Digital Design

Principles and Practices

Fourth Edition

John F. Wakerly

DIGITAL DESIGN

Principles and Practices

Fourth Edition

John F. Wakerly

Cisco Systems, Inc.
Stanford University

PEARSON Prentice Hall Upper Saddle River, New Jersey 07458

Wakerly, John F.
Digital design: principles and practices/ John F. Wakerly.--4th ed.
p. cm.
Includes index
ISBN 0-13-186389-4
1. Digital integrated circuits--Design and construction. I. Title
TK7874.65,W34 2005
621.39'5-dc22
2005048710

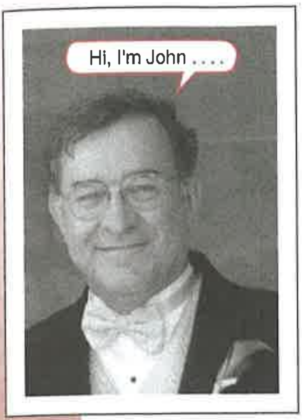
Vice President and Editorial Director, ECS: *Marcia Horton*
Editorial Assistant: *Richard Virginia*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *David A. George*
Production Editor: *Scott Disanno*
Director of Creative Services: *Paul Belfanti*
Art Director: *Kenny Beck*
Cover Designer: *Bruce Kenselaar*
Art Editor: *Xiaohong Zhu*
Manufacturing Manager: *Alexis Heydt-Long*
Manufacturing Buyer: *Lisa McDowell*
Senior Marketing Manager: *Holly Stark*
About the Cover: Original cover artwork © 2001 Ken Bakeman, www.kennyzen.com

ISBN 0-13-186389-4

11 12 13 14 15 16 17 18 19 20 V092 18 17 16 15

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

CONTENTS



chapter 1

Introduction

Welcome to the world of digital design. Perhaps you're a computer science student who knows all about computer software and programming, but you're still trying to figure out how all that fancy hardware could possibly work. Or perhaps you're an electrical engineering student who already knows something about analog electronics and circuit design, but you wouldn't know a bit if it bit you. No matter. Starting from a fairly basic level, this book will show you how to design digital circuits and subsystems.

We'll give you the basic principles that you need to figure things out, and we'll give you lots of examples. Along with principles, we'll try to convey the flavor of real-world digital design by discussing current, practical considerations whenever possible. And I, the author, will often refer to myself as "we" in the hope that you'll be drawn in and feel that we're walking through the learning process together.

1.1 About Digital Design

Some people call it "logic design." That's OK, but ultimately the goal of design is to build systems. To that end, we'll cover a whole lot more in this text than logic equations and theorems.

This book claims to be about principles and practices. Most of the principles that we present will continue to be important years from now;

some may be applied in ways that have not even been discovered yet. As for practices, they may be a little different from what's presented here by the time you start working in the field, and they will certainly continue to change throughout your career. So you should treat the "practices" material in this book as a way to reinforce principles, and as a way to learn design methods by example.

One of the book's goals is to present enough about basic principles for you to know what's happening when you use software tools to "turn the crank" for you. The same basic principles can help you get to the root of problems when the tools happen to get in your way.

Listed in the box on this page are several key points that you should learn through your studies with this text. Most of these items probably make no sense to you right now, but you should come back and review them later.

Digital design is engineering, and engineering means "problem solving." My experience is that only 5%–10% of digital design is "the fun stuff"—the creative part of design, the flash of insight, the invention of a new approach. Much of the rest is just "turning the crank." To be sure, turning the crank is much easier now than it was 25 or even 10 years ago, but you still can't spend 100% or even 50% of your time on the fun stuff.

IMPORTANT THEMES IN DIGITAL DESIGN

- Good tools do not guarantee good design, but they help a lot by taking the pain out of doing things right.
- Digital circuits have analog characteristics.
- Know when to worry and when not to worry about the analog aspects of digital design.
- Always document your designs to make them understandable to yourself and to others.
- Use consistent coding, organizational, and documentation styles in your HDL-based designs, following your company's guidelines.
- Understand and use standard functional building blocks.
- State-machine design is like programming; approach it that way.
- Design for minimum cost at the system level, including your own engineering effort as part of the cost.
- Design for testability and manufacturability.
- Use programmable logic to simplify designs, reduce cost, and accommodate last-minute modifications.
- Avoid asynchronous design. Practice synchronous design until a better methodology comes along (if ever).
- Pinpoint the unavoidable asynchronous interfaces between different subsystems and the outside world, and provide reliable synchronizers.

et. As for
y the time
to change
ı this book
ıethods by

les for you
crank" for
ıs when the

ıould learn
‹e no sense

n solving."
stuff'—the
ı approach.
ınk is much
nd 100% or

ȝ the pain

of digital

ıelf and to

our HDL-

ngineering

ıodate last-

er method-

ıubsystems

Besides the fun stuff and turning the crank, there are many other areas in which a successful digital designer must be competent, including the following:

- *Debugging.* It's next to impossible to be a good designer without being a good troubleshooter. Successful debugging takes planning, a systematic approach, patience, and logic: if you can't discover where a problem *is*, find out where it *is not*!
- *Business requirements and practices.* A digital designer's work is affected by a lot of nonengineering factors, including documentation standards, component availability, feature definitions, target specifications, task scheduling, office politics, and going to lunch with vendors.
- *Risk-taking.* When you begin a design project you must carefully balance risks against potential rewards and consequences, in areas ranging from component selection (will it be available when I'm ready to build the first prototype?) to schedule commitments (will I still have a job if I'm late?).
- *Communication.* Eventually, you'll hand off your successful designs to other engineers, other departments, and customers. Without good communication skills, you'll never complete this step successfully. Keep in mind that communication includes not just transmitting but also receiving; learn to be a good listener!

In the rest of this chapter, and throughout the text, I'll continue to state some opinions about what's important and what is not. I think I'm entitled to do so as a moderately successful practitioner of digital design.

Additional materials related to this book, such as supplemental chapter sections, selected exercise solutions, and downloadable source code for all programs, can be found at DDPPonline (via `www.ddpp.com`; see the Preface).

1.2 Analog versus Digital

Analog devices and systems process time-varying signals that can take on any value across a continuous range of voltage, current, or other metric. So do *digital* circuits and systems; the difference is that we can pretend that they don't! A digital signal is modeled as taking on, at any time, only one of two discrete values, which we call *0* and *1* (or LOW and HIGH, FALSE and TRUE, negated and asserted, Frank and Teri, or whatever).

analog
digital

0
1

Digital computers have been around since the 1940s and have been in widespread commercial use since the 1960s. Yet only in the past 10 to 20 years has the "digital revolution" spread to many other aspects of life. Examples of once-analog systems that have now "gone digital" include the following:

- *Still pictures.* Ten years ago, the majority of cameras still used silver-halide film to record images. Today, inexpensive digital cameras record a picture as a 1024×768 or larger array of pixels, where each pixel stores the intensities of its red, green, and blue color components as 8 or more bits each. This data, over 18 million bits in this example, is processed and compressed in JPEG format down to as few as 5% of the original number of bits. So, digital cameras rely on both digital storage and digital processing.

- *Video recordings.* A digital versatile disc (DVD) stores video in a highly compressed digital format called MPEG-2. This standard encodes a small fraction of the individual video frames in a compressed format similar to JPEG, and encodes each other frame as the difference between it and the previous one. The capacity of a single-layer, single-sided DVD is about 35 billion bits, sufficient for about 2 hours of high-quality video, and a two-layer, double-sided disc has four times that capacity.
- *Audio recordings.* Once made exclusively by impressing analog waveforms onto vinyl or magnetic tape, audio recordings now commonly use digital compact discs (CDs). A CD stores music as a sequence of 16-bit numbers corresponding to samples of the original analog waveform, one sample per stereo channel every 22.7 microseconds. A full-length CD recording (73 minutes) contains over 6 billion bits of information.
- *Automobile carburetors.* Once controlled strictly by mechanical linkages (including clever "analog" mechanical devices that sensed temperature, pressure, etc.), automobile engines are now controlled by embedded microprocessors. Various electronic and electromechanical sensors convert engine conditions into numbers that the microprocessor can examine to determine how to control the flow of fuel and oxygen to the engine. The microprocessor's output is a time-varying sequence of numbers that operate electromechanical actuators which, in turn, control the engine.
- *The telephone system.* It started out over a hundred years ago with analog microphones and receivers connected to the ends of a pair of copper wires (or was it string?). Even today, most homes still use analog telephones, which transmit analog signals to the phone company's central office (CO). However, in the majority of COs, these analog signals are converted into a digital format before they are routed to their destinations, be they in the same CO or across the world. For many years the private branch exchanges (PBXs) used by businesses have carried the digital format all the way to the desktop. Now many businesses, COs, and traditional telephony service providers are converting to integrated systems that combine digital voice with data traffic over a single IP (Internet Protocol) network.
- *Traffic lights.* Stop lights used to be controlled by electromechanical timers that would give the green light to each direction for a predetermined amount of time. Later, relays were used in controllers that could activate the lights according to the pattern of traffic detected by sensors embedded in the pavement. Today's controllers use microprocessors and can control

the lights in ways that maximize vehicle throughput or, in Sunnyvale, California, frustrate drivers with all kinds of perverse behavior.

- *Movie effects.* Special effects used to be created exclusively with miniature clay models, stop action, trick photography, and numerous overlays of film on a frame-by-frame basis. Today, spaceships, bugs, otherworldly scenes, and even babies from hell (in Pixar's animated short *Tin Toy*) are synthesized entirely using digital computers. Might the stunt man or woman someday no longer be needed, either?

The electronics revolution has been going on for quite some time now, and the "solid-state" revolution began with analog devices and applications like transistors and transistor radios. So why has there now been a *digital* revolution? There are in fact many reasons to favor digital circuits over analog ones:

- *Reproducibility of results.* Given the same set of inputs (in both value and time sequence), a properly designed digital circuit always produces exactly the same results. The outputs of an analog circuit vary with temperature, power-supply voltage, component aging, and other factors.
- *Ease of design.* Digital design, often called "logic design," is logical. No special math skills are needed, and the behavior of small logic circuits can be visualized mentally without any special insights about the operation of capacitors, transistors, or other devices that require calculus to model.
- *Flexibility and functionality.* Once a problem has been reduced to digital form, it can be solved using a set of logical steps in space and time. For example, you can design a digital circuit that scrambles your recorded voice so that it is absolutely indecipherable by anyone who does not have your "key" (password), but it can be heard virtually undistorted by anyone who does. Try doing that with an analog circuit.
- *Programmability.* You're probably already quite familiar with digital computers and the ease with which you can design, write, and debug programs for them. Well, guess what? Much of digital design is carried out today by writing programs, too, in *hardware description languages (HDLs)*. These languages allow both structure and function of a digital circuit to be specified or *modeled*. Besides a compiler, a typical HDL also comes with simulation and synthesis programs. These software tools are used to test the hardware model's behavior before any real hardware is built, and then to synthesize the model into a circuit in a particular component technology.

hardware description language (HDL)

hardware model

- *Speed.* Today's digital devices are very fast. Individual transistors in the fastest integrated circuits can switch in less than 10 picoseconds, and a complete, complex device built from these transistors can examine its inputs and produce an output in less than a nanosecond. This means that such a device can produce a billion or more results per second.

SHORT TIMES A *millisecond (ms)* is 10^{-3} second, and a *microsecond (μs)* is 10^{-6} second. A *nanosecond (ns)* is just 10^{-9} second, and a *picosecond (ps)* is 10^{-12} second. In a vacuum, light travels about a foot in a nanosecond, and an inch in 85 picoseconds. With individual transistors in the fastest integrated circuits now switching in less than 10 picoseconds, the speed-of-light delay between these transistors across a half-inch-square silicon chip has become a limiting factor in circuit design.

- *Economy.* Digital circuits can provide a lot of functionality in a small space. Circuits that are used repetitively can be "integrated" into a single "chip" and mass-produced at very low cost, making possible throw-away items like calculators, digital watches, and singing birthday cards. (You may ask, "Is this such a good thing?" Never mind!)
- *Steadily advancing technology.* When you design a digital system, you almost always know that there will be a faster, cheaper, or otherwise better technology for it in a few years. Clever designers can accommodate these expected advances during the initial design of a system, to forestall system obsolescence and to add value for customers. For example, desktop computers often have "expansion sockets" to accommodate faster processors or larger memories than are available at the time of the computer's introduction.

So, that's enough of a sales pitch on digital design. The rest of this chapter will give you a bit more technical background to prepare you for the rest of the book.

1.3 Digital Devices

gate

The most basic digital devices are called *gates*, and no, they were not named after the founder of a large software company. Gates originally got their name from their function of allowing or retarding ("gating") the flow of digital information. In general, a gate has one or more inputs and produces an output that is a function of the current input value(s). While the inputs and outputs may be analog conditions such as voltage, current, even hydraulic pressure, they are modeled as taking on just two discrete values, 0 and 1.

AND gate

Figure 1-1 shows symbols for the three most important kinds of gates. A 2-input *AND gate*, shown in (a), produces a 1 output if both of its inputs are 1; otherwise it produces a 0 output. The figure shows the same gate four times, with the four possible combinations of inputs that may be applied to it and the resulting outputs. A gate is called a *combinational circuit* because its output depends only on the current combination of input values (called an *input combination*).

combinational circuit
input combination
OR gate

A 2-input *OR gate*, shown in (b), produces a 1 output if one or both of its inputs are 1; it produces a 0 output only if both inputs are 0. Once again, there are four possible input combinations, resulting in the outputs shown in the figure.

cond. A
ond. In a
oseconds.
g in less
across a

in a small
nto a single
throw-away
cards. (You

system, you
rwise better
nodate these
estall system
esktop com-
r processors
computer's

chapter will
of the book.

re not named
ot their name
ow of digital
ices an output
d outputs may
ssure, they are

ds of gates. A
s inputs are 1;
our times, with
and the result-
output depends
combination).
e or both of its
again, there are
in the figure.

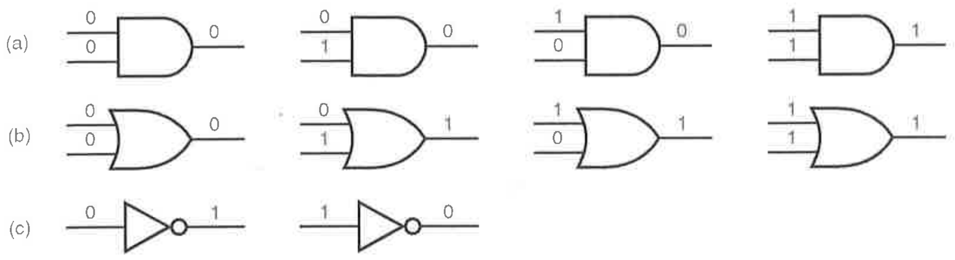


Figure 1-1 Digital devices: (a) AND gate; (b) OR gate; (c) NOT gate or inverter.

A *NOT gate*, more commonly called an *inverter*, produces an output value that is the opposite of the input value, as shown in (c). *NOT gate* *inverter*

We called these three gates the most important for good reason. Any digital function can be realized using just these three kinds of gates. In Chapter 3 we'll show how gates are realized using transistor circuits. You should know, however, that gates have been built or proposed using other technologies, such as relays, vacuum tubes, hydraulic devices, and molecular structures.

A *flip-flop* is a device that stores either a 0 or 1. The *state* of a flip-flop is the value that it currently stores. The stored value can be changed only at certain times determined by a "clock" input, and the new value may further depend on the flip-flop's current state and its "control" inputs. A flip-flop can be built from a collection of gates hooked up in a clever way, as we'll show in Section 7.2. *flip-flop* *state*

A digital circuit that contains flip-flops is called a *sequential circuit*, because its output at any time depends not only on its current input but also on the past sequence of inputs that have been applied to it. In other words, a sequential circuit has *memory* of past events. *sequential circuit* *memory*

1.4 Electronic Aspects of Digital Design

Digital circuits are not exactly a binary version of alphabet soup—with all due respect to Figure 1-1, they don't have little 0s and 1s floating around in them. As we'll see in Chapter 3, digital circuits deal with analog voltages and currents and are built with analog components. The "digital abstraction" allows analog behavior to be ignored in most cases, so circuits can be modeled as if they really did process 0s and 1s.

One important aspect of the digital abstraction is to associate a *range* of analog values with each logic value (0 or 1). As shown in Figure 1-2 on the next page, a typical gate is not guaranteed to have a precise voltage level for a logic 0 output. Rather, it may produce a voltage somewhere in a range that is a *subset* of the range guaranteed to be recognized as a 0 by other gate inputs. The difference between the range boundaries is called *noise margin*—in a real circuit, a gate's output can be corrupted by this much noise and still be correctly interpreted at the inputs of other gates. *noise margin*

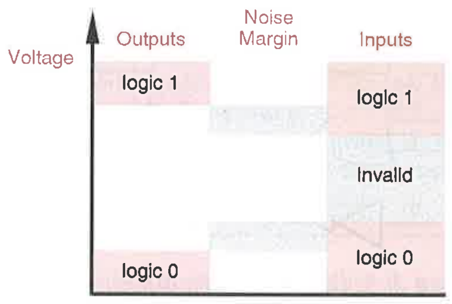


Figure 1-2
Logic values and noise margins.

Behavior for logic 1 outputs is similar. Note in the figure that there is an "invalid" region between the input ranges for logic 0 and logic 1. Although any given digital device operating at a particular voltage and temperature will have a fairly well defined boundary (or threshold) between the two ranges, different devices may have different boundaries. Still, all properly operating devices have their boundary *somewhere* in the "invalid" range. Therefore, any signal that is within the defined ranges for 0 and 1 will be interpreted identically by different devices. This characteristic is essential for reproducibility of results.

It is the job of an *electronic* circuit designer to ensure that logic gates produce and recognize logic signals that are within the appropriate ranges. This is an analog circuit-design problem; we touch upon some of its aspects in Chapter 3. It's not possible to design a circuit that has the desired behavior under every possible condition of power-supply voltage, temperature, loading, and other factors. Instead, the electronic circuit designer or device manufacturer provides *specifications* (also known as *specs*) that define the conditions under which correct behavior is guaranteed.

specifications (specs)

As a *digital* designer, then, you need not delve into the detailed analog behavior of a digital device to ensure its correct operation. Rather, you need only study enough about the device's operating environment to determine that it is operating within its published specifications. Granted, some analog knowledge is needed to perform this study, but not nearly what you'd need to design a digital device starting from scratch. In Chapter 3 we'll give you just what you need.

1.5 Software Aspects of Digital Design

Digital design need not involve any software tools. For example, Figure 1-3 shows the primary tool of the "old school" of digital design—a plastic template for drawing logic symbols in schematic diagrams by hand (the designer's name was engraved into the plastic with a soldering iron).

Today, however, software tools are an essential part of digital design. Indeed, the availability and practicality of hardware description languages (HDLs) and accompanying circuit simulation and synthesis tools have changed uts

ic 1

alid

ic 0

t there is an
lthough any
e will have a
es, different
devices have
signal that is
by different
s.
t logic gates
ranges. This
ts aspects in
havior under
loading, and
manufacturer
ditions under

tailed analog
ou need only
nine that it is
g knowledge
esign a digital
you need.

le, Figure 1-3
astic template
signer's name

ligital design.
on languages
have changed

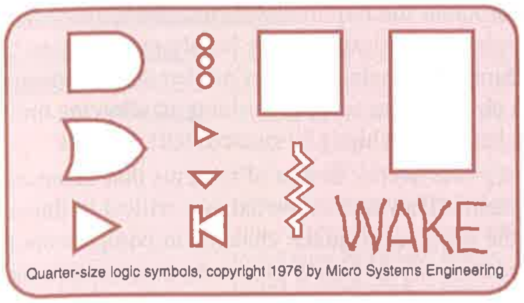

WAKE
Quarter-size logic symbols, copyright 1976 by Micro Systems Engineering

Figure 1-3 A logic-design template.

the entire landscape of digital design over the past several years. We'll make extensive use of HDLs throughout this book.

In *computer-aided design* (also called *computer-aided engineering*, *CAE*), various software tools improve the designer's productivity and help to improve the correctness and quality of designs. In a competitive world, the use of software tools is mandatory to obtain high-quality results on aggressive schedules. Important examples of software tools for digital design are listed below:

computer-aided design (CAD)

computer-aided engineering (CAE)

- *Schematic entry.* This is the digital designer's equivalent of a word processor. It allows schematic diagrams to be drawn "on-line," instead of with paper and pencil. The more advanced schematic-entry programs also check for common, easy-to-spot errors, such as shorted outputs, signals that don't go anywhere, and so on. Such programs are discussed in greater detail in Section CAD.2 at DDPPonline.
- *HDLs.* Hardware description languages, originally developed for circuit modeling, are now being used extensively for hardware *design*. They can be used to design anything from individual function modules to large, multichip digital systems. We'll introduce three commonly used HDLs, ABEL, VHDL, and Verilog, in Chapter 5, and we'll provide examples in later chapters. (Don't worry, you needn't learn all three languages!)
- *HDL text editors, compilers, and synthesizers.* A typical HDL software package has many components. The designer uses a text editor to write an HDL "program," and an HDL compiler checks it for syntax and related errors. The designer then can give the program to a synthesizer that creates a corresponding circuit realization that is targeted to a particular hardware technology. Most often, though, before synthesis, the designer runs the HDL program on a "simulator" to verify the behavior of the design.
- *Simulators.* The design cycle for a customized, single-chip digital integrated circuit is long and expensive. Once the first chip is built, it's very difficult, often impossible, to debug it by probing internal connections (they are really tiny), or to change the gates and interconnections. Usually, changes must be made in the original design database and a new chip must

be manufactured to incorporate the required changes. Since this process can take months to complete, chip designers are highly motivated to "get it right" on the first try. Simulators help designers predict the electrical and functional behavior of a chip without actually building it, allowing most if not all bugs to be found before the chip is fabricated.

- Simulators are also used in the overall design of systems that incorporate many individual components. They are somewhat less critical in this case because it's easier for the designer to make changes in components and interconnections on a printed-circuit board. However, even a little bit of simulation can save time by catching mistakes early.
- *Test benches.* HDL-based digital designs are simulated and tested in software environments called "test benches." The idea is to build a set of programs around the HDL programs to automatically exercise them, checking both their functional and timing behavior. This is especially handy when small design changes are made—the test bench can be run to ensure that bug fixes or "improvements" in one area do not break something else. Test-bench programs may be written in the same HDL as the design itself, in C or C++, or in a combination of languages including scripting languages like Perl.
- *Timing analyzers and verifiers.* The time dimension is very important in digital design. All digital circuits take time to produce a new output value in response to an input change, and much of a designer's effort is spent ensuring that such output changes occur quickly enough (or, in some cases, not too quickly). Specialized programs can automate the tedious task of drawing timing diagrams and specifying and verifying the timing relationships between different signals in a complex system.
- *Word processors.* HDL-specific text editors are useful for writing source code, but word processors supporting fancy fonts and pretty graphics also have an important use in every design—to create documentation!

In addition to using the tools above, designers may sometimes write specialized programs in high-level languages like C or C++, or scripts in languages like Perl, to solve particular design problems. For example, Section 9.1.6 gives a couple of examples of C programs that generate the "truth tables" for complex combinational logic functions.

Although CAD tools are important, they don't make or break a digital designer. To take an analogy from another field, you couldn't consider yourself to be a great writer just because you're a fast typist or very handy with a word processor. During your study of digital design, be sure to learn and use all the tools that are available to you, such as schematic-entry programs, simulators, and HDL compilers. But remember that learning to use tools is no guarantee that you'll be able to produce good results. Please pay attention to what you're producing with them!

is process
ed to "get
ctrical and
ng most if

ncorporate
n this case
onents and
ittle bit of

ted in soft-
ld a set of
cise them,
especially
n be run to
reak some-
HDL as the
s including

mportant in
output value
fort is spent
some cases,
edious task
the timing

iting source
graphics also
on!

etimes write
or scripts in
or example,
ate the "truth

eak a digital
ider yourself
with a word
nd use all the
s, simulators,
guarantee that
what you're

PROGRAMMABLE LOGIC DEVICES VERSUS SIMULATION

Later in this book you'll learn how programmable logic devices (PLDs) and field-programmable gate arrays (FPGAs) allow you to design a circuit or subsystem by writing a sort of program. PLDs and FPGAs are now available with up to tens of millions of gates, and the capabilities of these technologies are ever increasing. If a PLD- or FPGA-based design doesn't work the first time, you can often fix it by changing the program and physically reprogramming the device, without changing any components or interconnections at the system level. The ease of prototyping and modifying PLD- and FPGA-based systems can eliminate the need for simulation in board-level design; simulation is required only for chip-level designs.

The most widely held view in industry trends says that as chip technology advances, more and more design will be done at the chip level, rather than the board level. Therefore, the ability to perform complete and accurate simulation will become increasingly important to the typical digital designer.

However, another view is possible. The past decade has seen the emergence of programmable devices that include not only gates and flip-flops as building blocks, but also higher-level functions such as microprocessors, memories, and input/output controllers. Using these devices, the digital designer is relieved of the need to design the most complex and critical on-chip components and interconnections—they have already been designed and tested by the device manufacturer.

In this view, it is still possible to misapply high-level programmable functions, but it is also possible to fix mistakes simply by changing a program; detailed simulation of a design before simply "trying it out" could be a waste of time. Another, compatible view is that the PLD or FPGA is merely a full-speed simulator for the program, and this full-speed simulator is what gets shipped in the product!

Does this extreme view have any validity? To guess the answer, ask yourself the following question. How many software programmers do you know who debug a new program by "simulating" its operation rather than just trying it out?

In any case, modern digital systems are much too complex for a designer to have any chance of testing every possible input condition, with or without simulation. As with software, correct operation of digital systems is best accomplished through practices that ensure that the systems are "correct by design." It is a goal of this text to encourage such practices.

1.6 Integrated Circuits

A collection of one or more gates fabricated on a single silicon chip is called an *integrated circuit (IC)*. Large ICs with tens of millions of transistors may be half an inch or more on a side, while small ICs may be less than one-tenth of an inch on a side.

integrated circuit (IC)

Regardless of its size, an IC is initially part of a much larger, circular *wafer*, up to ten inches in diameter, containing dozens to hundreds of replicas of the same IC. All of the IC chips on the wafer are fabricated at the same time, like pizzas that are eventually sold by the slice, except in this case, each piece (IC

wafer

die
pad

chip) is called a *die*. Each die has *pads* around its periphery—electrical contact points that are much larger than other chip features, so wires can be connected later. After the wafer is fabricated, the dice are tested in place on the wafer using tiny, probing pins that temporarily contact the pads, and defective dice are marked. Then the wafer is sliced up to produce the individual dice, and the marked ones are discarded. (Compare with the pizza-maker who sells all the pieces, even the ones without enough pepperoni!) Each "good" die is mounted in a package, its pads are wired to the package pins, the packaged IC is subjected to a final test, and it is shipped to a customer.

Some people use the term "IC" to refer to a silicon die. Some use "chip" to refer to the same thing. Still others use "IC" or "chip" to refer to the combination of a silicon die and its package. Digital designers tend to use the two terms interchangeably, and they really don't care what they're talking about. They don't require a precise definition, since they're only looking at the functional and electrical behavior of these things. In the balance of this text, we'll use the term *IC* to refer to a packaged die.

IC

In the early days of integrated circuits, ICs were classified by size—small, medium, or large—according to how many gates they contained. The simplest type of commercially available ICs are still called *small-scale integration (SSI)* and contain the equivalent of 1 to 20 gates. SSI ICs typically contain a handful of gates or flip-flops, the basic building blocks of digital design.

small-scale integration (SSI)

The SSI ICs that you might encounter in an educational lab come in a 14-pin *dual inline-pin (DIP)* package. As shown in Figure 1-4(a), the spacing between pins in a column is 0.1 inch and the spacing between columns is 0.3 inch. Larger DIP packages accommodate functions with more pins, as shown in (b) and (c). A *pin diagram* shows the assignment of device signals to package pins, or *pinout*. Figure 1-5 shows the pin diagrams for a few common SSI ICs. Such diagrams are used only for mechanical reference, when a designer needs to determine the pin numbers for a particular IC. In the schematic diagram for a digital circuit, pin diagrams are not used. Instead, the various gates are grouped functionally, as we'll show in Section 6.1.

dual inline-pin (DIP) package

pin diagram
pinout

NOT A DICEY DECISION

A reader of a previous edition wrote to me to collect a $5 reward for pointing out my "glaring" misuse of "dice" as the plural of "die." According to the dictionary, she said, the plural form of "die" is "dice" *only* when describing those little cubes with dots on each side; otherwise it's "dies," and she produced the references to prove it.

Being stubborn, I recently did some searching using Google. It reported 259 web pages with the term "integrated-circuit dice" and only 113 with "integrated-circuit dies." None of the 259 pages appeared to be about a game-of-chance-on-a-chip, while at least one of the 113 was actually talking about chip failures. ("Sometimes it happens that an integrated circuit dies, and its soul transforms into a ghost that haunts the device.") So, I'm sticking with "dice"!

:al contact
connected
vafer using
e dice are
:e, and the
ells all the
mounted in
ubjected to

:e "chip" to
ombination
terms inter-
They don't
al and elec-
e term *IC* to

ize—small,
he simplest
ration (SSI)
a handful of

ome in a 14-
spacing be-
s is 0.3 inch.
hown in (b)
ackage pins,
SI ICs. Such
ier needs to
iagram for a
are grouped

ting out my
ionary, she
cubes with
to prove it.
ported 259
'integrated-
hance-on-a-
es. ("Some-
into a ghost

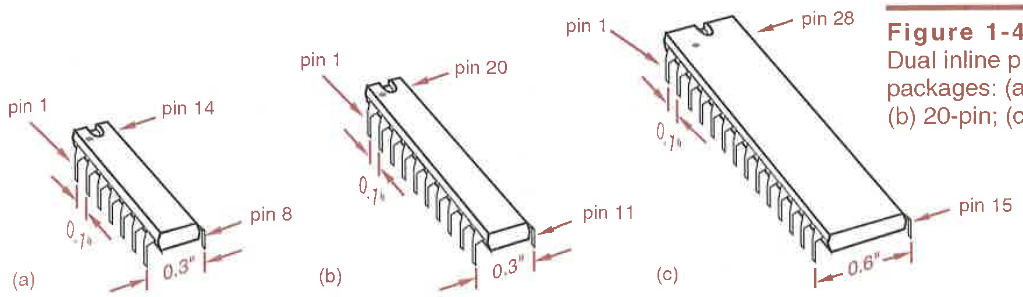


Figure 1-4 Dual inline pin (DIP) packages: (a) 14-pin; (b) 20-pin; (c) 28-pin.

Although SSI ICs are still sometimes used as "glue" to tie together larger-scale elements in complex systems, they have been largely supplanted by programmable logic devices (PLDs), which we'll study in Sections 6.3 and 8.3.

medium-scale integration (MSI)

The next larger commercially available ICs are called *medium-scale integration (MSI)* and contain the equivalent of about 20 to 200 gates. An MSI IC typically contains a functional building block, such as a decoder, register, or counter. In Chapters 6 and 8 we'll place a strong emphasis on these building blocks. Even though the use of discrete MSI ICs has declined, the equivalent building blocks are used extensively in the design of larger ICs.

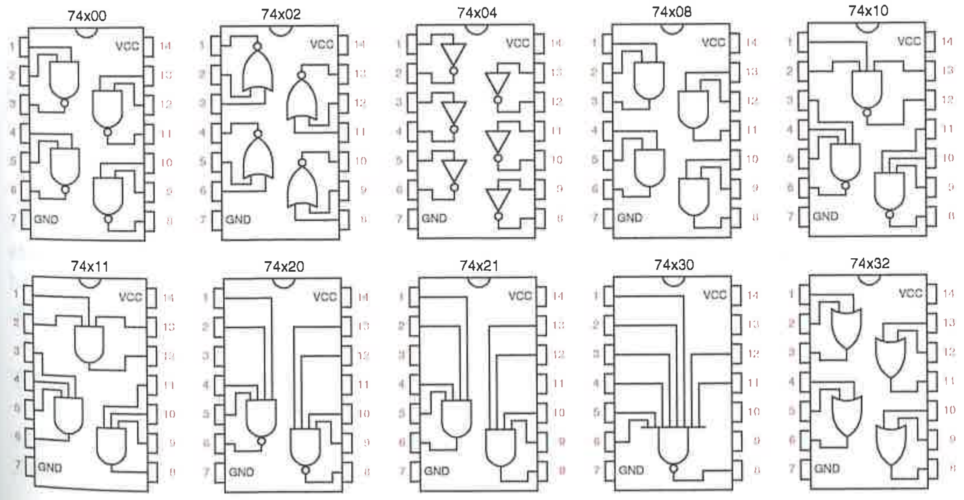
large-scale integration (LSI)

Large-scale integration (LSI) ICs are bigger still, containing the equivalent of 200 to 1,000,000 gates or more. LSI parts include small memories, microprocessors, programmable logic devices, and customized devices.

very large-scale integration (VLSI)

The dividing line between LSI and *very large-scale integration (VLSI)* is fuzzy and tends to be stated in terms of transistor count rather than gate count. Any IC with over a few million transistors is definitely VLSI, and that includes

Figure 1-5 Pin diagrams for a few 7400-series SSI ICs.



TINY-SCALE INTEGRATION

In the coming years, perhaps the most popular remaining use of SSI and MSI, especially in DIP packages, will be in educational labs. These devices will afford students the opportunity to "get their hands dirty" by "breadboarding" and wiring up simple circuits in the same way that their professors did decades ago.

However, much to my surprise, a segment of the IC industry actually introduced parts that went *down*scale from SSI during the past ten years. The idea was to sell individual logic gates in very small packages. These devices handle simple functions that are sometimes needed to match larger-scale components to a particular design, or in some cases they are used to work around bugs in the larger-scale components or their interfaces.

An example of such an IC was Motorola's 74VHC1G08. This chip was a single 2-input AND gate housed in a 5-pin package (power, ground, two inputs, and one output). The entire package, including pins, measured only 0.08 inches on a side and was only 0.04 inches high! Now that's what I would call "tiny-scale integration"!

most microprocessors and memories nowadays, as well as larger programmable logic devices and customized devices. In 2005, VLSI ICs with over 100 million transistors were available.

1.7 Programmable Logic Devices

There are a wide variety of ICs that can have their logic function "programmed" into them after they are manufactured. Most of these devices use technology that also allows the function to be *re*programmed, which means that if you find a bug in your design, you may be able to fix it without physically replacing or rewiring the device. In this book, we give a lot of attention to the design methods for such devices.

programmable logic array (PLA)

Historically, *programmable logic arrays (PLAs)* were the first programmable logic devices. PLAs contained a two-level structure of AND and OR gates with user-programmable connections. Using this structure, a designer could accommodate any logic function up to a certain level of complexity using the well-known theory of logic synthesis and minimization that we'll present in Chapter 4.

programmable array logic (PAL) device

programmable logic device (PLD)

PLA structure was enhanced and PLA costs were reduced with the introduction of *programmable array logic (PAL) devices*. Today, such devices are generically called programmable logic devices (PLDs) and are the "MSI" of the programmable logic industry. We'll have a lot to say about PLD architecture and technology in Sections 6.3 and 8.3.

The ever-increasing capacity of integrated circuits created an opportunity for IC manufacturers to design larger PLDs for larger digital-design applications. However, for technical reasons that we'll discuss in Section 9.5, the basic two-level AND-OR structure of PLDs could not be scaled to larger sizes. Instead,

nd MSI,
ll afford
viring up

lly intro-
ea was to
ple func-
particular
ale com-

ip was a
puts, and
on a side
gration"!

grammable
100 million

ogrammed"
hnology that
u find a bug
g or rewiring
ods for such

rst program-
nd OR gates
signer could
ity using the
'll present in

ed with the
such devices
the "MSI" of
) architecture

n opportunity
sign applica-
9.5, the basic
sizes. Instead,

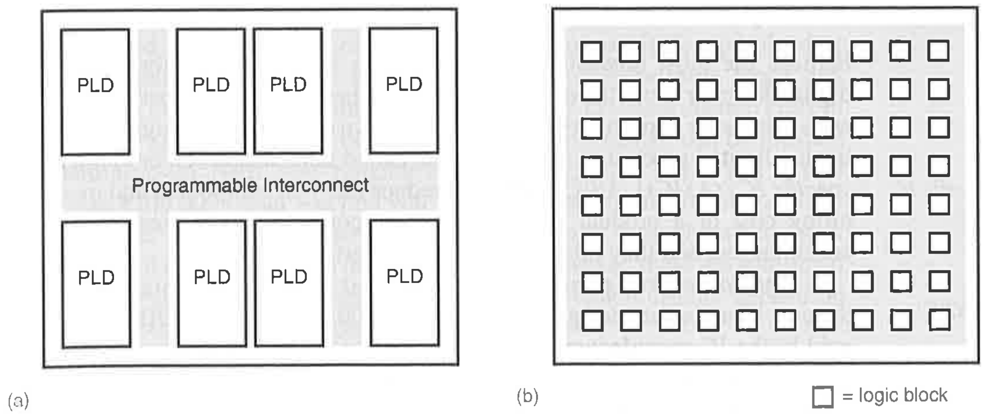


Figure 1-6 Large programmable-logic-device scaling approaches: (a) CPLD; (b) FPGA.

complex PLD (CPLD)

IC manufacturers devised *complex PLD (CPLD)* architectures to achieve the required scale. A typical CPLD is merely a collection of multiple PLDs and an interconnection structure, all on the same chip. In addition to the individual PLDs, the on-chip interconnection structure is also programmable, providing a rich variety of design possibilities. CPLDs can be scaled to larger sizes by increasing the number of individual PLDs and the richness of the interconnection structure on the CPLD chip.

field-programmable gate array (FPGA)

At about the same time that CPLDs were being invented, other IC manufacturers took a different approach to scaling the size of programmable logic chips. Compared to a CPLD, a field-programmable gate array (FPGA) contains a much larger number of smaller individual logic blocks and provides a large, distributed interconnection structure that dominates the entire chip. Figure 1-6 illustrates the difference between the two chip-design approaches.

Proponents of one approach or the other used to get into "religious" arguments over which way was better, but several leading manufacturers of large programmable logic devices acknowledge that there is a place for both approaches, and they manufacture both types of devices. What's more important than chip architecture is that both approaches support a style of design in which products can be moved from design concept to prototype and production in a very short time.

Also important in achieving short "time to market" for all kinds of PLD-based products is the use of HDLs in their design. HDLs like ABEL, Verilog, and VHDL, and their accompanying software tools, enable a design to be compiled, synthesized, and downloaded into a PLD, CPLD, or FPGA in minutes. Highly structured, hierarchical languages like VHDL and Verilog are especially powerful in helping designers to utilize the millions of gates provided in the largest CPLDs and FPGAs.

1.8 Application-Specific ICs

Perhaps the most interesting development in IC technology for the average digital designer is not the ever-increasing number of transistors per chip, but the ever-increasing opportunity to "design your own chip." Chips designed for a particular, limited product or application are called *semicustom ICs* or *application-specific ICs (ASICs)*. ASICs generally reduce the total component and manufacturing cost of a product by reducing chip count, physical size, and power consumption, and they often provide higher performance.

semicustom IC

application-specific IC (ASIC)

The *nonrecurring engineering (NRE) cost* for an ASIC design can exceed the cost of a discrete design by $10,000 to $500,000 or more. NRE charges are paid to the IC manufacturer and others who are responsible for designing the internal structure of the chip, creating tooling such as the metal masks for manufacturing the chips, developing tests for the manufactured chips, and actually making the first few sample chips. So, an ASIC design normally makes sense only if NRE cost is offset by the per-unit savings over the expected sales volume of the product.

nonrecurring engineering (NRE) cost

There are a few approaches to ASIC design, differing in their capabilities and cost. The NRE cost to design a *custom LSI* chip—a chip whose functions, internal architecture, and detailed transistor-level design is tailored for a specific customer—is very high, $500,000 or more. Thus, full custom LSI design is done only for chips that have general commercial application (e.g., microprocessors) or that will enjoy very high sales volume in a specific application (e.g., a digital watch chip, a network interface, or a bus-interface circuit for a PC).

custom LSI

To reduce NRE charges, IC manufacturers have developed libraries of *standard cells* including commonly used MSI functions such as decoders, registers, and counters and commonly used LSI functions such as memories, programmable logic arrays, and microprocessors. In a *standard-cell design*, the logic designer interconnects functions in much the same way as in a multichip MSI/LSI design. Custom cells are created (at added cost, of course) only if absolutely necessary. All of the cells are then laid out on the chip, optimizing the layout to reduce propagation delays and minimize the size of the chip. Minimizing the chip size reduces the per-unit cost of the chip, since it increases the number of chips that can be fabricated on a single wafer. The NRE cost for a standard-cell design is typically on the order of $250,000 or more.

standard cells

standard-cell design

Well, $250,000 is still a lot of money for most folks, so IC manufacturers have gone one step further to bring ASIC design capability to the masses. A *gate array* is an IC whose internal structure is an array of gates whose interconnections are initially unspecified. The logic designer specifies the gate types and interconnections. Even though the chip design is ultimately specified at this very low level, the designer typically works with "macrocells," the same high-level functions used in multichip MSI/LSI and standard-cell designs; software expands the high-level design into a low-level one.

gate array

The main difference between standard-cell and gate-array design is that the macrocells and the chip layout of a gate array are not as highly optimized as those in a standard-cell design, so the chip may be 25% or more larger and therefore may cost more. Also, there is no opportunity to create custom cells in the gate-array approach. On the other hand, a gate-array design can be finished faster and at lower NRE cost, ranging from about $10,000 (what you're told initially) to $100,000 (what you find you've spent when you're all done).

The basic digital design methods that you'll study throughout this book apply very well to the functional design of ASICs. However, there are additional opportunities, constraints, and steps in ASIC design, which usually depend on the particular ASIC vendor and design environment.

1.9 Printed-Circuit Boards

printed-circuit board (PCB)

printed-wiring board (PWB)

An IC is normally mounted on a *printed-circuit board (PCB)* [or *printed-wiring board (PWB)*] that connects it to other ICs in a system. The multilayer PCBs used in typical digital systems have copper wiring etched on multiple, thin layers of fiberglass that are laminated into a single board usually about 1/16 inch thick.

PCB traces

mil

fine-line

Individual wire connections, or *PCB traces*, are usually quite narrow, 10 to 25 mils in typical PCBs. (A *mil* is one-thousandth of an inch.) In *fine-line* PCB technology, the traces are extremely narrow, as little as 3 mils wide with 3-mil spacing between adjacent traces. Thus, up to 166 connections may be routed in a one-inch-wide band on a single layer of the PCB. If higher connection density is needed, then more layers are used.

surface-mount technology (SMT)

Most of the components in modern PCBs use *surface-mount technology (SMT)*. Instead of having the long pins of DIP packages that poke through the board and are soldered to the underside, the leads of SMT IC packages are bent to make flat contact with the top surface of the PCB. Before such components are mounted on the PCB, a special "solder paste" is applied to contact pads on the PCB using a stencil whose hole pattern matches the contact pads to be soldered. Then the SMT components are placed (by hand or by machine) on the pads, where they are held in place by the solder paste (or in some cases, by glue). Finally, the entire assembly is passed through an oven to melt the solder paste, which then solidifies when cooled.

Surface-mount technology, coupled with fine-line PCB technology, allows extremely dense packing of integrated circuits and other components on a PCB. This dense packing does more than save space. For very high-speed circuits, dense packing helps to minimize certain adverse analog phenomena, such as transmission-line effects and speed-of-light limitations.

multichip module (MCM)

To satisfy the most stringent requirements for speed and density, *multichip modules (MCMs)* have been developed. In this technology, IC dice are not mounted in individual plastic or ceramic packages. Instead, the IC dice for a high-speed subsystem (say, a processor and its cache memory) are bonded directly to a substrate that contains the required interconnections on multiple layers. The MCM is hermetically sealed and has its own external pins for power, ground, and just those signals that are required by the system that contains it.

1.10 Digital-Design Levels

Digital design can be carried out at several different levels of representation and abstraction. Although you may learn and practice design at a particular level, from time to time you'll need to go up or down a level or two to get the job done. Also, the industry itself and most designers have been steadily moving to higher levels of abstraction as circuit density and functionality have increased.

