctly. **Rule 2** distinguishes a ratioless logic circuit from a ratioed one. A CMOS logic circuit is said to be *ratioless* if it follows both rules and is *ratioed* logic if **Rule 2** is violated but the sizes of both pull-up and pull-down paths are set appropriately.

A ratioless logic circuit can always perform the designated switching function correctly regardless of the relative sizes of nMOS, pMOS, or TG switches. In contrast, for a ratioed logic circuit to function properly, the relative sizes of nMOS, pMOS, TG switches used in the circuit must be set appropriately.

1.2.5.2 Residues of a Switching Function The principle of CMOS switch logic design is governed by Shannon's expansion (or decomposition) theorem, which is stated as follows.

Theorem 1.2.1 (Shannon's expansion theorem.)
Let $f(x_{n-1}, \cdots, x_{i+1}, x_i, x_{i-1}, \cdots, x_0)$ be an n-variable switching function. Then f can be decomposed with respect to variable x_i, where $0 \leq i < n$, as follows.

$$\begin{aligned} f(x_{n-1}, \cdots, x_{i+1}, x_i, x_{i-1}, \cdots, x_0) &= x_i \cdot f(x_{n-1}, \cdots, x_{i+1}, 1, x_{i-1}, \cdots, x_0) + \\ &\quad \bar{x_i} \cdot f(x_{n-1}, \cdots, x_{i+1}, 0, x_{i-1}, \cdots, x_0) \end{aligned} \tag{1.6}$$

The proof of Shannon's expansion theorem is quite trivial; hence, we omit it here. For convenience, we will refer to the variable x_i as the *control variable* and the other variables as *function variables*.

Example 1-5: (Shannon's expansion theorem.) Consider the switching function $f(x, y, z) = xy + yz + xz$ and decompose it with respect to variable y.

$$\begin{aligned} f(x,y,z) &= xy + yz + xz \\ &= y \cdot f(x,1,z) + \bar{y} \cdot f(x,0,z) \\ &= y \cdot (x + z + xz) + \bar{y} \cdot (xz) \\ &= xy + yz + x\bar{y}z \end{aligned}$$

It is easy to show that the last line is indeed the same as the original switching function. Hence, we have shown the validity of Shannon's expansion theorem.

For convenience, we often distinguish a variable from a literal. A *variable* x is an object that can hold two values, 1 and 0, and can appear in one of two forms, x and $\bar{x}$. A *literal* is a variable in either true or complementary form. In other words, x and $\bar{x}$ denote two different literals but the same variable x. Once we have these, we define the residue of a switching function $f(X)$ with respect to a combination of a subset of $X = \{x_{n-1}, x_{n-2}, \cdots, x_1, x_0\}$ as follows.

Residues of a switching function. *Let $X = \{x_{n-1}, x_{n-2}, \cdots, x_1, x_0\}$ be the set of all n variables and $Y = \{y_{m-1}, y_{m-2}, \cdots, y_1, y_0\}$ be a subset of X, where $y_i \in X$ and $m \leq n$. The residue of $f(X)$ with respect to Y, denoted $f_Y(X)$, is defined to be the function value when all complementary literals in Y are set to 0 and all true literals in Y are set to 1.*

Based on this definition, the Shannon's expansion theorem can be thought of as a combination of two residues of the switching function, namely, $f(X) = x_i f_{x_i}(X) + \bar{x}_i f_{\bar{x}_i}(X)$.

Example 1-6: (Residue of a switching function.) Compute the residue of the following switching function with respect to variables x.

$$f(x, y, z) = xy + yz + xz$$

Solution: By definition, the residue of f with respect to variable x is obtained by setting x to 1 and computing the value of the switching function f. That is,

$$f_x(1, y, z) = 1 \cdot y + yz + 1 \cdot z = x + y$$

Hence, the residue of f with respect to variable x is $x + y$.

With the definition of residue, a switching function can be represented as a combination of two residues. For instance, $f(x, y, z) = x \cdot f_x(1, y, z) + \bar{x} \cdot f_{\bar{x}}(0, y, z)$. The reader interested in this is invited to verify this. By repeatedly applying Shannon's expansion theorem to further decompose residue functions, a binary tree is formed. Such a tree is often referred to as a *decomposition tree*. An example of a decomposition tree for a 4-variable switching function $f(w, x, y, z)$ is exhibited in Figure 1.16,

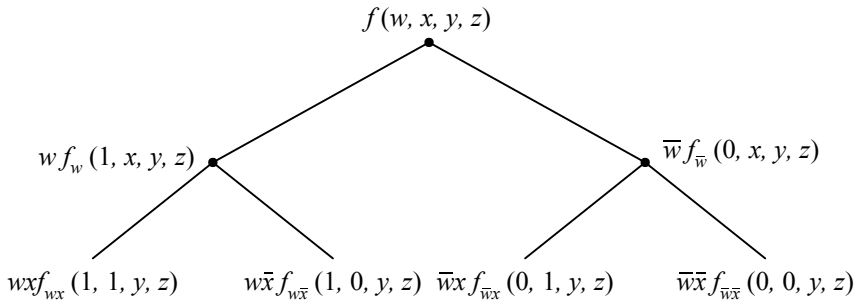


Figure 1.16: A decomposition tree for a four-variable switching function $f(w, x, y, z)$.

which is formed by decomposing the switching function with respect to variables w and x in order. The switching function f can then be expressed as a combination of the four residues as follows.

$$\begin{aligned} f(w,x,y,z) &= w \cdot f_w(1,x,y,z) + \bar{w} \cdot f_{\bar{w}}(0,x,y,z) \\ &= w \cdot [x \cdot f_{wx}(1,1,y,z) + \bar{x} \cdot f_{w\bar{x}}(1,0,y,z)] + \\ &\quad \bar{w} \cdot [x \cdot f_{\bar{w}x}(0,1,y,z) + \bar{x} \cdot f_{\bar{w}\bar{x}}(0,0,y,z)] \\ &= wxf_{wx}(1,1,y,z) + w\bar{x}f_{w\bar{x}}(1,0,y,z) + \\ &\quad \bar{w}xf_{\bar{w}x}(0,1,y,z) + \bar{w}\bar{x}f_{\bar{w}\bar{x}}(0,0,y,z) \end{aligned} \tag{1.7}$$

Each internal node of a decomposition tree can be realized by a 2-to-1 multiplexer with the control variable as its source selection variable and the residues as its two inputs. The resulting circuit is called a *tree network*.

The residues of a switching function f can also be found graphically. To this end, the residue map of a switching function f is defined as follows.

Residue map of a switching function. *A residue map is a two-dimensional graph with 2^m columns and 2^{n-m} rows. Each of 2^m columns corresponds to a combination of variables $Y = \{y_{m-1}, y_{m-2}, \cdots, y_1, y_0\}$, labeled in the increasing order, and each of 2^{n-m} rows corresponds to a combination of variables in the set of $X-Y$, labeled in Gray-code order. The cell at the intersection of each column and each row corresponds to a combination of variables $X = \{x_{n-1}, x_{n-2}, \cdots, x_1, x_0\}$ and is denoted the decimal value. To represent a switching function on the residue map, we circle the true minterm and leave the false minterm untouched. In addition, the don't care minterms are starred.*

The above definition is illustrated in Figure 1.17. Based on this definition, the variables in the set Y are referred to as *control variables*, and the variables in the set $X - Y$ as called *function variables*. Of course, the set of function variables may be empty.

Once we have prepared a residue map for a switching function with respect to a subset Y of the set of variables X, we can further use it to find the residues of the switching function. The general procedure is as follows.

1. Examine each column of the residue map from left to right. The residue of the column is 0 if no cell in the column is circled. The residue of the column is 1 if at least one cell in the column is circled and the other cells are circled or starred.

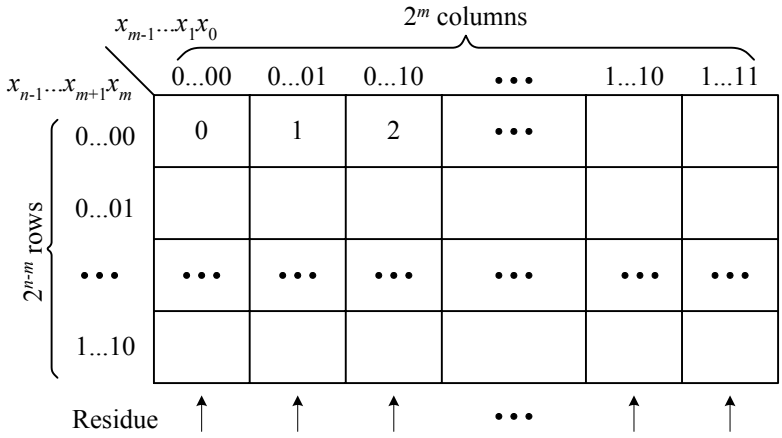


Figure 1.17: The general residue map of an n-variable switching function.

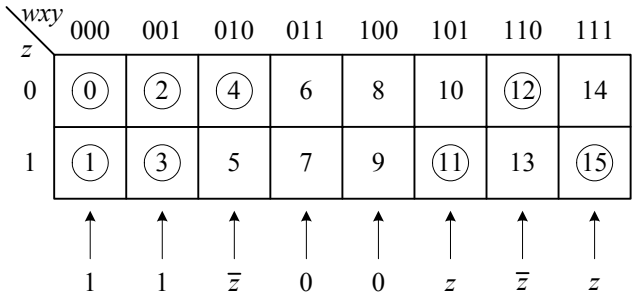


Figure 1.18: An example of using a residue map to find residues.

2. The residue is a function of function variables (namely, variables not in the subset Y) if only a proper subset of cells in the column is circled. The starred cells in the same column may be used to help simplify the residue function in this case.

The following example illustrates how to apply the above procedure to find the residues of a switching function.

Example 1-7: (An example of using a residue map.) Assume that $f(w,x,y,z) = \sum(0,1,2,3,4,11,12,15)$. Using the residue map, find the residues of $f(w,x,y,z)$ with respect to $Y = \{w,x,y\}$.

Solution: The residue map of f is depicted in Figure 1.18. According to the above procedure, the residues of columns 0 and 1 are 1 because all cells in these two columns are circled. The residues of columns 3 and 4 are 0 because no cells in these two columns are circled. The residues of columns 2 and 6 are $\bar{z}$ because only the cells corresponding to the row of $z = 0$ are circled. The residues of columns 5 and 7 are z because only the cells corresponding to the row of $z = 1$ are circled.

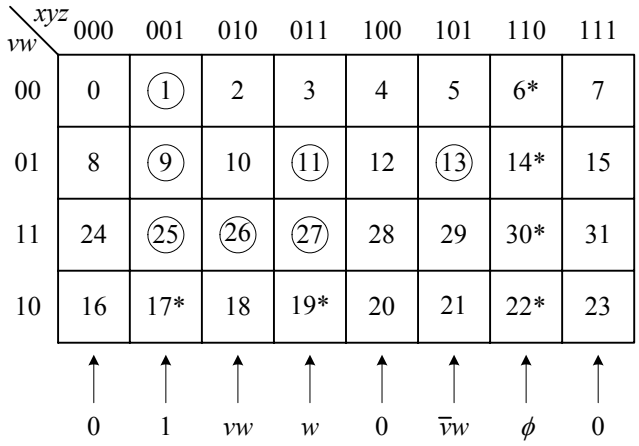


Figure 1.19: A more complicated example of using a residue map to find residues.

The following example shows how the don't care minterms may help find the residues of a switching function.

Example 1-8: (Another example of using a residue map.) Assume that $f(v,w,x,y,z) = \sum(1,9,11,13,25,26,27) + \sum_{\phi}(6,14,17,19,22,30)$. Using the residue map, find the residues of $f(v,w,x,y,z)$ with respect to $Y = \{x,y,z\}$.

Solution: The residue map of f is depicted in Figure 1.19. According to the above procedure, the residues of columns 0, 4, and 7 are 0 because no cells in these three columns are circled. The residue of column 1 is 1 because three cells in this column are circled and the remaining cell is starred. The residue of column 6 is ϕ because all cells in the column are don't care (i.e., no cell is circled). The residue of column 2 is vw since only the cell corresponding to row $vw = 11$ is circled. The residue of column 3 is w because the cells corresponding to rows $vw = 01$ and $vw = 11$ are circled. These two cells are combined into the result w. The residue of column 5 is $\bar{v}w$ because only the cell corresponding to the row of $vw = 01$ is circled.

1.2.5.3 Systematic Design Paradigms By using Shannon's expansion theorem, we can derive two basic systematic paradigms that can be used to design a ratioless logic circuit. Depending on how the control variable is chosen at each decomposition, tree networks can be further categorized into a *freeform-tree network* and a *uniform-tree network*.

