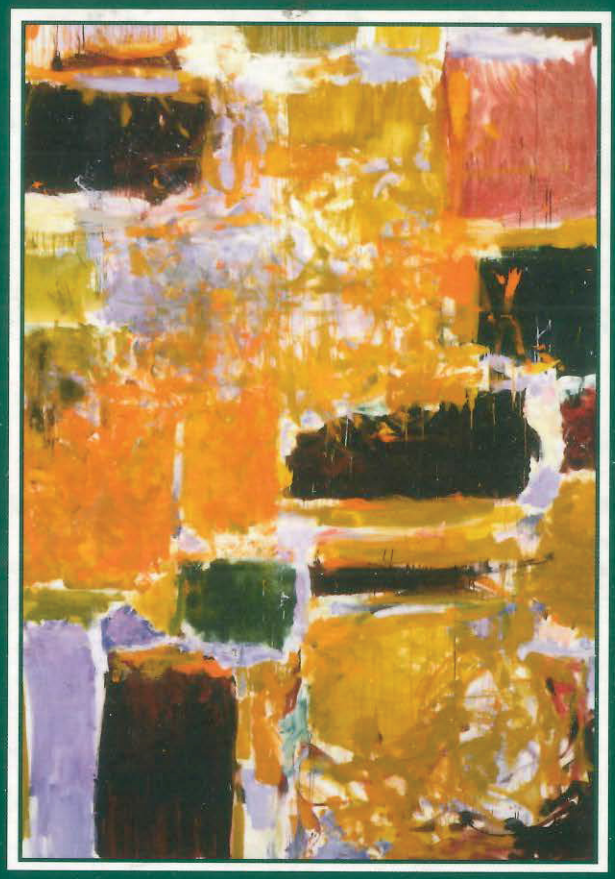
DIGITAL INTEGRATED CIRCUITS

A DESIGN PERSPECTIVE

SECOND EDITION



JAN M. RABAEY
ANANTHA CHANDRAKASAN
BORIVOJE NIKOLIĆ

PRENTICE HALL ELECTRONICS AND VLSI SERIES
CHARLES G. SODINI, SERIES EDITOR

DIGITAL INTEGRATED CIRCUITS

A DESIGN PERSPECTIVE

Prentice Hall Electronics and VLSI Series

Charles S. Sodini, Series Editor

LEE, SHUR, FIELDLY, YTTERDAL *Semiconductor Devices Modeling for VLSI*

LEUNG *VLSI for Wireless Communications*

PLUMMER, DEAL, GRIFFIN *Silicon VLSI Technology: Fundamentals, Practice, and Modeling*

RABAEY, CHANDRAKASAN, NIKOLIĆ *Digital Integrated Circuits: A Design Perspective*, Second Edition

DIGITAL INTEGRATED CIRCUITS

A DESIGN PERSPECTIVE
SECOND EDITION

JAN M. RABAEY
ANANTHA CHANDRAKASAN
BORIVOJE NIKOLIĆ

PRENTICE HALL ELECTRONICS AND VLSI SERIES
CHARLES G. SODINI, SERIES EDITOR

Pearson Education, Inc.
Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data on file.

Vice President and Editorial Director, ECS: *Marcia J. Horton*
Publisher: *Tom Robbins*
Editorial Assistant: *Eric Van Ostenbridge*
Vice President and Director of Production and Manufacturing, ESM: *David W. Riccardi*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Production Editor: *Daniel Sandin*
Director of Creative Services: *Paul Belfanti*
Creative Director: *Carole Anson*
Art and Cover Director: *Jayne Conte*
Art Editor: *Greg Dulles*
Manufacturing Manager: *Trudy Pisciotti*
Manufacturing Buyer: *Lisa McDowell*
Marketing Manager: *Holly Stark*

About the Cover: Detail of "Wet Orange," by Joan Mitchell (American, 1925–1992). Oil on canvas, 112 × 245 in. (284.5 × 622.3 cm). Carnegie Museum of Art, Pittsburgh, PA. Gift of Kaufmann's Department Store and the National Endowment for the Arts, 74.11. Photograph by Peter Harholdt, 1995.

Pearson Education, Inc.
Upper Saddle River, NJ 07458

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN 0-13-090996-3

Pearson Education Ltd., *London*
Pearson Education Australia Pty, Ltd., *Sydney*
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., *Hong Kong*
Pearson Education Canada Inc., *Toronto*
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education—Japan, *Tokyo*
Pearson Education Malaysia, Pte. Ltd.
Pearson Education Inc., *Upper Saddle River, New Jersey*

To Kathelijn, Karthiyayani, Krithivasan, and our Parents

"Qu'est-ce que l'homme dans la nature?
Un néant a l'égard de l'infini,
un tout al l'égard du néant,
un milieu entre rien et tout."

"What is man in nature?
Nothing in relation to the infinite,
everything in relation to nothing,
a mean between nothing and everything."

Blaise Pascal, *Pensées*, n. 4, 1670.

Contents

PART

1

The Fabrics

"The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term, this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000."

Gordon Moore,
Cramming more Components onto
Integrated Circuits, (1965).

CHAPTER

1

Introduction

The evolution of digital circuit design

Compelling issues in digital circuit design

How to measure the quality of a design

Valuable references

1.1 A Historical Perspective

The concept of digital data manipulation has made a dramatic impact on our society. One has long grown accustomed to the idea of digital computers. Evolving steadily from mainframe and minicomputers, personal and laptop computers have proliferated into daily life. More significant, however, is a continuous trend towards digital solutions in all other areas of electronics. Instrumentation was one of the first noncomputing domains where the potential benefits of digital data manipulation over analog processing were recognized. Other areas such as control were soon to follow. Only recently have we witnessed the conversion of telecommunications and consumer electronics into the digital format. Increasingly, telephone data is transmitted and processed digitally over both wired and wireless networks. The compact disk has revolutionized the audio world, and digital video is following in its footsteps.

The idea of implementing computational engines using an encoded data format is by no means an idea of our times. In the early 19th century, Babbage envisioned large-scale mechanical computing devices, which he called *Difference Engines* [Swade93]. Although these engines use the decimal number system rather than the binary representation now common in modern electronics, the underlying concepts are very similar. The Analytical Engine, developed in 1834, was perceived as a general-purpose computing machine, with features strikingly close to modern computers. Besides executing the basic repertoire of operations (addition, subtraction, multiplication, and division) in arbitrary sequences, the machine operated in a two-cycle sequence, called "store" and "mill" (execute), not unlike today's computers. It even used pipelining to speed up the execution of the addition operation! Unfortunately, the complexity and the cost of the designs made the concept impractical. For instance, the design of Difference Engine I (part of which is shown in Figure 1-1) required 25,000 mechanical parts at a total cost of £17,470 (in 1834!).

The electrical solution turned out to be more cost effective. Early digital electronics systems were based on magnetically controlled switches (or relays). They were mainly used in the implementation of very simple logic networks. Such systems are still used in train safety systems. The age of digital electronic computing only started in full with the introduction of the vacuum tube. While originally used almost exclusively for analog processing, it soon was recognized that the vacuum tube was useful for digital computations as well. Not long thereafter, the first complete computers were realized. The era of the vacuum-tube-based computer culminated in the design of machines like the ENIAC (intended for computing artillery firing tables) and the UNIVAC I (the first successful commercial computer). To get an idea about *integration density*, the ENIAC was 80 feet long, 8.5 feet high, and several feet wide; it also incorporated 18,000 vacuum tubes. It became rapidly clear, however, that this design technology had reached its limits. Reliability problems and excessive power consumption made the implementation of larger engines economically and practically infeasible.

All changed with the invention of the *transistor* at Bell Telephone Laboratories in 1947 [Bardeen48], followed by the introduction of the bipolar junction transistor by Schockley in

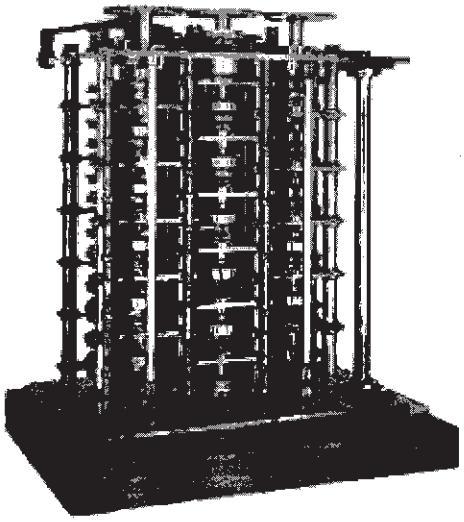


Figure 1-1 Working part of Babbage's Difference Engine I (1832), the first known automatic calculator (from [Swade93], courtesy of the Science Museum of London).

1949 [Schockley49].[1] It took until 1956 before this led to the first bipolar digital logic gate, made of discrete components introduced by Harris [Harris56]. In 1958, Jack Kilby at Texas Instruments conceived the *integrated circuit (IC)*, in which all components, passive and active, are integrated on a single semiconductor substrate—he was awarded the Nobel prize for this breakthrough. It led to the introduction of the first set of integrated-circuit commercial logic gates, called the *Fairchild Micrologic family* [Norman60]. The first truly successful IC logic family, *TTL (Transistor-Transistor Logic)* was pioneered in 1962 [Beeson62]. Other logic families were devised with higher performance in mind. Examples of these are the current switching circuits that produced the first subnanosecond digital gates and culminated in the *ECL (Emitter-Coupled Logic)* family [Masaki74]. TTL had the advantage, however, of offering a higher integration density and was the basis of the first integrated circuit revolution. In fact, the manufacturing of TTL components is what spearheaded the first large semiconductor companies such as Fairchild, National, and Texas Instruments. The family was so successful that it composed the largest fraction of the digital semiconductor market until the 1980s.

Ultimately, bipolar digital logic lost the battle for hegemony in the digital design world for exactly the reasons that haunted the vacuum tube approach: the large power consumption per gate puts an upper limit on the number of gates that can be reliably integrated on a single die,

[1] An intriguing overview of the evolution of digital integrated circuits can be found in [Murphy93]. (Most of the data in this overview has been extracted from this reference). It is accompanied by some of the historically ground-breaking publications in the domain of digital ICs.

package, housing, or box. Although attempts were made to develop high integration density, low-power bipolar families (such as I^2L—*Integrated Injection Logic* [Hart72]), the MOS digital integrated circuit approach eventually held sway.

The basic principle behind the MOSFET transistor (originally called IGFET) was proposed in a patent by J. Lilienfeld (Canada) as early as 1925, and, independently, by O. Heil in England in 1935. Insufficient knowledge of the materials and gate stability problems, however, delayed the practical usability of the device for a long time. Once these were solved, MOS digital integrated circuits started to take off in full in the early 1970s. Remarkably, the first MOS logic gates introduced were of the CMOS variety [Wanlass63], and this trend continued until the late 1960s. The complexity of the manufacturing process delayed the full exploitation of these devices for two more decades. Instead, the first practical MOS integrated circuits were implemented in PMOS-only logic and were used in applications such as calculators. The second age of the digital integrated circuit revolution was inaugurated with the introduction of the first microprocessors by Intel in 1972 (the 4004) [Faggin72] and 1974 (the 8080) [Shima74]. These processors were implemented in NMOS-only logic, which has the advantage of higher speed over the PMOS logic. Simultaneously, MOS technology enabled the realization of the first high-density semiconductor memories. For instance, the first 4Kbit MOS memory was introduced in 1970 [Hoff70].

These events were at the start of a truly astounding evolution towards ever higher integration densities and speed performances, a revolution that currently is still in full swing. The road to the current levels of integration has not been without hindrances, however. In the late 1970s, NMOS-only logic started to suffer from the same plague that made high-density bipolar logic unattractive or infeasible: power consumption. This realization, combined with progress in manufacturing technology, finally tilted the balance towards the CMOS technology, and this is where it remains today. Interestingly enough, power consumption concerns are rapidly becoming dominant in CMOS design as well, and this time there does not seem to be a new technology around the corner to alleviate the problem.

Although the large majority of the current integrated circuits are implemented in the MOS technology, other technologies come into play when very high performance is at stake. An example of this is the BiCMOS technology that combines bipolar and MOS devices on the same die. When even higher performance is necessary, other technologies emerge such as silicon-germanium, and even superconducting technologies. These technologies only play a very small role in the overall digital integrated circuit design scene. With the ever increasing performance of CMOS, this role is bound to be reduced further with time—hence, the focus of this textbook is on CMOS only.

1.2 Issues in Digital Integrated Circuit Design

Integration density and performance of integrated circuits have gone through an astounding revolution in the last two decades. In the 1960s, Gordon Moore, then with Fairchild Corporation and later cofounder of Intel, predicted that the number of transistors that can be integrated on a

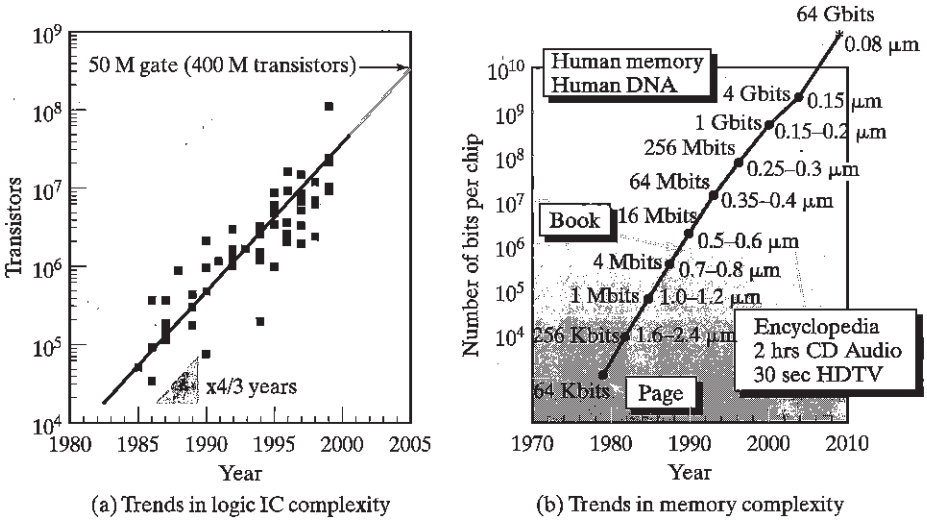


Figure 1-2 Evolution of integration complexity of logic ICs and memories as a function of time.

single die would grow exponentially with time. This prediction, later called *Moore's law*, has proven to be amazingly visionary [Moore65]. Its validity is best illustrated with the aid of a set of graphs. Figure 1-2 plots the integration density of both logic ICs and memory as a function of time. As can be observed, integration complexity doubles approximately every one to two years. As a result, memory density has increased by more than a thousandfold since 1970.

An intriguing case study is offered by the microprocessor. From its inception in the early 1970s, the microprocessor has grown in performance and complexity at a steady and predictable pace. The transistor counts for a number of landmark designs that are collected in Figure 1-3.

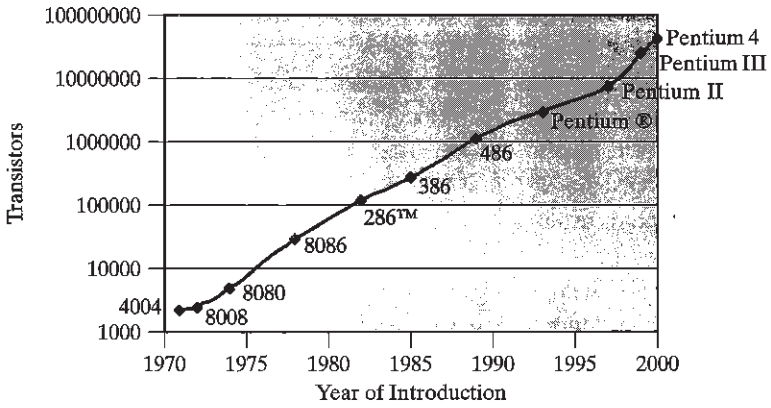


Figure 1-3 Historical evolution of microprocessor transistor count (from [Intel01]).

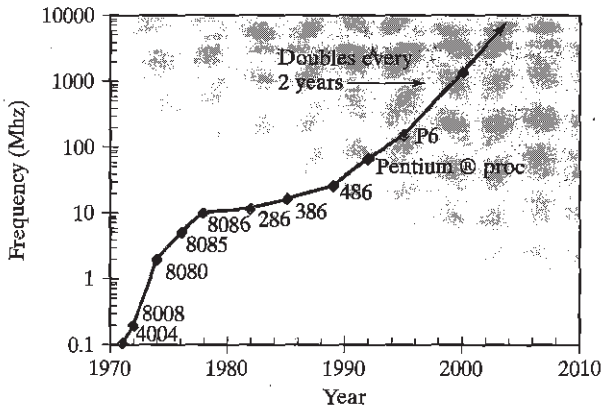


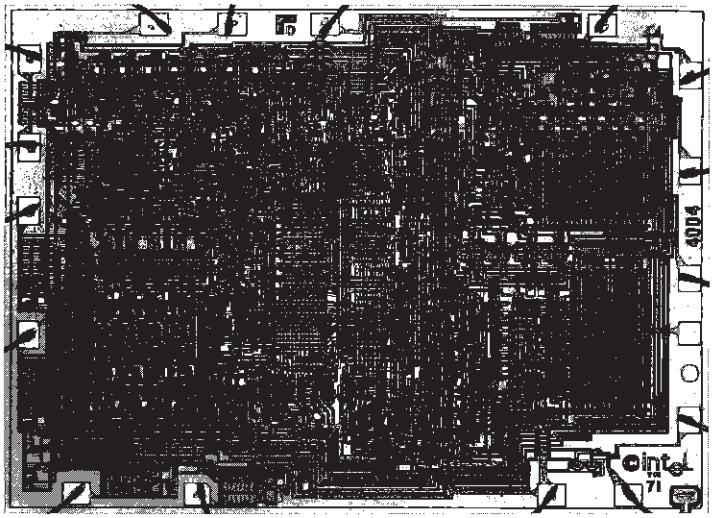
Figure 1-4 Microprocessor performance trends at the beginning of the 21st century. (Courtesy of Intel.)

The million-transistor/chip barrier was crossed in the late 1980s. Clock frequencies double every three years in the past decade and have reached into the GHz range. This is illustrated in Figure 1-4, which plots the microprocessor trends in terms of performance at the beginning of the 21st century. An important observation is that, as of now, these trends have not shown any signs of a slowdown.

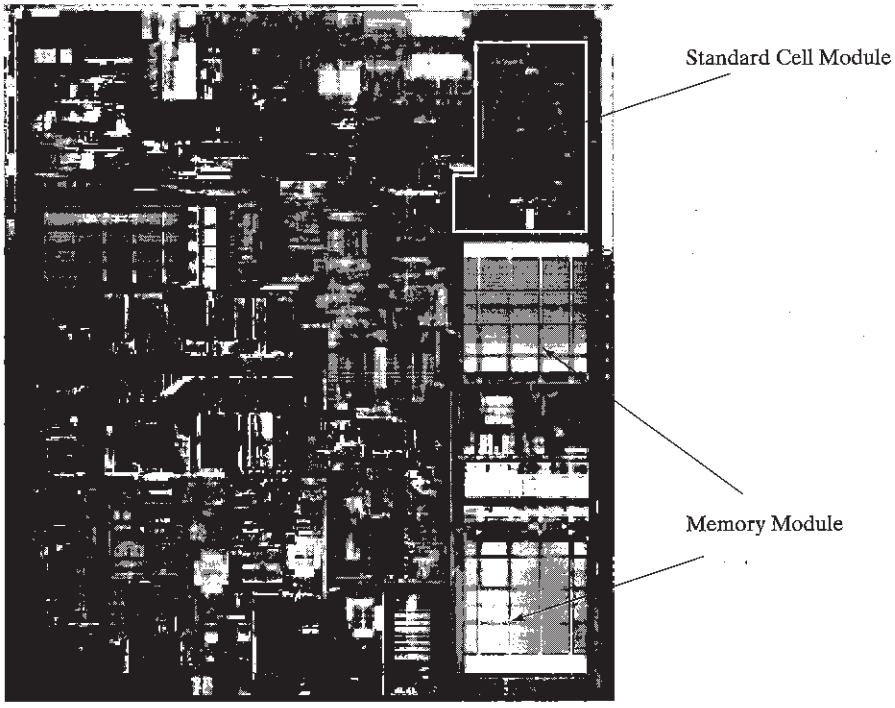
It should not surprise the reader that this revolution has had a profound impact on how digital circuits are designed. Early designs were truly handcrafted. Every transistor was laid out and optimized individually and carefully fitted into its environment. This is adequately illustrated in Figure 1-5a, which shows the design of the Intel 4004 microprocessor. Obviously, this approach is not appropriate when more than a million devices have to be created and assembled. With the rapid evolution of the design technology, time to market is one of the crucial factors in the ultimate success of a component.

As a result, designers have increasingly adhered to rigid design methodologies and strategies that are more amenable to design automation. The impact of this approach is apparent from the layout of one of the later Intel microprocessors, the Pentium® 4, shown in Figure 1-5b. Instead of the individualized approach of the earlier designs, a circuit is constructed in a hierarchical way: a processor is a collection of modules, each of which consists of a number of cells on its own. Cells are reused as much as possible to reduce the design effort and to enhance the chances for a first-time-right implementation. The fact that this hierarchical approach is at all possible is the key ingredient for the success of digital circuit design and also explains why, for instance, very large-scale analog design has never caught on.

The obvious next question is why such an approach is feasible in the digital world and not (or to a lesser degree) in analog designs. The crucial concept here, and the most important one in dealing with the complexity issue, is *abstraction*. At each design level, the internal details of a complex module can be abstracted away and replaced by a *black-box view* or *model*. This model contains virtually all the information needed to deal with the block at the next level of hierarchy.



(a) The 4004 microprocessor



(b) The Pentium ® 4 microprocessor

Figure 1-5 Comparing the design methodologies of the Intel 4004 (1971) and Pentium® 4 (2000) microprocessors (reprinted with permission from Intel).

For instance, once a designer has implemented a multiplier module, its performance can be defined very accurately and can be captured in a model. In general, the performance of this multiplier is only marginally influenced by the way it is utilized in a larger system. For all purposes, therefore, it can be considered a black box with known characteristics. As there exists no compelling need for the system designer to look inside this box, design complexity is substantially reduced. The impact of this *divide-and-conquer* approach is dramatic. Instead of having to deal with a myriad of elements, the designer has to consider only a handful of components, each of which are characterized in performance and cost by a small number of parameters.

This is analogous to a software designer using a library of software routines such as input/output drivers. Someone writing a large program does not bother to look inside those library routines. The only thing he cares about is the intended result of calling one of those modules. Imagine what writing software programs would be like if you had to fetch every bit individually from the disk and ensure its correctness instead of relying on handy "file open" and "get string" operators.

Abstraction levels typically used in digital circuit design are, in order of increasing abstraction, the device, circuit, gate, functional module (e.g., adder) and system levels (e.g., processor), as illustrated in Figure 1-6. A semiconductor device is an entity with a very complex behavior. No circuit designer will ever seriously consider the solid-state physics equations governing the behavior of the device when designing a digital gate. Instead, he will use a simplified model that adequately describes the input/output behavior of the transistor. For instance, an AND gate is adequately described by its Boolean expression ($Z = A.B$), its bounding box, the position of the input and output terminals, and the delay between the inputs and the output.

This design philosophy has been the enabler for the emergence of elaborate *computer-aided design* (CAD) frameworks for digital integrated circuits—without it the current design complexity would not have been achievable. Design tools include simulation at the various complexity levels, design verification, layout generation, and design synthesis. An overview of these tools and design methodologies is given in Chapter 8.

Furthermore, to avoid the redesign and reverification of frequently used cells, such as basic gates and arithmetic and memory modules, designers most often resort to *cell libraries.* These libraries not only contain the layouts, but also provide complete documentation and characterization of the behavior of the cells. The use of cell libraries is, for instance, apparent in the layout of the Pentium® 4 processor (Figure 1-5b). The integer and floating-point unit, just to name a few, contain large sections designed using one particular cell-based approach, called *standard cell.* Logic gates are placed in rows of cells of equal height and interconnected using routing channels. The layout of such a block can be generated automatically given that a library of cells is available.

The preceding analysis demonstrates that design automation and modular design practices have effectively addressed some of the complexity issues incurred in contemporary digital design. This leads to the following pertinent question: If design automation solves all our design problems, why should we be concerned with digital circuit design at all? Will the next-generation

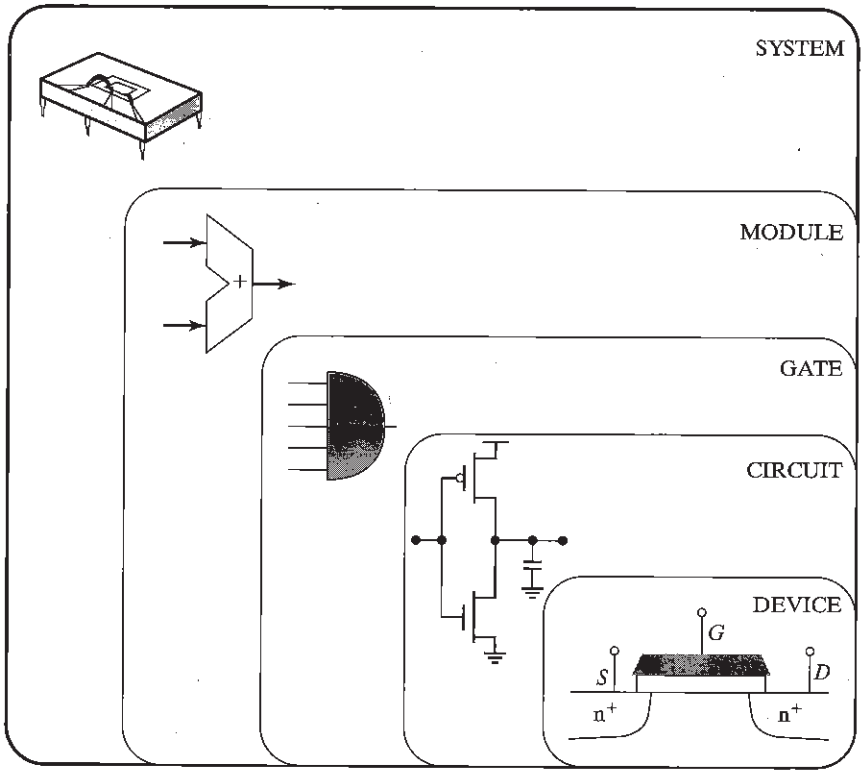


Figure 1-6 Design abstraction levels in digital circuits.

digital designer ever have to worry about transistors or parasitics, or is the smallest design entity he will ever consider be the gate and the module?

The truth is that the reality is more complex, and various reasons exist as to why an insight into digital circuits and their intricacies will still be an important asset for a long time to come:

- First, someone still has to *design and implement* the module libraries. Semiconductor technologies continue to advance from year to year. Until one has developed a foolproof approach towards "porting" a cell from one technology to another, each change in technology—which happens approximately every two years—requires a redesign of the library.
- Creating an adequate *model* of a cell or module requires an in-depth understanding of its internal operation. For instance, to identify the dominant performance parameters of a given design, one has to recognize the critical timing path first.
- The library-based approach works fine when the design constraints (speed, cost, or power) are not stringent. This is the case for a large number of *application-specific designs*, where

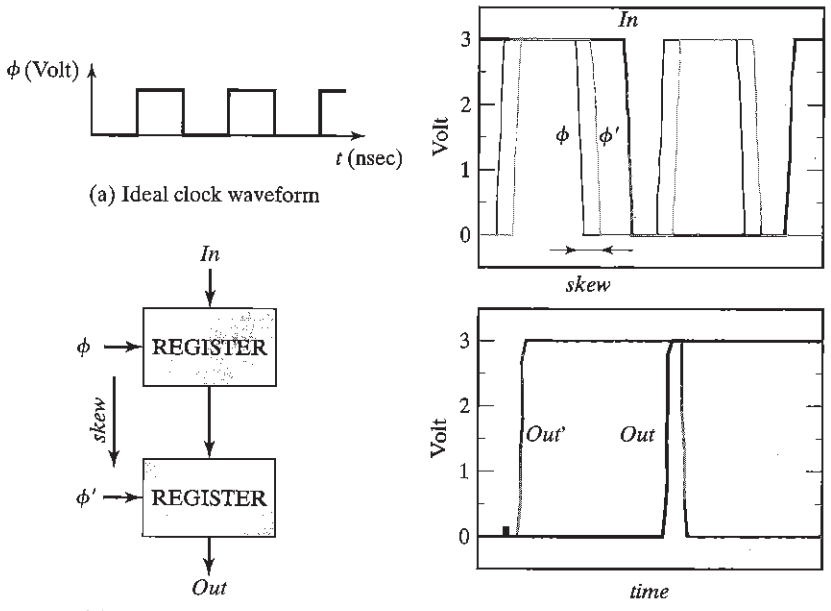
the main goal is to provide a more integrated system solution, and performance requirements are easily within the capabilities of the technology. Unfortunately, for a large number of other products such as microprocessors, success hinges on high performance, and designers therefore tend to push technology to its limits. At that point, the hierarchical approach tends to become somewhat less attractive. To resort to our previous analogy to software methodologies, a programmer tends to "customize" software routines when execution speed is crucial; compilers—or design tools—are not yet to the level of what human sweat or ingenuity can deliver.

- Even more important is the observation that the abstraction-based approach is only correct to a certain degree. The performance of, for instance, an adder can be substantially influenced by the way it is connected to its environment. The interconnection wires themselves contribute to delay because they introduce parasitic capacitances, resistances, and even inductances. The impact of the *interconnect parasitics* is bound to increase in the years to come with the scaling of the technology.
- Scaling tends to emphasize some other deficiencies of the abstraction-based model. Some design entities tend to be *global or external*. Examples of global factors are the clock signals, used for synchronization in a digital design, and the supply lines. Increasing the size of a digital design has a profound effect on these global signals. For instance, connecting more cells to a supply line can cause a voltage drop over the wire, which, in turn, can slow down all the connected cells. Issues such as clock distribution, circuit synchronization, and supply-voltage distribution are becoming more and more critical. Coping with them requires a profound understanding of the intricacies of digital circuit design.
- Another impact of technology evolution is that *new design issues* and constraints tend to emerge over time. A typical example of this is the periodical reemergence of power dissipation as a constraining factor, as was already illustrated in the historical overview. Another example is the changing ratio between device and interconnect parasitics. To cope with these unforeseen factors, one must at least be able to model and analyze their impact, requiring once again a profound insight into circuit topology and behavior.
- Finally, when things can go wrong, they often do. A fabricated circuit does not always exhibit the exact waveforms one might expect from advance simulations. Deviations can be caused by variations in the fabrication process parameters, or by the inductance of the package, or by a badly modeled clock signal. *Troubleshooting* a design requires circuit expertise.

For all these reasons, it is our profound belief that an in-depth knowledge of digital circuit design techniques and approaches is an essential asset for a digital-system designer. Even though she might not have to deal with the details of the circuit on a daily basis, the understanding will help her cope with unexpected circumstances and determine the dominant effects when analyzing a design.

Example 1.1 Clocks Defy Hierarchy

To illustrate some of the issues raised in the preceding discussion, let us examine the impact of deficiencies in one of the most important global signals in a design, the *clock*. The function of the clock signal in a digital design is to order the multitude of events happening in the circuit. This task can be compared to the function of a traffic light that determines which cars are allowed to move. It also makes sure that all operations are completed before the next one starts—a traffic light should be green long enough to allow a car or a pedestrian to cross the road. Under ideal circumstances, the clock signal is a periodic step waveform with transitions synchronized throughout the designed circuit (Figure 1-7a). In light of our analogy, changes in the traffic lights should be synchronized to maximize throughput while avoiding accidents. The importance of the *clock alignment* concept is illustrated with the example of two cascaded registers, both operating on the rising edge of the clock ϕ (Figure 1-7b). Under normal operating conditions, the input *In* gets sampled into the first register on the rising edge of ϕ and appears at the output exactly one clock period later. This is confirmed by the simulations shown in Figure 1-7c (signal *Out*).



(a) Ideal clock waveform

(b) Two cascaded registers

(c) Simulated waveforms

Figure 1-7 Impact of clock misalignment.

Due to delays associated with routing the clock wires, it may happen that the clocks become misaligned with respect to each other. As a result, the registers are interpreting time indicated by the clock signal differently. Consider the case in which the clock signal

for the second register is delayed—or skewed—by a value δ. The rising edge of the delayed clock ϕ' will postpone the sampling of the input of the second register. If the time it takes to propagate the output of the first register to the input of the second register is smaller than the clock delay, the latter will sample the wrong value. This causes the output to change prematurely, as clearly illustrated in the simulation, where the signal *Out'* goes high at the first rising edge of ϕ' instead of the second one. In terms of our traffic analogy, cars leaving the first traffic light hit the cars at the next light that have not yet left.

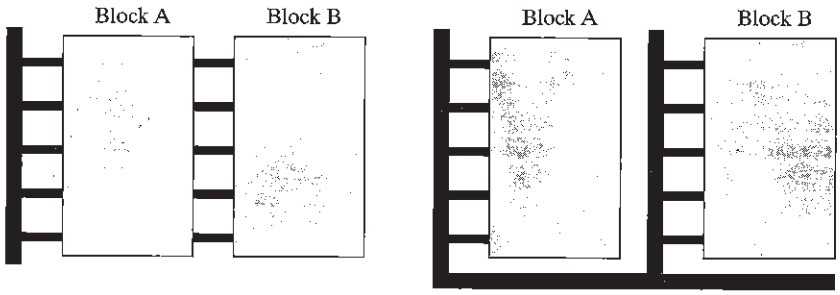
Clock misalignment, or *clock skew*, is an important example of how global signals may influence the functioning of a hierarchically designed system. Clock skew is actually one of the most critical design problems facing the designers of large high-performance systems.

Example 1.2 Power Distribution Networks Defy Hierarchy

While the clock signal is one example of a global signal that crosses the chip hierarchy boundaries, the power distribution network represents another. A digital system requires a stable DC voltage to be supplied to the individual gates. To ensure proper operation, this voltage should be stable within a few hundred millivolts. The power distribution system has to provide this stable voltage in the presence of very large current variations. The resistive nature of the on-chip wires and the inductance of the IC package pins make this a difficult proposition. For example, the average DC current to be supplied to a 100 W-1V microprocessor equals 100 A! The peak current can easily be twice as large, and current demand can readily change from almost zero to this peak value over a short time—in the range of 1 nsec or less. This leads to a current variation of 100 GA/s, which is a truly astounding number.

Consider the problem of the resistance of power distribution wires. For a current of 100 A, a wire resistance of merely 1.25 mΩ leads to a 5% drop in supply voltage (for a 2.5 V supply). Making the wires wider reduces the resistance and thus the voltage drop. While this sizing of the power network is relatively simple in a flat design approach, it is a lot more complex in a hierarchical design. For example, consider the two blocks shown in Figure 1-8a [Saleh01]. If power distribution for Block A is examined in isolation, the additional loading due to the presence of Block B is not taken into account. If power is routed through Block A to Block B, a larger IR drop will occur in Block B, since power is also being consumed by Block A before it reaches Block B.

Since the total IR drop is based on the resistance seen from the pin to the block, one could route around the block and feed power to each block separately, as shown in Figure 1-8b. Ideally, the main trunks should be large enough to handle all the current flowing through separate branches. Although routing power this way is easier to control and maintain, it also requires more area to implement. The large metal trunks of power have to



(a) Routing through the block (b) Routing around the block

Figure 1-8 Power distribution network design.

be sized to handle all the current for each block. This requirement forces designers to set aside area for power busing that takes away from the available routing area.

As more and more blocks are added, the complex interactions between the blocks determine the actual voltage drops. For instance, it is not always easy to determine which way the current will flow when multiple parallel paths are available between the power source and the consuming gate. Also, currents into the different modules do rarely peak at the same time. All these considerations make the design of the power distribution a challenging job. It requires a design methodology approach that supersedes the artificial boundaries imposed by hierarchical design.

The purpose of this book is to provide *a bridge between the abstract vision of digital design and the underlying digital circuit and its peculiarities*. While starting from a solid understanding of the operation of electronic devices and an in-depth analysis of the nucleus of digital design—the inverter—we will gradually channel this knowledge into the design of more complex entities, such as complex gates, datapaths, registers, controllers, and memories. The persistent quest for a designer when designing each of the mentioned modules is to identify the dominant design parameters, to locate the section of the design he should focus his optimizations on, and to determine the specific properties that make the module under investigation (e.g., a memory) different from any others.

We also address other compelling (global) issues in modern digital circuit design such as *power dissipation, interconnect, timing,* and *synchronization*.

1.3 Quality Metrics of a Digital Design

This section defines a set of basic properties of a digital design. These properties help to quantify the quality of a design from different perspectives: cost, functionality, robustness, performance, and energy consumption. Which one of these metrics is most important depends upon the application. For instance, pure speed is a crucial property in a computer server. On the other hand, energy consumption is a dominant metric for hand heldmobile applications such as cell

phones. The introduced properties are relevant at all levels of the design hierarchy—system, chip, module, and gate. To ensure consistency in the definitions throughout the design hierarchy stack, we propose a bottom-up approach: We start with defining the basic quality metrics of a simple inverter, and gradually expand these to the more complex functions such as gate, module, and chip.

1.3.1 Cost of an Integrated Circuit

The total cost of any product can be separated into two components: the recurring expenses or the *variable cost*, and the nonrecurring expenses or the *fixed cost*.

Fixed Cost

The fixed cost is independent of the sales volume or the number of products sold. An important component of the fixed cost of an integrated circuit is the effort in time and manpower it takes to produce the design. This design cost is strongly influenced by the complexity of the design, the aggressiveness of the specifications, and the productivity of the designer. Advanced design methodologies that automate major parts of the design process can help to boost the latter. Bringing down the design cost in the presence of an ever-increasing IC complexity is one of the major challenges always facing the semiconductor industry.

Additionally, one has to account for the *indirect costs*, the company overhead that cannot be billed directly to one product. It includes the company's research and development (R&D) costs, manufacturing equipment, marketing and sales costs, and building infrastructure, among others.

Variable Cost

The variable cost accounts for the cost that is directly attributable to a manufactured product, and is hence proportional to the product volume. Variable costs include the costs of the parts used in the product, assembly costs, and testing costs. The total cost of an integrated circuit is

$$\text{cost per IC} = \text{variable cost per IC} + \left(\frac{\text{fixed cost}}{\text{volume}}\right) \tag{1.1}$$

The impact of the fixed cost is more pronounced for small-volume products. This helps explain why it makes sense to have large design teams working for several years on a hugely successful product such as a microprocessor.

