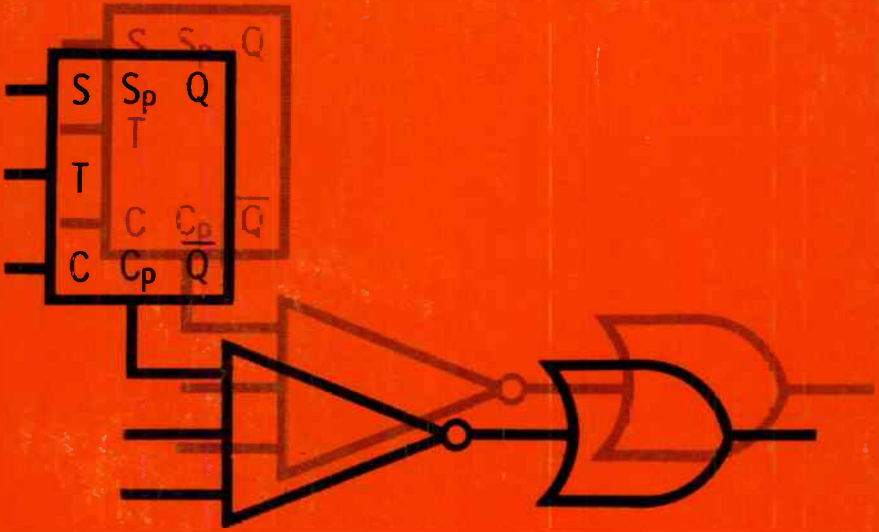
PUBLICATION 21081

how to use

INTEGRATED CIRCUIT LOGIC ELEMENTS

by Jack W. Streater



How to Use

INTEGRATED-CIRCUIT LOGIC ELEMENTS

by

Jack W. Streater

HOWARD W. SAMS & CO., INC.
THE BOBBS-MERRILL CO., INC.
INDIANAPOLIS · KANSAS CITY · NEW YORK

SECOND EDITION

FIRST PRINTING—1974

International Standard Book Number: 0-672-21081-9
Library of Congress Catalog Card Number: 70-99803

Preface

There have been two revolutions in electronics in the past twenty years. The transistor provided the first; the integrated circuit provided the second.

Ten years ago, it was clear that most electronic engineers and technicians were fast approaching technical obsolescence unless they had become familiar with the operation of transistors and transistor circuitry. Today, integrated circuits present another challenge. The integrated-circuit revolution which began several years ago is still gathering momentum, and nearly every branch of electronics has seen or will see radical changes before this revolution is over.

Integrated circuits can be classified broadly as linear or digital devices. But don't be mislead. The domain of digital circuits is widening; it now intrudes into what once was the domain only of linear circuits. The integrated circuit has so reduced the cost and size of digital functions that even engineers and technicians who have in the past worked exclusively with linear circuits(amplifiers, filters, etc.) must now frequently consider whether a job can be done better by digital techniques. Thus, not only must these men learn what they can do with the new linear integrated circuits, but they must also keep abreast of what they might do with digital circuits. For many, this will require tackling a branch of electronics previously belonging to the computer technologists—logic systems.

Electrical and electronic service technicians are another group who will be affected by the integrated-circuit revolution. Logic devices will soon be found in appliances, automobiles, and probably in television and hi-fi, too, as well as in many other places at home, in commerce, and in industry. Although integrated circuits hold promise of higher reliability than electronics has ever known, sheer volume of usage will create a demand for a large number of technicians who can service logic circuits containing these devices. Forward-looking service technicians should prepare themselves to meet this demand.

This book will give such people a broad general introduction to integrated-circuit logic technology. The necessary mathematics is developed first. This mathematics, Boolean algebra, does not require knowledge of higher mathematics to understand it. The gate circuit and combinations of gates are discussed. Flip-flops are also covered. Then logic families are introduced and are compared. Finally, the use of off-the-shelf logic elements is considered. A glossary is given in Appendix 2.

The explanations of circuit operation in this book are based on the assumption that current is in the same direction as the flow of electrons—from negative to positive. This is in keeping with what is being taught today in many technical schools and the armed services. Engineers and others who have learned to think in terms of "conventional current," from positive to negative, should have no difficulty in adapting to this newer assumption. The explanations can readily be restated by experienced persons in their own terms, using the conventional-current direction.

If you are an electronic hobbyist or experimenter, you will also find this book interesting and understandable. Fascinating logic circuits can now be assembled quickly and at very modest cost. The day may not be far off when home computers are almost as common as TV sets, or when building a home computer is a reasonable project for the experimenter.

JACK W. STREATER

Contents

Introduction

This is a practical book. Its purpose is to help you use devices you can readily obtain today from nearly any electronic distributor. These same devices are already being used by the millions in computers, machine control systems, process control systems, data transmission systems for warehousing and transportation, and in a myriad other ways, ranging from the simplest logic circuits to elaborate controls in space vehicles and aircraft.

There is a certain minimum theoretical background without which you cannot take full advantage of the potential of integrated-circuit logic elements. If you are already familiar with the binary number system and the elements of Boolean algebra, use the first two chapters for review or skip them entirely. Otherwise, study these chapters before going on.

To get the most out of this book you should have at least a basic understanding of elementary electronics and simple transistor circuits. If you do not have this background, obtain one of the many books available on these subjects and read it along with this book.

The mathematics presented here includes an introduction to binary arithmetic and Boolean algebra. If these subjects are new to you, do not be afraid of them; most certainly do not be afraid of the names. Neither is really very difficult to understand. You do not have to delve deeply into either to gain sufficient background for understanding digital logic devices and simpler systems. The mathematics included here was carefully selected; it has been kept to a minimum for the benefit of those technicians, servicemen, and experimenters who might otherwise be frightened away by a more thorough presentation. Engineers who enjoy the mathematics can find numerous books treating Boolean algebra in more detail.

In learning a new subject, you must learn a new vocabulary. In fact, it is often the vocabulary that can hinder you if the study material is derived from trade magazines, technical papers (including manufacturers' application notes), and device data and specifications. For this reason, your vocabulary development is an important secondary objective of this book. As an additional aid, a glossary of terms frequently encountered in technical literature is included as an appendix.

You need a foundation upon which to build further knowledge. But all learning is not step-by-step, despite what some authors would lead you to believe. Much learning is like going around in a circle. The early chapters are not completely meaningful until you have read the later chapters, and you cannot understand the later chapters until you have finished the early ones. So a good way to learn a new subject is to move along, not spending too much time where you have difficulty, and then when you are finished go back and start again. The next time around new concepts are clarified and the overall picture comes more into focus while you gain a broader perspective.

If you possibly can, experiment with actual integrated-circuit devices. Buy an assortment and see them work. Build some of the combinations shown in Chapters 4 and 5 and compare their operation with the Boolean equations used in their design and analysis. Measure the voltages and observe their ranges and their values under different conditions. Chapter 7 contains suggestions for "breadboarding" logic circuits to help you along.

When you study, try restating the ideas in your own words. Do not be satisfied to say, "I understand it but I cannot explain it." To be able to state something is the only proof that you know it and understand it. Learning fast and well requires active participation on your part. Ask yourself questions and try to answer them, but do not get bogged down. If you cannot immediately answer a question, write it down and perhaps the answer will be found later. Be discontent until you know the answers but be patient enough to keep moving forward, confident that the answers will come.

Chapter 1

Binary Numbers

In the binary number system all numbers are formed from only two symbols: 0 (zero) and 1 (one). Any two distinct symbols could be used, but 0 and 1 are employed almost universally. Electronic digital computers use binary numbers because electrical switches and gates, the basic elements in these systems, are two-state (on-off) devices.

Counting Numbers

Counting numbers in any system begin with zero and increase in a definite *order*. In our familiar decimal system we count:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, . . .

After each of the ten decimal symbols has been used alone, larger numbers are formed by combining the symbols in a specific orderly pattern. In these larger numbers the meaning of each decimal digit depends on its place in the sequence. For example, in the number 37,357 the first 3 does not mean the same as the second one, nor do both sevens stand for the same amount. The symbols 3 and 7 are called *decimal digits*, but their value depends on their place in the sequence. The total value of the group of decimal digits is the sum of each individual digit value. In the preceding example the total value is

30,000 + 7000 + 300 + 50 + 7

The 7 at the far right is called the *least significant digit*, while the left-most 3 is called the *most significant digit*.

In binary numbers a similar structure is used. Here there are only two digits: 0 and 1. Thus,

Decimal Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Binary Digits: 0, 1

Sometimes the term *bit* is used for binary digit (from the term *bi*nary digi*t*).

Larger binary numbers require a sequence of 0's and 1's. Beginning with zero, here are some of the lower binary counting numbers in order.

Binary Counting Number	*Equiv. Decimal Counting Number*
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8

Notice that the sequence of binary numbers can be generated by counting in our familiar decimal system but by skipping all numbers containing symbols other than 0 and 1.

Like decimal numbers, binary whole numbers have an order; that is, given any binary whole number, there is a next larger and a next smaller one, except for zero.

In both the decimal and binary number systems the place in the sequence determines the value of each symbol. In the decimal system the place value of an integer, counting from the far right, increases in powers of tens:

most right-hand digit	value of digit
2nd from right	10 × value of digit
3rd from right	100 × value of digit
4th from right	1000 × value of digit

In the binary system:

most right-hand digit	value of digit
2nd from right	2 × value of digit
3rd from right	4 × value of digit
4th from right	8 × value of digit
5th from right	16 × value of digit

The multipliers increase from right to left by the sequence 1, 2, 4, 8, 16, 32, etc., each being twice the previous one.

Here you see that we may talk about the binary system using decimal numbers. We do this because we are so familiar with decimal numbers; we feel more at home with them and we have a better feeling for their magnitudes. If we had grown up with the binary number system, we would have less need to "translate" to decimal numbers. Then it would not seem strange to write ourselves a note to buy 1100 No. 110 screws, or a No. 10010 drill (translation: 12 No. 6 screws, a No. 18 drill).

We often want to translate back-and-forth between number systems. Of course, a large table could be used (a short one is provided in the appendix of this book). However, if no table is available, the translation is not difficult, as is shown in the following examples.

Example 1: What is the decimal equivalent of 1011010?

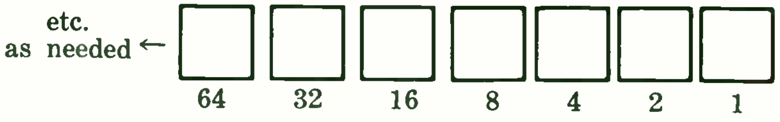
Solution: Taking the binary digits from right to left, multiply each by its decimal place value, then add all resulting terms.

```
          binary digits
         (right to left) ───┐   ┌─────decimal place value
least significant place →  0 × 1  = 0
                           1 × 2  = 2
                           0 × 4  = 0
                           1 × 8  = 8
                           1 × 16 = 16
                           0 × 32 = 0
most  significant  place → 1 × 64 = 64
                                   ──
                   decimal  sum  = 90 = decimal  equivalent  of
                                              1011010
```

Do yourself: What is the decimal equivalent of 1101010? (Answer: 106.)

Example 2: What is the binary equivalent of 39?

Solution: Fill in the boxes with 1's and 0's, so that the total value adds to 39.



The decimal value of each box is given below the box. Obviously, the 64 box and any box to the left of it must contain 0, since 39 cannot be the sum of numbers which include 64, 128, etc.

However, 32 can be one of the numbers adding to 39, so place a 1 in that box. The remaining 7 counts (39 minus 32) must be the sum of values from the remaining boxes. Now, 16 and 8 cannot be among those numbers, as each is larger than 7, so put 0's in those boxes. Of the remaining three boxes, inspection shows that 7 is made up from the sum of 4 plus 2 plus 1, so 1's must be placed in each of the boxes having these values. The final result is 39 = 0100111, or simply 100111. Left-hand zeroes may be omitted, as in the decimal system, where 039 is usually written simply as 39.

The answer may be checked by converting back to the decimal system using the method in example 1.

Do yourself: Find the binary equivalent of 73. (Answer: 1001001.)

Negative and Fractional Binary Numbers

Negative and fractional binary numbers are not normally used in logic systems, but they must be handled by computers in performing mathematical calculations.

Negative numbers are formed by prefixing a positive number with a negative sign. The sign of a number can be indicated with a 0 or 1 placed before it. The calculating system is then constructed to "recognize" the first digit as the sign, and to "know" whether 0 is minus and 1 is plus, or vice versa.

Fractions represent division. Fractions can be written in the binary system with a numerator and denominator separated by a fraction bar:

$$\frac{5}{6} \text{ (decimal) is equivalent to } \frac{101}{110} \text{ (binary)}$$

In the decimal system certain fractions can be written with a decimal point. Examples are:

$$\frac{2}{10} = .2$$

$$\frac{21}{100} = .21$$

Certain binary fractions can be written similarly, although it is more appropriate to call the dot a "binary point" instead of a "decimal point." In the binary system:

$$.1 \text{ (binary)} = \frac{1}{10} \text{ (binary)} = \frac{1}{2} \text{ (decimal)}$$

$$.01 \text{ (binary)} = \frac{1}{100} \text{ (binary)} = \frac{1}{4} \text{ (decimal)}$$

$$.11 \text{ (binary)} = \frac{11}{100} \text{ (binary)} = \frac{3}{4} \text{ (decimal)}$$

Example: What is the decimal equivalent of 101.11?
Answer: Now 101 = 5 and

$$.11 = \frac{11}{100} = \frac{3}{4}$$

Therefore, the answer is 101.11 = 5¾ or 5.75.

Binary-Coded Decimal

There is a "hybrid" or mixed binary-decimal system in common use in computers called *binary-coded decimal* or *BCD*. The difference between "pure" binary numbers, such as we have just discussed, and BCD is illustrated as follows:

Decimal Number	*"Pure" Binary Equivalent*	*BCD Equivalent*
39	100111	$\underbrace{0011}_{3}\ \underbrace{1001}_{9}$
27	11011	$\underbrace{0010}_{2}\ \underbrace{0111}_{7}$

In BCD, groups of four binary digits are used to represent decimal digits.

Binary Group	*Represents Decimal Digit*
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

There are some groups of four binary digits which are not used in BCD and which, accordingly, do not translate to decimal digits; that is, they do not represent any decimal digit at all. These groups are:

1010 1011 1100 1101 1110 1111

Notice that in pure binary these would be equivalent to 10, 11, 12, 13, 14, and 15. In the BCD representation, numbers larger than decimal 9 are represented by more than one group of four binary digits.

Comparison of Decimal, Binary, and BCD Numbers

Until you get used to them, binary numbers seem strange. After a while they lose their unfamiliarity and become almost as easy to work with as decimals. However, you have surely already noticed that binary numbers are longer than corresponding decimal numbers. Even after you have become quite used to binary numbers, it will still be difficult to handle large binary numbers in your mind—to compare magnitudes, remember them, and perform mental arithmetic. Look at these numbers:

1968 and 11110110000

The first is certainly easier to remember. Perhaps the mind is better adapted to remembering the greater variety of symbols used in the decimal system, rather than the long string of 0's and 1's used in the binary system. If this is true, the human mind is quite different from the modern digital computer, which is far more easily designed to use binary numbers.

BCD numbers are even longer than pure binary. For example,

1968 (decimal) = 0001 1001 0110 1000 (BCD)

However, with very little practice, translation back and forth between BCD and decimal can be done quickly and accurately in your head, and the magnitudes and relative sizes of large BCD numbers are more recognizable than those of binary numbers. Large BCD numbers are hard to remember, but conversion to decimal is so easy that their magnitudes are easily recognized by quick mental translations.

Codes

A string of binary digits (often abbreviated to "bits") can be thought of as a "code." Besides having equivalent decimal numbers, they can stand for almost any information at all. You only need a "codebook" or table to translate their meanings. If they represent codes in equipment someone else has designed, you need a codebook prepared by the designer in order to understand the code. If you are designing a logic system, you can let a string of bits stand for anything you choose, as long as in a single system or in two or more compatible systems the same string of bits stands for the same thing.

A string of bits is often used to stand for letters of the alphabet and also punctuation marks. Another common use is as "instructions." In a computer, applying one code may "tell" the computer to perform addition, while another code will tell it to subtract.

The size of the codebook depends on the number of bits used. A single bit allows only two codes. For example, 0 could mean "add," and 1 could mean "subtract." There is no way to include instructions to multiply or divide in this simple code without expanding it to a two-bit code.

The number of possible distinct codes increases with the number of bits used.

Number of Bits	*Number of Possible Distinct Codes*
1	2
2	4
3	8
4	16
n	2^n

The use of codes will become clearer as your familiarity with logic systems increases.

Binary Addition

Addition requires that an addition table be defined for any pair of digits. To add larger numbers the digits for corresponding places are added, according to the table, with carries to the next higher-value place as required.

The addition table for binary digits is much simpler than for decimal digits:

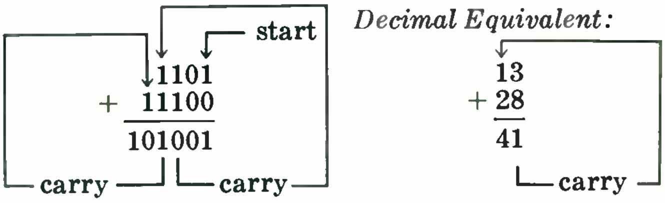
$$\begin{aligned}0+0&=0\\0+1&=1\\1+0&=1\\1+1&=10\end{aligned}$$

Sometimes this list is written in the form:

+	0	1
0	0	1
1	1	10

The addition of two binary numbers of more than one bit each is illustrated below. Add the least-significant digits first (at right unless otherwise stated). Then proceed a place at a time toward the higher significant places. "Carries" add to the next more significant column. If the addition of two 1's and a carry is required: $1 + 1 + 1 = 11$, or 1 with a carry.

Example: Add 1101 and 101001 in the binary system.



Do yourself: Add 101110 and 110011 in the binary number system. (Answer: 1100001.)

Binary Multiplication

The binary multiplication table is given in either of the following forms:

$$\begin{aligned}0\times 0&=0\\0\times 1&=0\\1\times 0&=0\\1\times 1&=1\end{aligned}$$

or

×	0	1
0	0	0
1	0	1

There are several ways to multiply larger numbers. One of these, illustrated below, follows the same pattern as our usual method for decimal multiplication.

Example: Multiply 1011 by 101.

```
       1011
 ×      101
     ------
       1011 ← 1 × 1011
      0000  ← 0 × 1011, shifted left once
     1011   ← 1 × 1011, shifted left twice
     ------
     110111 ← sum (answer)
```

You will recognize the similarity to decimal multiplication.

Try these binary multiplications yourself:

1. 110 × 1101. (Answer:1001110.)
2. 100 × 1001. (Answer: 100100.)

Did you notice that when the multiplier is 1 followed by *n* zeros, you can simply add *n* zeros to the multiplicand and get the correct answer?

In the decimal system: 1000 × 39 = 39,000.

In the binary system: 1000 × 1101 = 1101000.

Adding a zero at the right (least significant) end of a binary number doubles its value. Of course, it follows that taking a zero away from the same end halves the value.

1010 is twice as large as 101.

1101 is half as large as 11010.

(Translate these numbers to decimal numbers, as a check. Can you see why this must always be true?)

For much larger numbers than those used in the preceding examples it becomes necessary to add long columns of binary digits. In doing this it is very easy to make errors in the carries. A variation of the foregoing method of multiplication permits addition of only two binary numbers at a time, often simplifying the carries considerably.

```
        110101
 ×      101001
  ------------
        110101 ← 1 × 110101
     110101    ← shift left until a 1 is encoun-
  ------------   tered in multiplier
     111011101 ← add
   110101      ← shift left to next 1 in multiplier
  ------------
  100001111101 ← sum (answer)
```

The intermediate sums require additional writing, but eliminate an addition of $1 + 1 + 1 + 1 = 100$; that is, a carry two columns to the left. Of course, the carries are still more complex when the numbers are still larger.

Do yourself: Multiply 1110101 and 100111 both ways. (Answer: 1000111010011.) Which way was easier for you?

Binary Subtraction and Division

Since this book deals primarily with logic circuits rather than computers, we will not examine techniques for subtraction and division. These operations are described in many books on binary mathematics. They are not difficult, but are not required in order for the reader to understand the material in the remainder of this book.

Chapter 2

Boolean Algebra

Named after its inventor, the mathematician George Boole, Boolean algebra is a special kind of mathematics used to analyze and design logic systems. It is a complete "self-contained" mathematical system, with its own symbols and rules. You will find certain similarities to ordinary algebra; in fact, many rules for handling ordinary algebraic equations and expressions also apply to Boolean equations and expressions. But there are additional rules that work only for Boolean algebra.

Logic systems are composed of electric switches or electronic gates, both of which are two-state devices. Switches are either open or closed. Gates have high or low outputs and inputs (voltages or currents). Corresponding to these conditions, all quantities in Boolean algebra have only two possible values, symbolized as 0 or 1.

The relationship between the binary number system and Boolean algebra is often puzzling when the two are first encountered. Despite the facts that both use the binary symbols 0 and 1 and are associated with digital computers, they are separate subjects. Binary numbers are used in computers for calculating, in place of our familiar decimal numbers. Boolean algebra is used to design logic systems and digital control systems. However, instruction codes used in logic systems may sometimes be identified with the binary number which has the same sequence of binary digits.

Letters of the alphabet, capital or lower case, are used in Boolean algebra to stand for "variables," quantities whose values (0 or 1) may be assigned, or whose values are not known.

There are three basic "operations" in Boolean algebra. These are listed below along with their symbols and the way the symbols are read aloud.

Operation	*Symbol*	*Pronounced*
OR (logical addition)	$A + B$	A or B
AND (logical multiplication)	$\left\{\begin{matrix} A \times B \\ A \cdot B \\ AB \end{matrix}\right\}$	A and B
NOT (negation or complementation)	$\left\{\begin{matrix} \bar{A} \\ A' \end{matrix}\right\}$	not A (*or*) "the complement of A"

To "perform an operation," we determine a third quantity from two other quantities. To find this third quantity, tables and rules are used. For example, in conventional arithmetic, if we are given any two numbers we may perform the operation of addition with the help of rules for addition and the addition table. In this case, the third number is called the *sum*.

Since Boolean quantities can have only two values, 0 or 1, tables defining the three basic operations are short and easy to learn. You should memorize them thoroughly if you expect to be working with logic systems. These tables are given in the following sections.

The OR Operation (Logical Addition)

The OR operation is defined by the following table:

A	B	$A + B$
0	0	0
0	1	1
1	0	1
1	1	1

first quantity (A)
second quantity (B)
third quantity (logical sum) ($A + B$)

Every possibility for A and B is listed in the four lines above.

The same information is sometimes presented in this form:

	+ (OR)	A = 0	A = 1
B	0	0	1
B	1	1	1

The two possible values of A head the two columns. The two possible values of B identify the two rows. The intersection of the column and row gives the third quantity defined by the OR operation, sometimes called the *logical* sum of A and B.

Example: Find A OR B for $A = 0$, $B = 1$.

Answer: In the first form of the OR table the answer is on the second line, where we read that $A + B = 1$. In the second form of the table, the answer is in the square under the column $A = 0$, and on the line $B = 1$. Again, we read that the logical sum of $0 + 1$ is 1.

Notice that $A + B = 1$ if either A OR B is 1. Also from the table we see that $A + B = B + A$, for all values of A and B. In mathematical terms the OR operation *commutes;* in everyday language we can simply say that the order of the terms does not affect the answer.

The OR table is not the same as the binary addition table. The two tables do not refer to the same subject, nor to the same mathematical systems.

OR (logical addition): $1 + 1 = 1$ (say, "one OR one equals one")

binary addition: $1 + 1 = 10$ (say, "one plus one equals one-zero")

There is no such quantity as ten in Boolean algebra.

The AND Operation (Logical Multiplication)

The AND operation is defined by the following table:

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

first quantity ↑ (under A)
second quantity ↑ (under B)
↑ third quantity (logical product) (under AB)

where AB denotes A AND B.

An alternate form of the table is:

	× (AND)	A: 0	A: 1
B	0	0	0
B	1	0	1

Notice that $AB = 1$ only if both A AND B are 1. Also the order of the terms does not matter; that is, $AB = BA$. (Mathematicians say that the AND operation "commutes.") This is quickly confirmed from the table, which shows that $1 \times 0 = 0 \times 1 = 0$.

Although the logical multiplication table and the binary multiplication table appear the same, each is really part of its own mathematical system.

The NOT Operation

The NOT operation has only one "input" quantity instead of two, and in this sense is a different kind of operation from the OR and AND operations. However, this property is of no special significance to us. The defining table is:

A	$\bar{A}$
0	1
1	0

The complement of 0 is 1. The complement of 1 is 0. Stated in another way, if $A = 0$, then NOT A (or $\bar{A}$) $= 1$, and vice versa.