The lowest level of digital design is device physics and IC manufacturing processes. This is the level that is primarily responsible for the breathtaking advances in IC speed and density that have occurred over the past decades. The effects of these advances are summarized in *Moore's Law*, first stated by Intel founder Gordon Moore in 1965: that the number of transistors per square inch in the newest ICs will double every year. In recent years, the rate of advance has slowed down to doubling about every 24 months, but it is important to note that with each doubling of density has also come a significant increase in speed.

Moore's Law

This book does not reach down to the level of device physics and IC processes, but you need to recognize the importance of that level. Being aware of likely technology advances and other changes is important in system and product planning. For example, decreases in chip geometries have recently forced a move to lower logic-power-supply voltages, causing major changes in the way designers plan and specify modular systems and upgrades.

In this book, we jump into digital design at the transistor level and go all the way up to the level of logic design using HDLs. We stop short of the next level, which includes computer design and overall system design. The "center" of our discussion is at the level of functional building blocks.

To get a preview of the levels of design that we'll cover, consider a simple design example. Suppose you are to build a "multiplexer" with two data input bits, A and B, a control input bit S, and an output bit Z. Depending on the value of S, 0 or 1, the circuit is to transfer the value of either A or B to the output Z. This idea is illustrated in the "switch model" of Figure 1-7. Let us consider the design of this function at several different levels.

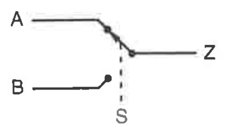


Figure 1-7 Switch model for multiplexer function.

Although logic design is usually carried out at a higher level, for some functions it is advantageous to optimize them by designing at the transistor level. The multiplexer is such a function. Figure 1-8 shows how the multiplexer can be designed in "CMOS" technology using specialized transistor circuit structures called "transmission gates," discussed in Section 3.7.1. Using this approach, the multiplexer can be built with just six transistors. Any of the other approaches that we describe requires at least 14 transistors.

n multiple
for power,
tains it.

ntation and
cular level,
e job done.
g to higher
ed.
nufacturing
reathtaking
cades. The
ted by Intel
uare inch in
dvance has
to note that
speed.
sics and IC
ing aware of
system and
ave recently
r changes in

el and go all
t of the next
The "center"

ider a simple
vo data input
on the value
utput Z. This
ler the design

vel, for some
ansistor level.
iplexer can be
uit structures
approach, the
er approaches

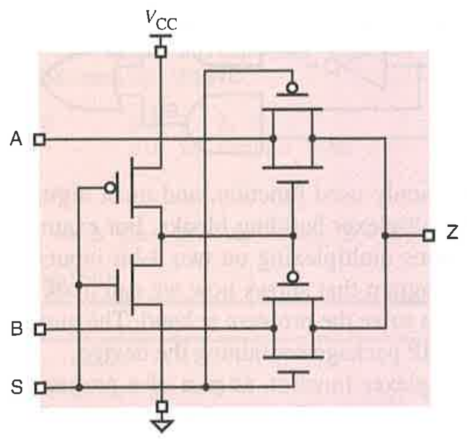


Figure 1-8
Multiplexer design using CMOS transmission gates.

In the traditional study of logic design, we would use a "truth table" to describe the multiplexer's logic function. A truth table lists all possible combinations of input values and the corresponding output values for the function. Since the multiplexer has three inputs, it has 2^3 or 8 possible input combinations, as shown in the truth table in Table 1-1.

Once we have a truth table, traditional logic design methods, described in Section 4.3, use Boolean algebra and well-understood minimization algorithms to derive an "optimal" two-level AND-OR equation from the truth table. For the multiplexer truth table, we would derive the following equation:

$$Z = S' \cdot A + S \cdot B$$

This equation is read "Z equals not S and A, or S and B." Going one step further, we can convert the equation into a set of logic gates that perform the specified logic function, as shown in Figure 1-9 on the next page. This circuit requires 14 transistors if we use standard CMOS technology for the four gates shown.

Table 1-1
Truth table for the multiplexer function.

S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

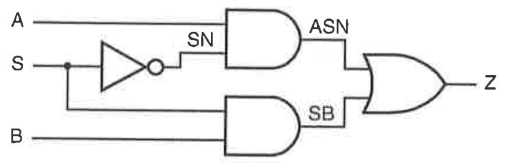


Figure 1-9
Gate-level logic diagram for multiplexer function.

A multiplexer is a very commonly used function, and most digital logic technologies provide predefined multiplexer building blocks. For example, the 74x157 is an MSI chip that performs multiplexing on two 4-bit inputs simultaneously. Figure 1-10 is a logic diagram that shows how we can hook up just one bit of this 4-bit building block to solve the problem at hand. The numbers in color are pin numbers of a 16-pin DIP package containing the device.

We can also realize the multiplexer function as part of a programmable logic device. HDLs like ABEL allow us to specify logic functions using Boolean equations similar to the one on the previous page, but an HDL's "higher-level" language elements can create a more readable program. For example, Table 1-2 is an ABEL program for the multiplexer function. The first three lines define the name of the program module and specify the type of PLD in which the function will be realized. The next two lines specify the device pin numbers for inputs and output. The "`when`" statement specifies the actual logic function in a way that's very easy to understand, even though we haven't covered ABEL yet.

Two even higher-level languages, VHDL and Verilog, can be used to specify the multiplexer function in a way that is very flexible and hierarchical. Table 1-3 is an example VHDL program for the multiplexer. The first two lines specify a standard library and set of definitions to use in the design. The next four lines specify only the inputs and outputs of the function, purposely hiding any details about the way the function is realized internally. The "architecture" section of the program is a behavioral specification of the multiplexer's function. VHDL syntax takes a little getting used to, but the single "`when`" statement says basically the same thing that the ABEL version did. A synthesizer can use this behavioral description to produce a circuit that has this behavior in a specified target digital-logic technology, such as a PLD or ASIC.

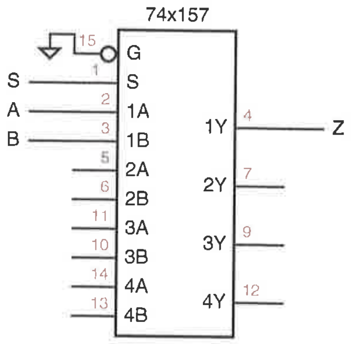
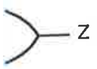


Figure 1-10
Logic diagram for a multiplexer using an MSI building block.



igital logic
ɩample, the
puts simul-
ook up just
numbers in
:.
ɩgrammable
ing Boolean
igher-level"
e, Table 1-2
ɜs define the
the function
ɔr inputs and
a way that's
t.
be used to
hierarchical.
irst two lines
gn. The next
ɔosely hiding
architecture"
er's function.
atement says
r can use this
in a specified

```
module chap1mux
title 'Two-input multiplexer example'
CHAP1MUX device 'P16V8'

A, B, S       pin 1, 2, 3;
Z             pin 13 istype 'com';

equations

when S == 0 then Z = A;  else Z = B;

end chap1mux
```

Table 1-2
ABEL program for the multiplexer.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Vchap1mux is
    port ( A, B, S: in  STD_LOGIC;
           Z:       out STD_LOGIC );
end Vchap1mux;

architecture Vchap1mux_arch of Vchap1mux is
begin
  Z <= A when S = '0' else B;
end Vchap1mux_arch;
```

Table 1-3
VHDL program for the multiplexer.

By explicitly enforcing a separation of input/output definitions ("entity") and internal realization ("architecture"), VHDL makes it easy for designers to define alternate realizations of functions without having to make changes elsewhere in the design hierarchy. For example, a designer could specify an alternate, structural architecture for the multiplexer as shown in Table 1-4. This architecture is basically a text equivalent of the logic diagram in Figure 1-9.

Going one step further, VHDL is powerful enough that we could actually define operations that model functional behavior at the transistor level (though we won't explore such capabilities in this book). Thus, we could come full circle by writing a VHDL program that specifies a transistor-level realization of the multiplexer equivalent to Figure 1-8.

```
architecture Vchap1mux_gate_arch of Vchap1mux is
signal SN, ASN, SB: STD_LOGIC;
  -- required component declarations have been omitted
  --   for brevity in this example.
begin
  U1: port map INV (S, SN);
  U2: port map AND2 (A, SN, ASN);
  U3: port map AND2 (S, B, SB);
  U4: port map OR2 (ASN, SB, Z);
end Vchap1mux_gate_arch;
```

Table 1-4
"Structural" VHDL program for the multiplexer.

```
module Vrchap1mux(A, B, S, Z);
  input A, B, S;
  output Z;