- **Freeform-tree networks.** In the freeform-tree network, each nontrivial node of the decomposition tree has its total freedom to select the control variable as it is further decomposed. Hence, the survived literals in two leaf nodes need not be the same variable. It should be noted that both sets of control and function variables need not be disjoint but are disjoint at the same branch.
- **Uniform-tree networks.** In the uniform-tree network, all nontrivial nodes at the same height of the decomposition tree use the same control variable when they

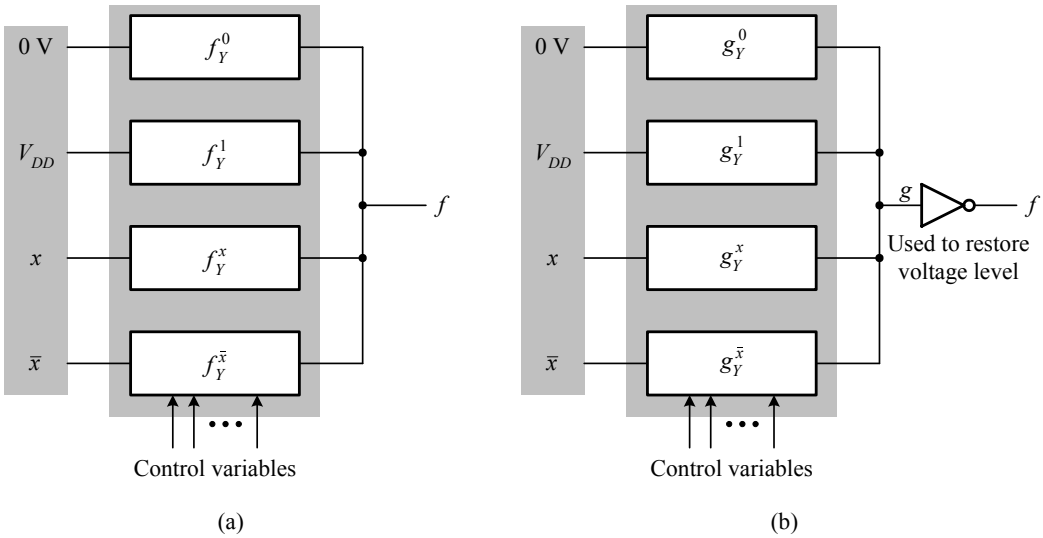


Figure 1.20: The general block diagrams of a 0/1-$x/\bar{x}$-tree network (x is a function variable): (a) using TG switches; (b) using nMOS switches.

are further decomposed. As in the freeform-tree network, the survived literals in two leaf nodes need not be the same variable. Two special cases are 0/1-$x/\bar{x}$-tree and 0/1-tree networks.

These networks can generally be realized in any type of nMOS, pMOS, and TG switches, or their combinations. However, excepting in 0/1-tree networks, the pMOS switches are usually not used due to their poor performance compared to nMOS switches.

1.2.5.4 0/1-x/$\bar{x}$-Tree Networks The 0/1-$x/\bar{x}$-tree network is a special case of uniform-tree network in which $n-1$ variables are used as control variables and all leaf nodes have the same height in the decomposition tree. This results in that each residue can only have one of four values, 0, 1, x, and $\bar{x}$. The combinations of control variables (Y) generating the same residue are then combined into a switching function, denoted as f_Y^s, where $s \in \{0, 1, x, \bar{x}\}$. Each switching function may be further simplified prior to being realized by a TG or an nMOS switch network, as shown in Figure 1.20.

Example 1-9: (A simple example of a 0/1-x/$\bar{x}$-tree network.) Realize the following switching function using a 0/1-$x/\bar{x}$-tree network and nMOS switches.

$$f(x, y, z) = xy + yz + xz$$

Solution: Assume that the inverter at the output is not necessary; that is, we can tolerate the degradation of logic-1 signal. By computing the residues with respect to all combinations of $Y = \{y, z\}$, the functions of f_Y^0, f_Y^1, f_Y^x, and $f_Y^{\bar{x}}$ are $\{\bar{y}\bar{z}\}$, $\{yz\}$, $\{\bar{y}z, y\bar{z}\}$, and ϕ (empty set), respectively. The resulting logic circuit is shown in Figure 1.21.

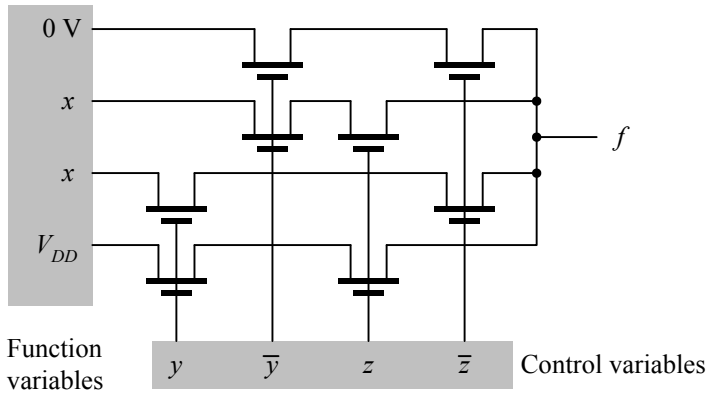


Figure 1.21: The realization of $f(x, y, z) = xy + yz + xz$ with a 0/1-$x/\bar{x}$-tree network.

The following example demonstrates the situation that the f_Y^s of 0/1-$x/\bar{x}$-tree network can be further simplified.

Example 1-10: (A more complicated example of a 0/1-x/$\bar{x}$-tree network.) Realize the following switching function using a 0/1-$x/\bar{x}$-tree network and nMOS switches.

$$f(w, x, y, z) = \overline{\bar{w}\bar{x} + x\bar{y}\bar{z} + wyz}$$

Solution: According to the block diagram shown in Figure 1.20(b) and assuming that an inverter is used at the output to restore the output voltage level, the actual switching function to be realized is the complement of $f(w, x, y, z)$. Let this function be $g(w, x, y, z)$.

$$g(w, x, y, z) = \bar{f}(w, x, y, z) = \bar{w}\bar{x} + x\bar{y}\bar{z} + wyz$$

According to the definition of a 0/1-$x/\bar{x}$-tree network and assuming that $Y = \{w, x, y\}$, the g_Y^0, g_Y^1, g_Y^z, and $g_Y^{\bar{z}}$ are as follows, respectively.

$$\begin{aligned} g_Y^0 &= \bar{w}xy + w\bar{x}\bar{y} \\ g_Y^1 &= \bar{w}\bar{x}\bar{y} + \bar{w}\bar{x}y = \bar{w}\bar{x} \\ g_Y^z &= w\bar{x}y + wxy = wy \\ g_Y^{\bar{z}} &= \bar{w}x\bar{y} + wx\bar{y} = x\bar{y} \end{aligned}$$

The resulting logic circuit is much simpler than an 8-to-1 multiplexer, as shown in Figure 1.22.

1.2.5.5 0/1-Tree Networks The 0/1-tree network is also a special case of uniform-tree network in which all of n variables are set as control variables and hence each residue can only have one of two constant values, 0, and 1. The combinations of control variables (Y) generating the same residue are combined into a switching function, denoted f_Y^0 and f_Y^1. The switching function f_Y^0 corresponds to $\bar{f}$ and f_Y^1 corresponds to f. Each of such switching functions can be further simplified prior to being realized

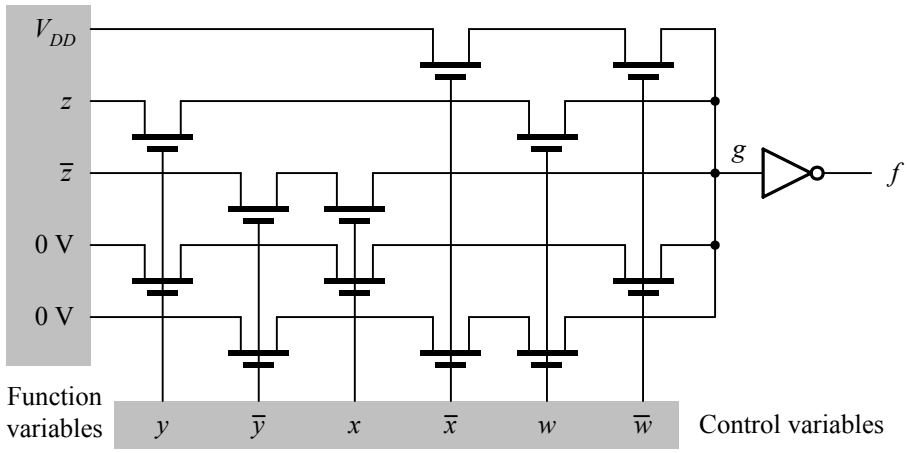


Figure 1.22: The realization of $f(w,x,y,z) = \overline{\bar{w}\bar{x} + x\bar{y}\bar{z} + wyz}$ with a 0/1-$x/\bar{x}$-tree network.

by a TG or an nMOS switch network, or CMOS logic, as the general block diagrams shown in Figure 1.23.

Example 1-11: (A simple example of 0/1-tree network.) Realize the following switching function using a 0/1-tree network with nMOS switches.

$$f(w,x,y,z) = \overline{\bar{w} \cdot x + y \cdot z}$$

Solution: Assuming that an inverter is employed at the output, hence let

$$g(w,x,y,z) = \bar{f}(w,x,y,z) = \bar{w} \cdot x + y \cdot z = g_Y^1$$

Function g realizes the complementary function $\bar{f}$ and function $\bar{g}$ implements the truth function f.

$$\bar{g}(w,x,y,z) = f(w,x,y,z) = \overline{\bar{w} \cdot x + y \cdot z} = (w + \bar{x})(\bar{y} + \bar{z}) = g_Y^0$$

The resulting logic circuit is depicted in Figure 1.24.

Unlike the 0/1-$x/\bar{x}$-tree networks, a mixed use of both nMOS and pMOS switches also finds its widespread use in the realizations of 0/1-tree networks. In this case, the 0/1-tree network is referred to as the $f/\bar{f}$ paradigm. The pMOS switches are used to implement the true function f while nMOS switches are employed to realize the complementary function $\bar{f}$. In a CMOS logic circuit, the $f_Y^1(X)$ tree network is usually called a *pull-up network* (PUN) because it connects the output to V_{DD} when it is on and the $f_Y^0(X)$ tree network is called a *pull-down network* (PDN) because it connects the output to ground when it is on.

Example 1-12: (An example of a 0/1-tree network with CMOS logic.) Realize a two-input NAND gate using a 0/1-tree network with CMOS logic.

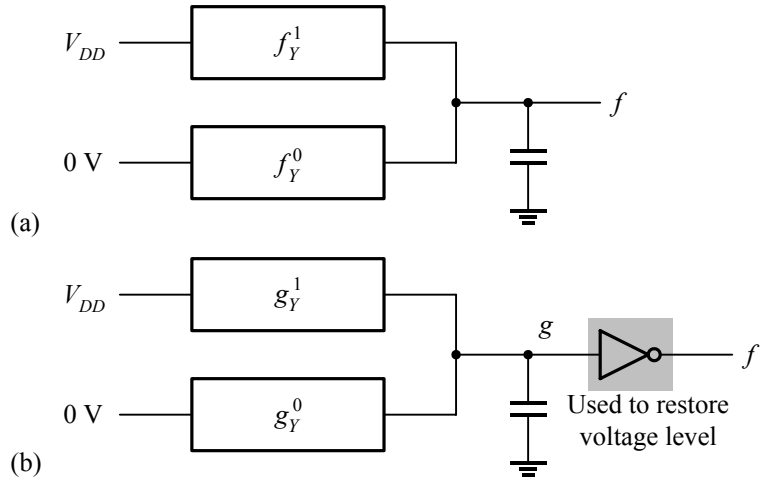


Figure 1.23: The general block diagrams of a 0/1-tree network: (a) using TG switches/CMOS logic; (b) using nMOS switches.

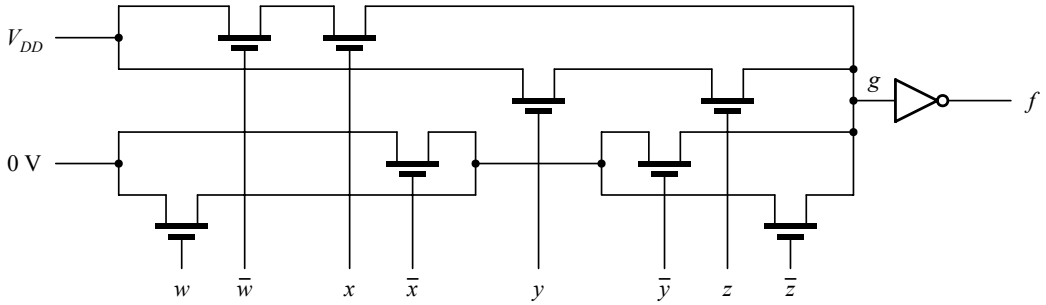


Figure 1.24: The realization of $f(w, x, y, z) = \overline{\bar{w} \cdot x + y \cdot z}$ with a 0/1-tree network.

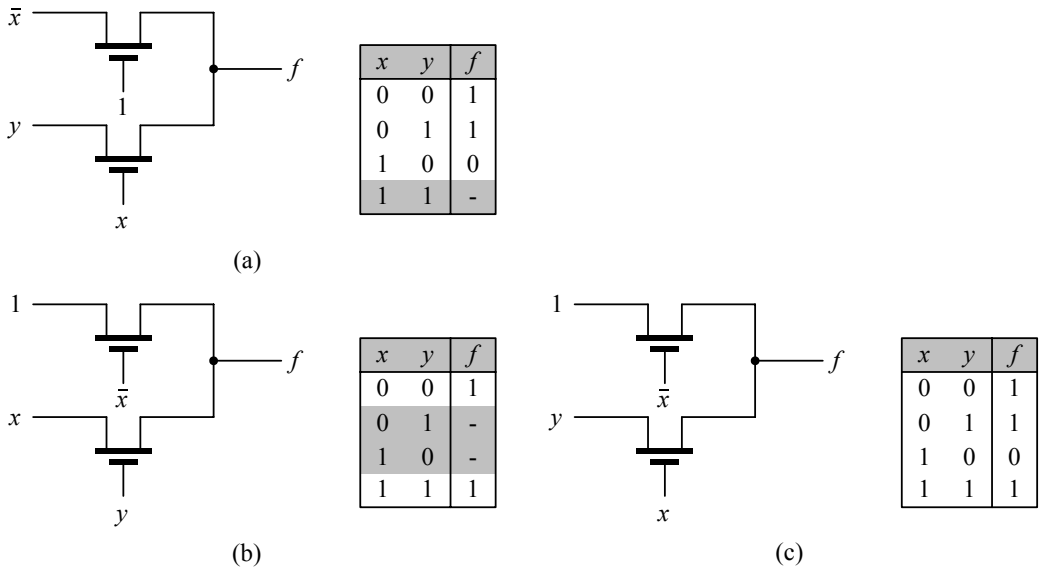
Solution: Since the output f of a two-input NAND gate is equal to 0 only when both inputs x and y are 1 and is 1, otherwise, the switching functions f_Y^0 and f_Y^1 are as follows.

$$\begin{aligned} f_Y^0 &= xy \\ f_Y^1 &= \bar{x}\bar{y} + \bar{x}y + x\bar{y} = \overline{xy} \end{aligned}$$

which are equal to $\bar{f}$ and f, respectively. Hence, it yields the same result as the $f/\bar{f}$ paradigm introduced in Section 1.2.4.