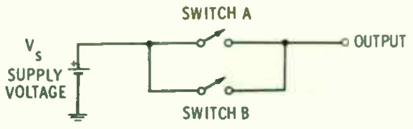
Boolean Algebra and Electric Switch Circuits

We are going to take a brief "time out" from the theory of Boolean algebra, in order to be introduced to its relationship to logic circuits. When we return to more abstract mathematics, this glimpse of its practical use should make the theory more palatable.

In this introduction we will use switches as our "logic elements" rather than electronic gates. Two or more interconnected switches will be called a *logic circuit*. Each switch can be either open or closed. When a switch is moved (by hand or by applying a voltage to a relay coil) to its closed position, we will say it has an *input* of 1. When it is open we will call its input 0. A voltage at the output of the switch is a 1 output; no voltage is a 0 output.

Fig. 2-1 shows an OR logic circuit consisting of switches A and B connected in parallel. With the circuit is a table which tells us for which combinations of switch positions there is an output voltage.

The table in Fig. 2-1 is the same as the OR table (logical addition) defined earlier. Thus, the Boolean OR operation symbolizes two switches connected in parallel. Conversely, parallel-connected switches are called an OR logic circuit. The Boolean

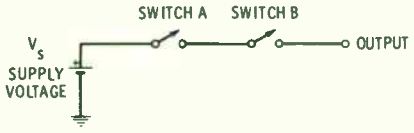


POSITION OF SWITCH A	POSITION OF SWITCH B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1
INPUTS 0 = OPEN	1 = CLOSED	0 = NO VOLTAGE 1 = V_S

Fig. 2-1. OR logic circuit using switches.

equation for the circuit shown is: Output $= A + B$, where A designates the closed state of switch $A = 1$,and $B = 1$ the closed state of switch B.

There is no limit to the number of switches that can be connected in parallel. For n switches the equation becomes $A_1 + A_2 + A_3 + \cdot \cdot \cdot + A_n =$ output. Notice that the order of listing the terms does not matter, any more than does the order of switches in the schematic. There is an output if A_1 OR A_2 OR A_3, etc., is closed.



POSITION OF SWITCH A	POSITION OF SWITCH B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1
INPUTS 0 = OPEN	1 = CLOSED	0 = NO VOLTAGE 1 = V_S

Fig. 2-2. AND circuit using switches.

Now look at the circuit in Fig. 2-2, where two switches are connected in series. Again, we can make a table showing the output for every switch combination. As before, there are four possible situations where switches A and B are open or closed (0 or 1). This table, you will notice, is the same as the Boolean AND table, and accordingly switches in series are called a logical AND circuit. The Boolean algebra equation which represents this circuit is output $= A \times B$, or simply output $= AB$, where $A = 1$ designates the closed state of switch A, and $B = 1$ the closed state of switch B.

We can connect any number of switches in series, but there is a 1 output only if all the switches are closed. In this case, the Boolean equation becomes

$$\text{Output} = A_1A_2A_3 \cdot\cdot\cdot A_n$$

where n is the number of switches.

In Fig. 2-3, a more complex switching circuit is shown. Switches A and B are shown twice. They are switches with two poles, where operating a single "handle" operates two sets of contacts at once. The dotted lines show that the contacts act together. If we say that $A = 1$, we mean that both pairs of contacts labeled A are closed; if $A = 0$, both are open. Switch B has two poles too, but whenever one is open the other is closed, and vice versa. We can represent this by calling one condition of these pairs of contacts B and the other condition $\bar{B}$ (NOT B). In this case, it becomes necessary to redefine slightly the 0 and 1 switch positions, in terms of the switch handle position. The handle position for which contact B is closed is 1, while 0 is the handle position for which contact $\bar{B}$ is closed.

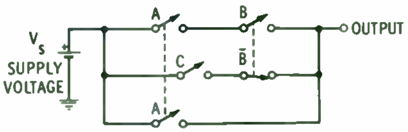


Fig. 2-3. Logic circuit with several switches.

Since there are three switches (inputs), there are eight lines in the table summarizing the output for each possible set of switch conditions (Table 2-1). Notice that there are five conditions for which there is a voltage output.

We can write a Boolean algebra equation describing the operation of the logic circuit of Fig. 2-3. Since there are three *parallel* branches, the equation will be of the form

$$\text{Output} = (\qquad) + (\qquad) + (\qquad)$$

Table 2-1. Output for All Switch Positions for Logic Circuit of Fig. 2-3

Position of Switch A	Position of Switch B	Position of Switch C	Output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

In the upper branch switches A and B are connected in series. The term for this branch to be a closed circuit is AB, so we substitute this in the first pair of parentheses. The term describing the middle branch is $\overline{B}C$, which we substitute in the second set of parentheses. In the last branch there is only one switch, so we place A inside the third pair of parentheses. The complete Boolean algebraic equation becomes

$$\text{Output} = AB + \overline{B}C + A$$

If we let the output be represented by D, then we can write

$$D = AB + \overline{B}C + A$$

Notice that the upper branch could be left off without affecting the logic of this particular circuit; that is, the circuit in Fig. 2-4, which is simpler, will do the same job, because it has the same logic table.

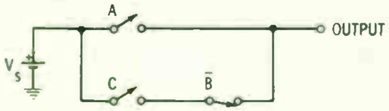


Fig. 2-4. Simplified equivalent logic circuit for Fig. 2-3.

In designing logic circuits, especially more complicated ones, it nearly always happens that there are many different circuits which will accomplish the desired objective. One of the problems of the circuit designer is to find the simplest, often lowest cost, way of achieving a certain logic function. It often is not obvious just what circuit this is. Writing Boolean equations and then rearranging and simplifying the equations following the rules of Boolean algebra is one technique for finding simpler circuits.

Notice that not only can we write the equation for a logic circuit from the circuit diagram, but we can also do the reverse—we can draw the circuit if we are given the Boolean equation. Each group of factors separated by a + (OR) sign stands for a parallel branch. The factors themselves (such as AB, $\overline{B}C$, etc.) represent conditions of switches in series along that particular branch.

The tables we have constructed for the various circuits are called *truth tables* or *logic tables*. They are used extensively in analysis and design, alone or hand in hand with the Boolean equations. To illustrate one use of these tables, look at Table 2-1 and observe that the output is 1 whenever switch A is closed ($A = 1$), or if switch C is closed ($C = 1$) at the same time that switch B is open ($B = 0$). When switch B is open ($B = 0$), switch $\overline{B}$ is closed ($\overline{B} = 1$), so we could alternately say that the output is 1 if A is 1 or if both C and $\overline{B}$ are 1. The equation for this is

$$\text{Output} = A + C\overline{B}$$

From this equation we can draw the circuit of Fig. 2-4 directly. Thus, had we not noticed the simplified circuit otherwise, we might have discovered it by examining the truth table.

In Fig. 2-4 you may be confused because the contacts of switch B are drawn closed and labeled $\overline{B}$. This representation resulted from an arbitrary decision when we drew Fig. 2-3. Switch B consists of two opposite-acting poles—one is open whenever the other is closed—it was clear enough that one should be labeled B and the other $\overline{B}$. When we eliminated one pair of contacts (one pole), it happened to be those called B, and we were left with only switch $\overline{B}$ in the circuit. In a practical situation such an assignment of symbols might have been based on the positions of the handle of a toggle switch, "1" meaning the handle is up and "0" meaning the handle is down. If the switch were relay operated, contacts which are open when the relay coil is unenergized would be labeled B, and the normally closed contacts $\overline{B}$.

In all branches of engineering the problem of analysis (finding how something works) is quite different from the problem of design or synthesis (putting elements together to work in a particular way). In general it is much easier to determine how something works. This is as true in the field of logic systems as in other areas of electronics. However, the two tools we are studying in this chapter—truth tables and Boolean algebra—are useful for both analyzing and synthesizing logic circuits.

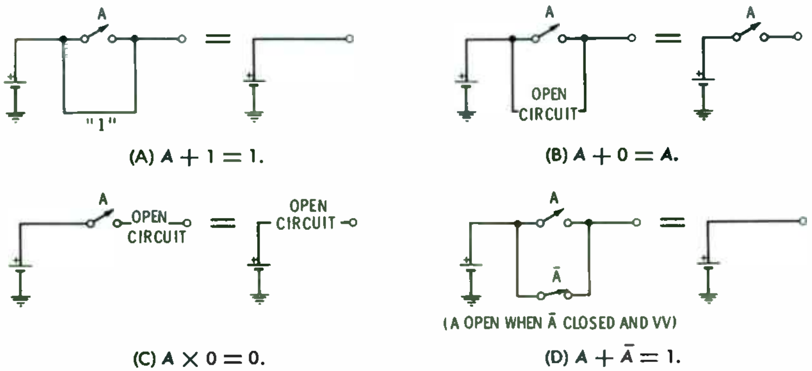


Fig. 2-5. Some basic Boolean expressions and equivalent switch circuits.

With these glimpses into the applications of Boolean algebra, we now return to learn more about it. So far we have learned the basic operations, how these correspond to parallel and series-switching circuits, and how Boolean expressions can represent these circuits. The next step is to study ways to rearrange and simplify these expressions, keeping in mind that the equations so obtained represent equivalent circuits for performing the same logic function.

Basic Relations and Laws of Boolean Algebra

The following basic relations are easily verified from the truth tables of the OR and AND operations. From the OR truth table:

$$A + 0 = A$$
$$A + 1 = 1$$
$$A + A + A + \cdots + A = A$$
$$A + \bar{A} = 1$$

You will find that the equations are valid whether $A = 0$ or whether $A = 1$.

From the AND truth table you can prove that:

$$A \times 0 = 0$$
$$A \times 1 = A$$
$$A \times A \times A \times \cdots \times A = A$$
$$A \times \bar{A} = 0$$

As before, let $A = 1$ and you will see that each equation is true; then let $A = 0$ and the equations will again be true.

A rather obvious, but frequently useful, relation is

$$\overline{(\bar{A})} = A$$

That is, if a Boolean quantity is complemented, then complemented again, the original quantity is obtained.

Arithmetic addition, arithmetic multiplication, logical addition (OR'ing), and logical multiplication (AND'ing) are all called *binary operations*. The word *binary* in this case does not refer to binary numbers, but rather means that the operations take *two* quantities to get an answer. Recall that the OR table gives the result of operating on or combining *two* quantities, just as the arithmetic addition table gives the *sum* of two quantities.

The *law of association* tells us how to combine three or more quantities. We first take any two of the quantities and replace them by the quantity obtained by performing the operation on them. The law is written in symbolic shorthand by these equations:

$$A + B + C = (A + B) + C = A + (B + C)$$
$$ABC = (AB)C = A(BC)$$

The first equation is for the OR operation; the second is for the AND operation. The parentheses indicate that the quantities inside are to be combined first and the result substituted in their place. The fact that the pairs of parentheses can enclose different quantities means that it does not matter which two quantities we choose to combine first—we get the same answer in any order we take them.

Table 2-2 shows a truth table which proves the law of association for the OR case. The eight lines contain the eight possible combinations of states that A, B, and C can have. For each combination, the quantities $(A + B)$ and $(B + C)$ are computed. Then each of these results is combined with C and A, respectively, to form the last two columns. The quantities (values) in the last two columns are the same, for each possible set of values for A, B, and C. Therefore, we can say it is always true that

$$(A + B) + C = A + (B + C)$$

Do yourself: Construct a truth table to prove that $(AB)C = A(BC)$.

The *law of distribution* is written as follows:

$$A(B + C) = AB + AC$$

Table 2-2. Truth Table for Law of Association With OR Operation

A	B	C	A+B	B+C	(A+B)+C	A+(B+C)
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	0	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

This is, of course, familiar to you from ordinary algebra, and it is equally valid for Boolean algebra. Working this law backwards, we perform the process of *factoring;* that is,

$$AB + AC = A(B + C)$$

More than one common factor may be "removed" from each term in a series of OR'ed terms and placed outside the parentheses. For example:

$$ABC + ABD = AB(C + D)$$
$$AB\bar{C}D + A\bar{C}D = A\bar{C}D(B + 1)$$

Once again, we can use a truth table to prove equivalency of two Boolean expressions, in this case those expressions illustrating the law of distribution. Look at Table 2-3. As always, where three variables are involved, there are eight lines to the truth table. For each possible set of values of A, B, and C we can readily "compute" $(B + C)$. This result is AND'ed with A, giving the logical product $A(B + C)$ and forming the fifth column. In the sixth and seventh columns we compute AB and AC. Finally, for each pair of values for AB and AC we find $(AB + AC)$ and place this quantity in the last column. Now compare the fifth and eighth columns: They are identical. We therefore can conclude that no matter what values A, B, and C have, it is always true that

$$A(B + C) = AB + AC$$

Another useful relation is this:

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

The equivalency of these two expressions is easily proved by using a truth table, and it is suggested that you construct a table and satisfy yourself that the expressions are equivalent.

Table 2-3. Truth Table for Law of Distribution With OR Operation

A	B	C	B+C	A(B+C)	AB	AC	AB+AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Two very useful relations are called *De Morgan's laws:*

$$(1) \qquad \overline{AB} = \bar{A} + \bar{B}$$
$$(2) \qquad \overline{A + B} = \bar{A}\bar{B}$$

Their greatest value arises in designing circuits containing electronic gates, as you will see in a later chapter. Some people find the following statements help them remember these laws:

"The complement of the product equals the sum of the complements."
"The complement of the sum equals the product of the complements."

Do yourself: Construct truth tables to prove De Morgan's laws. This will give you practice in making truth tables and will help you remember the two laws.

Although the main subject of this section has been the fundamental rules and relations in Boolean algebra, notice the part played by the truth table. By expanding it from the simple form which defines the AND and OR operations, we have used the truth table as a means of proving that two expressions are equivalent for all values of the variables. This is practical because the number of possible combinations of variable quantities is finite—four combinations for two variables, eight combinations for three variables, 2^n combinations for n variables. Of course, when n is large, there will be a great number of lines in the truth table. In our ordinary number system, such an approach to proving equivalence of two expressions could not work, as each variable can have an infinite number of possible values.

Each of the basic relations in this section represents an electric switch circuit. Whenever a *zero* (0) appears, it is equivalent to no connection at all—a permanent "open circuit." A *one* (1) represents a switch that is always closed; a piece of wire will do as well and costs less. Fig. 2-5 contains a number of examples of basic Boolean expressions and equivalent switch circuits.

Simplifying Boolean Expressions

We *evaluate* Boolean expressions by substituting numbers (0 or 1) for the variables and performing the operations. We have been evaluating expressions right along in making up truth tables. Almost any two expressions, when evaluated for certain sets of numbers, will produce the same answer. For example, if $A = 1$, $B = 0$, and $C = 1$,

$$AB + \overline{B}C = (1 \times 0) + (1 \times 1) = 0 + 1 = 1$$
$$ABC + A = (1 \times 0 \times 1) + 1 = 0 + 1 = 1$$

However, if we evaluate these expressions for $A = 1$, $B = 0$, and $C = 0$ we do *not* get the same answer from both.

Equivalent expressions are those which, when evaluated for *any* set of numbers, give the same result. They represent circuits of interconnected switches or electronic gates which perform the same logic function, even though like the Boolean expressions, they may appear quite different. The equals sign is used to show that two Boolean expressions are equivalent:

$$A\overline{B} + B = A + B$$

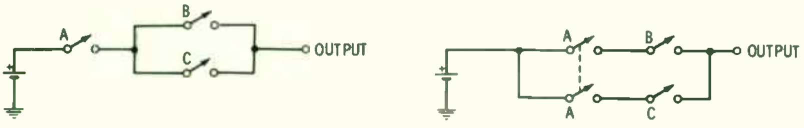
This is a correct Boolean statement of equivalency.

Boolean expressions connected by an equals sign have the appearance of an algebraic equation. However, it is not correct in Boolean algebra to "solve" these equations as we would in ordinary algebra. For example,

$$A\overline{B} + \cancel{B} = A + \cancel{B}$$
$$\cancel{A}\overline{B} = \cancel{A}$$
$$\overline{B} = 1$$

is not correct. In ordinary algebra, the above procedure involves the operations of subtraction and division. Neither of these operations are a part of Boolean algebra.

In Fig. 2-6, circuits 2-6A and 2-6B and their Boolean expressions are equivalent logically, even though both the schematic diagrams and the Boolean expressions are different. However, circuit 2-6A contains only three single-pole switches while circuit 2-6B contains two single-pole and one double-pole switches. Therefore circuit 2-6A is simpler and would probably cost less to construct. The object of simplifying Boolean expressions is to discover alternate logic schematics, usually for



(A) A(B+C). (B) AB + AC.

Fig. 2-6. Logically equivalent switch circuits.

the purpose of minimizing the number and complexity of switches and gates.

We will now examine several Boolean expressions and see how the basic relations and rules can be applied to find simpler equivalent expressions.

Example No. 1: Simplify $(A + AB)$.

	Reason
$A + AB = A(1 + B)$	factor
$= A \times 1$	$1 + B = 1$
$= A$	$A \times 1 = A$

Example No. 2: Simplify $AB + A\bar{B}$.

	Reason
$AB + A\bar{B} = A(B + \bar{B})$	factor
$= A \times 1$	$B + \bar{B} = 1$
$= A$	$A \times 1 = A$

Example No. 3: Simplify $A(A + B)$.

	Reason
$A(A + B) = AA + AB$	law of distribution
$= A + AB$	$A \times A = A$
$= A(1 + B)$	factor
$= A \times 1$	$1 + B = 1$
$= A$	$A \times 1 = A$

Example No. 4: Simplify $(A + B)(A + C)$.

	Reason
$(A + B)(A + C) = AA + AC + BA + BC$	law of distribution
$= A + AC + BA + BC$	$A \times A = A$
$= A(1 + C + B) + BC$	factor
$= (A \times 1) + BC$	$1 +$ any quantity $= 1$
$= A + BC$	$A \times 1 = A$

Do *yourself*: Simplify:

(1) $(A + \bar{B})B$

(2) $(A + B)(A + \bar{B})$

(3) $\bar{A}BC + A\bar{B}\bar{C} + AB\bar{C} + ABC$

Answers: (1) AB; (2) A; (3) $BC + A\bar{C}$.

In the third problem, you may have noticed that simplification can proceed along more than one route. Sometimes one route will lead to a simpler expression than will another route. It may be necessary to try several routes to arrive at the simplest expression, even though all routes lead to expressions which are less complex than the original one.

If in the third problem above you began by factoring B out of the first and third terms, you obtained

$$B(\bar{A}C + A\bar{C}) + A(\bar{B}\bar{C} + BC)$$

This does indeed have fewer terms, and therefore requires fewer switches to implement. However, if you begin by factoring BC from the first and last terms you get instead

$$BC(\bar{A} + A) + A\bar{C}(\bar{B} + B) = BC + A\bar{C}$$

since $\bar{A} + A = 1$ and $\bar{B} + B = 1$.

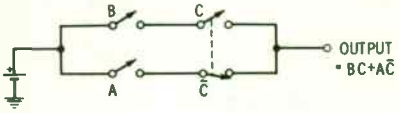


Fig. 2-7. Switch circuit for $BC + A\bar{C}$.

The switch circuit that accomplishes the logic of problem 3 is shown in Fig. 2-7. You can check this circuit by making a truth table for both the original expression and for the simplified circuit and expression.

Design Procedure Using Truth Tables and Boolean Expressions

In designing a logic system, the engineer usually begins with a word problem—a verbal description of what the system is to accomplish. From here he must progress to the final design, which will include a schematic showing how to interconnect certain logic elements to satisfy the logic requirements.

An approach, which is more and more useful as the complexity increases, consists of the following steps:

1. Word description of problem.
2. Truth table.
3. Boolean expression.
4. Simplified equivalent Boolean expression.
5. Schematic.

To illustrate this procedure, suppose we are to design a logic system where an electric motor is to operate from three switches.* The 1 condition of each switch is with its handle *up*. The motor is to operate if:

(1) Switch C is up and switches A and B are down.
(2) Or, if switches A and B are up and C is down.
(3) Or, if all three switches are up.

From this description, make a truth table. Try it yourself, then compare your table with the one in Table 2-4.

After the truth table, the next step is to write a Boolean expression. There is more than one way to do this, but the method which follows, sometimes called the *minterm method*, is usually the easiest.

From the truth table you can see that there are three ways to get a 1 at the output (motor running).

(1) If $\bar{A}$ and $\bar{B}$ and C are all 1's.
(2) Or, if A and B and $\bar{C}$ are all 1's.
(3) Or, if A and B and C are all 1's.

Notice how we have given a "different twist" to the problem. Instead of saying "if A is 0," we say "if $\bar{A}$ is 1." These are equivalent statements.

It is now easy to write the Boolean expression. Every place we have said "or" we use the OR sign (+), and every place we have said "and" we indicate the AND operation. Our expression is

$$\bar{A}\bar{B}C + AB\bar{C} + ABC$$

Table 2-4. Truth Table for Electric Motor Problem

A	B	C	Output
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

*Switches are used as logic elements for examples in this chapter because they illustrate logic concepts even to those not familiar with electronic gates.

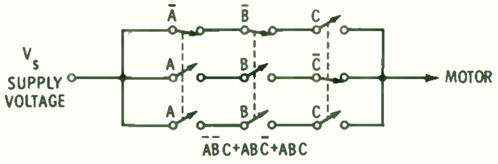
This expression will have a value of 1 if the values of A, B, and C correspond to those on lines two, seven, or eight in the truth table (Table 2-4). For any other values (other lines in the truth table) the value of the expression is 0.

Using a more mechanical procedure we could have written down a pair of blank parentheses for each line in the truth table having an output of 1. We would then have connected these with OR signs, and have filled in each pair of parentheses with the logical product of the variables—complemented when the corresponding line shows a *zero* for the variable and uncomplemented when that line shows a *one* for the variable. That is, first write

$$\underset{\uparrow}{(\quad)} + \underset{\uparrow}{(\quad)} + \underset{\uparrow}{(\quad)}\ \text{a term for each line where the output is 1}$$

Next fill in the parentheses.

Example: The first pair of parentheses correspond to the second line of the truth table (Table 2-4), where $A = 0$, $B = 0$, and $C = 1$. Therefore fill in the parentheses with $\bar{A} \times \bar{B} \times C$ or, more simply, $\bar{A}\bar{B}C$. (Use the same rules for filling in the other two pairs of parentheses. The resulting expression will, of course, be the same as before.)



(SWITCHES SHOWN IN '0' POSITION)

Fig. 2-8. Possible schematic for motor problem.

From the earlier discussion of the relation between electric switch connections and Boolean expressions, you will note that we can now draw a schematic (Fig. 2-8.) However, we should examine the expression to see if we can find an equivalent one which is simpler. Especially, we should look for common terms which may be factored, leaving expressions that can be replaced by 1. Try this yourself before reading on.

The correct simplification is:

$$\begin{aligned}\bar{A}\bar{B}C + AB\bar{C} + ABC &= \bar{A}\bar{B}C + AB(\bar{C} + C)\\ &= \bar{A}\bar{B}C + AB\end{aligned}$$

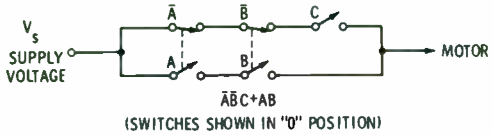


Fig. 2-9. Simplified motor-problem schematic.

No further simplification is possible, because each factor appears complemented or uncomplemented just one time.

We can now draw the simpler switch connection diagram of Fig. 2-9, the solution to the problem. Remember that the OR sign indicates parallel paths; the logical multiplication (AND operation) indicates series connections.

Chapter 3

Gates

Gates are the basic building blocks of electronic logic systems. In some ways, a gate functions like one or more electric switches. A gate has an output terminal, where the voltage can have approximately one of two nominal values. These two values can be called the "0" and "1" output states. A gate has two or more inputs, in the form of voltages or currents, and each input can be called either a "0" input or a "1" input. Like two or more switches, a gate can be equivalent to an OR logic circuit or an AND logic circuit. But that is about as far as we can go in finding similarities. The *differences* between electronic gates and electric switch circuits underlay the phenomenal growth in the usage of gates in digital logic systems.

The modern solid-state, integrated-circuit gate is characterized by:

1. *High speed*—The output changes from one state to the other in as little as 2 nanoseconds (2×10^{-9} second) following a signal change applied to the input.
2. *Small size*—Hundreds of gates, the equivalent of 500 or more switches, can be built on a silicon "chip" smaller than your little fingernail.
3. *Low power*—These same hundreds of gates may require less than 200 milliwatts of electric power.
4. *Low cost*—In large volumes the cost per gate may be as low as 5 cents.