  assign Z = (S==0) ? A : B;
endmodule
```

Table 1-5 Verilog program for the multiplexer.

Table 1-5 is a Verilog program for the same multiplexer function. Verilog syntax is somewhat C-like; see, for example, the conditional expression that tests the value of S so that either A or B is assigned to Z depending on the test's outcome. Like C, Verilog is less picky about variable and type definitions—for example, in Table 1-5 all of the variables default to being 1-bit "wires." Unlike VHDL, Verilog does not require separate definitions of entity and architecture—it's all together in one "module." But Verilog also provides a means for defining functions structurally as in the VHDL example of Table 1-4.

1.11 The Name of the Game

board-level design

Given the functional and performance requirements for a digital system, the name of the game in practical digital design is to minimize cost. For *board-level designs*—systems that are packaged on a single PCB—this usually means minimizing the number of IC packages. If too many ICs are required, they won't all fit on the PCB. "Well, just use a bigger PCB," you say. Unfortunately, PCB sizes are usually constrained by factors such as preexisting standards (e.g., add-in boards for PCs), packaging constraints (e.g., it has to fit in your pocket), or edicts from above (e.g., in order to get the project approved three months ago, you foolishly told your manager that it would all fit on a 3 × 5 inch PCB, and now you've got to deliver!). In each of these cases, the cost of using a larger PCB or multiple PCBs may be unacceptable.

Minimizing the number of ICs is usually the rule even though individual IC costs vary. For example, a typical SSI or MSI IC may cost 20 cents, while a small PLD may cost a dollar. It may be possible to perform a particular function with three SSI and MSI ICs (60 cents) or one PLD (a dollar). In many situations, the more expensive PLD solution is used, not because the designer owns stock in the PLD company, but because the PLD solution uses less PCB area and is also a lot easier to change if it's not right the first time.

ASIC design

In *ASIC design*, the name of the game is a little different, but the importance of structured, functional design techniques is the same. Although it's easy to burn hours and weeks creating custom macrocells and minimizing the total gate count of an ASIC, only rarely is this advisable. The per-unit cost reduction achieved by having a 10% smaller chip is negligible except in high-volume applications. In applications with low to medium volume (the majority), two other factors are more important: design time and NRE cost.

ıram for
xer.

ɔn. Verilog
ession that
n the test's
itions—for
es." Unlike
hitecture—
for defining

system, the
board-level
ıally means
l, they won't
nately, PCB
ıdards (e.g.,
our pocket),
months ago,
ch PCB, and
a larger PCB

individual IC
while a small
function with
ituations, the
ıs stock in the
ıd is also a lot

rent, but the
Although it's
ıinimizing the
per-unit cost
xcept in high-
(the majority),

A shorter design time allows a product to reach the market sooner, increasing revenues over the lifetime of the product. A lower NRE cost also flows right to the "bottom line" and in small companies may be the only way the project can be completed before the company runs out of money (believe me, I've been there!). If the product is successful, it's always possible and profitable to "tweak" the design later to reduce per-unit costs. The need to minimize design time and NRE cost argues in favor of a structured, as opposed to highly optimized, approach to ASIC design, using standard building blocks provided in the ASIC manufacturer's library.

The considerations in PLD, CPLD, and FPGA design are a combination of the above. The choice of a particular PLD technology and device size is usually made fairly early in the design cycle. Later, as long as the design "fits" in the selected device, there's no point in trying to optimize gate count or board area—the device has already been committed. However, if new functions or bug fixes push the design beyond the capacity of the selected device, that's when you must work very hard to modify the design to make it fit.

1.12 Going Forward

This concludes the introductory chapter. As you continue reading this book, keep in mind two things. First, the ultimate goal of digital design is to build systems that solve problems for people. While this book will give you the basic tools for design, it's still your job to keep "the big picture" in the back of your mind. Second, cost is an important factor in every design decision; and you must consider not only the cost of digital components, but also the cost of the design activity itself.

Finally, as you get deeper into the text, if you encounter something that you think you've seen before but don't remember where, please consult the index. I've tried to make it as helpful and complete as possible.

Drill Problems

1.1 Suggest some better-looking chapter-opening artwork to put on page 1 of the next edition of this book.

1.2 Give three different definitions for the word "bit" as used in this chapter.

1.3 Define the following acronyms: ASIC, CAD, CD, CO, CPLD, DIP, DVD, FPGA, HDL, IC, IP, LSI, MCM, MSI, NRE, PBX, PCB, PLD, PWB, SMT, SSI, VHDL, VLSI.

1.4 Research the definitions of the following acronyms: ABEL, CMOS, DDPP, JPEG, MPEG, OK, PERL. (Is OK really an acronym?)

1.5 Excluding the topics in Section 1.2, list three once-analog systems that have "gone digital" since you were born.

1.6 Draw a digital circuit consisting of a 2-input AND gate and three inverters, where an inverter is connected to each of the AND gate's inputs and its output. For each of the four possible combinations of inputs applied to the two primary inputs of this circuit, determine the value produced at the primary output. Is there a simpler circuit that gives the same input/output behavior?

1.7 When should you use the pin diagrams of Figure 1-5 in the schematic diagram of a circuit?

1.8 What is the relationship between "die" and "dice"?

ɔf Figure 2-5,

ıanical encod-

ːode and man-
lizing to *n*-bit
pair of values
. 7, 6, 8, 9, etc.
/ "bad" bound-
/ code?
: use Gray code
ntrollers. Why?
ɔ in some appli-
ff cycles rather
o suggest a way

cube are there?
ensional object)

ıbe for a specific

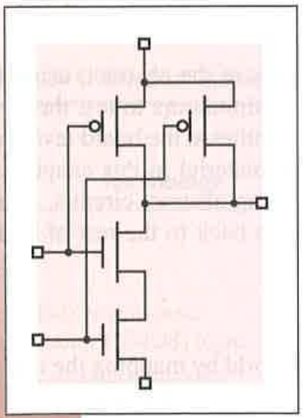
formation bits.
bit.
orner" parity bits

to the total num-
rable for efficient
ates of distance-2
) information bits.
nsional code or a
/le of Table 2-15,
ation bits of each

n bits. Write a list

'stem to write new
overed in the event
r of disk accesses

pattern 10101110
Manchester codes,
ht.



chapter 3

Digital Circuits

Marketing hype notwithstanding, we live in an analog world, not a digital one. Voltages, currents, and other physical quantities in real circuits take on values that are infinitely variable, depending on properties of the real devices that comprise the circuits. Because real values are continuously variable, we could use a physical quantity such as a signal voltage in a circuit to represent a real number (e.g., 3.14159265358979 volts represents the mathematical constant *pi* to 14 decimal digits of precision).

However, stability and accuracy in physical quantities are difficult to obtain in real circuits. They can be affected by manufacturing variations, temperature, power-supply voltage, cosmic rays, and noise created by other circuits, among other things. If we used an analog voltage to represent *pi*, we might find that instead of being an absolute mathematical constant, *pi* varied over a range of 10% or more.

Also, many mathematical and logical operations can be difficult or impossible to perform with analog quantities. While it is possible with some cleverness to build an analog circuit whose output voltage is the square root of its input voltage, no one has ever built a 100-input, 100-output analog circuit whose outputs are a set of voltages identical to the set of input voltages, but sorted arithmetically.

The purpose of this chapter is to give you a solid working knowledge of the electrical aspects of digital circuits, enough for you to understand and build real circuits and systems. We'll see in later chapters that with modern software tools, it's possible to "build" circuits in the abstract, using hardware design languages to specify their design and simulators to test their operation. But to build real, production-quality circuits, either at the board level or the chip level, you'll need to understand most of the material in this chapter. If you're really anxious to start designing and simulating abstract circuits, you can just read the first section of this chapter and come back to the rest of it later, after you've discovered that you really need it.

3.1 Logic Signals and Gates

digital logic

logic values

Digital logic hides the pitfalls of the analog world by mapping the infinite set of real values for a physical quantity into two subsets corresponding to just two possible numbers or *logic values*—0 and 1. Thus, digital logic circuits can be analyzed and designed functionally, using switching algebra, tables, and other abstract means to describe the operation of well-behaved 0s and 1s in a circuit.

binary digit

bit

A logic value, 0 or 1, is often called a *binary digit*, or *bit*. If an application requires more than two discrete values, additional bits may be used, with a set of n bits representing 2^n different values.

Examples of the physical phenomena used to represent bits in some modern (and not-so-modern) digital technologies are given in Table 3-1. With most phenomena, there is an undefined region between the 0 and 1 states (e.g., voltage = 1.8 V, dim light, capacitor slightly charged, etc.). This undefined region is needed so that the 0 and 1 states can be unambiguously defined and reliably detected. Noise can more easily corrupt results if the boundaries separating the 0 and 1 states are too close.

LOW

HIGH

When discussing electronic logic circuits such as CMOS and TTL, digital designers often use the words "LOW" and "HIGH" in place of "0" and "1" to remind them that they are dealing with real circuits, not abstract quantities:

LOW	A signal in the range of algebraically lower voltages, which is interpreted as a logic 0.
HIGH	A signal in the range of algebraically higher voltages, which is interpreted as a logic 1.

positive logic

negative logic

Note that the assignments of 0 and 1 to LOW and HIGH are somewhat arbitrary. Still, assigning 0 to LOW and 1 to HIGH seems natural and is called *positive logic*. The opposite assignment, 1 to LOW and 0 to HIGH, is not often used and is called *negative logic*.

buffer

Because a wide range of physical values represent the same binary value, digital logic is highly immune to component and power-supply variations and noise. Furthermore, *buffer* circuits can be used to regenerate (or amplify) "weak" values into "strong" ones, so that digital signals can be transmitted over arbitrary distances without loss of information. For example, a buffer for CMOS logic converts any HIGH input voltage into an output very close to 5.0 V, and any LOW input voltage into an output very close to 0.0 V.

ing hardware
eir operation.
/el or the chip
oter. If you're
, you can just
f it later, after

e infinite set of
ng to just two
circuits can be
bles, and other
1s in a circuit.
f an application
ed, with a set of

t bits in some
Table 3-1. With
id 1 states (e.g.,
This undefined
ısly defined and
the boundaries

and TTL, digital
f "0" and "1" to
t quantities:

oltages, which is

oltages, which is

mewhat arbitrary.
is called *positive*
not often used and

ame binary value,
ply variations and
or amplify) "weak"
itted over arbitrary
r for CMOS logic
5.0 V, and any LOW

Table 3-1 Physical states representing bits in different logic and memory technologies.

	State Representing Bit	
Technology	***0***	***1***
Pneumatic logic	Fluid at low pressure	Fluid at high pressure
Relay logic	Circuit open	Circuit closed
Complementary metal-oxide semiconductor (CMOS) logic	0–1.5 V	3.5–5.0 V
Transistor-transistor logic (TTL)	0–0.8 V	2.0–5.0 V
Dynamic memory	Capacitor discharged	Capacitor charged
Nonvolatile, erasable memory	Electrons trapped	Electrons released
Microprocessor on-chip serial number	Fuse blown	Fuse intact
Polymer memory	Molecule in state A	Molecule in state B
Fiber optics	Light off	Light on
Magnetic disk or tape	Flux direction "north"	Flux direction "south"
Compact disc (CD)	No pit	Pit
Writeable compact disc (CD-R)	Dye in crystalline state	Dye in noncrystalline state

A logic circuit can be represented with a minimum amount of detail simply as a "black box" with a certain number of inputs and outputs. For example, Figure 3-1 shows a logic circuit with three inputs and one output. However, this representation does not describe how the circuit responds to input signals.

From the point of view of electronic circuit design, it takes a lot of information to describe the precise electrical behavior of a circuit. However, since the inputs of a digital logic circuit can be viewed as taking on only discrete 0 and 1 values, the circuit's "logical" operation can be described with a table that ignores electrical behavior and lists only discrete 0 and 1 values.

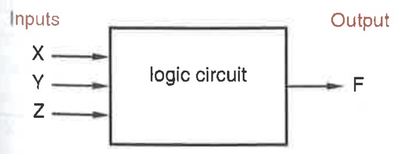


Figure 3-1
"Black-box" representation of a 3-input, 1-output logic circuit.

STATE TRANSITIONS The last four technologies in Table 3-1 don't actually use absolute states to represent bit values. Rather, they use transitions (or absence of transitions) between states to represent 0s and 1s using a code such as the Manchester code described on page 73.

combinational circuit
truth table

A logic circuit whose outputs depend only on its current inputs is called a *combinational circuit.* Its operation is fully described by a *truth table* that lists all combinations of input values and the output value(s) produced by each one. Table 3-2 is the truth table for a logic circuit with three inputs X, Y, and Z and a single output F. This truth table lists all eight possible combinations of values of X, Y, and Z and the circuit's output value F for each combination.

Table 3-2
Truth table for a combinational logic circuit.

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

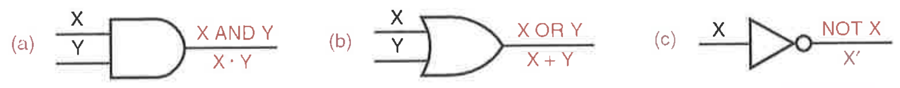
sequential circuit
state table

A circuit with memory, whose outputs depend on the current input *and* the sequence of past inputs, is called a *sequential circuit.* Its behavior may be described by a *state table* that specifies its output and next state as functions of its current state and input. Sequential circuits are introduced in Chapter 7.

As we'll show in Section 4.1, just three basic logic functions, AND, OR, and NOT, can be used to build any combinational digital logic circuit. Figure 3-2 shows the truth tables and symbols for logic "gates" that perform these functions. The symbols and truth tables for AND and OR may be extended to gates with any number of inputs. The gates' functions are easily defined in words:

AND gate
OR gate
NOT gate
inverter

- An *AND gate* produces a 1 output if and only if all of its inputs are 1.
- An *OR gate* produces a 1 if and only if one or more of its inputs are 1.
- A *NOT gate,* usually called an *inverter,* produces an output value that is the opposite of its input value.



X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

X	NOT X
0	1
1	0

Figure 3-2
Basic logic elements:
(a) AND; (b) OR;
(c) NOT (inverter).

Notic
input
when

is use
ple, t
or OF
for th

As w
may

acco
from
swit

time
how
sign
the a
a lag

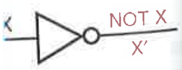
ıts is called a
ıble that lists
by each one.
', and Z and a
s of values of

ıt input *and* the
havior may be
as functions of
Chapter 7.
ions, AND, OR,
rcuit. Figure 3-2
orm these func-
xtended to gates
ed in words:

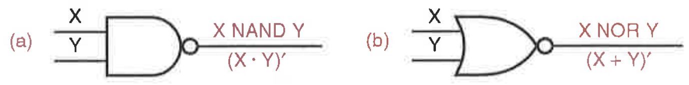
nputs are 1.

inputs are 1.

t value that is the



X	NOT X
0	1
1	0



X	Y	X NAND Y
0	0	1
0	1	1
1	0	1
1	1	0

X	Y	X NOR Y
0	0	1
0	1	0
1	0	0
1	1	0

Figure 3-3
Inverting gates:
(a) NAND; (b) NOR.

Notice that in the definitions of AND and OR functions, we only had to state the input conditions for which the output is 1, because there is only one possibility when the output is not 1—it must be 0.

The circle on the inverter symbol's output is called an *inversion bubble* and is used in this and other gate symbols to denote "inverting" behavior. For example, two more logic functions are obtained by combining inversion with an AND or OR function in a single gate. Figure 3-3 shows the truth tables and symbols for these gates. Their functions are also easily described in words:

inversion bubble

- A *NAND gate* produces the opposite of an AND gate's output, a 0 if and only if all of its inputs are 1.
- A *NOR gate* produces the opposite of an OR gate's output, a 0 if and only if one or more of its inputs are 1.

NAND gate

NOR gate

As with AND and OR gates, the symbols and truth tables for NAND and NOR may be extended to gates with any number of inputs.

Figure 3-4 is a logic circuit using AND, OR, and NOT gates that functions according to the truth table of Table 3-2. In Chapter 4 you'll learn how to go from a truth table to a logic circuit, and vice versa, and you'll also learn about the switching-algebra notation (in color) used in Figures 3-2 through 3-4.

Real logic circuits function in another very important analog dimension—time. For example, Figure 3-5 on the next page is a *timing diagram* that shows how the circuit of Figure 3-4 might respond to a time-varying pattern of input signals. The timing diagram shows that the logic signals do not change between the analog values corresponding to 0 and 1 instantaneously, and also that there is a lag between an input change and the corresponding output change. Later in this

timing diagram

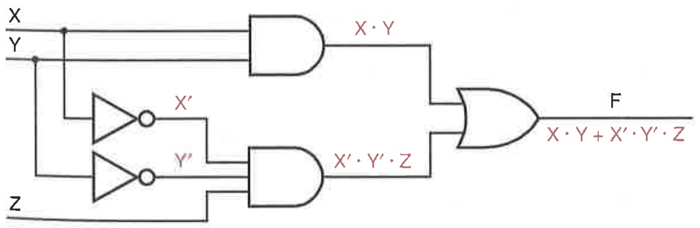


Figure 3-4
Logic circuit with the truth table of Table 3-2.

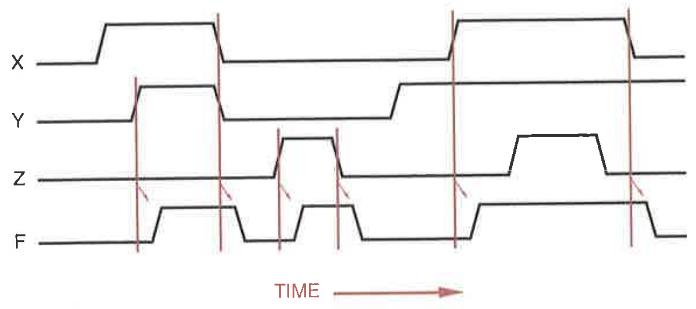


Figure 3-5
Timing diagram for a logic circuit.

chapter you'll learn some of the reasons for this timing behavior, and how it is specified and handled in real circuits. And you'll be happy to learn in a later chapter how this analog timing behavior can be generally ignored in most sequential circuits, and how instead the circuit can be viewed as moving between discrete states at precise intervals defined by a clock signal.

Thus, even if you know nothing about analog electronics, you should be able to understand the logical behavior of digital circuits. However, there comes a time in design and debugging when every digital logic designer must throw out "the digital abstraction" temporarily and consider the analog phenomena that limit or disrupt digital performance. The rest of this chapter prepares you for that day by discussing the electrical characteristics of digital logic circuits.

THERE'S HOPE FOR NON-EE'S

If all of this electrical "stuff" bothers you, don't worry, at least for now. The rest of this book is written to be as independent of this stuff as possible. But you'll need it later, if you ever have to design and build digital systems in the real world.

3.2 Logic Families

There are many, many ways to design an electronic logic circuit. The first electrically controlled logic circuits, developed at Bell Laboratories in the 1930s, were based on relays. In the mid-1940s, the first electronic digital computer, the Eniac, used logic circuits based on vacuum tubes. The Eniac had about 18,000 tubes and a similar number of logic gates, not a lot by today's standards of microprocessor chips with tens of millions of transistors. However, the Eniac could hurt you a lot more than a chip could if it fell on you—it was 100 feet long, 10 feet high, 3 feet deep, and consumed 140,000 watts of power!

semiconductor diode
bipolar junction transistor
integrated circuit (IC)

The inventions of the *semiconductor diode* and the *bipolar junction transistor* allowed the development of smaller, faster, and more capable computers in the late 1950s. In the 1960s, the invention of the *integrated circuit (IC)* allowed multiple diodes, transistors, and other components to be fabricated on a single chip, and computers got still better.

The 1960s also saw the introduction of the first integrated-circuit logic families. A *logic family* is a collection of different integrated-circuit chips that have similar input, output, and internal circuit characteristics, but that perform different logic functions. Chips from the same family can be interconnected to perform any desired logic function. Chips from different families may not be compatible; they may use different power-supply voltages or may use different input and output conditions to represent logic values.

logic family

The most successful *bipolar logic family* (one based on bipolar junction transistors) was *transistor-transistor logic (TTL)*. First introduced in the 1960s, TTL evolved into a family of logic families that were compatible with each other but differed in speed, power consumption, and cost. Digital systems could mix components from several different TTL families, according to design goals and constraints in different parts of the system.

bipolar logic family

transistor-transistor logic (TTL)

Ten years *before* the bipolar junction transistor was invented, the principles of operation were patented for another type of transistor, called the *metal-oxide semiconductor field-effect transistor (MOSFET)*, or simply *MOS transistor*. However, MOS transistors were difficult to fabricate in the early days, and it wasn't until the 1960s that a wave of developments made MOS-based logic and memory circuits practical. Even then, MOS circuits lagged bipolar circuits considerably in speed. They were attractive only in a few applications because of their lower power consumption and higher levels of integration.

metal-oxide semiconductor field-effect transistor (MOSFET)

MOS transistor

Beginning in the mid-1980s, advances in the design of MOS circuits, in particular *complementary MOS (CMOS)* circuits, tremendously increased their performance and popularity. Almost all new large-scale integrated circuits, such as microprocessors and memories, use CMOS. Likewise, small- to medium-scale applications, for which TTL was once the logic family of choice, are now much more likely to use CMOS devices with equivalent functionality or better, and higher speed and lower power consumption. CMOS circuits now account for the vast majority of the worldwide integrated-circuit market.

complementary MOS (CMOS)

CMOS logic is both the most capable and the easiest to understand commercial digital logic technology. Beginning in the next section, we describe the basic structure of CMOS logic circuits and introduce the most commonly used commercial CMOS logic families.

A LITTLE BIT OF TTL

Although TTL was largely replaced by CMOS in the 1990s, you still may encounter TTL components in your academic labs; therefore, basic TTL concepts are covered in Sections 3.10.3 through 3.10.7.

As a consequence of the industry's transition from TTL to CMOS over a long period of time, many CMOS families were designed to be somewhat compatible with TTL. Section 3.10.8 describes how TTL and CMOS families can be mixed within a single system.

GREEN STUFF Nowadays, the acronym "MOS" is usually spoken as "moss," rather than spelled out. Hence in this book we say "a MOS transistor," not "an MOS transistor." And "CMOS" has always been spoken as "sea moss."

3.3 CMOS Logic

The functional behavior of a CMOS logic circuit is fairly easy to understand, even if your knowledge of analog electronics is not particularly deep. The basic (and typically only) building blocks in CMOS logic circuits are MOS transistors, described shortly. But before getting into that, we need to talk about logic levels.

3.3.1 CMOS Logic Levels

Abstract logic elements process binary digits, 0 and 1. However, real logic circuits process electrical signals such as voltage levels. In any logic circuit, there is a range of voltages (or other circuit conditions) that is interpreted as a logic 0, and another, nonoverlapping range that is interpreted as a logic 1.

A typical CMOS logic circuit operates from a 5-volt power supply. Such a circuit may interpret any voltage in the range 0–1.5 V as a logic 0, and in the range 3.5–5.0 V as a logic 1. Thus, the definitions of LOW and HIGH for 5-volt CMOS logic are as shown in Figure 3-6. Voltages in the intermediate range (1.5–3.5 V) are not expected to occur except during signal transitions, and yield undefined logic values (i.e., a circuit may interpret them as either 0 or 1). CMOS circuits using other power-supply voltages, such as 3.3 or 2.7 volts, partition the voltage range similarly.

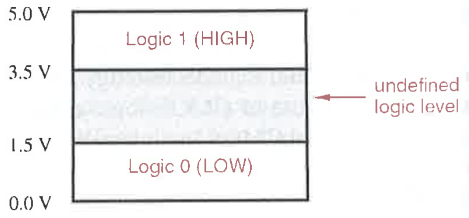


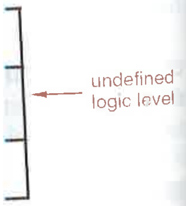
Figure 3-6 Logic levels for typical CMOS logic circuits.

MORE POWER (SUPPLIES) TO YOU! CMOS circuits can be designed to operate with power-supply voltages higher or, most often, lower than 5 volts, as described later in Sections 3.8 and 3.9. The logic levels used in these circuits are adjusted accordingly, though not necessarily proportionately. To keep things simple for now, we'll present CMOS logic levels and electrical characteristics based on a 5-volt power supply.

spelled out.
;istor." And

to understand,
leep. The basic
: MOS transis-
alk about logic

ever, real logic
iy logic circuit,
interpreted as a
; a logic 1.
r supply. Such a
gic 0, and in the
HIGH for 5-volt
ermediate range
sitions, and yield
er 0 or 1). CMOS
olts, partition the



ltages higher or,
nd 3.9. The logic
cessarily propor-
logic levels and

3.3.2 MOS Transistors

A MOS transistor can be modeled as a 3-terminal device that acts like a voltage-controlled resistance. As suggested by Figure 3-7, an input voltage applied to one terminal controls the resistance between the remaining two terminals. In digital logic applications, a MOS transistor is operated so its resistance is always either very high (and the transistor is "off") or very low (and the transistor is "on").

"off" transistor
"on" transistor

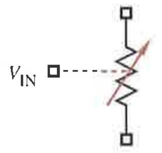


Figure 3-7
The MOS transistor as a voltage-controlled resistance.

There are two types of MOS transistors, *n*-channel and *p*-channel; the names refer to the type of semiconductor material used for the resistance-controlled terminals. The circuit symbol for an *n-channel MOS (NMOS) transistor* is shown in Figure 3-8. The terminals are called *gate, source,* and *drain.* (Note that the "gate" of a MOS transistor has nothing to do with a "logic gate.") As you might guess from the orientation of the circuit symbol, the drain is normally at a higher voltage than the source.

n-channel MOS (NMOS) transistor
gate
source
drain

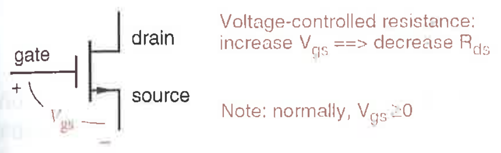

drain
gate
source
+
V_{gs}
−
Voltage-controlled resistance:
increase V_{gs} ==> decrease R_{ds}
Note: normally, $V_{gs} \geq 0$

Figure 3-8
Circuit symbol for an n-channel MOS (NMOS) transistor.

The voltage from gate to source (V_{gs}) in an NMOS transistor is normally zero or positive. If $V_{gs} = 0$, then the resistance from drain to source (R_{ds}) is very high, at least a megohm (10^6 ohms) or more. As we increase V_{gs} (i.e., increase the voltage on the gate), R_{ds} decreases to a very low value, 10 ohms or less in some devices.

The circuit symbol for a *p-channel MOS (PMOS) transistor* is shown in Figure 3-9. Operation is analogous to that of an NMOS transistor, except that the source is normally at a higher voltage than the drain, and V_{gs} is normally zero or negative. If V_{gs} is zero, then the resistance from source to drain (R_{ds}) is very high. As we algebraically decrease V_{gs} (i.e., *decrease* the voltage on the gate), R_{ds} decreases to a very low value.

p-channel MOS (PMOS) transistor

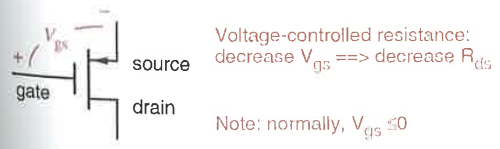

+
V_{gs}
−
source
gate
drain
Voltage-controlled resistance:
decrease V_{gs} ==> decrease R_{ds}
Note: normally, $V_{gs} \leq 0$

Figure 3-9
Circuit symbol for a *p*-channel MOS (PMOS) transistor.

IMPEDANCE VS. RESISTANCE Technically, there's a difference between the words "impedance" and "resistance," but electrical engineers often use the terms interchangeably. So do we in this text.

The gate of a MOS transistor has a very high impedance. That is, the gate is separated from the source and the drain by an insulating material with a very high resistance. However, the gate voltage creates an electric field that enhances or retards the flow of current between source and drain. This is the "field effect" in the "MOSFET" name.

leakage current
microampere, μA

Regardless of gate voltage, almost no current flows from the gate to source, or from the gate to drain for that matter. The resistance between the gate and the other terminals of the device is extremely high, well over a megohm. The small amount of current that flows across this resistance is very small, typically less than one *microampere* (μA, 10^{-6} A), and is called a *leakage current*.

The MOS transistor symbol itself reminds us that there is no connection between the gate and the other two terminals of the device. However, the gate of a MOS transistor is capacitively coupled to the source and drain, as the symbol might suggest. In high-speed circuits, the power needed to charge and discharge this capacitance on each input-signal transition accounts for a nontrivial portion of a circuit's power consumption.

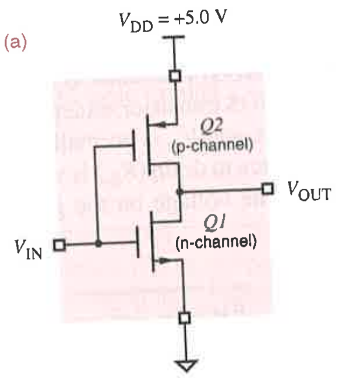
3.3.3 Basic CMOS Inverter Circuit

CMOS logic

NMOS and PMOS transistors are used together in a complementary way to form *CMOS logic*. The simplest CMOS circuit, a logic inverter, requires only one of each type of transistor, connected as shown in Figure 3-10(a). The power-supply voltage, V_{DD}, typically may be in the range 1–6 V and is often set at 5.0 V for compatibility with the older TTL family.

Figure 3-10 CMOS inverter: (a) circuit diagram; (b) functional behavior; (c) logic symbol.

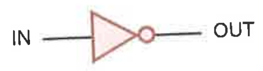
(a)



(b)

V_{IN}	Q1	Q2	V_{OUT}
0.0 (L)	off	on	5.0 (H)
5.0 (H)	on	off	0.0 (L)

(c)



WHAT'S IN A NAME? The "DD" in the name "V_{DD}" refers to the *drain* terminals of MOS transistors. This may seem strange, since in the CMOS inverter V_{DD} is actually connected to the *source* terminal of a PMOS transistor. However, CMOS logic circuits evolved from NMOS logic circuits, where the supply *was* connected to the drain of an NMOS transistor through a load resistor, and the name "V_{DD}" stuck.

Also note that ground is sometimes referred to as "V_{SS}" in CMOS and NMOS circuits. Some authors and most circuit manufacturers use "V_{CC}" as the symbol for the CMOS supply voltage, since this name is used in TTL circuits, which historically preceded CMOS. To get you used to both, we'll start using "V_{CC}" in Section 3.4.

Ideally, the functional behavior of the CMOS inverter circuit can be characterized by just two cases tabulated in Figure 3-10(b):

1. V_{IN} is 0.0 V. In this case, the bottom, *n*-channel transistor *Q1* is off, since its V_{gs} is 0, but the top, *p*-channel transistor *Q2* is on, since its V_{gs} is a large negative value (–5.0 V). Therefore, *Q2* presents only a small resistance between the power-supply terminal (V_{DD}, +5.0 V) and the output terminal (V_{OUT}), and the output voltage is 5.0 V.
2. V_{IN} is 5.0 V. Here, *Q1* is on, since its V_{gs} is a large positive value (+5.0 V), but *Q2* is off, since its V_{gs} is 0. Thus, *Q1* presents a small resistance between the output terminal and ground, and the output voltage is 0 V.

With the foregoing functional behavior, the circuit clearly behaves as a logical inverter, since a 0-volt input produces a 5-volt output, and vice versa.

Another way to visualize CMOS operation uses switches. As shown in Figure 3-11(a), the *n*-channel (bottom) transistor is modeled by a normally-open switch, and the *p*-channel (top) transistor by a normally-closed switch. Applying a HIGH voltage "pushes" each switch to the opposite of its normal state, as shown in (b).

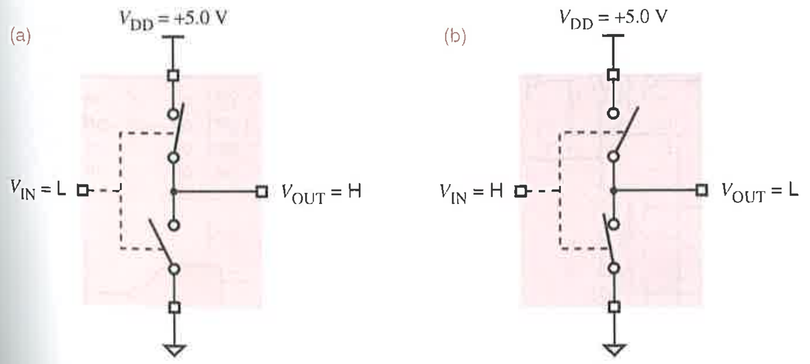


Figure 3-11 Switch model for CMOS inverter: (a) LOW input; (b) HIGH input.

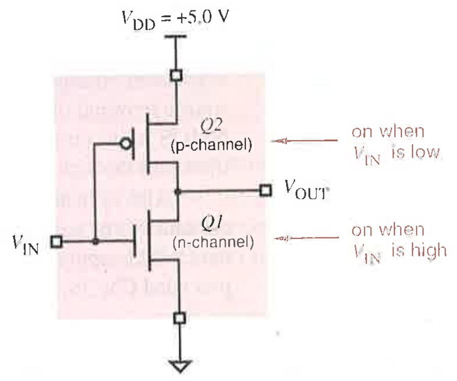


Figure 3-12
CMOS inverter logical operation.

The switch model gives rise to a way of drawing CMOS circuits that makes their logical behavior more readily apparent. As shown in Figure 3-12, different symbols are used for the *p*- and *n*-channel transistors to reflect their logical behavior. The *n*-channel transistor (*Q1*) is switched "on," and current flows between source and drain, when a HIGH voltage is applied to its gate; this seems natural enough. The *p*-channel transistor (*Q2*) has the opposite behavior. It is "on" when a LOW voltage is applied; the inversion bubble on its gate indicates this inverting behavior.

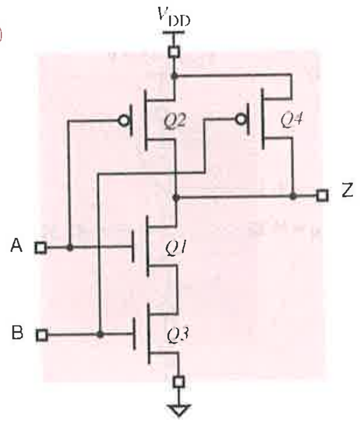
3.3.4 CMOS NAND and NOR Gates

Both NAND and NOR gates can be constructed using CMOS. A *k*-input gate uses *k p*-channel and *k n*-channel transistors.

Figure 3-13 shows a 2-input CMOS NAND gate. If either input is LOW, the output Z has a low-impedance connection to V_{DD} through the corresponding "on" *p*-channel transistor, and the path to ground is blocked by the correspond-

Figure 3-13
CMOS 2-input NAND gate:
(a) circuit diagram;
(b) function table;
(c) logic symbol.

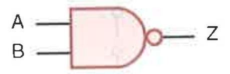
(a)



(b)

A	B	*Q1*	*Q2*	*Q3*	*Q4*	Z
L	L	off	on	off	on	H
L	H	off	on	on	off	H
H	L	on	off	off	on	H
H	H	on	off	on	off	L

(c)



on when V_{IN} is low

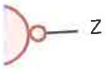
on when V_{IN} is high

uits that makes
3-12, different
ct their logical
l current flows
gate; this seems
behavior. It is
s gate indicates

MOS. A k-input

nput is LOW, the
e corresponding
the correspond-

Q3	*Q4*	Z
off	on	H
on	off	H
off	on	H
on	off	L



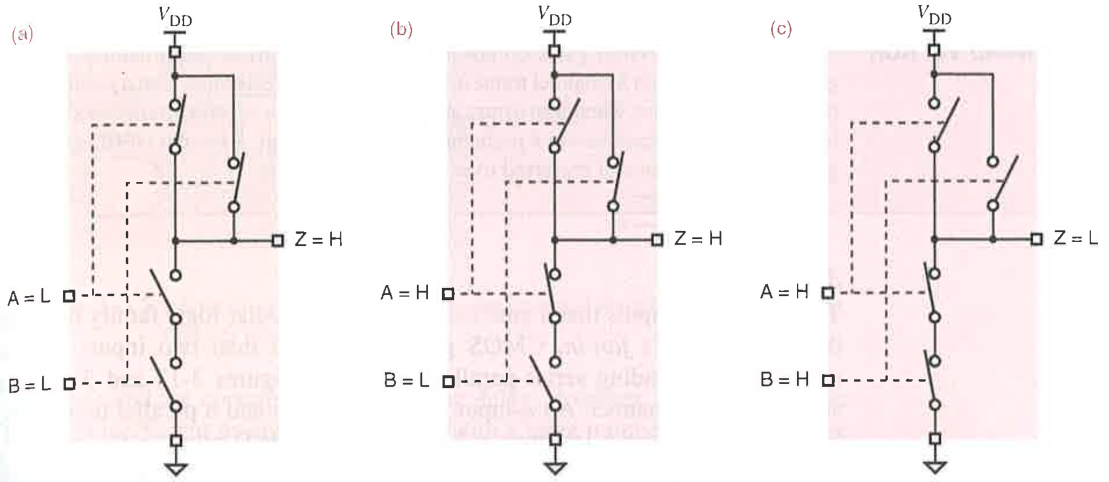
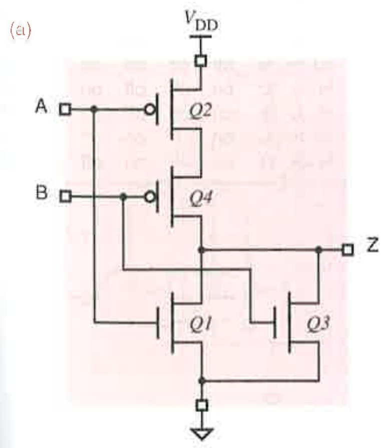


Figure 3-14 Switch model for CMOS 2-input NAND gate: (a) both inputs LOW; (b) one input HIGH; (c) both inputs HIGH.

ing "off" n-channel transistor. If both inputs are HIGH, the path to V_{DD} is blocked, and Z has a low-impedance connection to ground. Figure 3-14 shows the switch model for the NAND gate's operation.

Figure 3-15 shows a CMOS NOR gate. If both inputs are LOW, then the output Z has a low-impedance connection to V_{DD} through the "on" p-channel transistors, and the path to ground is blocked by the "off" n-channel transistors. If either input is HIGH, the path to V_{DD} is blocked, and Z has a low-impedance connection to ground.



(b)

A	B	*Q1*	*Q2*	*Q3*	*Q4*	Z
L	L	off	on	off	on	H
L	H	off	on	on	off	L
H	L	on	off	off	on	L
H	H	on	off	on	off	L

(c)

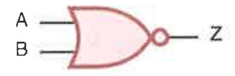


Figure 3-15 CMOS 2-input NOR gate: (a) circuit diagram; (b) function table; (c) logic symbol.

NAND VS. NOR CMOS NAND and NOR gates do not have identical electrical performance. For a given silicon area, an *n*-channel transistor has lower "on" resistance than a *p*-channel transistor. Therefore, when transistors are put in series, *k* *n*-channel transistors have lower "on" resistance than do *k* *p*-channel ones. As a result, a *k*-input NAND gate is generally faster than and preferred over a *k*-input NOR gate.

3.3.5 Fan-In

fan-in

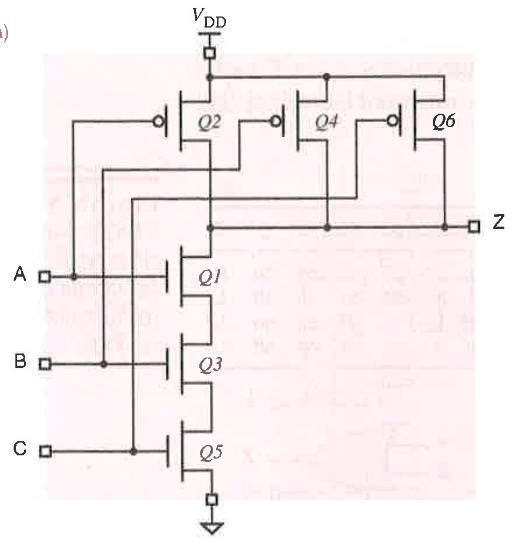
The number of inputs that a gate can have in a particular logic family is called the logic family's *fan-in*. CMOS gates with more than two inputs can be obtained by extending series-parallel designs on Figures 3-13 and 3-15 in a straightforward manner. An *n*-input gate has *n* series and *n* parallel transistors. For example, Figure 3-16 shows a 3-input CMOS NAND gate.

In principle, you could design a CMOS NAND or NOR gate with a very large number of inputs. In practice, however, the additive "on" resistance of series transistors limits the fan-in of CMOS gates, typically to 4 for NOR gates and 6 for NAND gates.

As the number of inputs is increased, designers of CMOS gate circuits may compensate by increasing the size of the series transistors to reduce their

Figure 3-16 CMOS 3-input NAND gate: (a) circuit diagram; (b) function table; (c) logic symbol.

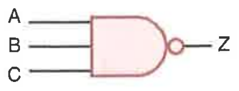
(a)



(b)

A	B	C	Q1	Q2	Q3	Q4	Q5	Q6	Z
L	L	L	off	on	off	on	off	on	H
L	L	H	off	on	off	on	on	off	H
L	H	L	off	on	on	off	off	on	H
L	H	H	off	on	on	off	on	off	H
H	L	L	on	off	off	on	off	on	H
H	L	H	on	off	off	on	on	off	H
H	H	L	on	off	on	off	off	on	H
H	H	H	on	off	on	off	on	off	L

(c)



nance. For a
n a *p*-channel
insistors have
NAND gate is

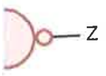
family is called
inputs can be
and 3-15 in a
allel transistors.

gate with a very
n" resistance of
4 for NOR gates

OS gate circuits
s to reduce their

) function table;

Q4	*Q5*	*Q6*	Z
on	off	on	H
on	on	off	H
off	off	on	H
off	on	off	H
on	off	on	H
on	on	off	H
off	off	on	H
off	on	off	L



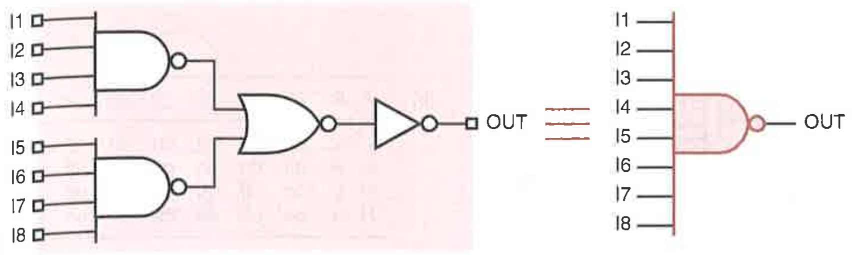


Figure 3-17 Logic diagram equivalent to the internal structure of an 8-input CMOS NAND gate.

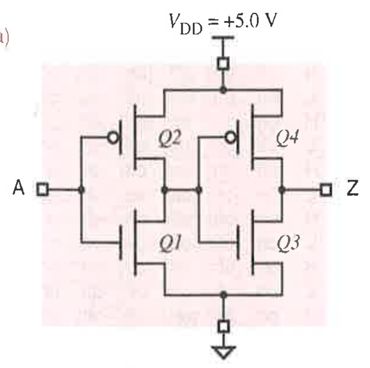
resistance and the corresponding switching delay. However, at some point this becomes inefficient or impractical. Gates with a large number of inputs can be made faster and smaller by cascading gates with fewer inputs. For example, Figure 3-17 shows the logical structure of an 8-input CMOS NAND gate. The total delay through a 4-input NAND, a 2-input NOR, and an inverter is typically less than the delay of a one-level 8-input NAND circuit.

3.3.6 Noninverting Gates

In CMOS, and in most other logic families, the simplest gates are inverters, and the next simplest are NAND gates and NOR gates. A logical inversion comes "for free," and it typically is not possible to design a noninverting gate with a smaller number of transistors than an inverting one.

CMOS noninverting buffers and AND and OR gates are obtained by connecting an inverter to the output of the corresponding inverting gate. For example, Figure 3-18 shows a noninverting buffer and Figure 3-19 shows an AND gate. Combining Figure 3-15(a) with an inverter yields an OR gate.

(a)



(b)

A	*Q1*	*Q2*	*Q3*	*Q4*	Z
L	off	on	on	off	L
H	on	off	off	on	H

(c)

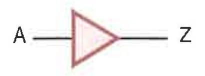
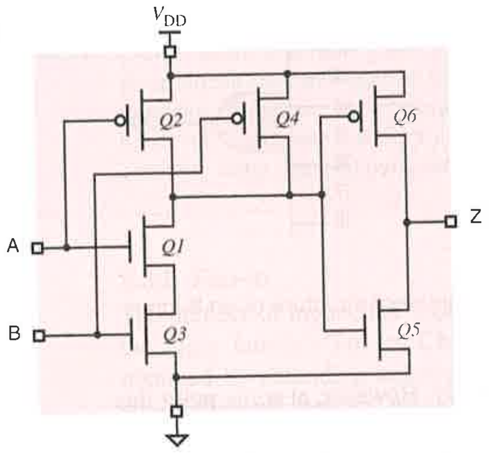


Figure 3-18 CMOS noninverting buffer: (a) circuit diagram; (b) function table; (c) logic symbol.

(a)



(b)

A	B	Q1	Q2	Q3	Q4	Q5	Q6	Z
L	L	off	on	off	on	on	off	L
L	H	off	on	on	off	on	off	L
H	L	on	off	off	on	on	off	L
H	H	on	off	on	off	off	on	H

(c)

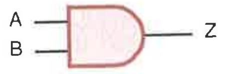


Figure 3-19 CMOS 2-input AND gate: (a) circuit diagram; (b) function table; (c) logic symbol.

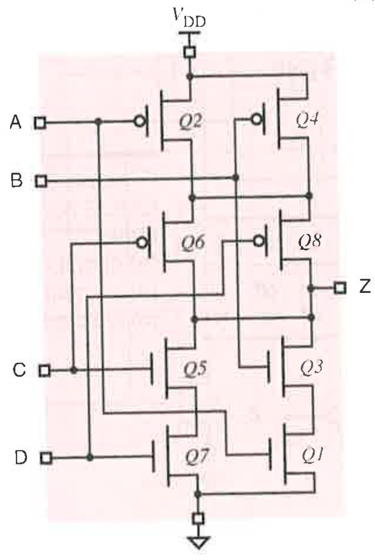
3.3.7 CMOS AND-OR-INVERT and OR-AND-INVERT Gates

AND-OR-INVERT (AOI) gate

CMOS circuits can perform two levels of logic with just a single "level" of transistors. For example, the circuit in Figure 3-20(a) is a 2-wide, 2-input CMOS *AND-OR-INVERT (AOI) gate*. The function table for this circuit is shown in (b) and a logic diagram for this function using AND and NOR gates is shown in

Figure 3-20 CMOS AND-OR-INVERT gate: (a) circuit diagram; (b) function table.

(a)



(b)

A	B	C	D	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Z
L	L	L	L	off	on	off	on	off	on	off	on	H
L	L	L	H	off	on	off	on	off	on	on	off	H
L	L	H	L	off	on	off	on	on	off	off	on	H
L	L	H	H	off	on	off	on	on	off	on	off	L
L	H	L	L	off	on	on	off	off	on	off	on	H
L	H	L	H	off	on	on	off	off	on	on	off	H
L	H	H	L	off	on	on	off	on	off	off	on	H
L	H	H	H	off	on	on	off	on	off	on	off	L
H	L	L	L	on	off	off	on	off	on	off	on	H
H	L	L	H	on	off	off	on	off	on	on	off	H
H	L	H	L	on	off	off	on	on	off	off	on	H
H	L	H	H	on	off	off	on	on	off	on	off	L
H	H	L	L	on	off	on	off	off	on	off	on	L
H	H	L	H	on	off	on	off	off	on	on	off	L
H	H	H	L	on	off	on	off	on	off	off	on	L
H	H	H	H	on	off	on	off	on	off	on	off	L

Q5	Q6	Z
on	off	L
on	off	L
on	off	L
off	on	H

— Z

unction table;

ngle "level" of
2-input CMOS
is shown in (b)
ıtes is shown in

Q6	Q7	Q8	Z
on	off	on	H
