62-1334

Radio Shack®

BASIC DIGITAL ELECTRONICS

Explains digital system functions and how digital circuits are used to build them.

A	B	Q
0	0	1
0	1	0
1	0	0

- For anyone who wants to understand digital electronics
- Stresses basic concepts of digital circuits and systems
- Discusses detailed circuits
- Includes an introduction to integrated circuits
- Detailed illustrations support easy-to-read text
- Easy-to-understand presentations
- Practical worked-out examples demonstrate circuit applications
- End-of-chapter quizzes and problems reinforce learning

By: Alvis J. Evans

Developed and Published by
Master Publishing, Inc.

The clock is digital! The audio is digital! The TV is digital! Digital—what does it mean? Why is it that electronic systems are being designed using digital electronic circuits? Find the answer to these questions, and more, as you learn the difference between analog and digital systems, the functions required to design digital systems, the circuits used to make decisions, code conversions, data selections, adding and subtracting, interfacing and storage, and the circuits that keep all operations in time and under control.

BASIC DIGITAL ELECTRONICS

COVERS IT ALL!

Learn about logic circuits, flip-flops, registers, multivibrators, counters, 3-state bus drivers, bidirectional line drivers and receivers, analog-to-digital (ADC) and digital-to-analog (DAC) converters using easy-to-read, easy-to-understand explanations coupled with detailed illustrations that bring seeing and doing together for a very meaningful learning experience.

Basic Digital Electronics contains worked-out examples within the text and quizzes and problem sets at the end of each chapter to complete and reinforce the learning cycle as follows:

1. A Look at Familiar Systems Uses familiar analog and digital systems to explain the difference between the two. Explains the ASCII code; decimal, binary, HEX and BCD numbers; and conversions between them.

2. Digital System Functions Identifies basic functions that are required in digital systems. Functons such as, transmission, timing (synchronization), code conversion, data selection, decision circuits, storage, arithmetic circuits, interfacing and signal conversions are covered.

3. How Do Decision Circuits Work? Discusses the states of a logic gate and how TTL and MOS gates are used for AND, INVERT, NAND, OR, and XOR decision circuits. Concludes by showing the difference between positive and negative logic.

4. How Temporary Storage Circuits Work The discussion is about sequential circuits—clocked and gated latches; J-K flip-flops; parallel and shift registers; bistable, monostable and astable multivibrators; ripple and synchronous counters and counter modulo.

5. How to Couple, Convert, and Compute Details the digital circuits for interfacing—open collector and 3-state bus drivers, including bidirectional. Includes discussion of analog-to-digital (ADC) and digital-to-analog (DAC) converters, operational amplifiers and comparators, and circuits to add, subtract, multiply and divide.

6. More Permanent Storage Elements Discusses static and dynamic random-access memory (RAM), read-only memory (ROM), and the circuits needed for addressing, reading and writing data to and from storage. Timing and refreshing circuits are included, as well as the variety of ROMs.

7. A Tour Through ICs Explains the complete integrated circuit (IC) process, and the layout of a MOSFET (CMOS) circuit and a bipolar circuit.

8. Putting Functions Together Into a System Shows how the digital functions are formed into a computer system—first a hard-wired system and then a programmed system. Explains buses, I/O, memory contents, addressing and ALU operations. Shows how the CPU is the nerve center of the computer, and how the controller is the nerve center of the CPU.

9. Looking at Digital Communications Explains new terms like carrier, modulation, demodulation, bandwidth, and multiplexing which are used in digital communications. Gives examples of multiplexing, and the modulation used by modems.

ABOUT THE AUTHOR

Alvis J. Evans is an Associate Professor of Electronics at Tarrant County Junior College in Fort Worth, Texas. He has taught for over 30 years and is a recipient of the Chancellor's Award for Exemplary Teaching. He also teaches seminars nationwide and has authored many books on electrical and electronic technology for audiences ranging from hobbyists to advanced technicians.

Radio Shack®
A DIVISION OF TANDY CORPORATION
FT. WORTH, TEXAS 76102

192 PAGES
PRINTED IN THE U.S.A.

BASIC DIGITAL ELECTRONICS

Digital System Circuits and Functions

How They Work and How They Are Used

By
Alvis J. Evans

This book was developed and published by:
Master Publishing, Inc.
Chicago, Illinois

Edited by:
Gerald Luecke
Charles Battle

Printing by:
Custom Printing Company
Owensville, Missouri

Photograph Credit:
All photographs that do not have a source identification are either courtesy of Radio Shack, the author, or Master Publishing, Inc.

First Edition
9 8 7 6 5 4 3 2

Table of Contents

Preface

In *Basic Electronics,* we explained the basic concepts and fundamentals of electronic devices and circuits. In this book, *Basic Digital Electronics,* we explain the basic concepts and fundamentals of digital system functions, and the circuits used to perform those functions and build digital electronic systems. No segment of electronics has expanded more rapidly than digital electronics, led by the small size, low power, high reliability and low cost of integrated circuits, where all the circuitry is formed and interconnected on a single chip of silicon.

Basic Digital Electronics explains in detail how digital circuits are designed and applied in systems that execute specific tasks. In a straightforward, easy-to-read language, using detailed illustrations, practical worked-out application examples solidify the understanding of the concepts as the discussions proceed. Anyone who wants to know about digital systems, and how engineers design digital circuits should be able to learn from the explanations in *Basic Digital Electronics.*

Basic Digital Electronics begins by identifying familiar systems to help understand the difference between analog and digital systems, and identifies the common functions required when designing digital systems. Digital circuits, and how they are designed, covering logic circuits that make decisions, sequential circuits for temporary storage, a variety of circuits to couple, convert and compute digitally, and memory circuits for more permanent storage of digital information are discussed in the next four chapters.

The wide application of digital electronics to systems would not be possible without integrated circuits. A complete chapter is devoted to explain the basic techniques and manufacturing steps used to fabricate, assemble and test integrated circuits.

The last two chapters conclude the learning by showing how the circuits and functions are combined into systems—first for computers, and then for digital communications.

Multiple-choice quizzes and a set of questions and problems at the end of each chapter reinforce the learning. Answers to quizzes are on the quiz page; worked-out answers to questions and problems are given in the Appendix.

Studying the material, understanding the worked-out examples, working the quizzes and solving the problems should result in an understanding of the basic concepts and fundamentals of digital electronics. That was the goal of writing *Basic Digital Electronics; we* hope we have succeeded.

A.J.E.
M.P.I.

CHAPTER 1

A Look at Familiar Systems

Comparing Analog and Digital

The field of electronics can be classified into two groups: analog and digital. Analog quantities vary at a continuous rate, whereas digital quantities vary in discrete values. We can find many examples of each all around us. All things that can be measured quantitatively in nature are either analog or digital. Let's look at some examples of both analog and digital quantities.

The difference between analog and digital can be seen at the entrance to your local library. There are the steps, which is the digital route; and there is the ramp, the analog route. In a wheelchair, you better take the analog route. Now consider pitching pennies at the steps and the ramp. Where would the pennies land? As we see in *Figure 1-1,* the pennies can land only on steps 1, 2, 3, or 4, whereas they can land anywhere at all on the ramp. The steps represent digital — a discrete group of values; the ramp represents analog — a continuous group of values.

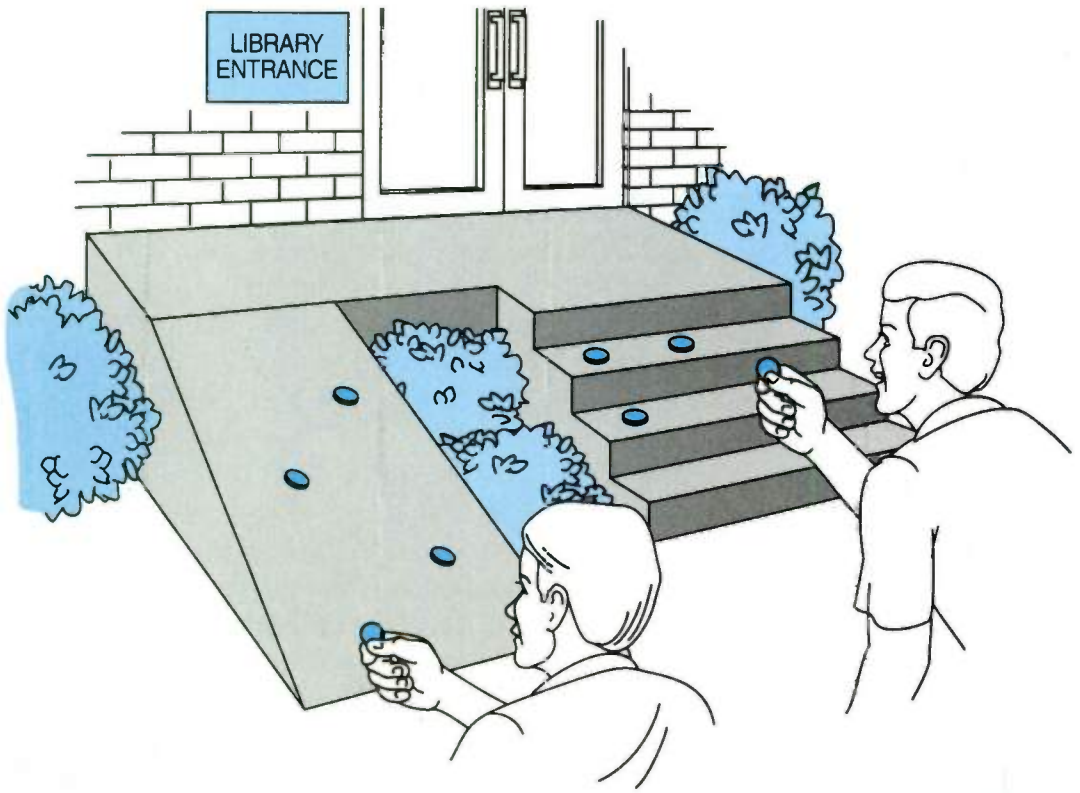


Figure 1-1. Pitching pennies at a ramp (analog) and steps (digital).

Another example of analog and digital quantities is shown in *Figure 1-2.* Here we have two buckets—one filled with water and the other filled with marbles. Suppose that we want to measure the contents of each bucket. The procedure would be different for each. For the bucket of water, we could dip the water out with a measuring cup or pour the water into the measuring cup. In either case we would probably leave a little in the bottom of the bucket or perhaps spill a little. Our measurement would not be very precise. With the bucket of marbles we could simply count them as we removed them. Our measurement would be much more precise. And, if we repeated the process on each over and over again, we would most likely get a different value for the quantity of water each time. But the number of marbles that we counted would be the same each time. This degree of precision is a characteristic difference between analog and digital. Digital is more precise. And how about if you were asked to reproduce each bucket at a distant location. Suppose that you were told one bucket had 6⅔ cups of water, and the other bucket had 723 marbles in it. Which bucket could you reproduce more accurately? The digital one, of course, because you can count out a discrete number of units.

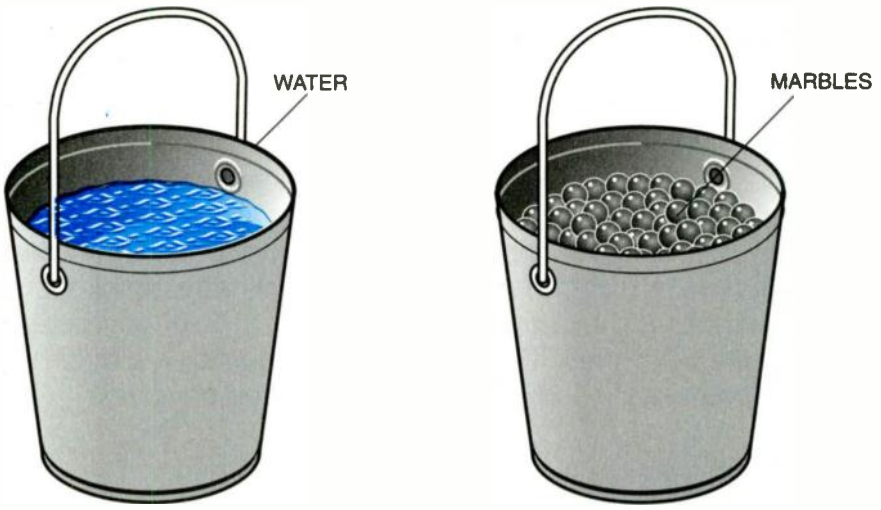


Figure 1-2. Two full buckets, one with water and one with marbles.

Familiar Examples of Analog and Digital Quantities

Most physical quantities that occur in nature are analog. Temperature can have an infinite number of values, even between 65° F and 66° F. Sound volume can vary continuously from very soft to extremely loud. The velocity and force with which a carpenter swings a hammer is analog in nature. If so many things in the world are analog, why do we use more and more digital to represent these things? The answer to this question is that electronic circuits can process information about these physical quantities easier and more accurately with digital than with analog.

Clocks

Many quantities are familiar in both analog and digital forms. *Figure 1-3a* shows examples of clocks. Modern mechanical clocks date from the late Middle Ages. Analog watches have served us well with improvements only in the basic design. For instance, by the end of the 15th century the spring had replaced the weight in some clocks, allowing them to be built small enough to be carried. Only within the last decade or two has the analog watch been almost completely replaced by the digital watch. (We wonder what the terms clockwise and counterclockwise will mean to the next generation.)

Thermometers

Figure 1-3b shows examples of thermometers. The analog thermometer has been used for temperature measurement since it was introduced around 1720 by Gabriel Daniel Fahrenheit (1686-1736). An example is a mercury thermometer, in which a rising or falling (expanding or contracting) column of mercury represents a rising or falling temperature. Digital thermometers have replaced these in most applications today, especially in the medical field.

Meters

Two basic types of meters are used to make electrical measurements. As shown in *Figure 1-3c,* the analog type has a needle that deflects along a scale and indicates the value of the quantity measured by the position of the needle on the scale. Measurements on a digital meter appear as a number on the display screen.

Audio Recordings

The fourth example of analog and digital is shown in *Figure 1-3d.* The human voice and musical instruments produce sound that is analog. The ear is also an analog device that responds to the sound signals. The first audio recording devices were analog. From about 1880 through 1980, the phonograph held the dominant position for audio recordings for the home. Around 1980, the audio cassette tape entered the scene. In the late 1980s came perhaps the most remarkable development in audio technology—the digitally recorded compact disc (CD).

Figure 1-3. Examples of the same quantities (time, temperature, electrical values, and sound) represented both as analog and digital values.

Recording devices for a CD convert analog signals to digital signals. The digital information is stored on the CD by making discrete indentations inline in a track on the surface of the CD. When the CD is played, a laser beam reflected from the surface converts these indentations into light pulses, which are detected and the digital data is converted back to analog signals by the CD player.

So why go to the trouble and extra steps to convert to and from the digital format? The CD digital audio vastly improves the audio quality over its analog counterparts—so much so that anyone who hears it will have a hard time going back to traditional analog audio. In addition, the digital format has virtually eliminated many problems of analog audio reproduction, such as noise and distortion due to fingerprints, smudges and scratches on the recording medium. Since a laser beam reads the information from the CD without mechanical contact with the CD, no matter how many times it is played, there is no wear on the disc.

Analyzing a Volume Control Circuit

Every radio and television receiver has a volume control, and most of them look like the circuit of *Figure 1-4a*. This is a partial block diagram of a radio receiver that shows the schematic of the volume control in more detail. The volume control is a continuously variable voltage divider; that is, it has an output voltage that changes as the control is varied. If the control is at the bottom (ground), there will be zero input voltage (V_{IN}) to the amplifier. If the control is at the top (maximum rotation), the input voltage to the amplifier is the full voltage, V_O. *Figure 1-4b* shows graphically the relationship between the input voltage and the amount the control is turned. Notice that the graph is a continuously varying line without jumps or breaks in it. The output is an analog of (or analogous to) the input voltage. The input voltage is a continuously proportional amount of the voltage V_O.

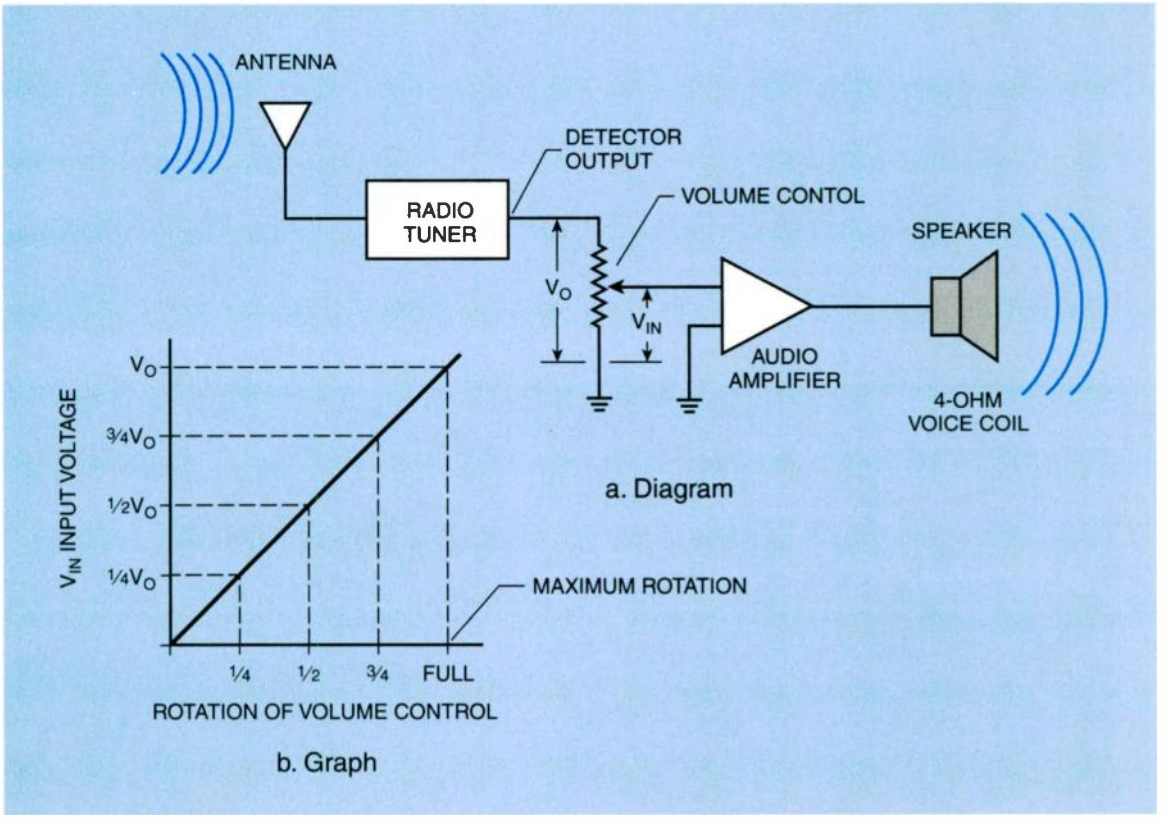


Figure 1-4. Partial block diagram of radio receiver with schematic showing volume control in more detail. Graph shows relationship between V_{IN} and volume control rotation.

Example 1. Power Delivered as Volume Control is Varied

If the amplification (or gain) of the audio amplifier in *Figure 1-4a* is 100 and the signal voltage output of the radio detector (V_O) is 0.1 volt, determine how much power is delivered to the speaker with the volume control set at full volume, one-half volume, and one-tenth volume.

Use $P = \frac{E^2}{R}$, where P is the power in watts, E is the EMF in volts, and R is the resistance in ohms.

Full Volume — At full volume, the full 0.1 volt of the detector is applied by the volume control to the input of the audio amplifier. The gain of the amplifier is 100, so the speaker receives 0.1 × 100 or 10 volts. The power is $P = 10^2/4 = 25$ watts when the volume control is full blast. The resistance of the speaker is 4 ohms.

Half Volume — With the volume control at one-half volume, the audio amplifier input is 0.1 × 0.5 or 0.05 volt. The amplifier 's gain of 100 delivers 0.05 × 100 = 5 volts to the speaker. This results in a power to the speaker of $P = 5^2/4 = 6.25$ watts. Notice that, though the voltage dropped by one-half, the power dropped by one-fourth.

One-tenth Volume — With the volume control at one-tenth of full volume, verify that the power to the speaker is 0.25 watt. The voltage is reduced by 1/10, but the power is reduced by 1/100!

The voltage varies in a linear manner and the power varies in a nonlinear manner, but both are analog. When we discuss integrated circuits (ICs) a little later, we will see them classified as linear and nonlinear. The term nonlinear in the case of ICs refers to digital. We must be careful in our use of the term nonlinear so it is clear what we mean.

What About a Race Track?

Is the speed control analog or digital for the miniature racetrack car shown in *Figure 1-5?* When the speed control slides from A to B, it causes the resistance of the rheostat to change linearly from 10 to 0 ohms. According to Ohm's law, the voltage and resistance of the circuit determine the current through the race car motor, which, in turn, controls the speed of the race car. The voltage is constant. The control resistance is at half-value when pressed half-way, and is at one-fourth value when pressed three-fourths way.

Example 2. Determining Current that Drives the Car Motor

What is the current in the circuit of *Figure 1-5b* if the speed control is set to one-half of the distance from A to B? The resistance of the car is 4 ohms and the resistance of the wire and track is negligibly small.

Use Ohm's law $I = \frac{E}{R}$, where I is the current in amperes, E is the EMF in volts, and R is the resistance in ohms.

$I = \frac{6}{(5 + 4)} = 0.667$ amperes or 667 milliamperes.

If the speed control is varied from A to B, the speed changes continuously as does the current. Therefore, this is an analog circuit.

The resistance varies linearly like the volume control of *Figure 1-4a;* however, there is a difference in the way the circuits operate. In *Figure 1-4a,* the control is in *parallel* with the detector output, and the rotating wiper selects different values of *voltage.* The *higher* the resistance value at the wiper, the louder the sound output. In *Figure 1-5,* the control is in *series* with the motor and power supply, and the rotating wiper varies *current* through the motor. The *lower* the resistance value at the wiper, the faster the race car goes.

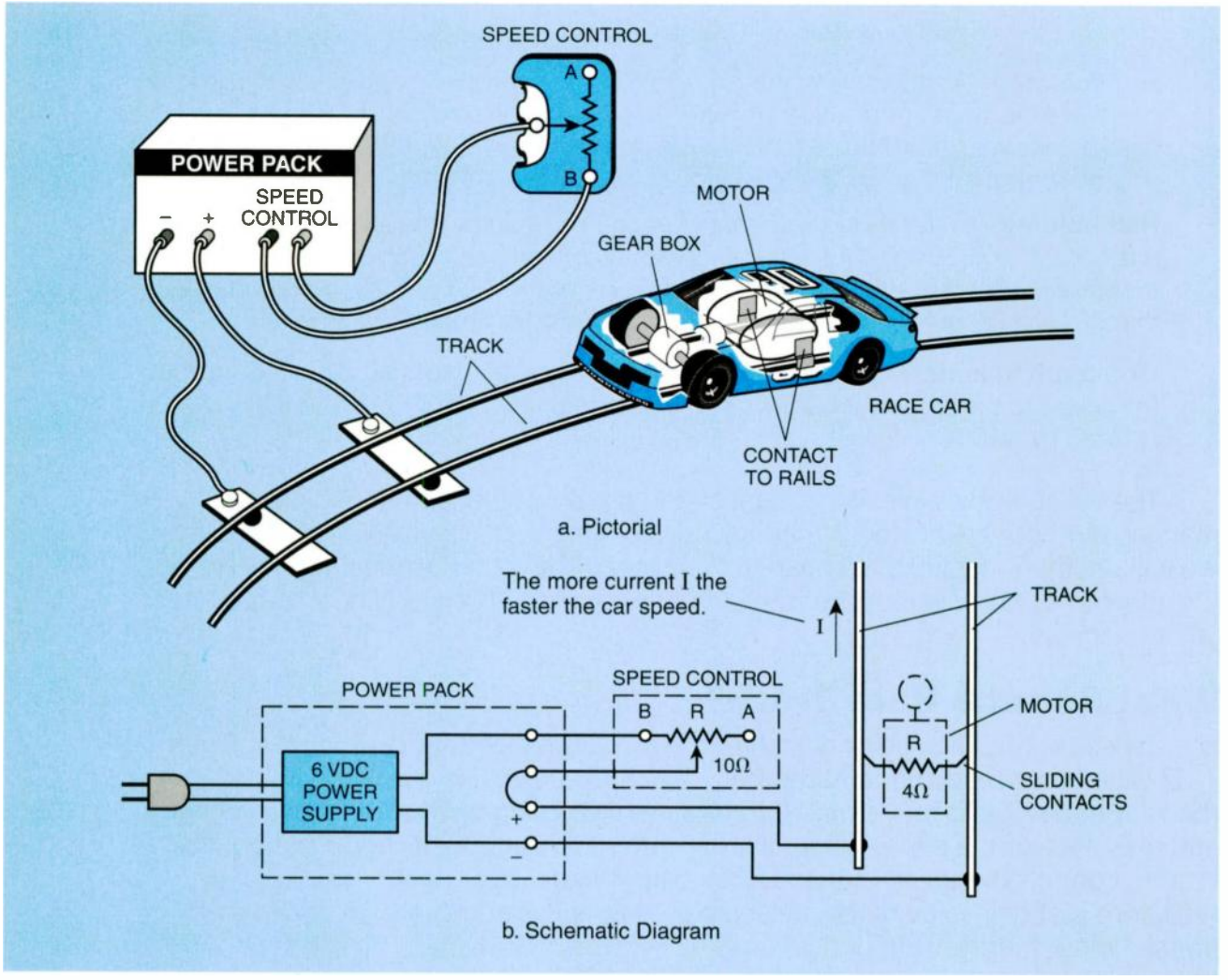


Figure 1-5. Speed controller for a miniature race track car.

A Digital Clock System

One digital system that is found in virtually every home is a digital clock. A simplified clock system that uses counters and frequency dividers is shown in *Figure 1-6.* These individual circuits will be discussed in detail in Chapter 4, but for now let's look at a overview of the clock system. The timing accuracy of the clock depends on the crystal oscillator, which does not vary with temperature, humidity, or power variations. Three counter blocks are shown in the block diagram: the first counts seconds, the next counts minutes, and the third counts hours. All counters output to the decoder which converts the count so it can be displayed. The maximum count of the first two counters is 59 (0 is a count, so there are 60 counts), the maximum count of the third is 12. The first and second counters start at zero, the third counter starts at one instead of zero so that it counts the hours from one o'clock to twelve o'clock. Each counter counts its input digital pulses and outputs a digital pulse to the next counter when it reaches its maximum count.

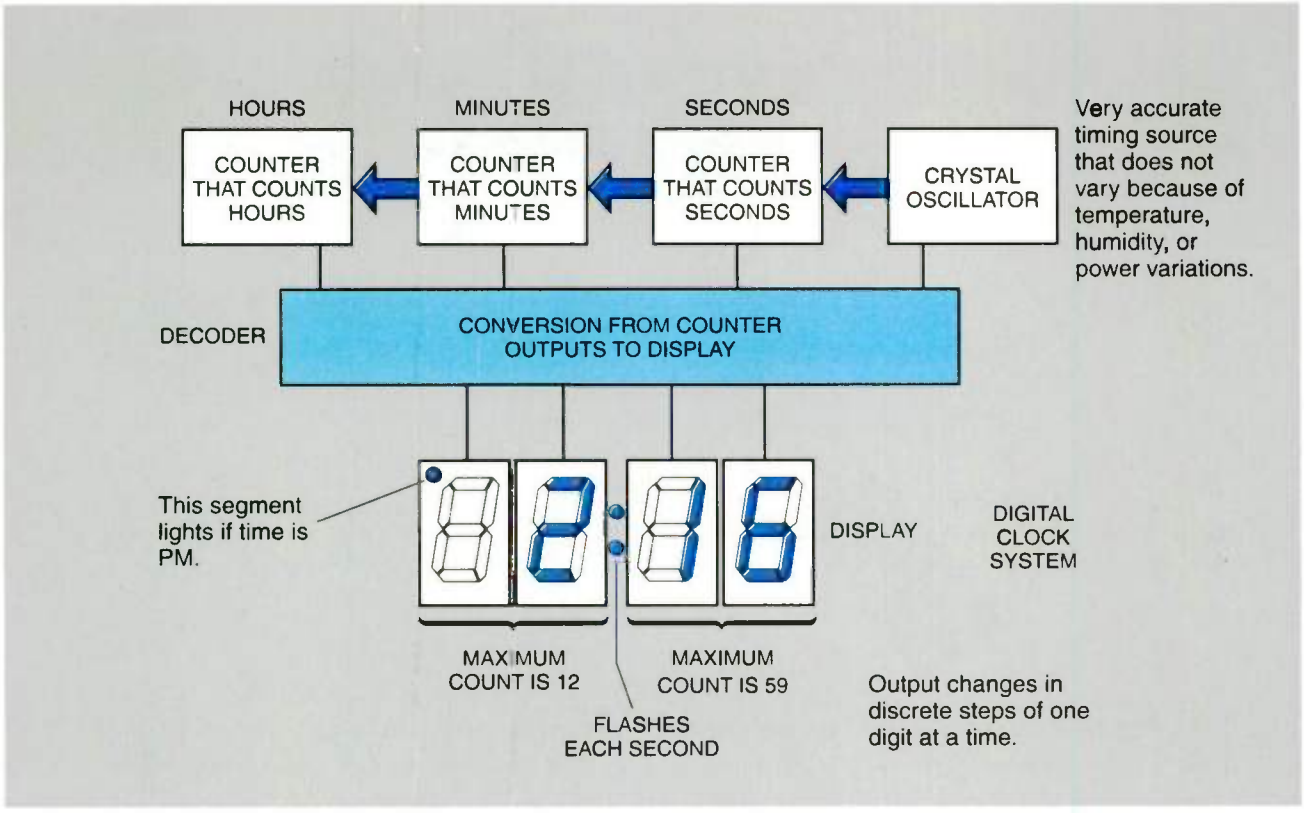


Figure 1-6. A digital clock system block diagram.

Example 3. Counting Digital Pulses

What time is it if the crystal oscillator in *Figure 1-6* has put out 37,200 one-second pulses. The clock is at 12:00 when the pulse counting starts, and with the first pulse the display shows 12:00:01.

Each hour contains 3600 one second counts, so 37,200 counts takes:

37,200 counts / 3600 counts / hr = 10.33333 hrs (10 hrs)
3600 × 0.33333 = 1200 counts
1200 counts / 60 counts / min = 20 min (20 min)

So the clock will show 10:20:00 after 37,200 pulse from the crystal oscillator.

We now have a good idea of analog (continuously varying) and digital (discrete parts) information. Since this book is about digital systems, from now on we will be dealing with digital information and digital systems. Let's look first at how information is represented digitally.

Codes

TV Remote Control

An example of using digital coding to represent information is as close as your TV remote control. A simplified version of the TV remote control system is shown in *Figure 1-7*. Each button on the keypad is assigned a particular code to cause a particular action in the TV. Pushing a button on the remote control generates a coded pattern that turns the infrared light transmitter on and off. The TV receiver detects the bursts of light and they are decoded into a required action, such as to turn on the power, change the channel, reduce or increase the volume, etc.

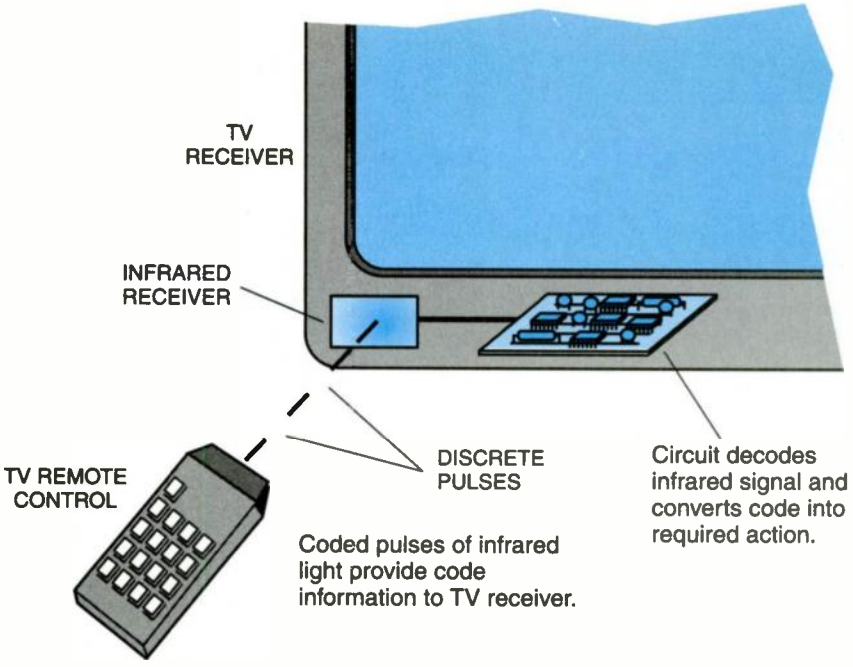


Figure 1-7. A simplified version of the TV remote control system.

Decoding Morse Code

Morse code is a signaling code first devised by Samuel F. B. Morse in 1838 for use with his electromagnetic telegraph. The code uses two basic symbols or signaling elements: the "dot," a short-duration electric current, which gave a quick deflection of the armature of Morse's receiver and so caused a dot to be printed on the strip of paper moving beneath the ink pen carried by the armature; and the "dash," a longer-duration signal that caused a dash to be printed. Using this code, the various alphanumeric characters needed to compose a message (letters and numerals) could be represented by groups of these two signal elements. The International Morse Code is shown in *Figure 1-8* along with a circuit that can be used to send a message by Morse code from one place to another. In the International Morse Code, for example, one dot followed by one dash (• —) symbolizes the letter A; the sequence "dash dash dot dot dot" (— — •••) symbolizes the numeral 7. When transmitting, the dot and dash elements are separated by a time interval that has the duration of one dot; the dash has a duration equal to three dots. The space between characters, whether letters or numbers, is equal to three dots; the separation between words is equal to six dots. Morse's code rapidly gained acceptance and its importance is not just historical; it illustrates the simplicity of a complete digital data communications system. A two-state communications system is the simplest, the easiest to build, and the most reliable. The two states can be On and Off, or Light and Dark with light beams, or with two durations of a tone, or, any other system with only two possible values. A two-state or two-valued system is referred to as a *binary system.*

Number Systems

What Do Numbers Mean?

Numbers are the primary language of all digital equipment. The information processed by digital devices and systems, including computers, is usually numerical in nature. As we will see, even the alphabet and other symbols can be expressed as numerical codes.

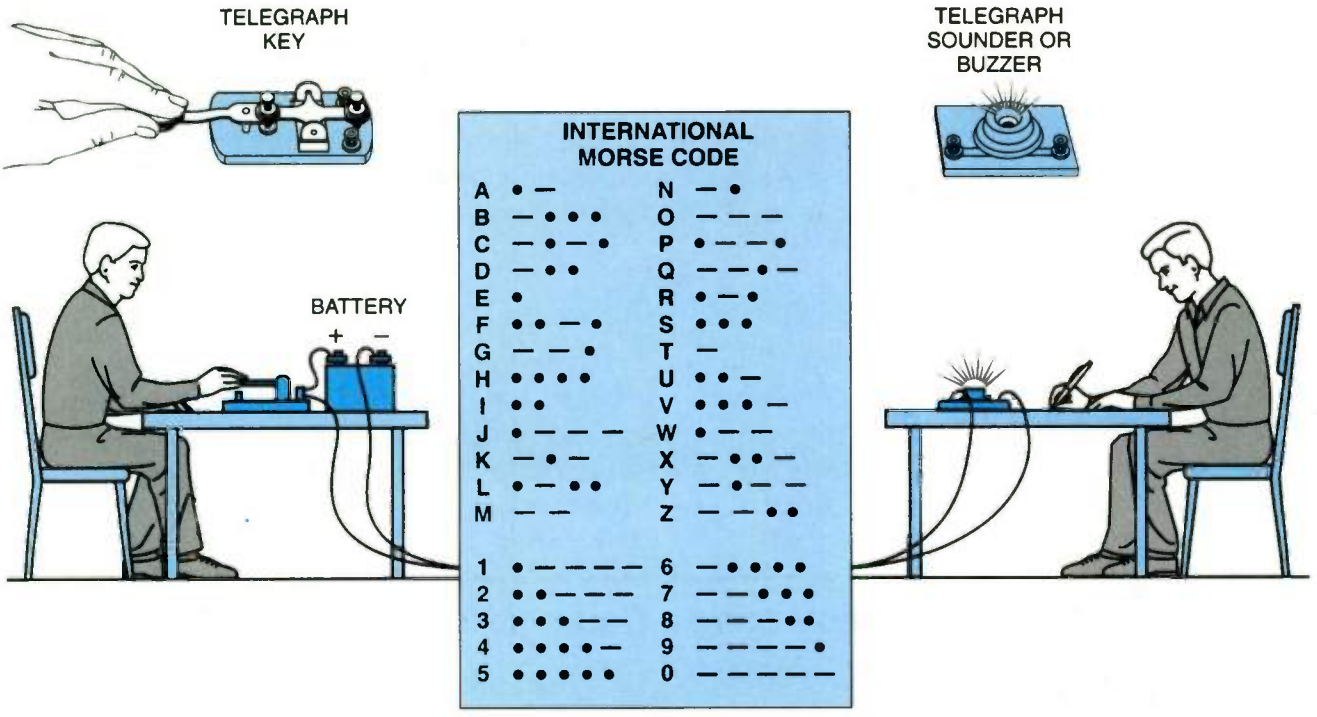


Figure 1-8. A circuit that can be used to send a message by Morse code.

Example 4. Morse Code Messages

Decode this message using the International Morse Code table in *Figure 1-8.* There are four words, one on each line.

• — — •••• • — —
•••• • — — ••••
— — • — — — — • •
• — — • — • — — — • • — — — • •••• —

Morse successfully demonstrated his magnetic telegraph by sending the above message in 1844. It said "What hath God wrought".

Now try this one. There are two words, one on each line.

• — • • — — — •• •• — — —
••• •••• • — — • — • — • — —

Solution: Write the decoded message in the space below. The correct solution is given at the end of this chapter.

We usually think of numbers as their symbols, but the numerical symbols themselves are quite versatile. Their meaning or value can vary according to the way they are utilized. It surprises some that there are other number systems than our "everyday" system that we use for phone numbers, our check book, or our street address. These numbers that we use everyday make up what is called the *decimal number system.* Although decimal numbers are common, the weight structure is not always understood. Let's review the characteristics of the place-value structure of number systems.

Decimal Numbers

The decimal number system, with which everyone is familiar, uses the symbols 0,1,2,3,4,5,6,7,8, and 9. Because it has 10 symbols, it is also referred to as a base-10 system or radix-10, which means the same thing. If we wish to express a quantity greater than nine, we use two or more digits, and the position of the digit within the number determines the magnitude it represents. Most number systems use a positional weighting scheme. The weights are powers of the base that increase from right to left. In the decimal system, the weight of each position is

$$\ldots 10^5, 10^4, 10^3, 10^2, 10^1, 10^0.$$

Remember that $10^0 = 1$, in fact, *any base raised to the zero power is equal to one.* 10^1 means the number itself; 10^2 means the number multiplied by itself twice (10×10); 10^3 means the number multiplied by itself three times ($10 \times 10 \times 10$); etc. The value of a number is then the sum of the digits after each digit has been multiplied by its digit position weight. The following example will illustrate the principle.

Example 5. Separating a Decimal Number by Its Weighted Digit Position

$328 = 3 \times 10^2 + 2 \times 10^1 + 8 \times 10^0 = [3 \times (10 \times 10)] + [2 \times (10)] + [8 \times 1]$
$328 = 300 + 20 + 8$
$328 = [3 \times 100] + [2 \times 10] + [8 \times 1]$

3rd digit weighted value	2nd digit weighted value	1st digit weighted value

Binary Numbers

Digital equipment uses a special number system to represent and process quantities. Because this system uses only two symbols or digits (0 and 1) to represent the quantities, it is called the binary number system. Because of the inherent bistable nature (having two states) of electronic circuits, binary numbers are more easily and quickly processed by electronic circuits than other numbers. Therefore, virtually all digital equipment uses binary numbers.

The binary system has two digits, 0 and 1, called **bi**nary digi**ts** or bits. It is a base-2 system, just as the decimal system with its ten digits is a base-10 system. To count in the binary system, we start with 0, count to 1, and we have run out of digits. So we can count only two values. We must go to the next digit position and continue. It too can only be 0 or 1. But it can be 0 while the first digit is 0 and 1, or it can be 1 while the 1st digit is 0 and 1. So we can count to four values with two digits. Adding a third digit position whose value is 0 or 1 allows a count to eight values. More and more digits provide more and more values. We illustrate the method in *Figure 1-9.* The decimal number is shown on the left with its binary equivalent in the center. The 4-bit binary number shown can represent 16 values. Two other number systems, *hexadecimal* and *binary coded decimal,* which we will see shortly, are on the right. Notice that the *least* significant bit (LSB) is in the 2^0 or 1's place. If a 1 appears in this place, a 1 is added to the binary count. The second place over from the right is the 2^1 or 2's place. A 1 appearing in this place means that 2 is added to the binary count. The weighted position value increases as digits are added from right to left. Just as in the decimal system, the value of a binary number is then the sum of the digits after each has been multiplied by its weight.

Decimal (XX_{10})	Binary ($XXXX_2$)	Hexidecimal (X_{16})	8421 BCD ($XXXX_{BCD}$)
0	0000 ← LSD	0	0000
1	0001	1	0001
2	0010	2	0010
3	0011	3	0011
4	0100	4	0100
5	0101	5	0101
6	0110	6	0110
7	0111	7	0111
8	1000	8	1000
9	1001	9	1001
10	1010	A	0001 0000
11	1011	B	0001 0001
12	1100	C	0001 0010
13	1101	D	0001 0011
14	1110	E	0001 0100
15	1111	F	0001 0101

Figure 1-9. Counting in the binary system with binary, hexadecimal and BCD.

Figure 1-10 shows a comparison between the decimal system and the binary system. The digit position values are 10 times the position value to the right in the decimal system (base-10), while the digit position values are two times the position value to the right in the binary system. Note that in both systems, the value of the number is obtained by adding together all the digit values after the digit has been multiplied by its weighted position value. Note also that a decimal number could be identified with a 10 subscript, while a binary number has a 2 subscript and a hexadecimal number has a 16 subscript. We normally do not write the 10 subscript with decimal numbers because they are so common that it is understood.

The following example will illustrate the principle of finding the decimal value of a binary number.

Example 6. Finding the Decimal Value of a Binary Number

What is the decimal equivalent of the binary number 101110_2?

Using the same procedure as for the decimal number in *Example 1-5,* each binary bit in the binary number produces a decimal equivalent for that binary place value. The *most significant bit* (MSB) of the binary number has a 1 or $1 \times 2^5 = 32$ is added to the value. The next significant bit has a 0 in its place, so it adds $0 \times 2^4 = 0$ to the value. The next three binary places have 1's, so they add $1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 = 8 + 4 + 2$. The *least significant bit* (LSB) is a 0 so it adds 0×2^0 to the value. The decimal value is, therefore, $32 + 0 + 8 + 4 + 2 + 0 = 46$.

It is also important to know how to convert from a decimal number to the equivalent binary number. A method of converting from decimal to binary is to divide repeatedly the decimal number by 2. The remainder generated by each division produces the binary number with the first remainder being the LSB. We illustrate in Example 7.

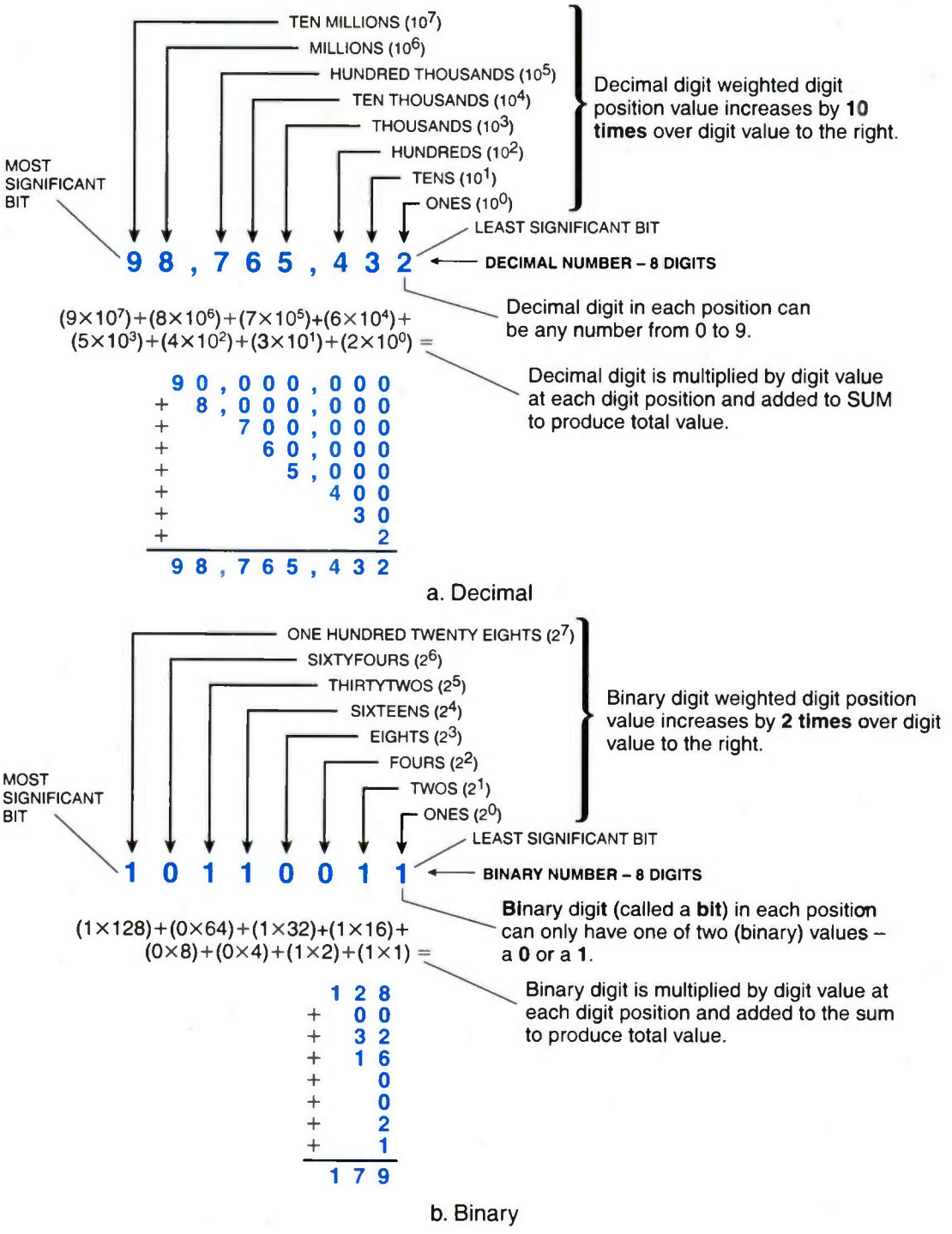


Figure 1-10. Decimal and binary numbering systems.

Example 7. Converting Decimal to Binary

Convert 87_{10} to its binary equivalent.

$87_{10} / 2 = 43$ remainder of $\mathbf{1}^{LSB}$
$43 / 2 = 21$ remainder of **1**
$21 / 2 = 10$ remainder of **1**
$10 / 2 = 5$ remainder of **0**
$5 / 2 = 2$ remainder of **1**
$2 / 2 = 1$ remainder of **0**
$1 / 2 = 0$ remainder of $\mathbf{1}^{MSB}$
$\mathbf{87_{10} = 1010111_2}$

Hexadecimal Numbers

Hexadecimal (called hex for short) is a popular number system used in digital electronics, especially in microcomputers. Each hex number is formed by grouping the binary bits in fours, starting from the right. This results in combinations from 0000_2 to 1111_2, or sixteen possible combinations. The first ten are represented by their decimal equivalents and the remaining six by the first six letters of the English alphabet; A, B, C, D, E, and F. Here is an example of a hex number converted to binary and decimal and vice versa.

Example 8. Converting Hex to Decimal and Decimal to Hex

Convert the hex number $3C_{16}$ to its binary and decimal equivalent. Use *Figures 1-9* and *1-10* to help you.

$3C_{16} = 0011\ \ 1100 = 00111100$
$3C_{16} = 111100 = 32 + 16 + 8 + 4 + 0 + 0 = 60$

The hex number may also be evaluated as a base-16 number:

$3C_{16} = 3 \times 16^1 + 12 \times 16^0 = 48 + 12 = 60$

To convert from binary to hex, reverse the process:

$11010101_2 = 1101\ \ 0101 = D5_{16}$

To convert from decimal to hex, apply the divide-by-16 method to convert 650 to its hex equivalent:

$650_{10} / 16 = 40$ with a remainder of $10^{LSD} = A_{16}$
$40 / 16 = 2$ with a remainder of $8 = 8$
$2 / 16 = 0$ with a remainder of $2^{MSD} = 2$
$650_{10} = 28A_{16}$

Its binary equivalent is:

$650_{10} = 0010\ \ 1000\ \ 1010$

Binary-Coded Decimal

The binary-coded decimal (BCD) code provides a way for decimal numbers to be encoded in a type of binary that is easily converted back to decimal. Each digit in the decimal number is assigned its equivalent binary code. Since digits from 0-9 need to be represented, a 4-bit code is required. This code is a weighted code more precisely known as 8421 BCD code. The 8421 part of the name gives the weight of each place in the 4-bit code. There are several other BCD codes that have other weights for the four places. The 8421 BCD, however, is the most popular so it is customary to refer to *it* as simply the BCD code. This code, shown in *Figure 1-9,* is useful for providing output to displays that are always numeric (0 to 9), such as those in digital multimeters, thermometers, and clocks. Converting decimal numbers to BCD and vise versa is quite simple as we shall see in the next example. To form a BCD number, simply convert each decimal digit to its 4-bit binary code. To convert BCD to decimal, just reverse the process. Note that the 4-bit code represents only numbers from 0 to 9, not the full possible sixteen numbers. It, therefore, is not as efficient as pure binary, but can save circuitry in certain applications.

Example 9. Converting Decimal to BCD and BCD to Decimal

Convert 369_{10} to BCD

$$\begin{array}{rccc} & 3 & 6 & 9 \\ 369_{10} = & 0011 & 0110 & 1001_{BCD} \end{array}$$

Convert $0101\ 1000\ 0111_{BCD}$ to decimal

$$\begin{array}{ccccc} & 0101 & 1000 & 0111 & \\ & 5 & 8 & 7 & = 587_{10} \end{array}$$

Communications Codes — Bits, Nibbles, and Bytes

Communications transmission codes are made up of binary digits. (Recall that "bits" is the abbreviated form for "binary digits", and that each bit can have one of two possible states (high and low, on or off, 0 or 1). Four-bits combined in a uniform group is called a *nibble,* and an 8-bit group is called a *byte.* Many modern computers operate with a 16-bit or 32-bit word size, so each word comprises two bytes or four bytes, respectively.

A code is a previously agreed upon set of meanings that defines a given set of symbols and characters. Digital systems process only codes consisting of 0's and 1's (binary codes). You have learned that coded patterns can represent operations on the TV remote control, and that by a telegraph system you can send a message of characters (letters, numbers or symbols). It was virtually impossible to automate the telegraph because of the varying number of dots and dashes for the various characters. What was needed was a code that had an equal number of equal duration signaling elements for each character. A standardized alphanumeric code called the American Standard Code for Information Interchange (ASCII) is the most widely used in industry.

ASCII

Figure 1-11 shows the ASCII characters and their associated codes. Compare this chart to the Morse code in *Figure 1-8.* Not only does the Morse code have a varying number of elements (dots and dashes) for each character, it is also restricted to the letters, numbers, and a few punctuation marks. ASCII has not only both upper and lower case for letters but also has a number of other features that make it very versatile. For example, the ASCII format is arranged so that lower case letters can be changed to upper case by changing only one bit of the seven. By changing bit six from a 0 to a 1, an upper case letter is changed into lower case. Also, bits 4 through 1 of the numeric characters 0-9 are the binary-coded-decimal (BCD) value of the number. ASCII is the most commonly used code for the input and output of computers. The keystrokes on the keyboard of a computer are converted directly into ASCII for processing by the digital circuits. In Example 10, we determine the codes that are entered into a computer's memory when a line of program is typed on its keyboard.

Signals and Switches

We pointed out in the beginning of this chapter that, while many quantities are inherently analog in nature, digital quantities also exist. In fact, virtually any analog quantity can also be represented digitally. Analog values are frequently converted to a digital value for more convenient processing and display purposes. An example is the gasoline pump where the smooth analog flow of gasoline is measured and the volume is displayed digitally to the nearest tenth of a gallon or liter.

Bit Position															
							0	1	0	1	1	0	0	1	
							0	0	1	1	1	1	0	0	
1	2	3	4	5	6	7	1	1	1	1	0	0	0	0	
0	0	0	0				@	P	‘	p	0	sp	NUL	DLE	
1	0	0	0				A	Q	a	q	1	!	SOH	DC1	
0	1	0	0				B	R	b	r	2	"	STX	DC2	
1	1	0	0				C	S	c	s	3	#	ETX	DC3	
0	0	1	0				D	T	d	t	4	$	EOT	DC4	
1	0	1	0				E	U	e	u	5	%	ENQ	NAK	
0	1	1	0				F	V	f	v	6	&	ACK	SYN	
1	1	1	0				G	W	g	w	7	’	BEL	ETB	
0	0	0	1				H	X	h	x	8	(	BS	CAN	
1	0	0	1				I	Y	i	y	9	)	HT	EM	
0	1	0	1				J	Z	j	z	:	*	LF	SUB	
1	1	0	1				K	[	k	{	;	+	VT	ESC	
0	0	1	1				L	\	l	|	<	,	FF	FS
1	0	1	1				M	]	m	}	=	-	CR	GS	
0	1	1	1				N	^	n	~	>	.	SO	RS	
1	1	1	1				O	_	o	DEL	?	/	SI	US	

Figure 1-11. ASCII characters and their associated codes.

Example 10. How ASCII Codes Represent Information

Encode the following line of program into ASCII using *Figure 1-11*. Remember bit position 1 is the LSB.

30 PRINT "C=";y

The solution is found by looking up each character in *Figure 1-11*.

Character	ASCII	Hexadecimal
3	0110011	33_{16}
0	0110000	30_{16}
Space	0100000	20_{16}
P	1010000	50_{16}
R	1010010	52_{16}
I	1001001	49_{16}
N	1001110	$4E_{16}$
T	1010100	54_{16}
Space	0100000	20_{16}
"	0100010	22_{16}
C	1000011	43_{16}
=	0111101	$3D_{16}$
"	0100010	22_{16}
;	0111011	$3B_{16}$
y	1111001	79_{16}

Analog and digital signals are used to represent all types of data, including physical data. Data is any kind of fundamental element of information, such as quantities, numbers values, letters, words, and symbols capable of being processed. We have considered several examples of how analog and digital signals represent data in this chapter. Both analog and digital signals can represent time, temperature, sound recording. We have seen a multimeter that can display electrical current, voltage, and resistance. However, we have not discussed in any detail how these two states even allow us to build any useful electronic circuits to do even simple processing or problem solving. So let's examine how we can use simple switches to represent numbers and provide the basis for digital electronics and even a digital computer.

Digital States Against Time — Timing Diagrams

What modern convenience comes to mind when you hear the terms "on" and "off"? A light switch, maybe? This simple example of a common digital circuit is illustrated in *Figure 1-12*. It has two states, no in-between, just light on and light off. When the switch is open, the light is off. When the switch is closed, the light is on. In digital circuits, the two voltage levels, the two current levels, or the two light conditions represent the two binary digits, 0 and 1. *Figure 1-12d* shows the current in the circuit or the voltage across the lamp plotted against time. It is called a timing diagram. Timing diagrams are used to show the "high" and "low" levels of a digital signal as it changes relative to time. The vertical axis of the plot displays the voltage or current level, and the horizontal axis displays the time. We will discuss the importance of timing in digital circuits in the next chapter.

Light-Emitting Diode Digital Circuit

Any type of lamp may be used to represent digital lows and highs (0 and 1), but a light-emitting diode (LED) is used extensively in digital circuits to display the "on " and "off" conditions of the digital signal. A typical LED and its circuit diagram symbol are shown in *Figure 1-13*. The LED is connected to a single-pole single-throw switch, a current limiting resistor, and a 5-volt power supply. When the switch is open, there is no current and the LED is "OFF." When the switch is closed, there is current and the LED is "ON."

The advantage of the LED over a incandescent lamp is its low power consumption. An LED can produce a visible amount of light with only a few milliamperes of current, so it produces very little heat. Another advantage is its very long life compared to a lamp.

The most common type of LED is the one that produces red light, though other colors are available. The LEDs actually produce different light colors, not just different color lens caps as with incandescent lamps. Unlike the incandescent lamps, the LED is a directional device; that is, it has to be connected in the circuit with a forward bias to produce light. When it is forward-biased by approximately two volts, it starts conducting current and produces light. Forward-bias is when the anode is positive in voltage relative to the cathode. When the LED is reversed-biased (negative voltage on its anode), it has zero current and gives off no light.

Groups of LEDs for Displaying Binary Numbers

Groups of four LEDs can be used to represent binary numbers that can be read as hexadecimal numbers. The conditions are: when an LED is ON, it represents a HIGH or 1 condition; when it is off, it represents a LOW or 0. The LEDs in *Figure 1-14* represent two hex numbers that can be changed by their corresponding switches. Example 11 shows how the switches can be used to represent (encode) hex numbers or ASCII characters.

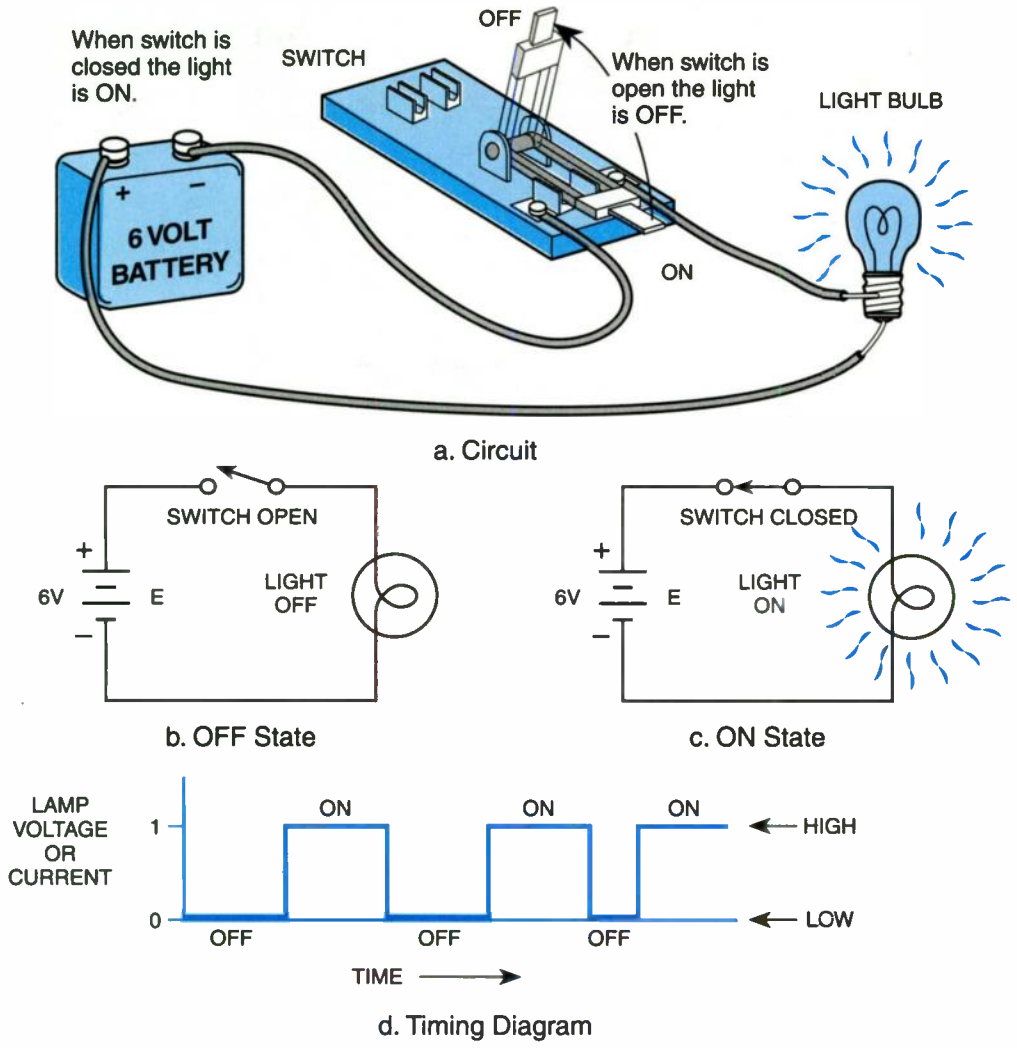


Figure 1-12. A light switch example of a digital circuit with its waveforms and timing diagrams.

Example 11. How LEDs Represent Binary Numbers

A: What are the hexadecimal numbers represented by the LEDs if the switches in *Figure 1-14* are in the following states? Use *Figure 1-9* to help you.

MSB ON-OFF-ON-ON OFF-ON-OFF-OFF LSB

1 0 1 1 0 1 0 0

B 4

B: What ASCII character is represented by the seven bits not counting the MSB if the switches are in the following states? Use the ASCII table in *Figure 1-11* to help you. Remember bit 7 is the one next to the MSB, and bit 1 is the LSB.

MSB ON-ON-OFF-OFF OFF-ON-ON-OFF LSB

1 1 0 0 0 1 1 0

F

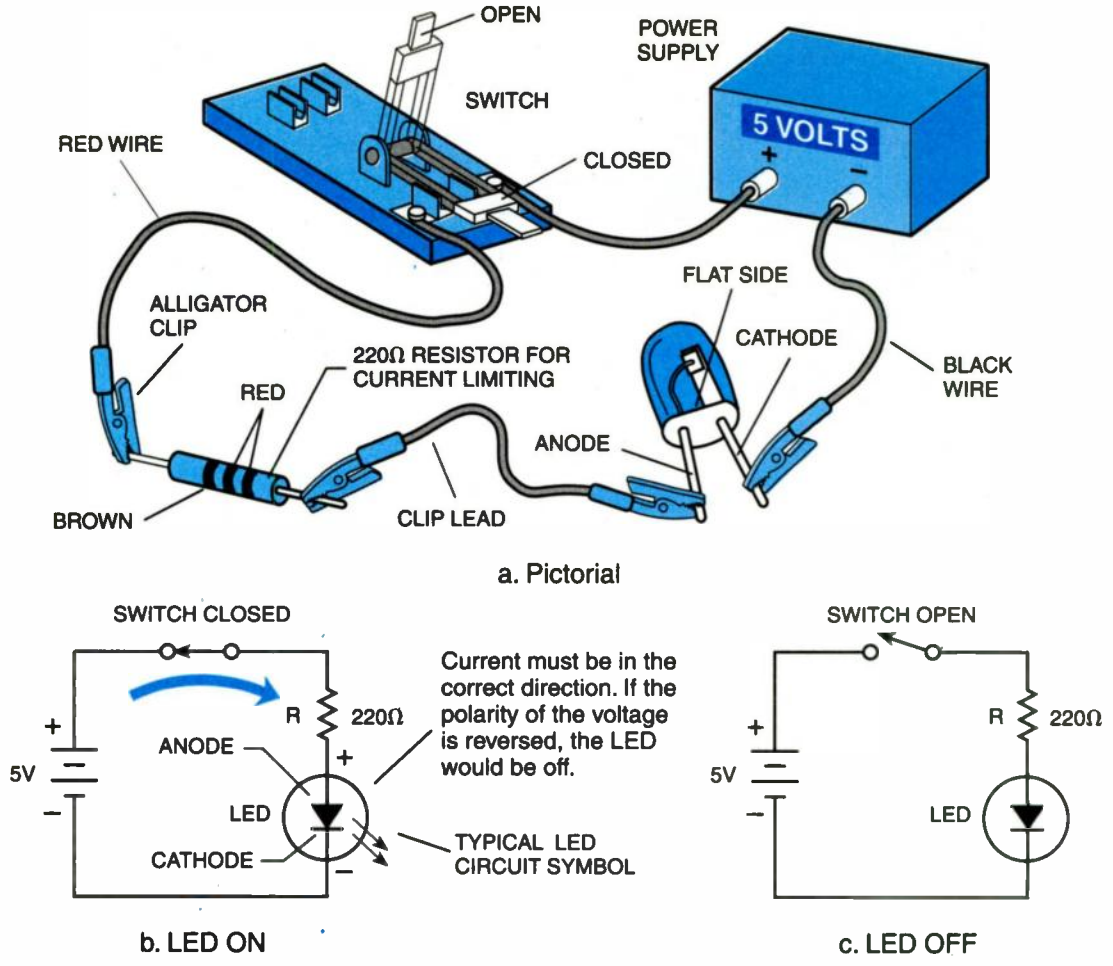


Figure 1-13. An LED in a typical digital circuit.

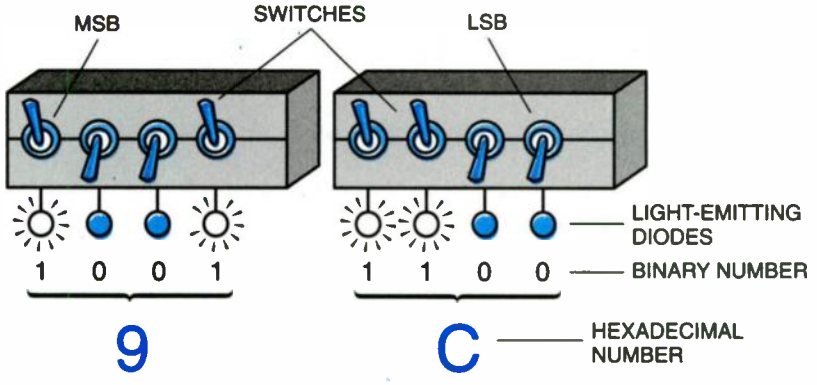


Figure 1-14. Switches control LEDs to represent hex numbers in binary code.

Digital Integrated Circuits

Most of the circuits in digital equipment are in the form of integrated circuits (ICs). This means that all of the circuit is fabricated on a single silicon chip; that is, all of the components, such as transistors, resistors and diodes are fully formed and interconnected on a single silicon chip. The components are very tiny, so many digital circuits can be formed on a very small chip. For example, the microprocessors in personal computers have digital integrated circuits with a million or more transistors on a silicon chip as small as a fingernail. The primary advantages of digital integrated circuits are small size, low cost, and outstanding reliability. We will investigate integrated circuits in more detail in Chapter 7. Now, in the next chapter, let's look at the important functions that are required in digital electronic systems.

Quiz for Chapter 1

1. Quantities that vary at a continuous rate are called
   a. linear
   b. digital
   c. analog
   d. logical
2. Electronic circuits can process information about physical quantities easier and more accurately with
   a. linear circuits
   b. digital circuits
   c. analog circuits
   d. logic circuits
3. An example of a digital audio source is
   a. AM radio
   b. compact disc
   c. phonograph
   d. standard cassette tape
4. If the amplification of the audio amplifier is 50 and the signal voltage output of the radio detector is 0.2 volt, at one-fourth volume control settings, the power delivered to an 8-ohm speaker is
   a. 1.25 watts
   b. 12.5 watts
   c. 0.78 watts
   d. 2.5 watts
5. For the clock in *Figure 1-6;* What time is it if the crystal oscillator has put out 52,200 one second pulses? The clock starts at 12:00.
   a. 10:20 AM
   b. 2:30 PM
   c. 4: 45 PM
   d. 10:20 PM
6. A TV remote control is an example of
   a. digital coding
   b. analog coding
   c. hexadecimal numbers
   d. Morse code
7. A two-state or two-valued system is referred to as a
   a. analog system
   b. digital system
   c. BCD system
   d. binary system
8. Decode this message from the International Morse Code:
   —•• •• ——• •• — •— •—••
   a. DECIMAL
   b. DECODE
   c. DECIBEL
   d. DIGITAL
9. The value of a number is the sum of the digit values after each digit has been multiplied by its
   a. weight
   b. power
   c. radix
   d. base
10. Convert 63_{10} to its binary equivalent.
    a. 0000111111
    b. 0001010101
    c. 0010110110
    d. 0011100011
11. What is the decimal equivalent of the binary number 101010_2?
    a. 24
    b. 40
    c. 42
    d. 124
12. Convert the hex number $B5_{16}$ to its binary equivalent.
    a. 1100 0011
    b. 1000 0111
    c. 0101 1011
    d. 1011 0101
13. Convert $0011\ 1001\ 0111_{BCD}$ to decimal.
    a. 468_{10}
    b. 284_{10}
    c. 953_{10}
    d. 397_{10}
14. Decode the ASCII characters below using the table of *Figure 1-11.*
    1010010 1010101 1001110 0100001
    a. RUN!
    b. JUMP
    c. WHO?
    d. RAMP
15. When positive polarity is applied to an LED's anode compared to the cathode, it is
    a. "OFF"
    b. Reversed-biased
    c. Forward-biased
    d. A high impedance

Answers:
1c, 2b, 3b, 4c, 5b, 6a, 7d, 8d, 9a, 10a, 11c, 12d, 13d, 14a, 15c
Example 4 message is RADIO SHACK

Questions and Problems for Chapter 1

1. For the circuit in *Figure 1-4a,* if the amplifier's gain is 50, the speaker is 8 ohms, and the detector's output is 50 mV, what is the power delivered to the speaker with the volume control set at full, one-half, and one-tenth volume?

2. a. For the circuit in *Figure 1-5b,* what is the current in the motor if the speed control is set to one-fourth the distance from A to B?

   b. Is this faster or slower than with the control set to one-half?

3. For the digital clock in *Figure 1-6,* if it starts at 12:00 noon, what time will it be after the crystal oscillator has put out 25,400 one-second pulses?

4. For the digital clock in *Figure 1-6,* if it starts at 12:00 noon, what time will it be after the crystal oscillator has put out:

   a. 3600 one-second pulses?

   b. 46,800 one-second pulses?

5. a. Is a binary system digital?

   b. Is a digital system binary?

6. Separate the digits of the decimal number 3572 into their weighted digit positions.

7. What is the decimal equivalent of the binary number 100101_2?

8. What is the binary equivalent of the decimal number 63?

9. Convert the hex number $5C_{16}$ to its: a. binary equivalent b. decimal equivalent?

10. Convert the decimal number 47 to its: a. hex equivalent b. binary equivalent?

11. Encode the following line of code into ASCII using *Figure 1-11:*

    DIGITAL ELECTRONICS

12. Decode the following ASCII message from its hex code:

    41, 4E, 41, 4C, 4F, 47, 20, 56, 4F, 4C, 54, 41, 47, 45

CHAPTER 2
Digital System Functions

Looking Back

In Chapter 1, the examples of common analog and digital systems showed the difference between the two. Recall that analog system values vary continuously, but digital system values are comprised of discrete parts formed into codes to represent the values. The digital system uses these codes that change with time to carry information from one point to another through the system. The codes are processed to accomplish particular functions that combine to make up the complete digital system.

This chapter provides an overview of basic functions that make up digital systems. This overview will help you understand how a digital system works and accomplishes tasks. Only basic concepts will be covered here; more details will be presented in later chapters.

Returning to Codes

Chapter 1 showed different types of digital codes, including Morse code and ASCII, for representing alphabetic characters, numbers, punctuation marks, special characters, and commands. Each of the codes uses *two* values (*two* levels, ON-OFF, 0 and 1) for each of the bits (bit is a contraction of **bi**nary digi**t**) in the *binary* code. Recall that the bit is the smallest unit of binary information, and since it is binary, it can have only *two* values. How then do we represent 10 different values?

Figure 2-1 will help explain. A 2-bit code is required to represent four values, a 3-bit code for eight values, and a 4-bit code for 10 values, and so on. But note that a 4-bit code can represent not just 10 values, but 16 values. To represent more and more values (or different things), we just add more bits to the code. For a base-2 (binary) system, the number of bits n required to distinguish K different things is given by

$$2^n = K$$

Note that, even though the codes are binary numbers, they can represent whatever the system designer designates them to be; e.g., the control system commands shown. In ASCII, a different binary code (binary number) is used for each of the characters.

If the digital code can represent more values than required, the coding efficiency (in percent) is:

$$\text{Percent Coding Efficiency} = \frac{\text{Values required}}{2^N} \times 100$$

Two Values	
0	First value
1	Second value

a. 1-Bit Code

Four Values	
0 0	First value
0 1	Second value
1 0	Third value
1 1	Fourth value

b. 2-Bit Code

Eight Values	
0 0 0	First value
0 0 1	Second value
0 1 0	Third value
0 1 1	Fourth value
1 0 0	Fifth value
1 0 1	Sixth value
1 1 0	Seventh value
1 1 1	Eighth value

c. 3-Bit Code

Ten Values		Control System Commands
0 0 0 0	First value	0. System Idle
0 0 0 1	Second value	1. Master Sw ON
0 0 1 0	Third value	2. Close Output Chute
0 0 1 1	Fourth value	3. Open Input Chute
0 1 0 0	Fifth value	4. Weigh Input Material
0 1 0 1	Sixth value	5. Close Input Chute
0 1 1 0	Seventh value	6. Start Conveyor Belt
0 1 1 1	Eighth value	7. Open Output Chute
1 0 0 0	Ninth value	8. Stop Conveyor Belt
1 0 0 1	Tenth value	9. Master Sw OFF
1 0 1 0	Eleventh value	Available for Additional Commands
1 0 1 1	Twelfth value	
1 1 0 0	Thirteenth value	
1 1 0 1	Fourteenth value	
1 1 1 0	Fifteenth value	
1 1 1 1	Sixteenth value	

d. 4-Bit Code

Figure 2-1. Bits are added to binary codes to represent more and more binary numbers, values, or commands.

Example 1. Determining Number of Bits to Represent Objects

How many bits are required to represent 22 different objects?

We can get the exact answer with logarithms, but it may be simpler by trying powers of 2. If we try $2^3 = 8$ and $2^4 = 16$, neither is enough. Trying $2^5 = 32$ is more than enough, but since a fraction of a bit cannot be implemented in practice, we must use the next highest whole number; that is, 5 bits. So, for a base-2 system, the number of bits n is 5 for $K = 22$ objects. Since $2^5 = 32$, there are 10 values included in the 5-bit code that are not used in this application. (Remember that the 32 values are 0 through 31, inclusive. The values 0 through 21 are used, so the unused values are 22 through 31.)

Transmission of Digital Information

Every digital system requires the movement of digital information from one point in the system to another. It may be called code transfer, binary bus transfer, digital communications, or data communications. Whatever it's called, it means that a binary code representing information is moved from one place to another. For want of a better term, we will use *data communications* because information termed "data" refers to numerical, alphabetical, or special-purpose characters which

represent some message or intelligence. Regardless of the origin of the data, a binary digital signal — a code — consists of a sequence of bits, appropriately grouped.

Movement of data requires a *data communications system*. The three components that make up a data communications system are: the source or transmitter, the channel or transmission path, and the receiver. Electrically, in a digital system, the bit value may be represented by a voltage or current value.

Data are commonly transferred between the transmitter and the receiver, as shown in *Figure 2-2,* by changes in current or voltage on the channel or transmission line(s) between the units. For example, to transmit an ASCII character, a "1" or MARK could be represented by a positive voltage or current pulse; a "0" or SPACE could be represented by a zero voltage or no current condition. The data — a code of 0100101, one digit at a time in sequence — is being transmitted from one point to another in *Figure 2-2.* Notice that a START bit is included just before the character code and a STOP bit is included immediately after the character code. The purpose of these bits is probably obvious, but they will be explained later.

In digital systems, two methods — parallel and serial — are used to move binary information from transmitter to receiver. Let's look at these two types of data transmission and the characteristics and advantages of each.

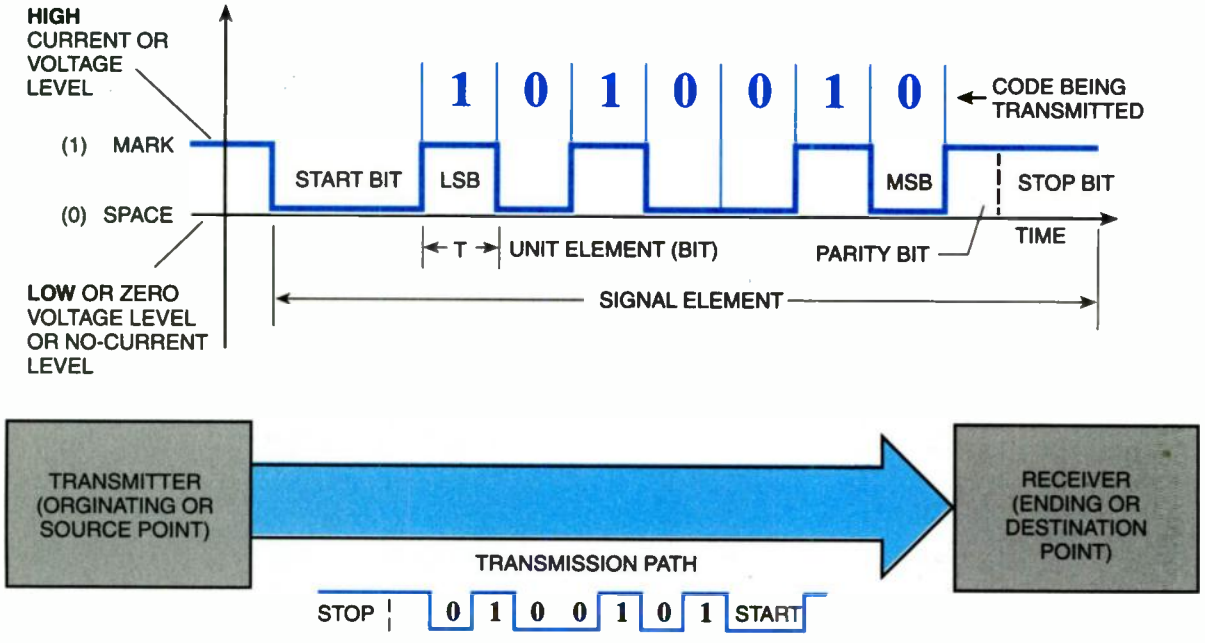


Figure 2-2. Data Communications System — Data transmitted from one point to another by voltage or current pulses.