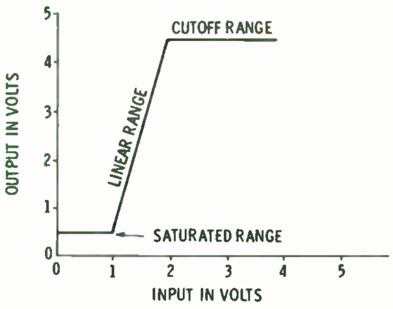


Fig. 3-1. Typical transistor amplifier input-output relation.

Gates Are a Special Class of Amplifiers

Except for diode gates which we will consider briefly later, electronic gates are really special forms of direct-coupled amplifiers. As amplifiers they are intended to be operated not in their "linear" operating range, but rather in their extreme bias conditions. Fig. 3-1 shows a typical relationship between the input voltage and output voltage in a transistor amplifier. (The same general relation also holds true for a vacuum-tube, direct-coupled amplifier, except that the voltage scales are different. However, there are very few vacuum-tube gates in use today, and those few that remain will nearly all be replaced eventually by equipment using smaller, more efficient, and more dependable solid-state gates.)

Amplifier input-output curves (Fig. 3-1) have a nearly linear section, normally used for signal amplification. Operating in this linear range an amplifier produces minimum distortion. The two end sections are often called the *saturated* and *cutoff* regions. However, in logic terminology the word "saturation" may also have a slightly different meaning, as we will see when we consider saturated logic versus nonsaturated logic.

Gates are not intended to be operated in the linear region of their input-output transfer curves. They are normally biased at either one end of the curve or the other. The in-between region is a logic "no-man's land." The two horizontal sections of the curve represent the 0 and 1 states of the gate.

Except for the NOT gate, or *inverter*, each gate has more than one input. In the case of OR gates the transfer curve applies to any input, but once one input is "high," the states of the other inputs make no difference in the output state. For AND gates the curve applies to any input when all other inputs are 1's.

Fig. 3-2A shows the schematic for one kind of a two-input gate. The inputs are at *A* and *B* and the output is at *C*. When the input voltages at *A* and *B* are both zero, or nearly so, no base current flows in either transistor. With no base current there is also no emitter-collector current, except for a small and negligible leakage current, and therefore essentially no current flows through the common collector resistor. With no current through the resistor, the output voltage at *C* is equal to the supply voltage. If sufficient voltage is applied to either input *A* or *B*, base current flows, collector current flows, and the voltage at *C* drops to about a quarter of a volt, the transistor "saturation" voltage. The input-output curve is shown in Fig. 3-2B, and you will notice that although it is reversed from the curve of Fig. 3-1, it nevertheless has the same flat high and low regions with a steep but nearly linear transition region between. We will consider this difference in more detail later.

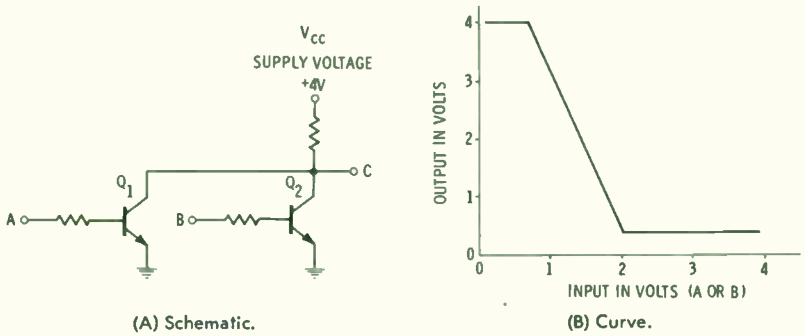


Fig. 3-2. Schematic and input-output curve for one kind of gate.

Although only two inputs are shown in Fig. 3-2, you can see we could add many more inputs simply by adding one transistor and one resistor for each additional one. The common collector resistor continues to serve all the transistors. There is, of course, a point where the total leakage current of many transistors can no longer be ignored, especially at higher temperatures where leakage currents increase. However, gates with up to eight inputs are quite common, and even here the limit is due more to such practical restrictions as the number of terminals available in standardized integrated-circuit packages.

Kinds of Gate Logic

Digital integrated circuits can be classified in a number of ways. The gate in Fig. 3-2 is called an RTL NOR gate and was one of the earliest kinds available in integrated-circuit form. "RTL" means *resistor-transistor logic* and refers to the details of the circuit—the types of electrical components and the way they are interconnected. "NOR" refers to the logic function performed, and is a contraction of NOT-OR, meaning a combination of an OR function with a NOT (or inverting) function. Other circuit-type classifications such as DTL, TTL, and ECL will be considered in more detail in a later chapter. In this section we will be concerned with the logic function performed by the gate.

It is sometimes helpful, but seldom completely necessary, to know just what *is* inside a gate that you plan to use. One advantage of integrated-circuit gates is that the gate circuitry is all designed for you. You can use the gates by simply following the rules set forth by the manufacturer, along with a knowledge of the capabilities and limitations of the gates, which are provided in the manufacturers' published technical data.

Gates are classified by logic function as OR, AND, NOR, and NAND gates. Table 3-1 gives the truth table defining these four functions for two- and three-input gates. You have already become familiar with the OR and AND functions in the last chapter. From the truth table you will notice that the NOR function is equal to an OR function followed by an inverter (NOT function). Similarly, the NAND function is equivalent to an AND gate followed by an inverter.

Table 3-1. Truth Tables for Several Kinds of Logic Gates

Outputs For Different Kinds of Two-Input Gates					
Inputs		Gate			
A	B	OR	AND	NOR	NAND
0	0	0	0	1	1
0	1	1	0	0	1
1	0	1	0	0	1
1	1	1	1	0	0

Outputs For Different Kinds of Three-Input Gates						
Inputs			Gate			
A	B	C	OR	AND	NOR	NAND
0	0	0	0	0	1	1
0	0	1	1	0	0	1
0	1	0	1	0	0	1
0	1	1	1	0	0	1
1	0	0	1	0	0	1
1	0	1	1	0	0	1
1	1	0	1	0	0	1
1	1	1	1	1	0	0

It is a characteristic of all four types that the output is at one state for *all except one* of the possible input conditions (see Table 3-1). For example, the OR gate has a 1 output unless all inputs are 0; the NAND gate has a 1 output unless all inputs are 1. This is true for any number of inputs.

By comparing the description of operation of the RTL NOR circuit in Fig. 3-2, with the OR columns in Table 3-1, you will see why the circuit is classified as NOR logic.

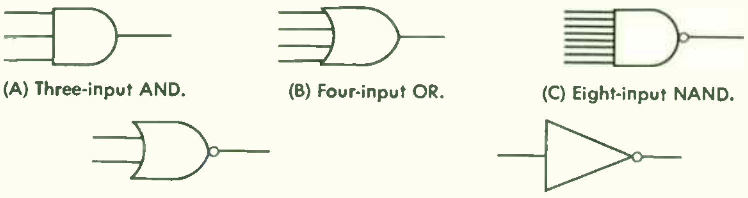
Schematic Symbols for Gates

There is no need to draw the internal circuit of each gate in most logic system schematics. Such a procedure might make an impressively complex schematic, but definitely does *not* help others to understand how the system operates. Certain simplified symbols have come into common use, although they are not completely standardized. The most common symbols are shown in Fig. 3-3, for the various gates. Usually there are either two, three, four, or eight inputs to a gate, the number of inputs being shown by the number of "wires" extending from the input side. There is always just one output.

The small circle at the output end of a gate is frequently used to symbolize inversion (negation), as shown here. Occasionally you may see the circle placed at the input of a gate, but in this book inversion at a gate input will be represented by a separate inverter symbol.

Fig. 3-4 illustrates a schematic showing several interconnected gates. The Boolean expressions are written over the wires to indicate the logic. Notice that the "system" in this example has two outputs, although each gate alone has only one. For the upper output, De Morgan's rules have been applied to give two equivalent Boolean expressions.

NAND and NOR gates are sometimes used as inverters by tying all inputs together. Study Table 3-1 to prove to yourself that this will work.


(A) Three-input AND. (B) Four-input OR. (C) Eight-input NAND.

(D) Two-input NOR. (E) Inverter (NOT function).

Fig. 3-3. Commonly used gate symbols.

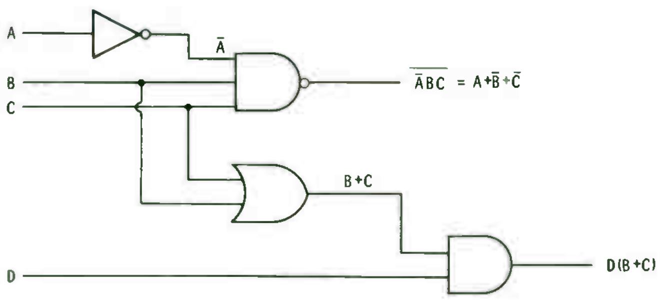


Fig. 3-4. Logic diagram showing interconnected gates.

Signal Ranges

Referring back to Fig. 3-1, you will observe that there is a range of voltages near zero that can be applied to the input without any effect on the output. In the curve of Fig. 3-1 this range is approximately 0 to 0.7 volt. Any input voltage within this range is assigned a binary symbol, either 0 or 1. For the present, we will call this range the logical 0 input voltage range, which is the most common assignment.

At the other extreme, any input voltage over 1.5 volts has about the same effect on the output, causing the output to become about 4.5 volts. Therefore any voltage in the range of 1.5 volts or higher can be called a logical 1 signal. Of course, there is an upper limit to the input voltage; too much voltage will cause permanent damage to any gate. Practically speaking, the power supply voltage, often 4 or 5 volts, is usually considered the upper limit of the 1 input.

There are many different gate designs, and each has its own ranges of voltages which are acceptable 0's and 1's. When the range nearest 0 volts is assigned the symbol "0," and the higher range is assigned the symbol "1," we say we are using *positive logic*. The opposite assignment is called *negative logic*, and is less common, but nevertheless sometimes very useful. In this book, unless we specify otherwise, we will always use positive logic. We will, however, briefly examine the uses of negative logic later.

Most often, gates are used in groups, with the output of one gate driving the input of one or more other gates. This requires that the input and output voltage ranges be compatible. It would not do to have the output of a gate at 0.8 volt in its 0 state, when a 0 input must be 0.6 volt or less, because then we could not connect the output to another gate of the same type and expect the second gate to perform properly. As a general rule, therefore, the 0 and 1 states represent the same ranges of voltages, whether we are speaking of the input or the output of a gate. A *compatible family* consists of a set of different gate types which have the same 0 and 1 voltage ranges and can be interconnected to form a logic system.

Loading and Fan-Out

Because of its simple internal structure, we will use the RTL gate of Fig. 3-2 to illustrate the loading limits of gates. Part of that schematic is redrawn in Fig. 3-5. There the collector resistor of the gate is shown, and we have added a load to the gate—in this case the input circuits of four more gates. Now assume the output of the first gate is in the "high" or 1 state.

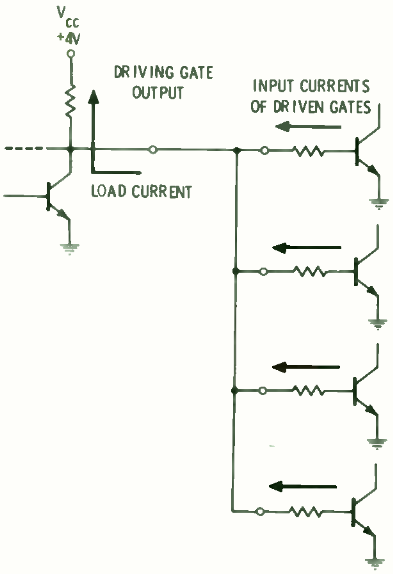


Fig. 3-5. Gate loading.

Although there is no current in any of the collectors of the transistors in the first gate, current flows into the first gate output terminal from the input terminals of each of the four gates connected to it. This output current causes a voltage drop across the collector resistor, so that the output voltage is not 4 volts at all, but quite a bit less. If too many gates are connected, the output voltage will fall low enough that it is no longer in the logical 1 range. Not being high enough, it cannot properly operate the gates connected as a load. The gate has been loaded too heavily. We say that its *fan-out* has been exceeded.

Fig. 3-6 shows the relationship between output voltage and output current. The actual numbers are for a typical RTL gate, but do not represent any particular manufacturer's design. Assume that for the gate represented by Fig. 3-6 a logical 0 is any voltage between 0 and 0.6 volt, and a logical 1 is any voltage between 2 and 4 volts. From the curve, we see that the logical 1 output will be 2 volts or higher as long as the output current does not exceed 6 mA. Now suppose we were given the additional information that when 2 volts is applied to the input of one of these gates, the current into it will be no greater than 2 milliamperes. We can see that we should not connect more than three gate inputs to the output if we want to be sure the 1 state output voltage does not fall below the minimum allowable 2 volts. The maximum fan-out is three.

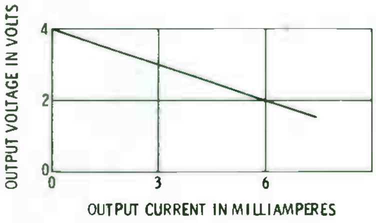


Fig. 3-6. Typical output characteristic of RTL gate.

Some manufacturers, instead of giving a curve such as in Fig. 3-6 along with the input currents of various gates that are compatible with one another, will give fan-out data in terms of a *loading factor*. Load factors are relatively small whole numbers that simplify the estimation of fan-out limits. For example, for a particular gate, the output load factor might be specified as 10. If this gate drives three other gates with input loading factors of 2, 2, and 3 respectively, we can check to see if the load capability of the driving gate has been exceeded by simply adding the three input loading factors:

$$2 + 2 + 3 = 7$$

Since seven is less than ten, the load is acceptable.

AC Loading

In the example of Fig. 3-5 the load is assumed to be a pure resistance. In actuality there is always some capacitance to ground at a gate input, as well as stray circuit capacitance from each conductor to ground. These capacitances do not affect steady-state voltages, but they do slow down the transition speed. Some bistable elements (flip-flops) require a fast transition time in order to change state. Too much capacitance at a driving gate output will prevent these bistable elements from "triggering." *Ac loading* refers to limits imposed by gate input capacitances on the ability of a gate to drive compatible bistable devices. Usually these capacitances are taken into consideration in the gate design and in the specifications, so that the dc and ac load factors or fan-outs are the same.

Buffers, Drivers, Expanders, and Translators

Certain special gate designs are intended to extend the capabilities of the more standard gates, either by increasing fan-out where many gates must be driven simultaneously, or by increasing the number of inputs to a gate. *Buffers* and *drivers* increase fan-out; *expanders* increase fan-in, the number of inputs.

A buffer, or driver, will usually be capable of driving from three to ten times as many gates as a standard gate in the same compatible family. In order to maintain the same propagation time as the rest of the family, current consumption is increased. This gives the system designer the most flexibility in keeping system current at a minimum: he can select gates drawing low supply currents where fan-out is low, and where necessary for high fan-out, he can choose a driver gate.

Buffers and drivers usually perform a logic function at the same time as they provide high fan-out. They may be any of the five standard logic-type gates—AND, OR, NAND, NOR, and NOT.

We have seen that the usual number of inputs to standard IC gates is two, three, four, or eight. Inputs can be increased beyond eight by using an *expandable gate* and an *expander*. This method may also be used to make several gates having five, six, or seven inputs using a minimum of integrated-circuit packages.

The logic principle of the expander is illustrated in Fig. 3-7. Gate 1 has an output of 0 at Y unless A, B, C, and X are all 1's. But X is a 1 only if D, E, F, and G are 1's. Therefore, Y is 1 only if A through G (seven inputs) are all 1's. Notice that the overall propagation delay of the four inputs to gate 2 is longer than for the inputs to gate 1 in this case.

There are several ways to accomplish the function illustrated in Fig. 3-7. These include (1) diode logic at the gate input, (2) collector logic at the gate output, and (3) use of more than one gate, as illustrated in Fig. 3-7. The details of the first two techniques will become clearer after you read later chapters in this book.

Besides being an alternate term for "driver," the word "buffer" is sometimes used with, or as a synonym for, the term *level translator*.* Level translation is required where gate families are intermixed, when logic voltage levels are not the same for both families. For example, ECL (emitter-coupled logic) gates require considerably different voltages than TTL (transistor-transistor logic) for the 0 and 1 states. If a designer wishes to use both types of logic elements in the same system, he must use a level translator at their *interfaces*—places in the system where an ECL gate drives a TTL gate, or vice versa.

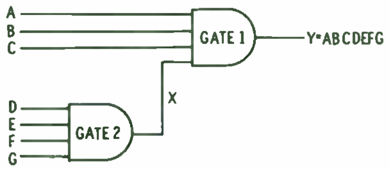


Fig. 3-7. Principle of an expander.

Speed And Propagation Time

Although gates operate very quickly, they nevertheless do require a finite nonzero time to change state. The propagation or delay time is the time it takes for a change at the input to produce a change of state at the output. The term may be applied to a series of gates, but for the present we are interested in one gate at a time.

The delay through a gate may range from over a microsecond (a millionth of a second) to only a few nanoseconds. A nanosecond is one billionth or 10^{-9} second. Delays of 10 to 50 nanoseconds are common for many of the modern, more popular IC gates. A *fast* or *high-speed gate* has a short propagation delay. The speed of a gate depends mostly on its internal design, but external factors such as loading, supply voltage, and temperature also affect speed to some degree. Some circuit types are inherently faster than others; for example, TTL (transistor-transistor logic) gates are generally considerably faster than RTL gates. However, within each logic family (RTL, TTL, etc.) there is still room for a range of speeds.

*Terminology is sometimes confusing. "Translator" can also mean a group of gates which change one set of binary codes into another. "Buffer" has sometimes been used as an alternate name for an OR gate.

Almost universally, higher speed is obtained at the expense of higher current drain from the power supply. This occurs because delays are caused in part by small unwanted but unavoidable capacitances which must be charged and discharged. To speed charging and discharging, resistor values must be reduced. For the same voltage levels, smaller resistances mean larger currents.

Additional information on the speeds of various gate types appears in a later chapter.

Noise

"Noise" is a general term referring to many kinds of unwanted signals and disturbances. In systems of IC gates, noise may occur as slowly or rapidly changing levels in power supplies, or as capacitively coupled electrical transients originating externally to the circuit under consideration. Excessive noise causes gates to change state when they are not supposed to.

Slowly changing power-supply voltages result from temperature, load, and line-voltage changes. These are reduced to small proportions by using well-regulated, temperature-compensated power supplies. However, economics requires that the gates not demand too sophisticated a power supply. In general, modern IC gates will work quite satisfactorily in most applications with a supply voltage held within 5 percent of the nominal voltage specified by the gate manufacturer.

Fast-changing power-supply voltage transients are usually caused by switching transients in the logic system, although with some regulated power supplies, line transients may pass through the supply and become a factor. Most types of gates require a sudden change in power-supply current when changing states. This current change causes voltage transients at the power-supply terminals. In addition, it causes voltage transients between points along the path from the supply to the device, due to the resistances and inductances of the conductive paths (wires or printed-circuit paths). A low output impedance power supply is helpful, but it is often necessary to use low-inductance bypass capacitors with short leads at crucial points along the supply "busses." These serve as tiny reservoirs of energy which fulfill the switching transient current requirements.

An external voltage pulse may be capacitively coupled into a gate circuit, especially from noisy conductors running parallel to critical gate leads. This situation is symbolized by the circuit in Fig. 3-8. The amount of capacitive coupling and the resistances of the generating circuit and the circuit into which the pulses are introduced determine the duration and magnitude of the pulses.

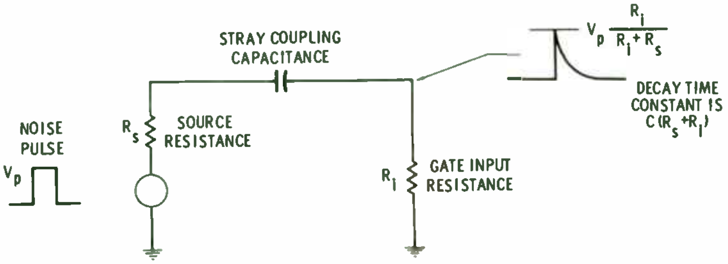
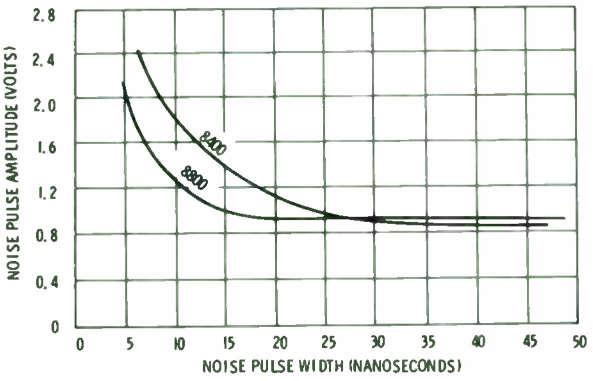

STRAY COUPLING CAPACITANCE
NOISE PULSE
V_p
R_s
SOURCE RESISTANCE
R_i
GATE INPUT RESISTANCE
$V_p \frac{R_i}{R_i + R_s}$
DECAY TIME CONSTANT IS $C(R_s + R_i)$

Fig. 3-8. Equivalent circuit for capacitively coupled noise pulses.

Faster gates tend to be more susceptible to noise. They will momentarily change state on even a very short transient voltage. Gates with low output impedances are less susceptible to noise than those with higher output impedances, since pulses capacitively coupled into the input of one gate will be more or less short-circuited to ground by the low output impedance of the driving gate.



Courtesy Signetics Corp.

Fig. 3-9. Minimum noise-pulse amplitude and duration causing gate operation.

The curve of Fig. 3-9 is typical of the relation between the amplitude and duration of pulses causing a gate to operate. The 8400 gate is of the diode-transistor logic (DTL) type and is slower than the 8800 TTL gate. For pulses of the same amplitude the slower gate requires longer pulses for false operation. Pulses of any duration below 0.8 volt do not cause operation.

The dc noise margin is related to the width of the "no man's land" between the 0 and 1 state voltage ranges. In Fig. 3-2 this width is approximately 1 volt, but this will, of course, vary from unit to unit and for temperature and supply voltage variations. Manufacturers allow a margin between the specified limits of the 0 and 1 states and the actual measured voltages which define the ends of the linear portion of the curve. Thus in Fig. 3-2, if the curve represents the worst case, the manufacturer might specify that

logical 0 = 0 to 0.5 volt,

logical 1 = 2.2 to 4.0 volts.

These specifications would allow for a worst-case noise margin of 0.2 volt. If an input signal of 2.2 volts is applied, it would act as a logical 1, and a down-going noise pulse of 0.2 volt would not cause momentary gate operation because even in the worst case it would lower the input only to the edge of the linear region.

Noise is generally a more severe problem in systems containing bistable elements (flip-flops) as well as gates. Whereas a gate will return to its original state following a noise transient, bistable elements will not. Thus the effect of a noise transient on flip-flops is more permanent and more likely to disrupt system operation.

Power Consumption

The power consumption of a gate is the product of the supply voltage and the current the gate draws. The total power consumption of all gates in a system determines the rating of the system power supply.

The supply current in most gates depends on the output state, and fan-out is also a factor. Many kinds of gates draw extra current while changing state, so that the average supply current then depends on how often the gates change state during an interval of time—that is, the frequency of operation. Some manufacturers specify a guaranteed maximum current consumption in order to simplify estimating how much current the power supply must provide. This will almost always result in selection of a supply capable of delivering more current than will ever be demanded, but the excess capacity serves as a good safety margin.

Power consumption must also be considered with respect to cooling requirements and temperature rises in different parts of the system.

Worst-Case Specifications

As mentioned earlier, curves such as those in Figs. 3-2 and 3-6 vary from unit to unit and with temperature and supply voltage changes. Temperature affects the 0 and 1 state voltage ranges, by changing base-emitter voltage drops, collector saturation voltages, transistor gains, and leakage currents. When a manufacturer publishes the 0 and 1 voltage ranges for a particular gate, an allowance has been made for the "worst case" of unit-to-unit variations, supply voltage, loading, and temperature within the specified limits of each. Load factors are also specified for the worst-case combinations of these factors.

You can see that by using standard integrated circuits you take advantage of an extensive analysis and design effort by the engineers who designed the gates. In effect, you have the assistance of thousands of engineers handling the details of your system design. This leaves you free to concentrate on the broader design aspects and lets you design more complex systems for the same expended effort.

Chapter 4

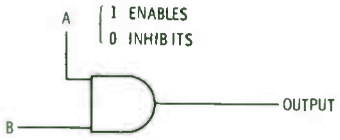
Gate Combinations

A single gate is rarely used alone. Usually two or more gates act in combination to perform a logic function which one gate alone cannot do.

Logic-Circuit Design

When you analyze or design logic circuits, you can take either of two approaches. One approach is to have a "library" of circuits—a cookbook, so to speak. If you come across an unfamiliar combination of gates, or if you want to design some logic function and do not immediately see the right combination of gates for the job, you can go to your "library" and look it up. The alternate approach is to use the tools to which you have already been introduced—truth tables and Boolean algebra. With these tools you can determine how a system works or you can create one to meet your own requirements.