on	on	off	H
off	off	on	H
off	on	off	L
on	off	on	H
on	on	off	H
off	off	on	H
off	on	off	L
on	off	on	H
on	on	off	H
off	off	on	H
off	on	off	L
on	off	on	L
on	on	off	L
off	off	on	L
off	on	off	L

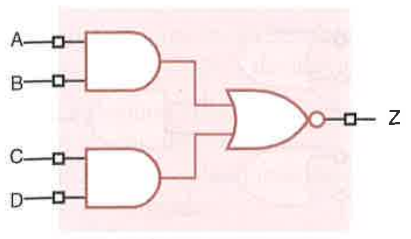


Figure 3-21
Logic diagram for CMOS AND-OR-INVERT gate.

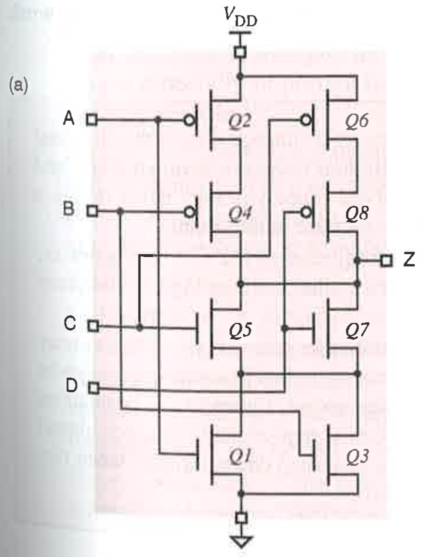
Figure 3-21. Transistors can be added to or removed from this circuit to obtain an AOI function with a different number of ANDs or a different number of inputs per AND.

The contents of each of the *Q1*–*Q8* columns in Figure 3-20(b) depends only on the input signal connected to the corresponding transistor's gate. The last column is constructed by examining each input combination and determining whether Z is connected to V_{DD} or to ground by "on" transistors for that input combination. Note that Z is never connected to *both* V_{DD} and ground for any input combination; in such a case the output would be a nonlogic value somewhere between LOW and HIGH, and the output structure would consume excessive power due to the low-impedance connection between V_{DD} and ground.

A circuit can also be designed to perform an OR-AND-INVERT function. For example, Figure 3-22(a) shows a 2-wide, 2-input CMOS *OR-AND-INVERT (OAI) gate*. The function table for this circuit is shown in (b); the values in each

OR-AND-INVERT (OAI) gate

Figure 3-22 CMOS OR-AND-INVERT gate: (a) circuit diagram; (b) function table.



(b)

A	B	C	D	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Z
L	L	L	L	off	on	off	on	off	on	off	on	H
L	L	L	H	off	on	off	on	off	on	on	off	H
L	L	H	L	off	on	off	on	on	off	off	on	H
L	L	H	H	off	on	off	on	on	off	on	off	H
L	H	L	L	off	on	on	off	off	on	off	on	H
L	H	L	H	off	on	on	off	off	on	on	off	L
L	H	H	L	off	on	on	off	on	off	off	on	L
L	H	H	H	off	on	on	off	on	off	on	off	L
H	L	L	L	on	off	off	on	off	on	off	on	H
H	L	L	H	on	off	off	on	off	on	on	off	L
H	L	H	L	on	off	off	on	on	off	off	on	L
H	L	H	H	on	off	off	on	on	off	on	off	L
H	H	L	L	on	off	on	off	off	on	off	on	H
H	H	L	H	on	off	on	off	off	on	on	off	L
H	H	H	L	on	off	on	off	on	off	off	on	L
H	H	H	H	on	off	on	off	on	off	on	off	L

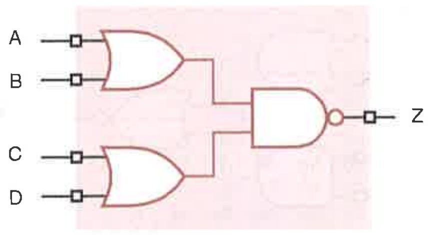


Figure 3-23
Logic diagram for CMOS OR-AND-INVERT gate.

column are determined just as we did for the CMOS AOI gate. A logic diagram for the OAI function using OR and NAND gates is shown in Figure 3-23.

The speed and other electrical characteristics of a CMOS AOI or OAI gate are quite comparable to those of a single CMOS NAND or NOR gate. As a result, these gates are very appealing because they can perform two levels of logic (AND-OR or OR-AND) with just one level of delay. Most digital designers don't bother to use AOI gates in their discrete designs. However, CMOS VLSI devices often use these gates internally, since many HDL synthesis tools can automatically convert AND/OR logic into AOI gates when appropriate.

3.4 Electrical Behavior of CMOS Circuits

The next three sections discuss electrical, not logical, aspects of CMOS circuit operation. It's important to understand this material when you design real circuits using CMOS or other logic families. Most of this material is aimed at providing a framework for ensuring that the "digital abstraction" is really valid for a given circuit. In particular, a circuit or system designer must provide adequate engineering design margins—insurance that the circuit will work properly even under the worst of conditions.

engineering design margins

IS ALL THIS REALLY NECESSARY?

The behaviors described in the next few sections are a consequence of the electrical design of the CMOS logic gates, including both their transistor-level structure and the analog properties of the transistors themselves. Since you may never design a logic gate yourself, you might think that these topics are unimportant.

However, these behaviors are also a consequence of the way that gates are selected and interconnected to form digital logic circuits, and creating such interconnections is exactly what a digital designer does.

Some technologies, such as field-programmable gate arrays (FPGAs), may hide the electrical consequences of on-chip interconnections from the designer, who can specify the design in a high-level language and use a software tool to generate an internal connection pattern that satisfies all electrical requirements. But it is almost *always* necessary for the designer to understand electrical characteristics when two or more chips are interconnected. So, please read on.

logic diagram
re 3-23.
AOI or OAI gate
ate. As a result,
levels of logic
designers don't
S VLSI devices
ls can automati-

of CMOS circuit
you design real
terial is aimed at
n" is really valid
ner must provide
circuit will work

of the electrical
vel structure and
y never design a
nt.
ay that gates are
ng such intercon-

s (FPGAs), may
the designer, who
ool to generate an
s. But it is almost
eristics when two

3.4.1 Overview

The topics that we examine in Sections 3.5–3.7 pertain to both the static and the dynamic behavior of CMOS devices and circuits:

- *Static behaviors.* These topics cover situations where a circuit's input and output signals are not changing. They include things like power consumption, the match-up and tolerances between input and output logic levels, and noise immunity.
- *Dynamic behaviors.* These topics cover situations where a circuit's input and output signals are changing. They include things like the extra power that is consumed as signals change, and the timing from an input-signal change to the resulting output-signal change.

When analyzing or designing a digital circuit, the designer must consider both static and dynamic behaviors. Most of the topics that we discuss in Sections 3.5–3.7 have both static and dynamic aspects. The topics include the following:

- *Logic voltage levels.* CMOS devices operating under normal conditions are guaranteed to produce output voltage levels within well-defined LOW and HIGH ranges. And they recognize LOW and HIGH input voltage levels over somewhat wider ranges. CMOS manufacturers specify these ranges and operating conditions very carefully to ensure compatibility among different devices in the same family and to provide a degree of interoperability (if you're careful) among devices in different families.
- *DC noise margins.* Positive DC noise margins ensure that the highest LOW voltage produced by an output is always lower than the highest voltage that an input can reliably interpret as LOW; and that the lowest HIGH voltage produced by an output is always higher than the lowest voltage that an input can reliably interpret as HIGH. A good understanding of noise margins is especially important in circuits that use devices from a number of different families.
- *Fanout.* This refers to the number and type of device inputs and other loads that are connected to a given output. If too many loads are connected to an output, the DC noise margins of the circuit may be inadequate. Fanout may also affect the speed at which the output changes from one state to another.
- *Speed.* The time that it takes a CMOS output to change from the LOW state to the HIGH state, or vice versa, depends on both the internal structure of the device and the characteristics of the other devices that it drives, even to the extent of being affected by the wire or printed-circuit-board traces connected to the output. We'll look at two separate components of "speed"—transition time and propagation delay.
- *Power consumption.* The power consumed by a CMOS device depends on a number of factors, including not only its internal structure, but also the

input signals that it receives, the other devices that it drives, and how often its output changes between LOW and HIGH.

- *Noise.* The main reason for providing engineering design margins is to ensure proper circuit operation in the presence of noise. Noise can be generated by a number of sources; several of them are listed below, from the least likely to the (perhaps surprisingly) most likely:
  - Cosmic rays.
  - Power-supply disturbances.
  - Magnetic fields from nearby equipment.
  - The switching action of the logic circuits themselves.
- *Electrostatic discharge.* Would you believe that you can destroy a CMOS device just by touching it? Ordinary "static electricity" can have a voltage potential of a thousand volts or more, enough to puncture and damage the thin insulating material between a MOS transistor's gate and its source and drain.
- *Open-drain outputs.* Some CMOS outputs omit the usual *p*-channel pull-up transistors. In the HIGH state, such an output behaves essentially like a "no-connection," which is useful in some applications.
- *Three-state outputs.* Some CMOS devices have an extra "output enable" control input that can be used to disable both the *p*-channel pull-up transistors and the *n*-channel pull-down transistors. Many such device outputs can be tied together to create a multisource bus, as long as the control logic is arranged so that at most one output is enabled at a time.

Among these topics, timing is probably the most important, since it's an area where designers typically must spend the most time, even if they're otherwise working at a strictly "logical" level. So, we'll keep coming back to timing in later chapters, even after we've dismissed the other electrical topics discussed here.

3.4.2 Data Sheets and Specifications

data sheet

The manufacturers of real-world devices provide *data sheets* that specify the devices' logical and electrical characteristics. The electrical specifications portion of a minimal data sheet for a simple CMOS device, the 54/74HC00 quadruple NAND gate, is shown in Table 3-3. ("Quadruple" means there are four gates in the same chip and package.) Different manufacturers typically specify additional parameters, and they may vary in how they specify even the "standard" parameters shown in the table. For example, they usually also show the test circuits and waveforms that they use to define various parameters, as in Figure 3-24. Note that this figure contains information for some additional parameters beyond those used with the 54/74HC00.

and how often

ı margins is to
. Noise can be
ed below, from

lestroy a CMOS
n have a voltage
and damage the
nd its source and

l *p*-channel pull-
essentially like a

ı "output enable"
el pull-up transis-
:h device outputs
s the control logic
e.

rtant, since it's an
n if they're other-
ing back to timing
al topics discussed

ts that specify the
ical specifications
:e, the 54/74HC00
ıeans there are four
rs typically specify
specify even the
y usually also show
us parameters, as in
or some additional

Table 3-3 Manufacturer's data sheet for a typical CMOS device, a 54/74HC00 quad NAND gate.

DC ELECTRICAL CHARACTERISTICS OVER OPERATING RANGE
The following conditions apply unless otherwise specified:
Commercial: $T_A = -40°C$ to $+85°C$, $V_{CC} = 5.0\ V \pm 5\%$; Military: $T_A = -55°C$ to $+125°C$, $V_{CC} = 5.0\ V \pm 10\%$

Sym.	Parameter	Test Conditions[(1)]		Min.	Typ.[(2)]	Max.	Unit
V_{IH}	Input HIGH level	Guaranteed logic HIGH level		3.15	—	—	V
V_{IL}	Input LOW level	Guaranteed logic LOW level		—	—	1.35	V
I_{IH}	Input HIGH current	V_{CC} = Max., $V_I = V_{CC}$		—	—	1	μA
I_{IL}	Input LOW current	V_{CC} = Max., V_I = 0 V		—	—	−1	μA
V_{IK}	Clamp diode voltage	V_{CC} = Min., I_N = −18 mA		—	−0.7	−1.2	V
I_{IOS}	Short-circuit current	V_{CC} = Max.,[(3)] V_O = GND		—	—	−35	mA
V_{OH}	Output HIGH voltage	V_{CC} = Min., $V_{IN} = V_{IL}$	$I_{OH} = -20\ \mu A$	4.4	4.499	—	V
			$I_{OH} = -4$ mA	3.84	4.3	—	V
V_{OL}	Output LOW voltage	V_{CC} = Min., $V_{IN} = V_{IH}$	$I_{OL} = 20\ \mu A$	—	.001	0.1	V
			$I_{OL} = 4$ mA		0.17	0.33	V
I_{CC}	Quiescent power supply current	V_{CC} = Max. V_{IN} = GND or V_{CC}, $I_O = 0$		—	2	10	μA

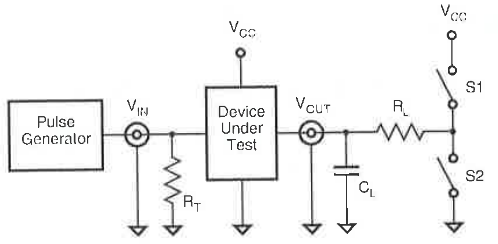
SWITCHING CHARACTERISTICS OVER OPERATING RANGE, C_L = 50 pF

Sym.	Parameter[(4)]	Test Conditions	Min.	Typ.	Max.	Unit
t_{PD}	Propagation delay	A or B to Y	—	9	19	ns
C_I	Input capacitance	V_{IN} = 0 V	—	3	10	pF
C_{pd}	Power dissipation capacitance per gate	No load	—	22	—	pF

NOTES:
1. *For conditions shown as Max. or Min., use appropriate value specified under Electrical Characteristics.*
2. *Typical values are at V_{CC} = 5.0 V, +25 °C ambient.*
3. *Not more than one output should be shorted at a time. Duration of short-circuit test should not exceed one second.*
4. *This parameter is guaranteed but not tested.*

WHAT'S IN A NUMBER? Two different prefixes, "74" and "54," are used in the part numbers of CMOS and TTL devices. These prefixes simply distinguish between commercial and military versions. A 74HC00 is the commercial part and the 54HC00 is the military version.

TEST CIRCUIT FOR ALL OUTPUTS

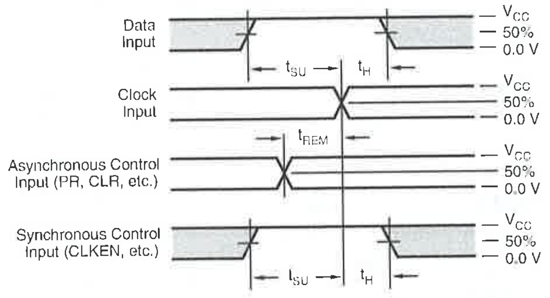


LOADING

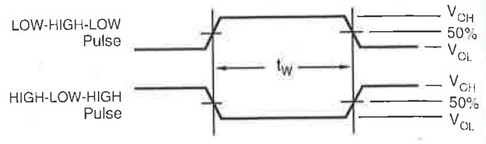
Parameter		R_L	C_L	S1	S2
t_{on}	t_{pZH}	1 KΩ	50 pF or 150 pF	Open	Closed
	t_{pZL}			Closed	Open
t_{dis}	t_{pHZ}	1 KΩ	50 pF or 150 pF	Open	Closed
	t_{pLZ}			Closed	Open
t_{pd}		—	50 pF or 150 pF	Open	Open

DEFINITIONS:
C_L = Load capacitance, includes jig and probe capacitance.
R_T = Termination resistance, should equal Z_{OUT} of the Pulse Generator.

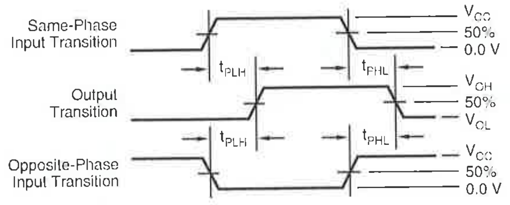
SETUP, HOLD, AND RELEASE TIMES


Data Input
Clock Input
Asynchronous Control Input (PR, CLR, etc.)
Synchronous Control Input (CLKEN, etc.)
t_SU
t_H
t_REM
V_CC
50%
0.0 V

PULSE WIDTH



PROPAGATION DELAY



THREE-STATE ENABLE AND DISABLE TIMES

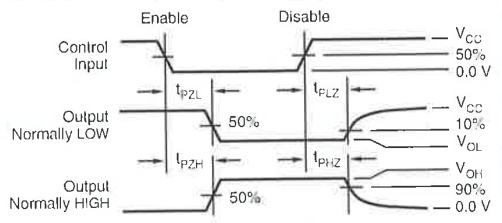
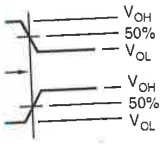


Figure 3-24 Test circuits and waveforms for HC-series logic.

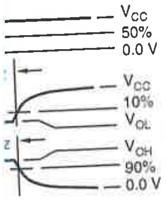
Most of the terms in the data sheet and the waveforms in the figure are probably meaningless to you at this point. However, after reading the next three sections you should know enough about the electrical characteristics of CMOS circuits that you'll be able to understand the salient points of this or any other data sheet. As a digital designer, you'll need this knowledge to create reliable and robust real-world circuits and systems.

DON'T BE AFRAID Computer science students and other non-EE readers should not have undue fear of the material in the next three sections. Only a basic understanding of electronics, at about the level of Ohm's law, is required.

nce.
Pulse Generator.



TIMES



in the figure are
ling the next three
eristics of CMOS
this or any other
to create reliable

ave undue fear of
of electronics, at

3.5 CMOS Static Electrical Behavior

This section discusses the "DC" or static behavior of CMOS circuits, that is, the circuits' behavior when inputs and outputs are not changing. Electrical engineers also call this "steady-state" behavior, because the electrical state of the inputs is not changing.

3.5.1 Logic Levels and Noise Margins

The table in Figure 3-10(b) on page 88 defined the CMOS inverter's behavior only at two discrete input voltages; other input voltages may yield different output voltages. The complete input-output transfer characteristic of a particular inverter can be described by a graph such as Figure 3-25. In this graph, the input voltage is varied from 0 to 5 V, as shown on the *X* axis; and the *Y* axis plots the output voltage.

If we believed the curve in Figure 3-25, we could define a CMOS LOW input level as any voltage under 2.4 V, and a HIGH input level as anything over 2.6 V. Only when the input is between 2.4 and 2.6 V does the inverter produce a nonlogic output voltage under this definition.

Unfortunately, the typical transfer characteristic shown in Figure 3-25 is just that—typical, but not guaranteed. It varies greatly under different conditions such as power-supply voltage, temperature, and output loading. For example, the transition in the middle of the curve may become more or less steep, and it may shift to the left or the right. The transfer characteristic may even vary depending on when the device was fabricated. For example, after months of trying to figure out why gates made on some days were good and on other days were bad, legend has it that one manufacturer discovered that the bad gates were victims of airborne contamination by a particularly noxious perfume worn by one of its production-line workers!

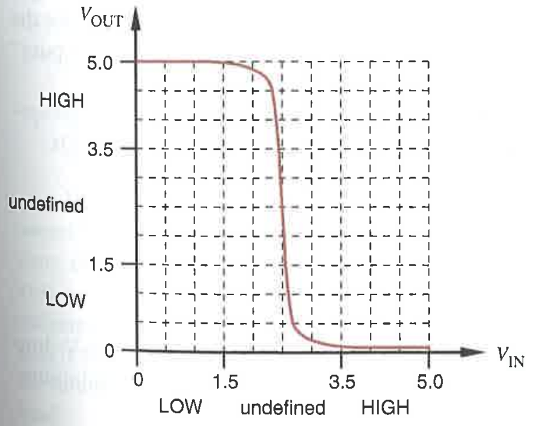


Figure 3-25
Typical input-output transfer characteristic of a CMOS inverter.

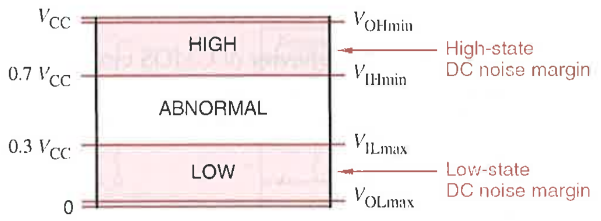


Figure 3-26 Logic levels and noise margins for the HC-series CMOS logic family.

Sound engineering practice dictates that we use specifications (or "specs") for LOW and HIGH that are more conservative. Conservative logic-level specs for a typical CMOS logic family (HC-series) are depicted in Figure 3-26. These parameters are specified by CMOS device manufacturers in data sheets like Table 3-3 on page 99, and are defined as follows:

V_{OHmin}	The minimum output voltage produced in the HIGH state.
V_{IHmin}	The minimum input voltage guaranteed to be recognized as a HIGH.
V_{ILmax}	The maximum input voltage guaranteed to be recognized as a LOW.
V_{OLmax}	The maximum output voltage produced in the LOW state.

The input voltages are determined mainly by switching thresholds of the two transistors, while the output voltages are determined mainly by the "on" resistance of the transistors.

All of the parameters in Figure 3-26 are guaranteed by CMOS manufacturers over a range of temperature and output loading. Parameters are also guaranteed over a range of power-supply voltage V_{CC}, typically 5.0 V±10%.

The data sheet in Table 3-3 specifies values for each of these parameters for HC-series CMOS. Notice that there are two values specified for V_{OHmin} and V_{OLmax}, depending on whether the output current (I_{OH} or I_{OL}) is large or small. When the device outputs are connected only to other CMOS inputs, the output current is low (e.g., $I_{OL} \le 20\ \mu A$), so there's very little voltage drop across the output transistors. In the next few subsections we'll focus on these "pure" CMOS applications.

power-supply rails

The power-supply voltage V_{CC} and ground are often called the *power-supply rails*. CMOS levels are typically a function of the power-supply rails:

V_{OHmin}	$V_{CC} - 0.1$ V
V_{IHmin}	70% of V_{CC}
V_{ILmax}	30% of V_{CC}
V_{OLmax}	ground + 0.1 V

Notice in Table 3-3 that V_{OHmin} is specified as 4.4 V. This is only a 0.1-V drop from V_{CC}, since the worst-case number is specified with V_{CC} at its minimum value of 5.0 − 10% = 4.5 V.

DC noise margin

DC noise margin is a measure of how much noise it takes to corrupt a worst-case output voltage into a value that may not be recognized properly by an input. For example, with HC-series CMOS in the LOW state, V_{ILmax} (1.35 V) exceeds V_{OLmax} (0.1 V) by 1.25 V, so the LOW-state DC noise margin is 1.25 V. With HC-series CMOS in the HIGH state, V_{IHmin} (3.15 V) is 1.25 V lower than V_{OHmin} (4.4 V), so there is 1.25 V of HIGH-state DC noise margin as well. In general, CMOS outputs have excellent DC noise margins when driving other CMOS inputs.

Regardless of the voltage applied to the input of a CMOS inverter, the input consumes very little current, only the leakage current of the two transistors' gates. The maximum amount of leakage current that can flow is specified by the device manufacturer:

I_{IH}	The maximum current that flows into the input in the HIGH state.
I_{IL}	The maximum current that flows into the input in the LOW state.

The input current shown in Table 3-3 for the 'HC00 is only $\pm 1\ \mu\text{A}$. Thus, it takes very little power to maintain a CMOS input in one state or the other. This is in sharp contrast to older bipolar logic circuits like TTL and ECL, whose inputs may consume significant current (and power) in one or both states.

3.5.2 Circuit Behavior with Resistive Loads

resistive load
DC load

As mentioned previously, CMOS gate inputs have very high impedance and consume very little current from the circuits that drive them. There are other devices, however, which require nontrivial amounts of current to operate. When such a device is connected to a CMOS output, we call it a *resistive load* or a *DC load.* Here are some examples of resistive loads:

- Discrete resistors may be included to provide transmission-line termination, discussed in Section Zo at DDPPonline.
- Discrete resistors may not really be present in the circuit, but the load presented by one or more TTL or other non-CMOS inputs may be modeled by a simple resistor network.
- The resistors may be part of or may model a current-consuming device such as a light-emitting diode (LED) or a relay coil.

When the output of a CMOS circuit is connected to a resistive load, the output behavior is not nearly as ideal as we described previously. In either logic state, the CMOS output transistor that is "on" has a nonzero resistance, and a load connected to the output terminal will cause a voltage drop across this resistance. Thus, in the LOW state, the output voltage may be somewhat higher than 0.1 V, and in the HIGH state it may be lower than 4.4 V. The easiest way to see how this happens is look at a resistive model of the CMOS circuit and its load.

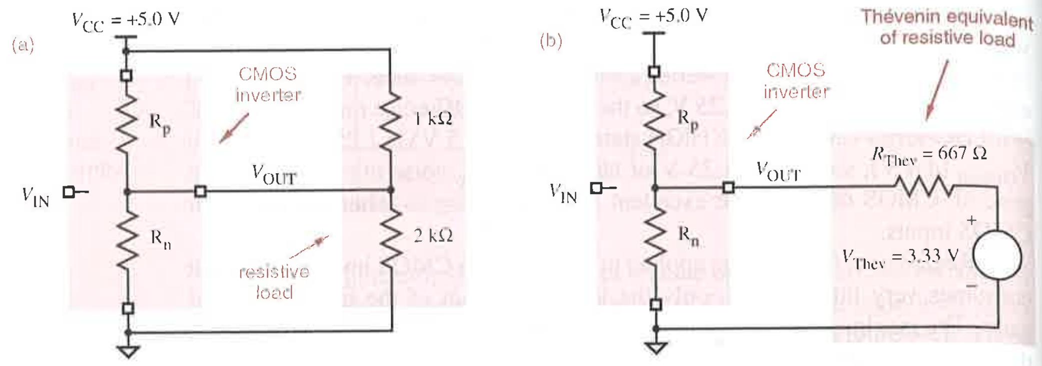


Figure 3-27 Resistive model of a CMOS inverter with a resistive load: (a) showing actual load circuit; (b) using Thévenin equivalent of load.

Figure 3-27(a) shows the resistive model. The *p*-channel and *n*-channel transistors have resistances R_p and R_n, respectively. In normal operation, one resistance is high (> 1 MΩ) and the other is low (perhaps 100 Ω), depending on whether the input voltage is HIGH or LOW. The load in this circuit consists of two resistors attached to the supply rails; a real circuit may have any resistor values, or an even more complex resistive network. In any case, a resistive load, consisting only of resistors and voltage sources, can always be modeled by a Thévenin equivalent network, such as the one shown in Figure 3-27(b).

When the CMOS inverter has a HIGH input, the output should be LOW; the actual output voltage can be predicted using the resistive model shown in Figure 3-28. The *p*-channel transistor is "off" and has a very high resistance,

REMEMBERING THÉVENIN

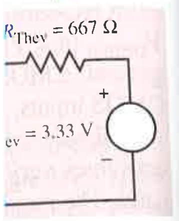
Any two-terminal circuit consisting of only voltage sources and resistors can be modeled by a *Thévenin equivalent* consisting of a single voltage source in series with a single resistor. The *Thévenin voltage* is the open-circuit voltage of the original circuit, and the *Thévenin resistance* is the Thévenin voltage divided by the short-circuit current of the original circuit.

In the example of Figure 3-27, the Thévenin voltage of the resistive load, including its connection to V_{CC}, is established by the 1-kΩ and 2-kΩ resistors, which form a voltage divider:

$$V_{Thev} = \frac{2\text{ k}\Omega}{2\text{ k}\Omega + 1\text{k}\Omega} \cdot 5.0\text{ V} = 3.33\text{ V}$$

The short-circuit current is (5.0 V)/(1 kΩ) = 5 mA, so the Thévenin resistance is (3.33 V) / (5 mA) = 667 Ω. Readers who are electrical engineers may recognize this as the parallel resistance of the 1-kΩ and 2-kΩ resistors.

évenin equivalent
of resistive load



stive load:
nin equivalent

el and *n*-channel
al operation, one
Ω), depending on
circuit consists of
have any resistor
e, a resistive load,
be modeled by a
e 3-27(b).
hould be LOW; the
model shown in
ry high resistance,

d resistors can be
urce in series with
of the original cir-
by the short-circuit

the resistive load,
kΩ resistors, which

Thévenin resistance
s may recognize this

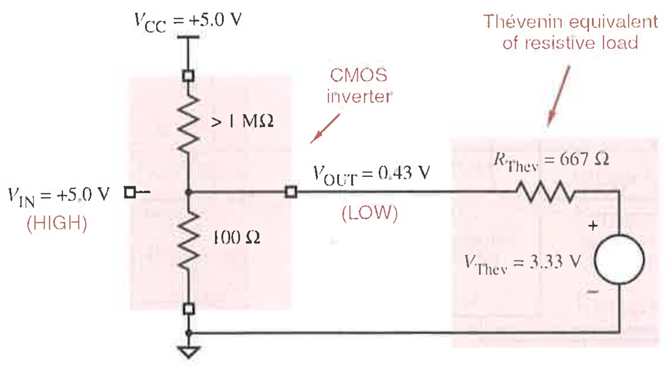


Figure 3-28
Resistive model for CMOS LOW output with resistive load.

high enough to be negligible in the calculations that follow. The *n*-channel transistor is "on" and has a low resistance, which we assume to be 100 Ω. (The actual "on" resistance depends on the CMOS family and other characteristics such as operating temperature and whether or not the device was manufactured on a good day.) The "on" transistor and the Thévenin-equivalent resistor R_{Thev} in Figure 3-28 form a simple voltage divider. The resulting output voltage can be calculated as follows:

$$\begin{aligned} V_{OUT} &= 3.33\text{ V}\cdot[100/(100+667)] \\ &= 0.43\text{ V} \end{aligned}$$

Similarly, when the inverter has a LOW input, the output should be HIGH, and the actual output voltage can be predicted with the model in Figure 3-29. We'll assume that the *p*-channel transistor's "on" resistance is 200 Ω. Once again, the "on" transistor and the Thévenin-equivalent resistor R_{Thev} in the figure form a simple voltage divider, and the resulting output voltage can be calculated as follows:

$$\begin{aligned} V_{OUT} &= 3.33\text{ V} + (5\text{ V} - 3.33\text{ V})\cdot[667/(200+667)] \\ &= 4.61\text{ V} \end{aligned}$$

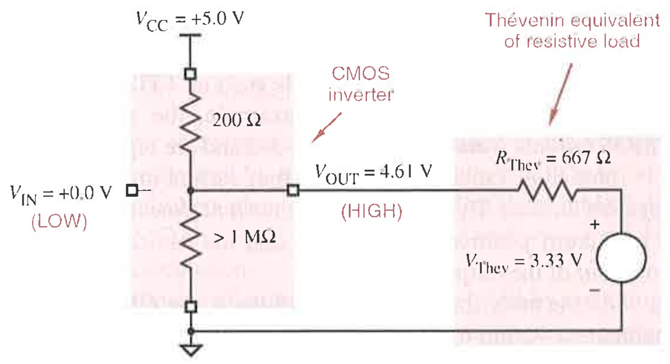


Figure 3-29
Resistive model for CMOS HIGH output with resistive load.

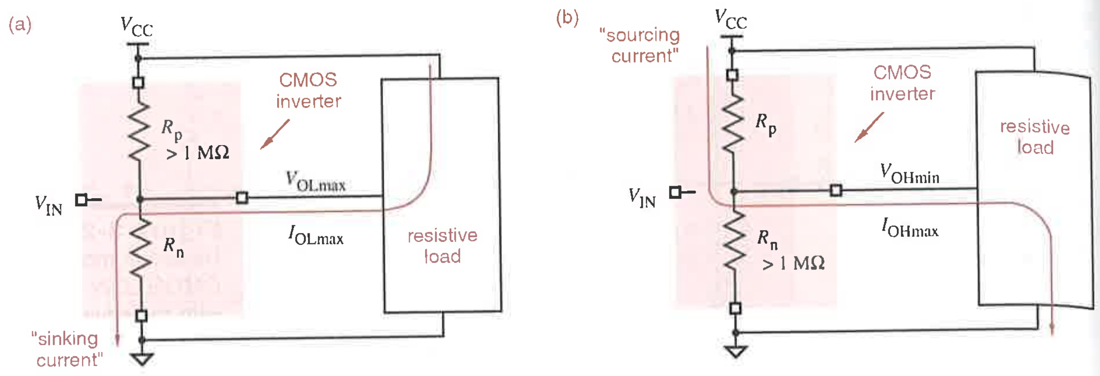


Figure 3-30 Circuit definitions of (a) I_{OLmax}; (b) I_{OHmax}.

In practice, it's seldom necessary to calculate output voltages as in the preceding examples. In fact, IC manufacturers usually don't specify the equivalent resistances of the "on" transistors, so you wouldn't have the necessary information to make the calculation anyway. Instead, IC manufacturers specify a maximum load for the output in each state (HIGH or LOW), and guarantee a worst-case output voltage for that load. The load is specified in terms of current:

I_{OLmax}	The maximum current that the output can sink in the LOW state while still maintaining an output voltage no greater than V_{OLmax}.
I_{OHmax}	The maximum current that the output can source in the HIGH state while still maintaining an output voltage no less than V_{OHmin}.

sinking current

sourcing current

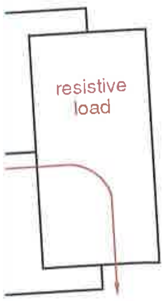
These definitions are illustrated in Figure 3-30. A device output is said to *sink current* when current flows from the power supply, through the load, and through the device output to ground as in (a). The output is said to *source current* when current flows from the power supply, out of the device output, and through the load to ground as in (b).

Most CMOS devices have two sets of loading specifications. One set is for "CMOS loads," where the device output is connected to other CMOS inputs, which consume very little current. The other set is for "TTL loads," where the output is connected to resistive loads such as TTL inputs or other devices that consume significant current. For example, the specifications for HC-series CMOS outputs were shown in Table 3-3 and are repeated in Table 3-4.

current flow

Notice in Table 3-4 that the output current in the HIGH state is shown as a negative number. By convention, the *current flow* measured at a device terminal is positive if positive current flows *into* the device; in the HIGH state, current flows *out* of the output terminal.

As the table shows, with CMOS loads, the CMOS gate's output voltage is maintained within 0.1 V of the power-supply rail. With TTL loads, the output



ɟes as in the pre-
y the equivalent
ɛessary informa-
turers specify a
and guarantee a
terms of current:

LOW state while
max·
HIGH state while
ı·

put is said to *sink*
gh the load, and
d to *source current*
utput, and through

ions. One set is for
her CMOS inputs,
ˌ loads," where the
r other devices that
ions for HC-series
Table 3-4.
d state is shown as a
at a device terminal
HIGH state, current

te's output voltage is
TL loads, the output

Table 3-4 Output loading specifications for HC-series CMOS with a 5V±10% supply.

	CMOS Load		***TTL Load***	
Parameter	***Name***	***Value***	***Name***	***Value***
Maximum LOW-state output current (mA)	I_{OLmaxC}	0.02	I_{OLmaxT}	4.0
Maximum LOW-state output voltage (V)	V_{OLmaxC}	0.1	V_{OLmaxT}	0.33
Maximum HIGH-state output current (mA)	I_{OHmaxC}	-0.02	I_{OHmaxT}	-4.0
Minimum HIGH-state output voltage (V)	V_{OHminC}	4.4	V_{OHminT}	3.84

voltage may degrade quite a bit. Also notice that for the same output current (±4 mA) the maximum voltage drop with respect to the power-supply rail is twice as much in the HIGH state (0.66 V) as in the LOW state (0.33 V). This suggests that the *p*-channel transistors in HC-series CMOS have a higher "on" resistance than the *n*-channel transistors do. This is natural, since in any CMOS circuit, a *p*-channel transistor has over twice the "on" resistance of an *n*-channel transistor with the same area. Equal voltage drops in both states could be obtained by making the *p*-channel transistors much larger than the *n*-channel transistors, but for various reasons this was not done.

Ohm's law can be used to determine how much current an output sources or sinks in a given situation. In Figure 3-28 on page 105, the "on" *n*-channel transistor modeled by a 100-Ω resistor has a 0.43-V drop across it; therefore it sinks (0.43 V)/(100 Ω) = 4.3 mA of current. Similarly, the "on" *p*-channel transistor in Figure 3-29 sources (0.39 V)/(200 Ω) = 1.95 mA.

The actual "on" resistances of CMOS output transistors usually aren't published, so it's generally not possible to use the exact models of the previous paragraphs. However, you can estimate "on" resistances using the following equations, which rely on specifications that are always published:

$$R_{\text{p(on)}} \approx \frac{V_{\text{DD}} - V_{\text{OHminT}}}{|I_{\text{OHmaxT}}|}$$

$$R_{\text{n(on)}} \approx \frac{V_{\text{OLmaxT}}}{I_{\text{OLmaxT}}}$$

These equations use Ohm's law to compute the "on" resistance as the voltage drop across the "on" transistor divided by the current through it with a worst-case resistive load. Using the numbers given for HC-series CMOS in Table 3-4, we can calculate $R_{\text{p(on)}} \approx$ 165 Ω and $R_{\text{n(on)}} \approx$ 82.5 Ω. Note that V_{DD} = 4.5 V (the minimum value) for this calculation.

Very good *worst-case* estimates of output current can be made by assuming that there is *no* voltage drop across the "on" transistor. This assumption simpli-

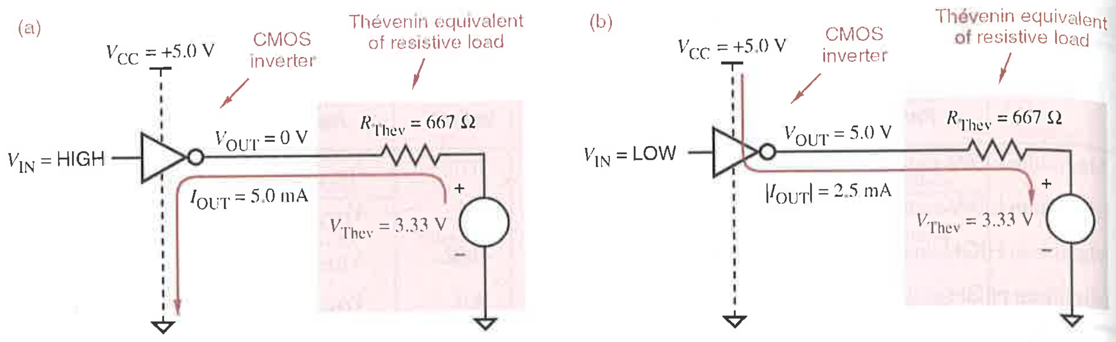


Figure 3-31 Estimating sink and source current: (a) output LOW; (b) output HIGH.

fies the analysis, and yields a conservative result that is almost always good enough for practical purposes. For example, Figure 3-31 shows a CMOS inverter driving the same Thévenin-equivalent load that we've used in previous examples. The resistive model of the output structure is not shown, because it is no longer needed; we assume that there is no voltage drop across the "on" CMOS transistor. In (a), with the output LOW, the entire 3.33-V Thévenin-equivalent voltage source appears across R_{Thev}, and the estimated sink current is $(3.33\text{ V})/(667\ \Omega) = 5.0$ mA. In (b), with the output HIGH and assuming a 5.0-V supply, the voltage drop across R_{Thev} is 1.67 V, and the estimated source current is $(1.67\text{ V})/(667\ \Omega) = 2.5$ mA.

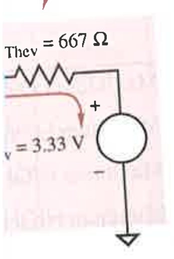
An important feature of the CMOS inverter (or any CMOS circuit) is that the output structure by itself consumes very little current in either state, HIGH or LOW. In either state, one of the transistors is in the high-impedance "off" state. All of the current flow that we've been talking about occurs when a resistive load is connected to the CMOS output. If there's no load, then there's no current flow, and the power consumption is zero. With a load, however, current flows through both the load and the "on" transistor, and power is consumed in both.

3.5.3 Circuit Behavior with Nonideal Inputs

So far, we have assumed that the HIGH and LOW inputs to a CMOS circuit are ideal voltages, very close to the power-supply rails. However, the behavior of a real CMOS inverter circuit depends on the input voltage as well as on the characteristics of the load. If the input voltage is not close to the power-supply rail, then the "on" transistor may not be fully "on" and its resistance may increase. Likewise, the "off" transistor may not be fully "off" and its resistance may be quite a bit less than one megohm. These two effects combine to move the output voltage away from the power-supply rail.

For example, Figure 3-32(a) shows a CMOS inverter's possible behavior with a 1.5-V input. The *p*-channel transistor's resistance has doubled at this

ıvenin equivalent
ıf resistive load



(b) output HIGH.

ost always good
shows a CMOS
used in previous
own, because it is
across the "on"
.33-V Thévenin-
ted sink current is
assuming a 5.0-V
ted source current

IOS circuit) is that
ther state, HIGH or
edance "off" state.
hen a resistive load
e's no current flow,
rrent flows through
d in both.

a CMOS circuit are
er, the behavior of a
e as well as on the
to the power-supply
d its resistance may
off" and its resistance
cts combine to move

er's possible behavior
e has doubled at this

PIDDLING CURRENT?

As we've stated elsewhere, an "off" transistor's resistance is over one megohm, but it's not infinite. Therefore, a very tiny leakage current actually does flow in "off" transistors, and the CMOS output structure does have a correspondingly tiny but nonzero power consumption. In most applications, this power consumption is tiny enough to ignore.

However, leakage current and the corresponding power consumption can be significant in "standby mode" in battery-powered devices, such as mobile phones and laptop computers. Leakage current can also be a significant concern in the densest, highest-performance IC technologies, with tens of millions of transistors per chip. As the transistors get smaller, their individual leakage currents increase, just as the total number of transistors per chip is increasing. The net effect is that as much as half of the chip's total power consumption may result from leakage.

point, and the *n*-channel transistor is beginning to turn on. (These values are just assumed for the purposes of illustration; the actual values depend on the detailed characteristics of the transistors.)

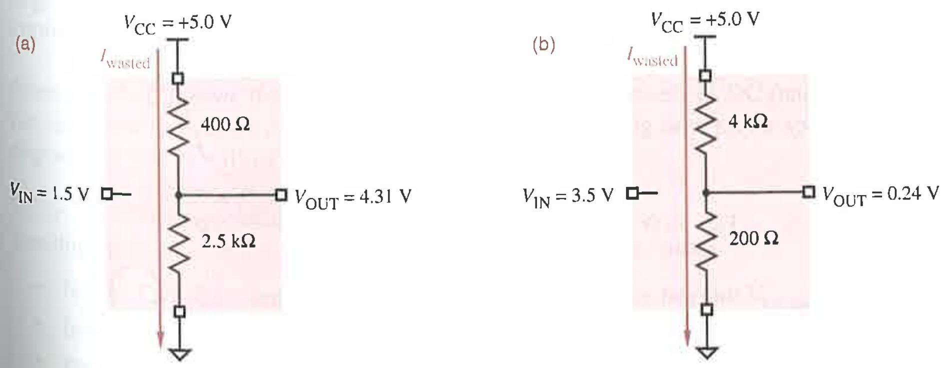
In the figure, the output at 4.31 V is still well within the valid range for a HIGH signal, but not quite the ideal of 5.0 V. Similarly, with a 3.5-V input in (b), the LOW output is 0.24 V, not 0 V. The slight degradation of output voltage is generally tolerable; what's worse is that the output structure is now consuming a nontrivial amount of power. The current flow with the 1.5-V input is

$$I_{\text{wasted}} = 5.0\ \text{V}/(400\ \Omega + 2.5\ \text{k}\Omega) = 1.72\ \text{mA}$$

and the power consumption is

$$P_{\text{wasted}} = 5.0\ \text{V} \cdot I_{\text{wasted}} = 8.62\ \text{mW}$$

Figure 3-32 CMOS inverter with nonideal input voltages: (a) equivalent circuit with 1.5-V input; (b) equivalent circuit with 3.5-V input.



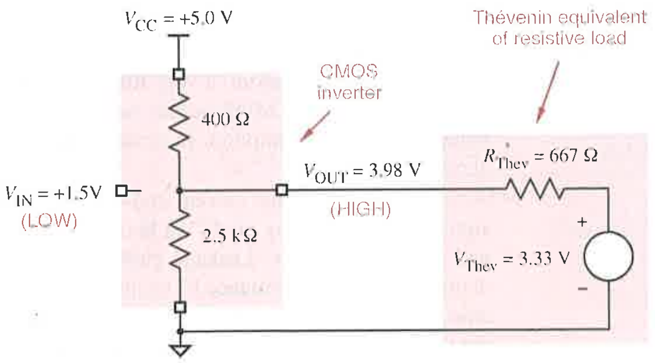


Figure 3-33
CMOS inverter with load and nonideal 1.5-V input.

The output voltage of a CMOS inverter deteriorates further with a resistive load. Such a load may exist for any of a variety of reasons discussed previously. Figure 3-33 shows the CMOS inverter's possible behavior with a resistive load. With a 1.5-V input, the output at 3.98 V is still within the valid range for a HIGH signal, but it is still farther from the ideal of 5.0 V. Similarly, with a 3.5-V input as shown in Figure 3-34, the LOW output is 0.93 V, not 0 V.

In "pure" CMOS systems, all of the logic devices in a circuit are CMOS. Since CMOS inputs have a very high impedance, they present very little resistive load to the CMOS outputs that drive them. Therefore, the CMOS output levels all remain very close to the power-supply rails (0 V and 5 V), and none of the devices waste power in their output structures. In "non-pure" CMOS systems, additional power can be consumed in two ways:

- If TTL outputs or other nonideal logic signals are connected to CMOS inputs, then the CMOS outputs use power in the way depicted in this subsection; this is formalized in the box on page 145.
- If TTL inputs or other resistive loads are connected to CMOS outputs, then the CMOS outputs use power in the way depicted in the preceding subsection.

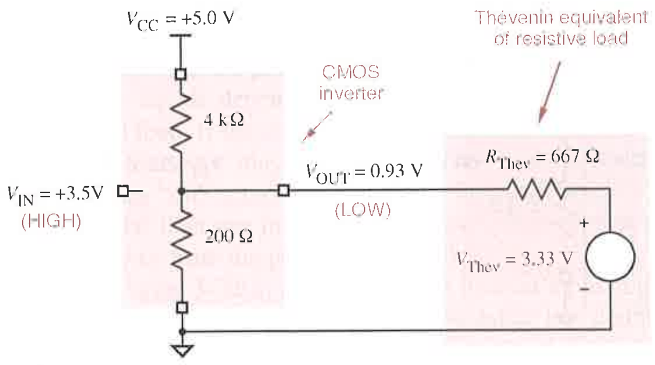


Figure 3-34
CMOS inverter with load and nonideal 3.5-V input.

3.5.4 Fanout

fanout

The *fanout* of a logic gate is the number of inputs that the gate can drive without exceeding its worst-case loading specifications. The fanout depends not only on the characteristics of the output, but also on the inputs that it is driving. Fanout must be examined for both possible output states, HIGH and LOW.

LOW-state fanout

HIGH-state fanout

For example, we showed in Table 3-4 on page 107 that the maximum LOW-state output current I_{OLmaxC} for an HC-series CMOS gate driving CMOS inputs is 0.02 mA (20 μA). We also stated previously that the maximum input current I_{Imax} for an HC-series CMOS input in any state is ± 1 μA. Therefore, the *LOW-state fanout* for an HC-series output driving HC-series inputs is 20. Table 3-4 also showed that the maximum HIGH-state output current I_{OHmaxC} is -0.02 mA (-20 μA) Therefore, the *HIGH-state fanout* for an HC-series output driving HC-series inputs is also 20.

overall fanout

Note that the HIGH-state and LOW-state fanouts of a gate aren't necessarily equal. In general, the *overall fanout* of a gate is the minimum of its HIGH-state and LOW-state fanouts, 20 in the foregoing example.

In the fanout example that we just completed, we assumed that we needed to maintain the gate's output at CMOS levels, that is, within 0.1 V of the power-supply rails. If we were willing to live with somewhat degraded, TTL output levels, then we could use I_{OLmaxT} and I_{OHmaxT} in the fanout calculation. Table 3-4 shows that these specifications are 4.0 mA and -4.0 mA, respectively. Therefore, the fanout of an HC-series output driving HC-series inputs at TTL levels is 4000—for practical purposes, virtually unlimited, apparently.

DC fanout

Well, not quite. The calculations that we've just carried out give the *DC fanout,* defined as the number of inputs that an output can drive *with the output in a constant state* (HIGH or LOW). Even if the DC fanout specification is met, a CMOS output driving a large number of inputs may not behave satisfactorily on transitions, LOW-to-HIGH or vice versa.

During transitions, the CMOS output must charge or discharge the stray capacitance associated with the inputs that it drives. If this capacitance is too large, the transition from LOW to HIGH (or vice versa) may be too slow, causing improper system operation.

AC fanout

The ability of an output to charge and discharge stray capacitance is sometimes called *AC fanout*, though it is seldom calculated as precisely as DC fanout. As you'll see in Section 3.6.1, it's more a matter of deciding how much speed degradation you're willing to live with.

3.5.5 Effects of Loading

Loading an output beyond its rated fanout has several effects:

- In the LOW state, the output voltage (V_{OL}) may increase beyond V_{OLmax}.
- In the HIGH state, the output voltage (V_{OH}) may fall below V_{OHmin}.
- Propagation delay to the output may increase beyond specifications.

- Output rise and fall times may increase beyond their specifications.
- The operating temperature of the device may increase, thereby reducing the reliability of the device and eventually causing device failure.

The first four effects reduce the DC noise margins and timing margins of the circuit. Thus, a slightly overloaded circuit may work properly in ideal conditions, but experience says that it will fail once it's out of the friendly environment of the engineering lab.

3.5.6 Unused Inputs

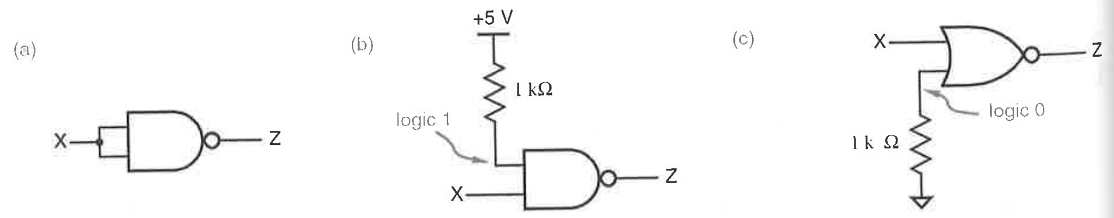
Sometimes not all of the inputs of a logic gate are used. In a real design problem, you may need an n-input gate but have only an $(n+1)$-input gate available. Tying together two inputs of the $(n+1)$-input gate gives it the functionality of an n-input gate. You can convince yourself of this fact intuitively now, or use switching algebra to prove it after you've studied Section 4.1. Figure 3-35(a) shows a NAND gate with its inputs tied together.