While the cost of producing a single transistor has dropped exponentially over the past few decades, the basic variable-cost equation has not changed:

$$\text{variable cost} = \frac{\text{cost of die} + \text{cost of die test} + \text{cost of packaging}}{\text{final test yield}} \tag{1.2}$$

As will be discussed further in Chapter 2, the IC manufacturing process groups a number of identical circuits onto a single *wafer* (see Figure 1-9). Upon completion of the fabrication, the wafer is chopped into *dies*, which are then individually packaged after being *tested*. We will

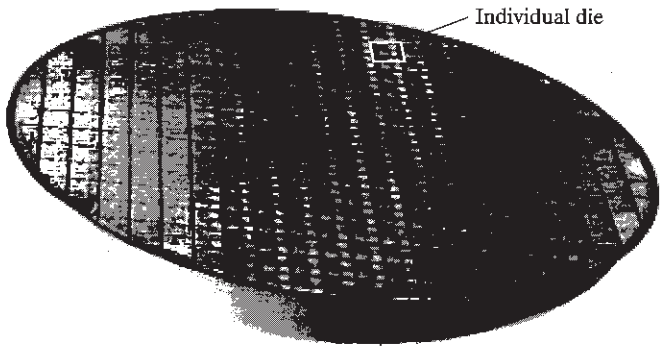


Figure 1-9 Finished wafer. Each square represents a die—in this case the AMD Duron™ microprocessor (reprinted with permission from AMD).

focus on the cost of the die in this discussion. The cost of packaging and testing is the topic of later chapters.

The die cost depends on the number of good dies on a wafer, and the percentage of those that are functional. The latter factor is called the *die yield*, given by

$$\text{cost of die} = \frac{\text{cost of wafer}}{\text{dies per wafer} \times \text{die yield}} \tag{1.3}$$

The number of dies per wafer is, in essence, the area of the wafer divided by the die area. The actual situation is somewhat more complicated as wafers are round, and chips are square. Dies around the perimeter of the wafer are therefore lost. The size of the wafer has been steadily increasing over the years, yielding more dies per fabrication run. Equation (1.3) also presents the first indication that the cost of a circuit is dependent upon the chip area—increasing the chip area simply means that fewer dies fit on a wafer.

The actual relation between cost and area is more complex, and depends upon the die yield. Both the substrate material and the manufacturing process introduce faults that can cause a chip to fail. Assuming that the defects are randomly distributed over the wafer and that the yield is inversely proportional to the complexity of the fabrication process, we obtain the equation

$$\text{die yield} = \left(1 + \frac{\text{defects per unit area} \times \text{die area}}{\alpha}\right)^{-\alpha} \tag{1.4}$$

where α is a parameter that depends upon the complexity of the manufacturing process, and it is roughly proportional to the number of masks. A good estimate for today's complex CMOS processes is $\alpha = 3$. The defects per unit area is a measure of the material and process-induced faults. A value between 0.5 and 1 defects/cm^2 is typical these days, but depends strongly upon the maturity of the process.

Example 1.3 Die Yield

Assume a wafer size of 12 inches, a die size of 2.5 cm^2, 1 defects/cm^2, and $\alpha = 3$. Determine the die yield of this CMOS process run.

The number of dies per wafer can be estimated with the following expression, which takes into account the lost dies around the perimeter of the wafer:

$$\text{dies per wafer} = \frac{\pi \times (\text{wafer diameter}/2)^2}{\text{die area}} - \frac{\pi \times \text{wafer diameter}}{\sqrt{2 \times \text{die area}}}$$

This means there are 252 (= 296 – 44) potentially operational dies for this particular example. The die yield can be computed with the aid of Eq. (1.4), and equals 16%! This means that, on average only 40 of the dies will be fully functional.

The bottom line is that the number of functional dies per wafer, and hence the cost per die, is a strong function of the die area. While the yield tends to be excellent for the smaller designs, it drops rapidly once a certain threshold is exceeded. Bearing in mind the equations derived previously and the typical parameter values, we can conclude that die costs are proportional to the fourth power of the area:

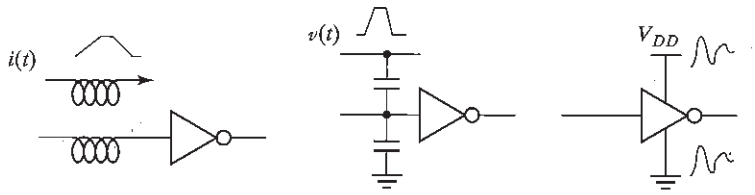
$$\text{cost of die} = f(\text{die area})^4 \tag{1.5}$$

The area is a function that is directly controllable by the designer(s), and it is the prime metric for cost. Small area is thus a desirable property for a digital gate. The smaller the gate, the higher the integration density and the smaller the die size. Smaller gates also tend to be faster and consume less energy—the total gate capacitance, which is one of the dominant performance parameters, often scales with the area.

The *number of transistors* in a gate is indicative of the expected implementation area but other parameters may have an impact. For instance, a complex interconnect pattern between the transistors can cause the wiring area to dominate. The *gate complexity*, as expressed by the number of transistors and the regularity of the interconnect structure, also has an impact on the design cost. Complex structures are more difficult to implement and tend to take more of the designer's valuable time. Simplicity and regularity is a precious property in cost-sensitive designs.

1.3.2 Functionality and Robustness

A prime requirement for a digital circuit is, obviously, that it performs the function it is designed for. The measured behavior of a manufactured circuit normally deviates from the expected response. One reason for this aberration is due to the variations in the manufacturing process. The dimensions and device parameters vary between runs or even on a single wafer or die. The electrical behavior of a circuit can be profoundly affected by those variations. The presence of disturbing noise sources on or off the chip is another source of deviations in circuit response.



(a) Inductive coupling (b) Capacitive coupling (c) Power and ground noise

Figure 1-10 Noise sources in digital circuits.

The word *noise,* in the context of digital circuits, means *unwanted variations of voltages and currents at the logic nodes.* Noise signals can enter a circuit in many ways. Some examples of digital noise sources are depicted in Figure 1-10. For instance, two wires placed side by side in an integrated circuit form a coupling capacitor and a mutual inductance. Hence, a change in voltage or current on one of the wires can influence the signals on the neighboring wire. Noise on the power and ground rails of a gate also influences the signal levels in the gate.

Most noise in a digital system is internally generated, and the noise value is proportional to the signal swing. Capacitive and inductive cross talk, and the internally generated power supply noise are examples of such. Other noise sources such as input power supply noise are external to the system, and their value is not related to the signal levels. For these sources, the noise level is directly expressed in volts or amperes. Noise sources that are a function of the signal level are better expressed as a fraction or percentage of the signal level. Noise is a major concern in the engineering of digital circuits. How to cope with all these disturbances is one of the main challenges in the design of high-performance digital circuits and is a recurring topic in this book.

The steady-state parameters (also called the *static behavior*) of a gate measure how robust the circuit is with respect to both variations in the manufacturing process and noise disturbances. The definition and derivation of these parameters requires a prior understanding of how digital signals are represented in the world of electronic circuits.

Digital circuits (DC) perform operations on *logical* (or *Boolean*) variables. A logical variable x can only assume two discrete values:

$$x \in \{0,1\}$$

As an example, the inversion (i.e., the function that an inverter performs) implements the following compositional relationship between two Boolean variables x and y:

$$y = \bar{x}: \{x = 0 \Rightarrow y = 1; x = 1 \Rightarrow y = 0\} \tag{1.6}$$

A logical variable is, however, a mathematical abstraction. In a physical implementation, such a variable is represented by an electrical quantity. This is most often a node voltage that is not discrete, but can adopt a continuous range of values. This electrical voltage is turned into a discrete variable by associating *a nominal voltage level* with each logic state: $1 \Leftrightarrow V_{OH}$, $0 \Leftrightarrow V_{OL}$, where V_{OH} and V_{OL} represent the *high* and the *low* logic levels, respectively. Applying V_{OH}

to the input of an inverter yields V_{OL} at the output and vice versa. The difference between the two is called the *logic* or *signal swing* V_{sw}.

$$V_{OH} = \overline{(V_{OL})}$$
$$V_{OL} = \overline{(V_{OH})} \tag{1.7}$$

The Voltage-Transfer Characteristic

Assume now that a logical variable *in* serves as the input to an inverting gate that produces the variable *out*. The electrical function of a gate is best expressed by its *voltage-transfer characteristic* (VTC) (sometimes called the *DC transfer characteristic*), which plots the output voltage as a function of the input voltage $V_{out} = f(V_{in})$. An example of an inverter VTC is shown in Figure 1-11. The high and low nominal voltages, V_{OH} and V_{OL}, can readily be identified—$V_{OH} = f(V_{OL})$ and $V_{OL} = f(V_{OH})$. Another point of interest of the VTC is the *gate* or *switching threshold voltage* V_M (not to be confused with the threshold voltage of a transistor), that is defined as $V_M = f(V_M)$. V_M can also be found graphically at the intersection of the VTC curve and the line given by $V_{out} = V_{in}$. The gate threshold voltage presents the midpoint of the switching characteristics, which is obtained when the output of a gate is short-circuited to the input. This point will prove to be of particular interest when studying circuits with feedback (also called *sequential circuits*).

Even if an ideal nominal value is applied at the input of a gate, the output signal often deviates from the expected nominal value. These deviations can be caused by noise or by the loading on the output of the gate (i.e., by the number of gates connected to the output signal). Figure 1-12a illustrates how a logic level is represented in reality by a range of acceptable voltages, separated by a region of uncertainty, rather than by nominal levels alone. The regions of acceptable high and low voltages are delimited by the V_{IH} and V_{IL} voltage levels, respectively. By definition, these represent the points where the gain ($= dV_{out} / dV_{in}$) of the VTC equals -1 as shown in Figure 1-12b. The region between V_{IH} and V_{IL} is called the *undefined region*

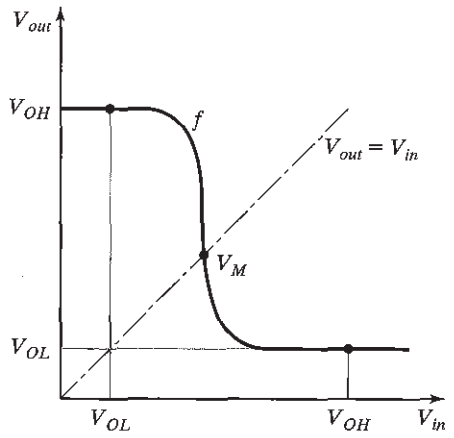
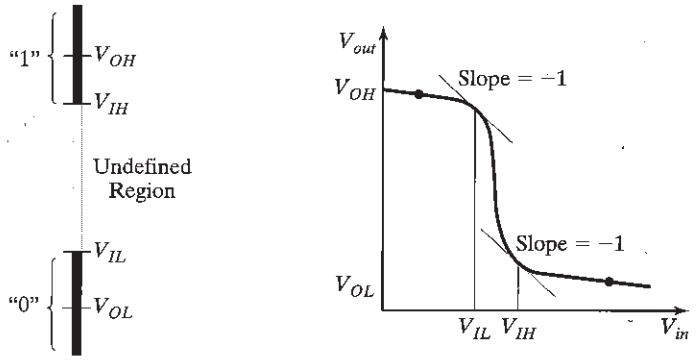


Figure 1-11 Inverter voltage-transfer characteristic.



(a) Relationship between voltage and logic levels (b) Definition of V_{IH} and V_{IL}

Figure 1-12 Mapping logic levels to the voltage domain.

(sometimes also called the *transition width,* or *TW*). Steady-state signals should avoid this region if proper circuit operation is to be ensured.

Noise Margins

For a gate to be robust and insensitive to noise disturbances, it is essential that the "0" and "1" intervals be as large as possible. A measure of the sensitivity of a gate to noise is given by the noise margins NM_L (*noise margin low*) and NM_H (*noise margin high*), which quantize the size of the legal "0" and "1," respectively, and set a fixed maximum threshold on the noise value:

$$\begin{aligned} NM_L &= V_{IL} - V_{OL} \\ NM_H &= V_{OH} - V_{IH} \end{aligned} \tag{1.8}$$

The noise margins represent the levels of noise that can be sustained when gates are cascaded as illustrated in Figure 1-13. It is obvious that the margins should be larger than 0 for a digital circuit to be functional, and by preference, they should be as large as possible.

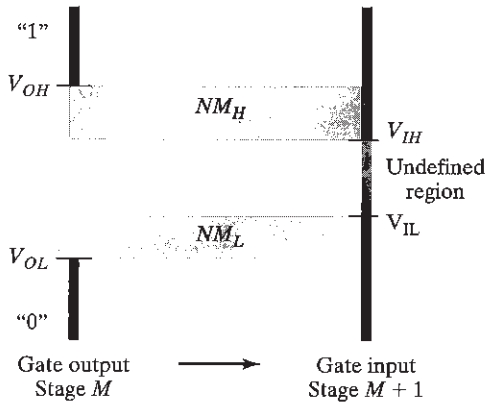
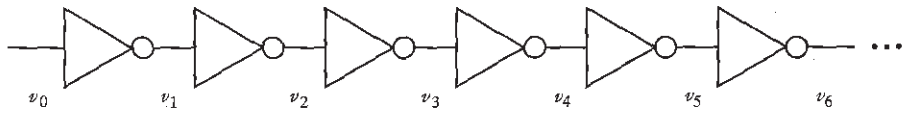
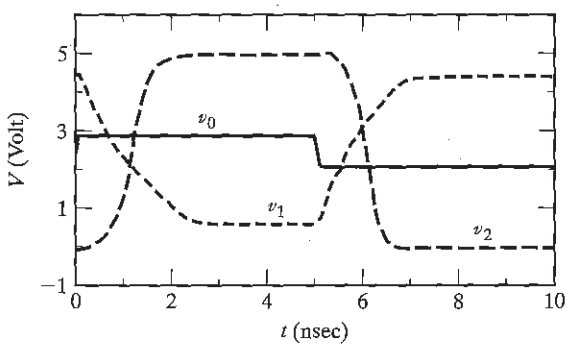


Figure 1-13 Cascaded inverter gates: definition of noise margins.



(a) A chain of inverters



(b) Simulated response of chain of MOS inverters

Figure 1-14 The regenerative property.

Regenerative Property

A large noise margin is desirable, but it is not the only requirement. Assume that a signal is disturbed by noise and differs from the nominal voltage levels. As long as the signal is within the noise margins, the gate that follows continues to function correctly, although its output voltage varies from the nominal one. This deviation is added to the noise injected at the output node and passed to the next gate. The effect of different noise sources may accumulate and eventually force a signal level into the undefined region. Fortunately, this does not happen if the gate possesses the *regenerative property*, which ensures that a disturbed signal gradually converges back to one of the nominal voltage levels after passing through a number of logical stages. This property can be understood as follows:

An input voltage v_{in} ($v_{in} \in$ "0") is applied to a chain of N inverters (Figure 1-14a). Assuming that the number of inverters in the chain is even, the output voltage v_{out} ($N \rightarrow \infty$) will equal V_{OL} if and only if the inverter possesses the regenerative property. Similarly, when an input voltage v_{in} ($v_{in} \in$ "1") is applied to the inverter chain, the output voltage will approach the nominal value V_{OH}.

Example 1.4 Regenerative Property

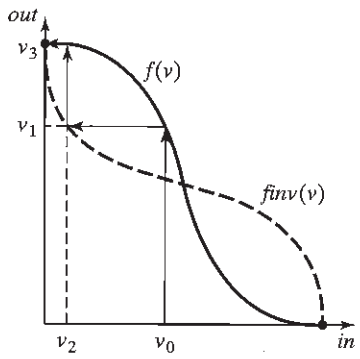
The concept of regeneration is illustrated in Figure 1-14b, which plots the simulated transient response of a chain of CMOS inverters. The input signal to the chain is a step wave-

form with a degraded amplitude, which could be caused by noise. Instead of swinging from rail to rail, v_0 only extends between 2.1 and 2.9 V. From the simulation, it can be observed that this deviation rapidly disappears while progressing through the chain; v_1, for instance, extends from 0.6 V to 4.45 V. Even further, v_2 already swings between the nominal V_{OL} and V_{OH}. The inverter used in this example clearly possesses the regenerative property.

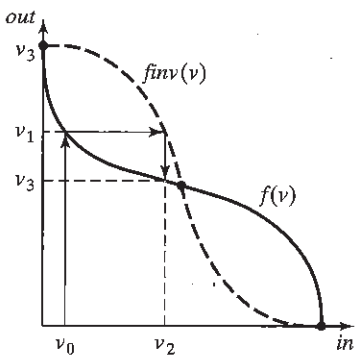
The conditions under which a gate is regenerative can be intuitively derived by analyzing a simple case study. Figure 1-15a plots the VTC of an inverter $V_{out} = f(V_{in})$ as well as its inverse function *finv*(), which reverts the function of the x- and y-axis and is defined as

$$in = f(out) \Rightarrow in = finv(out) \tag{1.9}$$

Assume that a voltage v_0, deviating from the nominal voltages, is applied to the first inverter in the chain. The output voltage of this inverter equals $v_1 = f(v_0)$ and is applied to the next inverter. Graphically, this corresponds to $v_1 = finv(v_2)$. The signal voltage gradually converges to the nominal signal after a number of inverter stages, as indicated by the arrows. In Figure 1-15b the signal does not converge to any of the nominal voltage levels, but to an intermediate voltage level. Hence, the characteristic is nonregenerative. The difference between the two cases is due to the gain characteristics of the gates. To be regenerative, the VTC should have a transient region (or undefined region) with a gain *greater than* 1 in absolute value, bordered by the two legal zones, where the gain should be *less than* 1. Such a gate has two stable operating points. This clarifies the definition of the V_{IH} and the V_{IL} levels that form the boundaries between the legal and the transient zones.



(a) Regenerative gate



(b) Nonregenerative gate

Figure 1-15 Conditions for regeneration.

Noise Immunity

While the noise margin is a meaningful means for measuring the robustness of a circuit against noise, it is not sufficient. It expresses the capability of a circuit to "overpower" a noise source. *Noise immunity*, on the other hand, expresses the ability of the system to process and transmit information correctly in the presence of noise [Dally98]. Many digital circuits with low noise margins have very good noise immunity because they *reject a noise source* rather than overpower it. These circuits have the property that only a small fraction of a potentially damaging noise source is coupled to the important circuit nodes. More precisely, the transfer function between noise source and signal node is far less than 1. Circuits that do not posses this property are *susceptible* to noise.

To study the noise immunity of a gate, we have to construct a noise budget that allocates the power budget to the various noise sources. As discussed earlier, the noise sources can be divided into the following types of sources:

- those that are *proportional* to the signal swing V_{sw}. The impact on the signal node is expressed as $g\ V_{sw}$; and
- those that are *fixed*. The impact on the signal node equals $f\ V_{Nf}$, with V_{nf} the amplitude of the noise source, and f the transfer function *from noise* to signal node.

We assume, for the sake of simplicity, that the noise margin equals half the signal swing (for both H and L). To operate correctly, the noise margin has to be larger than the sum of the coupled noise values:

$$V_{NM} = \frac{V_{sw}}{2} \geq \sum_i f_i V_{Nfi} + \sum_j g_j V_{sw} \tag{1.10}$$

Given a set of noise sources, we can derive the minimum signal swing necessary for the system to be operational:

$$V_{sw} \geq \frac{2\sum_i f_i V_{Nfi}}{1 - 2\sum_j g_j} \tag{1.11}$$

This makes it clear that the signal swing (and the noise margin) has to be large enough to overpower the impact of the fixed sources ($f\ V_{Nf}$). On the other hand, the sensitivity to internal sources depends primarily upon the noise suppressing capabilities of the gate, this is the proportionality or gain factors g_j. In the presence of large gain factors, increasing the signal swing does not do any good to suppress noise, as the noise increases proportionally. In later chapters, we will discuss some differential logic families that suppress most of the internal noise, and thus can get away with very small noise margins and signal swings.

Directivity

The directivity property requires a gate to be *unidirectional*—that is, changes in an output level should not appear at any unchanging input of the same circuit. Otherwise, an output-signal transition reflects to the gate inputs as a noise signal, affecting the signal integrity.

In real gate implementations, full directivity can never be achieved. Some feedback of changes in output levels to the inputs cannot be avoided. Capacitive coupling between inputs and outputs is a typical example of such a feedback. It is important to minimize these changes so that they do not affect the logic levels of the input signals.

Fan-In and Fan-Out

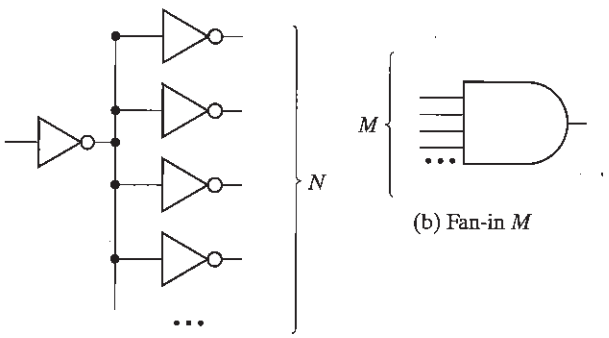
The *fan-out* denotes *the number of load gates N that are connected to the output of the driving gate* (see Figure 1-16). Increasing the fan-out of a gate can affect its logic output levels. From the world of analog amplifiers, we know that this effect is minimized by making the input resistance of the load gates as large as possible (minimizing the input currents) and by keeping the output resistance of the driving gate small (reducing the effects of load currents on the output voltage). When the fan-out is large, the added load can deteriorate the dynamic performance of the driving gate. For these reasons, many generic and library components define a *maximum fan-out* to guarantee that the static and dynamic performance of the element meet specification.

The *fan-in* of a gate is defined as the *number of inputs* to the gate (see Figure 1-16b). Gates with large fan-in tend to be more complex, which often results in inferior static and dynamic properties.

The Ideal Digital Gate

Based on these observations, we can define the *ideal* digital gate from a static perspective. The ideal inverter model is important because it gives us a metric by which we can judge the quality of actual implementations.

Its VTC is shown in Figure 1-17 and has the following properties: infinite gain in the transition region, and gate threshold located in the middle of the logic swing, with high and low



(a) Fan-out N (b) Fan-in M

Figure 1-16 Definition of fan-out and fan-in of a digital gate.

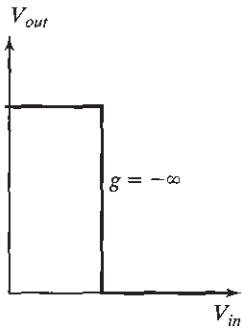


Figure 1-17 Ideal voltage-transfer characteristic.

noise margins equal to half the swing. The input and output impedances of the ideal gate are infinity and zero, respectively (i.e., the gate has unlimited fan-out). While this ideal VTC is unfortunately impossible in real designs, some implementations, such as the static CMOS inverter, come close.

Example 1.5 Voltage-Transfer Characteristic

Figure 1-18 shows an example of a voltage-transfer characteristic of an actual, but outdated, gate structure. The values of the dc parameters are derived from inspection of the graph.

$$V_{OH} = 3.5\text{ V}; \qquad V_{OL} = 0.45\text{ V}$$
$$V_{IH} = 2.35\text{ V}; \qquad V_{IL} = 0.66\text{ V}$$
$$V_M = 1.64\text{ V}$$
$$NM_H = 1.15\text{ V}; \qquad NM_L = 0.21\text{ V}$$

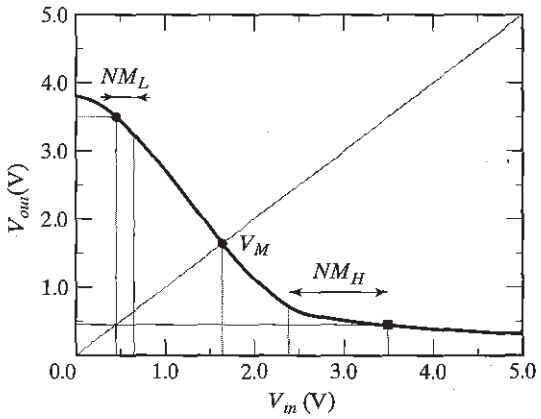


Figure 1-18 Voltage-transfer characteristic of an NMOS inverter of the 1970s.

The observed transfer characteristic, obviously, is far from ideal: it is asymmetrical, has a very low value for NM_L, and the voltage swing of 3.05 V is substantially below the maximum obtainable value of 5 V (which is the value of the supply voltage for this design).

1.3.3 Performance

From a system designer's perspective, the performance of a digital circuit expresses its computational ability. For instance, a microprocessor often is characterized by the number of instructions it can execute per second. This performance metric depends both on the architecture of the processor—for instance, the number of instructions it can execute in parallel—and the actual design of logic circuitry. While the former is crucially important, it is not the focus of this text. The reader may refer to the many excellent books on this topic [for instance, Hennessy02]. When focusing on the pure design, performance is most often expressed by the duration of the clock period (*clock cycle time*), or its rate (*clock frequency*). The minimum value of the clock period for a given technology and design is set by a number of factors such as the time it takes for the signals to propagate through the logic, the time it takes to get the data in and out of the registers, and the uncertainty of the clock arrival times. Each of these topics will be discussed in detail in this book. The performance of an individual gate lies at the core of the whole performance analysis, however.

The *propagation delay* t_p of a gate defines how quickly it responds to a change at its input(s). It expresses *the delay experienced by a signal when passing through a gate.* It is measured between the 50% transition points of the input and output waveforms, as shown in Figure 1-19 for

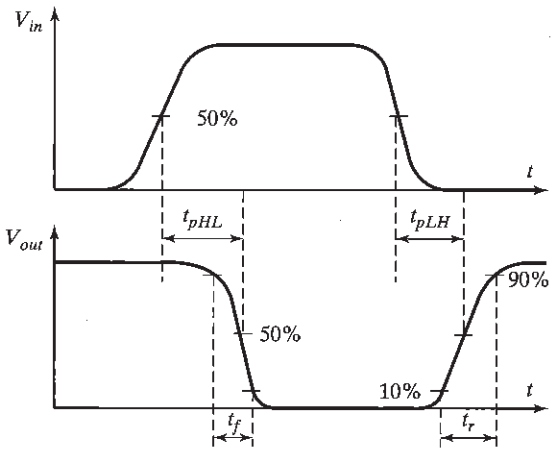


Figure 1-19 Definition of propagation delays and rise and fall times.

an inverting gate.[2] Because a gate displays different response times for rising or falling input waveforms, two definitions of the propagation delay are necessary. The t_{pLH} defines the response time of the gate for a *low-to-high* (or positive) output transition, while t_{pHL} refers to a *high-to-low* (or negative) transition. The propagation delay t_p is defined as the average of the two:

$$t_p = \frac{t_{pLH} + t_{pHL}}{2} \tag{1.12}$$

CAUTION: Observe that the propagation delay t_p, in contrast to t_{pLH} and t_{pHL}, is an artificial gate quality metric, and has no physical meaning per se. It is mostly used to compare different semiconductor technologies, circuit, or logic design styles.

The propagation delay is not only a function of the circuit technology and topology, but depends upon other factors as well. Most importantly, the delay is a function of the *slopes* of the input and output signals of the gate. To quantify these properties, we introduce the *rise and fall times*, t_r and t_f, which are metrics that apply to individual signal waveforms rather than to gates (see Figure 1-19), and they express how fast a signal transits between the different levels. The uncertainty over when a transition actually starts or ends is avoided by defining the rise and fall times between the 10% and 90% points of the waveforms, as shown in the figure. The rise/fall time of a signal is largely determined by the strength of the driving gate, and the load presented to it.

When comparing the performance of gates implemented in different technologies or circuit styles, it is important not to confuse the picture by including parameters such as load factors, fan-in, and fan-out. A uniform way of measuring the t_p of a gate so that technologies can be judged on an equal footing is desirable. The de facto standard circuit for delay measurement is the *ring oscillator*, which consists of an odd number of inverters connected in a circular chain (see Figure 1-20.) Due to the odd number of inversions, this circuit does not have a stable operating point and oscillates. The period T of the oscillation is determined by the propagation time of a signal transition through the complete chain, or $T = 2 \times t_p \times N$ with N the number of inverters in the chain. The factor 2 results from the observation that a full cycle requires both a low-to-high and a high-to-low transition. Note that this equation is only valid for $2Nt_p >> t_f + t_r$. If this condition is not met, the circuit might not oscillate—one "wave" of signals propagating through the ring will overlap with a successor and eventually dampen the oscillation. Typically, a ring oscillator needs at least five stages to be operational.

[2]The 50% definition is inspired by the assumption that the switching threshold V_M is typically located in the middle of the logic swing.

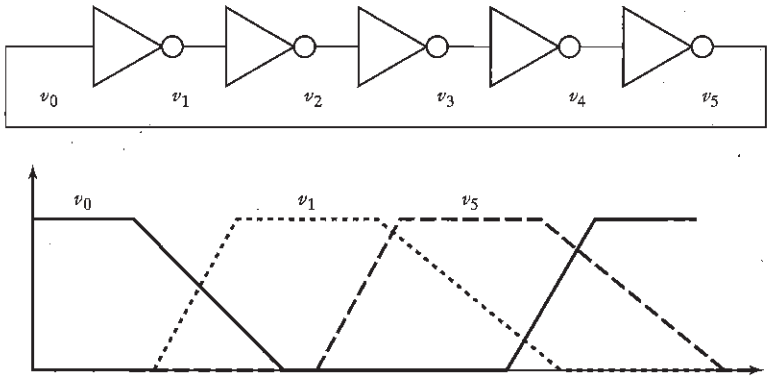


Figure 1-20 Ring oscillator circuit for propagation-delay measurement.

CAUTION: We must be extremely careful with results obtained from ring oscillator measurements. A t_p of 20 ps by no means implies that a circuit built with those gates will operate at 50 GHz. The oscillator results are primarily useful for quantifying the differences between various manufacturing technologies and gate topologies. The oscillator is an idealized circuit in which each gate has a fan-in and fan-out of exactly 1 and parasitic loads are minimal. In more realistic digital circuits, fan-ins and fan-outs are higher, and interconnect delays are non-negligible. The gate functionality is also substantially more complex than a simple invert operation. As a result, the achievable clock frequency, on average, is 50 to 100 times slower than the frequency obtained from ring oscillator measurements. This is an average observation; carefully optimized designs might approach use ideal frequency more closely.

Example 1.6 Propagation Delay of First-Order *RC* Network

Digital circuits are often modeled as first-order *RC* networks of the type shown in Figure 1-21. The propagation delay of such a network is thus of considerable interest.

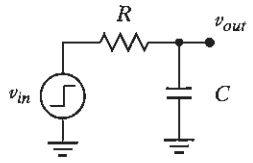


Figure 1-21 First-order *RC* network.

When applying a step input (with v_{in} going from 0 to V), the transient response of this circuit is known to be an exponential function, and is given by the following expression (where $\tau = RC$, the time constant of the network):

$$v_{out}(t) = (1 - e^{-t/\tau})\, V \tag{1.13}$$

The time to reach the 50% point is easily computed as $t = \ln(2)\tau = 0.69\tau$. Similarly, it takes $t = \ln(9)\tau = 2.2\tau$ to get to the 90% point. It is worth memorizing these numbers because they are used extensively throughout this text.

1.3.4 Power and Energy Consumption

The power consumption of a design determines how much energy is consumed per operation, and how much heat the circuit dissipates. These factors influence a great number of critical design decisions, such as the power supply capacity, the battery lifetime, supply line sizing, packaging and cooling requirements. Therefore, power dissipation is an important property of a design that affects feasibility, cost, and reliability. In the world of high-performance computing, power consumption limits, dictated by the chip package and the heat removal system, determine the number of circuits that can be integrated onto a single chip, and how fast they are allowed to switch.With the increasing popularity of mobile and distributed computation, energy limitations put a firm restriction on the number of computations that can be performed given a minimum time between battery recharges.

Depending on the design problem at hand, different dissipation measures must be considered. For instance, the peak power P_{peak} is important when studying supply line sizing. When addressing cooling or battery requirements, one is predominantly interested in the average power dissipation P_{av}. The measures are defined as

$$P_{peak} = i_{peak} V_{supply} = max[p(t)]$$
$$P_{av} = \frac{1}{T}\int_0^T p(t)dt = \frac{V_{supply}}{T}\int_0^T i_{supply}(t)dt \tag{1.14}$$

where $p(t)$ is the instantaneous power, i_{supply} is the current being drawn from the supply voltage V_{supply} over the interval $t \in [0,T]$, and i_{peak} is the maximum value of i_{supply} over that interval.

The dissipation can further be decomposed into *static* and *dynamic* components. The latter occurs only during transients, when the gate is switching. It is attributed to the charging of capacitors and temporary current paths between the supply rails; therefore it is proportional to the switching frequency: *the higher the number of switching events, the greater the dynamic power consumption.* On the other hand, the static component is present even when no switching occurs and is caused by static conductive paths between the supply rails or by leakage currents. It is always present, even when the circuit is in standby. Minimization of this consumption source is a worthwhile goal.

The propagation delay and the power consumption of a gate are related—the propagation delay is mostly determined by the speed at which a given amount of energy can be stored on the gate capacitors. The faster the energy transfer (or the higher the power consumption), the faster the gate. For a given technology and gate topology, the product of power consumption and propagation delay is generally a constant. This product is called the *power-delay product* (or PDP),

and can be considered as a quality measure for a switching device. The PDP is simply the *energy* consumed by the gate *per switching event*. The ring oscillator is again the circuit of choice for measuring the PDP of a logic family.

An ideal gate is one that is fast and consumes little energy. The *energy-delay* product (E-D) is a combined metric that brings those two elements together, and is often used as the ultimate quality metric. Thus, it should be clear that the E-D is equivalent to *power-delay*^{2}.

Example 1.7 Energy Dissipation of First-Order *RC* Network

Let us again consider the first-order *RC* network (shown in Figure 1-21). When applying a step input (with V_{in} going from 0 to V), an amount of energy is provided by the signal source to the network. The total energy delivered by the source (from the start of the transition to the end) can be readily computed:

$$E_{in} = \int_0^\infty i_{in}(t)v_{in}(t)dt = V\int_0^\infty C\frac{dv_{out}}{dt}dt = (CV)\int_0^V dv_{out} = CV^2 \tag{1.15}$$

It is interesting to observe that the energy needed to charge a capacitor from 0 to V volts with a step input is a function of the size of the voltage step and the capacitance, but is independent of the value of the resistor. We can also compute how much of the delivered energy gets stored on the capacitor at the end of the transition.

$$E_C = \int_0^\infty i_C(t)v_{out}(t)dt = \int_0^\infty C\frac{dv_{out}}{dt}v_{out}dt = C\int_0^V v_{out}dv_{out} = \frac{CV^2}{2} \tag{1.16}$$

This is exactly half of the energy delivered by the source. For those who wonder what happened to the other half—a simple analysis shows that an equivalent amount gets dissipated as heat in the resistor during the transaction. It is left to the reader to demonstrate that during the discharge phase (for a step from V to 0), the energy originally stored on the capacitor gets dissipated in the resistor as well, and then turned into heat.

1.4 Summary

In this introductory chapter, we learned about the history and the trends in digital circuit design. We also introduced the important quality metrics used to evaluate the quality of a design: cost, functionality, robustness, performance, and energy/power dissipation.

1.5 To Probe Further

The design of digital integrated circuits has been the topic of many textbooks and monographs. To help the reader find more information on some selected topics, an extensive list of references

follows. The state-of-the-art developments in the area of digital design are generally reported in technical journals or conference proceedings, the most important of which are listed.

Journals and Proceedings

IEEE Journal of Solid-State Circuits
IEICE Transactions on Electronics (Japan)
Proceedings of The International Solid-State and Circuits Conference (ISSCC)
Proceedings of the VLSI Circuits Symposium
Proceedings of the Custom Integrated Circuits Conference (CICC)
European Solid-State Circuits Conference (ESSCIRC)

Bibliography

MOS

M. Annaratone, *Digital CMOS Circuit Design*, Kluwer, 1986.
T. Dillinger, *VLSI Engineering*, Prentice Hall, 1988.
M. Elmasry, ed., *Digital MOS Integrated Circuits*, IEEE Press, 1981.
M. Elmasry, ed., *Digital MOS Integrated Circuits II*, IEEE Press, 1992.
L. Glasser and D. Dopperpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley, 1985.
A. Kang and Leblebici, *CMOS Digital Integrated Circuits*, 2nd Ed., McGraw-Hill, 1999.
C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.
K. Martin, *Digital Integrated Circuit Design*, Oxford University Press, 2000.
D. Pucknell and K. Eshraghian, *Basic VLSI Design*, Prentice Hall, 1988.
M. Shoji, *CMOS Digital Circuit Technology*, Prentice Hall, 1988.
J. Uyemura, *Circuit Design for CMOS VLSI*, Kluwer, 1992.
H. Veendrick, Deep-Submicron *CMOS IC's: From Basics to ASICS*, Second Edition, Kluwer Academic Publishers, 2000.
N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, 1985, 1993.

High-Performance Design

K. Bernstein et al, *High Speed CMOS Design Styles*, Kluwer Academic, 1998.
A. Chandrakasan, F. Fox, and W. Bowhill, ed., *Design of High-Performance Microprocessor Circuits*, IEEE Press, 2000.
M. Shoji, *High-Speed Digital Circuits*, Addison-Wesley, 1996.

Low-Power Design

A. Chandrakasan and R. Brodersen, ed., *Low-Power Digital CMOS Design*, IEEE Press, 1998.
M. Pedram and J. Rabaey, ed., *Power-Aware Design Methodologies*, Kluwer Academic, 2002.
J. Rabaey and M. Pedram, ed., *Low-Power Design Methodologies*, Kluwer Academic, 1996.
G. Yeap, *Practical Low-Power CMOS Design*, Kluwer Academic, 1998.

Memory Design

K. Itoh, *VLSI Memory Chip Design*, Springer, 2001.
B. Keeth and R. Baker, *DRAM Circuit Design*, IEEE Press, 1999.
B. Prince, *Semiconductor Memories*, Wiley, 1991.
B. Prince, *High Performance Memories*, Wiley, 1996.
D. Hodges, *Semiconductor Memories*, IEEE Press, 1972.

Interconnections and Packaging

H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, 1990.

W. Dally and J. Poulton, *Digital Systems Engineering*, Cambridge University Press, 1998.

E. Friedman, ed., *Clock Distribution Networks in VLSI Circuits and Systems*, IEEE Press, 1995.

J. Lau et al, ed., *Electronic Packaging: Design, Materials, Process, and Reliability*, McGraw-Hill, 1998.

Design Tools and Methodologies

V. Agrawal and S. Seth, *Test Generation for VLSI Chips*, IEEE Press, 1988.

D. Clein, *CMOS IC Layout*, Newnes, 2000.

G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.