Parallel Transmission of Data

Parallel transmission is often used for short distance communications between digital equipment. Parallel transmission moves the data characters using multiple lines — one for each bit. The 1 or 0 on each data line representing a bit of the digital code appears on all of the multiple lines at the same time. In *Figure 2-3,* we are transmitting one bit of a 7-bit ASCII character on each line of the seven lines. Each bit of a character travels on its own line. As shown in *Figure 2-3,* each of the 7-bit ASCII characters appears in sequence as time proceeds. Thus, the transmission is also known as *parallel-by-bit, serial-by-character* transmission. It is often used over short distances, such as from a computer to a printer, or from one circuit to another inside a piece of digital equipment. Parallel transmission is much faster, so it is employed

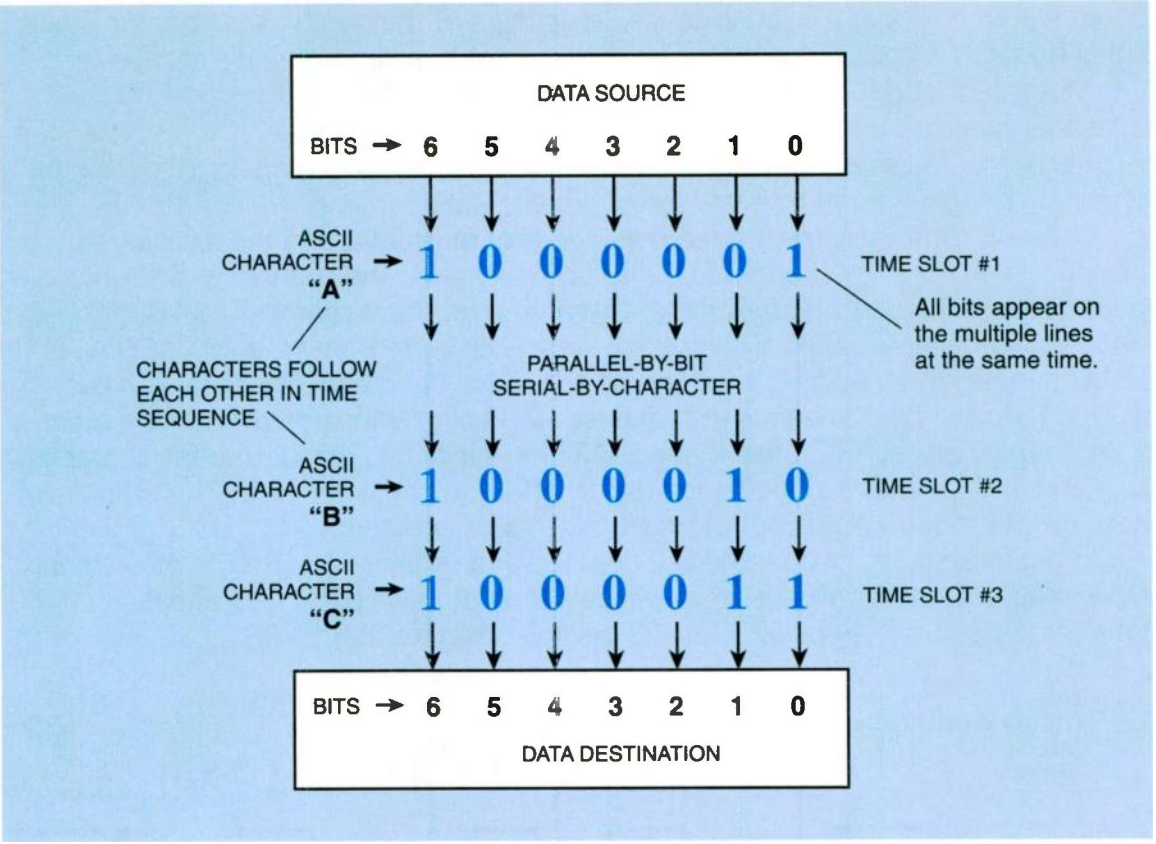


Figure 2-3. Parallel-by-bit, serial-by-character transmission of data.

when the equipment being connected is close by (usually less than 25 feet). As the distance between equipment increases, the multiple lines not only become more costly, but the difficulty of transmitting pulse signals over long lines and keeping them in correct time relationship becomes a major limitation. Over long distances, serial transmission is preferred because it involves only one transmission line.

Serial Transmission of Data

To send data over a single line, the bits must be transmitted in series (appropriately called serial transmission). The 7-bit ASCII character "C" would appear in a series time sequence as shown in *Figure 2-4.* Both the transmitter and the receiver use the same nominal unit element time (called clock rate) shown in *Figure 2-2,* so the system must provide a way for the receiver to duplicate the transmitter clock so it can decode the data correctly. The communications system must ensure that the receiver clock is timed so that errors do not occur. If the characters are seven bits long, the receiving device is designed to know which of the seven bits it is supposed to look for at any one instant.

You might ask: "How does the receiver know when one character ends and the next one begins. How does the equipment get *synchronized*?"

Timing Is Very Important

We will get back to the data communications example shortly, but first let's define synchronization. Synchronization (or sync for short) means "To do things at the *same time*" and "To cause to occur or operate with *exact coincidence in time or rate.*" Since digital systems depend on digital signals which are changing from 0 to 1 or 1 to 0 levels as the system goes through its tasks, it is very important that the system has accurate synchronization or accurate timing.

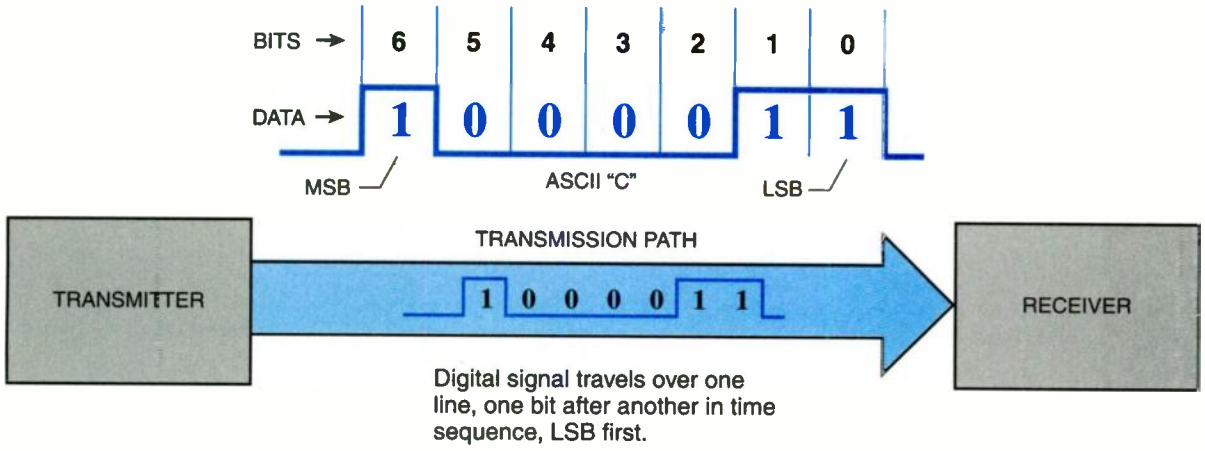


Figure 2-4. Serial Transmission — Bits transmitted over a single line in series time sequence; in this case, an ASCII code representing the letter "C".

Synchronization

Let's demonstrate synchronization using *Figure 2-5.* A timing generator, called a *clock*, generates regular pulses, called *clock pulses*, that are very accurate in frequency; therefore, the *time interval* when the clock pulses are at the 1 level and at the 0 level are very accurately controlled. In addition, the *transitions* from 1 to 0 and 0 to 1 occur at very accurate times, and are used to time or "trigger" action in digital circuits throughout the digital system. For example, in *Figure 2-5,* the transition from 1 to 0 of the clock pulse triggers a sampling circuit that samples the bit of the received data in the middle of the bit time. This allows a variation in the received data of one-half the bit time before an error occurs. We will see many applications of clock pulses throughout this book as we discuss digital electronic circuits.

Synchronizing Transmitter and Receiver

Now, back to the data communications system. We can use it to demonstrate synchronization. First of all, both transmitter and receiver must be operating at the same clock rate. Thus, in serial transmission, you will hear that the digital system is operating at the same *baud rate*. That means the unit element time shown in *Figure 2-5* is the same. In addition to synchronizing its clock with the transmitter clock, the receiving device must be able to determine when to start counting an individual character so that the individual bits can be sampled and identified. Unique marker characters or bits are required to indicate the beginning and end of a character or group of characters. *Figure 2-2* shows start and stop bits, and *Figure 2-5* shows the sampling. In parallel transmission, a signal called the *strobe* or clock on an additional line indicates to the receiver when all the bits are present so they can be read or sampled.

In *synchronous systems* for high-speed data communications, data is transferred in blocks which are framed by STX (start-of-text) and ETX (end-of-text) characters, as shown in *Figure 2-6.* The receiving device synchronizes its clock when it receives an agreed upon SYN character. After the series of SYN characters, the actual data is preceded by an STX character and followed by an ETX character. These two characters tell the receiver when the actual data begins and ends.

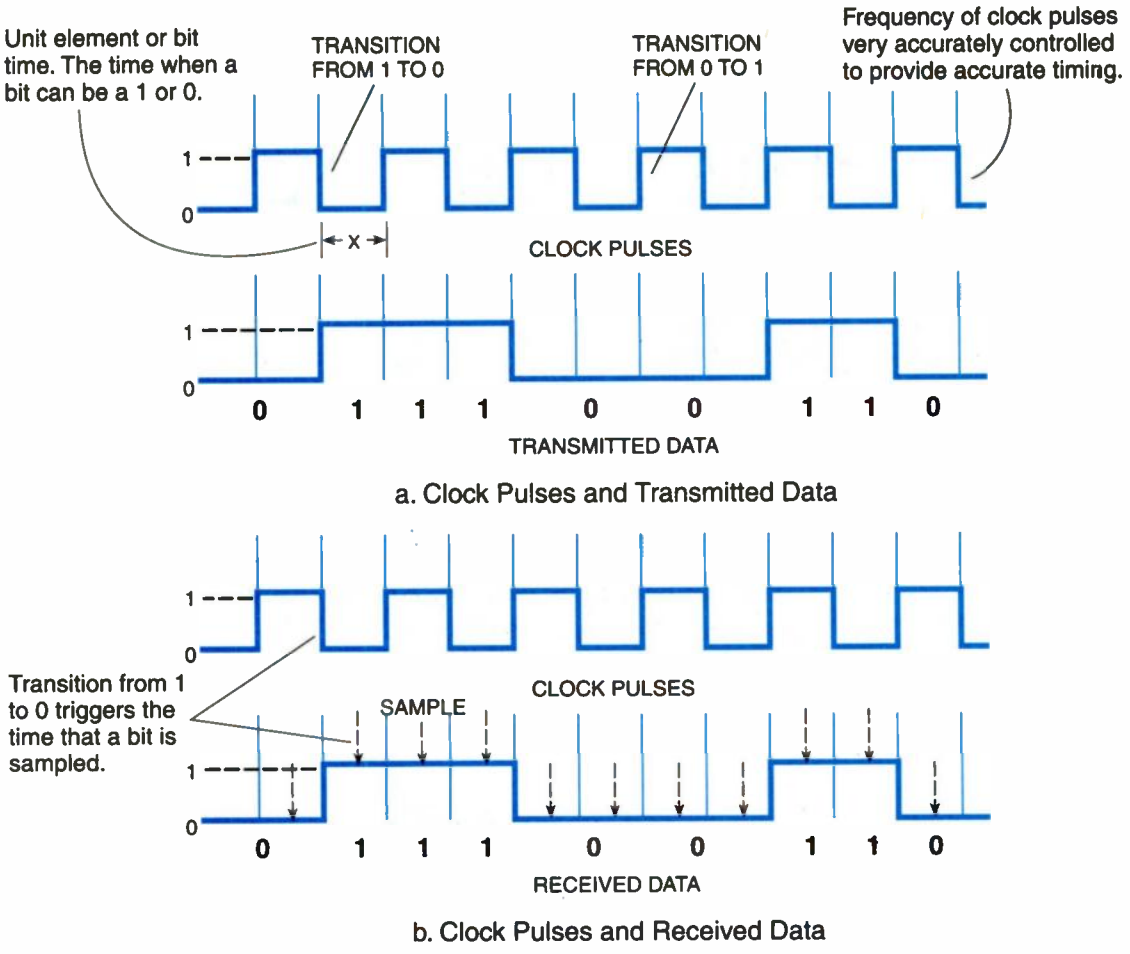


Figure 2-5. Synchronization is provided by a clock generator controlled to a very accurate frequency. Bit times and transition times are held very accurately.

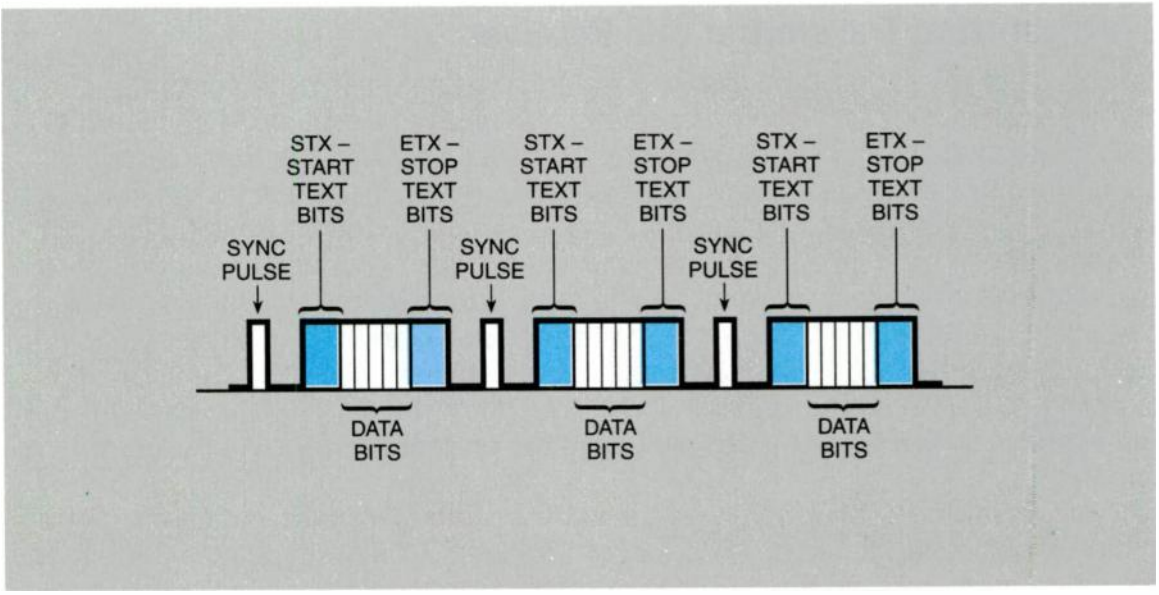


Figure 2-6. The "start text" bits tell the receiver that characters are coming and "stop text" bits tell it that the characters have ended. Sync pulses synchronize transmitter and receiver.

Asynchronous Operation

Asynchronous operation means "not in sync." It means that information is received at a random rate — not at set times. It means that the time between the transmission of characters can vary. In many digital systems, only portions of the system are asynchronous, such as those that handle manual and random operations. For example, when a person presses a key on a keyboard, the transmitted character code is preceded by a START bit and followed by an optional PARITY bit and one or more STOP bits, as shown in *Figure 2-2*. The START bit is always a 0 (SPACE). The STOP bit is always a 1 (MARK). The PARITY bit is sometimes used for error detection; it may be a 0 or a 1, depending on the type of parity used. Characters may be of any *predetermined* number of bits. For example, ASCII uses 7 bits, EBCDIC uses 8 bits, and Baudot uses five bits to represent one character. Whatever the character length, the receiver is "activated" when it reads a START bit, then samples the code at a set baud rate until it reads the STOP bit. It then "idles" until it receives another START bit. (EBCDIC (pronounced "eb see dick") is a code developed and used extensively in IBM® equipment. Baudot (pronounced "baw' dough"), a code developed by Emile Baudot, and its derivatives were used for early communications, particularly on low-speed teletype equipment.)

Code Conversion

Chapter 1 described the decimal, binary and BCD number systems and how each uses its own code to represent numbers. Conversion from one number system to another is often required in digital systems. Converting from one code system to another is called *encoding*; converting back is called *decoding*.

A good example is a common keypad to input decimal numbers into a calculator, as indicated in *Figure 2-7*. Between the keypad and the digital processing circuit is an *encoder* which converts the decimal number pressed on the keyboard into a binary code. As other number keys and operation keys are pressed, the digital processing circuits perform the arithmetic operations using the binary codes. A *decoder* converts the output binary code to a special code that lights the correct segments on the 7-segment LED display to display the result in decimal numbers.

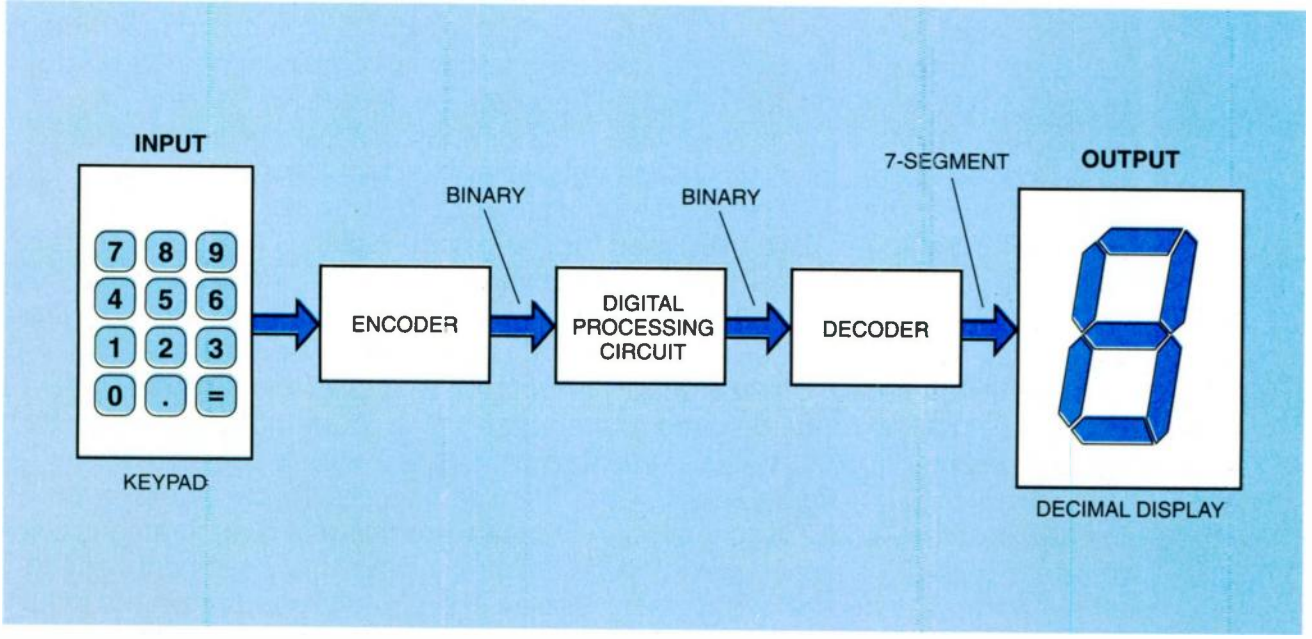


Figure 2-7. A common decimal keypad used as an input device to a calculator, showing encoding and decoding functions.

Encoding

The purpose of the encoder in the calculator of *Figure 2-7* is to translate from "people" language to "machine" language; that is, from decimal to the BCD binary form. Let's examine the simplified encoder circuit shown in *Figure 2-8* to see how it works. The encoder itself is an integrated circuit which has all the encoder circuitry inside. This one is referred to as a 74147A by its manufacturer.

The encoder has nine inputs on the left and four outputs on the right. All of the encoder inputs have +5V, which represents a binary 1, applied when no switch button is pressed. When a switch button is pressed, 0V, which represents a binary 0, is applied to its respective encoder input. All of the encoder outputs are at +5V, or binary 1, when no switch button is pressed and all LEDs are off. When an encoder output goes to 0, it causes its LED to light. Note that these LEDs do not actually exist in the calculator; we are using them to indicate the generated BCD code shown in the encoding table in *Figure 2-8.* A lighted LED indicates a binary 1 for its respective position in the BCD code.

For example, if switch button "7" is pressed, a 0 is applied to terminal 7 of the 74147A. The encoder circuits interpret this input and cause 0 volts to be output at terminals 1, 2, and 4 of the 74147A. This causes LEDs 1, 2, and 4 to light. Since a lighted LED is a 1, the BCD code is 0111, which, of course, is the BCD code for 7. For any switch button pressed, the BCD code will be indicated by the LEDs.

Notice that there is no input terminal on the encoder for the number 0. When the 0 switch button is pressed, it applies no input to the encoder. All encoder input terminals have a 1 (+5V) applied and no LED is lighted; thus, the BCD output code is 0000. The 74147A output is 1111. In a full digital system, the output of this 74147A would be stored in a latch at a particular time with a clock signal.

The 74147A is also called a 10-line-to-4-line priority encoder. The *priority* function means that the encoder will produce a BCD output corresponding to the highest-order decimal digit appearing on the inputs, and will ignore all others. For example, if the 7 and 4 keys on the keyboard are pressed at the same time, both inputs are 0 at the same time, but the BCD output will be 0111 (or decimal 7).

Decoding

A decoder detects the presence of a certain *combination* of bits on its input and then indicates that combination or code by delivering a specific combination on its output. It may be thought of as the opposite of an encoder. The decoder in *Figure 2-7* translates the binary code from the digital processing circuit to a special code that lights the correct segments of the 7-segment display which outputs a decimal number. Let's look at the 7-segment display a little closer.

Several technologies have been used for 7-segment displays: (1) a gas discharge or fluorescent tube (the earliest type); (2) the common light-emitting diode (LED) display; and (3) the liquid-crystal display (LCD), which is the newest and most popular now. Since the LED is the easiest to use of the three, we will cover it in some detail.

A 7-segment display format is shown in *Figure 2-9.* Each of the segments is a single LED that can be forward-biased to produce light. An output of 0 volts from the integrated circuit will light the LED segment to which the output is connected.

The input circuit of *Figure 2-9* simulates the digital processing circuit of *Figure 2-7.* It uses switches and LEDs to input a 4-bit binary number to the integrated circuit decoder in *Figure 2-9.* The decoder converts the binary input to a 7-segment code to light the appropriate decimal digits on the display. The 220-ohm resistors in the input and output circuits limit the LED current to prevent burn out. Pins 3, 4, and 5 are connected through a 1k-ohm resistor to 15 volts. Pin 14 is connected directly to 15 volts and pin 8 is connected to ground. The IC is numbered 7447A by the manufac-

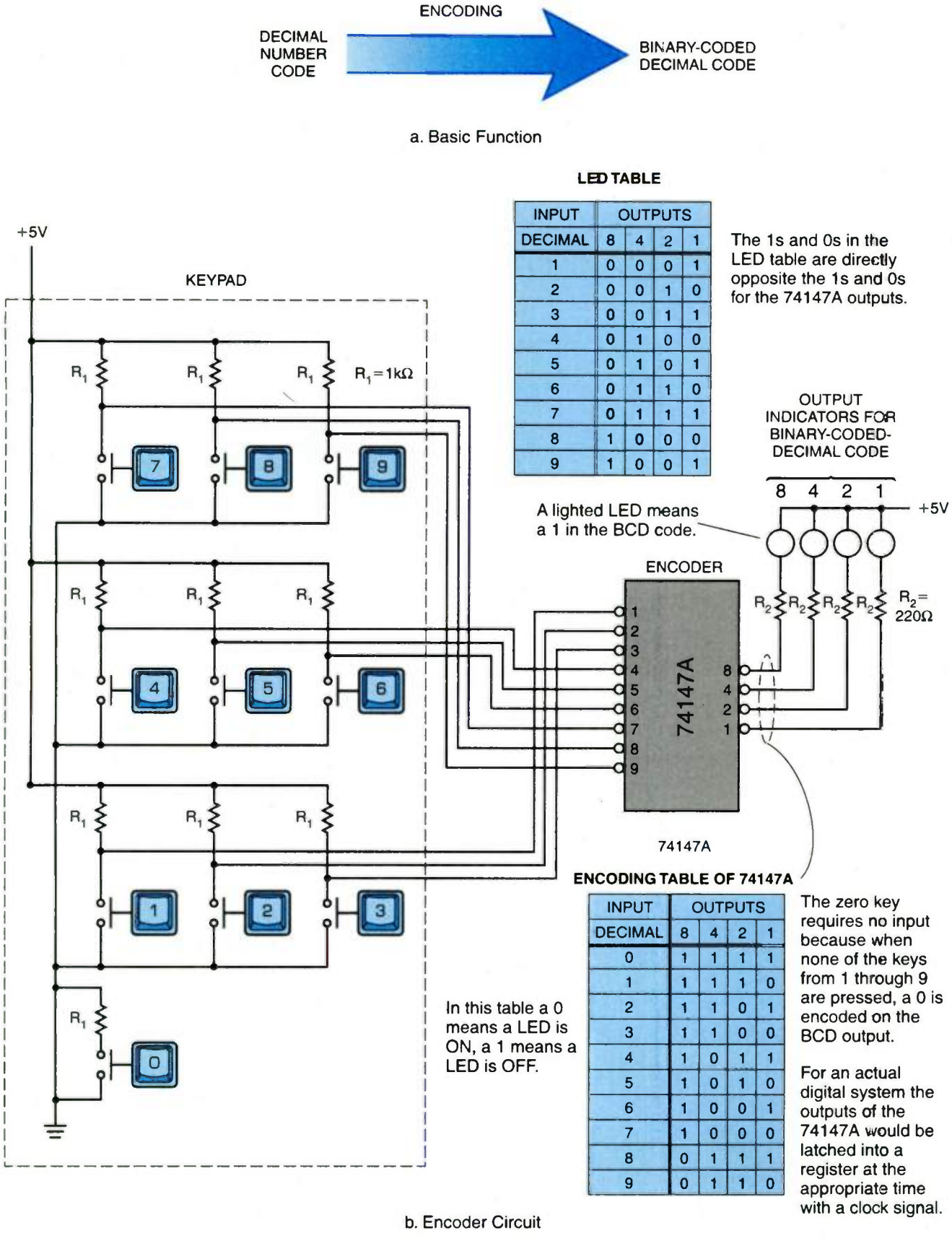
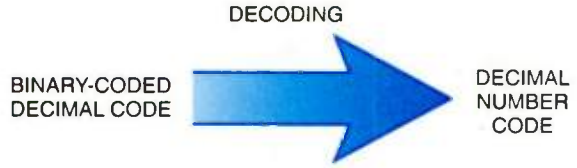


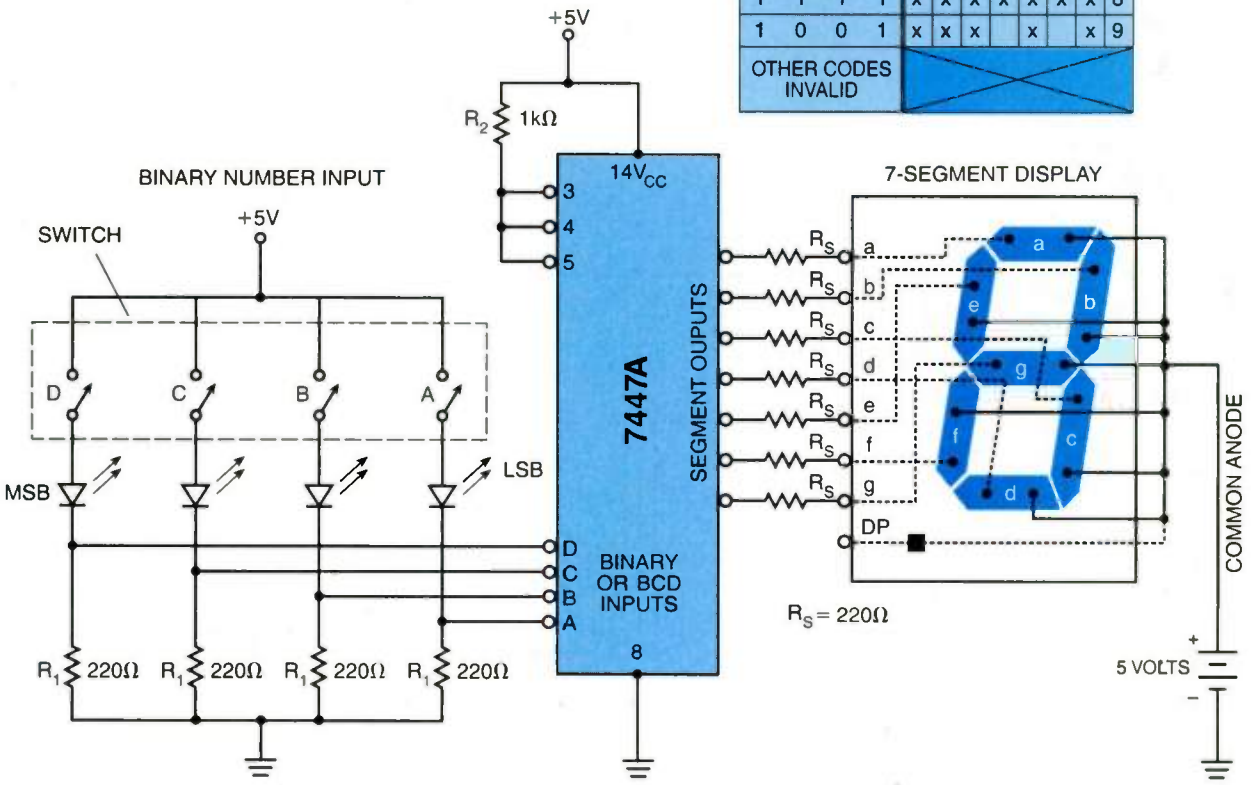
Figure 2-8. An encoder changes one digital code to another. For example, a decimal-to-BCD encoder that uses a commercial integrated circuit shows how it is done.



a. Basic Function

DECODING TABLE

BINARY INPUT				OUTPUT							DECIMAL
D	C	B	A	a	b	c	d	e	f	g	
0	0	0	0	x	x	x	x	x	x		0
0	0	0	1		x	x					1
0	0	1	0	x	x		x		x	x	2
0	0	1	1	x	x	x	x			x	3
0	1	0	0		x	x		x		x	4
0	1	0	1	x		x	x	x		x	5
0	1	1	0				x	x	x	x	6
0	1	1	1	x	x	x					7
1	1	1	1	x	x	x	x	x	x	x	8
1	0	0	1	x	x	x		x		x	9
OTHER CODES INVALID											



b. Decoder Circuit

Figure 2-9. A decoder also changes one digital code to another. It usually is considered as doing the opposite of an encoder. For example, a BCD-to-decimal decoder using a 7-segment display format to output the decimal number.

Example 2. Determine Which Segments are Lighted

A: With all four switches in *Figure 2-9b* set for a binary code of 0000, which segments are activated (lighted)?

Segments a, b, c, d, e, and f, which displays a decimal 0.

B: With switch A on, and switches B, C, and D off, which segments are lighted?

Segments b and c are lighted, which displays a decimal 1.

As an exercise, go through the rest of the BCD codes to determine which segments are activated and the decimal numbers displayed. Because the last six binary numbers are invalid input codes, unique and meaningless codes will be shown on the 7-segment display for them.

turer. A closed "B" switch in the input circuit produces binary code 0010 (which is decimal 2) on the A-B-C-D terminals of the decoder. The decoder produces 0 volts (ground) on the outputs a, b, d, f, and g to display the number "2" on the 7-segment display.

As shown in *Figure 2-9,* modern day LCD displays have a decimal point (DP) LED segment. A special input code is used to activate another decimal point output line from the IC.

Data Selection Function

Mechanical and electronic systems often have a need to select one or more items from a group of items. TVs have channel selectors to select one channel from several possible channels. Washing machines have operating mode selectors to select the wash cycle, water temperature, and time of cycle. *Figure 2-10a* shows a mechanical selector switch on a washing machine for selecting water temperature.

Digital systems also need selectors to direct the flow of digital codes. *Figure 2-10b* shows an electrical diagram of a mechanical switch used as a data selector. The data selector selects and provides one closed path for data to flow. The other options are excluded.

Signal flow through a mechanical data selector can be controlled to flow in either direction or both directions, although the flow is usually one-way. However, the data selector used in digital systems allows signals to flow through it in only one direction — from input to output.

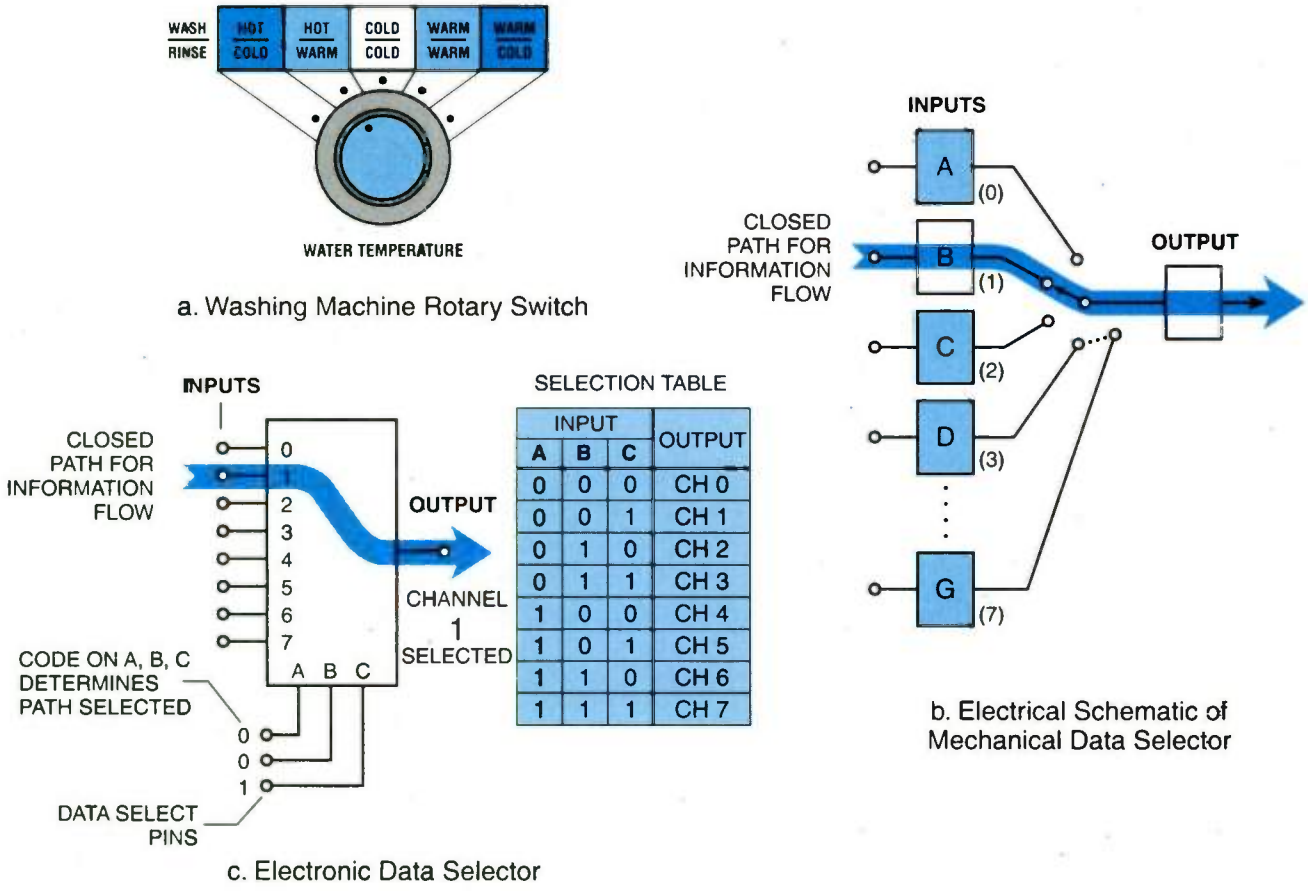


Figure 2-10. Data selectors — A mechanical switch, its electrical diagram, and an electronic version, a digital integrated circuit for digital systems.

In its more advanced form for digital systems, the data selector is a digital electronic integrated circuit *(Figure 2-10c)* that is capable of selecting one set of data or signals from a group and providing a closed path for the selected data to a single output line. It's important to remember that, on the electronic data selector, the digital inputs cannot be used for outputs, and vice versa. As shown in *Figures 2-10b* and *2-10c,* the eight inputs are on the left and the one output is on the right in both the mechanical and electronic versions. Using the mechanical data selector, the switch rotor would have to be physically rotated to the desired position. Using the electronic data selector, the input is selected by placing the proper binary number on the data select inputs (A, B, or C). The electronic version can be thought of as having an "electronic magical genie" inside that rotates the rotor on the rotary switch. This electronic data selector, as we will see in later chapters, is quite a versatile device. It is commonly used as a multiplexer and demultiplexer. In fact, demonstrating the use of the data selector will provide an opportunity to show how the functions of timing and data selectors are combined in digital systems.

Data Selector Demonstration

Look at *Figure 2-11a.* The objective is to take data coming in on one of the data inputs D_0, D_1, D_2 or D_3 and transmit the data over the data line and output it on the data line that matches the data input selected. *Figure 2-11b* is a diagram called a timing diagram, which shows events as they occur in time. In time period #1, data on the D_2 data input is selected, placed on the data line, and transferred to the D_2 data output by the second data selector. In digital systems, the transmitting data selector (from multiple inputs to a smaller number of outputs) is called a *multiplexer*, and the receiving data selector (from small number of inputs to a larger number of outputs) is called a *demultiplexer*.

In time period #2, data on the D_1 input line is transferred over the data line to the output D_1 data line. This demonstrates that the data on the data line depends on time and when you look at the data line. This demonstrates how timing is very important and also explains multiplexing — data is placed on the data line in time sequence based on the data selector input code present during the valid data time period. Since the same data selector code is also applied to the demultiplexer at the same time, the data on the data line is selected through to the proper data output line.

To understand the full operation, the enable signal and its purpose must be explained. The enable signal is a timing signal that says, "Everything is stable at the data and code inputs, so activate the circuit." The enable signal must be a 0 to activate the circuit. Look again at the timing diagram in *Figure 2-11b.* Note that during time period #1 and time period #2, the enable signal is 0 and the circuit is active. Such timing is very important to digital systems because it allows data input and control inputs to change and stabilize before their information is used. The enable timing signal comes from the master clock signal for the system.

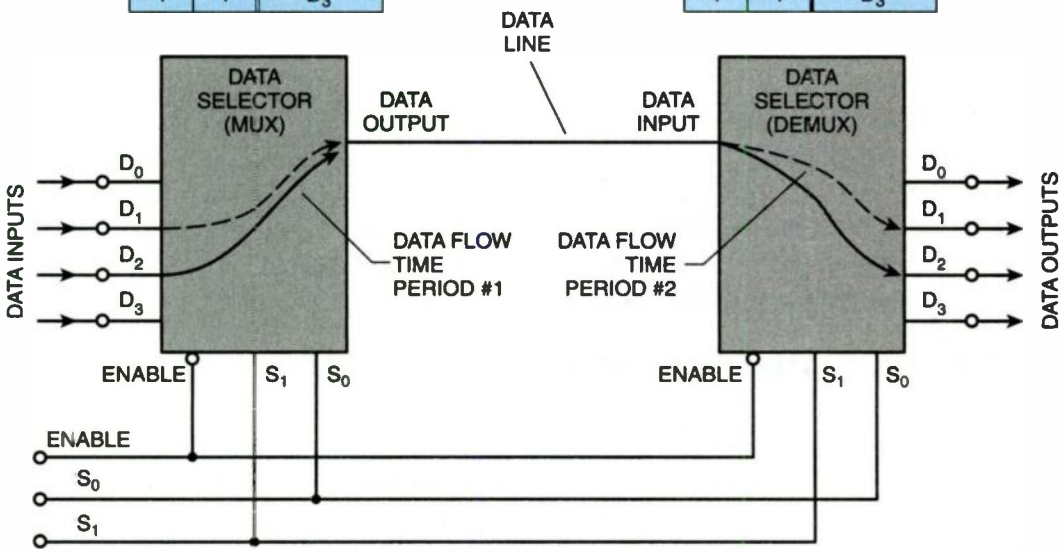
Several other rules are necessary to completely understand *Figure 2-11.*

1. The data outputs from the data selectors are at 0 when the selector is not enabled (enable signal at 1.)
2. The data on the data inputs and code inputs is allowed to change only when the selectors are not enabled.

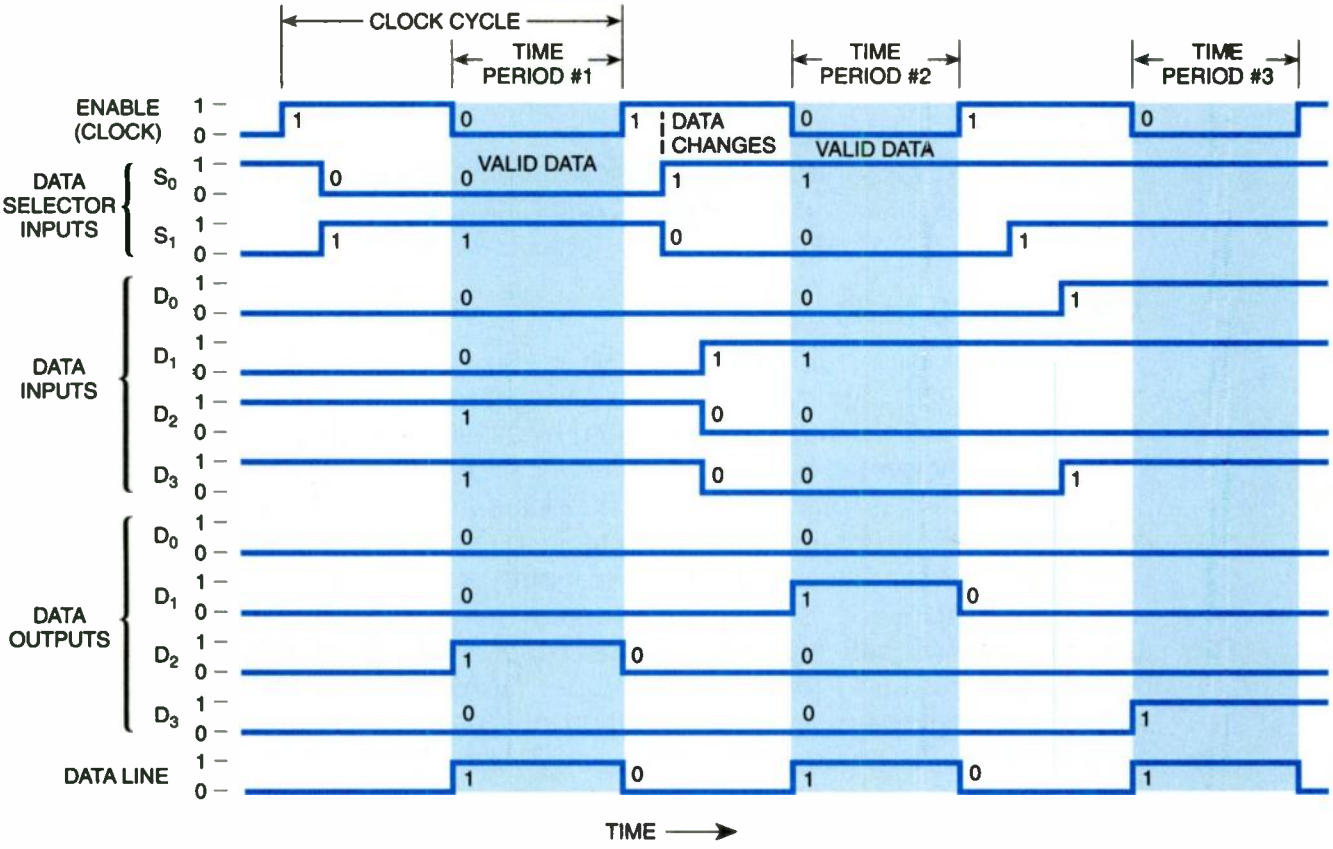
Examine the timing diagram in *Figure 2-11b* to verify how timing is very important and how it is used to assure that digital circuits operate only on reliable data. The timing process just described is technically known as *time division multiplexing*. We'll come back to it in Chapter 9.

DATA SELECTOR INPUTS		INPUT SELECTED
S_1	S_0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

DATA SELECTOR INPUTS		OUTPUT SELECTED
S_1	S_0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3



a. Circuit Diagram



b. Timing Diagram

Figure 2-11. The use of data selectors to demonstrate data flow, timing, multiplexing, and demultiplexing.

Example 3. Determine Data Input and Output

In *Figure 2-11*, during time period #3, determine from the timing diagram which data input and output are selected by the code on S_0 and S_1, and what the state of the output should be, 0 or 1?

Since $S_0=1$, and $S_1=1$, the data input and date output selected is D_3. Since the data input D_3 is a 1, then the D_3 data output is a 1 during the time that the enable signal is a 0.

Notice that, for the data selectors, there are two data-select code lines. Remember that with two select bits, any of four data input or output lines can be selected. Data selectors/multiplexers are commercially available with more data lines than four. With eight data lines (1-of-8), the multiplexer would require three data select lines; with sixteen data lines (1-of-16), it would require four. The number of data select lines and, therefore, the number of data select bits, is just like we saw in the earlier section on Code Conversion.

The Decision Function

Digital systems have a requirement for circuits that make *logical decisions*. This *decision function* is a very important part of digital systems. The fundamental building blocks that make decisions are called *logic gates*. In fact, the name *logic gate* is derived from the ability of the circuit to produce one output level when certain combinations of input levels are present and a different output for other combinations. This is its decision-making ability or ability to perform logical decisions. Many situations in our everyday lives can be expressed in the form of logic statements or functions that are either in "true or false" or "yes or no" statements. Such statements can be implemented using binary digital circuits. There are three basic kinds of digital circuits used to implement logic functions. Enormously complex logic statements are implemented by using the three basic circuits over and over in a set combination. We will introduce the basic three here using simple mechanical switch circuits that are analogous to the actual electronic logic gates. Chapter 3 will provide a more thorough discussion.

AND Logic Circuit

The first basic logic circuit, the *AND* circuit, is shown in *Figure 2-12a*. In digital circuit terms, it is an AND logic gate, or just an AND gate. Notice that the two switches are in series. A "true" condition is represented by a closed switch, and a "false" condition is represented by an open switch. The table that represents the logical results of the various switch conditions is called a *truth table*. It shows the output for all conditions of the inputs. In the truth table, 0 represents an open switch (false) and 1 represents a closed switch (true) for the inputs, and 0 represents zero volts (false) and 1 represents +5 volts (true) at the output. In *Figure 2-12a*, we see by the truth table that an AND gate *is a device whose output is a 1 if, and only if, all its inputs are 1*. S_1 AND S_2 must be 1 for the output to be 1. We say that S_1 AND S_2 equals V_O, or 1 AND 1 = 1. Logic gates are represented on logic diagrams by symbols. The symbol for the 2-input AND gate is shown in *Figure 2-12a*.

OR Logic Circuit

The second basic logic circuit is the *OR* circuit shown in *Figure 2-12b*. In digital circuit terms, it is an OR logic gate, or just an OR gate. This circuit is very similar to the AND circuit except that the two switches are in parallel instead of series. Looking at the

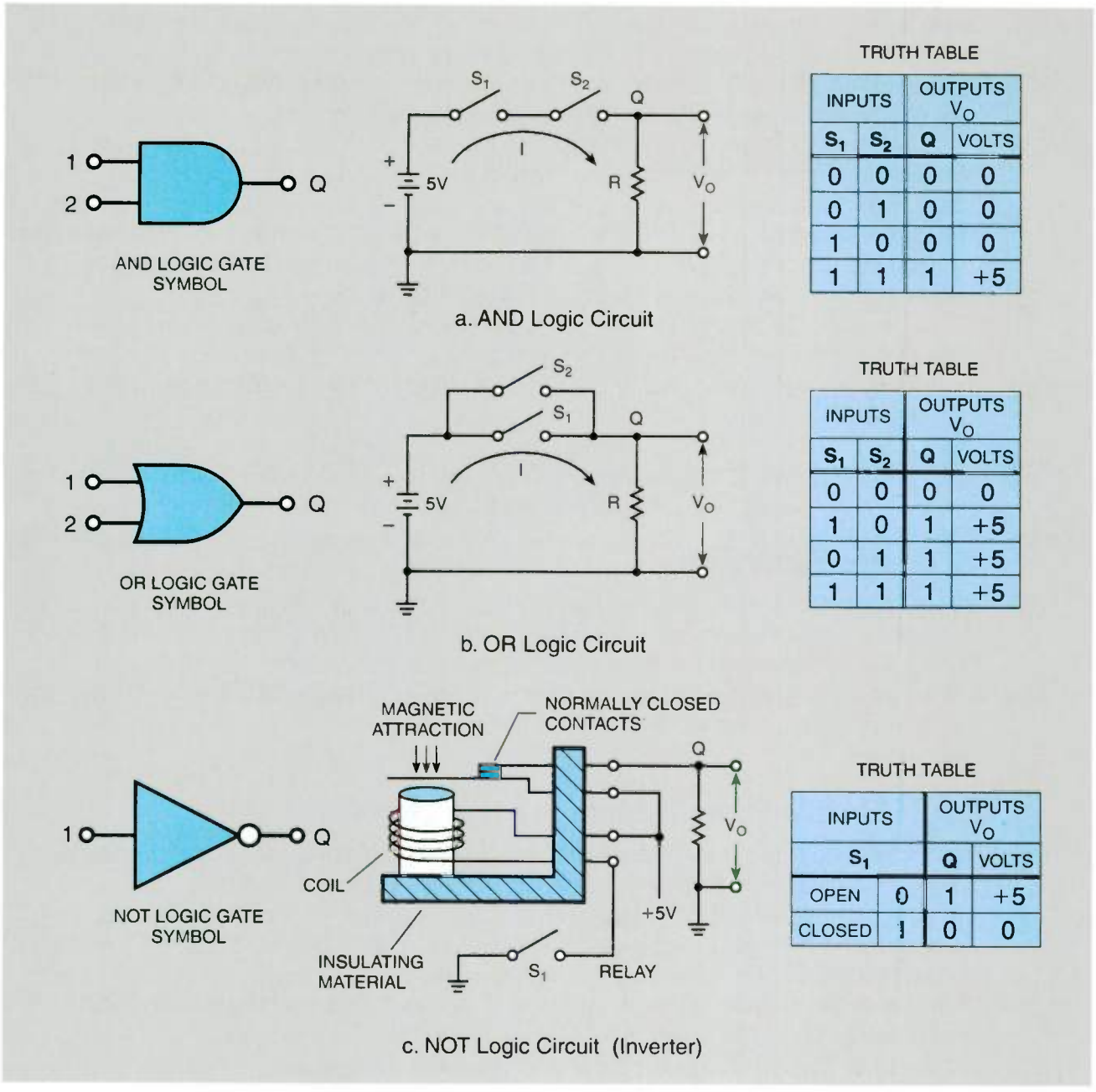


Figure 2-12. The three basic logic circuits, AND, OR, and NOT, and their truth tables. Truth tables show the conditions of the output for all input conditions.

truth table, we see that when either S_1 OR S_2 OR both inputs are 1, then the output V_O equals 1. Thus, an OR gate *is a device whose output is 1 if at least one of its inputs is 1*. The 2-input OR logic gate symbol is shown in *Figure 2-12b*.

NOT Logic Circuit

The third basic circuit is necessary to complete the assortment required to implement all types of decisions with digital logic circuits. It is the *NOT logic circuit*. In digital circuit terms, it is commonly called an *inverter*, but it may be called a NOT gate, or a NOT gate inverter. Our mechanical NOT circuit in *Figure 2-12c* is implemented with an electromechanical relay. It has contacts like a manual switch, but the contacts are controlled by energizing the relay coil with a voltage. When S_1 is closed, the relay coil is energized, and the contacts are pulled open. From the NOT circuit truth table,

shown in the *Figure 2-12c,* the output is 1 when the input is 0, and the output is 0 when the input is 1; that is, *the output is just the opposite of the input.* Remember that an open S1 is a 0 output and a closed S1 is a 1 output. Thus, the NOT gate is *called an inverter because the output state is the inversion of the input state.*

Another common way to refer to the NOT circuit operation and truth table function in digital circuit terms is to say that the NOT gate *complements* a digital value. The complement of a value is its opposite state. The complement of a 0 is a 1; the complement of a 1 is a 0. The complement of a value is shown by drawing a line, or bar, over it. For example, if A = 0, then $\overline{A}$ = 1. The symbol for the NOT gate inverter commonly used on digital logic diagrams is shown in *Figure 2-12c.*

The AND, OR, and NOT logic gates are the fundamental building blocks of digital electronic systems. We will employ these gates in Chapter 3 to form other frequently encountered gates and to perform many more logic operations. The NOT gate always has only one input and one output. The AND and OR logic gates can have many inputs — not just two as shown in the figures. The more inputs on the AND and OR gates, the more input columns will be added to the truth table and the more rows will be added to handle all the combinations of states for the inputs.

Storage Function

In digital electronic systems, the means for storing digital information (data) on a temporary or permanent basis for future use is provided by memory circuits. Storing data is a very important function necessary in most digital systems. Because of the variety of ways that the storage function is used in digital electronic systems, a number of different types of memories have evolved, each suited to particular applications.

Logic Circuits Form Bistable Storage Elements

To illustrate how the storage of digital information is accomplished in digital systems, a very primitive circuit that can store a bit of information will be used. Such an element can be formed from two inverters cross-coupled as shown in *Figure 2-13a.* Such a digital storage device is called a bistable element; that is, it has the two stable states indicated in *Figure 2-13a*. It is this ability to reside in a given state indefinitely that makes the bistable element useful as a memory (storage) device. A bistable circuit like this is called a *flip-flop* because it can "flip" to one state and "flop" to the other state. The flip-flop is the basic building block for registers, which are short term or intermediate memories used in digital systems. Flip-flops can also be the basic element for computer memories that are generally used for longer-term storage of larger amounts of data.

A 4-bit register stores four bits, sometimes called a *nibble (Figure 2-13b)*; an 8-bit register stores eight bits, called a *byte (Figure 2-13c)*; and a larger register stores a *word*. Typical word sizes are 16, 32 and 64 bits. A 16-bit register is shown in *Figure 2-13d.* The state of a bit stored in any of these registers is determined by examining the Q output. In most cases, the complementary output, $\overline{Q}$, is also available.

Large Memory Storage

Figure 2-14 shows a much larger memory than the registers of *Figure 2-13.* This is used only as an example to demonstrate the technique used to provide the function. The size (256 bytes) is primitive with respect to today's state-of-the-art computer memories that may have 16 megabytes or more of storage capacity in one IC package. But the purpose and technique is the same; that is, to provide a matrix of storage elements, with each element capable of storing one bit as either a 1 or a 0, and organized into bytes or words which can be located at a specific address by a decoder.

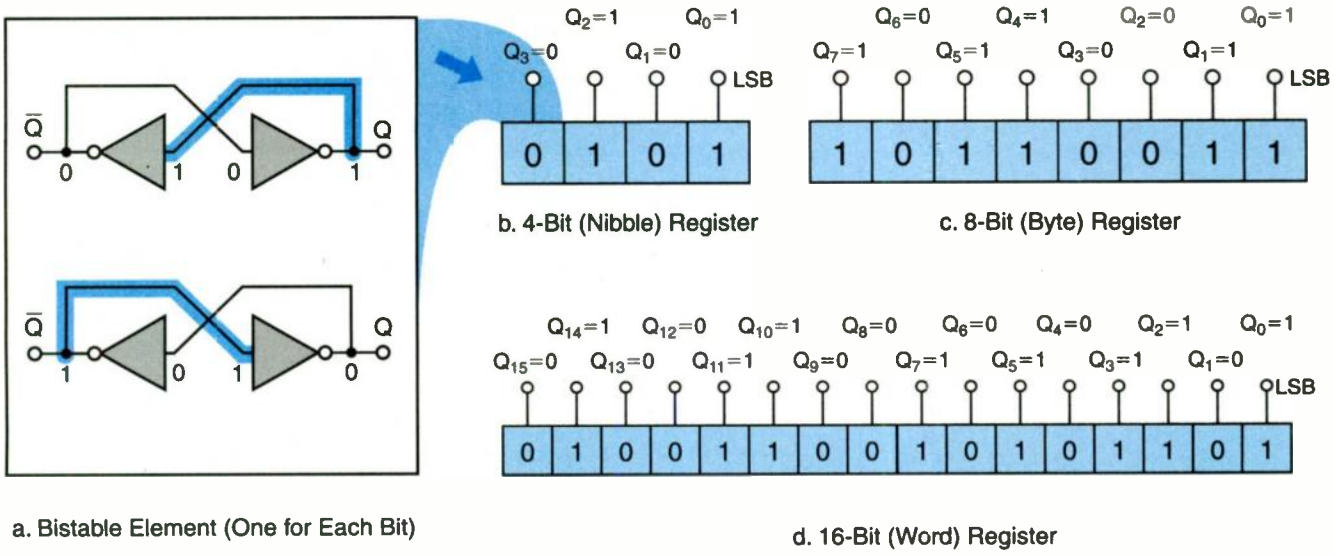


Figure 2-13. Storage of digital information can be accomplished using bistable elements that have two states, Q = 0 or Q = 1. These bistable elements can be combined to form registers for nibbles, bytes, or words.

Example 4. Finding Decimal Equivalents for Binary Numbers

In *Figure 2-13,* what are the decimal number equivalents in the 4-bit, 8-bit and 16-bit registers?

32768	16384	8142	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	Binary Weight
												0	1	0	1	Binary Number
													4		+1	= 5 Decimal
								1	0	1	1	0	0	1	1	Binary Number
								128		+32	+16			+2	+1	= 179 Decimal
0	1	0	0	1	1	0	0	1	0	1	0	1	1	0	1	BinaryNumber
	16384			+2048	+1024			+128		+32		+8	+4		+1	= 19629 Decimal

In the 256-byte memory of *Figure 2-14,* each of the 256 memory cells stores an 8-bit byte; therefore, a decoder that can provide 256 combinations is required. An 8-bit binary number provides the address to locate a specific memory cell. Cell #71 is located by the input address shown in *Figure 2-14.*

Digital system memories are divided into two types — random access memories and read-only memories. Let's look at random access memory first.

Random Access Memory

In digital systems, random access memory (RAM) is used for temporary storage of data, programs, control status, etc. It permits *random access* to the stored information, which means that all storage locations in it are equally accessible in approximately the same amount of time. (In contrast, a magnetic tape requires *sequential access*; that is, the "storage locations" on the tape must be read from the beginning (or some designated starting point) in sequence until the data is found.)

RAM has both read and write capabilities; therefore, read/write memory would be a better description than random access memory, but RAM is the accepted and commonly used term.

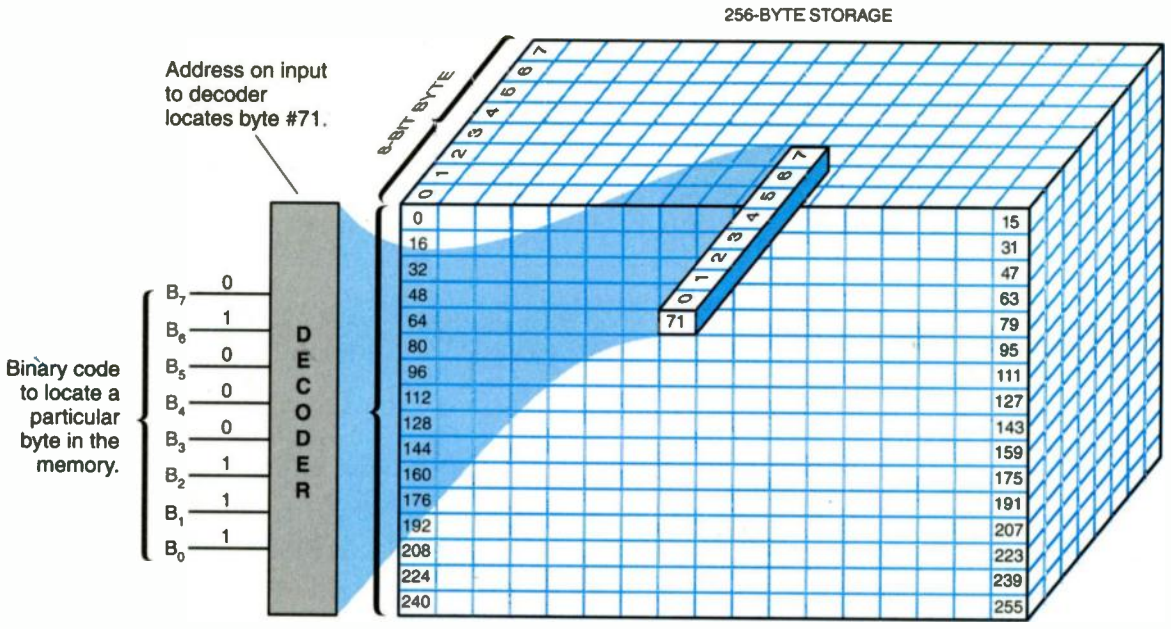


Figure 2-14. Bistable memory cells can be combined into large memories. The individual bytes are located by applying a specific binary input to the decoder.

RAM is *volatile*, which means the stored data are lost if operating power is removed. The two basic types of RAM are *static* and *dynamic*. Static RAM (SRAM) uses bistable circuits as the storage elements. As long as power is on, the data remains stored. Dynamic RAM (DRAM) uses capacitors as the storage elements. The capacitors discharge over a short time, even with power applied, so they must be recharged (refreshed) periodically to maintain the data.

Example 5. Finding Binary Equivalent for a Decimal Number

What binary address will locate memory cell #187 in the RAM of *Figure 2-14?* Choose the weighted values that are required to make the decimal number. Place a 1 in the weights required and 0 in the weights not required.

128	64	32	16	8	4	2	1	Binary Weight
128		+32	+16	+8		+2	+1	= 187 Decimal
1	0	1	1	1	0	1	1	Binary Number

Read-Only Memory

In a read-only memory (ROM), data are permanently stored and can be read from the memory at any time. The stored data is retained even when the power is removed, so it is a *nonvolatile* memory. A ROM cannot be written to because the information is either programmed into the device during manufacturing or semi-permanently programmed into the device by the user and not easily altered. There is no write operation for a ROM; the data can be altered only by a special "burner."

Arithmetic Logic Unit (ALU) Functions

A very important function required in digital systems is the execution of arithmetic and logical operations. We want to give you an understanding of what is meant by ALU operations. There is no magic in doing these operations electronically. Instead, they follow step-by-step procedures that are very similar to what we do with decimal numbers. Different rules apply, but the technique is the same. All operations are with

binary numbers, and since electronic circuits are very fast, successive simple operations are used to perform a complex operation. For example, instead of multiplying two complex numbers with a complex multiply routine, successive additions are performed to do the multiplication.

Binary Addition

The procedure for adding numbers in binary is similar to adding in decimal except that the binary sum is made up only of 1's and 0's. As in decimal, when the sum exceeds the radix, you must *carry* a 1 to the next higher place. There are four basic rules for adding binary digits:

Rule 1	0 + 0 = 0	
Rule 2	0 + 1 = 1	
Rule 3	1 + 0 = 1	
Rule 4	1 + 1 = 10	(0 with a carry of 1)

Notice that in Rule 4, the 1 in the sum is carried to the next column as in regular decimal addition.

Here are some examples of binary addition with the decimal equivalent shown for reference.

0101	5	1111	15	110	6
+0010	+2	+1100	+12	+100	+4
0111	7	11011	27	1010	10

Example 6. Adding Binary Numbers

Add the following two bytes of data:

		Decimal
10101101	Byte #1	173
11001110	Byte #2	206
101111011	Sum	379

Check yourself using decimal equivalents.

Binary Subtraction

There are also four basic rules for subtracting binary digits:

	Minuend		Subtrahend		Difference	
Rule 1	0	−	0	=	0	
Rule 2	0	−	1	=	1	(with a borrow 1)
Rule 3	1	−	0	=	1	
Rule 4	1	−	1	=	0	

In subtraction, the first (top) number is called the *minuend,* the second (bottom) number is called the *subtrahend,* and the answer is called the *difference.* Notice that in Rule 2 a 1 is being subtracted from the smaller number 0. Therefore, a 1 must be *borrowed* from the 2's column, that is a 10_2 (or a 2_{10}). Now the subtrahend 1 is subtracted from the minuend 10_2 leaving a difference of 1 in the 1's column.

Here are a few examples, again with the decimal equivalent shown for reference.

11	3	101	5	1111	15
−10	−2	−011	−3	−1010	−10
01	1	010	2	0101	5

Two's Complement Arithmetic

The borrowing becomes tedious in digital subtraction so digital system designers devised a system whereby subtraction could be accomplished by using complements and adding. Thus, the *two's complement method* evolved for subtraction. It is now the most widely used.

- The *two's complement* of a binary number is found by adding 1 to the one's complement.
- The *one's complement* of a binary number is found by simply changing all 1's to 0's and all 0's to 1's.

This is illustrated as follows:

Binary number	110	10101	0110 0010
Swap 0's and 1's to get 1's complement	001	01010	1001 1101
Add 1 to 1's complement to get 2's complement	+1 010	+1 01011	+1 1001 1110

When adding numbers in the two's complement form, simply perform a regular binary addition to get the result. When subtracting numbers in the two's complement form, it makes a difference which is larger — the minuend or the subtrahend. This results in different procedures for the two cases.

Case 1. To subtract a smaller number from a larger one:

1. Determine the 2's complement of the smaller number.
2. Add the 2's complement to the larger number.
3. Discard the final carry (for this case there is always a carry).

Case 2. To subtract a larger number from a smaller one:

1. Determine the 2's complement of the larger number.
2. Add the 2's complement to the smaller number.
3. There will be no carry. The result is in 2's complement form and is negative.
4. To get the final answer, take the 2's complement and change the MSB. The MSB is actually the sign bit. If the MSB is a 1, then the number is negative. If the MSB is a 0, then the number is positive.

Example 7. Subtract 28_{10} from 19_{10} Using 2's Complement:

$28_{10} = 11100_2$, $19_{10} = 10011_2$

1's complement of 28_{10}	00011
add 1	+ 1
2's complement of 28_{10}	00100
add 19_{10}	+ 10011
result	10111
1's comp. of the result	01000
add 1	+ 1
2's comp. of the result	01001
change the sign bit (MSB)	$11001_2 = -9_{10}$

The sign bit is 1 so the number is negative.

In digital systems, there is an integrated circuit that not only does arithmetic, but logical operations as well. It is called an *arithmetic logic unit* (ALU). The specific arithmetic or logical operation to be performed is chosen by placing a specific binary code on the *mode select* inputs. We will look at the ALU in more detail in Chapter 5, and show operation in a system in Chapter 8.

Interfacing Function

Digital systems, or at least portions of them, do not normally operate only by themselves. They must be coupled to other systems. This is called the *interfacing function*. An *interface* is the *coupling or connection* between the digital electronic devices. A typical interface between two digital systems is shown in *Figure 2-15.*

If only one manufacturer made all of the digital electronic equipment on the market, there would be fewer, maybe no, interface problems. But anyone who has tried to couple different digital devices knows that this is not the case. The potential for confusion is immense. When talking about interfaces, it is important to make sure of some basic facts:

1. That both systems use the same connector type and the same pins and wires on the connecting cable
2. That the correct format (called protocol) of digital information is used
3. That the clock timing of both systems is the same
4. That the correct control characters are present
5. That the right voltage levels are used.

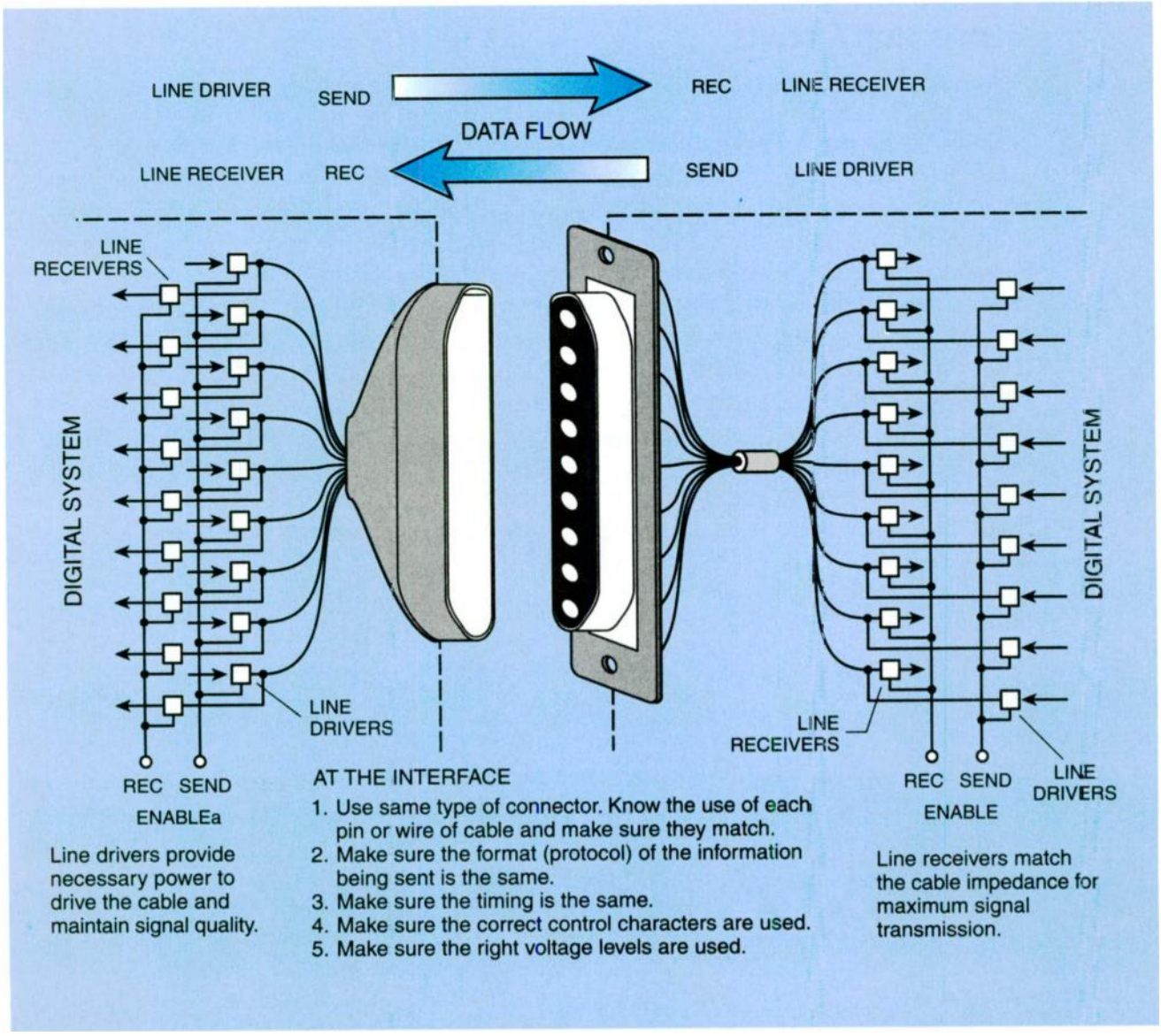


Figure 2-15. Digital systems must work together. The interface between the systems is very important and must be correct. Many times the interface is governed by standards.

These are only a few of the factors that must be considered. Sometimes these factors and others are taken care of by "standards" established by organizations of the designers and manufacturers of equipment, such as the Electronic Industries Association (EIA) or the Institute of Electrical and Electronic Engineers (IEEE). Two examples of common standards are RS-232 and IEEE-488. The standards are intended to make sure that information or data can be understood by another device. These may not mean anything to you now, but they will as you get further into your study of digital electronics.

Line Drivers and Line Receivers

Look at *Figure 2-15* again. Line drivers and line receivers are shown on each side of the interface with enable signals called SEND and REC. If data is to flow to the right, the left-side line drivers and the right-side line receivers are enabled, and vice versa for data flow to the left. The line drivers and receivers assure proper match to the interconnecting cable for maximum signal quality and transmission. Line drivers and receivers may be circuits using standard semiconductor devices or they may be optical semiconductor devices.

Conversion Circuits

Remember from Chapter 1 that most real-world data is analog by nature. Temperature is not simply cold or hot, nor is light simply dark or bright. Therefore, conversion factors are required to allow digital systems to *interface* with analog systems. To interface from analog to digital, a special encoder, called an *analog-to-digital converter* (ADC) translates the analog information at the input into a digital output. A digital signal has discrete voltage levels and an analog signal is one that varies continuously from a minimum to a maximum voltage or current. After processing the data in the digital system, the signal must be converted to interface from digital to analog. This is done with another interfacing circuit called a *digital-to-analog converter* (DAC). The combined ADC-DAC is sometimes referred to as a *data acquisition* system, as shown in *Figure 2-16.* Here a physical parameter, such as temperature, is converted to an analog voltage by a *sensor*, and the analog voltage becomes the input to the ADC. The digital processing unit handles the data digitally for speed and accuracy and outputs a digital number to the DAC. The output of the DAC is usually a voltage or current that runs a motor, or a heater, or a solenoid to complete the action.

In the next chapter, we will see how electronic circuits are used to make decisions.

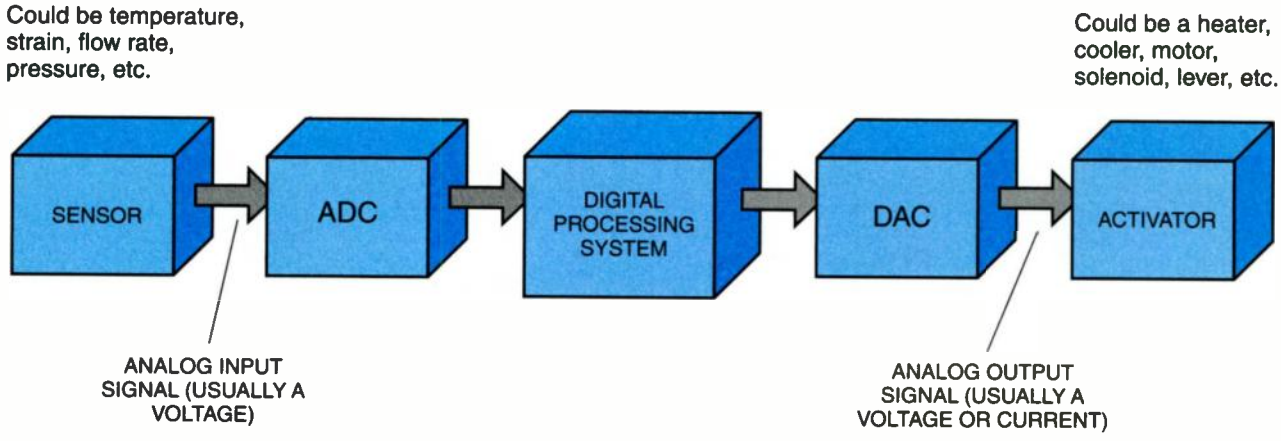


Figure 2-16. A data acquisition and control system using analog-to-digital and digital-to-analog converters.

Quiz for Chapter 2

1. A _____ is the smallest unit of binary information.
   a. digit
   b. bit
   c. base
   d. radix
2. The total number of pieces of information communicated by a code is dependent on the _____ .
   a. total number of bits
   b. code word used
   c. degree of uncertainty
   d. clock rate
3. How many bits are required to represent 88 different objects in binary?
   a. 5 bits
   b. 6 bits
   c. 7 bits
   d. 8 bits
4. Using the minimum number of bits, what would the coding efficiency be to represent 88 keys of a piano?
   a. 76.4%
   b. 81.7%
   c. 87.5%
   d. 68.75%
5. The primary use of parity is for _____ .
   a. accuracy
   b. simplicity
   c. error correction
   d. redundancy
6. In BCD, the binary representations of the numbers 10 through 15 .
   a. are the easiest to represent.
   b. are the most difficult to represent.
   c. are not used and are invalid.
   d. are represented in octal only.
7. If an IC has bubbles on its output, it means that
   a. the outputs are normally 1s or HIGHS until activated.
   b. the outputs are normally 0s or LOW until activated.
   c. the output will only go LOW.
   d. the output will only go HIGH.
8. _____ has the advantage of being able to send several data signals on a single line.
   a. Encoding
   b. Multiplexing
   c. Synchronizing
   d. ASCII
9. A data multiplexer is a digital electronic circuit like the _____ .
   a. synchronizer.
   b. encoder.
   c. demultiplexer.
   d. data selector.
10. With eight data lines (1-of -8) , the MUX would require _____ data select lines.
    a. two
    b. three
    c. four
    d. five
11. The _____ is also known as a data distributor.
    a. synchronizer.
    b. demultiplexer
    c. encoder.
    d. data selector.
12. Two or more switchers in series represent an _____ logic gate.
    a. AND
    b. OR
    c. NOT
    d. NOR
13. In _____ data are permanently stored and can be read from the memory at any time.
    a. magnetic tape memories
    b. random access memories (RAM)
    c. read-only memories (ROM)
    d. volatile memories
14. A _____ uses a capacitor to store the data and it must be recharged (refreshed) periodically.
    a. static ROM
    b. static RAM
    c. dynamic ROM
    d. dynamic RAM
15. An _____ is the coupling or connection between digital electronic devices.
    a. encoder
    b. A/D converter
    c. gate
    d. interface

Answers:
1b, 2a, 3c, 4d, 5c, 6c, 7a, 8b, 9d, 10b, 11b, 12a, 13c, 14d, 15d

Questions and Problems for Chapter 2

1. How many binary bits are required to represent 36 different objects?
2. How does a communication receiver know when one character ends and the next one begins?
3. Describe the advantage of serial and parallel data transmission, respectively.
4. What are the three basic blocks between the keypad and the display of a calculator and what is the purpose of each?
5. For the encoder circuit of *Figure 2-8b,* if the "5" button is pushed, what is the outputs of the 74147A?
6. For the decoder circuit of *Figure 2-9b,* if the A and C switches are ON and B and D are OFF, what outputs of the 7447A are LOW?
7. In *Figure 2-11b,* during the time period #2, determine from the timing diagram which data input and output are selected by the code on S_0 and S_1, and what state the output should be, 0 or 1?
8. Draw the symbols for the three basic logic gates with their truth tables.
9. Draw a circuit diagram showing how two inverters can be used to form a bistable flip-flop.
10. If the 8-bit register of *Figure 2-13c* has the bits 10001100_2 stored in it, what is the decimal equivalent?
11. What binary address will locate cell #154 in the RAM of *Figure 2-14?*
12. Add the two bytes of binary data 10001011 and 01001101.
13. What is the 2's complement of 11001?
14. Use 2's complement to subtract 5_{10} from 12_{10}.