Review Questions

Q1-22. Describe Shannon's expansion theorem.
Q1-23. Define the residue of a switching function $f(x_{n-1}, \cdots, x_1, x_0)$.
Q1-24. Define the control variables and function variables.
Q1-25. What is the rationale behind the $f/\bar{f}$ paradigm?



x	y	f
0	0	1
0	1	1
1	0	0
1	1	-

x	y	f
0	0	1
0	1	-
1	0	-
1	1	1

x	y	f
0	0	1
0	1	1
1	0	0
1	1	1

Figure 1.25: Pitfalls of realizing tree networks based on Shannon's expansion theorem: (a) wrong realization I; (b) wrong realization II; (c) right realization.

Q1-26. What do pull-up and pull-down networks mean?

1.2.5.6 Pitfalls of Realizing Tree Networks When using CMOS switches to realize a tree network decomposed by Shannon's expansion theorem, we have to take care of the places where control and function variables are to be applied. As stated, control variables should be applied to the gates of MOS/TG switches while the function variables and constants should be applied at the inputs (namely, source/drain nodes) of MOS/TG switches. Otherwise, some undefined floating or conflict nodes might arise. Some instances are illustrated in the following example.

Example 1-13: (The effects of misplacing control and function variables.) Supposing that we want to implement the switching function, $f(x, y) = \bar{x} + y$, with nMOS switches, by using Shannon's expansion theorem, the switching function can be decomposed with respect to variable x into the following.

$$f(x, y) = \bar{x} \cdot 1 + x \cdot y$$

where variable x is the control variable and variable y is the function variable. As shown in Figure 1.25(a), since the variable $\bar{x}$ and constant 1 are interchanged, the resulting logic circuit can only correctly function in three out of four combinations of both inputs, x and y. In the fourth case, when both variables x and y are 1, the output node f will get both 1 and 0 at the same time, namely, a conflict situation.

A similar situation occurs when the switching function is realized with the circuit shown in Figure 1.25(b). The output node f will be in a conflict situation and be floated when the variables x and y are 0 and 1, as well as 1 and 0, respectively.

Figure 1.25(c) shows the correct implementation, where the control variable x is applied to the gates of nMOS switches, and the constant 1 and function variable y are

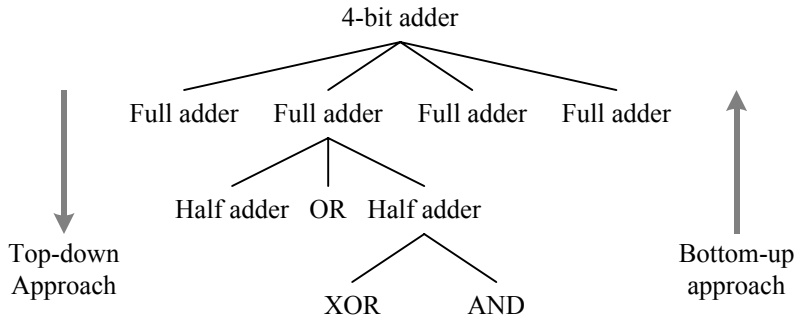


Figure 1.26: The concept of hierarchical design.

applied to the inputs of nMOS switches. It is easy to verify that it correctly functions in all four combinations of the input variables, x and y.

1.3 VLSI Design and Fabrication

The design of a VLSI chip is a complex process and is often proceeded with a hierarchical design process. The rationale behind this is the capability of repeatedly reusing some basic building blocks. To facilitate such reusable building blocks, logic circuits are designed and implemented as cells. Therefore, in the following we first introduce the hierarchical design process and then address the concept of cell designs. After this, we briefly describe the CMOS fabrication process, the layouts of CMOS circuits, and the related layout design rules.

1.3.1 Design Techniques

The design of a VLSI chip is a complex process and is often proceeded with a hierarchical design process, including *top-down* and *bottom-up approaches*. In the design, the term *abstraction* is often used to mean the ability to generalize an object into a set of primitives that can describe the functionality of the object. In the following, we first cope with the concept of hierarchical design and then address the design abstraction of VLSI.

1.3.1.1 Hierarchical Design As in software programming, the *divide-and-conquer paradigm* is usually used to partition a large hardware system into several smaller subsystems. These subsystems are further partitioned into even smaller ones and this partition process is repeated until each subsystem can be easily handled. This system design methodology is known as a *hierarchical design approach*.

Generally speaking, the VLSI design methodology can be classified into two types: the top-down approach and bottom-up approach. In the top-down approach, functional details are progressively added into the system. That is to say, it creates lower-level abstractions from higher levels. These lower-level abstractions are in turn used to create even lower-level abstractions. The process is repeated until the created abstractions can be easily handled. An illustration is depicted in Figure 1.26. The 4-bit adder is decomposed into four full adders, with each consisting of two half adders and an OR gate. The half adder further comprises an XOR gate and an AND gate.

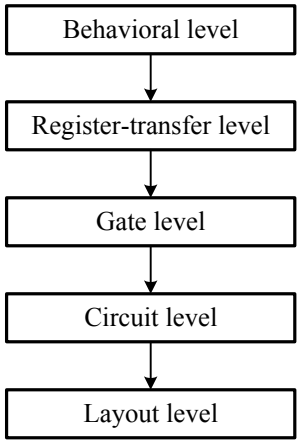


Figure 1.27: The design abstractions of VLSI systems.

In contrast, in the bottom-up approach, the abstractions from lower-level behavior are created and added up to build its higher-level abstractions. These higher-level abstractions are then used to construct even higher-level abstractions. The process is repeated until the entire system is constructed.

However, a practical project usually needs the mixing efforts of both top-down and bottom-up approaches. During the design phase, the top-down approach is often used to partition a design into smaller subsystems; during the realization phase, the bottom-up approach is used to implement and verify the subsystems and their combinations.

The important features of hierarchical design are *modularity*, *locality*, and *regularity*. The modularity means that modules have well-defined functions and interfaces. Locality implies that the detailed implementation of each module is hidden from outside. The use of the module is completely through the well-defined interface. Regularity is sometimes synonymous with *reusability*; namely, modules can be reused many times in a design.

Although there are many factors that also need to be taken into account in designing a VLSI chip, area budget, power dissipation, and performance are the three most important factors that designers should always keep in mind. To make a practical project successful, the designer often needs to trade off among these three factors in some sense during the course of design and implementation.

1.3.1.2 Design Abstractions A VLSI chip is usually designed by following a design process or a synthesis flow. The general VLSI design abstraction levels are shown in Figure 1.27, which includes *behavioral level*, *register-transfer level* (*RTL* or *RT level*), *gate level*, *circuit level*, and *layout level*. In the following, we illustrate this design process with examples.

To begin with, a set of requirements, also called specifications, tell what the system should do, how fast it should run, what amount of power dissipation it could have, and so on. The specifications are not a design, and are often incomplete and subject to be changed later. The following is an example of specifications.

Specifications: Design an adder to calculate the sum of two numbers.

The abstraction levels of this design may be described in either a schematic or text way.

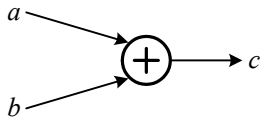


Figure 1.28: A behavioral level for an adder to calculate the sum of two numbers.

Behavioral level. Once the specifications of a VLSI chip design have been set up, the next step is to model it at a behavioral level. At the behavioral level, the chip design is generally modeled as a set of executable programs, usually written in C, Verilog HDL, VHDL, or other high-level programming languages, and is much more precise than the specifications. Through the careful verification of behavioral modeling, we could know whether the specifications meet the requirements or not. The simulation results of behavioral modeling are used to annotate the specifications of chip design.

Figure 1.28 shows the behavioral level for adding up two 4-bit numbers. Here, an adder is employed to calculate the sum of two numbers. At this behavioral level, only the behavior of design is verified. It leaves the area, delay, and power dissipation, of the resulting adder untouched. A possible text description in Verilog HDL of the adder is as follows.

```
module adder_four_bit_behavior(
       input  [3:0] x, y,
       input  c_in,
       output [3:0] sum,
       output c_out);
// the body of an n-bit adder
always @(x or y)
   {c_out, sum} = x + y + c_in;
endmodule
```

Register-transfer level. When the specifications are assured, the next step is to describe the more detailed design at RTL. At the RTL, a design is described by a set of Boolean functions. The input and output values of the design on every cycle are exactly known. The components used in RTL are the basic combinational and sequential logic modules such as full-adders, decoders, encoders, multiplexers, demultiplexers, registers, counters, and so on. At this level, the delay and area can be roughly estimated.

Figure 1.29 depicts the RTL structure of the 4-bit adder. From the figure, we can see that the 4-bit adder consists of four 1-bit adders in turn. A possible text description of the adder in Verilog HDL is as follows.

```
// gate-level description of 4-bit adder
module adder_four_bit_RTL (
       input  [3:0] x, y,
       input  c_in,
       output [3:0] sum,
       output c_out);
wire   c1, c2, c3; // intermediate carries
// -- four_bit adder body-- //
// instantiate the full adder
   full_adder fa_1 (x[0], y[0], c_in, c1, sum[0]);
```

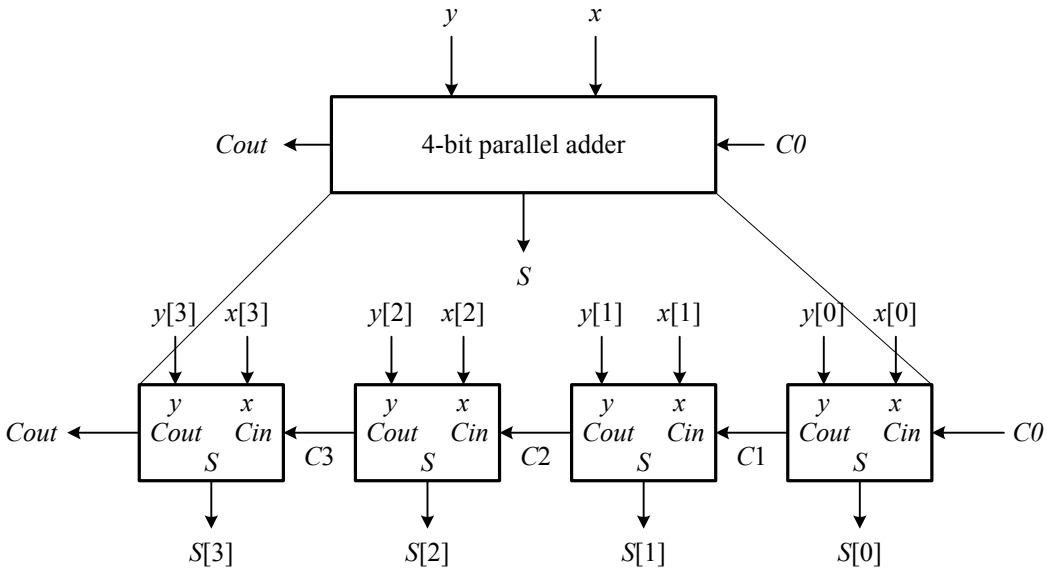


Figure 1.29: An RTL example of a 4-bit adder.

```
    full_adder fa_2 (x[1], y[1], c1, c2, sum[1]);
    full_adder fa_3 (x[2], y[2], c2, c3, sum[2]);
    full_adder fa_4 (x[3], y[3], c3, c_out, sum[3]);
endmodule
```

Gate level. The gate level of a design is a structure of gates, flip-flops, and latches. At this level, more accurate delay and area can be estimated. To describe a design at the gate level, the switching functions of the design need to be derived. To illustrate this, let us consider the truth table of a full adder shown in Figure 1.30(a). From the Karnaugh maps given in Figure 1.30(b), we can derive the switching functions for c_{out} and sum as in the following.

$$c_{out}(x, y, c_{in}) = c_{in}(x \oplus y) + xy \tag{1.8}$$
$$sum(x, y, c_{in}) = x \oplus y \oplus c_{in} \tag{1.9}$$

If we define the function of a half adder as follows.

$$\begin{aligned} S_{ha}(x, y) &= x \oplus y \\ C_{ha}(x, y) &= x \cdot y \end{aligned} \tag{1.10}$$

Then, a full adder can be represented as a function of two half adders and an OR gate. That is, the carry out and sum of a full adder can be expressed as the following two functions.