S. Rubin, *Computer Aids for VLSI Design*, Addison-Wesley, 1987.

J. Uyemura, *Physical Design of CMOS Integrated Circuits Using L-Edit*, PWS, 1995.

A. Vladimirescu, *The Spice Book*, John Wiley and Sons, 1993.

W. Wolf, *Modern VLSI Design*, Prentice Hall, 1998.

Bipolar and BiCMOS

A. Alvarez, *BiCMOS Technology and Its Applications*, Kluwer, 1989.

M. Elmasry, ed., *BiCMOS Integrated Circuit Design*, IEEE Press, 1994.

S. Embabi, A. Bellaouar, and M. Elmasry, *Digital BiCMOS Integrated Circuit Design*, Kluwer, 1993.

General

J. Buchanan, *CMOS/TTL Digital Systems Design*, McGraw-Hill, 1990.

H. Haznedar, *Digital Micro-Electronics*, Benjamin/Cummings, 1991.

D. Hodges and H. Jackson, *Analysis and Design of Digital Integrated Circuits*, 2nd ed., McGraw-Hill, 1988.

M. Smith, *Application-Specific Integrated Circuits*, Addison-Wesley, 1997.

R. K. Watts, *Submicron Integrated Circuits*, Wiley, 1989.

References

[Bardeen48] J. Bardeen and W. Brattain, "The Transistor, a Semiconductor Triode," *Phys. Rev.*, vol. 74, p. 230, July 15, 1948.

[Beeson62] R. Beeson and H. Ruegg, "New Forms of All Transistor Logic," *ISSCC Digest of Technical Papers*, pp. 10–11, Feb. 1962.

[Dally98] B. Dally, *Digital Systems Engineering*, Cambridge University Press, 1998.

[Faggin72] F. Faggin, M.E. Hoff, Jr, H. Feeney, S. Mazor, M. Shima, "The MCS-4 - An LSI MicroComputer System," 1972 IEEE Region Six Conference Record, San Diego, CA, pp. 1–6, April 1972.

[Harris56] J. Harris, "Direct-Coupled Transistor Logic Circuitry in Digital Computers," *ISSCC Digest of Technical Papers*, p. 9, Feb. 1956.

[Hart72] C. Hart and M. Slob, "Integrated Injection Logic—A New Approach to LSI," *ISSCC Digest of Technical Papers*, pp. 92–93, Feb. 1972.

[Hoff70] E. Hoff, "Silicon-Gate Dynamic MOS Crams 1,024 Bits on a Chip," *Electronics*, pp. 68–73, August 3, 1970.

[Intel01] "Moore's Law", *http://www.intel.com/research/silicon/mooreslaw.htm*

[Masaki74] A. Masaki, Y. Harada and T. Chiba, "200-Gate ECL Master-Slice LSI," *ISSCC Digest of Technical Papers*, pp. 62–63, Feb. 1974.

[Moore65] G. Moore, "Cramming more Components into Integrated Circuits," Electronics, Vol. 38, Nr 8, April 1965.

[Murphy93] B. Murphy, "Perspectives on Logic and Microprocessors," *Commemorative Supplement to the Digest of Technical Papers, ISSCC*, pp. 49–51, San Francisco, 1993.

[Norman60] R. Norman, J. Last and I. Haas, "Solid-State Micrologic Elements," *ISSCC Digest of Technical Papers,* pp. 82–83, Feb. 1960.

[Hennessy02] J. Hennessy, D. Patterson, and David Goldberg, *Computer Architecture A Quantitative Approach*, Third Edition, Morgan Kaufmann Publishers, 2002.

[Saleh01] R. Saleh, M. Benoit, and P. McCrorie, "Power Distribution Planning", Simplex Solutions, *http://www.simplex.com/wt/sec.php?page_name=wp_powerplan*

[Schockley49] W. Schockley, "The Theory of pn Junctions in Semiconductors and pn-Junction Transistors," *BSTJ,* vol. 28, p. 435, 1949.

[Shima74] M. Shima, F. Faggin and S. Mazor, "An N-Channel, 8-bit Single-Chip Microprocessor," *ISSCC Digest of Technical Papers,* pp. 56–57, Feb. 1974.

[Swade93] D. Swade, "Redeeming Charles Babbage's Mechanical Computer," *Scientific American,* pp. 86–91, February 1993.

[Wanlass63] F. Wanlass, and C. Sah, "Nanowatt logic Using Field-Effect Metal-Oxide Semiconductor Triodes," *ISSCC Digest of Technical Papers,* pp. 32–32, Feb. 1963.

Exercises

Please refer to **http://bwrc.eecs.berkeley.edu/IcBook** for up-to-date problem sets and exercises. By making the exercises electronically available rather than in print, we can provide a dynamic environment that tracks the rapid evolution of today's digital integrated circuit design technology.

CHAPTER

4

The Wire

Determining and quantifying interconnect parameters

Introducing circuit models for interconnect wires

Detailed wire models for SPICE

Technology scaling and its impact on interconnect

4.1 Introduction

Throughout most of the history of integrated circuits, on-chip interconnect wires were almost like second class citizens, only considered in special cases or when performing high-precision analysis. With the introduction of deep submicron semiconductor technologies, this picture has undergone rapid changes. The parasitic effects introduced by the wires display a scaling behavior that differs from the active devices such as transistors, and they tend to gain in importance as device dimensions are reduced and circuit speed is increased. In fact, they start to dominate some of the relevant metrics of digital integrated circuits such as speed, energy consumption, and reliability. This situation is aggravated by the fact that improvements in technology make the production of ever larger die sizes economically feasible, which results in an increase in the average length of an interconnect wire and in the associated parasitic effects. A careful and in-depth analysis of the role and behavior of the interconnect wire in a semiconductor technology is, therefore, not only desirable, but essential.

4.2 A First Glance

The designer of an electronic circuit has multiple choices in realizing the interconnections between the various devices that make up the circuit. State-of-the-art processes offer multiple layers of aluminum or copper, and at least one layer of polysilicon. Even the heavily doped n^+ or p^+ diffusion layers typically used for the realization of source and drain regions can be employed for wiring purposes. These wires appear in the schematic diagrams of electronic circuits as simple lines with no apparent impact on the circuit performance. From our discussion of the integrated circuit manufacturing process, it should be clear that this picture is too simplistic, and that the wiring of today's integrated circuits forms a complex geometry that introduces capacitive, resistive, and inductive parasitics. All three have multiple effects on the circuit's behavior:

1. They all cause an increase in propagation delay, or, equivalently, a drop in performance.
2. They all have an impact on the energy dissipation and the power distribution.
3. They all cause the introduction of extra noise sources, which affect the reliability of the circuit.

A designer can decide to play it safe and include all these parasitic effects in her analysis and design optimization process. This conservative approach is not very constructive, however, and most often it is not even feasible. First of all, a "complete" model is dauntingly complex and is only applicable to very small topologies. Hence, it is totally useless for today's integrated circuits, with their millions of circuit nodes. Furthermore, this approach has the disadvantage of "not seeing the forest for the trees," so to speak. The circuit behavior at a given circuit node is only determined by a few dominant parameters. Bringing all possible effects to bear may obscure the picture and turn the optimization and design process into a "trial-and-error" operation, rather than an enlightened and focused search.

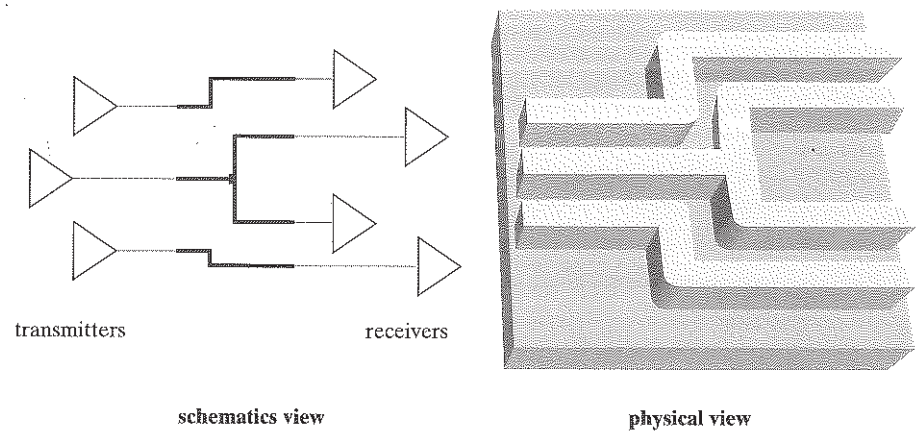


Figure 4-1 Schematic and physical views of wiring of bus network. The latter shows only a limited area (as indicated by the emphasis in the schematics).

Thus, it is important for the designer to have a clear insight into the parasitic wiring effects, their relative importance, and their models. This is best illustrated with a simple example, as shown in Figure 4-1. Each wire in a bus network connects a transmitter (or transmitters) to a set of receivers and is implemented as a chain of wire segments of various lengths and geometries. Assume that all segments are implemented on a single interconnect layer and isolated from the silicon substrate and from each other by a layer of dielectric material. (Be aware that the reality may be far more complex.)

A full-fledged circuit model, that takes into account the parasitic capacitance, resistance, and the inductance of the interconnections is shown in Figure 4-2a. Observe that these extra circuit elements are not located in a single physical point, but are distributed over the length of the wire. This is necessary when the length of the wire becomes significantly greater than its width. In addition, interwire parasitics are present, creating coupling effects between the different bus signals that were not present in the original schematics.

Analyzing the behavior of this schematic, which only models a small part of the circuit, is slow and cumbersome. Fortunately, substantial simplifications often can be made, including the following:

- Inductive effects can be ignored if the resistance of the wire is substantial enough—this is the case for long aluminum wires with a small cross section, for example, or if the rise and fall times of the applied signals are slow.
- When the wires are short, the cross section of the wire is large, or the interconnect material used has a low resistivity, a capacitance-only model can be used (see Figure 4-2b).
- Finally, when the separation between neighboring wires is large, or when the wires only run together for a short distance, interwire capacitance can be ignored, and all the parasitic capacitance can be modeled as capacitance to ground.

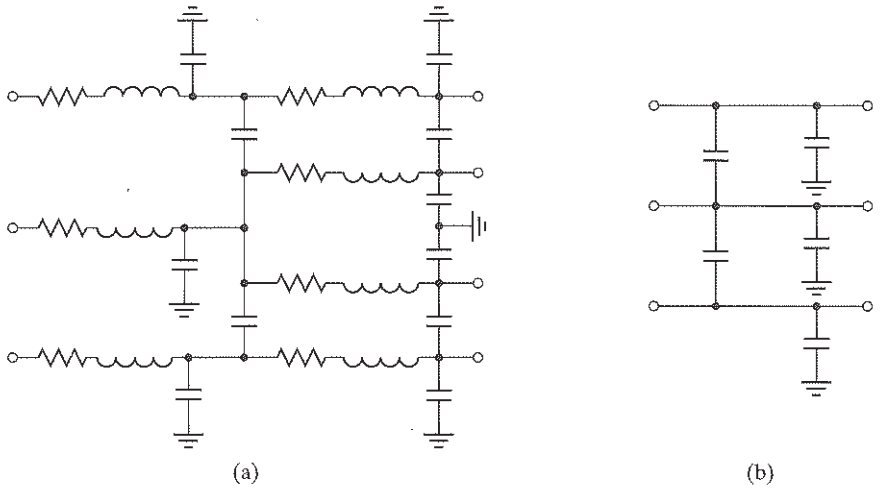


Figure 4-2 Wire models for the circuit of Figure 4-1. Model (a) considers most of the wire parasitics (with the exception of interwire resistance and mutual inductance), while model (b) only considers capacitance.

Obviously, the latter cases are the easiest to model, analyze, and optimize. The experienced designer knows to differentiate between dominant and secondary effects. The goal of this chapter is to present the basic techniques of estimating the values of the various interconnect parameters, simple models to evaluate their impact, and a set of rules of thumb for deciding when and where a particular model or effect should be considered.

4.3 Interconnect Parameters—Capacitance, Resistance, and Inductance

4.3.1 Capacitance

An accurate modeling of the wire capacitance(s) in a state-of-the-art integrated circuit is a nontrivial task, and even today it is the subject of advanced research. The task is further complicated by the fact that the interconnect structure of contemporary integrated circuits is three-dimensional, as was clearly demonstrated in the integrated-circuit cross section of Figure 2-8. The capacitance of such a wire is a function of its shape, its environment, its distance to the substrate, and the distance to surrounding wires. Rather than getting lost in complex equations and models, a designer typically will use an advanced extraction tool to get precise values of the interconnect capacitances of a completed layout. Most semiconductor manufacturers also provide empirical data for the various capacitance contributions, as measured from a number of test dies. Yet, some simple, first-order models come in handy to provide a basic understanding of the nature of interconnect capacitance and its parameters, and of how wire capacitance will evolve with future technologies.

Consider first a simple rectangular wire placed above the semiconductor substrate, as shown in Figure 4-3. If the width of the wire is substantially larger than the thickness of the

CHAPTER

6

Designing Combinational Logic Gates in CMOS

In-depth discussion of logic families in CMOS—static and dynamic, pass-transistor, nonratioed and ratioed logic

Optimizing a logic gate for area, speed, energy, or robustness

Low-power and high-performance circuit-design techniques

6.1 Introduction

The design considerations for a simple inverter circuit were presented in the previous chapter. We now extend this discussion to address the synthesis of arbitrary digital gates, such as NOR, NAND, and XOR. The focus is on *combinational logic* or *nonregenerative* circuits—that is, circuits having the property that at any point in time, the output of the circuit is related to its current input signals by some Boolean expression (assuming that the transients through the logic gates have settled). No intentional connection from outputs back to inputs is present.

This is in contrast to another class of circuits, known as *sequential* or *regenerative*, for which the output is not only a function of the current input data, but also of previous values of the input signals (see Figure 6-1). This can be accomplished by connecting one or more outputs intentionally back to some inputs. Consequently, the circuit "remembers" past events and has a sense of *history*. A sequential circuit includes a combinational logic portion and a module that holds the state. Example circuits are registers, counters, oscillators, and memory. Sequential circuits are the topic of the next chapter.

There are numerous circuit styles to implement a given logic function. As with the inverter, the common design metrics by which a gate is evaluated are area, speed, energy, and power. Depending on the application, the emphasis will be on different metrics. For example, the switching speed of digital circuits is the primary metric in a high-performance processor, while in a battery operated circuit, it is energy dissipation. Recently, power dissipation also has become an important concern and considerable emphasis is placed on understanding the sources of power and approaches to dealing with power. In addition to these metrics, robustness to noise and reliability are also very important considerations. We will see that certain logic styles can significantly improve performance, but they usually are more sensitive to noise.

6.2 Static CMOS Design

The most widely used logic style is static complementary CMOS. The static CMOS style is really an extension of the static CMOS inverter to multiple inputs. To review, the primary advantage of the CMOS structure is robustness (i.e., low sensitivity to noise), good performance, and low power consumption with no static power dissipation. Most of those properties are carried over to large fan-in logic gates implemented using a similar circuit topology.

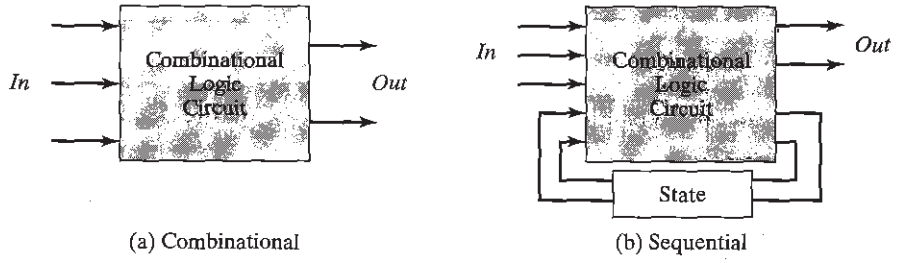


Figure 6-1 High-level classification of logic circuits.

The complementary CMOS circuit style falls under a broad class of logic circuits called *static* circuits in which **at every point in time, each gate output is connected to either V_{DD} or V_{SS} via a low-resistance path**. Also, the outputs of the gates assume at all times the value of the Boolean function implemented by the circuit (ignoring, the transient effects during switching periods). This is in contrast to the *dynamic* circuit class, which relies on temporary storage of signal values on the capacitance of high-impedance circuit nodes. The latter approach has the advantage that the resulting gate is simpler and faster. Its design and operation are, however, more involved and prone to failure because of increased sensitivity to noise.

In this section, we sequentially address the design of various static circuit flavors, including complementary CMOS, ratioed logic (pseudo-NMOS and DCVSL), and pass-transistor logic. We also deal with issues of scaling to lower power supply voltages and threshold voltages.

6.2.1 Complementary CMOS

Concept

A static CMOS gate is a combination of two networks—the *pull-up network* (PUN) and the *pull-down* network (PDN), as shown in Figure 6-2. The figure shows a generic N-input logic gate where all inputs are distributed to both the pull-up and pull-down networks. The function of the PUN is to provide a connection between the output and V_{DD} anytime the output of the logic gate is meant to be 1 (based on the inputs). Similarly, the function of the PDN is to connect the output to V_{SS} when the output of the logic gate is meant to be 0. The PUN and PDN networks are constructed in a mutually exclusive fashion such that *one and only one* of the networks is conducting in steady state. In this way, once the transients have settled, a path always exists between V_{DD} and the output F for a high output ("one"), or between V_{SS} and F for a low output ("zero"). This is equivalent to stating that the output node is always a *low-impedance* node in steady state.

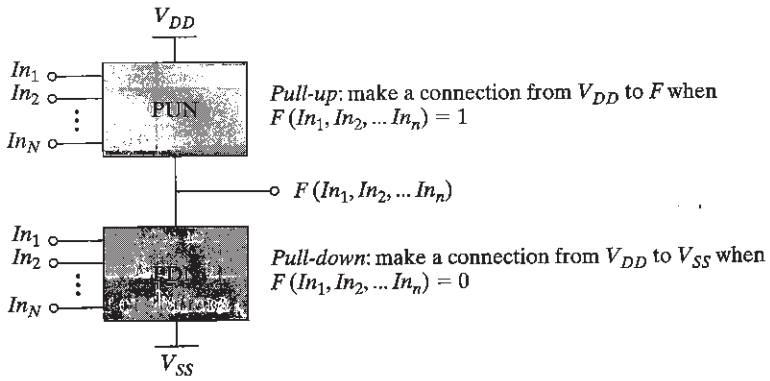
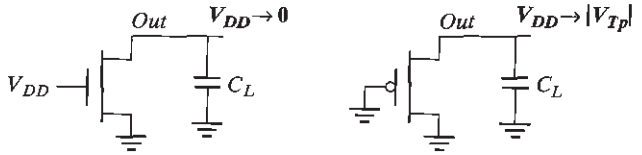


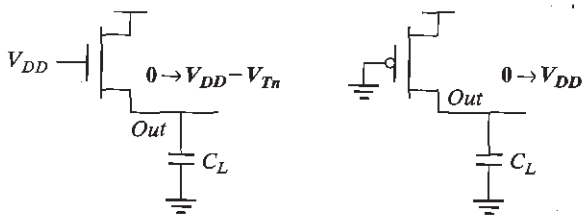
Figure 6-2 Complementary logic gate as a combination of a PUN (pull-up network) and a PDN (pull-down network).

In constructing the PDN and PUN networks, the designer should keep the following observations in mind:

- A transistor can be thought of as a switch controlled by its gate signal. An NMOS switch is *on* when the controlling signal is high and is *off* when the controlling signal is low. A PMOS transistor acts as an inverse switch that is *on* when the controlling signal is low and *off* when the controlling signal is high.
- The PDN is constructed using NMOS devices, while PMOS transistors are used in the PUN. The primary reason for this choice is that NMOS transistors produce "strong zeros," and PMOS devices generate "strong ones." To illustrate this, consider the examples shown in Figure 6-3. In Figure 6-3a, the output capacitance is initially charged to V_{DD}. Two possible discharge scenarios are shown. An NMOS device pulls the output all the way down to GND, while a PMOS lowers the output no further than $|V_{Tp}|$—the PMOS turns *off* at that point and stops contributing discharge current. NMOS transistors are thus the preferred devices in the PDN. Similarly, two alternative approaches to charging up a capacitor are shown in Figure 6-3b, with the output initially at GND. A PMOS switch succeeds in charging the output all the way to V_{DD}, while the NMOS device fails to raise the output above $V_{DD} - V_{Tn}$. This explains why PMOS transistors are preferentially used in a PUN.
- A set of rules can be derived to construct logic functions (see Figure 6-4). NMOS devices connected in series correspond to an AND function. With all the inputs high, the series combination conducts and the value at one end of the chain is transferred to the other end. Similarly, NMOS transistors connected in parallel represent an OR function. A conducting path exists between the output and input terminal if at least one of the inputs is high. Using similar arguments, construction rules for PMOS networks can be formulated. A series con-



(a) Pulling down a node by using NMOS and PMOS switches



(b) Pulling down a node by using NMOS and PMOS switches

Figure 6-3 Simple examples illustrate why an NMOS should be used as a pull-down, and a PMOS should be used as a pull-up device.

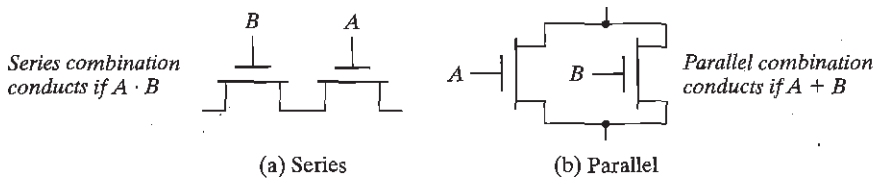


Figure 6-4 NMOS logic rules—series devices implement an AND, and parallel devices implement an OR.

nection of PMOS conducts if both inputs are low, representing a NOR function ($\overline{A} \cdot \overline{B} = \overline{A + B}$), while PMOS transistors in parallel implement a NAND ($\overline{A} + \overline{B} = \overline{A \cdot B}$).

- Using De Morgan's theorems ($\overline{A + B} = \overline{A} \cdot \overline{B}$ and $\overline{A \cdot B} = \overline{A} + \overline{B}$), it can be shown that the pull-up and pull-down networks of a complementary CMOS structure are *dual* networks. This means that a parallel connection of transistors in the pull-up network corresponds to a series connection of the corresponding devices in the pull-down network, and vice versa. Therefore, to construct a CMOS gate, one of the networks (e.g., PDN) is implemented using combinations of series and parallel devices. The other network (i.e., PUN) is obtained using the duality principle by walking the hierarchy, replacing series subnets with parallel subnets, and parallel subnets with series subnets. The complete CMOS gate is constructed by combining the PDN with the PUN.
- The complementary gate is naturally *inverting*, implementing only functions such as NAND, NOR, and XNOR. The realization of a noninverting Boolean function (such as AND OR, or XOR) in a single stage is not possible, and requires the addition of an extra inverter stage.
- The number of transistors required to implement an N-input logic gate is $2N$.

Example 6.1 Two-Input NAND Gate

Figure 6-5 shows a two-input NAND gate ($F = \overline{A \cdot B}$). The PDN network consists of two NMOS devices in series that conduct when both A and B are high. The PUN is the dual

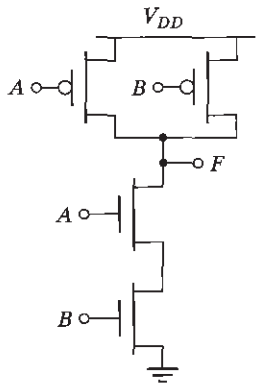


Figure 6-5 Two-input NAND gate in complementary static CMOS style.

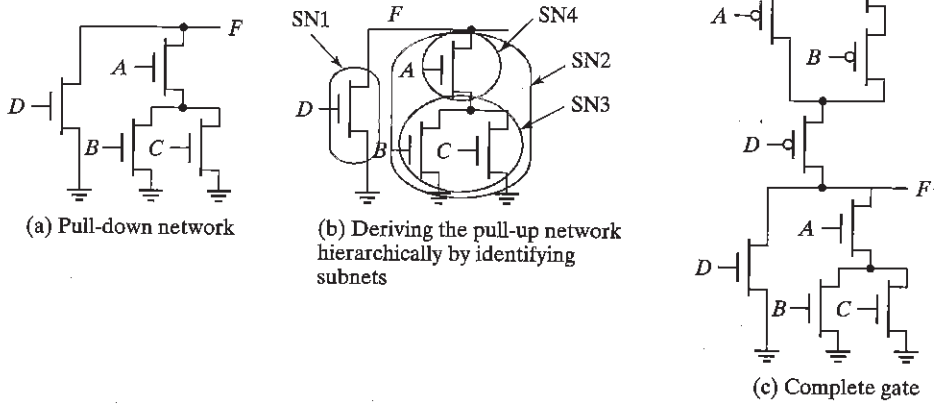
network, and it consists of two parallel PMOS transistors. This means that F is 1 if $A = 0$ or $B = 0$, which is equivalent to $F = \overline{A \cdot B}$. The truth table for the simple two input NAND gate is given in Table 6-1. It can be verified that the output F is always connected to either V_{DD} or GND, but never to both at the same time.

Table 6-1 Truth Table for two-Input NAND.

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Example 6.2 Synthesis of Complex CMOS Gate

Using complementary CMOS logic, consider the synthesis of a complex CMOS gate whose function is $F = \overline{D + A \cdot (B + C)}$. The first step in the synthesis of the logic gate is to derive the pull-down network as shown in Figure 6-6a by using the fact that NMOS devices in series implements the AND function and parallel device implements the OR function. The next step is to use duality to derive the PUN in a hierarchical fashion. The PDN network is broken into smaller networks (i.e., subset of the PDN) called subnets that simplify the derivation of the PUN. In Figure 6-6b, the subnets (SN) for the pull-down net-



(a) Pull-down network

(b) Deriving the pull-up network hierarchically by identifying subnets

(c) Complete gate

Figure 6-6 Complex complementary CMOS gate.

work are identified. At the top level, SN1 and SN2 are in parallel, so that in the dual network they will be in series. Since SN1 consists of a single transistor, it maps directly to the pull-up network. On the other hand, we need to sequentially apply the duality rules to SN2. Inside SN2, we have SN3 and SN4 in series, so in the PUN they will appear in parallel. Finally, inside SN3, the devices are in parallel, so they appear in series in the PUN. The complete gate is shown in Figure 6-6c. The reader can verify that for every possible input combination, there always exists a path to either V_{DD} or GND.

Static Properties of Complementary CMOS Gates

Complementary CMOS gates inherit all the nice properties of the basic CMOS inverter. They exhibit rail-to-rail swing with $V_{OH} = V_{DD}$ and V_{OL} = GND. The circuits also have no static power dissipation, since the circuits are designed such that the pull-down and pull-up networks are mutually exclusive. The analysis of the DC voltage transfer characteristics and the noise margins is more complicated than for the inverter, as these parameters **depend upon the data input patterns** applied to gate.

Consider the static two-input NAND gate shown in Figure 6-7. Three possible input combinations switch the output of the gate from high to low: (a) $A = B = 0 \rightarrow 1$, (b) $A = 1, B = 0 \rightarrow 1$, and (c) $B = 1, A = 0 \rightarrow 1$. The resulting voltage transfer curves display significant differences. The large variation between case (a) and the others (b and c) is explained by the fact that in the former case, both transistors in the pull-up network are on simultaneously for $A = B = 0$, representing a strong pull-up. In the latter cases, only one of the pull-up devices is on. The VTC is shifted to the left as a result of the weaker PUN.

The difference between (b) and (c) results mainly from the state of the internal node *int* between the two NMOS devices. For the NMOS devices to turn on, both gate-to-source voltages

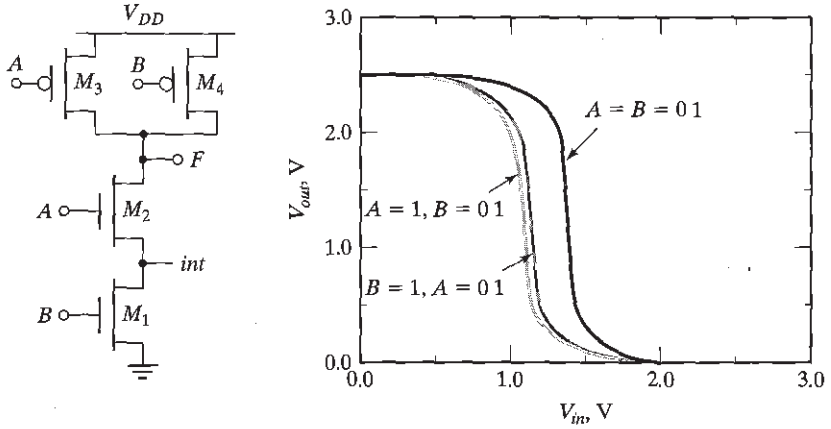


Figure 6-7 The VTC of a two-input NAND is data dependent. NMOS devices are 0.5 μm/0.25 μm while the PMOS devices are sized at 0.75 μm/0.25 μm.

must be above V_{Tn}, with $V_{GS2} = V_A - V_{DS1}$ and $V_{GS1} = V_B$. The threshold voltage of transistor M_2 will be higher than transistor M_1 due to the body effect. The threshold voltages of the two devices are given by the following equations:

$$V_{Tn2} = V_{Tn0} + \gamma((\sqrt{|2\phi_f| + V_{int}}) - \sqrt{|2\phi_f|}) \quad (6.1)$$

$$V_{Tn1} = V_{Tn0} \quad (6.2)$$

For case (b), M_3 is turned *off*, and the gate voltage of M_2 is set to V_{DD}. To a first order, M_2 may be considered as a resistor in series with M_1. Since the drive on M_2 is large, this resistance is small and has only a small effect on the voltage transfer characteristics. In case (c), transistor M_1 acts as a resistor, causing a V_T increase in M_2 due to body effect. The overall impact is quite small, as seen from the plot.

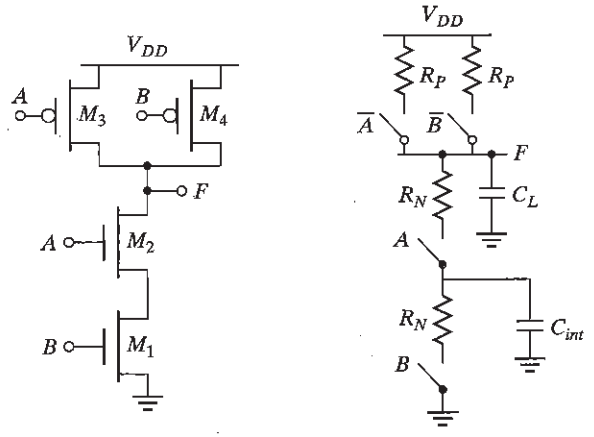
Design Consideration

The important point to take away from the preceding discussion is that the **noise margins are input/pattern dependent**. In Example 6.2, a glitch on only one of the two inputs has a larger chance of creating a false transition at the output than if the glitch were to occur on both inputs simultaneously. Therefore, the former condition has a lower low-noise margin. A common practice when characterizing gates such as NAND and NOR is to connect all the inputs together. Unfortunately, this does not represent the worst case static behavior; the data dependencies should be carefully modeled. ■

Propagation Delay of Complementary CMOS Gates

The computation of propagation delay proceeds in a fashion similar to the static inverter. For the purpose of delay analysis, each transistor is modeled as a resistor in series with an ideal switch. The value of the resistance is dependent on the power supply voltage and an equivalent large signal resistance, scaled by the ratio of device width over length, must be used. The logic is transformed into an equivalent *RC* network that includes the effect of internal node capacitances. Figure 6-8 shows the two-input NAND gate and its equivalent *RC* switch level model. Note that the internal node capacitance C_{int}—attributable to the source/drain regions and the gate overlap capacitance of M_2 and M_1—is included here. While complicating the analysis, the capacitance of the internal nodes can have quite an impact in some networks such as large fan-in gates. In a first pass, we ignore the effect of the internal capacitance.

A simple analysis of the model shows that, similarly to the noise margins, **the propagation delay depends on the input patterns**. Consider, for instance, the low-to-high transition. Three possible input scenarios can be identified for charging the output to V_{DD}. If both inputs are driven low, the two PMOS devices are on. The delay in this case is $0.69 \times (R_p/2) \times C_L$, since the two resistors are in parallel. This is not the worst case low-to-high transition, which occurs when only one device turns on, and is given by $0.69 \times R_p \times C_L$. For the pull-down path, the output is discharged only if both A and B are switched high, and the delay is given by $0.69 \times (2R_N) \times C_L$ to a first order. In other words, adding devices in series slows down the circuit, and devices must be made wider to avoid a performance penalty. When sizing the transistors in a gate with multiple inputs, we should pick the combination of inputs that triggers the worst case conditions.



(a) Two-input NAND (b) *RC*-equivalent model

Figure 6-8 Equivalent *RC* model for a two-input NAND gate.

For the NAND gate to have the same pull-down delay (t_{phl}) as a minimum-sized inverter, the NMOS devices in the PDN stack must be made twice as wide so that the equivalent resistance of the NAND pull-down network is the same as the inverter. The PMOS devices can remain unchanged.[1]

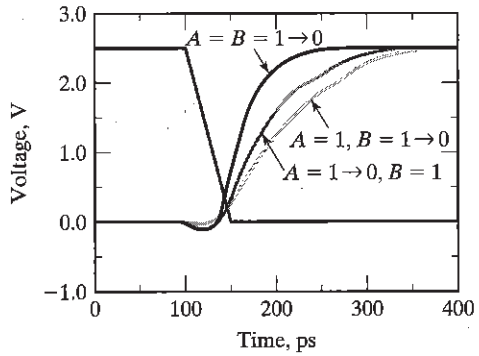
This first-order analysis assumes that the extra capacitance introduced by widening the transistors can be ignored. This is not a good assumption, in general, but it allows for a reasonable first cut at device sizing.

Example 6.3 Delay Dependence on Input Patterns

Consider the NAND gate of Figure 6-8a. Assume NMOS and PMOS devices of 0.5 μm/0.25 μm and 0.75 μm/0.25 μm, respectively. This sizing should result in approximately equal worst case rise and fall times (since the effective resistance of the pull-down is designed to be equal to the pull-up resistance).

Figure 6-9 shows the simulated low-to-high delay for different input patterns. As expected, the case in which both inputs transition go low ($A = B = 1 \rightarrow 0$) results in a smaller delay, compared with the case in which only one input is driven low. Notice that the worst case low-to-high delay depends upon which input (A or B) goes low. The reason for this involves the internal node capacitance of the pull-down stack (i.e., the source of M_2). For the case in which $B = 1$ and A transitions from $1 \rightarrow 0$, the pull-up PMOS device only has to charge up the output node capacitance (M_2 is turned off). On the other hand, for the case in which $A = 1$ and B transitions from $1 \rightarrow 0$, the pull-up PMOS device

[1] In deep-submicron processes, even larger increases in the width are needed due to the on-set of velocity saturation. For a two-input NAND, the NMOS transistors should be made 2.5 times as wide instead of 2 times.



Input Data Pattern	Delay (ps)
$A = B = 0 \rightarrow 1$	69
$A = 1, B = 0 \rightarrow 1$	62
$A = 0 \rightarrow 1, B = 1$	50
$A = B = 1 \rightarrow 0$	35
$A = 1, B = 1 \rightarrow 0$	76
$A = 1 \rightarrow 0, B = 1$	57

Figure 6-9 Example showing the delay dependence on input patterns.

has to charge up the sum of the output and the internal node capacitances, which slows down the transition.

The table in Figure 6-9 shows a compilation of various delays for this circuit. The first-order transistor sizing indeed provides approximately equal rise and fall delays. An important point to note is that the high-to-low propagation delay depends on the initial state of the internal nodes. For example, when both inputs transition from 0 → 1, it is important to establish the state of the internal node. The worst case happens when the internal node is initially charged up to $V_{DD} - V_{Tn}$, which can be ensured by pulsing the A input from 1 → 0 → 1, while input B only makes the 0 → 1 transition. In this way, the internal node is initialized properly.

The important point to take away from this example is that estimation of delay can be fairly complex, and requires a careful consideration of internal node capacitances and data patterns. Care must be taken to model the worst case scenario in the simulations. A brute force approach that applies all possible input patterns may not always work, because it is important to consider the state of internal nodes.

The CMOS implementation of a NOR gate ($F = \overline{A + B}$) is shown in Figure 6-10. The output of this network is high, if and only if both inputs A and B are low. The worst case pull-down transition happens when only one of the NMOS devices turns on (i.e., if either A or B is high). Assume that the goal is to size the NOR gate such that it has approximately the same delay as an inverter with the following device sizes: NMOS of 0.5 μm/0.25 μm and PMOS of 1.5 μm/0.25 μm. Since the pull-down path in the worst case is a single device, the NMOS devices (M_1 and M_2) can have the same device widths as the NMOS device in the inverter. For the output to be pulled high, both devices must be turned on. Since the resistances add, the devices must be made two times larger compared with the PMOS in the inverter (i.e., M_3 and M_4 must have a size of 3 μm/0.25 μm). Since PMOS devices have a lower mobility relative to NMOS devices, stacking devices in series must be avoided as much as possible. A NAND implementation is clearly preferred over a NOR implementation for implementing generic logic.

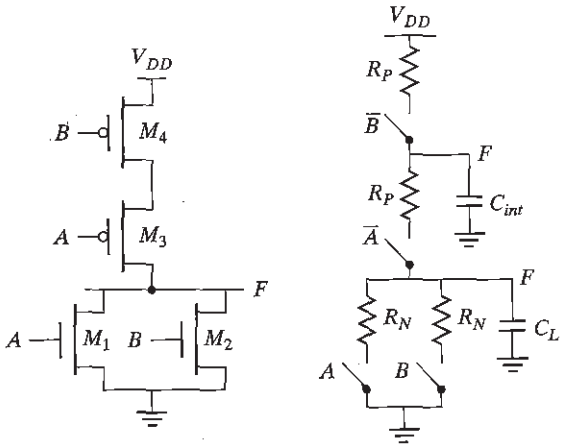


Figure 6-10 Sizing of a NOR gate.

Problem 6.1 Transistor Sizing in Complementary CMOS Gates

Determine the sizes of the transistors in Figure 6-6c such that it has approximately the same t_{plh} and t_{phl} as an inverter with the following sizes: NMOS: 0.5 μm/0.25 μm and a PMOS: 1.5 μm/0.25 μm.

So far in the analysis of propagation delay, we have ignored the effect of internal node capacitances. This is often a reasonable assumption for a first-order analysis. However, in more complex logic gates with large *fan-ins*, the internal node capacitances can become significant. Consider a four-input NAND gate, as drawn in Figure 6-11, which shows the equivalent *RC* model of the gate, including the internal node capacitances. The internal capacitances consist of the junction capacitances of the transistors, as well as the gate-to-source and gate-to-drain capacitances. The latter are turned into capacitances to ground using the Miller equivalence. The delay analysis for such a circuit involves solving distributed *RC* networks, a problem we already encountered when analyzing the delay of interconnect networks. Consider the pull-down delay of the circuit. The output is discharged when all inputs are driven high. The proper initial conditions must be placed on the internal nodes (i.e., the internal nodes must be charged to $V_{DD} - V_{TN}$) before the inputs are driven high.