CHAPTER 3

How Do Decision Circuits Work?

Using Transistor Switches for Binary Bits

We have seen that the transitions between 0 and 1 levels in digital circuits are caused by switching from one voltage level to another (voltage levels in widespread use are 0V and +5V). The manual switches and electromechanical relays that we have used up to now have almost ideal electrical characteristics. Their contacts have almost zero resistance(0Ω) when they are closed (ON) and almost infinite resistance (∞Ω) when they are open (OFF). Except for the fact that they must be operated manually, toggle switches have ideal electrical parameters. Transistors are substituted for the mechanical switches in digital electronic circuits. Even though they only approach the ideal ON an OFF parameters, they do the job very well and are very reliable.

Today, the majority of digital electronic switching circuits use transistors in either discrete circuits or integrated circuit form. There are two kinds of transistors, based mostly on their operation and construction. One is called *bipolar*; the other *field-effect*. A metal-oxide-semiconductor field-effect transistor, or MOSFET, is a very common field-effect transistor used in integrated circuit form. We are going to concentrate on the bipolar junction transistor (BJT) first because MOSFETs are rarely used in discrete form. We will then look at the MOSFET in more detail, and again in Chapter 7 when we discuss integrated circuits. The bipolar transistor has been used in circuits extensively in both discrete and integrated form. Let's first see how it is used in a circuit as a digital logic switch.

The Bipolar Transistor as a Switch

The bipolar junction transistor (BJT) is constructed with three doped semiconductor regions separated by two PN junctions as shown in *Figure 3-1a*. The three regions of the transistor are the emitter, base, and collector. There are two types of bipolar junction transistors as shown in the figure. The only significant difference is that one has a P-type semiconductor region "sandwiched" between two N-type semiconductor regions — an NPN transistor, and the other has an N-type region between two P-type regions — a PNP transistor. The transistors are represented by the schematic symbols shown in *Figure 3-2b*. Note that an NPN transistor has the emitter arrow pointing out and the PNP transistor has the emitter arrow pointing in.

Logic States

Recall that in order for a transistor to conduct base current, the base-emitter junction must be forward biased. The base current turns on current between collector and emitter. Therefore, turning ON base current turns ON the transistor from collector to emitter. Turning OFF base current turns OFF the transistor from collector to emitter. If

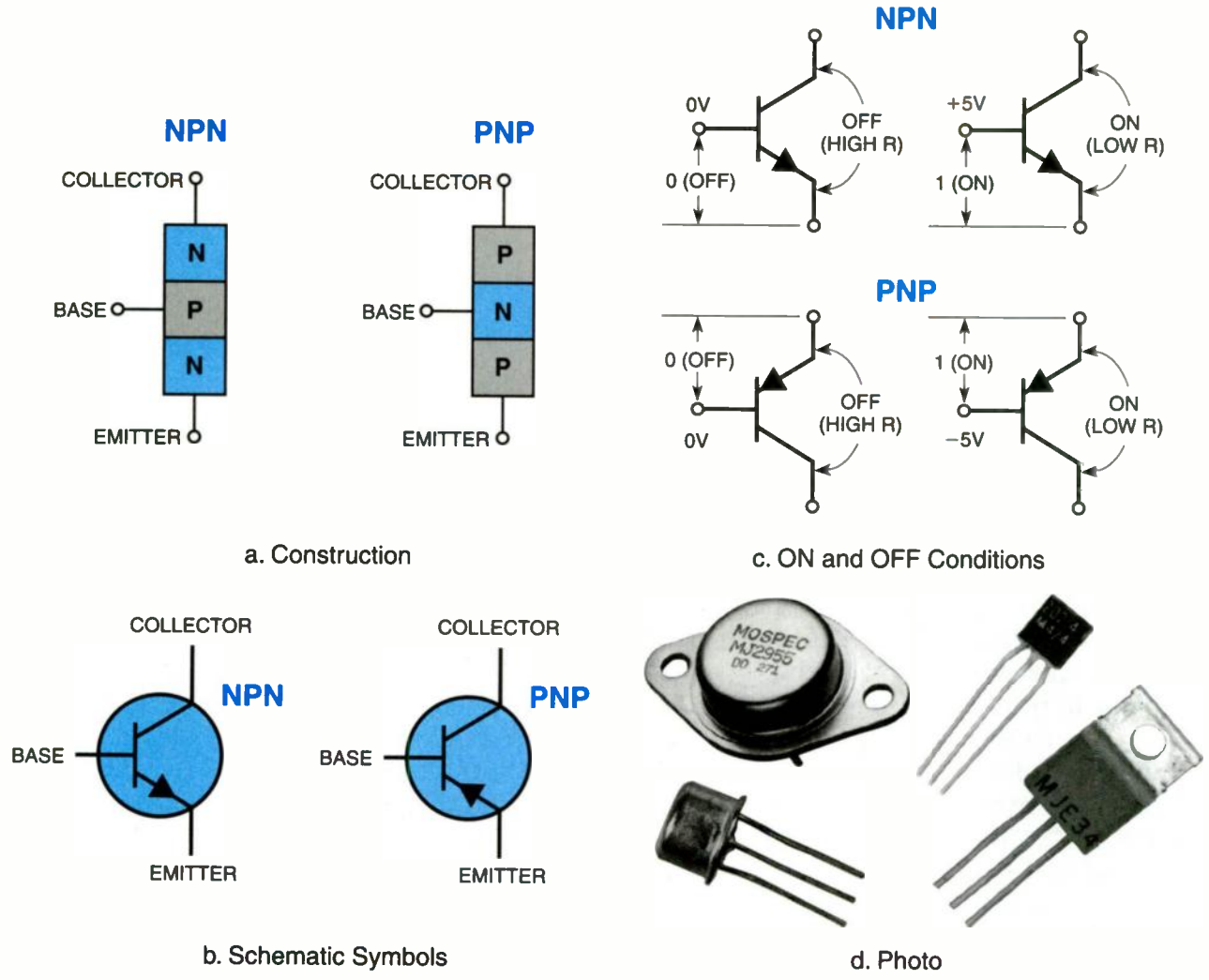


Figure 3-1. NPN and PNP bipolar junction transistor construction, schematic symbols, logic states, and photo showing size.

logic level 0 is represented by zero volts between base and emitter of an NPN transistor shown in *Figure 3-1c,* and a logic level 1 is represented by +5V with the base more positive than the emitter, then a 0 applied to the NPN base will provide on OFF logic state at the collector-to-emitter output, and a 1 applied to the NPN base will provide an ON logic state at the collector-to-emitter output. The states are shown in *Figure 3-1c.* In the collector-to-emitter output, the OFF state is a high resistance, and the ON state is a low resistance.

A similar condition exists for the PNP transistor as shown in *Figure 3-1c;* however, now the 1 logic level is −5V rather than +5V, because in a PNP transistor, the base must be negative with respect to the emitter to forward bias the PN junction. Typical packaged transistors are shown in *Figure 3-1d.*

Actual States

The basic circuits for the operation of a NPN and PNP bipolar transistor as a switch are illustrated in *Figure 3-2.* The NPN transistor uses $+V_{CC}$ and 0V and +5V logic levels as shown. When a PNP is used, the polarity is reversed and the logic levels are 0V and −5V. When used as a digital electronic switch, a BJT is normally operated in two states — either *cutoff* or *saturation, but not in between*. When a transistor is saturated, its emitter-base junction is forward-biased and base current is large enough to cause its collector current to reach its saturated value. When a transistor is saturated, it is like a closed switch from the emitter to the collector. When a transistor

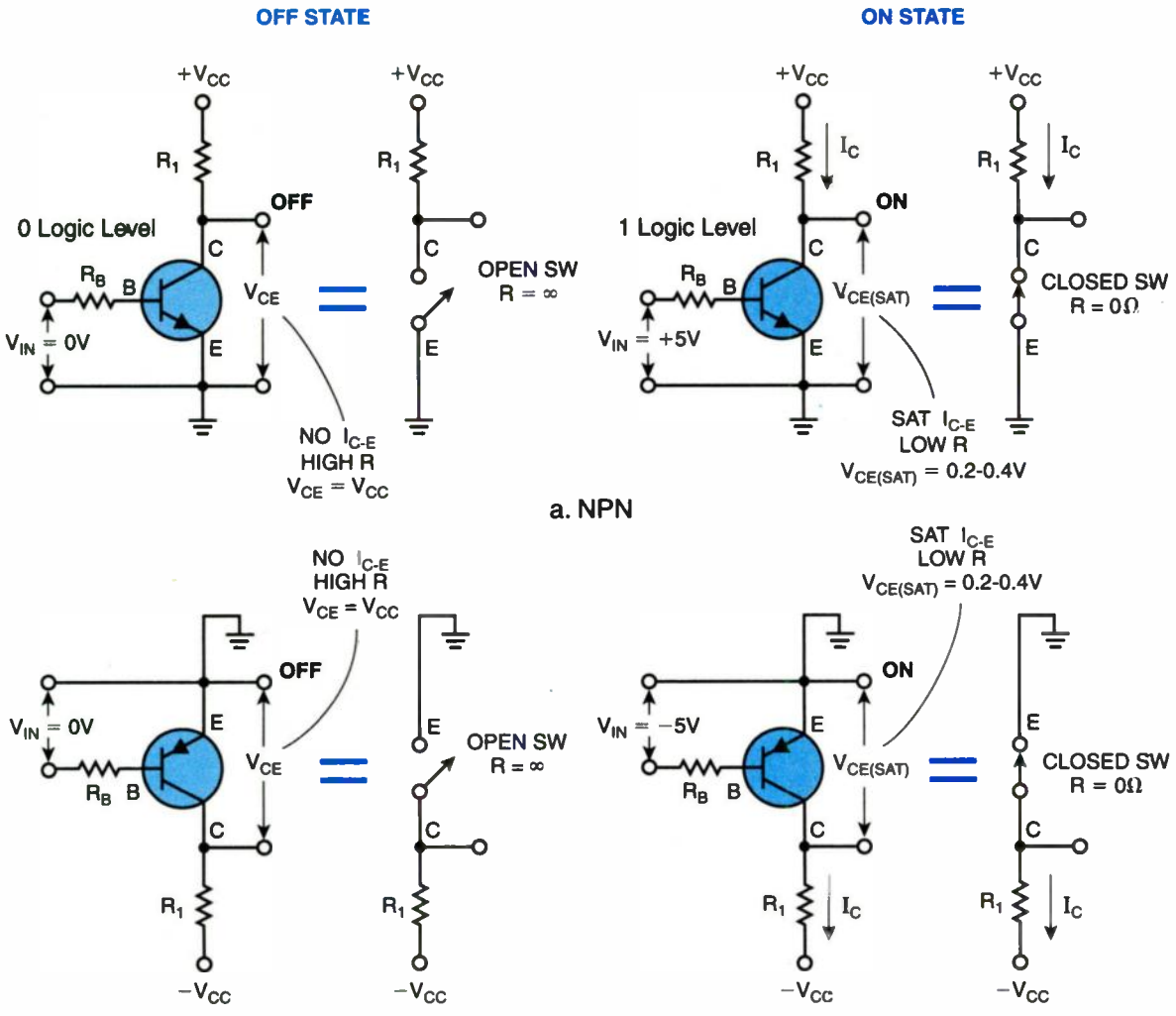


Figure 3-2. Basic circuits for the operation of a BJT as a switch, both NPN and PNP. The OFF (cutoff) and ON (saturation) conditions of a BJT are shown with the mechanical toggle switch equivalent circuit comparisons.

is cutoff, its emitter-base junction is reversed biased and base current is zero which causes its collector current to be zero. When a transistor is cutoff, it is like an open switch from the emitter to the collector. *Figure 3-2* shows the toggle switch comparisons.

The input voltage, V_{IN}, and output voltage, V_{CE}, for both ON and OFF states are shown in *Figure 3-2*. Since the resistance from emitter-to-collector is almost zero when the transistor is saturated (ON), the collector-to-emitter voltage, $V_{CE(SAT)}$, is almost zero (0.2V to 0.4V). If V_{CC} = 5V, the 5V is all dropped across the resistor. When the transistor is cutoff (OFF), all currents are approximately zero. There is, therefore, almost zero volts dropped across the resistor, R_1, so the output voltage (V_{CE}) is approximately equal to 5V (V_{CC}).

If you were to look at the V_{IN} and V_{OUT} (V_{CE}) of particular BJT circuits inside a digital electronic system, such as a computer, you would likely see changes in the voltages with time as shown in *Figure 3-3b.* As described in *Figure 3-2,* notice that when the input to each circuit is zero, the transistor is OFF (cutoff), and the output voltage is +5V. When the input is +5 V, the transistor is ON (in saturation) and the output voltage, $V_{CE(SAT)}$, is approximately zero. Although $V_{CE(SAT)}$ is 0.2 V instead of zero, it is still considered small and usually ignored.

Example 1. Calculating I_C and V_{CE}

Calculate the current through the NPN collector resistor and the V_{CE} output voltage in *Figure 3-2a* for each input logic level. Assign a value of 2 kilohms to R_1.

INPUT = 0: The transistor base current is zero, so the collector-to-emitter current is zero. The collector resistor is in series with the collector and +5V, which means that its current is zero. With no current through the load resistor, there is no voltage drop across it; therefore, $V_{CE} = +5V$.

INPUT = 1: The transistor base current is turned on, so there is collector-to-emitter current. Ignoring $V_{CE(SAT)} = 0.2$ V dropped across the transistor from collector-to-emitter, the collector current (and therefore the series collector resistor current) is given by:

$$I_C = \frac{V_{CC}}{R_C}$$

$$I_C = \frac{5V}{2\ k\Omega}$$

$$I_C = 2.5\ mA$$

and the output voltage is given by:

$$V_{CE} = V_{CC} - I_C \times R_C$$
$$V_{CE} = 5\ V - (2.5\ mA \times 2\ k\Omega)$$
$$V_{CE} = 5\ V - 5V = 0\ V$$

Each Circuit is Really an Inverter

Notice in *Figure 3-3a* that the output of each circuit goes LOW when the input goes HIGH, and vice versa. This means that each circuit is acting as an *inverter*. The inverter takes a digital signal level at its input and *complements* it to the opposite state at its output (1 becomes a 0, and 0 becomes a 1). For this reason, the inverter is also known as a *complementing switch*. The *truth table* for each inverter shown in *Figure 3-3a* is the same as the NOT logic gate (inverter) of *Figure 2-12c*.

Logic Circuits Drive Other Logic Circuits

The first inverter is directly coupled to the second inverter. A 1 on the input to the first inverter produces a 0 at the input to the second inverter. The 0 at the input to the second inverter produces a 1 at the output. The result of one logic circuit driving another is that the state of V_{IN1} (a 1) has been produced at the second inverter output ($V_{OUT2} = 1$). Since $V_{CE(SAT)}$ of each inverter output is about 0.2V and it takes 0.7V to forward bias a silicon transistor's base-emitter junction, when inverter #1 is ON (logic state 0), since the output is directly coupled to the input of inverter #2, it holds the transistor in inverter #2 OFF (logic state 1). To be able to propagate decisions through logic gates formed by electronic circuits, the logic circuits must have enough current output to drive more than one logic circuit coupled to the output. The number of logic circuits that the output can drive is called the *fanout* of the logic circuit. Note, as shown in *Figure 3-3b,* that a digital serial code input on V_{IN1} varying with time propagates through to V_{OUT2}.

Safety Margins on Voltage Levels

Note one other point. There is a safety margin built into logic circuit designs. It only takes 0.7V to forward bias a silicon transistor base-emitter junction. In *Figure 3-3a,* +5V is the V_{CC} level when inverter #1 is OFF. The current, I_{IN}, into the forward-biased

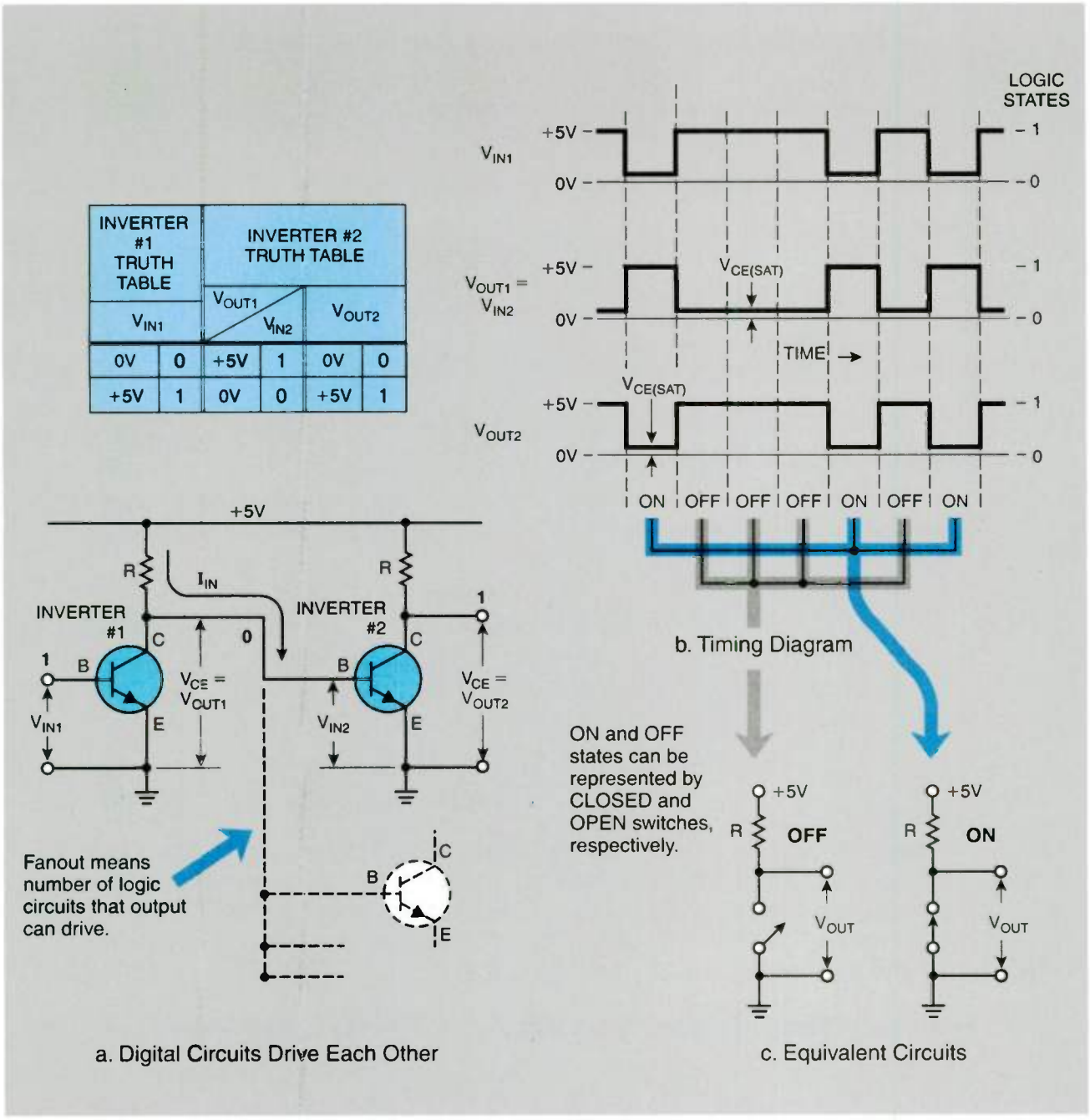


Figure 3-3. Two inverter circuits show how digital circuits drive each other and the equivalent circuits. A timing diagram shows the time relationship of signals.

transistor must be 1 mA in order to drive inverter #2 ON into saturation. Even if R is equal to 4 kΩ, which would cause a 4V drop across R, there is still 1V to forward bias the base-emitter junction. There is a safety margin of +0.3V. With a smaller R, it would be greater.

When inverter #1 is ON, we've already stated that V_{OUT1} ($V_{CE(SAT)}$) is about 0.2V. Since it takes 0.7V to forward bias the base-emitter junction, there is a safety margin of 0.5V. This means circuits, voltages, and transistors can vary, but the logic circuits still perform without error. The safety margin voltages are called *noise margins.* Interferences, variations, and deviations can be up to the noise margins before an error occurs.

Example 2. Determining Logic Gate Noise Margin

Using *Figure 3-3a* with R = 2 kΩ, determine the noise margin of the voltage level that turns on inverter #2 when inverter #1 is OFF. Assume it takes 1.5 mA to turn ON inverter #2.

Since $V_{CC} - I_{IN} R = V_{IN2}$:

$5V - (1.5 \times 10^{-3})(2 \times 10^{3}) = V_{IN2}$
$5V - 3V = V_{IN2}$
$2V = V_{IN2}$

The noise margin is:

$V_{IN2} - V_{BE}$ (for forward bias)
$2V - 0.7V = 1.3V$

Logic Circuit Families

Over the years there have many types of digital logic circuit families. Their names were based on the interconnection of the components or upon the technologies used. Here are some of them:

DCTL	Direct-Coupled Transistor Logic
RTL	Resistor-Transistor Logic
RCTL	Resistor-Capacitor-Transistor Logic
DTL	Diode-Transistor Logic
ECL	Emitter-Coupled Logic
TTL	Transistor-Transistor Logic
MOS	Metal-Oxide Semiconductor Logic
CMOS	Complementary Metal-Oxide Semiconductor Logic

Because most decision circuits use logic circuits that are made in integrated circuit form so that hundreds of thousands of gates can be made at the same time, TTL, MOS, and CMOS circuits have won out over the other families. TTL uses bipolar junction transistors and MOS uses a field-effect transistor made of a metal-oxide semiconductor sandwich. Let's look at TTL first.

Putting Transistors Together to Form Gate — TTL Logic

Transistor-transistor logic (TTL) is so named because it depends on transistors alone to perform logic operations. To understand, let's look at the input stage of a typical logic gate shown in *Figure 3-4a.* It is a basic inverter. Q_1 is an NPN transistor that has multiple N emitters fabricated in a P base where each emitter-base diode serves as an input to the inverter gate. When a base-emitter of Q_1 is forward biased, Q_1 is in saturation and $V_{CE(SAT)}$ is less than the forward-biased V_{BE} of Q_2. Thus, if V_{IN} is 0V, then Q_2 will be OFF because the $V_{BE} = V_{CE(SAT)} = 0.2V$. No collector current in Q_2 means $V_{OUT} = +5V$. Therefore, using only one input, the truth table would show an inverter. With an input of 0V, a 0 logic level, the output is +5V, a 1 logic level.

Even when V_{IN} is supplied by another transistor in saturation (V_{IN} would be $V_{CE(SAT)} = 0.2V$), there is still 0.3V noise margin because:

V_{BE} of $Q_2 = V_{IN} + V_{CE(SAT)}$ of Q_1
$= 0.2V + 0.2V$
$= 0.4V$

Since V_{BE} of Q_2 needs to be 0.7V to be forward biased, there is 0.3V noise margin. Incidentally, D_A and D_B are diodes on the inputs to protect the logic gates from static electricity damage. Also, Q_1 transistors have been fabricated with as many as 10 emitters.

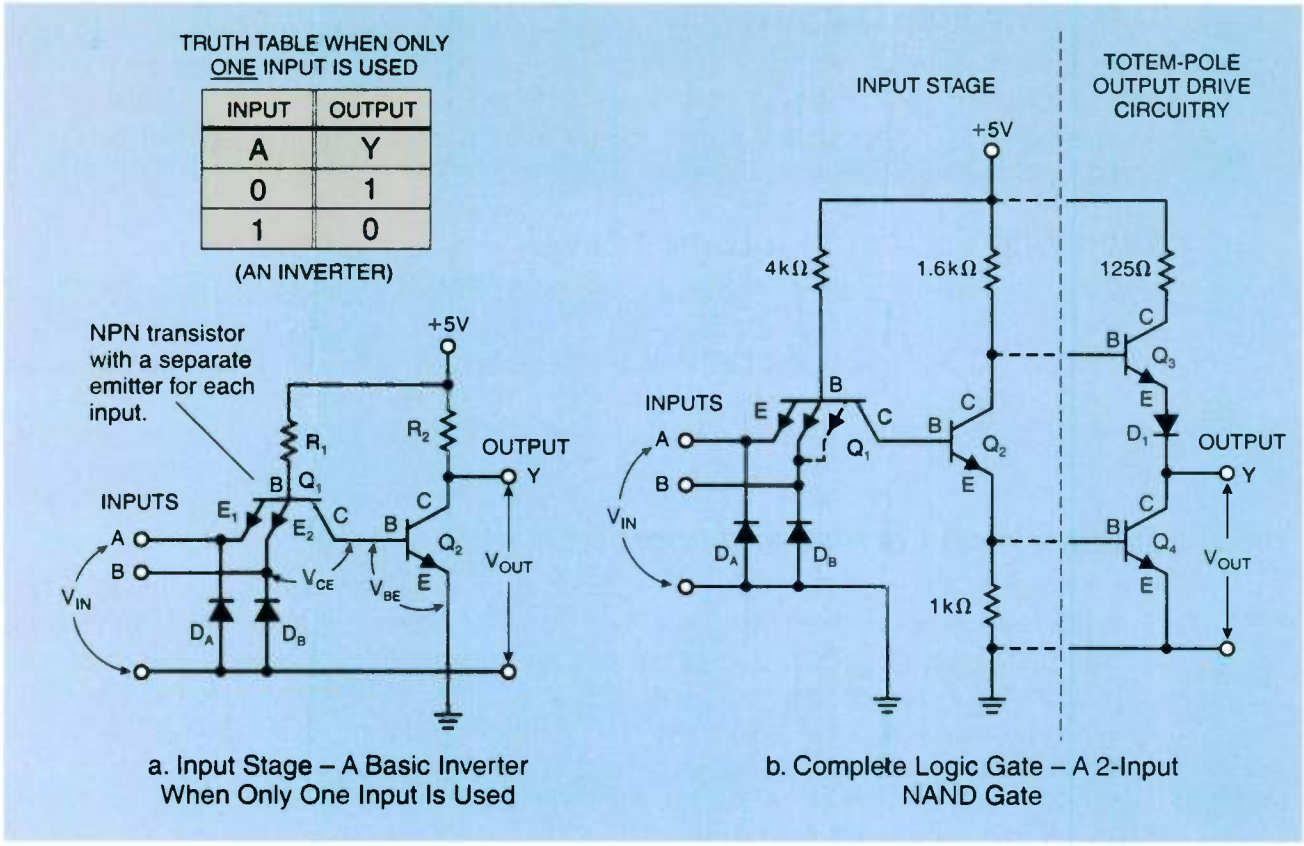


Figure 3-4. A complete TTL 2-input NAND gate is made up of an input stage and a totem-pole output.

When V_{IN} = +5V, the function of Q_1 essentially changes, the emitter-base diode becomes reverse-biased (no current) and the base-collector diode becomes forward biased. It turns into a diode in series with the base-emitter diode of Q_2 conducting current through R_1 from V_{CC} to ground through the forward-biased base-emitter of Q_2. Q_2 is turned ON and its output $V_{OUT} = V_{CE(SAT)}$ or approximately 0V. For an input of +5V, a 1 logic level, the output is 0V, a 0 logic level.

Example 3. TTL Logic Minimum High-Level Voltage

What is the minimum input voltage, V_{IN}, to keep Q_1 from transistor action when Q_2 is ON?

When Q_2 is ON:

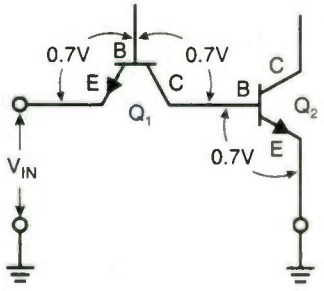
V_{BE2} of Q_2 = 0.7V

V_{BC1} of Q_1 = 0.7V

$V_{BE2} + V_{BC1}$ = 1.4V

Since V_{BE1} = 0.7V

$V_{IN} = 1.4V - 0.7V = 0.7V$



So the absolute minimum V_{IN} is 0.7V. However, to have at least 0.3V noise margin, the minimum V_{IN} = 1.0V. Typically, in TTL logic the minimum high-level input voltage is 2V.

The Totem Pole Output

To improve the simple inverter into a logic gate with fast switching speeds and be able to drive a large number of other gates (a high fanout), the designers added what is called an "active pull-up" or "totem-pole" output circuit. A totem-pole circuit is added to the inverter in *Figure 3-4b.*

Supplying Current to a Logic 1 Level

Figure 3-5a shows the conditions when the circuit output supplies a large drive current to an output at a logic 1 level. V_{IN} is at 0 so Q_2 is OFF and its output is at a 1. As a result, Q_3 is ON and Q_4 is OFF and draws no current. Therefore, I_{ON} into the base of Q_3 is increased by the current gain of Q_3 to provide a large drive current to the output. The mechanical switch equivalent circuit is shown to help you understand the circuit.

Sinking High Current at a Logic 0 Level

The conditions are opposite in *Figure 3-5b.* V_{IN} is now +5V so Q_2 is ON. With Q_2 ON, Q_4 is ON because it is driven by Q_2. Q_3 is OFF and supplies no current. With Q_4 ON, a low resistance path is provided from the output to ground to be able to "sink" a large amount of current when the output is at a 0 logic level. The mechanical switch equivalent circuit is again shown to help you understand the circuit. Diode D_1 is added in the circuit to assure that Q_3 is held OFF when Q_2 is ON.

What Kind of Logic Gate?

We talked about using only one input for the logic gate of *Figure 3-4* and ended up with an inverter. What kind of logic gate do we have if we use both inputs? Let's see if we can reason it through. Let's say that +5V = 1 and 0V = 0. If in *Figure 3-4,* input A is at 0V, Q_1 is ON and Q_2 is OFF. With Q_2 OFF, Q_3 is ON and Q_4 is OFF. The output Y is a 1. Let's put that in our truth table. Remember a truth table shows all combinations of outputs for all combinations of inputs. We have two inputs and one output.

INPUTS		OUTPUT
A	B	Y
0		1

What happens if we apply different inputs to input B. If we apply a 0 to input B at the same time as input A, the conditions of Q_1, Q_2, Q_3, and Q_4 do not change. The output Y is still a 1. Our truth table becomes:

INPUTS		OUTPUT
A	B	Y
0	0	1

If we apply a 1 to input B at the same time as input A is 0, this means that input B is reverse-biased because the forward bias of input A controls. It holds the condition of Q_1, Q_2, Q_3, and Q_4 so that output Y is still a 1. Our truth table becomes:

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	1

Now let's reverse the two input conditions. A is now a 1 and B is now a 0. Obviously, we have the same conditions as A = 0 and B = 1. Our truth table becomes:

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	1
1	0	1

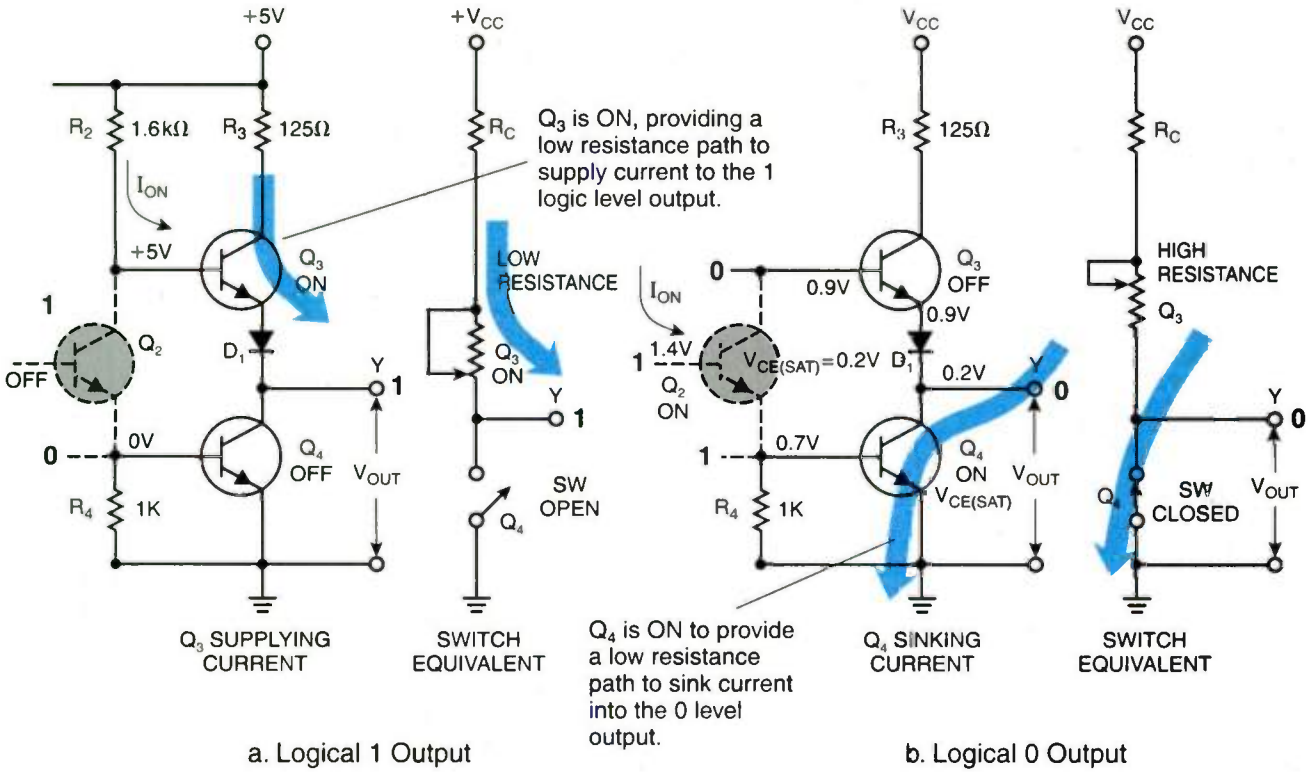


Figure 3-5. The totem-pole output of a TTL gate has excellent capability to supply current when the output is a 1, and to sink current when the output is a 0.

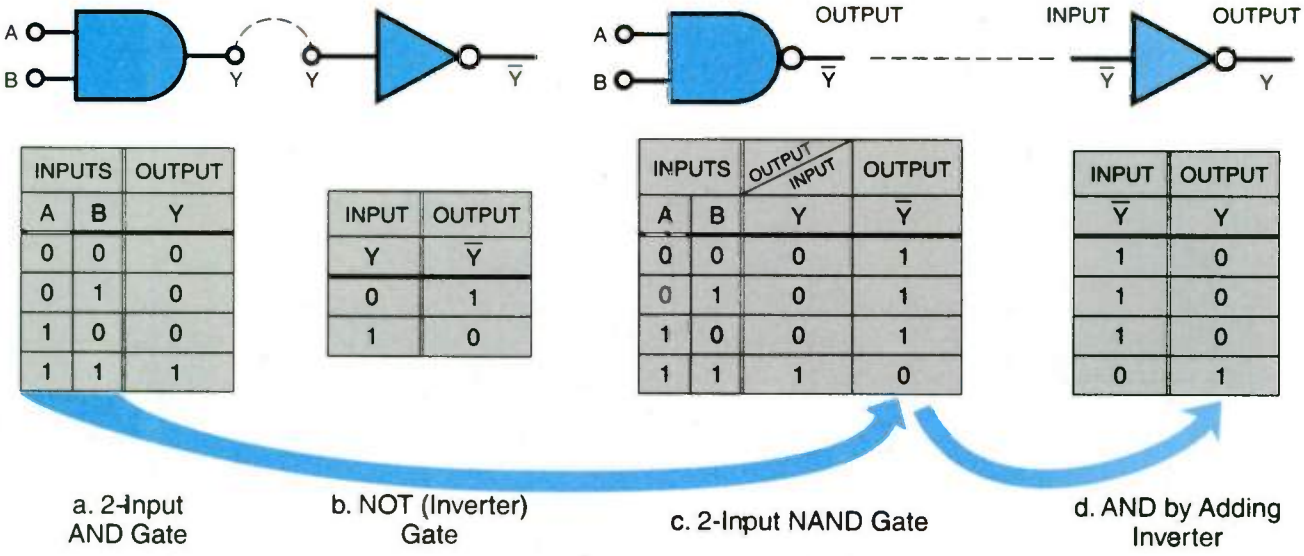
We have one condition left: A = 1 and B = 1. With both inputs at +5V, the base-collector diode of Q_1 becomes a forward-biased diode to turn ON Q_2 and Q_4 and turn OFF Q_3. Y is at a 0. Our truth table becomes:

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

In Chapter 2, *Figures 2-12a* and *2-12c,* we described an AND gate and an inverter and showed their truth tables. These are repeated in *Figure 3-6.* To match the truth table we developed for the TTL logic gate, an AND gate and an inverter need to be combined to form what is called a NAND gate (a **N**egative **AND** gate). The NAND gate has the same symbol as an AND gate, but it has a small circle on the output lead to indicate the inversion. So the TTL gate of *Figure 3-4* is a 2-input NAND logic gate. If both A an B are 1, there is an inverse of 1 (a 0) at the output.

In *Figure 3-6d,* an inverter has been added to the output of the TTL 2-input NAND, and an AND gate results. So if an AND gate is required in a digital logic circuit design, one way of obtaining one is to add an inverter to a NAND gate. This is actually what is done, but it is done inside the integrated circuit itself because, by doing so, it saves power, switching delays, and space. Look at Example 4. It shows the circuit of a 2-input AND gate. An additional inverter is placed between the input stage of *Figure 3-4b* and the totem-pole output.

AND Gate + Inverter = NAND Gate + Inverter = AND Gate



INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

a. 2-Input AND Gate

INPUT	OUTPUT
Y	$\overline{Y}$
0	1
1	0

b. NOT (Inverter) Gate

INPUTS		OUTPUT / INPUT	OUTPUT
A	B	Y	$\overline{Y}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

c. 2-Input NAND Gate

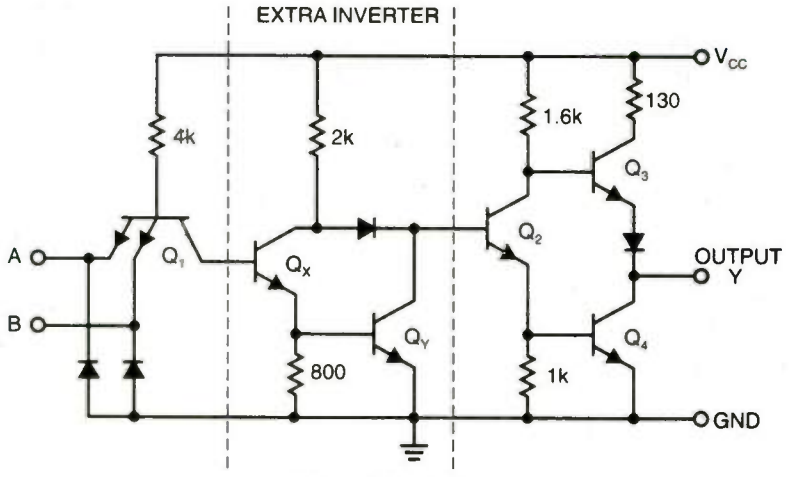
INPUT	OUTPUT
$\overline{Y}$	Y
1	0
1	0
1	0
0	1

d. AND by Adding Inverter

Figure 3-6. A NAND gate is an AND gate connected to an inverter. If an AND gate is required, another inverter can be added.

Example 4. Determining a Logic Circuit's Truth Table

Determine the truth table of the circuit shown.



2-Input AND Gate

INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

1. When V_{IN} = 0V on A or B, no matter what the other input(s) are, Q_1 is ON, Q_X is OFF, Q_Y is OFF, Q_2 is ON, Q_4 is ON, Q_3 is OFF and Y is 0.
2. When A and B are 1, Q_1 is OFF, Q_X is ON, Q_Y is ON, Q_2 is OFF, Q_4 is OFF, Q_3 is ON, and Y is 1.

The circuit is a 2-input AND gate.

An OR Gate — TTL Logic

So far we have shown how an inverter and an AND gate can be made with digital electronic circuits, and specifically with TTL. We need the missing link — an OR gate.

The TTL circuit should have the same output drive so that the OR gate can be easily coupled to the other TTL gates; therefore, in our OR circuit, we will use the same totem-pole output as the NAND TTL gate; we will just change the input stage. The circuit is shown in *Figure 3-7.* Let's develop the truth table from what we know about the operation of a TTL circuit.

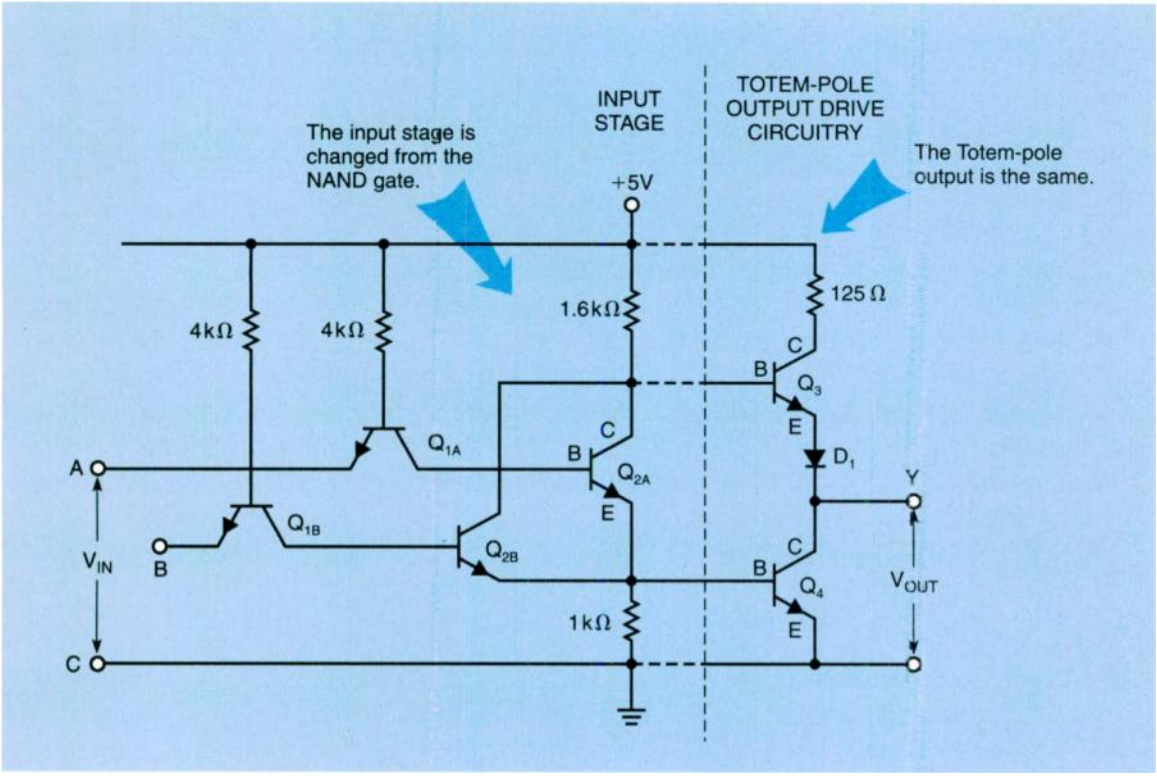


Figure 3-7. A TTL NOR gate has the same totem-pole output, but a different input stage than a TTL NAND gate.

What Kind of Logic Gate

Start with V_{IN} = 0V, a logic 0 level, on A. Q_{1A} will be ON to hold Q_{2A} OFF, Q_4 OFF, and Q_3 ON. Y will be a 1.

INPUTS		OUTPUT
A	B	Y
0		1

If B is also a 0, Q_{1B} is ON an Q_{2B} is OFF, and the same condition exists for Y; it is a 1.

INPUTS		OUTPUT
A	B	Y
0	0	1

The condition is different, however, if B is a 1 (V_{IN} = +5V). With B a 1, and A still a 0, the base-collector of Q_{1B} becomes forward-biased to turn ON Q_{2B} and Q_4 and turn OFF Q_3. Y is a 0.

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	0

If A and B conditions are reversed, A = 1 and B = 0, nothing changes at the output from when A = 0 and B = 1. The base-collector of Q_{1A} is now forward-biased to turn on Q_{2A} and Q_4 and turn OFF Q_3. Y is a 0.

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	0
1	0	0

The remaining condition is A = 1 and B = 1. Both Q_{2A} and Q_{2B} are ON, Q_4 is ON and Q_3 is OFF. Y is 0. The complete truth table is:

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Figure 3-8 repeats the OR gate and the inverter and their truth tables from *Figures 2-12b* and *2-12c*. Again, as in the NAND gate, an OR gate and an inverter must be combined to match the truth table for the circuit in *Figure 3-7*. In the combined form, it is called a NOR gate (a **N**egative **OR** gate). The NOR gate has the same symbol as an OR gate except it has a small circle on the output lead to indicate the inversion. Our circuit of *Figure 3-7* is a 2-input NOR logic gate. If A OR B is a 1, OR if both are a 1, there is an inverse of 1 at the output.

Another inverter can be added to the NOR gate *(Figure 3-8d)* to end up with an OR gate. Like the NAND gate, the addition of an inverter to make an OR gate from a NOR gate is done inside an integrated circuit to save power, increase switching time, and save space.

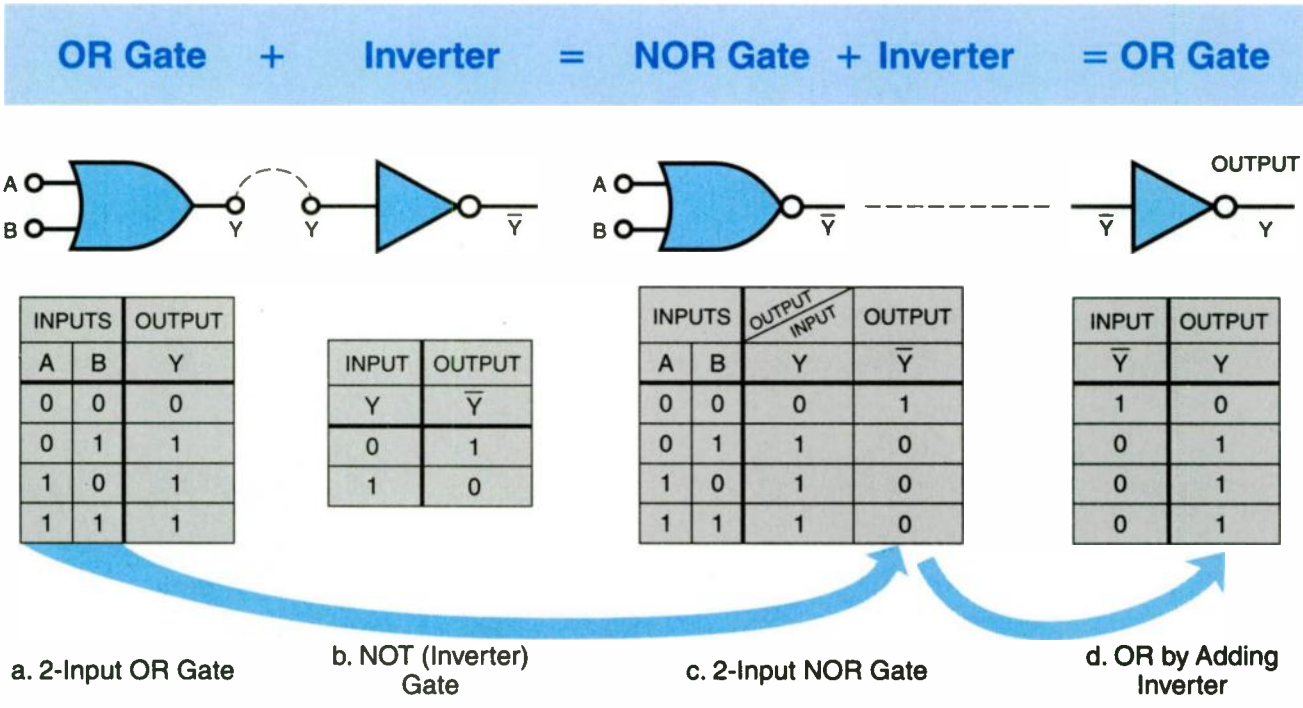


Figure 3-8. A NOR gate is an OR gate connected to an inverter. If an OR gate is required another inverter can be added.

TTL Logic Circuits a Winner

Since the original version of TTL logic circuits in 1965, TTL has been the leading digital electronic logic circuit designed into digital systems worldwide. Variations from the original standard family include low-power, high-power, and high-speed families. One of the major improvements, as shown in *Figure 3-9a,* has been the inclusion of a *Schottky* diode in parallel with the base-collector junction of the transistors. As we have stated many times, transistors are driven into saturation when turned ON, and the excess charge carriers in the base must be removed before the transistor can be turned OFF. The time required to remove these carriers, called *storage time*, is responsible for the time delay that slows down the switching speed of the gate. Because the forward voltage drop of the Schottky diode is less than that of the base-collector of the transistor, the transistor's junction cannot become heavily forward biased so it is kept out of heavy saturation. When the base current is diverted around the transistor's base-collector junction through the Schottky diode, the junction is said to be *clamped* to the diode's voltage. As a result, the transistor can be switched rapidly without storage-time delays. The equivalent symbol for a Schottky-clamped NPN transistor is shown in *Figure 3-9b.* Additional logic families using Schottky transistors — low-power, advanced Schottky, and advanced low-power Schottky — have resulted.

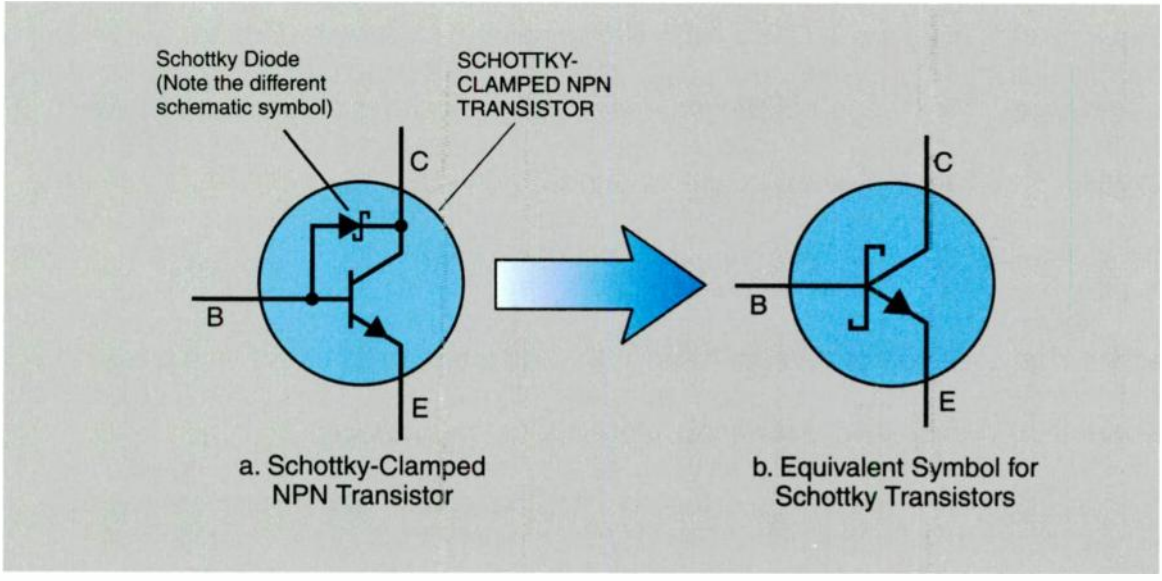


Figure 3-9. A Schottky-clamped NPN transistor and its equivalent symbol.

Field-Effect Transistors — MOS Logic

Digital systems continue to demand that logic gates be packed into smaller and smaller space. As circuit densities increase, electron devices must be found that operate at lower and lower power, but are still able to turn ON and OFF at competitive speeds. Such a device is the field-effect transistor (FET). We will be concerned mostly with a MOS (metal-oxide-semiconductor) field-effect device because it has become the device of choice for logic circuit designers because of the reasons stated above. How does a field-effect transistor work?

How a Field-Effect Transistor Works

Depletion Mode

We are going to use water flowing in a flexible plastic tube to help understand how a field-effect transistor works. Look at *Figure 3-10a.* Here water is flowing at its maximum rate through a flexible plastic tube into a container and the container fills in one minute. If the tube is pinched between the fingers as in *Figure 3-10b,* the water flow is restricted and it takes five minutes to fill the container. If the tube is pinched hard enough, there will be no flow because the flow is pinched off. We have depleted the flow as the tube is pinched off. We call this the *depletion mode.*

Enhancement Mode

In *Figure 3-11a,* we start with the flow pinched off. There is no flow. The pressure is released some in *Figure 3-11b,* there is a restricted flow and it takes five minutes to fill the container. All pinching is released in *Figure 3-11c,* there is maximum flow and the container fills in one minute. As the pinching is released, we have enhanced the water flow from being pinched off to maximum flow. We call this the *enhancement mode.*

MOS Transistor Action

The transistor in *Figure 3-12* is a MOS N-channel enhancement mode transistor. The transistor is made from semiconductor material separated so that a P-type is between two N-types. Over the top of the material is a very thin layer of silicon oxide, which is an insulator. On top of the silicon oxide is an aluminum metal plate isolated to just overlap the P-type material. There is no current from the metal plate to the semiconductor material because of the insulating silicon oxide. The transistor gets its MOS name because of the Metal-Oxide Semiconductor sandwich. The metal plate is called a *gate;* one N-type is called a *source* and the other N-type is called a *drain.*

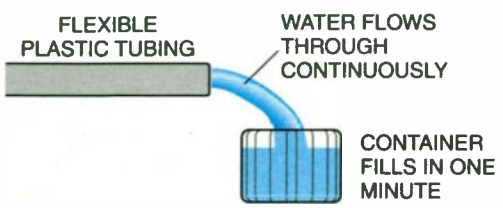
*Figure 3-12*a shows S_1 open so no voltage is applied to the gate, but +10V is applied between source and drain. Since a P-N junction must be forward biased to conduct current, there is no current from drain to source. It is just like the case of *Figure 3-11a* where all the current (water flow) is "pinched off."

When a voltage is applied to the gate (between gate and source), an electric field is set up across the silicon oxide insulator that creates an N-type channel in the P-type material below the plate. The N-type channel allows current between the N-type drain and N-type source. In *Figure 3-12b,* the current is restricted, but with the application of a higher voltage to the gate, as in *Figure 3-12c,* there is maximum current from drain to source — just like *Figure 3-11c* has maximum water flow. We see how the gate acts as a switch, and controls the current from drain to source from OFF to ON.

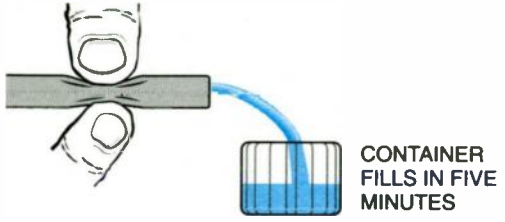
Types of MOS Transistors

There are many types of MOS transistors. Reversing the P and N-type in *Figure 3-12* makes the transistor a P-channel enhancement mode type. The operation is the same, but the voltages are reversed in polarity (a minus voltage on gate compared to source, and on the drain compared to source). Each N-channel and P-channel can be either an enhancement mode or a depletion mode device. Remember:

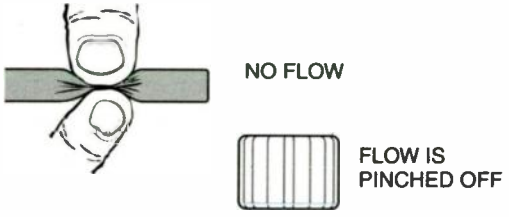
1. In a depletion mode MOS transistor, current is ON when the gate voltage is low, and OFF when the gate voltage is high.
2. In an enhancement mode MOS transistor, current is OFF when the gate voltage is low, and ON when the gate voltage is high.



a. Maximum Flow



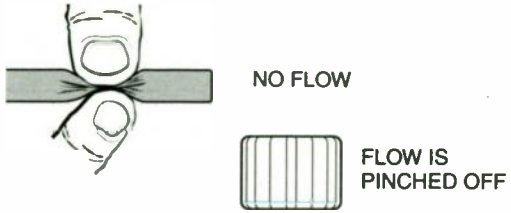
b. Restricted Flow
Due to Pinched Tube



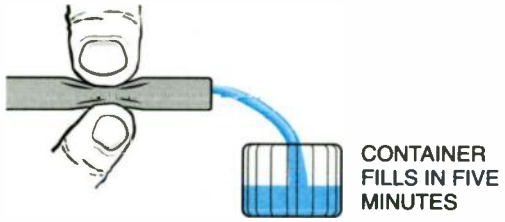
c. Flow Completely
Pinched OFF

I. Depletion Mode

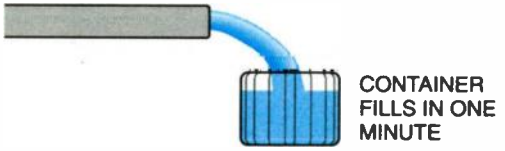
Figure 3-10. Example of the depletion mode operation of a field-effect transistor.



a. Flow Completely Pinched OFF



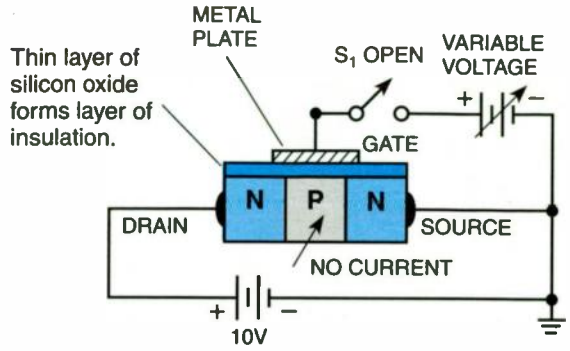
b. Restricted Flow
Due to Pinched Tube



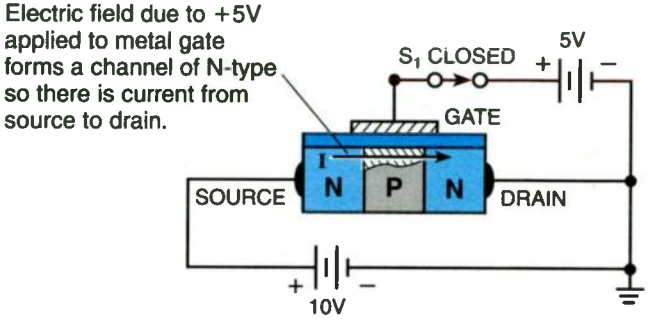
c. Maximum Flow

II. Enhanced Mode

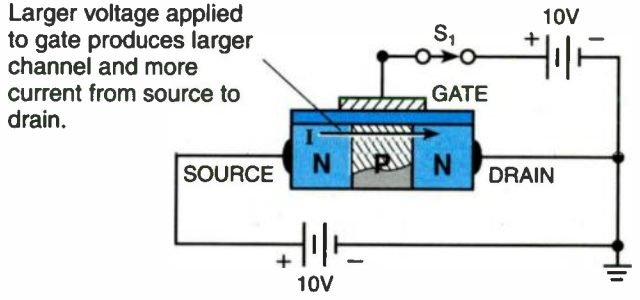
Figure 3-11. Example of the enhancement mode operation of a field-effect transistor.



a. Flow Completely Pinched OFF



b. Restricted Flow



c. Maximum Flow

III. Enhanced Mode MOS Transistor (N-Channel)

Figure 3-12. Construction and operation of an N-channel enhancement mode MOS transistor.

Figure 3-13 shows the schematic symbols for the four types of MOS transistors. Note the solid line from drain to source for the depletion mode and the dotted line between drain and source for the enhancement mode. Note the direction of the arrows to indicate the channel type.

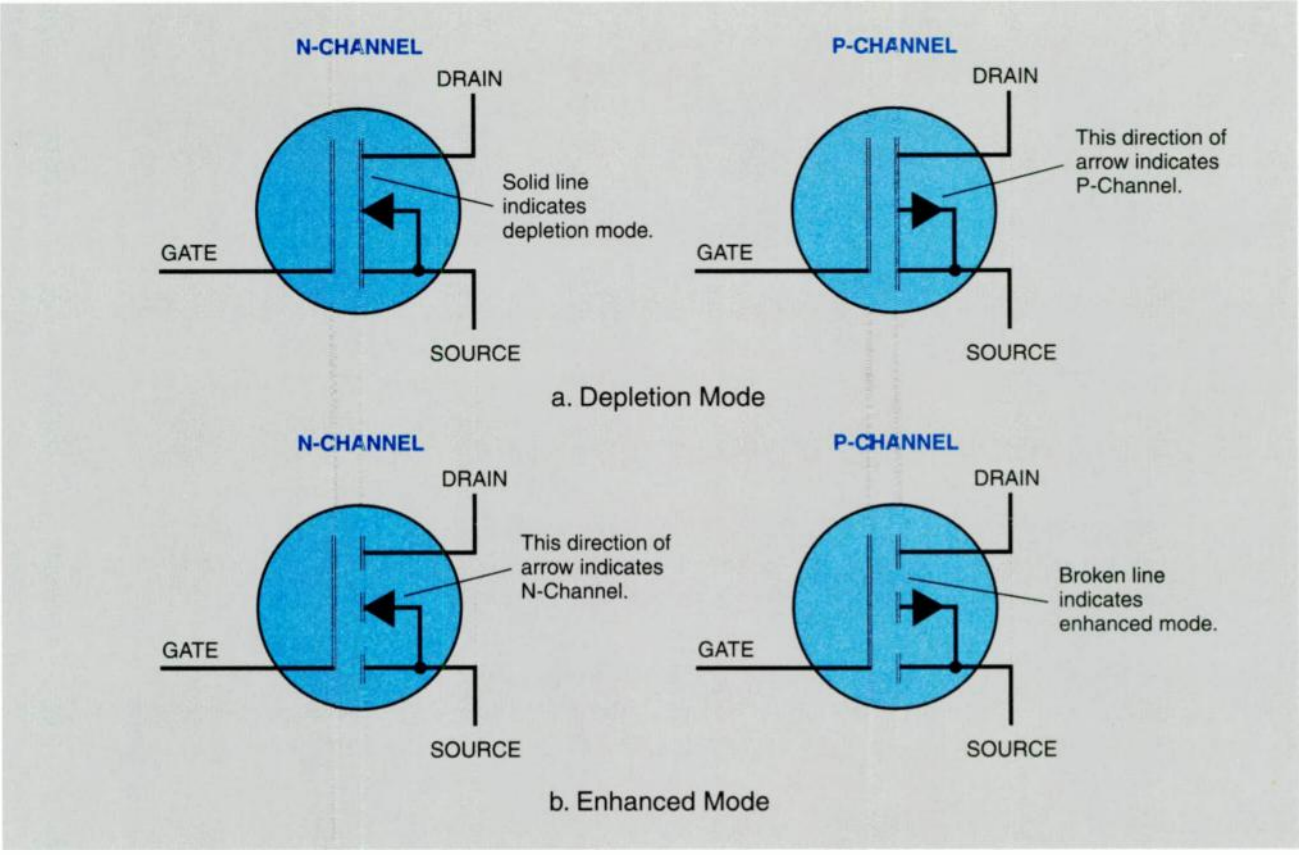


Figure 3-13. MOS field-effect transistor (MOSFET) schematic symbols.

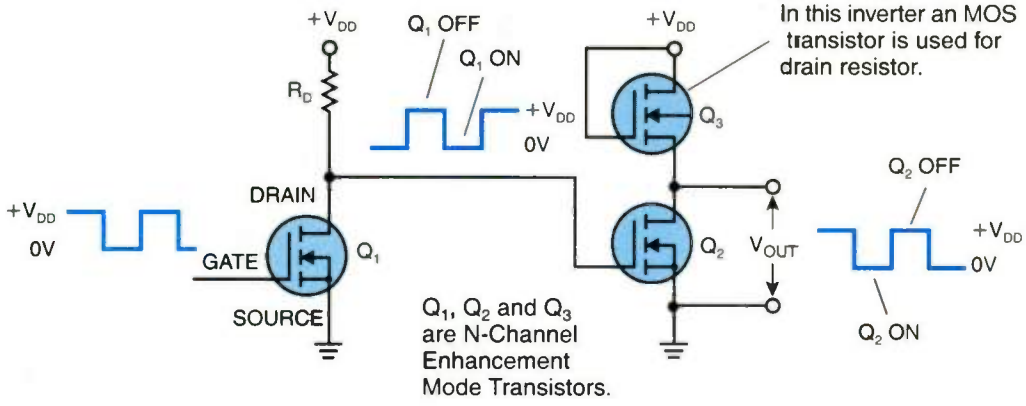
MOS Logic Gates

MOS transistors can be interconnected in the same fashion as bipolar transistors in order to form logic gates. *Figure 3-14a* shows circuits for an inverter. Trace through the circuit operation to assure yourself that when Q_1 is ON, Q_2 is OFF and vice versa. For N-channel MOS, V_{DD} is a plus voltage, in this case +10V. +10V on the gate turns the transistor ON. One significant advantage of MOS transistors for putting in integrated circuit form is the fact that a MOS transistor can be used as a drain load resistor. Q_3 is used this way in the second inverter of *Figure 3-14a.*

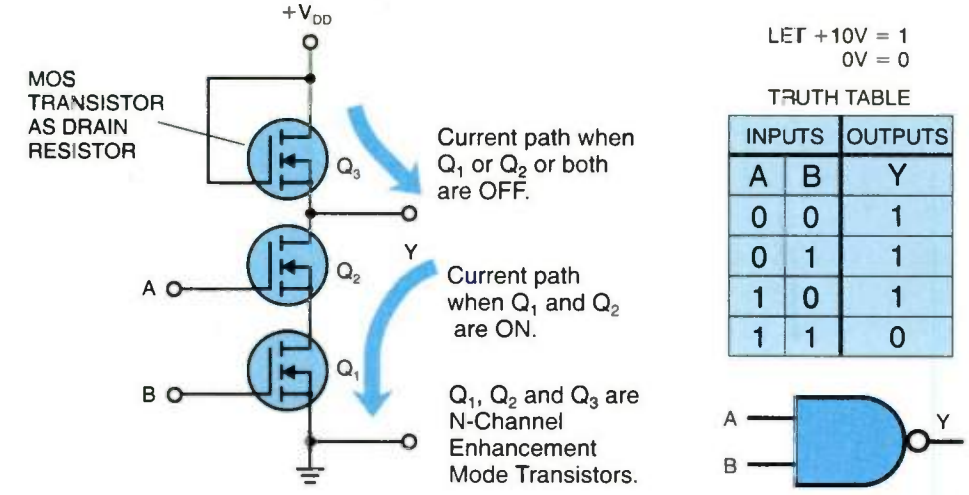
Figure 3-14b shows the circuit for a MOS 2-input NAND gate, and *Figure 3-14c* shows the circuit for a MOS 2-input NOR gate. Verify that the truth tables are correct when +10V = 1 logic level and 0V = 0 logic level. A 1 turns the transistor ON. Also, as with TTL, and when packed closely together in an integrated circuit, an extra inverter can be added to the NAND to get an AND gate, and to the NOR to get an OR gate.

CMOS Logic

Since P-channel and N-channel transistors require opposite polarity voltages to operate, it seems logical that they could be combined and operate together. That is exactly what occurs in Complementary Metal-Oxide Semiconductor (CMOS) circuits. *Figure 3-15a* shows a P-channel inverter in which the transistor is ON when the gate voltage is 0V, and OFF when gate voltage is $+V_{DD}$. The N-channel inverter has the

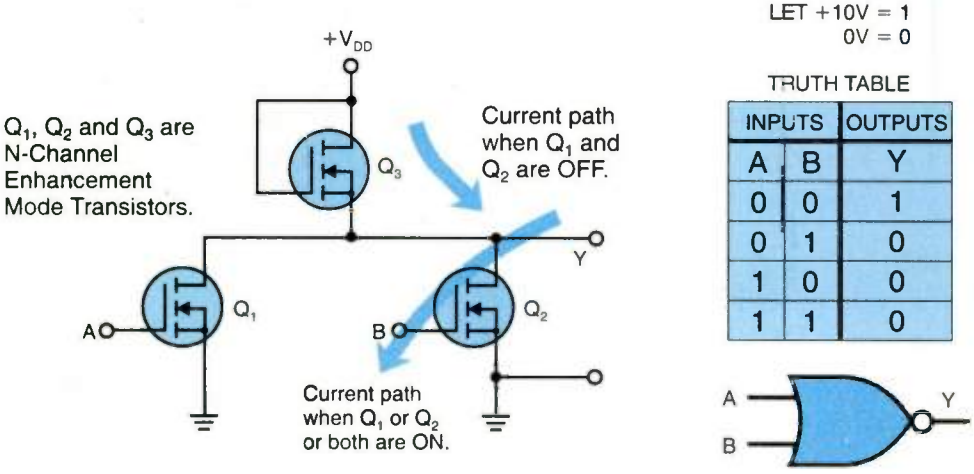


a. MOS Inverters (NOT Logic Gates)



INPUTS		OUTPUTS
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

b. MOS 2-Input NAND Gate



INPUTS		OUTPUTS
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

c. MOS 2-Input NOR Gate

Figure 3-14. MOS logic gates — NOT, NAND, NOR with truth tables and symbols.

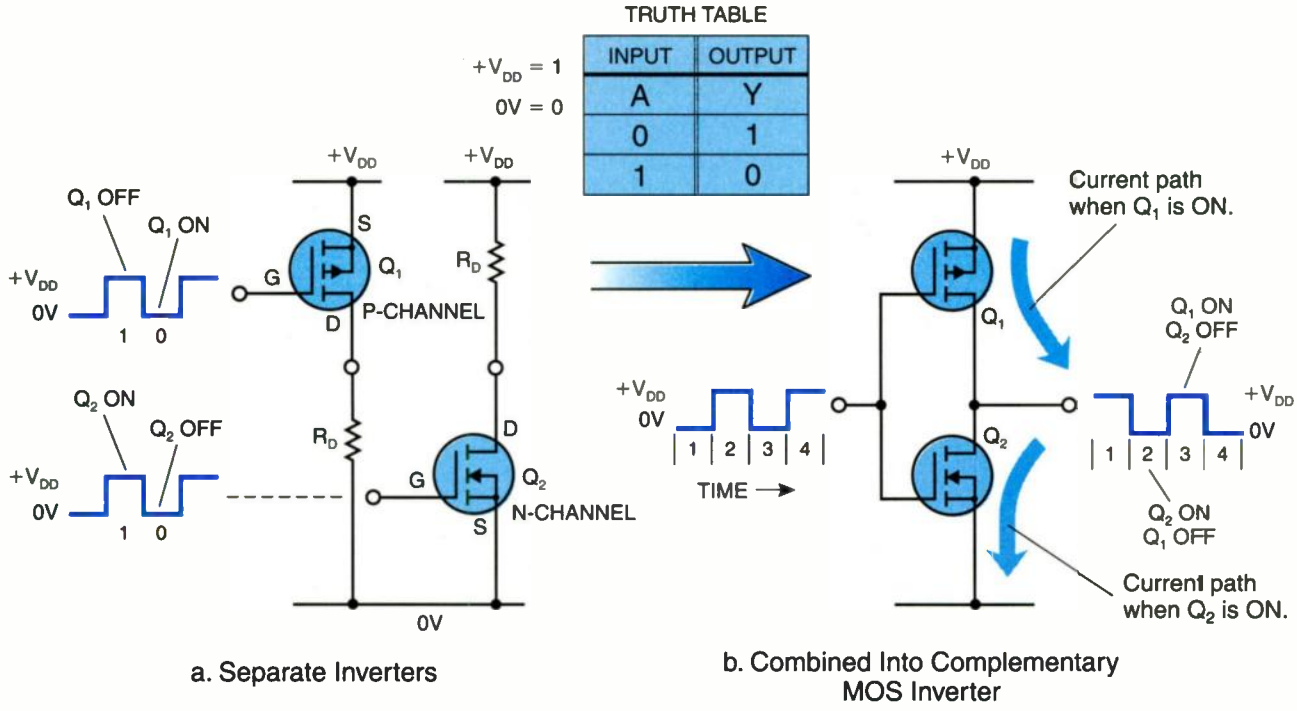


Figure 3-15. CMOS logic combines P-channel and N-channel MOS transistors. It operates with low power and has excellent drive when the output is at the 1 level, and excellent sink capability when the output is at the 0 level.

opposite conditions. Q_2 is ON when the gate voltage is $+V_{DD}$, and OFF when the gate voltage is 0V. Q_1 is ON when Q_2 is OFF, and Q_1 is OFF when Q_2 is ON. Therefore, coupling them together and throwing away the load resistor results in a complementary MOS logic inverter. Since Q_2 is OFF when Q_1 is ON, and vice versa, there is no power lost in the OFF transistor. All the current to the output is delivered through a low resistance ON transistor, both when supplying through Q_1 at the $+V_{DD}$ level or sinking current through Q_2 at the 0V level. It is just like the totem pole output of TTL. Thus, CMOS is used because it has fast switching and low standby power. The only time power is used is during the switching times. AND, OR and, as we have shown, a NOT gate can be constructed using CMOS circuitry just like those using MOS circuitry.

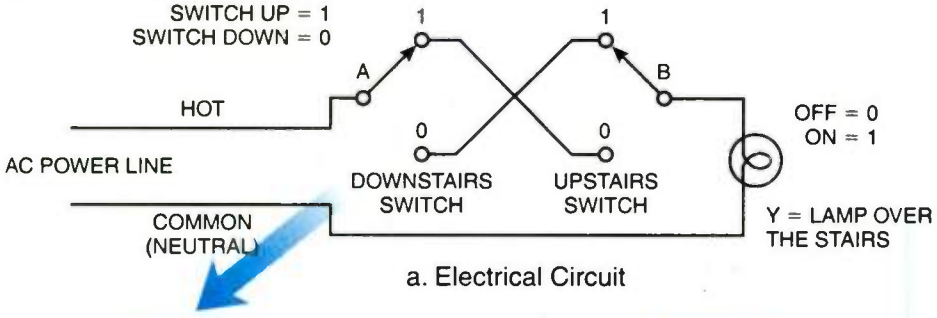
Exclusive OR and Exclusive NOR

Two other logic gates that are used a great deal by digital system designers are:

1. An OR gate, called an exclusive OR, that has special conditions applied to its truth table.
2. The inverse of the exclusive OR, called an exclusive NOR.

You probably use an exclusive OR circuit and don't even realize it. It is a 3-way light switch that controls the lights from the ends of a room, or from inside and outside the house, etc. One for upstairs/downstairs control is shown in *Figure 3-16a*. The downstairs switch is identified as input A and the upstairs switch as input B. When the switches are UP they input a 1; when they are DOWN, they input a 0. When the lamp over the stairs is ON, the output Y is a 1; when the lamp is OFF, the output Y is a 0. Note that when A OR B is a 1, Y is a 1, just like a standard OR gate. However, when both A and B are a 1, the output is 0, which is different from a

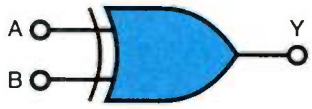
standard OR gate. This logic gate is called an XOR (e**X**clusive **OR**) because some standard OR conditions are excluded. The truth table and symbol are shown in *Figures 3-16b* and *c*. Building in an inversion in the gate produces an XNOR logic gate. The extension of the truth table and the schematic symbol for a XNOR are shown in *Figure 3-16d*.



a. Electrical Circuit

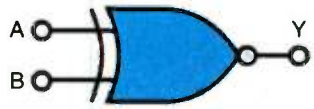
INPUTS		OUTPUT	XOR OUTPUT
B	A	LAMP	Y
1	1	OFF	0
0	1	ON	1
1	0	ON	1
1	1	OFF	0

b. Truth Table



c. XOR Schematic Symbol

XNOR OUTPUT
Y
1
0
0
1

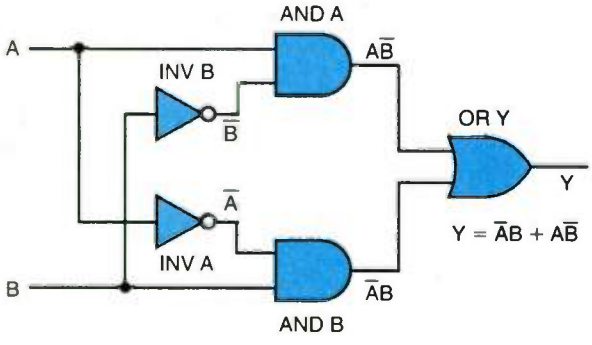


d. XNOR Extension

Figure 3-16. A 3-way switch for electric lights is an example of an eXclusive OR logic gate. An adder inverter provides an XNOR.

Example 5. Combining Gates to Form XOR Gate

Here is an example of using AND, OR and NOT logic gates to form another kind of gate. Prove to yourself that this gate satisfies the truth table of an XOR gate. Use 1s and 0s as the logic levels.



INPUTS		OUTPUT
A	B	Y
0	0	0
1	0	1
0	1	1
1	1	0

1. When A=0 and B=0, neither AND gate will be active, so the OR gate has two 0s at its input and the output Y=0.
2. When A=1 and B=0, AND A has both inputs (A AND $\overline{B}$) equal to 1 so its output is a 1. With a 1 to the OR gate input, Y=1.
3. When A=0 and B=1, now AND B has both inputs (B AND $\overline{A}$) equal to 1 so its output is a 1. With a 1 to the OR gate, input Y=1.
4. When A=1 and B=1, the inverter input to each AND gate has a 0, thus, each AND gate has a 0 output to the OR gate. Therefore, Y=0. The gate is an XOR gate.