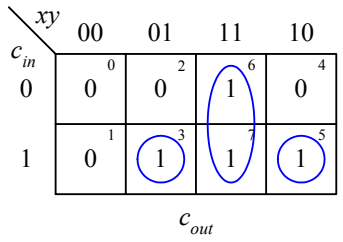
$$\begin{aligned} S_{fa} &= S_{ha2}(C_{in}, S_{ha1}(x, y)) \\ C_{fa} &= C_{ha2}(C_{in}, S_{ha1}(x, y)) + C_{ha1}(x, y) \end{aligned} \tag{1.11}$$

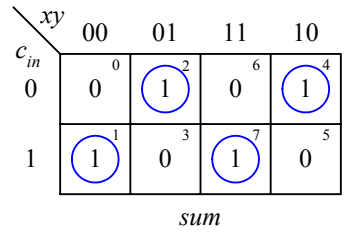
The resulting gate-level full adder is shown in Figure 1.31.

A possible text description of the full adder in Verilog HDL at the gate level is as follows.

x	y	c_{in}	c_{out}	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

(a)





(b)

Figure 1.30: The (a) truth table and (b) Karnaugh maps for c_{out} and sum of a full adder.

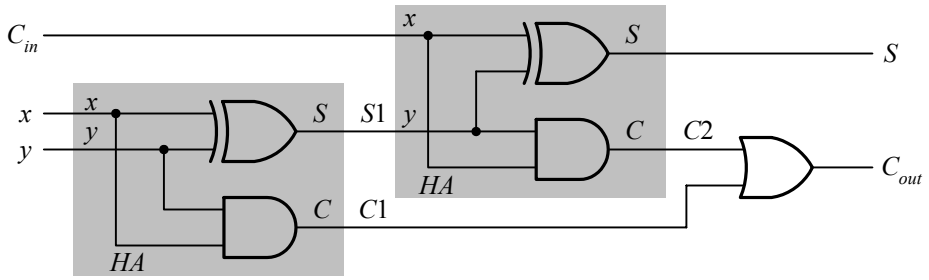


Figure 1.31: A gate-level example of a full adder.

```
// gate-level description of full adder
module full_adder_gate (
       input  x, y, c_in,
       output sum, c_out);
wire s1, c1, c2;  // outputs of both half adders
// -- full adder body-- //
// instantiate the half adder
   half_adder ha_1 (x, y, c1, s1);
   half_adder ha_2 (c_in, s1, c2, sum);
   or (c_out, c1, c2);
endmodule

// gate-level description of half adder
module half_adder_gate (
       input  x, y,
       output c, s);
// -- half adder body-- //
// instantiate primitive gates
   xor (s,x,y);
   and (c,x,y);
endmodule
```

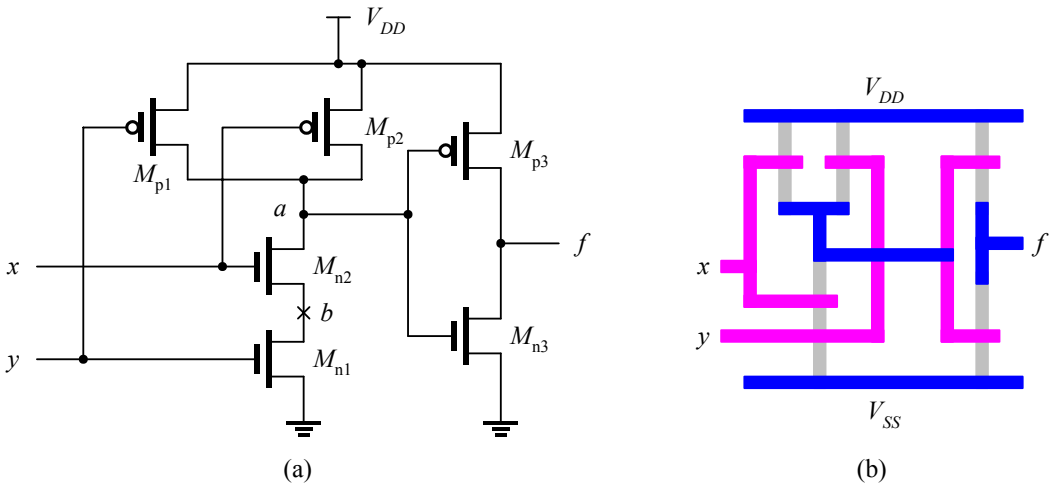


Figure 1.32: The (a) logic circuit and (b) stick diagram of a two-input AND gate.

Circuit level. At the circuit level, the design is implemented with transistors. The circuit level of a two-input AND gate is shown in Figure 1.32. In CMOS technology, the basic gate is an inverting circuit such as a NAND gate or a NOR gate. The simplest way to implement a noninverting gate, such as the AND gate or OR gate, is to add an *inverter* (NOT gate) at the output node of its corresponding basic gate, such as the combination of NAND+NOT or NOR+NOT. Hence, a two-input AND gate is implemented by cascading a two-input NAND gate with an inverter.

A possible text description of the two-input AND gate in SPICE at the circuit level is as follows.

```
AND Gate —— 0.35−um process
******** Parameters and model *********
.lib   '..\models\cmos35.txt' TT
.param Supply=3.3V   * Set value of Vdd
.opt   scale=0.175u
******** AND Subcircuit description **********
.global Vdd Gnd
.subckt AND x y f
Mn1   b   y  Gnd Gnd nmos L = 2 W = 6
Mn2   a   x    b Gnd nmos L = 2 W = 6
Mp1   a   y  Vdd Vdd pmos L = 2 W = 6
Mp2   a   x  Vdd Vdd pmos L = 2 W = 6
Mn3   f   a  Gnd Gnd nmos L = 2 W = 3
Mp3   f   a  Vdd Vdd pmos L = 2 W = 6
.ends
********* Circuit description ***********
Vdd Vdd Gnd 'Supply'
X1  t  w  z  AND  ** instantiate an AND gate
Vt  t Gnd pulse 0 'Supply' 0ps 100ps 100ps 2ns 4ns
Vw  w Gnd pulse 0 'Supply' 0ps 100ps 100ps 1ns 2ns
******** Analysis statement *******
.tran 1ps 4ns
```

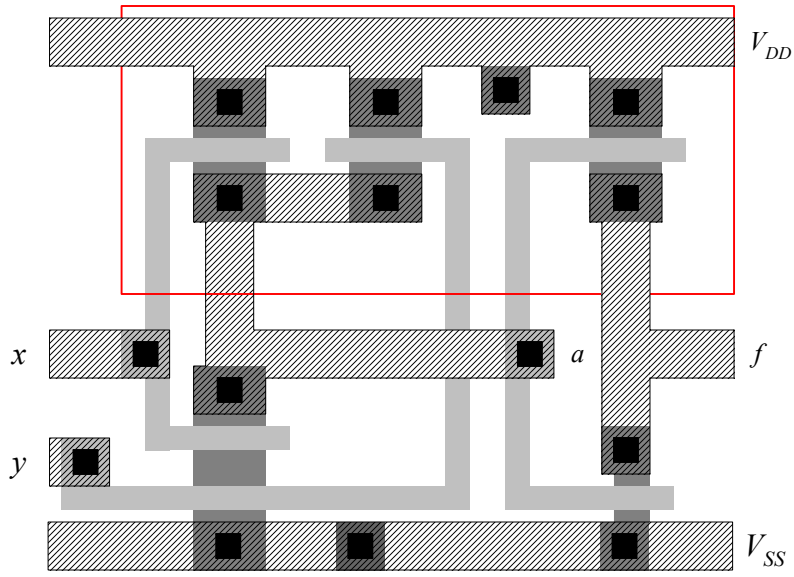


Figure 1.33: A layout example of the two-input AND gate shown in Figure 1.32.

```
******** Output statements *******
.probe V(t) V(w) V(z)
.option post
.end
```

Layout level. The last step of a VLSI chip design is to represent the circuit in mask layout layers. Before proceeding to the design of layout (i.e., physical layout) of a logic circuit, it is useful to draw the logic circuit in a stick diagram. A *stick diagram* is a simplified layout, which only represents the relative relationship among wires and components without being confined to layout design rules. A stick diagram of the AND logic circuit shown in Figure 1.32(a) is depicted in Figure 1.32(b).

After a stick diagram of a circuit is completed, the layout of a logic circuit can be obtained by enforcing the layout design rules to all features of the stick diagram. A complete layout of the AND logic circuit exhibited in Figure 1.32(a) is given in Figure 1.33. This layout defines all mask layout layers required to fabricate the AND logic circuit.

Through these mask layout layers, the design can be manufactured in an IC foundry. After the layout is done, parasitics including resistance, capacitance, and inductance, can be extracted and the performance of layout can then be estimated through circuit-level simulators, such as the SPICE or its descendent programs, including HSPICE and Spectra.

A partial text description of the AND gate after extracting *RC* parasitics from the layout shown in Figure 1.33 is as follows.

```
.subckt PM_AND2   *** F 1 3 15 20 22
c5 20 0 0.333511f
c6 14 0 0.160047f
c7 12 0 0.172583f
r8 17 20 0.0938868
r9 22 16 0.6138
```

```
r10 15 16 1.83333
r11 13 17 0.0195
r12 13 14 0.0599444
r13 12 17 0.0195
r14 11 15 0.0342493
r15 11 12 0.0642778
r16 3 2 0.347907
r17 1 14 0.0314389
r18 1 2 0.833333
.ends
```

The parasitic capacitance and resistance of each node are extracted and represented as a SPICE data file. From this information, the characteristics of the layout can be quantified more accurately.

Review Questions

Q1-27. Describe the divide-and-conquer paradigm.

Q1-28. Distinguish the top-down approach from the bottom-up approach.

Q1-29. Describe the general VLSI design abstraction levels.

Q1-30. What are the three major factors that designers should always keep in mind when designing a VLSI system?

1.3.2 Cell Designs

The essence of hierarchical design is the capability of reusing some basic cells over and over again. During the course of bottom-up composition, a group of cells may be combined into a submodule and it can be in turn used as a submodule for another higher-level submodule. This process continues until the entire system is constructed. Therefore, the basic building blocks are cells. But what kind of circuit can be referred to as a cell? How complex a function should it contain? In fact, the function of a cell could range from a simple gate to a very complicated module, such as an MP3 decoder or even a 32-bit microprocessor.

In the following, we will introduce the three most basic types of cells corresponding to those often introduced in basic texts: combinational cells, sequential cells, and subsystem cells. Their details will be treated in greater detail in dedicated chapters later in the book.

1.3.2.1 Combinational Cells Combinational cells are logic circuits whose outputs are only determined by the present inputs, i.e., only functions of present inputs. The most basic CMOS circuit is the NOT gate (i.e., inverter), which is composed of two MOS transistors, a pMOS transistor, and an nMOS transistor, as shown in Figure 1.34(a). The pMOS transistor connects the output to V_{DD} when it is on whereas the nMOS transistor connects the output to ground when it is on. Since both MOS transistors cannot be turned on at the same time, there is no direct path from power (V_{DD}) to ground. Hence, the power dissipation is quite small, only due to a small leakage current and the transient current during switching. The average power dissipation is on the order of nanowatts. The layout and side view of a CMOS inverter are shown in Figure 1.35.

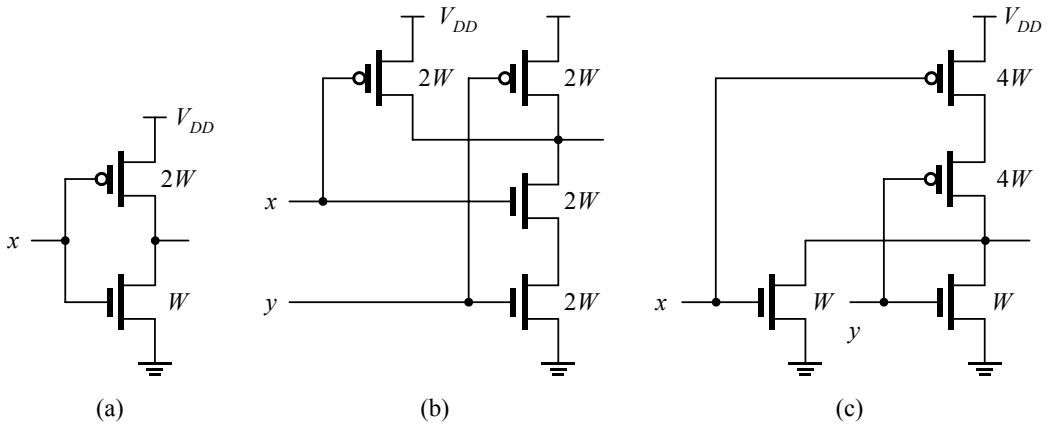


Figure 1.34: The three basic CMOS gates: (a) NOT gate; (b) NAND gate; (c) NOR gate.

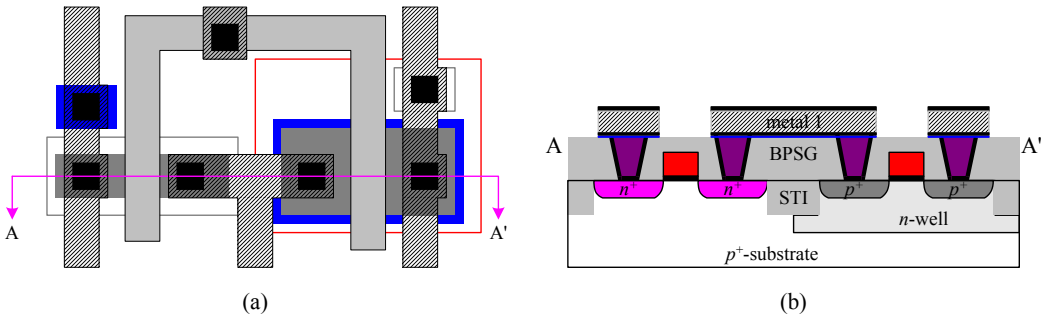


Figure 1.35: The (a) layout and (b) side view of a CMOS inverter.