The propagation delay can be computed by using the Elmore delay model:

$$t_{pHL} = 0.69(R_1 \cdot C_1 + (R_1 + R_2) \cdot C_2 + (R_1 + R_2 + R_3) \cdot C_2 + (R_1 + R_2 + R_3 + R_4) \cdot C_L) \quad (6.3)$$

Notice that the resistance of M_1 appears in all the terms, which makes this device especially important when attempting to minimize delay. Assuming that all NMOS devices have an equal size, Eq. (6.3) simplifies to

$$t_{pHL} = 0.69R_N(C_1 + 2 \cdot C_2 + 3 \cdot C_3 + 4 \cdot C_L) \quad (6.4)$$

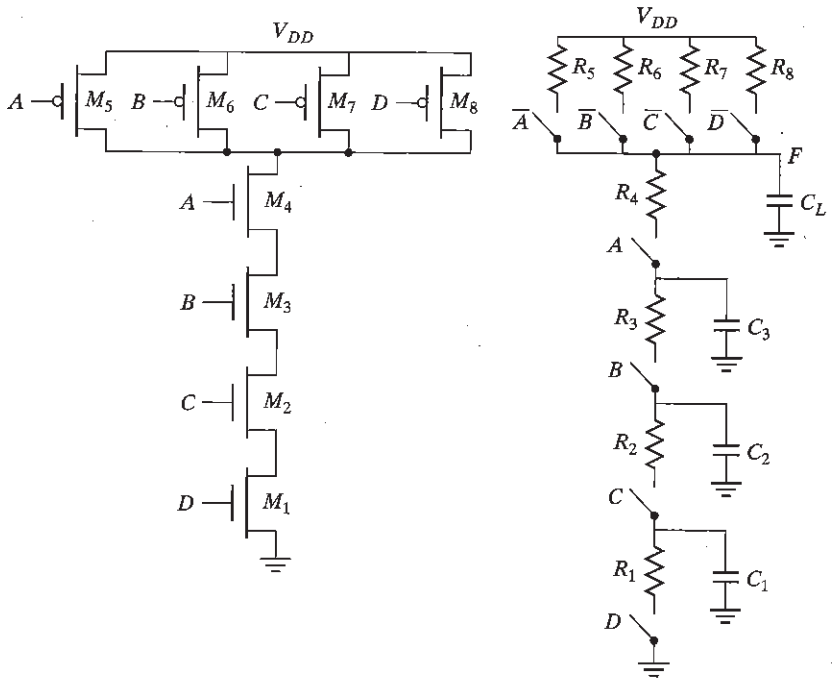


Figure 6-11 Four-input NAND gate and its *RC* model.

Example 6.4 A Four-Input Complementary CMOS NAND Gate

In this example, we evaluate the *intrinsic (or unloaded) propagation delay* of a four-input NAND gate (without any loading) is evaluatedusing hand analysis and simulation. The layout of the gate is shown in Figure 6-12. Assume that all NMOS devices have a *W/L* of 0.5 μm/0.25 μm, and all PMOS devices have a device size of 0.375 μm/0.25 μm. The devices are sized such that the worst case rise and fall times are approximately equal to a first order (ignoring the internal node capacitances).

By using techniques similar to those employed for the CMOS inverter in Chapter 5, the capacitance values can be computed from the layout. Notice that in the pull-up path, the PMOS devices share the drain terminal, in order to reduce the overall parasitic contribution. Using our standard design rules, we find that the area and perimeter for various devices can be easily computed, as shown in Table 6-2.

In this example, we focus on the pull-down delay, and the capacitances will be computed for the high-to-low transition at the output. While the output makes a transition from V_{DD} to 0, the internal nodes only transition from $V_{DD} - V_{Tn}$ to GND. We need to linearize the internal junction capacitances for this voltage transition, but, to simplify the analysis, we use the same K_{eff} for the internal nodes as for the output node.

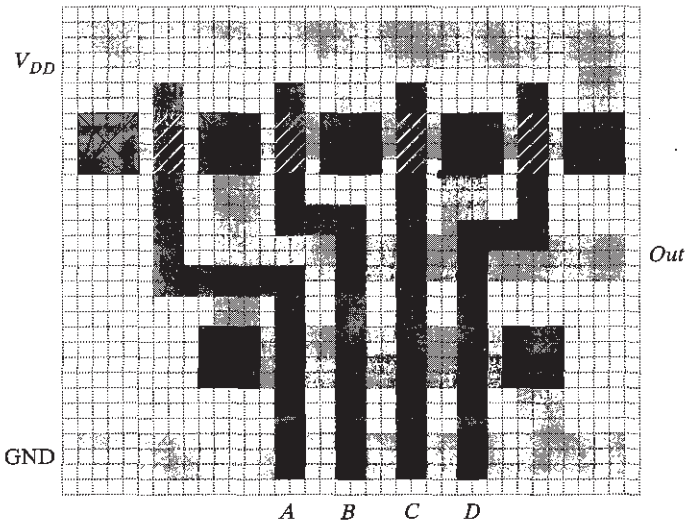


Figure 6-12 Layout a four-input NAND gate in complementary CMOS. See also Colorplate 7.

Table 6-2 Area and perimeter of transistors in four-input NAND gate.

Transistor	W (μm)	AS (μm²)	AD (μm²)	PS (μm)	PD (μm)
1	0.5	0.3125	0.0625	1.75	0.25
2	0.5	0.0625	0.0625	0.25	0.25
3	0.5	0.0625	0.0625	0.25	0.25
4	0.5	0.0625	0.3125	0.25	1.75
5	0.375	0.297	0.172	1.875	0.875
6	0.375	0.172	0.172	0.875	0.875
7	0.375	0.172	0.172	0.875	0.875
8	0.375	0.297	0.172	1.875	0.875

It is assumed that the output connects to a single, minimum-size inverter. The effect of intracell routing, which is small, is ignored. The various contributions are summarized in Table 6-3. For the NMOS and PMOS junctions, we use $K_{eq} = 0.57$, $K_{eqsw} = 0.61$, and $K_{eq} = 0.79$, $K_{eqsw} = 0.86$, respectively. Notice that the gate-to-drain capacitance is multiplied by a factor of two for all internal nodes as well as the output node, to account for the Miller effect. (This ignores the fact that the internal nodes have a slightly smaller swing due to the threshold drop.)

Table 6-3 Computation of capacitances for high-to-low transition at the output. The table shows the intrinsic delay of the gate without extra loading. Any *fan-out* capacitance would simply be added to the C_L term.

Capacitor	Contributions (H → L)	Value (fF) (H → L)
C1	$C_{d1} + C_{s2} + 2 * C_{gd1} + 2 * C_{gs2}$	(0.57 * 0.0625 * 2 + 0.61 * 0.25 * 0.28) + (0.57 * 0.0625 * 2 + 0.61 * 0.25* 0.28) + 2 * (0.31 * 0.5) + 2 * (0.31 * 0.5) = 0.85 fF
C2	$C_{d2} + C_{s3} + 2 * C_{gd2} + 2 * C_{gs3}$	(0.57 * 0.0625 * 2 + 0.61 * 0.25 * 0.28) + (0.57 * 0.0625 * 2 + 0.61 * 0.25* 0.28) + 2 * (0.31 * 0.5) + 2 * (0.31 * 0.5) = 0.85 fF
C3	$C_{d3} + C_{s4} + 2 * C_{gd3} + 2 * C_{gs4}$	(0.57 * 0.0625 * 2+ 0.61 * 0.25 * 0.28) + (0.57 * 0.0625 * 2+ 0.61 * 0.25* 0.28) + 2 * (0.31 * 0.5) + 2 * (0.31 * 0.5) = 0.85 fF
CL	$C_{d4} + 2 * C_{gd4} + C_{d5} + C_{d6} + C_{d7} + C_{d8} + 2 * C_{gd5} + 2 * C_{gd6} + 2 * C_{gd7} + 2 * C_{gd8} = C_{d4} + 4 * C_{d5} + 4 * 2 * C_{gd6}$	(0.57 * 0.3125 * 2 + 0.61 * 1.75 *0.28) + 2 * (0.31 * 0.5)+ 4 * (0.79 * 0.171875* 1.9+ 0.86 * 0.875 * 0.22)+ 4 * 2 * (0.27 * 0.375) = 3.47 fF

Using Eq. (6.4), we compute the propagation delay, as follows:

$$t_{pHL} = 0.69\left(\frac{13\text{K}\Omega}{2}\right)(0.85\,\text{fF} + 2 \cdot 0.85\,\text{fF} + 3 \cdot 0.85\,\text{fF} + 4 \cdot 3.47\,\text{fF}) = 85\ \text{ps}$$

The simulated delay for this particular transition was found to be 86 ps! The hand analysis gives a fairly accurate estimate, given all of the assumptions and linearizations that were made. For example, we assume that the gate–source (or gate–drain) capacitance only consists of the overlap component. This is not entirely the case, because, during the transition, some other contributions come in place depending upon the operating region. Once again, the goal of hand analysis is not to provide a totally accurate delay prediction, but rather to give intuition into what factors influence the delay and to aid in initial transistor sizing. Accurate timing analysis and transistor optimization is usually done using SPICE. The simulated worst case low-to-high delay time for this gate was 106 ps.

While complementary CMOS is a very robust and simple approach for implementing logic gates, there are two major problems associated with using this style as the complexity of the gate (i.e., *fan-in*) increases. First, the number of transistors required to implement an N fan-in gate is $2N$. This can result in a significantly large implementation area.

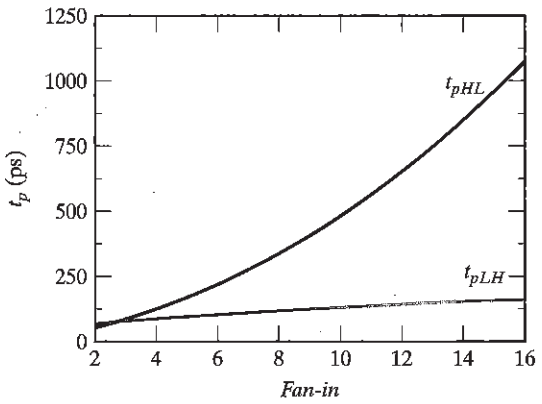


Figure 6-13 Propagation delay of CMOS NAND gate as a function of fan-in. A fan-out of one inverter is assumed, and all pull-down transistors are minimal size.

The second problem is that propagation delay of a complementary CMOS gate deteriorates rapidly as a function of the fan-in. In fact, the *unloaded intrinsic delay* of the gate is, at worst, a *quadratic function of the fan-in.*

- The large number of transistors ($2N$) increases the overall capacitance of the gate. For an N-input gate, the *intrinsic capacitance* increases linearly with the fan-in. Consider, for instance, the NAND gate of Figure 6-11. Given the linear increase in the number of PMOS devices connected to the output node, we expect the low-to-high delay of the gate to increase linearly with fan-in—while the capacitance goes up linearly, the pull-up resistance remains unchanged.
- The series connection of transistors in either the PUN or PDN of the gate causes an additional slowdown. We know that the *distributed RC network* in the PDN of Figure 6-11 comes with a delay that is quadratic in the number of elements in the chain. The high-to-low delay of the gate should hence be a quadratic function of the fan-in.

Figure 6-13 plots the (intrinsic) propagation delay of a NAND gate as a function of fan-in assuming a fixed fan-out of one inverter (NMOS: 0.5 μm and PMOS: 1.5 μm). As predicted, t_{pLH} is a linear function of fan-in, while the simultaneous increase in the pull-down resistance and the load capacitance cause an approximately quadratic relationship for t_{pHL}. Gates with a *fan-in* greater than or equal to 4 become excessively slow and must be avoided.

Design Techniques for Large Fan-in

The designer has a number of techniques at his disposition to reduce the delay of large fan-in circuits:

- **Transistor Sizing** The most obvious solution is to increase the transistor sizes. This lowers the resistance of devices in series and lowers the time constants. However, increasing the transistor sizes results in larger parasitic capacitors, which not only affect the *propagation delay* of the gate in question, but

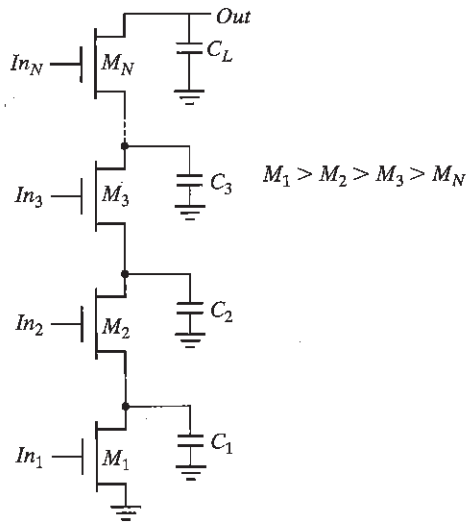


Figure 6-14 Progressive sizing of transistors in large transistor chains copes with the extra load of internal capacitances.

also present a larger load to the preceding gate. This technique should therefore be used with caution. If the load capacitance is dominated by the intrinsic capacitance of the gate, widening the device only creates a "self-loading" effect, and the *propagation delay* is unaffected. Sizing is only effective when the load is dominated by the fan-out. A more comprehensive approach toward sizing transistors in complex CMOS combinational networks is discussed in the next section.

- **Progressive Transistor Sizing** An alternate approach to uniform sizing (in which each transistor is scaled up uniformly), is to use progressive transistor sizing (Figure 6-14). Referring back to Eq. (6.3), we see that the resistance of M_1 (R_1) appears N times in the delay equation, the resistance of M_2 (R_2) appears $N-1$ times, etc. From the equation, it is clear that R_1 should be made the smallest, R_2 the next smallest, etc. Consequently, a progressive scaling of the transistors is beneficial: $M_1 > M_2 > M_3 > M_N$. This approach reduces the dominant resistance, while keeping the increase in capacitance within bounds. For an excellent treatment on the optimal sizing of transistors in a complex network, we refer the interested reader to [Shoji88, pp. 131–143]. You should be aware, however, of one important pitfall of this approach. While progressive resizing of transistors is relatively easy in a schematic diagram, it is not as simple in a real layout. Very often, design-rule considerations force the designer to push the transistors apart, which causes the internal capacitance to grow. This may offset all the gains of the resizing!
- **Input Reordering** Some signals in complex combinational logic blocks might be more critical than others. Not all inputs of a gate arrive at the same time (due, for instance, to the propagation delays of the preceding logical gates). An input signal to a gate is called *critical* if it is the last signal of all inputs to assume a stable value. The path through the logic which determines the ultimate speed of the structure is called the *critical path.*

  Putting the critical-path transistors closer to the output of the gate can result in a speed up, as demonstrated in Figure 6-15. Signal In_1 is assumed to be a critical signal. Suppose further that In_2 and In_3 are high, and that In_1 undergoes a $0 \rightarrow 1$ transition. Assume also that C_L is initially charged high. In case (a), no path to GND exists until M_1 is turned on, which, unfortunately, is the last event to happen. The delay between the arrival of In_1 and the output is therefore determined by the time it takes to discharge C_L, C_1, and C_2. In the second case, C_1 and C_2 are already

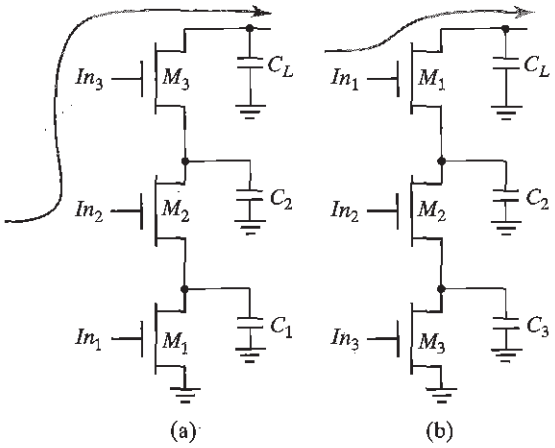


Figure 6-15 Influence of transistor ordering on delay. Signal In_1 is the critical signal.

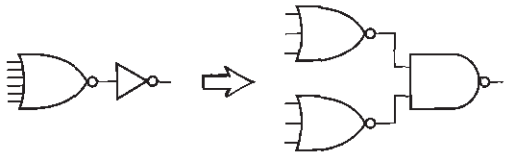


Figure 6-16 Logic restructuring can reduce the gate fan-in.

discharged when In_1 changes. Only C_L still has to be discharged, resulting in a smaller delay.

- **Logic Restructuring** Manipulating the logic equations can reduce the fan-in requirements and thus reduce the gate delay, as illustrated in Figure 6-16. The quadratic dependency of the gate delay on *fan-in* makes the six-input NOR gate extremely slow. Partitioning the NOR gate into two three-input gates results in a significant speedup, which by far offsets the extra delay incurred by turning the inverter into a two-input NAND gate. ■

Optimizing Performance in Combinational Networks

Earlier, we established that minimization of the propagation delay of a gate in isolation is a purely academic effort. The sizing of devices should happen in its proper context. In Chapter 5, we developed a methodology to do so for inverters. We also found that an optimal fan-out for a chain of inverters driving a load C_L is $(C_L/C_{in})^{1/N}$, where N is the number of stages in the chain, and C_{in} the input capacitance of the first gate in the chain. If we have an opportunity to select the number of stages, we found out that we would like to keep the fan-out per stage around 4. Can this result be extended to determine the size of any combinational path for minimal delay? By extending our previous approach to address complex logic networks, we find out that this is indeed possible [Sutherland99].[2]

[2]The approach introduced in this section is commonly called logical effort, and was formally introduced in [Sutherland99], which presents an extensive treatment of the topic. The treatment offered here represents only a glance over of the overall approach.

Table 6-4 Estimates of intrinsic delay factors of various logic types, assuming simple layout styles, and a fixed PMOS–NMOS ratio.

Gate type	p
Inverter	1
n-input NAND	n
n-input NOR	n
n-way multiplexer	$2n$
XOR, NXOR	$n2^{n-1}$

To do so, we modify the basic delay equation of the inverter that we introduced in Chapter 5, namely,

$$t_p = t_{p0}\left(1 + \frac{C_{ext}}{\gamma C_g}\right) = t_{p0}(1 + f/\gamma) \tag{6.5}$$

to

$$t_p = t_{p0}(p + gf/\gamma) \tag{6.6}$$

with t_{p0} still representing the intrinsic delay of an inverter and f the *effective fan-out*, defined as the ratio between the external load and the input capacitance of the gate. In this context, f is also called the *electrical effort*, and p represents the ratio of the intrinsic (or unloaded) delays of the complex gate and the simple inverter, and is a function of gate topology, as well as layout style. The more involved structure of the multiple-input gate causes its intrinsic delay to be higher than that of an inverter. Table 6-4 enumerates the values of p for some standard gates, assuming simple layout styles, and ignoring second-order effects such as internal node capacitances.

The factor g is called the *logical effort*, and represents the fact that, for a given load, complex gates have to work harder than an inverter to produce a similar response. In other words, the logical effort of a logic gate tells how much worse it is at producing output current than an inverter, given that each of its inputs may present only the same input capacitance as the inverter. Equivalently, logical effort is how much more input capacitance a gate presents to deliver the same output current as an inverter. Logical effort is a useful parameter, because it depends only on circuit topology. The logical efforts of some common logic gates are given in Table 6-5.

Table 6-5 Logic efforts of common logic gates, assuming a PMOS–NMOS ratio of 2.

Gate Type	Number of Inputs			
	1	2	3	n
Inverter	1			
NAND		4/3	5/3	(n + 2)/3
NOR		5/3	7/3	(2n + 1)/3
Multiplexer		2	2	2
XOR		4	12	

Example 6.5 Logical Effort of Complex Gates

Consider the gates shown in Figure 6-17. Assuming PMOS–NMOS ratio of 2, the input capacitance of a minimum-sized symmetrical inverter equals three times the gate capacitance of a minimum-sized NMOS (called C_{unit}). We size the two-input NAND and NOR such that their equivalent resistances equal the resistance of the inverter (using the techniques described earlier). This increases the input capacitance of the two-input NAND to 4 C_{unit}, or 4/3 the capacitance of the inverter. The input capacitance of the two-input NOR is 5/3 that of the inverter. Equivalently, for the same input capacitance, the NAND and NOR gate have 4/3 and 5/3 less driving strength than the inverter. This affects the delay component that corresponds to the load, increasing it by this same factor, called the *logical effort*. Hence, $g_{NAND} = 4/3$, and $g_{NOR} = 5/3$.

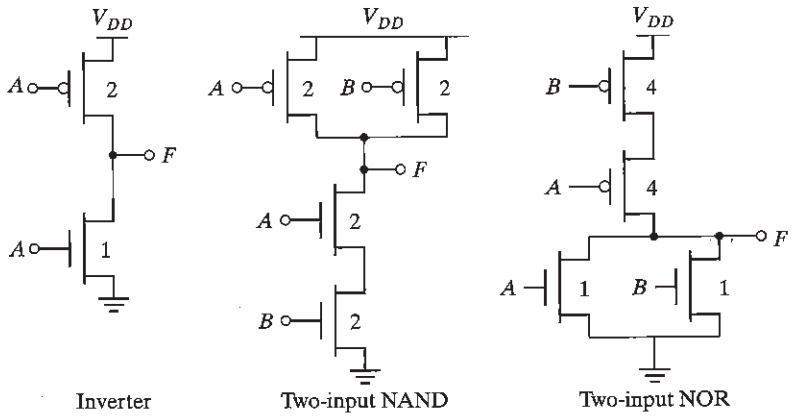


Figure 6-17 Logical effort of two-input NAND and NOR gates.

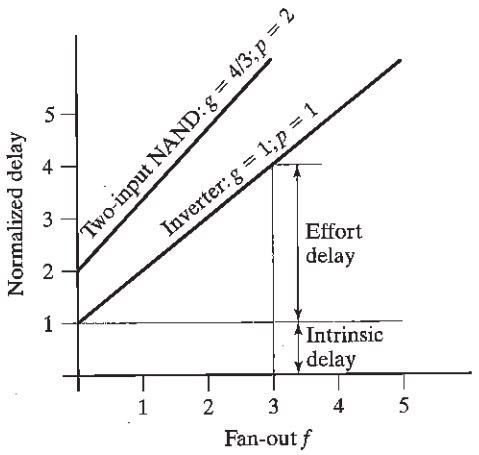


Figure 6-18 Delay as a function of fan-out for an inverter and a two-input NAND.

The delay model of a logic gate, as represented in Eq. (6.6), is a simple linear relationship. Figure 6-18 shows this relationship graphically: the delay is plotted as a function of the fan-out for an inverter and for a two-input NAND gate. The slope of the line is the logical effort of the gate; its intercept is the intrinsic delay. The graph shows that we can adjust the delay by adjusting the effective fan-out (by transistor sizing) or by choosing a logic gate with a different logical effort. Observe also that fan-out and logical effort contribute to the delay in a similar way. We call the product of the two $h = fg$, the *gate effort*.

The total delay of a path through a combinational logic block can now be expressed as

$$t_p = \sum_{j=1}^{N} t_{p,j} = t_{p0} \sum_{j=1}^{N} \left(p_j + \frac{f_j g_j}{\gamma} \right) \tag{6.7}$$

We use a similar procedure as we did for the inverter chain in Chapter 5 to determine the minimum delay of the path. By finding $N-1$ partial derivatives and setting them to zero, we find that **each stage should bear the same gate effort**:

$$f_1 g_1 = f_2 g_2 = \ldots = f_N g_N \tag{6.8}$$

The logical effort along a path in the network compounds by multiplying the logical efforts of all the gates along the path, yielding the *path logical effort* G:

$$G = \prod_{1}^{N} g_i \tag{6.9}$$

We also can define a *path effective fan-out* (or *electrical effort*) F, which relates the load capacitance of the last gate in the path to the input capacitance of the first gate:

$$F = \frac{C_L}{C_{g1}} \tag{6.10}$$

To relate F to the effective fan-outs of the individual gates, we must introduce another factor to account for the logical fan-out within the network. When fan-out occurs at the output of a node, some of the available drive current is directed along the path we are analyzing, and some is directed off the path. We define the *branching effort b* of a logical gate on a given path to be

$$b = \frac{C_{\text{on-path}} + C_{\text{off-path}}}{C_{\text{on-path}}} \tag{6.11}$$

where $C_{\text{on-path}}$ is the load capacitance of the gate along the path we are analyzing and $C_{\text{off-path}}$ is the capacitance of the connections that lead off the path. Note that the branching effort is, if the path does not branch (as in a chain of gates). The *path branching effort* is defined as the product of the branching efforts at each of the stages along the path, or

$$B = \prod_{1}^{N} b_i \tag{6.12}$$

The path electrical effort can now be related to the electrical and branching efforts of the individual stages:

$$F = \prod_{1}^{N} \frac{f_i}{b_i} = \frac{\prod f_i}{B} \tag{6.13}$$

Finally, the total path effort H can be defined. Using Eq. (6.13), we write

$$H = \prod_{1}^{N} h_i = \prod_{1}^{N} g_i f_i = GFB \tag{6.14}$$

From here on, the analysis proceeds along the same lines as the inverter chain. The gate effort that minimizes the path delay is

$$h = \sqrt[N]{H} \tag{6.15}$$

and the minimum delay through the path is

$$D = t_{p0}\left(\sum_{j=1}^{N} p_j + \frac{N(\sqrt[N]{H})}{\gamma}\right) \tag{6.16}$$

Note that the path intrinsic delay is a function of the types of logic gates in the path and is not affected by the sizing. The size factors of the individual gates in the chain s_i can then be derived by working from front to end (or vice versa). We assume that a unit-size gate has a driving capability equal to a minimum-size inverter. Based on the definition of the logical effort, this means that its input capacitance is g times larger than that of the reference inverter, which equals C_{ref}. With s_1 the sizing factor of the first gate in the chain, the input capacitance of the chain C_{g1}

equals $g_1 s_1 C_{ref}$. Including the branching effort, we know that the input capacitance of gate 2 is (f_1/b_1) larger, or

$$g_2 s_2 C_{ref} = \left(\frac{f_1}{b_1}\right) g_1 s_1 C_{ref} \tag{6.17}$$

For gate i in the chain, this yields

$$s_i = \left(\frac{g_1 s_1}{g_i}\right) \prod_{j=1}^{i-1} \left(\frac{f_j}{b_j}\right) \tag{6.18}$$

Example 6.6 Sizing Combinational Logic for Minimum Delay

Consider the logic network of Figure 6-19, which may represent the critical path of a more complex logic block. The output of the network is loaded with a capacitance which is five times larger than the input capacitance of the first gate, which is a minimum-sized inverter. The effective fan-out of the path thus equals $F = C_L/C_{g1} = 5$. Using the entries in Table 6-5, we find the path logical effort as follows:

$$G = 1 \times \frac{5}{3} \times \frac{5}{3} \times 1 = \frac{25}{9}$$

Since there is no branching, $B = 1$. Hence, $H = GFB = 125/9$, and the optimal stage effort h is $\sqrt[4]{H} = 1.93$. Taking into account the gate types, we derive the following fan-out factors: $f_1 = 1.93$; $f_2 = 1.93 \times (3/5) = 1.16$; $f_3 = 1.16$; $f_4 = 1.93$. Notice that the inverters are assigned larger than the more complex gates because they are better at driving loads.

Finally, we derive the gate sizes (with respect to the minimum-sized versions) using Eq. (6.18). This leads to the following values: $a = f_1 g_1/g_2 = 1.16$; $b = f_1 f_2 g_1/g_3 = 1.34$; and $c = f_1 f_2 f_3 g_1/g_4 = 2.60$.

These calculations do not have to be very precise. As discussed in Chapter 5, sizing a gate too large or too small by a factor of 1.5 still results in circuits within 5% of minimum delay. Therefore, the "back of the envelope" hand calculations using this technique are quite effective.

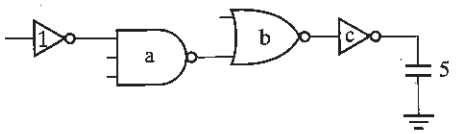


Figure 6-19 Critical path of combinational network.

Problem 6.2 Sizing an Inverter Network

Revisit Problem 5.5, but this time around use the branching-effort approach to produce the solution.

Power Consumption in CMOS Logic Gates

The sources of power consumption in a complementary CMOS inverter were discussed in detail in Chapter 5. Many of these issues apply directly to complex CMOS gates. The power dissipation is a strong function of transistor sizing (which affects physical capacitance,) input and output rise–fall times (which determine the short-circuit power,) device thresholds and temperature (which impact leakage power,) and switching activity. The dynamic power dissipation is given by $\alpha_{0\to1} C_L V_{DD}^2 f$. Making a gate more complex mostly affects the *switching activity* $\alpha_{0\to1}$, which has two components: a static component that is only a function of the topology of the logic network, and a dynamic one that results from the timing behavior of the circuit. (The latter factor is also called glitching.)

Logic Function The transition activity is a strong function of the logic function being implemented. For static CMOS gates with statistically independent inputs, the static transition probability is the probability p_0 that the output will be in the *zero* state in one cycle, multiplied by the probability p_1 that the output will be in the *one* state in the next cycle:

$$\alpha_{0\to1} = p_0 \cdot p_1 = p_0 \cdot (1 - p_0) \tag{6.19}$$

Assuming that the inputs are independent and uniformly distributed, any N-input static gate has a transition probability given by

$$\alpha_{0\to1} = \frac{N_0}{2^N} \cdot \frac{N_1}{2^N} = \frac{N_0 \cdot (2^N - N_0)}{2^{2N}} \tag{6.20}$$

where N_0 is the number of *zero* entries, and N_1 is the number of *one* entries in the output column of the truth table of the function. To illustrate, consider a static two-input NOR gate whose truth table is shown in Table 6-6. Assume that only one input transition is possible during a clock cycle and that the inputs to the NOR gate have a uniform input distribution (in other words, the four possible states for inputs A and B—00, 01, 10, 11—are equally likely).

Table 6-6 Truth table of a two-input NOR gate.

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

From Table 6-6 and Eq. (6.20), the output transition probability of a two-input static CMOS NOR gate can be derived:

$$\alpha_{0\to 1} = \frac{N_0 \cdot (2^N - N)}{2^{2N}} = \frac{3 \cdot (2^2 - 3)}{2^{2\cdot 2}} = \frac{3}{16} \tag{6.21}$$

Problem 6.3 ***N*-Input XOR Gate**

Assuming the inputs to an *N*-input XOR gate are uncorrelated and uniformly distributed, derive the expression for the switching activity factor.

Signal Statistics The switching activity of a logic gate is a strong function of the input signal statistics. Using a uniform input distribution to compute activity is not a good technique, since the propagation through logic gates can significantly modify the signal statistics. For example, consider once again a two-input static NOR gate, and let p_a and p_b be the probabilities that the inputs A and B are *one*. Assume further that the inputs are not correlated. The probability that the output node is 1 is given by

$$p_1 = (1 - p_a)(1 - p_b) \tag{6.22}$$

Therefore, the probability of a transition from 0 to 1 is

$$\alpha_{0\to 1} = p_0 p_1 = (1 - (1 - p_a)(1 - p_b))(1 - p_a)(1 - p_b) \tag{6.23}$$

Figure 6-20 shows the transition probability as a function of p_a and p_b. Observe how this graph degrades into the simple inverter case when one of the input probabilities is set to 0. From

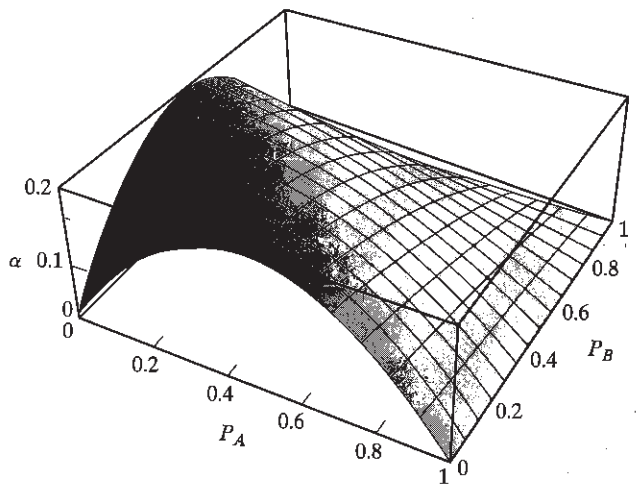


Figure 6-20 Transition activity of a two-input NOR gate as a function of the input probabilities (p_A, p_B).

this plot, it is clear that understanding the signal statistics and their impact on switching events can be used to significantly impact the power dissipation.

Problem 6.4 Power Dissipation of Basic Logic Gates

Derive the $0 \rightarrow 1$ output transition probabilities for the basic logic gates (AND, OR, XOR). The results to be obtained are given in Table 6-7.

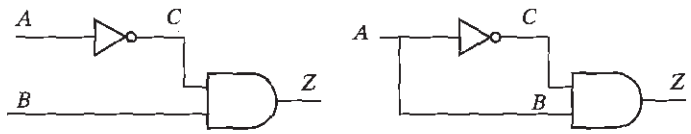
Table 6-7 Output transition probabilities for static logic gates.

	$\alpha_{0\rightarrow1}$
AND	$(1-p_A p_B)p_A p_B$
OR	$(1-p_A)(1-p_B)[1-(1-p_A)(1-p_B)]$
XOR	$[1-(p_A+p_B-2p_A p_B)](p_A+p_B-2p_A p_B)$

Intersignal Correlations The evaluation of the switching activity is further complicated by the fact that signals exhibit correlation in space and time. Even if the primary inputs to a logic network are uncorrelated, the signals become correlated or "colored," as they propagate through the logic network. This is best illustrated with a simple example. Consider first the circuit shown in Figure 6-21a, and assume that the primary inputs A and B are uncorrelated and uniformly distributed. Node C has a **1** (**0**) probability of 1/2, and a $0 \rightarrow 1$ transition probability of 1/4. The probability that the node Z undergoes a power consuming transition is then determined using the AND-gate expression of Table 6-7:

$$p_{0\rightarrow1} = (1-p_a p_b)\, p_a p_b = (1 - 1/2 \cdot 1/2)\, 1/2 \cdot 1/2 = 3/16 \tag{6.24}$$

The computation of the probabilities is straightforward: signal and transition probabilities are evaluated in an ordered fashion, progressing from the input to the output node. This approach, however, has two major limitations: (1) it does not deal with circuits with feedback as found in sequential circuits, and (2) it assumes that the signal probabilities at the input of each gate are independent. This is rarely the case in actual circuits, where reconvergent fan-out often causes intersignal dependencies. For instance, the inputs to the AND gate in Figure 6-21b (C and B) are interdependent because both are a function of A. The approach to computing



(a) Logic circuit without reconvergent fan-out

(b) Logic circuit with reconvergent fan-out

Figure 6-21 Example illustrating the effect of signal correlations.

probabilities that we presented previously fails under these circumstances. Traversing from inputs to outputs yields a transition probability of 3/16 for node Z, similar to the previous analysis. This value clearly is false, as logic transformations show that the network can be reduced to $Z = C \cdot B = A \cdot \overline{A} = 0$, and thus no transition will ever take place.

To get the precise results in the progressive analysis approach, its is essential to take signal interdependencies into account. This can be accomplished with the aid of conditional probabilities. For an AND gate, Z equals 1 if and only if B and C are equal to 1. Thus,

$$p_Z = p(Z = 1) = p(B = 1, C = 1) \tag{6.25}$$

where $p(B = 1, C = 1)$ represents the probability that B and C are equal to 1 simultaneously. If B and C are independent, $p(B = 1, C = 1)$ can be decomposed into $p(B = 1) \cdot p(C = 1)$, and this yields the expression for the AND gate derived earlier: $p_Z = p(B = 1) \cdot p(C = 1) = p_B p_C$. If a dependency between the two exists (as is the case in Figure 6-21b), a conditional probability has to be employed, such as the following:

$$p_Z = p(C = 1|B = 1) \cdot p(B = 1) \tag{6.26}$$

The first factor in Eq. (6.26) represents the probability that $C = 1$ given that $B = 1$. The extra condition is necessary because C is dependent upon B. Inspection of the network shows that this probability is equal to 0, since C and B are logical inversions of each other, resulting in the signal probability for Z, $p_Z = 0$.

Deriving those expressions in a structured way for large networks with reconvergent fanout is complex, especially when the networks contain feedback loops. Computer support is therefore essential. To be meaningful, the analysis program has to process a typical sequence of input signals, because the power dissipation is a strong function of statistics of those signals.

Dynamic or Glitching Transitions When analyzing the transition probabilities of complex, multistage logic networks in the preceding section, we ignored the fact that the gates have a nonzero propagation delay. In reality, the finite propagation delay from one logic block to the next can cause spurious transitions known as *glitches or dynamic hazards* to occur: a node can exhibit multiple transitions in a single clock cycle before settling to the correct logic level.

A typical example of the effect of glitching is shown in Figure 6-22, which displays the simulated response of a chain of NAND gates for all inputs going simultaneously from 0 to 1. Initially, all the outputs are 1 since one of the inputs was 0. For this particular transition, all the odd bits must transition to 0, while the even bits remain at the value of 1. However, due to the finite propagation delay, the even output bits at the higher bit positions start to discharge, and the voltage drops. When the correct input ripples through the network, the output goes high. The glitch on the even bits causes extra power dissipation beyond what is required to strictly implement the logic function. Although the glitches in this example are only partial (i.e., not from rail to rail), they contribute significantly to the power dissipation. Long chains of gates often occur in important structures such as adders and multipliers, and the glitching component can easily dominate the overall power consumption.

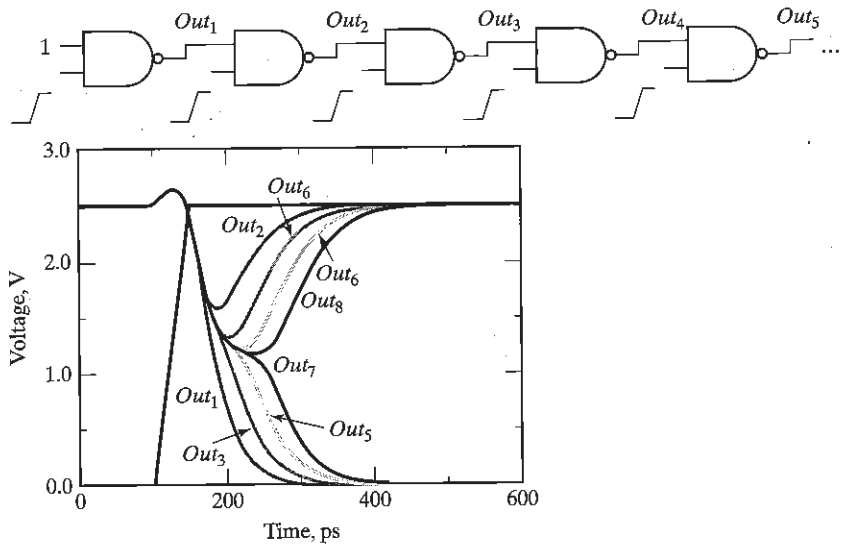


Figure 6-22 Glitching in a chain of NAND gates.

Design Techniques to Reduce Switching Activity

The dynamic power of a logic gate can be reduced by minimizing the physical capacitance and the switching activity. The physical capacitance can be minimized in a number ways, including circuit style selection, transistor sizing, placement and routing, and architectural optimizations. The switching activity, on the other hand, can be minimized at all levels of the design abstraction, and is the focus of this section. Logic structures can be optimized to minimize both the fundamental transitions required to implement a given function and the spurious transitions.