You can also tie unused inputs to a constant logic value. An unused AND or NAND input should be tied to logic 1, as in (b), and an unused OR or NOR input should be tied to logic 0, as in (c). In high-speed circuit design, it's usually better to use method (b) or (c) rather than (a), which increases the capacitive load on the driving signal and may slow things down. In (b) and (c), a resistor value in the range 1–10 kΩ is typically used, and a single pull-up or pull-down resistor can serve multiple unused inputs. It is also possible to tie unused inputs directly to the appropriate power-supply rail.

floating input

Unused CMOS inputs should never be left unconnected (or *floating*). On one hand, such an input will behave as if it had a LOW signal applied to it and will normally show a value of 0 V when probed with an oscilloscope or voltmeter. So you might think that an unused OR or NOR input can be left floating, because it will act as if a logic 0 is applied and will not affect the gate's output. However, since CMOS inputs have such high impedance, it takes only a small amount of circuit noise to temporarily make a floating input look HIGH, creating some very nasty intermittent circuit failures.

Figure 3-35 Unused inputs: (a) tied to another input; (b) NAND pulled up; (c) NOR pulled down.



ications.
ereby reducing
ailure.

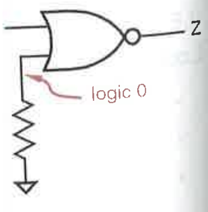
rgins of the cir-
leal conditions,
environment of

design problem,
available. Tying
ctionality of an
ely now, or use
. Figure 3-35(a)

n unused AND or
OR or NOR input
it's usually better
apacitive load on
resistor value in
ull-down resistor
ed inputs directly

(or *floating*). On
applied to it and
illoscope or volt-
an be left floating,
the gate's output.
takes only a small
ok HIGH, creating

IAND pulled up;



SUBTLE BUGS Floating CMOS inputs are often the cause of mysterious circuit behavior, as an unused input erratically changes its effective state based on noise and conditions elsewhere in the circuit. When you're trying to debug such a problem, the extra capacitance of an oscilloscope probe touched to the floating input is often enough to damp out the noise and make the problem go away. This can be especially baffling if you don't realize that the input is floating!

3.5.7 How to Destroy a CMOS Device

Hit it with a sledgehammer. Or simply walk across a carpet and then touch an input pin with your finger. Because CMOS device inputs have such high impedance, they are subject to damage from *electrostatic discharge (ESD)*.

electrostatic discharge (ESD)

ESD occurs when a buildup of charge ("static electricity") on one surface arcs through a dielectric to another surface with the opposite charge. In the case of a CMOS input, the dielectric is the insulation between an input transistor's gate and its source and drain. ESD may damage this insulation, causing a short-circuit between the device's input and its output.

Ordinary activities of people, such as walking on a carpet, can create static electricity with surprisingly high voltage potentials—1000 V or more. The input structures of modern CMOS devices use various measures to reduce their susceptibility to ESD damage, but no device is completely immune. Therefore, to protect CMOS devices from ESD damage during shipment and handling, manufacturers normally package their devices in conductive bags, tubes, or foam. To prevent ESD damage when handling loose CMOS devices, circuit assemblers and technicians usually wear conductive wrist straps that are connected by a coil cord to earth ground; this prevents a static charge from building up on their bodies as they move around the factory or lab.

Ordinary operation of some equipment, such as repeated or continuous movement of mechanical components like doors or fans, can also create static electricity. For that reason, printed-circuit boards containing CMOS circuits are carefully designed with ESD protection in mind. Typically, this means grounding connector housings, the edges of the board, and any other points where static might be encountered because of proximity to people or equipment. This "encourages" ESD to take a safe, metallic path to ground, rather than through the pins of CMOS chips mounted on the board.

Once a CMOS device is installed in a system, another possible source of damage is *latch-up*. The physical input structure of just about any CMOS device contains parasitic bipolar transistors between V_{CC} and ground configured as a "silicon-controlled rectifier (SCR)." In normal operation, this "parasitic SCR" has no effect on device operation. However, an input voltage that is less than ground or more than V_{CC} can "trigger" the SCR, creating a virtual short-circuit between V_{CC} and ground. Once the SCR is triggered, the only way to turn it off

latch-up

ELIMINATE RUDE, SHOCKING BEHAVIOR!

Some design engineers consider themselves above such inconveniences, but to be safe you should follow several ESD precautions in the lab:

- Before handling a CMOS device, touch the grounded metal case of a plugged-in instrument or another source of earth ground.
- Before transporting a CMOS device, insert it in conductive foam.
- When carrying a circuit board containing CMOS devices, handle the board by the edges, and touch a ground terminal on the board to earth ground before poking around with it.
- When handing over a CMOS device to a partner, especially on a dry winter day, touch the partner first. He or she will thank you for it.

is to turn off the power supply. Before you have a chance to do this, enough power may be dissipated to destroy the device (i.e., you may see smoke).

One possible trigger for latch-up is "undershoot" on high-speed HIGH-to-LOW signal transitions, discussed in Section Zo at DDPPonline. In this situation, the input signal may go several volts below ground for several nanoseconds before settling in the normal LOW range. However, modern CMOS logic circuits are fabricated with special structures that prevent latch-up in this transient case.

Latch-up can also occur when CMOS inputs are driven by the outputs of another system or subsystem with a separate power supply. If a HIGH input is applied to a CMOS gate before power is present, the gate may come up in the "latched-up" state when power is applied. Again, modern CMOS logic circuits are fabricated with special structures that prevent this in most cases. However, if the driving output is capable of sourcing lots of current (e.g., tens of milliamperes), latch-up is still possible. One solution to this problem is to apply power before hooking up input cables.

3.6 CMOS Dynamic Electrical Behavior

Both the speed and the power consumption of a CMOS device depend to a large extent on "AC" or dynamic characteristics of the device and its load, that is, what happens when the output changes between states. As part of the internal design of CMOS ASICs, digital designers must carefully examine the effects of output loading, and resize or redesign circuits where the load is too high. Even in board-level design, the effects of loading must be considered for clocks, buses, and other signals that have high fanout or long interconnections.

Speed depends on two characteristics, transition time and propagation delay, discussed in the next two subsections. Power dissipation is discussed in the third subsection, and a few nasty real-world effects are discussed in the last three subsections.

s, but to be

f a plugged-

the board by
round before

y winter day,

do this, enough
smoke).
speed HIGH-to-
In this situation,
ral nanoseconds
OS logic circuits
is transient case.
by the outputs of
a HIGH input is
y come up in the
OS logic circuits
ases. However, if
g., tens of milli-
blem is to apply

e depend to a large
s load, that is, what
the internal design
ne effects of output
igh. Even in board-
clocks, buses, and

ne and propagation
ation is discussed in
discussed in the last

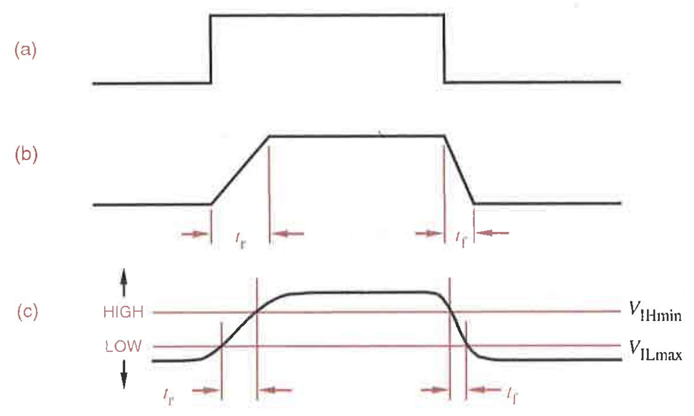


Figure 3-36 Transition times: (a) ideal case of zero-time switching; (b) a more realistic approximation; (c) actual timing, showing rise and fall times.

3.6.1 Transition Time

The amount of time that the output of a logic circuit takes to change from one state to another is called the *transition time*. Figure 3-36(a) shows how we might like outputs to change state—in zero time. However, real outputs cannot change instantaneously, because they need time to charge the stray capacitance of the wires and other components that they drive. A more realistic view of a circuit's output is shown in (b). An output takes a certain time, called the *rise time* (t_r), to change from LOW to HIGH, and a possibly different time, called the *fall time* (t_f), to change from HIGH to LOW.

transition time

rise time (t_r)

fall time (t_f)

Even Figure 3-36(b) is not quite accurate, because the rate of change of the output voltage does not change instantaneously, either. Instead, the beginning and the end of a transition are smooth, as shown in (c). To avoid difficulties in defining the endpoints, rise and fall times are normally measured at the boundaries of the valid logic levels as indicated in the figure.

With the convention in (c), the rise and fall times indicate how long an output voltage takes to pass through the "undefined" region between LOW and HIGH. The initial part of a transition is not included in the rise- or fall-time number. Instead, the initial part of a transition contributes to the "propagation delay" number discussed in the next subsection.

The rise and fall times of a CMOS output depend mainly on two factors, the "on" transistor resistance and the load capacitance. A large capacitance increases transition times; since this is undesirable, it is very rare for a digital designer to purposely connect a capacitor to a logic circuit's output. However, *stray capacitance* is present in every circuit; it comes from at least three sources:

stray capacitance

1. Output circuits, including a gate's output transistors, internal wiring, and packaging, have some capacitance associated with them, in the range of 2–10 picofarads (pF) in typical logic families, including CMOS.

2. The wiring that connects an output to other inputs has capacitance, about 1 pF per inch or more, depending on the wiring technology.
3. Input circuits, including transistors, internal wiring, and packaging, have capacitance, from 2 to 15 pF per input in typical logic families.

capacitive load
AC load

Stray capacitance is sometimes called a *capacitive load* or an *AC load*.

A CMOS output's rise and fall times can be analyzed using the equivalent circuit shown in Figure 3-37. As in the preceding section, the *p*-channel and *n*-channel transistors are modeled by resistances R_p and R_n, respectively. In normal operation, one resistance is high and the other is low, depending on the output's state. The output's load is modeled by an *equivalent load circuit* with three components:

equivalent load circuit

R_L, V_L These two components represent the DC load. They determine the voltages and currents that are present when the output has settled into a stable HIGH or LOW state. The DC load doesn't have too much effect on transition times when the output changes state.

C_L This capacitance represents the AC load. It determines the voltages and currents that are present while the output is changing, and how long it takes to change from one state to the other.

When a CMOS output drives only CMOS inputs, the DC load is negligible. To simplify matters, we'll analyze only this case, with $R_L = \infty$ and $V_L = 0$, in the remainder of this subsection. The presence of a nonnegligible DC load would affect the results, but not dramatically (see Exercise 3.68).

We can now analyze the transition times of a CMOS output. For the purpose of this analysis, we'll assume C_L= 100 pF, a moderate capacitive load. Also, we'll assume that the "on" resistances of the *p*-channel and *n*-channel transistors are 200 Ω and 100 Ω, respectively, as in the preceding subsection. The rise and fall times depend on how long it takes to charge or discharge the capacitive load C_L.

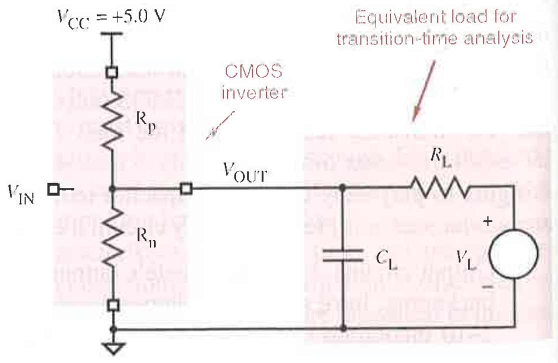


Figure 3-37
Equivalent circuit for analyzing transition times of a CMOS output.

acitance, about
ackaging, have
ilies.
C load.
g the equivalent
p-channel and
respectively. In
epending on the
oad circuit with

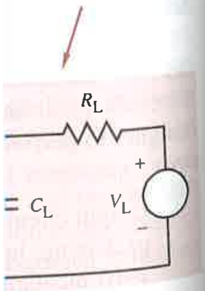
y determine the
has settled into a
o much effect on

the voltages and
, and how long it

l is negligible. To
nd $V_L = 0$, in the
e DC load would

S output. For the
te capacitive load.
nel and *n*-channel
ceding subsection.
e or discharge the

Equivalent load for
nsition-time analysis



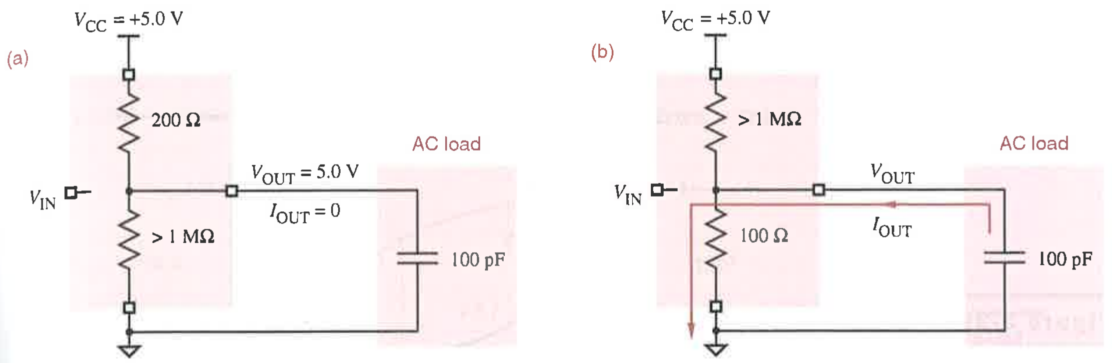


Figure 3-38 Model of a CMOS HIGH-to-LOW transition: (a) in the HIGH state; (b) after *p*-channel transistor turns off and *n*-channel transistor turns on.

First, we'll look at fall time. Figure 3-38(a) shows the electrical conditions in the circuit when the output is in a steady HIGH state. (R_L and V_L are not drawn; they have no effect, since we assume $R_L = \infty$.) For the purposes of our analysis, we'll assume that when CMOS transistors change between "on" and "off," they do so instantaneously. We'll assume that at time $t = 0$ the CMOS output changes to the LOW state, resulting in the situation depicted in (b).

At time $t = 0$, V_{OUT} is still 5.0 V. (A useful electrical engineering maxim is that the voltage across a capacitor cannot change instantaneously.) At time $t = \infty$, the capacitor must be fully discharged and V_{OUT} will be 0 V. In between, the value of V_{OUT} is governed by an exponential law:

$$\begin{aligned} V_{OUT} &= V_{DD} \cdot e^{-t/(R_n C_L)} \\ &= 5.0 \cdot e^{-t/(100 \cdot 100 \cdot 10^{-12})}\ \text{V} \\ &= 5.0 \cdot e^{-t/(10 \cdot 10^{-9})}\ \text{V} \end{aligned}$$

The factor $R_n C_L$ has units of seconds and is called an *RC time constant.* The preceding calculation shows that the *RC* time constant for HIGH-to-LOW transitions is 10 nanoseconds (ns).

RC time constant

Figure 3-39 plots V_{OUT} as a function of time. To calculate fall time, recall that 1.5 V and 3.5 V are the defined boundaries for LOW and HIGH levels for CMOS inputs being driven by the CMOS output. To obtain the fall time, we must solve the preceding equation for $V_{OUT} = 3.5$ and $V_{OUT} = 1.5$, yielding:

$$t = -R_n C_L \cdot \ln \frac{V_{OUT}}{V_{DD}} = -10 \cdot 10^{-9} \cdot \ln \frac{V_{OUT}}{5.0}$$

$$t_{3.5} = 3.57\ \text{ns}$$

$$t_{1.5} = 12.04\ \text{ns}$$

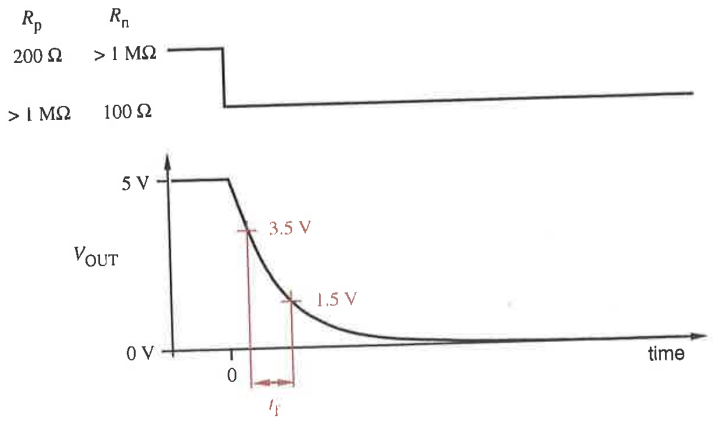


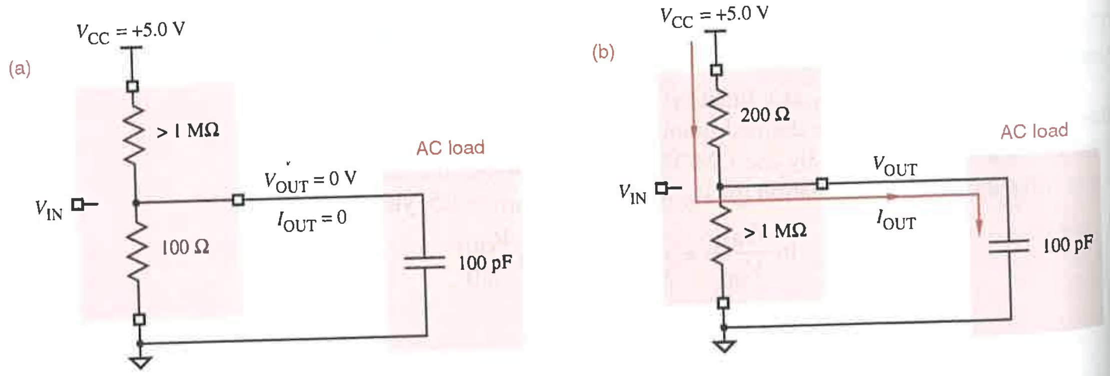
Figure 3-39 Fall time for a HIGH-to-LOW transition of a CMOS output.

The fall time t_f is the difference between these two numbers, or about 8.5 ns.

Rise time can be calculated in a similar manner. Figure 3-40(a) shows the conditions in the circuit when the output is in a steady LOW state. If at time $t = 0$ the CMOS output changes to the HIGH state, the situation depicted in (b) results. Once again, V_{OUT} cannot change instantly, but at time $t = \infty$, the capacitor will be fully charged and V_{OUT} will be 5.0 V. Once again, the value of V_{OUT} in between is governed by an exponential law:

$$\begin{aligned} V_{OUT} &= V_{DD} \cdot (1 - e^{-t/(R_p C_L)}) \\ &= 5.0 \cdot (1 - e^{-t/(200 \cdot 100 \cdot 10^{-12})})\ \text{V} \\ &= 5.0 \cdot (1 - e^{-t/(20 \cdot 10^{-9})})\ \text{V} \end{aligned}$$

Figure 3-40 Model of a CMOS LOW-to-HIGH transition: (a) in the LOW state; (b) after n-channel transistor turns off and p-channel transistor turns on.

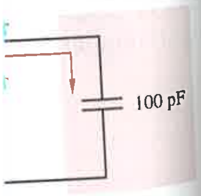


time

about 8.5 ns.
·40(a) shows the
e. If at time $t = 0$
ted in (b) results.
capacitor will be
V_{OUT} in between

in the LOW state;
hannel transistor

AC load



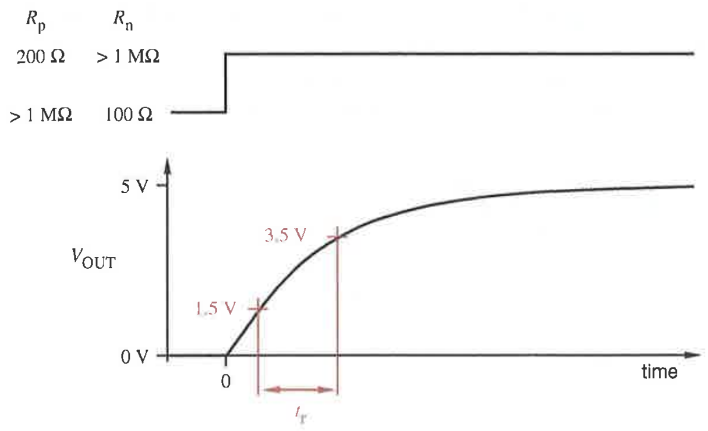


Figure 3-41
Rise time for a LOW-to-HIGH transition of a CMOS output.

The RC time constant in this case is 20 ns. Figure 3-41 plots V_{OUT} as a function of time. To obtain the rise time, we must solve the preceding equation for $V_{OUT} = 1.5$ and $V_{OUT} = 3.5$, yielding

$$\begin{aligned} t &= -RC \cdot \ln \frac{V_{DD} - V_{OUT}}{V_{DD}} \\ &= -20 \cdot 10^{-9} \cdot \ln \frac{5.0 - V_{OUT}}{5.0} \\ t_{1.5} &= 7.13 \text{ ns} \\ t_{3.5} &= 24.08 \text{ ns} \end{aligned}$$

The rise time t_r is the difference between these two numbers, or about 17 ns.

The foregoing example assumes that the p-channel transistor has twice the resistance of the n-channel one, and as a result the rise time is twice as long as the fall time. It takes longer for the "weak" p-channel transistor to pull the output up than it does for the "strong" n-channel transistor to pull it down; the output's drive capability is "asymmetric." High-speed CMOS devices are sometimes fabricated with larger p-channel transistors to make the transition times more nearly equal and output drive more symmetric.

Regardless of the transistors' characteristics, an increase in load capacitance causes an increase in the RC time constant and a corresponding increase in the transition times of the output. Thus, it is a goal of high-speed circuit designers to minimize load capacitance, especially on the most timing-critical signals. This can be done by minimizing the number of inputs driven by the signal, by creating multiple copies of the signal, and by careful physical layout of the circuit.

When working with real digital circuits, it's often useful to estimate transition times, without going through a detailed analysis. A useful rule of thumb is that the transition time approximately equals the *RC* time constant of the charging or discharging circuit. For example, estimates of 10 and 20 ns for fall and rise time in the preceding example would have been pretty much on target, especially considering that most assumptions about load capacitance and transistor "on" resistances are approximate to begin with.

Manufacturers of commercial CMOS circuits typically do not specify transistor "on" resistances on their data sheets. If you search carefully, you might find this information published in the manufacturers' application notes. In any case, you can estimate an "on" resistance as the voltage drop across the "on" transistor divided by the current through it with a worst-case resistive load, as we showed in Section 3.5.2:

$$R_{\mathrm{p(on)}} \approx \frac{V_{\mathrm{DD}} - V_{\mathrm{OHminT}}}{|I_{\mathrm{OHmaxT}}|}$$

$$R_{\mathrm{n(on)}} \approx \frac{V_{\mathrm{OLmaxT}}}{I_{\mathrm{OLmaxT}}}$$

THERE'S A CATCH! Calculated transition times are actually quite sensitive to the choice of logic levels. In the examples in this subsection, if we used 2.0 V and 3.0 V instead of 1.5 V and 3.5 V as the thresholds for LOW and HIGH, we would calculate shorter transition times. On the other hand, if we used 0.0 and 5.0 V, the calculated transition times would be infinity! You should also be aware that in some logic families (most notably TTL), the thresholds are not symmetric around the voltage midpoint. Still, it is the author's experience that the "time-constant-equals-transition-time" rule of thumb usually works for practical circuits.

3.6.2 Propagation Delay

signal path
propagation delay t_{p}

Rise and fall times only partially describe the dynamic behavior of a logic element; we need additional parameters to relate output timing to input timing. A *signal path* is the electrical path from a particular input signal to a particular output signal of a logic element. The *propagation delay* t_{p} of a signal path is the amount of time that it takes for a change in the input signal to produce a change in the output signal.

A complex logic element with multiple inputs and outputs may specify a different value of t_{p} for each different signal path. Also, different values may be specified for a particular signal path, depending on the direction of the output change. Assuming zero rise and fall times for simplicity, Figure 3-42(a) shows

estimate transi-
rule of thumb is
constant of the
nd 20 ns for fall
much on target,
:apacitance and

do not specify
h carefully, you
lication notes. In
p across the "on"
istive load, as we

: of logic levels.
ead of 1.5 V and
horter transition
transition times
c families (most
midpoint. Still, it
on-time" rule of

ehavior of a logic
ng to input timing.
gnal to a particular
a signal path is the
o produce a change

tputs may specify a
erent values may be
ection of the output
igure 3-42(a) shows

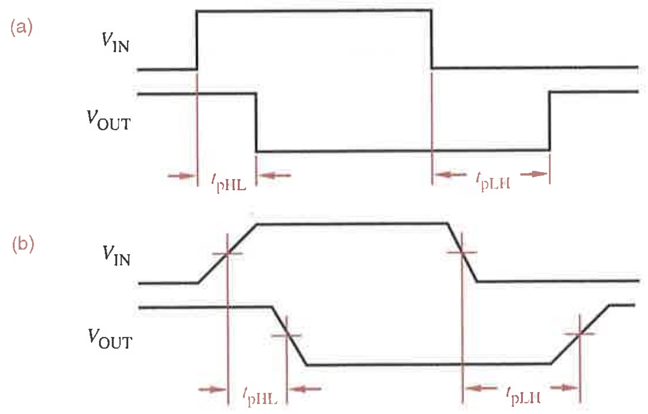


Figure 3-42 Propagation delays for a CMOS inverter: (a) ignoring rise and fall times; (b) measured at midpoints of transitions.

two different propagation delays for the input-to-output signal path of a CMOS inverter, depending on the direction of the output change:

t_{pHL} The time between an input change and the corresponding output change when the output is changing from HIGH to LOW. t_{pHL}

t_{pLH} The time between an input change and the corresponding output change when the output is changing from LOW to HIGH. t_{pLH}

Several factors lead to nonzero propagation delays. In a CMOS device, the rate at which transistors change state is influenced both by the semiconductor physics of the device and by the circuit environment, including input-signal transition rate, input capacitance, and output loading. Multistage devices such as noninverting gates or more complex logic functions may require several internal transistors to change state before the output can change state. And even when the output begins to change state, with nonzero rise and fall times it takes quite some time to cross the region between states, as we showed in the preceding subsection. All of these factors are included in propagation delay.

To factor out the effect of rise and fall times, manufacturers usually specify propagation delays at the midpoints of input and output transitions, as shown in Figure 3-42(b). However, sometimes the delays are specified at the logic-level boundary points, especially if the device's operation may be adversely affected by slow rise and fall times. For example, Figure 3-43 shows how the minimum input pulse width for an S-R latch (discussed in Section 7.2.1) might be specified.

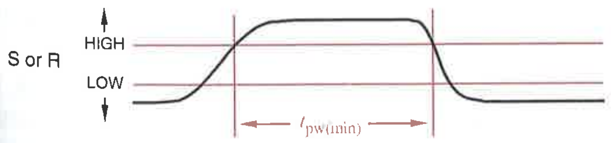


Figure 3-43 Worst-case timing specified using logic-level boundary points.

In addition, a manufacturer may specify absolute maximum input rise and fall times that must be satisfied to guarantee proper operation. High-speed CMOS circuits may consume excessive current or oscillate if their input transitions are too slow.

3.6.3 Power Consumption

static power dissipation
quiescent power dissipation
dynamic power dissipation

The power consumption of a CMOS circuit whose output is not changing is called *static power dissipation* or *quiescent power dissipation.* Most CMOS circuits have very low static power dissipation. This is what makes them so attractive for laptop computers and other low-power applications—when computation pauses, very little power is consumed. A CMOS circuit consumes significant power only during transitions; this is called *dynamic power dissipation.*

One source of dynamic power dissipation is the partial short-circuiting of the CMOS output structure. When the input voltage is not close to one of the power supply rails (0 V or V_{CC}), both the p-channel and n-channel output transistors may be partially "on," creating a series resistance of 600 Ω or less. In this case, current flows through the transistors from V_{CC} to ground. The amount of power consumed in this way depends on both the value of V_{CC} and the rate at which output transitions occur, according to the formula

$$P_T = C_{PD} \cdot V_{CC}^2 \cdot f$$

The following variables are used in the formula:

P_T	The circuit's internal power dissipation due to output transitions.
V_{CC}	The power-supply voltage. As all electrical engineers know, power dissipation across a resistive load (the partially-on transistors) is proportional to the *square* of the voltage.
f	The *transition frequency* of the output signal. This implies the number of power-consuming output transitions per second. (But note that the number of transitions per second is the transition frequency times 2.)
C_{PD}	The *power-dissipation capacitance*. This constant, normally specified by the device manufacturer, completes the formula. C_{PD} turns out to have units of capacitance, but does not represent an actual output capacitance. Rather, it embodies the dynamics of current flow through the changing output-transistor resistances during a single pair of output transitions, HIGH-to-LOW and LOW-to-HIGH. For example, C_{PD} for HC-series CMOS gates is typically 20–24 pF, even though the actual output capacitance is much less.

transition frequency
power-dissipation capacitance

The P_T formula is valid only if input transitions are fast enough, leading to fast output transitions. If the input transitions are too slow, then the output transistors stay partially on for a longer time, and power consumption increases.

CONSUMPTION VS. DISSIPATION The words *consumption* and *dissipation* are used pretty much interchangeably when discussing how much power a device uses. To be precise, however, dissipation includes only the power that is used in the device itself, generating heat in the device. *Consumption* includes additional power that the device consumes from the power supply and delivers to other devices connected to it (such as resistive loads).

Device manufacturers usually recommend a maximum input rise and fall time, below which the value specified for C_{PD} is valid.

A second, and often more significant, source of CMOS power consumption is the capacitive load (C_L) on the output. During a LOW-to-HIGH transition, current flows through a *p*-channel transistor to charge C_L. Likewise, during a HIGH-to-LOW transition, current flows through an *n*-channel transistor to discharge C_L. In each case, power is dissipated in the "on" resistance of the transistor. We'll use P_L to denote the total amount of power dissipated by charging and discharging C_L. C_L P_L

The units of P_L are power, or energy usage per unit time. The energy for one transition could be determined by calculating the current through the charging transistor as a function of time (using the *RC* time constant as in Section 3.6.1), squaring this function, multiplying by the "on" resistance of the charging transistor, and integrating over time. An easier way is described below.

During a transition, the voltage across the C_L changes by $\pm V_{CC}$. According to the definition of capacitance, the total amount of charge that must flow to make a voltage change of V_{CC} across C_L is $C_L \cdot V_{CC}$. The total amount of energy used in one transition is charge times the average voltage change. The first little bit of charge makes a voltage change of V_{CC}, while the last bit of charge makes a vanishingly small voltage change; hence the average change is $V_{CC}/2$. The total energy per transition is therefore $C_L \cdot V_{CC}^2/2$. If there are $2f$ transitions per second, the total power dissipated due to the capacitive load is

$$\begin{aligned} P_L &= C_L \cdot (V_{CC}^2/2) \cdot 2f \\ &= C_L \cdot V_{CC}^2 \cdot f \end{aligned}$$

The total dynamic power dissipation of a CMOS circuit is the sum of P_T and P_L:

$$\begin{aligned} P_D &= P_T + P_L \\ &= C_{PD} \cdot V_{CC}^2 \cdot f + C_L \cdot V_{CC}^2 \cdot f \\ &= (C_{PD} + C_L) \cdot V_{CC}^2 \cdot f \end{aligned}$$

Based on this formula, dynamic power dissipation is often called *CV^2f power*. In most applications of CMOS circuits, CV^2f power is by far the major contributor *CV^2f power* to total power dissipation. Note that CV^2f power is also consumed by bipolar logic circuits like TTL and ECL, but at low to moderate frequencies it is insignificant compared to the static (DC or quiescent) power dissipation of bipolar circuits.

3.6.4 Current Spikes and Decoupling Capacitors

When a CMOS output switches between LOW and HIGH, current flows from V_{CC} to ground through the partially-on *p*- and *n*-channel transistors. These currents, often called *current spikes* because of their brief duration, may show up as noise on the power-supply and ground connections in a CMOS circuit, especially when multiple outputs are switched simultaneously.

current spikes

For this reason, systems that use CMOS circuits require *decoupling capacitors* between V_{CC} and ground. These capacitors must be distributed throughout the printed-circuit board, at least one within an inch or so of each chip, to supply current during transitions. The large *filtering capacitors* typically found in the power supply itself don't satisfy this requirement, because stray wiring inductance prevents them from supplying the current fast enough, hence the need for a *physically distributed* system of decoupling capacitors.

decoupling capacitors

filtering capacitors

3.6.5 Inductive Effects

Digital logic circuits rarely contain any discrete inductors but, just like stray capacitance, *stray inductance* arises in circuit wiring, even in straight wires. (Electrical engineers know that discrete inductors are usually formed by a coil of wire.)

stray inductance

When the amount of current flowing through an inductor changes, a voltage is developed across that inductor according to the formula

$$V = L \cdot \frac{dI}{dt}$$

where L is the inductance in henries and dI/dt is the current's rate of change in amperes per second. Stray inductance can be on the order of 10 nanohenries (nH, 10^{-9} H) per inch of wire on a printed circuit board.

henries

nanohenries

With such tiny stray inductances, it may not seem possible that significant voltages could be developed across them, and that inductive effects could be safely ignored. This was the case with most digital circuits until the late 1990s.

However, two factors have combined to make inductance a significant factor and sometimes an obstacle in high-speed CMOS design, especially at the printed-circuit-board level. First, the output transistors in modern CMOS circuits are able to switch on or off in extremely short times—on the order of tens of picoseconds or less in the fastest circuits. Changing so quickly from a no-current condition to one in which even just a few milliamperes of current is flowing results in a *rate of change* (dI/dt) that is very high. Second, CMOS circuits' power-supply voltage (V_{CC}) has been steadily declining from 5 V to 1.2 V or less in the densest ASICs. This has resulted in smaller noise margins between logic levels, exacerbating the error-inducing effects of any voltage disturbances.

Under reasonable assumptions (see references), the maximum value of dI/dt when driving a resistive load can be approximated by the formula

$$\left(\frac{dI}{dt}\right)_{\text{Max-resistor}} = \frac{\Delta V}{T_t} \cdot \frac{1}{R}$$

where R is the load resistance, and ΔV is the voltage change and T_t is the rise or fall time for the transition. So, let's consider the voltage that could be developed across a 1-inch PCB trace driving a 2-kΩ load, for a couple of different CMOS logic families. A 5-V 74HC output can have transition times as low as 5 ns. Based on the preceding formula,

$$\left(\frac{dI}{dt}\right)_{\text{Max-resistor}} = \frac{5\text{ V}}{5\text{ ns}} \cdot \frac{1}{2000\ \Omega} = 5 \cdot 10^{5}\text{ A/s}$$

Wow, 500,000 amps per second! Of course, the current doesn't continue to ramp up or down for anywhere near a second, but the rate of change during the 5-ns output transition really is that high. Now, we can plug that number into the voltage formula to see how much voltage is developed across our 1-inch, 10-nH PCB trace:

$$V = 10 \cdot 10^{-9} \cdot 5 \cdot 10^{5} = 5\text{ mV}$$

When all's said and done, the voltage change across the PCB trace is only 5 mV (plus or minus, depending on the direction of current change). This is nothing to worry about in a logic family that has 1.35 V of DC noise margin in either state.

Now let's consider the case for a more advanced technology, 74AC, which can source or sink six times as much current as 74HC and do so with transition times as short as 1 ns. The maximum current-change rate for 74AC driving a 1-kΩ load is

$$\left(\frac{dI}{dt}\right)_{\text{Max-resistor}} = \frac{5\text{ V}}{1\text{ ns}} \cdot \frac{1}{1000\ \Omega} = 5 \cdot 10^{6}\text{ A/s}$$

or 10 times higher than the previous case. So the voltage change across a 1-inch, 10-nH PCB trace is also 10 times higher, or 50 mV. That's still not quite enough to worry about, but wait, there's more!

So far we've considered only resistive loads. As discussed in Section 3.6.1, gate inputs and wiring have stray capacitance, and current must flow to charge or discharge this capacitance. Under reasonable assumptions (once again, see references), the maximum value of dI/dt when driving a capacitive load C can be approximated by the formula

$$\left(\frac{dI}{dt}\right)_{\text{Max-capacitor}} = \frac{1.52\Delta V}{T_t^2} \cdot C$$

On a good day, our 74AC output can drive a 50-pF load at the end of a 1-inch PCB trace and deliver a transition time of about 5 ns. Based on the preceding formula,

$$\left(\frac{dI}{dt}\right)_{\text{Max-capacitor}} = \frac{1.52 \cdot 5 \text{ V}}{(25 \cdot 10^{-18}) \text{ ns}^2} \cdot 50 \cdot 10^{-12} \text{ F} = 1.52 \cdot 10^7 \text{ A/s}$$

Plugging that into the voltage formula, the voltage developed across our 1-inch, 10-nH PCB trace is

$$V = 10 \cdot 10^{-9} \cdot 1.52 \cdot 10^7 = 152 \text{ mV}$$

Although this case eats into the noise margin even more than the last example, it's probably not enough to cause an incorrect logic value to be produced. The real problem occurs when the inductive effects of several changing outputs are concentrated on a single wire, as discussed in the next subsection.

HAND WAVING In this subsection, we assumed some transition times for 74HC and 74AC outputs driving certain resistive and capacitive loads. Where did these numbers come from? Minimum transition times, especially as a function of loading, are seldom if ever specified in manufacturers' data sheets. Actually, these numbers came from the author's experience in the lab.

You might also be wondering, what if we had a 10-inch instead of a 1-inch PCB trace and a 500-pF load instead of a 50-pF load? Could there be 15.2 V across that trace during a transition? No, of course not. Experience shows that the transition time would be much longer, and *dI/dt* would be much less. How can that be? The answer is that the actual electrical model of the circuit output, the PCB trace, and the load is much more complicated than we've shown, with each element having resistive, capacitive, and inductive components.

The approximations in this subsection are intended only to give you a rough feel for inductive effects. A more detailed study, typically using a circuit analysis tool such as SPICE, is needed to predict the dynamic effects of output transitions more accurately. Most IC manufacturers provide SPICE models (or equivalent) for their high-speed output circuits to aid electrical engineers who need to analyze such dynamic effects.

3.6.6 Simultaneous Switching and Ground Bounce

The current that flows through a gate's output pin has to come from or go to somewhere—from the device's V_{CC} pin when an output is sourcing current, and to the ground pin when it's sinking current. Now let's consider what happens when multiple gates use the same ground pin.

Figure 3-44 shows the situation when eight inverters are fabricated on a single chip with one ground and one V_{CC} pin. The connection from the chip's

ənd of a 1-inch
ı the preceding

)^7 A/s

ːoss our 1-inch,

ɩe last example,
produced. The
ɟing outputs are
n.

74AC outputs
's come from?
eldom if ever
ame from the

ɪd of a 1-inch
15.2 V across
t the transition
ı that be? The
· trace, and the
t having resis-

e you a rough
ircuit analysis
put transitions
:quivalent) for
ɔ analyze such

:ome from or go
ourcing current,
ɛr what happens

fabricated on a
from the chip's

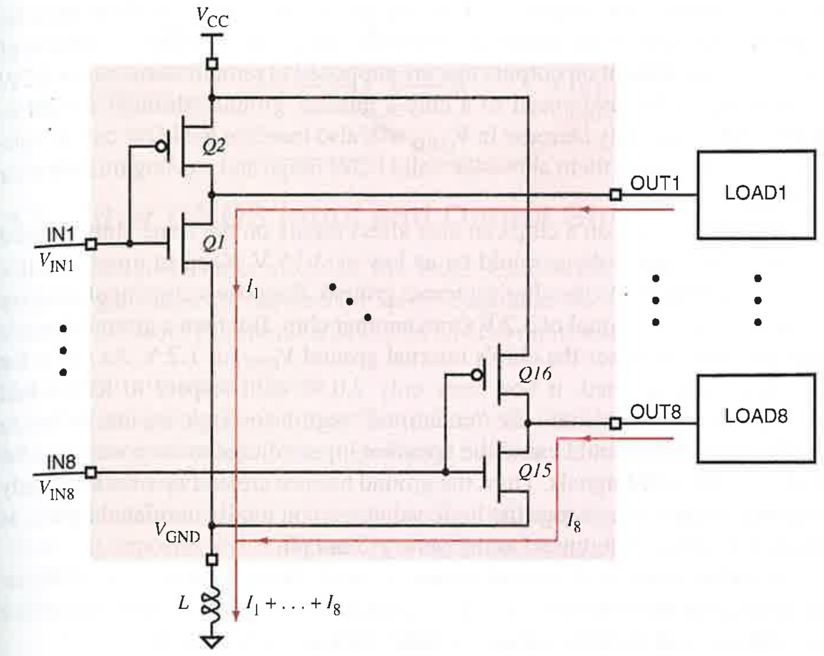


Figure 3-44 Ground bounce in an IC with eight inverters and one ground pin.

internal ground has stray inductance due to the chip and package substrates, the bonding wire between the chip and its package, and the wiring between the package and the PCB's ground plane. In the figure, this is shown as a lumped inductance L between the chip's ground pin and the actual ground on the PCB. The amount of stray inductance varies greatly with different packaging technologies, but in a 20-pin plastic DIP package with the ground pin in the corner, L is on the order of 10 nH.

Now consider the situation when all eight inputs are LOW, so all eight outputs are HIGH, and all eight inputs are simultaneously changed to HIGH. This kind of event is often called *simultaneous switching*. At that moment, all of the outputs change to LOW, and the single ground pin must sink the current from all eight loads. Assuming these are 74AC outputs each driving a 50 pF load as in the previous subsection, maximum value of dI/dt for each output is $1.52 \cdot 10^7$ A/s. With simultaneously switching outputs, the current change across the stray inductance L will be eight times this amount, and the voltage drop across L will be

simultaneous switching

$$V_{GND} = L \cdot \left(8 \cdot \frac{dI}{dt}\right) = 10 \cdot 10^{-9} \cdot 8 \cdot 1.52 \cdot 10^7 \text{ A/s} = 1.216 \text{ V}$$

This change in the chip's internal ground voltage compared to the PCB and system ground is called ground bounce, and its effects can be significant. A chip

ground bounce

has many inputs and outputs, and at any given time some of them may be changing while others are supposed to remain static. But consider the effects of a ground-bounce event on outputs that are supposed to remain static. Since LOW output voltages are referenced to a chip's internal ground (through an ON *n*-channel transistor), any increase in V_{GND} will also increase the LOW output voltages, possibly raising them above the valid LOW range and causing misbehavior elsewhere.

Ground bounce on a chip can also affect *inputs* on the same chip. A valid CMOS HIGH input voltage could be as low as 3.15 V. Keep in mind that this voltage is referenced to the chip's internal ground. Suppose a chip input receives a static, valid HIGH signal of 3.2 V from another chip. But then a ground-bounce event temporarily raises the chip's internal ground V_{GND} to 1.2 V. As far as the chip input is concerned, it now sees only 2.0 V with respect to its internal ground, and this is well into the "undefined" region for logic inputs. In fact, a slightly larger event could cause the apparent input voltage to drop well into the valid range for LOW signals. Thus, the ground bounce created by simultaneously switching outputs can change the logic value seen on totally unrelated inputs, as long as they are all referenced to the same ground pin.

A certain amount of ground bounce is inevitable in high-speed CMOS circuit design, but there are several ways that chip and system designers can reduce it enough to mask it safely within the noise margins of the circuit:

- Create or use a logic family whose output circuits are explicitly designed to have slower transition times, such as 74FCT versus 74AC/ACT.
- Place the ground pins on the IC package so that the lead lengths to the chip will be shorter and hence inductance will be lower. For example, many high-speed circuits packaged in DIPs now have V_{CC} and ground pins in the middle of each row of pins instead of on the corners.
- Use an IC package with lower inductance, such as a square PLCC versus a long rectangular DIP.
- Use multiple ground pins to split the current demand across multiple paths and thereby reduce the voltage drop across any one path. This is one reason that high-pin-count ICs are designed with lots and lots of ground pins.