Boolean Algebra

In Example 5, the output Y is expressed in an equation, $Y = \overline{A}B + A\overline{B}$. This is what is called a Boolean Algebra equation that expresses when Y is "true" or a 1. It says, "Y=1 when A=0 AND B=1 OR when A=1 AND B=0." Technically, there should be a dot between the AB terms to indicate the AND function, but in practice it is usually omitted. The plus sign (+) means an OR function. Boolean Algebra and its rules are beyond the scope of this book, but for an introduction, here are some common Boolean Algebra equations:

Gate		Boolean Equation
3-input AND	$Y = A \bullet B \bullet C$	Y = 1 when A AND B AND C = 1
3-input NAND	$Y = \overline{A \bullet B \bullet C}$	Y = the inverse of A AND B AND C = 1
3-input OR	$Y = A+B+C$	Y = 1 when A OR B OR C (OR comb) = 1
3-input NOR	$Y = \overline{A+B+C}$	Y = the inverse of A OR B OR C (OR comb) = 1
NOT (inverter)	$\overline{Y} = Y$	Y = the inverse of the input
XOR	$Y = A \oplus B$	Y = 1 when A=1 OR B=1, not when both = 1 or 0
XNOR	$Y = \overline{A \oplus B}$	Y = the inverse of $A \oplus B$

Positive and Negative Logic

Another subject that you need to be aware of, especially if you are going to continue your study of digital systems, is that there are two ways of expressing logic truth tables — as positive logic or as negative logic. We have seen that in digital systems two voltage levels represent the two binary digits 0 and 1. In all of our previous discussions and examples, the logic levels were defined so that the most positive (HIGH) voltage corresponding to logical 1 and that the least positive (LOW) voltage corresponded to logical 0. This logic level assignment is called *positive logic*. The assignment of logical 0 to the most positive voltage level and 1 to the least positive is called *negative logic*. Regardless of actual voltage levels, a *voltage truth table* that represents an AND gate under positive logic always becomes an OR gate under negative logic. Also, a positive-logic NAND gate always becomes a negative-logic NOR gate and vice versa. To translate a positive-logic truth table into a negative-logic truth table, just replace every 0 by a 1 and every 1 by a 0 in *both* the input and the output columns of the truth table. *Figure 3-17* illustrates the definition of both positive-logic and negative-logic with both positive and negative voltage levels.

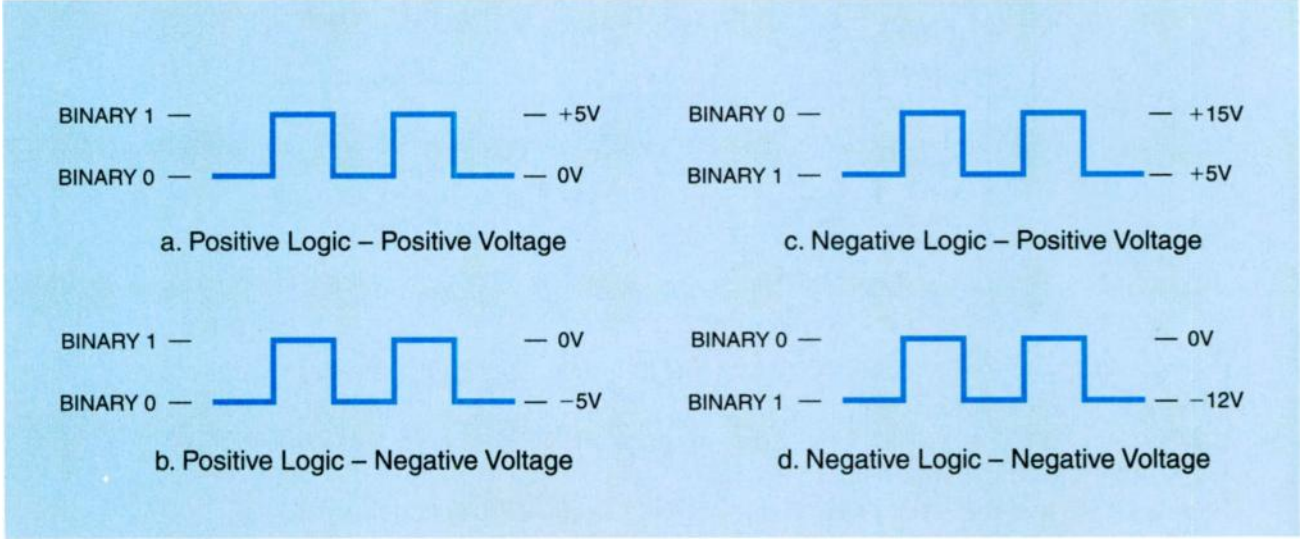


Figure 3-17. An illustration of both positive logic and negative logic with both positive and negative voltage.

A Data Selector

Now that the AND, OR, NOT, NAND and NOR logic circuits are understood, circuits which include more mixtures of these can be formed to make what are known as *combinational* circuits. Combinational circuits always produce the same logic level outputs for a unique combination of logic levels at the inputs.

Example 6 shows a combinational logic circuit that is quite useful. It is called a data selector or multiplexer (MUX). As we pointed out in Chapter 2, such circuits are used to route digital information over selected paths. In digital systems the name "bus" is given to the bundled wires that carry the many bits of a digital code in parallel (at the same time). Data selectors are used to switch digital codes coming in on a common bus to selected output buses. As shown, the output bus is selected by the logic level on a control line.

The data selector of Example 6 is built from AND gates. SN7408 ICs (Quad 2-input AND gates) are used to switch input logic levels on d_3, d_2, d_1, and d_0 to the corresponding bit line for bus A or bus B. The A or B output bus is determined by a control signal of 1 on the respective control line. S_1 is a DPDT switch wired to apply +5V on the control line of the chosen bus and ground on the one that is not selected. Similar kinds of data selectors can be designed that have multiple bus inputs and select the input bus to be put onto one output bus. The principles are the same as for the data selector that we have shown here.

Summary

We have looked at logic gates, the techniques used to build them, and showed examples of how the gates are combined to form combinational digital circuits to perform specific functions. In the next chapter, we'll see how these gates are put together to form sequential digital circuits.

c. NOT
d. NOR

d. NOT

Answers:
1b, 2d, 3a, 4c, 5b, 6b, 7d, 8a, 9c, 10c, 11b,
12c, 13a, 14d, 15b

What Is a Storage Circuit?

Digital circuits can be divided into two types: *combinational* and *sequential.* The digital circuits we described in Chapter 3 are *combinational* logic circuits. Their output at any instant of time depends on the combination of 0s and 1s on the inputs. The combinational AND gate, OR gate, NAND gate, and NOR gate circuits are used to design and construct more complicated combinational circuits, but the complicated circuits still satisfy the definition of a combinational circuit — at any time the output depends on the combination of the states of the inputs. *Sequential* circuits are distinctly different. They are bistable and the output, either a 0 or a 1, depends on the inputs just before the application of the next input. Thus, sequential circuits have the ability to "remember" the order, or sequence, of the applied inputs. This "memory" capability; that is, the ability to remain in one of two stable states (bistable) after application of inputs, is the special feature that identifies sequential circuits.

Basic examples of sequential circuits are the *latch* and the *flip-flop.* As we shall see, they are very similar to each other. Combinational logic circuits can be connected to form bistable sequential circuits. Let's use the latch to demonstrate how this is done, and how the latch sequential circuit works.

The Latch

The term latch is used because it describes a logic function that is similar to the action of a mechanical latch. When a mechanical latch is moved to one of its two stable positions, it will stay in that position until it is *forced* into its other stable position. The latch stays in the second stable position until it is *forced* back to the first stable position. An *external force* must be applied to make it change positions.

The digital sequential circuit latch we will discuss first is the NAND gate S-R latch shown in *Figure 4-1.* S means SET; R means RESET. LOWs on the SET or RESET are activating inputs that change the state (position) of the bistable latch. Notice that the NAND S-R latch has two inputs brought out on its left side — one called $\overline{\text{SET}}$ and the other called $\overline{\text{RESET}}$. $\overline{\text{SET}}$ and $\overline{\text{RESET}}$ indicate that the inputs will activate the circuit when a 0 (LOW) signal is applied. Each input is normally connected to a 1 (HIGH), but

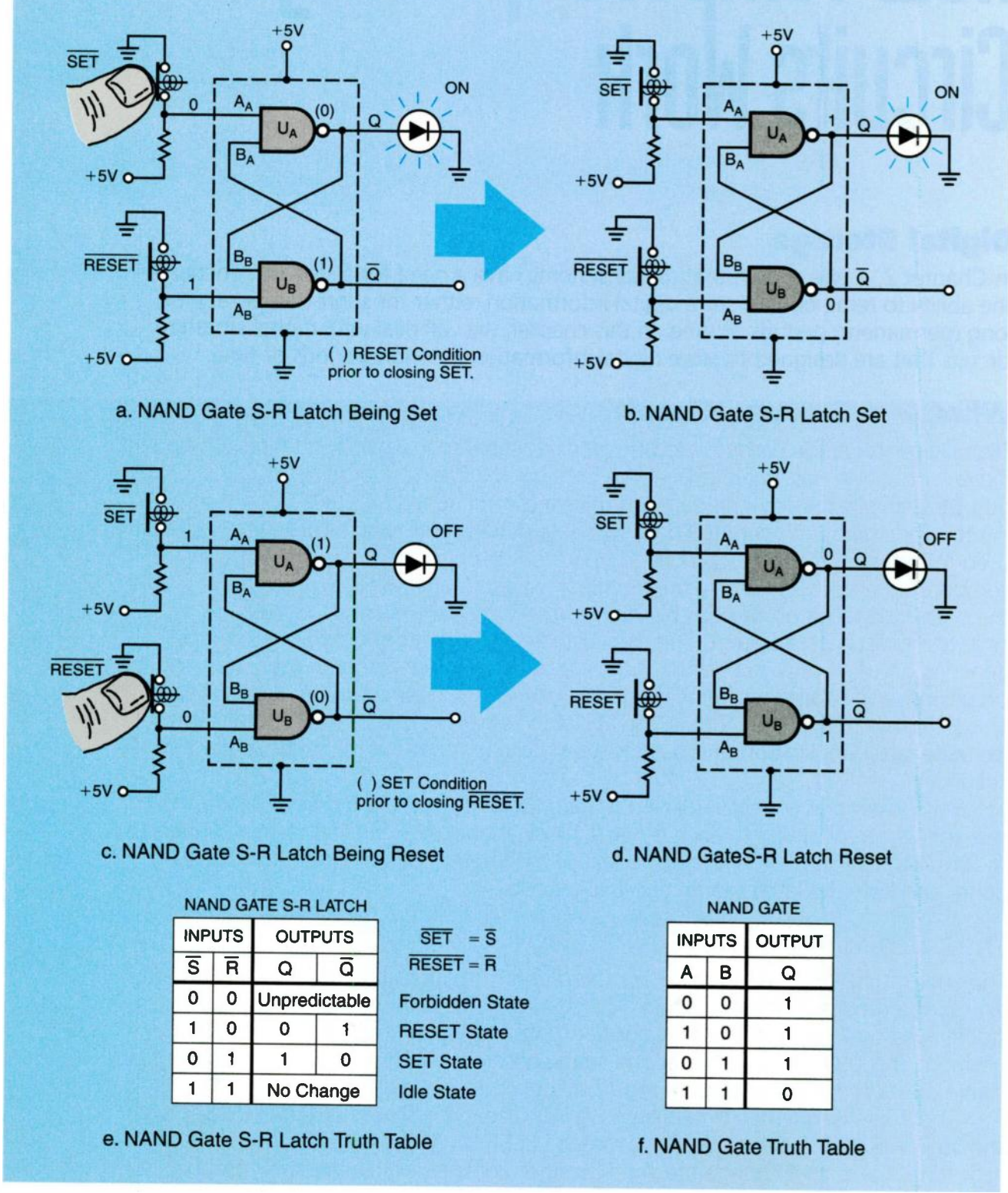
Example 6. Analyzing a Data Selector

AND gates are used to build the data selector shown. The objective is to transfer the

Questions and Problems for Chapter 3

1. Calculate the current through the NPN collector resistor and the V_{CE} output voltage in *Figure 3-2a* for each input logic level. Assign a value of 5 kilohms to R_1 and +5V to V_{CC}.
2. Calculate the current through the PNP collector resistor and the V_{CE} output voltage in *Figure 3-2b* for each input logic level. Assign a value of 1 kilohms to R_1 and −5V to V_{CC}.
3. Using *Figure 3-3a* with R = 1.5 kΩ, determine the noise margin of the voltage level that turns ON inverter #2 when inverter #1 is OFF. Assume it takes 1.2 mA to turn on inverter #2.
4. In *Figure 3-4a,* the input voltage, V_{IN} = 0.5V, to input A. Is this enough to keep Q_1 from transistor action when Q_2 is ON?
5. If the two inputs of a NAND gate are tied together, describe the resulting truth table.
6. If the two inputs of an OR gate are tied together, describe the resulting truth table.
7. Describe the two modes of operation of a MOSFET.
8. Describe the characteristics of CMOS logic.
9. If −10V is defined as a HIGH, and +5V is defined as a LOW in a system, the system is said to use

when the button is pushed on its normally-open switch, the input is connected to ground so that to a 0 (LOW) logic level is applied to that input as long as the switch button is held down. When the button is released, the switch opens and the input returns to a 1 (HIGH).



NAND GATE S-R LATCH

INPUTS		OUTPUTS	
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$
0	0	Unpredictable	
1	0	0	1
0	1	1	0
1	1	No Change	

$\overline{SET} = \overline{S}$
$\overline{RESET} = \overline{R}$

Forbidden State
RESET State
SET State
Idle State

e. NAND Gate S-R Latch Truth Table

NAND GATE

INPUTS		OUTPUT
A	B	Q
0	0	1
1	0	1
0	1	1
1	1	0

f. NAND Gate Truth Table

Figure 4-1. A NAND gate S-R latch can be SET or RESET into either of its bistable states.

Inside the latch are two 2-input NAND gates that have their gate outputs cross-connected to the opposite gate's input. The second input to NAND gate U_A is the $\overline{SET}$ input, and the second input to NAND gate U_B is the $\overline{RESET}$ input. The cross coupling makes the outputs depend on the inputs, and the inputs depend on the outputs. The latch has two stable states, SET and RESET. Let's assume the latch in *Figure 4-1a* is in the RESET state before the $\overline{SET}$ switch is closed. This means that Q = 0 (LOW), and $\overline{Q}$ = 1 (HIGH). Both pushbutton switches are open, so $\overline{SET}$ and $\overline{RESET}$ are 1 (HIGH). Since $\overline{Q}$ is 1 (HIGH), the two inputs of NAND gate U_A are at 1 and its output Q is at 0 (as given in a NAND gate's truth table). With Q = 0, NAND gate U_B has its A input = 1 and its B input = 0, therefore, its output is a 1 (as given in a NAND gate's truth table). The circuit is in the RESET stable state. It stays in this RESET state until forced by input signals to switch to the SET state.

Setting the Latch

When the $\overline{SET}$ button is pushed, it puts a momentary 0 (LOW) on the A input of NAND gate U_A. From the truth table, this causes output Q = 1 (HIGH), which forces $\overline{Q}$ = 0 (LOW) since now A_B = 1 and B_B = 1, and sets the latch in its SET stable state (Q = 1 and $\overline{Q}$ = 0). With Q = 1, the LED on the Q output will be lit.

In *Figure 4-1b,* we have removed the LOW on the $\overline{SET}$ input of gate U_A by removing our finger from the pushbutton. This does not change the output of gate U_A because the feedback connection from $\overline{Q}$ keeps the B input at 0 and with a 0 and a 1 at the U_A gate inputs, its output Q = 1. The latch stays in its SET stable state. Thus, the *definition* for the SET state is that $\overline{SET}$ = *0 sets Q = 1.* Now you see the purpose of the $\overline{SET}$ notation. The 0 (LOW) that was momentarily applied to the $\overline{SET}$ input has *latched* the circuit into the SET state.

Example 1. Determining the RESET State

Use *Figure 4-1c* to show that a $\overline{RESET}$ = 0 resets Q to a 0. The *definition* of the RESET state is that $\overline{RESET}$ = *0 resets Q = 0* or $\overline{Q}$ = 1.

The solution begins by starting with Q = 1 (HIGH) and the input button switches open; the respective inputs = 1 (HIGH). The inputs to gate U_B are A_B = 1 and B_B = 1 and output $\overline{Q}$ = 0. Pushing the RESET button switch forces U_B gate input A_B momentarily to 0 (LOW). This forces the output $\overline{Q}$ to a 1 (HIGH). U_A gate now has 1s on both inputs so its output Q = 0 (LOW) (i.e., the latch RESETs.) In *Figure 4-1d*, with the $\overline{RESET}$ button released, the latch remains in the RESET state because the feedback connection from Q = 0 to input B_B of gate U_B keeps the output $\overline{Q}$ = 1. The LED on the Q output will not be lit.

NAND Gate S-R Latch Review

Notice these conditions about the latch in *Figure 4-1:*

1. The RESET state of *Figure 4-1d* is where we initially started the discussion of the latch circuit.
2. The outputs are always complements of each other: Q = 1 and $\overline{Q}$ = 0, or Q = 0 and $\overline{Q}$ = 1.
3. LOW or 0 is the active low input that forces the latch from one stable state to the other.
4. The active LOW input latch does not allow the condition where both $\overline{SET}$ = 0 and $\overline{RESET}$ = 0 at the same time. This is a *not allowed or forbidden condition.* If these inputs are applied, the state of the latch is unpredictable.
5. When both inputs are 1s or HIGH, there is *no change* in the latch state.

The active LOW NAND S-R latch truth table is shown in *Figure 4-1e,* and a NAND gate truth table in *Figure 4-1f.*

NOR Gate S-R Latch

To help with your understanding of a latch, you might want to substitute NOR gates and a NOR gate truth table and analyze the SET and RESET states of the latch. Study, in particular, the input signals and feedback sequences that force the latch from one stable state to the other. The NOR gate S-R latch has the truth table shown in *Figure 4-2*.

Burglar Alarm

Figure 4-2 shows a burglar alarm circuit that can be constructed using a NOR gate S-R latch. If S_1 is opened by opening the window or door, then the latch will SET and turn on the alarm. To stop the alarm requires that a key or digital code close switch S_2 after S_1 is closed. When the system is RESET, the LED lights to indicate that the system is armed.

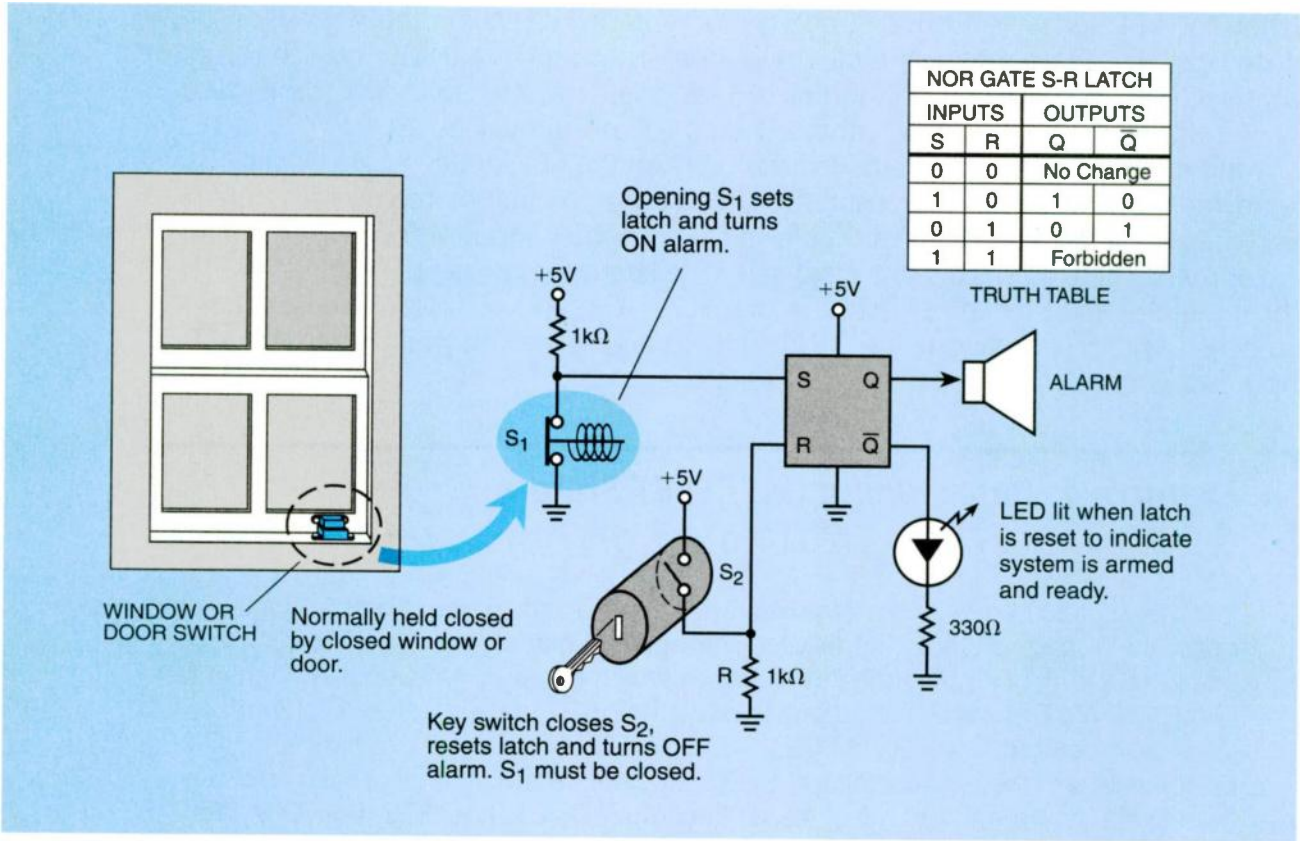


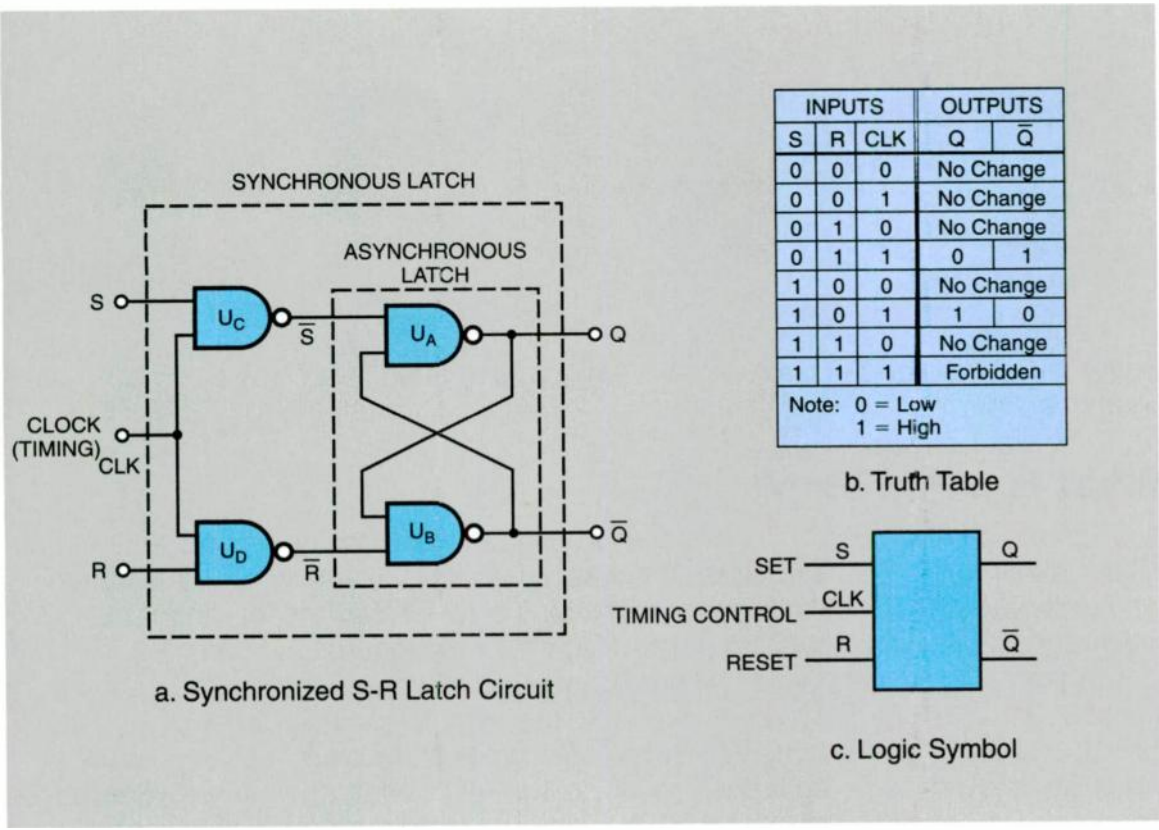
Figure 4-2. Burglar alarm using a NOR gate S-R latch.

The Clocked S-R Latch

Recall in Chapter 2 we discussed the importance of timing. The NAND gate S-R latch just discussed has no synchronization. We say that the S-R latch is an *asynchronous* device because it does not operate under the control of a timing signal called the clock.

The S-R latch of *Figure 4-1* can be made synchronous by adding two 2-input NAND gates as shown in *Figure 4-3*. The outputs of the NAND gates U_C and U_D are connected to the $\overline{S}$ ($\overline{SET}$) and $\overline{R}$ ($\overline{RESET}$) inputs, respectively, of the asynchronous latch. One of the two inputs of U_C becomes the S (SET) input, and one of the two inputs of U_D becomes the R (RESET) input, of the synchronous latch. The remaining inputs of U_C and U_D are wired together and become the clock (timing) input. All four gates combined are a synchronous latch which is normally called a *clocked flip-flop*.

The S and R inputs control the state of the latch, but the control is dependent on the state of the clock input. Only *when a 1 (HIGH) logic level is present at the clock input* will the signals on S and R control the latch. Referring to the truth table of *Figure 4-3b,* the outputs of the latch may not change unless the clock input (CLK) is a 1 (HIGH); therefore, the state of the latch outputs not only depend on the inputs S and R, but also will not change until the CLK signal is HIGH. Note also that the inverting effect of gates U_C and U_D makes their SET and RESET inputs active HIGH instead of active LOW; therefore, they are identified as S and R, respectively, rather than $\overline{S}$ and $\overline{R}$.



INPUTS			OUTPUTS	
S	R	CLK	Q	$\overline{Q}$
0	0	0	No Change	
0	0	1	No Change	
0	1	0	No Change	
0	1	1	0	1
1	0	0	No Change	
1	0	1	1	0
1	1	0	No Change	
1	1	1	Forbidden	

Note: 0 = Low
1 = High

b. Truth Table

Figure 4-3. A synchronized S-R latch can be designed by adding two NAND gates to an asynchronous NAND gate S-R latch.

The Gated D Latch

We can modify the clocked S-R latch to create a gated D latch by adding an inverter as shown in *Figure 4-4. Gated* means it is a synchronous circuit. Notice that the D latch has only one input besides the clock input (CK). The SET input of the clocked S-R latch becomes a D (D for "data") input. It does not have an external RESET input. The RESET input is derived from the SET input of the clocked S-R latch through the inverter. When the D input is a 1 (HIGH) and the CK input is a 1 (HIGH), the D latch will SET so Q = 1 (HIGH). When the D input is a 0 (LOW) and the CK input is a 1 (HIGH), the D latch will RESET so Q = 0 (LOW). We can say that the Q output "follows" the D input when CK is a 1 (HIGH). The clock is normally at the 0 (LOW) level; thus, data is "gated" (allowed to pass) to the output only when the clock is at a 1 (HIGH) level. The logic symbol is shown in *Figure 4-4c.* Many times only the Q output is provided. Also, notice that the clock input is identified as CK, while in *Figure 4-3,* it was CLK. Both of these notations are used in digital systems.

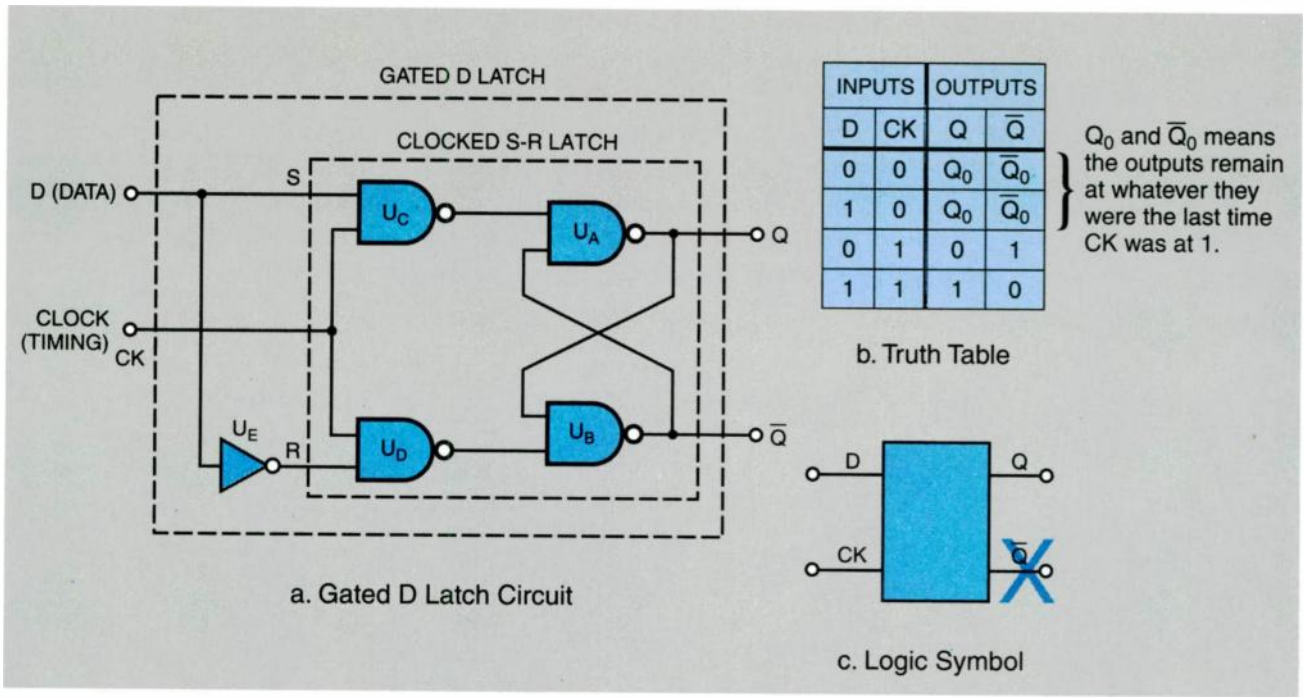


Figure 4-4. A clocked S-R latch can be changed to a gated D latch by connecting an additional inverter.

What Is a Flip-Flop?

A flip-flop is a circuit which has two stable states; it "flips" to the second state or "flops" back to the first state when an input signal activates the circuit. A latch is a type of flip-flop. The usual difference between the two is that the flip-flop is synchronous, or clocked, while the latch may be asynchronous.

In the gated D latch, which is commonly called a D flip-flop, the flip-flop input signals are enabled by the level of the clock. The *edge-triggered flip-flop* is a *unique type of flip-flop* that is ***enabled by the clock signal transition.*** It changes state either on the positive (rising) edge (from 0 to 1 level) or on the negative (falling) edge (from 1 to 0 level) of the clock signal. The edge-triggered flip-flop is a very popular sequential circuit for digital systems. It is used in shift registers, data registers, counters, memories, and many other sequential digital circuits.

Edge-Triggered S-R Flip-Flop

The two basic kinds of edge-triggered flip-flops are positive edge-triggered and negative edge-triggered. The triggering transition on the edges of the clock pulses is shown in *Figure 4-5b* and *Figure 4-5c* compared to level-triggering in *Figure 4-5a.* Notice the triangular symbol on the clock input for an edge-triggered flip-flop. It is called the *dynamic input indicator* and means that the flip-flop is activated on the *edge* of the clock signal. A small circle on the CK (or CLK) input *(Figure 4-5c)* indicates the flip-flop is activated on the falling (negative) edge; that is, the 1 (HIGH) to 0 (LOW) transition. No small circle *(Figure 4-5b)* on the CK input indicates the flip-flop is activated on the rising (positive) edge; that is, the 0 (LOW) to 1 (HIGH) transition.

Edge-triggering may be more reliable than level triggering. For example, in the D flip-flop (gated-D latch), the data is transferred from D to Q when CK is at a 1 (HIGH) *level.* If the *level* of D changes during the time that CK is a 1, the change will be transmitted to the output. As a result errors may occur. Edge-triggered flip-flops allow a much shorter time to transfer the data, thus, the chance for incorrect data transfer is reduced.

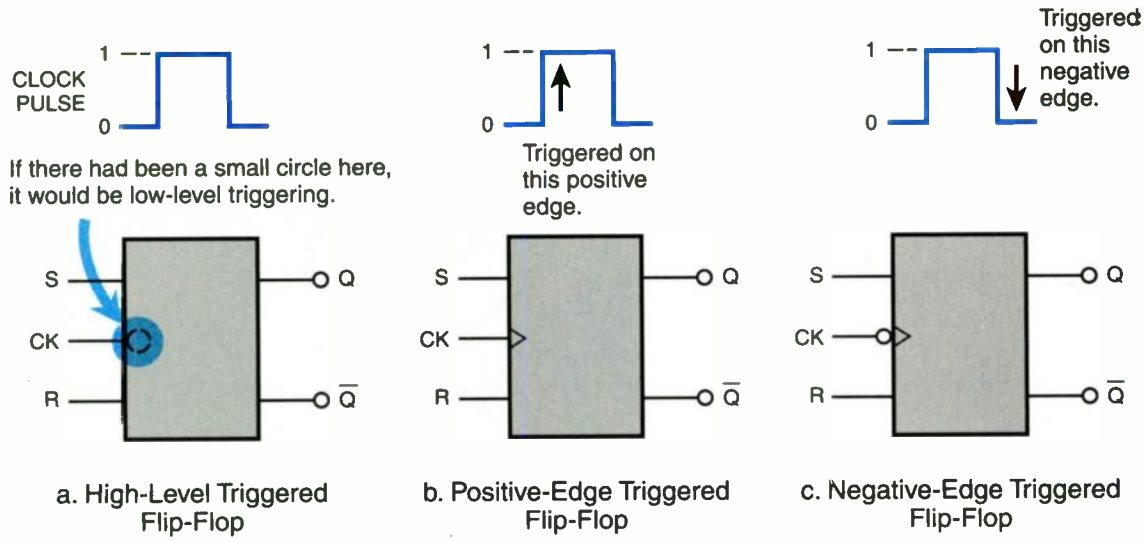


Figure 4-5. Synchronization with clock can occur by clock signal level or by edge triggering.

J-K Flip-Flop

The most versatile and most widely used type of flip-flop is the J-K, shown in *Figure 4-6.* It has two data inputs and one clock input. For the flip-flop in *Figure 4-6a,* CK activates the flip-flop on the negative-edge transition. The data on the J and K inputs determines how the outputs change.

The J-K flip-flop might even be considered the universal flip-flop because it can be configured to emulate the function of all the other types. If we add an inverter from the J input to the K input, we have a D flip-flop. As shown by the truth table of *Figure 4-6b,* if we wire the J and K inputs together *and force them both to a logic 1 (HIGH),* we have a *toggle* flip-flop that divides the clock (CK) by a factor of two.

Dividing the Clock Frequency

The toggle feature of the J-K flip-flop gives us the ability to divide the clock signal to generate new clock signals. One J-K will divide the clock signal by two. If we feed this output to a second J-K, we divide by two again for a total division of four. Each J-K flip-flop we add to the clock chain will divide its input signal frequency by a factor of two to provide various clock frequencies.

Preset and Clear Inputs

Look at *Figure 4-6a.* We haven't talked about the two dotted-line inputs. These are asynchronous inputs and they operate just like the SET and RESET in the S-R latches we discussed. The inputs are PRESET (PR) and CLEAR (CLR). The small circle on each of these inputs on the logic symbol of *Figure 4-6c* means that they are active LOW. The PR SETs the flip-flop; that is, when a LOW is applied to the PR input, it presets Q = 1 (HIGH). When a LOW is applied to the CLR input it RESETs Q = 0 (LOW). With Q = 0, the flip-flop is said to be cleared. It is important to point out that these asynchronous inputs *override* the synchronous inputs.

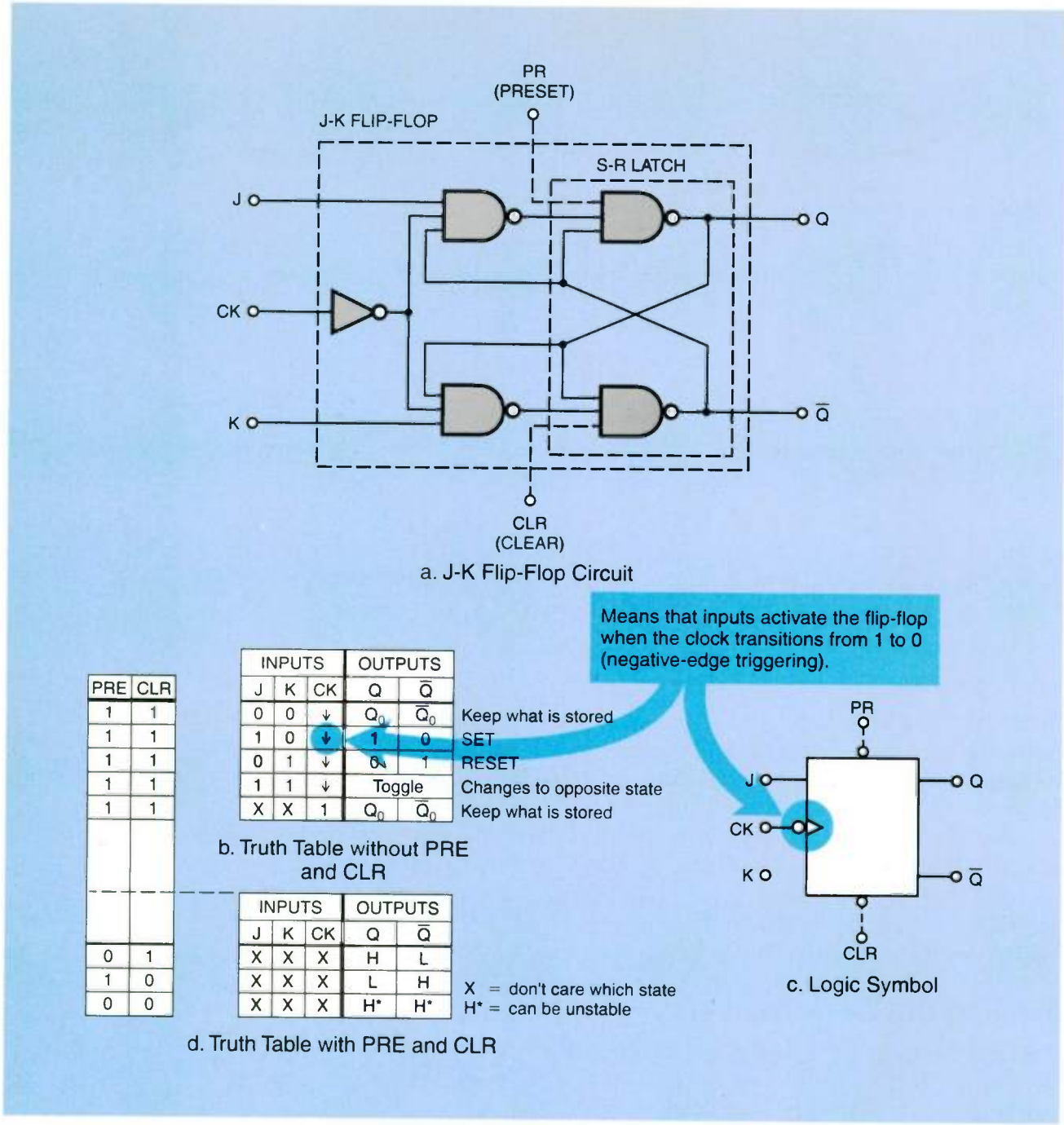


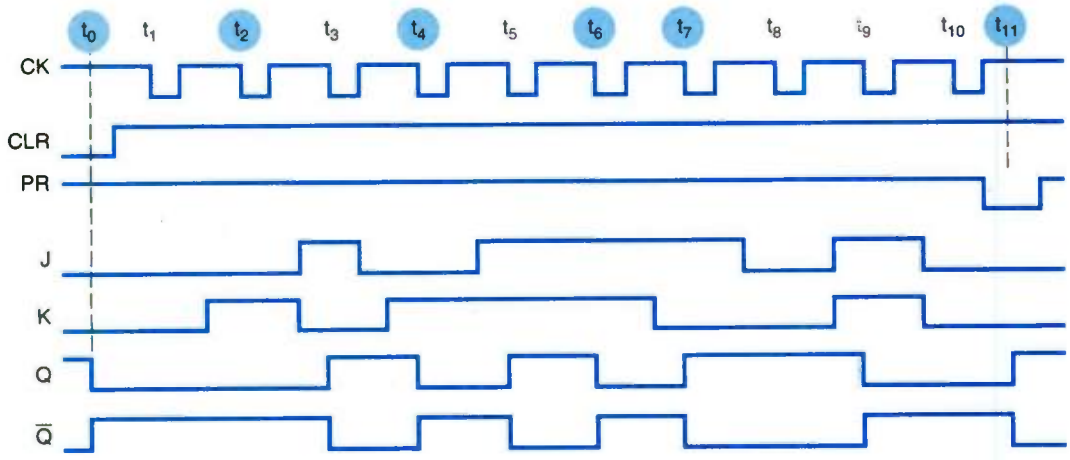
Figure 4-6. A J-K flip-flop with negative-edge triggering, preset and clear.

Registers

Flip-flops, grouped together to temporarily store digital data, are called registers. Each FF stores one bit, thus, the storage capacity in bits is equal to the number of flip-flops (FFs) in the register. A very common flip-flop used for registers is a D FF. As we previously discussed, the data that is on the D input when the active clock signal is applied appears at the Q output. If the input is a 1, Q = 1; if the input is a 0, Q = 0. After the clock goes inactive, the level of Q (the data) remains the same, thus, the data is stored in the register.

Example 2. J-K Negative-Edge-Triggered Flip-Flop Timing Diagram

The timing diagram of the J-K flip-flop shows the relationships between the J-K flip-flop signals. It clearly shows the immediate memory and time dependency properties of a sequential circuit like a J-K flip-flop. At the indicated time transitions of the clock, identify what conditions are occurring at the outputs.



Timing Diagram
J-K Negative-Edge Triggered FF

Solution:

t_0 CLEAR, Q = 0; $\overline{Q}$ = 1
t_2 Since J = 0 and K = 1, a RESET of Q = 0, $\overline{Q}$ = 1
t_4 Since J = 0 and K = 1, a RESET of Q = 0, $\overline{Q}$ = 1
t_6 Since J = 1 and K = 1, a TOGGLE of Q to 0, $\overline{Q}$ to 1
t_7 Since J = 1 and K = 0, a SET of Q = 1, $\overline{Q}$ = 0
t_{11} PRESET of Q = 1, $\overline{Q}$ = 0

The states on the two inputs, J and K, cause four possible output conditions:

1. If both J and K are LOW, the flip-flop will remain in its present state on the next clock pulse.
2. If J is 1 (HIGH) and K is 0 (LOW), the flip-flop will be SET (Q = 1) on the next negative-edge transition of the clock.
3. If K is 1 (HIGH) and J is 0 (LOW), the flip-flop will be RESET (Q = 0) on the next negative-edge transition of the clock.
4. A unique property of the J-K flip-flop occurs when both J and K are HIGH (active). This condition causes the J-K flip-flop to change output state on each clock pulse transition. This behavior is called toggling and is unique to the J-K flip-flop. *This input state was not allowed for the S-R flip-flop.* This is what makes the J-K the most versatile of the flip-flops.

Calculator Display

One example of a register application is the display on a pocket or desk calculator. When the calculator is ON, pressing a number key enters the digit in the first digit position of the display. Each subsequent number key that is pressed enters the new digit into the first digit position and shifts the displayed digits one position to the left. The display illustrates two of the basic functional characteristics of flip-flops. First, it illustrates temporary memory storage, and second, it illustrates shifting data. The shifting can be either to the left or to the right. Let's look at shift registers.

Shift Registers

The *shifting* capability of a shift register allows us to move the contents of data from one cell (individual flip-flop) to another cell inside the register with each clock pulse. Data may also be shifted into or out of the register at the same time. A shift register consists of a group of flip-flops that are serially connected so that each output transfers its information to the next flip-flop of the register when the clock pulse occurs. Four types of shift registers are commonly used:

1. Serial-in, serial-out (SISO)
2. Serial-in, parallel-out (SIPO)
3. Parallel-in, serial-out (PISO)
4. Parallel-in, parallel-out (PIPO)

These four types of registers are shown symbolically in *Figures 4-7a–d*. Each block represents the cell of a register. The arrows indicate the type and direction of data movement. Additionally shown in *Figures 4-7e* and *4-7f* are registers in which data is rotated out of the register and then right back into the register. In other words, the data stored in the register is shifted out to be used, but is put back in for storage.

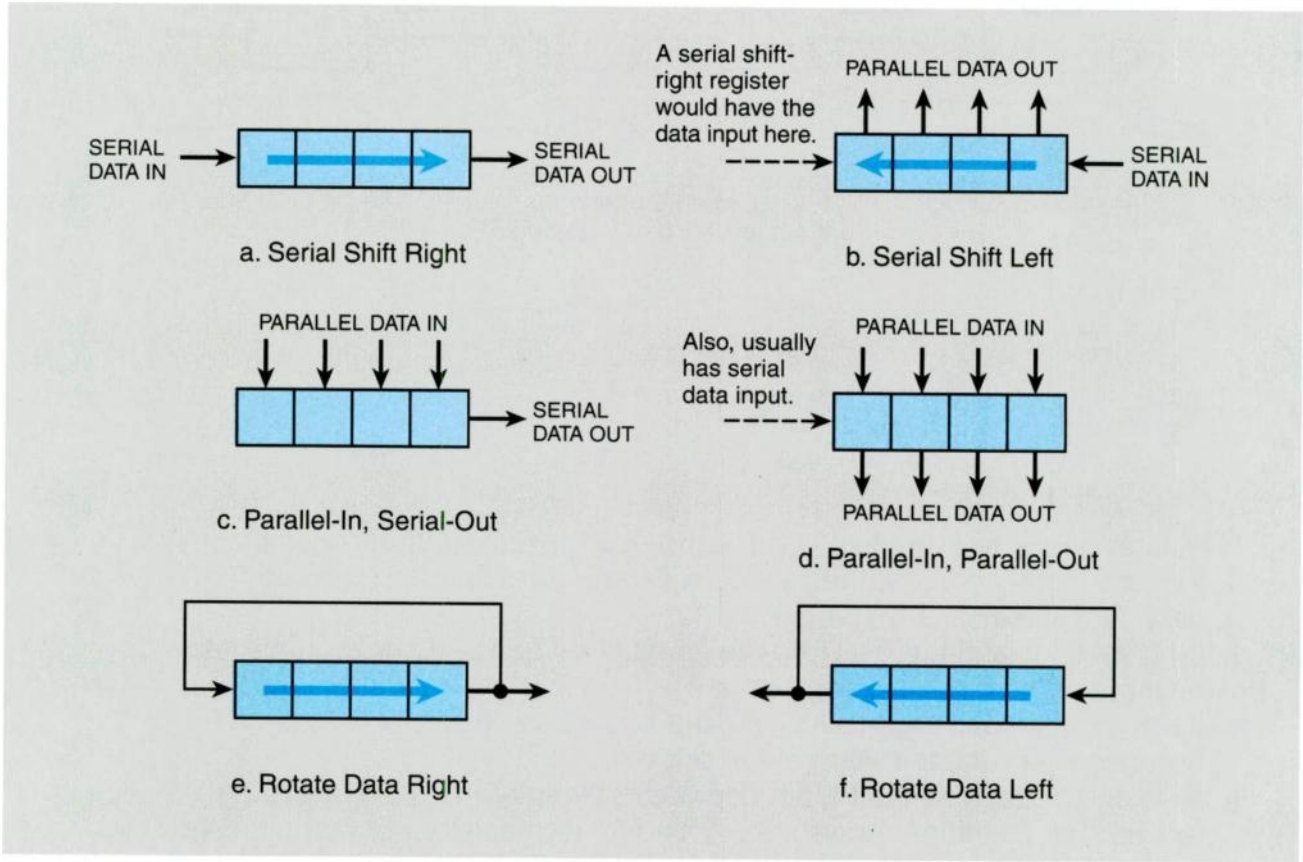


Figure 4-7. Block diagrams of various types of shift registers.

Serial-In, Parallel-Out

Figure 4-8 shows a typical 4-bit, serial-in, parallel-out, shift-right register. It uses four D flip-flops. Data bits are fed serially into the D input of flip-flop A. This shift register was represented in the block diagram of *Figure 4-7b*. The CLR (Clear) input RESETS all four flip-flops to 0 when the level of CLR is taken to 0 (LOW). A single clock pulse clocks all FFs together to shift the data from the D input to the Q output of each FF. After shifting, the FF Q outputs can be read out from D_0, D_1, D_2, and D_3. D_0 is the LSB and D_3 is the MSB. All four bits are available to be read out in parallel. The data is shifted in serially and is read out in parallel.

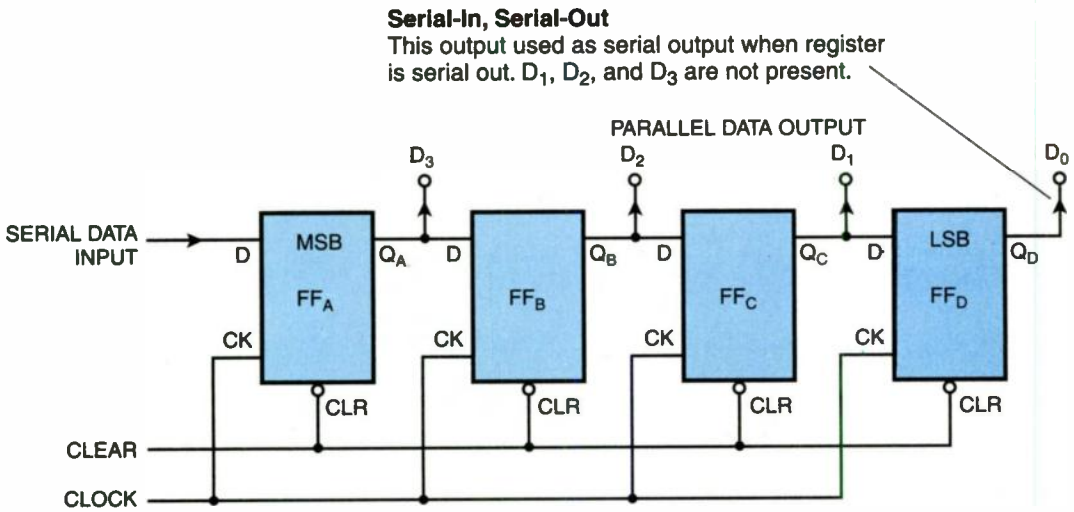


Figure 4-8. A 4-bit serial-in, parallel-out shift-right register.

Example 3. Load a 4-Bit Number Into a Serial-In, Parallel-Out Shift Register

We want to load the 4-bit number, 0101, into the shift register of *Figure 4-8*. First, we apply a 0 to the CLR input to initialize the register so that D_0, D_1, D_2, and D_3 are 0000. We return the CLR input to a 1. Indicate in the four shift register blocks the state of each FF after each clock pulse is applied and returned to a 1 level.

Solution:
Input 0101

	D_3	D_2	D_1	D_0
Clock Pulse #1	1	0	0	0
Clock Pulse #2	0	1	0	0
Clock Pulse #3	1	0	1	0
Clock Pulse #4	0	1	0	1

Whatever data is on the D input of a particular FF at the time of the clock will appear at the Q output of the FF after clocking is over. To check yourself, before the second clock pulse, here is the data at the D input of the four FFs:

	FF_A	FF_B	FF_C	FF_D
D	0	1	0	0

Parallel-In, Serial-Out

The block diagram of a parallel-in, serial-out register was shown in *Figure 4-7c*. Such a register accepts data in parallel from four input bit lines, one to each FF, and outputs the four-bit data in a serial stream on one signal line, one bit after the other until all four bits are read out. In Example 4, the 0101 would be applied to four parallel lines at the same time to load the information through D inputs. A clock pulse would then shift this information out to a single line, forming a data stream that would duplicate the parallel input data.

Example 4. Load a 4-Bit Number Into a Parallel-In, Serial-Out Shift Register

Repeat the same procedure as in Example 3 using the same input data, 0101. Assume that the D input to FF_A is 0 after the initial parallel input of 0101.

Solution:
D Input 0000

	D_3	D_2	D_1	D_0	
Clock Pulse #1	0	1	0	1	With parallel inputs present
Clock Pulse #2	0	0	1	0	
Clock Pulse #3	0	0	0	1	
Clock Pulse #4	0	0	0	0	

The D values would be input to a register as in Example 3.

Parallel-In, Parallel-Out

Figure 4-9 shows a simple 4-bit, parallel-in, parallel-out shift register. It was shown in *Figure 4-7d.* Such registers are used to shift a binary number one digit place to the right or left after input, and then output the number in parallel. In many cases, such registers also have a serial input. When data is already in parallel form, the parallel-load shift register can be a real advantage. The register in *Figure 4-9* employs J-K flip-flops and uses the CLR and PR inputs. Some parallel-load shift registers have PR inputs only. They have a disadvantage because a 1 in any stage cannot be changed back to a 0 with the PR input. The entire register must be cleared to 0s and reloaded with new data. The register in *Figure 4-9* is also a rotating shift register which was shown in *Figure 4-7e.* PR and CLR are active LOW signals.

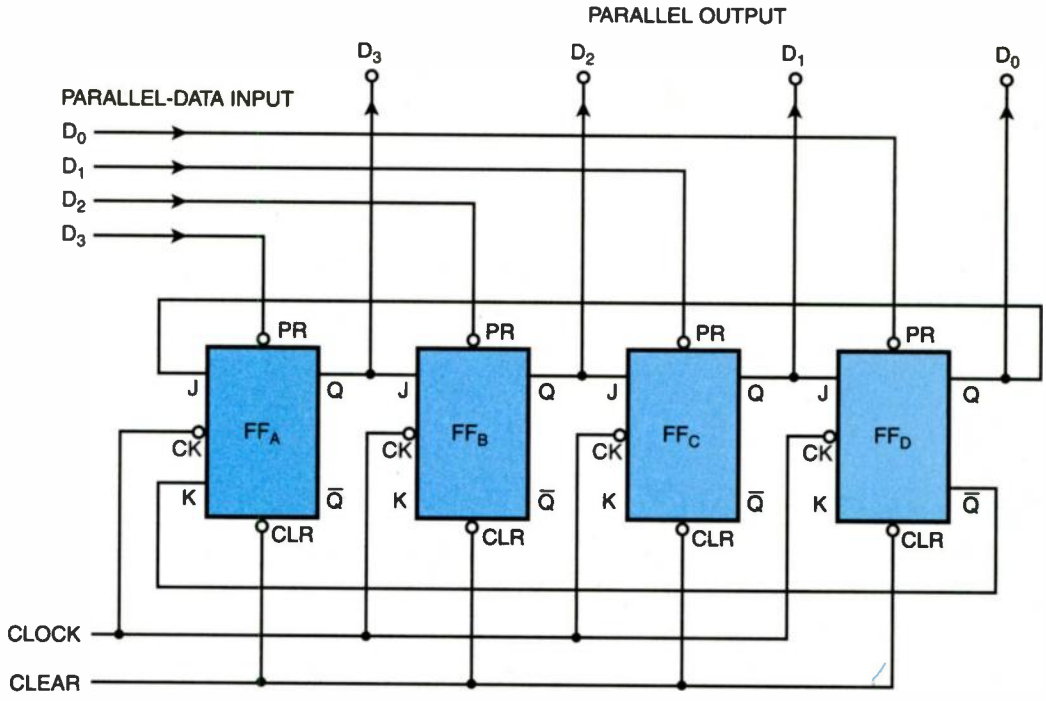


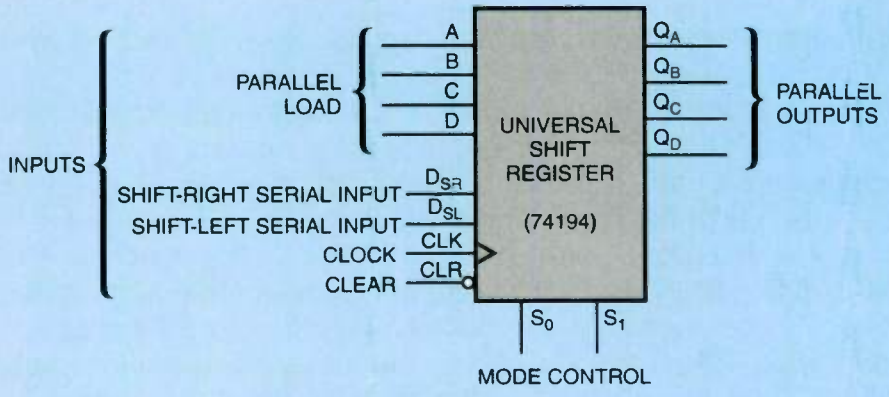
Figure 4-9. A 4-bit parallel-in, parallel-out shift-right register that uses PRESET to input data.

Serial-In, Serial-Out

The serial-in, serial-out shift register processes input data applied serially and delivers serial data out one bit at a time on a single line. It was shown in *Figure 4-7a.* If in *Figure 4-8,* the parallel outputs D_3, D_2, D_1 were not present and D_0 were used as the output for serial data, it would be a serial-in, serial-out shift register. The shifting is as described in Examples 2 and 3. Take note of the fact that while the original four bits are being shifted out, a new four bits can be shifted in.

Universal Shift Register

The 74194 shown in *Figure 4-10* is a universal shift register. All of the functions we've talked about are contained in one integrated circuit (IC). The type of function is selected by two mode control bits, S_0 and S_1. Special gates included in the IC provide the control. For example, bidirectional shifting (right or left) is controlled by setting $S_0 = 1$ and $S_1 = 0$ (shift right) or $S_0 = 0$ and $S_1 = 1$ (shift left). Other modes — parallel load and clocking inhibited — can be selected. Parallel output is available, and if serial data is to be inputted, data is placed on the respective input line. The logic symbol and a truth table are included in *Figure 4-10.*



a. Logic Symbol

INPUTS										OUTPUTS				MODE
CLEAR	MODE S_1	MODE S_0	CLOCK	SERIAL Left	SERIAL Right	PARALLEL A	PARALLEL B	PARALLEL C	PARALLEL D	Q_A	Q_B	Q_C	Q_D	
0	X	X	X	X	X	X	X	X	X	0	0	0	0	CLEAR
1	X	X	0	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}	Keep what is stored
1	1	1	↑	X	X	a	b	c	d	a	b	c	d	Parallel Load
1	0	1	↑	X	1	X	X	X	X	1	Q_{An}	Q_{Bn}	Q_{Cn}	Shift Right – 1 Input
1	0	1	↑	X	0	X	X	X	X	0	Q_{An}	Q_{Bn}	Q_{Cn}	Shift Right – 0 Input
1	1	0	↑	1	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	1	Shift Left – 1 Input
1	1	0	↑	0	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	0	Shift Left – 0 Input
1	0	0	X	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}	Clocking Inhibited

1 = high level (steady state)
0 = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level
a, b, c, d = the level of steady-state input at input A, B, C, or D, respectively

Q_{A0}, Q_{B0}, Q_{C0}, Q_{D0} = the level of Q_A, Q_B, Q_C, or Q_D, respectively, before the indicated steady-state input conditions were established.

Q_{An}, Q_{Bn}, Q_{Cn}, Q_{Dn} = the level of Q_A, Q_B, Q_C, or Q_D, respectively, before the most-resent ↑ transition of the clock.

b. Truth Table

Figure 4-10. A 4-bit universal shift register that operates in modes determined by binary code on S_0 and S_1 inputs.

Multivibrators

Remember, a flip-flop is a sequential bistable digital circuit. Another sequential bistable digital circuit is the *multivibrator.* It is usually used to generate pulses — clock pulses are a good example. The three types of multivibrators are: *bistable, monostable*, and *astable.*

Bistable Multivibrator

The flip-flop is a bistable multivibrator. It comes under the general umbrella of multivibrators and has carved for itself the special name separate from the bistable multivibrator. Many times, multivibrators are triggered from one bistable state to another by ac-coupled signals, while a flip-flop in digital circuits can operate with dc coupling. In any case, a bistable multivibrator has two stable states. It stays in a stable state until forced to its other stable state, and requires input signals to do the forcing (triggering).

Monostable Multivibrator

The monostable multivibrator is named as such because it has only one stable state. It is commonly called a *one-shot* multivibrator. When it is triggered, the one-shot multivibrator produces a single rectangular output pulse of a predetermined width. The pulse width, which can vary from a few nanoseconds to several seconds, is normally determined by a resistor and capacitor external to the main circuit.

Shown in *Figure 4-11* is a simple monostable multivibrator built from two 7400 2-input NAND gate ICs and a RC (resistor-capacitor) timing circuit. It is very similar to the S-R latch circuit except it has a capacitor in one of the cross-coupled output to input connections. Here's how the circuit works: When power is first applied to the circuit, both inputs of NAND gate U_A are 1 (HIGH) and Q = 0. The cross connection inputs the 0 to U_B and $\overline{Q} = 1$. When the normally open push-button S_1 is pressed, that trigger input of U_A goes to 0 (LOW), forcing Q to 1 (HIGH). The 1 input to U_B changes $\overline{Q}$ to a 0. The capacitor voltage across C_1 cannot change instantaneously, so point X (the junction of the resistor and capacitor) goes to almost zero volts (LOW). Point X is the second input of U_A and the 0 holds the NAND gate output Q = 1, even if the push-button S_1 is released and the trigger input returns HIGH. Point X increases toward +5V as the capacitor charges through the resistor to V_{CC}. Q remains at 1. When the voltage at point X reaches a HIGH level, Q switches back to a 0 (LOW), and the circuit is back in its initial stable state. This completes the cycle initiated by pressing S_1. The output is a single pulse with a width in time that depends on the value of R_1 and C_1. The product of R_1 in ohms and C_1 in farads is called a *time constant* which is measured in seconds. Therefore, digital circuit designers say the pulse width is determined by the R_1C_1 time constant.

Several IC monostable multivibrators are available. Two popular ones are the 74121 and the 74123. The 74121 is *non-retriggerable.* If a trigger is applied before the circuit completes its timing cycle, that trigger is ignored. The 74123 is *retriggerable.* It will start a new timing cycle each time a new trigger is applied.

Astable Multivibrator

You probably have guessed what astable means. You're right! It means this type of multivibrator has no stable state. It just keeps switching back and forth between two unstable states as long as power is applied. It is also called a *free-running* multivibrator. As it switches between its two unstable states, it generates an output that is a regular pattern of rectangular pulses. The astable multivibrator can be used as an oscillator that outputs periodic square waves to provide clock signals for synchronization.

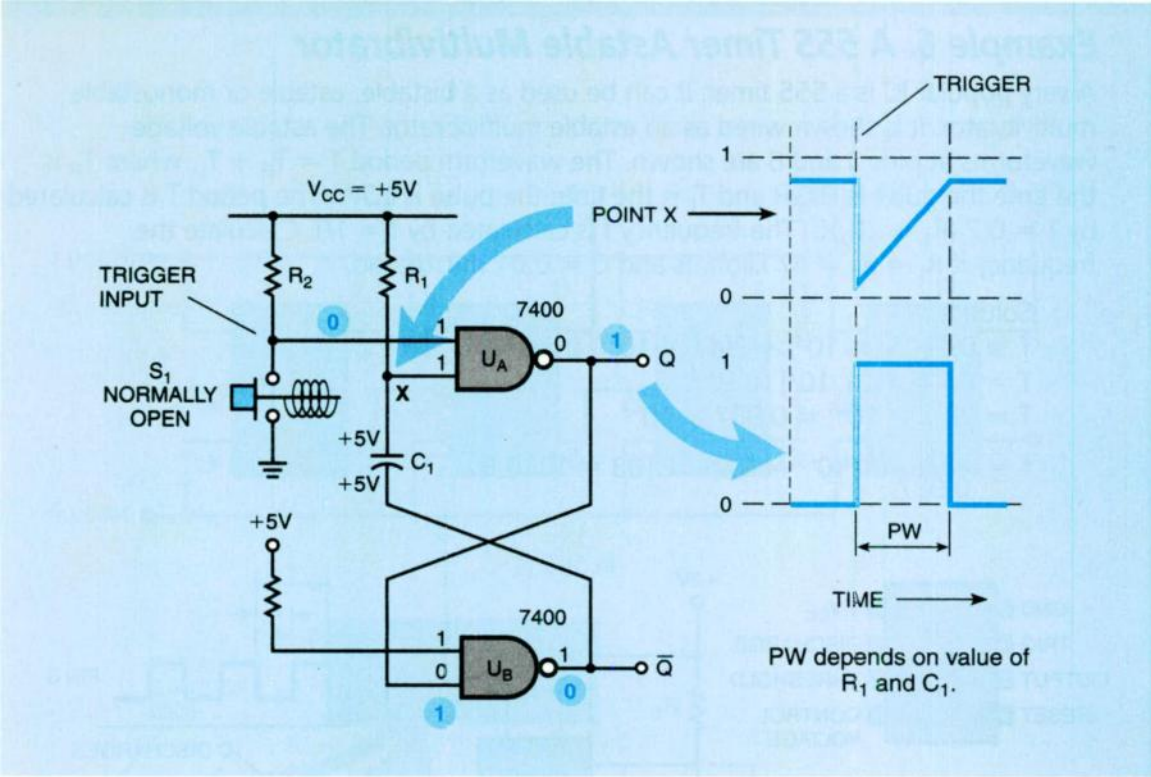


Figure 4-11. A monostable multivibrator circuit built from two NAND gates.

Example 5. An Astable Multivibrator

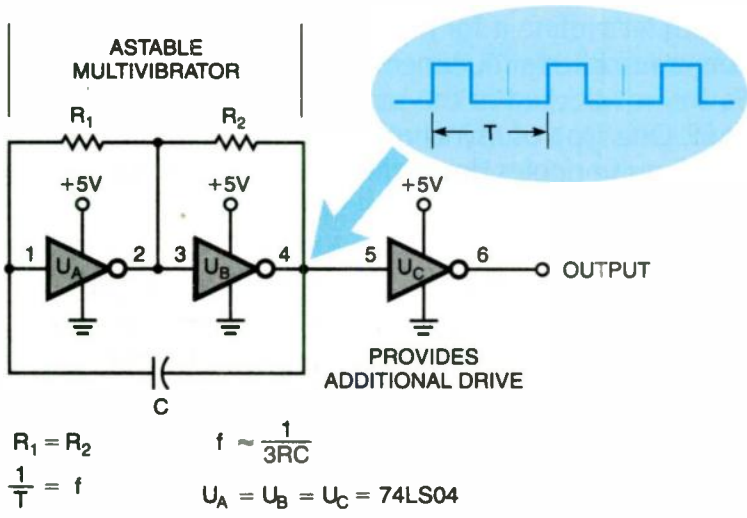
An astable multivibrator constructed using two TTL inverters, two resistors and a capacitor is shown. The output of one inverter is coupled back to the input of the other inverter so the circuit forms a basic oscillator. The pulse will be ON for about 33% of the cycle and its frequency can be approximated as f = 1/(3RC), which is the reciprocal of the period T. Calculate the frequency if R = 10 kilohms and C = 0.001 microfarad.

Solution:

$RC = 10 \times 10^3 \times 1 \times 10^{-9} = 1 \times 10^{-5}$ second

$3RC = 3 \times 10^{-5}$ second

$$f = \frac{1}{3 \times 10^{-5}} = 0.333 \times 10^5 = 33{,}300 \text{ Hz}$$



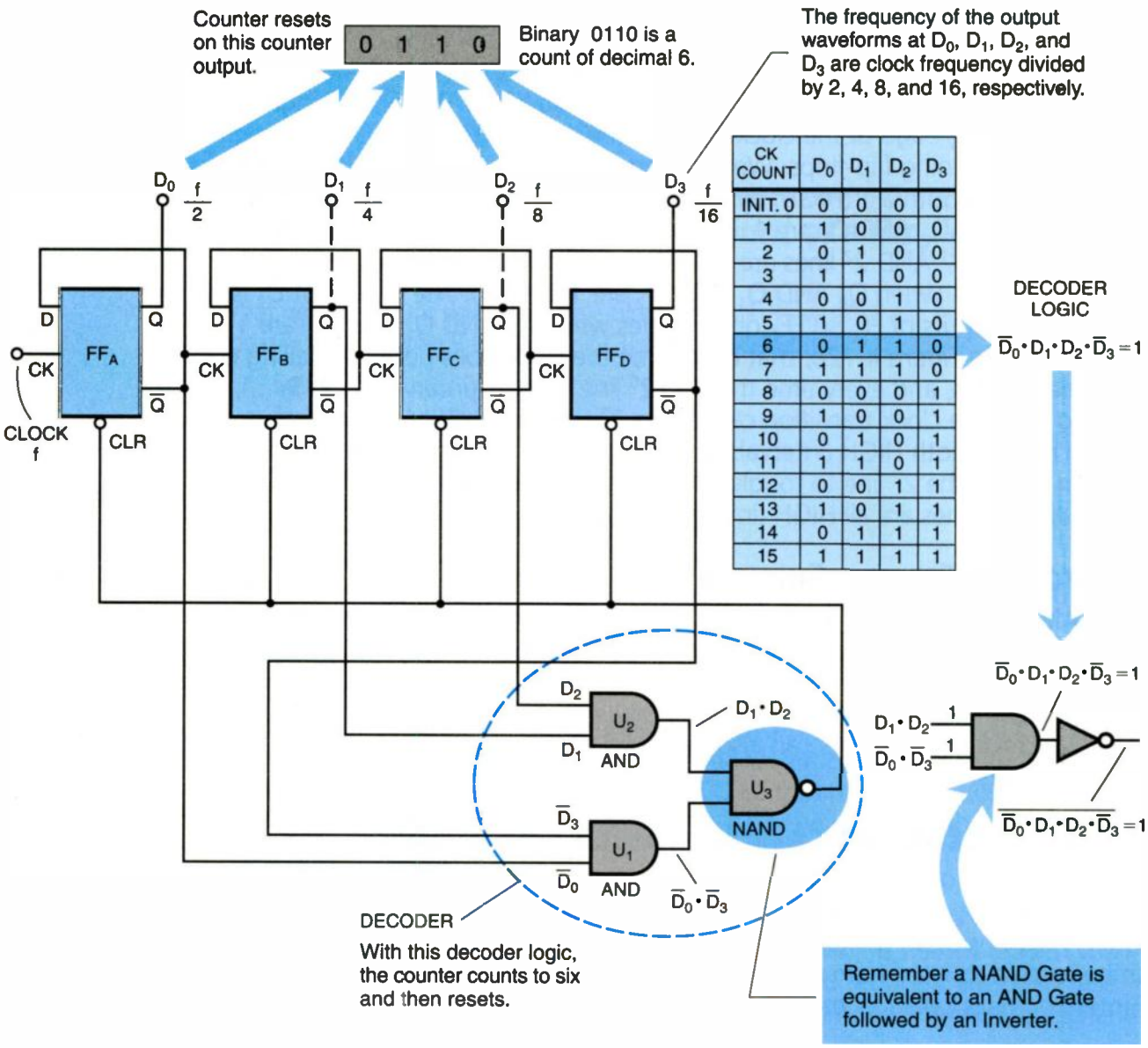


Figure 4-14. An asynchronous modulo-6 counter using D-type flip-flops. A decoder resets the counter at the desired count.

which is $\overline{Q}_0$, and $\overline{D}_3$, which is $\overline{Q}_3$, are ANDed together in U_1; and D_1, which is Q_1, is ANDed together with D_2, which is Q_2, in U_2. The outputs of U_1 and U_2 are ANDed together in U_3 to form the total logic statement $\overline{D}_0$ AND $\overline{D}_3$ AND D_1 AND D_2. This logic statement is 1 when all inputs are 1. If U_3 had been an AND gate, the output would be a 1. Because the FFs in the counter need a 0 to reset (clear) the FFs to the initial state of 0, U_3 is a NAND gate (an AND gate followed by an inverter). No other combination of flip-flop states will produce a 1 output from the decoder. Decoders like this can be used to make the counter operate at any modulus up to its natural modulus. After the desired count is reached, the output of the decoder is used to CLR (clear) or RESET all of the flip-flops in the counter and the count resumes at zero.

Integrated Circuit Counters

Integrated circuit designers and manufacturers have made available a wide variety of IC counters so that a digital system designer has little problem choosing a synchronous or asynchronous counter for an application. And if the desired modulo counter is not available, existing ones can be combined to obtain the correct modulus. For example, a modulo-6 and a modulo-10 counter can be combined as a divide-by-6-by-10 combination, resulting in a modulo-60 device.

Programmable Counters

Programmable counters capable of variable modulus configurations are also available. The modulus is determined by applying binary codes to programming inputs. The code determines the modulus, which can be set at any value up to the maximum count. Of course, there is a price to pay; the programmable counters are more expensive than the fixed modulo type. Let's use an example to show how the counter principles we have presented can be used in a practical application — a time-of-day clock.

A Digital Clock

This real-time, 12-hour digital clock displays seconds, minutes, and hours. A block diagram is shown in *Figure 4-15*. A signal obtained from the 60-Hz power line is divided by 10 and then by 6 to obtain a 1 Hz clock to generate one pulse per second. Modulo-10 and modulo-6 counters are used. The 1 Hz signal drives a modulo-10 counter and its output provides the clock pulse for a modulo-6 counter. The modulo-10 counter generates the least significant digit (LSD) for the seconds display. It counts from 0 to 9 and then RESETS. At RESET, the modulo-10 counter outputs a pulse that becomes the clock for the modulo-6 counter to increment one digit for every ten pulses input to the modulo-10 counter. The modulo-6 counter generates the most significant digit (MSD) for the seconds display. When the modulo-6 reaches the binary count of 0110, it resets so fast that its "6" state is not recognized on the display. At the same time, the LSD counter is resetting from a 9 count so both "seconds display" counters reset and the seconds display starts over at 00 seconds.

When the "seconds display" counters RESET, a 1 pulse-per-minute clock is generated to drive the modulo-10 counter that generates the LSD for the minutes display. The minutes modulo-10 counter drives the minutes modulo-6 counter and they both reset in the same fashion as the seconds combination. The modulo-6 counter generates the MSD for the minutes display. When the "minutes display" counters RESET, a 1 pulse per hour clock is generated that drives a modulo-10 and a modulo-2 counter for the hours display. When the modulo-2 counter reaches the binary count of 0010, it resets so fast that the 2 is not recognized. Special logic comes into play when the modulo-2 counter has counted to display a 1 as the MSD. It restricts the modulo-10 LSD hours digit counter to a modulo-3 counter. Under these conditions, when the binary count goes to 0011, the converted modulo-10 counter resets, causing the modulo-2 counter to reset. The highest count that will be displayed is 12:59:59. When the counters are in that state, the next 1 second pulse will RESET all the counters except the hours 1's digit modulo-10 counter, which is reset to display a 1; thus, the display becomes 01:00:00. Note the extra logic provided to turn on and off a "PM" LED after a count of 12 hours, and to turn on LEDs that flash every second. Note that toggling D-type FFs are used. For the 7-segment decoder, refer to *Figure 2-9* for a refresher.

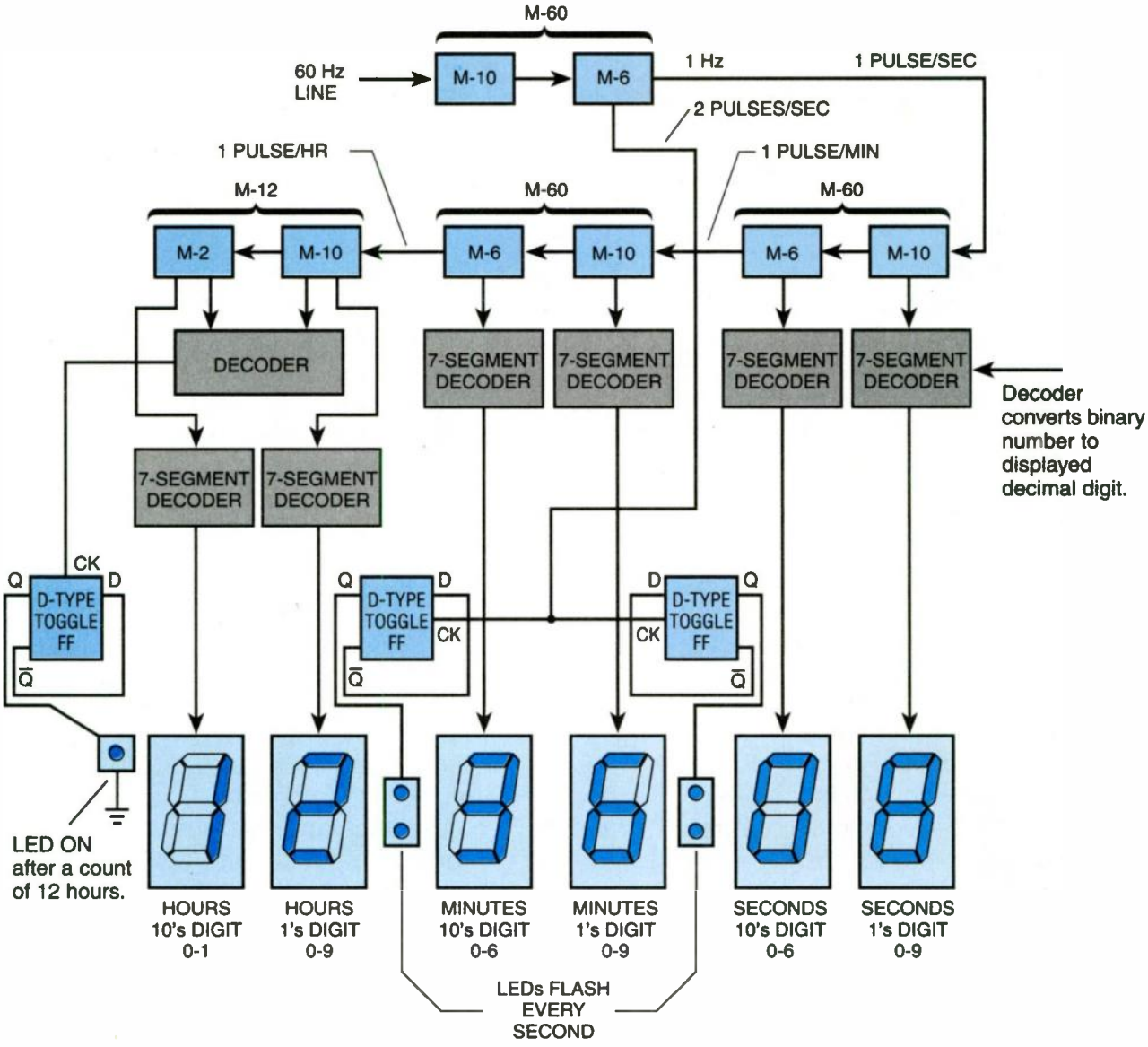


Figure 4-15. A real-time time-of-day clock block diagram. Seven-segment decoders convert the binary counter output to a digital display.

Summary

In this chapter we have learned about sequential digital circuits called latches and flip-flops, showed how they are used in registers and counters, and gave examples of how they are combined to build more complicated functions. In the next chapter, we will find out how to couple, convert, and compute.