Combinational cells typically include basic gates: AND, OR, NOT, NAND, NOR, XOR, and XNOR gates. Also, these basic gates are often designed with various transistor sizes in order to provide different current-driving capabilities for different applications. The following examples give some representative combinational cells that are widely used in various digital system designs and will appear over and over again throughout the book.

Example 1-14: (Basic Gates.) Three basic CMOS gates are depicted in Figure 1.34. These are a NOT gate (or inverter), a two-input NAND gate, and a two-output NOR gate. The reader may notice that a basic CMOS gate is an inverting gate; thus, it is necessary to append a NOT gate at the output when a noninverting function is needed. For example, a two-input AND gate is formed by appending an inverter at the output of a two-input NAND gate, as we have done in Figure 1.32.

In addition to basic gates, multiplexers and demultiplexers are often used in digital system designs. A multiplexer is a device capable of routing the input data from one out of its many inputs specified by the value of a set of source selection inputs to its

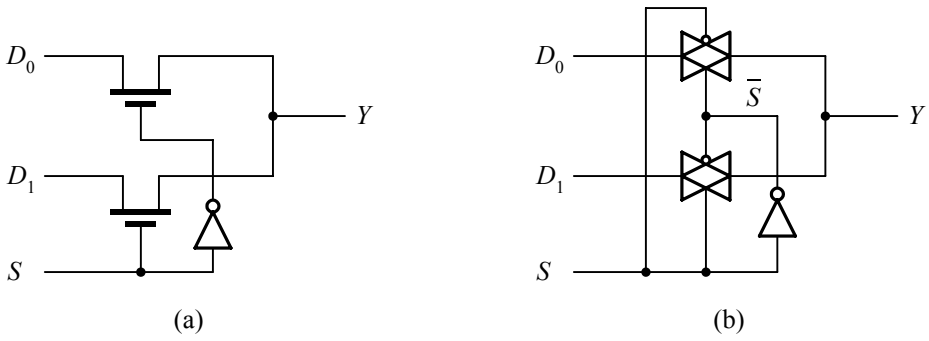


Figure 1.36: The examples of (a) nMOS-based and (b) TG-switch-based 2-to-1 multiplexers.

output. A demultiplexer operates in the reverse way; that is, it routes the input data to one of its many outputs selected by the value of a set of destination selection inputs.

The following two examples separately illustrate how a simple 2-to-1 multiplexer and demultiplexer can be constructed.

Example 1-15: (A 2-to-1 multiplexer.) Figure 1.36 shows two examples of a CMOS 2-to-1 multiplexer. Figure 1.36(a) is constructed by using two nMOS transistors and an inverter. The value of output Y is equal to D_0 or D_1 determined by the value of source selection input S. The upper nMOS transistor is turned on when the source selection input S is 0 and is turned off otherwise. The lower nMOS transistor operates in the opposite way; that is, it is on when the source selection input S is 1 and is turned off otherwise. As a result, it is operated as a 2-to-1 multiplexer. Figure 1.36(b) shows the case when transmission gates are used.

Example 1-16: (A 1-to-2 demultiplexer.) Figure 1.37 shows two examples of a CMOS 1-to-2 demultiplexer. Figure 1.37(a) is constructed by using two nMOS transistors and an inverter. The input D is routed to Y_0 or Y_1 determined by the value of destination selection input S. The input D is routed to Y_0 when the destination selection input S is 0 and routed to Y_1 otherwise. To facilitate this, the upper nMOS transistor is turned on only when the destination selection input S is 0 and the lower nMOS transistor is turned on only when the selection input S is 1. Hence, it results in a 1-to-2 demultiplexer. Figure 1.37(b) shows the case when transmission gates are used.

It is worth noting that the above 2-to-1 multiplexers and 1-to-2 demultiplexers virtually have the same circuit structure. In fact, they are interchangeable. That is, the 2-to-1 multiplexer may be used as a 1-to-2 demultiplexer and vice versa.

1.3.2.2 Sequential Cells Sequential cells are logic circuits whose outputs are not only determined by the current inputs but also dependent on their previous output

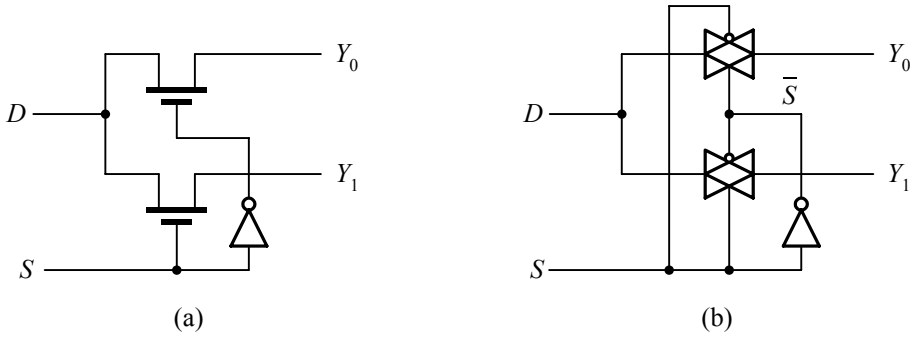


Figure 1.37: The examples of (a) nMOS-based and (b) TG-switch-based 1-to-2 demultiplexers.

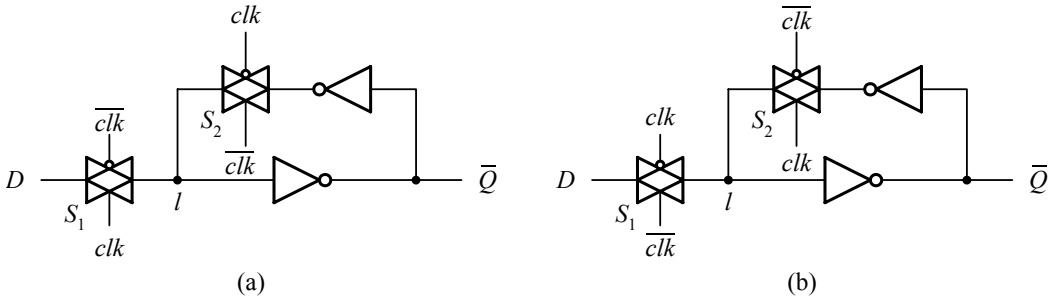


Figure 1.38: The examples of (a) positive and (b) negative D-type latches based on TG switches.

values; namely, their outputs are functions of both current inputs and previous outputs. Typical sequential cells include basic memory devices: *latches* and *flip-flops*.

The following examples first show how two transmission gates (TGs) and two inverters can be combined to build positive and negative latches, and then show how to use these two latches to construct a *master-slave flip-flop*.

Example 1-17: (A positive D-type latch.) Figure 1.38(a) shows a TG-based positive D-type latch. A latch is essentially a *bistable circuit* in which two stable states exist. These two states are called 0 and 1, respectively. In order to change the state of the bistable circuit, an external signal must be directed into the circuit to override the internal state. As shown in the figure, this operation is done by cutting off the feedback path and connecting the external signal to the bistable circuit; that is, it is achieved by turning off switch S_2 and turning on switch S_1. Since the latch receives the input data at the high level of clock *clk* and retains the input data as the clock *clk* falls to 0. Such a type of latch is called a *positive latch*.

The following is an example of a negative D-type latch based on TGs.

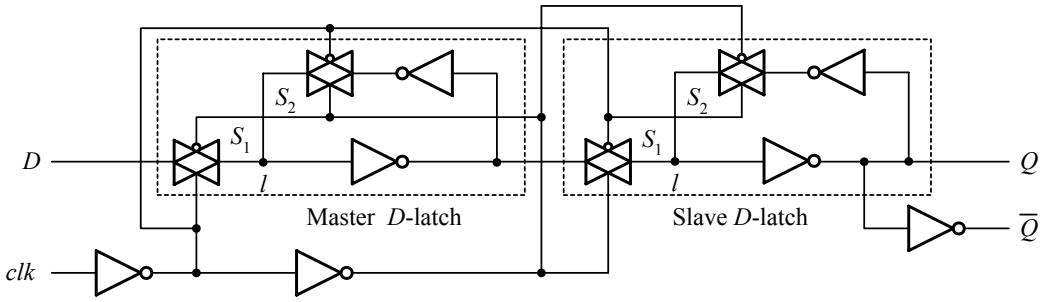


Figure 1.39: A master-slave positive edge-triggered D-type flip-flop.

Example 1-18: (A negative D-type latch.) Figure 1.38(b) shows a TG-based negative latch. It essentially operates in the same way as the positive latch except that now the latch receives the external data at the low level of clock *clk* and retains the input data as the clock *clk* rises high. Such a type of latch is called a *negative latch*.

The following example shows how to combine the above two latches into a master-slave flip-flop. The essential difference between latches and flip-flops is on the transparent property. For both positive and negative latches, their outputs exactly follow their inputs as the clocks are high and low, respectively. This property is known as *transparent property* since under this situation the latch sitting between the output and input seems not to exist at all. For flip-flops, their outputs are only a sample of their inputs at a particular instant of time, usually, the positive or negative edge of clock. Hence, flip-flops do not own the transparent property. The detailed discussion of latches and flip-flops with more examples is deferred to Chapter 9.

Example 1-19: (A master-slave positive edge-triggered D-type flip-flop.) In CMOS technology, a flip-flop is often created by cascading two latches with opposite polarities, that is, a negative latch followed by a positive latch, or the reverse order. Such a flip-flop is called a *master-slave flip-flop*. Figure 1.39 shows a master-slave D-type flip-flop created by cascading negative and positive D-type latches. The input data is routed to the master D-latch when the clock *clk* is low and then to the slave D-latch when the clock *clk* is high. Due to the transparent property of latches, the input data routed to the slave-D latch is the data just before the clock changing from low to high. As a consequence, it indeed functions as a positive edge-triggered D-type flip-flop.

Using the same principle, a negative edge-triggered D-type flip-flop can be built by cascading a positive and a negative D-type latches.

1.3.2.3 Subsystem Cells The third type of cells widely used in VLSI designs are *subsystem cells*. A subsystem cell is a logic circuit being able to function as an independent module such as an 8-bit *arithmetic logic unit* (ALU), an 8-bit or larger multiplier,

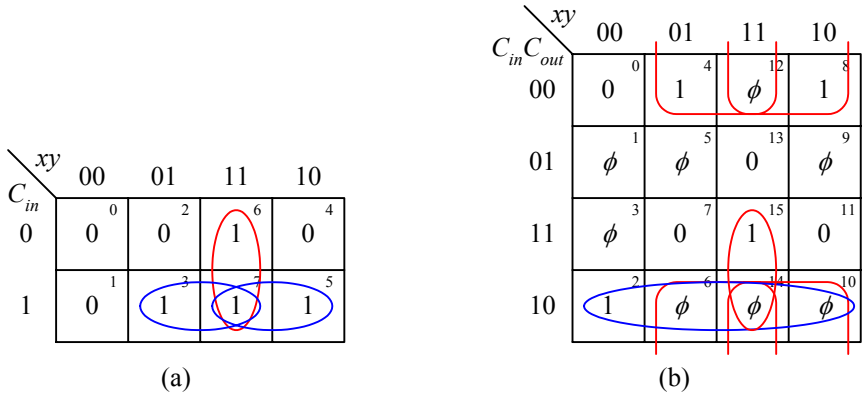


Figure 1.40: The Karnaugh maps for (a) c_{out} and (b) *sum* of a full adder.

and so on. A subsystem cell is usually designed as a single primitive module to optimize an area, a performance, or both.

In the following, we only give the reader a sense for the subsystem cell by using a simple example showing how to directly design and implement a full adder from the circuit level without resorting to the combination of basic gates, such as the one shown in Figure 1.31.

Example 1-20: (A full adder.) At the circuit level, the designed logic circuit is implemented with transistors. The Karnaugh maps for c_{out} and *sum* of a full adder are depicted in Figures 1.40(a) and (b), respectively. From these two Karnaugh maps, the switching functions for c_{out} and *sum* can be obtained and expressed as follows.

$$\begin{aligned} c_{out} &= c_{in}(x+y)+xy \\ sum &= \overline{c_{out}}(x+y+c_{in})+xyc_{in} \end{aligned}$$

The circuit-level realization of the above two functions is shown in Figure 1.41. The general approaches to realizing a switching function using CMOS transistors have been explored in Section 1.2.5 in depth.

In industry, cells with a broad variety of functions and driving capabilities are usually designed in advance. These cells along with their layouts and parameters, such as area and propagation delay, are then aggregated together in a library called a *cell library* so that they can be reused whenever they are needed. The design based on a cell library is referred to as a *cell-based design*. It is often accomplished by using synthesis flow based on Verilog HDL or VHDL.

1.3.3 CMOS Processes

A *CMOS process* is a manufacturing technology capable of incorporating both pMOS and nMOS transistors in the same chip. Due to ultra-low power dissipation and high integration density, CMOS processes have become the major technology for modern VLSI circuits.