1. **Logic Restructuring** Changing the topology of a logic network may reduce its power dissipation. Consider, for example, two alternative implementations of $F = A \cdot B \cdot C \cdot D$, as shown in Figure 6-23. Ignore glitching and assume that all primary inputs (A,B,C,D) are uncorrelated and uniformly distributed (this is, $p_{1\ (a,b,c,d)} = 0.5$). Using the expressions from Table 6-7, the activity can be computed for the two topologies, as shown in Table 6-8. The results indicate that the chain implementation has an overall lower switching activity than the tree implementation for random inputs. However, as mentioned before, it is also important to consider the timing behavior to

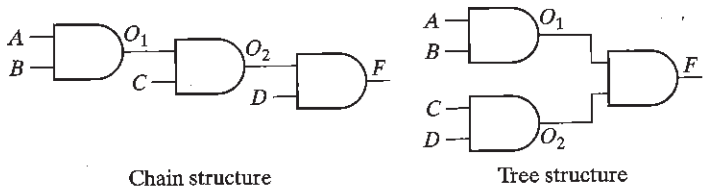


Figure 6-23 Simple example to demonstrate the influence of circuit topology on activity.

Table 6-8 Probabilities for tree and chain topologies.

	O_1	O_2	F
p_1 (chain)	1/4	1/8	1/16
$p_0 = 1\text{-}p_1$ (chain)	3/4	7/8	15/16
$p_{0\to1}$ (chain)	3/16	7/64	15/256
p_1 (tree)	1/4	1/4	1/16
$p_0 = 1\text{-}p_1$ (tree)	3/4	3/4	15/16
$p_{0\to1}$ (tree)	3/16	3/16	15/256

accurately make power trade-offs. In this example, the tree topology experiences (virtually) no glitching activity since the signal paths are balanced to all the gates.

2. **Input ordering** Consider the two static logic circuits of Figure 6-24. The probabilities that *A*, *B*, and C are equal to **1** are listed in the Figure. Since both circuits implement identical logic functionality, it is clear that the activity at the output node *Z* is equal in both cases. The difference is in the activity at the intermediate node. In the first circuit, this activity equals $(1 - 0.5 \times 0.2)\ (0.5 \times 0.2) = 0.09$. In the second case, the probability that a $0 \to 1$ transition occurs equals $(1 - 0.2 \times 0.1)\ (0.2 \times 0.1) = 0.0196$, a substantially lower value. From this, we learn that it is beneficial to postpone the introduction of signals with a high transition rate (i.e., signals with a signal probability close to 0.5). A simple reordering of the input signals is often sufficient to accomplish that goal.

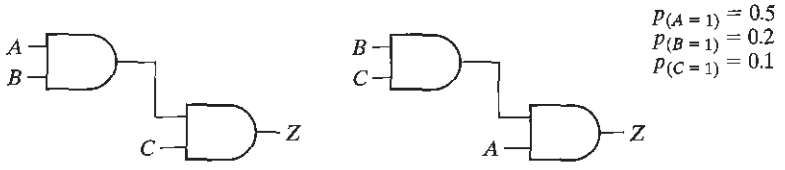
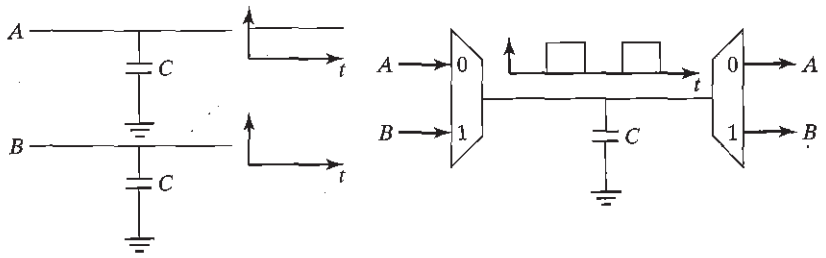


Figure 6-24 Reordering of inputs affects the circuit activity.

3. **Time-multiplexing resources** Time-multiplexing a single hardware resource—such as a logic unit or a bus—over a number of functions is a technique often used to minimize the implementation area. Unfortunately, the minimum area solution does not always result in the lowest switching activity. For example, consider the transmission of two input bits (*A* and *B*) using either dedicated resources or a time-multiplexed approach, as shown in Figure 6-25. To the first order, ignoring the multiplexer overhead, it would seem that the degree of time multiplexing should not affect the switched capacitance, since the time-multiplexed solution has half the physical capacitance switched at twice the frequency (for a fixed throughput).

   If the data being transmitted are random, it will make no difference which architecture is used. However, if the data signals have some distinct properties (such as temporal correlation), the power dissipation of the time-multiplexed solution can be significantly higher. Suppose, for instance, that *A* is always (or mostly) 1, and *B* is (mostly) 0. In the parallel solution, the switched capacitance is very low since there are very few transitions on the data bits. However, in the time-multiplexed solution,

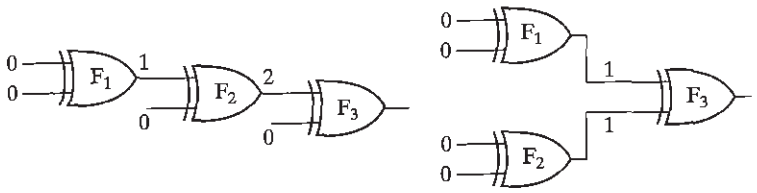


(a) Parallel data transmission (b) Serial data transmission

Figure 6-25 Parallel versus time-multiplexed data busses.

the bus toggles between 0 and 1. Care must be taken in digital systems to avoid time-multiplexing data streams with very distinct data characteristics.

4. **Glitch Reduction by balancing signal paths** The occurrence of glitching in a circuit is mainly due to a mismatch in the path lengths in the network. If all input signals of a gate change simultaneously, no glitching occurs. On the other hand, if input signals change at different times, a dynamic hazard might develop. Such a mismatch in signal timing is typically the result of different path lengths with respect to the primary inputs of the network. This is illustrated in Figure 6-26. Assume that the XOR gate has a unit delay. The first network (a) suffers from glitching as a result of the wide disparity between the arrival times of the input signals for a gate. For example, for gate F_3, one input settles at time 0, while the second one only arrives at time 2. Redesigning the network so that all arrival times are identical can dramatically reduce the number of superfluous transitions (network b).



(a) Network sensitive to glitching (b) Glitch-free network

Figure 6-26 Glitching is influenced by matching of signal path lengths. The annotated numbers indicate the signal arrival times.

■

Summary

The CMOS logic style described in the previous section is highly robust and scalable with technology, but requires $2N$ transistors to implement an N-input logic gate. Also, the load capacitance is significant, since each gate drives two devices (a PMOS and an NMOS) per *fan-out*. This has opened the door for alternative logic families that either are simpler or faster.

6.2.2 Ratioed Logic

Concept

Ratioed logic is an attempt to reduce the number of transistors required to implement a given logic function, often at the cost of reduced robustness and extra power dissipation. The purpose

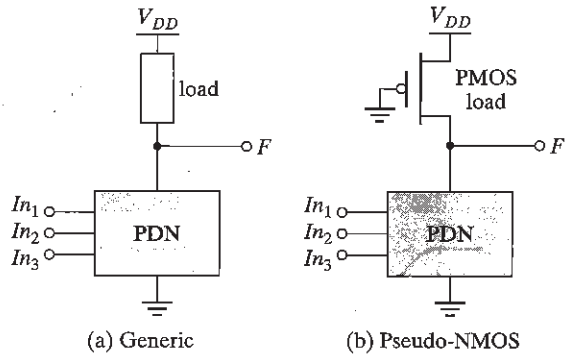


Figure 6-27 Ratioed logic gate.

of the PUN in complementary CMOS is to provide a conditional path between V_{DD} and the output when the PDN is turned *off*. In ratioed logic, the entire PUN is replaced with a single unconditional load device that pulls up the output for a high output as in Figure 6-27a. Instead of a combination of active pull-down and pull-up networks, such a gate consists of an NMOS pull-down network that realizes the *logic function*, and a simple *load device*. Figure 6-27b shows an example of ratioed logic, which uses a grounded PMOS load and is referred to as a pseudo-NMOS gate.

The clear advantage of a pseudo-NMOS gate is the reduced number of transistors ($N + 1$, versus $2N$ for complementary CMOS). The nominal high output voltage (V_{OH}) for this gate is V_{DD} since the pull-down devices are turned *off* when the output is pulled high (assuming that V_{OL} is below V_{Tn}). On the other hand, the **nominal low output voltage is not 0 V**, since there is contention between the devices in the PDN and the grounded PMOS load device. This results in reduced noise margins and, more importantly, static power dissipation. The sizing of the load device relative to the pull-down devices can be used to trade off parameters such as *noise margin*, *propagation delay*, and *power dissipation*. Since the voltage swing on the output and the overall functionality of the gate depend on the ratio of the NMOS and PMOS sizes, the circuit is called *ratioed*. This is in contrast to the *ratioless* logic styles, such as complementary CMOS, where the low and high levels do not depend on transistor sizes.

Computing the dc-transfer characteristic of the pseudo-NMOS proceeds along paths similar to those used for its complementary CMOS counterpart. The value of V_{OL} is obtained by equating the currents through the driver and load devices for $V_{in} = V_{DD}$. At this operation point, it is reasonable to assume that the NMOS device resides in linear mode (since, ideally, the output should be close to 0V), while the PMOS load is saturated:

$$k_n\left((V_{DD} - V_{Tn})V_{OL} - \frac{V_{OL}^2}{2}\right) + k_p\left((-V_{DD} - V_{Tp}) \cdot V_{DSATp} - \frac{V_{DSATp}^2}{2}\right) = 0 \qquad (6.27)$$

Assuming that V_{OL} is small relative to the gate drive ($V_{DD} - V_T$), and that V_{Tn} is equal to V_{Tp} in magnitude, V_{OL} can be approximated as

$$V_{OL} \approx \frac{k_p(V_{DD} + V_{Tp}) \cdot V_{DSATp}}{k_n(V_{DD} - V_{Tn})} \approx \frac{\mu_p \cdot W_p}{\mu_n \cdot W_n} \cdot V_{DSATp} \tag{6.28}$$

In order to make V_{OL} as small as possible, the PMOS device should be sized much smaller than the NMOS pull-down devices. Unfortunately, this has a negative impact on the *propagation delay* for charging up the output node since the current provided by the PMOS device is limited.

A major disadvantage of the pseudo-NMOS gate is the static power that is dissipated when the output is low through the direct current path that exists between V_{DD} and GND. The static power consumption in the low-output mode is easily derived:

$$P_{low} = V_{DD} I_{low} \approx V_{DD} \cdot \left| k_p \left((-V_{DD} - V_{Tp}) \cdot V_{DSATp} - \frac{V_{DSATp}^2}{2} \right) \right| \tag{6.29}$$

Example 6.7 Pseudo-NMOS Inverter

Consider a simple pseudo-NMOS inverter (where the PDN network in Figure 6-27 degenerates to a single transistor) with an NMOS size of 0.5 μm/0.25 μm. In this example, we study the effect of sizing the PMOS device to demonstrate the impact on various parameters. The *W–L* ratio of the grounded PMOS is varied over values from 4, 2, 1, 0.5 to 0.25. Devices with a $W–L < 1$ are constructed by making the length greater than the width. The voltage transfer curve for the different sizes is plotted in Figure 6-28.

Table 6-9 summarizes the nominal output voltage (V_{OL}), static power dissipation, and the low-to-high propagation delay. The low-to-high delay is measured as the time it takes to reach 1.25 V from V_{OL} (which is not 0V for this inverter)—by definition. The trade-off between the static and dynamic properties is apparent. A larger pull-up device not only improves performance, but also increases static power dissipation and lowers noise margins by increasing V_{OL}.

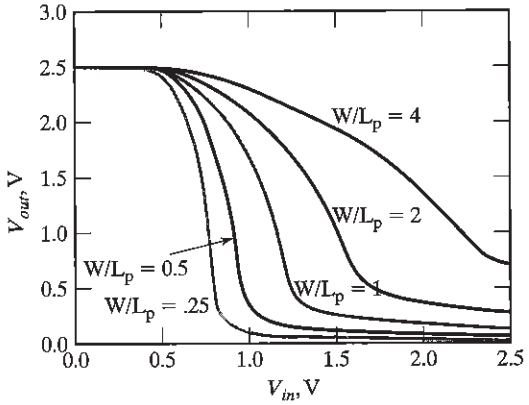


Figure 6-28 Voltage-transfer curves of the pseudo-NMOS inverter as a function of the PMOS size.

Table 6-9 Performance of a pseudo-NMOS inverter.

Size	V_{OL}	Static Power Dissipation	t_{plh}
4	0.693 V	564 μW	14 ps
2	0.273 V	298 μW	56 ps
1	0.133 V	160 μW	123 ps
0.5	0.064 V	80 μW	268 ps
0.25	0.031 V	41 μW	569 ps

Notice that the simple first-order model to predict V_{OL} is quite effective. For a PMOS W–L of 4, V_{OL} is given by (30/115) (4) (0.63V) = 0.66V.

The static power dissipation of pseudo-NMOS limits its use. When area is most important however, its reduced transistor count compared with complementary CMOS is quite attractive. Pseudo-NMOS thus still finds occasional use in large fan-in circuits. Figure 6-29 shows the schematics of pseudo-NMOS NOR and NAND gates.

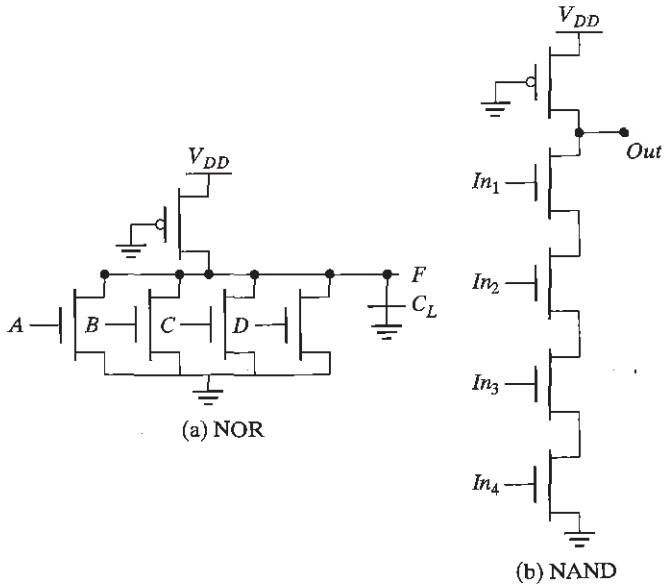


Figure 6-29 Four-input pseudo-NMOS NOR and NAND gates.

Problem 6.5 NAND versus NOR in Pseudo-NMOS

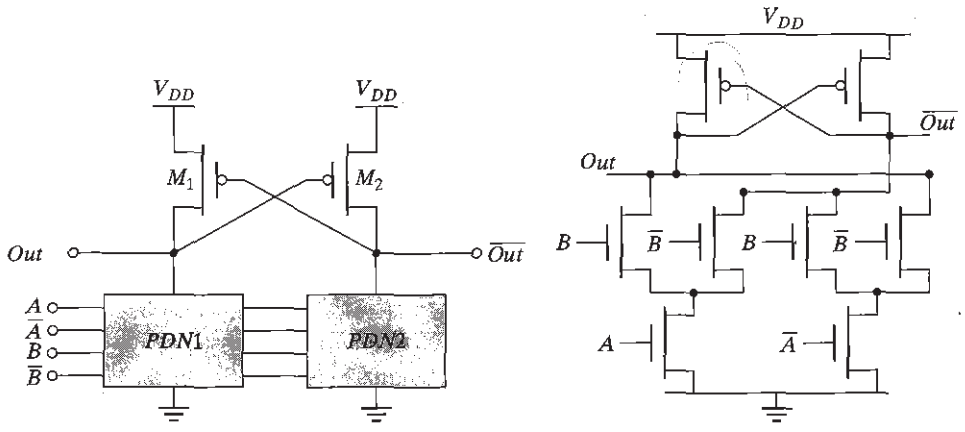
Given the choice between NOR or NAND logic, which one would you prefer for implementation in pseudo-NMOS?

How to Build Even Better Loads

It is possible to create a ratioed logic style that completely eliminates static currents and provides rail-to-rail swing. Such a gate combines two concepts: *differential logic* and *positive feedback*. A differential gate requires that each input is provided in complementary format, and it produces complementary outputs in turn. The feedback mechanism ensures that the load device is turned off when not needed. An example of such a logic family, called *Differential Cascode Voltage Switch Logic* (or DCVSL), is presented conceptually in Figure 6-30a [Heller84].

The pull-down networks PDN1 and PDN2 use NMOS devices and are mutually exclusive—that is, when PDN1 conducts, PDN2 is off, and when PDN1 is off, PDN2 conducts—such that the required logic function and its inverse are simultaneously implemented. Assume now that, for a given set of inputs, PDN1 conducts while PDN2 does not, and that *Out* and $\overline{Out}$ are initially high and low, respectively. Turning *on PDN1*, causes *Out* to be pulled down, although there is still contention between M_1 and PDN1. $\overline{Out}$ is in a high impedance state, as M_2 and PDN2 are both turned *off*. PDN1 must be strong enough to bring *Out* below $V_{DD} - |V_{Tp}|$, the point at which M_2 turns *on* and starts charging $\overline{Out}$ to V_{DD}, eventually turning off M_1. This in turn enables *Out* to discharge all the way to GND. Figure 6-30b shows an example of an XOR–XNOR gate. Notice that it is possible to share transistors among the two pull-down networks, which reduces the implementation overhead.

The resulting circuit exhibits a rail-to-rail swing, and the static power dissipation is eliminated: in steady state, none of the stacked pull-down networks and load devices are



(a) Basic principle (b) XOR–XNOR gate

Figure 6-30 DCVSL logic gate.

simultaneously conducting. However, the circuit is still ratioed since the sizing of the PMOS devices relative to the pull-down devices is critical to functionality, not just performance. In addition to the problem of increased design complexity, this circuit style has a power-dissipation problem that is due to cross-over currents. During the transition, there is a period of time when PMOS and PDN are turned on simultaneously, producing a short circuit path.

Example 6.8 DCVSL Transient Response

An example transient response is shown in Figure 6.31 for an AND/NAND gate in DCVSL. Notice that as *Out* is pulled down to $V_{DD} - |V_{Tp}|$, $\overline{Out}$ starts to charge up to V_{DD} quickly. The delay from the input to *Out* is 197 ps and to $\overline{Out}$ is 321 ps. A static CMOS AND gate (NAND followed by an inverter) has a delay of 200 ps.

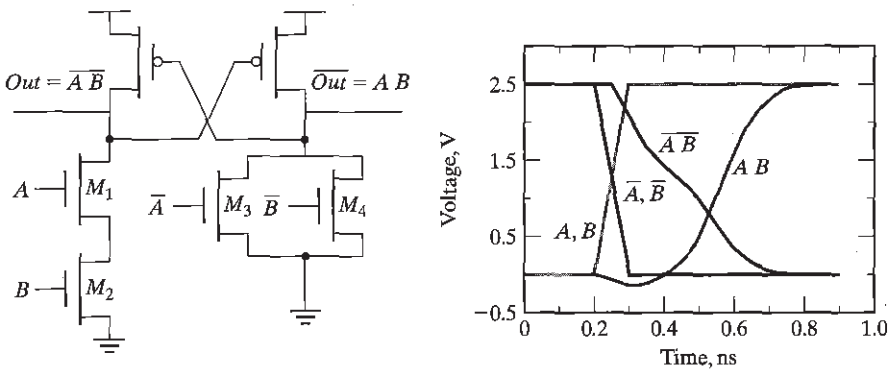
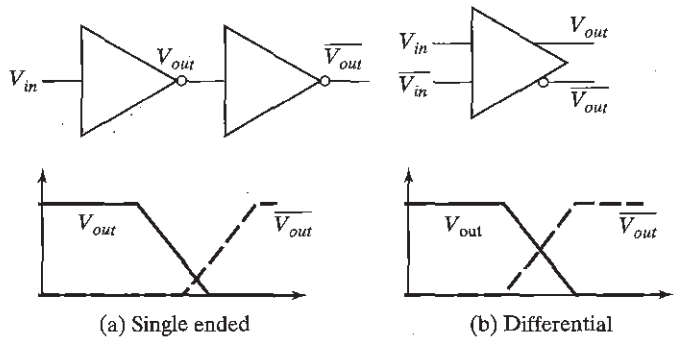


Figure 6-31 Transient response of a simple AND/NAND DCVSL gate. M_1 and M_2 1 μm/0.25 μm, M_3 and M_4 are 0.5 μm/0.25 μm and the cross-coupled PMOS devices are 1.5 μm/0.25 μm.

Design Consideration—Single-Ended versus Differential

The DCVSL gate provides *differential* (or *complementary*) *outputs*. Both the output signal (V_{out}) and its inverted value ($\overline{V}_{out}$) are simultaneously available. This is a distinct advantage, because it eliminates the need for an extra inverter to produce the complementary signal. It has been observed that a differential implementation of a complex function may reduce the number of gates required by a factor of two! The number of gates in the critical timing path is often reduced as well. Finally, the approach prevents some of the time-differential problems introduced by additional inverters. For example, in logic design, it often happens that both a signal and its complement are needed simultaneously. When the complementary signal is generated using an inverter, the inverted signal is delayed with respect to the original (Figure 6-32a). This causes timing problems, especially in very high-speed designs. Logic families with differential output capability avoid this problem to a major extent, if not completely (Figure 6-32b).

With all these positive properties, why not always use differential logic? The reason is that the differential nature virtually doubles the number of wires that have to be routed, often leading to unwieldy designs on top of the additional implementation overhead in the individual gates. The dynamic power dissipation also is high.



(a) Single ended (b) Differential

Figure 6-32 Advantage of over single-ended (a) differential (b) gate. ■

6.2.3 Pass-Transistor Logic

Pass-Transistor Basics

A popular and widely used alternative to complementary CMOS is *pass-transistor logic*, which attempts to reduce the number of transistors required to implement logic by allowing the primary inputs to drive gate terminals as well as source–drain terminals [Radhakrishnan85]. This is in contrast to logic families that we have studied so far, which only allow primary inputs to drive the gate terminals of MOSFETS.

Figure 6-33 shows an implementation of the AND function constructed that way, using only NMOS transistors. In this gate, if the B input is high, the top transistor is turned on and copies the input A to the output F. When B is low, the bottom pass-transistor is turned on and passes a 0. The switch driven by $\overline{B}$ seems to be redundant at first glance. Its presence is essential to ensure that the gate is static—a low-impedance path must exist to the supply rails under all circumstances (in this particular case, when B is low).

The promise of this approach is that fewer transistors are required to implement a given function. For example, the implementation of the AND gate in Figure 6-33 requires 4 transistors (including the inverter required to invert B), while a complementary CMOS implementation would require 6 transistors. The reduced number of devices has the additional advantage of lower capacitance.

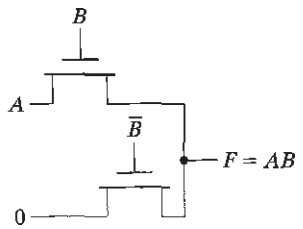


Figure 6-33 Pass-transistor implementation of an AND gate.

Unfortunately, as discussed earlier, an NMOS device is effective at passing a 0, but it is poor at pulling a node to V_{DD}. When the pass-transistor pulls a node high, the output only charges up to $V_{DD} - V_{Tn}$. In fact, the situation is worsened by the fact that the devices experience body effect, because a significant source-to-body voltage is present when pulling high. Consider the case in which the pass-transistor is charging up a node with the gate and drain terminals set at V_{DD}. Let the source of the NMOS pass-transistor be labeled x. The node x will charge up to $V_{DD} - V_{Tn}(V_x)$. We obtain

$$V_x = V_{DD} - (V_{tn0} + \gamma((\sqrt{|2\phi_f| + V_x}) - \sqrt{|2\phi_f|})) \tag{6.30}$$

Example 6.9 Voltage Swing for Pass-Transistors Circuits

The transient response of Figure 6-34 shows an NMOS charging up a capacitor. The drain voltage of the NMOS is at V_{DD}, and its gate voltage is being ramped from 0 V to V_{DD}. Assume that node x is initially at 0 V. We observe that the output initially charges up quickly, but the tail end of the transient is slow. The current drive of the transistor (gate-to-source voltage) is reduced significantly as the output approaches $V_{DD} - V_{Tn}$, and the current available to charge up node x is reduced drastically. Manual calculation using Eq. (6.30) results in an output voltage of 1.8 V, which is close to the simulated value.

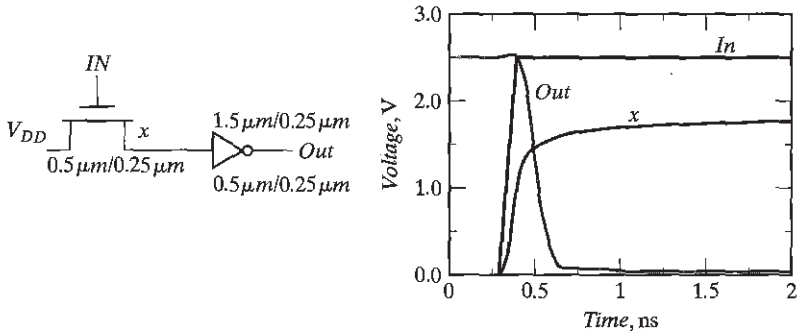


Figure 6-34 Transient response of charging up a node using an *N* device. Notice the slow tail after an initial quick response. $V_{DD} = 2.5$ V.

WARNING: The preceding example demonstrates that **pass-transistor gates cannot be cascaded by connecting the output of a pass gate to the gate input of another pass-transistor**. This is illustrated in Figure 6-35a, where the output of M_1 (node x) drives the gate of another MOS device. Node x can charge up to $V_{DD} - V_{Tn1}$. If node C has a rail-to-rail swing, node Y only charges up to the voltage on node $x - V_{Tn2}$, which works out to $V_{DD} - V_{Tn1} - V_{Tn2}$. Figure 6-35b, on the other hand, has the output of M_1 (x) driving the junction of M_2, and there is only one threshold drop. This is the proper way of cascading pass gates.

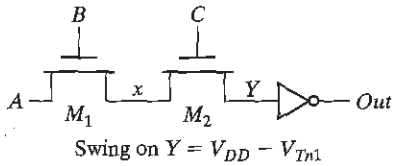


Figure 6-35 Pass-transistor output (drain–source) terminal should not drive other gate terminals to avoid multiple threshold drops.

Example 6.10 VTC of the Pass-Transistor AND Gate

The voltage transfer curve of a pass-transistor gate shows little resemblance to complementary CMOS. Consider the AND gate shown in Figure 6-36. Similar to complementary CMOS, the VTC of pass-transistor logic is data dependent. For the case when $B = V_{DD}$, the top pass-transistor is turned *on*, while the bottom one is turned *off*. In this case, the output just follows the input A until the input is high enough to turn *off* the top pass-transistor (i.e., reaches $V_{DD} - V_{Tn}$). Next, consider the case in which $A = V_{DD}$, and B makes a transition from $0 \rightarrow 1$. Since the inverter has a threshold of $V_{DD}/2$, the bottom pass-transistor is turned *on* until then and the output remains close to zero. Once the bottom pass-transistor turns *off*, the output follows the input B minus a threshold drop. A similar behavior is observed when both inputs A and B transition from $0 \rightarrow 1$.

Observe that a pure pass-transistor gate is not regenerative. A gradual signal degradation will be observed after passing through a number of subsequent stages. This can be remedied by the occasional insertion of a CMOS inverter. With the inclusion of an inverter in the signal path, the VTC resembles one of the CMOS gates.

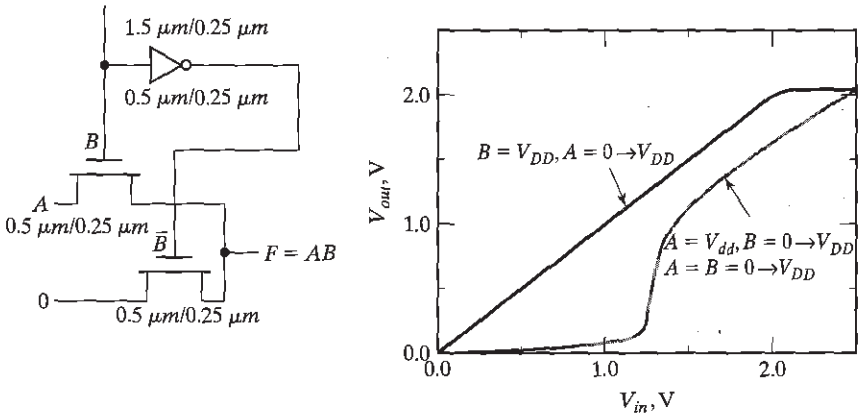


Figure 6-36 Voltage transfer characteristic for the pass-transistor AND gate of Figure 6-33.

Pass-transistors require lower switching energy to charge up a node, due to the reduced voltage swing. For the pass-transistor circuit in Figure 6-34, assume that the drain voltage is at

V_{DD} and the gate voltage transitions to V_{DD}. The output node charges from 0V to $V_{DD} - V_{Tn}$ (assuming that node x was initially at 0V), and the energy drawn from the power supply for charging the output of a pass-transistor is given by

$$E_{0\to 1} = \int_0^T P(t)dt = V_{DD}\int_0^T i_{supply}(t)dt$$
$$= V_{DD}\int_0^{(V_{DD}-V_{Tn})} C_L dV_{out} = C_L \cdot V_{DD} \cdot (V_{DD} - V_{Tn}) \tag{6.31}$$

While the circuit exhibits lower switching power, it may also consume static power when the output is high—the reduced voltage level may be insufficient to turn off the PMOS transistor of the subsequent CMOS inverter.

Differential Pass-Transistor Logic

For high performance design, a differential pass-transistor logic family, called *CPL* or *DPL*, is commonly used. The basic idea (similar to DCVSL) is to accept true and complementary inputs and produce true and complementary outputs. Several CPL gates (AND/NAND, OR/NOR, and XOR/NXOR) are shown in Figure 6-37. These gates possess some interesting properties:

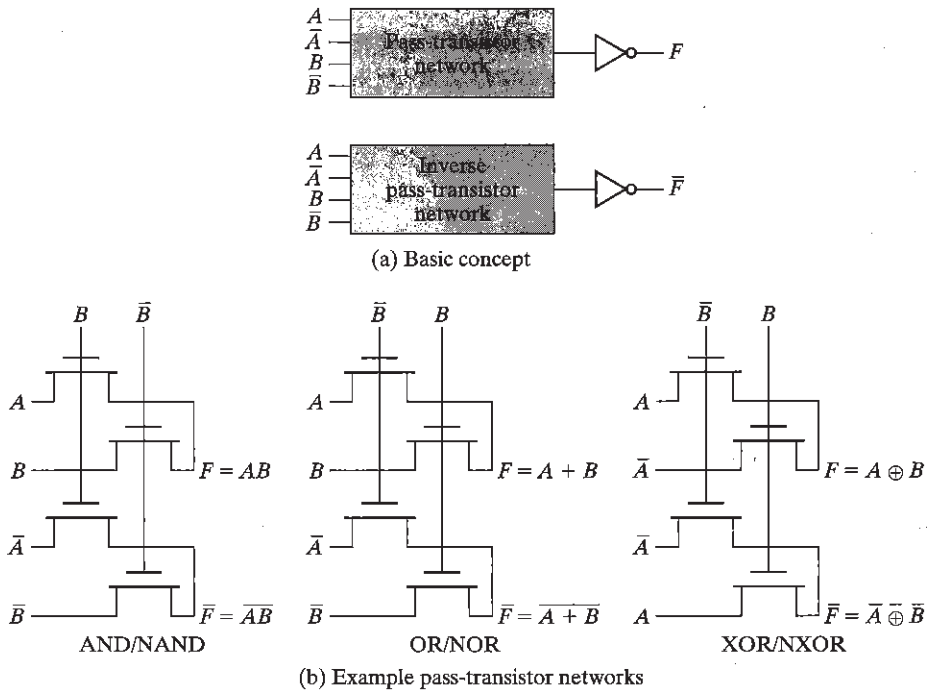


Figure 6-37 Complementary pass-transistor logic (CPL).

- Since the circuits are *differential*, complementary data inputs and outputs are always available. Although generating the differential signals requires extra circuitry, the differential style has the advantage that some complex gates such as XORs and adders can be realized efficiently with a small number of transistors. Furthermore, the availability of both polarities of every signal eliminates the need for extra inverters, as is often the case in static CMOS or pseudo-NMOS.
- CPL belongs to the class of *static* gates, because the output-defining nodes are always connected to either V_{DD} or GND through a low-resistance path. This is advantageous for the noise resilience.
- The design is very modular. In effect, all gates use exactly the same topology. Only the inputs are permutated. This makes the design of a library of gates very simple. More complex gates can be built by cascading the standard pass-transistor modules.

Example 6.11 Four-Input NAND in CPL

Consider the implementation of a four-input AND/NAND gate using CPL. Based on the associativity of the boolean AND operation [$A \cdot B \cdot C \cdot D = (A \cdot B) \cdot (C \cdot D)$], a two-stage approach has been adopted to implement the gate (Figure 6-38). The total number of transistors in the gate (including the final buffer) is 14. This is substantially higher than previously discussed gates.[3] This factor, combined with the complicated routing requirements, makes this circuit style not particularly efficient for this gate. One should, however, be aware of the fact that the structure simultaneously implements the AND and the NAND functions, which might reduce the transistor count of the overall circuit.

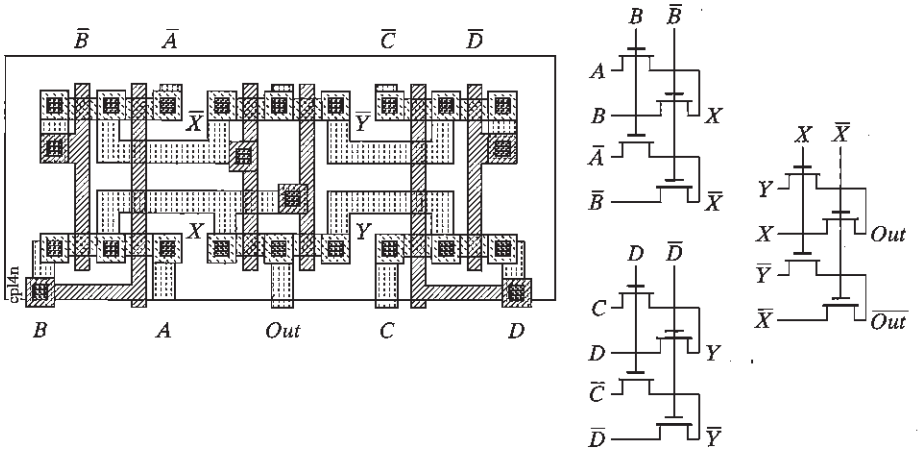


Figure 6-38 Layout and schematics of four-input NAND gate using CPL. The final inverter stage is omitted.

[3]This particular circuit configuration is only acceptable when zero-threshold pass-transistors are used. If not, it directly violates the concepts introduced in Figure 6-35.

In sum, CPL is a conceptually simple and modular logic style. Its applicability depends strongly on the logic function to be implemented. The availability of a simple XOR and the ease of implementing multiplexers makes it attractive for structures such as adders and multipliers. Some extremely fast and efficient implementations have been reported in that application domain [Yano90]. When considering CPL, the designer should not ignore the implicit routing overhead of the complementary signals, which is apparent in the layout of Figure 6-38.

Robust and Efficient Pass-Transistor Design

Unfortunately, differential pass-transistor logic, like single-ended pass-transistor logic, suffers from static power dissipation and reduced noise margins, since the high input to the signal-restoring inverter only charges up to $V_{DD} - V_{Tn}$. There are several solutions proposed to deal with this problem, outlined as follows:

Solution 1: Level Restoration A common solution to the voltage drop problem is the use of a *level restorer*, which is a single PMOS configured in a feedback path (see Figure 6-39). The gate of the PMOS device is connected to the output of the inverter its drain is connected to the input of the inverter and the source is connected to V_{DD}. Assume that node X is at 0V (*out* is at V_{DD} and the M_r is turned *off*) with $B = V_{DD}$ and $A = 0$. If input A makes a 0 to V_{DD} transition, M_n only charges up node X to $V_{DD} - V_{Tn}$. This is, however, enough to switch the output of the inverter low, turning on the feedback device M_r and pulling node X all the way to V_{DD}. This eliminates any static power dissipation in the inverter. Furthermore, no static current path can exist through the level restorer and the pass-transistor, since the restorer is only active when A is high. In sum, this circuit has the advantage that all voltage levels are either at GND or V_{DD}, and no static power is consumed.

While this solution is appealing in terms of eliminating static power dissipation, it adds complexity since the circuit is ratioed. The problem arises during the transition of node X from high to low (seeFigure 6-40). The pass-transistor network attempts to pull down node X, while

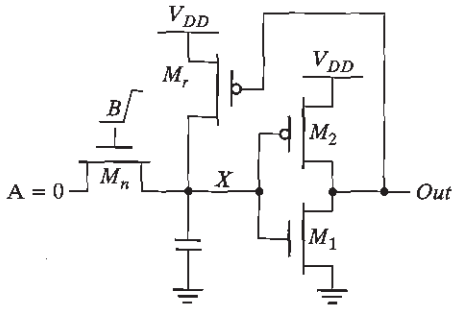


Figure 6-39 Transistor-sizing problem in level-restoring circuits.

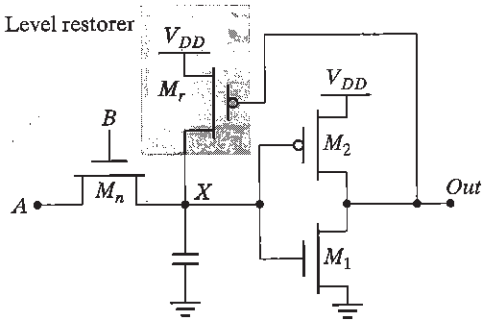


Figure 6-40 Level-restoring circuit.

the level restorer pulls X to V_{DD}. Therefore, the pull-down network, represented by M_n, must be stronger than the pull-up device Mr to switch node X (and the output). Careful transistor sizing is necessary to make the circuit function correctly. Assume the notation R_1 to denote the equivalent on-resistance of transistor M_1, R_2 for M_2, and R_r for M_r. When R_r is too small, it is impossible to bring the voltage at node X below the switching threshold of the inverter. Hence, the inverter output never switches to V_{DD}, and the gate is locked in a single state. The problem can be resolved by sizing transistors M_n and M_r such that the voltage at node X drops below the threshold of the inverter V_M, which is a function of R_1 and R_2. This condition is sufficient to guarantee the switching of the output voltage V_{out} to V_{DD} and the turning off of the level-restoring transistor.