Quiz for Chapter 4

1. Circuits that have the ability to remember the order of the applied inputs are called _____ circuits.
   a. combinational
   b. gate
   c. sequential
   d. synchronous
2. When a S-R latch is SET the Q output is _____ .
   a. toggling
   b. LOW
   c. HIGH
   d. unstable
3. A synchronous latch made of four gates is normally called a _____ .
   a. clocked flip-flop
   b. shift register
   c. universal counter
   d. ripple counter
4. An edge-triggered flip-flop is a type that is enabled by the _____ of the clock signal.
   a. peak voltage
   b. negative alternation
   c. dc level
   d. transition
5. A small circle on the clock input indicates that the flip-flop is activated on the _____ of the clock signal.
   a. falling edge
   b. rising edge
   c. peak
   d. average value
6. The most versatile and most widely used type of flip-flop is the _____ .
   a. S-R
   b. D
   c. J-K
   d. universal
7. Flip-flops grouped together to temporarily store digital data are called _____ .
   a. synchronous counters
   b. ripple counters
   c. multivibrators
   d. registers
8. The _____ capability allows us to move the contents of data from one cell to another inside a register.
   a. shifting
   b. synchronous
   c. astable
   d. bistable
9. A flip-flop is an example of a/an _____ multivibrator.
   a. astable
   b. bistable
   c. monostable
   d. one-shot
10. A _____ multivibrator produces a single rectangular output pulse when triggered.
    a. astable
    b. bistable
    c. monostable
    d. tristable
11. If all of the flip-flops in a circuit are clocked at the same time then the circuit is said to be _____ .
    a. universal
    b. ripple
    c. asynchronous
    d. synchronous
12. A 4-bit counter has the binary number 0110 stored in it. This is equivalent to the decimal number _____ .
    a. 3
    b. 5
    c. 6
    d. 8
13. An advantage of a synchronous counter over an asynchronous counter is that it _____ .
    a. counts to a higher number.
    b. counts at a higher speed.
    c. requires less clock signal power.
    d. is less complicated
14. The number of counting states that a counter can attain before it begins to repeat itself is called its _____ .
    a. modulus
    b. steady state
    c. maxcount
    d. latch state
15. Counters capable of variable modulus configurations are called _____ counters.
    a. ripple
    b. programmable
    c. astable
    d. sequential

Answers:
1c, 2c, 3a, 4d, 5a, 6c, 7d, 8a, 9b, 10c, 11d, 12c, 13b, 14a, 15b

Questions and Problems for Chapter 4

1. Use *Figure 4-2* and show that a SET = 1 sets Q to a 1 and turns on the alarm. Then show that R = 1, after S = 0, resets the latch and the alarm.
2. How can a NAND gate S-R latch be made synchronous?
3. How can a clocked S-R latch be modified to create a gated D latch?
4. Compare the reliability of level triggering to edge triggering in flip-flops.
5. Why is the J-K flip-flop considered the universal flip-flop?
6. If three toggle flip-flops are connected together with the Q output of one to the clock input of the next and a 16-MHz clock is input to the first, what will be the frequency of the output of the third flip-flop? Explain.
7. How many bits can be stored in a register made of 8 flip-flops? Why?
8. In the figure of Example 2, at the time transition t_5 of the clock, identify what conditions are occurring at the outputs.
9. Name the four types of shift registers commonly used.
10. Describe how to load the 4-bit binary number 1001 into an SIPO register if the register is cleared to start (0000).
11. Describe what the 4-bit binary number output will be if a 1001 is loaded into an PISO register after the register is cleared to start (0000), and then shifted right with four clock pulses. Assume the D input of first flip-flop is 0 after the initial parallel load.
12. Calculate the frequency of oscillation of the 555 timer shown in Example 6 if $R_1 = R_2 = 22$ kilohms and C = 0.005 microfarad.

CHAPTER 5

How to Couple, Convert, and Compute

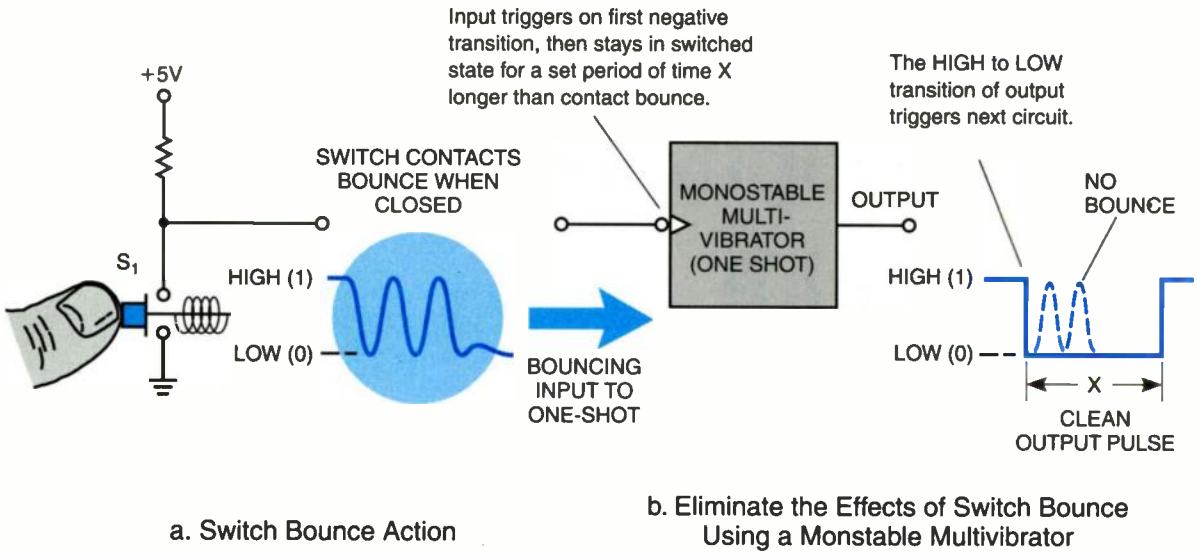
Interfacing Digital Circuits

To *couple* two or more circuits or systems means to associate them in such a way that a signal (power) may be transferred from one to another. This *coupling* may be from the output of one gate to the input of another gate, to or from a data line or register, or from almost any type digital device to another. A common application is to couple data to or from a bus. Remember that a bus is a group of conductors (data lines) used to transfer signals from one device to another. In most cases, the digital code used for the information is traveling in parallel along the bus in a given time period. The coupling may involve some form of *interfacing* — meaning to connect in a compatible manner. In the case of a digital system, the LOW and HIGH voltage levels that are output by a driving device must provide adequate noise margins with respect to the LOW and HIGH input voltage level specifications of the receiving device. Let's look at some examples of digital coupling circuits, starting with a simple switch interface.

Switch Debouncing

As we have seen in previous chapters, mechanical switches form the interface between human beings and many digital systems, such as computers. Unfortunately, however, most switches exhibit a phenomenon called switch *bounce,* which occurs when the normal metal contact on a switch closes or opens. This bounce prevents the contacts from making and breaking the circuit smoothly. Instead they bounce or physically vibrate, causing the contact to make and break the circuit many times before they come to rest. Although these bounces are minute, they produce voltage pulses that are often not acceptable in a digital system. This switch bounce action is shown in *Figure 5-1a.*

There are several ways to eliminate the effects of switch bounce. A simple method using a monostable multivibrator (one-shot) latch is shown in *Figure 5-1b.* Switch S_1 is normally open, which keeps the input HIGH and the output HIGH. When the switch is pushed, spring contacts inside the switch close to cause the input to go LOW as the switch first makes contact. Because of the springy nature of the mechanical construction of the switch, the contacts do not stay closed, but bounce, that is, they break contact for an instant and cause the input to go momentarily HIGH. The contacts bounce again and then stay closed. The first transition from HIGH to LOW is sufficient to trigger the one-shot and cause its output to go low. Any further voltage pulses do not affect the circuit, so its output remains LOW until the time X shown in *Figure 5-1b* goes by. Time X is longer than the bouncing of the contacts. After time X, the one-shot returns to the HIGH state. The HIGH to LOW



a. Switch Bounce Action

b. Eliminate the Effects of Switch Bounce Using a Monstable Multivibrator

Figure 5-1. Eliminating contact bounce in digital systems.

transition of the output, which is bounce free, triggers the next circuit and eliminates the effect of the contact bounce. You can see why the monostable multivibrator is called a "one-shot." It triggers, puts out one pulse of a given length of time, and then resets, ready for the next input trigger.

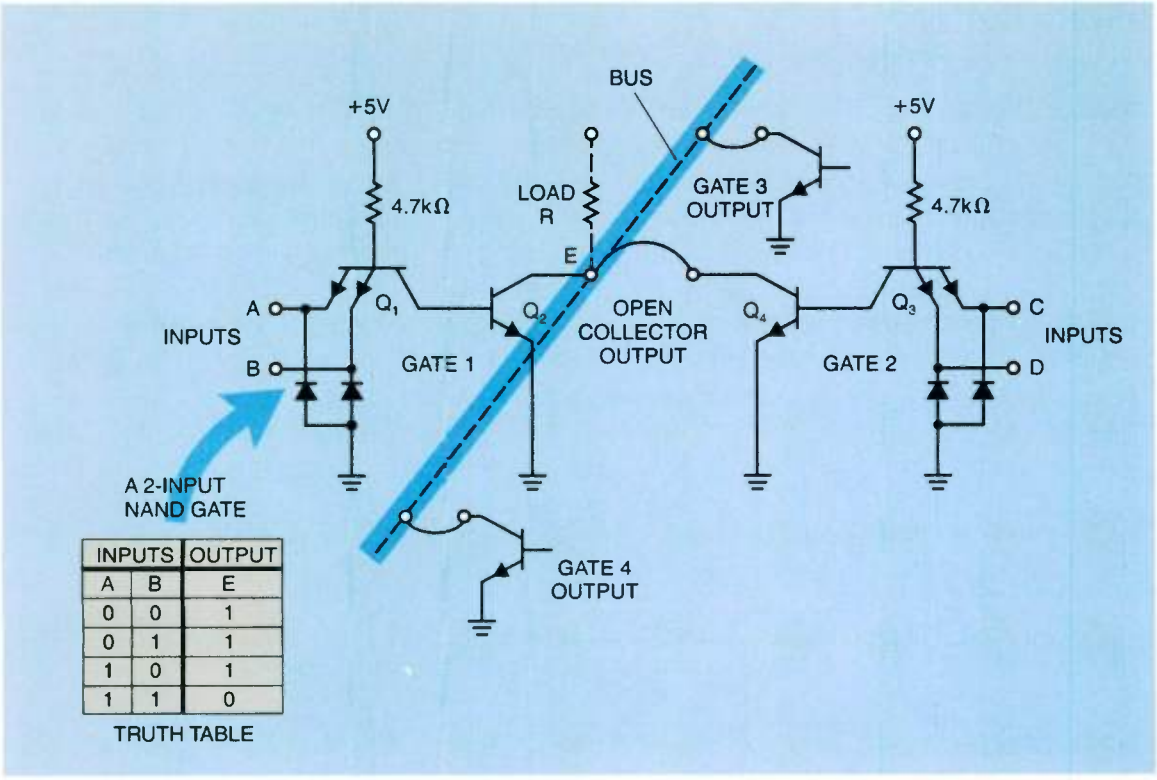
Buffers and Bus Drivers

What is a buffer? A buffer is a special type of logic gate to isolate conventional gates from other circuits, and to provide high driving currents for heavy circuit loads or high fan-out. A non-inverting buffer is a single input circuit that does not alter the signal: a 1 in, a 1 out. It is common also to have an inverting buffer that has an output that complements the input: a 1 in, a 0 out. These gates usually have heavy-duty output transistors with power handling capabilities that allow them to supply (source) high currents. They provide isolation, or *buffering,* between a digital circuit and a common data bus, thus, they are called bus drivers. They also provide isolation for a conventional gate by providing the sink or source current required by any device connected to its output without loading down the input device. A very popular type buffer is the three-state buffer. Before we discuss it, let's look at open-collector drivers.

Open-Collector Bus Driver

As we discussed in Chapter 3, most TTL gates have totem pole output stages *(Figure 3-4b).* Such outputs present an interconnection problem. They cannot be tied directly together because if one output goes HIGH while another goes LOW, a direct connection will cause an indeterminate output level. Damage to one or both of the gates is also very likely because one output in the LOW state would act as a *short to ground* to another output in the HIGH state, excessive current would result, and high-temperature failure of the junction probably would occur. The only way to tie the totem-pole outputs together is to couple them through the inputs of another gate.

A special gate shown in *Figure 5-2* solves the problem. It is an open collector TTL NAND gate, in this case, a 2-input gate. Notice that the output collector does not have a built-in load resistor. A load resistor must be supplied. With a load resistor, as



INPUTS		OUTPUT
A	B	E
0	0	1
0	1	1
1	0	1
1	1	0

TRUTH TABLE

Figure 5-2. Open collector TTL NAND gates used as bus drivers.

shown by the truth table, the gate is a 2-input NAND gate. Coupling another open-collector gate to the E output of gate 1 causes the E output to be controlled by the state of the combination of the inputs of gate 1 and gate 2. It turns out that the combination is a NOR of two AND gate outputs. Here are two conditions that govern the coupling of open collector gate outputs:

1. All gate outputs connected to the bus are pulled LOW if any gate output goes LOW.
2. The only time that the common output bus can be HIGH is when all gate outputs connected to it are HIGH.

The open collector gate was an early solution to the problem of connecting several gate outputs directly to the same bus, but a better solution is the 3-state buffer, which is also called the tri-state™ buffer.

Example 1. Develop a Truth Table for a Bus Driver

For the circuit in *Figure 5-2* with two open collector 2-input NAND gates coupled together by connecting the outputs to load resistor R, develop the truth table for the complete circuit. Convince yourself that the circuit satisfies $E = \overline{AB + CD}$. (Remember the + sign means an OR function.)

TRUTH TABLE							
INPUTS				OUTPUTS			
A	B	C	D	E			
0	0	0	0	1			
0	0	0	1	1			
0	0	1	0	1			
0	0	1	1	0	$\overline{A}\overline{B}CD$		
0	1	0	0	1			
0	1	0	1	1			
0	1	1	0	1			
0	1	1	1	0	$\overline{A}B\overline{C}D$	$\overline{XXCD}$	
1	0	0	0	1			
1	0	0	1	1			
1	0	1	0	1			
1	0	1	1	0	$A\overline{B}CD$		$\overline{AB+CD}$
1	1	0	0	0	$AB\overline{C}\overline{D}$		
1	1	0	1	0	$AB\overline{C}D$	$\overline{ABXX}$	
1	1	1	0	0	$ABC\overline{D}$		
1	1	1	1	0	$ABCD$		

3-State Buffers

A 3-state buffer combines the output advantages of the totem pole and the open collector circuits. A 3-state device has the usual HIGH and LOW logic output states and also a third state which is called a *high-impedance* state. This state is simply an open circuit between the 3-state device's internal circuitry and the output terminal that is connected to the bus. *Figure 5-3* shows these three states as the 3-state buffer is used as a bus driver. The EN represents an enable/disable control input on the 3-state buffer and can be either active HIGH or active LOW. When the EN is activated, the 3-state operates in the same way as a regular gate. When a 3-state buffer is disabled, it is in its high-impedance state and the output is effectively disconnected from the rest of the circuit. 3-state devices are available in both TTL and CMOS families.

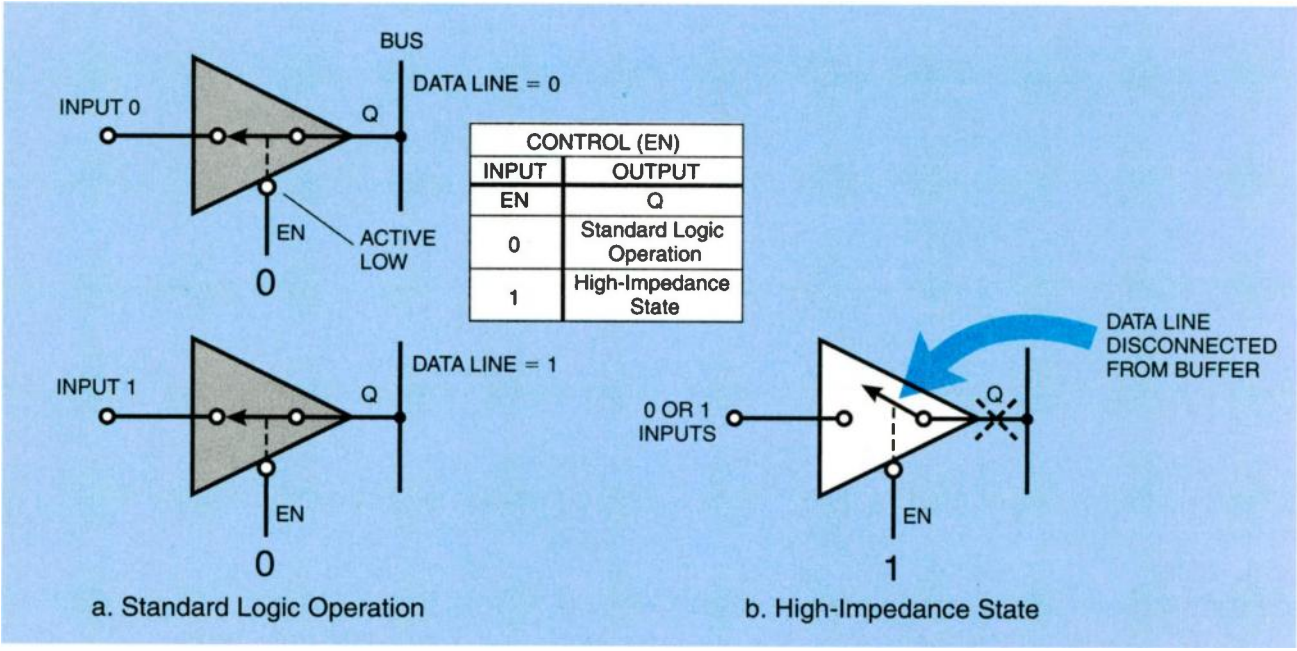


Figure 5-3. 3-state buffer operation.

Bidirectional Bus Drivers

A bus in a digital system can usually carry data in both directions and the device connected to it may be transmitting data to the bus, receiving data from the bus, or both — but not usually at the same time. A device that transmits data to the bus is called a *talker* while one that receives data from the bus is called a *listener.* Some 3-state devices can do both, either talk or listen to the bus; when they can, the bidirectional devices are called *transceivers* (transmitter/receiver). This capability is necessary for interfacing devices that are used for both input to and output from a digital system.

An example of a popular octal 3-state non-inverting bus transceiver is the 74LS245. *Figure 5-4* shows the actual pin configuration and internal logic diagram. This transceiver is ideal for controlling the bus of microprocessors. Notice that the buffers are configured in two groups of eight. One group passes data from the *A port to the B port,* and the other group passes data the other way; that is, from the *B port to the A port.* When $\overline{EN}$ is LOW, gates 1 and 2 are enabled. When the direction control DIR goes HIGH, the output of gate 1 is 0, and the buffers that pass data from the *B port to the A port* are put in the high-impedance state and are disconnected from the *A port* side terminals. At the same time, gate 2 produces a 1 output, which enables all the buffers that pass data from the A port to the B port. Data flows from

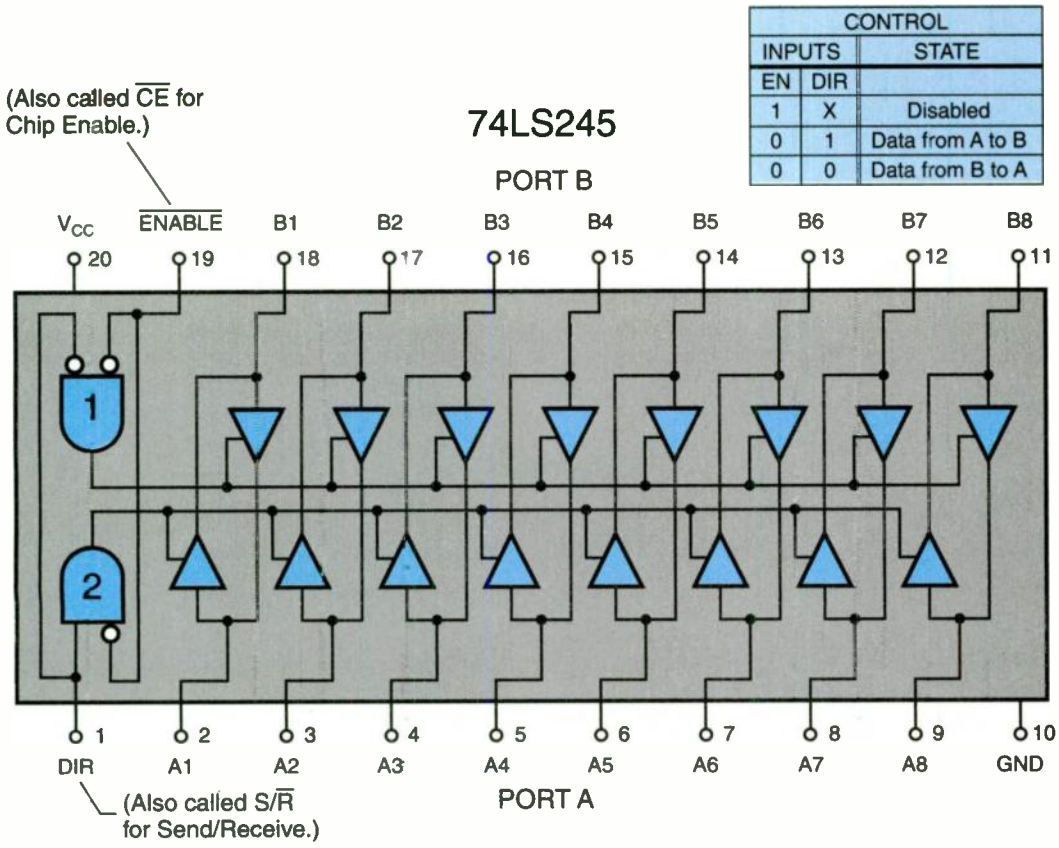


Figure 5-4. 74LS245 octal 3-state transceiver.

the *A port* to the *B port.* Changing the direction control DIR to LOW, the output of gate 1 is a 1, the buffers that pass data from the *B port* to the *A port* are enabled, the buffers that pass data from the *A port* to the *B port* are put in the high-impedance state and are disconnected from the *B port* terminals, and data flows from the *B port* to the *A port.* Only one group of buffers is active at a time, so no conflict can occur.

Application of Bidirectional Bus Drivers

Bidirectional bus drivers or transceivers can be used for communication between a microcomputer bus and input/output (peripheral) devices when parallel transfer of the data is required. This capability is necessary for interfacing devices that are used for both input and output to a microprocessor; for example, a disk drive. *Figure 5-5* shows a common way to connect I/O devices to a data bus with transceivers. To make I/O device 1 the active interface, the $\overline{CE}$ (Chip Enable) pin of its octal transceiver must be made LOW. When $\overline{CE}$ is HIGH, the transceiver disconnects I/O device 1 from the bus by making all of that transceiver's outputs high impedance, which effectively open circuits I/O device 1. This makes the device 1 output connections float. After enabling device 1's transceiver, the microprocessor then puts the appropriate level on the S/$\overline{R}$ (Send/Receive) pin depending on whether it wants to *send data to* device 1 or *receive data from* device 1. If the S/$\overline{R}$ is made HIGH, the transceiver allows data to be sent to I/O device 1 from the microcomputer (from port A to port B). If the S/$\overline{R}$ is made LOW, the transceiver allows data to be received from I/O device 1 (from port B to port A). When $\overline{CE}$ of device 2's transceiver is made LOW, and $\overline{CE}$ of device 1's transceiver is made HIGH, then I/O device 2 becomes the active device and the microcomputer sends the appropriate control signal to the transceivers to send or receive data.

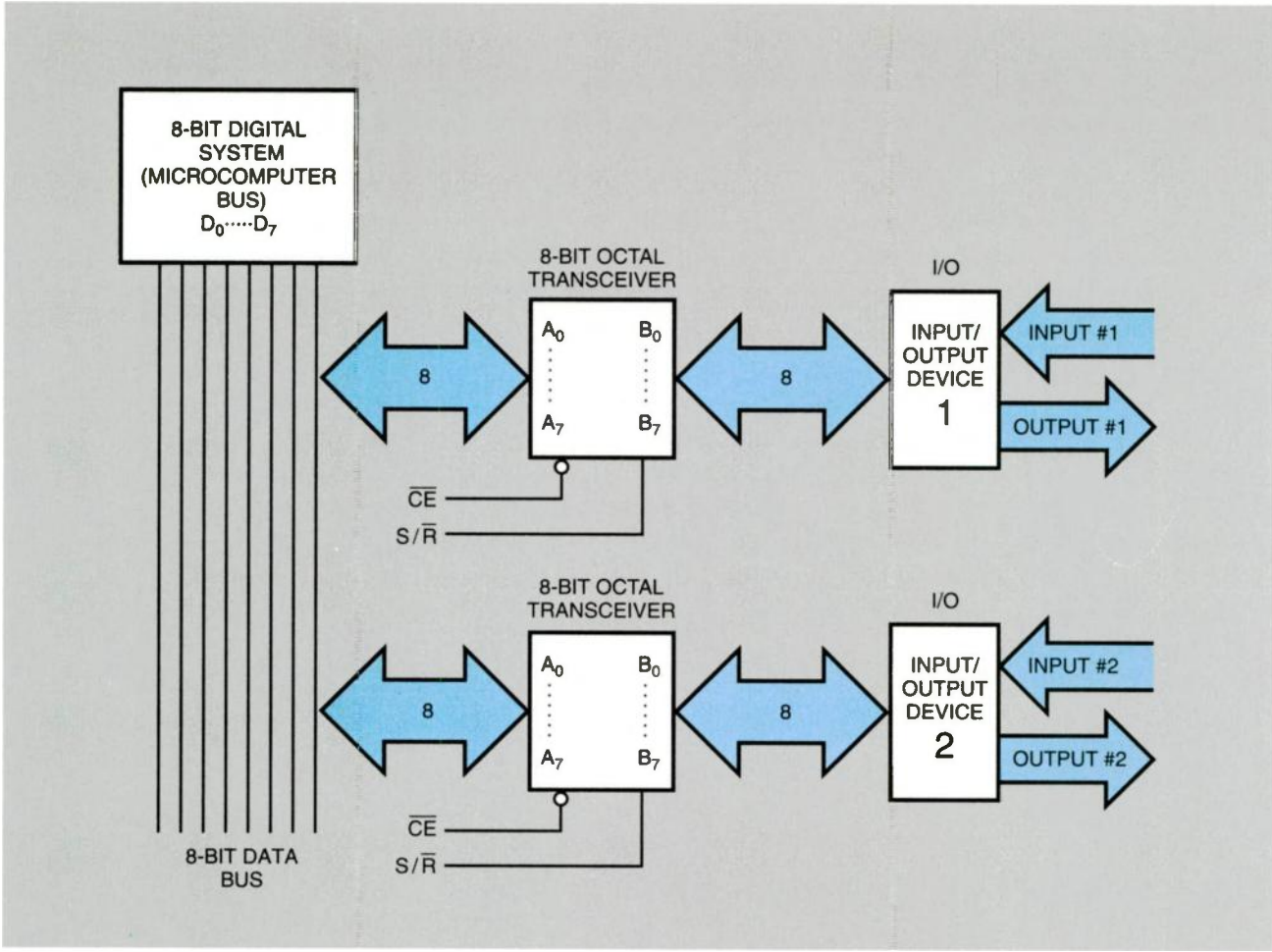


Figure 5-5. An octal transceiver interfaces input/output devices to an 8-bit data bus

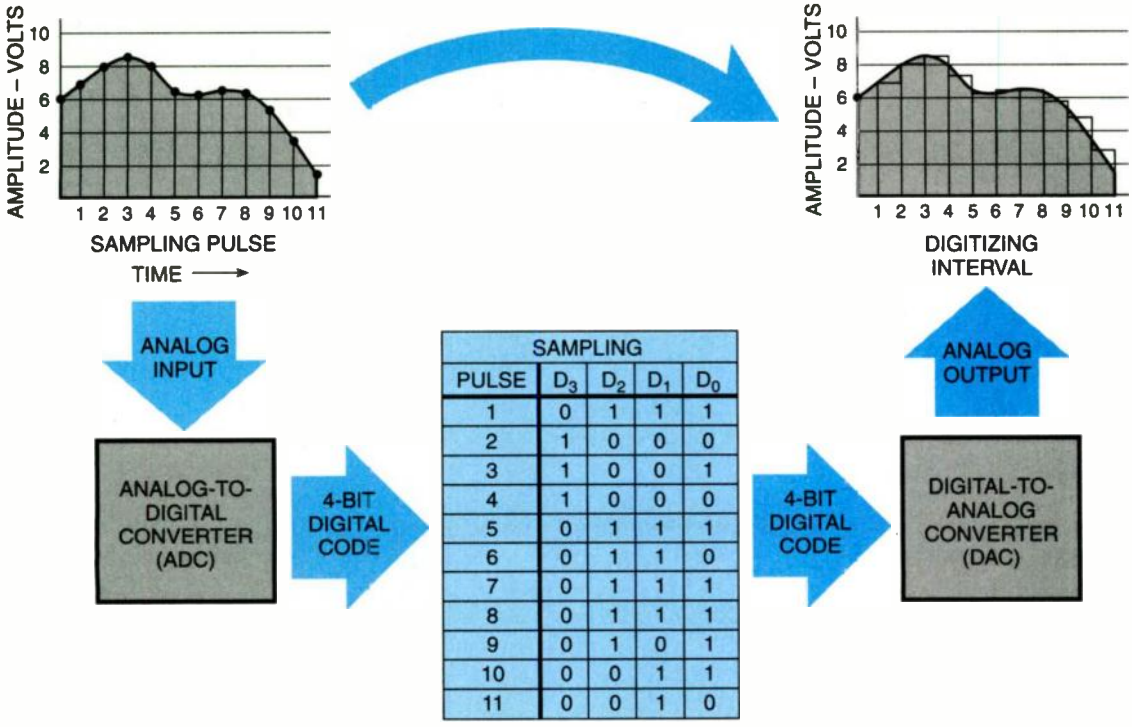
Interfacing the Analog and Digital World

In Chapter 1 we discussed analog and digital quantities at great length. Recall that analog information represents *continuously changing* information but digital information is represented by *two-state* (ON and OFF) conditions. We saw that a great many, probably most, of the physical quantities that we deal with in this world are analog in nature. Before a digital circuit or system can process analog information, it must be converted to digital form by a circuit called an *analog-to-digital converter* (ADC, but sometimes written as A/D converter). Digital system outputs are required to interface with analog system inputs; therefore, the digital output must be converted to an analog output by a circuit called a *digital-to-analog converter* (DAC, but sometimes written as D/A converter).

Basic ADC and DAC Functions

Figure 5-6 summarizes both the A/D conversion and the D/A conversion. An analog signal is input to the ADC, sampled at regular intervals, and the sampled value converted into a digital code to represent the value. A 4-bit output code is shown in *Figure 5-6.*

For the DAC, a digital code is input to the DAC and converted to an analog voltage amplitude represented by the digital code. A 4-bit code is input in *Figure 5-6* and the respective analog voltage is output at the respective digitizing interval.



SAMPLING PULSE	D_3	D_2	D_1	D_0
1	0	1	1	1
2	1	0	0	0
3	1	0	0	1
4	1	0	0	0
5	0	1	1	1
6	0	1	1	0
7	0	1	1	1
8	0	1	1	1
9	0	1	0	1
10	0	0	1	1
11	0	0	1	0

Figure 5-6. An ADC converts an analog signal to a digital code. A DAC converts a digital code to an analog signal.

Sampling

A sampling circuit is a major portion of an ADC. One of the simplest ways to digitize an analog signal is to connect and disconnect the signal path with a switch controlled by a short width (low-duty cycle) pulse as shown in *Figure 5-7a.* All analog-to-digital converters require a period of time to convert the analog input into its digital equivalent. This time, shown as "S" in the figure, is called the conversion time or aperture time. Information will be lost during the time that the switch is open, but if the input is sampled at a frequency that is *greater than twice* that of the analog input frequency, then the original waveform can be reproduced accurately. This is called the *Nyquist criterion.* If the Nyquist criterion is not met, the input signal is not reproduced accurately and low-frequency components, called *alias* frequencies, are added.

Two special cases of sampling are shown in *Figures 5-7b* and *5-7c.* If the sample rate is equal to the analog signal, the sampled signal will be at the same place on every cycle. *Figure 5-7b* shows the sampling at the zero crossing of the input waveform. This obviously is not enough information to reconstruct the original analog signal, so "equal to" two times is generally too slow a sampling rate. A special case where this two times rate *is* adequate is if the sampler is *synchronously phase-locked* to the peaks of the analog signal. This special case is shown in *Figure 5-7c. The point is that the sampling must be greater than twice the input frequency.*

We saw natural sampling without the capacitor C in *Figure 5-7a.* The top of each sample pulse follows the intelligence signal during the pulse-width time of the sampling signal. With the capacitor C in the circuit, as shown in *Figure 5-7a,* the capacitor charges to the input signal during the pulse-width of the sampling signal, then holds the level until the next sample pulse. An amplifier with high input impedance follows the capacitor so that little of the charge will leak off during the time between sampling pulses. The result is a staircase signal, as shown in *Figure 5-7a,* that gives more accurate conversion.

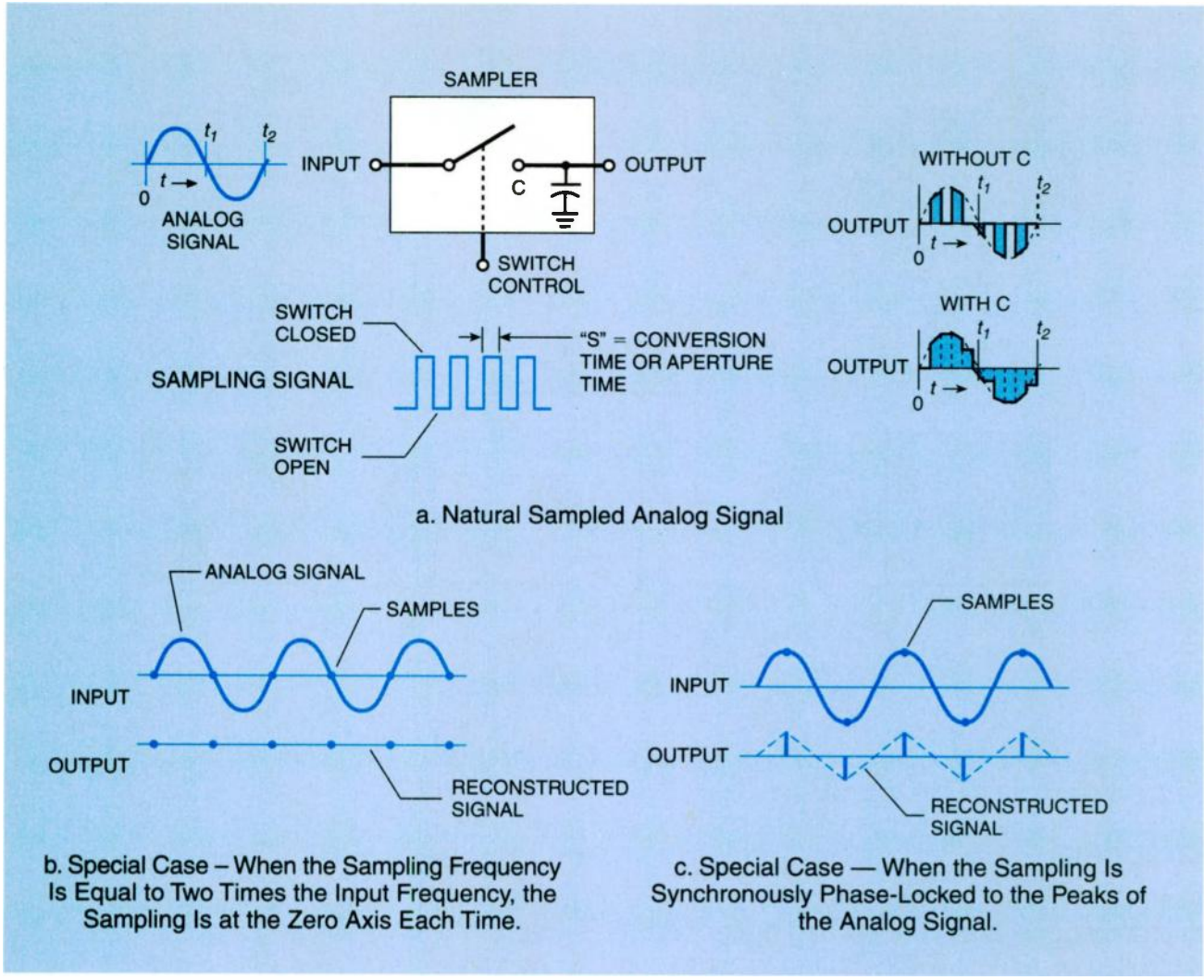


Figure 5-7. Sampling an analog signal.

Before we discuss the actual analog-to-digital converters and the digital-to-analog converters, we need to discuss two integrated circuits that are used extensively in ADCs and DACs. The circuits are the operational amplifier and the comparator. The comparator is an off-shoot of the operational amplifier.

Operational Amplifiers and Comparators

Operational Amplifiers

Most D/A (and many A/D) converters make use of an operational amplifier for signal conditioning. An operational amplifier is a linear amplifier that has two inputs, called inverting (–) and non-inverting (+), and one output. It has three ideal amplifier characteristics: (1) very high voltage gain, (2) very high input impedance, (3) very low output impedance.

When the operational amplifier is used as an *inverting amplifier,* as shown in *Figure 5-8a,* the feedback resistor R_F and the input resistor R_{IN} control the voltage gain according to the following equation. The minus sign means 180° out of phase:

$$\frac{V_{OUT}}{V_{IN}} = -\frac{R_F}{R_{IN}}$$

Thus, the gain of the amplifier can be set just by setting a resistor ratio with resistors external to the amplifier. Very precise gains can be obtained by using precision resistors. Signals that are applied to the inverting input (−), with the non-inverting (+) input grounded, produce an output that is 180° out of phase with the input signal. Signals applied to the non-inverting input (+) produce output signals that are in phase with the input signal.

Example 2. Calculate Voltage Gain of Amplifier

If R_F = 10 kΩ and R_{IN} = 2 kΩ in *Figure 5-8a,* what is the voltage gain of the amplifier? The voltage gain of the inverting amplifier is:

$-R_F / R_{IN} = -10\ k\Omega / 2\ k\Omega = -5$ (the minus sign indicates that the signal is inverted or 180° out of phase).

Comparators

When the operational amplifier is used as a comparator, the two voltages to be compared are applied to the two inputs. When these voltages differ, even by a very small amount, the operational amplifier (comparator) is driven into one of its two saturated output states — HIGH and LOW — depending on which of the input voltages is greater.

Figure 5-8b shows that a comparator is an operational amplifier operating with maximum open-loop gain (no feedback resistor). The inverting input is at the reference voltage, V_{REF}. The voltage to be compared, V_{IN}, is applied to the non-inverting input. The relationship between V_{IN} and V_{REF} is shown in the characteristic response curve of *Figure 5-8b.* When V_{IN} is negative compared to V_{REF}, the output is in a LOW state; when V_{IN} is positive compared to V_{REF}, the output is in a HIGH state. High open-loop gain, wide bandwidth, and fast switching to change states as fast as possible, are characteristics that a digital designer looks for in comparators. Usually a logic gate is placed inside the comparator circuit at the output so the output logic levels are the same as standard TTL gates.

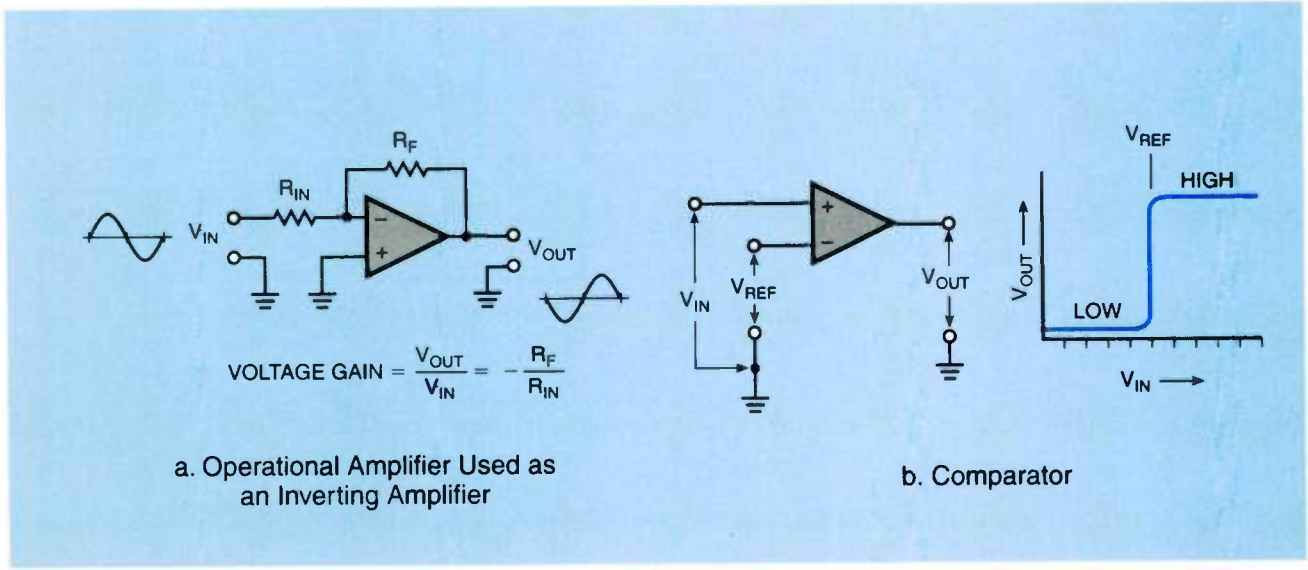


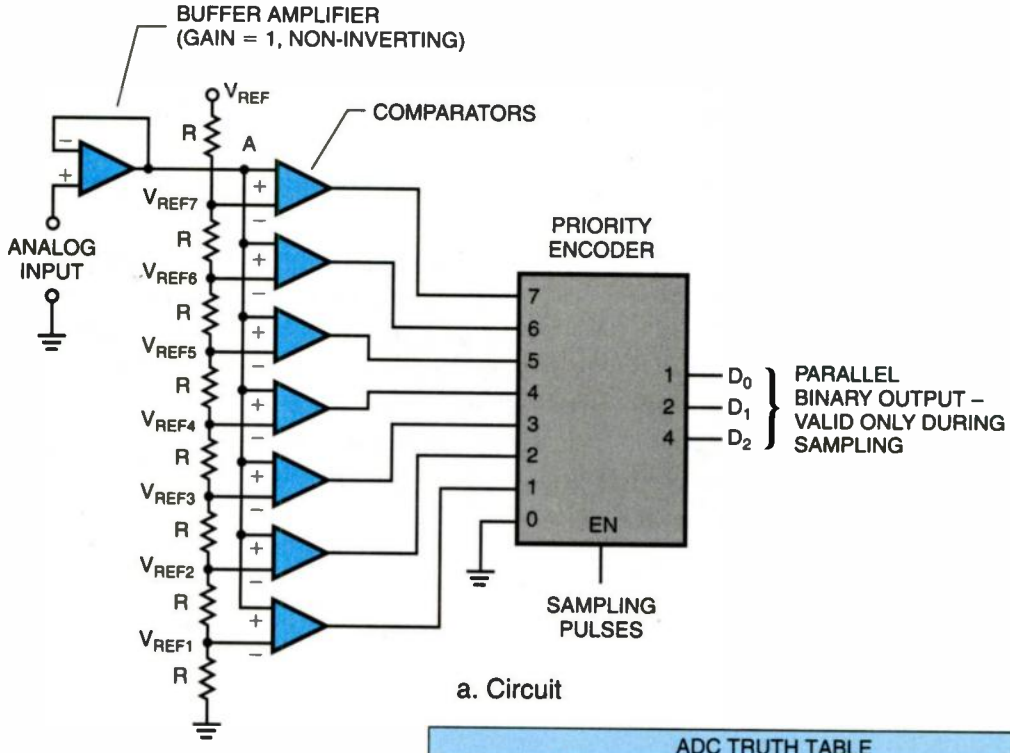
Figure 5-8. Using an operational amplifier.

Analog-to-Digital Converter Circuits

The two broad classifications of analog-to-digital converters are parallel and serial. Parallel converters, also called simultaneous converters, are simple in concept and the faster of the two. They also take more hardware to implement than the serial (sequential) converter, making them more expensive.

A Parallel ADC — The Simultaneous (Flash) Converter

Figure 5-9a shows a basic circuit for a parallel A/D converter using voltage comparators that compare a reference voltage with the analog input voltage. The truth table is shown in *Figure 5-9b*. The total precision reference voltage, V_{REF} determines the step size and, therefore, the *resolution* of the A/D converter. In *Figure 5-9a,* the precision voltage divider divides this reference voltage into eight equal parts. So, a reference voltage of 8 volts would yield a resolution of one volt. Each V_{REF} is one volt in value from the reference voltage next to it.



a. Circuit

ADC TRUTH TABLE											
ANALOG INPUT	ENCODER INPUT								OUTPUT		
V	0	1	2	3	4	5	6	7	D_2	D_1	D_0
0	L	L	L	L	L	L	L	L	0	0	0
1	L	H	L	L	L	L	L	L	0	0	1
2	L	H	H	L	L	L	L	L	0	1	0
3	L	H	H	H	L	L	L	L	0	1	1
4	L	H	H	H	H	L	L	L	1	0	0
5	L	H	H	H	H	H	L	L	1	0	1
6	L	H	H	H	H	H	H	L	1	1	0
7	L	H	H	H	H	H	H	H	1	1	1

b. Truth Table

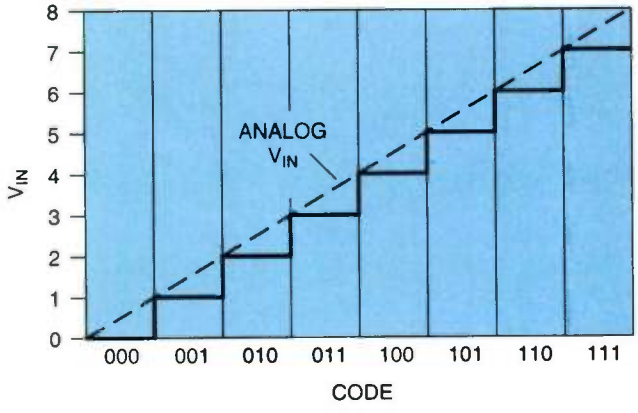
Figure 5-9. Parallel ADC using comparators and encoder.

The analog voltage is applied to the voltage-follower buffer amplifier that has a gain equal to 1, and its output is applied to one input on each comparator. The individual reference voltage for each comparator (its switch voltage) is set by the precision resistive voltage-divider network. With a reference voltage of 8 volts, comparator 1 will switch when the analog input to the buffer is 1 volt, then comparator 2 will switch when the analog input is 2 volts, and so on through each comparator. With the input voltage more negative than V_{REF} at the inverting input, the comparator is a LOW logic level. When the analog input voltage at the non-inverting input is more positive than V_{REF}, the comparator output is a HIGH logic level. The output of each comparator is connected to the input of the priority encoder which produces a 3-bit output representing the value of the analog input. As we discussed above, the sampling rate (remember Nyquist) determines the accuracy with which the digital code represents the analog input signal. The more samples taken in a given unit of time, the more accurately the analog signal is represented in digital form. The output of the priority encoder is valid only during the sampling pulse.

Example 3. Determine Digital Code Output of ADC

Plot the input analog voltage of the ADC of *Figure 5-9* against the output digital code. What is the digital code output when V_{IN} = 5.6 volts? When V_{IN} = 7.3 volts?

V_{IN}	CODE
5.6 V	101
7.3 V	111



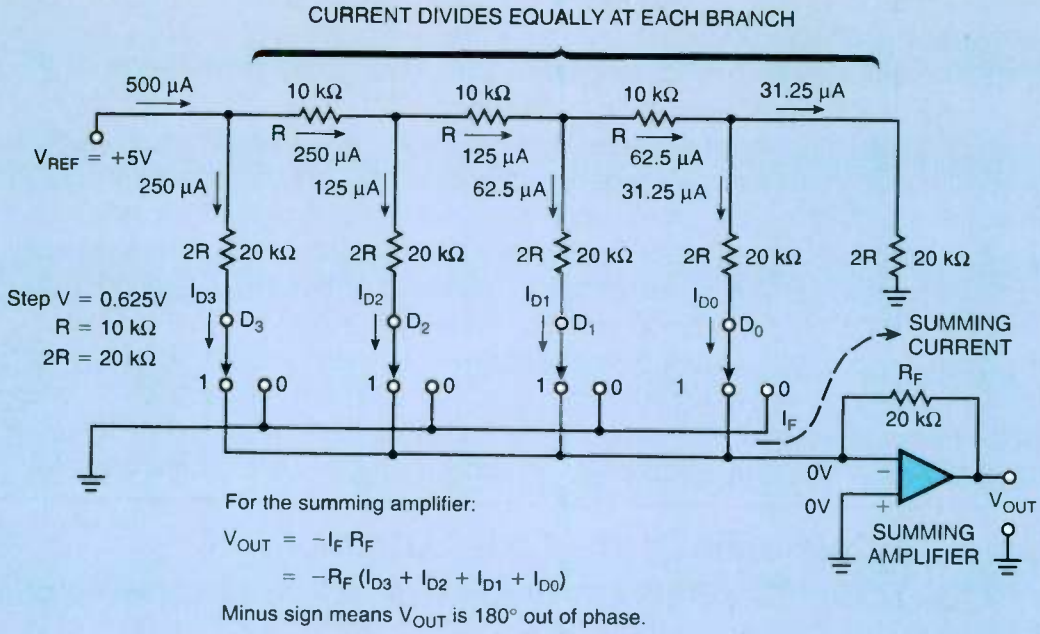
Digital-to-Analog Converters

Digital-to-analog conversion hardware is generally simpler than analog-to-digital conversion hardware. Also, many ADCs include DACs as part of their circuitry. Let's look at a DAC using operational amplifiers.

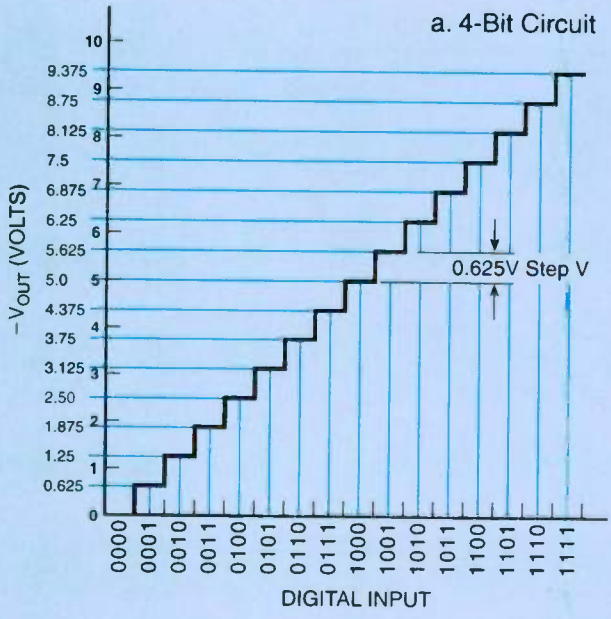
R/2R Ladder D/A Converter

D/A conversion in integrated circuits is most often accomplished with an operational amplifier used as a summing amplifier. The circuit consists of a resistor network and an operational amplifier. A commonly used resistor network is called a R/2R ladder circuit. It requires two different, but closely matched, resistor values. It can be fabricated easily on integrated circuits for converters of 8, 10, 12, or even more bits of resolution. For simplicity, we will look at only a 4-bit resolution converter as shown in *Figure 5-10a*. The four bits of digital information to be converted to analog are entered on the data switches D0 to D3. The R/2R ladder circuit forms a binary-weighted current-divider. Let's go through the operation. We represent the logic input as switches, but logic gate outputs can supply the inputs.

We need to recall some of the things we have learned about operational amplifiers. First, the input impedance of the op-amp is very high. Any current that reaches the inverting input will continue past it and through R_F, the summing resistor.



a. 4-Bit Circuit



D3	D1	D2	D0	$-V_{OUT}$ (V)
0	0	0	0	0.000
0	0	0	1	0.625
0	0	1	0	1.250
0	0	1	1	1.875
0	1	0	0	2.500
0	1	0	1	3.125
0	1	1	0	3.750
0	1	1	1	4.375
1	0	0	0	5.000
1	0	0	1	5.625
1	0	1	0	6.250
1	0	1	1	6.875
1	1	0	0	7.500
1	1	0	1	8.125
1	1	1	0	8.750
1	1	1	1	9.375

b. V_{OUT} vs. Digital Input (4-Bit Conversion)

Figure 5-10. R/2R ladder DAC uses precision resistors and operational amplifier.

Second, since there is no input current from the − to + terminals of the op-amp (very high input impedance), both of the inputs are at the same potential; therefore, since the + input is at ground (0 volts), the inverting input (−) is at virtual ground (0 volts). As a result, notice that the position of any of the data switches does not affect the current through any rung of the ladder because when a data switch is in the 1 position, it is connected to virtual ground, and when it is in the 0 position, it is connected to ground. As shown in *Figure 5-10a,* the current I_F is made up of the sum of the branch currents, I_{D0}, I_{D1}, I_{D2}, I_{D3}, and V_{OUT} equals $I_F \times R_F$.

To find the total current from the reference voltage V_{REF}, we must find the equivalent resistance of the ladder. Starting at the right side of the ladder, we find two 2R resistors in parallel which are equal to R (for our circuit R = 10k and 2R = 20k). Now this equivalent R is in series with R making 2R again in parallel with the next 2R, and so forth all the way down the ladder to the left rung. The equivalent resistance seen by V_{REF} is 10 kΩ. So the total current leaving V_{REF} is 5V / 10 kΩ = 500 μA. Now, here is the beautiful part. When the current reaches the first rung, it divides equally into the two branches because each branch is 20 kΩ. Then the current (250 μA) that reaches the second rung divides equally again with 125 μA down the second rung. This divide-by-two of the current continues for each rung of the ladder forming a binary-weighted ratio that is then available to the operational amplifier summing resistor R_F.

Analog Voltage Output for Binary Input

If any of the switches (D_0 to D_3) are in position 1, the current in that rung is routed through the summing resistor. If a switch is in position 0, then the current in that rung is routed to ground and, therefore, does not contribute to the voltage output, $V_{OUT} = I_F R_F$. The formula for the output voltage for a R/2R ladder network DAC converter is determined by the binary number formed by the switches, the amount of the reference voltage, and the number of bits, N. The equation is:

$$V_{OUT} = -(\text{decimal equivalent of binary number input})(V_{REF}/2^N)$$

The negative sign results from the inverting operational amplifier. Here are two examples using the equation.

Example 4. Output Voltage for DAC Using R/2R Ladder

What would be the output voltage of an 8-bit R/2R DAC if the binary number equal to 78_{10} (01001110_2) is on the digital input? The reference voltage is 5V; N = 8; $2^N = 256$.

$$V_{OUT} = -(\text{decimal equivalent of binary number})(V_{REF}/2^N)$$

$$= -(78)\frac{5V}{256}$$

$$= -1.523V$$

Example 5. Output Voltage for DAC Using R/2R Ladder

What would be the output voltage of a 4-bit R/2R DAC if the binary number equal to 12_{10} (1100_2) is on the digital input? The reference voltage is 10V; N = 4; $2^N = 16$.

$$V_{OUT} = -(\text{decimal equivalent of binary number})(V_{REF}/2^N)$$

$$= -(12)\frac{10}{16}$$

$$= -7.500\,V$$

We can use the output voltage equation to plot the values for each value of binary count from 0000 to 1111. *Figure 5-10b* shows the results.

The formula for the analog output voltage for each binary input can be seen to be the sum of the output voltages generated by all of the individual weighted-bit outputs.

Example 6. Weighted Outputs for V_{OUT}

What are the individual weighted-bit outputs and the sum of these for a binary input of 1010_2? Check your answer by referring to *Figure 5-10b.*

WEIGHT	INPUT	
8	1	MSB = 1 × 8 × 10/16 = 5.00V
4	0	2nd MSB = 0 × 4 × 10/16 = 0.00V
2	1	3rd MSB = 1 × 2 × 10/16 = 1.25V
1	0	LSB = 0 × 1 × 10/16 = 0.00V
		SUM = 6.25V = the analog output voltage

(Remember the negative sign is due to the inverting op-amp.)

A Popular DAC

A very popular and inexpensive integrated circuit DAC is shown in *Figure 5-11* with a 741 op-amp providing the output analog voltage. This DAC requires a reference current of 2 mA which is provided by V_{REF} = 10V and the 5 kΩ resistor on pin 14. The negative reference (pin 15) is returned to ground through the same size (5 kΩ) resistor. The output current through R_F develops the output voltage equal to I_{OUT} R_F.

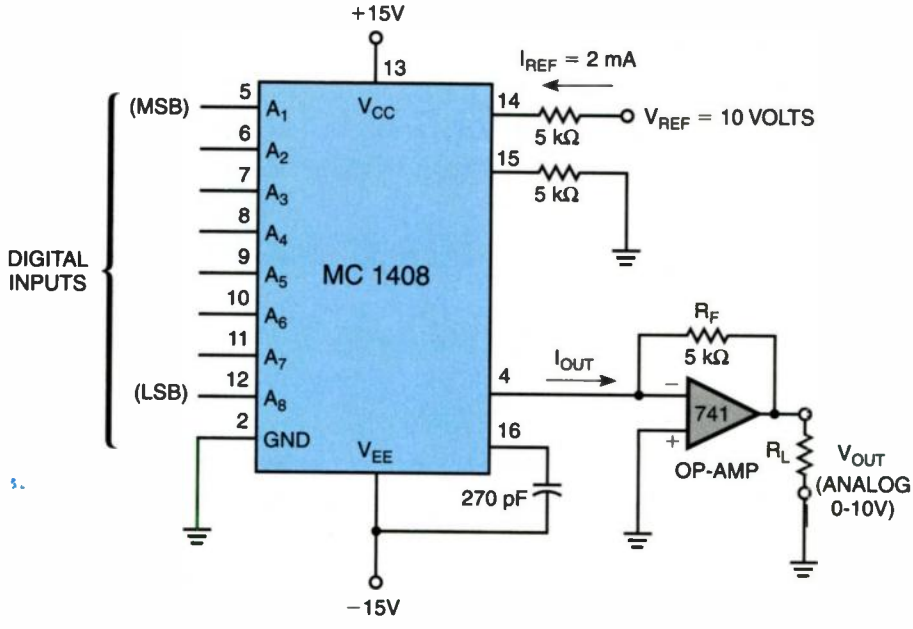


Figure 5-11. An 8-bit integrated circuit DAC converter with a 741 op-amp providing the output analog voltage.

An ADC That Uses a DAC

There are a number of ADCs that use a DAC in the circuit. Let's look at one of these — the successive-approximation ADC. It also has a serial binary output.

Successive-Approximation Analog-to-Digital Converter

The successive-approximation analog-to-digital converter has a much shorter conversion time than other methods with the exception of the flash method. *Figure 5-12* shows that this converter consists of a D/A converter, a successive-approxima-

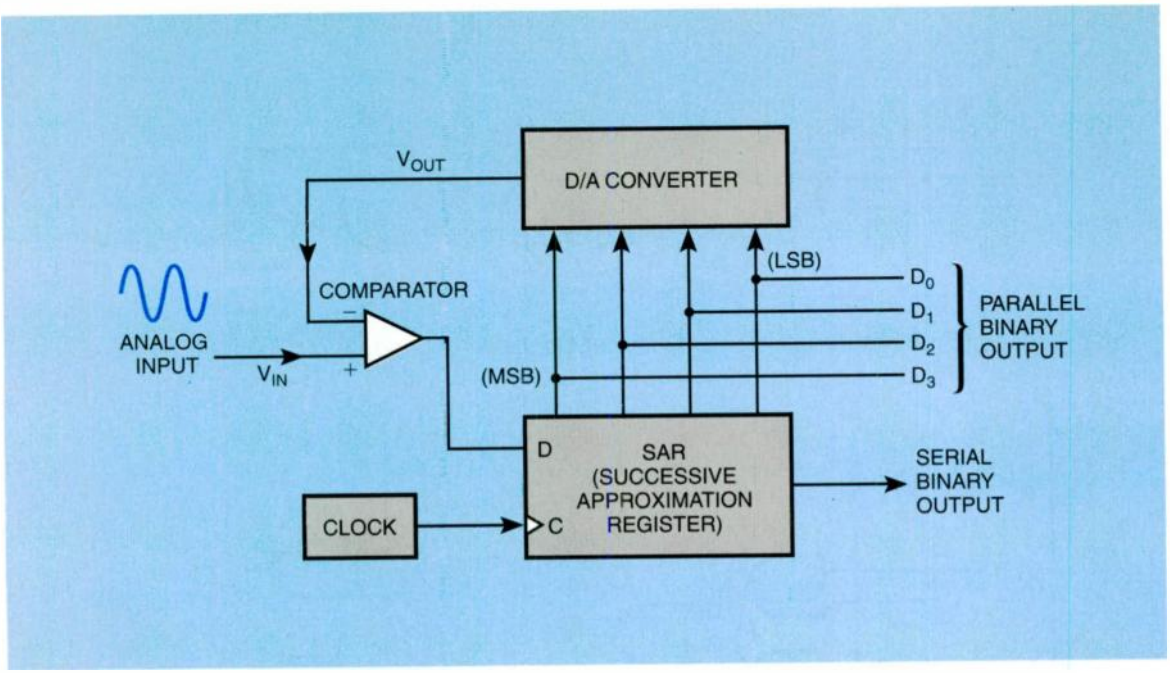


Figure 5-12. Successive-approximation A/D converter.

tion register (SAR), and a comparator. The A/D conversion process begins by entering a 1 into the MSB of the SAR so that it would feed a binary code of 1000 to the DAC. The DAC output is then compared with the analog input. If the register contents represent a value of V_{OUT} larger than the analog input voltage, the output of the comparator is LOW (0), causing the MSB of the SAR to be RESET to a 0. If the register contents represent a V_{OUT} value less than the analog input voltage, the comparator output is HIGH (1), then the SAR holds the 1 in its MSB. A 1 is then tried in the next most significant bit. If the register contents represent a value less than the analog voltage as indicated by the comparator, the next most significant bit is held as a 1. If the register contents are equivalent to a V_{OUT} that is greater than the analog voltage, the flip-flop is RESET to 0. This process is continued down to the least significant bit (LSB). When all of the bits have been tested in this fashion, the register contains the binary digital equivalent of the analog voltage. The binary output is available either in parallel or serial form.

How Do Digital Circuits Compute

All arithmetic operations from the simplest to the most complex can be reduced to addition, so let's see how digital circuits perform addition. Then, we'll see how the adder can be used for subtraction, multiplication, and division.

Half-Adder Circuit

Recall from Chapter 2 there are four basic rules for adding binary digits:

Rule 1	0 + 0 = 0	
Rule 2	0 + 1 = 1	
Rule 3	1 + 0 = 1	
Rule 4	1 + 1 = 10	(sum is 0 with a carry of 1)

We can take these rules directly and make a logic circuit truth table from them as shown in *Figure 5-13a.* From the truth table, the decoder circuit equations for the sum, S, and the carry, C, can be written as shown in *Figure 5-13b.* The decoder circuit

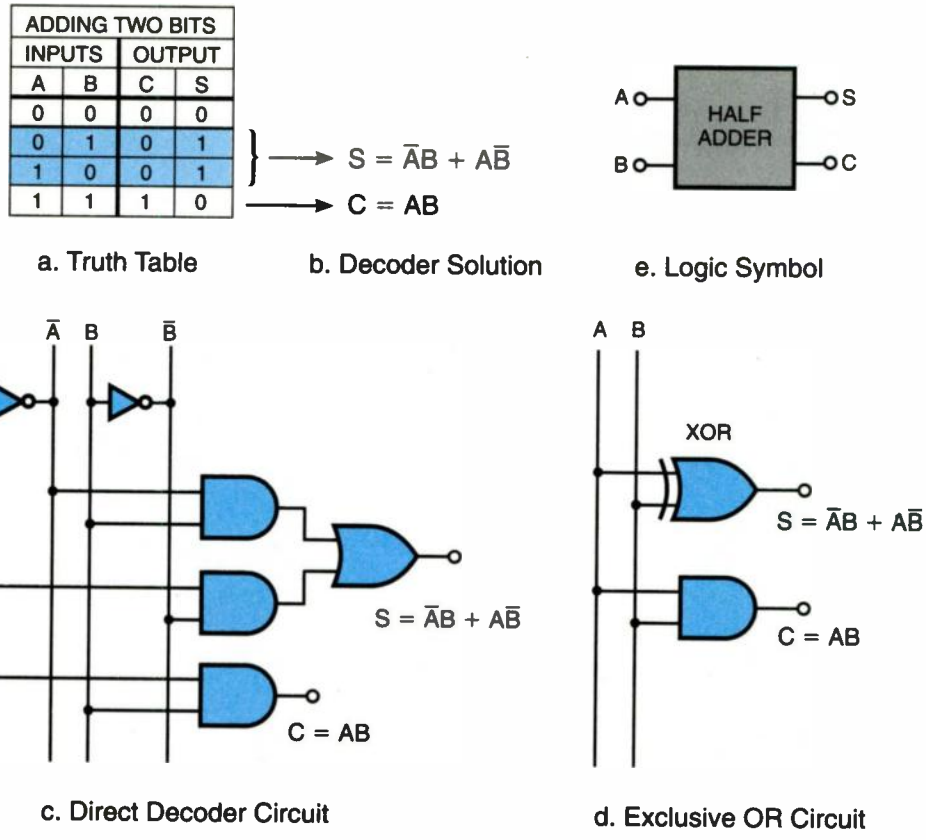


Figure 5-13. Half-adder — A decoder circuit to add two 1-bit binary numbers and produce a sum and carry output.

is shown in *Figure 5-13c* using AND, OR, and INV gates. If we are sharp, we will recognize that the S function is the same as an XOR gate output. Therefore, the circuit can be built as shown in *Figure 5-13d.* The logic symbol for a half-adder is shown in *Figure 5-13e.* Note that there are four operations corresponding to the four rules.

It is called a *half-adder* because it does not have the provisions for adding a carry from the previous addition. Most binary numbers that are to be added have more than one bit, so a sum would pass the carry from one bit position to the next. A circuit that can accept a carry from a previous addition and add two bits is called a *full-adder.*

Full-Adder Circuit

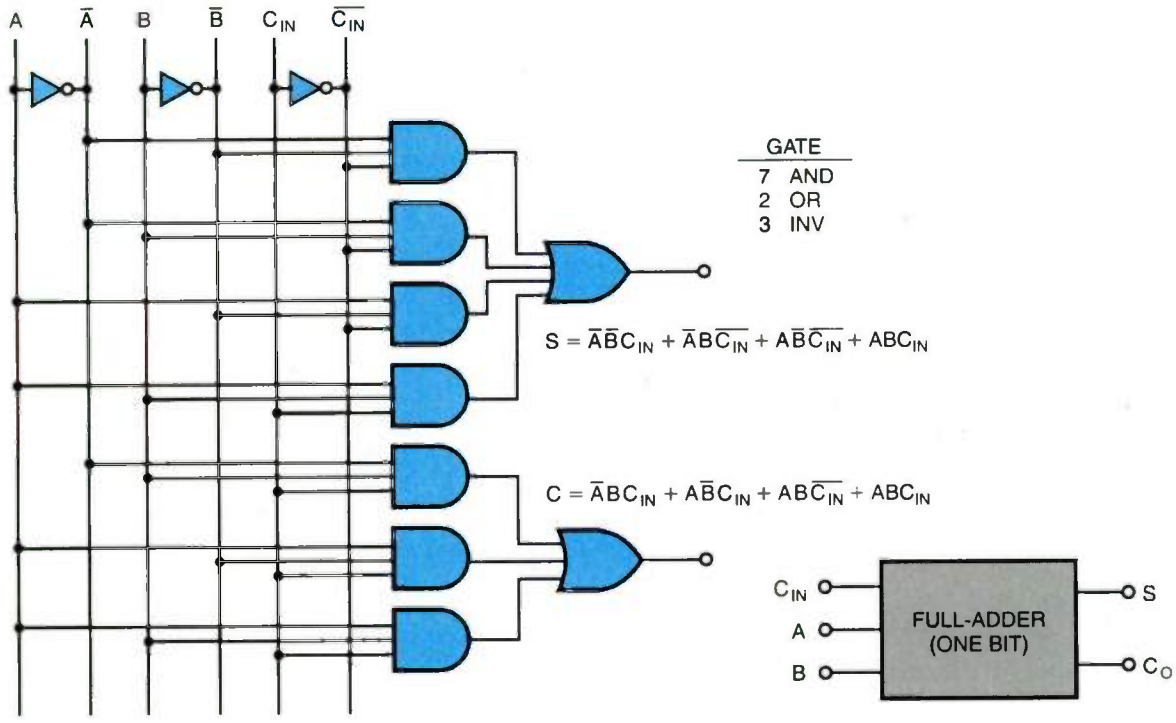
Whereas the half-adder adds two inputs, the full-adder adds three inputs. For example, it can add A and B ***and*** a carry from a previous addition, with an output of a sum and a carry. The functional block diagram symbol and truth table for the full-adder is shown in *Figure 5-14.* In the truth table of *Figure 5-14a,* we added an input column for the incoming carry, C_{IN}. The sum, S, and the carry, C_O, are still the outputs.

A	B	C_{IN}	S	C_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

a. Truth Table

SUM	CARRY
$S = \bar{A}\bar{B}C_{IN}$	
$+ \bar{A}B\overline{C_{IN}}$	
	$C = \bar{A}BC_{IN}$
$+ A\bar{B}\overline{C_{IN}}$	
	$+ A\bar{B}C_{IN}$
	$+ AB\overline{C_{IN}}$
$+ ABC_{IN}$	$+ ABC_{IN}$

b. Decoder Solution



c. Decoder Solution

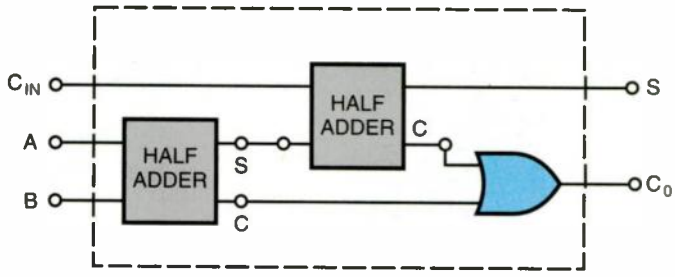
d. Logic Symbol

Figure 5-14. Full-adder — A decoder circuit to add two 1-bit binary numbers plus a previous carry and produce a sum and carry output.

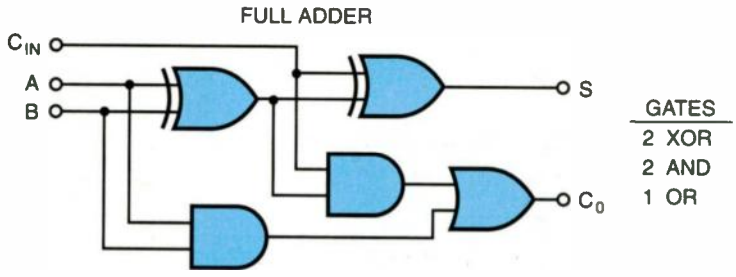
The same procedure can be followed as for the half-adder to arrive at a direct decoder circuit. The solution is shown in *Figure 5-14*. The circuit uses seven AND, two OR, and three INV gates. Can it be done simpler? Since we have half-adders, could we use two to make a full adder? The solution is shown in *Figure 5-15*. The gate count has been reduced from 12 in *Figure 5-14c* to 5 for *Figure 5-15b*.

Adders for Multiple-Bit Binary Numbers

To provide adders for multiple-bit binary numbers, half-adders and full-adders are ganged together to handle as many bits as needed. An example of a 4-bit adder is shown in *Figure 5-16*. Note that the LSB adder is a half-adder because there is no need for a carry input. However, if the adder is to be used for subtraction, the LSB adder will likely be a full-adder because the input of a 1 on the carry input is an easy way to form a two's complement needed in subtraction.



a. Full Adder



b. Exclusive OR Circuit for Full Adder

Figure 5-15. Full-adder using two half-adders.

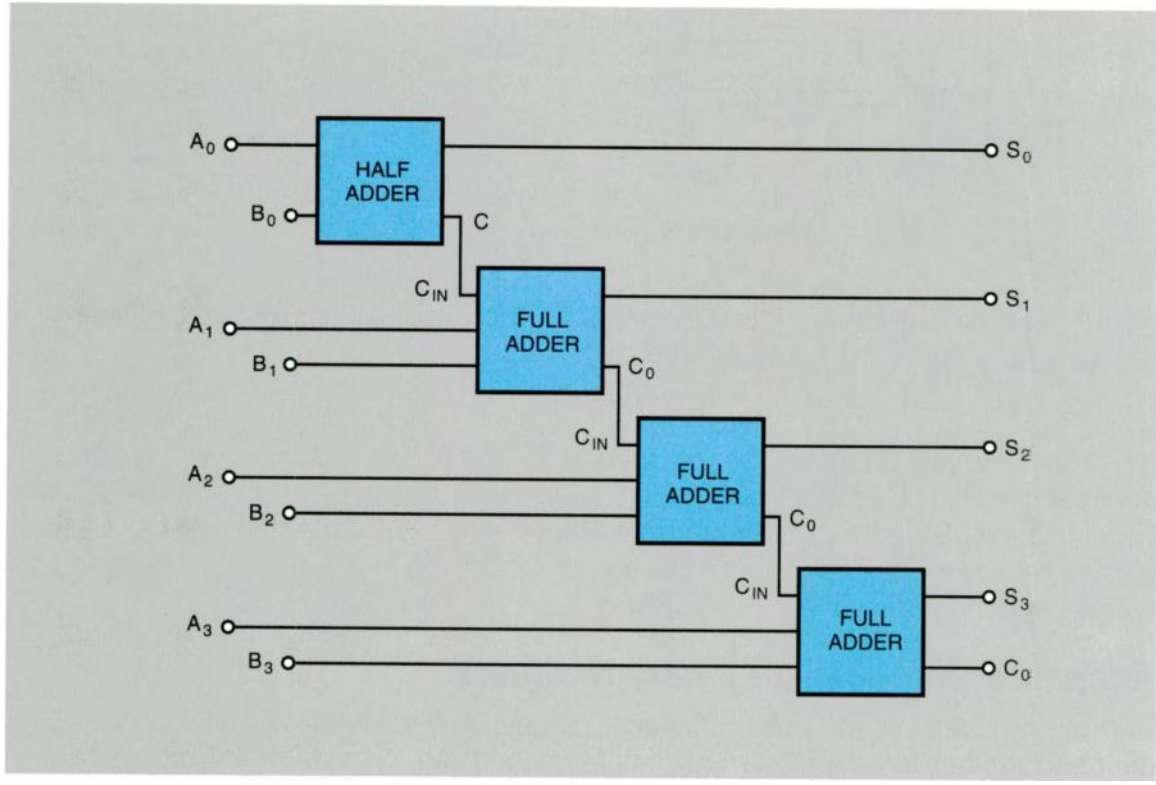


Figure 5-16. A 4-bit adder — Half- and full-adders are ganged together to form multiple-bit adders for binary numbers with 4, 8, 16, 32 or 64 bits.

Parity Bits

Communications using digital techniques need to be accurate. For this reason, the binary transmissions have a check bit in them called a parity bit. Notice from the truth table of the full-adder that the sum is 1 each time that the total number of inputs A, B, and C_{IN} is odd. In communication transmissions, a parity bit is sometimes used for error detection and correction. Since the output is 1 for an odd number of input 1s, this circuit can be used for an even parity generator. A parity bit is attached to the group of information bits to make the total number of 1s always even or always odd. An even parity bit makes the total number of 1s even, and an odd parity bit makes the total number of 1s odd. By checking the binary information after it is transmitted for a given parity, single-bit errors in transmission can be detected and corrected.

Subtraction

Subtraction of binary numbers is most easily done by the method known as two's complement that we described in Chapter 2. Recall that this is essentially subtraction by changing the sign and adding. A circuit to do the subtraction is shown in *Figure 5-17*, as well as addition and subtraction examples. It is an 8-bit adder which has a full-adder with a C_{IN} input in the LSB stage. Data selectors provide the one's complement for the B binary number on subtraction and $C_{IN} = 1$ to provide the two's complement addition to complete the subtraction. Checks are made with decimal numbers. In the majority of digital arithmetic systems, subtraction is performed by this two's complement addition, simply because adders are readily available. There are, however, some circumstances where it is better to use circuits that are designed to subtract without complementing.