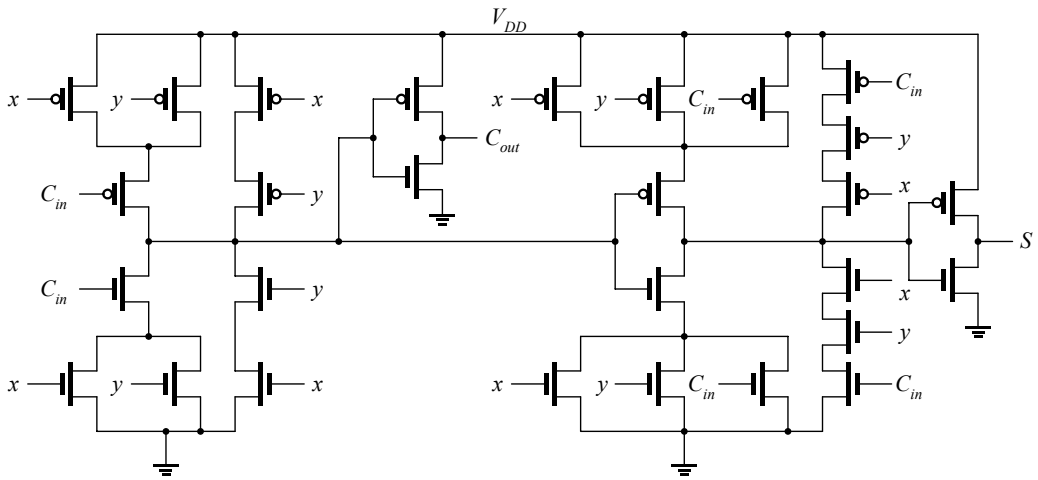


Figure 1.41: A circuit-level implementation of a full adder.

A specific CMOS circuit can be fabricated in a variety of ways. Figure 1.42 shows the three most commonly used CMOS fabrication structures. Figure 1.42(a) is an *n-well* (or *n*-tub) structure, where an *n*-well is first implanted and then a pMOS transistor is built on the surface of it. The nMOS transistor is built on top of the *p*-type substrate. Of course, a CMOS circuit can also be built with a *p*-well on an *n*-type substrate. However, this process yields an inferior performance to the *n*-well process on the *p*-type substrate. Hence, it is not widely used in industry.

In the *n*-well structure, the *n*-type dopant concentration must be high enough to overcompensate for the *p*-type substrate doping in order to form the required *n*-well. One major disadvantage of this structure is that the channel *mobility* is degraded because mobility is determined by the total concentration of dopant impurities, including both *p* and *n* types.

To improve channel mobility, most recent CMOS processes used in industry are the structure called a *twin-well structure* or a *twin-tub*, as shown in Figure 1.42(b). In this structure, a *p*-type epitaxial layer is first grown and then the desired *p*-type and *n*-type wells are separately grown on top of the epitaxial layer. Finally, the nMOS and pMOS transistors are then manufactured on the surface of *p*-type and *n*-type wells, respectively. The field-oxide is grown by using *local oxidation of silicon* (LOCOS). Due to no overcompensation problem, a high channel mobility can be achieved.

The third CMOS structure shown in Figure 1.42(c) is still a twin-well structure but uses *shallow trench isolation* (STI) instead of LOCOS. Due to the lack of a *bird's beak effect* occurring in LOCOS, it can provide higher integration density. This structure is widely used in deep submicron (DSM) processes, in particular, below 0.18 μm. The STI used here has a depth less than 1 μm. Some processes use a deep-trench isolation with a depth deeper than the depth of a well. In such a structure, an oxide layer is thermally grown on the bottom and side walls of the trench. The trench is then refilled by depositing polysilicon or silicon dioxide. The objective of this structure is to eliminate the inherent *latch-up problem* associated with CMOS processes. The latch-up problem is dealt with in more detail in Chapter 4.

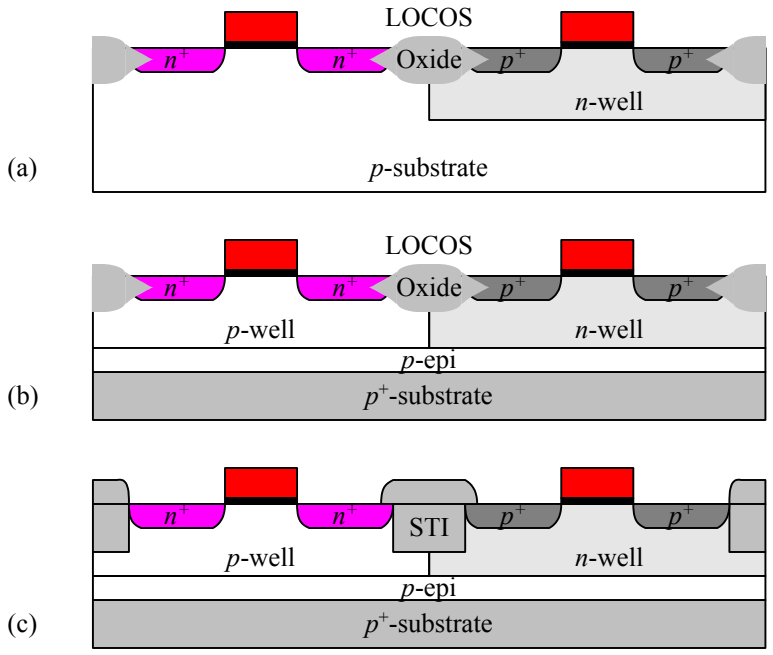


Figure 1.42: An illustration of various CMOS structures: (a) n-well structure; (b) twin-well structure; (c) twin-well structure with refilled trench.

1.3.4 CMOS Layout

As it can be seen from Figure 1.33, a layout is another view of the circuit. In other words, both layout and circuit denote the same thing just like a coin being looked at from both sides. Nevertheless, different viewpoints about a digital integrated circuit do exist between circuit engineering and computer science. From the circuit engineering viewpoint, a digital integrated circuit is a system composed of circuits built on the surface of a silicon chip; from the computer science viewpoint, a digital integrated circuit is a set of geometrical patterns on the surface of a silicon chip.

As a consequence, from the circuit engineering/computer science viewpoint, a VLSI design is a system design discipline with the following two features. First, groups of circuits/patterns, called *modules*, represent different logic functions and can be repeated many times in a system. Second, complexity could be dealt with using the concept of repeated circuits/patterns that are put together hierarchically.

Fundamentally, a circuit is an abstract design that represents only the idea in one's mind. To implement the circuit, physical devices, either discrete devices or integrated circuits, must be used. As a result, a layout of a circuit is also only an abstract representation of the design. It denotes all information required for fabricating the circuit in an IC foundry. To illustrate the relationship between a layout and an actual IC fabrication, consider Figure 1.43, which shows a layout of a CMOS inverter along with the major steps for fabricating the inverter.

A layout of a circuit indeed defines the set of masks needed in manufacturing the circuit. As it can be seen from Figure 1.43(a), there are seven mask layers required for manufacturing such a simple CMOS inverter circuit. It is worthy to note that an IC is made in a layer-by-layer fashion from bottom up. Since an n-well process is assumed to be used, the first mask is employed to define the n-well, where a pMOS transistor

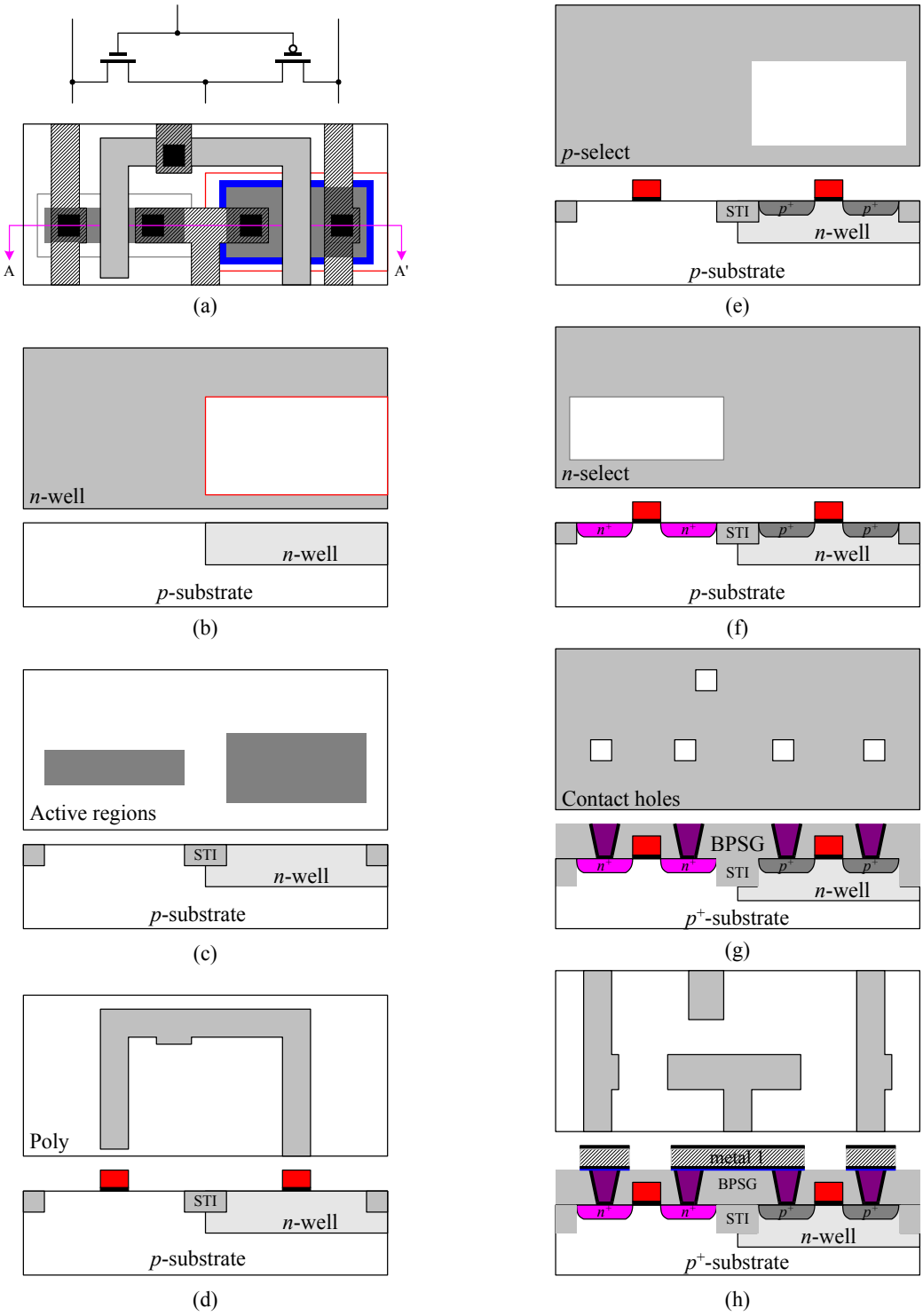


Figure 1.43: A CMOS inverter layout and its related mask set: (a) layout; (b) formation of n-well; (c) definition of nMOS transistors; (d) definition of polysilicon gates; (e) definition of p^+ diffusion; (f) definition of n^+ diffusion; (g) definition of contact holes; (h) definition of metal1 connections.

can be made, as depicted in Figure 1.43(b). After the n-well is defined, it is necessary to reserve areas needed by all MOS transistors and fill the remaining part with *field oxide*, a thick silicon dioxide formed by the STI process, so as to isolate each MOS transistor electrically. This is defined by the active mask, as shown in Figure 1.43(c).

Once the active regions have been defined, the next step is to use the polysilicon mask to make the gates of all MOS transistors, including both pMOS and nMOS transistors, and all wires using polysilicon as well. An illustration is exhibited in Figure 1.43(d). The next two masks are separately used to implant p^+ and n^+ diffusions needed in forming the drain and source regions of MOS transistors. These two masks are called p-select and n-select masks, respectively, and are derived masks obtained by bloating the size of active regions. These two masks and their effects are shown in Figures 1.43(e) and (f), respectively.

The last two masks are used to define contact holes and the metal layer, respectively. As depicted in Figure 1.43(g), all source and drain regions of MOS transistors need to be connected together in the same way as its original circuit so as to perform the same function. To achieve this, a mask is used to define the contact holes through which the required interconnect points can be prepared. The next step is to manufacture the required metal wires. This is done by using the metal mask, as shown in Figure 1.43(h).

Of course, in addition to the above steps there are many steps that still need to be done. Generally, a CMOS circuit usually requires many masks, ranging from seven to over twenty, depending how many metal layers are used. A later chapter is dedicated to the details of CMOS fabrication.

1.3.5 Layout Design Rules

A layout design must follow some predefined rules to specify geometrical objects, namely, polygons, that are either touching or overlapping, on each masking layer. Such a set of predefined rules is called *layout design rules*. The layout design rules may be specified as either of the following two forms:

- *Lambda* (λ) *rules*: In the λ rules, all rules are defined as a function of a single parameter λ. Typically, the feature size of a process is set to be 2λ. The λ rule is a scalable rule and widely used in the academic realm.
- *Micron* (μ) *rules*: In the μ rules, all rules are defined in absolute dimensions and can therefore exploit the features of a given process to its maximum degree. The μ rule is adopted in most IC foundries in industry.

The μ rules tend to differ from company to company and even from process to process. The λ rules are conservative in the sense that they use the integer multiples of λ. We will use λ rules throughout the book. Some sample design rules are shown in Figure 1.44, referring to Chapter 4 for more details.

Review Questions

Q1-31. Draw the logic circuit of a negative edge-triggered D-type flip-flop.

Q1-32. Design a gate-based 2-to-1 multiplexer.

Q1-33. What is the distinction between CMOS-based and gate-based 2-to-1 multiplexers?

Q1-34. Design a gate-based 1-to-2 demultiplexer.