At this point, you might be wondering, what about "V_{CC} bounce"? After all, V_{CC} wiring paths have stray inductance similar to ground paths, and they suffer voltage drops when multiple outputs switch from LOW to HIGH. However, logic levels are referenced to ground, not V_{CC}, and CMOS inputs are more sensitive to an input's voltage relative to ground than to V_{CC}. Thus, "V_{CC} bounce" is seldom a problem. Still, most high-pin-count ICs are designed with lots of V_{CC} pins to handle dynamic as well as static current demands with little voltage drop. A typical VLSI chip has at least half as many V_{CC} pins as ground pins and, quite often, just as many.

ɔf them may be
ler the effects of
atic. Since LOW
ough an ON n-
OW output volt-
ng misbehavior

ıe chip. A valid
ı mind that this
ɔ input receives
ground-bounce
V. As far as the
to its internal
puts. In fact, a
ɒp well into the
imultaneously
ated inputs, as

ed CMOS cir-
ers can reduce

citly designed
/ACT.

hs to the chip
ample, many
nd pins in the

PLCC versus

ıultiple paths
is one reason
ınd pins.

ınce"? After
ınd they suf-
H. However,
more sensi-
bounce" is
lots of V_{CC}
ɔltage drop.
s and, quite

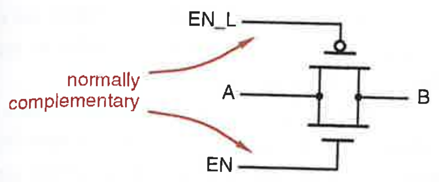


Figure 3-45
CMOS transmission gate.

3.7 Other CMOS Input and Output Structures

Circuit designers have modified the basic CMOS circuit in many ways to produce gates that are tailored for specific applications. This section describes some of the more common variations in CMOS input and output structures.

3.7.1 Transmission Gates

A p-channel and n-channel transistor pair can be connected together to form a logic-controlled switch. Shown in Figure 3-45, this circuit is called a CMOS *transmission gate*.

transmission gate

A transmission gate is operated so that its input signals EN and EN_L are always at opposite levels. When EN is HIGH and EN_L is LOW, there is a low-impedance connection (as low as 2–5 Ω) between points A and B. When EN is LOW and EN_L is HIGH, points A and B are disconnected.

Once a transmission gate is enabled, the propagation delay from A to B (or vice versa) is very short. Because of their short delays and conceptual simplicity, transmission gates are often used internally in larger-scale CMOS devices such as multiplexers and flip-flops. For example, Figure 3-46 shows how transmission gates can be used to create a "2-input multiplexer." When S is LOW, the X "input" is connected to the Z "output"; when S is HIGH, Y is connected to Z.

At least one manufacturer (Integrated Device Technology [IDT]) makes a variety of logic functions based on transmission gates. In their multiplexer devices, it takes several nanoseconds for a change in the "select" inputs (such as

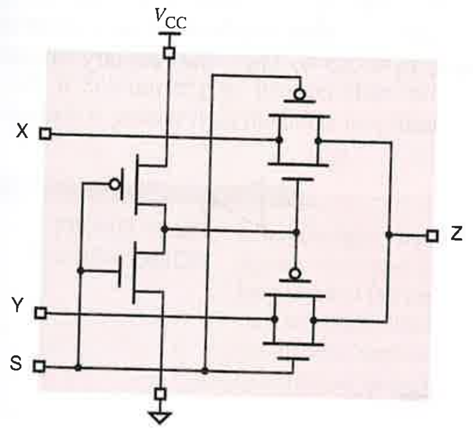


Figure 3-46
Two-input multiplexer using CMOS transmission gates.

in Figure 3-46) to affect the input-output path (X or Y to Z). Once a path is se however, the propagation delay from input to output is specified to be at m 0.25 ns; this is the fastest discrete CMOS multiplexer you can buy.

The *p*-channel (top) transistor in Figure 3-45 has a low impedance when gate (EN_L) is LOW. The *n*-channel transistor has a low impedance when EN HIGH. Two transistors are used because a typical "on" *p*-channel transistor ca conduct a LOW voltage between points A and B very well, and a typical "o *n*-channel transistor can't conduct a HIGH voltage very well, but the paral transistors cover the entire voltage range just fine. Some manufacturers, such IDT, have improved their *n*-channel transistors enough to omit the *p*-chann transistor. Besides saving a transistor, this approach also eliminates a parasi diode to V_{CC} that would otherwise result from the chip's physical structure.

3.7.2 Schmitt-Trigger Inputs

Schmitt-trigger input

The input-output transfer characteristic for a typical CMOS gate was shown i Figure 3-25 on page 101. The corresponding transfer characteristic for a ga with *Schmitt-trigger inputs* is shown in Figure 3-47(a). A Schmitt trigger is special circuit that uses feedback internally to shift the switching threshol depending on whether the input is changing from LOW to HIGH or from HIGH to LOW.

For example, suppose the input of a Schmitt-trigger inverter is initially a 0 V, a solid LOW. Then the output is HIGH, close to 5.0 V. If the input voltage i increased, the output will not go LOW until the input voltage reaches about 2.9 V. However, once the output is LOW, it will not go HIGH again until the input is decreased to about 2.1 V. Thus, the switching threshold for positive-going inpu changes, denoted V_{T+}, is about 2.9 V, and for negative-going input changes, denoted V_{T-}, is about 2.1 V. The difference between the two thresholds is called *hysteresis*. The Schmitt-trigger inverter provides about 0.8 V of hysteresis.

hysteresis

To demonstrate the usefulness of hysteresis, Figure 3-48(a) shows an input signal with long rise and fall times and about 0.5 V of noise on it. An ordinary inverter, without hysteresis, has the same switching threshold for both positive-going and negative-going transitions, $V_T \approx 2.5$ V. Thus, the ordinary inverter

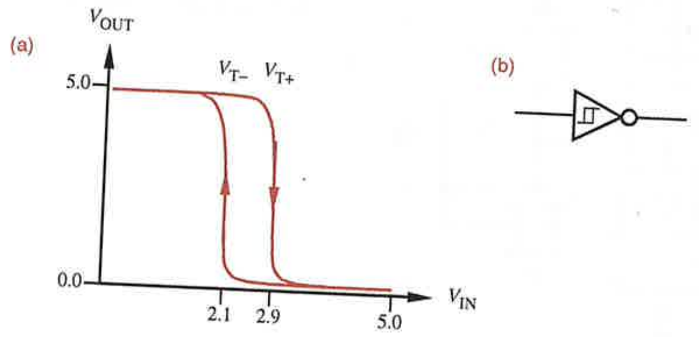


Figure 3-47 A Schmitt-trigger inverter: (a) input-output transfer characteristic; (b) logic symbol.

ath is set up,
be at most

ce when its
when EN is
sistor can't
ypical "on"
the parallel
ers, such as
e p-channel
a parasitic
ucture.

s shown in
for a gate
trigger is a
threshold
from HIGH

initially at
t voltage is
about 2.9
he input is
going input
t changes,
is is called
esis.

s an input
n ordinary
positive-
y inverter

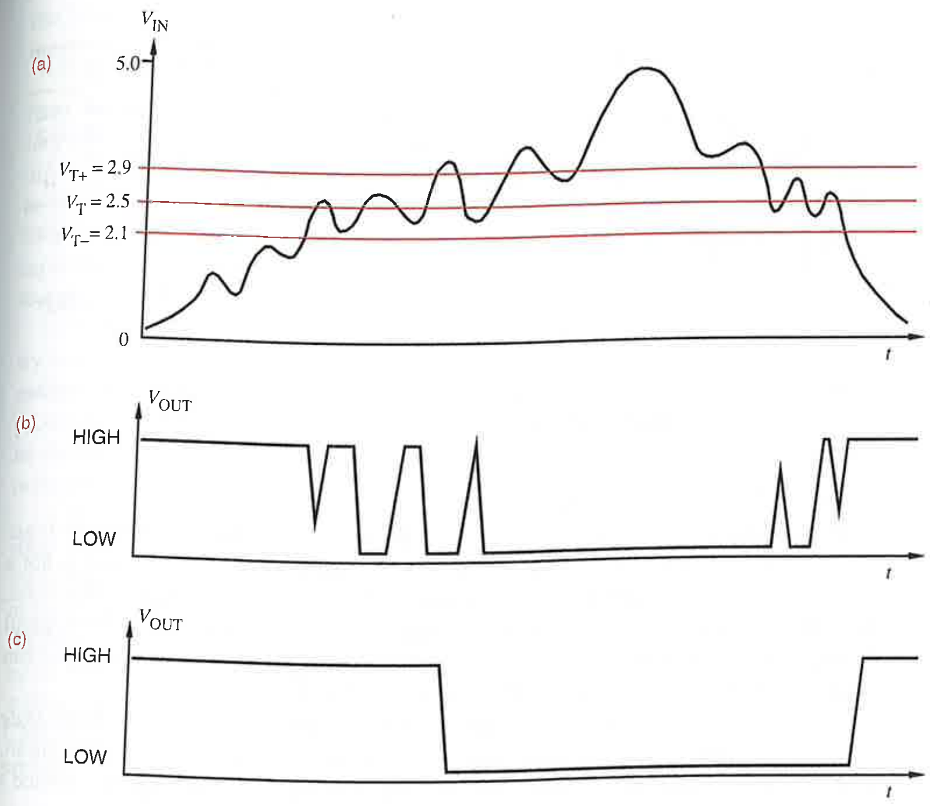


Figure 3-48 Device operation with slowly changing inputs: (a) a noisy, slowly changing input; (b) output produced by an ordinary inverter; (c) output produced by an inverter with 0.8 V of hysteresis.

responds to the noise as shown in (b), producing multiple output changes each time the noisy input voltage crosses the switching threshold. However, as shown in (c), a Schmitt-trigger inverter does not respond to the noise, because its hysteresis is greater than the noise amplitude.

FIXING YOUR TRANSMISSION

Schmitt-trigger inputs have better noise immunity than ordinary gate inputs for signals with transmission-line reflections, discussed at DDPPonline in Section Zo, or long rise and fall times. Such signals typically occur in physically long connections, such as input-output buses and computer interface cables. Noise immunity is important in these applications, since long signal lines are more likely to have reflections or to pick up noise from adjacent signal lines, circuits, and appliances.

e a path is set up,
ed to be at most
ly.
edance when its
nce when EN is
transistor can't
f a typical "on"
but the parallel
cturers, such as
the *p*-channel
tes a parasitic
structure.

was shown in
tic for a gate
tt trigger is a
ing threshold
r from HIGH

is initially at
ut voltage is
es about 2.9
l the input is
going input
ut changes,
lds is called
eresis.
ws an input
n ordinary
h positive-
y inverter

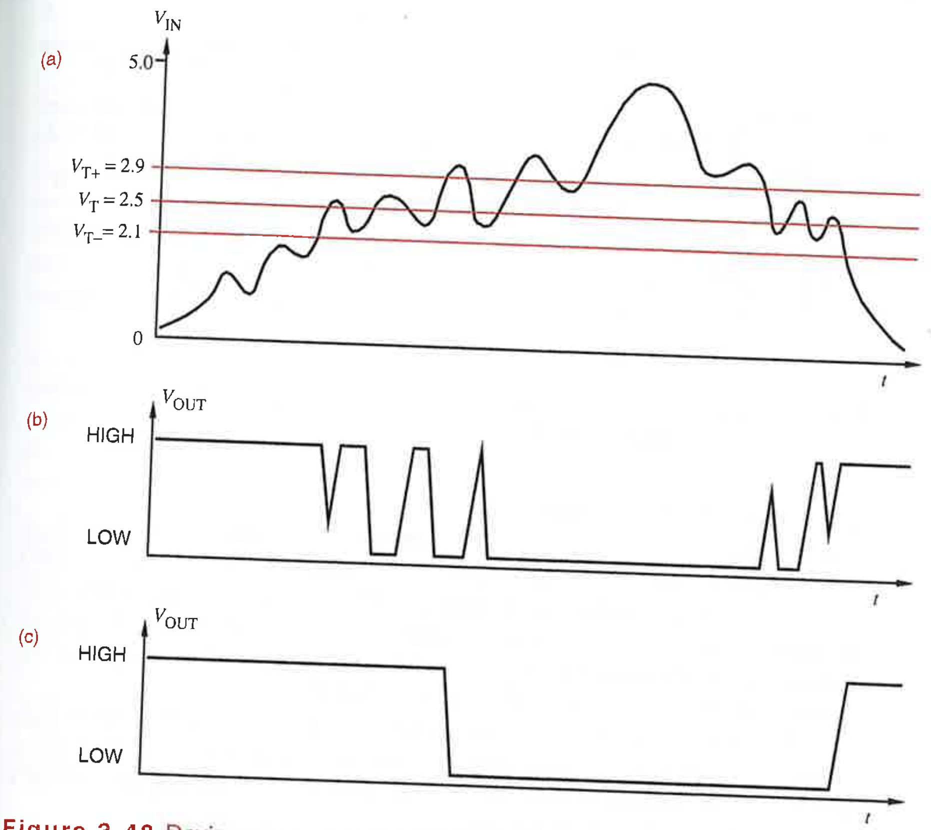


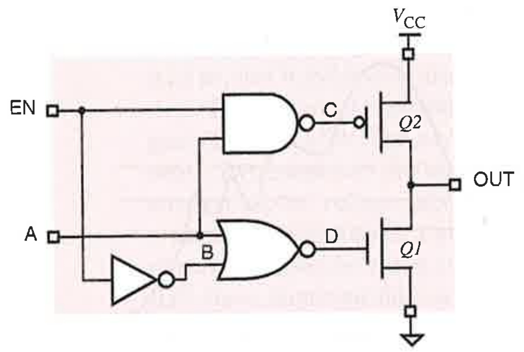
Figure 3-48 Device operation with slowly changing inputs: (a) a noisy, slowly changing input; (b) output produced by an ordinary inverter; (c) output produced by an inverter with 0.8 V of hysteresis.

responds to the noise as shown in (b), producing multiple output changes each time the noisy input voltage crosses the switching threshold. However, as shown in (c), a Schmitt-trigger inverter does not respond to the noise, because its hysteresis is greater than the noise amplitude.

FIXING YOUR TRANSMISSION

Schmitt-trigger inputs have better noise immunity than ordinary gate inputs for signals with transmission-line reflections, discussed at DDPPonline in Section Zo, or long rise and fall times. Such signals typically occur in physically long connections, such as input-output buses and computer interface cables. Noise immunity is important in these applications, since long signal lines are more likely to have reflections or to pick up noise from adjacent signal lines, circuits, and appliances.

(a)



(b)

EN	A	B	C	D	*Q1*	*Q2*	OUT
L	L	H	H	L	off	off	Hi-Z
L	H	H	H	L	off	off	Hi-Z
H	L	L	H	H	on	off	L
H	H	L	L	L	off	on	H

(c)

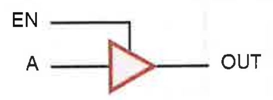


Figure 3-49 CMOS three-state buffer: (a) circuit diagram; (b) function table; (c) logic symbol.

3.7.3 Three-State Outputs

Logic outputs have two normal states, LOW and HIGH, corresponding to logic values 0 and 1. However, some outputs have a third electrical state that is not a logic state at all, called the *high-impedance, Hi-Z,* or *floating state.* In this state, the output behaves as if it isn't even connected to the circuit, except for a small leakage current that may flow into or out of the output pin. Thus, an output can have one of three states—logic 0, logic 1, and Hi-Z.

high-impedance state
Hi-Z state
floating state

An output with three possible states is called (surprise!) a *three-state output* or, sometimes, a *tri-state output*. Three-state devices have an extra input, usually called "output enable" or "output disable," for placing the device's output(s) in the high-impedance state.

three-state output
tri-state output

A *three-state bus* is created by wiring several three-state outputs together. Control circuitry for the "output enables" must ensure that at most one output is enabled (not in its Hi-Z state) at any time. The single enabled device can transmit logic levels (HIGH and LOW) on the bus. Examples of three-state bus design are given in Section 6.6.

three-state bus

A circuit diagram for a CMOS *three-state buffer* is shown in Figure 3-49(a). To simplify the diagram, the internal NAND, NOR, and inverter functions are shown in functional rather than transistor form; they actually use a total of 10 transistors (see Exercise 3.82). As shown in the function table (b), when the enable (EN) input is LOW, both output transistors are off, and the output is in the Hi-Z state. Otherwise, the output is HIGH or LOW as controlled by

three-state buffer

LEGAL NOTICE The name "TRI-STATE" is a trademark of the National Semiconductor Corporation (`www.national.com`). Their lawyer thought you'd like to know.

the "data" input A. Logic symbols for three-state buffers and gates are normally drawn with the enable input coming into the top, as shown in (c).

In practice, the three-state control circuit may be different from what we have shown, in order to provide proper dynamic behavior of the output transistors during transitions to and from the Hi-Z state. In particular, devices with three-state outputs are normally designed so that the output-enable delay (Hi-Z to LOW or HIGH) is somewhat longer than the output-disable delay (LOW or HIGH to Hi-Z). Thus, if a control circuit activates one device's output-enable input and simultaneously deactivates a second's, the second device is guaranteed to enter the Hi-Z state before the first places a HIGH or LOW level on the bus.

If two three-state outputs on the same bus are enabled at the same time and try to maintain opposite states, the situation is similar to tying standard active-pull-up outputs together as in Figure 3-57 on page 139—a nonlogic voltage is produced on the bus. If fighting is only momentary, the devices probably will not be damaged, but the large current drain through the tied outputs can produce noise pulses that affect circuit behavior elsewhere in the system.

There is a leakage current of up to 10 μA associated with a CMOS three-state output in its Hi-Z state. This current, as well as the input currents of receiving gates, must be taken into account when calculating the maximum number of devices that can be placed on a three-state bus. That is, in the LOW or HIGH state, an enabled three-state output must be capable of sinking or sourcing up to 10 μA of leakage current for every other three-state output on the bus, as well as handling the current required by every input on the bus. As with standard CMOS logic, separate LOW-state and HIGH-state calculations must be made to ensure that the fanout requirements of a particular circuit configuration are met.

*3.7.4 Open-Drain Outputs[1]

The p-channel transistors in CMOS output structures are said to provide *active pull-up*, since they actively pull up the output voltage on a LOW-to-HIGH transition. These transistors are omitted in gates with *open-drain outputs*, such as the NAND gate in Figure 3-50(a). The drain of the topmost n-channel transistor is left unconnected internally, so if the output is not LOW it is "open," as indicated in (b). The underscored diamond in the symbol in (c) is sometimes used to indicate an open-drain output. A similar structure, called an "open-collector output," is provided in TTL logic families as described at DDPPonline in Section TTL.

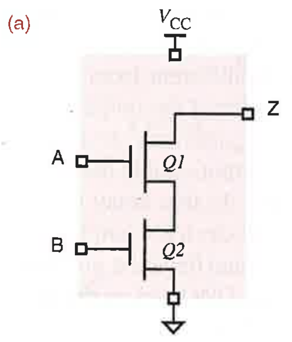
active pull-up
open-drain output

An open-drain output requires an external *pull-up resistor* to provide *passive pull-up* to the HIGH level. For example, Figure 3-51 shows an open-drain CMOS NAND gate, with its pull-up resistor, driving a load.

pull-up resistor
passive pull-up

For the highest possible speed, an open-drain output's pull-up resistor should be as small as possible; this minimizes the RC time constant for LOW-to-

1 *Throughout this book, optional sections are marked with an asterisk.



(b)

A	B	*Q1*	*Q2*	Z
L	L	off	off	open
L	H	off	on	open
H	L	on	off	open
H	H	on	on	L

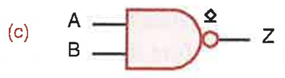


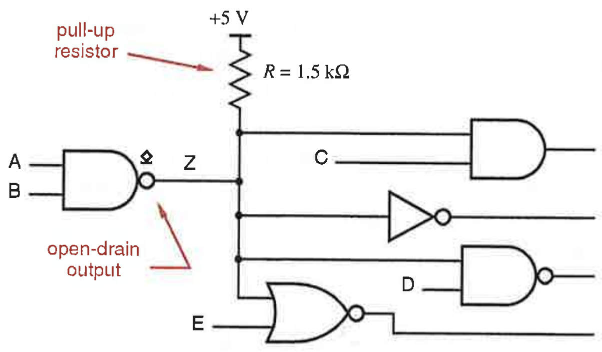
Figure 3-50
Open-drain CMOS NAND gate: (a) circuit diagram; (b) function table; (c) logic symbol.

HIGH transitions (rise time). However, the pull-up resistance cannot be arbitrarily small; the minimum resistance is determined by the open-drain output's maximum sink current, I_{OLmax}. For example, in HC- and HCT-series CMOS, I_{OLmax} is 4 mA, and the pull-up resistor can be no less than 5.0 V/4 mA, or 1.25 kΩ. Since this is an order of magnitude greater than the "on" resistance of the *p*-channel transistors in a standard CMOS gate, the LOW-to-HIGH output transitions are much slower for an open-drain gate than for standard gate with active pull-up.

As an example, let us assume that the open-drain gate in Figure 3-51 is HC-series CMOS, the pull-up resistance is 1.5 kΩ, and the load capacitance is 100 pF. We showed in Section 3.5.2 that the "on" resistance of an HC-series CMOS output in the LOW state is about 80 Ω. Thus, the *RC* time constant for a HIGH-to-LOW transition is about 80 Ω · 100 pF = 8 ns, and the output's fall time is about 8 ns. However, the *RC* time constant for a LOW-to-HIGH transition is about 1.5 kΩ · 100 pF = 150 ns, and the rise time is about 150 ns. This relatively slow rise time is contrasted with the much faster fall time in Figure 3-52. A friend of the author calls such slow rising transitions *ooze*.

ooze

Figure 3-51
Open-drain CMOS NAND gate driving a load.



ce cannot be
he open-drain
nd HCT-series
n 5.0 V/4 mA,
on" resistance
-HIGH output
lard gate with

Figure 3-51 is
capacitance is
an HC-series
constant for a
put's fall time
l transition is
his relatively
gure 3-52. A

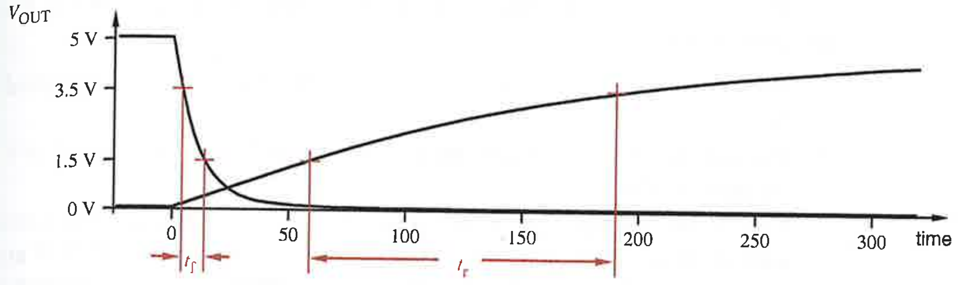


Figure 3-52 Rising and falling transitions of an open-drain CMOS output.

So why use open-drain outputs? Despite slow rise times, they can be useful in three applications discussed next: driving light-emitting diodes (LEDs) and other devices; driving multisource buses; and performing wired logic.

*3.7.5 Driving LEDs

An open-drain output can drive an LED as shown in Figure 3-53. If either input A or B is LOW, the corresponding *n*-channel transistor is off and the LED is off. When A and B are both HIGH, both transistors are on, the output Z is LOW, and the LED is on. The value of the pull-up resistor *R* is chosen so that the proper amount of current flows through the LED in the "on" state.

Typical LEDs require 10 mA for normal brightness. HC- and HCT-series CMOS outputs are only specified to sink or source 4 mA and are not normally used to drive LEDs. However, the outputs in advanced CMOS families such as 74ACT and 74FCT can sink 24 mA or more and can be used quite effectively to drive LEDs.

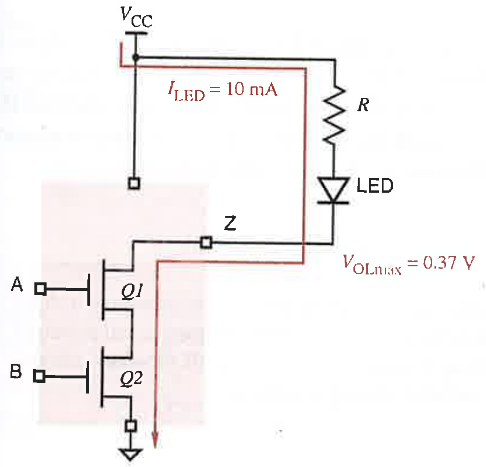


Figure 3-53 Driving an LED with an open-drain output.

Three pieces of information are needed to calculate the proper value of the pull-up resistor R:

1. The LED current I_{LED} needed for the desired brightness, 10 mA for typical LEDs.
2. The voltage drop V_{LED} across the LED in the "on" condition, about 1.6 V for typical LEDs.
3. The output voltage V_{OL} of the open-drain output that sinks the LED current. In the 74AC and 74ACT CMOS families, V_{OLmax} is 0.37 V. If an output can sink I_{LED} and maintain a lower voltage, say 0.2 V, then the calculation below yields a resistor value that is a little too low, but normally with no harm done. A little more current than I_{LED} will flow and the LED will be just a little brighter than expected.

Using the above information, we can write the following equation:

$$V_{OL} + V_{LED} + (I_{LED} \cdot R) = V_{CC}$$

Assuming $V_{CC} = 5.0$ V and the other typical values above, we can solve for the required value of R:

$$\begin{aligned} R &= \frac{V_{CC} - V_{OL} - V_{LED}}{I_{LED}} \\ &= (5.0 - 0.37 - 1.6)\ \text{V}/10\ \text{mA} = 303\ \Omega \end{aligned}$$

Note that you don't have to use an open-drain output to drive an LED. Figure 3-54(a) shows an LED driven by an ordinary CMOS NAND-gate output with active pull-up. If both inputs are HIGH, the bottom (n-channel) transistors pull the output LOW as in the open-drain version. If either input is LOW, the output is HIGH; although one or both of the top (p-channel) transistors is on, no current flows through the LED.

With some CMOS families, you can turn an LED "on" when the output is in the HIGH state, as shown in Figure 3-54(b). This is possible if the output can source enough current to satisfy the LED's requirements. However, method (b) isn't used as often as method (a), because most CMOS and TTL outputs cannot source as much current in the HIGH state as they can sink in the LOW state.

RESISTOR VALUES

In most applications, the precise value of LED series resistors is unimportant, as long as groups of nearby LEDs have similar drivers and resistors to give equal apparent brightness. In the example in this subsection, one might use an off-the-shelf resistor value of 270, 300, or 330 ohms, whatever is readily available.

ılue of the

for typical

ıout 1.6 V

the LED
7 V. If an
then the
normally
the LED

ve for the

an LED.
e output
ansistors
OW, the
is on, no

output is
tput can
thod (b)
s cannot
te.

long
arent
sistor

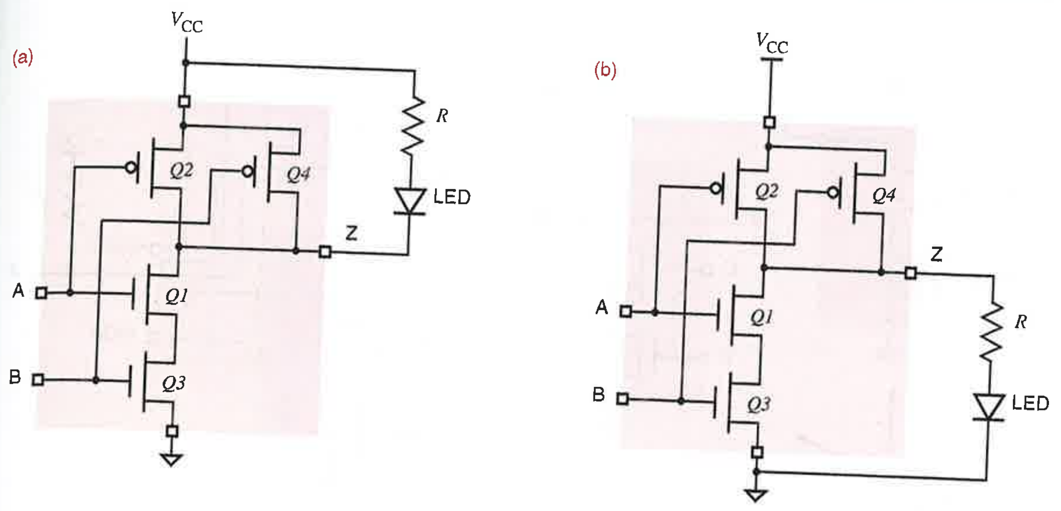


Figure 3-54 Driving an LED with an ordinary CMOS output: (a) sinking current, "on" in the LOW state; (b) sourcing current, "on" in the HIGH state.

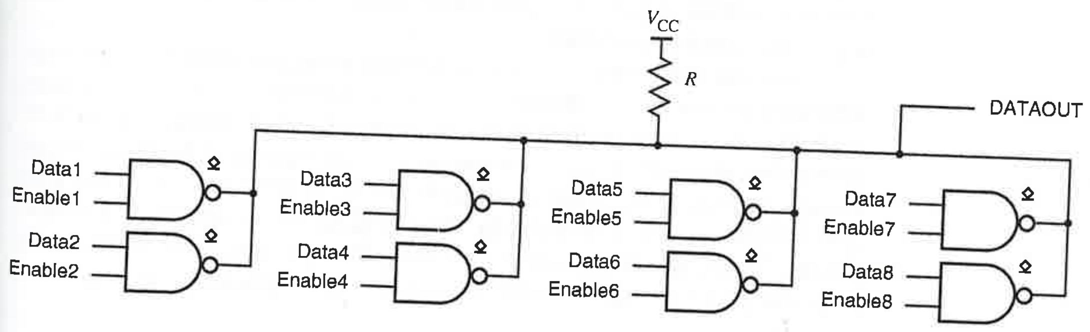
*3.7.6 Multisource Buses

Open-drain outputs can be tied together to allow several devices, one at a time, to put information on a common bus. At any time all but one of the outputs on the bus are in their HIGH (open) state. The remaining output either stays in the HIGH state or pulls the bus LOW, depending on whether it wants to transmit a logical 1 or a logical 0 on the bus. Control circuitry selects the particular device that is allowed to drive the bus at any time.

open-drain bus

For example, in Figure 3-55, eight 2-input open-drain NAND-gate outputs drive a common bus. The top input of each NAND gate is a data bit, and the

Figure 3-55 Eight open-drain outputs driving a bus.



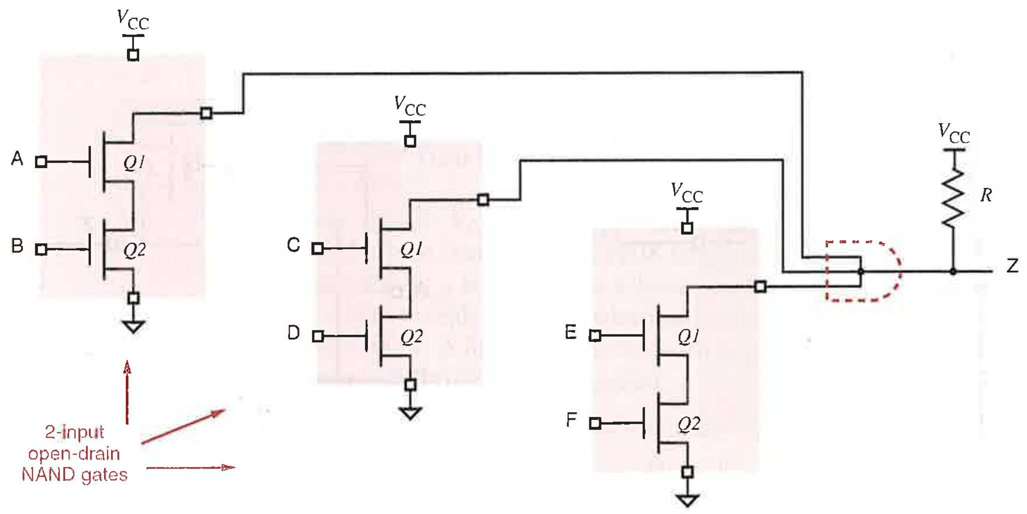


Figure 3-56 Wired-AND function on three open-drain NAND-gate outputs.

bottom input of each is a control bit. At most one control bit is HIGH at any time, enabling the corresponding data bit to be passed through to the bus. (Actually, the complement of the data bit is placed on the bus.) The other gate outputs are HIGH, that is, "open," so the data input of the enabled gate determines the value on the bus.

*3.7.7 Wired Logic

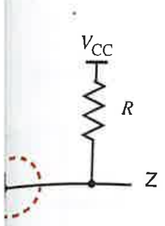
wired logic

If the outputs of several open-drain gates are tied together with a single pull-up resistor, then *wired logic* is performed. (That's *wired*, not *weird*!) An AND function is obtained, since the wired output is HIGH if and only if all of the individual gate outputs are HIGH (actually, open); any output going LOW is sufficient to pull the wired output LOW. For example, a 3-input *wired AND* function is shown in Figure 3-56. If any of the individual 2-input NAND gates has both inputs HIGH, it pulls the wired output LOW; otherwise, the pull-up resistor *R* pulls the wired output HIGH.

wired AND

Note that wired logic cannot be performed using gates with active pull-up. Two such outputs wired together and trying to maintain opposite logic values result in a very high current flow and an abnormal output voltage. Figure 3-57 shows this situation, which is sometimes called *fighting*. The exact output voltage depends on the relative "strengths" of the fighting transistors, but with 5-V CMOS devices it is typically about 1–2 V, almost always a nonlogic voltage. Worse, if outputs fight continuously for more than a few seconds, the chips can get hot enough to sustain internal damage *and* to burn your fingers!

fighting



te outputs.

H at any time,
us. (Actually,
te outputs are
ines the value

single pull-up
rd!) An AND
if all of the
oing LOW is
ed AND func-
ND gates has
ll-up resistor

ctive pull-up.
logic values
Figure 3-57
exact output
ors, but with
a nonlogic
seconds, the
r fingers!

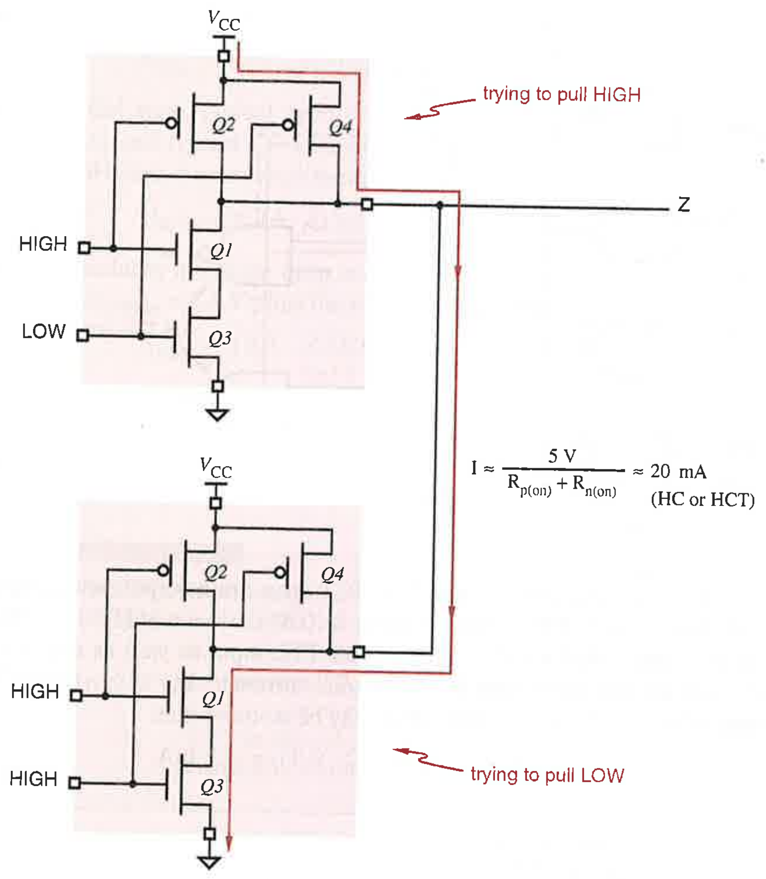


Figure 3-57 Two CMOS outputs trying to maintain opposite logic values on the same line.

*3.7.8 Pull-Up Resistors

A proper choice of value for the pull-up resistor R must be made in open-drain applications. Two calculations are made to bracket the allowable values of R:

pull-up-resistor calculation

Minimum The sum of the current through R in the LOW state and the LOW-state input currents of the gates driven by the wired outputs must not exceed the LOW-state driving capability of the active output, for example, 4 mA for HC/HCT and 8 mA for AHC/AHCT devices.

Maximum The voltage drop across R in the HIGH state must not reduce the output voltage below 2.4 V, which is V_{IHmin} for typical driven gates plus a 400-mV noise margin. This drop is produced by the HIGH-state output leakage current of the wired outputs and the HIGH-state input currents of the driven gates.

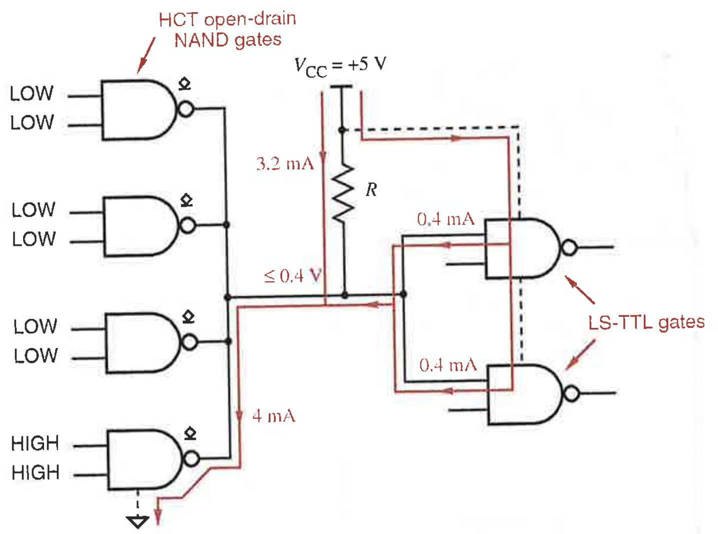


Figure 3-58
Four open-drain outputs driving two inputs in the LOW state.

For example, suppose that four HCT open-drain outputs are wired togetl and drive two LS-TTL inputs (Section 3.10.6) as shown in Figure 3-58. A LC output must sink 0.4 mA from each LS-TTL input as well as sink the curr through the pull-up resistor R. For the total current to stay within the HCT I_{OL} spec of 4 mA, the current through R may be no more than

$$I_{R(max)} = 4 - (2 \cdot 0.4) = 3.2 \text{ mA}$$

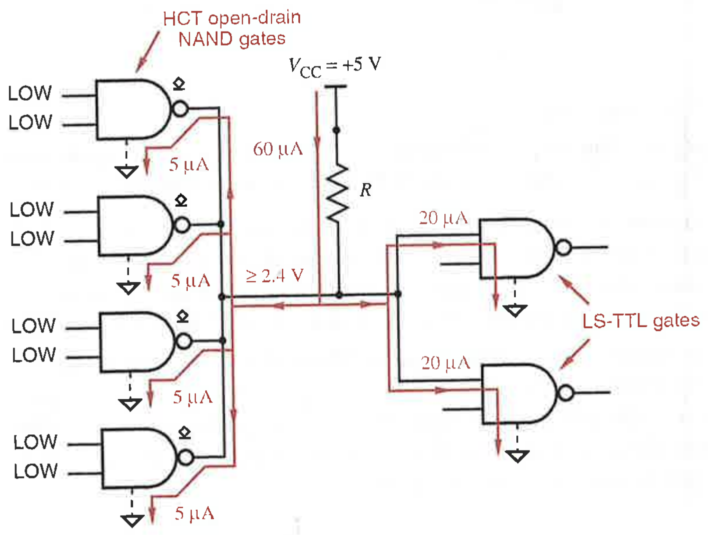


Figure 3-59
Four open-drain outputs driving two inputs in the HIGH state.

Assuming that V_{OL} of the open-drain output is 0.0 V, the minimum value of R is

$$R_{min} = (5.0 - 0.0)/I_{R(max)} = 1562.5\ \Omega$$

In the HIGH state, typical open-drain outputs have a maximum leakage current of 5 μA, and typical LS-TTL inputs require 20 μA of source current. Hence, the HIGH-state current requirement as shown in Figure 3-59 is

$$I_{R(leak)} = (4 \cdot 5) + (2 \cdot 20) = 60\ \mu A$$

This current produces a voltage drop across R, and must not lower the output voltage below $V_{OHmin} = 2.4$ V; thus the maximum value of R is

$$R_{max} = (5.0 - 2.4)/I_{R(leak)} = 43.3\ k\Omega$$

Hence, any value of R between 1562.5 Ω and 43.3 kΩ may be used. Higher values reduce power consumption and improve the LOW-state noise margin, while lower values increase power consumption but improve both the HIGH-state noise margin and the speed of LOW-to-HIGH output transitions.

OPEN-DRAIN ASSUMPTION

In our open-drain resistor calculations, we assume that the output voltage can be as low as 0.0 V rather than 0.4 V (V_{OLmax}) in order to obtain a worst-case result. That is, even if the open-drain output is so strong that it can pull the output voltage all the way down to 0.0 V (it's only required to pull down to 0.4 V), we'll never allow it to sink more than 4 mA, so it doesn't get overstressed. Some designers prefer to use 0.4 V in this calculation, figuring that if the output is so good that it can pull lower than 0.4 V, a little bit of excess sink current beyond 4 mA won't hurt it.

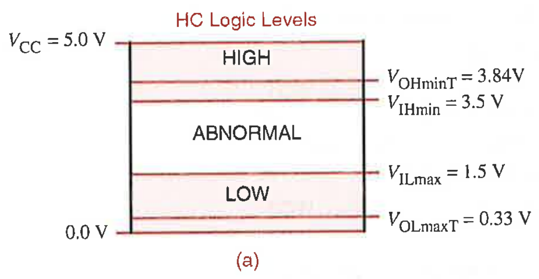
3.8 CMOS Logic Families

The first commercially successful CMOS family was *4000-series CMOS.* Although 4000-series circuits offered the benefit of low power dissipation, they were fairly slow and were not easy to interface with the most popular logic family of the time, bipolar TTL. Thus, the 4000 series was supplanted in most applications by the more capable CMOS families discussed in this section.

4000-series CMOS

All of the CMOS devices that we discuss have part numbers of the form "74FAM*nn*," where "FAM" is an alphabetic family mnemonic and *nn* is a numeric function designator. Devices in different families with the same value of *nn* perform the same function. For example, the 74HC30, 74HCT30, 74AC30, 74ACT30, 74AHC30, and 74AHCT30 are all 8-input NAND gates.

The prefix "74" is simply a number that was used by an early, popular supplier of TTL devices, Texas Instruments. The prefix "54" is used for identical parts that are specified for operation over a wider range of temperature and power-supply voltage, for use in military applications. Such parts are usually



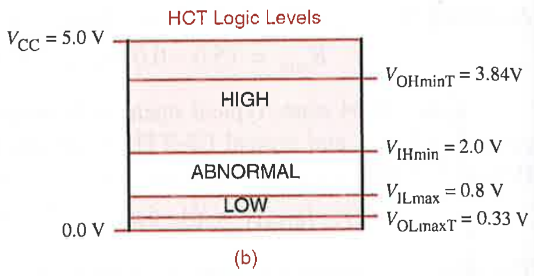


Figure 3-60 Input and output levels for CMOS devices using a 5-V supply: (a) HC; (b) HCT.

fabricated in the same way as their 74-series counterparts, except that they are tested, screened, and marked differently, a lot of extra paperwork is generated, and a higher price is charged, of course.

3.8.1 HC and HCT

HC (High-speed CMOS)

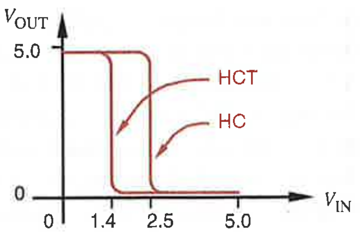
HCT (High-speed CMOS, TTL compatible)

The first two 74-series CMOS families are *HC (High-speed CMOS)* and *HCT (High-speed CMOS, TTL compatible)*. Compared with the original 4000 family, HC and HCT both have higher speed and better current sinking and sourcing capability. The HCT family uses a power-supply voltage V_{CC} of 5 V and can be intermixed with TTL devices, which also use a 5-V supply.

The HC family is optimized for use in systems that use CMOS logic exclusively, and can use any power-supply voltage between 2 and 6 V. A higher voltage is used for higher speed, and a lower voltage for lower power dissipation. Lowering the supply voltage is especially effective, since most CMOS power dissipation is proportional to the square of the voltage (CV^2f power).

Even when used with a 5-V supply, HC devices are not quite compatible with TTL. In particular, HC circuits are designed to recognize CMOS input levels. Assuming a supply voltage of 5.0 V, Figure 3-60(a) shows the input and output levels of HC devices. The output levels produced by TTL devices do not quite match this range, so HCT devices use the different input levels shown in (b). These levels are established in the fabrication process by making transistors with different switching thresholds, producing the different transfer characteristics shown in Figure 3-61.

Figure 3-61 Transfer characteristics of HC and HCT circuits under typical conditions.



We'll say more about CMOS/TTL interfacing in Section 3.10.8. For now, it is useful to note that HC and HCT have essentially identical output specs. Only their input levels differ, with the "T" suffix denoting TTL compatibility.

3.8.2 AHC and AHCT

AHC (Advanced High-speed CMOS)

AHCT (Advanced High-speed CMOS, TTL compatible)

Several new CMOS families were introduced in the 1980s and the 1990s. Two of the most recent and probably the most versatile are *AHC (Advanced High-speed CMOS)* and *AHCT (Advanced High-speed CMOS, TTL compatible)*. These families are two to three times as fast as HC/HCT while maintaining backward compatibility with their predecessors. Like HC and HCT, the AHC and AHCT families differ from each other only in the input levels that they recognize; their output characteristics are the same.

symmetric output drive

asymmetric output drive

Also like HC/HCT, AHC/AHCT outputs have *symmetric output drive*. That is, an output can sink or source equal amounts of current; the output is just as "strong" in both states. Other logic families, including the FCT and TTL families introduced later, have *asymmetric output drive;* they can sink much more current in the LOW state than they can source in the HIGH state.

3.8.3 HC, HCT, AHC, and AHCT Electrical Characteristics

Electrical characteristics of the HC, HCT, AHC, and AHCT families are summarized in this subsection. The specifications assume that the devices are used with a nominal 5-V power supply, although (derated) operation is possible with any supply voltage in the range 2–5.5 V (up to 6 V for HC/HCT). We'll take a closer look at low-voltage and mixed-voltage operation in Section 3.9.

Commercial (74-series) parts are intended to be operated at temperatures between 0°C and 70°C, while military (54-series) parts are characterized for operation between −55°C and 125°C. The specs in Table 3-5 assume an operating temperature of 25°C. A full manufacturer's data sheet provides additional specifications for device operation over the entire temperature range.

Most devices within a given logic family have the same electrical specifications for inputs and outputs, typically differing only in power consumption and propagation delay. Table 3-5 includes specifications for a 74x00 2-input NAND gate and a 74x138 3-to-8 decoder in the HC, HCT, AHC, and AHCT families. The '00 NAND gate is included as the smallest logic-design building block in each family, while the '138 is a "medium-scale" part containing the equivalent of about 15 NAND gates. (The '138 spec is included to allow

VERY = ADVANCED, SORT OF

The AHC and AHCT logic families are manufactured by several companies, including Texas Instruments and Philips. Compatible families with similar but not identical specifications are manufactured by STMicro, Fairchild, and Toshiba; they are called VHC and VHCT, where the "V" stands for "Very."

Table 3-5 Speed and power characteristics of CMOS families operating at 5 V.

				Family			
Description	***Part***	***Symbol***	***Condition***	***HC***	***HCT***	***AHC***	***AHCT***
Typical propagation delay (ns)	'00	t_{PD}		9	10	3.7	5
	'138			18	20	5.7	7.6
Quiescent power-supply current (μA)	'00	I_{CC}	$V_{in} = 0$ or V_{CC}	2.5	2.5	5.0	5.0
	'138		$V_{in} = 0$ or V_{CC}	40	40	40	40
Quiescent power dissipation (mW)	'00		$V_{in} = 0$ or V_{CC}	0.0125	0.0125	0.025	0.025
	'138		$V_{in} = 0$ or V_{CC}	0.2	0.2	0.2	0.2
Power-dissipation capacitance (pF)	'00	C_{PD}		22	15	2.4	2.6
	'138	C_{PD}		55	51	13	14
Dynamic power dissipation (mW/MHz)	'00			0.55	0.38	0.06	0.065
	'138			1.38	1.28	0.33	0.35
Total power dissipation (mW)	'00		f = 100 kHz	0.068	0.050	0.031	0.032
	'00		f = 1 MHz	0.56	0.39	0.085	0.09
	'00		f = 10 MHz	5.5	3.8	0.63	0.68
	'138		f = 100 kHz	0.338	0.328	0.23	0.24
	'138		f = 1 MHz	1.58	1.48	0.53	0.55
	'138		f = 10 MHz	14.0	13.0	3.45	3.7
Speed-power product (pJ)	'00		f = 100 kHz	0.61	0.50	0.11	0.16
	'00		f = 1 MHz	5.1	3.9	0.31	0.45
	'00		f = 10 MHz	50	38	2.3	3.38
	'138		f = 100 kHz	6.08	6.55	1.33	1.79
	'138		f = 1 MHz	28.4	29.5	2.99	4.2
	'138		f = 10 MHz	251	259	19.7	28.1

comparison with the faster FCT family in Section 3.8.5; '00 gates are not manufactured in the FCT family.)

The first row of Table 3-5 specifies propagation delay. As discussed in Section 3.6.2, two numbers, t_{pHL} and t_{pLH}, may be used to specify delay; the number in the table is the worst case of the two. Skipping ahead to Table 3-10 on page 167, you can see that HC and HCT are about the same speed as LS TTL, and that AHC and AHCT about the same as ALS TTL. The propagation delay

NOTE ON NOTATION

The "x" in the notation"74x00" takes the place of a family designator such as HC, HCT, AHC, AHCT, FCT, LS, ALS, AS, or F. We may also refer to such a generic part simply as a " '00" and leave off the "74x."

V.

y

AHC	AHCT
3.7	5
5.7	7.6
5.0	5.0
40	40
0.025	0.025
0.2	0.2
2.4	2.6
13	14
0.06	0.065
0.33	0.35
.031	0.032
.085	0.09
0.63	0.68
0.23	0.24
.53	0.55
.45	3.7
.11	0.16
.31	0.45
2.3	3.38
.33	1.79
.99	4.2
.7	28.1

tes are not

iscussed in
delay; the
ble 3-10 on
s LS TTL,
ation delay

as HC,
generic

QUIETLY GETTING MORE DISS'ED HCT and AHCT circuits can also be driven by TTL devices, which may produce HIGH output levels as low as 2.4 V. As we explained in Section 3.5.3, a CMOS output may draw additional current from the power supply if any of the inputs are nonideal. In the case of an HCT or AHCT inverter with a HIGH input of 2.4 V, the bottom, *n*-channel output transistor is fully "on." However, the top, *p*-channel transistor is also partially "on." This allows the additional quiescent current flow, specified as ΔI_{CC} or I_{CCT} in the data sheet, which can be as much as 2–3 mA per nonideal input in HCT and AHCT devices.

for the '138 is somewhat longer than for the '00, since signals must travel through three or four levels of gates internally.

The second and third rows of the table show that the quiescent power dissipation of these CMOS devices is practically nil, well under a milliwatt (mW) if the inputs have CMOS levels—0 V for LOW and V_{CC} for HIGH. (Note that in the table, the quiescent power dissipation numbers given for the '00 are per gate, while for the '138 they apply to the entire MSI device.)

As we discussed in Section 3.6.3, the dynamic power dissipation of a CMOS gate depends on the voltage swing of the output (usually V_{CC}), the output transition frequency (f), and the capacitance that is being charged and discharged on transitions, according to the formula

$$P_D = (C_L + C_{PD}) \cdot V_{DD}^2 \cdot f$$

Here, C_{PD} is the power-dissipation capacitance of the device and C_L is the capacitance of the load attached to the CMOS output in a given application. The table lists both C_{PD} and an equivalent dynamic power-dissipation factor in units of milliwatts per megahertz, assuming that $C_L = 0$. Using this factor, the total power dissipation is computed at various frequencies as the sum of the dynamic power dissipation at that frequency and the quiescent power dissipation.

speed-power product

Shown next in the table, the *speed-power product* is simply the product of the propagation delay and power consumption of a typical gate; the result is measured in picojoules (pJ). Recall from physics that the joule is a unit of energy, so the speed-power product measures a sort of efficiency—how much energy a logic gate uses to switch its output. In this day and age, it's obvious that the lower the energy usage, the better.

SAVING ENERGY There are practical as well as geopolitical reasons for saving energy in digital systems. Lower energy consumption means lower cost of power supplies and cooling systems. Also, a digital system's reliability is improved more by running it cooler than by any other single reliability improvement strategy.

Table 3-6 Input specifications for CMOS families with V_{CC} between 4.5 and 5.5 V.

			Family			
Description	**Symbol**	**Condition**	**HC**	**HCT**	**AHC**	**AHCT**
Input leakage current (μA)	I_{Imax}	V_{in} = any	±1	±1	±1	±1
Maximum input capacitance (pF)	C_{INmax}		10	10	10	10
LOW-level input voltage (V)	V_{ILmax}		1.35	0.8	1.35	0.8
HIGH-level input voltage (V)	V_{IHmin}		3.85	2.0	3.85	2.0

Table 3-6 gives the input specs of typical CMOS devices in each of the families. Some of the specs assume that the 5-V supply has a ±10% margin; that is, V_{CC} can be anywhere between 4.5 and 5.5 V. These parameters were discussed in previous sections, but for reference purposes their meanings are summarized here:

I_{Imax}	The maximum input current for any value of input voltage. This spec states that the current flowing into or out of a CMOS input is 1 μA or less for any value of input voltage. In other words, CMOS inputs create almost no DC load on the circuits that drive them.
C_{INmax}	The maximum capacitance of an input. This number can be used when figuring the AC load on an output that drives this and other inputs. Most manufacturers also specify a lower, typical input capacitance of 2 to 5 pF, which gives a good estimate of AC load if you're not unlucky.
V_{ILmax}	The maximum voltage that an input is guaranteed to recognize as LOW. Note that the values are different for HC/AHC versus HCT/AHCT. The "CMOS" value, 1.35 V, is 30% of the minimum power-supply voltage, while the "TTL" value is 0.8 V for compatibility with TTL families.

CMOS VS. TTL POWER DISSIPATION

At high transition frequencies (f), CMOS families actually use more power than TTL. For example, compare HCT CMOS in Table 3-5 at f = 10 MHz with LS TTL in Table 3-10; a CMOS gate uses three times as much power as a TTL gate at this frequency. Both HCT and LS may be used in systems with maximum "clock" frequencies of up to about 20 MHz, so you might think that CMOS is not so good for high-speed systems. However, the transition frequencies of most outputs in typical systems are much less than the maximum frequency present in the system (e.g., see Exercise 3.79). Thus, typical CMOS systems have a lower total power dissipation than they would have if they were built with TTL.

Table 3-7 Output specifications for CMOS families operating with V_{CC} between 4.5 and 5.5 V.

			Family			
Description	**Symbol**	**Condition**	**HC**	**HCT**	**AHC**	**AHCT**
LOW-level output current (mA)	I_{OLmaxC}	CMOS load	0.02	0.02	0.05	0.05
	I_{OLmaxT}	TTL load	4.0	4.0	8.0	8.0
LOW-level output voltage (V)	V_{OLmaxC}	$I_{out} \le I_{OLmaxC}$	0.1	0.1	0.1	0.1
	V_{OLmaxT}	$I_{out} \le I_{OLmaxT}$	0.33	0.33	0.44	0.44
HIGH-level output current (mA)	I_{OHmaxC}	CMOS load	−0.02	−0.02	−0.05	−0.05
	I_{OHmaxT}	TTL load	−4.0	−4.0	−8.0	−8.0
HIGH-level output voltage (V)	V_{OHminC}	$\lvert I_{out}\rvert \le \lvert I_{OHmaxC}\rvert$	4.4	4.4	4.4	4.4
	V_{OHminT}	$\lvert I_{out}\rvert \le \lvert I_{OHmaxT}\rvert$	3.84	3.84	3.80	3.80

V_{IHmin} The minimum voltage that an input is guaranteed to recognize as HIGH. The "CMOS" value, 3.85 V, is 70% of the maximum power-supply voltage, while the "TTL" value is 2.0 V for compatibility with TTL families. (Unlike CMOS levels, TTL input levels are not symmetric with respect to the power-supply rails.)

The specifications for TTL-compatible CMOS outputs usually have two sets of output parameters; one set or the other is used depending on how an output is loaded. A *CMOS load* is one that requires the output to sink and source very little DC current, 20 μA for HC/HCT and 50 μA for AHC/AHCT. This is, of course, the case when the CMOS outputs drive only CMOS inputs. With CMOS loads, CMOS outputs maintain an output voltage within 0.1 V of the supply rails, 0 and V_{CC}. (A worst-case $V_{CC} = 4.5$ V is used for the table entries; hence, $V_{OHminC} = 4.4$ V.)

CMOS load

A *TTL load* can consume much more sink and source current, up to 4 mA from an HC/HCT output and 8 mA from an AHC/AHCT output. In this case, a higher voltage drop occurs across the "on" transistors in the output circuit, but the output voltage is still guaranteed to be within the normal range of TTL output levels.

TTL load

Table 3-7 lists CMOS output specifications for both CMOS and TTL loads. These parameters have the following meanings:

I_{OLmaxC} The maximum current that an output can supply in the LOW state while driving a CMOS load. Since this is a positive value, current flows *into* the output pin.

I_{OLmaxT} The maximum current that an output can supply in the LOW state while driving a TTL load.

V_{OLmaxC}	The maximum voltage that a LOW output is guaranteed to produce while driving a CMOS load, that is, as long as I_{OLmaxC} is not exceeded.