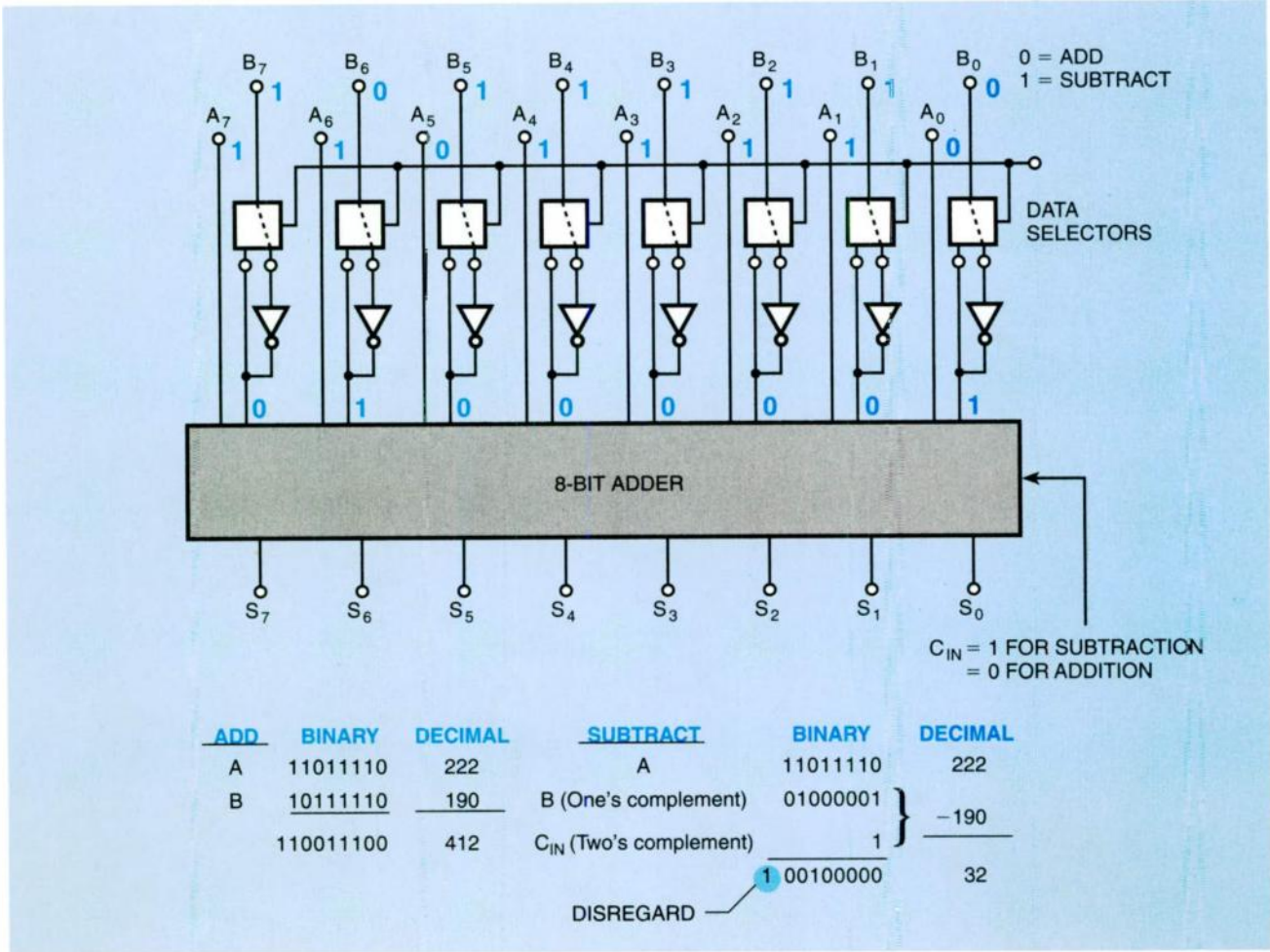
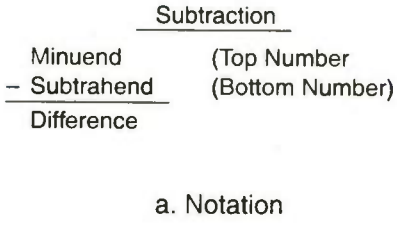


Figure 5-17. 8-bit adder using two's complement addition for subtraction.

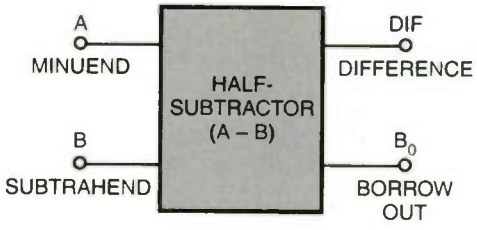
If we use the same technique as we used for the adder to develop a subtractor, let's again start with the rules of Chapter 2:

Rule 1	0	–	0	=	0	
Rule 2	0	–	1	=	1	(with a borrow 1)
Rule 3	1	–	0	=	1	
Rule 4	1	–	1	=	0	

From the rules, we can develop the truth table and the XOR circuitry for the half-subtractor shown in *Figures 5-18c* and *5-18d*. The logic symbol is shown in *Figure 5-18b*. A full-subtractor and its truth table are shown in *Figure 5-19*.



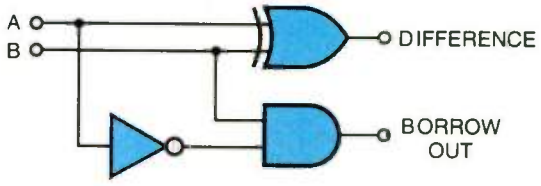
a. Notation



a. Logic Symbol

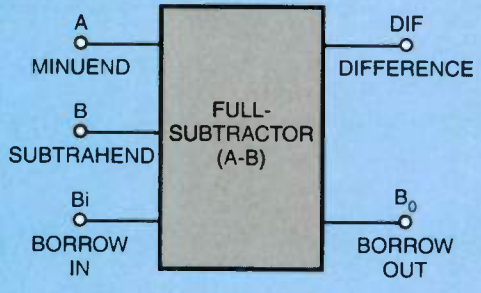
A	B	DIFF	BORROW OUT
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

c. Truth Table



d. XOR Circuit

Figure 5-18. Half-subtractor — A decoder circuit to subtract two 1-bit binary numbers and produce a difference and borrow out output.



a. Logic Symbol

INPUTS			OUTPUTS	
A	B	Bi	DIF	B_0
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

An additional borrow input (Bi) is added to a half-subtractor to allow a borrow input from a previous stage of a multiple-bit subtractor.

b. Truth Table

Figure 5-19. Full-subtractor — An additional borrow input (Bi) is added to a half-subtractor to allow a borrow input from a previous stage of a multiple-bit subtractor.

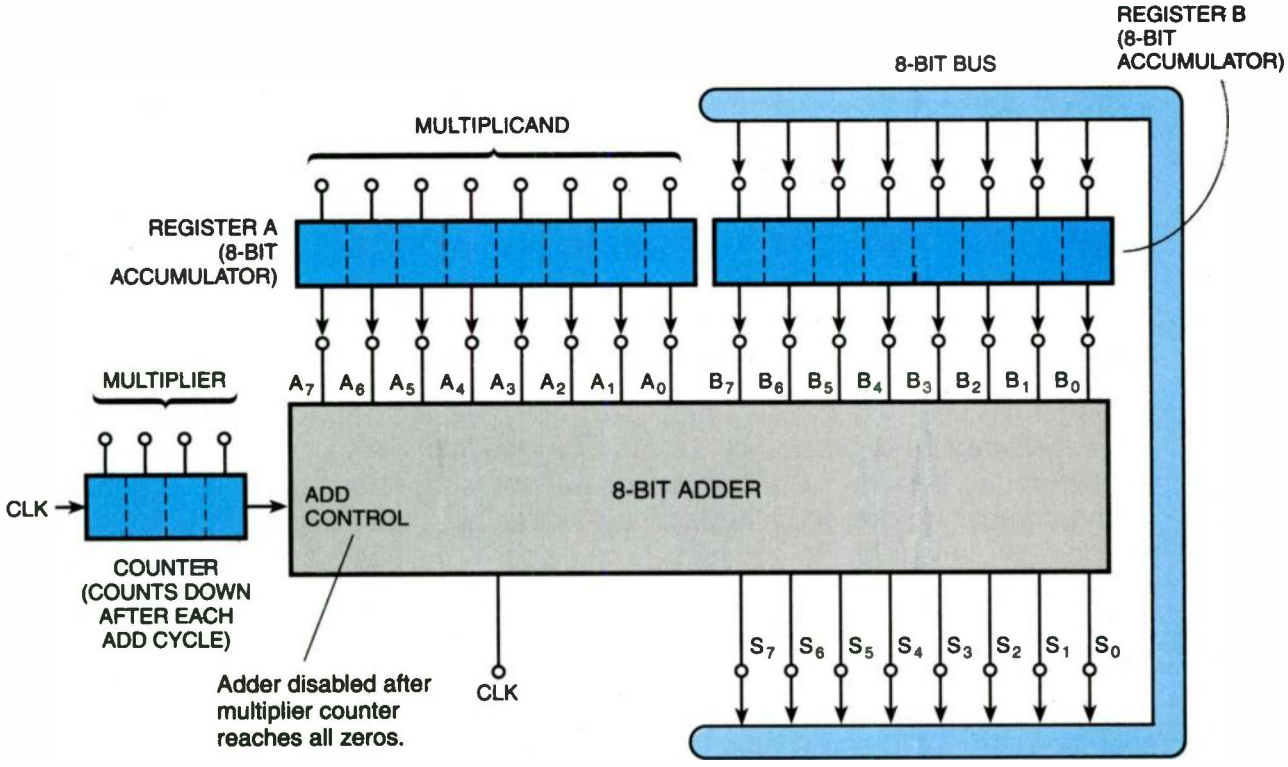
Multiple-bit subtractors can be made using a half-subtractor for the LSB and a full-subtractor for each higher bit just as we did for the adder. The Borrow Out of each subtractor is connected to the Borrow In of the next higher bit. And as with the adder, a full-subtractor can be made from two half-subtractors.

Multiplication by Successive Addition

Digital computers can perform addition extremely rapidly, and, therefore, can be programmed to multiply by a series of successive addition steps. The adder circuits add the two numbers and store the partial products in an accumulator register for the next addition.

A binary multiplier using the repeated addition method is shown in *Figure 5-20.* An 8-bit adder, two registers, and a counter make up the circuit. A clock synchronizes the operation. The multiplication cycle is given in *Figure 5-20.*

The steps of decreasing the counter by 1, ADD, and Load Accumulator are repeated until the counter is at all zeros. One ADD and Load Accumulator provides the final value in register B.



MULTIPLICATION CYCLE

1. Load register A with multiplicand
2. Load counter with multiplier
3. Clear register B (accumulator) to all zeros
4. Decrement counter by 1
5. Add
6. Load accumulator with sum
7. Decrement counter
8. Add
9. Load accumulator with sum
10. Continue steps 7, 8 and 9 until counter has all zeros
11. Add
12. Load accumulator with sum
13. Final value in accumulator

Figure 5-20. Multiplier for two 4-bit binary numbers using successive addition.

Example 7. Multiplication by Successive Addition

Using the multiplier of *Figure 5-20,* go through the steps of the multiplication cycle until the result is obtained. Multiply 8 × 8.

STEP	MULTIPLICAND Register A	MULTIPLIER Counter	ACCUMULATOR Register B	SUM	
1	1000				Load Multiplicand
2	1000	1000			Load Multiplier
3	1000	1000	00000		Clear Accumulator
4	1000	0111	00000		Decrement Counter
5				1000	Add
6	1000	0111	01000		Load Accumulator
7	1000	0110	01000		Decrement Counter
8				10000	Add
9	1000	0110	10000		Load Accumulator
10	1000	0101	10000		Decrement Counter
11				11000	Add
12	1000	0101	11000		Load Accumulator
13	1000	0100	11000		Decrement Counter
14				100000	Add
15	1000	0100	100000		Load Accumulator
16	1000	0011	100000		Decrement Counter
17				101000	Add
18	1000	0011	101000		Load Accumulator
19	1000	0010	101000		Decrement Counter
20				110000	Add
21	1000	0010	110000		Load Accumulator
22	1000	0001	110000		Decrement Counter
23				111000	Add
24	1000	0001	111000		Load Accumulator
25	1000	0000	111000		Decrment Counter
26				1000000	Add
27	1000	0000	1000000		Final Value – 64 Decimal

Division by Successive Subtraction

An adder has been used to do multiplication by repeated addition. This procedure is also easily adapted to division by a series of subtractions. Also, these same techniques can be used to perform more complex operations, such as extracting roots and raising numbers to powers. Many of these operations that we have discussed up to this point (both logical and arithmetic) can be performed with one special integrated circuit called an ALU.

Arithmetic-Logic Unit

An arithmetic logic unit (ALU) performs addition and subtraction as well as logical operations, like AND and OR, on the data applied to its inputs. The basic ALU in the 7400 TTL series is the 74181. In one mode, the ALU performs logic operations, such as AND, OR, NAND, NOR, or XOR. In another mode, the ALU performs arithmetic operations, such as add, subtract, and compare. The control inputs select 16 different arithmetic operations or 16 different logical operations. It is a very useful circuit and is used in all computers and most complex digital control systems. Whenever a microprocessor is used in a digital system, there is an ALU because the microprocessor contains an ALU.

Summary

Now that we understand some basic details of coupling, converting, and computing circuits, let's look at more permanent storage circuits.

Quiz for Chapter 5

1. A special type of logic gate to isolate conventional gates from other circuits is a/an
   a. XOR
   b. totem-pole
   c. buffer
   d. bistable
2. Totem-pole outputs have this characteristic:
   a. they cannot be tied directly together.
   b. they are 3-state.
   c. they are easily interconnected.
   d. they are sometimes called open-collector.
3. Several gate outputs can be connected directly to the same bus using
   a. totem-pole outputs
   b. TTL and gates
   c. separate load resistors
   d. 3-state buffers
4. A 3-state buffer has the output states LOW, HIGH, and _____ .
   a. MEDIUM
   b. high-impedance
   c. low-impedance
   d. short circuit
5. A device that transmits data to the bus is called a
   a. driver
   b. talker
   c. port
   d. controller
6. The Nyquist criterion is met if an input analog signal is sampled at a frequency
   a. equal to the highest input frequency.
   b. just greater than the input frequency.
   c. one and one-half times the input frequency.
   d. greater than twice the input frequency.
7. A comparator is actually an operational amplifier with
   a. one input.
   b. a gain of one.
   c. a gain of less than one.
   d. no feedback resistor.
8. Usually a logic gate is placed at the output of a comparator so the output level is
   a. the same as standard TTL gates.
   b. at the power supply voltage level.
   c. at ground level.
   d. greater than the power supply.
9. A parallel A/D converter using voltage comparators and a reference voltage is the
   a. R/2R converter.
   b. summing converter.
   c. flash converter.
   d. universal converter.
10. A circuit that can accept a carry from a previous addition and add two bits is called a
    a. summing amplifier
    b. full-adder
    c. half-adder
    d. look-ahead adder
11. A _____ bit may be added to a group of bits to make the total number of 1s either even or odd for error detection.
    a. carry
    b. borrow
    c. sign
    d. parity
12. Subtraction of binary numbers is most easily done by the method of
    a. successive addition.
    b. two's complement.
    c. look-around carry
    d. add-and-accumulate
13. A binary multiplier is usually formed using
    a. three AND gates and inverters.
    b. adders and accumulators
    c. inverters and half-subtractors
    d. XOR gates in series
14. A/An _____ performs addition and subtraction as well as logic operations.
    a. universal counter
    b successive-approximation A/D
    c. successive-approximation D/A
    d. arithmetic-logic unit
15. Two inputs are common to an XOR gate and an AND gate. Resulting outputs are _____ and _____ respectively.
    a. true, false
    b. sum, carry
    c. complement, NOT
    d. AND, OR

Answers:
1c, 2a, 3d, 4b, 5b, 6d, 7d, 8a, 9c, 10b, 11d, 12b, 13b, 14d, 15b

Questions and Problems for Chapter 5

1. a. What is a buffer used for?

   b. Name a popular buffer.

2. Explain why totem pole outputs cannot be tied directly together.

3. What does it mean to say that a 3-state buffer is "disabled"?

4. On the 74S245 bus transceiver, can data go both ways at the same time? Explain.

5. What determines the minimum sampling frequency?

6. If the voltage gain of an inverting amplifier is $V_{out}/V_{in} = -R_F/R_{IN}$, what is the gain when $R_F = 50k\Omega$ and $R_{IN} = 5k\Omega$.

7. What characteristics does a designer look for in comparators?

8. What would be the digital code output for a $V_{in} = 3.5$ volts for the ADC of *Figure 5-9?*

9. What would be the output voltage of an 8-bit R/2R DAC if the binary number equal to 56_{10} is on the digital input? The reference voltage is 5V, N = 8.

10. For the R/2R ladder DAC of *Figure 5-10,* what are the individual weighted-bit outputs and the sum of these for a binary output of 1001_2.

11. Refer to the DAC of *Figure 5-10,* with a binary input of 01010100_2 to this type of DAC, what would V_{out} be if the reference voltage equals 10V?

12. Prove with a truth table that XOR, AND and OR gates can be combined as in *Figure 5-15b* to make a full adder.

CHAPTER 6

More Permanent Storage Elements

More Permanent Digital Storage

In Chapter 2, the function of storage circuits in digital electronic systems was discussed in which digital information is stored for use at a later time, either temporarily or on a more permanent basis. In Chapter 4, temporary storage circuits were discussed. In this chapter, we will deal with digital circuits that provide more permanent storage of digital information.

Recall that *Figure 2-14* showed how bistable memory cells (cells that store either a 0 or a 1) are combined into a matrix of storage elements, which provide, using today's integrated circuit technology, 4, 16, or even 64 megabits of storage capacity in one integrated circuit package — that's 64 million bits of storage capacity! Recall, also, that in Chapter 2, random access memory (RAM) and read-only memory (ROM) were introduced. We will explain both of these memory types in more detail in this chapter.

There is another category of memory used in digital electronic systems that has not been mentioned. It is called mass memory and may be sequential access or random access. It is the media that holds digital information permanently for long periods of time, even off-line or separated from the digital system in which it is used. Punched paper tape, punched cards, magnetic tape, magnetic disks, and optical disks have progressively served this memory category. Magnetic disk storage, which is a type of random-access memory, with capacities up to 2 gigabytes are common in personal computer systems of today. We will not deal with mass memory, but keep our discussion to permanent storage circuits using semiconductor technology.

A Typical Random Access Memory

The memory system we will deal with first is random-access memory. In a RAM, the information stored at all locations in the memory is equally accessible in approximately the same amount of time. In today's digital systems, especially personal computers, RAM usually means the memory system that holds the programs and related data which the digital system needs to execute its current operations. Thus, this RAM must have enough storage capacity to hold the program(s) and associated data for tasks the system is going to execute. Several years ago, it was sufficient to have 1 megabyte (MB) of semiconductor RAM (1 megabyte = 1,048,576 eight-bit bytes, however, for convenience, we usually just call it 1 million bytes). But today's operating systems and application programs need much more semiconductor RAM for efficient operation — 8, 16, 32, 64 MB and up.

(In our continuing discussion, RAM means the broad scope memory system, not just the definition of the term. This is an important point to understand because read-only memory (ROM,) which we will discuss later, is also a random-access memory.)

A RAM has both read and write capability. In fact, a better term for RAM is *read/write* memory because all semiconductor and disk memories have random access. Depending on the state of the control inputs, the RAM must either store input data or it must output previously stored information with equal speed to/from any storage location for use by the digital system. RAM is further classified as static or dynamic, but, for now, let's look at the overall system.

System Organization

The information, or data, stored in memory may consist of thousands, millions, or millions of millions of bits. A major concern is how to locate a particular bit or group of bits among all of the ones available. In *Figure 2-14*, we showed the concept of locating a particular word in a memory array. In *Figure 6-1*, a typical RAM system is shown in more detail. *Figure 6-1* applies to random-access memory systems in general; that is, both RAM and ROM.

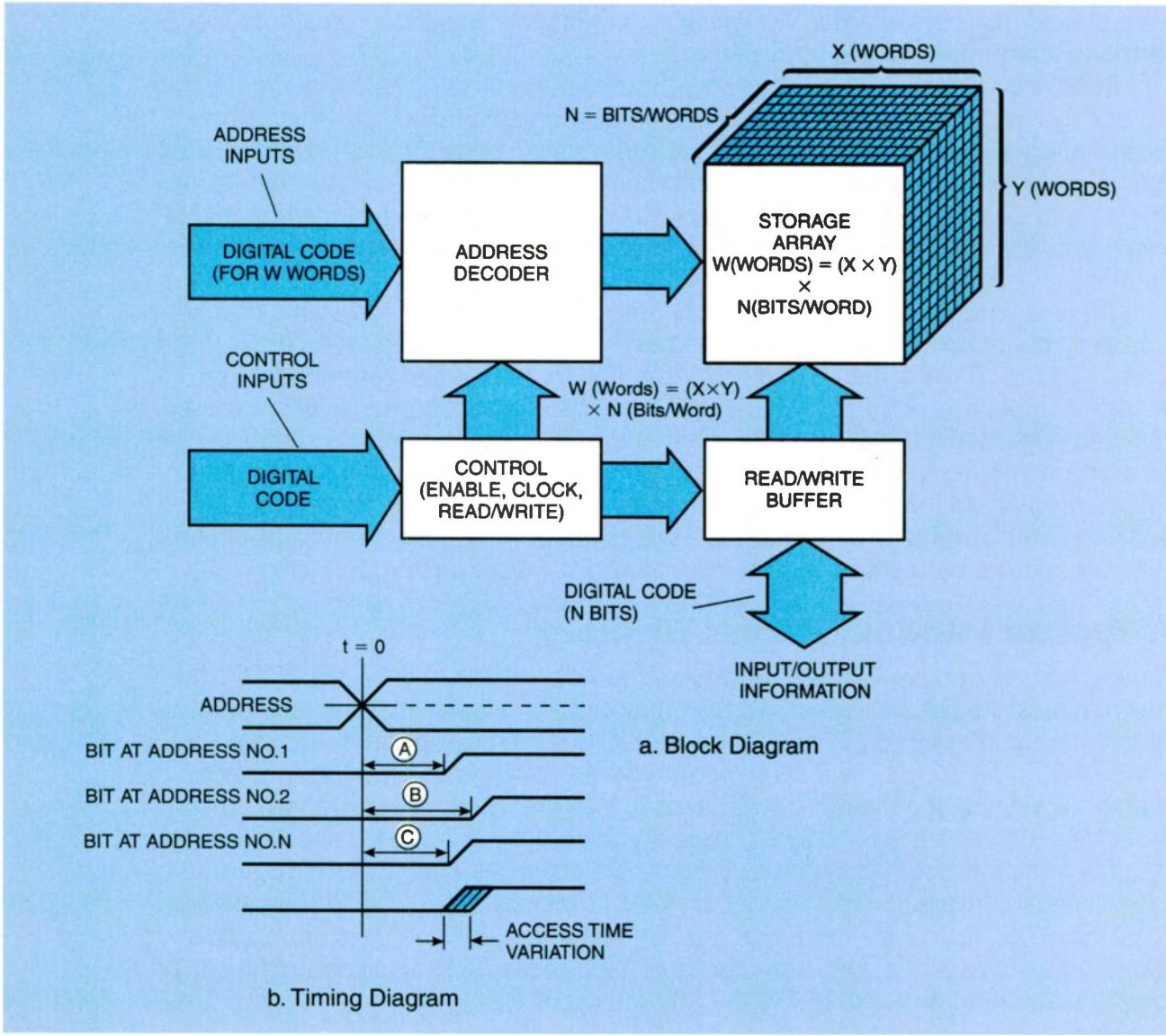


Figure 6-1. Typical RAM System.

When designing a memory system and its associated memory cells, the location of each bit is specified by a digital code called the *address*. RAM integrated circuits include within them, as shown in *Figure 6-1a*, address decoders which select the storage location indicated by the address. The number of address input lines (bits in the address code) is determined by the number of words stored in the IC. A memory storing 2^N words requires N address lines. Many modern digital system processing units (usually a microprocessor) operate on groups of eight bits (a byte) instead of individual bits, and the addressing may access a byte at a time, thus, a 16 MB memory requires 24 address lines. The RAM system shown in *Figure 6-1a* is organized with each word containing N bits. The total storage capacity is W words × N bits. Remember, the storage array must contain a memory cell for each bit. If N = 8 bits for a byte, the total storage capacity is W bytes. A 4 MB RAM contains four million bytes or 32 million bits (remember, we are rounding to the nearest million). When each memory word contains a byte, the method of organization is referred to as a byte-wide organized RAM.

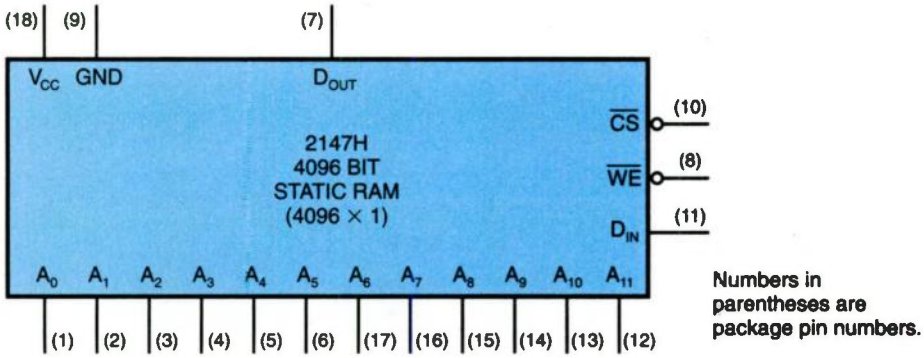
As shown in *Figure 6-1a*, RAM accepts both address and control inputs, and includes read/write buffers, which prepare the selected storage location for the particular memory operation (read or write) as indicated by the control inputs. Depending on the state of the address code and the control code, the RAM must either write (accept and store) input data or it must read (copy) previously stored information from the addressed location. As shown in *Figure 6-1b*, the variation of access times for reading data from the memory is quite small — confirming the fact that this memory system is called random-access.

The following two examples illustrate methods of storage element address organization:

Example 1. Number of Bits in an Address Code

How many address bits are required to address the storage locations in a 2147H static RAM?

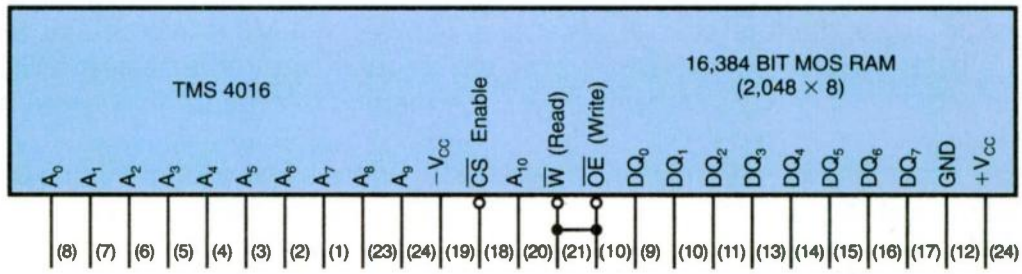
The logic diagram symbol for a 2147H MOS static RAM is as shown. This memory has a W × N organization of 4096 × 1, which means that there is storage capacity for 4096 one-bit words. Since $4096 = 2^{12}$, then 12 address input lines ($A_0 - A_{11}$) are required. (Note: 4096 is referred to as 4K. In computer terminology, the "K" multiplier is equal to 1024, not 1000 as for the "k" multiplier in electronic circuits.) Notice that there is only one data input and one data output line.



Example 2. Number of Bits in an Address Code

How many address bits are required to address the word storage in a TMS4016 RAM?

The logic diagram symbol for a TMS4016 MOS static RAM is as shown. This memory has a 2048 × 8 byte-wide organization, which means that there is storage capacity for 2048 eight-bit (byte) words. (This is referred to as 2K bytes or 2 KB.) Notice that there are eleven address input lines ($A_0 - A_{10}$) because $2^{11} = 2048$. Therefore, 11 bits are required in the address. Notice the TMS4016 has eight data lines that are used to input and output data.



RAM Family

Recall that we said there are two types of RAMs — static and dynamic. Two types of semiconductor technology can be used for the design and production of RAMs. *Figure 6-2* shows an overview of the RAM family. Both bipolar and MOS technology can be used as shown; however, bipolar, because of its higher power dissipation and higher speed, is only used in static RAMs. MOS, and specifically CMOS technology, is most widely used because of its low power dissipation and acceptable speed.

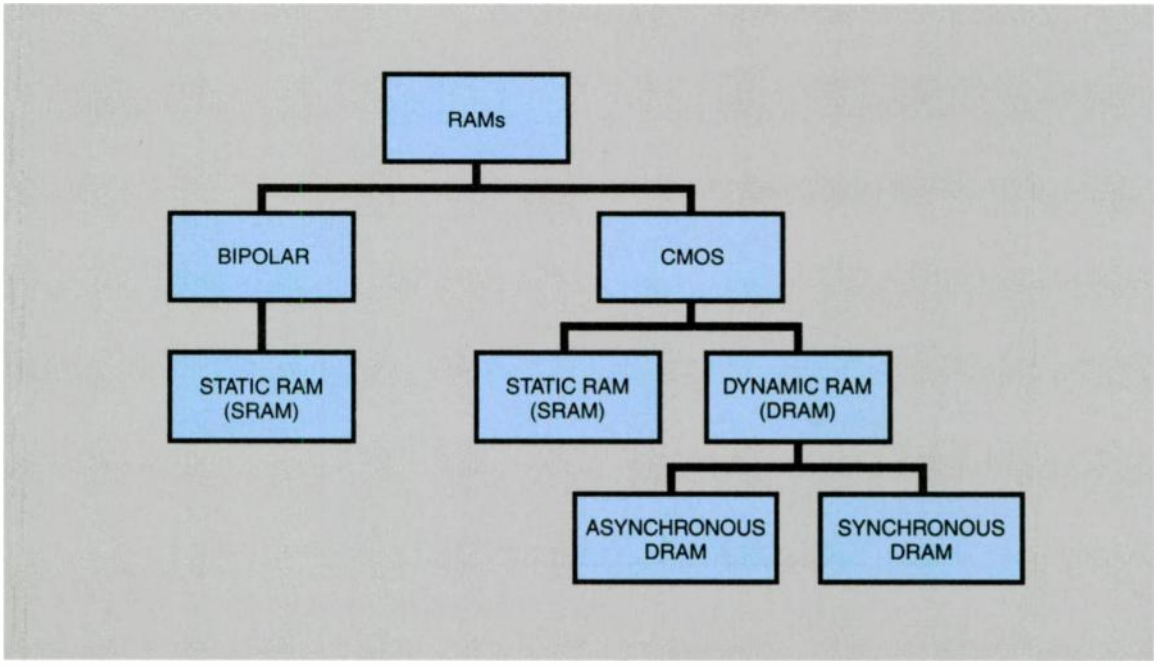


Figure 6-2. RAM family uses both bipolar and CMOS technology. Bipolar is used only for static RAM.

Bipolar Static RAM Cell

The storage cell in static RAM (SRAM), both bipolar (TTL) and MOS, is the flip-flop. *Figure 6-3a* shows an early circuit[1] used for a bipolar RAM cell. It is used to show the simplicity and familiar cross-coupled inverters used for flip-flops. The two multiple-emitter transistors and their associated load resistors form a flip-flop circuit capable of storing one bit. When the WORD LINE is not selected, its voltage is 0.3V. If Q_1 is latched ON, emitter E_1 is the conducting emitter and there is current to the WORD LINE. When this cell is selected by the address-decoding circuitry, the WORD LINE is raised to the logic 1 level (3.0V). If the memory cell is to be read, the READ/WRITE buffer of *Figure 6-1a* connects a sense amplifier to the BIT LINE, the ON latched current of Q_1 switches from E_1 to E_2 and the sense amplifier detects a 1. If Q_1 had been OFF, the sense amplifier would have detected no current on the line.

If a bit is to be written into the cell (the READ/WRITE buffers are in the WRITE state), WORD LINE is again set to logic 1. Then, the input data bit is applied to $\overline{\text{BIT LINE}}$, and its complement is placed on BIT LINE. For example, to store a logic 0, 0 is placed on $\overline{\text{BIT LINE}}$ and 1 is placed on BIT LINE. Q_2 conducts and Q_1 is cut off, thereby storing logic 0.

Anytime the cell is not selected by the address-decoding logic, WORD LINE is at the logic 0 level (0.3 v.) The E1 emitter connected to the WORD LINE becomes the active element of the transistors, and the BIT LINE/$\overline{\text{BIT LINE}}$ E_2 emitters do not function. The sense amplifier connected to BIT LINE, therefore, detects no data.

Bipolar RAM cells are very fast, and bipolar RAM memories typically have access times from 10 to 50 nanoseconds (ns). Their chief disadvantage is that they dissipate relatively large amounts of power, typically 0.5 milliwatt (mW) per bit. This is because one transistor in the flip-flop circuit is always saturated, conducting current through its load resistor.

[1] *Semiconductor Memory Design and Applications,* G. Luecke, J.P.Mize, W.N.Carr, p. 95, ©1973, Texas Instruments Incorporated, McGraw-Hill Book Company.

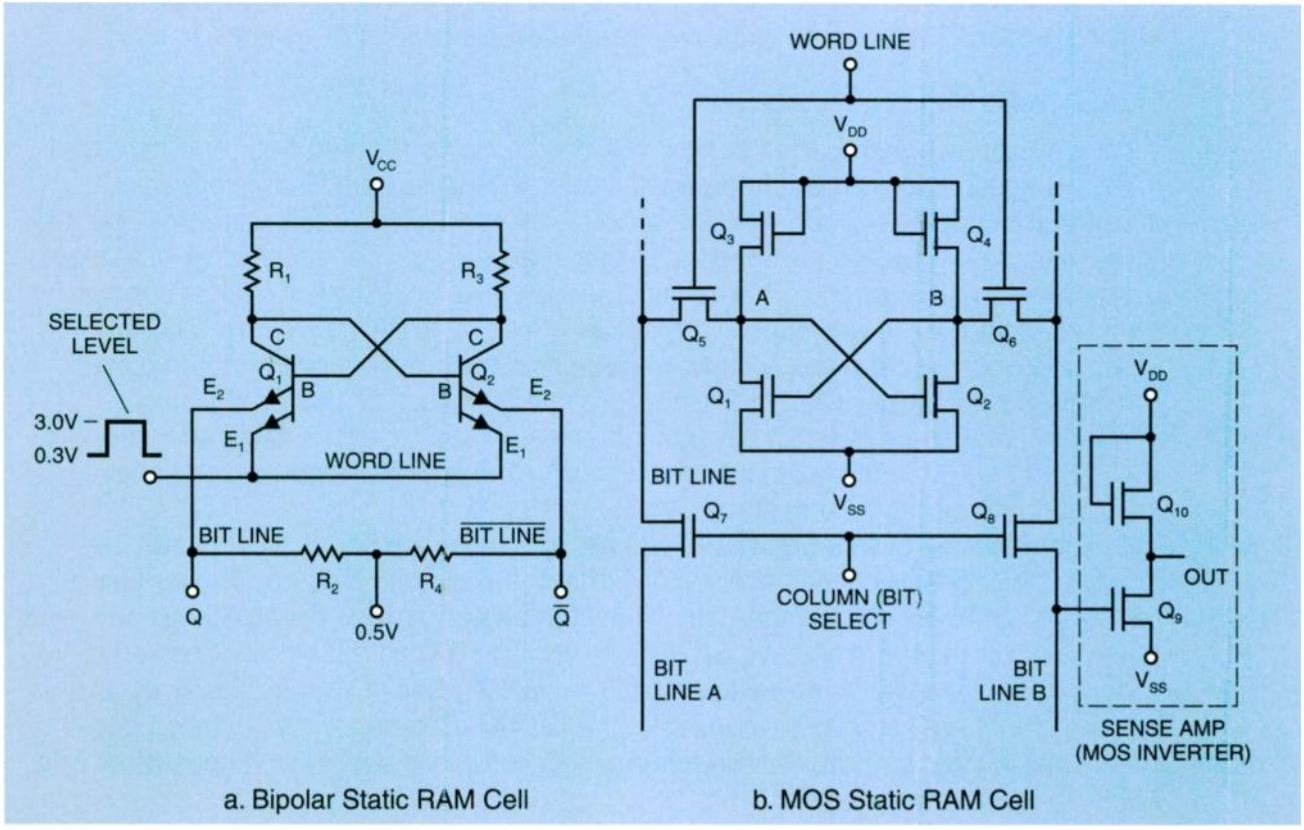


Figure 6-3. Static RAM cells.

MOS Static RAM Cell

An early MOS static RAM cell circuit[2] is shown in *Figure 6-3b*. We again use an early circuit for simplicity. Q_1 and Q_2 correspond to the cross-coupled inverter bipolar transistors in *Figure 6-3a*. MOS transistors Q_3 and Q_4 are used as the load resistors. MOS transistors Q_5, Q_6, Q_7, and Q_8 are activated by the decoding and serve to isolate the cell so it is not disturbed. Q_5 and Q_6 are controlled by the WORD decoder, and Q_7 and Q_8 are controlled by the COLUMN (BIT) decoder. The MOS transistors are P-channel enhancement transistors; therefore, V_{SS} is the most positive voltage and V_{DD} is the most negative. The transistors turn ON when the gate is a V_{DD} potential referenced to the source. Q_5 and Q_6 are enabled by applying V_{DD} to their gates, which the WORD line decoding does for either read or write of the cell. At the same time, V_{DD} is applied to the COLUMN (BIT) select to connect the bit lines.

If Q_1 is ON and Q_2 is OFF, point A will be close to V_{SS} and point B will be near V_{DD}. If the cell is read by activating Q_5 and Q_6 with the WORD line decoding, and Q_7 and Q_8 with the BIT decoding, then bit line B will be at the voltage of point B — near V_{DD} — and inverter transistor Q_9 will be ON to output a 1. The inverter sense amplifier would output a 0 if Q_2 had been ON.

To write into the cell and turn Q_2 ON, Q_5, Q_6, Q_7, and Q_8 are again activated and bit line A is forced to V_{DD} and bit line B is forced to V_{SS}. This places point A at V_{DD} and point B at V_{SS} to turn ON Q_2 and turn OFF Q_1. Removing the decoding isolates the cell and leaves it storing the written data.

The MOS cell has lower power dissipation than the bipolar cell. Typical power dissipation is about 0.15 mW per bit compared to 0.5 mW for bipolar. Bipolar transistors switch much faster than MOS transistors, so the access time for MOS static RAM is slower, typically in the range of 40 to 80 ns.

To conserve power, static RAMs have a standby mode when they are not selected. MOS static RAMs dissipate considerably less power in the standby mode than in the selected mode. The difference is not as great for bipolar static RAMs. Even though a static RAM uses MOS transistors, the output and input logic levels are at the standard TTL levels so that they interface easily with TTL systems.

Dynamic RAM Systems

As digital systems increased in complexity, there was continuing need for increased RAM storage capacity. That condition still exists. Because static RAM cells use considerable power and a significant amount of integrated circuit real estate per cell, there was a need to find a memory cell that used only a small amount of power and used only a small amount of IC real estate. The dynamic RAM (DRAM) cell and memory system were developed to meet this need. The DRAM system is called "dynamic" because the memory would lose its data if it were not restored (called *refreshing*) periodically. MOS transistors are used because of their low-power dissipation and small physical size. Capacitors are used to store charge to represent the data. In addition, they are compatible and can be manufactured using the integrated circuit processing steps.

There are many varieties of DRAM cells and more are being developed. The emphasis has been on small size. DRAM integrated circuit chips are already being produced commercially that contain 16 million bits of storage capacity, and new ones have been developed that have 64 million bits. The DRAM cell to be discussed is shown in *Figure 6-4*. It is an early circuit[3] that is used because of its simplicity and helps to show easily the basic concepts of a DRAM memory. It is a so-called *one-transistor cell,* and probably is the minimum in complexity because it consists of one MOS transistor and one capacitor.

[2] Ibid, p. 118
[3] Ibid, p. 123

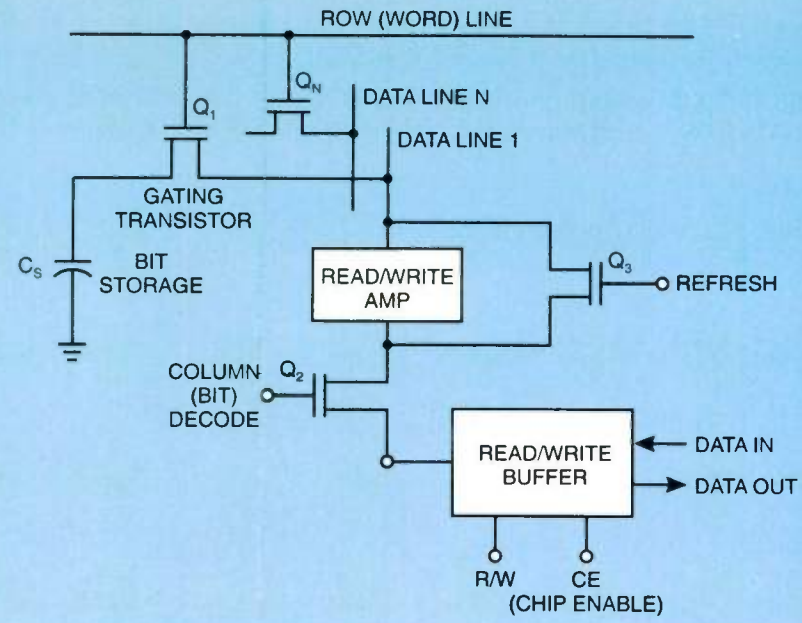


Figure 6-4. One-transistor DRAM cell showing row and column decode, sense, I/O and refresh functions.

One-Transistor DRAM Cell

In the cell of *Figure 6-4*, the bit of data is stored on capacitor C_S. The capacitor holds data after it is charged (written) from the data line through the MOS transistor Q_1. If the data on the capacitor is to be read, Q_1 is again activated to read the data onto the data line. Q_1 is activated by the ROW (word) line decoding, and Q_2, another MOS transistor, is activated by the COLUMN (bit) line decoding. The bit line, organized into a column, is common to the bit position of many of the words in the memory; therefore, Q_2 is not part of the one-transistor cell. When Q_1 is OFF, it isolates the capacitor C_S so its data is not disturbed.

As the memory cell sits idle, the charge leaks off capacitor C_S. In this early design, within a two millisecond time period, the charge would decrease so that the correct logic state could not be reliably determined. To recharge the capacitor, a read/write amplifier is inserted in the data line. It recreates a full logic level at its output from a below par logic level. Here's the refresh cycle. The ROW (word) decode activates Q_1 and reads the data into the read/write amplifier. The REFRESH signal activates Q_3 to close a feedback path to the data line. The full logic level is reestablished on the data line and on C_S.

On read, Q_2 is activated by the COLUMN (bit) decode and the read/write amplifier output is fed to the read/write buffer which passes it to the Data Out line. A R/W control signal tells the buffer that the memory is in the READ mode. CE (chip enable) has previously activated the particular DRAM integrated circuit containing the correct addressed memory word and its bits.

Self-Contained Refresh Circuitry

Refreshing the memory, because it must be done on demand due to the time requirement, requires separate circuitry consisting of a row address generator, a multiplexer to switch between the system address bus and the refresh address generator, and logic to isolate the memory from the data bus during the refresh operation. Early on, the refresh circuitry was not on the memory integrated circuit chip. As a result, the circuitry required additional design, additional space, and added

complexity to the system. In present designs, manufacturers put the circuitry right on the DRAM chip. *Figure 6-5* shows an example. Amplifier A is active on READ to output data from C_S. Amplifier B is active on WRITE to write data from the Data In line to the memory cell. Amplifier A and B are both active on REFRESH so that the data on the data line is reestablished at the proper level and C_S refreshed.

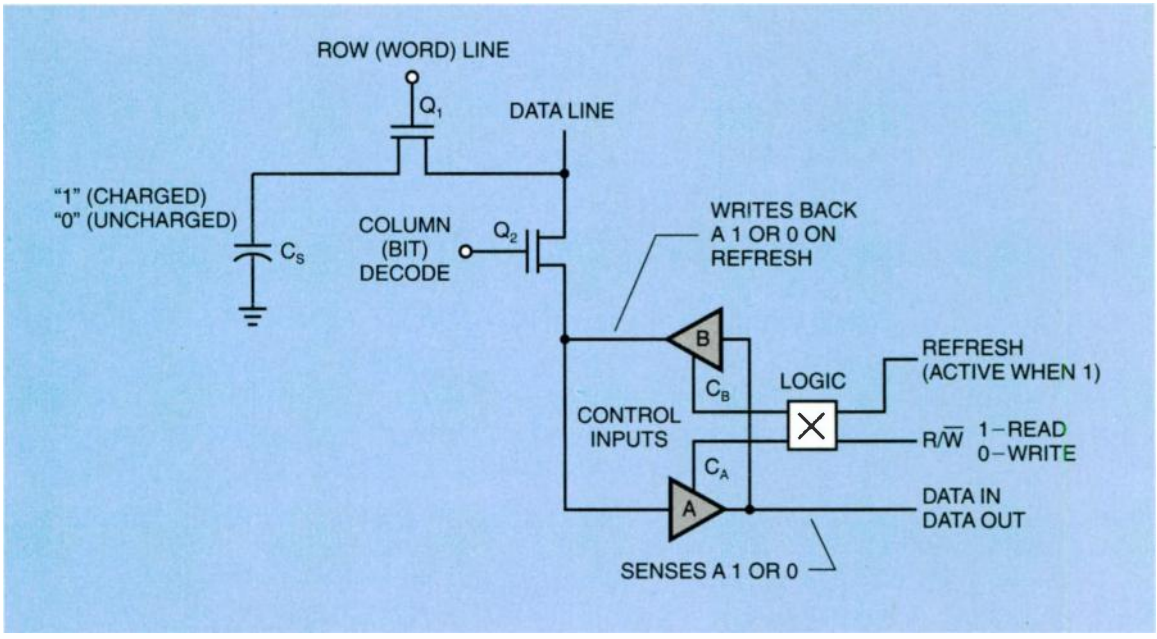


Figure 6-5. On-chip refresh circuitry restores 0 or 1 on C_S.

Timing Generator

Within a RAM or DRAM system, or any memory system, the application of the various digital codes occurs in a preset time sequence under the supervision of a controller. The controller needs a timing generator. An example of a timing generator is shown in *Figure 6-6*. It uses four clocked D-type FFs to generate four different timing signals called Phase 1 (ϕ_1), Phase 2 (ϕ_2), Phase 3 (ϕ_3) and Phase 4 (ϕ_4). These

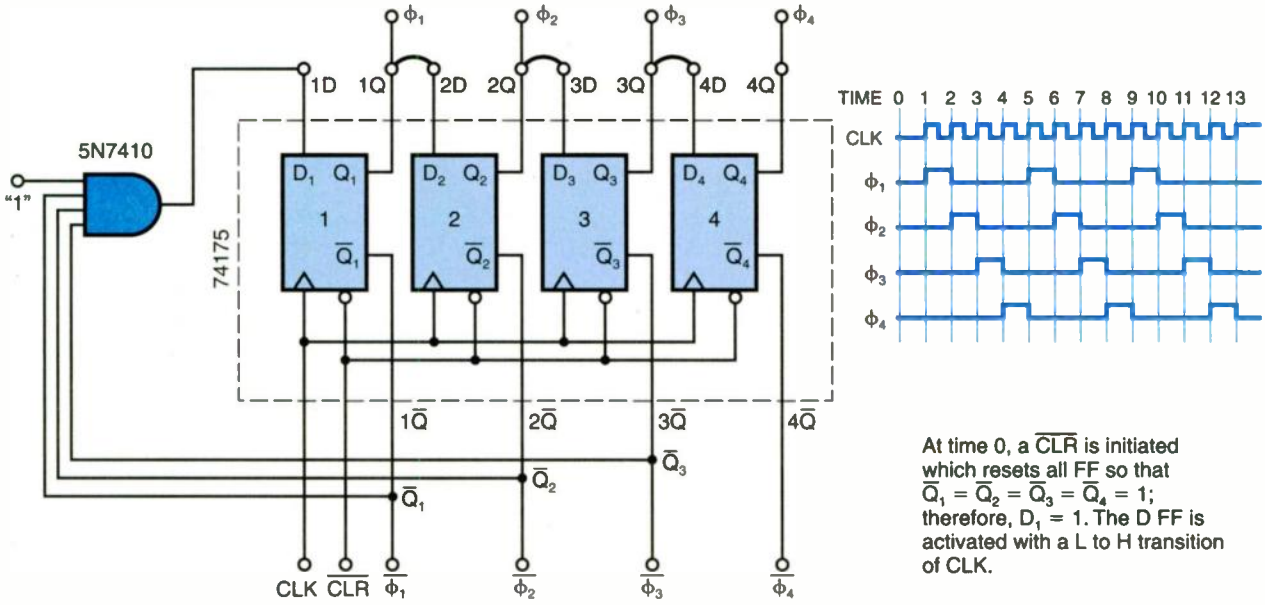


Figure 6-6. Timing generator generates control signals used to control memory functions in sequence.

four timing signals are based on a master clock so they can be used to time the operation of chip enable, address decoding, read/write and the other necessary control signals.

Example 3. Logic for Control of Refresh Circuitry

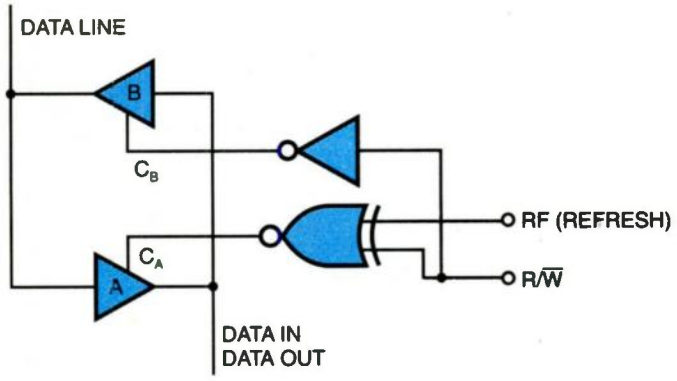
What kind of logic is needed for the logic block X in *Figure 6-5*?

The first step is to construct a truth table, then draw the logic. REFRESH (RF) is active on 1, READ (R) is active on 1, WRITE (W) is active on 0, amplifier A is ON when control signal C_A is a 1, and amplifier B is ON when control signal C_B is a 1.

	INPUTS		OUTPUTS	
Operation	$R/\overline{W}$	RF	C_B	C_A
Read	1	0	0	1
Write	0	0	1	0
Don't Care	1	1	0	0
Refresh	0	1	1	1

$C_A = R/\overline{W}\ \overline{RF} + \overline{R/\overline{W}}\ RF$ (XOR)

$C_B = \overline{R/\overline{W}}$ (INVERTER)



Example 4. Timing Generator Waveforms

Verify that the waveforms for the phases of the timing generator of *Figure 6-6* are correct. Whatever data appears on the D input of the D-type FF gets clocked in and appears on the Q output. If you follow the signals through the FFs and the logic gate, you can verify the waveforms. Start with all FFs cleared to zero.

Overall System Block Diagram

A typical 4-MB MOS DRAM is shown in *Figure 6-7*. Notice that it takes 20 address bits to address the storage array since the DRAM is organized as 1024K × 4. (This means there are 1024K words and each word has 4 bits.) Of the 20 address bits, 10 are decoded to select one of the 1024 rows, 8 are decoded to select one of the 256 columns, and 2 are decoded to select the desired 4-bit word from the 16 bits in each column. The final word selection is accomplished in the I/O buffers because all 16 bits from a column are read at once into the buffers. The 2 column bits (which provide four code combinations) are used to select the 4 bits desired.

To save package pins, only 10 address lines are available externally, thus, a two-step procedure is required to get all 20 address bits in. First, the 10 row address bits are applied to the 10 address pins, the control signal $\overline{RAS}$ is activated, and the row address bits are latched into the row address buffers. Next, the 10 column address bits are applied to the 10 address pins, the control signal $\overline{CAS}$ is activated, and the column address bits are latched into the column address buffers.

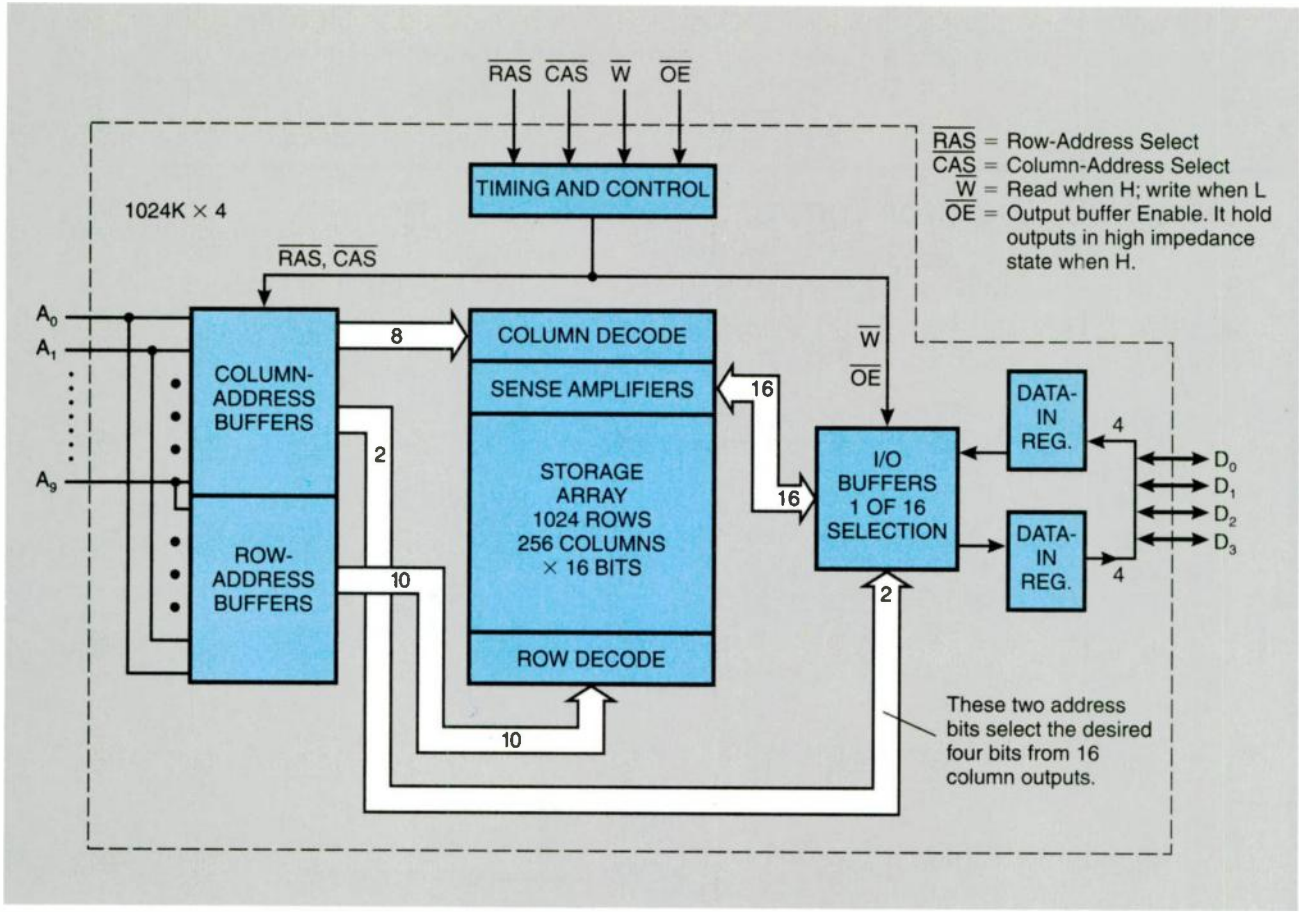


Figure 6-7. A 4,194,304-bit (4 MB) CMOS DRAM organized as 1024K words by 4 bits.

The timing is provided by the timing and control circuitry. The $\overline{W}$ is a read/write control signal. When it is a high logic level, the memory is read and the four bits are output on the data lines. When $\overline{W}$ is a low logic level, data (4 bits) is placed on the data lines and written (stored) into the array at the selected address. $\overline{OE}$ is an output enable that keeps the output line from memory in a high-impedance state (remember the 3-state output on a logic gate that we discussed) so the output line will not load the data bus until data is to be placed on the bus.

Refreshing

CMOS technology and DRAM design has progressed significantly so that the time between refresh operations has been extended to 16 milliseconds for the DRAM in *Figure 6-7*. There is refresh circuitry designed into the chip similar in function to that shown in *Figure 6-5*. At least every 16 milliseconds, every row address must be sequenced so that the data stored is refreshed. The refreshing is done in two ways. A block diagram of the external system interconnection is shown in *Figure 6-8*. The normal row addressing is replaced with row addresses generated in sequence by a refresh counter. Read/write and enable control signals are applied from the control bus. Only row addresses are applied and the counter steps through each row address to refresh the DRAM. This is called *burst refreshing*.

A method called *cycle-stealing refresh* is the second way. At a selected time, a cycle is used to refresh a row address. This continues until all row addresses are selected. The process must be completed in the 16-millisecond time period. In either case, the normal operation of the digital system using the DRAM must be suspended until the DRAM is refreshed.

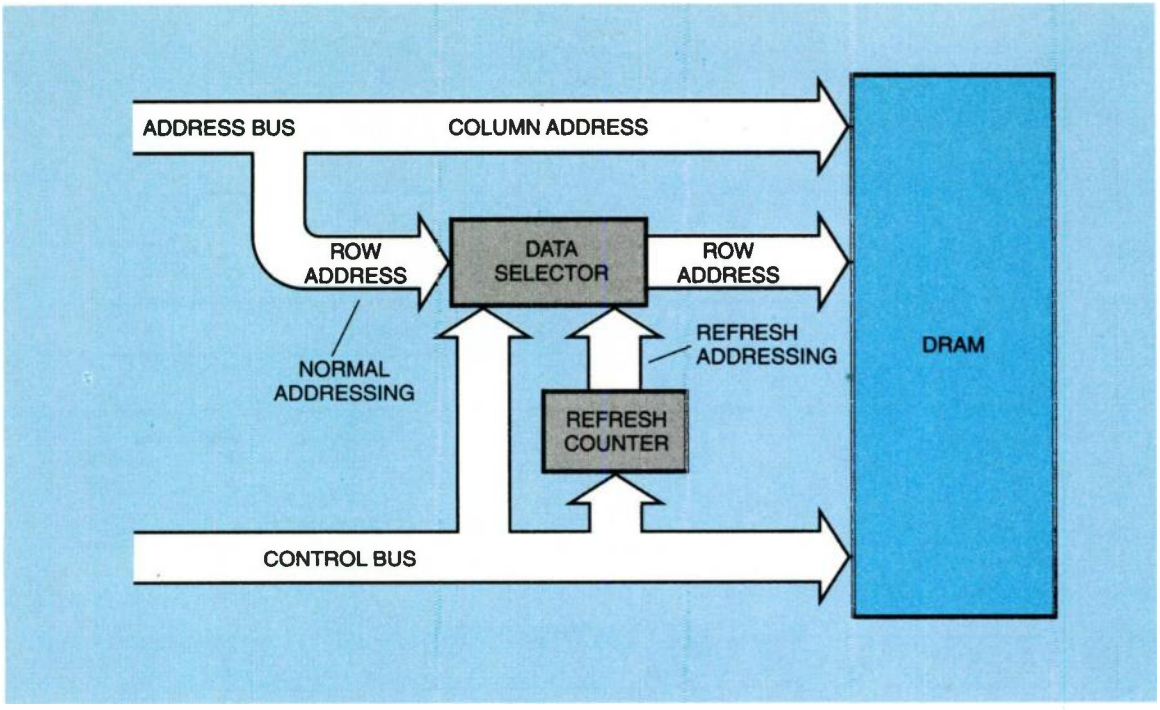


Figure 6-8. The DRAM is refreshed by sequencing through the row addresses which are selected when the refresh signal on the control bus activates the data selector.

Power-Up and Initialization

Because DRAM systems contain large amounts of capacitance and need to have time to stabilize before beginning reliable operation, manufacturers require a certain pause time, a certain number of initialization cycles, and a complete burst refresh before beginning operation of the DRAM.

Timing Diagram

An example timing diagram is shown in *Figure 6-9* to provide a sense of the type of timing signals that control the operation of the DRAM. The specified times have been stripped to simplify the understanding, but be aware that the time from when $\overline{RAS}$ latches in row addresses to when valid data appears at the output, t_{RAC} shown in *Figure 6-9*, is between 60 and 80 ns. That's 60 to 80 $\times$ 10^{-9} second. That's 0.000000080 second maximum! That seems fast, but it's slow compared to the 10 ns or less cycle time of microprocessors currently used in personal computers.

Asynchronous Versus Synchronous RAM

Recall that we saw in *Figure 6-2* that there were two members of the DRAM family — asynchronous and synchronous. Several companies are currently manufacturing and supplying DRAMs. All of these are what is called asynchronous because they do not run directly off of the system clock; that is, the memory integrated circuit is not interfaced with the system clock except through the control signals that are based on the clock. As a result, systems using asynchronous DRAMs must incorporate *wait states* so the system waits until the DRAM operation is completed. Command timing is based upon DRAM speed of operation rather than on system speed.

A major manufacturer of DRAMs is currently producing synchronous DRAMs (SDRAMs). They are called synchronous because they run directly off of the system clock. SDRAMs allow for better interaction between the RAM memory and the system because all commands are referenced to the system clock directly so wait states are unnecessary. SDRAMs have the same basic cell design, but vary in memory

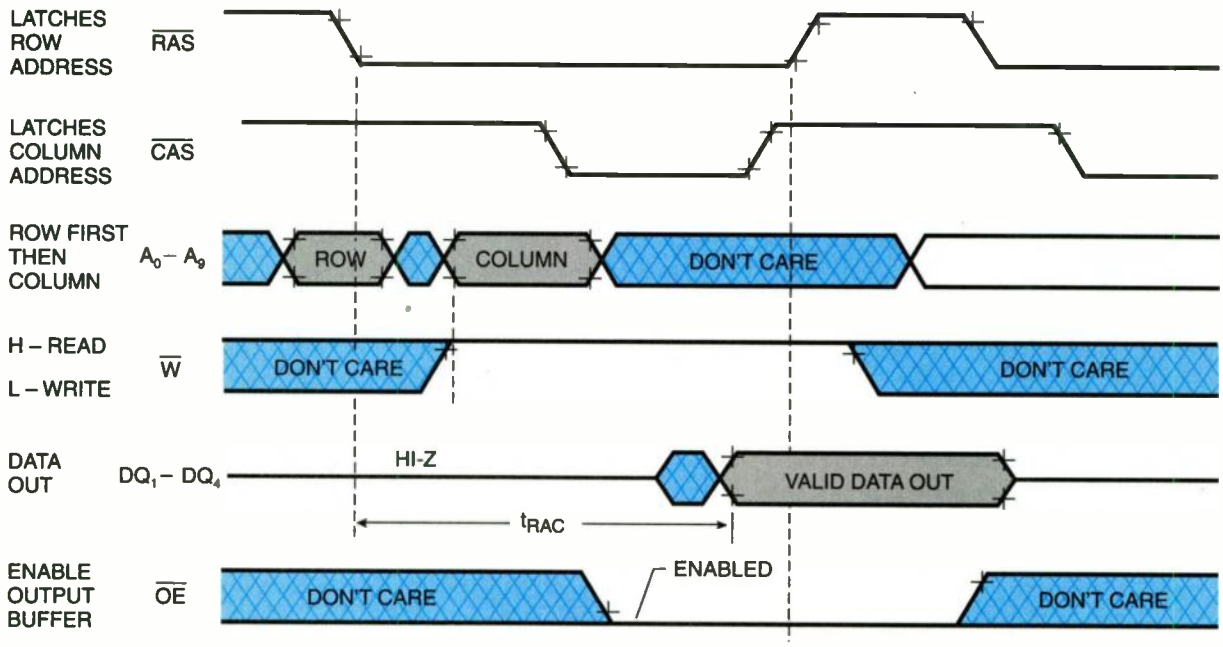


Figure 6-9. DRAM read cycle latches row addresses first, then column addresses. READ control signal provides valid data after address decoding is stabilized.

organization and operation. SDRAMs allow for greater data throughput. For example, to transfer 32 bytes using DRAM takes 1300 ns, while using SDRAM takes 370 ns. This performance has caused personal computer manufacturers to use SDRAMs in new designs. Another type of DRAM, called Enhanced Data Out (EDO) RAM has recently been introduced. It also gives performance gains in computer systems.

This concludes the discussion of RAMs. Let's now turn our attention to ROMs.

Read-Only Memory (ROM)

ROMs are solid-state memories used to store data on a permanent or semipermanent basis. They are capable of random access and are nonvolatile, which means that they do not lose their memory contents when the power is removed. Normal RAM is volatile and loses its memory contents if power is removed. As their name implies, ROMs are generally used for read-only operations and are usually not written to after they are initially programmed. Examples of ROM applications are lookup tables for such things as math functions, constants for performing conversions and special character codes; operating system programs for "booting up" personal computers, and tables of instructions to perform fixed-program routines.

Basic Concept

To understand the basic concept, look at *Figure 6-10*. The memory structure is the same as RAM. It has a row decoder, a column decoder, sense amplifiers, a read/write I/O buffer and control circuitry. It is a random-access memory, and either bipolar or MOS technology can be used. The same characteristics exist for bipolar and MOS as before — fastest speed and highest power dissipation for bipolar; slower speed and lower power dissipation for MOS. MOS and bipolar technology are shown in *Figure 6-10* to demonstrate the concept. A particular cell in the storage matrix is located by

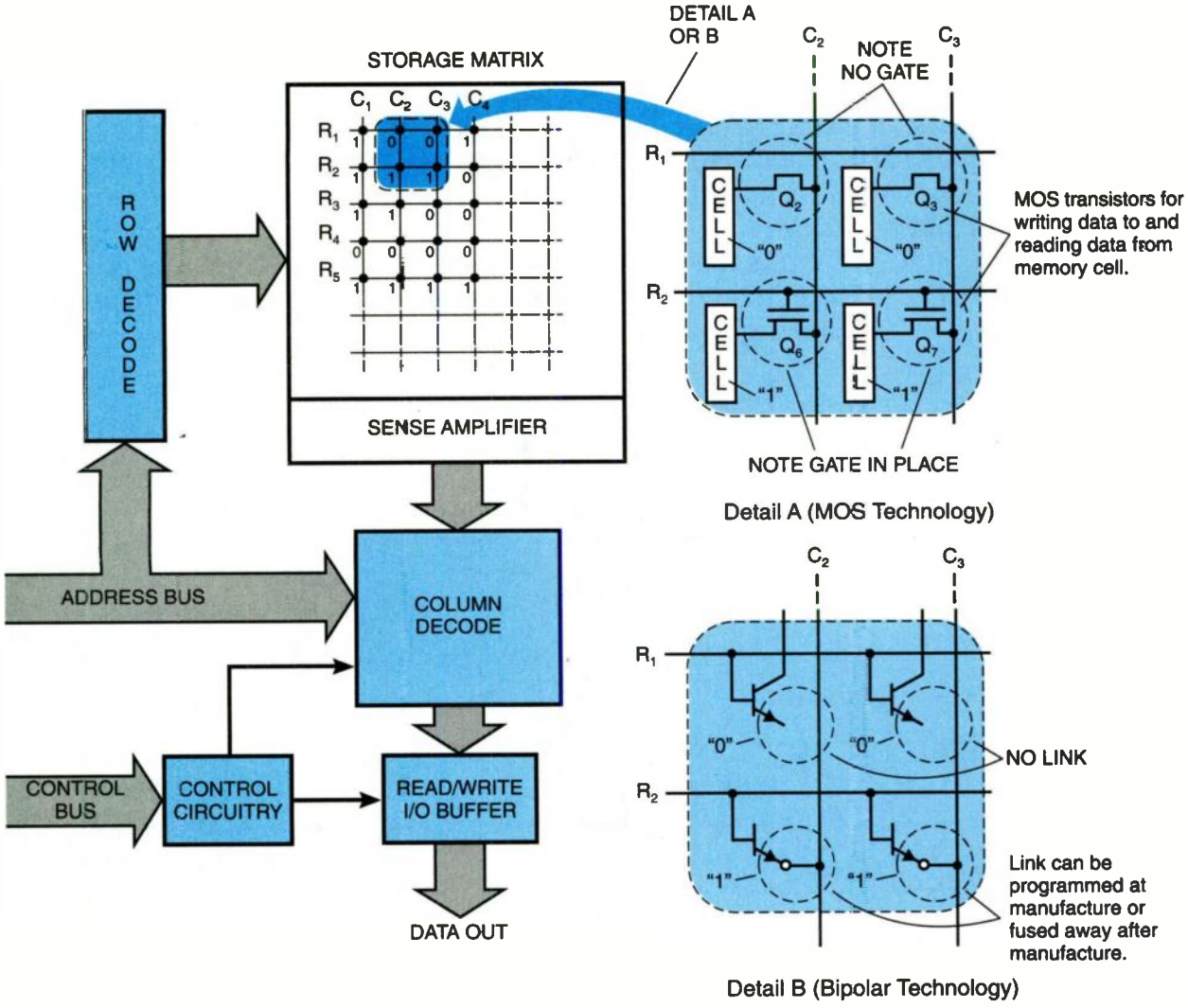


Figure 6-10. A ROM memory system with its fixed-program storage.

addressing a particular column and a particular row. Storage cells for digital information are located at each intersection. The difference for ROM is that the digital information is fixed after the initial programming. It is not meant to be changed for a relatively long period of time.

Note in *Figure 6-10* that there is a specific pattern of 1s and 0s stored in the matrix. In its least expensive and most permanent form, which are ROMs designed for high-volume systems, the pattern of information is placed in the storage matrix during the manufacture of the ROM. For example, look at Detail A of *Figure 6-10*. Note that for the storage cells that have a 1 stored, the MOS transistor used to write data to the cell has a gate connection, and for the storage cells that have a 0 stored, the MOS transistor does not have its gate connected. The gate connections are *left out* during manufacture on cells where a 0 is to be stored, and *left in* where a 1 is to be stored. On power-up and initialization, 1s are stored and held in all locations where the gates are connected, and 0s are stored and held in all locations where the gate connection is missing. The cell in this case is a FF and not a capacitor. This is the basic concept of a ROM. For bipolar technology, the programming of the storage cell occurred because a transistor emitter connection was made or not made during manufacture as shown in Detail B of *Figure 6-10*. Programming the ROM during manufacturing is called *mask programming* because a mask used during manufacture is modified to do the programming.

Programmable ROMs

It is relatively expensive to have a new mask made each time a new programmed ROM is required. Only if large quantities of the ROMs are needed does it become cost effective; therefore, a need quickly developed for a ROM that could be programmed after manufacture. Such ROMs are very useful for small-quantity production, system changes to provide new features, research projects, or a quick solution proving an idea.

Figure 6-11 shows the ROM family of products and the fan-out of programmable ROMs. Mask programming has already been discussed. The other types are described in the following paragraphs.

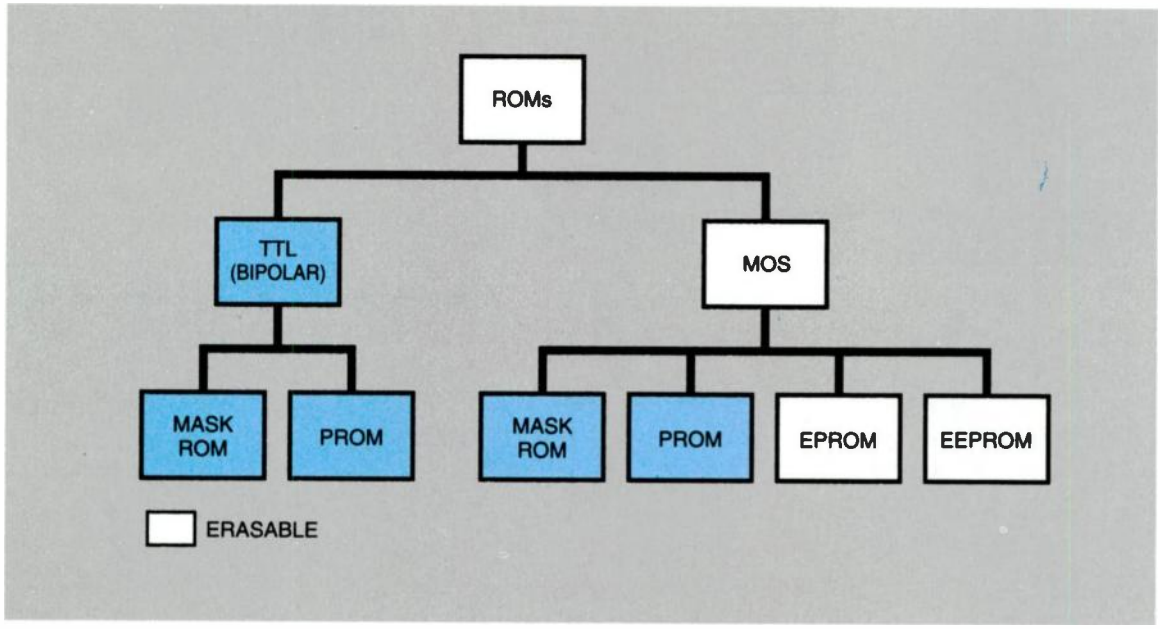


Figure 6-11. Semiconductor ROM family.

PROM:

Describes a ROM memory that can be programmed after manufacture. It originally described ROM memory storage arrays whose programming could be changed by "blowing" fusible links inside the ROM. For example, look at *Figure 6-10*, Detail B. The emitter link connection shown was in place when the ROM was manufactured resulting in 1s in all storage locations. After manufacture, the emitter links where 0s occur in the storage matrix were blown open with a high-current surge. Such PROMs are called "fusible-link" PROMs.

EPROM:

Describes an MOS (or CMOS) PROM that can be programmed by the user and then erased by the user and programmed again. These PROMs are packaged with a quartz window over the chip. The chip is exposed to a strong ultraviolet light beamed through the window to erase it. The erasure time is from 20 to 60 minutes.

EEPROM:

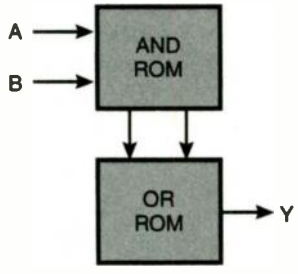
Describes an MOS (or CMOS) PROM that can be programmed by the user, erased by using electrical means, and reprogrammed again; thus the term EEPROM for Electrically Erasable Programmable ROM. An MOS transistor structure called a "floating gate" is used in EEPROM memory cells.

ROMs have filled a real need in the design cycle of digital systems. EEPROMs are first used in the experimental stage, PROMs are used in small-quantity production systems, and mask ROMs when the production reaches high quantity.

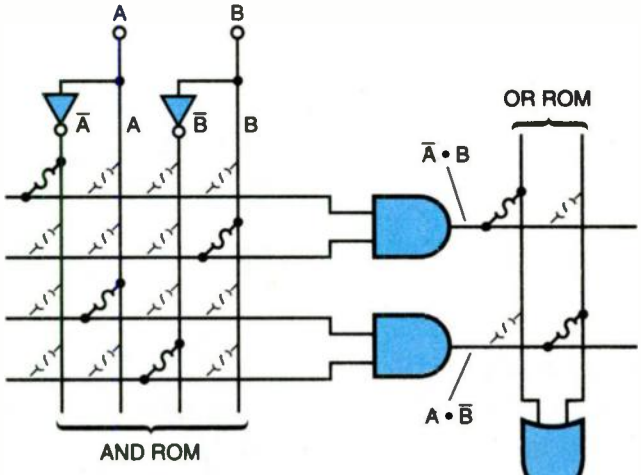
Programmable Logic Arrays (PLA)

There is one other product that really isn't a memory, but has come out of the development of the ROM. It is called a PLA, for Programmable Logic Array, because it is used for logic circuits in place of interconnecting individual IC gates. Overall, the device may be called a PLD (programmable logic device) or such names as PAL (programmable array logic), PLS (programmable logic sequencer), or GAL (generic array logic.)

Figure 6-12 shows the basic concept and shows why the devices are related to ROMs. A PLA can be considered as a combination of an AND ROM connected to an OR ROM as shown in *Figure 6-12a*. The detailed interconnections are shown in *Figure 6-12b*. The interconnections started with all the dotted line fusible links in place just like any PROM after initial manufacture. All links are "blown" except the solid line ones. The output of the top AND gate will be a 1 when $\overline{A}$ AND B are 1s. The output of the bottom AND gate will be 1 when A AND $\overline{B}$ are 1s. Both outputs of the AND gates are each coupled to an input of an OR gate, and the output of the OR gate will be a 1 when $\overline{A}B$ OR $A\overline{B}$ are 1s. The truth table for the logic is shown. Note how the 1s in the truth table correspond to where the links remain in the AND ROM when the OR gate output is to be a 1.



a. Block Diagram



TRUTH TABLE

INPUTS				OUTPUT
A	$\overline{A}$	B	$\overline{B}$	Y
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	0	0

b. Detail Interconnnection

Figure 6-12. A programmable logic array derived from ROM technology.

Example 5. Programming a PLA

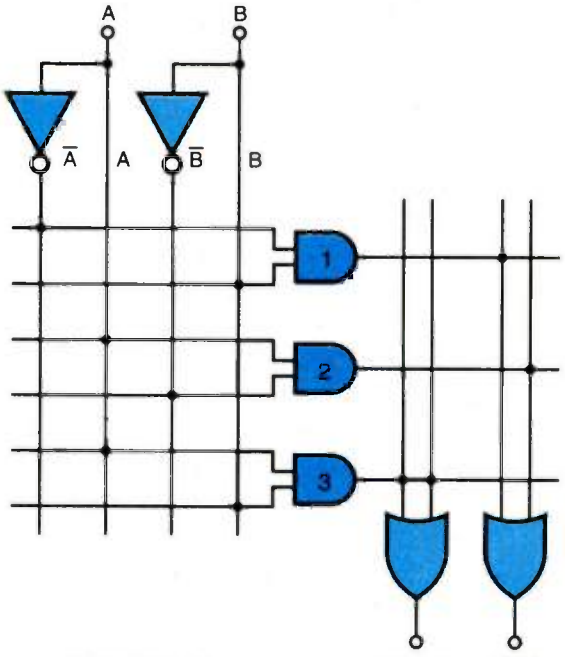
What links in the PLA of *Figure 6-12b* would be connected if $Y = AB + \overline{A}\overline{B}$?

The inputs to the top AND gate would be to line A and line B. The inputs to the bottom AND gate would be to line $\overline{A}$ and line $\overline{B}$.

Example 6. A PLA Half Adder

Is the logic in the PLA shown a half adder?

Satisfy yourself by constructing a truth table. AND gate #1 is a 1 when $\bar{A}B = 1$. AND gate #2 is a 1 when $A\bar{B} = 1$. AND gate #3 is a 1 when $AB = 1$. AND gates #1 and #2 are inputs to the SUM OR gate, and AND gate #3 is the input to the CARRY OR gate. The PLA performs this function: $SUM = \bar{A}B + A\bar{B}$ and $C = AB$.



TRUTH TABLE

INPUTS		OUTPUTS	
A	B	S	C
0	1	1	0
1	0	1	0
1	1	0	1
0	0	0	0

$SUM = \bar{A}B + A\bar{B}$

$CARRY = AB$

} CONDITION NOT SHOWN IN PLA BUT STILL VALID

Summary

In this chapter we have covered RAM and ROM memory. RAM is memory used by digital systems for storing present programs and data that the system is executing. ROM is used for storing information that remains the same as it is used repeatedly for fixed programs to initialize a digital system, and for conversions. Be aware that integrated circuit technology has advanced to the point where a memory cell MOS transistor is 100 times smaller than the thickness of a human hair — and it will continue to be reduced in size in the future. In the next chapter, we will see the techniques used to make integrated circuits.