Q1-35. What is the distinction between CMOS-based and gate-based 1-to-2 demultiplexers?

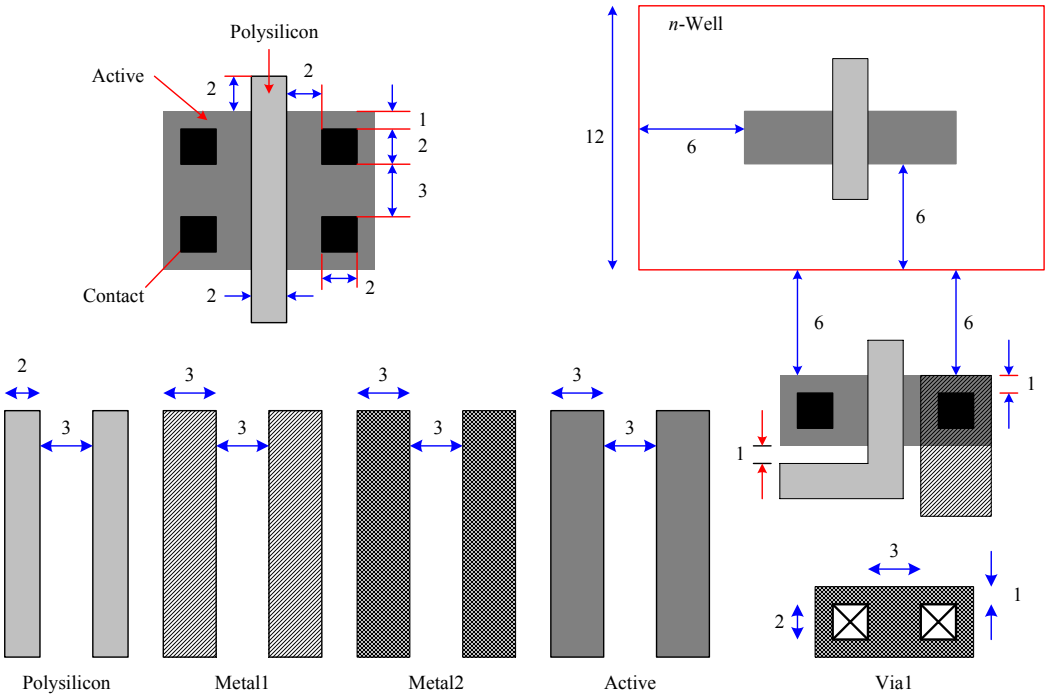


Figure 1.44: A simplified set of layout design rules (all units are in λ).

1.4 Implementation Options of Digital Systems

In this section, we first look at the future trends of VLSI (digital systems) systems design.[4] Then, we introduce a broad variety of implementation options available for digital systems currently.

1.4.1 Future Trends

The NRE cost (fixed cost) and *time to market* are two major factors that affect the future trends of VLSI designs. Recall that the cost of a VLSI chip is determined by the NRE and variable costs. The NRE cost is significantly increased with the decreasing feature sizes of manufacture processes due to the exponentially increased cost of related equipments, photolithography masks, CAD tools, and R&D. Recall that there are three important issues in designing a VLSI system with DSM processes: IR drop, Ldi/dt effect, and hot-spot problems. To accurately model and analyze these three issues, it is inevitable to heavily rely on the aid of computer-aided design (CAD) tools. This means that the expensive CAD tools are indispensable for deep-submicron VLSI designs. To make the product more competitive or acceptable by the end users, the NRE cost has to be reduced profoundly. Consequently, for a VLSI chip to be successful in the market, the product volume must be large enough so as to lower the amortized NRE cost to an acceptable level by the market.

[4]Even though the words "design" and "implementation" mean two different things, they are often strongly related. Hence, we will use these two words interchangeably when their meanings are unambiguous from the context.

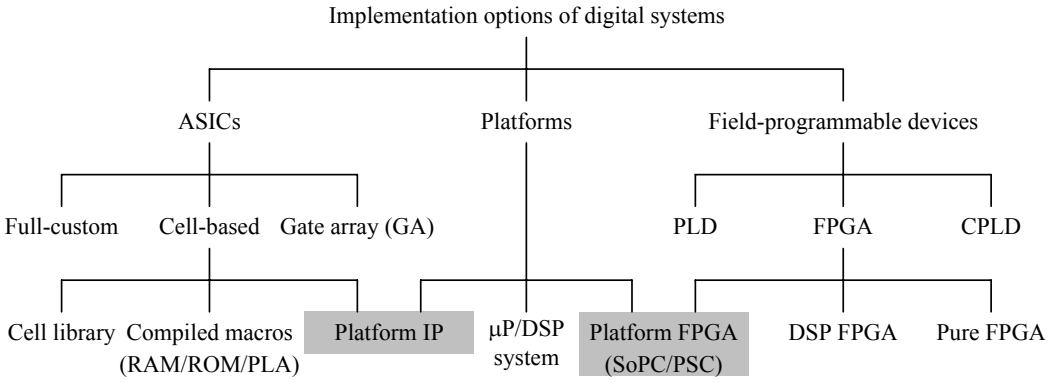


Figure 1.45: Implementation options of digital systems.

Example 1-21: (Amortized NRE cost.) Suppose that the NRE cost for getting the first prototype of a 50 M transistor chip is about 10 M dollars. Calculate the amortized NRE cost if the total product volume of the chip is estimated to be (a). 100 k, and (b). 100 M.

Solution: (a). The amortized NRE cost is $10 \times 10^6/100 \times 10^3 = 100$ dollars. (b). The amortized NRE cost is $10 \times 10^6/100 \times 10^6 = 0.1$ dollars. Therefore, as the product volume increases to some point, the amortized NRE cost might be ignored.

The other factor that affects the future trends of VLSI design is the time to market. Late products are often irrelevant to modern consumer markets. In addition, all electronic products have increasing system complexity with the reduction of feature sizes and hence hardware cost. Although the divide-and-conquer paradigm can be used to partition the system into many smaller modules so that each module can be easily dealt with, the combination of these different functionality modules becomes more difficult and challenging, and the accompanied testing for the combined system is even more complicated. This means that to shorten the time to market of a product, some effective design alternatives must be explored and used.

Based on the aforementioned factors, the future trends of VLSI (digital) system designs can be classified into three classes: ASICs, platforms, and field-programmable devices, as shown in Figure 1.45. One needs to choose an appropriate one from these options to meet the design specifications at the lowest cost and shortest time to market.

1.4.2 Implementation Options

The future trends of VLSI (digital) system implementations can be classified into three classes: *platforms*, *field-programmable devices*, and *ASICs*, as shown in Figure 1.45. We introduce these in the following briefly.

1.4.2.1 Platforms We are first concerned with the platform option. This option includes microprocessor and/or *digital-signal processor* (μP/DSP), platform IP, and platform FPGA. Many consumer products may be designed with a single μP and/or

a simple DSP system. Depending on the desired performance, a variety of 8-bit to 32-bit μPs are available for use today.

A μP/DSP-based system is actually a microcontroller chip and is an embedded system built on silicon. Usually, the system contains an 8-bit or a 32-bit *center-processor unit* (CPU), nonvolatile memory (Flash memory or MRAM), static memory, and a variety of periphery modules, such as *universal serial bus* (USB), *general-purpose input-output* (GPIO), timer, and so on. Most digital systems are designed with this approach.

1.4.2.2 Field-Programmable Devices The field-programmable devices are the second design option. A field-programmable device is the one that contains many uncommitted logic modules that can be committed to the desired functions on-demand in laboratories. Currently, three major types of field-programmable devices are *field-programmable gate arrays* (FPGAs), *complex-programmable logic devices* (CPLDs), and *programmable logic devices* (PLDs). The FPGAs may be further subdivided into pure FPGAs, DSP FPGAs, and platform FPGAs. A platform FPGA combines features from both platforms and field-programmable devices into a single device; it is an FPGA device containing one or more CPUs in a hard, soft, or hardwired IP form, some periphery modules, and field-programmable logic modules.

Here, the IP is the acronym of *intellectual property* and is a predesigned component that can be reused in larger designs. It is often referred to as a *virtual component*. There are three common types of IP: *hard IP*, *soft IP*, and *hardwired IP*. A hard IP comes as a predesigned layout and routing but can be synthesized with other soft IP modules. The block size, performance, and power dissipation of a hard IP can be accurately measured. A soft IP is a synthesizable module in HDL, Verilog HDL or VHDL. Soft IPs are more flexible to new technologies but are harder to characterize, occupy more area, and perform more poorly than hard IPs. A hardwired IP is a circuit module that is already fabricated along with FPGA fabrics.

1.4.2.3 ASICs The third design option is *application-specific integrated circuits* (ASICs). Although the acronym ASIC represents any integrated circuit (IC) defined by a specific application, in industry the term ASIC is often reserved to identify an integrated circuit that needs to be processed in an IC foundry. That is, an ASIC denotes any IC that is designed and implemented with one of the following three methods: *full-custom*, *cell-based*, and *gate array*. One essential feature of an ASIC is that its final logic function needs to be committed through a partial or full set of mask layers in an IC foundry.

Full-custom design starts from scratch and needs to design the layouts of every transistor and wire. It is suitable for very high-regularity chips such as DRAM/SRAM and for very high-speed chips such as CPUs, GPUs, or FPGAs or other special-purpose chips such as GPUs. The cell-based design combines various full-custom cells, blocks, and IPs into a chip. It is suitable for a product with a successful market only. The gate-array-based design combines a variety of IPs and builds the resulting design on a gate array, which is a wafer with prefabricated transistors. It is suitable for a product that has a successful market and needs a short time to market.

In summary, due to enough complexity, FPGA devices will replace ASICs and dominate the market of VLSI-size systems due to the following reasons. First, the NRE cost of an ASIC is profoundly increasing with the reduction of feature sizes of CMOS processes. Second, the functionality of an ASIC chip is increasing with the progress of feature sizes. This makes the design and implementation more complicated,

challenging, and difficult. Third, the increasing complexity of an ASIC design will become a difficult problem for most people.

Review Questions

Q1-36. What are the three major classes of options for designing a VLSI system?
Q1-37. What are the three types of platforms?
Q1-38. What are the three types of field-programmable devices?
Q1-39. What is the meaning of the term ASIC in industry?
Q1-40. What options can be used to design an ASIC?
Q1-41. Explain the reasons why FPGA devices will dominate the VLSI-size market.

1.5 Summary

In the introduction section, we first reviewed the history of VLSI technology in brief. Next, we introduced the silicon planar process on which nowadays CMOS processes are founded, and ultimate limitations of feature size. Then, we were concerned with the classification of VLSI circuits, the benefits of using VLSI circuits, the appropriate technologies for manufacturing VLSI circuits, and scaling theory. Finally, design challenges in terms of DSM devices and wires were also dealt with in detail. Economics and future perspectives of VLSI technology were explored briefly.

Both nMOS and pMOS transistors are regarded as simple ideal switches, known as nMOS switches and pMOS switches, respectively. The combination of both nMOS and pMOS transistors as a combined switch is referred to as a transmission gate (TG) or a CMOS switch. The features of the three types of switches above when used in logic circuits were also discussed and the basic principles of switch logic design were coped with in detail along with examples.

The design of a VLSI chip is a complex process and is often proceeded with a hierarchical design process, including top-down and bottom-up approaches. To facilitate the reusable building blocks, logic circuits are designed and implemented as cells. After introducing the concepts of cell designs, we briefly described the CMOS fabrication process, the layout of CMOS circuits, and the related layout design rules.

The final section first looked at the future trends of VLSI (digital systems) systems design and then introduced the three major classes of implementation options of VLSI systems, including platforms, field-programmable devices, and ASICs.

References

1. B. Bailey, G. Martin, and A. Piziali, *ESL Design and Verification: A Prescription for Electronic System-Level Methodology.* San Francisco: Morgan Kaufmann, 2007.
2. J. A. Cunningham, "The use and evaluation of yield models in integrated circuit manufacturing," *IEEE Trans. on Semiconductor Manufacturing,* Vol. 3, No. 2, pp. 60-71, May 1990.
3. B. Davari,"CMOS technology: present and future," *Symposium VLSI Circuits Digest Technology Papers,* pp. 5-10, 1999.
4. R. Dennard et al.,"Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE J. of Solid State Circuits*, Vol. 9, No. 5, pp. 256-268, October 1974.

5. P. Gelsinger, "Microprocessors for the new millennium: challenges, opportunities, and new frontiers," *Proc. of IEEE Int'l Solid-State Circuits Conf.*, pp. 22-25, 2001.

6. G. Gerosa et al., "A sub-2 W low power IA processor for mobile Internet devices in 45 nm high-k metal gate CMOS," *IEEE J. of Solid State Circuits*, Vol. 44, No. 1, pp. 73-82, January 2009.

7. D. A. Hodges, H. G. Jackson, and R. A. Saleh, *Analysis and Design of Digital Integrated Circuits: In Deep Submicron Technology,* 3rd ed. New York: McGraw-Hill Books, 2004.

8. J. S. Kilby, "Invention of the integrated circuit," *IEEE Trans. on Electronic Devices*, Vol. 23, No. 7, pp. 648-654, July 1976.

9. M. B. Lin, *Digital System Design: Principles, Practices, and ASIC Realization,* 3rd ed. Taipei, Taiwan: Chuan Hwa Book Ltd., 2002.

10. M. B. Lin, *Basic Principles and Applications of Microprocessors: MCS-51 Embedded Microcomputer System, Software, and Hardware,* 2nd ed. Taipei, Taiwan: Chuan Hwa Book Ltd., 2003.