A complete library of circuits is not practical. There are too many ways to accomplish too many logic functions. It is clear that we cannot forego a good working knowledge of the tools of design and analysis. However, there are some simple combinations of only a few gates that appear so often that it is wise to learn to recognize these at sight, or to be able to call on them from memory when designing a new system. Many complex systems are simply groups of small subsystems, each consisting of several gates in simple basic combinations. In this chapter we will examine some of these basic combinations.



(A) Diagram.

A	B	OUTPUT = AB
0	0	0
0	1	0
1	0	0
1	1	1

OUTPUT = 0 WHEN A = 0

OUTPUT = B WHEN A = 1

(B) Truth table.

Fig. 4-1. AND gate used as enable/inhibit gate.

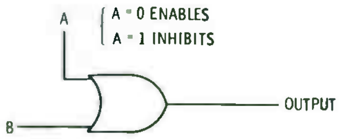
The Inhibit or Enable/Disable Gate

In its simplest form the *inhibit*, or *enable/disable* gate is a single gate seen from a particular point of view. Any of the four logic gates standing alone can be used for the purpose. Gates may also be used in combination.

Consider the AND gate and its truth table, in Fig. 4-1. Notice that if $A = 1$ the output is equal to B. We could say that making $A = 1$ *enables* the gate to pass the signal at B on to the next circuit. Conversely, if we make $A = 0$, the gate is *inhibited* or *disabled*, since the signal at B does not pass and the output is always 0.

The OR gate in Fig. 4-2 will do the same thing, except that $A = 0$ enables the gate while $A = 1$ disables it. There is the further difference that when the gate is disabled, the output is always 1.

Since NAND and NOR gates are logically equivalent to AND and OR gates, respectively, followed by inverters, they also may be used as enable/disable gates. If the inverted output is not suitable for the remainder of the logic system, an inverter may be placed after the gate.



(A) Diagram.

A	B	OUTPUT = A+B
0	0	0
0	1	1
1	0	1
1	1	1

OUTPUT = B WHEN A = 0

(B) Truth table.

Fig. 4-2. OR gate used as enable/inhibit gate.

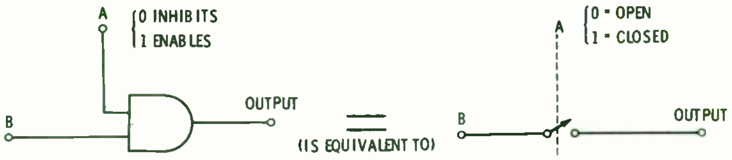
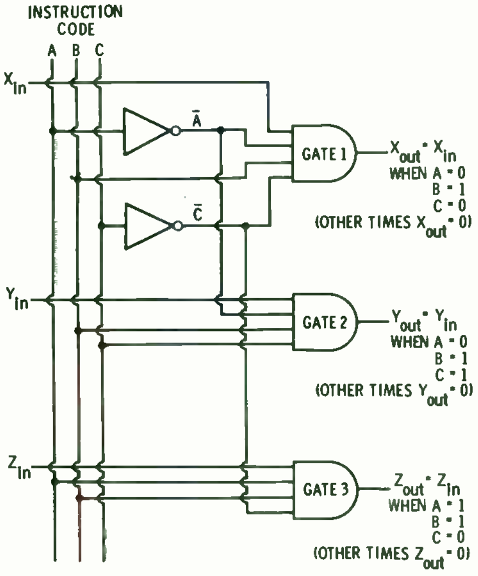


Fig. 4-3. Equivalence of AND gate to spst switch.

You can see that the enable/disable gate is equivalent to an electronically operated single-pole, single-throw switch (Fig. 4-3). The *A* input of the AND gate corresponds to the manually operated position of the switch handle. The *B* input is like the absence or presence of an input voltage (0 or 1).

Gates with more than two inputs can be used as inhibit/enable gates which respond to a multiple-bit coded input. For example, in Fig. 4-4, assume that the voltages on the three wires *A*, *B*, and *C* change every second in a sequence given by the table. Using inverters as necessary, gates 1, 2, and 3 can be made to "enable" their input signals X_{in}, Y_{in}, and Z_{in} to pass through their respective gates only at certain times. As shown, 2 seconds after the sequence begins, X_{in} will pass through to X_{out}. Three seconds after the beginning X_{in} is inhibited again, and gate 2 becomes enabled instead, so that Y_{in} will pass through. Another second later all gates become inhibited and



(A) Diagram.

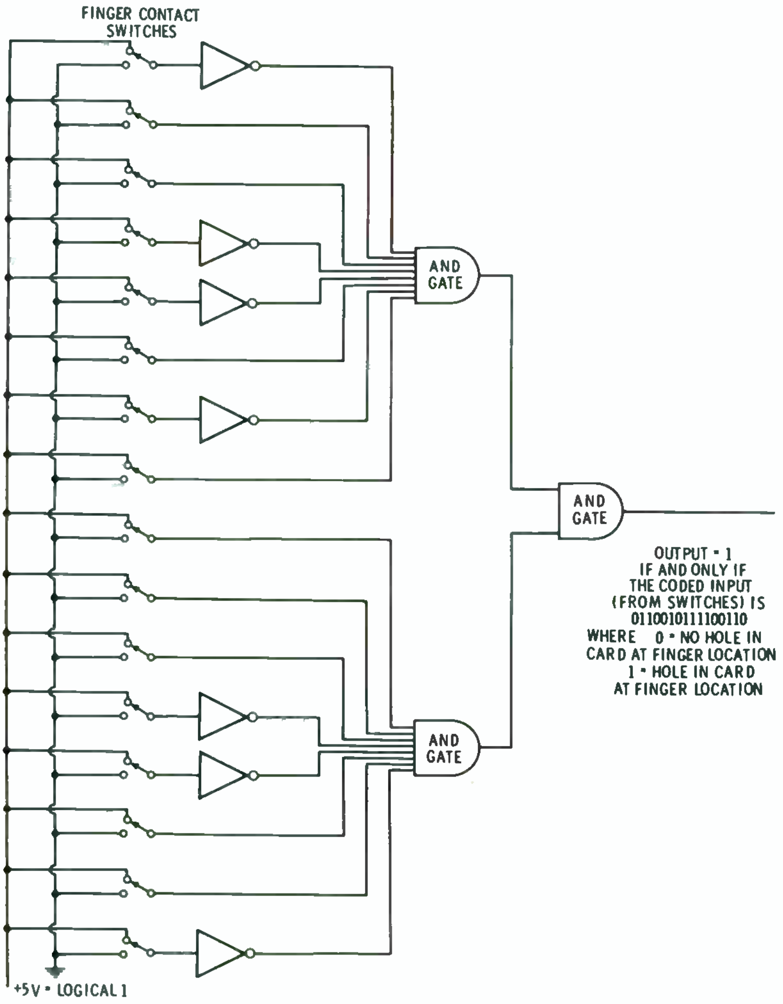
TIME (SECONDS)	BINARY CODE		
	A	B	C
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0
	ETC., REPEATS		

(B) Sequence.

Fig. 4-4. Logic subsystem which passes three signals according to coded instructions.

remain so until 6 seconds have elapsed from the starting time, whereupon the third gate is enabled for 1 second. The combinations of voltages on wires *A*, *B*, and *C* can be thought of as coded instructions to each gate, "telling" the gate whether to pass or inhibit a signal.

Gate combinations like these are used extensively in logic systems where a series of logic functions are to be performed in sequence, such as in digital computers. The binary instruction code goes through a sequence selected by the system designers. Binary signals can then be routed from one part of



FINGER CONTACT SWITCHES
AND GATE
AND GATE
AND GATE
OUTPUT = 1 IF AND ONLY IF THE CODED INPUT (FROM SWITCHES) IS 0110010111100110 WHERE 0 = NO HOLE IN CARD AT FINGER LOCATION 1 = HOLE IN CARD AT FINGER LOCATION
+5V = LOGICAL 1

Fig. 4-5. Circuit producing logical 1 output for one 16-bit code.

the system to another, at a preselected time following initiation of the sequence.

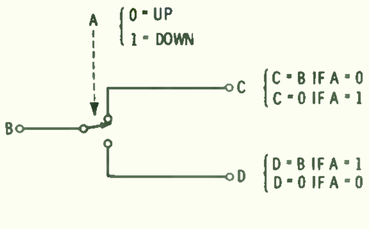
Fig. 4-5 shows another possible application of a multiple-bit coded gate. Cards with holes either punched or omitted at certain locations on their surfaces are dropped into a slot. The absence or presence of holes is detected by a series of tiny "fingers" acting to close switches when they fall into holes.

A particular combination of open and closed switches will cause each of the eight-input gates to have a 1 output. For inputs with no inverter, logical 1 signals must be applied; for inputs with an inverter, a logical 0 at each inverter input will produce the necessary 1 at the eight-input gate input terminal. When both eight-input gates have all 1 inputs, both will also have 1 outputs. These two 1 signals applied to the final two-input gate will cause the output to be a logical 1. There is only one input combination out of $2^{16} = 65{,}536$ possible different input combinations which will produce a 1 output. This output could be used to unlock a door or give some signal, or to operate a relay and cause some mechanical action to occur.

Transfer Circuits

We have seen how gates may be equivalent to a single-pole, single-throw switch. Gates in combination may also be equivalent to transfer circuits—switches with more than one pole. In Fig. 4-6 one such circuit is shown, along with a truth table defining its logical operation. Note that there are two outputs, *C* and *D*. There are also two inputs: *B*, the signal (which may be 0 or 1), and *A*, the switch position.

Fig. 4-7 shows two AND gates and an inverter connected to perform the same logic. This logic circuit directs one signal (*B*) along either of two paths. Stated another way, it *transfers* an incoming binary signal to either line *C* or line *D*. Instead of



INPUTS		OUTPUTS	
A	B	C	D
0	0	0	0
0	1	1	0
1	0	0	0
1	1	0	1

BOOLEAN EQUATIONS: $C = \bar{A}B$ $D = AB$

(A) Circuit. (B) Truth table.

Fig. 4-6. Transfer switch (spdt) and truth table.

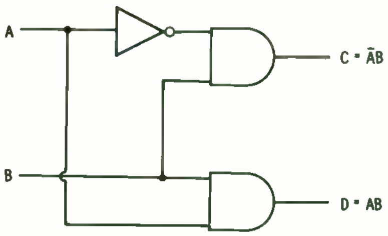
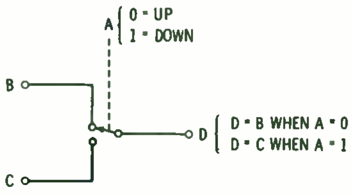


Fig. 4-7. Gates connected as spdt switch.

being manually operated, as is a toggle switch, it is electronically operated by the application of a 0 or 1 voltage on line *A*.

Fig. 4-8 shows the same double-pole switch turned around so that the output receives a signal from either of two sources. There are really three inputs: *A*, *B*, and *C*. Therefore, the



(A) Circuit.

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$D = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC$
$= AC + \bar{A}B$

(B) Truth table.

Fig. 4-8. Two-pole selector switch with truth table and Boolean expressions.

truth table has eight lines. The Boolean expression can be written from the truth table by the method described in chapter 2, and then simplified by the rules of Boolean algebra. From the simplified expression, the circuit in Fig. 4-9 is drawn.

In either the signal-selecting or signal-directing transfer circuits, there may be more than two poles. However, it then is

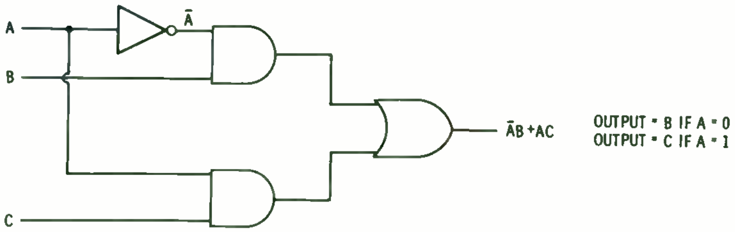
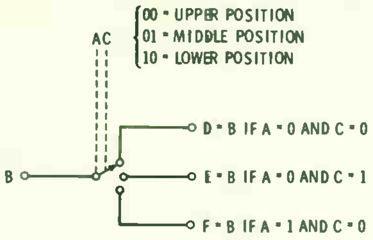


Fig. 4-9. Gates connected as two-pole selector switch.



(A) Circuit.

A	B	C	D	E	F
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	NOT USED		
1	1	0	0	0	1
1	1	1	NOT USED		

$D = \bar{A}B\bar{C}$ $E = \bar{A}BC$ $F = AB\bar{C}$

(B) Truth table.

Fig. 4-10. Three switch positions represented by two-bit code.

no longer possible to represent the switch position with one binary digit. A longer instruction code is required. In Fig. 4-10 a two-bit code is used to represent the three positions of a three-pole transfer circuit of the signal-directing type.

The equivalent logic circuit is shown in Fig. 4-11. It is derived from the Boolean equations written from the truth table in Fig. 4-10. By adding one more three-input gate this circuit is readily converted to a four-pole transfer switch. The instruction code for the fourth pole (switch position) is $A = 1$ and $C = 1$.

If the number of poles of the signal-directing transfer circuit exceeds four, the two-bit instruction code is no longer sufficient, as there are only four different code combinations possi-

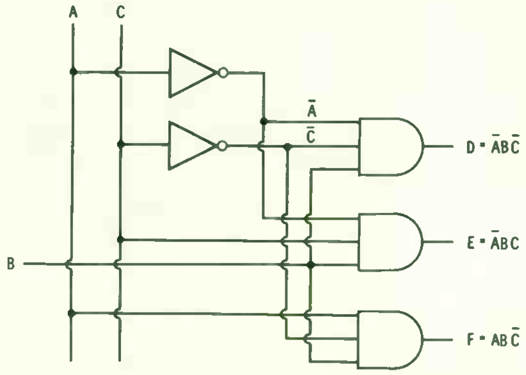
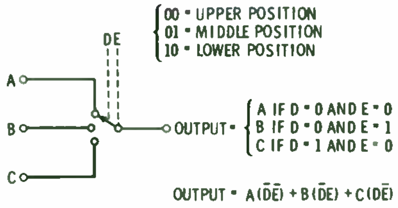


Fig. 4-11. Three-pole transfer switch using electronic gates.

ble from a two-bit code. In general, the maximum number of poles will be 2^n, where n is the number of binary digits (bits) in the instruction code. For five to eight poles, $n = 3$ and each gate must have four inputs. The number of inputs required per gate is $n + 1$. Up to n inverters will also be required.



00 = UPPER POSITION
01 = MIDDLE POSITION
10 = LOWER POSITION
DE
A
B
C
OUTPUT = A IF D = 0 AND E = 0; B IF D = 0 AND E = 1; C IF D = 1 AND E = 0
OUTPUT = $A(\bar{D}\bar{E}) + B(\bar{D}E) + C(D\bar{E})$

Fig. 4-12. Three-pole selector switch with coded positions.

For the signal-selector switch with more than two poles, the size of the truth table becomes quite unwieldy. In Fig. 4-12 the manually operated equivalent is shown for three inputs: A, B, and C. A two-bit instruction code of D and E is required. Since altogether there are five inputs (A, B, C, D, and E) the truth table would require 32 lines. However, for this kind of "orderly" logic function we can take a short-cut and write the logic expression directly from the word description of the output. It is helpful if we redraw the switch diagram (Fig. 4-13), showing the three-pole switch replaced by three single-pole switches in parallel, each enabled by a unique instruction code. The parallel paths indicate that three OR'ed terms are required in the Boolean expression. Each term is the product of the signal logic value (0 or 1) and the switch instruction code:

$$\text{Output} = A(\bar{D}\bar{E}) + B(\bar{D}E) + C(D\bar{E})$$

Study the expression and you will see that when evaluated for each instruction code, only one term can be a logical 1 for any code. This term will be either 0 or 1 depending on the value of its input. For example, when $D = 0$ and $E = 0$ the second and third terms must both be 0, and the first term will be 0 if A is 0, or 1 if A is 1.

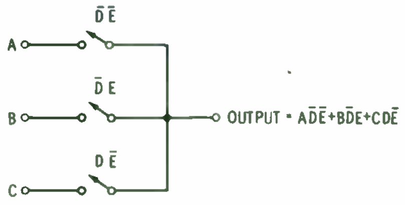


Fig. 4-13. Three-pole switch redrawn.

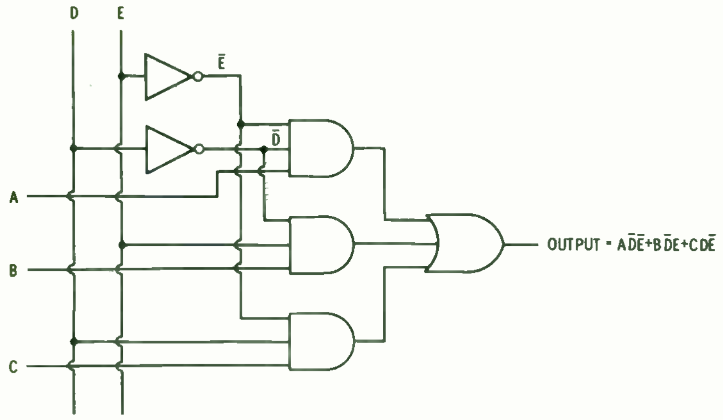


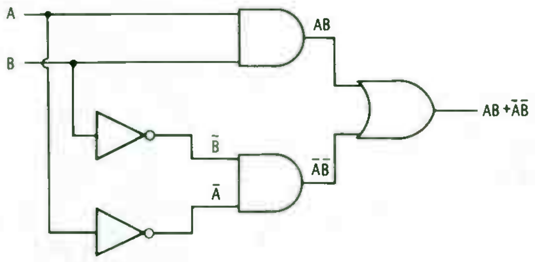
Fig. 4-14. Three-pole selector switch using gates.

The equivalent logic circuit using gates, drawn directly from the Boolean expression, is that of Fig. 4-14. One gate is required for each input (pole). Each gate must have $n + 1$ inputs, and the largest number of poles is 2^n as before, n being the number of bits in the instruction code. Up to n inverters are required. Finally, an OR gate, with as many inputs as there are poles, is required.

In all of the foregoing switch circuits each circuit was first shown with familiar mechanical contact switches. Then an equivalent logic circuit using electronic gates was developed. You should note that while the gate circuits are *logically equivalent* to the corresponding circuits using mechanical switches, they are not *electrically equivalent*. For example, for mechanical switches, when a switch is closed its output and input voltages are equal, due to the direct connection between them. For gates this is not true—both voltages need merely be within the acceptable voltage ranges for 0 or 1 logic levels. Another difference is that with mechanical switches the output impedance of the circuit is infinite when the switch is open and is equal to the source resistance of the voltage source when the switch is closed. For gates, the output impedance is a function of the gate design and is nearly independent of the source impedance of the input logic signal.

The Comparator and the Exclusive OR

The *comparator* circuit detects whether or not two or more logic signals have the same value—either 0's or 1's. The truth table and logic diagram for a two-bit comparator are shown in



(A) Diagram.

A	B	OUTPUT
0	0	1
0	1	0
1	0	0
1	1	1

OUTPUT = 1 IF A = B
= $AB+\bar{A}\bar{B}$

(B) Truth table.

Fig. 4-15. Two-bit comparator.

Fig. 4-15. Despite the apparent simplicity of the truth table, five gates are required (for this purpose we classify inverters as gates). The output is 1 if $A = B$; otherwise the output is 0.

More sophisticated comparators have added logic elements with two additional outputs, one having a 1 output if A is greater than B and the other having a 1 output is B is greater than A. Try adding the necessary circuitry to Fig. 4-15 yourself. Only two more AND gates are required.

Comparators can be constructed for more than two bits. One inverter is needed for each bit, and the AND gates must have as many inputs as there are bits. One OR gate is all that is needed.

Do yourself: Draw a logic diagram for a three-bit comparator. (Answer: See Fig. 4-16.)

The exclusive-OR circuit has a 1 output if A or B is 1, but not if both are 1's. The truth table and logic diagram are shown in Fig. 4-17. Note the similarity to the two-bit comparator—both require the same number and types of gates. In fact, either can be made from the other by a slight wiring change, or by adding an inverter at the output.

One application of the comparator or exclusive-OR circuit which illustrates the narrowing gap between digital and analog

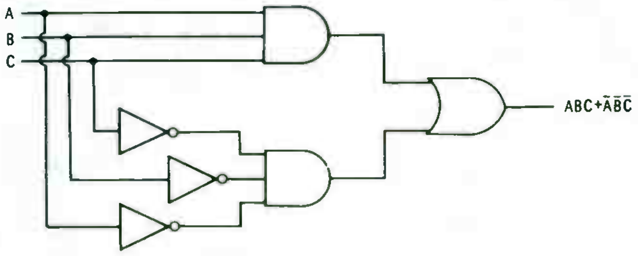
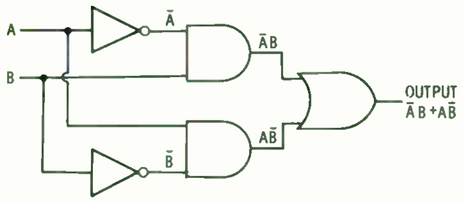
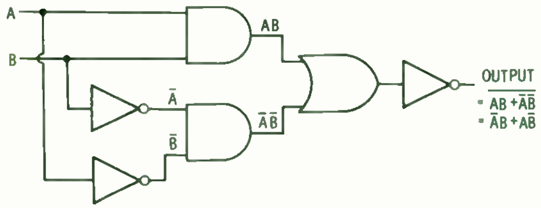


Fig. 4-16. Three-bit comparator.



A	B	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

(A) Direct method.



NOTE:

$$\overline{AB+\bar{A}\bar{B}} = \overline{AB}\cdot\overline{\bar{A}\bar{B}} = (\bar{A}+\bar{B})\cdot(A+B)$$

(BY DE MORGAN'S LAWS)

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

(LAW OF DISTRIBUTION)

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

(SINCE $\bar{A}A = 0$ AND $B\bar{B} = 0$)

(B) Using comparator followed by inverter.

Fig. 4-17. Two methods of performing the exclusive-OR function.

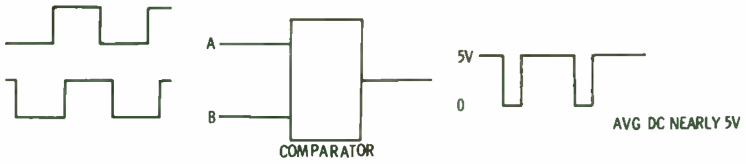
(linear) circuitry is the phase detector for square waveforms, shown in Fig. 4-18. Although a comparator is shown, the exclusive-OR circuit works equally well. For the comparator, maximum average dc output occurs when the *A* and *B* inputs are exactly in phase, and therefore always equal. When the two input signals are 180° out of phase, they are never equal and the output is nearly 0 volts. The average dc output is recovered by low-pass filters following the output.

Translators, Encoders, and Decoders

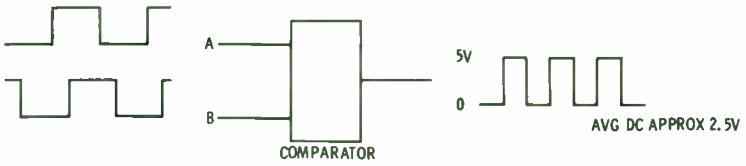
Binary codes represent a kind of language. Just as letters of the alphabet can be combined to form different words, so strings of binary digits can form binary "words" with distinct meanings when used in a logic system. But logic systems, like people, can speak different languages. It sometimes becomes necessary to "translate" one set of codes into another set. A circuit for doing this is called a *translator.**

In the broadest sense, every logic circuit is a translator. For every input code (combination of binary inputs) there is a cor-

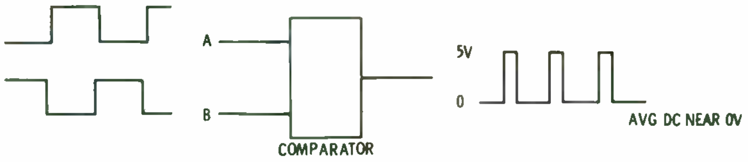
*Many logic terms have more than one meaning. "Translator" can also refer to a circuit which permits an element from one logic family to drive an element of another family—for example, an RTL gate to drive a TTL gate.



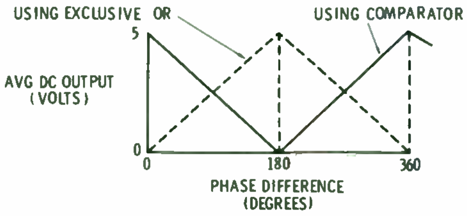
(A) *B* lags *A* by small amount.