Quiz for Chapter 6

1. A _____ has both read/write and random access capabilities.
   a. EPROM
   b. ROM
   c. RAM
   d. PLA
2. The location of each bit in memory is specified by a digital code called a/an _____ .
   a. program
   b. address
   c. timing slot
   d. data bit
3. The total number of bits stored in a 4K x 8 RAM is:
   a. 8
   b. 2000
   c. 4096
   d. 32,768
4. A memory device with ten address inputs has how many word storage locations?
   a. 10
   b. 1024
   c. 4096
   d. 10K
5. A statement which is not true about dynamic RAM is that:
   a. cells are smaller than static RAM cells.
   b. it dissipates less power than static RAM.
   c. it must be refreshed during operation.
   d. cells are constructed with MOS flip-flops.
6. To say that a memory is volatile means that:
   a. storage cells are erased by reading.
   b. storage cells must be refreshed.
   c. it dissipates relatively large amounts of power.
   d. removing power erases the memory.
7. From a user's point of view, the most inflexible read-only memory is:
   a. EPROM
   b. EAROM
   c. PROM
   d. ROM
8. The storage cell in static RAM is the
   a. capacitor
   b. flip-flop
   c. bubble
   d. transistor
9. A characteristic of dynamic RAM is that it
   a. must be refreshed.
   b. dissipates relatively large amounts of power.
   c. takes up more room on the silicon chip.
   d. is very fast compared to static RAM.
10. A DRAM that runs directly off the system clock is _____ DRAM.
    a. programmable
    b. static
    c. synchronous
    d. asynchronous
11. A solid-state memory used to store data on a permanent or semipermanent basis is a
    a. flip-flop
    b. RAM
    c. DRAM
    d. ROM
12. The type of memory that is erased by a strong ultraviolet light is the
    a. ERAM
    b. fusible link PROM
    c. EPROM
    d. EEPROM
13. For very large quantities of production _____ are used for memories.
    a. mask ROM
    b. EPROM
    c. EEPROM
    d. fusible link PROM
14. A device made from the combination of an AND ROM and an OR ROM is the
    a. RAM
    b. PROM
    c. PRAM
    d. PLA
15. To program a PAL or a GAL you must
    a. blow the links
    b. down-load the data
    c. set external toggle switches
    d. stop the MPU

Answers:
1c, 2b, 3d, 4b, 5d, 6d, 7d, 8b, 9a, 10c, 11d, 12c, 13a, 14d, 15a

Questions and Problems for Chapter 6

1. How many address lines are required to address 16K storage locations of 1-bit words?
2. How many address lines are required to address 16K storage locations of 8-bit words?
3. Compare bipolar and CMOS technologies for use in static RAM.
4. What are some characteristics of DRAM?
5. In addition to the storage cell, what else is usually contained in a one-transistor DRAM cell?
6. How many address lines and how many data lines does the CMOS RAM in *Figure 6-7* have? How is it possible that 1,024 kilowords can be addressed?
7. What are the two methods of CMOS DRAM refreshing?
8. What is the difference between asynchronous and synchronous RAM?
9. What are some applications of ROM?
10. How is a ROM usually programmed?
11. How is a EPROM usually erased?
12. Describe the internal construction of a PLD.

CHAPTER 7

A Tour Through ICs

Introduction

Some of the biggest changes in our world are taking place on a scale that is remarkably small. The information age is literally driven by the semiconductor revolution. This invasion of electronics in our lives is possible because semiconductor manufacturers continue to make devices increasingly more innovative, powerful, smaller, and cheaper. The modern electronics era bloomed when thousands of transistors and other electrical components were integrated on a small chip of silicon crystalline material called an *integrated circuit,* or IC for short. Today, ICs are the heart of the microelectronics communication and computer industries.

Inside a computer are rows of integrated circuits — each capable of executing millions of operations a second and some storing up to millions and millions of bits of information. Bulky, power-consuming vacuum tubes were used first in electronic digital systems. They were replaced by discrete transistors, and now ICs have taken over. ICs evolved from transistor technology and are fabricated in the same semiconductor factories. Texas Instruments, Motorola, National Semiconductor, and Intel are prominent American manufacturers. There are as many procedures for the fabrication of integrated circuits as there are companies that produce them; however, the processes are similar.

Categories of ICs

Integrated circuits are categorized as *monolithic* or *hybrid*. The term monolithic is from a Greek word meaning *single stone*. This is truly an apt description because all of the components on the IC are formed on or within a single chip of silicon. The monolithic process is best suited for most digital circuits, including the central processing unit (CPU) of a computer. It is the most economical for mass production. Because of their size and structure, monolithic ICs are limited in how much power they can handle.

Hybrid IC technology is appropriate for small quantity runs and special circuit requirements, such as microwave and high-power circuits, whose requirements cannot be met by monolithic ICs. There are two general types of hybrid ICs: *thin-film* and *thick-film*. In both types, some components and connecting leads are deposited on an insulating substrate. If evaporation is used to deposit the components and interconnection pattern, the result is called thin-film. If silk screening is used, then the result is called a thick-film hybrid IC. Active devices — transistors and integrated circuits — and additional passive components are attached to the hybrid circuit by tiny wires (bonding) or by a solder reflow or surface mount process.

In this chapter, we will discuss the production of monolithic integrated circuits. The production of all silicon ICs begins with a slice (some call it a wafer) of single-crystal silicon. The Czochraiski method is currently used for producing more than 90% of all single-crystal semiconductor silicon, so let's begin there.

Czochraiski Silicon Ingot Growth Process

It all begins with the growth of pure silicon crystals. Silicon is a common element found in sand. It composes 28% of the earth's crust and is second only to oxygen in abundance. The silicon from the sand is refined, purified, and heated to a molten state. A small solid piece of single crystal called a *seed* is gently lowered into a vat of molten silicon. The principle process is shown in *Figure 7-1*. As the seed is pulled from the molten silicon, the hot liquid silicon in contact with the seed begins to cool and solidify as it is gently pulled from the molten region. A new crystal forms as a symmetrical extension of the original seed. After pulling 20 hours of growth, a single-crystal ingot results. In early processes, the ingot was only one inch in diameter, but current processes produce diameters up to eight inches and lengths of one to two feet. These improvements in processes are very important because many more individual circuits can be produced on a larger diameter slice and many more slices can be obtained from a longer ingot.

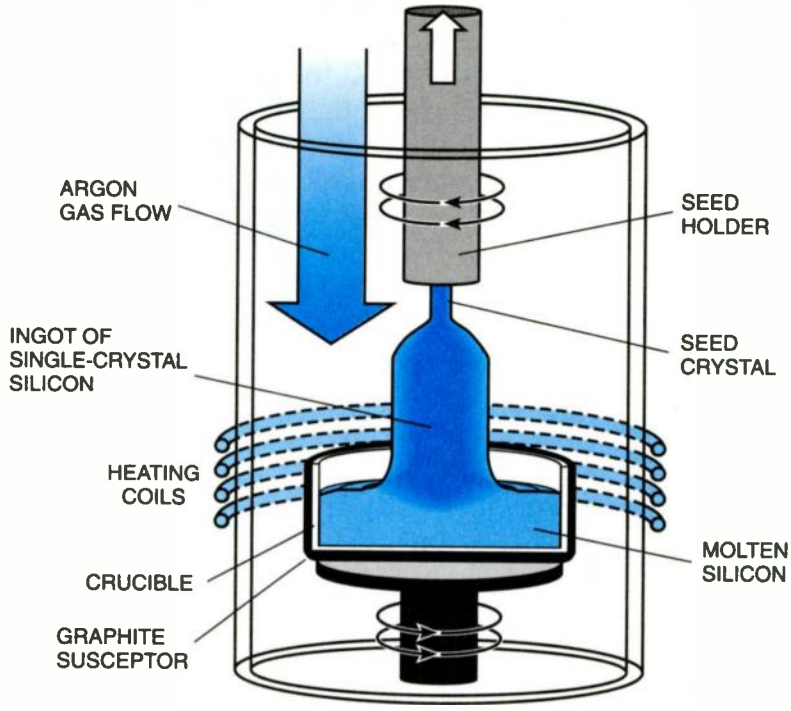


Figure 7-1. A crystal called a seed is gently lowered into a vat of molten silicon. As it is pulled from the molten silicon, a single-crystal ingot results.

Sawed Slices

As shown in *Figure 7-2*, the ingot is sawed into slices that are 10 to 12 mils thick (1 mil = 10^{-3} inch). As the ingot is pulled, N-type dopants, such as arsenic, or P-type dopants, such as boron, are added to make the single-crystal ingot an N-type or a P-type. Ingots and their wafers have primary flats to indicate the type and orientation of the single-crystal ingot.

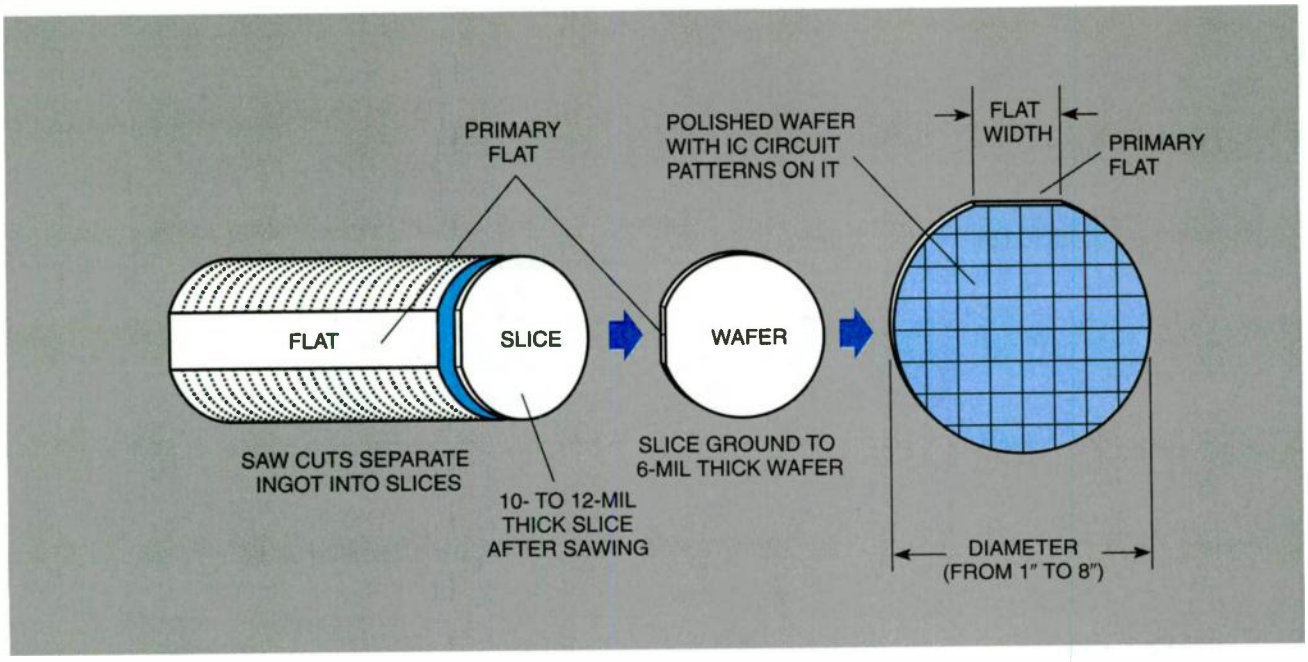


Figure 7-2. Slices are sawed from the ingot, ground down in thickness, and one side polished to a mirror finish. The flat on the wafer indicates the type and crystal orientation of the material.

The sawed slices shown in *Figure 7-2* are then polished to a mirror-like finish. After polishing, the slice is called a *wafer* and is about 6 mils thick. The final polish is done on only one side of the wafer. The characteristic mirror-like luster is free of contamination and scratches. The wafers are inspected and measured for resistivity, which is a function of dopant concentration. They are packaged and sent to fabrication areas where they will be made into ICs.

A Digital Circuit as an IC

Digital electronic circuits of the type we have been discussing in this book are designed to be manufactured as ICs using transistors of either bipolar (BJT) or MOS field-effect (FET) types. The TTL family is the most widely used IC of the bipolar type and CMOS of the MOSFET type. We explained in Chapter 3 how MOS transistors work, including a complementary metal oxide semiconductor (CMOS) inverter. We will use the CMOS inverter circuit to show how digital circuits are manufactured in IC form.

Simple CMOS Circuit

The CMOS inverter in *Figure 7-3a* is the circuit. Let's review how the circuit works. The P-channel MOSFET Q_1 acts as a drain resistor for the N-channel MOSFET Q_2, and the N-channel MOSFET Q_2 acts as a drain resistor for the P-channel MOSFET Q_2. A positive V_{IN} voltage near V_{DD} on the gates of Q_1 and Q_2 turns ON the N-channel transistor Q_2, pulling V_{OUT} to ground (V_{SS}). Q_2 sinks current from the output to ground. The positive V_{IN} voltage turns OFF the P-channel transistor Q_1 so there is no current through it. When V_{IN} is zero volts (near V_{SS}), Q_1 is ON and the N-channel transistor Q_2 is OFF. Now the current to the output is supplied through Q_1 from V_{DD}, and V_{OUT} is close to V_{DD}. A digital 1 is represented by a high-level V_{OUT} close to V_{DD}, and a digital 0 is represented by a low-level V_{OUT} close to ground (V_{SS}). Since Q_2 is OFF when Q_1 is ON, and Q_1 is OFF when Q_2 is ON, power is only used during the switching times. Power increases as switching speeds increase, but standby power is very low.

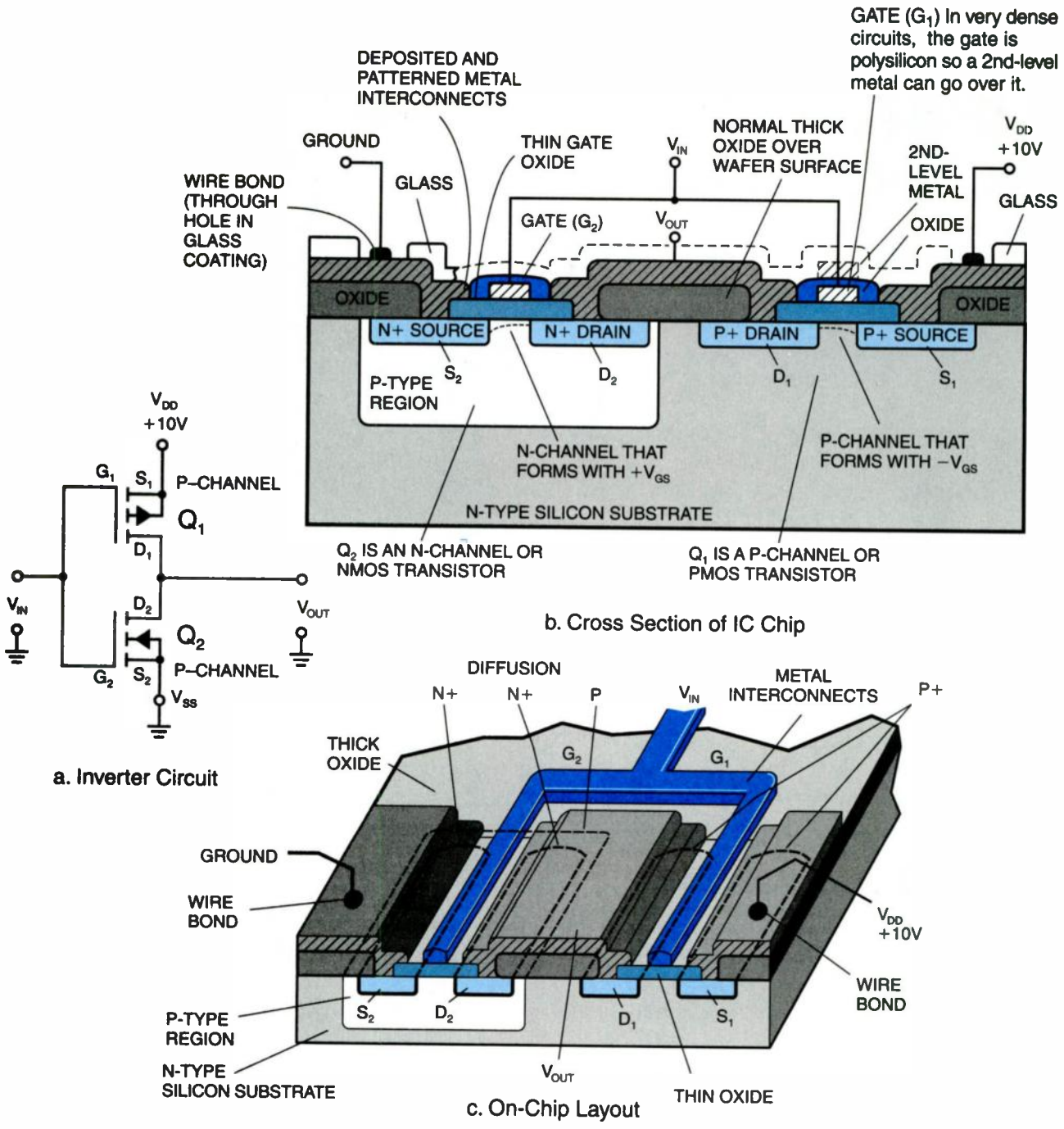


Figure 7-3. CMOS inverter made as an integrated circuit.

IC Layout

The CMOS inverter in *Figure 7-3a* is shown laid out in integrated circuit form in *Figures 7-3b* and *7-3c*. *Figure 7-3b* is a cross section to show how the various regions of the IC chip are modified to make the drain and source of the N-channel and P-channel MOSFETs, and to show how the various regions are connected together with deposited and patterned aluminum. The N-channel has two heavily doped electron-rich N-type source and drain defined and diffused into an electron-poor P-type substrate, which itself was defined and diffused into the original N-type wafer. The N+ source and drain have a gate electrode between them. The gate electrode (aluminum) is electrically isolated by silicon oxide from the P-type region below it. The application of a positive voltage to G_2 creates an N-channel between the source and

drain, turning Q_2 ON. When the G_2 voltage returns to zero, the transistor turns OFF. To form the P-channel transistor, heavily doped P-type regions are patterned and diffused into a lightly doped N-type substrate to form the source and drain of Q_1, the P-channel. Gate G_1 is between the source and drain and is electrically isolated by silicon oxide from the N-type substrate, just like G_2 for the N-channel Q_2. A voltage near V_{SS} (a negative gate-source voltage) on gate G_1 forms a P-channel between source and drain and turns Q_1 ON. Returning G_1 to a voltage near V_{DD} turns Q_1 OFF.

Figure 7-3c is an exploded view to show in more detail how the circuit components are laid out on the chip. It shows how the aluminum is deposited over the chip and removed in a selected pattern to connect the circuit. Note how G_1 and G_2 are connected to V_{IN}, and how V_{OUT} is the connection of D_1 and D_2.

Designing the Circuit

Computer-aided design systems are used to design these circuits. Once the overall function of the chip is specified, engineers divide the layout into easily managed blocks of circuitry. After each block is carefully designed, the computer checks the accuracy of the design and generates master blueprints. These drawings are usually four or five hundred times the actual size of the chip and enable engineers to visually check for errors. The computer then produces a tape of the final design which has been divided into a series of patterns, one for each mask layer. The tape is fed into a computer-controlled electron beam machine to produce the masks.

The end product of the design process is the geometrical layout information separated into mask layers, as shown in *Figure 7-4,* which control where the transistors (and additional components, if necessary) will be located and how they will be interconnected. Using the masks for each layer, the circuit is fabricated on silicon in specific regions of the silicon surface through a series of repetitive steps, such as oxidation, patterning, etching, diffusion, deposition, and etching.

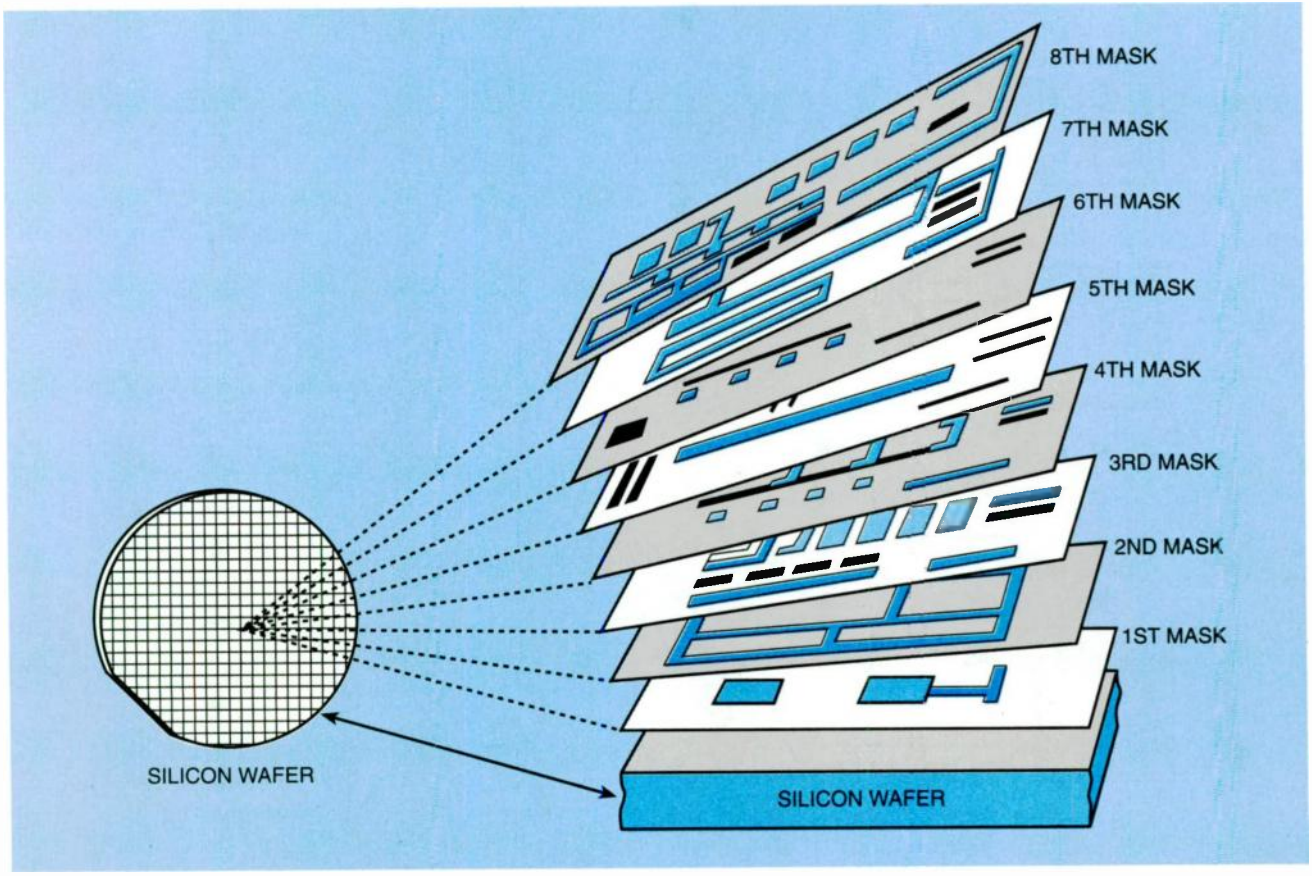


Figure 7-4. Photomask set used to fabricate the CMOS inverter.

Making a Mask

In an ultra-clean environment, a fine electron beam will etch the patterns onto a series of chrome-plated glass plates. Once the glass plates are etched, they become the mask to transfer the circuit patterns onto the wafer. The fabrication of an IC requires from five to twenty or more different masks. The following sections show how the masks are used to build integrated circuits step-by-step.

Fabrication of Integrated Circuits

The major steps in the fabrication process of the CMOS inverter IC are illustrated in *Figure 7-5*. Some minor steps, even though discussed, are omitted in the figure for simplification. We've covered preparing the wafers and preparing the masks, so we begin in *Figure 7-5* with Mask #1.

The fabrication process used to take several weeks. Today, it takes about eight days, working around the clock. To have a successful run, control of contamination is extremely important. Most contamination comes from the human body. Just a few microscopic dust particles on the wafer or mask surfaces can drastically reduce the yield of acceptable circuits; therefore, all operations are performed in ultra, ultra clean areas.

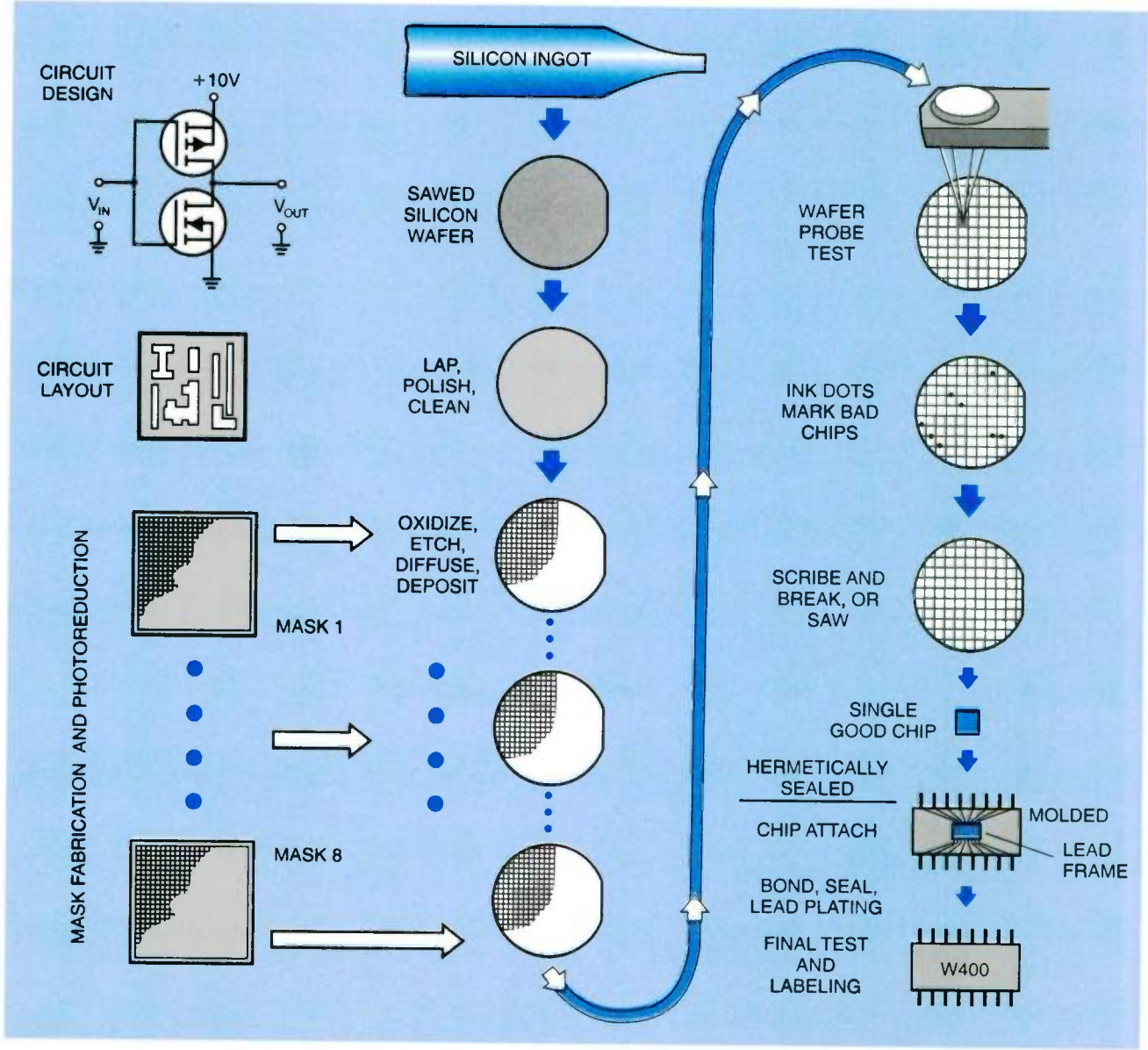


Figure 7-5. Summary of IC fabrication process.

Be aware that silicon dioxide on the surface of a wafer protects the wafer from being modified by dopants that may be on the surface of the silicon dioxide. However, when a hole is cut in the oxide, any dopants that are placed on the surface of the silicon dioxide will be able to modify the wafer in the areas where the holes are cut. Placing the wafer in a furnace will drive (diffuse) the dopants into the wafer and modify the wafer in a patterned way as determined by the holes cut in the oxide.

Also, note that placing a wafer in a high-temperature furnace which has an oxygen-rich atmosphere will cause a layer of silicon dioxide to grow on the surface. Thus, a major subset of steps in the IC manufacturing process is as follows:

a. Grow an oxide.
b. Cut holes in the oxide to modify selected areas on the surface of the wafer.
c. Deposit on the oxide a dopant or place the wafer in an atmosphere that contains the selected doping materials.
d. Heat the wafer to allow the dopant to modify the selected area.
e. Grow additional oxide to re-protect the hole areas.

After the furnace step, and many times in the process, the wafers are cleaned carefully to remove any dopants and then rinsed before new oxide is grown. This prevents one diffusion step from contaminating another.

When the components are to be interconnected, silicon dioxide is the insulator between the aluminum and the wafer. Holes are cut in the oxide where connections are to be made. Aluminum is deposited over the complete surface making connection through the holes to the wafer component. Etching the aluminum with a mask pattern determines how the interconnecting leads will run on the IC chip.

To start the CMOS circuit fabrication, N-type wafers with a specific resistivity are selected. The wafers are cleaned in hot acids to make sure that the wafer surfaces are absolutely free of any contaminants. Then the wafers are rinsed in deionized water and spun dry with filtered nitrogen gas.

Oxidation

In a high temperature oxidation furnace, silicon dioxide will be grown on the wafer. Silicon dioxide is a good insulator, which protects the silicon substrate beneath it from unwanted reactions. The wafers are placed in a hot furnace with pure oxygen which reacts with the silicon surface to form a thin layer of silicon dioxide. This process is similar to the way the sun's heat and the air's oxygen turn a car's shiny new paint job to a dull coat over the years. As stated above, the silicon dioxide is the protecting layer to keep the wafer from being changed.

Photolithography Process

The photolithography process transfers the circuit image from the mask to the silicon wafer. Making a CMOS FET inverter IC involves eight different masks as shown in *Figure 7-4*. Refer back to *Figure 7-3b* to determine what happens using the first mask. Note the P-type region into which the N+ source and drain of Q_2 are diffused. This is the first region to be modified with Mask #1. Mask #1 will cut holes in the silicon dioxide on the wafer surface to allow dopants to change the N-type wafer into P-type material in selected areas.

Photoresist

The first mask pattern is transferred onto the wafer using photoresist, a material that is sensitive to light, and can be dissolved in solvents. It is evenly spread over the surface of the wafer and baked at a low temperature. There are two kinds of photoresist — negative and positive. A negative photoresist hardens when exposed to light and remains on the wafer when developed. If a positive photoresist is exposed to light, the photoresist changes chemically and dissolves when developed.

Alignment and Exposure

A computer-controlled machine called a "stepper" positions the wafer underneath the selected mask pattern. Ultraviolet light exposes the pattern across the surface of the wafer. For our Mask #1, the area for the P-type diffusion is exposed to ultraviolet light and will be dissolved by the solvent, making a hole where the silicon dioxide will be etched away. In this way, many chips are made from a single wafer. (The larger the diameter of the wafer, the more chips can be made on each wafer.) Using hydrofluoric acid, the holes are etched in the silicon dioxide. The hardened resist is then removed with hot acids leaving the silicon dioxide layer in place except where there are holes. This photolithography process is used with each of the masks. This technique allows us to isolate microscopic regions of the wafer and construct them into components of the electronic circuit.

Ion Implantation and Diffusion

The process of applying the dopant for the P-type regions starts with an ion implanter. P-type dopant ions bombard the wafers in the exposed wafer areas. Placing the wafers in a diffusion furnace drives the ions deep into the exposed areas, producing the P-type well into which the N-channel MOSFET is to be built. The depth of the ion implantation depends on the amount of energy used. If the wafers had been P-type, then N-type dopants would have been implanted. So either P-type or N-type regions can be formed into the exposed areas. A new thinner layer of silicon dioxide is now regrown on the wafer.

Mask #2 and #3

Photoresist is again evenly spread on the wafer and baked in preparation for the second mask. Using a computerized machine, each new mask is perfectly aligned to the pattern already on the wafer. Ultraviolet light is applied to expose the second mask and its patterns. The wafers are developed to remove the exposed photoresist, etched to open the diffusion holes, ion implanted for the dopant, and diffused to change the wafer in the selected areas.

The photoresist is removed and the wafers go back into the oxidation furnace. After many steps, a thick insulating layer of silicon dioxide, known as field oxide, has grown over most of the wafer. To grow these thick layers of oxide, oxygen combined with hydrogen is introduced on the wafers as steam. The thick field oxide reduces the electric fields between the surface and the underlying regions. It will block the current from leaking between devices, allowing thousands of transistors to coexist in a small area. That is why modern computers can be so fast — there is not much distance for a signal to travel between transistors. Mask #3, used for a drain and source N+ diffusion has the same steps as Mask #2.

In the case of our *Figure 7-3b* example, Mask #2 forms the P+ areas for the source and drain of Q_1, and Mask #3 forms the N+ areas for the source and drain of Q_2.

As shown in *Figure 7-3b* at gate G_1 and gate G_2, the oxide underneath the gates is much thinner than the field oxide so that the transistor characteristics will be as needed. In addition, look at Gate G_1. In very dense circuit layouts for CMOS ICs, the gate metal is actually a polysilicon — a silicon doped with phosphorous to make it more conductive. It is polysilicon so that an oxide can be grown over it to insulate it from a second-level layer of metal connections that run over it in order to interconnect all of the dense circuitry.

Special masking is required to form the gate oxide and pattern it, open the holes for the polysilicon metal layer, and pattern the polysilicon. In our simple example, only Masks #4 and #5 are needed, but extra masks would be needed if the gate were polysilicon.

Mask #6, #7 and #8 for Interconnections

Photolithography and mask #6 define the holes through which electrical contact will be made between the aluminum metal and the silicon area used for the MOSFETs. The unmasked holes in the silicon dioxide are created using plasma etching. Mask #7 patterns the aluminum into its connection conductors. Either wet acid etching or dry plasma etching are used to remove the excess aluminum. As a result, we have an interconnected integrated circuit.

There are wide aluminum pads all along the outer edges of the chip. They are the contact points where wires are bonded to the pads and to leads that extend through the package to make external connections. At low temperatures, a final layer of glass is deposited to protect the fragile aluminum-silicon interconnects. Using Mask #8, photolithography steps remove only the glass on top of the bonding pads. An example is shown in *Figure 7-3b.* The wafers are then stripped of photoresist and the run is complete.

Final Wafer Processing

We are now at the wafer probe step in *Figure 7-5.* After the wafers have been fabricated, they move on to electrical testing and packaging. In testing, the wafer slides beneath a probe card. Thin metal probes make contact with each aluminum pad on the circuitry and a machine tests the electric parameters against predetermined specifications. A computer keeps track of which circuits have failed. In the inking machine, the failed circuits are marked with a black dot. After the individual circuits have been tested, they are ready to be separated either by sawing or by using a procedure called dicing. Dicing is accomplished by scribing the wafer and then breaking it into individual chips.

If the sawing process is used, a diamond saw cuts through only the wafers that have been mounted on sticky mylar tape to keep the chips in place after separation. An electronic eye detects the ink blots that mark the failed chips and skips them. The good dies are removed and go on to further assembly.

Assembling the Integrated Circuit

Assembly is the most expensive part of the fabrication process. When processing the wafers, large numbers of circuits are processed simultaneously. During assembly, however, each circuit is handled individually and connections to each die are made separately. The chips are mounted on a lead frame, shown in *Figure 7-6a,* if the package is a molded package, or directly to a plated surface on a package substrate that has pins that extend through it. The lead frame contains the number of leads on the molded package. The substrate package has a seal placed over the top of the package.

Next, thin gold wires that make the electrical connections between the chip and the lead frame are physically attached. As shown in *Figure 7-6b,* thermocompression bonds attach one end of a wire to the chip bonding pads and the other end to the lead frame pins. When the wire bonding step is finished, the chip is sitting on a metal pad (for heat dissipation) in the center of the lead frame with the gold wires fanning out to the lead frame pins — much like a spider sitting in the center of an its web. The substrate package looks similar with the gold wires connecting from the bonding pads to the package pins.

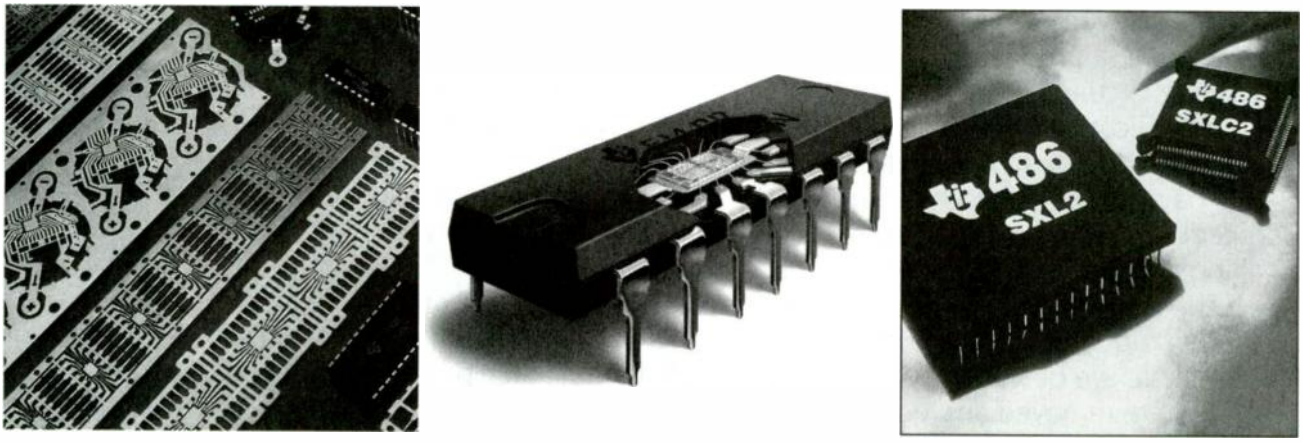
Packaging the Integrated Circuit (Molded Package)

Molding

To protect the circuitry and its fragile wire bonds, the IC is now encapsulated to protect it environmentally. A molded package is shown in *Figure 7-6b*, and several hermetically sealed packages are shown in *Figure 7-6c*. Plastic molded packages require a two-sided mold. Plastic pellets are heated and the melted plastic flows into rectangular cavities which hold the ICs mounted on lead frames. After several minutes the molding is complete. Excess plastic is removed from the mold, and it is ready to use again. In ovens, the plastic molding is fully cured so it can withstand the plating process.

Lead Plating

The leads are plated so that connection to external circuitry will be easy and reliable. In the plating process, the molded packages are loaded onto carriers and dipped in a series of chemical and water wash baths. In the plating baths, with anodes of tin and lead, the leads are plated with a layer of solder so that the chips can be easily attached to a printed circuit board. We have not pointed it out before, but the lead frame actually accommodates eight ICs at a time, as shown in *Figure 7-6a*. Therefore, the molding machine molds eight ICs at a time. After molding, the eight ICs are sawed apart from the strip. The leads are cut from the frame, then formed and trimmed to their final configuration. The ICs are now in individual packages.



a. Lead Frame Strip b. Molded Package c. Hermetically Sealed Package

Figure 7-6. ICs are protected from the environment using plastic molded packages or hermetically sealed packages. Courtesy of Texas Instruments Incorporated

Burn-In and Testing the IC

The individual ICs are mounted on special boards and electrical power is applied to them. They are then placed in special chambers where the temperature is cycled between very hot and very cold. This procedure stresses the ICs far beyond normal conditions so that weak chips are forced to fail at this point rather than have a premature failure in a customer's product.

For final electrical tests, a large computer controls the handling of the ICs, the type of electrical tests that are run, and the recording of test results. The computer sorts the ICs according to their test results — some have failed, some have met full specifications, and some have met lower specifications. Each IC is marked with a type number for the specification level it met and with the manufacturer's logo. It is visually inspected and packaged for distribution. It is now ready for a customer to use it as part of an electronic system.

Summary of IC Fabrication Process

All of the IC fabrication technologies and all of the devices that are integrated on a chip share some common features. These include circuit design and layout, mask fabrication and photoreduction. As we have seen, there are many processes that are repeated over and over again. These include oxidation, application of photoresist, exposure to ultraviolet light, etching, and diffusion. Each of the steps (including those that are proprietary for each manufacturer) determine the cost and quality of the final chip.

This monolithic fabrication process is best suited for high-volume production of ICs, especially for digital circuits, but applies to analog (linear) ICs as well. Generally, CMOS ICs are preferred for high-density circuity, but TTL logic circuits using bipolar transistors for monolithic integration have been, and continue to be, very popular. Let's look at a very simple example of a bipolar integrated circuit.

A Bipolar Junction Transistor Integrated Circuit

The simple emitter follower of *Figure 7-7a* will be used to show the bipolar monolithic IC fabrication process. Whereas the CMOS IC had no diffused resistors, this circuit has two, R_B and R_E. The circuit has four external connection points:

Pin 1. The input signal through R_B to the base.
Pin 2. The +5 volt power supply.
Pin 3. The output signal at the emitter.
Pin 4. The signal and power supply ground.

The emitter resistor R_E is connected between the emitter and the ground terminal. A cross section of the monolithic IC structure is shown in *Figure 7-7b,* along with its overall layout.

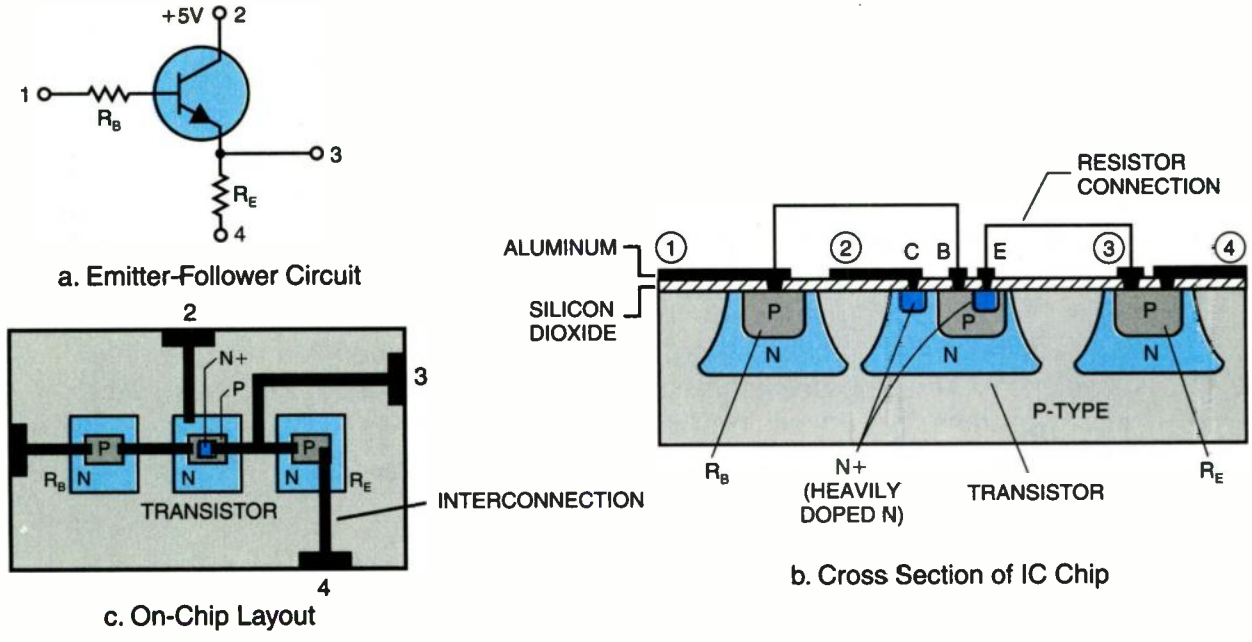


Figure 7-7. Monolithic IC process of simple emitter follower.

The fabrication of this emitter follower requires five masks. This is typical of a bipolar monolithic IC regardless of the complexity of the circuit being integrated. The processing steps for this chip are as follows:

1. Start with a P-type substrate.
2. A layer of silicon dioxide is formed by oxidation.
3. Photoresist is applied to the silicon dioxide layer.
4. Mask #1 is placed over the wafer, the photoresist is exposed to ultraviolet light, and establishes the openings through the silicon dioxide layer when it is washed in solvent.

5. Hydrofluoric acid is used to etch away the unexposed photoresist.
6. An N-type silicon layer is diffused for transistor collector and resistor wells.
7. The wafer is cleaned, a new layer of silicon dioxide is grown by oxidation, and photoresist is applied again.
8. Mask #2 is placed over the wafer and the photoresist is exposed and dissolved. When the silicon dioxide is etched, this mask provides the windows for the diffusion of P-type dopant for the transistor base and the two resistors. The length, width and resistivity of the P-diffusion determines the resistance value.
9. The P-type dopant is diffused through the windows to form the base and resistors.
10. The wafer is cleaned and a new layer of silicon dioxide is grown
11. Photoresist is applied, mask #3 is placed on the wafer, and the photoresist is exposed to form two windows for N-type diffusion of the emitter and collector regions.
12. The slice is etched and the N-diffusion is made.
13. The wafer is cleaned and a new layer of silicon dioxide is grown.
14. Photoresist is again applied, mask #4 is aligned, and the photoresist is exposed and dissolved. The silicon dioxide is etched and provides windows for contacts of connections to the terminals of the components.
15. The wafer is cleaned and the entire top of the substrate is deposited with a metallic aluminum film for interconnection of the circuit elements.
16. Photoresist is applied.
17. Mask #5 is applied to expose the photoresist and, when dissolved, provides the interconnection pattern.
18. The excess aluminum is removed by etching.
19. The wafer is cleaned, probed and the chip is packaged and tested.

The NPN transistor like the one above is ideal for manufacture of monolithic ICs. If a PNP transistor is needed, it may be formed by a thin N-type region between two P-type regions.

Evolution of IC Fabrication

The concept of integrated circuits was an outgrowth of printed circuit techniques and diffused transistor fabricating technology. The trend of the integrated circuit industry from its beginning has been to reduce the geometry of the size of the IC components. This has been true for both bipolar and MOS ICs. Size reduction requires improved resolution in photolithography operations, as well as the processing techniques. These improvements have resulted in a tremendous increase in the density of transistors on a chip. The following table shows this exploding evolution.

Year	Integration	Number of Transistors
1950	small scale	1 -10
1960	medium	10 -100
1970	large	1000 -10,000
1980	very large	10,000 -100,000
1990	ultra large	1,000,000+

Summary

In this chapter, the basics of the technology used to fabricate modern monolithic integrated circuits have been introduced. The emphasis has been on the CMOS technology because of its increasing popularity. We also discussed the long-time standard, the bipolar junction transistor fabrication in an IC. The trend of more CMOS type of ICs will undoubtedly continue. In the next chapter, we will see how integrated circuits are interconnected to form a computer system.

Quiz for Chapter 7

1. The _____ family is the most widely used IC of the bipolar type.
   a. TTL
   b. CMOS
   c. hybrid
   d. thick-film
2. On the CMOSFET inverter, the two gates G_1 and G_2:
   a. are driven by out of phase signals.
   b. are connected together.
   c. are both made of N material.
   d. are both made of P material.
3. The master blueprints of the circuitry to make an IC are usually
   a. drawn to scale.
   b. reduced to about one-hundredth their original size.
   c. four or five hundred times the actual size of the chip.
   d. drawn by hand.
4. Most contamination in the IC fabrication process comes from
   a. the human body.
   b. operators failing to wash their hands.
   c. outer space radiation.
   d. dirt in the silicon from its original sand.
5. Placing a wafer in a high temperature furnace with an oxygen-rich atmosphere will cause
   a. the formation of N-type semiconductor.
   b. moisture to form on the wafer.
   c. aluminum to be deposited on the wafer.
   d. silicon dioxide to be deposited on the wafer.
6. The first mask is transferred to the wafer using a light-sensitive material called
   a. dopant
   b. silicon dioxide
   c. photoresist
   d. photosynthesis
7. The depth of ion implantation depends on the amount of _____ used.
   a. pressure
   b. energy
   c. water
   d. nitrogen
8. Whether a P- or N-type region is desired determines the type of _____ used.
   a. photoresist
   b. exposure
   c. dopant
   d. emitter
9. A black dot is placed with an inking machine on certain circuits on the wafer to indicate that the circuit
   a. is to be sold at premium price.
   b. failed the predetermined specifications.
   c. contains inductors.
   d. is a MOSFET.
10. The connections between the chip and the lead frame are made with
    a. solder
    b. copper clad
    c. aluminum wire
    d. gold wire
11. The IC is commonly encapsulated to protect it environmentally in a
    a. plastic package.
    b. wooden case.
    c. aluminum box.
    d. polyethylene bag.
12. In a silicon IC, the silicon dioxide is used to:
    a. allow chemical contaminants to reach the silicon surface.
    b. provide electrical connections to the silicon surface.
    c. define geometric patterns to be photographed.
    d. block diffusing material from reaching the silicon.
13. The MOS fabrication process:
    a. uses totally different physical procedures from bipolar processing.
    b. uses essentially the same physical procedures as bipolar processing.
    c. is primarily a method for fabricating resistors.
    d. requires many more processing steps than bipolar processing.
14. Diffusion is the process of:
    a. changing metal to metal oxide.
    b. changing silicon to silicon dioxide.
    c. providing metal contacts to silicon.
    d. introducing an impurity into a substance.
15. The trend in the IC industry is to
    a. decrease the operating speed.
    b. decrease the component size.
    c. increase the power dissipation.
    d. decrease component count on a chip.

Answers: 1a, 2b, 3c, 4a, 5d, 6c, 7b, 8c, 9b, 10d, 11a, 12d, 13b, 14d, 15b

Questions and Problems for Chapter 7

1. How does a CMOS inverter get by without a drain resistor?
2. What are the power requirements of a CMOS circuit?
3. Name the series of repetitive steps in the IC fabrication process.
4. How many masks does the fabrication of an IC take?
5. Describe the area where IC manufacturing operations are performed.
6. How is one diffusion step prevented from contaminating another?
7. What is used in the process as an insulator and to protect the silicon substrate from unwanted reactions?
8. How is the circuit image transferred from the mask to the silicon wafer?
9. What is photoresist and what is it used for?
10. Why have wafer manufacturers attempted to make larger wafers?
11. What are the wide aluminum pads along the outer edge of the chip used for?
12. What is used to make the connection between the chip and lead frame in a molded IC package?
13. What fabrication method is best suited for high-volume production of ICs?
14. What has been the trend of the IC industry from its beginning?

CHAPTER 8

Putting Functions Together Into a System

Introduction

We have now covered the details of the digital electronic functions for which we gave an overview in *Chapter 2,* and are ready to combine these functions into a total system. We will use a computer system for study because it is a complete digital system and is representative of other digital systems. The system must accept inputs from outside the system (usually from humans) and provide outputs that can be used by external systems or humans. Typically, inputs come from keyboards, mouse or track-ball pointers, magnetic disks, and CD-ROMs (compact disk read-only memory). Typically, outputs go to printers, video displays (monitors), magnetic disks, magnetic tape, and speakers.

Computer Systems

Keep in mind that the term "computer system" can be applied to many types of systems. Personal computers are either desk-top or lap-top units, and are intended for single-user operation. Minicomputers are more powerful and faster than personal computers and are intended to serve several users simultaneously. Mainframe computers are even larger and faster, and serve a large number of users such as banks, government, and universities. Or a computer system can be a system within a system—a single IC in a hand-held game, a combination of ICs in a manufacturing control system, or a dedicated computer to control functions in an automobile, airplane, or various types of consumer products.

All of these systems can be described by the block diagram in *Figure 8-1a.* They vary by the number or size of the Input/Output (I/O), the Central Processing Unit (CPU) and the Memory (ROM or RAM.) Of course, large systems will have a large CPU and memory, and usually will have many input and output terminals. On the other end of the scale, a single-chip microcomputer designed on one IC will have all subsystems on the same piece of silicon. The main restriction is the amount of memory that can be placed on the single chip. These are used in electronic games and toys, identification cards, and small controllers.

We will limit our discussion to the general system in *Figure 8-1,* and orient it to a system, similar to a basic personal computer, with keyboard and mouse inputs, and video display and printer outputs. We will introduce the concept of *programmed* digital electronic systems—the category into which most digital computer systems fall.

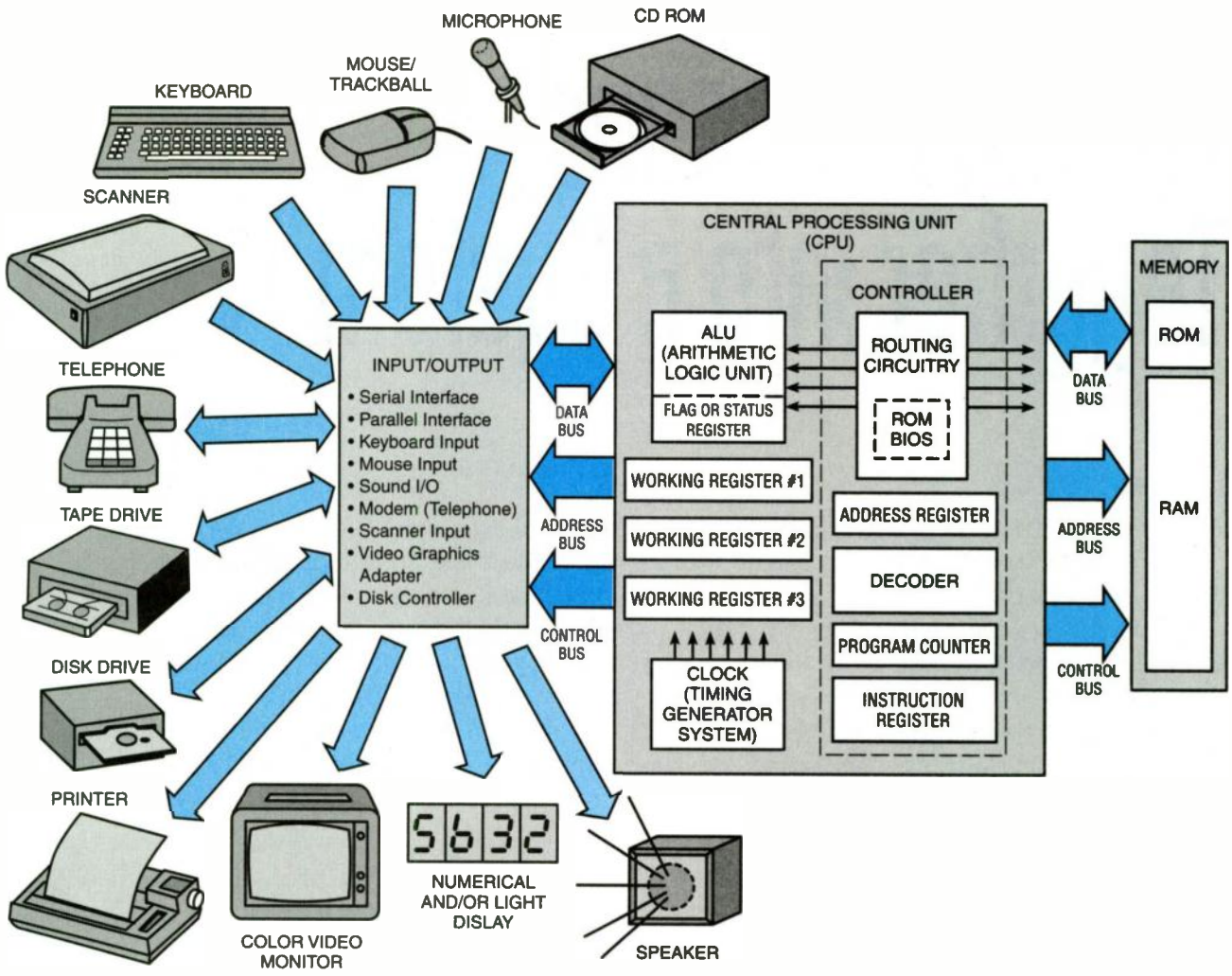


Figure 8-1. A basic computer system made up of I/O, CPU and Memory.

Bus Structures

Notice in *Figure 8-1* that three buses are shown: control, address, and data. *Figure 8-2* expands these buses so we can examine them in more detail. Note that the combination Input/Output block of *Figure 8-1* is divided into separate blocks in *Figure 8-2.* A bus is a group of electrical conductors that make electrical connections between various subsystems. The conductors may be a bundle of insulated wires, often in a "ribbon" configuration, or a group of parallel traces on a printed circuit board. The purpose of a bus is to have common conductors connecting common signals to the subsystems. Standard practice is to refer to the individual lines of a bus by numbers from 0 (for LSB) to the highest number—7 for an 8-bit bus, 15 for a 16-bit bus, 31 for a 32-bit bus, and 63 for a 64-bit bus. Usually, a prefix is used with the number to identify the bus: A for address, D for data, and C for control. Thus, the address bus might be identified with A_0 - A_{15}.

Control Bus

The control bus signals activate the I/O units, memory, and various subsystems and coordinate the timed operation of their digital circuits. They control whether a memory is in a read or write cycle, which way the data flows through an I/O unit, register load or transfer, and ALU operations.

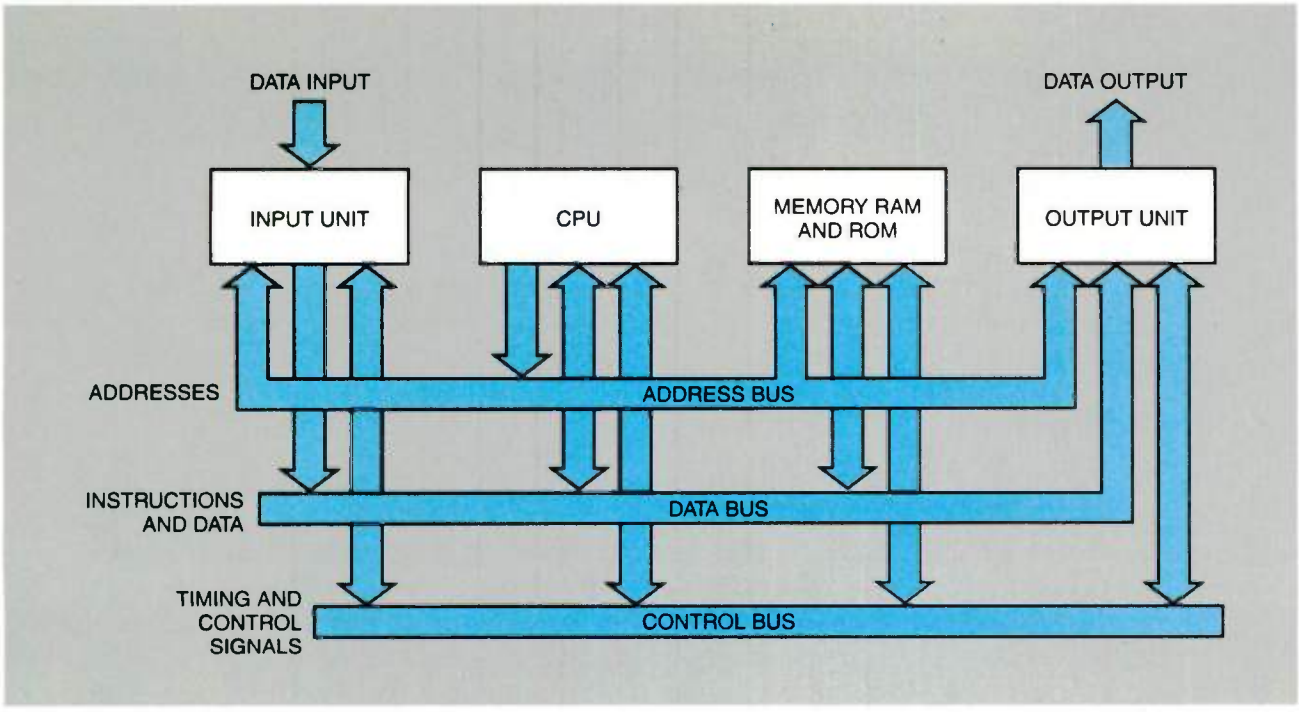


Figure 8-2. A bus is a group of wires bundled together that carry digital signals.

Address Bus

Address bus signals are digital codes that identify a specific I/O unit or a unique location in memory. Remember, if the address bus has 16 lines, it can address 2^{16} or 65,536 individual memory locations.

Data Bus

Data bus signals may flow either way on the bus, but only one way at a time. Unique digital codes that represent a number, letter, or special character, transfer information from one digital system circuit to another. The CPU uses the data bus to transfer data between I/O units, CPU and memory. The data bus is also connected directly to registers in the controller so that information can be transferred quickly from memory to these registers.

CPU

As shown in *Figure 8-1,* the CPU is made up of a controller, an ALU, working registers, and a timing generator which produces clock signals that synchronizes all of the computer system operations. The CPU is the nerve center of the computer system. Through the circuitry in the controller, the CPU controls all operations, interprets data, and switches the interconnections and flow of information to or from the I/O, to or from the memory, to or from the ALU, and to or from the working registers. All these signals occur in precisely timed sequences to complete the task the CPU is presently executing. It contains almost all of the different types of decoding, decision, selection, conversion, buffering and temporary storage circuits we have talked about.

When the CPU is contained on and within a single IC, and packaged as such, it is called a *microprocessor.* Most personal computers use a microprocessor that operates at clock signals approaching 200 MHz. A modern day microprocessor has millions of transistors made with CMOS technology on a single chip of silicon. The final package is about 2½" to 3" square.

I/O

As shown in *Figure 8-1,* I/O units communicate with the external world. Some of the devices only input data, e.g., keyboards, mice, and scanners. Some of the devices only output data, e.g., monitors and printers. Others may do both; e.g., magnetic disks, magnetic tapes, modems, and telephones. In many cases, a unique digital subsystem is required to interface an I/O device to the digital system. These subsystems often are contained on plug-in cards that can be added to the digital system when the I/O device is added. The subsystems contain decoder, data selector, register, 3-state buffer, temporary storage and logic functions—all timed by the controller in the CPU.

Addressing

The CPU uses a method called *addressing* to select a particular I/O device from the several that may be connected. The basic concept of addressing is shown in *Figure 8-3a.* It is the same as assigning house addresses on a street, but in the digital system, the street is the address bus and the house addresses are instruction addresses. As shown in *Figure 8-3b,* when the address, which is a unique code for a particular I/O device, comes down the bus, decoder circuits in the I/O units with that address activate. With the addressed I/O unit activated, the controller in the CPU directs the I/O interface circuits to deliver data to the CPU or receive data from the CPU. Some I/O units send or receive data in parallel; others use serial transmission.

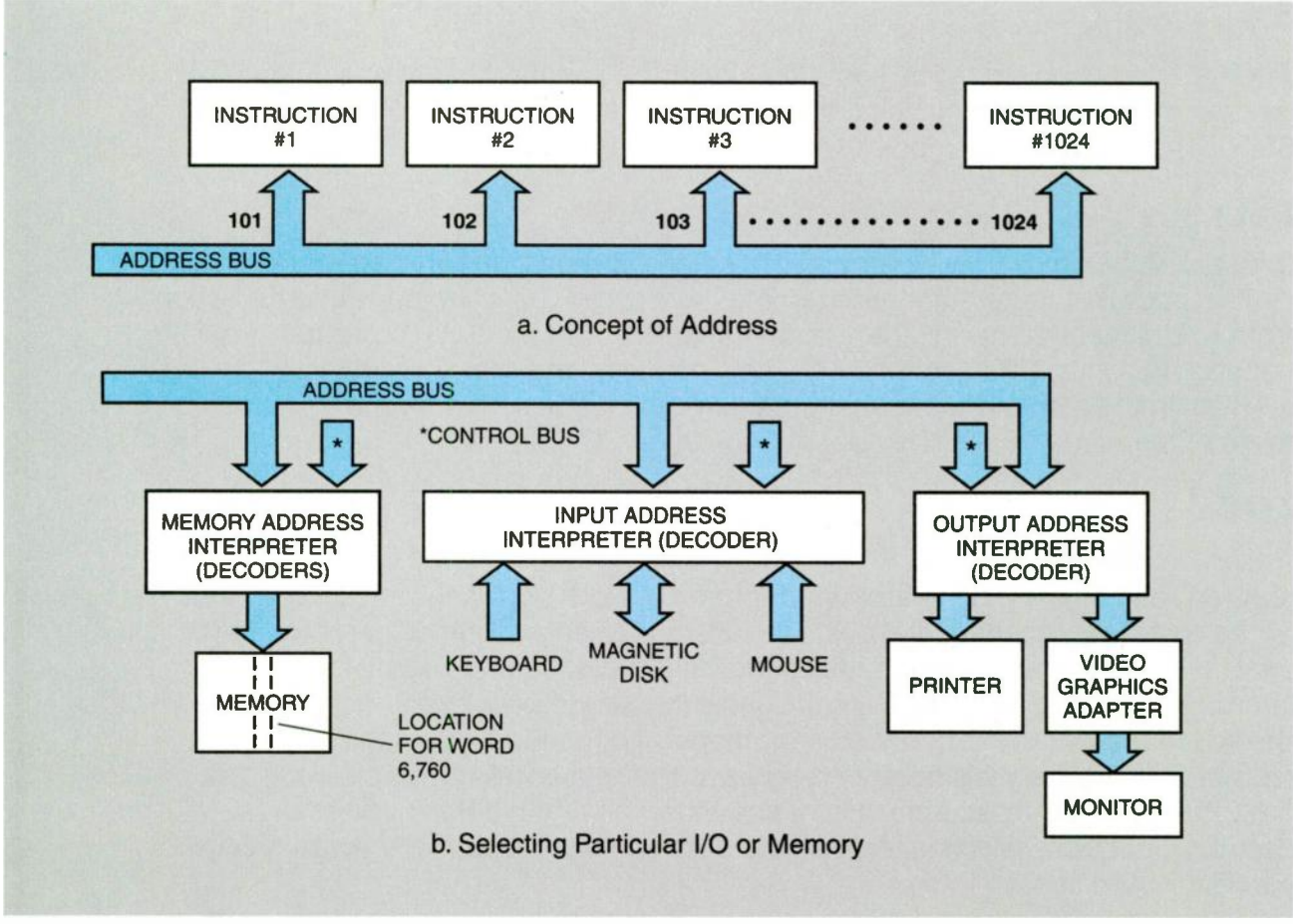
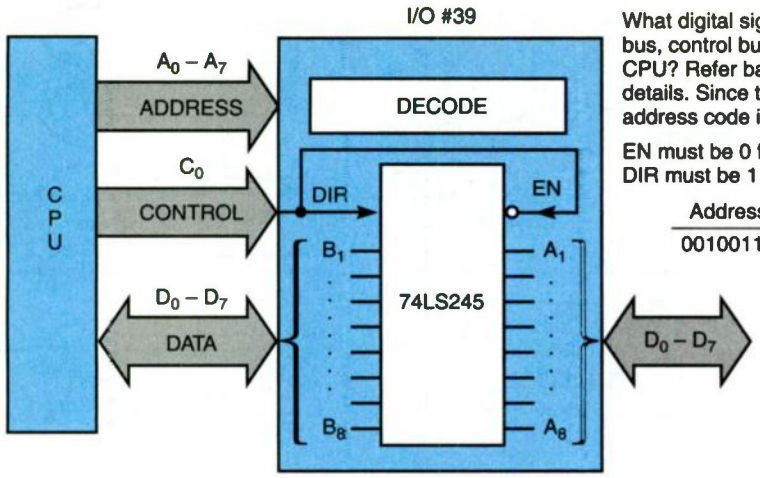


Figure 8-3. A unique address is assigned to each I/O unit as well as to each memory location to allow the CPU to activate it when needed.

Example 1. Input of Data from I/O



What digital signals must be on the address bus, control bus, and EN to input data to the CPU? Refer back to *Figure 5-4* for 74LS245 details. Since the I/O unit is No. 39, its address code is 00100111.

EN must be 0 for enable.
DIR must be 1 for A to B.

Address	Control	EN
00100111	1	0

CONTROL		
INPUTS		STATE
EN	DIR	
1	X	Disabled
0	1	A to B
0	0	B to A

Memory

The system memory, as we discussed in Chapter 6, can be either RAM or ROM. The information in ROM is meant to be used as is without changes. The information in RAM is meant to be stored only temporarily and it can be changed readily. Recall, for that reason, we called RAM a read/write memory. The stored data is read when needed, and changed by writing in new data as needed. The information in RAM can be a variety of different digital codes: a series of codes called the operating system, instructions to the controller, data that the digital system needs for the task that it is executing, or it can be special reference data, such as, time of day, date and year, etc. A memory manager makes sure that the different digital codes are stored in the correct portion of the memory.

An example of ROM data is information to initialize a system. The information is a series of steps that the digital system must go through to "boot-up" a system. As soon as power is applied, the system is directed to the first address in ROM and from there follows the initialization routine. Notice in Figure 8-1, there is a ROM BIOS indicated inside the controller. The BIOS (Basic Input Output System) holds the "boot-up" instructions and specific instructions for controlling input and output. In modern day PCs, the BIOS is a separate ROM.

Each piece of information in memory, either ROM or RAM, is stored at a particular address. When the CPU needs the information, it puts the address on the address bus. At the same time, it sends timing and control signals via the control bus to put the memory in the read mode. The memory is read and the data is sent on the data bus to the CPU. If the CPU wanted to write data to memory for storage, the addressing sequence would be the same, but the CPU would put the data on the data bus and put the memory in the write mode.

We now know the operation of the main subsystems of a digital system. Let's now examine the CPU in more detail.

The CPU ALU — Its Operations

The arithmetic-logic unit in the CPU is very important. It is made up of digital electronic circuits that perform arithmetic, logic, signal routing and function selection . It may use the working registers of the CPU, or it may have its own registers. A block diagram of an ALU is shown in *Figure 8-4.* The control signals on the control bus determine which operation is performed on the two inputs A and B. The typical kinds of ALU operations are shown in *Figure 8-4a.*

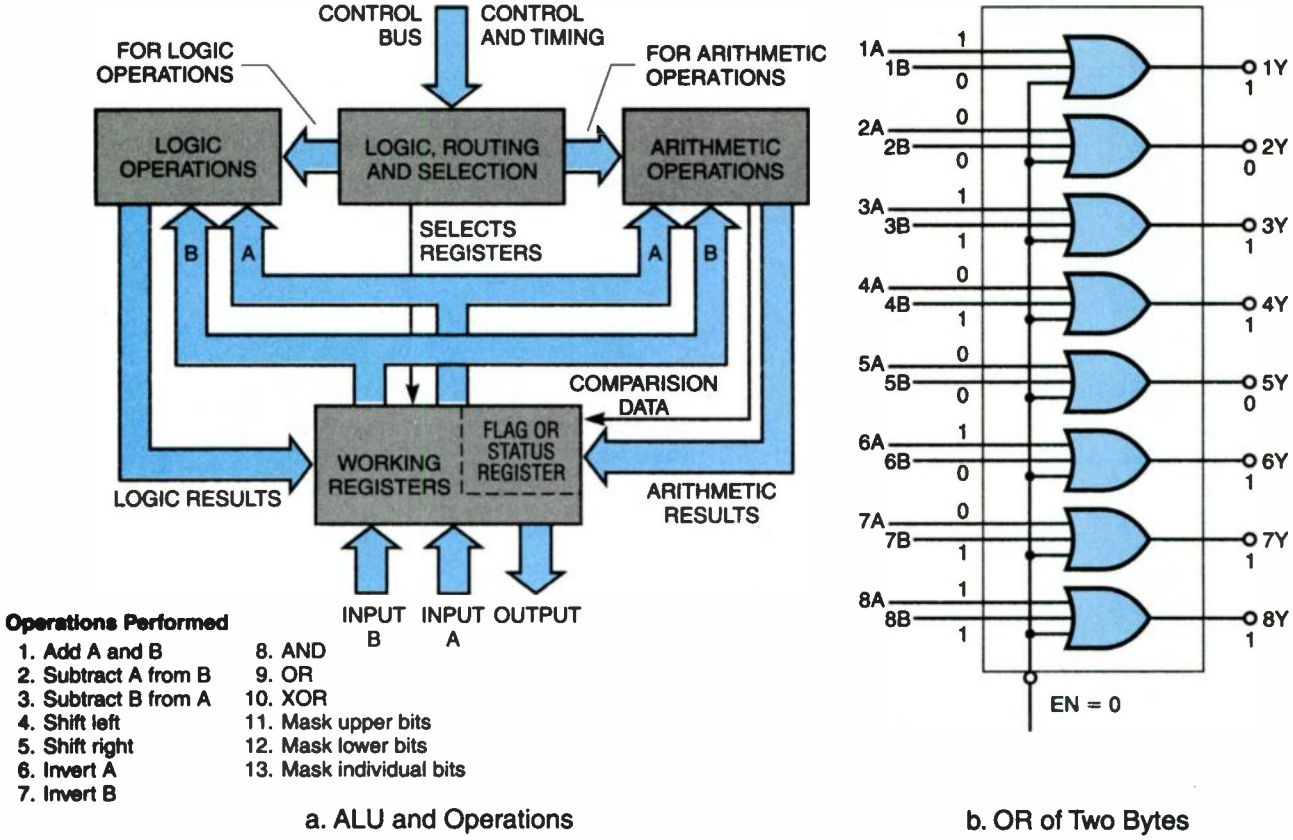


Figure 8-4. An ALU can do many operations other than adding and subtracting.

Arithmetic Operations

With properly timed instructions, even though it only has addition and subtraction capabilities, the ALU can multiply by successive addition as we showed in Chapter 5, and divide by successive subtraction. ALUs are available to handle from 4-bit to 32-bit addition. (ALUs that do 32-bit addition must handle 64 bits.) Comparison of numbers is accomplished by using subtraction to generate a digital signal that indicates that A is equal to B, A is greater than B, or B is greater than A. The comparison result, called a *flag,* is stored in a flag register. The CPU periodically checks the flag register to determine the status of the task it is executing. Another task that the ALU performs is the inversion of bits to provide the 1's complement of a number for subtraction.

Logical Operations

The logical functions performed by the ALU are AND, OR, and XOR. The function is performed on all bits of A and B in parallel. An example of an OR operation between an A byte and a B byte is shown in *Figure 8-4b.* The output byte corresponds bit-by-bit to the result of the operation which is enabled by placing a 0 on the EN(enable) line. Both inputs A and B must be present and stable when EN=0 to make any logic operation valid. The XOR operation is very useful for comparing two binary codes to determine if they are identical. Upon completing an XOR operation, any code bits that are not the same will have a 1 in the output result bit position.

Masking Operations

The ALU also can do masking of bits of digital words sent to it. *Masking* means to cover up, or ignore, some of the bits so the next operation is performed only on the unmasked bits. The ALU in *Figure 8-4a* can mask lower bits or upper bits of a byte, and individual bits.

Example 2. Masking and Comparing Bits

a. An AND ALU operation can be used to mask bits.
Do an AND operation using byte A and the masking code B shown for masking the lower four bits of a byte, upper four bits and for examining the LSB.

	Lower Bits	Upper Bits	Examining LSB
Byte A	10101101	10101101	10101101
B (masking code)	11110000	00001111	00000001
Result	10100000	00001101	00000001

b. Compare the two digital codes A and B to determine if they are the same.
Send the two codes to the ALU and do an XOR operation on them. If a 1 appears in any bit position in the output, the codes are not the same.

	Same Codes	Different Codes
Code A	01011001	01011001
Code B	01011001	01111001
After XOR	00000000	00100000

Working Registers

ALU shift operations occur in the working registers. Input to, and output from, the registers is usually accomplished by moving all the bits at once in parallel. Some of the registers have serial links between them so that bits can be shifted left or right between the registers. If two registers linked together are shifted in the same direction at the same time, the bits of one register shift to the other. The shifting is timed and sequenced by the controller. Such an operation is very useful in some multiply operations. The working registers can be incremented (one added to the contents) or decremented (one subtracted from the contents). These are very important operations to a programmed digital system.

In *Figure 8-1,* the working registers are identified as #1, #2 and #3; however, in a real system, the registers often have names that indicate their function. The A and B inputs to the ALU are called operands, so the register that holds them may be called the *operand register.* As the ALU steps through an arithmetic operation, the register that holds the results is called the *accumulator*. Other registers shown in *Figure 8-1* are the address register and the instruction register, both in the controller, and the flag register, which we have discussed.

Controller

If the nerve center of a digital system is the CPU, then the nerve center of the CPU is the controller. Signals from it tell all the other parts of the digital system what to do and when to do it. It directs received data to the correct register, clocks the register to receive the data, shifts the data if necessary, and stores the data in the specified memory location. It tells the ALU which operation to perform and when to perform it. For example, it places the address digital code in the address register for the memory to use when it tells the memory to read or write. If the memory is read, the memory contents may be placed in the instruction register to execute further operations.

Timing Generator System

All the operations of a digital system are synchronized by a clock. *Figure 6-6* of *Chapter 6* showed a timing generator in which four separate timing pulses were generated from an input clock. Such a generator is know as a four-phase clock. The very accurate clock pulses are derived from a high-frequency crystal-controlled oscillator that is held in precise time by the stability of the crystal. The crystal is often contained in a controlled-temperature oven to prevent frequency variations due to ambient temperature changes. As we stated previously, some PCs are operating at 200 MHz. That's only 5 nanoseconds (5×10^{-9} seconds) per cycle!

As timing circuits distribute the clock signals to the point of use, logic gate propagation time and the charging of distribution-line capacitance may delay the triggering transition time to occur later than it should. This can cause operational errors. These delays are very small or non-existent in microprocessors because the circuitry is very close together. Thus, using microprocessors can provide a significant advantage for the CPU, and for the accurate timing of the system.