11. M. B. Lin, *Digital System Designs and Practices: Using Verilog HDL and FPGAs.* New York: John Wiley & Sons, 2008.

12. K. Mistry et al., "A 45nm logic technology with high-k+metal gate transistors, strained silicon, 9 Cu interconnect layers, 193 nm dry patterning, and 100% Pb-free packaging," *Proc. of IEEE Int'l Electron Devices Meeting (IEDM),* pp. 247-250, 2007.

13. G. Moore, "No exponential is forever: but 'forever' can be delayed!" *Proc. of IEEE Int'l Solid-State Circuits Conf.*, pp. 1-19, 2003.

14. G. Moore, "Cramming more components onto integrated circuits," *Electronics,* Vol. 38, No. 8, April 1965.

15. R. S. Muller and T. I. Kamins, *Device Electronics for Integrated Circuits,* 3rd ed. New York: John Wiley & Sons, Inc., 2003.

16. E. Nowak, "Maintaining the benefits of CMOS scaling when scaling bogs down," *IBM J. of Research and Development,* Vol. 46, No. 2/3, pp. 169-180, March/May 2002.

17. S. Parihar et al.,"A high density 0.10 μm CMOS technology using low-k dielectric and copper interconnect," *Proc. of IEEE Int'l Electron Devices Meeting (IEDM),* pp. 11.4.1-11.4.4, 2001.

18. S. Rusu et al., "A 45 nm 8-core enterprise Xeon processor," *IEEE J. of Solid State Circuits*, Vol. 45, No. 1, pp. 7-14, January 2010.

19. D. Sylvester and K. Keutzer,"Getting to the bottom of deep submicron," *Proc. of IEEE/ACM Int'l Conf. of Computer-Aided Design,* pp. 203-211, 1998.

20. S. Tyagi, "An advanced low power, high performance, strained channel 65 nm technology," *Proc. of IEEE Int'l Electron Devices Meeting (IEDM),* pp. 1070-1072, 2005.

21. J. P. Uyemura, *Introduction to VLSI Circuits and Systems.* New York: John Wiley & Sons, Inc., 2002.

22. F. Wanlass and C. Sah, "Nanowatt logic using field effect metal-oxide semiconductor triodes," *Proc. of IEEE Int'l Solid-State Circuits Conf.*, pp. 32-33, 1963.

Problems

1-1. Suppose that the diameter of a wafer is d and each die has an area of $A = a \times a$. Show that the number of dies in a wafer, excluding fragmented dies on the boundary, can be approximated as the following equation:

$$\text{Dies per wafer} = \frac{3}{4}\frac{d^2}{A} - \frac{1}{2\sqrt{A}}d$$

1-2. Assume that the diameter of a wafer is 30 cm and the die area is 0.65 cm^2. The defect density D_0 is 0.5 defects/cm^2 and the manufacturing complexity α is 4. The wafer price is 1,200 USD. Calculate the cost of each die without involving the fixed cost.

1-3. Assume that the diameter of a wafer is 30 cm and the die area is 0.86 cm^2. The defect density D_0 is 0.6 defects/cm^2 and the manufacturing complexity α is 4. The wafer price is 1,200 USD. Calculate the cost of each die without involving the fixed cost.

1-4. A half subtractor (also subtracter) is a device that accepts two inputs, x and y, and produces two outputs, b and d. The full subtractor is a device that accepts three inputs, x, y, and b_{in}, and produces two outputs, b_{out} and d, according to the truth table shown in Table 1.3.

Table 1.3: The truth table of a full subtractor.

x	y	b_{in}	b_{out}	d
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

(a) Derive the minimal expressions of both b_{out} and d of the full subtractor.

(b) Draw the logic diagram of switching expressions: b_{out} and d in terms of two half subtractors and one two-input OR gate.

1-5. Consider the logic circuit shown in Figure 1.46:

(a) Show that the logic circuit cannot correctly realize the switching function: $f(x, y, z) = xy + yz + xz$.

(b) Give a correct version using $f/\bar{f}$ design paradigm.

1-6. A universal logic module is a circuit that can realize any switching function by only using the circuit as many copies as needed. Show that both circuits shown in Figure 1.47 are universal logic modules if both true and complementary forms of control variable are available.

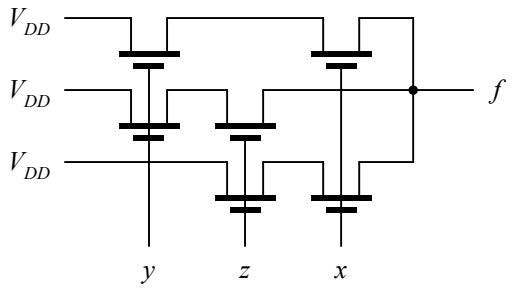


Figure 1.46: An example of an incorrect logic circuit.

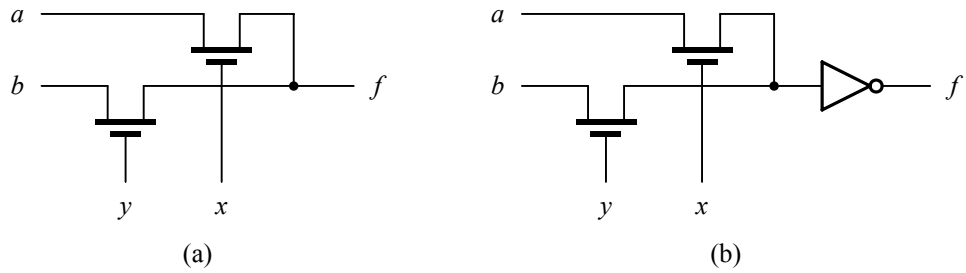


Figure 1.47: Two examples of universal logic modules: (a) unbuffered; (b) buffered.

1-7. Use a combination of CMOS logic gates to realize each of the following switching functions.

(a) $f(x, y, z) = \bar{x}\bar{y} + \bar{x}\bar{z} + \bar{y}\bar{z}$

(b) $f(w, x, y, z) = \overline{w \cdot \overline{(x + y)} + y \cdot z}$

Suppose that only true literals are available. Your implementation needs to use the minimal number of transistors.

1-8. Using the $f/\bar{f}$ paradigm, implement the following logic gates and sketch their switch logic circuits.

(a) Three-input AND gate

(b) Three-input OR gate

1-9. Using the $f/\bar{f}$ paradigm, design a CMOS complex logic gate to realize each of the switching functions.

(a) $f(w, x, y, z) = \overline{wx + x \cdot y \cdot z}$

(b) $f(w, x, y, z) = \overline{w \cdot (x + y + z)}$

1-10. Suppose that both true and complementary forms of variables are available. Using the $f/\bar{f}$ paradigm, implement the following specified logic circuits.

(a) Two-input XOR gate

(b) Two-input XNOR gate

(c) 2-to-1 multiplexer

1-11. Using the following switching functions, verify the validity of Shannon's expansion theorem.

(a) $f(w,x,y,z) = \overline{w+x} \cdot (y \cdot z)$

(b) $f(w,x,y,z) = \bar{w} \cdot (\overline{x+y} + z)$

Assume that the control variable is x.

1-12. Compute the residues of the following switching functions with respect to all combinations of variables w, x, and y.

(a) $f(w,x,y,z) = \bar{w}\bar{x} + xy\bar{z} + wyz$

(b) $f(w,x,y,z) = \bar{w}y + xyz + \bar{w}\bar{y}z$

1-13. Compute the residues of the following switching functions with respect to all combinations of variables x, y, and z.

(a) $f(w,x,y,z) = \sum(0,1,4,6,7,9,11,13,15)$

(b) $f(w,x,y,z) = \sum(1,3,6,7,9,11,12) + \sum_{\phi}(2,5,13,15)$

1-14. Assuming that freeform-tree networks are used, plot the simplified decomposition trees for the following switching functions under the specified decomposition order.

(a) $f(w,x,y,z) = \bar{w}\bar{x}\bar{y}z + \bar{w}x\bar{y} + \bar{w}yz + w\bar{y}\bar{z} + wxy\bar{z} + wxz$. The function is first decomposed with respect to variable w. Then, the decomposition sequence of function f_w is variables z and y, and of function $f_{\bar{w}}$ is y and z.

(b) $f(w,x,y,z) = w\bar{x}\bar{z} + w\bar{x}y\bar{z} + w\bar{x}z + wx\bar{y}\bar{z} + \bar{w}x\bar{y}z + xyz$. The function is first decomposed with respect to variable x. Then, the decomposition sequence of function f_x is y and z, and of function $f_{\bar{x}}$ is variables z and w.

1-15. Assuming that uniform-tree networks are used, plot the simplified decomposition trees for the following switching functions under the specified decomposition order.

(a) $f(w,x,y,z) = \bar{x}\bar{y}z + w\bar{x}y + x\bar{y} + xy\bar{z}$. The decomposition order is x and y.

(b) $f(w,x,y,z) = \bar{x}\bar{y}z + \bar{w}\bar{x}y\bar{z} + w\bar{x}yz + wx\bar{y} + \bar{w}xyz + wxy\bar{z}$. The decomposition order is x, y, and w.

1-16. Using freeform-tree networks with nMOS switches, implement the following logic gates and sketch their switch logic circuits. Assume that the output inverter is not needed.

(a) Three-input NAND gate

(b) Three-input NOR gate

1-17. Realize each of the following switching functions using a freeform-tree network with nMOS switches.

(a) $f(x, y, z) = \bar{x}\bar{y} + \bar{x}\bar{z} + \bar{y}\bar{z}$

(b) $f(x, y, z) = xy + yz + xz$

1-18. Using a freeform-tree network with nMOS switches, design a complex logic gate to realize each of the switching functions.

(a) $f(w, x, y, z) = \overline{w + x \cdot y \cdot z}$

(b) $f(w, x, y, z) = \overline{(w + x) \cdot (y + z)}$

1-19. Using a uniform-tree network with nMOS switches, design a complex logic gate to realize each of the switching functions.

(a) $f(w, x, y, z) = \overline{w \cdot (x + y + z)}$

(b) $f(w, x, y, z) = \overline{(w \cdot x) + (y \cdot z)}$

1-20. Implement the following switching functions with nMOS switches using uniform-tree networks.

(a) $f(w, x, y, z) = \bar{w}x + x\bar{y}z + wy\bar{z}$

(b) $f(w, x, y, z) = w\bar{x} + \bar{x}\bar{y}z + wyz$

1-21. Assuming that the inverter at the output is not needed, use 0/1-$x/\bar{x}$-tree networks with nMOS switches to implement the following logic gates.

(a) Three-input NAND gate

(b) Three-input NOR gate

1-22. Realize each of the following switching functions using a 0/1-$x/\bar{x}$-tree network with TG switches.

(a) $f(x, y, z) = \bar{x}\bar{y} + \bar{x}\bar{z} + \bar{y}\bar{z}$

(b) $f(x, y, z) = xy + yz + xz$

1-23. Using a 0/1-$x/\bar{x}$-tree network with nMOS switches, design a logic circuit to realize each of the switching functions.

(a) $f(w, x, y, z) = \overline{w + x \cdot y \cdot z}$

(b) $f(w, x, y, z) = \overline{w \cdot (x + y + z)}$

1-24. Implement the following switching functions with nMOS switches using 0/1-$x/\bar{x}$-tree networks.

(a) $f(w, x, y, z) = \sum(3, 4, 5, 7, 9, 13, 14, 15)$

(b) $f(w, x, y, z) = \sum(3, 5, 6, 7, 9, 12, 13, 15)$

1-25. Consider a four-input NAND gate and answer the following questions.

(a) Sketch a CMOS logic circuit.

(b) Sketch a stick diagram.

(c) Estimate the area of your four-input NAND gate from the stick diagram.

(d) Using a CAD tool of your choice, layout your four-input NAND gate.

1-26. Consider a four-input NOR gate and answer the following questions.

(a) Sketch a CMOS logic circuit.
(b) Sketch a stick diagram.
(c) Estimate the area of your four-input NOR gate from the stick diagram.
(d) Using a CAD tool of your choice, layout your four-input NOR gate.

1-27. Consider a CMOS compound OR-AND-INVERT (OAI21) gate computing a switching function of $f(x, y, z) = \overline{(x + y) \cdot x}$.

(a) Sketch a CMOS logic circuit.
(b) Sketch a stick diagram.
(c) Estimate the area of your OAI21 gate from the stick diagram.
(d) Using a CAD tool of your choice, layout your OAI21 gate.

1-28. Consider a CMOS compound OR-AND-INVERT (OAI22) gate computing a switching function of $f(w, x, y, z) = \overline{(w + x) \cdot (y + z)}$.

(a) Sketch a CMOS logic circuit.
(b) Sketch a stick diagram.
(c) Estimate the area of your OAI22 gate from the stick diagram.
(d) Using a CAD tool of your choice, layout your OAI22 gate.

1-29. A majority circuit is a circuit that outputs 1 whenever more than half of its inputs are 1. Assume that only three inputs are considered.

(a) Sketch a CMOS logic circuit.
(b) Sketch a stick diagram.
(c) Estimate the area of your majority circuit from the stick diagram.
(d) Using a CAD tool of your choice, layout your majority circuit.

1-30. A minority circuit is a circuit that outputs 1 whenever less than half of its inputs are 1. Assume that only three inputs are considered.

(a) Sketch a CMOS logic circuit.
(b) Sketch a stick diagram.
(c) Estimate the area of your minority circuit from the stick diagram.
(d) Using a CAD tool of your choice, layout your minority circuit.

This page intentionally left blank