(B) *B* lags *A* by 90°.



(C) *B* lags *A* by almost 180°.

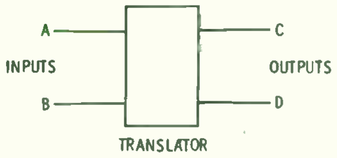


(D) Input-output relation.

Fig. 4-18. Phase detector using comparator.

responding output code. Thus "translator" is a name which distinguishes the purpose of the logic circuit rather than its construction or logic diagram. Encoders and decoders are special forms of translators.

Fig. 4-19 shows a translator represented by a box with two inputs and two outputs. A truth table is given which defines the translation. Note that the logic function, or translation, is defined for only three of the four possible inputs. This implies that the fourth possible input combination is not pertinent to the problem and can be anything at all. The design of a logic



(A) Diagram.

INPUTS		OUTPUTS	
A	B	C	D
0	1	1	1
1	0	0	1
1	1	0	0

$C = \bar{A}$ $\quad$ $D = \bar{A}B + A\bar{B}$

(B) Truth table.

Fig. 4-19. Translator block diagram and truth table.

circuit to perform this function is straightforward. One such design is given in Fig. 4-20; this particular circuit would translate the "unused" input of $A = 0$, $B = 0$ into $C = 1$, $D = 0$.

The longer the codes, the more complex the translator may be, although long codes do not require complex circuits in every case. Table 4-1 gives a truth table for translating a four-bit binary code into a binary-coded-decimal code. The simplified Boolean equations are below the table. Some of these equations

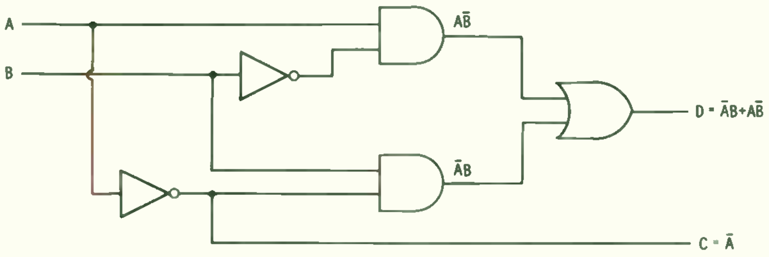


Fig. 4-20. Logic diagram for simple translator.

can be written directly by studying the table; for example, it is clear simply by close examination of the table that $I = D$ and $F = A\overline{B}\overline{C}$. A logic circuit drawn from the Boolean equations is shown in Fig. 4-21.

The binary to binary-coded-decimal translator illustrates that the number of inputs and outputs do not have to be the same. In s translator, notice that since there are five outputs there are 32 possible output codes, but only 16 are used. However, the operation of this translator is defined for all sixteen possible inputs.

The "read-only-memory," or ROM, is a translator. The input is called the *address*, and outputs which result for each address are referred to as the binary *words* stored in the memory. The

Table 4-1. Truth Table and Boolean Expressions for Binary-to-BCD translator

Decimal	Binary A	B	C	D	Binary-Coded Decimal E	F	G	H	I
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	1
2	0	0	1	0	0	0	0	1	0
3	0	0	1	1	0	0	0	1	1
4	0	1	0	0	0	0	1	0	0
5	0	1	0	1	0	0	1	0	1
6	0	1	1	0	0	0	1	1	0
7	0	1	1	1	0	0	1	1	1
8	1	0	0	0	0	1	0	0	0
9	1	0	0	1	0	1	0	0	1
10	1	0	1	0	1	0	0	0	0
11	1	0	1	1	1	0	0	0	1
12	1	1	0	0	1	0	0	1	0
13	1	1	0	1	1	0	0	1	1
14	1	1	1	0	1	0	1	0	0
15	1	1	1	1	1	0	1	0	1

$$\begin{aligned} I &= D \\ H &= \bar{A}C+A\bar{B}C \\ G &= \bar{A}B+ABC = B(\bar{A}+AC) \\ F &= A\bar{B}\bar{C} \\ E &= A(B+C) \end{aligned}$$

stored words are built-in; they are due to the way the gates are connected.

The logic circuit represented by Figs. 4-19 and 4-20 could be called a simple read-only-memory. The address consists of the two-bit code applied to A and B. The circuit contains four two-bit words in the memory. Three of these are defined by the table in Fig. 4-19. For example, if the address code is $A = 0$, $B = 1$, then the output will be the two-bit word $C = 1$, $D = 1$. In short:

Address	*Memory Output*
01	11
10	01
11	00
00	10

The last line applies to the circuit of Fig. 4-20.

The words stored in the ROM cannot be changed without rewiring the gate connections. This is the significance of the phrase "read-only," as most memories have both "read" and

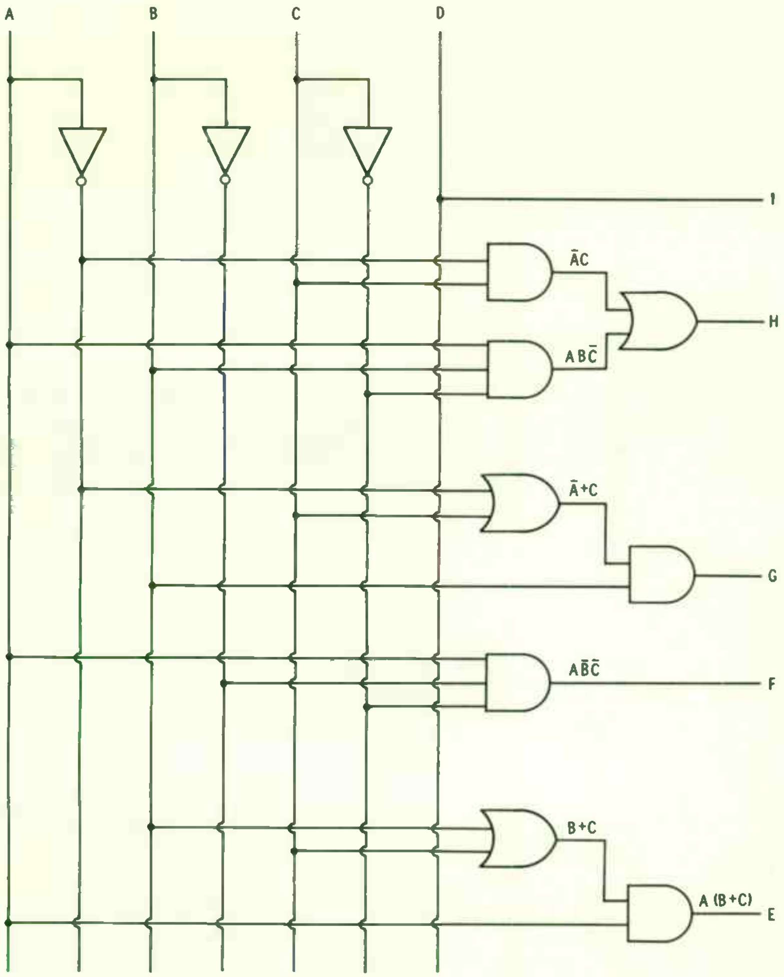
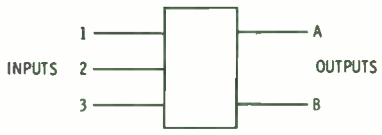


Fig. 4-21. Logic diagram for binary-to-BCD translator.

"write" capabilities, whereby the stored words may be changed or rewritten by applying binary electrical signals. Despite this inflexibility, in some applications the ROM has distinct advantages over other types of memories. It is a permanent memory which is not affected by use (reading out words). The memorized words are not lost if the power supply is disconnected, nor can the words be accidentally erased.

Encoders belong to a special class of translators where the translation is defined only for certain input combinations. For example, in Fig. 4-22, a three-input encoder is shown, along



(A) Diagram.

INPUTS			OUTPUTS	
1	2	3	A	B
0	0	0	0	0
1	0	0	0	1
0	1	0	1	0
0	0	1	1	1

B = 1+3 A = 2+3

(B) Table and expressions.

Fig. 4-22. Block diagram, truth table, and Boolean expressions for simple encoder.

with the truth table. Notice that the four defined input codes are those for which either all inputs are 0 or else a 1 is applied to only one input at a time. The outputs in this case are the first four binary numbers, including zero. This encoder could be used to translate the outputs of three switches, called number one, number two, and number three, into the equivalent binary number. When no switch is operated, the output is zero in the binary number system; when switch number one is operated, the output is the binary number 1, etc. It is assumed that only one switch will be operated at a time. A logic circuit to accomplish this is shown in Fig. 4-23.

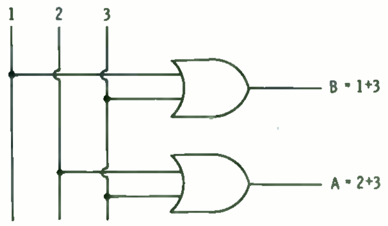
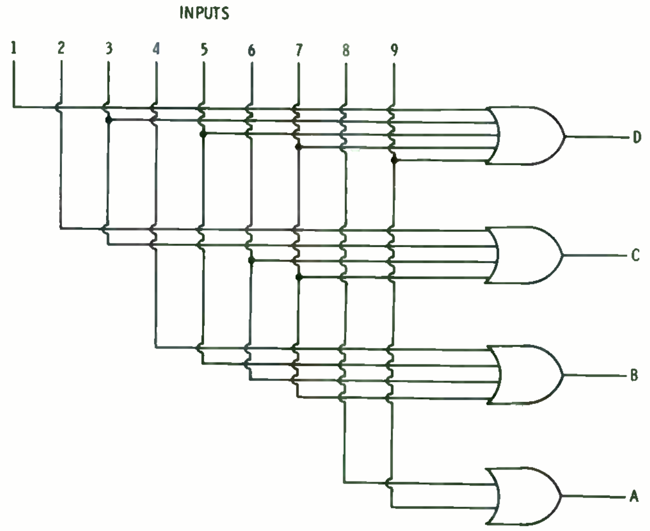


Fig. 4-23. Logic diagram for simple encoder.

A more common encoder has nine inputs and four outputs, and converts decimal numbers (represented by a 1 on one of the nine input wires, or all zeros for a decimal zero) into binary or BCD numbers. Fig. 4-24 shows a logic circuit which will do this.

Decoders are the "opposites" of encoders. Only one output terminal at a time can have a logical 1. Refer to Fig. 4-25. The truth table defines the logic and the logic diagram shows one way to connect nine four-input AND gates to accomplish the desired result. Four inverters are also required. Circuits like these are commonly used to energize indicator lamps labeled 1 through 9 to "read out" a BCD code. Another gate could be added to operate a "zero" lamp. This way, one lamp should be energized at all times, eliminating confusion between the zero code and the the condition where all power is removed and the circuits are not operating.



OUTPUT = $\boxed{A}\boxed{B}\boxed{C}\boxed{D}$, A 4-BIT BINARY NUMBER EQUAL TO WIRE NUMBER ENERGIZED

EXAMPLE: ENERGIZE INPUT WIRE 6; OUTPUT = 0110

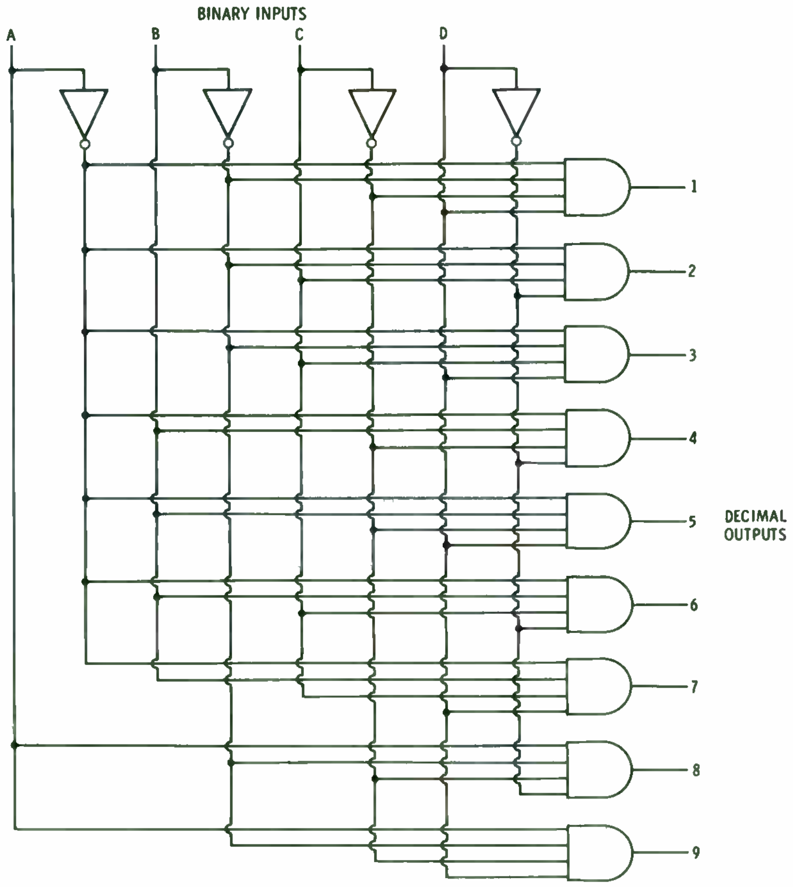
Fig. 4-24. Decimal-to-binary encoder.

MSI ROM's and PROM's

As integrated circuit technology developed, manufacturers developed methods of including more and more gates in a single package. Instead of one, two, or four gates in one package, it became practical to fabricate packages containing 50 or more gates. The terms "medium scale integration" (MSI) and "large scale integration" (LSI) were invented to describe these increasingly complex and more elaborate integrated circuits. By 1973, MSI devices were in widespread use, and LSI was growing rapidly.

One of the earliest uses of MSI was for read-only-memories (ROM's), already discussed. Larger memories have more address inputs and store more and longer words. They of course require more gates, but this is exactly what is available through MSI.

The capacity of the memory is expressed in bits, a quantity which is usually a power of two (16, 32, 63, 128, 256, etc.). The



(A) Diagram.

INPUTS				OUTPUTS								
A	B	C	D	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	1

(B) Truth table.

Fig. 4-25. Binary-to-decimal decoder.

number of words the memory stores is 2^n, where n is the number of address inputs. The memory size in bits is the product of the number of words times the number of bits in each word.

As an example, the Texas Instrument Type SN7488 is an MSI 256-bit read-only-memory. It is organized into 32 words of 8 bits each ($32 \times 8 = 256$). There are five address inputs. This memory is custom-programmed at the factory during manufacture, in accordance with instructions furnished by a particular customer. These instructions are in the form of a table, such as the one in Table 4-2. Notice that although different in size, the tables of Fig. 4-19 and Table 4-2 have the same form. The circuit of Fig. 4-19 is only an 8-bit ROM, organized as four 2-bit words.

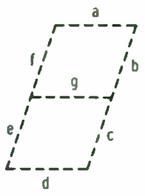
Certain ROM applications are so common that off-the-shelf preprogrammed models are available. One such application is

Table 4-2. Typical Truth Table for 256-Bit ROM

	Address					Word Outputs							
Word No.	A_1	A_2	A_3	A_4	A_5	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
1	0	0	0	0	0	0	0	1	1	0	1	0	1
2	0	0	0	0	1	0	0	0	1	0	0	0	1
3	0	0	0	1	0	0	1	1	0	0	0	1	1
4	0	0	0	1	1	0	1	1	1	0	1	0	0
etc.			etc.					etc. as required by customer					
32	1	1	1	1	1	0	0	1	0	1	1	0	0

the conversion of the BCD code to the seven-bit code commonly used to operate seven-segment numerical readouts. Fig. 4-26A shows the layout of the seven segments and the letters usually used to identify each segment. The digits 0 through 9 are formed by lighting combinations of two or more segments, according to the table in Fig. 4-26B. For example, to display a "3," the BCD code for 3, which is 0011, is applied to the four converter (ROM) inputs. The converter outputs appear on the seven output wires, each of which connects to one readout segment as indicated. If an output is a "1," the corresponding readout segment lights up.

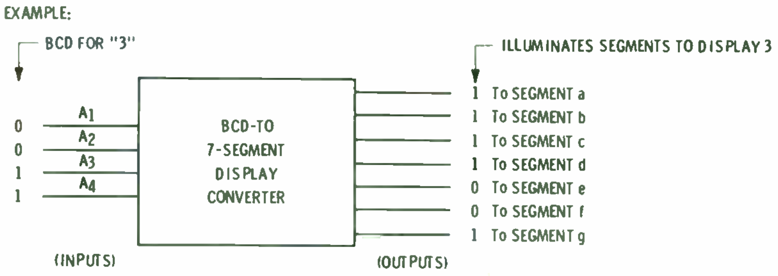
Ten words, each 7 bits long, must be stored in the memory. Therefore, a 70-bit ROM woud be sufficient. But since there must be four address inputs for BCD, a 128-bit memory organized into 16 words of 7 bits is generally used. The six extra words may be programmed to display the letter "E" (for "er-



(A) Seven segment layout.

TO DISPLAY DECIMAL DIGIT	BCD	ILLUMINATE SEGMENTS
0	0000	a-b-c-d-e-f
1	0001	b-c
2	0010	a-b-g-e-d
3	0011	a-b-g-c-d
4	0100	b-c-f-g
5	0101	a-f-g-c-d
6	0110	f-e-d-c-g
7	0111	a-b-c
8	1000	a-b-c-d-e-f-g
9	1001	a-b-c-f-g

(B) Table.



(C) Converter.

Fig. 4-26. ROM used as a converter from BCD to 7-segment readout code.

ror"), or to consist of all zeroes, which would leave the display blank.

For custom applications, factory programmed ROM's have certain disadvantages. One of these is the relatively long time required between preparing the instruction table and having the completed devices on hand. For smaller quantities, such ROM's are also relatively expensive, since manufacturers impose a fixed charge to cover the cost of custom design and setting up the manufacturing process. These disadvantages are largely overcome by the programmable-read-only-memory, or PROM.

PROM's are readily programmed by the user, starting with an off-the-shelf "blank" device. The mechanism varies by which a PROM is internally altered to store the desired words, but the procedure is similar for all types. The words are addressed one at a time, and while addressed, current pulses are applied to the output leads. The pulses change the internal circuit—one way is by melting tiny links—so that whenever that address is applied again, the previously impressed word bits will appear at the output terminals. This programming process can be done slowly, one bit at a time, by simple circuits described by the manufacturer, or much more quickly by more elaborate programming equipment. Some of the latter are also capable of copying another ROM in a matter of minutes.

Two examples of PROM's are the Type IM5600, made by Intersil, and the Type N8223, made by Signetics. Both store 32 words of 8 bits each, for 256 bits total capacity.

Using NAND and NOR Logic Elements

The examples in this chapter, up to this point, have used AND, OR, and NOT gates. In fact, however, NAND and NOR integrated-circuit gates have been used much more extensively in actual practice, and this pattern will probably continue for some time to come.

One reason NOR gates have been used widely is that the earliest IC gates were mostly of this type. The RTL NOR gate was introduced first because it was simple with few components—important factors when integrated circuits first began to be manufactured. Not long after RTL NOR gates were introduced, DTL (diode-transistor logic) gates became available. Although they cost a little more because they were more complex, they offered several advantages over the the RTL gates. They were most readily designed to function as NAND gates, and in this form came into widespread use.

TTL (transistor-transistor logic), the most popular family of IC logic for new designs, is an extension of DTL, and in its simplest form also results in a gate which performs the NAND function.

For several years many logic systems composed of integrated circuits consisted almost entirely of NOR or NAND gates because these were the kinds of IC gates that logic system designers could readily obtain "off the shelf." Although today the system designer has available a continually broadening variety of gates from which to choose, and is no longer limited to using only NOR and NAND gates, the latter are still the primary components in many logic systems.

Besides availability, another reason NAND and NOR gates are widely used is that they are in a sense "universal" gates. Any logic function can be realized using only NOR gates, or only NAND gates. They are the only logic types that have this characteristic. AND gates alone cannot be combined to perform many logic functions. Neither can OR gates. Even AND and OR gates used together require inverters as well to perform some logic functions. Notice how many of the examples given in this chapter use AND and OR gates, but also require inverters. Yet each of these functions could be obtained by the right combination of NOR or NAND gates.

Using just one kind of gate has certain advantages. Standardization reduces some costs by simplifying purchasing and inventory control. Another advantage is that since gates come several to a single package, left-over gates in a package in one subsystem may sometimes be put to use in another subsystem.

A disadvantage of using only NAND or NOR gates is that more gates are often required. More gates mean higher cost, more packages and space, more power and heat dissipation, and added propagation delays.

Even with the larger variety of gates now available, one trend in integrated circuits is certain to keep NAND gates in the forefront in system design. Today the trend is toward medium- and large-scale integration (MSI and LSI), which means more and more interconnected gates on a single semiconductor chip in a single package. In order to accommodate the tremendous variety of logic systems that are possible, with the minimum number of different IC designs, manufacturers are offering gate arrays. These are standardized single packages containing up to 25 or more gates which can be interconnected in an almost unlimited number of ways to form relatively complex logic systems. At least for the next few years these gate arrays will contain all of the same kind of gates, since this is easiest for the manufacturer to produce with minimum cost and maximum reliability. If only one kind of gate is to be used, the natural choices are NAND or NOR logic, as they are the only gates which can be connected to perform any logic function.

Some simple logic expressions are easily implemented by NAND gates if one only remembers that a NAND gate acts as an inverter if all its inputs are tied together. Then two NAND gates can be equivalent to one AND gate, as in Fig. 4-27.

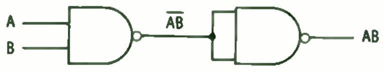


Fig. 4-27. AND gate made from two NAND gates.

Using this technique, the double-pole transfer circuit of Fig. 4-6 can be made from only NAND gates. The logic diagram is shown in Fig. 4-28. Compare this with Fig. 4-7 and you see that you pay a penalty of two additional gates if you restrict your design to using only NAND gates.

The logic designer needs ways to convert the Boolean expressions derived from truth tables into equivalent expressions which indicate directly how NAND or NOR gates can be used.

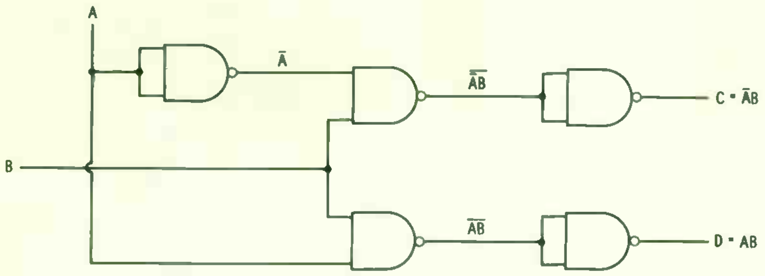


Fig. 4-28. Two-pole transfer switch using NAND gates.

The basic tools for this conversion are De Morgan's Laws, which are repeated below:

$$(1) \qquad \overline{AB} = \overline{A} + \overline{B}$$
$$(2) \qquad \overline{A + B} = \overline{A}\,\overline{B}$$

Notice that in both equations, one side contains an AND operation and the other side contains an OR operation. From the second equation, if we invert* both sides, the resulting pair of expressions are still equivalent; that is,

$$\overline{\overline{A + B}} = \overline{\overline{A}\,\overline{B}}$$

But remember that if a quantity is inverted twice, the original expression results, so that

$$\overline{\overline{A + B}} = A + B$$

Therefore

$$A + B = \overline{\overline{A}\,\overline{B}}$$

The right-hand side represents a NAND gate with two inverters, one at each input, as in Fig. 4-29. This combination is shown to be equivalent to an OR gate.

*In Boolean algebra *invert*, *complement*, and *negate* have the same meaning.

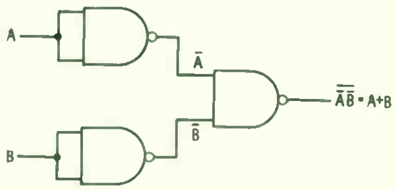


Fig. 4-29. OR function from NAND gates.

We have now seen how to use NAND gates to produce NOT, AND, and OR functions. The only remaining basic function is the NOR, and this is obtained by connecting an inverter at the output of an OR circuit. Fig. 4-30 summarizes all five basic functions, using only NAND gates.