Example 6.12 Sizing of a Level Restorer

Analyzing the circuit as a whole is nontrivial, because the restoring transistor acts as a feedback device. One way to simplify the circuit for manual analysis is to open the feedback loop and to ground the gate of the restoring transistor when determining the switching point (this is a reasonable assumption, as the feedback only becomes active once the inverter starts to switch). Hence, M_r and M_n form a configuration that resembles pseudo-NMOS with M_r the load transistor, and M_n acting as a pull-down network to GND. Assume that the inverter M_1, M_2 is sized to have its switching threshold at $V_{DD}/2$ (NMOS: 0.5 μm/0.25 μm and PMOS: 1.5 μm/0.25 μm). Therefore, node X must be pulled below $V_{DD}/2$ to switch the inverter and to shut off M_r.

This is confirmed in Figure 6-41, which shows the transient response as the size of the level restorer is varied, while keeping the size of M_n fixed (0.5 μm/0.25 μm). As the simulation indicates, for sizes above 1.5 μm/0.25 μm, node X cannot be brought below the switching threshold of the inverter, and can't switch the output.

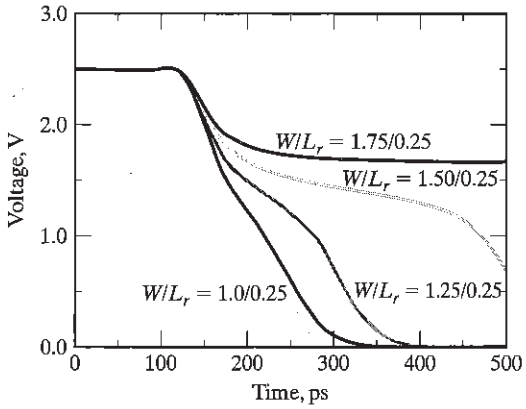


Figure 6-41 Transient response of the circuit in Figure 6-39. A level restorer that is too large results in incorrect evaluation.

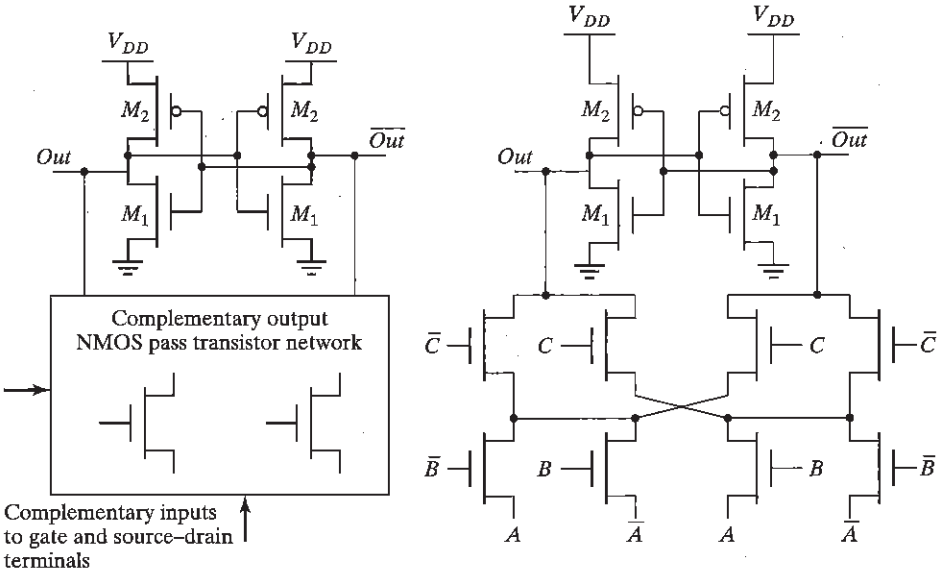
Another concern is the influence of the level restorer on the switching speed of the device. Adding the restoring device increases the capacitance at the internal node X, slowing down the gate. In addiction, the rise time of the gate is affected negatively. The level restoring transistor M_r fights the decrease in voltage at node X before being switched off. On the other hand, the level restorer reduces the fall time, since the PMOS transistor, once turned on, accelerates the pull-up action.

Problem 6.6 Device Sizing in Pass-Transistors

For the circuit shown in Figure 6-39, assume that the pull-down device consists of six pass-transistors in series each with a device size of 0.5 μm/0.25 μm (replacing transistor M_n). Determine the maximum W–L size for the level restorer transistor for correct functionality.

A modification of the level restorer concept is shown in Figure 6-42. It is applicable in differential networks and is known as *swing-restored pass-transistor logic*. Instead of a simple inverter at the output of the pass-transistor network, two back-to-back inverters configured in a cross-coupled fashion are used for level restoration and performance improvement. Inputs are fed to both the gate and source–drain terminals, as in the case of conventional pass-transistor networks. Figure 6-42 shows a simple XOR/XNOR gate of three variables A, B, and C. The complementary network can be optimized by sharing transistors between the true and complementary outputs. This logic family comes with a major caveat: When cascading gates, buffers may have to be included in between the gates. If not, contention between the level-restoring devices of the cascaded gates negatively impacts the performance.

Solution 2: Multiple-Threshold Transistors A technology solution to the voltage-drop problem associated with pass-transistor logic is the use of multiple-threshold devices. Using *zero-*



(a) General concept (b) XOR/XNOR gate

Figure 6-42 Swing-restored pass-transistor logic [Landman91, Parameswar96].

threshold devices for the NMOS pass-transistors eliminates most of the threshold drop, and passes a signal close to V_{DD}. All devices other than the pass-transistors (i.e., the inverters) are implemented using standard high-threshold devices. The use of multiple-threshold transistors is becoming more common, and involves simple modifications to existing process flows. Observe that even if the device implants were carefully calibrated to yield thresholds of exactly zero, the body effect of the device still would prevent a full swing to V_{DD}.

The use of zero-threshold transistors has some negative impact on the power consumption due to the subthreshold currents flowing through the pass-transistors, even if V_{GS} is below V_T. This is demonstrated in Figure 6-43, which points out a potential sneak dc-current path. While these leakage paths are not critical when the device is switching constantly, they do pose a significant energy overhead when the circuit is in the idle state.

Solution 3: Transmission-Gate Logic The most widely used solution to deal with the voltage-drop problem is the use of *transmission gates*.[4] This technique builds on the complementary properties of NMOS and PMOS transistors: NMOS devices pass a strong 0, but a weak 1, while PMOS transistors pass a strong 1 but a weak 0. The ideal approach is to use an NMOS to pull down and a PMOS to pull up. The transmission gate combines the best of both device flavors by placing an NMOS device in parallel with a PMOS device as in Figure 6-44a. The control

[4]The transmission gate is only one of the possible solutions. Other styles of pass-transistor networks that combine NMOS and PMOS transistors have been devised. Double pass-transistor logic (DPL) is an example of such [Bernstein98, pp. 84].

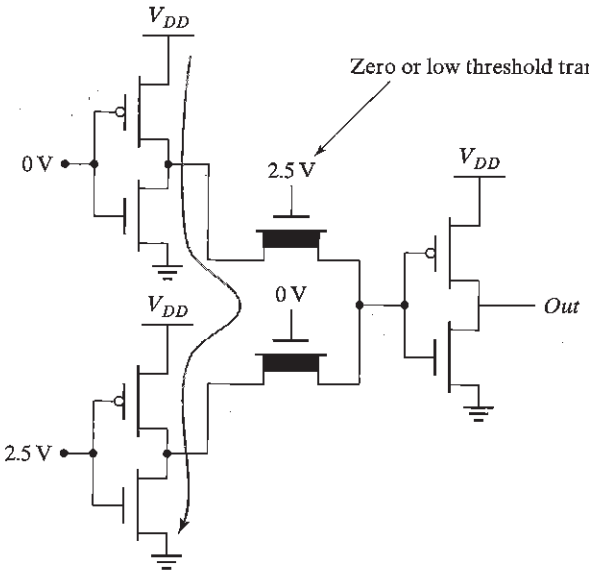
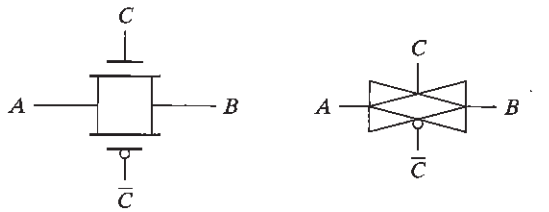


Figure 6-43 Static power consumption when using zero-threshold pass-transistors.



(a) Circuit (b) Symbolic representation

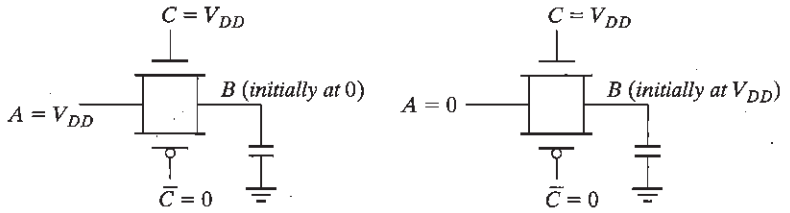
Figure 6-44 CMOS transmission gate.

signals to the transmission gate (C and $\overline{C}$) are complementary. The transmission gate acts as a bidirectional switch controlled by the gate signal C. When $C = 1$, both MOSFETs are on, allowing the signal to pass through the gate. In short,

$$A = B \quad \text{if} \quad C = 1 \tag{6.32}$$

On the other hand, $C = 0$ places both transistors in cutoff, creating an open circuit between nodes A and B. Figure 6-44b shows a commonly used transmission-gate symbol.

Consider the case of charging node B to V_{DD} for the transmission-gate circuit in Figure 6-45a. Node A is set at V_{DD}, and the transmission gate is enabled ($C = 1$ and $\overline{C} = 0$). If only the NMOS pass device were present, node B would only charge up to $V_{DD} - V_{Tn}$, at which point the NMOS device would turn off. However, since the PMOS device is present



(a) Charging node B (b) Discharging node B

Figure 6-45 Transmission gates enable rail-to-rail switching.

and is "on" ($V_{GSp} = -V_{DD}$), the output charges all the way up to V_{DD}. Figure 6-45b shows the opposite case—that is, discharging node B to 0. B is initially at V_{DD} when node A is driven low. The PMOS transistor by itself can only pull-down node B to V_{Tp} at which point it turns *off*. The parallel NMOS device stays turned on, however (since its $V_{GSn} = V_{DD}$), and pulls node B all the way to GND. Although the transmission gate requires two transistors and more control signals, it enables rail-to-rail swing.

Transmission gates can be used to build some complex gates very efficiently. Figure 6-46 shows an example of a simple inverting two-input multiplexer. This gate either selects input A or B on the basis of the value of the control signal S, which is equivalent to implementing the following Boolean function:

$$\bar{F} = (A \cdot S + B \cdot \bar{S}) \tag{6.33}$$

A complementary implementation of the gate requires eight transistors instead of six.

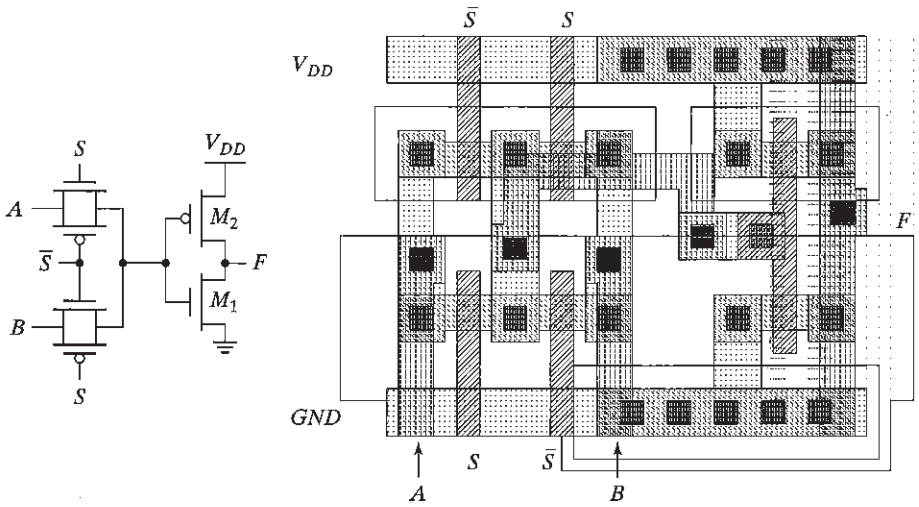


Figure 6-46 Transmission-gate multiplexer and its layout.

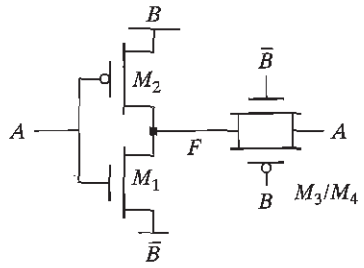


Figure 6-47 Transmission-gate XOR.

Another example of the effective use of transmission gates is the popular XOR circuit shown in Figure 6-47. The complete implementation of this gate requires only 6 transistors (including the inverter used for the generation of $\bar{B}$), compared with the 12 transistors required for a complementary implementation. To understand the operation of this circuit, we need only analyze the $B = 0$ and $B = 1$ cases separately. For $B = 1$, transistors M_1 and M_2 act as an inverter, while the transmission gate M_3/M_4 is off; hence, $F = \bar{A}B$. In the opposite case, M_1 and M_2 are disabled, and the transmission gate is operational, or $F = A\bar{B}$. The combination of both leads to the XOR function. Notice that regardless of the values of A and B, node F always has a connection to either V_{DD} or GND and thus is a low-impedance node. When designing static-pass-transistor networks, it is essential to adhere to the low-impedance rule under all circumstances. Other examples in which transmission-gate logic is effectively used are fast adder circuits and registers.

Performance of Pass-Transistor and Transmission-Gate Logic

The pass-transistor and the transmission gate are, unfortunately, not ideal switches, and they have a series resistance associated with them. To quantify the resistance, consider the circuit in Figure 6-48, which involves charging a node from 0 V to V_{DD}. In this discussion, we use the large-signal definition of resistance, which involves dividing the voltage across the switch by the drain current. The effective resistance of the switch is modeled as a parallel connection of the resistances R_n and R_p of the NMOS and PMOS devices, defined as $(V_{DD} - V_{out})/I_{Dn}$ and $(V_{DD} - V_{out})/(-I_{Dp})$, respectively. The currents through the devices obviously are dependent on the value of V_{out} and the operating mode of the transistors. During the low-to-high transition, the pass-transistors traverse through a number of operation modes. For low values of V_{out}, the NMOS device is saturated and the resistance is approximated as

$$R_p = \frac{V_{out} - V_{DD}}{I_{Dp}} = \frac{V_{out} - V_{DD}}{k_p \cdot \left((-V_{DD} - V_{Tp})(V_{out} - V_{DD}) - \frac{(V_{out} - V_{DD})^2}{2}\right)} \approx \frac{1}{k_p(-V_{DD} - V_{Tp})} \tag{6.34}$$

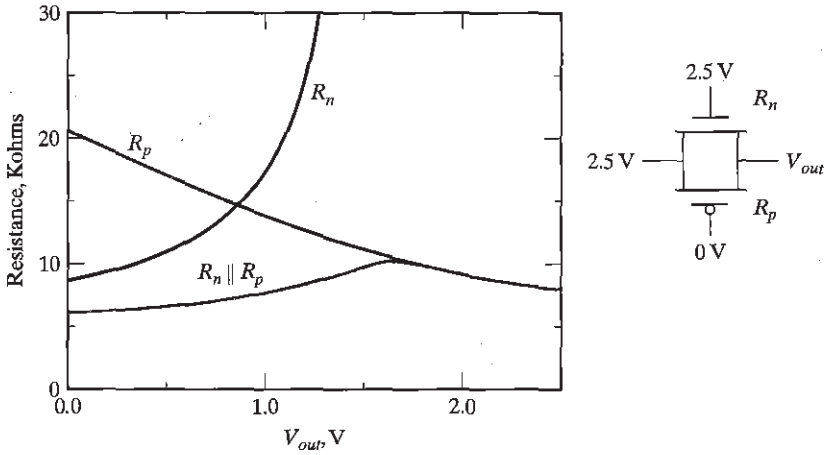


Figure 6-48 Simulated equivalent resistance of transmission gate for low-to-high transition (for $(W\text{–}L)_n = (W\text{–}L)_p = 0.5\ \mu m/0.25\ \mu m$). A similar response for overall resistance is obtained for the high-to-low transition.

The resistance goes up for increasing values of V_{out} and approaches infinity when V_{out} reaches $V_{DD} - V_{Tn}$ and the device shuts off. Similarly, we can analyze the behavior of the PMOS transistor. When V_{out} is small, the PMOS is saturated, but it enters the linear mode of operation for V_{out} approaching V_{DD}. This gives the following approximated resistance:

$$R_p = \frac{V_{out} - V_{DD}}{I_{Dp}} = \frac{V_{out} - V_{DD}}{k_p \cdot \left((-V_{DD} - V_{Tp})(V_{out} - V_{DD}) - \frac{(V_{out} - V_{DD})^2}{2} \right)} \approx \frac{1}{k_p(-V_{DD} - V_{Tp})} \tag{6.35}$$

The simulated value of $R_{eq} = R_p \parallel R_n$ as a function of V_{out} is plotted in Figure 6-48. It can be observed that R_{eq} is relatively constant ($\approx 8\ k\Omega$ in this particular case). The same is true in other design instances (for example, when discharging C_L). When analyzing transmission-gate networks, the simplifying assumption that the switch has a constant resistive value is therefore acceptable.

Problem 6.7 Equivalent Resistance during Discharge

Determine the equivalent resistance by simulation for the high-to-low transition of a transmission gate. (In other words, produce a plot similar to the one presented in Figure 6-48).

An important consideration is the delay associated with a chain of transmission gates. Figure 6-49 shows a chain of n transmission gates. Such a configuration often occurs in circuits such as adders or deep multiplexors. Assume that all transmission gates are turned on and a step

Obviously, the number of switches per segment grows with increasing values of t_{buf}. In current technologies, m_{opt} typically equals 3 or 4. The presented analysis ignores that tp_{buf} itself is a function of the load m. A more accurate analysis taking this factor into account is presented in Chapter 9.

Example 6.14 Transmission-Gate Chain

Consider the same 16-transmission-gate chain. The buffers shown in Figure 6-51 can be implemented as inverters (instead of two cascaded inverters). In some cases, it might be necessary to add an extra inverter to produce the correct polarity. Assuming that each inverter is sized such that the NMOS is 0.5 μm/0.25 μm and PMOS is 0.5 μm /0.25 μm, Eq. (6.39) predicts that an inverter must be inserted every 3 transmission gates. The simulated delay when placing an inverter every two transmission gates is 154 ps; for every three transmission gates, the delay is 154 ps; and for four transmission gates, it is 164 ps. The insertion of buffering inverters reduces the delay by a factor of almost 2.

CAUTION: Although many of the circuit styles discussed in the previous sections sound very interesting, and might be superior to static CMOS in many respects, none has the *robustness and ease of design* of complementary CMOS. Therefore, use them sparingly and with caution. For designs that have no extreme area, complexity, or speed constraints, complementary CMOS is the recommended design style.

6.3 Dynamic CMOS Design

It was noted earlier that static CMOS logic with a fan-in of N requires $2N$ devices. A variety of approaches were presented to reduce the number of transistors required to implement a given logic function including pseudo-NMOS, pass-transistor logic, etc. The pseudo-NMOS logic style requires only $N + 1$ transistors to implement an N input logic gate, but unfortunately it has static power dissipation. In this section, an alternate logic style called *dynamic logic* is presented that obtains a similar result, while avoiding static power consumption. With the addition of a clock input, it uses a sequence of *precharge* and conditional *evaluation* phases.

6.3.1 Dynamic Logic: Basic Principles

The basic construction of an (n-type) dynamic logic gate is shown in Figure 6-52a. The PDN (pull-down network) is constructed exactly as in complementary CMOS. The operation of this circuit is divided into two major phases—*precharge* and *evaluation*—with the mode of operation determined by the *clock signal CLK*.

Precharge

When $CLK = 0$, the output node *Out* is precharged to V_{DD} by the PMOS transistor M_p. During that time, the evaluate NMOS transistor M_e is off, so that the pull-down path is disabled. The

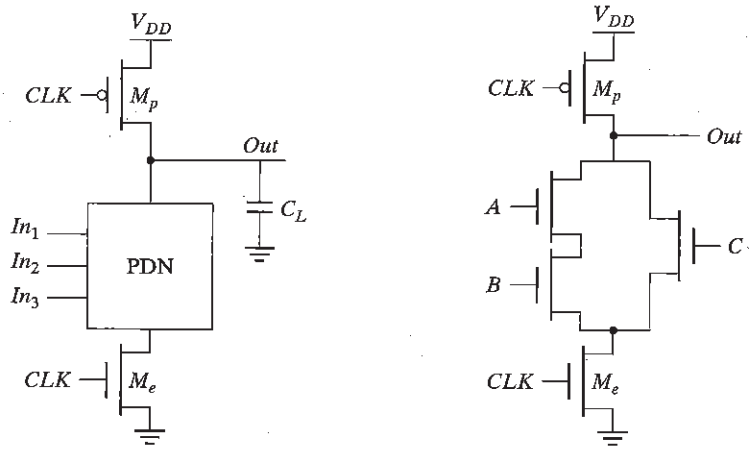


Figure 6-52 Basic concepts of a dynamic gate.

evaluation FET eliminates any static power that would be consumed during the precharge period (i.e., static current would flow between the supplies if both the pull-down and the precharge device were turned on simultaneously).

Evaluation

For $CLK = 1$, the precharge transistor M_p is off, and the evaluation transistor M_e is turned on. The output is conditionally discharged based on the input values and the pull-down topology. If the inputs are such that the PDN conducts, then a low resistance path exists between *Out* and GND, and the output is discharged to GND. If the PDN is turned off, the precharged value remains stored on the output capacitance C_L, which is a combination of junction capacitances, the wiring capacitance, and the input capacitance of the fan-out gates. During the evaluation phase, the only possible path between the output node and a supply rail is to GND. Consequently, once *Out* is discharged, it cannot be charged again until the next precharge operation. **The inputs to the gate can thus make at most one transition during evaluation.** Notice that the output can be in the *high-impedance state* during the evaluation period if the pull-down network is turned off. This behavior is fundamentally different from the static counterpart that always has a low resistance path between the output and one of the power rails.

As an example, consider the circuit shown in Figure 6-52b. During the precharge phase ($CLK = 0$), the output is precharged to V_{DD} regardless of the input values, because the evaluation device is turned off. During evaluation ($CLK = 1$), a conducting path is created between *Out* and GND if (and only if) $A \cdot B + C$ is TRUE. Otherwise, the output remains at the precharged state of V_{DD}. The following function is thus realized:

$$Out = \overline{CLK} + \overline{(A \cdot B + C)} \cdot CLK \tag{6.40}$$

A number of important properties can be derived for the dynamic logic gate:

- The logic function is implemented by the NMOS pull-down network. The construction of the PDN proceeds just as it does for static CMOS.
- The *number of transistors* (for complex gates) is substantially lower than in the static case: $N + 2$ versus $2N$.
- It is *nonratioed*. The sizing of the PMOS precharge device is not important for realizing proper functionality of the gate. The size of the precharge device can be made large to improve the low-to-high transition time (of course, at a cost to the high-to-low transition time). There is, however, a trade-off with power dissipation, since a larger precharge device directly increases clock-power dissipation.
- It only consumes *dynamic power*. Ideally, no static current path ever exists between V_{DD} and GND. The overall power dissipation, however, can be significantly higher compared with a static logic gate.
- The logic gates have *faster switching speeds*, for two main reasons. The first (obvious) reason is due to the reduced load capacitance attributed to the lower number of transistors per gate and the single-transistor load per *fan-in*. This translates in a *reduced logical effort*. For instance, the logical effort of a two-input dynamic NOR gate equals 2/3, which is substantially smaller than the 5/3 of its static CMOS counterpart. The second reason is that the dynamic gate does not have short circuit current, and all the current provided by the pull-down devices goes towards discharging the load capacitance.

The low and high output levels of V_{OL} and V_{OH} are easily identified as GND and V_{DD}, and they are not dependent on the transistor sizes. The other VTC parameters are dramatically different from static gates. Noise margins and switching thresholds have been defined as static quantities that are not a function of time. To be functional, a dynamic gate requires a periodic sequence of precharges and evaluations. Pure static analysis, therefore, does not apply. During the evaluation period, the pull-down network of a dynamic inverter starts to conduct when the input signal exceeds the threshold voltage (V_{Tn}) of the NMOS pull-down transistor. Therefore, it is reasonable to assume that the switching threshold (V_M) as well as V_{IH} and V_{IL} are equal to V_{Tn}. This translates to a low value for the NM_L.

Design Consideration

It is also possible to implement dynamic logic using the dual approach, where the output node is connected by a predischarge NMOS transistor to GND, and the evaluation PUN network is implemented in PMOS. The operation is similar: During precharge, the output node is discharged to GND; during evaluation, the output is conditionally charged to V_{DD}. This *p*-type dynamic gate has the disadvantage of being slower than the *n*-type because of the lower current drive of the PMOS transistors. ■

6.3.2 Speed and Power Dissipation of Dynamic Logic

The main advantages of dynamic logic are increased speed and reduced implementation area. Fewer devices to implement a given logic function implies that the overall load capacitance is much smaller. The analysis of the switching behavior of the gate has some interesting peculiarities to it. After the precharge phase, the output is high. For a low input signal, no additional switching occurs. As a result, $t_{pLH} = 0$! The high-to-low transition, on the other hand, requires the discharging of the output capacitance through the pull-down network. Therefore, t_{pHL} is proportional to C_L and the current-sinking capabilities of the pull-down network. The presence of the evaluation transistor slows the gate somewhat, as it presents an extra series resistance. Omitting this transistor, while functionally not forbidden, may result in static power dissipation and potentially a performance loss.

The preceding analysis is somewhat unfair because it ignores the influence of the precharge time on the switching speed of the gate. The precharge time is determined by the time it takes to charge C_L through the PMOS precharge transistor. During this time, the logic in the gate cannot be utilized. Very often, however, the overall digital system can be designed in such a way that the precharge time coincides with other system functions. For instance, the precharge of the arithmetic unit in a microprocessor could coincide with the instruction decode. The designer has to be aware of this "dead zone" in the use of dynamic logic and thus should carefully consider the pros and cons of its usage, taking the overall system requirements into account.

Example 6.15 A Four-Input Dynamic NAND Gate

Figure 6-53 shows the design of a four-input NAND example designed using the dynamic-circuit style. Due to the dynamic nature of the gate, the derivation of the voltage-transfer

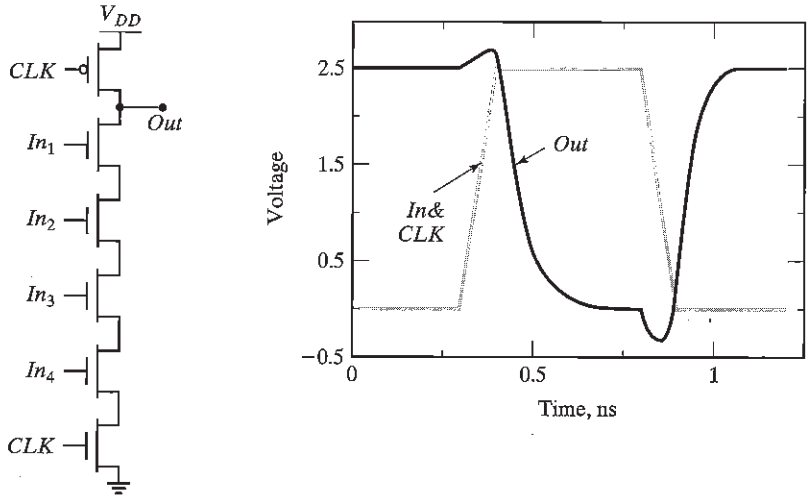


Figure 6-53 Schematic and transient response of a four-input dynamic NAND gate.

characteristic diverges from the traditional approach. As discussed earlier, we assume that the switching threshold of the gate equals the threshold of the NMOS pull-down transistor. This results in asymmetrical noise margins, as shown in Table 6-10.

Table 6-10 The dc and ac parameters of a four-input dynamic NAND.

Transistors	V_{OH}	V_{OL}	V_M	NM_H	NM_L	t_{pHL}	t_{pLH}	t_{pre}
6	2.5 V	0 V	V_{TN}	$2.5 - V_{TN}$	V_{TN}	110 ps	0 ps	83 ps

The dynamic behavior of the gate is simulated with SPICE. It is assumed that all inputs are set high when the clock goes high. On the rising edge of the clock, the output node is discharged. The resulting transient response is plotted in Figure 6-53, and the propagation delays are summarized in Table 6-10. The duration of the precharge cycle can be adjusted by changing the size of the PMOS precharge transistor. Making the PMOS too large should be avoided, however, as it both slows down the gate and increases the capacitive load on the clock line. For large designs, the latter factor might become a major design concern because the clock load can become excessive and hard to drive.

As mentioned earlier, the static gate parameters are time dependent. To illustrate this, consider a four-input NAND gate with all the partial inputs tied together, and are making a low-to-high transition. Figure 6-54 shows a transient simulation of the output voltage for three different input transitions—from 0 to 0.45 V, 0.5 V and 0.55 V, respectively. In the preceding discussion, we have defined the switching threshold of the dynamic gate as the device threshold. However, notice that the amount by which the output voltage drops is a strong function of the input voltage and the *available evaluation time*. The noise voltage needed to corrupt the signal has to be larger if the evaluation time is short. In other words, the switching threshold is truly time dependent.

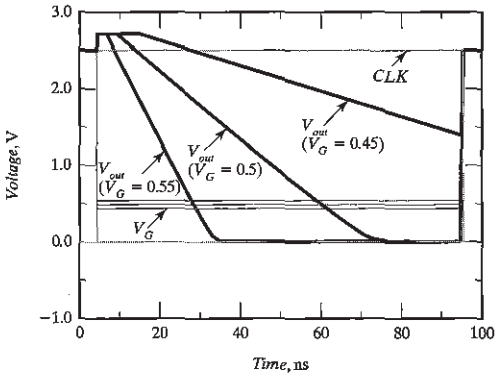


Figure 6-54 Effect of an input glitch on the output. The switching threshold depends on the time for evaluation. A larger glitch is acceptable if the evaluation phase is shorter.

It would appear that dynamic logic presents a significant advantage from a power perspective. There are three reasons for this. First, the physical capacitance is lower since dynamic logic uses fewer transistors to implement a given function. Also, the load seen for each fan-out is one transistor instead of two. Second, dynamic logic gates *by construction* can have at most one transition per clock cycle. Glitching (or dynamic hazards) does not occur in dynamic logic. Finally, dynamic gates do not exhibit short-circuit power since the pull-up path is not turned on when the gate is evaluating.

While these arguments generally are true, they are offset by other considerations: (1) the clock power of dynamic logic can be significant, particularly since the clock node has a guaranteed transition on every single clock cycle; (2) the number of transistors is greater than the minimal set required for implementing the logic; (3) short-circuit power may exist when leakage-combatting devices are added (as will be discussed further); and (4), most importantly, dynamic logic generally displays a higher switching activity due to the periodic *precharge* and *discharge* operations. Earlier, the transition probability for a static gate was shown to be $p_0\, p_1 = p_0\,(1 - p_0)$. For dynamic logic, the output transition probability does not depend on the state (history) of the inputs, but rather on the signal probabilities. For an n-tree dynamic gate, the output makes a $0 \rightarrow 1$ transition during the precharge phase only if the output was discharged during the preceding evaluate phase. Hence, the $0 \rightarrow 1$ transition probability for an n-type dynamic gate is given by

$$a_{0\rightarrow 1} = p_0 \tag{6.41}$$

where p_0 is the probability that the output is zero. This number is always greater than or equal to $p_0\, p_1$. For uniformly distributed inputs, the transition probability for an N-input gate is

$$a_{0\rightarrow 1} = \frac{N_0}{2^N} \tag{6.42}$$

where N_0 is the number of zero entries in the truth table of the logic function.

Example 6.16 Activity Estimation in Dynamic Logic

To illustrate the increased activity for a dynamic gate, consider again a two-input NOR gate. An n-tree dynamic implementation is shown in Figure 6-55, along with its static counterpart. For equally probable inputs, there is a 75% probability that the output node of the dynamic gate discharges immediately after the precharge phase; implying that the activity for such a gate equals 0.75 (i.e., $P_{NOR} = 0.75\, C_L V_{dd}^2 f_{clk}$). The corresponding activity is a lot smaller, 3/16, for a static implementation. For a dynamic NAND gate, the transition probability is 1/4 (since there is a 25% probability the output will be discharged) while it is 3/16 for a static implementation. Although these examples illustrate that the switching activity of dynamic logic is generally higher, it should be noted that dynamic logic has lower physical capacitance. Both factors must be accounted for when analyzing dynamic power dissipation.

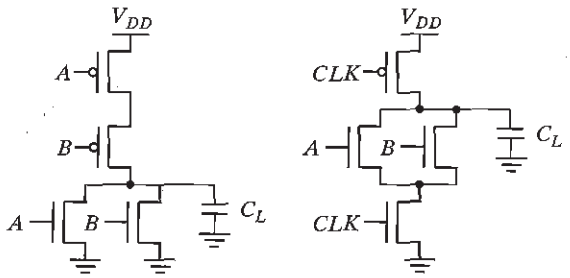


Figure 6-55 Static NOR versus *n*-type dynamic NOR.

Problem 6.8 Activity Computation

For the four-input dynamic NAND gate, compute the activity factor with the following assumption for the inputs: They are independent, and $p_{A=1} = 0.2, p_{B=1} = 0.3, p_{C=1} = 0.5$, and $p_{D=1} = 0.4$.

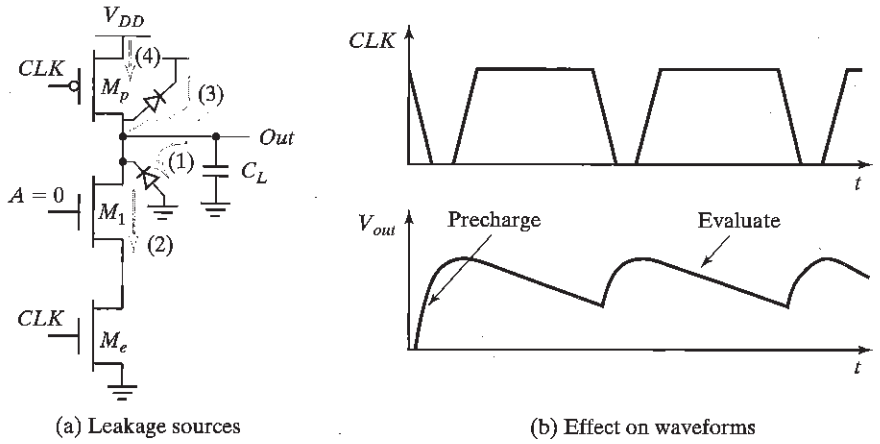
6.3.3 Signal Integrity Issues in Dynamic Design

Dynamic logic clearly can result in high-performance solutions compared to static circuits. However, there are several important considerations that must be taken into account if one wants dynamic circuits to function properly. These include charge leakage, charge sharing, capacitive coupling, and clock feedthrough. These issues are discussed in some detail in this section.

Charge Leakage

The operation of a dynamic gate relies on the dynamic storage of the output value on a capacitor. If the pull-down network is *off*, ideally, the output should remain at the precharged state of V_{DD} during the evaluation phase. However, this charge gradually leaks away due to leakage currents, eventually resulting in a malfunctioning of the gate. Figure 6-56a shows the sources of leakage for the basic dynamic inverter circuit.

Source 1 and 2 are the *reverse-biased diode* and *subthreshold leakage* of the NMOS pull-down device M_1, respectively. The charge stored on C_L will slowly leak away through these leakage channels, causing a degradation in the high level (Figure 6-56b). Dynamic circuits therefore require a minimal clock rate, which is typically on the order of a few kHz. This makes the usage of dynamic techniques unattractive for low-performance products such as watches, or processors that use conditional clocks (where there are no guarantees on minimum clock rates). Note that the PMOS precharge device also contributes some leakage current due to the reverse bias diode (source 3) and the subthreshold conduction (source 4). To some extent, the leakage current of the PMOS counteracts the leakage of the pull-down path. As a result, the output voltage is going to be set by the resistive divider composed of the pull-down and pull-up paths.



(a) Leakage sources (b) Effect on waveforms

Figure 6-56 Leakag e issues in dynamic circuits.

Example 6.17 Leakage in Dynamic Circuits

Consider the simple inverter with all devices set at 0.5 μm/0.25 μm. Assume that the input is low during the evaluation period. Ideally, the output should remain at the precharged state of V_{DD}. However, as seen from Figure 6-57, the output voltage drops. Once the output drops below the switching threshold of the fan-out logic gate, the output is interpreted as a low voltage. Notice that the output settles to an intermediate voltage, due to the leakage current provided by the PMOS pull-up.

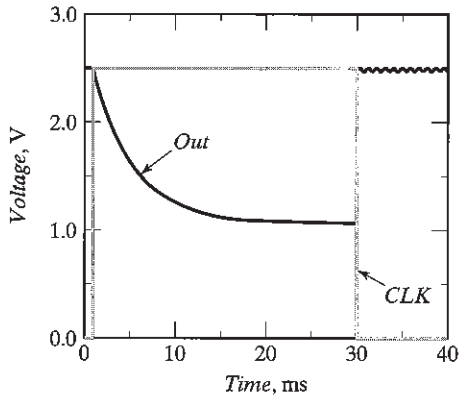


Figure 6-57 Impact of charge leakage. The output settles to an intermediate voltage determined by a resistive divider of the pull-down and pull-up devices.

Leakage is caused by the high-impedance state of the output node during the evaluate mode, when the pull-down path is turned off. The leakage problem may be counteracted by reducing the output impedance on the output node during evaluation. This often is done by

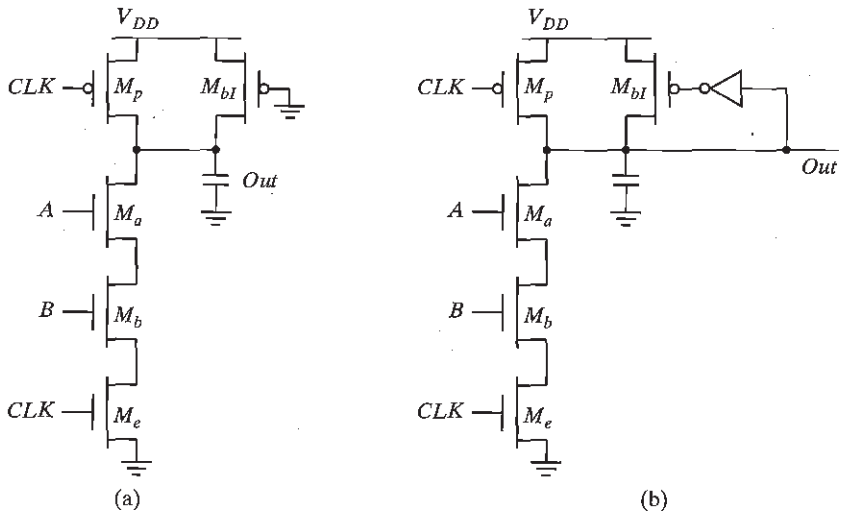


Figure 6-58 Static bleeders compensate for the charge leakage.

adding a *bleeder transistor*, as shown in Figure 6-58a. The only function of the bleeder—an NMOS style pull-up device—is to compensate for the charge lost due to the pull-down leakage paths. To avoid the ratio problems associated with this style of circuit and the associated static power consumption, the bleeder resistance is made high (in other words, the device is kept small). This allows the (strong) pull-down devices to lower the *Out* node substantially below the switching threshold of the next gate. Often, the bleeder is implemented in a feedback configuration to eliminate the static power dissipation altogether (Figure 6-58b).