Simple 4-Bit Computer

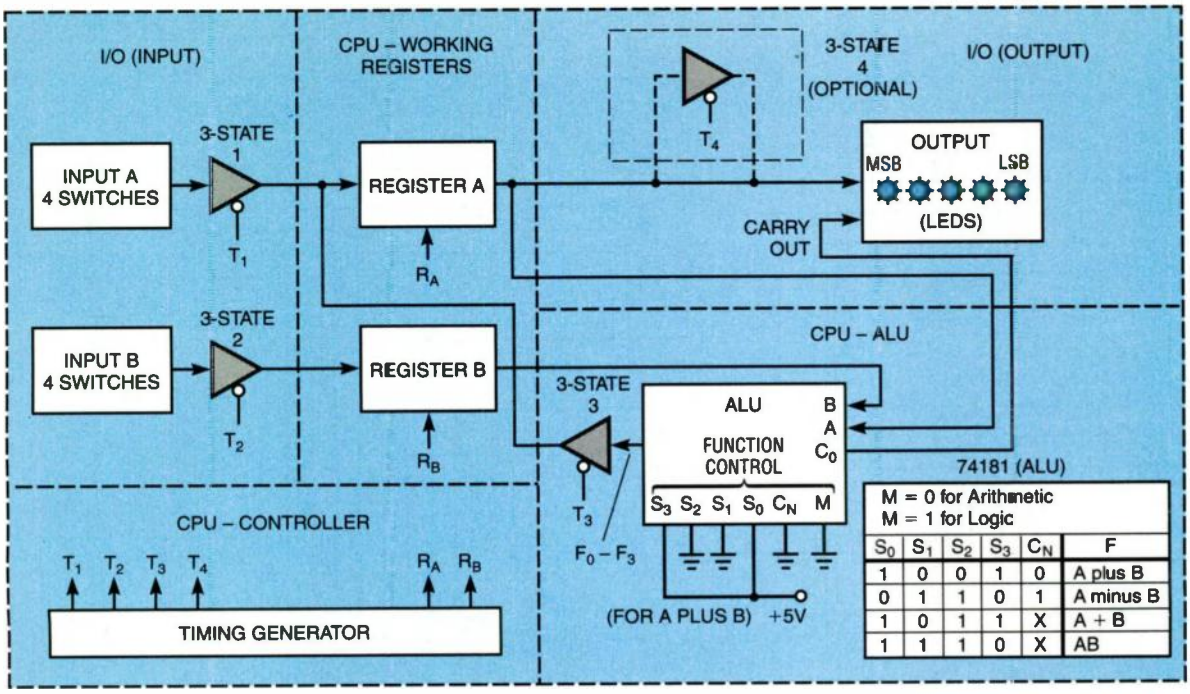
We are going to call the digital electronic system in *Figure 8-5* a computer. It has I/O and it has a CPU, but it has no memory, so it truly is not a computer. We have a specific purpose for using it; we want to demonstrate how a computer works. It is not a complicated machine at all. In fact, you could very easily build it as an experiment. It would provide hands-on experience in how a simple digital system works.

This simple digital system is called a "hard-wired" digital system. It only performs the function that is wired into it by the interconnection of its components. *If the function is to change, the interconnection must change.* For example, look at *Figure 8-5*. The ALU is a type 74181 integrated circuit. When M=0, the ALU is set to arithmetic functions, and the function performed is determined by the logic levels on S_0, S_1, S_2, S_3, and C_N. Since $S_0 = 1$, $S_1 = 0$, $S_2 = 0$, $S_3 = 1$, $C_N = 0$, and $M = 0$, the ALU is set to add A plus B. By changing the $S_0 - S_3$, C_N and M logic levels, the ALU will subtract B from A, do a logic OR, or do a logic AND. The physical interconnections must change to change the function. With the interconnection shown, the "computer" of *Figure 8-5* is set to add two 4-bit binary numbers (A and B), and display the sum.

How It Works

A block diagram is shown in *Figure 8-5*, and the timing diagram of the output signals from the controller is shown in *Figure 8-6*. A full schematic diagram to show how the components are wired is shown in *Figure 8-7*. You may want to refer to it for more detail as the explanation continues. The two 4-bit binary numbers, called operands, are input to the computer by using the 4-bit switches, INPUT A and INPUT B, to set their binary values. The logic levels are: HIGH (+5V) = 1 and LOW (0V) = 0. After the input switches are set (assume A = 0011 and B = 0100), the timing sequence from the controller is started by opening the "T" switch momentarily. When the trigger "T" goes momentarily HIGH, the controller begins its sequence of signal outputs as shown in *Figure 8-6*. We will find, as we stated before about controllers, that the timing and sequence of the controller signals is what keeps the computer's functions synchronized.

After the "T" input goes momentarily HIGH, the very next positive-going clock pulse produces a LOW at controller output T_1 which activates 3-state #1, putting the first number on the inputs of register A. Remember, the 3-state driver output remains in a high-impedance state until activated. The second positive-going clock pulse



S_0	S_1	S_2	S_3	C_N	F
1	0	0	1	0	A plus B
0	1	1	0	1	A minus B
1	0	1	1	X	A + B
1	1	1	0	X	AB

Figure 8-5. Block diagram of simple 4-bit "Computer" (not a complete computer because it has no memory).

causes controller output R_A to go HIGH. R_A activates register A and loads the 4-bit binary number (0011) into register A. Immediately, the output of register A appears at input A of the ALU and also at the I/O output, which displays 0011 on the LEDs. The LED is ON for a 1 register output and is OFF for a 0 output. The next (third) positive-going clock pulse causes these controller outputs to change: T_1 goes HIGH, R_A and T_2 go LOW. This action turns OFF 3-state #1, deactivates register A, and turns ON 3-state #2, which puts the second 4-bit number (0100) on the input of register B.

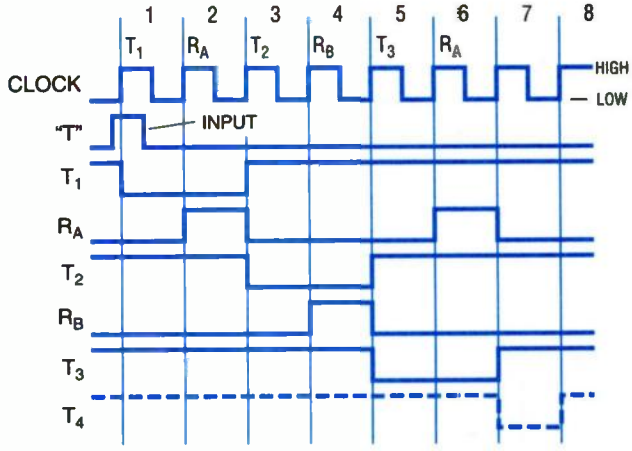
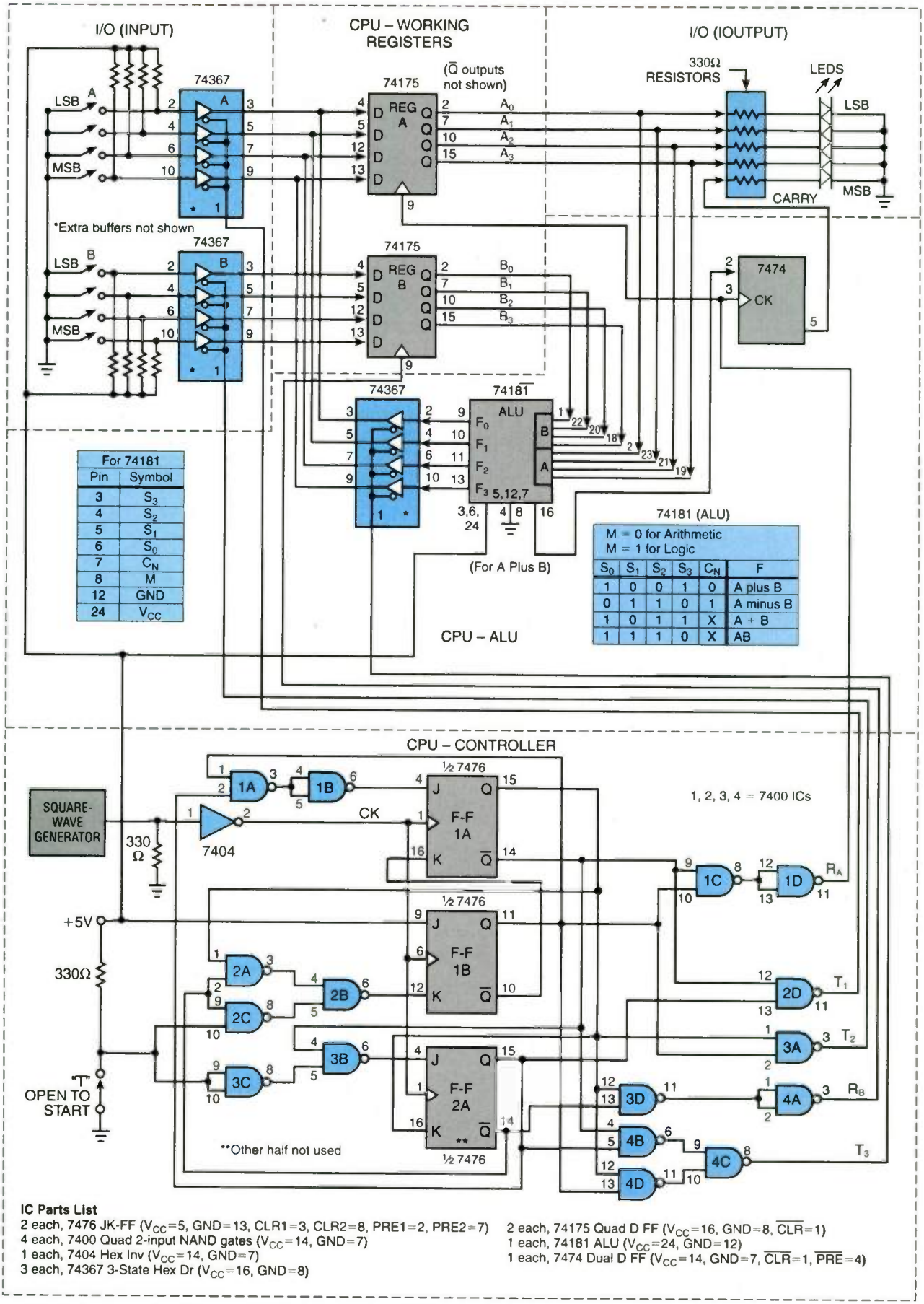


Figure 8-6. Timing diagram for simple 4-bit "Computer."

At the next (fourth) positive-going clock pulse, R_B goes HIGH and loads 0100 into register B. The output of register B appears immediately at the input B of the ALU, and since register A is inputting A, immediately the sum of operands A and B (0111) appears at the output (F0-F3) of the ALU. At the fifth positive-going clock pulse, T_2 goes HIGH, R_B goes LOW, and T_3 goes LOW. Register B is deactivated so nothing disturbs its contents, 3-state #2 is deactivated, and 3-state #3 is activated. This puts the output of the ALU (the sum) on the input of register A. The A input is disconnected because T_1 is HIGH so 3-state #1 is not active. The sixth positive-going clock pulse causes R_A to go HIGH, activating register A again and loading the sum of the two operands into register A. Immediately, the output of register A (the sum) is displayed on the I/O output LEDs. At this point, the computer has really completed its task; however, one more clock pulse is required to set T_3 HIGH and R_A LOW. An optional 3-state #4 can be added if desired. It is activated by timing signal T_4 which goes LOW at the seventh clock pulse. It activates the I/O output only



IC Parts List

2 each, 7476 JK-FF (V_{CC}=5, GND=13, CLR1=3, CLR2=8, PRE1=2, PRE2=7)
4 each, 7400 Quad 2-input NAND gates (V_{CC}=14, GND=7)
1 each, 7404 Hex Inv (V_{CC}=14, GND=7)
3 each, 74367 3-State Hex Dr (V_{CC}=16, GND=8)
2 each, 74175 Quad D FF (V_{CC}=16, GND=8, $\overline{CLR}$=1)
1 each, 74181 ALU (V_{CC}=24, GND=12)
1 each, 7474 Dual D FF (V_{CC}=14, GND=7, $\overline{CLR}$=1, $\overline{PRE}$=4)

Figure 8-7. Schematic diagram of the hard-wired 4-bit computer system.

when the sum is at its input, and so the sum is all that is displayed on the LEDs. T_4 goes HIGH (back to initial conditions) on the eighth clock pulse.

The computer can be built with discrete parts as shown in *Figure 8-7,* or the logic circuits may be implemented with a PLD. If a square-wave generator is not available, the clock can be single stepped with a debounced push-button switch. Or a monostable multivibrator, such as a 555 timer, could be used to generate the clock pulses. Obviously, the I/O, the registers, and the ALU could be changed to handle 8-, 16-, or 32-bit numbers, but the operation of this simple computer would remain the same.

Example 3. Determining the Controller Logic for T_1

Referring to the schematic of *Figure 8-7,* what must the output $\overline{Q}$ of FF1A and the output Q of FF2A be in order for the NAND gate 2D to output a LOW signal for T_1?

Since a NAND gate outputs a 0 or LOW when both A AND B are 1s or HIGH, output Q of FF1A must be a 0 or LOW so $\overline{Q}$ can be a 1, and output Q of FF2A must be a 1. It is a good exercise to determine what sets FF1A and FF2A.

Programmed Digital Systems

Digital system designers very quickly realized that hard-wired designs limited the flexibility of digital systems. As a result, they came up with digital systems that perform functions and do tasks based on a program. A *program* is a sequence of operations that the digital system follows to perform a task. Each step in the sequence is determined by a digital code called an *instruction.* Thus, from the instruction digital code, the digital system knows which operation to perform, and it follows these instructions one after the other, performing all the operations and completing its programmed task.

Someone writes a program to accomplish a task by writing instruction codes, organized in sequence so that one instruction follows another in a definite order, usually at the next higher address than the address of the one before it, and stores the program in RAM. Thus, to direct the digital system to run the program, the address of the first instruction in RAM is located, and the RAM addresses stepped through until the program is complete. Now, to change the digital system's task, all that needs to be done is to change the program, *not any of the hard wiring.*

Programmed System Information Flow

Let's revisit *Figure 8-1* and *Figure 8-2* and look at the flow of information within a programmed digital system. Stored in memory are programs of instructions in sequence and digital codes representing data that pass between CPU and memory on the data bus. Address codes to locate memory storage locations and I/O flow on the address bus from CPU to I/O or memory. Timing and control signals pass over the control bus from the controller in the CPU to memory or I/O. Note in *Figure 8-1,* the controller has a program counter. The program counter is a counting register (it can count up or count down) that *holds the memory address of the next instruction that the digital system will execute.* Note also there is an instruction register. We have mentioned it before. The instruction register *holds the digital code that tells the controller what operation the CPU is presently executing.* The address register is loaded with digital codes that are placed on the address bus.

Instruction Fetch Cycle

To help us understand programmed digital systems, let's look at how the instructions of a program are located. *Figure 8-8a* shows how the memory has the instructions of

a program stored in particular areas of memory separate from the area where data is stored. Suppose a program starts at location 00 (this is a digital code byte where each 4-bit segment is a BCD number.) As shown in *Figure 8-8b,* to start the program, the CPU loads the 00 byte into the program counter. The program counter is the address register for an instruction. The digital code it contains is the address in memory where the first instruction is located. With the program counter address on the address bus, the controller tells the CPU to read memory and load the digital code from memory into the instruction register. Timing and control signals in sequence continue to decode the instruction and route the control signals so the operation directed by the instruction is executed. The complete process is called an instruction fetch cycle. Note there are quite a few steps in an *instruction fetch cycle.* At the end of the instruction fetch cycle, the program counter is incremented by one to start the next instruction fetch cycle. The total sequence is referred to as "fetching an instruction."

Data Cycle

If the present instruction is to fetch data, then the address for data is loaded by the CPU into the address register, and a command given to read memory. As shown in *Figure 8-8c,* the contents of the memory address (could be I/O or RAM) are loaded into one of the working registers. If the task is to add two numbers, several of these data cycles might be executed in order to have data in two working registers, which then are sent to the ALU to produce a sum of the binary numbers in the registers.

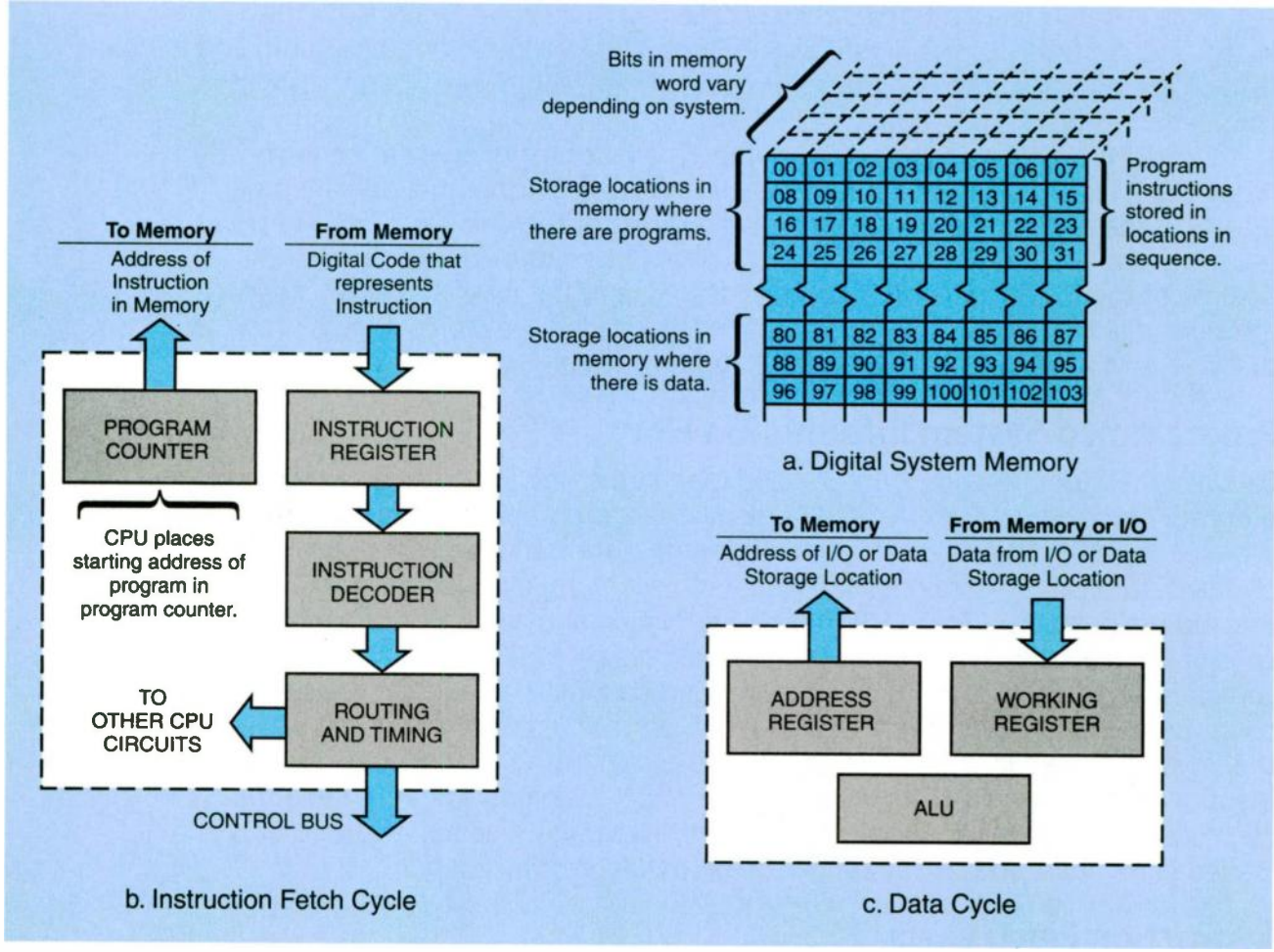


Figure 8-8. Details of how a programmed digital system locates an instruction or data.

Jump Instruction

If the instruction to be executed is not the instruction in the next higher address in memory, then the CPU is instructed to load a new address into the program counter. Such a sequence is called a *jump instruction.* With jump instructions, a program can use small segments of a program (called subroutines) repeatedly in different programs without writing the specific steps over again in each program.

Program for a 4-Bit Computer

To demonstrate how the same task as was done previously with the hard-wired 4-bit computer can be executed using a programmed digital system, we will write a program to add binary number B to binary number A and produce the sum. Then the program will continue and load a new binary number B and subtract it from the first sum.

Program Details

From *Figure 8-8a,* the area of memory where programs are to be stored starts with address 00. Remember, the address is a byte divided into two 4-bit BCD numbers. As an example, address 16 would be a binary code of 0001 0110 (in BCD, 0001 is one, and 0110 is six.) The program doesn't have to start at 00, it could start at 11, or 20, or 29, but for simplicity we will start it at 00. The program uses a similar digital system to that shown in *Figure 8-5* except the controller is like that shown in *Figure 8-1* with its program counter, instruction register, address register, decoder and routing circuitry. The timing generator is designed to provide the correct timing signals to the total system at the correct time as we demonstrated for the hard-wired system with the timing signals of *Figure 8-6.*

Detailed in *Figure 8-9a* are the program instructions located at RAM storage locations in sequence starting at 00. The instructions are numbered and an explanation of each instruction is given, but in the actual digital system, the instruction is a digital code that instructs the CPU to do what the explanation says. In fact, the CPU action in *Figure 8-9b* explains further the instruction decoded by the controller.

Let's review the basic action to execute this program: The CPU loads the starting address for the first instruction in the program counter and initiates an instruction

MEMORY

Instructions

Address in BCD	No.	Instruction Explanation
00	1	Clear all registers to 0, increment PC*
01	2	Load I/O A address, increment PC
02	3	Load Register A, increment PC
03	4	Load I/O B address, increment PC
04	5	Load Register B, increment PC
05	6	Add Reg. A and Reg. B, increment PC
06	7	Load Reg. A with sum, increment PC
07	8	Clear Reg. B, increment PC
08	9	Load address 83, increment PC
09	10	Load Reg. B, increment PC
10	11	Subtract Reg. B from Reg. A, increment PC
11	12	Load Reg. A with result, increment PC
12	13	Jump, load 00 in program counter
00	14	Clears all registers to 0 and stop

a. Instructions In Memory

CPU ACTION

Clears all registers to zero
Puts 80 in Address Register
Addresses and reads memory, loads contents in Reg. A
Puts 81 in Address Register
Addresses and reads memory, loads contents in Reg. B
Enables ALU, places Reg. A and B on inputs and adds A and B
Loads sum in Reg. A, sets carry flag to 1 if carry, disables ALU
Clears Reg. B to zero
Puts 83 in address register
Addresses and reads memory, loads contents in Reg. B
Enables ALU, places A and B on inputs and subtracts B from A
Loads Reg. A with result, sets borrow flag to 1 if borrow, disables ALU
Puts 00 in program counter, jumps and executes
Clears all registers to zero and stops

b. CPU Action as Result of Controller Decode

DATA

Address in BCD		Contents in BCD		Explanation
8	0	0	6	Data from I/O A
8	1	0	3	Data from I/O B
8	2	0	0	Data from Program
8	3	0	4	Data from Program
8	4	0	9	Data from Program

c. Data In Memory

*Increment PC means that 1 is added to the program counter contents and a new instruction cycle initiated.

Figure 8-9. Details of a program and resulting CPU action for a programmed digital system.

fetch cycle. The CPU requests that memory read the digital code for the first instruction, brings it into the instruction register, from which it is decoded and executed by the CPU under the direction of the control and timing signal sent out by the controller. At the end of the instruction is a command "increment PC" which adds one to the program counter to provide the next instruction address and initiate a new instruction fetch cycle.

Program Counter		Instruction Register	Address Register		Register A		Register B		ALU Output	
		Code for Inst #1								
		Code for Inst #2								
		Code for Inst #3								
		Code for Inst #4								
		Code for Inst #5								
		Code for Inst #6								
		Code for Inst #7								
		Code for Inst #8								
		Code for Inst #9								
		Code for Inst #10								
		Code for Inst #11								
		Code for Inst #12								
		Code for Inst #13								
		Code for Inst #1								

You Be the CPU

Follow through the instructions in *Figure 8-9a* as if you are the CPU, and assure yourself that the programmed digital system executes the same task of adding two 4-bit binary numbers and produces their sum. And that it continues and subtracts a new binary number from the first sum and arrives at a result. Place the contents of the program counter, registers, and ALU output in the chart as the instructions of the program are executed. The solution is on the questions and problems page at the end of the chapter. Note in the program and digital system of *Figure 8-9,* we are really dealing with bytes (8 bits), and therefore, could have handled much larger numbers than the 4-bit hard-wired system of *Figure 8-7.*

Summary

When we state that an instruction is a digital code and is loaded into the instruction register and decoded, or that a digital code address is sent to memory and must be decoded to find the specific location, you must understand that the decoding function is implemented the way that it was described earlier in the book. The same applies for the loading of registers, counting, reading and writing memory, and all types of logic. If you don't remember the use of these terms, you should go back into earlier parts of the book and review how the functions are accomplished with digital circuits.

We now have given the basic concepts of hard-wired and programmed digital systems. Because of the flexibility of changing what a digital system can do by changing its programs, there is an overwhelming variety of ways to do a particular task. Also, be aware that an instruction is not completed in one clock cycle. Rather, it may take many clock cycles to complete an instruction. In a computer system, for example, if the clock rate is 100 MHz, the rate of executing complete instructions may be only 1/50 of that; however, that is still *one instruction in two millionths of a second!*

We have not said anything about programming languages, but there are several levels. The lowest level is called *machine language.* It is the detailed 1s and 0s that must be present in the instruction digital codes to make the decoding, logic, registers, counters, flip-flops operate as desired. If humans programmed in this language, it would be so tedious that programmers would probably go crazy. Therefore, *high-level programming languages,* such as, Basic, Fortran, Pascal, C, C++ have been developed so that humans can develop programs in a language closer to how they think and express themselves. Other programs called compilers and assemblers then convert the programs written in higher-level language to machine code for a particular digital system.

In the final chapter, we will look at how digital electronics are used in digital communication systems.

Quiz for Chapter 8

1. The controller is that portion of the CPU responsible for many functions, one of these is:
   a. switches the interconnections and flow of information to and from I/O.
   b. inputs data to the CPU.
   c. outputs data from the CPU.
   d. stores the timing signals in memory.
2. The main buses in a microcomputer are:
   a. control bus, register bus, and address bus.
   b. right bus, center bus, left bus.
   c. memory bus, data bus, logic bus.
   d. address bus, data bus, control bus.
3. A certain 8-bit microprocessor is capable of addressing 65K of memory. How many address lines does it have?
   a. 4
   b. 8
   c. 16
   d. 32
4. A group of common connectors connecting common signals to subsystems is called a
   a. register
   b. cable train
   c. three-state system
   d. bus
5. Timed operations for digital circuits are coordinated through the:
   a. ALU bus
   b. address bus
   c. data bus
   d. control bus
6. Information goes to and from the CPU on two buses but only one way on one bus, that is, only:
   a. from the CPU on the address bus.
   b. to the CPU on the address bus.
   c. from the CPU on the data bus.
   d. to the CPU on the data bus.
7. The CPU uses a method called _____ to select a particular I/O device.
   a. instructing
   b. addressing
   c. interpreting
   d. assigning
8. I/O units send data to or receive data from the CPU
   a. some in serial.
   b. some in parallel.
   c. some in serial; and some in parallel.
   d. neither serial or parallel
9. A _____ makes sure that the different instruction codes are stored in sequence in the correct portion of memory.
   a. memory manager
   b. control bus
   c. memory address interpreter
   d. program counter
10. The _____ signals on the _____ bus determine which operation is performed on the two inputs of the ALU.
    a. address, address
    b. address, data
    c. control, address
    d. control, control
11. Along with addition, the ALU in our study has capabilities to
    a. do subtraction and logic functions.
    b. multiply, divide, and take square root.
    c. subtract and divide.
    d. take square root.
12. Ignoring some of the bits in an operation is refereed to as _____ .
    a. jumping
    b. masking
    c. flagging
    d. aliasing
13. The A and B inputs to the ALU are called _____ .
    a. instructions
    b. controls
    c. operands
    d. performers
14. All of the operations of a digital system are synchronized by a_____ .
    a. clock
    b. trigger pulse
    c. address register
    d. status flag
15. A/An _____ is a sequence of operations that the digital system follows to perform a task.
    a. instruction
    b. program
    c. code
    d. control

Answers: 1a, 2d, 3c, 4d, 5d, 6a, 7b, 8c, 9d, 10d, 11a, 12b, 13c, 14a, 15b

Questions and Problems for Chapter 8

1. Name the three buses of a basic computer system.
2. What is the purpose of a bus structure?
3. What signals determine whether a memory is in a read or write cycle?
4. What signals identify a specific I/O unit or location in memory?
5. What signals are used to transfer information like numbers, letters, or special characters from one digital system to another?
6. What section is called the "nerve center" of the computer?
7. What section of the computer communicates with the outside world?
8. Describe the similarities and differences between RAM and ROM.
9. How is information that is stored in memory located ?
10. How does an ALU perform multiplication and division?
11. What is the XOR logic operation used for in the ALU?
12. What is meant by the term "masking"?
13. What is the accumulator?
14. What tells the ALU which operation to perform and when to perform it?
15. Describe the "jump" instruction.

Solution to Program of Figure 8-9

Program Counter	Instruction Register	Address Register	Register A	Register B	ALU Output
0 0	Code for Inst #1	0 0	0 0	0 0	0 0
0 1	Code for Inst #2	8 0	0 0	0 0	0 0
0 2	Code for Inst #3	8 0	0 6	0 0	0 0
0 3	Code for Inst #4	8 1	0 6	0 0	0 0
0 4	Code for Inst #5	8 1	0 6	0 3	0 0
0 5	Code for Inst #6	8 1	0 6	0 3	0 9
0 6	Code for Inst #7	8 1	0 9	0 3	0 0
0 7	Code for Inst #8	8 1	0 9	0 0	0 0
0 8	Code for Inst #9	8 3	0 9	0 0	0 0
0 9	Code for Inst #10	8 3	0 9	0 4	0 0
1 0	Code for Inst #11	8 3	0 9	0 4	0 5
1 1	Code for Inst #12	8 3	0 5	0 4	0 0
1 2	Code for Inst #13	8 3	0 5	0 4	0 0
0 0	Code for Inst #1	0 0	0 0	0 0	0 0

CHAPTER 9

Looking at Digital Communications

Introduction

This chapter is about the use of digital electronics in communication systems. Earlier in this book, we discussed communications, but in a restricted sense — the transmission of digital codes between two computers or other digital systems that are physically close to each other. We learned that we need a transmitter to send the information, a receiver to receive the information, and a transmission link (a means of interconnecting the transmitter and receiver) to transfer the information. We know that transmitting information *one-way* in the transmission link at any one time is called *simplex* operation, and transmitting information *both ways* at the same time is called *duplex* operation.

Expanding Our Horizons

In this chapter, we will discuss communication between digital communication systems that are separated by long distances. To do this, we must define these new terms and concepts: carrier, modulation, demodulation, bandwidth, and multiplexing. In addition, we will expand the definition of the transmission link.

Carrier is just what the name implies. It is a high-frequency signal upon which information is superimposed for transmission; thus, the high-frequency signal becomes the carrier of the information.

Modulation is the process used to superimpose the information on the carrier signal. The common processes change the carrier signal amplitude, phase or frequency. Modulation occurs at the sending end.

Demodulation, which occurs at the receiving end, is the process used to recover the original information from the carrier.

Bandwidth is a measure of the frequency range of a signal transmission that a communication system can handle. It is a range of frequencies (f_1 to f_2) over which the system response stays above a specified level so the transmission will be successful.

Multiplexing is a process that allows a single transmission link to carry more than one voice or data channel at the same time.

Figure 9-1 shows media that can make up a telephone system transmission link. The first portion of the transmission link from a house or business is usually a hard-wired cable to the central office. Other portions of the link may use coaxial cable, fiber optic cable, wireless microwave transmission, or wireless satellite transmission. Both voice and data can be transmitted through this communication link; however, data transmission is limited by the subsystem that has the narrowest bandwidth, which is usually the hard-wired telephone line.

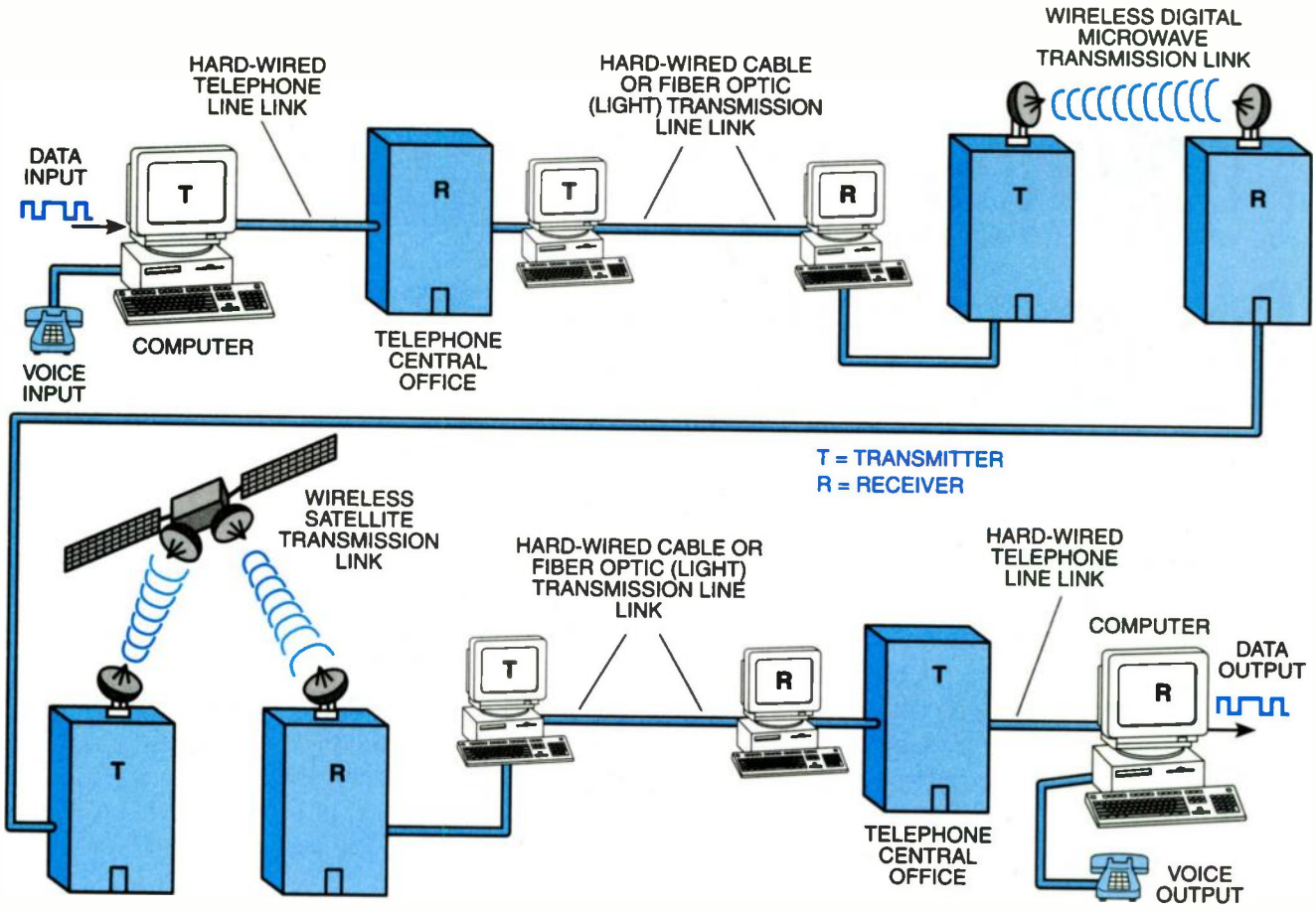


Figure 9-1. Communication system with a variety of transmission links.

Bandwidth and Multiplexing

Let's continue further using the familiar standard telephone system. When you pick up the handset, a switch inside the telephone completes a direct current circuit, called the *local loop,* between the telephone and the telephone company's central office. A battery or power supply at the central office causes current in the circuit. Your voice produces changes in sound pressure on the microphone in the handset which causes variations in the circuit current. The current variations are amplified by the central office and then transmitted over the remaining communication system until they reach the destination telephone. There, the current variations are changed back into sound pressure variations that the receiving person can hear.

The bandwidth of the standard telephone voice channel is 300 to 4000 hertz as shown in *Figure 9-2a.* The specified response is shown between f_1 = 300 Hz and f_2 = 4000 Hz. When this voice signal is modulated onto a 8140 kHz carrier, the resultant signal is shown in *Figure 9-2b.* This is shown as Channel 1 in *Figure 9-2c.* Note that the bandwidth of Channel 1 after modulation is now 8140-8144 kHz, the addition of the carrier frequency and its modulation voice channel bandwidth.

Telephone systems would not be economical if only one voice conversation were transmitted on the link beyond the telephone central office, so multiple conversations are multiplexed onto a transmission link at the same time as shown in *Figure 9-2c.* This example shows six voice channels, each with a 4-kHz bandwidth, multiplexed together. Each channel has its own carrier frequency which is separated from the next channel by 4 kHz. Note that the output transmission link must now have a 24-kHz bandwidth (6 × 4 kHz.). This method, called *frequency division multiplexing* (FDM), is used in analog systems. Later in this chapter we will see that a different, but quite similar, multiplexing technique is used to combine digital signal channels onto a common transmission link for digital communication systems.

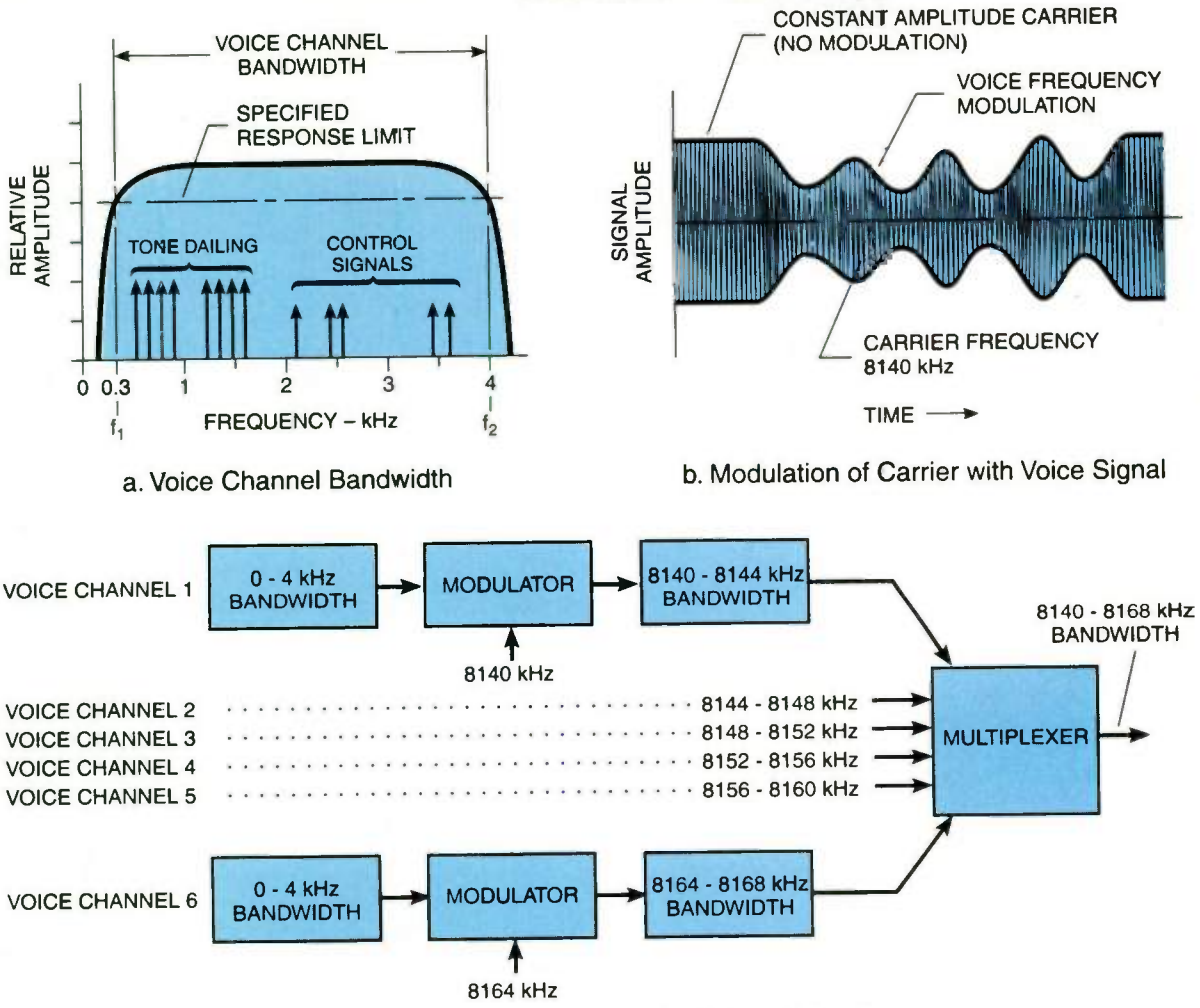


Figure 9-2. How a telephone voice channel modulates a carrier and is multiplexed so it can be combined onto a common transmission link.

Example 1. Determining Bandwidth Requirements

Determine the channel frequencies and the total bandwidth required for an analog system that multiplexes 10 channels with a channel bandwidth of 5 kHz onto carrier frequencies starting at 10,000 kHz.

Channel	Frequencies	Channel	Frequencies
1	10,000 - 10,005 kHz	6	10,025 - 10,030 kHz
2	10,005 - 10,010 kHz	7	10,030 - 10,035 kHz
3	10,010 - 10,015 kHz	8	10,035 - 10,040 kHz
4	10,015 - 10,020 kHz	9	10,040 - 10,045 kHz
5	10,020 - 10,025 kHz	10	10,045 - 10,050 kHz

Using the 50 kHz band (10 channels at 5 kHz each) the total bandwidth is 10,000 to 10,050 kHz.

Digital Systems

The use of digital electronics has advanced rapidly in the last few years because digital communication systems are more efficient than analog systems in meeting the basic goals of communication systems. These goals are:

- Transmit the most information possible
- Use the least amount of bandwidth
- Hold the number of errors to acceptable limits

Digital circuits can operate at very high frequencies — up to 200 million cycles per second — so large amounts of information can be communicated in a short period of time. Since digital circuits operate on binary signals, the signal level can vary a great deal before it causes an error. In other words, the fidelity of the signal is maintained even in the presence of some transmission noise. In addition, error detection and correction circuits can be used to maintain error-free transmission.

A major advantage of digital communication systems is the lower cost because digital circuits cost less that analog circuits. We have discussed many of the digital circuits used in communication systems; for example, encoding, decoding, data selection, registers, logic functions, line drivers, line receivers, memory, and synchronization circuits. These circuits are used in large quantities, so volume discounts help keep costs down. Some analog circuits must be used, but they are held to a minimum because they cost more.

A major disadvantage of digital communication systems is that they require more bandwidth than analog systems for a given number of *voice* channels that must be transmitted.

ADC and DAC Converters

When data is already in digital code, the digital communication system can handle it directly, using digital electronic circuits. When the source information is in analog form, an analog-to-digital (ADC) converter must be used at the input end. When the destination information must be in analog form, a digital-to-analog (DAC) converter must be used at the output end. We discussed ADC and DAC circuits in detail in Chapter 5.

Figure 9-3a summarizes the ADC function and its resulting pulse code. *Figures 9-3b and 9-3c* show the complete block diagram of a *pulse code modulation* (PCM) digital communication system. The digital code output of the ADC is the source for the pulse code modulation in the system. The code is output in parallel to a shift register and shifted out in series. The serial PCM is then input to a multiplexer and combined with other channels to be sent out over the transmission link.

A demultiplexer at the receiver sorts out the signals to individual channels. The serial PCM is converted to a parallel-bit code that is stored in a register. The PCM code is fed to a DAC which converts it back to a reproduction of the analog signal that was input to the transmitter. Note that low-pass filters are used in the transmitter and receiver channels. The one in the transmitter channel limits the pass band of the input analog signal. The one in the receiver channel helps smooth the steps in the output of the DAC so the analog output is a more faithful reproduction of the input.

Time Division Multiplexing

Earlier, we discussed FDM for analog systems. Now we'll discuss *time division multiplexing* (TDM) for digital systems as illustrated in *Figure 9-4*. Many channels can be multiplexed, but we will limit the number of channels to four for simplicity.

As the name implies, time division multiplexing is done by dividing time into predetermined intervals, called time slots. Each channel has a designated time slot, so in our example, there are four time slots. The channels are switched sequentially to their time slots by switches in the multiplexer and demultiplexer. The switches in our example are shown as mechanical switches, but in a real circuit, they are electronic data selectors. The switches are synchronized to operate together so each is selecting the same channel at the same time. The value of the data in each channel at any given time is represented by the 3-bit code in the current time slot.

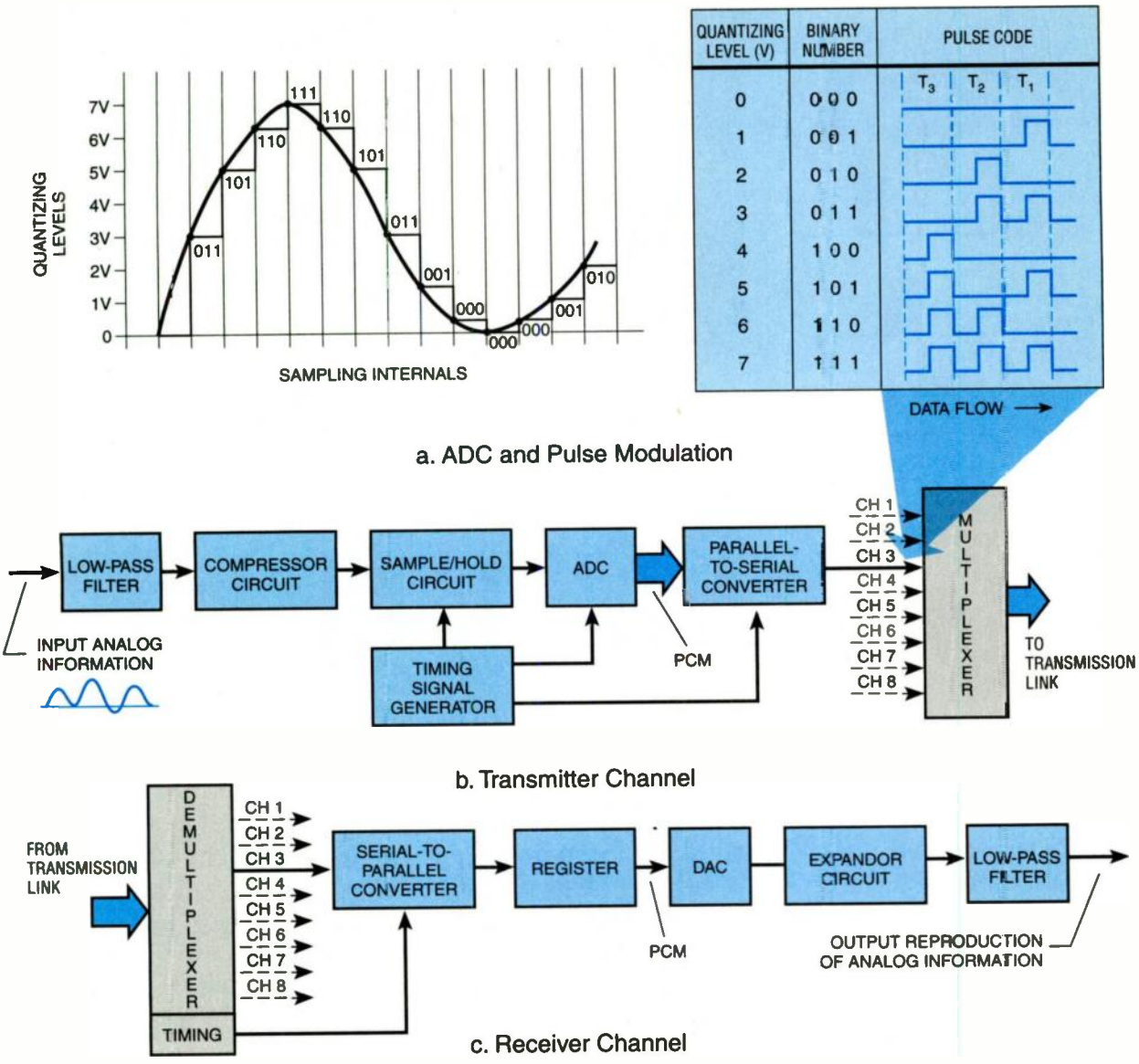


Figure 9-3. Multiplexed PCM digital communication system.

Notice in *Figure 9-4* that the switches first select Channel 1, then Channel 2, then Channel 3, then Channel 4. Thus, Channel 2's 3-bit code is placed behind Channel 1's code in time sequence, Channel 3's code behind Channel 2, and Channel 4's code behind Channel 3. When all four channels have been multiplexed, that group of signals is called a *frame*. The multiplexer adds a frame bit after the data of Channel 4 to identify the start of a new frame. The frame bit is used by the demultiplexer to synchronize the timing. The receiver sees the 3-bit code from Channel 1 first, then the 3-bit code from Channel 2, then from Channel 3, then from Channel 4. After Channel 4, the new 3-bit code for Channel 1 appears and the cycle repeats. The repetition rate of the cycle is the *multiplexing frequency.*

Bit Rate and Signal Frequency Limit

Recall from the ADC and DAC discussions that the input signal must be sampled at least twice in a cycle to accurately reproduce the signal through a DAC; therefore, the multiplexing frequency, f_M, is at least twice the signal frequency, f_1, and each multiplexed channel, Ch, must be sampled at least once in the $1/f_M$ period. The

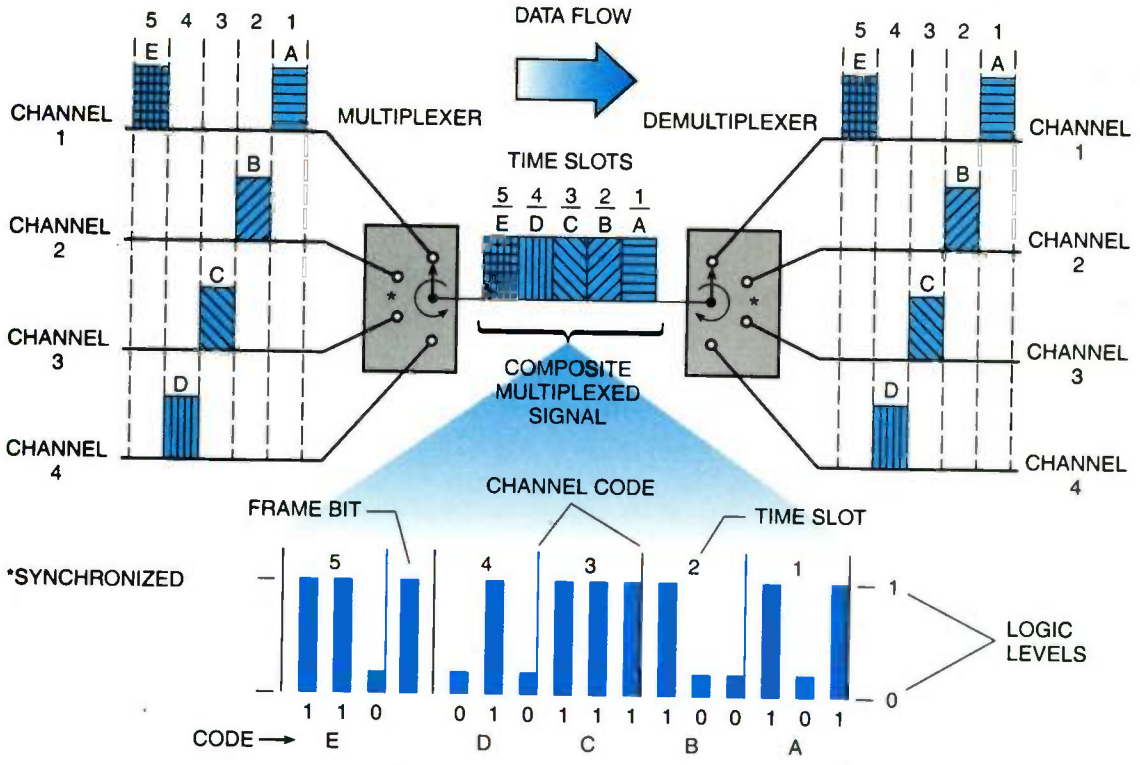


Figure 9-4. Time division multiplexing using four channels and 3-bit codes. Each group of four channels comprises a frame.

bandwidth (bit rate) depends on f_M, the number of channels (Ch) and the number of code bits (N) per channel. The PCM/TDM system of *Figure 9-3* uses pulses that are constant in amplitude, duration or width, and relative position. The code generated by the ADC represents the value of the channel signal at the time of the sampling pulse. One can easily see that if the input signal is a digital code itself, the n-bit code could be multiplexed into a time slot and handled in the same way as the signals in *Figure 9-4*. In fact, if the n-bit codes are compatible, and the timing is compatible, the digital codes for data and the digital codes for analog input signals could be mixed.

Example 2. Determining Transmission Link Bit Rate

If 32 8-bit channels with a maximum f_1 = 10 kHz are multiplexed, what is the minimum transmission link bit rate?

Remember, $f_M \geq 2\ f_1$, N = bits/Ch
Bit rate = f_M [(Ch × N) + 1 (framing bit)]
Bit rate = 20×10^3 [(32 × 8) + 1] = 5140×10^3 bits/sec = 5.14 Mb/sec

Companding

Notice in *Figure 9-3* that there is a compressor circuit in the transmitter and an expandor circuit in the receiver. These are used to accomplish a process called *companding*, a name derived from COMPressor and expANDor. Companding is used to reduce channel noise caused by digitization and to improve transmitter efficiency. *Figure 9-5* indicates the process. A common PCM system might compress a 12-bit linear code to an 8-bit code, and then expand it to 12 bits again. The 12-bit to 8-bit compression and the 8-bit to 12-bit expansion can be accomplished with digital logic circuits, but a simple and economical way is to use a ROM containing a look-up

table. The compressor would look up the 12-bit code at a ROM address to find the 8-bit compressed code to transmit. The expandor would look up the 8-bit compressed code at a ROM address to find the 12-bit expanded code.

Using modern semiconductor technology, all of the coding and decoding functions of a PCM system is placed on a single IC called a CODEC (which stands for coder/decoder). It contains the filter, the sample-and-hold circuit, and the ADC of the transmitter, and the converter, register, DAC and filter of the receiver.

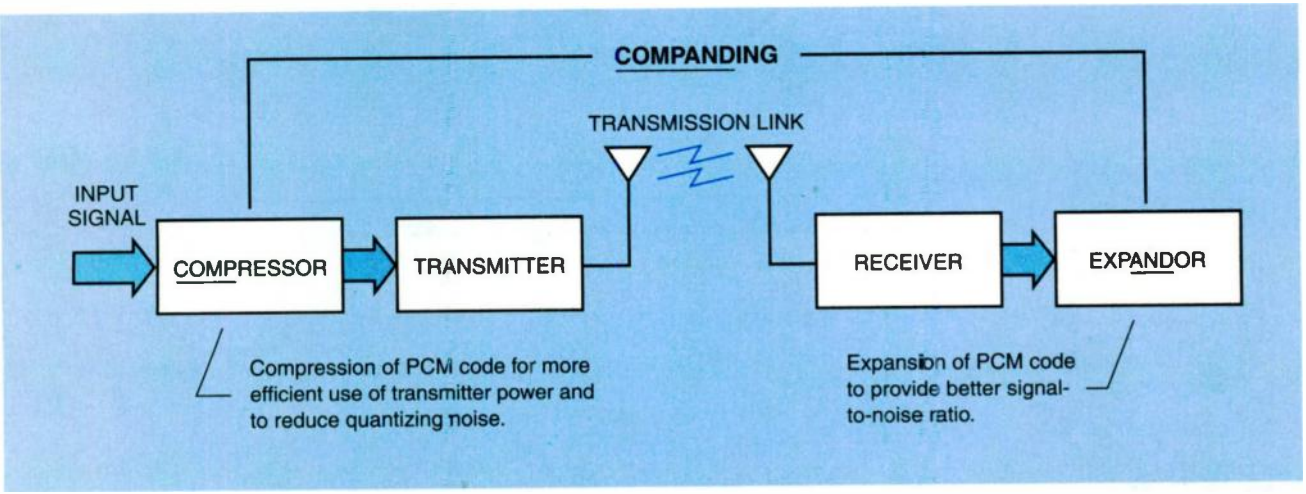


Figure 9-5. Companding means compressing the signal at the transmitter and expanding it at the receiver.

Modems

The term *modem* is derived from *MO*dulator and *DEM*odulator, which are the basic functions of the device. A modem is not a digital system in itself, but is actually an interface or adapter device that permits digital systems to communicate with each other. It has become a very important subsystem for distance communications between digital systems using ordinary telephone lines and the standard telephone system.

Figure 9-6a shows a typical use. The modem in each computer is coupled to a standard telephone line using a standard modular connector. The transmission link between the two computers may include any of the methods shown in *Figure 9-1*. If computer #1 is sending information to computer #2, the modem in computer #1 converts the computer's digital signals into analog signals that can be transmitted over a telephone line. This process is called modulation. The modem in computer #2 *demodulates* the analog signals and reconstructs the digital information sent by computer #1. If computer #2 is sending the information, the roles of the two modems are reversed.

Figure 9-6b is a block diagram of a modem. It is usually a separate printed circuit card that plugs into the computer's internal connector. One part of the circuitry is a modulator, the other part a demodulator. Amplifiers boost the signal level, filters eliminate unwanted frequencies, and interface circuits provide the proper coupling to the computer's circuits. The hybrid coil couples the modem to the telephone line to provide information flow in both directions and to isolate the telephone line from the computer.

Amplifiers and Filters

We haven't spoken much about amplifiers and filters in this book because our emphasis has been on digital circuits. Of course, amplifiers serve the same function wherever they are used; that is, they increase the voltage level, current level, or

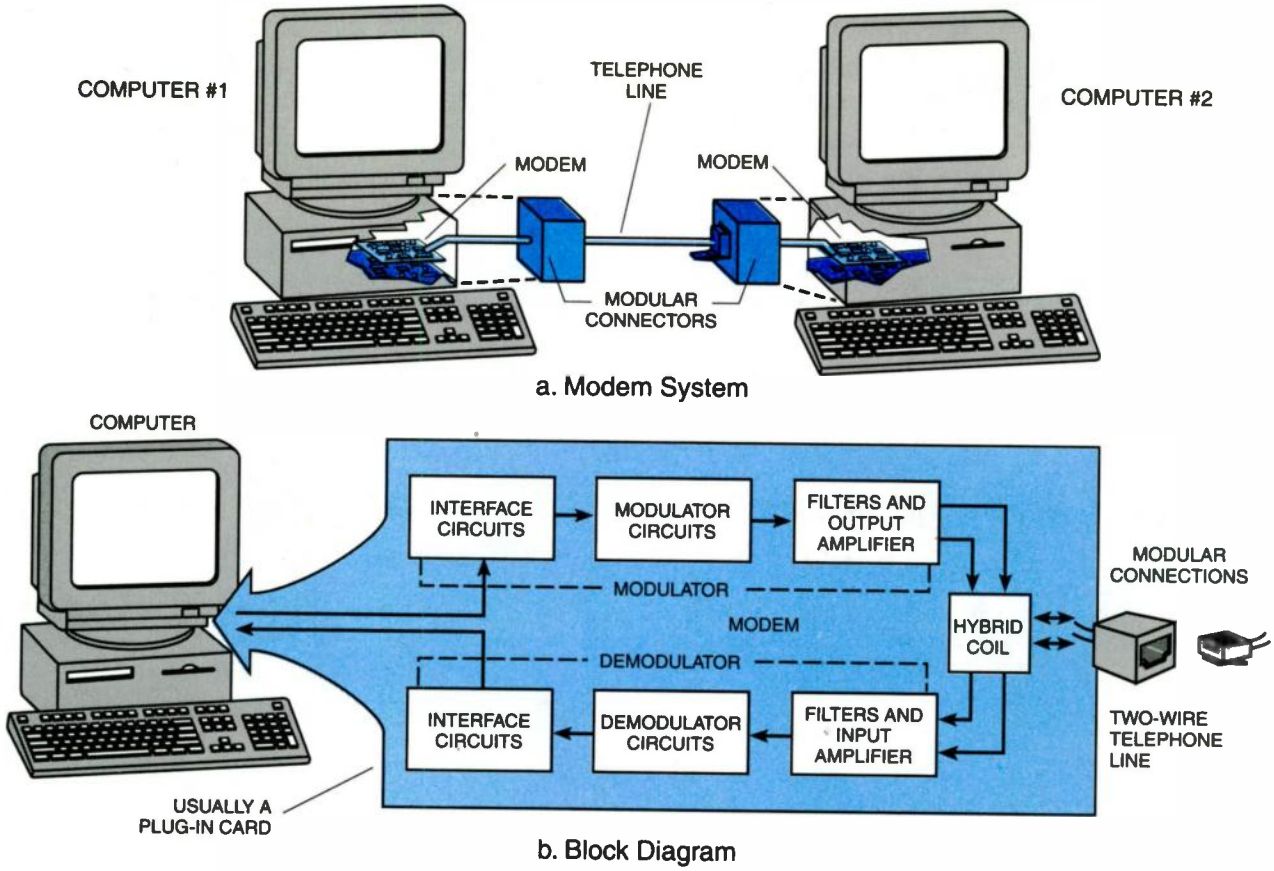


Figure 9-6. A modem is used to communicate digital information between digital systems over telephone lines.

power level of a signal. They reestablish or reinforce the signal which may be reduced or deteriorated as it moves through the transmission link.

Filters restrict the passage of certain signal frequencies and allow others to pass. A low-pass filter allows passage of low frequencies while high frequencies are restricted. (We saw low-pass filters used in *Figure 9-3.*) A high-pass filter allows passage of high frequencies while low frequencies are restricted. A band-pass filter restricts some lower frequencies and some higher frequencies while allowing passage of a band of frequencies in between.

Modem Modulation Techniques

The type of modulation used in a modem depends on the information bit rate being transmitted because standard voice-grade telephone lines are bandwidth-limited as shown in *Figure 9-2a.* When using the simplest modulation method, the bit rate and the baud rate are equal. The baud rate is the change in the signaling event that is occurring on the transmission link. To increase the bit rate and stay within the bandwidth, more bits per baud are used to change the signaling event. Changing the signaling event is the modulation.

Binary Phase-Shift Keying (BPSK)

To illustrate the type of modulation used in modems, let's look at *Figure 9-7.* The system block diagram is shown in *Figure 9-7a.* A carrier oscillator is modulated so that the output signal's phase, as compared to the carrier's original signal, is shifted by the binary input from a serial shift register. The shifting is shown in *Figure 9-7b.* If

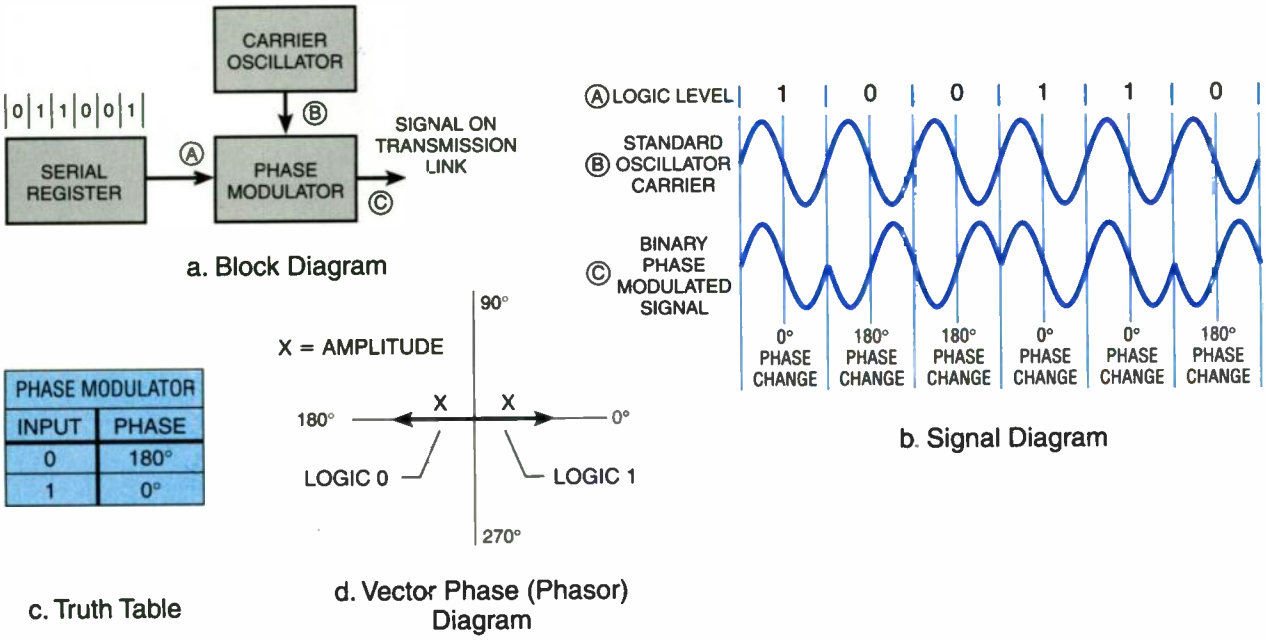


Figure 9-7. Modem Binary Phase-Shift Keying Modulation (BPSK).

the binary input logic level to the phase modulator is a 1, the output signal phase is not changed (it has 0° phase change) from the carrier signal. If the binary input logic level is a 0, the phase of the carrier signal is changed by 180°. The truth table for the phase modulator is shown in *Figure 9-7c*. The phase of the signal for the logic level can be represented by the phasor diagram in *Figure 9-7d*. The amplitude of the carrier is held constant to provide a constant signal amplitude X. The problem with BPSK is that we use only one bit for a single signal change, thus, we get no change in baud and bit rate.

QPSK

A block diagram of the quaternary (or quadrature) phase-shift keying (QPSK) modulator is shown in *Figure 9-8a*. The truth table of *Figure 9-8b* shows that it takes a combination of two bits to set the phase of the signal frequency, therefore, the bit

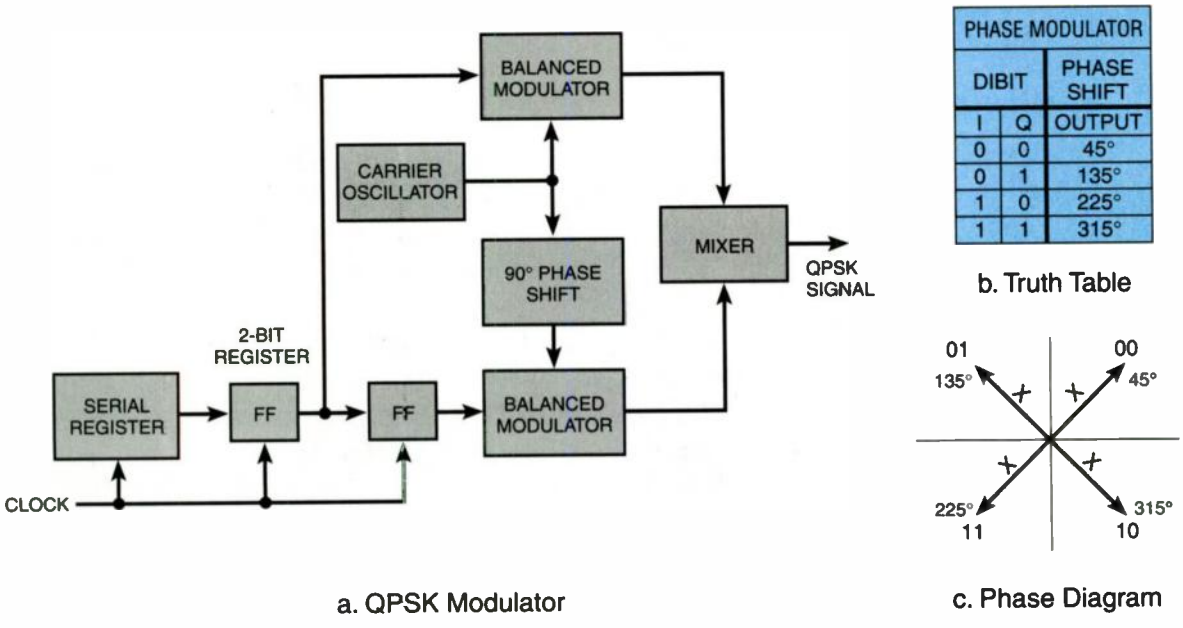


Figure 9-8. A 2-bit quadrature modulation for QPSK of modem.

rate can now be twice the baud rate. QPSK produces four phase changes in the carrier signal as shown by the phasor diagram of *Figure 9-8c.* The phasor diagram describes the four outputs from the phase modulator, and shows that the signal amplitude is a constant value X. The two bits that cause the phase shift are named the I bit (for in phase) and the Q bit (for quadrature), and the combination of the two bits is called a *dibit.*

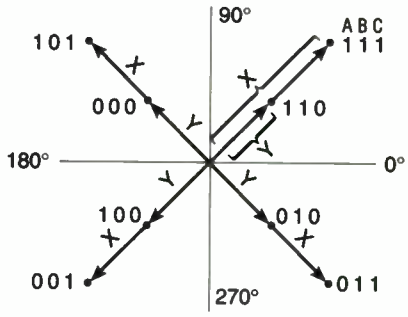


Figure 9-9. Phasor diagram for amplitude-phase keying (APK) modulation.

APK

Increasing the number of bits encoded per baud raises the bit rate further. One of the most popular modulations used in modems is called amplitude-phase keying (APK.) Its phasor diagram is shown in *Figure 9-9.* Now the signal amplitude, as well as the phase, out of the phase modulator is varied. With the four vectors of four different phases, varying the vector amplitude provides four additional signals for a total of eight. Eight signals require three bits to identify the different phase and amplitude variations.

Example 3. Truth Table for Amplitude-Phase Keying (APK)

From the phasor diagram of *Figure 9-9,* construct the truth table for the phase modulator for APK. Consider the full amplitude of the vector as X and the half amplitude as Y. The bits are A, B and C.

PHASE MODULATOR				
INPUTS			OUTPUTS	
A	B	C	AMPLITUDE	PHASE
0	0	0	Y	135°
0	0	1	X	225°
0	1	0	Y	315°
0	1	1	X	315°
1	0	0	Y	225°
1	0	1	X	135°
1	1	0	Y	45°
1	1	1	X	45°

Data Channels

When microwave, fiber optics, or satellites are used for the transmission links for digital communication, then bit rates of many millions of bits per second are possible because of the much wider bandwidth of these transmission links. Groupings of data bits into frames and packets, and the use of special identification bits for error detection and correction make sure that the digital data is communicated efficiently, accurately, and at great speed.

Summary

Digital communication systems can be very complex and we have just "scratched the surface" in this chapter. However, we have tried to explain the basic concepts of bandwidth, bit rate, multiplexing, modulation, demodulation, and transmission link. With an understanding of these, you have a foundation on which to build further studies.

This chapter concludes our discussion of basic digital electronics. We have discussed the functions required in digital systems, showed how these functions are implemented in circuits, and how they are applied to form systems to accomplish specific tasks. We reviewed integrated circuits, and repeatedly stressed that almost all digital circuits are available in ICs. They are small, low power, reliable, and relatively low cost. More circuit functions are being placed on each chip — and the trend will continue.We hope that this book has sparked your interest in digital electronics, has increased your knowledge of digital electronics and has whetted your appetite for more. If we have succeeded in any one of the three, we have accomplished our goal.

Quiz for Chapter 9

1. A high-frequency signal upon which information is superimposed is called a:
   a. link
   b. channel
   c. bank
   d. carrier
2. A process that allows a single transmission link to carry more than one voice or data channel at the same time is:
   a. duplexing
   b. modeming
   c. companding
   d. multiplexing
3. The bandwidth of a standard telephone voice channel is
   a. 100 to 1000 Hz
   b. 300 to 4000 Hz
   c. 600 to 8000 Hz
   d. 1000 to 8000 Hz
4. An analog method of multiplexing that has been used in telephone work is
   a. space scattering multiplexing
   b. frequency shift keying multiplexing
   c. frequency division multiplexing
   d. time division multiplexing
5. Which of these is not a basic goal of communication systems?
   a. transmit the most information possible.
   b. use the least amount of bandwidth.
   c. hold the number of errors to acceptable limits.
   d. keep the number of channels to a minimum.
6. Large amounts of information can be communicated in a short period of time by
   a. operating at very high frequencies.
   b. keeping the clock speed low.
   c. reducing the bandwidth.
   d. not using party bits.
7. A circuit that must be used to send analog information over a digital communications system is a/an
   a. analog-to-digital converter at the input end.
   b. digital-to-analog converter at the input end.
   c. expandor at the input end.
   d. compressor at the output end.
8. When several channels have been multiplexed, the group of signals is called a/an
   a. channel cluster
   b. parity group
   c. frame
   d. register group
9. A _____ is used to reduce channel noise caused by digitization and to improve transmitter efficiency.
   a. filter
   b. compandor
   c. dynamic microphone
   d. crystal microphone
10. A single IC that contains the transmit and receive ADC and DAC is called a
    a. modem
    b. compandor
    c. modulator
    d. CODEC
11. The _____ couples the modem to the telephone line to provide information flow in both directions.
    a. hybrid coil
    b. amplifier
    c. compandor
    d. DAC
12. The change in the signaling event that is occuring on a transmission link is called a
    a. carrier
    b. sideband
    c. baud
    d. packet
13. The bit rate of transmission can be twice the baud rate in
    a. FSK
    b. QPSK
    c. BPSK
    d. AM
14. One of the most popular modulation types used in modems is
    a. AM
    b. FM
    c. AKK
    d. APK
15. Bit rates of many millions per second are possible using
    a. copper cable pairs
    b. fiber optic cable
    c. balanced copper pairs
    d. low enough carrier frequencies

Answers:
1d, 2d, 3b, 4c, 5d, 6a, 7a, 8c, 9b, 10d, 11a, 12c, 13b, 14d, 15b

Questions and Problems for Chapter 9

1. What is the process called that is used to superimpose information on a carrier?
2. What is the process called that is used to send more than one channel over a single transmission link?
3. What is the bandwidth of the standard telephone voice channel?
4. What are two methods of multiplexing?
5. How many 5-kHz voice channels can be frequency-division multiplexed in a 20-kHz transmission channel?
6. What are the basic goals of a communication system?
7. What is the major advantage of a digital communication system over an analog system?
8. When all of the voice channels have been time-division multiplexed, what is this group of channels called?
9. If 16 channels of an 8-bit code are multiplexed onto a transmission link at a multiplexing frequency of 50 kHz, what is the bit rate on the transmission link?
10. What is the process used to reduce channel noise caused by digitization and to improve transmitter efficiency?
11. What is the interface device called that permits digital systems to communicate with each other over the telephone lines?
12. Describe Binary Phase-Shift Keying (BPSK).
13. What is the modulation method called that takes a combination of two bits to set the phase of the signal carrier?
14. Describe Amplitude-Phase keying.

Appendix

Answers to Chapter Questions and Problems

Chapter 1

1. 50 mv = 0.05 volts
   0.05 volts × 50 × 1 (full volume) = 2.5 volts
   $P = E^2/R$ $P = (2.5V)^2/8\Omega = 0.78$ watts
   0.05 volts × 50 × 1/2 (half volume) = 1.25 volts
   $P = (1.25V)^2/8\Omega = 0.195$ watts
   2.5V × 1/10 = 0.25V
   $P = (0.25V)^2/8\Omega = 0.0078$ watts or 7.8 mW
2. $I = E/R_t$ $R_t = (10/4 + 4)\Omega = 6.5\Omega$
   a) $I = 6V/6.5\Omega = 0.923$ amps or 923 mA
   b) Faster
3. 25,400 ÷ 3600 = 7.05555
   0.05555 × 3600 = 200 counts
   200 counts/60 counts/min = 3.33 min
   0.33 min × 60 sec/min = 20 sec
   Time = 7:03:20 PM
4. a) 3600 sec/60 sec/min = 60 min = 1 hr, thus 1:00 PM
   b) 46,800 sec/60 sec/min = 780 min/60 min/hr = 13 hrs, thus 1:00 AM
5. a) yes
   b) sometimes, but not always (could be octal, trinary)
6. 3572 = (3 × 1000) + (5 × 100) + (7 × 10) + (2 × 1)
7. $100101_2 = (1 \times 2^5) + (0 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$
   [32 + 0 + 0 + 4 + 0 + 1] = 37
8. 63/2 = 31 remainder of 1
   31/2 = 15 remainder of 1
   15/2 = 7 remainder of 1
   7/2 = 3 remainder of 1
   3/2 = 1 remainder of 1
   1/2 = 0 remainder of 1
   $63_{10} = 111111_2$
9. a) $5C_{16} = 0101\ 1100$
   b) $5C_{16} = (5 \times 16^1) + (12 \times 16^0) = 92$
   Check: $01011100 = (0 \times 2^7) + (1 \times 2^6) + (0 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) =$
   64 + 16 + 8 + 4 = 92
10. a) 47/16 = 2 with a remainder of 15
    15/16 = 0 with a remainder of 15 or F_{16}
    so $47 = 2F_{16}$
    b) $2F_{16} = 0010\ 1111_2$
    Check: $(1 \times 2^5) + (1 + 2^3) + (1 + 2^2) + (1 + 2^1) + (1 + 2^0) = 32 + 8 + 4 + 2 + 1 = 47$
11.

	ASCII		HEX
D	100	0100	44
I	100	1001	49
G	100	**0111**	47
I	100	1001	49
T	**101**	0100	54
A	100	0001	41
L	100	1100	4C
SP	**010**	0000	20
E	100	**0101**	45
L	100	1100	4C
E	100	**0101**	45
C	100	0011	43
T	**101**	0100	54
R	**101**	0010	52
O	100	**1111**	4F
N	100	**1110**	4E
I	100	1001	49
C	100	0011	43
S	**101**	0011	53

12. ANALOG VOLTAGE

Chapter 2

1. $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$ so 6 bits are required to represent 36 different objects with 28 codes unused.
2. Unique marker bits called "start" and "stop" bits.
3. Serial transmission – only single line required. Parallel transmission - much faster.
4. Encoder – converts decimal number to binary code.
   Digital processing unit – performs the arithmetic operations.
   Decoder – converts output binary digital code to 7-segment code for digital display.
5. 8-4-2-1 outputs 1010, see encoding table.
6. a,e,g,c,d – making the decimal number 5.
7. $S_0 = 1$, $S_1 = 0$, output $D_1 = 1$
8. See Figure 2-12.
9. See Figure 2-13a.
10. $10001100_2 = 128 + 8 + 4 = 140$
11. $10011010_2 = 128 + 16 + 8 + 2 = 154$
    154/2 = 77 = 0
    77/2 = 38 = 1
    38/2 = 19 = 0
    19/2 = 9 = 1
    9/2 = 4 = 1
    4/2 = 2 = 0
    2/2 = 1 = 0
    1/2 = 0 = 1
12.

10001011	139
01001101	77
11011000	216