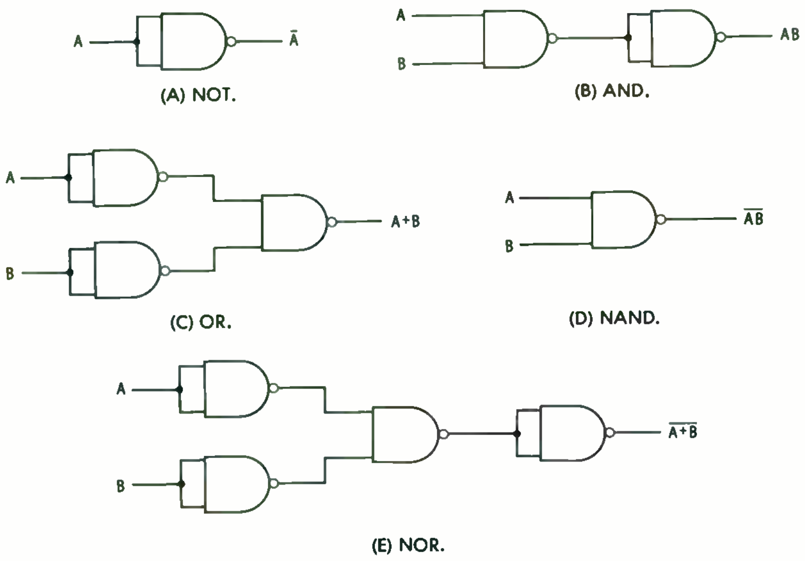


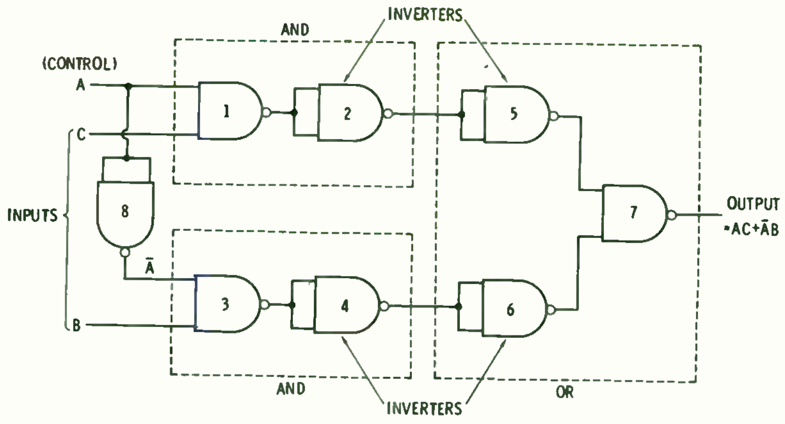
Fig. 4-30. Logic functions using only NAND gates.

Consider the transfer circuit of Fig. 4-8. The Boolean expression is $AC + \bar{A}B$. By selecting and combining two AND circuits and one OR circuit from Fig. 4-30, the logic circuit of Fig. 4-31A is constructed. While this circuit will perform the desired function, it turns out that in two places two pairs of inverters are connected in series. Since the second inverter changes the Boolean signal back to its state at the input to the first inverter, the two inverters may be eliminated. This results in the diagram of Fig. 4-31B. The latter contains four gates, exactly the same number used in the logic circuit of Fig. 4-9, which uses AND and OR gates!

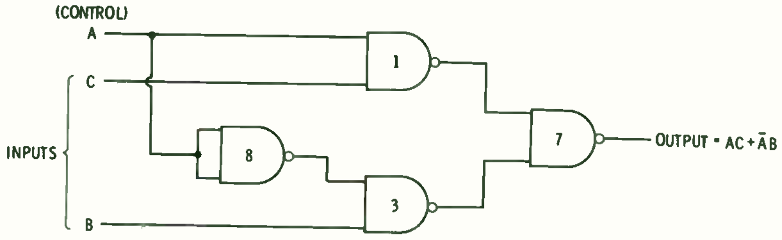
Notice that restricting a design to using only NAND gates sometimes results in a penalty, requiring more total gates—but not always, as in the last example. In general, no added gates are required if the Boolean expression is of the form

$$(\qquad) + (\qquad) + \cdots + (\qquad)$$

where the terms inside the parentheses are logical products.



(A) Using functions from Fig. 4-30.



(B) Simplified circuit.

Fig. 4-31. Two-pole selector switch using NAND gates.

Positive Logic vs Negative Logic

Earlier it was mentioned that the assignment of the symbols 0 or 1 to the two signal voltage ranges was arbitrary. The most common assignment is for the symbol 0 to represent the lower voltage state and the symbol 1 to represent the higher voltage state. You may have noticed that in this chapter, all discussion of gate behavior has been in terms of the symbols 0 and 1, without reference to the voltage ranges to which they correspond.

When you buy a gate of a particular logic type, the manufacturer has almost invariably named it in terms of the positive logic function it performs. Suppose, for example, that you have purchased a two-input TTL AND gate. The manufacturer specifies that the 0 state corresponds to voltages from 0 to 0.6 volt, while the 1 state corresponds to voltages from 2.5 to 4.5 volts, all being measured with respect to a common ground terminal. Somewhere on the data sheet the term "positive logic" usually appears. Assume that Table 4-3 applies to this gate. This table has three sections. The first section expresses the output voltage in terms of the four possible input voltage combinations, but lists voltage ranges instead of the symbols 0 and 1. In the second section, 0's are substituted for voltages in the lower range and 1's for voltages in the upper range. This is the positive logic truth table and shows that the gate performs the AND function. In the third section, for negative logic, 0's are assigned to the higher voltage-range signals and 1's to the lower-voltage signals. This assignment is made for both the input and output signals. Study the third section (negative logic) and you will see that the gate now acts as an OR gate.

Table 4-3. Positive and Negative Logics

Voltages			Positive Logic Symbols			Negative Logic Symbols		
A	B	Output	A	B	Output	A	B	Output
0-0.6	0-0.6	0.3	0	0	0	1	1	1
0-0.6	2.5-4.5	0.3	0	1	0	1	0	1
2.5-4.5	0-0.6	0.3	1	0	0	0	1	1
2.5-4.5	2.5-4.5	2.5-4.5*	1	1	1	0	0	0
			AND Function			OR Function		

* Depends on fan-out, supply voltage, etc.

Except for negation, using negative logic symbolization changes the logic function of a gate. In Table 4-4 the left-hand column lists the five basic logic functions. Opposite each function, in the second column, is the function obtained if negative logic is used instead of the usual positive logic.

Table 4-4. How Gate Logic Functions Change Using Negative Logic

Positive Logic	Negative Logic
Inverter	Inverter
AND	OR
OR	AND
NAND	NOR
NOR	NAND

Negative logic is not used extensively, but references to it are seen occasionally, and you should at least understand the meaning of the term.

Chapter 5

Bistable Elements and Their Uses

Bistable elements play important parts in many logic systems. Although there are a great many variations differing in operating details, all have the common characteristics of possessing a "memory." This means that their outputs will remain in one state even after the signal which put them in that state is removed.

Gates do not have memories. For example, if either of two 1 inputs is removed from a two-input AND gate, the output will return to the 0 state. The output depends only on the *present status* of all the inputs. On the other hand, the output of a bistable element, or flip-flop, depends not only on the present input state, but also on its previous input state or states.

The two primary uses of bistable elements are for electronic memories and for counters or frequency dividers. Many logic systems operate in a sequence of steps. As memories, flip-flops hold data (information) from one period in the sequence to another, so that data is available when needed. The circuits for counters and frequency dividers are very similar to each other; often one circuit can be used for either purpose.

A flip-flop may be constructed as a complete integrated circuit on a single silicon chip, or it may be formed by combining several IC gates of the proper kinds. The latter method will be increasingly in evidence as standard arrays of gates become readily available and are applied to logic systems or subsystems which include one or more flip-flops. At the present time, standard integrated-circuit packages containing four or more flip-flops can be purchased off-the-shelf from many electronic distributors.

There are two broad classes of bistable elements; they are as follows:

1. Level, or threshold, triggered.
2. Transient, or edge, triggered.

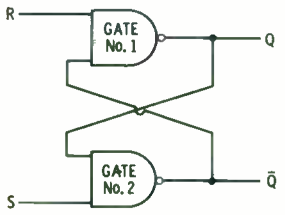
In the first class the change of the output state occurs when the dc voltage level of the input (or inputs) reaches a threshold level. It does not matter how quickly or slowly the voltage approaches this level. In contrast, the edge-triggered flip-flop changes state when one of the inputs (the "clock" or "toggle" input) *changes*. In some devices triggering occurs on a down-going transition, when the input falls from a high or 1 state to a low or 0 state. Other devices toggle on the up-going transition. In either case, not only must the initial and final levels fall within the acceptable voltage ranges, but the transition must occur with at least a minimum speed. Such flip-flops will not trigger if the change occurs too slowly. A further requirement for both classes is that the input signal change be of at least some minimum duration. If a change that would normally cause triggering is followed immediately by a change back to the original state, the flip-flop may not respond at all.

The R-S Flip-Flop

The simplest bistable device is the R-S flip-flop, or *latch*. It is readily made from either two inverters, or two NOR or NAND gates. Fig. 5-1 shows an R-S latch made from two "cross-coupled" NAND gates. The inputs are labeled R (reset), or sometimes C (clear), and S (set), and the outputs are labeled Q and $\overline{Q}$. Labeling the outputs this way implies that their states are always opposite. However, if R and S are both 0, then you will notice that both outputs must be 1, in accordance with the truth table of a NAND gate. This input combination is normally avoided in the design of the rest of the system in which the latch is used, as indicated in the truth table in Fig. 5-1. The remaining three input combinations are the ones normally used. In the table the second and third lines represent the reset and set states, respectively. As long as the other input is a 0, a 1 on the set terminal "sets" the output to 1, while a 1 on the reset terminal "resets" the output (Q) to 0. These results are forced by the logical operation of the gates. You can substitute these values for the inputs and prove to yourself that the second and third lines must always be true.

The fourth line is another matter. To analyze the behavior, assume that the third input combination exists, so that $Q = 1$ and $\bar{Q} = 0$. Now let R become 1, which is equivalent to moving from the third line of the table to the fourth line. Gate 1 is clearly unaffected by this change, since its other input is a 0 from $\bar{Q}$. The output of a NAND gate is 1 unless *both* inputs are 1's; therefore Q remains 1. As long as gate 1 is unaffected, neither is gate 2, since its inputs are Q and S, and neither of these have changed.

A similar situation occurs if you assume the second line as the starting point, and then let S become 1. This is equivalent to moving from the second to the fourth line of the truth table. Once again there is no change in either Q or $\bar{Q}$. Gate 2 already has a 0 on the input connected to Q, so changing S from 0 to 1 has no effect. Then since $\bar{Q}$ and R are unchanged, Q remains a 0.



(A) Diagram.

R	S	Q*
0	0	1 (NOT NORMALLY USED)
1	0	0
0	1	1
1	1	DEPENDS ON PREVIOUS STATE

* EXCEPT FOR FIRST LINE, $\bar{Q}$ IS ALWAYS OPPOSITE OF Q

(B) Truth table.

Fig. 5-1. R-S flip-flop using NAND gates.

In summary, if the input conditions move from either the second or third lines to the fourth line, Q and $\bar{Q}$ do not change. They "remember" their last state. But if the gate inputs are those of line four and change to those of line two or line three, the outputs will be forced to the values shown in the table.

The R-S latch is a level-triggered bistable circuit. Once a critical input voltage level is reached, regeneration (positive feedback) causes the very rapid completion of the switching process.

A simple circuit, but one very useful to the experimenter or engineer in any digital laboratory, is the combination of a switch and R-S latch shown in Fig. 5-2. This circuit eliminates the effects on logic circuits of contact bounce, a phenomenon occurring in all mechanical switches. When switch contacts close, the initial closure is followed by several quick make-break cycles before the contacts stabilize in the final closed position. The number of cycles is not always the same, so that when such

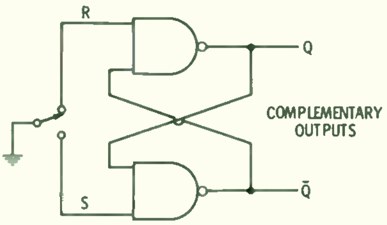


Fig. 5-2. R-S latch used to overcome switch-contact bounce.

a switch is used to drive a flip-flop, the flip-flop will change states an unpredictable number of times. In the switch-latch combination, the latch triggers the first time the contacts close, and thereafter is unaffected by any further contact bouncing, as long as the contacts do not bounce so violently as to make contact again with the other pole. Two push-button switches could also be used to ground either the *R* or *S* input. The circuit is very useful for generating pulses manually for testing logic circuits containing flip-flops.

The regenerative action of the R-S flip-flop can be put to use in the circuit of Fig. 5-3. Here, by adding one gate used as an inverter, the signals to the set and reset terminals are always ensured of being complementary. Certain periodic waveforms applied to the input will generate rectangular waves at the output. The rise and fall times of these rectangular output pulses can be very fast: 10 nanoseconds can be obtained with standard gates. This circuit is especially useful where the input signal is sinusoidal, or where an older "square-wave" generator with slow rise and fall times (such as 1 or 2 microseconds) is available but will not drive edge-triggered flip-flops. The main requirement of the input signal is that it must have maximum and minimum voltages above and below the threshold level of triggering, which is generally about 1 volt.

A way to use the R-S latch as a memory is shown in Fig. 5-4. Whenever the signal on the enable terminal (sometimes called the clock terminal) is 0, both *R* and *S* must be 1's. This places

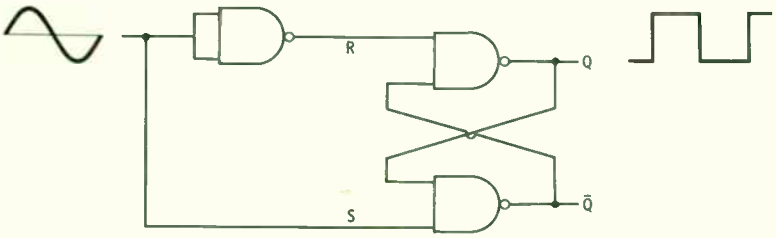


Fig. 5-3. Circuit for generating rectangular waveforms with fast rise and fall times.

the latch in the condition where its output depends on the previous input conditions. When a short positive pulse is applied to the enable terminal, R takes on the value of $\bar{D}$ and S takes on the value of D. Notice that R and S are then complementary; that is, if $R = 0$ then $S = 1$, and vice versa. This forces Q to have the value of D, which is held or "remembered" even after the enabling pulse is removed.

Gates 1 and 2 can be replaced with gates having more than two input terminals, so that the enabling signal can be a binary code of two or more bits. Inverters are required for those bits whose values are 0 when enabling is to occur.

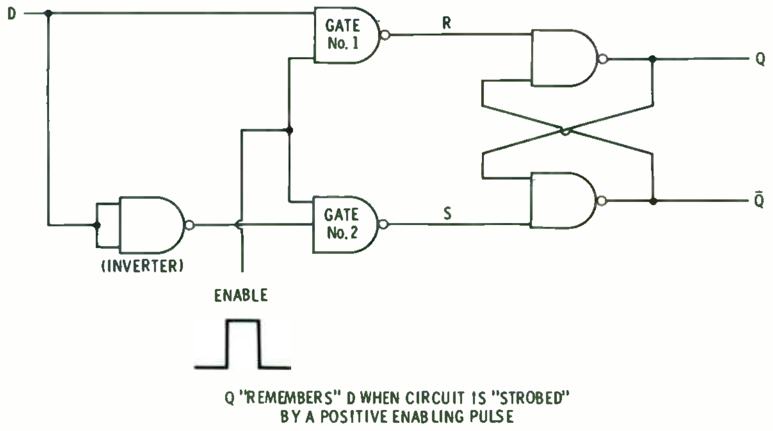

D
GATE No. 1
R
Q
GATE No. 2
S
Q̄
(INVERTER)
ENABLE
Q "REMEMBERS" D WHEN CIRCUIT IS "STROBED" BY A POSITIVE ENABLING PULSE

Fig. 5-4. R-S flip-flop used as memory.

The R-S-T Flip-Flop

The next step in complexity is the R-S-T flip-flop (Fig. 5-5). The T terminal is also called the *toggle* or *trigger terminal.* The output of this flip-flop changes state on the down-going change of the input signal to the T terminal (the up-going change in some designs), provided that the dc inputs to the R and S terminals are correct. Thus, it belongs to the class of edge-triggered flip-flops. If the T terminal is not used, the device will operate as an R-S flip-flop, but the additional T terminal provides much more flexibility of application. In particular, the R-S-T flip-flop can be used as a frequency divider. If, for example, only a down-going signal on T causes its output (Q) to change state, then application of a periodically changing input causes Q to change state once each complete input cycle. The result is an output with half as many "pulses" per unit of time. Furthermore, if the input pulses have a constant repetition rate, then whether or not they have equal high and low durations the output will be a square wave. This is illustrated in the timing diagram in Fig. 5-5B. Note that the assumptions listed in this figure are just one possible set of assumptions. The behavior with respect to the *R* and *S* dc inputs could be different, and, of course, some edge-triggered flip-flops toggle on the upgoing pulse.

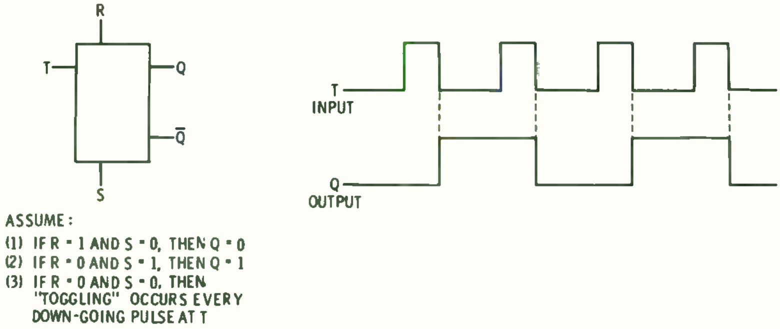

R
T
Q
$\overline{Q}$
S
ASSUME:
(1) IF R = 1 AND S = 0, THEN Q = 0
(2) IF R = 0 AND S = 1, THEN Q = 1
(3) IF R = 0 AND S = 0, THEN "TOGGLING" OCCURS EVERY DOWN-GOING PULSE AT T
T INPUT
Q OUTPUT
(A) Symbol and states.
(B) Frequency division.

Fig. 5-5. R-S-T flip-flop.

R-S-T flip-flops can be connected in series to divide the input frequency by powers of 2 (Fig. 5-6). Two flip-flops will divide the frequency by 4; three will divide by 8, etc. This same arrangement can be used to *count*. Counters are characterized by a discrete number of different combinations of flip-flop outputs that always occur in the same sequence. When all combinations have occurred, the counter starts over. Flip-flops connected to divide by any number *n* will also count to *n* before starting over.

When used as a counter, the flip-flops are first "cleared" or reset, so that all outputs are 0. Then a string of pulses is applied to the input. If the number of pulses is less than 2^n, where in this case *n* is the *number of flip-flops*, then the number of pulses can be determined by noting the values of all the outputs. For example, in the four-stage counter of Fig. 5-6, if the values of Q_1, Q_2,Q_3, and Q_4 are 1, 0, 1, and 0, respectively, there must have

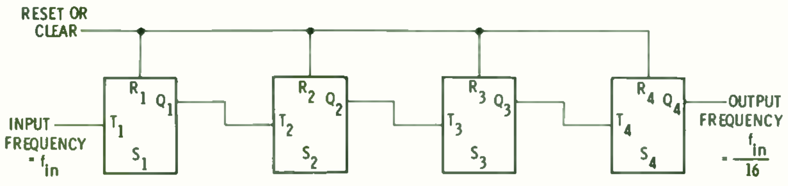


Fig. 5-6. Divide-by-16 circuit using four R-S-T flip-flops.

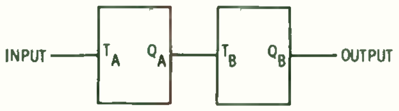
been five pulses since all flip-flops were cleared. In this case the number of pulses is given by the binary number formed by the values of the outputs. However, note that the "most significant" flip-flop is at the right, while the binary number would ordinarily be written in the opposite order.

Counters or frequency dividers which count to or divide by numbers other than 2, 4, 8, 16, etc., may be constructed by adding logic circuits. The general principle is to allow the count to procede normally up to the desired "last count," which is always less than the maximum possible (2^n), then cause the next trigger pulse to return the outputs to the starting combination, usually all 0's. One or more output combinations which would normally occur are jumped over, or bypassed. Fig. 5-7 illustrates a simple arrangement to count up to 3, or to divide by 3. The normal counting sequence is given in Fig. 5-7*A*. To divide by 3, (Fig. 5-7B) the count should proceed normally for the first two down-going pulses, but on the third should return the circuit to the starting count. Notice that this can be accomplished after the second pulse by inhibiting Q_A from changing, and by making Q_B toggle on the third pulse as well as on the second.

Assume that in Fig. 5-7B the starting count is 00. The input AND gate is enabled because one input connects to the complement of Q_B, which is 1 at the starting count. Assume that each flip-flop toggles when the signal on its toggle input goes from 1 to 0. When the input goes high for the first time, nothing further happens. When it then falls to 0, the first flip-flop (*A*) toggles, and its output (Q_A) becomes a 1. Since the second flip-flop will not toggle until its input (T_B) *falls*, it does not change state yet, so AND gate 1 remains enabled. The input goes high again, with no effect on either flip-flop, but when the input falls to 0, several things will happen. Flip-flop *A* toggles again; this time when its output falls from 1 to 0 the second flip-flop is also toggled. Q_B becomes 1, and the first gate, with one input tied to $\overline{Q}_B$, becomes inhibited. AND gate 2, however, now is enabled. The next input signal rise and fall passes through AND gate 2, and since its output is OR'ed with Q_A (which has remained at 0 during the last input pulse), flip-flop *B* is toggled. At this time, after the third complete pulse, both flip-flops are in their 0 states, as they were at the beginning of the cycle.

The sequence just described is represented by the timing diagram in Fig. 5-7B. Notice that there is one complete output pulse at Q_B for every three pulses at the input. The duration of the high and low states are not equal; that is, the output of the circuit when used to divide by three is not a square wave.

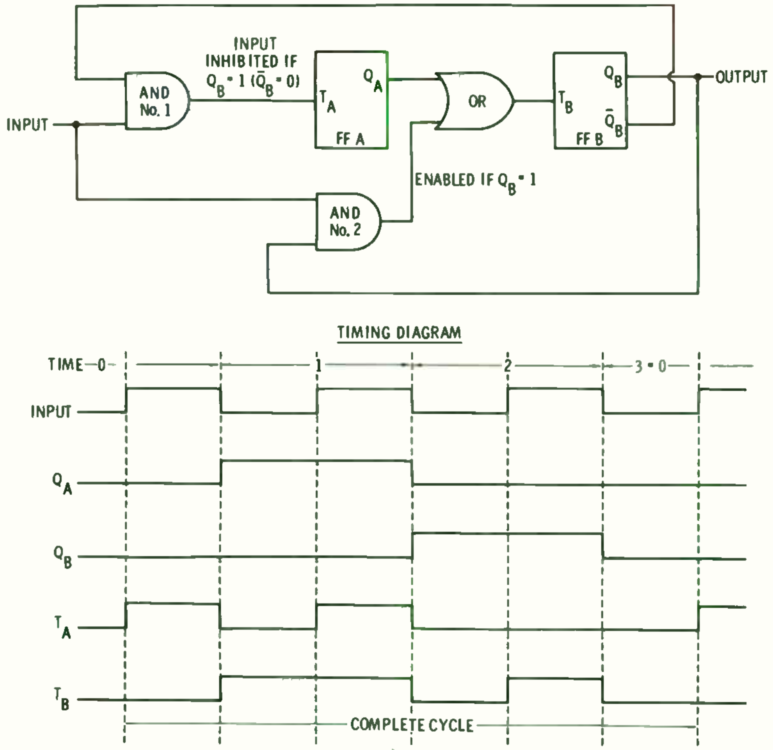
Counters which count up to or divide by any number can be made in a similar way. Such circuits which divide by 5 or 6 (using three flip-flops), or by 10 or 12 (using four flip-flops) are very common. Today it is usually easier and less costly to purchase integrated circuits which contain all the flip-flops and gates properly interconnected in a single package, to divide by



TIME*	Q_A	Q_B
0	0	0
1	1	0
2	0	1
3	1	1
4	0	0 (START OVER)

*0 = STARTING COUNT; 1 = AFTER FIRST DOWN-GOING INPUT PULSE

(A) Without additional gates two R-S-T flip-flops count to 4 (divide by 4).



(B) With additional gates two R-S-T flip-flops can count to 3 and return to starting count (divide by 3).

Fig. 5-7. Divider counters.

2, 4, 5, 6, 8, 10, 12, or 16, or larger numbers containing these numbers as factors.

The counters of Figs. 5-6 and 5-7 are called *ripple-through* or *serial* counters. Each stage must wait for pulses to propagate through the preceding stages before performing its function. There is a short delay between the time that the trigger signal to a flip-flop changes and the time that the output changes. In a ripple-through counter the total delay from the signal change at the input to the change of the last or nth output is the sum of all the delays of each flip-flop. In some applications this limits the maximum frequency of usefulness of the ripple-through counter.