V_{OLmaxT}	The maximum voltage that a LOW output is guaranteed to produce while driving a TTL load, that is, as long as I_{OLmaxT} is not exceeded.
I_{OHmaxC}	The maximum current that an output can supply in the HIGH state while driving a CMOS load. Since this is a negative value, positive current flows out of the output pin.
I_{OHmaxT}	The maximum current that an output can supply in the HIGH state while driving a TTL load.
V_{OHminC}	The minimum voltage that a HIGH output is guaranteed to produce while driving a CMOS load, that is, as long as I_{OHmaxC} is not exceeded.
V_{OHminT}	The minimum voltage that a HIGH output is guaranteed to produce while driving a TTL load, that is, as long as I_{OHmaxT} is not exceeded.

The voltage parameters above determine DC noise margins. The LOW-state DC noise margin is the difference between V_{OLmax} and V_{ILmax}. This depends on the characteristics of both the driving output and the driven inputs. For example, the LOW-state DC noise margin of HCT driving a few HCT inputs (a CMOS load) is 0.8 – 0.1 = 0.7 V. With a TTL load, the noise margin for the HCT inputs drops to 0.8 – 0.33 = 0.47 V. Similarly, the HIGH-state DC noise margin is the difference between V_{OHmin} and V_{IHmin}. In general, when different families are interconnected, you have to compare the appropriate V_{OLmax} and V_{OHmin} of the driving gate with V_{ILmax} and V_{IHmin} of all the driven gates to determine the worst-case noise margins.

The I_{OLmax} and I_{OHmax} parameters in the table determine fanout capability and are especially important when an output drives inputs in one or more different families. Two calculations must be performed to determine whether an output is operating within its rated fanout capability:

HIGH-state fanout	The I_{IHmax} values for all of the driven inputs are added. The sum must not exceed I_{OHmax} of the driving output.
LOW-state fanout	The I_{ILmax} values for all of the driven inputs are added. The sum must not exceed I_{OLmax} of the driving output

Note that the input and output characteristics of specific components may vary from the representative values given in Table 3-7, so you must always consult the manufacturers' data sheets when analyzing a real design.

*3.8.4 AC and ACT

AC (Advanced CMOS)
ACT (Advanced CMOS, TTL compatible)

Introduced in the mid-1980s, a pair of more advanced CMOS families are aptly named— *AC (Advanced CMOS)* and *ACT (Advanced CMOS, TTL compatible)*. These families are very fast, and they can source or sink a lot of current, up to

eed to produce
s not exceeded.
ɜed to produce
ɿot exceeded.
he HIGH state
value, positive

he HIGH state

ed to produce
not exceeded.
ɜd to produce
ot exceeded.
ɪs. The LOW-
V_{ILmax}. This
ɟriven inputs.
v HCT inputs
ɿargin for the
ate DC noise
hen different
V_{OLmax} and
ven gates to

ut capability
ɔne or more
ɜ whether an

added. The
ɿt.
added. The
it
ɜs may vary
ays consult

ɜs are aptly
ɔmpatible).
rent, up to

24 mA in either state. Like HC and HCT, and AHC and AHCT, the AC and ACT families differ only in the input levels that they recognize; their output characteristics are the same. Also like the other CMOS families, AC/ACT outputs have symmetric output drive.

Devices in the AC and especially ACT families were popular because of their ability to drive heavy DC loads, including TTL devices. Their outputs also have very fast rise and fall times, which contributes to faster overall system operation, but at a price. The rise and fall times are so fast that they are often a major source of "analog" problems, including switching noise and ground bounce. As a result, the families in the next subsection were developed, and they gradually supplanted the ACT family in most applications requiring TTL compatibility.

*3.8.5 FCT and FCT-T

FCT (Fast CMOS, TTL compatible)

In the early 1990s, yet another CMOS family was launched. The key benefit of the *FCT (Fast CMOS, TTL compatible)* family was its ability to meet or exceed the speed and the output drive capability of the best TTL families while reducing power consumption and maintaining full compatibility with TTL. FCT output circuits are specifically designed with rise and fall times that are more controlled as compared to those of AC/ACT outputs, so FCT outputs do not create quite the same magnitude of "analog" problems.

FCT-T (Fast CMOS, TTL compatible with TTL V_{OH})

Still, the original FCT family had the drawback of producing a full 5-V CMOS V_{OH}, creating enormous CV^2f power dissipation and circuit noise as its outputs swung from 0 V to almost 5 V in high-speed (25 MHz+) applications. A variation of the family, *FCT-T (Fast CMOS, TTL compatible with TTL V_{OH})*, was quickly introduced with circuit innovations to reduce the HIGH-level output voltage, thereby reducing both power consumption and switching noise while maintaining the same high operating speed as the original FCT. A suffix of "T" is used on part numbers to denote the FCT-T output structure, for example, 74FCT138T versus 74FCT138.

The FCT-T family remains very popular today. A key application of FCT-T is driving buses and other heavy loads. To reduce transmission-line reflections (Section Zo at DDPPonline), another high-speed design worry, some FCT-T outputs have built-in 25-Ω series resistors. Compared with other CMOS families, FCT can source or sink gobs of current, up to 64 mA in the LOW state.

*3.8.6 FCT-T Electrical Characteristics

Electrical characteristics of the 5-V FCT-T family are summarized in Table 3-8. The family is specifically designed to be intermixed with TTL devices, so its operation is only specified with a nominal 5-V supply and TTL logic levels.

Individual logic gates are not manufactured in the FCT family. Perhaps the simplest FCT logic element is a 74FCT138T decoder, which has six inputs, eight outputs, and contains the equivalent of about a dozen 4-input gates internally.

Table 3-8 Specifications for a 74FCT138T decoder in the FCT-T logic family.

Description	Symbol	Condition	Value
Maximum propagation delay (ns)	t_{PD}		5.8
Quiescent power-supply current (μA)	I_{CC}	$V_{in} = 0$ or V_{CC}	200
Quiescent power dissipation (mW)		$V_{in} = 0$ or V_{CC}	1.0
Dynamic power-supply current (mA/MHz)	I_{CCD}	Outputs open, one input changing	0.12
Quiescent power-supply current per TTL input (mA)	ΔI_{CC}	$V_{in} = 3.4$ V	2.0
Total power dissipation (mW)		$f = 100$ kHz	0.60
		$f = 1$ MHz	1.06
		$f = 10$ MHz	1.6
Speed-power product (pJ)		$f = 100$ kHz	6.15
		$f = 1$ MHz	9.3
		$f = 10$ MHz	41
Input leakage current (μA)	I_{Imax}	$V_{in} =$ any	±5
Typical input capacitance (pF)	C_{INtyp}		5
LOW-level input voltage (V)	V_{ILmax}		0.8
HIGH-level input voltage (V)	V_{IHmin}		2.0
LOW-level output current (mA)	I_{OLmax}		64
LOW-level output voltage (V)	V_{OLmax}	$I_{out} \le I_{OLmax}$	0.55
HIGH-level output current (mA)	I_{OHmax}		−15
HIGH-level output voltage (V)	V_{OHmin}	$\|I_{out}\| \le \|I_{OHmax}\|$	2.4
	V_{OHtyp}	$\|I_{out}\| \le \|I_{OHmax}\|$	3.3

(This function is described later, in Section 6.4.3.) Comparing its propagation delay and power consumption in Table 3-8 with the corresponding HCT and AHCT numbers in Table 3-5 on page 144, you can see that the FCT-T family is superior in both speed and power dissipation. When comparing, note that FCT-T manufacturers specify only maximum, not typical propagation delays.

Unlike other CMOS families, FCT-T does not have a C_{PD} specification. Instead, it has an I_{CCD} specification:

I_{CCD} Dynamic power-supply current, in units of mA/MHz. This is the amount of additional power-supply current that flows when one input is changing at the rate of 1 MHz.

-T logic family.

	Value
	5.8
	200
	1.0
	0.12
ging	
	2.0
	0.60
	1.06
	1.6
	6.15
	9.3
	41
	±5
	5
	0.8
	2.0
	64
	0.55
	−15
	2.4
	3.3

its propagation
ıding HCT and
=CT-T family is
ıote that FCT-T
elays.
D specification.

is is the amount
input is chang-

The I_{CCD} specification gives the same information as C_{PD}, but in a different way. The circuit's internal power dissipation due to transitions at a given frequency f can be calculated by the formula

$$P_T = V_{CC} \cdot I_{CCD} \cdot f$$

Thus, I_{CCD}/V_{CC} is algebraically equivalent to the C_{PD} specification of other CMOS families (see Exercise 3.85). FCT-T also has a ΔI_{CC} specification for the extra quiescent current that is consumed with nonideal HIGH inputs (see box at the top of page 145).

*3.9 Low-Voltage CMOS Logic and Interfacing

Two important factors have led the IC industry to move toward lower power-supply voltages in CMOS devices:

- In most applications, CMOS output voltages swing from rail to rail, so the V in the CV^2f equation is the power-supply voltage. Cutting power-supply voltage reduces dynamic power dissipation more than proportionally.
- As the industry moves toward ever-smaller transistor geometries, the oxide insulation between a CMOS transistor's gate and its source and drain is getting ever thinner, and thus incapable of insulating voltage potentials as "high" as 5 V.

As a result, JEDEC, an IC industry standards group, selected 3.3 V ± 0.3V, 2.5 V ± 0.2V, 1.8 V ± 0.15V, 1.5 V ± 0.1V, and 1.2 V ± 0.1V as the next "standard" logic power-supply voltages. JEDEC standards specify the input and output logic voltage levels for devices operating with these power-supply voltages.

The migration to lower voltages has occurred in stages and will continue to do so. For discrete logic families, the trend has been to produce parts that operate and produce outputs at the lower voltage but can also tolerate inputs at the higher voltage. This approach has allowed 3.3-V CMOS families to operate with 5-V CMOS and TTL families, as we'll see in the next section.

MORE POWER (SUPPLIES) TO YOU

Many microprocessors and ASICs use a simple approach to accommodate different internal and external logic levels—they have two power-supply voltages. A low voltage, such as 1.8 V, is supplied to operate the chip's internal gates, or *core logic*. A higher voltage, such as 3.3 V, is supplied to operate the external input and output circuits, or *pad ring*, for compatibility with older-generation devices in the system. Special buffer circuits are used internally to translate safely and quickly between the core-logic and the pad-ring logic voltages. With microprocessors, the internal voltage may even be varied dynamically depending on the application's needs—a lower voltage for lower power, and a higher voltage for higher speed.

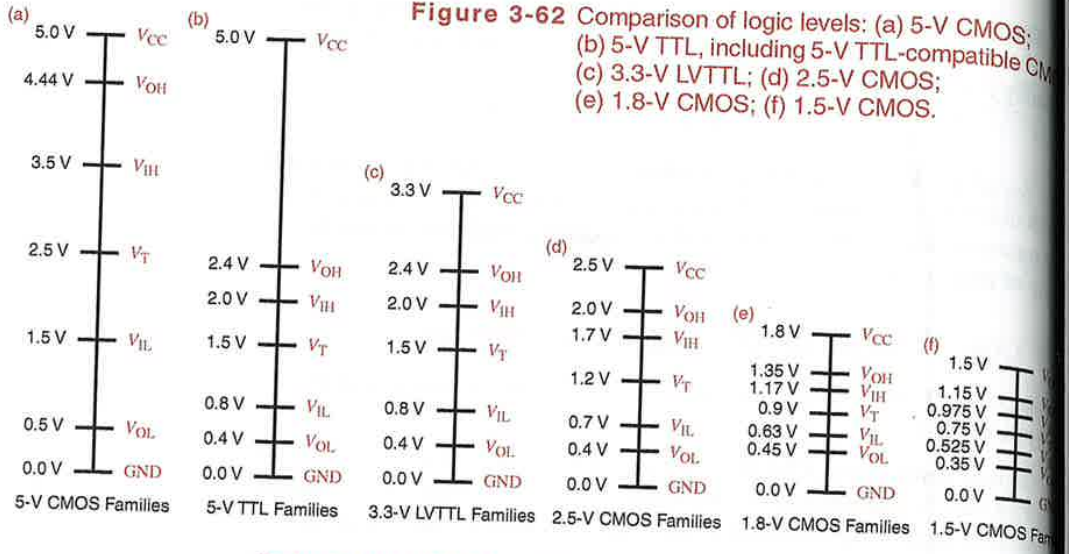


Figure 3-62 Comparison of logic levels: (a) 5-V CMOS; (b) 5-V TTL, including 5-V TTL-compatible C (c) 3.3-V LVTTL; (d) 2.5-V CMOS; (e) 1.8-V CMOS; (f) 1.5-V CMOS.

*3.9.1 3.3-V LVTTL and LVCMOS Logic

The relationships among signal levels for standard TTL and low-voltage C devices operating at their nominal power-supply voltages are illustrated ni in Figure 3-62, adapted from a Texas Instruments application note. The orig symmetric signal levels for pure 5-V CMOS families such as HC and AHC shown in (a). TTL-compatible CMOS families such as HCT, AHCT, and shift the voltage levels downward for compatibility with TTL as shown in (

The first step in the progression of lower CMOS power-supply volt was 3.3 V. The JEDEC standards for 3.3-V logic actually define two set levels. *LVCMOS (low-voltage CMOS)* levels are used in pure CMOS appl tions where outputs have light DC loads (less than 100 μA), so V_{OL} and are maintained within 0.2 V of the power-supply rails. *LVTTL (low-voltage T* levels, shown in (c), are used in applications where outputs have significant loads, so V_{OL} can be as high as 0.4 V and V_{OH} can be as low as 2.4 V.

LVCMOS (low-voltage CMOS)

LVTTL (low-voltage TTL)

The positioning of TTL's logic levels at the low end of the 5-V range really quite fortuitous. As shown in Figure 3-62(b) and (c), it was possibl define the LVTTL levels to match up with TTL levels exactly. Thus, an LV output can drive a TTL input with no problem, as long as its output cur specifications (I_{OLmax}, I_{OHmax}) are respected. Similarly, a TTL output can d an LVTTL input, except for the problem of driving it beyond LVTTL's 3. V_{CC}, as discussed in the next subsection.

Notice the narrowing of the ranges of valid logic levels and the DC n margins in the even lower-voltage standards in (d) through (f). This narrow further drives the importance of minimizing analog effects such as switch noise and ground bounce in modern high-speed designs.

IOS;
ble CMOS;

IOS Families

ge CMOS
ted nicely
e original,
AHC are
and FCT
n in (b).
voltages
o sets of
applica-
and V_{OH}
age TTL)
icant DC

inge was
ssible to
LVTTL
current
an drive
's 3.3-V

C noise
rowing
itching

*3.9.2 5-V Tolerant Inputs

The inputs of a gate won't necessarily tolerate voltages greater than V_{CC}. This is a problem when two different logic voltage ranges are used in a system. For example, 5-V CMOS devices easily produce 4.9-V outputs when lightly loaded, and both CMOS and TTL devices routinely produce 4.0-V outputs even when moderately loaded. The inputs of 3.3-V devices may not like these high voltages.

The maximum voltage V_{Imax} that an input can tolerate is listed in the "absolute maximum ratings" section of the manufacturer's data sheet. For HC devices, V_{Imax} equals V_{CC}. Thus, if an HC device is powered by a 3.3-V supply, its inputs cannot be driven by any 5-V CMOS or TTL outputs without damage. For AHC devices, on the other hand, V_{Imax} is 5.5 V; thus, AHC devices with a 3.3-V power supply may be used to convert 5-V outputs to 3.3-V levels for use with 3.3-V microprocessors, memories, and other devices in a pure 3.3-V subsystem.

Figure 3-63 explains why some inputs are 5-V tolerant and others are not. As shown in (a), the HC and HCT input structure actually contains two reverse-biased *clamp diodes*, which we haven't shown before, between each input signal and V_{CC} and ground. The purpose of these diodes is specifically to shunt any transient input signal voltage less than 0 through *D1* or greater than V_{CC} through *D2* to the corresponding power-supply rail. Such transients can occur as a result of transmission-line reflections, as described in Section Zo at DDPPonline. Shunting the so-called "undershoot" or "overshoot" to ground or V_{CC} reduces the magnitude and duration of reflections.

clamp diode

Of course, diode *D2* can't distinguish between transient overshoot and a persistent input voltage greater than V_{CC}. Hence, if a 5-V output is connected to one of these inputs, it will not see the very high impedance normally associated with a CMOS input. Instead, it will see a relatively low impedance path to V_{CC} through the now forward-biased diode *D2*, and excessive current will flow.

Figure 3-63(b) shows a 5-V tolerant CMOS input. This input structure simply omits *D2*; diode *D1* is still provided to clamp undershoot. The AHC family uses this input structure.

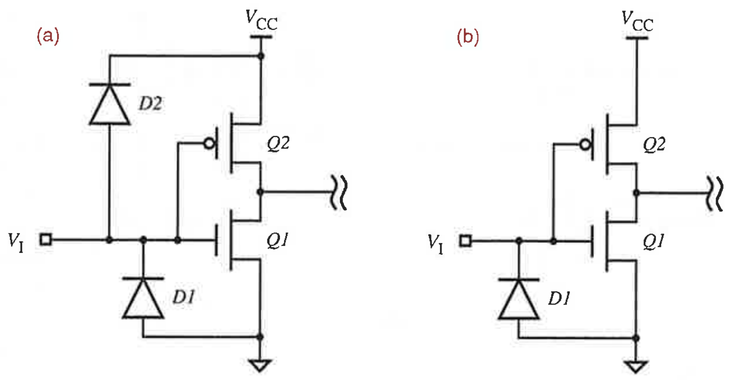


Figure 3-63
CMOS input structures: (a) non-5-V tolerant HC; (b) 5-V tolerant AHC.

The kind of input structure shown in Figure 3-63(b) is necessary but no sufficient to create 5-V tolerant inputs. The transistors in a device's particul fabrication process must also be able to withstand voltage potentials higher tha V_{CC}. On this basis, V_{Imax} in the AHC family is limited to 5.5 V. In many 3.3- ASIC processes, it's not possible to get 5-V tolerant inputs, even if you're wil ing to give up the transmission-line benefits of diode *D2*.

*3.9.3 5-V Tolerant Outputs

Five-volt tolerance must also be considered for outputs, in particular, when bot 3.3-V and 5-V three-state outputs are connected to a bus. When the 3.3-V outp is in the disabled, Hi-Z state, a 5-V device may be driving the bus, and a 5- signal may appear on the 3.3-V device's *output*.

In this situation, Figure 3-64 explains why some outputs are 5-V toleran and others are not. As shown in (a), the standard CMOS three-state output has a *n*-channel transistor *Q1* to ground and a *p*-channel transistor *Q2* to V_{CC}. Whe the output is disabled, circuitry (not shown) holds the gate of *Q1* near 0 V, an the gate of *Q2* near V_{CC}, so both transistors are off and Y is Hi-Z.

Now consider what happens if V_{CC} is 3.3 V and a different device applies 5-V signal to the output pin Y in (a). Then the drain of *Q2* (Y) is at 5 V while th gate (V_2) is still at only 3.3 V. With the gate at a lower potential than the drain *Q2* will begin to conduct and provide a relatively low-impedance path from Y t V_{CC}, and excessive current will flow. Both HC and AHC three-state outputs hav this structure and therefore are not 5-V tolerant.

Figure 3-64(b) shows a 5-V tolerant output structure. An extra *p*-channe transistor *Q3* is used to prevent *Q2* from turning on when it shouldn't. Whe V_{OUT} is greater than V_{CC}, *Q3* turns on. This forms a relatively low impedanc path from Y to the gate of *Q2*, which now stays off because its gate voltage V can no longer be below the drain voltage. This output structure is used in Texa Instruments' LVC (Low-Voltage CMOS) family.

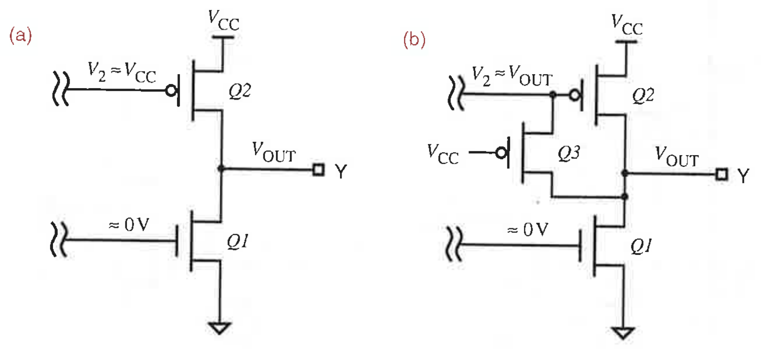


Figure 3-64 CMOS three-state output structures: (a) non-5-V tolerant HC and AHC; (b) 5-V tolerant LVC.

ecessary but not
ice's particular
ials higher than
In many 3.3-V
if you're will-

lar, when both
3.3-V output
us, and a 5-V

5-V tolerant
output has an
V_{CC}. When
near 0 V, and

ice applies a
V while the
an the drain,
th from Y to
outputs have

a *p*-channel
dn't. When
impedance
voltage V_2
ed in Texas

*3.9.4 TTL/LVTTL Interfacing Summary

Based on the information in the preceding subsections, TTL (5-V) and LVTTL (3.3-V) devices can be mixed in the same system subject to just three rules:

1. LVTTL outputs can drive TTL inputs directly, subject to the usual constraints on output current (I_{OLmax}, I_{OHmax}) of the driving devices.
2. TTL outputs can drive LVTTL inputs if the inputs are 5-V tolerant.
3. TTL and LVTTL three-state outputs can drive the same bus if the LVTTL outputs are 5-V tolerant.

*3.9.5 Logic Levels Less Than 3.3 V

The transition from 3.3-V to 2.5-V logic is not so easy. It is true that 3.3-V outputs can drive 2.5-V inputs as long as the inputs are 3.3-V tolerant. However, a quick look at Figure 3-62(c) and (d) on page 152 shows that V_{OH} of a 2.5-V output equals V_{IH} of a 3.3-V input. In other words, there is zero HIGH-state DC noise margin when a 2.5-V output drives a 3.3-V input—not a good situation, but it could be worse.

Comparing the logic levels for 2.5-V and 1.8-V logic, you can see that the minimum HIGH output voltage for 1.8-V logic is quite a bit higher than what can be recognized as HIGH by a 2.5-V input. A smaller mismatch occurs between 1.8-V and 1.5-V logic, but it still cannot be ignored.

level translator
level shifter

The solution to this problem is to use a *level shifter* (or *level translator*), a device which is powered by both supply voltages and which internally boosts the lower logic levels to the higher ones. For example, the 74ALVC164245 level shifter can connect two 16-bit buses with different logic levels on each side. One side could use 5.0-V or 3.3-V power and logic levels, while the other side uses 2.5-V or 1.8-V power and logic levels.

Many of today's ASICs and microprocessors contain level translators internally. This allows them to operate, for example, with a 1.8-V or lower core and a 3.3-V pad ring, as we discussed in the box on page 151.

*3.10 Bipolar Logic

diode logic
transistor-transistor logic (TTL)
emitter-coupled logic (ECL)
BiCMOS logic

Bipolar logic families use semiconductor diodes and bipolar junction transistors as the basic building blocks of logic circuits. The simplest bipolar logic elements use diodes and resistors to perform logic operations; this is called *diode logic*. The original *transistor-transistor logic (TTL)* families used transistors both to perform logic functions and to boost output drive capability. Many newer TTL families use diode logic internally and use transistors only to boost their output drive capability. *Emitter-coupled logic (ECL)* families use transistors as current switches to achieve very high speed. *BiCMOS logic* uses both bipolar and MOS transistors—input and logic circuits are CMOS for low power consumption, while outputs use bipolar transistors to achieve higher driving capability.

As you already know, bipolar logic families have been largely supplanted by the CMOS families that we studied in previous sections. Still, it is useful to study basic TTL operation for the occasional application that requires TTL/CMOS interfacing, discussed in Section 3.10.8. Also, an understanding of TTL may give you insight into the fortuitous similarity of logic levels that allowed the industry to migrate smoothly from TTL to 5-V CMOS logic, and later to lower-voltage, higher-performance 3.3-V CMOS logic.

This section covers the basic operation of bipolar logic circuits at the "black box" level. More details can be found at DDPPonline, as noted as we go along in the sections below.

*3.10.1 Diode Logic

diode
anode
cathode
diode action

The schematic symbol for a *diode* is shown in Figure 3-65(a). The physical properties of a diode are such that positive current can easily flow only in the direction shown by the arrow in the figure,from *anode* to *cathode*; current flow in the other direction is blocked. This is called *diode action*.

reverse-biased diode
forward-biased diode

The transfer characteristic of an ideal diode, shown in Figure 3-65(b), further illustrates this principle. If the anode-to-cathode voltage, *V*, is negative, the diode is said to be *reverse biased* and the current *I* through the diode is zero. If *V* is nonnegative, the diode is said to be *forward biased* and *I* can be an arbitrarily large positive value. In fact, *V* can never get larger than zero, because an ideal diode acts like a zero-resistance short circuit when forward biased.

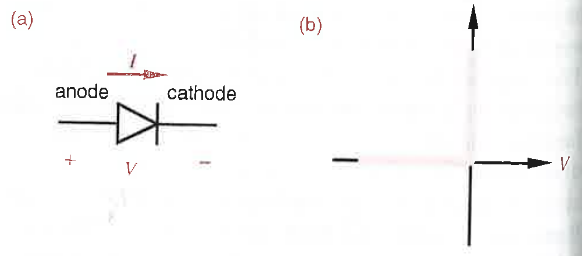
Stated another way, an ideal diode acts like a short circuit as long as the voltage across the anode-to-cathode junction is nonnegative. If the anode-to-cathode voltage is negative, the diode acts like an open circuit and no current flows.

leakage current

forward resistance
diode-drop

Real diodes do not behave as ideally as this, of course. When a real diode is reverse biased, it's not quite an open circuit; a small *leakage current* flows. When the diode is forward biased, it acts like a small resistance, R_f, in series with V_d, a small voltage source. R_f is called the *forward resistance* of the diode, and V_d is called a *diode-drop*, about 0.6 V for typical silicon diodes. For more information, see Section Diode.1 at DDPPonline.

Figure 3-65
Diodes: (a) schematic symbol; (b) transfer characteristic of an ideal diode.



supplanted
is useful to
uires TTL/
ing of TTL
allowed the
er to lower-

cuits at the
ed as we go

he physical
only in the
urrent flow

ure 3-65(b),
is negative,
ode is zero.
can be an
ro, because
biased.
long as the
e anode-to-
no current

eal diode is
rent flows.
series with
diode, and
more infor-

Signal Level	Designation	Binary Logic Value
0–2 volts	LOW	0
2–3 volts	noise margin	undefined
3–5 volts	HIGH	1

Table 3-9 Logic levels in a simple diode logic system.

Diode action can be exploited to perform logical operations. Consider a logic system with a 5-V power supply and the definitions shown in Table 3-9. Within the 5-volt range, signal voltages are partitioned into two ranges, LOW and HIGH, with a 1-volt noise margin between. A voltage in the LOW range is considered to be a logic 0, and a voltage in the HIGH range is a logic 1.

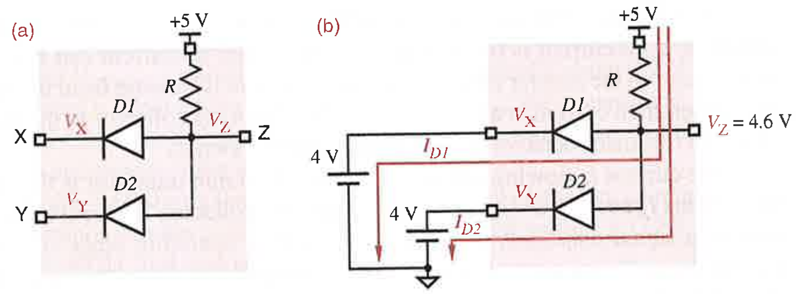
LOW

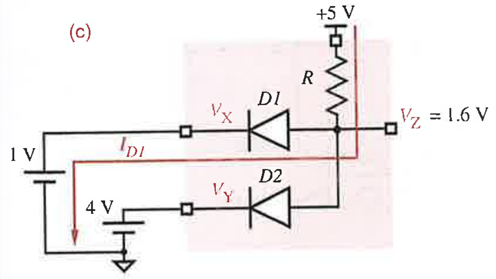
HIGH

diode AND gate

With these definitions, a *diode AND gate* can be constructed as shown in Figure 3-66(a). In this circuit, suppose that both inputs X and Y are connected to HIGH voltage sources, say 4 V, so that V_X and V_Y are both 4 V as in (b). Then both diodes are forward biased, and the output voltage V_Z is one diode-drop above 4 V, or about 4.6 V. A small amount of current, determined by the value of R, flows from the 5-V supply through the two diodes and into the 4-V sources. The colored arrows in the figure show the path of this current flow.

Figure 3-66 Diode AND gate: (a) electrical circuit; (b) both inputs HIGH; (c) one input HIGH, one LOW; (d) function table; (e) truth table.





(d)

V_X	V_Y	V_Z
low	low	low
low	high	low
high	low	low
high	high	high

(e)

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

When diode logic gates are cascaded, the voltage levels of the logic signals move away from the power-supply rails and toward the undefined region, as a result of the voltage drops across the diodes and resistors. For an example, see Section Diode.2 at DDPPonline. Thus, in practice, a diode AND gate normally must be followed by a transistor amplifier to restore the logic levels; this is the scheme used in TTL NAND gates, described in Section 3.10.3. Still, board-level digital designers are occasionally tempted to use discrete diodes to perform logic under special circumstances; for example, see Exercise 3.86.

*3.10.2 Bipolar Junction Transistors

bipolar junction transistor

A *bipolar junction transistor* is a three-terminal device that, in most logic circuits, acts like a current-controlled switch. There are two basic types of bipolar junction transistors, with schematic symbols shown in Figure 3-67. If we put a small current into one of the terminals, called the *base*, then the switch is "on"—current may flow between the other two terminals, called the *emitter* and the *collector*. If no current is put into the base, then the switch is "off"—no current flows between the emitter and the collector.

base
emitter
collector

npn transistor
pnp transistor

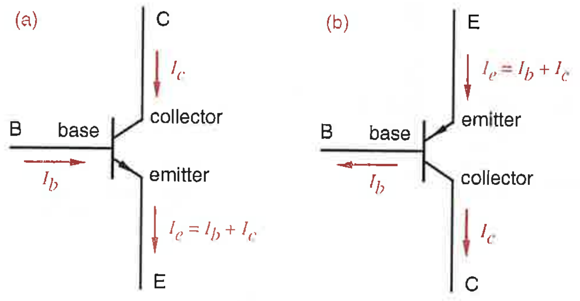
The schematic symbol of an *npn transistor* is shown in (a). The symbol for a *pnp transistor* is shown in (b); however, *pnp* transistors are seldom used in digital circuits, so we won't discuss them any further.

The base-to-emitter junction of an *npn* transistor acts somewhat like a diode—positive current can flow only in the direction of the symbol's subtle arrow. But the base has a more important role as the "control terminal" of the transistor. If no current is flowing into the base, then no current can flow from the collector to the emitter either. However, if current is flowing from the base to the emitter, then current is also enabled to flow from the collector to the emitter. Thus, the transistor behaves as a current-controlled switch.

The current I_e flowing out of the emitter of an *npn* transistor is the sum of the currents I_b and I_c flowing into the base and the collector. A transistor is often used as a signal *amplifier*, because over a certain operating range (the *active region*) the collector current is equal to a fixed constant times the base current ($I_c = \beta \cdot I_b$). However, in digital circuits, we normally use a transistor as a simple switch that's always fully "on" or fully "off," as explained next.

amplifier
active region

Figure 3-67 Bipolar transistor symbols: (a) *npn* transistor; (b) *pnp* transistor.



the logic signals
ned region, as a
an example, see
ↄ gate normally
evels; this is the
Still, board-level
to perform logic

most logic cir-
types of bipolar
-67. If we put a
witch is "on"—
emitter and the
ff"—no current

The symbol for
om used in dig-

omewhat like a
symbol's subtle
erminal" of the
can flow from
rom the base to
r to the emitter.

or is the sum of
ansistor is often
nge (the *active*
he base current
stor as a simple

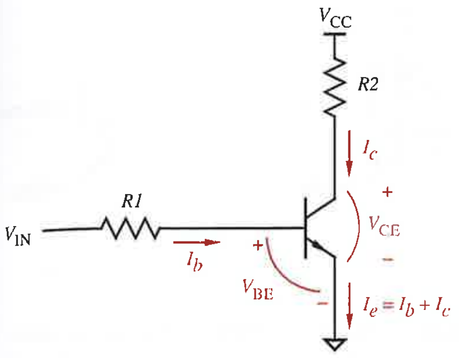


Figure 3-68
Common-emitter configuration of an *npn* transistor.

common-emitter configuration

Figure 3-68 shows the *common-emitter configuration* of an *npn* transistor, which is most often used in digital switching applications. This configuration uses two discrete resistors, *R1* and *R2*, in addition to a single *npn* transistor. In this circuit, if V_{IN} is 0 or negative, then the base-to-emitter diode junction is reverse biased, and no base current (I_b) can flow. If no base current flows, then no collector current (I_c) can flow, and the transistor is said to be *cut off (OFF)*. In this state, there is no current through and hence no voltage drop across *R2*, so $V_{CE} = V_{CC}$.

cut off (OFF)

When base current is flowing, a certain amount of current may flow from the collector to the emitter, directly proportional to the base current. This current creates a voltage drop across *R2* and lowers V_{CE}. However, V_{CE} can never drop lower than $V_{CE(sat)}$, a transistor parameter that is typically about 0.2 V. When the base current is great enough to drop V_{CE} to $V_{CE(sat)}$, the transistor is said to be *saturated (ON)*. In digital logic applications, most transistors are operated so that they are always either saturated or cut off.

saturated (ON)

Figure 3-69 shows that we can make a logic inverter from an *npn* transistor in the common-emitter configuration. When the input voltage is LOW, the output voltage is HIGH, and vice versa.

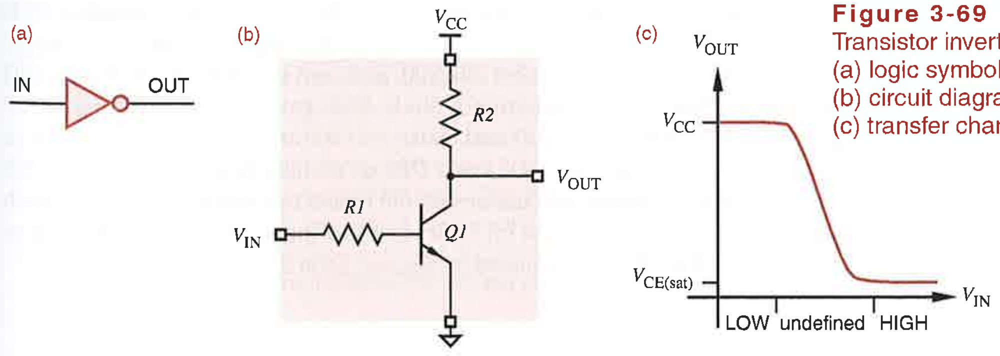


Figure 3-69
Transistor inverter:
(a) logic symbol;
(b) circuit diagram;
(c) transfer characteristic.

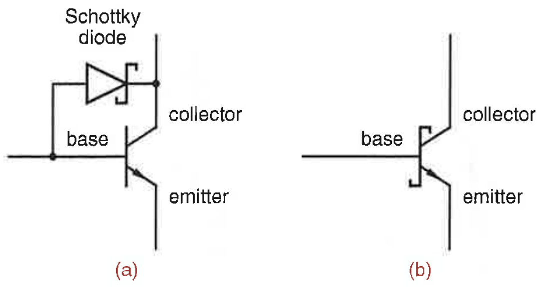


Figure 3-70
Schottky-clamped transistor: (a) circuit; (b) symbol.

storage time

When the input of a saturated transistor is changed, the output does not change immediately; it takes extra time, called *storage time*, to come out of saturation. In fact, storage time accounts for a significant portion of the propagation delay in the original TTL logic family.

Schottky diode
Schottky-clamped transistor
Schottky transistor

Storage time can be eliminated and propagation delay can be reduced by ensuring that transistors do not saturate in normal operation. Contemporary TTL logic families do this by placing a *Schottky diode* between the base and collector of each transistor that might saturate, as shown in Figure 3-70. The resulting transistors, which do not saturate, are called *Schottky-clamped transistors* or *Schottky transistors* for short.

Additional information about bipolar transistors, transistor inverters, and Schottky transistors can be found at DDPPonline in Section BJT.

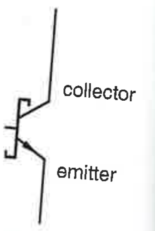
*3.10.3 Transistor-Transistor Logic

There are many different TTL families, with a range of speed, power consumption, and other characteristics. These families use basically the same logic levels as the TTL-compatible CMOS families in previous sections. We'll use the following definitions of LOW and HIGH in our discussions of TTL circuit behavior:

LOW	0–0.8 volts.
HIGH	2.0–5.0 volts.

diode AND gate
clamp diode

The circuit examples in this section are based on a representative TTL family, Low-power Schottky (LS, or LS-TTL). The circuit diagram for a 2-input LS-TTL NAND gate, part number 74LS00, is shown in Figure 3-71 . The NAND function is obtained by combining a diode AND gate with an inverting buffer amplifier. Diodes *D1X* and *D1Y* and resistor *R1* in form a *diode AND gate,* as in Section 3.10.1. *Clamp diodes D2X* and *D2Y* do nothing in normal operation, but limit undesirable negative excursions on the inputs to a single diode-drop. Such negative excursions may occur on HIGH-to-LOW input transitions as a result of transmission-line effects, discussed in Section Zo at DDPPonline.



b)

:tput does not
:ne out of sat-
: propagation

: reduced by
:nporary TTL
:nd collector
he resulting
:ansistors or

:verters, and

consump-
:ogic levels
:se the fol-
behavior:

:ive TTL
:a 2-input
:e NAND
:g buffer
:te, as in
:ion, but
:p. Such
:esult of

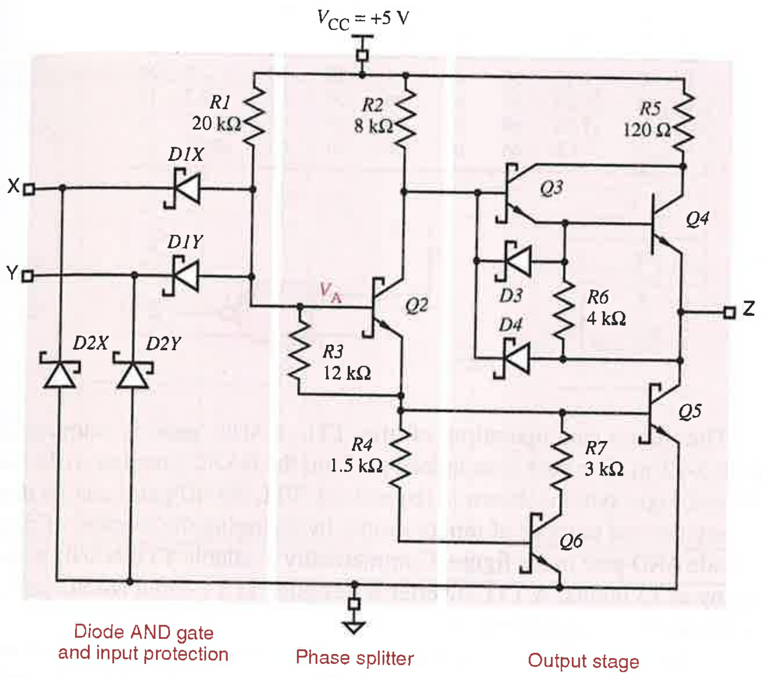


Figure 3-71
Circuit diagram of 2-input LS-TTL NAND gate.

Transistor $Q2$ and the surrounding resistors form a *phase splitter* that controls the output stage. Depending on whether the diode AND gate produces a "low" or a "high" voltage at V_A, $Q2$ is either cut off or turned on.

phase splitter

The *output stage* has two transistors, $Q4$ and $Q5$, only one of which is on at any time. The TTL output stage is sometimes called a *totem-pole* or *push-pull output*. Similar to the *p*-channel and *n*-channel transistors in CMOS, $Q4$ and $Q5$ provide active pull-up and pull-down to the HIGH and LOW states, respectively. Additional details of TTL circuit operation can be found at DDPPonline in Section TTL.

output stage
totem-pole output
push-pull output

WHERE IN THE WORLD IS *Q1*?

Notice that there is no transistor *Q1* in Figure 3-71, but the other transistors are named in a way that's traditional; some TTL devices do in fact have a transistor named *Q1*. Instead of diodes like *D1X* and *D1Y*, these devices use a multiple-emitter transistor *Q1* to perform logic. This transistor has one emitter per logic input, as shown in the figure to the right. Pulling any one of the emitters LOW is sufficient to turn the transistor ON and thus pull V_A LOW.

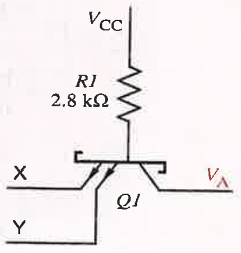


Figure 3-72 Functional operation of a TTL 2-input NAND gate: (a) function table; (b) truth table; (c) logic symbol.

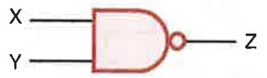
(a)

X	Y	V_A	$Q2$	$Q3$	$Q4$	$Q5$	$Q6$	V_Z	Z
L	L	1.05	off	on	on	off	off	2.7	H
L	H	$\le$1.05	off	on	on	off	off	2.7	H
H	L	$\le$1.05	off	on	on	off	off	2.7	H
H	H	1.2	on	off	off	on	on	$\le$0.35	L

(b)

X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

(c)



The functional operation of the TTL NAND gate is summarized in Figure 3-72(a). The gate does indeed perform the NAND function, with the truth table and logic symbol shown in (b) and (c). TTL NAND gates can be designed with any desired number of inputs simply by changing the number of diodes in the diode AND gate in the figure. Commercially available TTL NAND gates have as many as 13 inputs. A TTL inverter is designed as a 1-input NAND gate, omitting diodes *D1Y* and *D2Y* in Figure 3-71.

So far we have shown the input signals to a TTL gate as ideal voltage sources. Figure 3-73 shows the situation when a TTL input is driven LOW by the output of another TTL gate. Transistor *Q5A* in the driving gate is ON, and thereby provides a path to ground for the current flowing out of the diode *D1XB* in the driven gate. When current flows *into* a TTL output in the LOW state, as in this case, the output is said to be *sinking current*.

sinking current

Figure 3-74 shows the same circuit with a HIGH output. In this case, *Q4A* in the driving gate is turned on enough to supply the small amount of leakage current flowing through reverse-biased diodes *D1XB* and *D2XB* in the driven gate. When current flows out of a TTL output in the HIGH state, the output is said to be *sourcing current*.

sourcing current

*3.10.4 TTL Logic Levels and Noise Margins

At the beginning of this section, we indicated that we would consider TTL signals between 0 and 0.8 V to be LOW, and signals between 2.0 and 5.0 V to be HIGH. Actually, we can be more precise by defining TTL input and output levels in the same way as we did for CMOS:

V_{OHmin}	The minimum output voltage in the HIGH state, 2.7 V for most TTL families.
V_{IHmin}	The minimum input voltage guaranteed to be recognized as a HIGH, 2.0 V for all TTL families.

iarized in
h the truth
designed
diodes in
;ates have
ate, omit-

l voltage
W by the
ON, and
le *D1XB*
ite, as in

se, *Q4A*
leakage
driven
itput is

r TTL
' to be
levels

TTL

IGH,

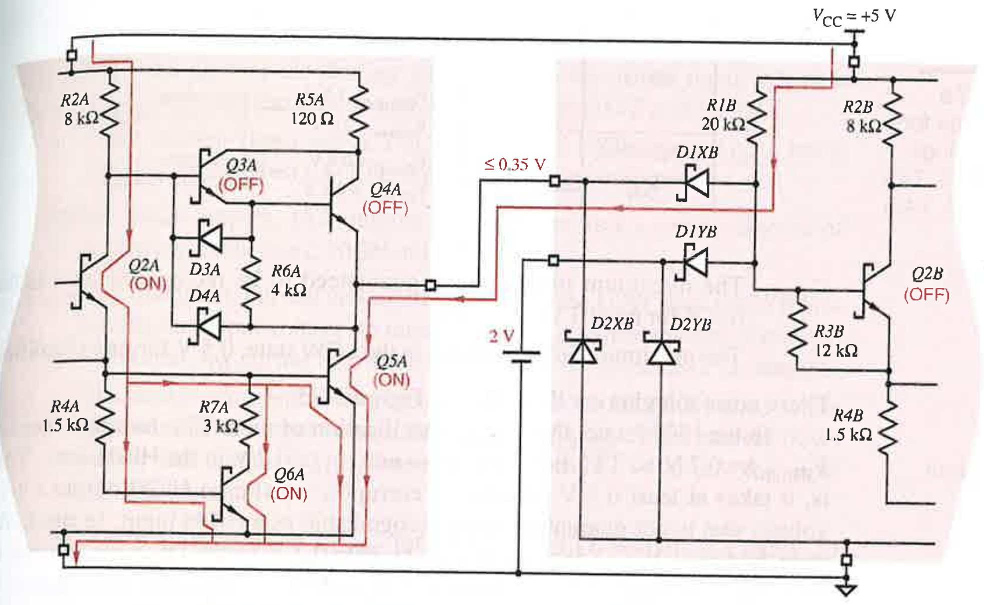


Figure 3-73 A TTL output driving a TTL input LOW.

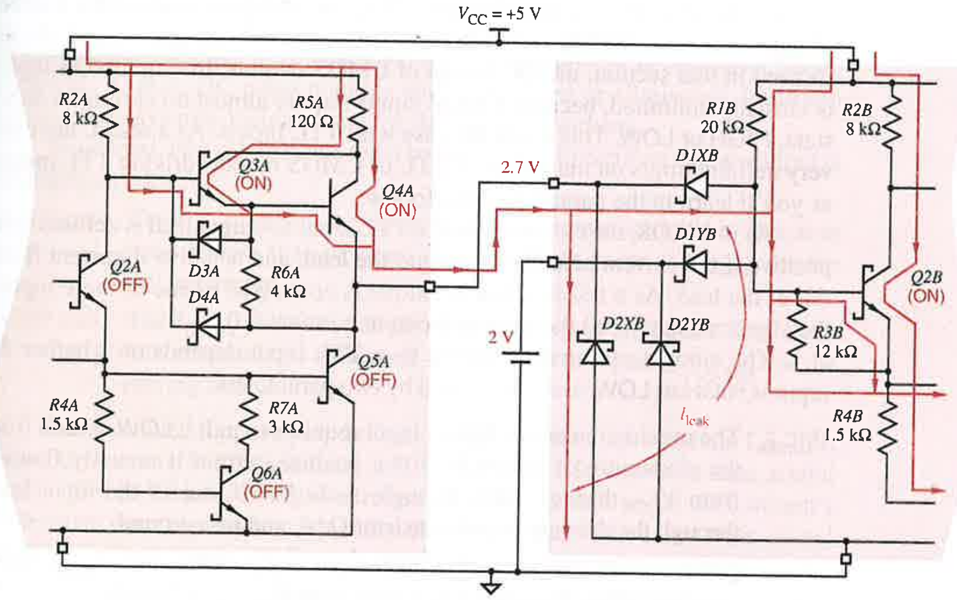


Figure 3-74 A TTL output driving a TTL input HIGH.

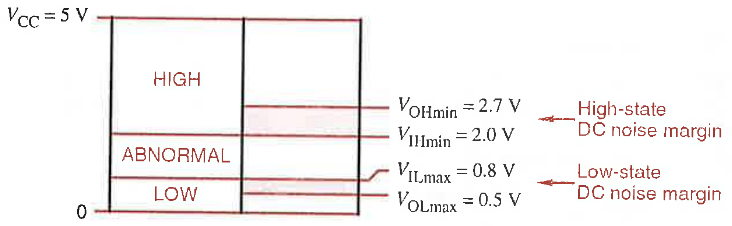


Figure 3-75 Noise margins for popular TTL logic families (74LS, 74S, 74ALS, 74AS, 74F).

V_{ILmax} The maximum input voltage guaranteed to be recognized as a LOW, 0.8 V for most TTL families.

V_{OLmax} The maximum output voltage in the LOW state, 0.5 V for most families.

These noise margins are illustrated in Figure 3-75.

DC noise margin

In the HIGH state, the V_{OHmin} specification of most TTL families exceeds V_{IHmin} by 0.7 V, so TTL has a *DC noise margin* of 0.7 V in the HIGH state. That is, it takes at least 0.7 V of noise to corrupt a worst-case HIGH output into a voltage that is not guaranteed to be recognizable as a HIGH input. In the LOW state, however, V_{ILmax} exceeds V_{OLmax} by only 0.3 V, so the DC noise margin in the LOW state is only 0.3 V. In general, TTL and TTL-compatible circuits tend to be more sensitive to noise in the LOW state than in the HIGH state.

*3.10.5 TTL Fanout

fanout

As we defined it previously in Section 3.5.4, *fanout* is a measure of the number of gate inputs that are connected to (and driven by) a single gate output. As we showed in that section, the DC fanout of CMOS outputs driving CMOS inputs is virtually unlimited, because CMOS inputs require almost no current in either state, HIGH or LOW. This is not the case with TTL inputs. As a result, there are very definite limits on the fanout of TTL or CMOS outputs driving TTL inputs, as you'll learn in the paragraphs that follow.

current flow

As in CMOS, the *current flow* in a TTL input or output lead is defined to be positive if the current actually flows *into* the lead, and negative if current flows *out* of the lead. As a result, when an output is connected to one or more inputs, the algebraic sum of all the input and output currents is 0.

The amount of current required by a TTL input depends on whether the input is HIGH or LOW, and is specified by two parameters:

I_{ILmax} The maximum current that an input requires to pull it LOW. Recall from the discussion of Figure 3-73 that positive current is actually flowing from V_{CC}, through *R1B*, through diode *D1XB*, out of the input lead, through the driving output transistor *Q5A*, and into ground.

Since current flows out of a TTL input in the LOW state, I_{ILmax} has a negative value. Most LS-TTL inputs have I_{ILmax} = –0.4 mA.

gin

gin

d as a LOW,

nost families.

ilies exceeds
H state. That
output into a
In the LOW
ise margin in
rcuits tend to

the number
itput. As we
MOS inputs
ent in either
ılt, there are
TTL inputs,

lefined to be
ırrent flows
nore inputs,

whether the

Recall from
lly flowing
input lead,

ILmax has a

I_{IHmax} The maximum current that an input requires to pull it HIGH. As shown in Figure 3-74 on page 163, positive current flows from V_{CC}, through *R5A* and *Q4A* of the driving gate, and *into* the driven input, where it leaks to ground through reverse-biased diodes *D1XB* and *D2XB*.

Since current flows *into* a TTL input in the HIGH state, I_{IHmax} has a positive value. Most LS-TTL inputs have $I_{\text{IHmax}} = 20\ \mu\text{A}$.

Like CMOS outputs, TTL outputs can source or sink a certain amount of current depending on the state, HIGH or LOW:

I_{OLmax} The maximum current an output can sink in the LOW state while maintaining an output voltage no more than V_{OLmax}. Since current flows into the output, I_{OLmax} has a positive value, 8 mA for most LS-TTL outputs.

I_{OHmax} The maximum current an output can source in the HIGH state while maintaining an output voltage no less than V_{OHmin}. Since current flows out of the output, I_{OHmax} has a negative value, –400 μA for most LS-TTL outputs.

Notice that the value of I_{OLmax} for typical LS-TTL outputs is exactly 20 times the absolute value of I_{ILmax}. As a result, LS-TTL is said to have a *LOW-state fanout* of 20, because an output can drive up to 20 inputs in the LOW state. Similarly, the absolute value of I_{OHmax} is exactly 20 times I_{IHmax}, so LS-TTL is said to have a *HIGH-state fanout* of 20 also. The *overall fanout* is the lesser of the LOW- and HIGH-state fanouts.

LOW-state fanout

HIGH-state fanout

overall fanout

Loading a TTL output with more than its rated fanout has the same deleterious effects that were described for CMOS devices in Section 3.5.5 on page 111. That is, DC noise margins may be reduced or eliminated, transition times and delays may increase, and the device may overheat.