Charge Sharing

Another important concern in dynamic logic is the impact of charge sharing. Consider the circuit in Figure 6-59. During the precharge phase, the output node is precharged to V_{DD}. Assume that all inputs are set to 0 during precharge, and that the capacitance C_a is discharged. Assume further that input B remains at 0 during evaluation, while input A makes a $0 \rightarrow 1$ transition, turning transistor M_a on. The charge stored originally on capacitor C_L is redistributed over C_L and C_a. This causes a drop in the output voltage, which cannot be recovered due to the dynamic nature of the circuit.

The influence on the output voltage is readily calculated. Under the assumptions given previously, the following initial conditions are valid: $V_{out}(t = 0) = V_{DD}$ and $V_X(t = 0) = 0$. As a result, two possible scenarios must be considered:

1. $\Delta V_{out} < V_{Tn}$. In this case, the final value of V_X equals $V_{DD} - V_{Tn}(V_X)$. Charge conservation then yields

$$C_L V_{DD} = C_L V_{out}(\text{final}) + C_a[V_{DD} - V_{Tn}(V_X)]$$

or

$$\Delta V_{out} = V_{out}(\text{final}) + (-V_{DD}) = -\frac{C_a}{C_L}[V_{DD} - V_{Tn}(V_X)] \tag{6.43}$$

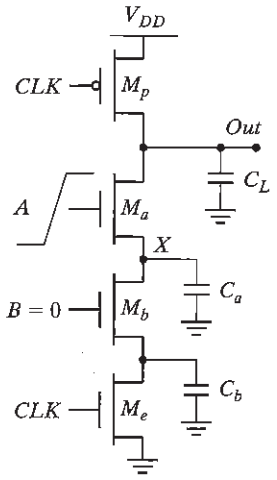


Figure 6-59 Charge sharing in dynamic networks.

2. $\Delta V_{out} > V_{Tn}$. V_{out} and V_X then reach the same value:

$$\Delta V_{out} = -V_{DD}\left(\frac{C_a}{C_a + C_L}\right) \tag{6.44}$$

We determine which of these scenarios is valid by the capacitance ratio. The boundary condition between the two cases can be determined by setting ΔV_{out} equal to V_{Tn} in Eq. (6.44), yielding

$$\frac{C_a}{C_L} = \frac{V_{Tn}}{V_{DD} - V_{Tn}} \tag{6.45}$$

Case 1 holds when the (C_a/C_L) ratio is smaller than the condition defined in Eq. (6.45). If not, Eq. (6.44) is valid. Overall, it is desirable to keep the value of ΔV_{out} below $|V_{Tp}|$. The output of the dynamic gate might be connected to a static inverter, in which case the low level of V_{out} would cause static power consumption. One major concern is a circuit malfunction if the output voltage is brought below the switching threshold of the gate it drives.

Example 6.18 Charge Sharing

Let us consider the impact of charge sharing on the dynamic logic gate shown in Figure 6-60, which implements a three-input EXOR function $y = A \oplus B \oplus C$. The first question to be resolved is what conditions cause the worst case voltage drop on node y. For simplicity, ignore the load inverter, and assume that all inputs are low during the precharge operation and that all isolated internal nodes (V_a, V_b, V_c, and V_d) are initially at 0 V.

Inspection of the truth table for this particular logic function shows that the output stays high for 4 out of 8 cases. The worst case change in output is obtained by exposing the maximum amount of internal capacitance to the output node during the evaluation

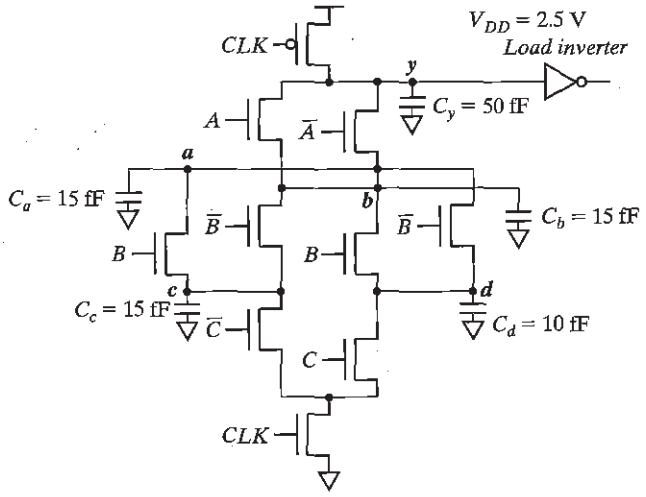


Figure 6-60 Example illustrating the charge-sharing effect in dynamic logic.

period. This happens for $\overline{A}\ B\ C$ or $A\ \overline{B}\ C$. The voltage change can then be obtained by equating the initial charge with the final charge as done with equation Eq. (6.44), yielding a worst case change of 30/(30 + 50) * 2.5 V = 0.94 V. To ensure that the circuit functions correctly, the switching threshold of the connecting inverter should be placed below 2.5 – 0.94 = 1.56 V.

The most common and effective approach to deal with the charge redistribution is to also precharge critical internal nodes, as shown in Figure 6-61. Since the internal nodes are charged

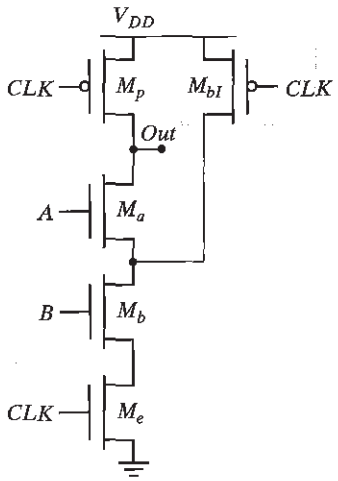


Figure 6-61 Dealing with charge sharing by precharging internal nodes. An NMOS precharge transistor may also be used, but this requires an inverted clock.

to V_{DD} during precharge, charge sharing does not occur. This solution obviously comes at the cost of increased area and capacitance.

Capacitive Coupling

The relatively high impedance of the output node makes the circuit very sensitive to crosstalk effects. A wire routed over or next to a dynamic node may couple capacitively and destroy the state of the floating node. Another equally important form of capacitive coupling is *backgate (*or *output-to-input) coupling.* Consider the circuit shown in Figure 6-62a, in which a dynamic two-input NAND gate drives a static NAND gate. A transition in the input *In* of the static gate may cause the output of the gate (Out_2) to go low. This output transition couples capacitively to the other input of the gate (the dynamic node Out_1) through the gate–source and gate–drain capacitances of transistor M_4. A simulation of this effect is shown in Figure 6-62b. It demonstrates how the coupling causes the output of the dynamic gate Out_1 to drop significantly. This further causes the output of the static NAND gate not to drop all the way down to 0 V and a small amount of static power to be dissipated. If the voltage drop is large enough, the circuit can evaluate incorrectly, and the NAND output may not go low. When designing and laying out dynamic circuits, special care is needed to minimize capacitive coupling.

Clock Feedthrough

A special case of capacitive coupling is clock feedthrough, an effect caused by the capacitive coupling between the clock input of the precharge device and the dynamic output node. The coupling capacitance consists of the gate-to-drain capacitance of the precharge device, and includes both the overlap and channel capacitances. This capacitive coupling causes the output of the dynamic node to rise above V_{DD} on the low-to-high transition of the clock, assuming that the pull-down network is turned off. Subsequently, the fast rising and falling edges of the clock couple onto the signal node, as is quite apparent in the simulation of Figure 6-62b.

The danger of clock feedthrough is that it may cause the normally reverse-biased junction diodes of the precharge transistor to become forward biased. This causes electron injection into the substrate, which can be collected by a nearby high-impedance node in the **1** state, eventually resulting in faulty operation. CMOS latchup might be another result of this injection. For all purposes, high-speed dynamic circuits should be carefully simulated to ensure that clock feedthrough effects stay within bounds.

All of the preceding considerations demonstrate that the design of dynamic circuits is rather tricky and requires extreme care. It should therefore be attempted only when high performance is required, or high quality design-automation tools are available.

6.3.4 Cascading Dynamic Gates

Besides the signal integrity issues, there is one major catch that complicates the design of dynamic circuits: Straightforward cascading of dynamic gates to create multilevel logic structures does not work. The problem is best illustrated with two cascaded *n*-type dynamic

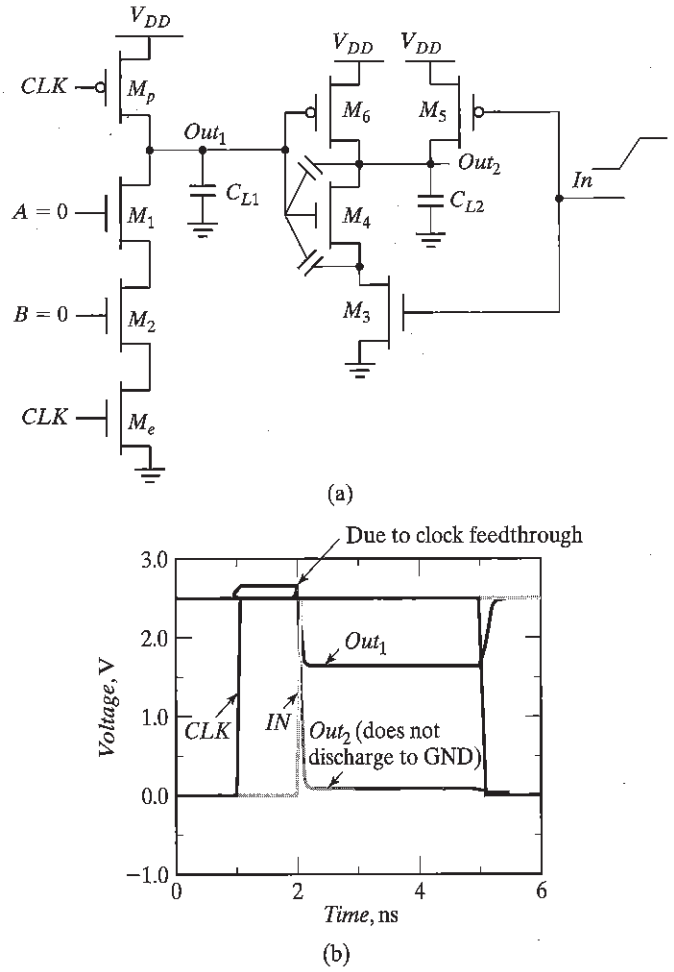


Figure 6-62 Example demonstrating the effect of backgate coupling: (a) circuit schematics; (b) simulation results.

inverters, shown in Figure 6-63a. During the precharge phase (i.e., $CLK = 0$), the outputs of both inverters are precharged to V_{DD}. Assume that the primary input *In* makes a $0 \rightarrow 1$ transition (Figure 6-63b). On the rising edge of the clock, output Out_1 starts to discharge. The second output should remain in the precharged state of V_{DD} as its expected value is 1 (Out_1 transitions to 0 during evaluation). However, there is a finite propagation delay for the input to discharge Out_1 to GND. Therefore, the second output also starts to discharge. As long as Out_1 exceeds the switching threshold of the second gate, which approximately equals V_{Tn}, a conducting path exists between Out_2 and GND, and precious charge is lost at Out_2. The conducting path is only disabled once Out_1 reaches V_{Tn}, and turns off the NMOS pull-down transistor. This leaves Out_2 at an intermediate voltage level. The correct level will not be recovered, because dynamic gates rely on capacitive storage, in contrast to static gates, which have dc restoration. The charge loss leads to reduced noise margins and potential malfunctioning.

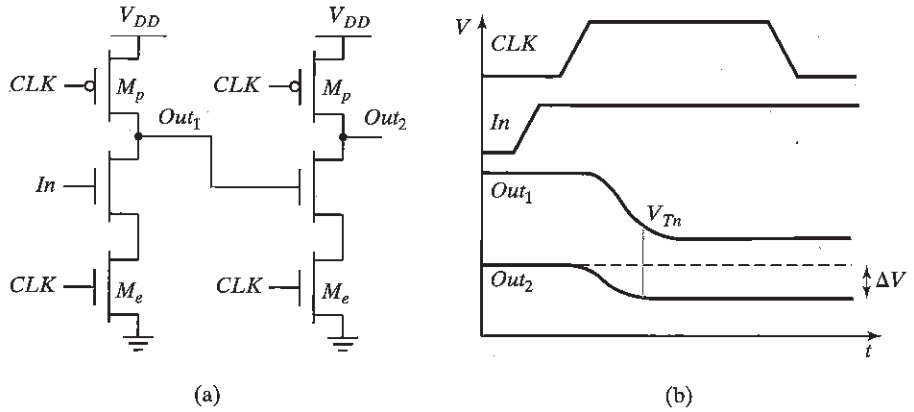


Figure 6-63 Cascade of dynamic *n*-type blocks.

The cascading problem arises because the outputs of each gate—and thus the inputs to the next stages—are precharged to 1. This may cause inadvertent discharge in the beginning of the evaluation cycle. Setting all the inputs to 0 during precharge addresses that concern. When doing so, all transistors in the pull-down network are turned off after precharge, and no inadvertent discharging of the storage capacitors can occur during evaluation. In other words, correct operation is guaranteed as long as **the inputs can only make a single 0 → 1 transition during the evaluation period.**[5] Transistors are turned on only when needed—and at most, once per cycle. A number of design styles complying with this rule have been conceived, but the two most important ones are discussed next.

Domino Logic

Concept A domino logic module [Krambeck82] consists of an *n*-type dynamic logic block followed by a static inverter (Figure 6-64). During precharge, the output of the *n*-type dynamic gate is charged up to V_{DD}, and the output of the inverter is set to 0. During evaluation, the dynamic gate conditionally discharges, and the output of the inverter makes a conditional transition from 0 → 1. If one assumes that all the inputs of a domino gate are outputs of other domino gates,[6] then it is ensured that all inputs are set to 0 at the end of the precharge phase, and that the only transitions during evaluation are 0 → 1 transitions. Hence, the formulated rule is obeyed. The introduction of the static inverter has the additional advantage that the fan-out of the gate is driven by a static inverter with a low-impedance output, which increases noise immunity. Also, the buffer reduces the capacitance of the dynamic output node by separating internal and load capacitances. Finally, the inverter can be used to drive a bleeder device to combat leakage and charge redistribution, as shown in the second stage of Figure 6-64.

[5]This ignores the impact of charge distribution and leakage effects, discussed earlier.

[6]It is required that all other inputs that do not fall under this classification (for instance, primary inputs) stay constant during evaluation.

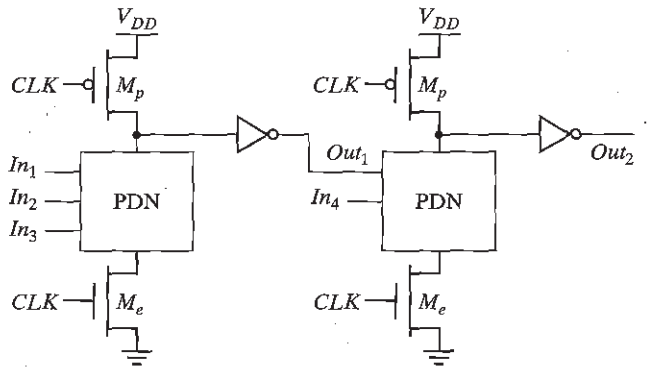


Figure 6-64 Domino CMOS logic.

Consider now the operation of a chain of domino gates. During precharge, all inputs are set to 0. During evaluation, the output of the first domino block either stays at 0 or makes a $0 \rightarrow 1$ transition, affecting the second gate. This effect might ripple through the whole chain, one after the other, similar to a line of falling dominoes—hence the name. Domino CMOS has the following properties:

- Since each dynamic gate has a static inverter, only noninverting logic can be implemented. Although there are ways to deal with this, as discussed in a subsequent section, this is a major limiting factor, and pure domino design has thus become rare.
- Very high speeds can be achieved: only a rising edge delay exists, while t_{pHL} equals zero. The inverter can be sized to match the *fan-out*, which is already much smaller than in the complimentary static CMOS case, as only a single gate capacitance has to be accounted for per fan-out gate.

Since the inputs to a domino gate are low during precharge, it is tempting to eliminate the evaluation transistor because this reduces clock load and increases pull-down drive. However, eliminating the evaluation device extends the precharge cycle—the precharge now has to ripple through the logic network as well. Consider the logic network shown in Figure 6-65, where the evaluation devices have been eliminated. If the primary input In_1 is 1 during evaluation, the output of each dynamic gate evaluates to 0, and the output of each static inverter is 1. On the falling edge of the clock, the precharge operation is started. Assume further that In_1 makes a high-to-low transition. The input to the second gate is initially high, and it takes two gate delays before In_2 is driven low. During that time, the second gate cannot precharge its output, as the pull-down network is fighting the precharge device. Similarly, the third gate has to wait until the second gate precharges before it can start precharging, etc. Therefore, the time taken to precharge the logic circuit is equal to its critical path. Another important negative is the extra power dissipation when both pull-up and pull-down devices are on. Therefore, it is good practice to always utilize evaluation devices.

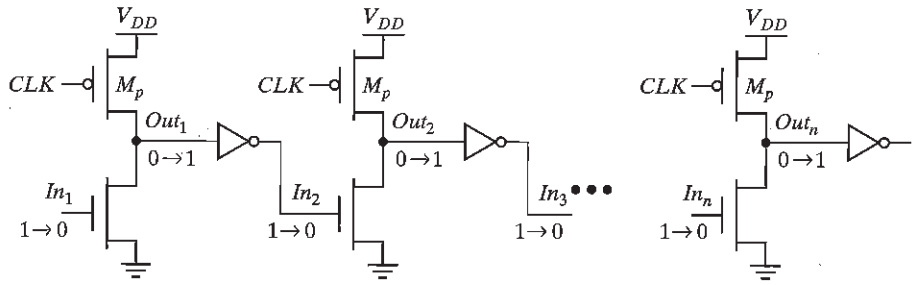
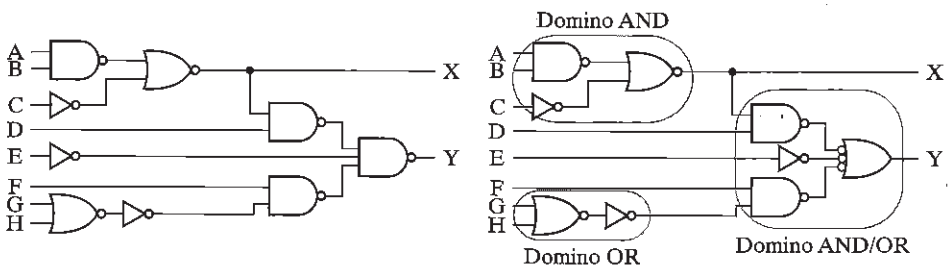


Figure 6-65 Effect of ripple precharge when the evaluation transistor is removed. The circuit also exhibits static power dissipation.

Dealing with the Noninverting Property of Domino Logic A major limitation in domino logic is that only noninverting logic can be implemented. This requirement has limited the widespread use of pure domino logic. There are several ways to deal with it, though. Figure 6-66 shows one approach to the problem—reorganizing the logic using simple boolean transforms such as De Morgan's Law. Unfortunately, this sort of optimization is not always possible, and more general schemes may have to be used.

A general (but expensive) approach to solving the problem is the use of differential logic. *Dual-rail domino* is similar in concept to the DCVSL structure discussed earlier, but it uses a precharged load instead of a static cross-coupled PMOS load. Figure 6-67 shows the circuit schematic of an AND/NAND differential logic gate. Note that all inputs come from other differential domino gates. They are low during the precharge phase, while making a conditional $0 \rightarrow 1$ transition during evaluation. Using differential domino, it is possible to implement any arbitrary function. This comes at the expense of an increased power dissipation, since a transition is guaranteed every single clock cycle regardless of the input values—either O or $\overline{O}$ must make a $0 \rightarrow 1$ transition. The function of transistors M_{f1} and M_{f2} is to keep the circuit static when the clock is high for extended periods of time (*bleeder*). Notice that this circuit is not ratioed, even in the presence of the PMOS pull-up devices! Due to its high performance, this differential approach is very popular, and is used in several commercial microprocessors.



(a) Before logic transformation (b) After logic transformation

Figure 6-66 Restructuring logic to enable implementation by using noninverting domino logic.

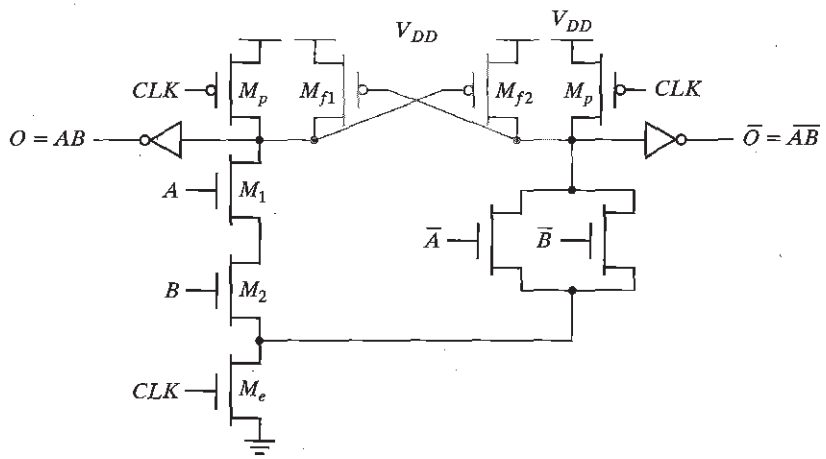


Figure 6-67 Simple dual rail (differential) domino logic gate.

Optimization of Domino Logic Gates Several optimizations can be performed on domino logic gates. The most obvious performance optimization involves the sizing of the transistors in the static inverter. With the inclusion of the evaluation devices in domino circuits, all gates precharge in parallel, and the precharge operation takes only two gate delays—charging the output of the dynamic gate to V_{DD}, and driving the inverter output low. The critical path during evaluation goes through the pull-down path of the dynamic gate and through the PMOS pull-up transistor of the static inverter. Therefore, to speed up the circuit during evaluation, the beta ratio of the static inverter should be made high so that its switching threshold is close to V_{DD}. This can be accomplished by using a small (minimum-sized) NMOS and a large PMOS device. The minimum-sized NMOS only affects the precharge time, which is generally limited due to the parallel precharging of all gates. The only disadvantage of using a large beta ratio is a reduction in noise margin. Hence, a designer should consider reduced noise margin and performance impact simultaneously during the device sizing.

Numerous variations of domino logic have been proposed [Bernstein98]. One optimization that reduces area is *multiple-output domino logic*. The basic concept is illustrated in Figure 6-68. It exploits the fact that certain outputs are subsets of other outputs to generate a number of logical functions in a single gate. In this example, $O3 = C + D$ is used in all three outputs, and thus it is implemented at the bottom of the pull-down network. Since $O2$ equals $B \cdot O3$, it can reuse the logic for $O3$. Notice that the internal nodes have to be precharged to V_{DD} to produce the correct results. Given that the internal nodes precharge to V_{DD}, the number of devices driving precharge devices is not reduced. However, the number of evaluation transistors is drastically reduced because they are amortized over multiple outputs. Additionally, this approach results in a reduction of the fan-out factor, again due to the reuse of transistors over multiple functions.

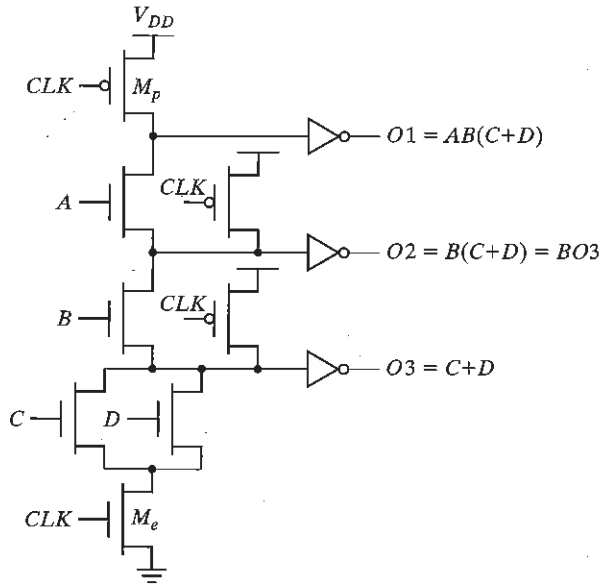


Figure 6-68 Multiple-output domino.

Compound domino (Figure 6-69) represents another optimization of the generic domino gate, once again minimizing the number of transistors. Instead of each dynamic gate driving a static inverter, it is possible to combine the outputs of multiple dynamic gates with the aid of a complex static CMOS gate, as shown in Figure 6-69. The outputs of three dynamic structures (implementing $O1 = \overline{A\ B\ C}$, $O2 = \overline{D\ E\ F}$ and $O3 = \overline{G\ H}$) are combined using a single complex CMOS static gate that implements $O = \overline{(O1 + O2)\ O3}$. The total logic function realized this way is $O = A\ B\ C\ D\ E\ F + GH$.

Compound domino is a useful tool for constructing complex dynamic logic gates. Large dynamic stacks are replaced by parallel structures with small fan-in and complex CMOS gates. For example, a large *fan-in* domino AND can be implemented as a set of parallel dynamic NAND structures with lower *fan-in*, combined with a static NOR gate. One important consideration in Compound domino is the problem associated with backgate coupling. Care must be taken to ensure that the dynamic nodes are not affected by the coupling between the output of the static gates and the output of dynamic nodes.

np-CMOS

An alternative approach to cascading dynamic logic is provided by *np*-CMOS, which uses two flavors (*n*-tree and *p*-tree) of dynamic logic, and avoids the extra static inverter in the critical path that comes with domino logic. In a *p*-tree logic gate, PMOS devices are used to build a pull-up logic network, including a PMOS evaluation transistor ([Gonçalvez83, Friedman84, Lee86]).

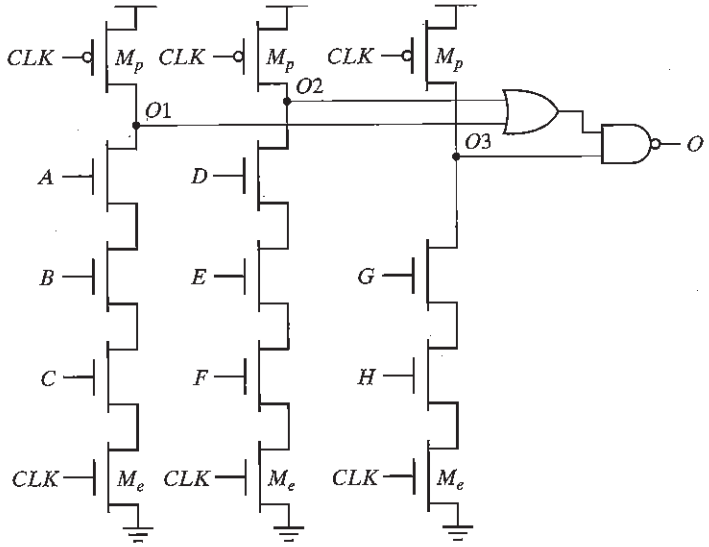


Figure 6-69 Compound domino logic uses complex static gates at the output of the dynamic gates.

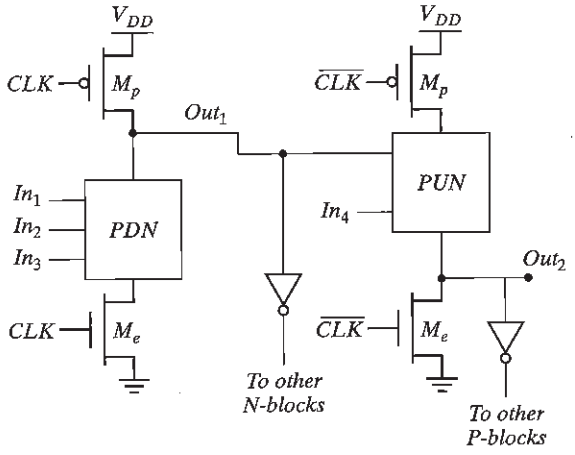


Figure 6-70 The *np*-CMOS logic circuit style.

(see Figure 6-70). The NMOS predischarge transistor drives the output low during precharge The output conditionally makes a $0 \rightarrow 1$ transition during evaluation depending on its inputs.

np-CMOS logic exploits the duality between *n*-tree and *p*-tree logic gates to eliminate the cascading problem. If the *n*-tree gates are controlled by *CLK*, and *p*-tree gates are controlled using $\overline{CLK}$, *n*-tree gates can directly drive *p*-tree gates, and vice versa. Similar to domino, *n*-tree outputs must go through an inverter when connecting to another *n*-tree gate. During the

precharge phase ($CLK = 0$), the output of the n-tree gate, Out_1, is charged to V_{DD}, while the output of the p-tree gate, Out_2, is predischarged to 0 V. Since the n-tree gate connects PMOS pull-up devices, the PUN of the p-tree is turned off at that time. During evaluation, the output of the n-tree gate can only make a $1 \rightarrow 0$ transition, conditionally turning on some transistors in the p-tree. This ensures that no accidental discharge of Out_2 can occur. Similarly, n-tree blocks can follow p-tree gates without any problems, because the inputs to the n-gate are precharged to 0. A disadvantage of the np-CMOS logic style is that the p-tree blocks are slower than the n-tree modules, due to the lower current drive of the PMOS transistors in the logic network. Equalizing the propagation delays requires extra area. Also, the lack of buffers requires that dynamic nodes are routed between gates.

6.4 Perspectives

6.4.1 How to Choose a Logic Style?

In the preceding sections, we have discussed several gate-implementation approaches using the CMOS technology. Each of the circuit styles has its advantages and disadvantages. Which one to select depends upon the primary requirement: ease of design, robustness, area, speed, or power dissipation. No single style optimizes all these measures at the same time. Even more, the approach of choice may vary from logic function to logic function.

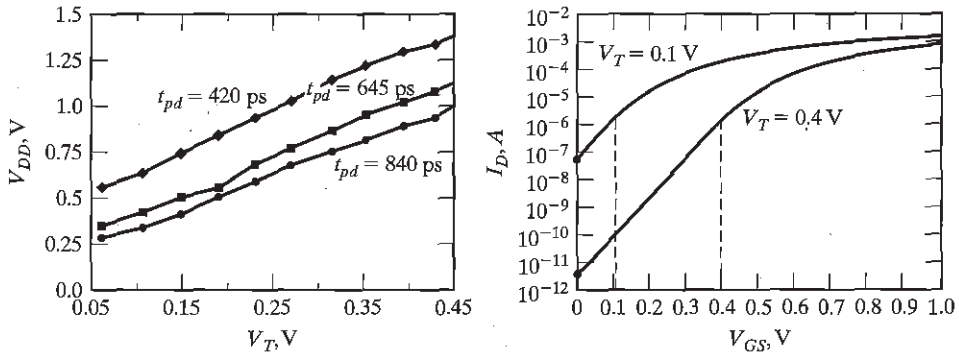
The static approach has the advantage of being robust in the presence of noise. This makes the design process rather trouble free and amenable to a high degree of automation. It is clearly the best general-purpose logic design style. This ease of design does come at a cost: For complex gates with a large fan-in, complementary CMOS becomes expensive in terms of area and performance. Alternative static logic styles have therefore been devised. Pseudo-NMOS is simple and fast at the expense of a reduced noise margin and static power dissipation. Pass-transistor logic is attractive for the implementation of a number of specific circuits, such as multiplexers and XOR-dominated logic like adders.

Dynamic logic, on the other hand, makes it possible to implement fast and small complex gates. This comes at a price, however. Parasitic effects such as charge sharing make the design process a precarious job. Charge leakage forces a periodic refresh, which puts a lower bound on the operating frequency of the circuit.

The current trend is towards an increased use of complementary static CMOS. This tendency is inspired by the increased use of design-automation tools at the logic design level. These tools emphasize optimization at the logic level, rather than at the circuit level, and they put a premium on robustness. Another argument is that static CMOS is more amenable to voltage scaling than some of the other approaches discussed in this chapter.

6.4.2 Designing Logic for Reduced Supply Voltages

In Chapter 3, we projected that the supply voltage for CMOS processes will continue to drop over the coming decade, and may go as low as 0.6 V by 2010. To maintain performance under



(a) V_{DD}/V_T *for fixed performance* (b) *Leakage as a function of* V_T

Figure 6-71 Voltage Scaling (V_{DD}/V_T on delay and leakage).

those conditions, it is essential that the device thresholds scale as well. Figure 6-71a shows a plot of the (V_T, V_{DD}) ratio required to maintain a given performance level (assuming that other device characteristics remain identical).

This trade-off is not without penalty. Reducing the threshold voltage increases the subthreshold leakage current exponentially, as we derived in Eq. (3.39) (repeated here for the sake of clarity):

$$I_{leakage} = I_S 10^{\frac{V_{GS}-V_{th}}{S}} \left(1 - 10^{\frac{nV_{DS}}{S}}\right) \tag{6.46}$$

In Eq. (6.46), S is the *slope factor* of the device. The subthreshold leakage of an inverter is the current of the NMOS for V_{in} = 0 V and V_{out} = V_{DD} (or the PMOS current for V_{in} = V_{DD} and V_{out} = 0). The exponential increase in inverter leakage for decreasing thresholds is illustrated in Figure 6-71b.

These leakage currents are a concern particularly for designs that feature intermittent computational activity separated by long periods of inactivity. For example, the processor in a cellular phone remains in idle mode for a majority of the time. While the processor is in idle mode, ideally, the system should consume zero or near-zero power. This is only possible if leakage is low—that is, the devices have a high threshold voltage. This is in contrast to the scaling scenario that we just depicted, where high performance under low supply voltage means reduced thresholds. To satisfy the contradicting requirements of high performance during active periods and low leakage during standby, several process modifications or leakage-control techniques have been introduced in CMOS processes. Most processes with feature sizes at or below 0.18 μm CMOS support devices with different thresholds—typically a device with low threshold for high-performance circuits, and a transistor with high threshold for leakage control. Another approach gaining popularity is the

dynamic control of the threshold voltage of a device by exploiting the body effect of the transistor. Use of this approach to control individual devices requires a dual-well process (see Figure 2-2).

Clever circuit design can also help reduce the leakage current, which is a function of the circuit topology and the value of the inputs applied to the gate. Since V_T depends on body bias (V_{BS}), the subthreshold leakage of an MOS transistor depends not only on the gate drive (V_{GS}), but also on the body bias. In an inverter with $In = 0$, the subthreshold leakage of the inverter is set by the NMOS transistor with its $V_{GS} = V_{BS} = 0$ V. In more complex CMOS gates, such as the two-input NAND gate of Figure 6-72, the leakage current depends on the input vector. The subthreshold leakage current of this gate is the least when $A = B = 0$. Under these conditions, the intermediate node X settles to

$$V_X \approx V_{th} \ln(1 + n) \tag{6.47}$$

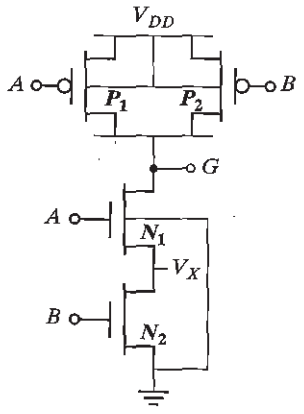
The leakage current of the gate is then determined by the topmost NMOS transistor with $V_{GS} = V_{BS} = -V_X$. Clearly, the subthreshold leakage under this condition is smaller than that of the inverter. This reduction due to stacked transistors is called the *stack effect*. The Table in Figure 6-72 analyzes the leakage components for the two-input NAND gate under different input conditions.

The reality is even better. In short-channel MOS transistors, the subthreshold leakage current depends not only on the gate drive (V_{GS}) and the body bias (V_{BS}), but also on the drain voltage (V_{DS}). The threshold voltage of a short-channel MOS transistor decreases with increasing V_{DS} due to *drain-induced barrier lowering* (DIBL). Typical values for DIBL can range from a 20- to a 150-mV change in V_T per voltage change in V_{DS}. Because of this, the impact of the stack effect is even more significant for short-channel transistors. The intermediate voltage reduces the drain–source voltage of the topmost device, increases its threshold, and thus lowers its leakage.

Example 6.19 Stack Effect in Two-Input NAND Gate

Consider again the two-input NAND gate of Figure 6-72a, when both $N_1$and N_2 are *off* ($A = B = 0$). From the simulated load lines shown in Figure 6-72c, we see that V_X settles to approximately 100 mV in steady state. The steady-state subthreshold leakage in the gate is therefore due to $V_{GS} = V_{BS} = -100$ mV and $V_{DS} = V_{DD} - 100$ mV, which is 20 times smaller than the leakage of a stand-alone NMOS transistor with $V_{GS} = V_{BS} = 0$ mV and $V_{DS} = V_{DD}$ [Ye98].

In sum, the subthreshold leakage in complex stacked circuits can be significantly lower than in individual devices. Observe that the maximum leakage reduction occurs when all the transistors in the stack are *off*, and the intermediate node voltage reaches its steady–state value. Exploiting this effect requires a careful selection of the input signals to every gate during standby or sleep mode.



A	B	V_X	I_{SUB}
0	0	$V_{th} \ln(1+n)$	$I^N_{SUB}(V_{GS} = V_{BS} = -V_X)$
0	1	0	$I^N_{SUB}(V_{GS} = V_{BS} = 0)$
1	0	$V_{dd} - V_T$	$I^N_{SUB}(V_{GS} = V_{BS} = 0)$
1	1	0	$2\,I^P_{SUB}(V_{SG} = V_{SB} = 0)$

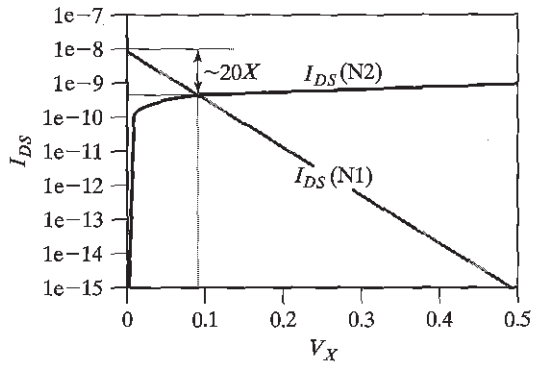


Figure 6-72 Subthreshold leakage reduction in a two-input NAND gate (a) due to stack effect for different input conditions (b). Figure (c) plots the simulated load lines of the gate for $A = B = 0$.