We have already noted that the trigger pulse must meet certain requirements, one being the duration of the pulse. In a single divide-by-2 flip-flop this sets the limit on the maximum frequency or pulse repetition rate for which the flip-flop will toggle. In ripple-through or serial counters which do not complete the full cycle, there are usually additional limitations on maximum speed.

For example, in Fig. 5-7B when the input pulse falls for the second time, there is a sequence of events which must occur before the next input pulse falls. The pulse must pass through gate 1, which delays it slightly. Then flip-flop A must toggle, which takes a little more time. Next the change at Q_A must pass through the OR gate, with additional delay, and then flip-flop B must toggle, in order that $\overline{Q}_B$ can change to 0 and inhibit AND gate 1. If the total of these delays is too long relative to the pulse rate, the input gate may not be inhibited until too late—before the next input pulse has arrived. The counter will then not operate as intended.

When a number of flip-flops are used, the ripple-through time can substantially reduce the maximum input frequency the counter can handle. One way to permit faster operation is to connect all the T inputs in parallel, then use logic circuits to allow some to toggle and to inhibit others from toggling. Such an arrangement is called a *parallel* counter. An example appears in the next section.

The J-K Flip-Flop

In considering the R-S-T flip-flop we have assumed that the R and S terminals operated much as they did in the R-S latch made from two cross-connected NAND or NOR gates. For as long as it is applied a set or reset signal overrides the toggle signal. For example, a reset signal forces the Q output to go immediately to 0 and remain there while the reset signal continues, regardless of any pulses at the trigger terminal. Input signals which cause this kind of behavior are called *dc* or *asynchronous inputs*.

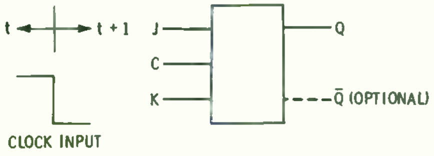
Like the R-S-T binary the basic J-K flip-flop also has three kinds of input terminals and a Q output. The "clock" terminal corresponds to the trigger terminal of the R-S-T device. The two terminals labeled J and K are control terminals, but they do not directly affect the output at Q. Instead, signals on the J and K terminals determine what Q does on the next down-going (or up-going, for some models) change of the signal on the clock terminal. This behavior is called *synchronous operation*—the output changes are "synchronized" with clock changes.

Fig. 5-8 shows a simple J-K flip-flop with a truth table describing its operation. Q_t is the value of Q (either 0 or 1) *before* the clock input falls. Q_{t+1} is the value of Q after the clock input falls. The table states, in a shorthand language, that:

1. If J and K are both zero, $Q_{t+1} = Q_t$; that is, the clock pulse is inhibited and has no effect on the output.
2. If J and K are opposite (complementary), Q takes on the value of J at the time of the next down-going pulse.
3. If J and K are both 1, then the clock terminal acts like a trigger terminal. Q complements on every down-going pulse for as long as $J = K = 1$.

The values of J and K at the instant of the down-going pulse determines how Q changes. In between two down-going pulses, J and K may take on a series of values without affecting Q.

Some J-K flip-flops have more than one each J and K terminals. The terminals of the same type may be OR'ed or AND'ed This is illustrated in Fig. 5-9. In the OR'ed case, a 1 on any of the J terminals makes the flip-flop perform as though it had a single J terminal with a 1 applied. In the case of AND'ed inputs, all J's (or K's) must have 1's to obtain the equivalent of a single input with a 1 applied.

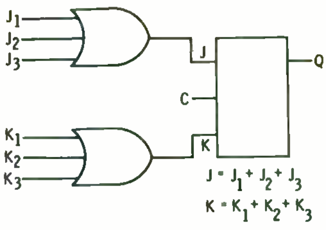


(A) Diagram.

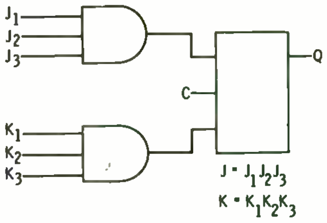
J	K	Q_{t+1}
0	0	Q_t
1	0	1
0	1	0
1	1	$\overline{Q}_t$

(B) Sequence.

Fig. 5-8. J-K flip-flop.



(A) OR'ed inputs.



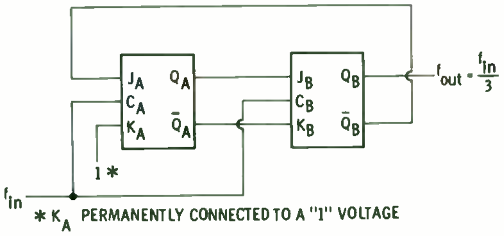
(B) AND'ed inputs.

Fig. 5-9. J-K flip-flops with multiple inputs.

In addition to the synchronous J and K inputs, a device may also have dc or asynchronous R and S inputs. In some designs the clock input may be missing, but may be created by connecting a J and K input together and applying trigger signals to that connection. For synchronous operation the Q and $\overline{Q}$ outputs of a J-K flip-flop are always complementary. However, for some units having dc inputs, simultaneous application of set and reset signals can cause both Q and $\overline{Q}$ to be 1 (or 0, depending on the exact design).

J-K flip-flops are useful in a wide variety of applications due to the flexibility offered by their control modes. Fig. 5-10 shows the circuit diagram of a divide-by-3 parallel counter using two J-K flip-flops, along with a table giving the sequence of output states. The input frequency is applied to both clock terminals in parallel. Using the table from Fig. 5-8, it is easy to follow the operation of the divider as the sequence progresses.

Assume that Q_A and Q_B are both 0 to start. On the first down-going pulse, flip-flop A will toggle, since both of its J and K inputs are 1. The second flip-flop, however, has complementary signals on its J and K inputs, so its output will take on the value of J, which is 0. Since this is the value Q_B had to begin with, it does not change. The output states after the first down-going pulse are given on the second line of the sequence table.



(A) Diagram.

Q_A	Q_B
0	0
1	0
0	1
0	0

(B) Sequence.

Fig. 5-10. Divide-by-3 parallel counter using J-K flip-flops.

When the next down-going pulse arrives, the first flip-flop will toggle again, since it still "sees" a 1 on both its J and K inputs. The second flip-flop, as always, has complementary signals on J and K, and will thus take on the value of J, which this time is 1. The outputs become those of the third line of the table.

Finally, on the next down-going pulse, Q_A will remain at 0, since the signals on J_A and K_A are now complementary, with 0 on J_A. In the second flip-flop, now with 0 on J_B, Q_B will take on the value 0. Thus, after three pulses, both outputs are back to their original states, completing the sequence.

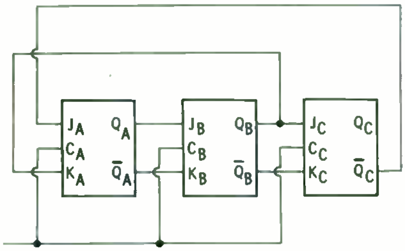
If a square-wave signal is applied to the input, the output will be a rectangular wave with one pulse for every three pulses at the input. The same output appears at either Q_A or Q_B, except that they are not "in phase" with each other. At either point the output signal is "high" for one-third of the cycle and "low" for two-thirds of the cycle.

Counters which do not use all possible output combinations in their natural sequence may have undesired modes of operation. This can occur if for some reason, such as a start-up or noise transient, the outputs take on one of the combinations not normally used. If the sequence from that point on does not naturally step into one of the combinations in the desired count sequence, the divider may divide by some number other than the desired one. In Fig. 5-10 we may investigate what happens if the divider should get into the state where Q_A and Q_B are both 1. We see that after the next down-going pulse Q_A will be 0—because $J_A = 0$ and $K_A = 1$. Q_B will remain 1, but the combination is now one of those which occurs naturally in the intended sequence. Thus, despite the temporary disturbance, the divider will pick up the normal sequence of counting. In some counters, circuitry which on the surface appears unnecessary is added to prevent "latch-up" in some state from which the counter cannot escape, or from which the intended count sequence will not result.

Fig. 5-11 shows a divide-by-5 counter of the parallel type. Since all clock terminals are connected in parallel, there are no long delays due to a signal having to pass through a number of flip-flops and gates. The delay is never more than that for one flip-flop. This helps keep the maximum speed (frequency) of the counter high. This counter is sometimes called a *shift counter*. The output may be taken at Q_A, Q_B, or Q_C, these signals differing only in their relative phases.

All major manufacturers of logic circuit elements can supply circuit suggestions for connecting their flip-flops to divide by any number. As previously mentioned, most newer designs now use MSI (medium-scale integration) packages, containing two, four, or more flip-flops and additional gates, all properly internally connected to perform division by 2, 4, 5, 6, 8, 10, 12, and 16.

For division by larger numbers not factorable into numbers for which standard dividers are available, the usual technique is to use special-purpose decade counters connected in series (cascade). Each counter has the feature that it can be set to any BCD count from 0 to 9. After all decades are set, the counter *counts down* until all Q outputs are 0's, at which time a pulse is produced. At that time, all of the counters are again set ("strobed") to their initial count and the count-down begins again. This approach provides great flexibility; not only can division by any number be accomplished, but the number may be varied easily and quickly. Dividers with this capability are called *variable-modulus* or *mod-N dividers* (or counters).



(A) Diagram.

Q_A	Q_B	Q_C
0	0	0
1	0	0
1	1	0
0	1	1
0	0	1
0	0	0

(B) Counting sequence.

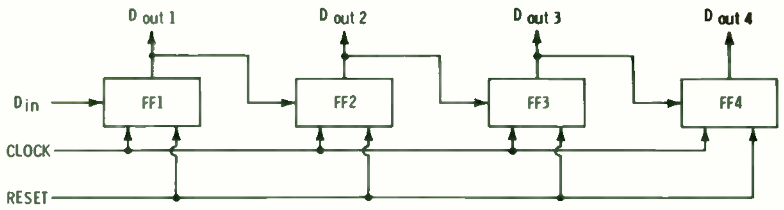
Fig. 5-11. Divide-by-5 parallel counter.

They have been used in applications ranging from shutting off a duplicating machine after it produces a preset number of copies, to special-purpose computers, to frequency dividers in phase-locked frequency synthesizers operating into the vhf range.

The Shift Register

A shift register is a type of memory consisting of a group of flip-flops connected to operate together in a specified way. Each flip-flop is called a *stage;* a four-stage register contains four flip-flops, an eight-stage register contains eight, etc.

A basic shift register is illustrated in Fig. 5-12, which shows a four-stage register. This register has one input terminal, four output terminals, and two control terminals—clock and



RESET = 0; if RESET = 1, ALL OUTPUTS BECOME 0

TIME	D out 1	D out 2	D out 3	D out 4
t+1*	D in	D out 1	D out 2	D out 3

*t+1 MEANS A SHORT TIME AFTER CLOCK INPUT CHANGES FROM 0 TO 1

Fig. 5-12. Four-stage shift register.

reset. The operation table shows that binary data (0 or 1) is shifted to the right on each upgoing clock pulse. Data at the D_{in} terminal shifts into the first flip-flop. Data stored in the last flip-flop is "discarded" after each clock pulse. In the absence of clock pulses, data remains where it is, as long as supply voltage to the register is maintained, of course.

One use of the register is to obtain a time delay in a sequence of data. If a periodic square wave is applied to the clock terminal, binary data applied at the same rate at D_{in} reappears at each output delayed in time. For example, if clock pulses occur once every millisecond (1/1000th second), then a series of data pulses at D_{in} reappears at D_{out2} delayed by one to two milliseconds. The exact delay depends on the elapsed time between the arrival of the data pulse at D_{in} and the occurrence of the clock pulse. At each subsequent register output the data pulses are delayed an additional millisecond.

Another application of the shift register is to convert *serial data* into *parallel data*. Serial data consists of a time-series of quantities on a single wire. Sending Morse Code by telegraph is an example of serial data. Another example of serial data is found in electronic calculators. Numbers and operations are entered one digit or operation at a time. To multiply 25 × 42, the user pushes the keyboard buttons serially, one at a time, in the order 2, 5, × 4, 2, = . The answer (1050) appears as parallel data; all dights in the answer are displayed simultaneously.

When a shift register is used for serial-to-parallel conversion, the data inputs, in the form of 0's and 1's, must be coordinated with the clock pulses. The latter must occur only when a new input data bit has arrived on the input wire. If the clock occurs regularly, such as every millisecond, then each bit of input data must arrive in time to be transferred to the first flip-flop.

Besides the serial-to-parallel shift register, there are also parallel-to-serial registers, combinations of the two, and left-right registers in which data can be shifted in either direction. In the parallel-to-serial register, there is an input terminal for each stage. Input data is applied simultaneously, and there is a single output terminal where the applied data appears serially as a string of 0's and 1's.

The Language Problem

The vocabulary of logic elements, and their mathematical and symbolic representations, has not been well standardized. This can lead to considerable confusion to the newcomer in the field. The same term may be used in a number of different ways. Conversely, several terms may all refer to the same thing, or to minor variations, the differences being obscure or unexplained. In studying technical articles manufacturer's data sheets, and manufacturer's application notes, it is necessary to keep a very open and flexible mind and to be alert to clues as to the meaning of nearly every important term.

For example, the dc inputs to an R-S-T flip-flop may be called "set" and "reset," or "set" and "clear," or "preset" and "preclear." The *J* and *K* inputs, which are synchronous or ac inputs, have been called "set" and "clear" by some manufacturers. As another example, the terms "trigger," "toggle," and "clock" may all refer to the same input terminal of a flip-flop—or there may be a distinction between each pair of terms.

In the appendix of this book is a glossary of terms in which an attempt has been made to give the more common meanings of most-used words. However, in any particular case, the meaning may be suggested by the context in which a word is used. A similar comment can apply to graphic symbols.

Chapter 6

How Logic Families Compare

In the preceding chapters we have treated all gates performing the same logic function (all NAND gates, all NOR gates, etc.) as though they were alike. We rather indiscriminantly called the input and output signals simply 0's and 1's. We did consider briefly, in Chapter 3, just what voltage levels can constitute a 0 or 1, with the observation that these symbols do not always stand for the same ranges of voltages. In this chapter we will make additional comparisons which will bring out more specifically the differences between the families of logic elements.

Classification by Families

Besides classifying gates by function, we can classify them by "family." Some of the more common families are:

Diode logic.
RTL (resistor-transistor logic).
DTL (diode-transistor logic).
TTL or T^2L or TCL (transistor-transistor logic or transistor-coupled logic).
ECL or CML (emitter-coupled logic or current-mode logic).
CTL (complementary-transistor logic).
MOS (metal-oxide-semiconductor) logic.
CMOS (Complementary metal-oxide semiconductor) logic.

Each broad family may be further broken down into subfamilies or into families assigned trademark names by particular manufacturers. Subfamily classifications are usually based on differences in voltage levels, or differences in power, speed, and fan-out capabilities. Family names assigned by manufacturers are partly for advertising purposes, but they also generally imply compatibility, meaning that all gates in the family may be used together in systems as long as fan-out rules are observed. Examples of manufacturers' family names are "SUHL II" (Sylvania), "Utilogic" (Signetics), "7400 Series" (Texas Instruments), and, of course, there are many more.

Broad family names (RTL, DTL, etc.) do not necessarily assure the user of compatability. For example, you cannot always be certain that TTL gates made by one manufacturer will be fully compatible with TTL gates made by another, although in many cases they will be. The designer must be even more cautious about attempting to intermix broad families in large systems.

Up to now we have tried to ignore the details of what goes on inside a gate. It has even been stressed that it often is not necessary to know these details—that what really counts is the performance as a whole. In fact, one advantage in using integrated-circuit logic elements is that the user need not be concerned with internal design details and is thus free to concentrate on system design.

In this chapter we will not deviate far from this philosophy. However, while we can avoid a detailed knowledge, we can nevertheless gain from a general idea of the internal circuitry of gates in the various families. This knowledge helps us to understand the differences between the various logic families.

Diode Logic

Diode logic suffers from certain limitations which make it unsuitable for exclusive use in larger systems. Nevertheless, it still finds widespread use in simple logic circuits, or combined with other logic families in larger systems.

Diode OR and AND circuits are shown in Figs. 6-1A and 6-1B. In Fig. 6-1A, if terminals *A* and *B* are grounded (inputs are both 0), or if there is no connection to either, there is no output voltage. If a positive signal voltage is impressed on either terminal, that voltage appears at the output, reduced by the forward voltage drop of the diode—about 0.25 volt for germanium diodes or about 0.6 volt for silicon diodes. For the AND circuit in Fig. 6-1B, the output is "high" if both inputs have voltages applied which are approximately equal to or higher than V_s. The exact value of the output voltage will depend primarily on V_s and on how much current flows *into* the output terminal, since this current must flow through R_s and, accordingly, reduce the output voltage below V_s.

Notice that for the OR circuit, currents flow *out of* the input terminals, and no connection is equivalent to a 0 signal. On the other hand, for the AND circuit, currents flow *into* the input terminals, and no connection is equivalent to a 1 signal.

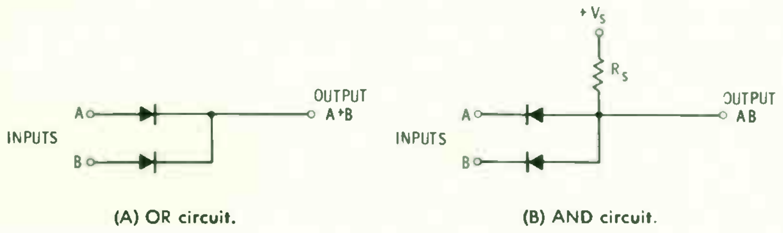


Fig. 6-1. Diode logic circuits.

One limitation of diode-logic circuits is the absence of the NOT function. Thus, inverter amplifiers are required for the construction of many diode-logic combinations. Another limitation is that the diode drop in each stage limits the number of stages that can be "cascaded," since the voltage is reduced with each stage. Also, fan-out is limited because there is no signal power amplification as there is with other logic families. On the other hand, propagation delays can be short if fast diodes are chosen, and the circuitry is simple and easy to understand.

Fig. 6-2 shows a circuit for developing a BCD code from a ten-position switch using diodes connected in a "matrix." Assume the switch is in position 5. The voltages at terminals *A* and *C* will be high, while the voltages at *B* and *D* will be low—equal to the diode forward voltage drop of about 0.6 volt for the silicon diodes. Thus, the output can be expressed as 0101, which is the BCD code for 5.

The "high" outputs in Fig. 6-2 will always be less than V_s, due to current through R_s. Part of this current flows into the output terminal out of the load. The remaining current consists of the sum of numerous small reverse currents through the diodes. One such path is indicated by dashed lines in the diagram. These tiny currents increase with temperature, and this fact limits the maximum practical values of R_s and the load resistance. Silicon diodes are preferred to germanium diodes because the reverse leakage currents are lower.

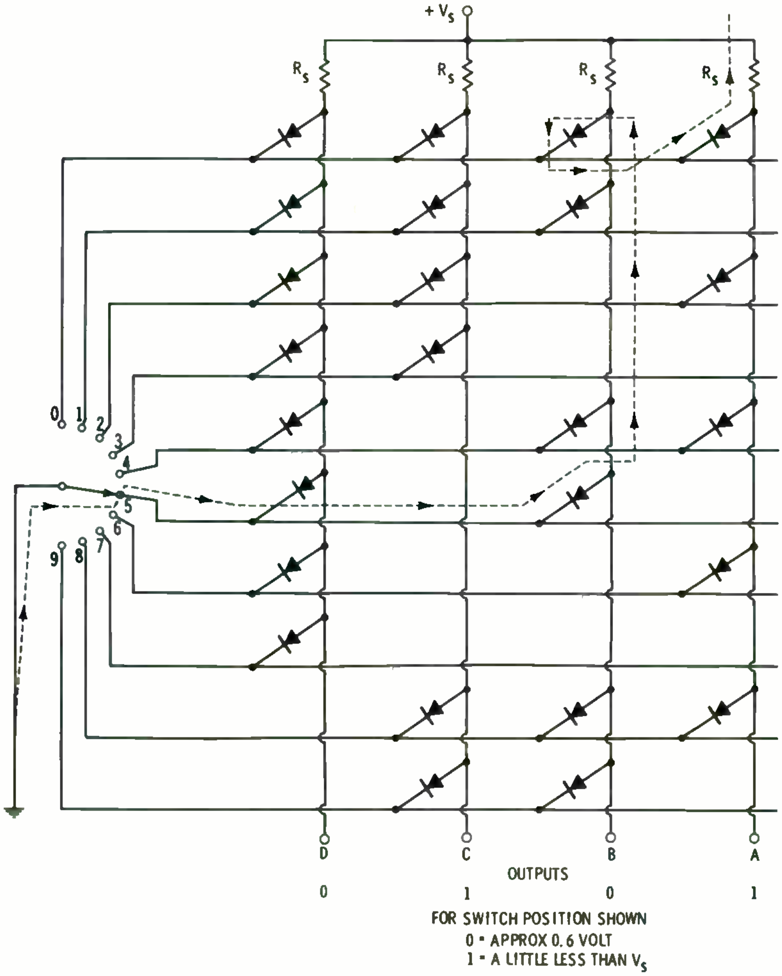


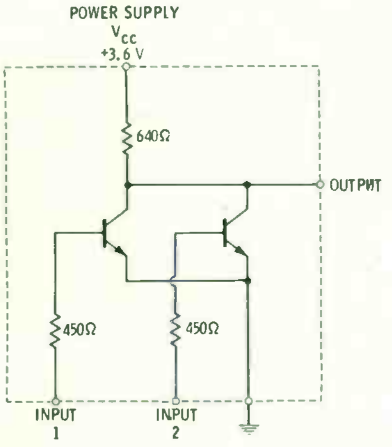
Fig. 6-2. Diode matrix for producing BCD code.

Resistor-Transistor Logic

Historically the first readily available active integrated logic circuits were of the RTL or resistor-transistor logic family. These were relatively easy to manufacture, and since the circuitry is simple with few "parts" there were fewer rejections. Higher "yields" meant lower costs, and even today RTL is the least expensive logic family. Where economy is the most important design consideration, RTL elements are found in new designs as well as in older equipment.

Fig. 6-3 shows the internal circuit of a typical two-input RTL gate (Motorola Type MC 724P). Four such gates are contained in a single plastic package having 14 pins, or terminals. The diagram shows only one gate circuit, since each gate is like but independent of the others. The four gates do, however, have common terminals for the ground and power supply connections.

You can see that the RTL circuit contains only two kinds of elements: resistors and transistors. There is one transistor for each gate input. If you compare this circuit with other circuits in this chapter, the relative simplicity will be obvious.



+ 0.4V OR LESS = LOGICAL 0
+ 0.8V OR MORE = LOGICAL 1

APPROX POWER DISSIPATION:
25 mW/GATE WHEN OUTPUT = 0
6 mW/GATE WHEN OUTPUT = 1

PROPAGATION DELAY:
APPROX 24 NANOSECONDS

LOADING FACTORS:
INPUT = 3; OUTPUT = 16

Fig. 6-3. Typical RTL gate (Motorola Type MC 724P).

The operation of the RTL NOR circuit was explained in Chapter 3. Recall that current flows *out* of the input terminal when a 1 signal is applied. This input current becomes the base current of the transistor, and, when large enough, it causes the transistor to conduct current from emitter to collector. This in turn lowers the collector voltage to the transistor saturation voltage of only a few tenths of a volt.

The first observation we can make about RTL is that the basic gate performs the NOR logic function. As we have previously seen, NOR gates may be connected as inverters, by using only one input and leaving other inputs open, or by tying all inputs together. NOR gates have the property of being combinable into OR, AND, or NAND functions. Thus, using a chip containing a sufficient number of NOR gates, connections can be manufactured on the chip to form these other logic functions. Indeed, such devices are readily available."

Another observation we can make from Fig. 6-3 is that if the input is an open circuit there is no input current. In other words, an open input terminal produces the same result as if it were connected to ground—both cases represent 0 input signals. This is not true for all logic families, as we will see.

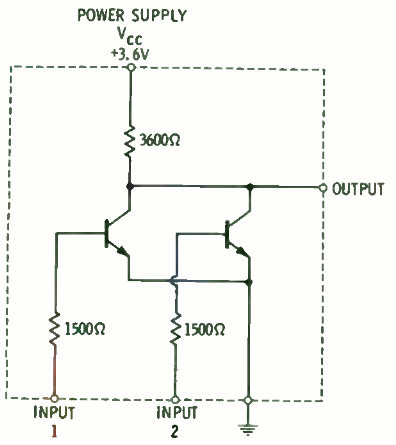
Other RTL characteristics are not so obvious from the schematic, especially as they compare with the other logic families. These characteristics include speed or propagation delay, power consumption, noise immunity, and fan-out or loading capabilities. Data on some of these characteristics is listed in Fig. 6-3.