In general, two calculations must be carried out to confirm that an output is not being overloaded:

HIGH state The I_{IHmax} values for all of the driven inputs are added. This sum must be less than or equal to the absolute value of I_{OHmax} for the driving output.

LOW state The I_{ILmax} values for all of the driven inputs are added. The absolute value of this sum must be less than or equal to I_{OLmax} for the driving output.

For example, suppose you designed a system in which a certain LS-TTL output drives ten LS-TTL and three S-TTL gate inputs. In the HIGH state, a total of $10 \cdot 20 + 3 \cdot 50\ \mu\text{A} = 350\ \mu\text{A}$ is required. This is within an LS-TTL output's HIGH-state current-sourcing capability of 400 μA. But in the LOW state, a total of $10 \cdot 0.4 + 3 \cdot 2.0\ \text{mA} = 10.0\ \text{mA}$ is required. This is more than an LS-TTL output's LOW-state current-sinking capability of 8 mA, so the output is overloaded.

BURNED FINGERS If a TTL or CMOS output is forced to sink a lot more than I_{OLmax}, the device may be damaged, especially if high current is allowed to flow for more than a second or so. For example, suppose that a TTL output in the LOW state is short-circuited directly to the 5 V supply. The ON resistance, $R_{CE(sat)}$, of the saturated *Q5* transistor in a typical TTL output stage is less than 10 Ω. Thus, *Q5* must dissipate about $5^2/10$ or 2.5 watts. Don't try this yourself unless you're prepared to deal with the consequences! That's enough heat to destroy the device (and burn your finger) in a very short time.

*3.10.6 TTL Families

TTL families have evolved over the years in response to the demands of digital designers for better performance. As a result, several TTL families have come and gone, and today there are just a few surviving TTL families. Instead, TTL-compatible CMOS families are generally preferred for new designs. Some of the history of TTL is described in DDPPonline in Section TTL. All of these TTL families are compatible in that they use the same 5-V power-supply voltage and logic levels, but each family has its own advantages in terms of speed, power consumption, and cost.

74S (Schottky TTL)
74LS (Low-power Schottky TTL)
74AS (Advanced Schottky TTL)
74ALS (Advanced Low-power Schottky TTL)
74F (Fast TTL)

All current TTL families use Schottky transistors to improve their speed. The oldest of these are *74S (Schottky TTL)* and *74LS (Low-power Schottky TTL)*. Subsequent IC processing and circuit innovations led to three more Schottky logic families. The *74AS (Advanced Schottky TTL)* family offers speeds about twice as fast as 74S with about the same power consumption. The *74ALS (Advanced Low-power Schottky TTL)* family offers both lower power and higher speeds than 74LS. The *74F (Fast TTL)* family is positioned between 74AS and 74ALS in the speed/power tradeoff, and is probably the most popular choice for high-speed requirements in new TTL designs.

The important characteristics of these TTL families are summarized in Table 3-10. The first two rows of the table list the propagation delay (in nanoseconds) and the power consumption (in milliwatts) of a typical 2-input NAND gate in each family. The third row lists the corresponding speed-power product.

The remaining rows in Table 3-10 describe the input and output parameters of typical TTL gates in each of the families. Using this information, you can analyze the external behavior of TTL gates without knowing the details of the internal TTL circuit design. The input and output characteristics of specific components may vary from the representative values given in Table 3-10, so you must always consult the manufacturer's data book when analyzing a real design.

*3.10.7 A TTL Data Sheet

Table 3-11 shows part of a typical manufacturer's data sheet for the 74LS00. The 54LS00 listed in the data sheet is identical to the 74LS00, except that it is

levice may be
second or so.
uited directly
istor in a typ-
t $5^2/10$ or 2.5
onsequences!
ry short time.

ands of digital
ies have come
Instead, TTL-
s. Some of the
of these TTL
y voltage and
speed, power

e their speed.
chottky TTL).
ore Schottky
speeds about
The *74ALS*
er and higher
en 74AS and
ar choice for

nmarized in
(in nanosec-
NAND gate
roduct.
parameters
on, you can
etails of the
ecific com-
-10, so you
real design.

LS00. The
t that it is

Table 3-10 Characteristics of gates in TTL families.

		Family				
Description	***Symbol***	***74S***	***74LS***	***74AS***	***74ALS***	***74F***
Maximum propagation delay (ns)		3	9	1.7	4	3
Power consumption per gate (mW)		19	2	8	1.2	4
Speed-power product (pJ)		57	18	13.6	4.8	12
LOW-level input voltage (V)	V_{ILmax}	0.8	0.8	0.8	0.8	0.8
LOW-level output voltage (V)	V_{OLmax}	0.5	0.5	0.5	0.5	0.5
HIGH-level input voltage (V)	V_{IHmin}	2.0	2.0	2.0	2.0	2.0
HIGH-level output voltage (V)	V_{OHmin}	2.7	2.7	2.7	2.7	2.7
LOW-level input current (mA)	I_{ILmax}	−2.0	−0.4	−0.5	−0.2	−0.6
LOW-level output current (mA)	I_{OLmax}	20	8	20	8	20
HIGH-level input current (μA)	I_{IHmax}	50	20	20	20	20
HIGH-level output current (μA)	I_{OHmax}	−1000	−400	−2000	−400	−1000

specified to operate over the full "military" temperature and voltage range, and it costs more. Most TTL parts have corresponding 54-series (military) versions. Three sections of the data sheet are shown in the table:

recommended operating conditions

- *Recommended operating conditions* specify power-supply voltage, input-voltage ranges, DC output loading, and temperature values under which the device is normally operated.

electrical characteristics

- *Electrical characteristics* specify additional DC voltages and currents that are observed at the device inputs and output when it is operated under the recommended conditions:

I_{I}	Maximum input current for a very high HIGH input voltage.
I_{OS}	Output current with HIGH output shorted to ground.
I_{CCH}	Power-supply current when all outputs (on four NAND gates) are HIGH. (The number given is for the entire package, which contains four NAND gates, so the current per gate is one-fourth of the specified amount.)
I_{CCL}	Power-supply current when all outputs (on four NAND gates) are LOW.

Table 3-11 Typical manufacturer's data sheet for the 74LS00.

RECOMMENDED OPERATING CONDITIONS

Parameter	Description	SN54LS00			SN74LS00			Unit
		Min.	Nom.	Max.	Min.	Nom.	Max.	
V_{CC}	Supply voltage	4.5	5.0	5.5	4.75	5.0	5.25	V
V_{IH}	High-level input voltage	2.0			2.0			V
V_{IL}	Low-level input voltage			0.7			0.8	V
I_{OH}	High-level output current			–0.4			–0.4	mA
I_{OL}	Low-level output current			4			8	mA
T_A	Operating free-air temperature	–55		125	0		70	°C

ELECTRICAL CHARACTERISTICS OVER RECOMMENDED FREE-AIR TEMPERATURE RANGE

Parameter	Test Conditions(1)	SN54LS00			SN74LS00			Unit
		Min.	Typ.(2)	Max.	Min.	Typ.(2)	Max.	
V_{IK}	V_{CC} = Min., I_N = –18 mA			–1.5			–1.5	V
V_{OH}	V_{CC} = Min., V_{IL} = Max., I_{OH} = –0.4 mA	2.5	3.4		2.7	3.4		V
V_{OL}	V_{CC} = Min., V_{IH} = 2.0 V, I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
	V_{CC} = Min., V_{IH} = 2.0 V, I_{OL} = 8 mA					0.35	0.5	V
I_I	V_{CC} = Max., V_I = 7.0 V			0.1			0.1	mA
I_{IH}	V_{CC} = Max., V_I = 2.7 V			20			20	μA
I_{IL}	V_{CC} = Max., V_I = 0.4 V			–0.4			–0.4	mA
I_{OS}(3)	V_{CC} = Max.	–20		–100	–20		–100	mA
I_{CCH}	V_{CC} = Max., V_I = 0 V		0.8	1.6		0.8	1.6	mA
I_{CCL}	V_{CC} = Max., V_I = 4.5 V		2.4	4.4		2.4	4.4	mA

SWITCHING CHARACTERISTICS, V_{CC} = 5.0 V, T_A = 25°C

Parameter	From (Input)	To (Output)	Test Conditions	Min.	Typ.	Max.	Unit
t_{PLH}	A or B	Y	R_L = 2 kΩ, C_L = 15 pF		9	15	ns
t_{PHL}					10	15	ns

NOTES:

1. For conditions shown as Max. or Min., use appropriate value specified under Recommended Operating Conditions.

2. All typical values are at V_{CC} = 5.0 V, T_A = 25 °C.

3. Not more than one output should be shorted at a time; duration of short-circuit should not exceed one second.

.S00	
Max.	Unit
5.25	V
	V
0.8	V
−0.4	mA
8	mA
70	°C

RANGE

00	
Max.	Unit
−1.5	V
	V
0.4	V
0.5	V
0.1	mA
20	μA
-0.4	mA
·100	mA
1.6	mA
1.4	mA

ax.	Unit
5	ns
5	ns

ns.

- *Switching characteristics* give maximum and typical propagation delays under "typical" operating conditions of $V_{CC} = 5$ V and $T_A = 25°C$. A conservative designer must increase these delays by 5%–10% to account for different power-supply voltages and temperatures, and even more under heavy loading conditions.

switching characteristics

A fourth section is also included in the manufacturer's data book:

- *Absolute maximum ratings* indicate the worst-case conditions for operating or storing the device without damage.

absolute maximum ratings

A complete data book also shows test circuits that are used to measure the parameters when the device is manufactured, and graphs that show how the typical parameters vary with operating conditions such as power-supply voltage (V_{CC}), ambient temperature (T_A), and load (R_L, C_L).

*3.10.8 CMOS/TTL Interfacing

A digital designer selects a "default" logic family to use in a system, based on general requirements of speed, power, cost, and so on. However, the designer may select devices from other families in some cases because of availability or other special requirements. (For example, not all 74LS part numbers are available in 74HCT, and vice versa.) Thus, it's important for a designer to understand the implications of connecting TTL outputs to CMOS inputs, and vice versa.

There are several factors to consider in TTL/CMOS interfacing, and the first is noise margin. The LOW-state DC noise margin depends on V_{OLmax} of the driving output and V_{ILmax} of the driven input, and equals $V_{ILmax} - V_{OLmax}$. Similarly, the HIGH-state DC noise margin equals $V_{OHmin} - V_{IHmin}$. Figure 3-76 shows the relevant numbers for TTL and CMOS families.

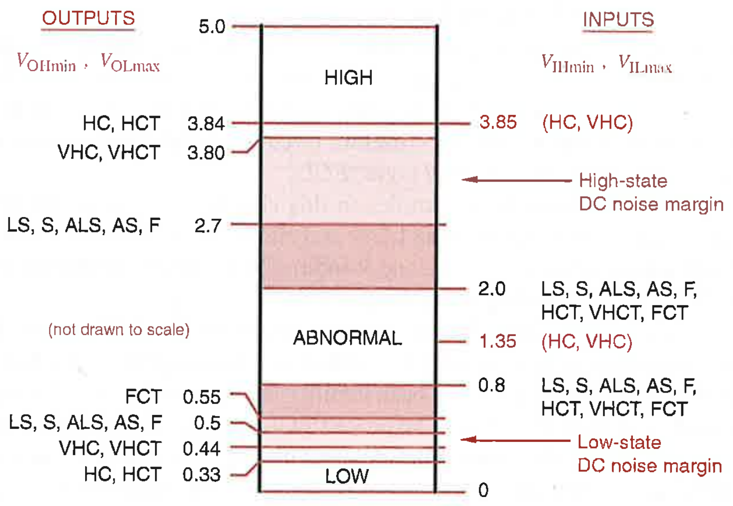


Figure 3-76 Output and input levels for interfacing TTL and CMOS families. (Note that HC and VHC inputs are not TTL compatible.)

For example, the LOW-state DC noise margin of HC or HCT driving TTL is 0.8 – 0.33 = 0.47 V, and the HIGH-state is 3.84 – 2.0 =1.84 V. On the other hand, the HIGH-state margin of TTL driving HC or VHC is 2.7 – 3.85 = –1.15 V. In other words, TTL driving HC or AC doesn't work, unless the TTL HIGH output happens to be higher and the CMOS HIGH input threshold happens to be lower by a total of 1.15 V compared to their worst-case specs. To drive CMOS inputs properly from TTL outputs, the CMOS devices should be HCT, VHCT, or FCT rather than HC or VHC.

The next factor to consider is fanout. As with pure TTL (Section 3.10.5), a designer must sum the input current requirements of devices driven by an output and compare with the output's capabilities in both states. Fanout is not a problem when TTL drives CMOS, since CMOS inputs require almost no current in either state. On the other hand, TTL inputs, especially in the LOW state, require substantial current, especially compared to HC and HCT output capabilities. For example, an HC or HCT output can drive ten LS or only two S-TTL inputs.

The last factor is capacitive loading. We've seen that load capacitance increases both the delay and the power dissipation of logic circuits. Increases in delay are especially noticeable with HC and HCT outputs, whose transition times increase about 1 ns for each 5 pF of load capacitance. The transistors in FCT outputs have very low "on" resistances, so their transition times increase only about 0.1 ns for each 5 pF of load capacitance.

For a given load capacitance, power-supply voltage, and application, all of the CMOS families have similar dynamic power dissipation, since each variable in the CV^2f equation is the same. On the other hand, TTL outputs have somewhat lower dynamic power dissipation, since the voltage swing between TTL HIGH and LOW levels is smaller.

*3.10.9 Emitter-Coupled Logic

current-mode logic (CML)

emitter-coupled logic (ECL)

The key to reducing propagation delay in a bipolar logic family is to prevent a gate's transistors from saturating. In Section 3.10.2, we learned how Schottky diodes prevent saturation in TTL gates. However, it is also possible to prevent saturation by using a radically different circuit structure, called *current-mode logic (CML)* or *emitter-coupled logic (ECL)*.

Unlike the other logic families in this chapter, ECL does not produce a large voltage swing between the LOW and HIGH levels. Instead, it has a small voltage swing, less than a volt, and it internally switches current between two possible paths, depending on the output state.

The first ECL logic family was introduced by General Electric in 1961. The concept was later refined by Motorola and others to produce the still popular 10K and 100K ECL families. These families are extremely fast, offering propagation delays as short as 1 ns. The newest ECL family, ECLinPS (literally, ECL in picoseconds), offers maximum delays under 0.5 ns (500 ps), including the signal delay getting on and off of the IC package. Throughout the evolution of

HCT driving TTL
34 V. On the other
' −3.85 = −1.15 V
ss the TTL HIGH
old happens to be
. To drive CMOS
e HCT, VHCT, or

ection 3.10.5), a
ven by an output
is not a problem
current in either
V state, require
capabilities. For
TL inputs.
ad capacitance
ts. Increases in
hose transition
e transistors in
times increase

lication, all of
each variable
ive somewhat
n TTL HIGH

to prevent a
ow Schottky
e to prevent
urrent-mode

: produce a
has a small
etween two

c in 1961.
ill popular
ing propa-
ally, ECL
uding the
olution of

digital circuit technology, some type of ECL has always been the fastest technology for discrete, packaged logic components.

Still, commercial ECL families aren't nearly as popular as CMOS and TTL, mainly because they consume much more power. In fact, high power consumption made the design of ECL supercomputers, such as the Cray-1 and Cray-2, as much of a challenge in cooling technology as in digital design. Also, ECL has a poor speed-power product, does not provide a high level of integration, has fast edge rates requiring design for transmission-line effects in most applications, and is not directly compatible with TTL and CMOS. Nevertheless, ECL still finds its place as a logic and interface technology in very high-speed communications gear, including fiber-optic transceiver interfaces for gigabit Ethernet and Asynchronous Transfer Mode (ATM) networks.

The basic idea of current-mode logic is illustrated by the inverter/buffer circuit in Figure 3-77. This circuit has both an inverting output (OUT1) and a noninverting output (OUT2). Two transistors are connected as a *differential amplifier* with a common emitter resistor. The supply voltages for this example are $V_{CC} = 5.0$, $V_{BB} = 4.0$, and $V_{EE} = 0$ V, and the input LOW and HIGH levels are defined to be 3.6 and 4.4 V. This circuit actually produces output LOW and HIGH levels that are 0.6 V higher (4.2 and 5.0 V), but this is corrected in real ECL circuits.

differential amplifier

When V_{IN} is HIGH, as shown in the figure, transistor *Q1* is on, but not saturated, and transistor *Q2* is OFF. This is true because of a careful choice of resistor values and voltage levels. Thus, V_{OUT2} is pulled to 5.0 V (HIGH) through *R2*, and it can be shown that the voltage drop across *R1* is about 0.8 V, so that V_{OUT1} is about 4.2 V (LOW).

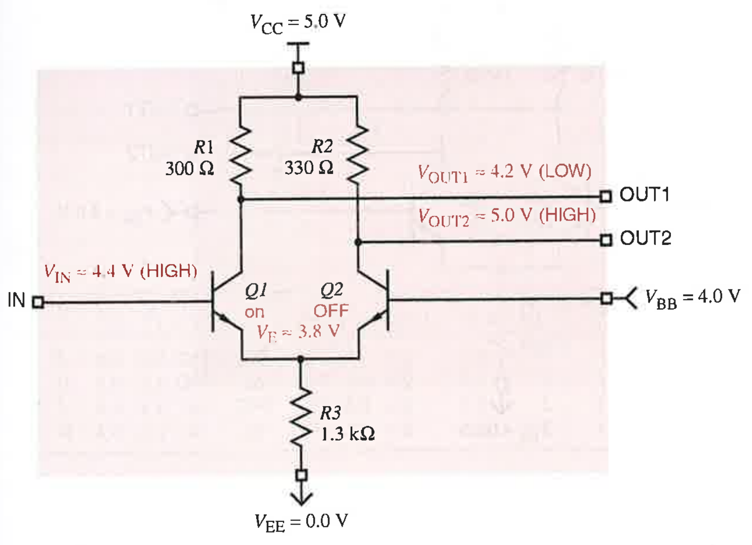


Figure 3-77
Basic ECL inverter/buffer circuit with input HIGH.

When V_{IN} is LOW, as shown in Figure 3-78, transistor *Q2* is on, but not saturated, and transistor *Q1* is OFF. Thus, V_{OUT1} is pulled to 5.0 V through *R1*, and it can be shown that V_{OUT2} is about 4.2 V.

differential outputs

The outputs of this inverter are called *differential outputs* because they are always complementary, and it is possible to determine the output state by looking at the difference between the output voltages ($V_{OUT1} - V_{OUT2}$) rather than their absolute values. That is, the output is 1 if ($V_{OUT1} - V_{OUT2}$) > 0, and it is 0 if ($V_{OUT1} - V_{OUT2}$) < 0. It is possible to build input circuits with two wires per logical input that define the logical signal value in this way; these are called *differential inputs*.

differential inputs

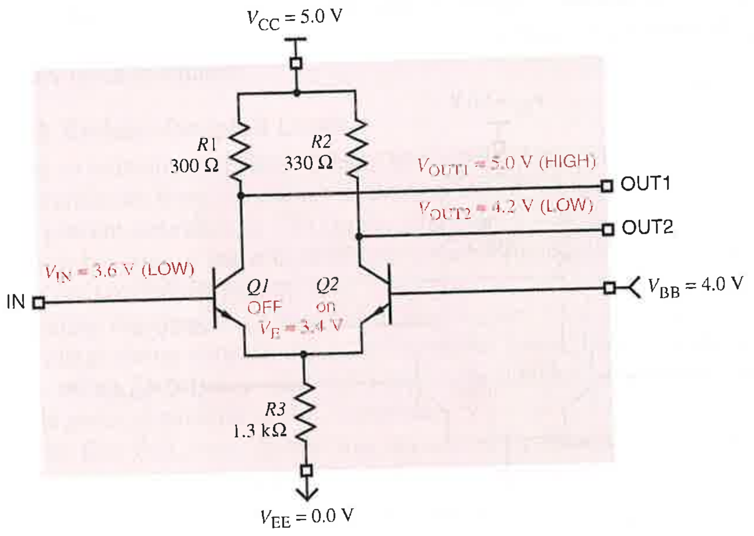
Differential signals are used in most ECL "interfacing" and "clock distribution" applications because of their low skew and high noise immunity. They are "low skew" because the timing of a 0-to-1 or 1-to-0 transition does not depend critically on voltage thresholds, which may change with temperature or between devices. Instead, the timing depends only on when the voltages cross over relative to each other. Similarly, the "relative" definition of 0 and 1 provides outstanding noise immunity, since noise created in the power supply distribution or coupled from external sources tends to be a *common-mode signal* that affects both differential signals similarly, leaving the difference value unchanged.

common-mode signal

It is also possible, of course, to determine the logic value by sensing the absolute voltage level of one input signal, called a *single-ended input*. Single-ended signals are used in most ECL "logic" applications to avoid the obvious

single-ended input

Figure 3-78
Basic ECL inverter/buffer circuit with input LOW.



on, but not
through $R1$,

use they are
ut state by
UT2) rather
) > 0, and it
1 two wires
e are called

lock distri-
inity. They
1 does not
perature or
ages cross
1 provides
istribution
hat affects
ged.
ensing the
t. Single-
e obvious

) V

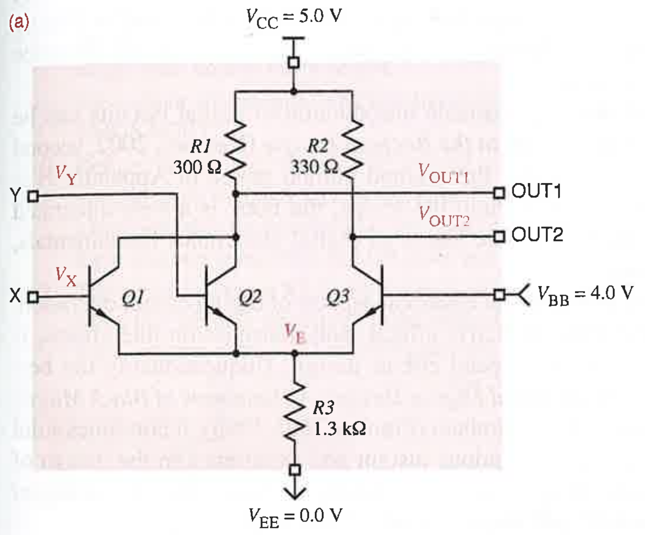
expense of doubling the number of signal lines. The basic CML inverter in Figure 3-77 and 3-78 has a single-ended input. It always has both "outputs" available internally; the circuit is actually either an inverter or a noninverting buffer, depending on whether we use OUT1 or OUT2.

To perform logic with the basic circuit of Figure 3-78, we place additional transistors in parallel with $Q1$. For example, Figure 3-79 shows a 2-input ECL OR/NOR gate. If any input is HIGH, the corresponding input transistor is active, and V_{OUT1} is LOW (NOR output). At the same time, $Q3$ is OFF, and V_{OUT2} is HIGH (OR output).

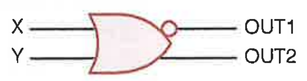
Additional information on ECL circuits can be found at DDPPonline in Section ECL. This includes a discussion of the common ECL families—10K and 100K—as well as positive ECL (PECL) operation.

Figure 3-79 ECL 2-input OR/NOR gate: (a) circuit diagram; (b) function table; (c) logic symbol; (d) truth table.

(a)



(c)



(b)

X	Y	V_X	V_Y	$Q1$	$Q2$	$Q3$	V_E	V_{OUT1}	V_{OUT2}	OUT1	OUT2
L	L	3.6	3.6	OFF	OFF	on	3.4	5.0	4.2	H	L
L	H	3.6	4.4	OFF	on	OFF	3.8	4.2	5.0	L	H
H	L	4.4	3.6	on	OFF	OFF	3.8	4.2	5.0	L	H
H	H	4.4	4.4	on	on	OFF	3.8	4.2	5.0	L	H

(d)

X	Y	OUT1	OUT2
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

References

Students who need to study the basics may wish to consult "Electrical Circuits Review" by Bruce M. Fleischer. This 20-page tutorial covers all of the basic circuit concepts that are used in this chapter. It appears both as an appendix in this book's first edition and as a `.pdf` file in Section Elec at DDPPonline.

After seeing the results of last few decades' amazing pace of development in digital electronics, it's easy to forget that logic circuits had an important place in technologies that came before the transistor. In Chapter 5 of *Introduction to the Methodology of Switching Circuits* (Van Nostrand, 1972), George J. Klir shows how logic can be (and has been) performed by a variety of physical devices, including relays, vacuum tubes, and pneumatic systems.

For another perspective on the electronics material in this chapter, you can consult almost any modern electronics text. Many contain a much more analytical discussion of digital circuit operation; for example, see *Introduction to Electronic Circuit Design* by R. Spencer and M. Ghausi (Prentice Hall, 2003). A good introduction to ICs and important logic families can be found in *Digital Integrated Circuits* by J. M. Rabaey, A. Chandrakasan, and B. Nikolic (Prentice Hall, 2003, second edition).

A light-hearted and very readable introduction to digital circuits can be found in Clive Maxfield's *Bebop to the Boolean Boogie* (Newnes, 2002, second edition). Some people think that the seafood gumbo recipe in Appendix H is alone worth the price! Even without the recipe, the book is a well-illustrated classic that guides you through the basics of digital electronics fundamentals, components, and processes.

A sound understanding of the electrical aspects of digital circuit operation, including capacitive effects, inductive effects, and transmission-line effects, is mandatory for successful high-speed circuit design. Unquestionably the best book on this subject is *High-Speed Digital Design: A Handbook of Black Magic*, by Howard Johnson and Martin Graham (Prentice Hall, 1993). It combines solid electronics principles with tremendous insight and experience in the design of practical digital systems. Also see Johnson's follow-on book, *High-Speed Signal Propagation: Advanced Black Magic* (Prentice Hall, 2003).

Characteristics of today's logic families can be found in the data sheets published by the device manufacturers. Old-time digital designers are proud of their collections of thick databooks published by the device manufacturers, but nowadays all of the latest specs can be found on the Web. Among the better sites for logic-family data sheets and design application notes are `www.ti.com` (Texas Instruments), `www.philips.com`, and `www.fairchildsemi.com`.

The JEDEC (Joint Electron Device Engineering Council) standards for digital logic levels can be found on JEDEC's web site, `www.jedec.org`. The JEDEC standards for 3.3-V, 2.5-V, 1.8-V, and 1.5-V logic were published in 1994, 1995, 1997, and 2001, respectively.

Circuits
asic cir-
x in this

lopment
nt place
ction to
J. Klir
hysical

ou can
nalyti-
tion to
03). A
Digital
rentice

an be
econd
H is
trated
ntals,

ation,
ts, is
best
agic,
solid
n of
gnal

eets
d of
but
ites
xas

for
The
l in

Drill Problems

3.1 The Stub Series Terminated low Voltage Logic (SSTV) family, used for SDRAM modules, defines a LOW signal to be in the range 0.0–0.7 V and a HIGH signal to be in the range 1.7–2.5 V. Under a positive-logic convention, indicate the logic value associated with each of the following signal levels:

(a)	0.0 V	(b)	0.7 V	(c)	1.7 V	(d)	–0.6 V
(e)	1.6 V	(f)	–2.0 V	(g)	2.5 V	(h)	3.3 V

3.2 Repeat Drill 3.1 using a negative-logic convention.

3.3 Discuss how a logic buffer amplifier is different from an audio amplifier.

3.4 Is a buffer amplifier equivalent to a 1-input AND gate or a 1-input OR gate?

3.5 True or false: For a given set of input values, a NAND gate produces the opposite output as an OR gate with inverted inputs.

3.6 Write two completely different definitions of "gate" used in this chapter.

3.7 How many transistors are used in a 2-input CMOS NAND gate? How many of each type are used?

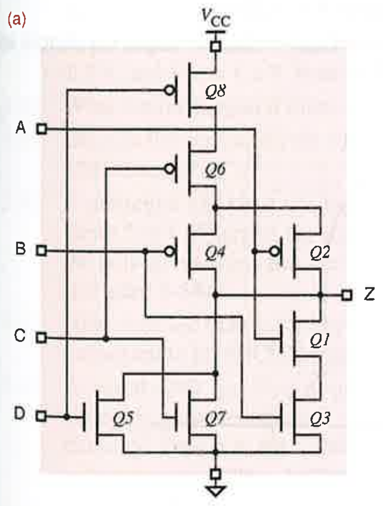
3.8 (Hobbyists only.) Draw an equivalent circuit for a CMOS NAND gate using two single-pole, double-throw relays.

3.9 For a given silicon area, which is likely to be faster, a CMOS NAND gate or a CMOS NOR?

3.10 Define "fan-in" and "fanout." Which one are you likely to have to calculate?

3.11 The circuit in Figure X3.11(a) is a type of CMOS AND-OR-INVERT gate. Write a function table for this circuit in the style of Figure 3-15(b), and a corresponding logic diagram using AND and OR gates and inverters.

3.12 The circuit in Figure X3.11(b) is a type of CMOS OR-AND-INVERT gate. Write a function table for this circuit in the style of Figure 3-15(b), and a corresponding logic diagram using AND and OR gates and inverters.



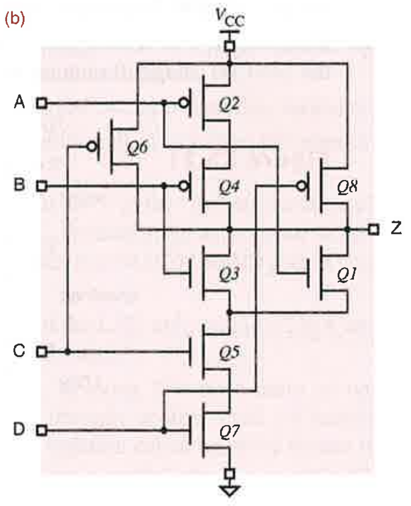


Figure X3.11

3.13 Draw the circuit diagram, function table, and logic symbol for a 3-input CMOS NOR gate in the style of Figure 3-16.

3.14 Draw switch models in the style of Figure 3-14 for a 2-input CMOS NOR gate for all four input combinations.

3.15 Draw a circuit diagram, function table, and logic symbol for a CMOS OR gate in the style of Figure 3-19.

3.16 Which has fewer transistors, a CMOS inverting gate or a noninverting gate?

3.17 Name and draw the logic symbols of four different 3-input CMOS gates that each use six transistors.

3.18 Which 8-input CMOS gate would you expect to be faster, NAND or AND? Why?

3.19 How is it that perfume can be bad for digital designers?

3.20 Using the data sheet in Table 3-3, determine the worst-case LOW-state and HIGH-state DC noise margins of the 74HC00. State any assumptions required by your answer.

3.21 How much high-state DC noise margin is available in an inverter whose transfer characteristic under worst-case conditions is shown in Figure X3.21? How much low-state DC noise margin is available? (Assume 1.5-V and 3.5-V thresholds for LOW and HIGH.)

3.22 Section 3.5 defines seven different electrical parameters for CMOS circuits. Using the data sheet in Table 3-3, determine the worst-case value of each of these for the 74HC00. State any assumptions required by your answer.

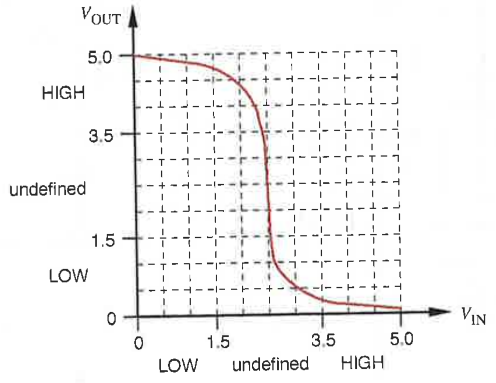
3.23 Based on the conventions and definitions in Section 3.4, if the current at a device output is specified as a negative number, is the output sourcing current or sinking current?

3.24 Across the range of valid HIGH input levels, 3.15–5.0 V, at what input level would you expect the 74HC00 (Table 3-3) to consume the most power?

3.25 Determine the LOW-state and HIGH-state DC fanout of the 74HC00 when it drives 74ALS00-like inputs. (Refer to Tables 3-3 and 3-10.)

3.26 Estimate the "on" resistances of the *p*-channel and *n*-channel output transistors of the 74HC00 using information in Table 3-3.

Figure X3.21



3.27 For each of the following resistive loads, determine whether the output drive specifications of the 74HC00 over the commercial operating range are exceeded. Refer to Table 3-3, and use $V_{OLmax} = 0.33$ V, $V_{OHmin} = 3.84$ V, and $V_{CC} = 5.0$ V. You may not exceed I_{OLmax} or I_{OHmax} in any state.

(a)	120 Ω to V_{CC}	(b)	270 Ω to V_{CC} and 330 Ω to GND
(c)	820 Ω to GND	(d)	470 Ω to V_{CC} and 470 Ω to GND
(e)	1 kΩ to V_{CC}	(f)	1.2 kΩ to V_{CC} and 820 Ω to GND
(g)	4.7 kΩ to V_{CC}	(h)	1.2 kΩ to V_{CC} and 1 kΩ to GND

3.28 Under what circumstances is it safe to allow an unused CMOS input to float?

3.29 Explain "latch up" and the circumstances under which it occurs.

3.30 Explain why replacing small decoupling capacitors to larger ones with larger capacitance may not be a good idea.

3.31 When is it important to hold hands with a friend?

3.32 Name the two components of CMOS logic gate's delay. How are either or both affected by the direction of the output transition?

3.33 Determine the *RC* time constant for each of the following resistor-capacitor combinations:

(a)	$R = 100\ \Omega$, $C = 50$ pF	(b)	$R = 4.7\ \text{k}\Omega$, $C = 150$ pF
(c)	$R = 47\ \Omega$, $C = 47$ pF	(d)	$R = 1\ \text{k}\Omega$, $C = 100$ pF

3.34 Which would you expect to have a bigger effect on the power consumption of a CMOS circuit, a 5% increase in power-supply voltage or a 5% increase in internal and load capacitance?

3.35 Explain why the number of CMOS inputs connected to the output of a CMOS gate generally is not limited by DC fanout considerations.

3.36 It is possible to operate 74VHC CMOS devices with a 2.5-volt power supply. How much power does this typically save, compared to 5-volt operation?

3.37 A particular Schmitt-trigger inverter has $V_{ILmax} = 0.8$ V, $V_{IHmin} = 2.0$ V, $V_{T+} = 1.7$ V, and $V_{T-} = 1.2$ V. How much hysteresis does it have?

3.38 What would happen if three-state outputs turned on faster than they turned off?

3.39 Discuss the pros and cons of larger vs. smaller pull-up resistors for open-drain CMOS outputs.

3.40 A particular LED has a voltage drop of about 2.0 V in the "on" state and requires about 5 mA of current for normal brightness. Determine an appropriate value for the pull-up resistor when the LED is connected to a 74AC00 NAND gate as shown in Figure 3-54(a).

3.41 How does the answer for Drill 3.40 change if the LED only requires 2 mA and is connected to a 74HC00 as shown in Figure 3-54(b)?

3.42 A wired-AND function is obtained simply by tying two open-drain or open-collector outputs together, without going through another level of transistor circuitry. How is it, then, that a wired-AND function can actually be slower than a discrete AND gate?

3.43 Which CMOS or TTL logic family in this chapter has the strongest output driving capability?

3.44 Concisely summarize the difference between HC and ACT logic families.

3.45 Why don't the specifications for FCT devices include parameters like V_{OLmaxC} that apply to CMOS loads, as HCT and ACT specifications do?

3.46 How do FCT-T devices reduce power consumption compared to FCT devices?

3.47 How many diodes are required for an n-input diode AND gate?

3.48 Are TTL outputs more capable of sinking current or sourcing current?

3.49 Compute the maximum fanout for each of the following cases of a TTL output driving multiple TTL inputs. Also indicate how much "excess" driving capability is available in the LOW or HIGH state for each case.

(a)	74LS driving 74AS	(b)	74LS driving 74F
(c)	74F driving 74LS	(d)	74F driving 74AS
(e)	74AS driving 74S	(f)	74S driving 74ALS
(g)	74ALS driving 74S	(h)	74F driving 74F

3.50 Which resistor dissipates more power, the pull-down for an unused LS-TTL NOR-gate input, or the pull-up for an unused LS-TTL NAND-gate input? Use the maximum allowable resistor value in each case.

3.51 Which would you expect to be faster, a CMOS AND gate or a CMOS AND-OR-INVERT gate, assuming all transistors switch at the same speed? Why?

3.52 Describe the key benefit of Schottky transistors in TTL.

3.53 Using the data sheet from the Texas Instruments (`www.ti.com`), determine the worst-case LOW-state and HIGH-state DC noise margins of the 74ALS00.

3.54 Sections 3.10.4 and 3.10.5 define eight different electrical parameters for TTL circuits. Using the data sheet from Texas Instruments (`www.ti.com`), determine the worst-case value of each of these for the 74ALS00.

3.55 For each of the following resistive loads, determine whether the output drive specifications of the 74LS00 over the commercial operating range are exceeded. (Refer to Table 3-11, and use $V_{OLmax} = 0.5$ V and $V_{CC} = 5.0$ V.)

(a)	470 Ω to V_{CC}	(b)	330 Ω to V_{CC} and 470 Ω to GND
(c)	6.8 kΩ to GND	(d)	910 Ω to V_{CC} and 1200 Ω to GND
(e)	620 Ω to V_{CC}	(f)	510 Ω to V_{CC} and 470 Ω to GND
(g)	5.1 kΩ to GND	(h)	464 Ω to V_{CC} and 510 Ω to GND

3.56 Compute the LOW-state and HIGH-state DC noise margins for each of the following cases of a TTL output driving a TTL-compatible CMOS input, or vice versa.

(a)	74HCT driving 74LS	(b)	74ALS driving 74HCT
(c)	74AS driving 74VHCT	(d)	74VHCT driving 74F

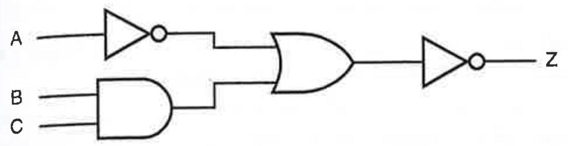


Figure X3.59

3.57 Compute the maximum fanout for each of the following cases of a TTL-compatible CMOS output driving multiple inputs in a TTL logic family. Also indicate how much "excess" driving capability is available in the LOW or HIGH state for each case.

(a) 74HCT driving 74LS (b) 74VHCT driving 74S
(c) 74VHCT driving 74ALS (d) 74HCT driving 74AS

3.58 For a given load capacitance and transition rate, which logic family in this chapter has the highest dynamic power dissipation? How does it compare to the family with the lowest dynamic power dissipation?

Exercises

3.59 Design a CMOS circuit that has the functional behavior shown in Figure X3.59. (*Hint:* Only eight transistors are required.)

3.60 Design a CMOS circuit that has the functional behavior shown in Figure X3.60. (*Hint:* Only eight transistors are required.)

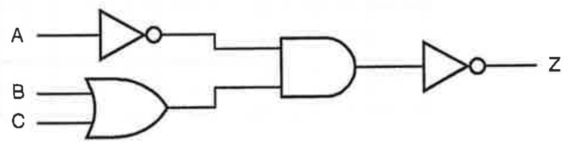
3.61 Draw a circuit diagram, function table, and logic symbol in the style of Figure 3-19 for a CMOS gate with two inputs A and B and an output Z, where Z=1 if A=0 and B=1, and Z=0 otherwise. (*Hint:* Only six transistors are needed.)

3.62 Draw a circuit diagram, function table, and logic symbol in the style of Figure 3-19 for a CMOS gate with two inputs A and B and an output Z, where Z=0 if A=1 and B=0, and Z=1 otherwise. (*Hint:* Only six transistors are needed.)

3.63 Draw a figure showing the logical structure of an 8-input CMOS NAND gate, assuming that at most 4-input NAND and 2-input NOR gate circuits are practical. Using your general knowledge of CMOS characteristics, select a circuit structure that minimizes the NAND gate's propagation delay for a given silicon area, and explain why this is so.

3.64 The circuit designers of TTL-compatible CMOS families presumably could have made the voltage drop across the "on" transistor under load in the HIGH state as little as it is in the LOW state, simply by making the *p*-channel transistors bigger. Why do you suppose they didn't bother to do this?

Figure X3.60



3.65 How much current and power are "wasted" in Figure 3-32(b)?

3.66 Perform a detailed calculation of V_{OUT} in Figures 3-33 and 3-34. (*Hint:* Create a Thévenin equivalent for the CMOS inverter in each figure.)

3.67 Consider the dynamic behavior of a CMOS output driving a given capacitive load. If the resistance of the charging path is double the resistance of the discharging path, is the rise time exactly twice the fall time? If not, what other factors affect the transition times?

3.68 Analyze the fall time of the CMOS inverter output of Figure 3-37 with R_L=900 Ω and V_L=2.0 V. Compare your result with the result in Section 3.6.1 and explain.

3.69 Repeat Exercise 3.68 for rise time.

3.70 Assuming that the transistors in an FCT CMOS three-state buffer are perfect zero-delay on-off devices that switch at an input threshold of 1.5 V, determine the value of t_{PLZ} for the test circuit and waveforms in Figure 3-24. (*Hint:* You have to determine the time using an *RC* time constant.)

3.71 Repeat Exercise 3.70 for t_{PHZ}.

3.72 Using the specifications in Table 3-7, estimate the "on" resistances of the *p*-channel and *n*-channel transistors in 74HCT-series CMOS logic.

3.73 Create a 4×4×2×2 matrix of worst-case DC noise margins for the following CMOS interfacing situations: an (HC, HCT, VHC, or VHCT) output driving an (HC, HCT, VHC, or VHCT) input with a (CMOS, TTL) load in the (LOW, HIGH) state; Figure X3.73 illustrates. (*Hints:* There are 64 different combinations, but many give identical results. Some combinations yield negative margins.)

3.74 The 74LVC00 is capable of being driven by up to 5.5 V, even when it is operating at only 1.8 V. Using Figure 3-62, determine the DC noise margins when a 74LVC00 is driven by a) 3.3-V CMOS, and b) 2.5-V CMOS.

3.75 Using Figure 3-62, determine the DC noise margins for 5-V-tolerant, 3.3-V CMOS driving 5-V CMOS logic with TTL input levels, and vice versa.

3.76 Using Figure 3-62, determine the DC noise margins for 3.3-V-tolerant, 2.5-V CMOS driving 3.3-V CMOS, and vice versa.

3.77 Using Figure 3-62, determine the DC noise margins for a) 2.5-V CMOS driving itself, and b) 1.8-V CMOS driving itself.

Figure X3.73

Output \ Input	HC		HCT		VHC		VHCT	
HC	CL	TL	CL	TL	CL	TL	CL	TL
	CH	TH	CH	TH	CH	TH	CH	TH
HCT	CL	TL	CL	TL	CL	TL	CL	TL
	CH	TH	CH	TH	CH	TH	CH	TH
VHC	CL	TL	CL	TL	CL	TL	CL	TL
	CH	TH	CH	TH	CH	TH	CH	TH
VHCT	CL	TL	CL	TL	CL	TL	CL	TL
	CH	TH	CH	TH	CH	TH	CH	TH

Key:
CL = CMOS load, LOW
CH = CMOS load, HIGH
TL = TTL load, LOW
TH = TTL load, HIGH

3.78 In the
and fc
drivin
powei

3.79 Consi
purpo
transi
the pu
what

3.80 Using
perfo

3.81 Calcu
gates

3.82 Redr
trans
for th
draw

3.83 Mod
High
no m

3.84 Usin
each

3.85 Show
be d

3.86 A di
had
circu
gate
colo
the

3.78 In the LED example in Section 3.7.5, a designer chose a resistor value of 390 Ω and found that the open-drain gate was able to maintain its output at 0.3 V while driving the LED. How much current flows through the LED, and how much power is dissipated by the pull-up resistor in this case?

3.79 Consider a CMOS 8-bit binary counter (Section 8.4) clocked at 16 MHz. For the purpose of computing the counter's dynamic power dissipation, what is the transition frequency of the least significant bit? Of the most significant bit? For the purpose of determining the dynamic power dissipation of the eight output bits, what frequency should be used?

3.80 Using only AND and NOR gates, draw a logic diagram for the logic function performed by the circuit in Figure 3-56.

3.81 Calculate the approximate output voltage at Z in Figure 3-57, assuming that the gates are HCT-series CMOS.

3.82 Redraw the circuit diagram of a CMOS 3-state buffer in Figure 3-49 using actual transistors instead of NAND, NOR, and inverter symbols. Can you find a circuit for the same function that requires a smaller total number of transistors? If so, draw it.

3.83 Modify the CMOS 3-state buffer circuit in Figure 3-49 so that the output is in the High-Z state when the enable input is HIGH. The modified circuit should require no more transistors than the original.

3.84 Using information in Table 3-3, estimate how much current can flow through each output pin if the outputs of two different 74HC00s are fighting.

3.85 Show that at a given power-supply voltage, an FCT-type I_{CCD} specification can be derived from an HCT/ACT-type C_{PD} specification, and vice versa.

3.86 A digital designer found a problem in a certain circuit's function after the circuit had been released to production and 1000 copies of it built. A portion of the circuit is shown in Figure X3.86 in black; all of the gates are 74HCT00 NAND gates. The digital designer fixed the problem by adding the two diodes shown in color. What do the diodes do? Describe both the logical effects of this change on the circuit's function and the electrical effects on the circuit's noise margins.

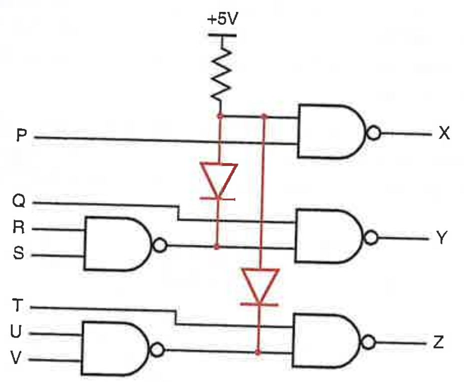


Figure X3.86

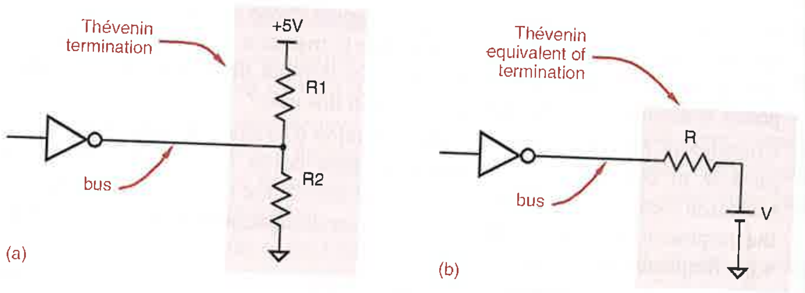


Figure X3.87

3.87 A *Thévenin termination* for an open-collector or three-state bus has the structure shown in Figure X3.87(a). The idea is that, with appropriate values of *R1* and *R2*, this circuit is equivalent to the termination in (b) for any desired values of *V* and *R*. The value of *V* determines the voltage on the bus when no device is driving it, and the value of *R* is selected to match the characteristic impedance of the bus for transmission-line purposes (Section Zo at DDPPonline). For each of the following pairs of *V* and *R*, determine the required values of *R1* and *R2*.

(a) $V = 3.0, R = 120$ (b) $V = 2.7, R = 179$

(c) $V = 2.4, R = 150$ (d) $V = 1.5, R = 50$

3.88 For each of the *R1* and *R2* pairs in Exercise 3.87, determine whether the termination can be properly driven by a three-state output in each of the following logic families: 74LS, 74S, 74FCT-T. For proper operation, the family's I_{OL} and I_{OH} specs must not be exceeded when $V_{OL} = V_{OLmax}$ and $V_{OH} = V_{OHmin}$, respectively.

3.89 Determine the total power dissipation of the circuit in Figure X3.89 as function of transition frequency *f* for two realizations: (a) using 74LS gates; (b) using 74HC gates. Assume that input capacitance is 3 pF for a TTL gate and 7 pF for a CMOS gate, that a 74LS gate has an internal power-dissipation capacitance of 20 pF, and that there is an additional 20 pF of stray wiring capacitance in the circuit. Also assume that the X, Y, and Z inputs are always HIGH, and that input C is driven with a CMOS-level square wave with frequency *f*. Other information that you need for this problem can be found in Tables 3-5 and 3-10. State any other assumptions that you make. At what frequency does the TTL circuit dissipate less power than the CMOS circuit?

3.90 Find a commercially available 74-series device with a very long part number, based on the logic family and the device number, but excluding the package type, temperature range, and so on. You should be able to beat 74ALVCH16244.

Figure X3.89