Problem 6.9 Computing V_X

Equation (6.47) calculates the intermediate node voltage for a two-input NAND with less than 10% error, for $A = B = 0$. Derive Eq. (6.47) assuming (1) V_T and I_S of N_1 and N_2 are approximately equal, (2) NMOS transistors are identically sized, and (3) $n < 1.5$.

6.5 Summary

In this chapter, we have extensively analyzed the behavior and performance of combinational CMOS digital circuits with regard to area, speed, and power. We summarize the major points as follows:

- *Static complementary* CMOS combines dual pull-down and pull-up networks, only one of which is enabled at any time.

- The performance of a CMOS gate is a strong function of the *fan-in*. Techniques to deal with fan-in include transistor sizing, input reordering, and partitioning. The speed is also a linear function of the fan-out. Extra buffering is needed for large fan-outs.
- The *ratioed logic* style consists of an active pull-down (-up) network connected to a load device. This results in a substantial reduction in gate complexity at the expense of static power consumption and an asymmetrical response. Careful transistor sizing is necessary to maintain sufficient noise margins. The most popular approaches in this class are the pseudo-NMOS techniques and differential DCVSL, which require complementary signals.
- *Pass-transistor logic* implements a logic gate as a simple switch network. This results in very simple implementations for some logic functions. Long cascades of switches are to be avoided due to a quadratic increase in delay with respect to the number of elements in the chain. NMOS-only pass-transistor logic produces even simpler structures, but might suffer from static power consumption and reduced noise margins. This problem can be addressed by adding a level-restoring transistor.
- The operation of *dynamic logic* is based on the storage of charge on a capacitive node and the conditional discharging of that node as a function of the inputs. This calls for a two-phase scheme, consisting of a precharge followed by an evaluation step. Dynamic logic trades off noise margin for performance. It is sensitive to parasitic effects such as leakage, charge redistribution, and clock feedthrough. Cascading dynamic gates can cause problems and thus should be addressed carefully.
- The *power consumption* of a logic network is strongly related to the switching activity of the network. This activity is a function of the input statistics, the network topology, and the logic style. Sources of power consumption such as glitches and short-circuit currents can be minimized by careful circuit design and transistor sizing.
- Threshold voltage scaling is required for *low-voltage operation*. Leakage control is critical for low-voltage operation.

6.6 To Probe Further

The topic of (C)MOS logic styles is treated extensively in the literature. Numerous texts have been devoted to the issue. Some of the most comprehensive treatments can be found in [Weste93] and [Chandrakasan01]. Regarding the intricacies of high-performance design, [Shoji96] and [Bernstein98] offer the most in-depth discussion of the optimization and analysis of digital MOS circuits. The topic of power minimization is relatively new, but comprehensive reference works are available in [Chandrakasan95], [Rabaey95], and [Pedram02].

Innovations in the MOS logic area are typically published in the proceedings of the ISSCC Conference and the VLSI circuits symposium, as well as the *IEEE Journal of Solid State Circuits* (especially the November issue).

References

[Bernstein98] K. Bernstein et al., *High-Speed CMOS Design Styles*, Kluwer Academic Publishers, 1998.

[Chandrakasan95] A. Chandrakasan and R. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995.

[Chandrakasan01] A. Chandrakasan, W. Bowhill, and F. Fox, ed., *Design of High-Performance Microprocessor Circuits*, IEEE Press, 2001.

[Gonçalvez83] N. Gonçalvez and H. De Man, "NORA: A Racefree Dynamic CMOS Technique for Pipelined Logic Structures," *IEEE Journal of Solid State Circuits*, vol. SC-18, no. 3, pp. 261–266, June 1983.

[Heller84] L. Heller et al., "Cascade Voltage Switch Logic: A Differential CMOS Logic Family," *Proc. IEEE ISSCC Conference*, pp. 16–17, February 1984.

[Krambeck82] R. Krambeck et al., "High-Speed Compact Circuits with CMOS," *IEEE Journal of Solid State Circuits*, vol. SC-17, no. 3, pp. 614–619, June 1982.

[Landman91] P. Landman and J. Rabaey, "Design for Low Power with Applications to Speech Coding," *Proc. International Micro-Electronics Conference*, Cairo, December 1991.

[Parameswar96] A. Parameswar, H. Hara, and T. Sakurai, "A Swing Restored Pass-Transistor Logic-Based Multiply and Accumulate Circuit for Multimedia Applications," *IEEE Journal of Solid State Circuits*, vol. SC-31, no. 6, pp. 805–809, June 1996.

[Pedram02] M. Pedram and J. Rabaey, ed., *Power-Aware Design Methodologies*, Kluwer, 2002.

[Rabaey95] J. Rabaey and M. Pedram, ed., *Low Power Design Methodologies*, Kluwer, 1995.

[Radhakrishnan85] D. Radhakrishnan, S. Whittaker, and G. Maki, "Formal Design Procedures for Pass-Transistor Switching Circuits," *IEEE Journal of Solid State Circuits*, vol. SC-20, no. 2, pp. 531–536, April 1985.

[Shoji88] M. Shoji, *CMOS Digital Circuit Technology*, Prentice Hall, 1988.

[Shoji96] M. Shoji, *High-Speed Digital Circuits*, Addison-Wesley, 1996.

[Sutherland99] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort*, Morgan Kaufmann, 1999.

[Weste93] N. Weste and K. Eshragian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley, 1993.

[Yano90] K. Yano et al., "A 3.8 ns CMOS 16 × 16 b Multiplier Using Complimentary Pass-Transistor Logic," *IEEE Journal of Solid State Circuits*, vol. SC-25, no. 2, pp. 388–395, April 1990.

[Ye98] Y. Ye, S. Borkar, and V. De, "A New Technique for Standby Leakage Reduction in High-Performance Circuits," *Symposium on VLSI Circuits*, pp. 40–41, 1998.

Exercises

For the latest problem sets and design challenges in CMOS digital logic, log in to **http://bwrc.eecs.berkeley.edu/IcBook**.

DESIGN METHODOLOGY INSERT

C

How to Simulate Complex Logic Circuits

Timing- and Switch-Level Simulation

Logic and Functional Simulation

Behavioral Simulation

Register-Transfer Languages

While circuit simulation in the SPICE style proves to be an extremely valuable element of the designers tool box, it has one major deficiency. By taking into account all the peculiarities and second-order effects of the semiconductor devices, it tends to be time consuming. It rapidly becomes unwieldy when designing complex circuits, unless one is willing to spend days of computer time. Even though computers are always getting faster and simulators are getting better, circuits are getting complex even faster. The designer can address the complexity issue by giving up modeling accuracy and resorting to higher representation levels. A discussion of the different abstraction levels available to the designer and their impact on simulation accuracy is the topic of this insert.

The best way of differentiating among the myriad of simulation approaches and abstraction levels is to identify how the data and time variables are represented—as analog, continuous variables, as discrete signals, or as abstract data models.

C.1 Representing Digital Data as a Continuous Entity

Circuit Simulation and Derivatives

In Design Methodology Insert B, we established that a circuit simulator is "digitallyagnostic," meaning that it is, in essence, an analog simulator. Voltage, current, and time are treated as analog variables. This accurate modeling, combined with the nonlinearity of most of the devices leads to a high overhead in simulation time.

Substantial effort has been invested to decrease the computation time at the expense of generality. Consider an MOS digital circuit. Due to the excellent isolation property of the MOS gate, it is often possible to partition the circuit into a number of sections that have limited interaction. A possible approach is to solve each of these partitions individually over a given period, assuming that the inputs from other sections are known or constant. The resulting waveforms can then be iteratively refined. This *relaxation-based* approach has the advantage of being computationally more effective than the traditional technique by avoiding expensive matrix inversions, but it is restricted to MOS circuits [White87]. When the circuit contains feedback paths, the partitions can become large, and simulation performance degrades.

Another approach is to reduce the complexity of the transistor models used. For example, linearization of the model leads to a dramatic reduction in the computational complexity. Yet another approach is to employ a simplified table-lookup model. While this approach, by necessity, leads to a decreased accuracy of the waveforms, it still allows for a good estimation of timing parameters such as propagation delay and rise and fall times. This explains why these tools often are called *timing simulators*. The big advantage is in the execution speed, which can be one or two orders of magnitude higher than that of SPICE-like tools. Another advantage is that, in contrast to the tools that are discussed next, timing simulators still can incorporate second-order effects such as leakage, threshold drops, and signal glitches. Examples of an offering in this class is the NanoSim (formerly TimeMill/PowerMill) tool set from Synopsys [TimeMill]. As a point of reference, simulators in this class typically give up 5 to 10% in accuracy on timing parameters, with respect to full-blown circuit simulators.

C.2 Representing Data as a Discrete Entity

In digital circuits, we generally are not interested in the actual value of the voltage variable, but only in the digital value it represents. Therefore, it is possible to envision a simulator in which data signals are either in the 0 or 1 range. Signals that do no comply with either condition are denoted as X, or undefined.

This tertiary representation $\{0, 1, X\}$ is used extensively in simulators at both the device and gate level. By augmenting this set of allowable data values, we can obtain more detailed information, while retaining the capability of handling complex designs. Possible extensions are the Z-value for a tristate node in the high-impedance state, and R- and F-values for the rising and falling transients. Some commercially offered simulators provide as many as a dozen possible signal states.

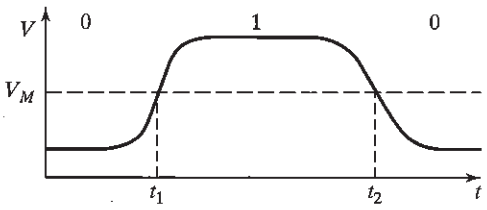


Figure C-1 Discretizing the time variable.

While substantial performance improvement is obtained by making the data representation space discrete, similar benefits can be obtained by making time a discrete variable as well. Consider the voltage waveform of Figure C-1, which represents the signal at the input of an inverter with a switching threshold V_M. It is reasonable to assume that the inverter output changes its value one propagation delay after its input crossed V_M. When one is not strictly interested in the exact shape of the signal waveforms, it is sufficient to evaluate the circuit only at the interesting time points, t_1 and t_2.

Similarly, the interesting points of the output waveform are situated at $t_1 + t_{pHL}$ and $t_2 + t_{pLH}$. A simulator that only evaluates a gate at the time an event happens at one of its inputs is called an *event-driven* simulator. The evaluation order is determined by putting projected events on a time queue and processing them in a time-ordered fashion. Suppose that the waveform of Figure C-1 acts as the input waveform to a gate. An event is scheduled to occur at time t_1. Upon processing that event, a new event is scheduled for the fan-out nodes at $t_1 + t_{pHL}$ and is put on the time queue. This event-driven approach is evidently more efficient than the time-step-driven approach of the circuit simulators. To take the impact of fan-out into account, the propagation delay of a circuit can be expressed in terms of an intrinsic delay (t_{in}) and a load-dependent factor (t_l), and it can differ over edge transitions:

$$t_{pLH} = t_{inLH} + t_{lLH} \times C_L \tag{C.1}$$

The load C_L can be entered in absolute terms (in pF) or as a function of the number of fan-out gates. Observe how closely this equation resembles the *logical-effort* model we introduced in the preceding chapter.

While offering a substantial performance benefit, the preceding approach still has the disadvantage that events can happen any time. Another simplification could be to make the time even more discrete and allow events to happen only at integer multiples of a *unit time* variable. An example of such an approach is the *unit-delay* model, where each circuit has a single delay of one unit. Finally, the simplest model is the *zero-delay model,* in which gates are assumed to be free of delay. Under this paradigm, time proceeds from one clock event to the next, and all events are assumed to occur instantaneously upon arrival of a clocking event. These concepts can be applied on a number of abstraction levels, resulting in the simulation approaches discussed next.

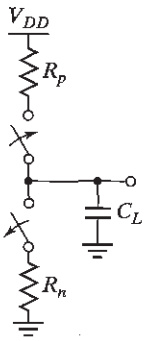


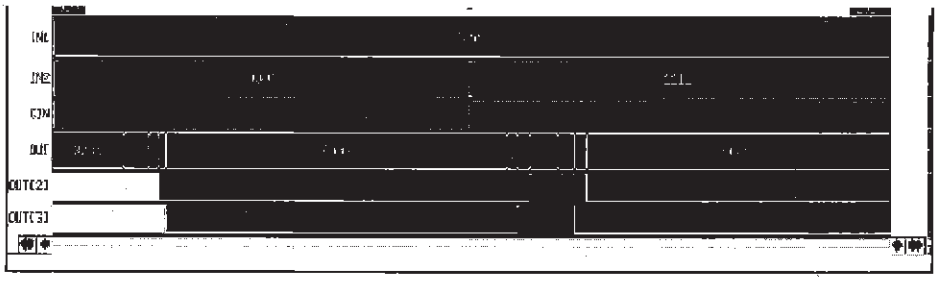
Figure C-2 Switch-level model of CMOS inverter.

Switch-Level Simulation

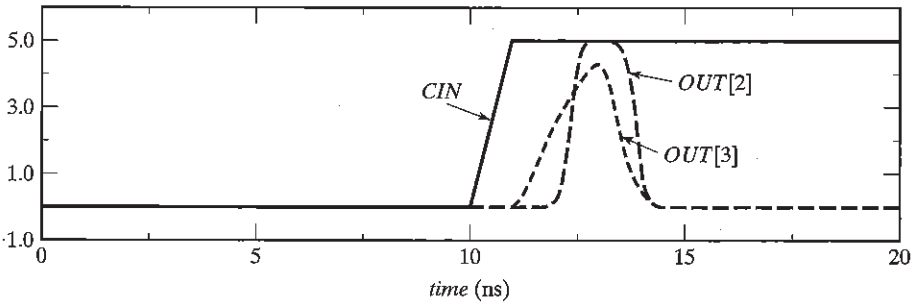
The nonlinear nature of semiconductor devices is one of the major impediments to higher simulation speeds. The switch-level model [Bryant81] overcomes this hurdle by approximating the transistor behavior with a linear resistance whose value is a function of the operating conditions. In the off-mode, the resistance is set to infinity, while in the on-mode, it is set to the average "on" resistance of the device (Figure C-2). The resulting network is a time-variant, linear network of resistors and capacitors that can be more efficiently analyzed. Evaluation of the resistor network determines the steady-state values of the signals and typically employs a {0, 1, *X*} model. For instance, if the total resistance between a node and GND is substantially smaller than the resistance to V_{DD}, the node is set to the 0-state. The timing of the events can be resolved by analyzing the *RC* network. Simpler timing models such as the unit-delay model are also employed.

Example C.1 Switch versus Circuit-Level Simulation

A four-bit adder is simulated using the switch-level simulator IRSIM ([Salz89]). The simulation results are plotted in Figure C-3. Initially, all inputs (*IN*1 and *IN*2) and the carry-input *CIN* are set to 0. After 10 nsec, all inputs *IN*2 as well as *CIN* are set to 1. The display window plots the input signals, the output vector *OUT*[0–3], and the most significant output bits *OUT*[2] and *OUT*[3]. The output converges to the correct value 0000 after a transition period. Notice how the data assumes only 0 and 1 levels. The glitches in the output signals go rail to rail, although in reality they might represent only partial excursions. During transients, the signal is marked *X*, which means "undefined." To put this result in perspective, Figure C-3 plots the SPICE results for the same input vectors. Notice the partial glitches. Also, it shows that the IRSIM timing, which is based on an *RC* model, is relatively accurate and sufficient to get a first-order impression.



(a) IRSIM results.



(b) SPICE results.

Figure C-3 Comparison between circuit and switch-level simulations.

Gate-Level (or Logic) Simulation

Gate-level simulators use the same signal values as the switch-level tools, but the simulation primitives are gates instead of transistors. This approach enables the simulation of more complex circuits at the expense of detail and generality. For example, some common VLSI structures such as tristate busses and pass transistors are hard to deal with at this level. Since gate level is the preferred entry level for many designers, this simulation approach remained extremely popular until the introduction of logic synthesis tools, which moved the focus to the functional or behavioral abstraction layer. The interest in logic simulation was so great that special and expensive hardware accelerators were developed to expedite the simulation process (e.g., [Agrawal90]).

Functional Simulation

Functional simulation can be considered as a simple extension of logic simulation. The primitive elements of the input description can be of an arbitrary complexity. For instance, a simulation element can be a NAND gate, a multiplier, or an SRAM memory. The functionality of one of these complex units can be described using a modern programming language or a dedicated hardware description language. For instance, the THOR simulator uses the C programming language to determine the output values of a module as a function of its inputs [Thor88].

The *SystemC* language [SystemC] uses most of the syntax and semantics of C, but adds a number of constructs and data types to deal with the peculiarities of hardware design—such as the presence of concurrency. On the other hand, *VHDL (VHSIC Hardware Description Language)* [VHDL88] is a specially developed language for the description of hardware designs.

In the *structural mode*, VHDL describes a design as a connection of functional modules. Such a description often is called a *netlist*. For example, Figure C-4 shows a description of a 16-bit accumulator consisting of a register and adder.

The adder and register can in turn be described as a composition of components such as full-adder or register cells. An alternative approach is to use the *behavioral mode* of the language that describes the functionality of the module as a set of input/output relations regardless of the chosen implementation. As an example, Figure C-5 describes how the output of the adder is the two's-complement sum of its inputs.

```
entity accumulator is
    port ( -- definition of input and output terminals
        DI: in bit_vector(15 downto 0) -- a vector of 16 bit wide
        DO: inout bit_vector(15 downto 0);
        CLK: in bit
    );
end accumulator;

architecture structure of accumulator is
    component reg -- definition of register ports
        port (
            DI : in bit_vector(15 downto 0);
            DO : out bit_vector(15 downto 0);
            CLK : in bit
        );
    end component;
    component add -- definition of adder ports
        port (
            IN0 : in bit_vector(15 downto 0);
            IN1 : in bit_vector(15 downto 0);
            OUT0 : out bit_vector(15 downto 0)
        );
    end component;
-- definition of accumulator structure
signal X : bit_vector(15 downto 0);
begin
    add1 : add
        port map (DI, DO, X); -- defines port connectivity
    reg1 : reg
        port map (X, DO, CLK);
end structure;
```

Figure C-4 Functional description of an accumulator in VHDL.

```
entity add is
    port (
        IN0 : in bit_vector(15 downto 0);
        IN1 : in bit_vector(15 downto 0);
        OUT0 : out bit_vector(15 downto 0)
    );
end add;

architecture behavior of add is
begin
    process(IN0, IN1)
        variable C : bit_vector(16 downto 0);
        variable S : bit_vector(15 downto 0);
    begin
        loop1:
        for i in 0 to 15 loop
            S(i) := IN0(i) xor IN1(i) xor C(i);
            C(i+1):= IN0(i) and IN1(i) or C(i) and (IN0(i) or IN1(i));
        end loop loop1;
        OUT0 <= S;
    end process;
end behavior;
```

Figure C-5 Behavioral description of 16-bit adder.

The signal levels of the functional simulator are similar to the switch and logic levels. A variety of timing models can be used—for example, the designer can describe the delay between input and output signals as part of the behavioral description of a module. Most often the zero-delay model is employed, since it yields the highest simulation speed.

C.3 Using Higher-Level Data Models

When conceiving a digital system such as a compact disk player or an embedded microcontroller, the designer rarely thinks in terms of bits. Instead, she envisions data moving over busses as integer or floating-point words, and patterns transmitted over the instruction bus as members of an enumerated set of instruction words (such as {*ACC*, *RD*, *WR*, or *CLR*}). Modeling a discrete design at this level of abstraction has the distinct advantage of being more understandable, and it also results in a substantial benefit in simulation speed. Since a 64-bit bus is now handled as a single object, analyzing its value requires only one action instead of the 64 evaluations it formerly took to determine the current state of the bus at the logic level. The disadvantage of this approach is another sacrifice of timing accuracy. Since a bus is now considered to be a single entity, only one global delay can be annotated to it, while the delay of bus elements can vary from bit to bit at the logic level.

It also is common to distinguish between *functional* (or *structural)* and *behavioral* descriptions. In a functional-level specification, the description mirrors the intended hardware

structure. Behavioral-level specifications only mimic the input/output functionality of a design. Hardware delay loses its meaning, and simulations are normally performed on a per clock-cycle (or higher) basis. For instance, the behavioral models of a microprocessor that are used to verify the completeness and the correctness of the instruction set are performed on a per instruction basis.

The most popular languages at this level of abstraction are the VHDL and VERILOG hardware-description languages. VHDL allows for the introduction of user-defined data types such as 16-bit, two's-complement words or enumerated instruction sets. Many designers tend to use traditional programming approaches such as C or C++ for their first-order behavioral models. This approach has the advantage of offering more flexibility, but it requires the user to define all data types and to essentially write the complete simulation scenario.

Example C.2 Behavioral-Level VHDL Description

To contrast the functional and behavioral description modes and the use of higher level data models, consider again the example of the accumulator (see Figure C-6). In this case, we use a fully behavioral description that employs integer data types to describe the module operation.

Figure C-7 shows the results of a simulation performed at this level of abstraction. Even for this small example, the simulation performance in terms of CPU time is three times better than what is obtained with the structural description of Figure C-4.

```
entity accumulator is
    port (
        DI : in integer;
        DO : inout integer := 0;
        CLK : in bit
    );
end accumulator;

architecture behavior of accumulator is
begin
    process(CLK)
    variable X : integer := 0; -- intermediate variable
    begin
        if CLK = '1' then
            X <= DO + D1;
            DO <= X;
        end if;
    end process;
end behavior;
```

Figure C-6 Accumulator for Example C.2.

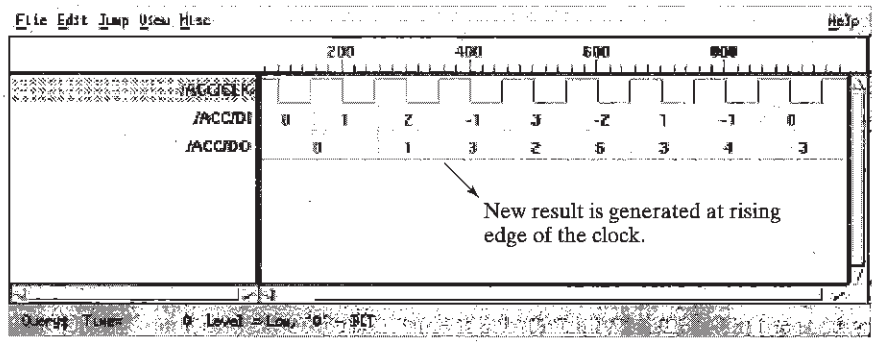


Figure C-7 Display of simulation results for accumulator example as obtained at the behavioral level. The WAVES display tool (and VHDL simulator) are part of the Synopsis VHDL tool suite (Courtesy of Synopsys.).

References

[Agrawal90] P. Agrawal and W. Dally, "A Hardware Logic Simulation System," *IEEE Trans. Computer-Aided Design*, CAD-9, no. 9, pp. 19–29, January 1990.

[Bryant81] R. Bryant, *A Switch-Level Simulation Model for Integrated Logic Circuits*, Ph. D. diss., MIT Laboratory for Computer Science, report MIT/LCS/TR-259, March 1981.

[Salz89] A. Salz and M. Horowitz, "IRSIM: An Incremental MOS Switch-Level Simulator," *Proceedings of the 26th Design Automation Conference*, pp. 173–178, 1989.

[SystemC] *Everything You Wanted to Know about SystemC, http://www.systemc.org/*

[Thor88] R. Alverson et al., "THOR User's Manual," Technical Report CSL-TR-88-348 and 349, Stanford University, January 1988.

[TimeMill] *http://www.synopsys.com/products/mixedsignal/mixedsignal.html*

[VHDL88] VHDL Standards Committee, IEEE Standard VHDL Language Reference Manual, IEEE standard 1076–1077, 1978.

[White87] J. White and A. Sangiovanni-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits*, Kluwer Academic, 1987.

DESIGN METHODOLOGY INSERT

D

Layout Techniques for Complex Gates

Weinberger and standard-cell layout techniques

Euler graph approach

In Chapter 6, we discussed in detail how to construct the schematics of complex gates and how to size the transistors. The last step in the design process is to derive a layout for the gate or cell; in other words, we must determine the exact shape of the various polygons composing the gate layout. The composition of a layout is strongly influenced by the *interconnect approach*. How does the cell fit in the overall chip layout, and how does it communicate with neighboring cells? Keeping these questions in mind from the start results in denser designs with less parasitic capacitance.

Weinberger and Standard-Cell Layout Techniques

We now examine two important layout approaches, although many others can be envisioned. In the *Weinberger approach* [Weinberger67], the data wires (inputs and outputs) are routed (in metal) parallel to the supply rails and perpendicular to the diffusion areas, as illustrated in Figure D-1. Transistors are formed at the cross points of the polysilicon signal wires (connected to the horizontal metal wires) and the diffusion zones. The "over-the-cell" wiring approach makes the Weinberger technique particularly suited for bit-sliced datapaths. While it is still used on an occasional base, the Weinberger technique has lost its appeal over the years in favor of the standard-cell style.

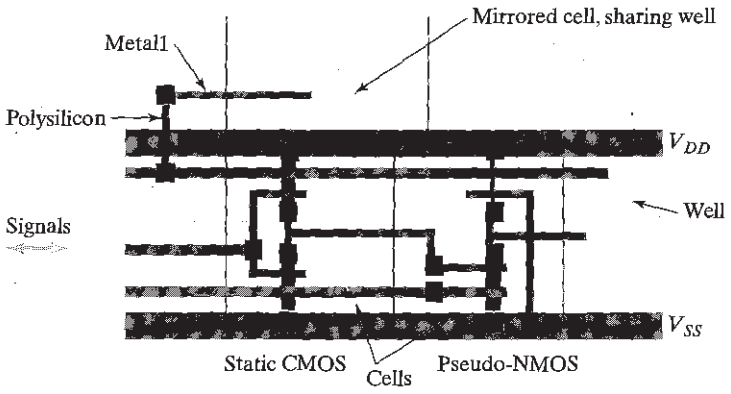


Figure D-1 The Weinberger approach for complex gate layout (using a single metal layer).

In the *standard-cell technique*, signals are routed in polysilicon perpendicular to the power distribution (Figure D-2). This approach tends to result in a dense layout for static CMOS gates, as the vertical polysilicon wire can serve as the input to both the NMOS and the PMOS transistors. An example of a cell implemented using the standard-cell approach is shown in Figure 6-12. Interconnections between cells generally are established in so-called routing channels, as demonstrated in Figure D-2. The standard-cell approach is very popular at present due to its high degree of automation. (For a detailed description of the design automation tools supporting the standard-cell approach, see Chapter 8.)

Layout Planning using the Euler Path Approach

The common use of this layout strategy makes it worth analyzing how a complex Boolean function can be mapped efficiently onto such a structure. For density reasons, it is desirable to realize the NMOS and PMOS transistors as an unbroken row of devices with abutting source–drain con-

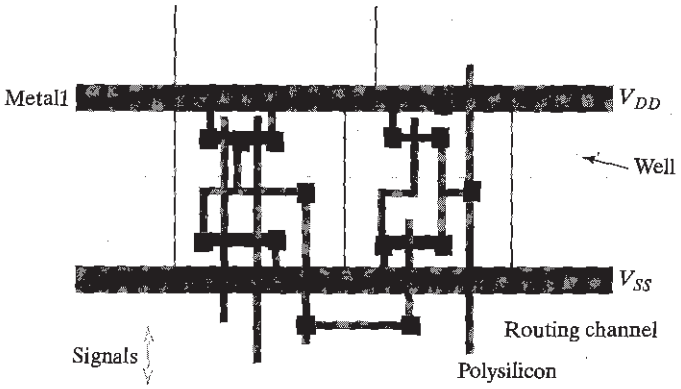


Figure D-2 The standard-cell approach for complex gate layout.

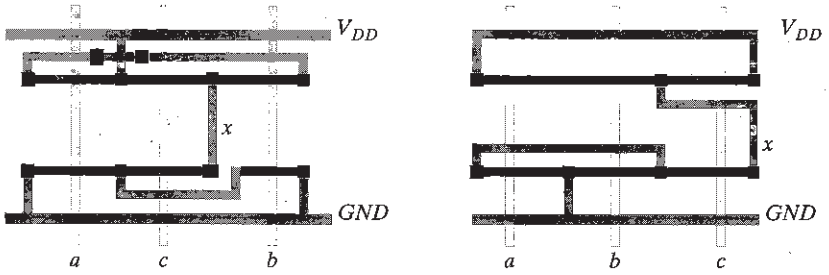
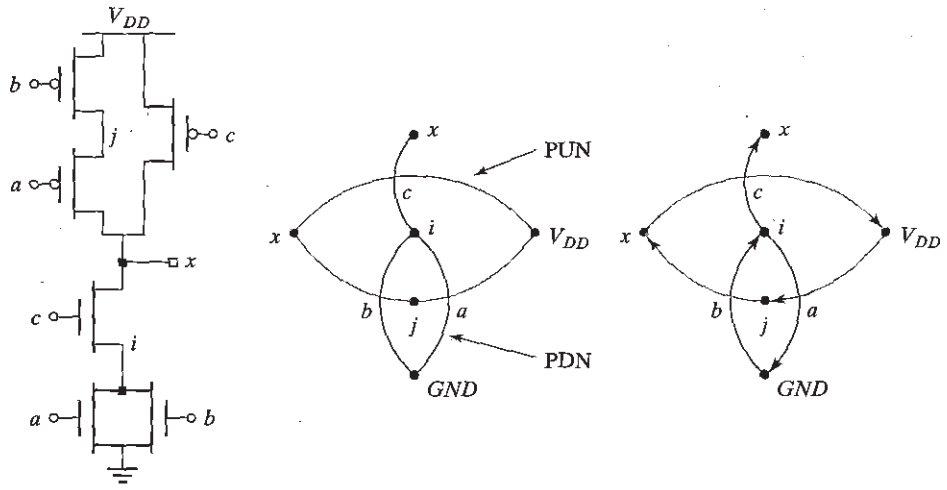


Figure D-3 Stick Diagram for $x = (a + b) \cdot c$.

nections, and with the gate connections of the corresponding NMOS and PMOS transistors aligned. This approach requires only a single strip of diffusion in both wells. To achieve this goal, a careful ordering of the input terminals is necessary. This is illustrated in Figure D-3, where the logical function $\bar{x} = (a + b) \cdot c$ is implemented. In the first version, the order $\{a\ c\ b\}$ is adopted. It can easily be seen that no solution will be found using only a single diffusion strip. A reordering of the terminals (for instance, using $\{a\ b\ c\}$), generates a feasible solution, as shown in Figure D-3b. Observe that the "layouts" in Figure D-3 do not represent actual mask geometries, but are rather symbolic diagrams of the gate topologies. Wires and transistors are represented as dimensionless objects, and positioning is relative, not absolute. Such conceptual representations are called *stick diagrams*, and often are used at the conception time of the gate, before determining the actual dimensions. We use stick diagrams whenever we want to discuss gate topologies or layout strategies.

Fortunately, a systematic approach has been developed to derive the permutation of the input terminals so that complex functions can be realized by uninterrupted diffusion strips that minimize the area [Uehara81]. The systematic nature of the technique also has the advantage that it is easily automated. It consists of the following two steps:

1. **Construction of *logic graph*.** The logic graph of a transistor network (or a switching function) is the graph of which the vertices are the nodes (signals) of the network, and the edges represent the transistors. Each edge is named for the signal controlling the corresponding transistor. Since the PUN and PDN networks of a static CMOS gate are dual, their corresponding graphs are dual as well—that is, a parallel connection is replaced by a series one and vice versa. This is demonstrated in Figure D-4, where the logic graphs for the PDN and PUN networks of the Boolean function $\bar{x} = (a + b) \cdot c$ are overlaid (notice that this approach can be used to derive dual networks).
2. **Identification of Euler paths.** An Euler path in a graph is defined as a path through all nodes in the graph such that each edge in the graph is only visited once. Identification of such a path is important, because an ordering of the inputs leading to an uninterrupted diffusion strip of NMOS (PMOS) transistors is possible only if there exists an Euler path in



(a) Schematic diagram (b) Logic graphs for PDN and PUN (c) Consistent Euler paths for PDN and PUN

Figure D-4 Schematic diagram, logic graph, and Euler paths for $x = (a + b) \cdot c$.

the logic graph of the PDN (PUN) network. The reasoning behind this finding is as follows:

To form an interrupted strip of diffusion, all transistors must be visited in sequence; that is, the drain of one device is the source of the next one. This is equivalent to traversing the logic graph along an Euler path. Be aware that Euler paths are not unique: many different solutions may exist.

The sequence of edges in the Euler path equals the ordering of the inputs in the gate layout. To obtain the same ordering in both the PUN and PDN networks, as is necessary if we want to use a single poly strip for every input signal, the Euler paths must be *consistent*—that is, they must have the same sequence.

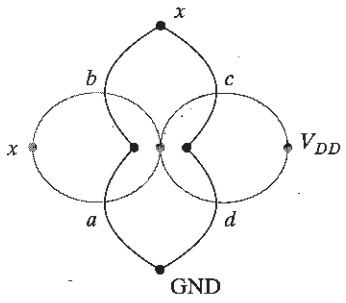
Consistent Euler paths for the example of Figure D-4a are shown in Figure D-4c. The layout associated with this solution is shown in Figure D-3b. An inspection of the logic diagram of the function shows that $\{a\ c\ b\}$ is an Euler path for the PUN, but not for the PDN. A single-diffusion-strip solution is, hence, nonexistent (Figure D-3a).

Example D.1 Derivation of Layout Topology of Complex Logic Gate

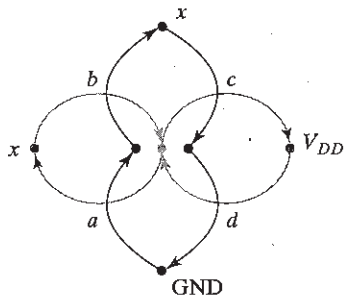
As an example, let us derive the layout topology of the following logical function:

$$x = \overline{ab + cd}$$

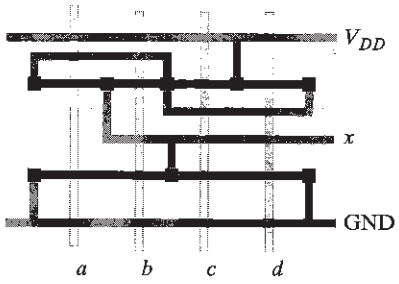
The logical function and one consistent Euler path are shown in Figure D-5a and Figure D-5b. The corresponding layout is shown in Figure D-5c.



(a) Logic graphs for $\overline{(ab + cd)}$



(b) Euler paths $\{a\ b\ c\ d\}$



(c) Stick diagram for ordering $\{a\ b\ c\ d\}$

Figure D-5 Deriving the layout topology for $x = (ab + cd)$.

The reader should be aware that the existence of consistent Euler paths depends on the way the Boolean expressions (and the corresponding logic graphs) are constructed. For example, no consistent Euler paths can be found for $\overline{x} = a + b \cdot c + d \cdot e$, but the function $\overline{x} = b \cdot c + a + d \cdot e$ has a simple solution (confirm that this is true by preserving the ordering of the function when constructing the logic graphs). A restructuring of the function is sometimes necessary before a set of consistent paths can be identified. This could lead to an exhaustive search over all possible path combinations. Fortunately, a simple algorithm to avoid this plight has been proposed [Uehara81]. A discussion of this is beyond our scope, however, and we refer the interested reader to that text.

Finally, it is worth mentioning that the layout strategies presented are not the only possibilities. For example, sometimes it might be more effective to provide multiple diffusion strips stacked vertically. In this case, a single polysilicon input line can serve as the input for multiple transistors. This might be beneficial for certain gate structures, such as the NXOR gate, and therefore, case-by-case analysis is recommended.

To Probe Further

A good overview of cell-generation techniques can be found in [Rubin87, pp. 116–128]. Some of the landmark papers in this area include the following:

[Clein00] D. Clein, *CMOS IC Layout*, Newnes, 2000.

[Rubin87] S. Rubin, *Computer Aids for VLSI Design*, Addison-Wesley, 1987.

[Uehara81] T. Uehara and W. Van Cleemput, "Optimal Layout of CMOS Functional Arrays," *IEEE Trans. on Computers*, vol. C-30, no. 5, pp. 305–311, May 1981.

[Weinberger67] A. Weinberger, "Large Scale Integration of MOS Complex Logic: A Layout Method," *IEEE Journal of Solid State Circuits*, vol. 2, no. 4, pp. 182–190, 1967.

SECOND EDITION

DIGITAL INTEGRATED CIRCUITS

A DESIGN PERSPECTIVE

JAN M. RABAEY ■ ANANTHA CHANDRAKASAN
BORIVOJE NIKOLIĆ

Since the publication of the first edition of this book in 1996, CMOS manufacturing technology has continued its breathtaking pace, scaling to ever-smaller dimensions. Minimum feature sizes are now reaching the 100-nm realm. Circuits are becoming more complex, challenging the productivity of the designer, while the plunge into the deep-submicron space causes devices to behave differently and brings to the forefront a number of new issues that impact the reliability, cost, performance, power dissipation, and reliability of the digital IC. This updated text reflects the ongoing (r)evolution in the world of digital integrated circuit design, caused by this move into the deep-submicron realm. This means increased importance of deep-submicron transistor effects, interconnect, signal integrity, high-performance and low-power design, timing, and clock distribution. In contrast to the first edition, the present text focuses entirely on CMOS ICs.

http://bwrc.eecs.berkeley.edu/IcBook—A Dynamic Companion

Even more than for the first edition, this book uses its companion website to evolve and grow over time. It contains complete Microsoft PowerPoint presentations covering all the material, updates, corrections, design projects, and extensive instructor material. Most importantly, all problem sets are now available on the website (and have been removed from the text).

Outstanding Features of the Text

- It focuses solely on deep-submicron CMOS devices, the workhorses of today's digital integrated circuits. A simple transistor model for manual analysis, called the unified MOS model, has been developed and is used throughout.
- Design Examples stress the design of Digital ICs from a real-world perspective. Design challenges and guidelines are highlighted. 0.25-micron CMOS technology is used for all the examples and problems.
- Design Methodology inserts are interspersed throughout the text, highlighting the importance of methodology and tools in today's design process.
- A Perspective section at the end of each chapter gives an insight into future evolutions.

About the Cover: Detail of "Wet Orange," by Joan Mitchell (American, 1925–1992). Oil on canvas, 112 × 245 in. (284.5 × 622.3 cm). Carnegie Museum of Art, Pittsburgh, PA. Gift of Kaufmann's Department Store and

Pearson Education
Prentice Hall
Upper Saddle River, NJ 07458
www.prenhall.com

Prentice Hall
SECOND EDITION
DIGITAL INTEGRATED CIRCUITS
A DESIGN PERSPECTIVE
RABAEY
CHANDRAKASAN
NIKOLIĆ