Within the RTL family of NOR gates, different characteristics can be achieved by varying the transistor specifications and the resistor values. These variations will affect many of the characteristics. The most common variations are the values of resistors. Higher resistances reduce the power drawn by the gate and decrease the input current when a logical 1 is applied. However, speed and loading capability are sacrificed. In RTL, as well as in all gate families, the gate designer can "buy" speed at the cost of more power drawn from the power supply. This is due to the presence of small unavoidable capacitances associated with the transistors, which are charged and discharged more rapidly through smaller resistances.

To give the user more flexibility, Motorola makes two kinds of RTL. Fig. 6-3 gives the circuit for the standard RTL gate, but low-power, or "milliwatt," RTL is also available. The circuit diagram for the corresponding low-power RTL two-input NOR gate is shown in Fig. 6-4. The higher resistor values are seen to be the primary difference. For the milliwatt RTL, the input currents are about a third as much as for the standard RTL. The power dissipation is about one-fifth as much. Although load factors are reduced about three times, fan-out is only slightly less as long as the system uses all low-power gates. Propagation time is doubled. The low-power devices would usually be selected for systems where the higher speed is not required, to take advantage of the less severe power supply requirements and heat dissipation problems.

Diode-Transistor Logic

Diode-transistor logic is a step beyond RTL in complexity, in that each gate has more total parts. The circuit of a typical DTL three-input gate (Fairchild Type DTμL 962) is shown in



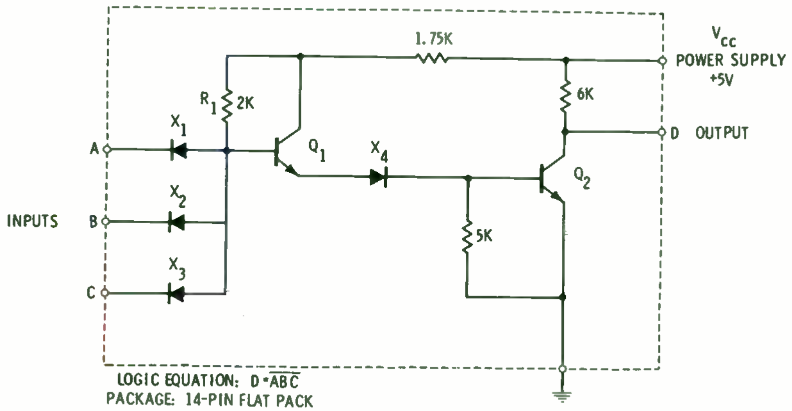
APPROX POWER DISSIPATION:
5 mW/GATE WHEN OUTPUT = 0
2 mW/GATE WHEN OUTPUT = 1

PROPOGATION DELAY:
APPROX 45 NANOSECONDS

LOADING FACTORS:
INPUTS = 1; OUTPUTS = 4

Fig. 6-4. Low-power RTL logic (Motorola MC 717P).

Fig. 6-5. This circuit is one of three such circuits that are manufactured together on one silicon chip and sold in a 14-pin "flat-pack" configuration. The three gates share common power supply and ground terminals.



LOGIC EQUATION: $D = \overline{ABC}$
PACKAGE: 14-PIN FLAT PACK

Fig. 6-5. Typical DTL gate (Fairchild DTμL 962).

From the circuit diagram, you will immediately notice that the directions of the input diodes indicates that current must flow *into* the input terminals, and this is indeed the case. The value of base bias resistor R_1 is such that the first transistor Q_1 is biased to conduct current if the input terminals are all open. Most of the resulting emitter current of Q_1 becomes the base current of Q_2, so that the latter too is "turned on." For this condition, the output voltage is equal to the saturation voltage of the second transistor—only a few tenths of a volt.

With no current through any of the input diodes, the voltage at the base of Q_1 is approximately 1.8 volts. This is the sum of the three forward voltage drops of the base-emitter junctions of Q_1 and Q_2 and the diode X_4. If a voltage lower than this is applied to one of the input terminals, the diode associated with that terminal is forward biased. The Q_1 base voltage falls, turning both Q_1 and Q_2 "off," so that the output voltage will rise to a voltage near the supply voltage.

For DTL a logical 1 input (high voltage) has the same effect as an open circuit. In contrast, you will remember that for RTL an open circuit was equivalent to a logical 0. In order to establish a 0 on the input terminal of a DTL gate it is necessary to ground it or apply a voltage less than about 0.5 volt.

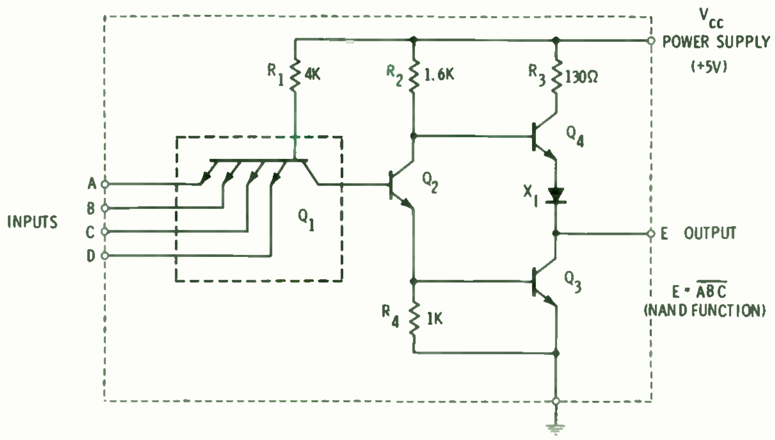
Since a low signal (logical 0) on any one or more input terminals causes the output to be high (logical 1), the basic DTL gate performs the NAND function. As with other NAND gates the device operates as an inverter if unused inputs are left open or connected to a logical 1 voltage such as the supply voltage.

One of the advantages of DTL is its improved noise immunity. This is mostly due to diode X_4, which raises the Q_1 base voltage by an extra 0.6 volt. A noise pulse at an input terminal must exceed the three forward drops of Q_1, X_4, and Q_2, less the drop of the input diode, a net voltage of about 1.2 volts, in order to operate the gate.

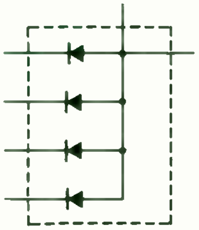
The speed, power dissipation, and fan-out vary among the designs of different manufacturers but in general are comparable to the standard types of RTL. Like RTL, DTL devices are quite economical, in part because they have been in production for a relatively long time. This has permitted yields to be maximized, a broad market to be developed, and automated production equipment to be placed in operation.

Transistor-Transistor Logic

Transistor-transistor logic was an outgrowth of diode-transistor logic. The diodes in each input lead have been replaced by a multiple-emitter transistor, which performs almost exactly the same function as the diodes, but with higher speed. To obtain still shorter propagation delays the output circuit is also modified in many designs.



(A) TTL gate.



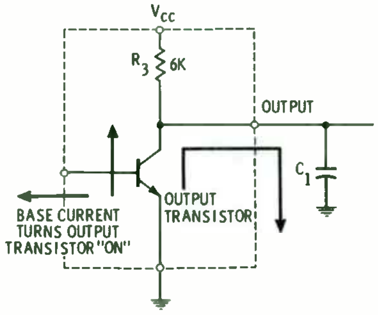
(B) Approximately equivalent input circuit.

Fig. 6-6. Four-input TTL gate (TI Type SN7420).

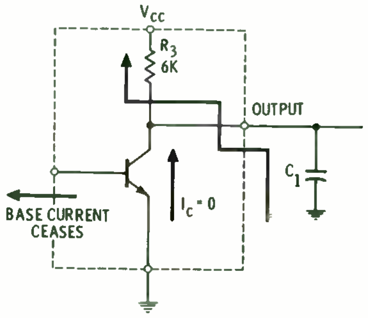
A typical TTL circuit diagram is shown in Fig. 6-6, which represents one of two identical four-input gates in the same package (Texas Instruments Type SN7420). The equivalency of the multiple-emitter and DTL input circuits is indicated. The base-emitter junction of Q_2 performs a similar function as X_4 in Fig. 6-5. However, the collector resistor of the DTL gate is replaced in this gate by transistor Q_4. This arrangement is sometimes called an *active pull-up* and has the effect of providing a low output impedance in both the high and low output states.

The shorter propagation delay of TTL gates is due in part to the multiple-emitter input circuit but is also a result of the low output impedance for both output states. The low impedance permits the gates to charge any small capacitance at the output very quickly. This capacitance may arise from relatively long wires leading to other circuits, or the capacitance between conductors on a printed-circuit board, or it may be the sum of the input capacitances of other gates and flip-flops. The low output impedance has a further advantage: It reduces the sensitivity of the logic system to noise pulses capacitively coupled into the system, by tending to "short out" such pulses to ground.

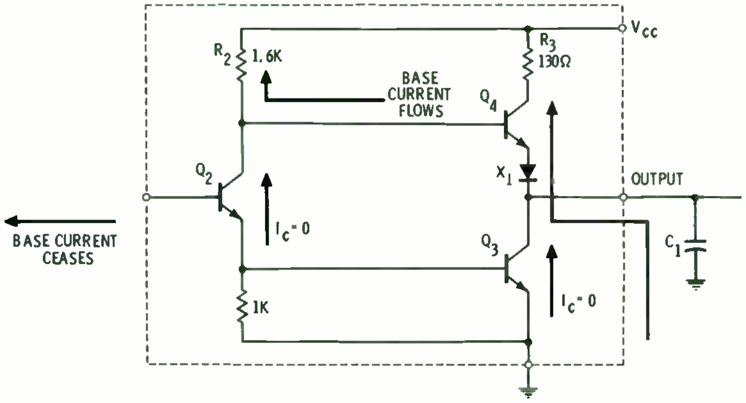
The operation of the active pull-up is illustrated in Fig. 6-7. In Fig. 6-7A, which shows a passive or resistor pull-up, suppose that the output has been high and is changing to the low or 0 state. The capacitor represents any capacitance seen by the output, of which there will always be at least a little. While the output was high, this capacitor has been charged to the logical 1 output voltage. When the output transistor changes rapidly to its fully conducting state, electrons flow from one



(A) Low transistor resistance permits capacitor voltage to fall quickly.



(B) C_1 charges through passive collector pull-up resistor R_3 when gate output goes high.



(C) C_1 charges through active pull-up transistor when output goes high.

Fig. 6-7. Operation of the active pull-up in TTL gates.

terminal of the capacitor to the other, through the transistor. The low "on" resistance of the transistor permits the capacitor voltage, which is equal to the gate output voltage, to fall quickly to within the logical 0 range.

Now suppose that the gate output is changing to the high state. In Fig. 6-7B the transistor abruptly becomes nonconducting, but the capacitor, which has been charged with only a few tenths of a volt, must charge to the higher voltage before the output is truly high. With the transistor nonconducting, the capacitor charging current flows through collector "pull-up" resistor R_3. Since R_3 is much larger than the "on" resistance of the output transistor, it takes much longer for C_1 to charge than it did to discharge. If R_3 is made smaller, the charging speed is increased, but more power is drawn by the gate.

In Fig. 6-7C, the active pull-up transistor is shown. When the output changes from high to low, the behavior is exactly as in Fig. 6-7A, but when the output changes from low to high there is no resistor to limit the rate of change of voltage across C_1. Instead, upper pull-up transistor Q_4 will conduct in the following manner. Transistor Q_2 becomes nonconducting, so that it ceases conducting the base current of output transistor Q_3 and turns the latter transistor off (nonconducting). At the same time, there is no current through the collector of Q_2, so that the base voltage of Q_4 rises, allowing Q_4 to conduct current from emitter to collector. The low "on" resistance of Q_4 allows a relatively high charging current in the path consisting of Q_4, diode X_1, R_3, C_1, and the power supply, charging C_1 quickly. Thus the operation of Q_4 is quite transitory—it does its job quickly and then waits for the next time it is needed to provide current again to charge the output capacitor.

The same feature that gives an active pull-up TTL gate its capacitive drive capability also contributes to one of the disadvantages of TTL. During the short transition from one output state to the other, there is a momentary period when both Q_3 and Q_4 are conducting. This draws a current pulse from the power supply. The pulse can become an interfering noise pulse to another part of the logic system. Also, when the gates are changing states frequently—up to millions of times per second in some systems—these transient current pulses impose a significant additional load on the power supply.

The function of the resistor in the collector circuit of the pull-up transistor is to limit output current to a value that will not destroy the pull-up transistor should the output terminal be accidentally short-circuited to ground.

Like DTL, current flows into the input terminals of a TTL gate. To get a 0 input, the input terminal must be grounded. A voltage source with a low voltage (less than about 0.4 volt) will do this. A 1 input for most TTL gates is about 1.5 volts, and any voltage above this, or an open input termination, will result in no input current and therefore a logical 1 input. However, Q_1, Q_2, and Q_3 (in Fig. 6-6) will all conduct unless logical 1 signals are applied to *all* inputs. Since this condition turns on the output transistor, making the output low, the gate performs the NAND logic function.

TTL achieves shorter propagation times with less increase in power consumption than would be required with RTL or DTL gates. This has contributed to its present popularity. Propagation delays of 10 or 12 nanoseconds are "standard," with faster gates also readily available having delays as short as 5 or 6 nanoseconds.

Although more costly than RTL, TTL is nevertheless not an expensive form of logic. The variety of off-the-shelf gates available has been constantly expanded until now a number of manufacturers offer wide lines which include variations of the basic NAND gate as to speed, power, and loading capabilities, as well as AND, OR, NOR gates, and more complex logic functions. The TTL family will probably form the basis of extensive lines of medium- and large-scale integrated circuits (MSI and LSI), wherein many gates are interconnected on one silicon chip in a single package.

In new designs TTL gates are frequently selected where delays of 6 to 12 nanoseconds are required, or where pulse repetition frequencies are in the range of 5 to 20 MHz. TTL flip-flops are available which toggle up to frequencies in excess of 50 MHz.

The speed of TTL can be further increased by connecting Schottky barrier diodes across all transistors which would normally saturate. The diodes prevent the transistors from saturating, thus eliminating the time delay associated with removing stored charges from saturated junctions. Schottky diodes themselves have no stored charge, so the voltage across their junctions can change extremely rapidly. Propagation delays of 2-3 nanoseconds are standard for many Schottky TTL gates. A further advantage is that the gate characteristics are less affected by temperature changes. However, since they cost more, Schottky TTL devices are better reserved for applications where their higher speed is essential to system operation or performance.

Emitter-Coupled Logic or Current-Mode Logic

The logic families discussed so far all belong to a broad class of logic called *saturated logic*. This means that the transistors swing between their extreme bias conditions where they either do not conduct at all or they conduct heavily with a very low emitter-collector voltage. A saturated transistor has a delay in coming out of saturation, termed *storage time*. This storage time is inherent in the physics of transistor operation. To avoid this delay, the switching transistors can be operated always in their active or unsaturated operating regions. This is done in emitter-coupled logic, with the result that delay times as short as 2 nanoseconds are practical.

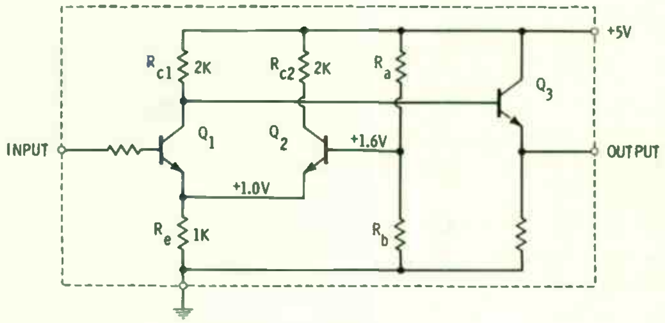


Fig. 6-8. Basic emitter-coupled logic circuit.

The basic emitter-coupled logic circuit is shown in Fig. 6-8. Although this simplified circuit does not represent any actual device, it does illustrate the principle of operation. In the diagram, assume first that there is a low or 0 input signal, and, accordingly, no base current in Q_1. Bias resistors R_a and R_b set the base voltage of Q_2 at 1.6 volts. Since the base-emitter voltage drop of a forward-biased silicon junction is about 0.6 volt, the emitter voltage at Q_2 is 1.0 volt. The current through R_e must therefore be 1 millampere, by Ohm's law.

When Q_1 is "off" (no base or emitter current), all the current through R_e is through Q_2. If we assume that the collector and emitter currents are nearly equal, the 1-milliampere current through R_{c2} (2000 ohms) causes a 2.0-volt drop there. The emitter-collector voltage of Q_2 is the difference between the supply voltage and the voltage drops across R_e and R_{c2}:

$$\begin{aligned}\text{emitter-collector voltage} &= 5.0 - (2.0 + 1.0)\\ &= 2.0 \text{ volts}\end{aligned}$$

Thus, with Q_1 off, Q_2 is not saturated, as the typical emitter-collector voltage for the saturated condition is about 0.2 volt.

Now let an input voltage of approximately 2 volts be applied. Base current immediately flows in Q_1, which in turn causes this transistor to "turn on." But both emitters are "clamped" to 1.0 volt by the base bias on Q_2 and the constant base-emitter voltage drop of Q_2. Therefore the current through R_e remains nearly constant, and any increase in emitter current in Q_1 must be offset by a corresponding decrease in emitter current from Q_2. The entire 1-mA current switches from Q_2 to Q_1. The circuit is symmetrical so that Q_1, although conducting, is not saturated. Since Q_2 was not previously saturated, no storage time is involved and the current switching occurs very rapidly.

In a practical ECL gate additional circuits are required to make the output and input voltage swings compatible. The design details and additional circuitry allow the input and output 0 and 1 voltage ranges to be approximately equal. Also, the output is usually taken from an emitter follower (shown in Fig. 6-8) in order to achieve low output impedance for good noise immunity and for driving capacitive loads. Notice that current flows out of the input terminal when a logical 1 signal is applied.

ECL gates are characterized primarily by their very high speed, obtained because the transistors are never saturated. Another advantage is that there is virtually no generation of power supply transients, because the current is nearly the same whether the output is a logical 1 or a logical 0 or in the process of changing between the two. The current merely switches back and forth between two paths, but the total current remains constant.

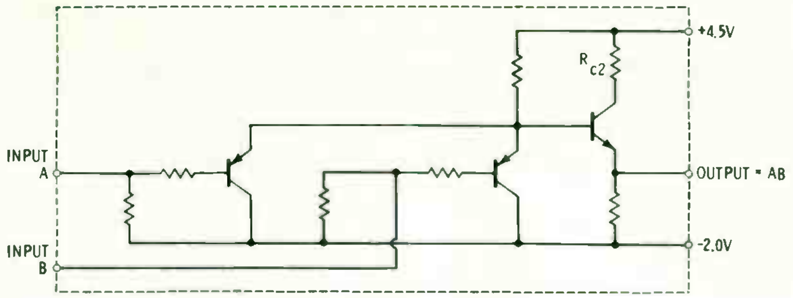
At the present time the cost per gate of ECL is higher than for RTL, DTL, or TTL. Also, there are fewer standard "complex functions" available off-the-shelf, partly due to the power consumption per gate, which is higher and therefore limits gate density in small packages capable of only limited heat dissipation. But where the ultimate in speed is required, ECL reigns supreme.

Complementary-Transistor Logic

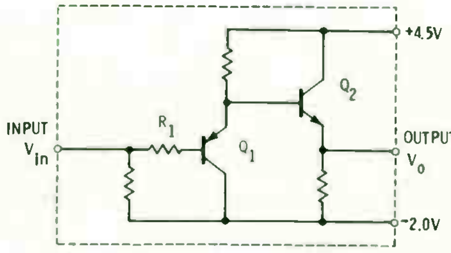
Low-cost complementary-transistor logic with speeds approaching, and sometimes equaling, those of ECL has recently become available. Like ECL, CTL is a type of unsaturated logic, with high speeds obtained in part by avoiding the storage time associated with transistors coming out of saturation. The word "complementary" refers to the use of both npn and pnp transistors in the same direct-coupled circuit.

Fig. 6-9A shows one basic circuit, a two-input AND gate. In Fig. 6-9B, the basic circuit with only one input is repeated, simplified to aid in the analysis which follows. Notice the use of both pnp and npn transistors and also the fact that both a negative and positive power supply are required. Resistor R_{c2}, omitted in Fig. 6-9B, is a low-value resistor used to protect the output transistor in case the output is accidentally short-circuited.

In the circuits of Fig. 6-9, a logical 0 represents a voltage of approximately 0 volts, and a logical 1 a voltage of approximately 3 volts. The entire circuit of Fig. 6-9B operates like a



(A) Typical CTL two-input gate.



(B) Simplified circuit.

Fig. 6-9. Basic CTL circuits.

linear amplifier with a gain just less than unity. With 0 volts input, there is approximately 0 volts at the output; with 3 volts applied to the input, there is just under 3 volts at the output. Both Q_1 and Q_2 are connected as common-emitter stages, with voltage gains of nearly unity. Notice that except for the small voltage drop caused by base current through R_1, the input and output voltages are nearly equal, since the base-emitter voltage drops of the two transistors are approximately equal but of opposite polarity. This can be expressed mathematically by

$$V_o = V_{in} - I_{b1}R_1 + V_{be1} - V_{be2}$$

If $I_{b1}R_1$ is very small, and $V_{be1} \approx V_{be2}$, then

$$V_o \approx V_{in}$$

To show that neither transistor saturates, note that when V_{in} and V_o are zero, both emitters are at or near ground potential, while both collectors are at one or the other power supply voltage. In neither case is the collector-emitter voltage only several tenths of a volt, as would be the case for a saturated transistor. For $V_{in} \approx V_o \approx 3$ volts, the collector-emitter voltage of Q_1 is approximately 5 volts, while for Q_2 it is approximately $4.5 - 3$, or 1.5 volts.

The advantages of CTL are low cost and high speed. Because the output transistor operates as an emitter follower, the output impedance is low and the fan-out is high. The requirement for both positive and negative power supplies is a disadvantage. Although the basic circuit illustrated performs the AND logic function, a somewhat modified circuit produces a NOR gate which can also be used as an inverter. Since the AND gate gain is slightly less than unity, in larger systems additional signal level restoring gates must be included when several gates are cascaded.

Metal-Oxide-Semiconductor Logic

Although it is the newest of the logic families, MOS logic also called CMOS) is rapidly capturing a large share of the total market for integrated circuit logic devices. The popularity of MOS logic is due to a unique combination of characteristics, which include the following:

1. Very high input resistance, permitting high fan-out (up to 50).
2. Very low quiescent power requirements, allowing the design of low-power logic systems and large-scale integrated circuits whose gate densities are less restricted by heat dissipation problems.
3. Relativey high-signal swings—up to 10 volts.
4. Good noise immunity.
5. Relatively longer propagation delays.

Of these, only the last characteristic limits the application of MOS logic, preventing its use in high speed circuits, primarily computers, high speed data handling systems, and in specialized areas such as high frequency, vhf, and uhf frequency synthesizers. But MOS is ideal for a great many other applications. In battery operated equipment, the very low current requirement and insensitivity to power supply voltage changes is an especial advantage. MOS logic has been extensively designed into hand and desktop calculators, digital clocks and watches, numeral control, and automotive logic systems such as seat belt interlocks. Since the low heat dissipation permits large scale integration in small packages, and since package cost is a significant part of total IC cost, the cost per gate can be quite low.

Earlier MOS gates had propagation delays up to several hundred nanoseconds. However, continuous improvements have resulted in gates being available today with delays on the order of 30 to 50 nanoseconds—approximately the same as many RTL gates.

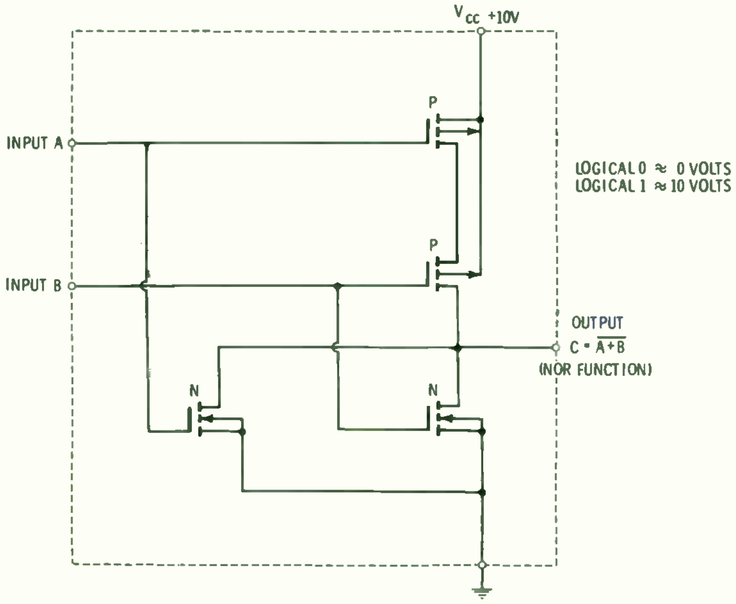
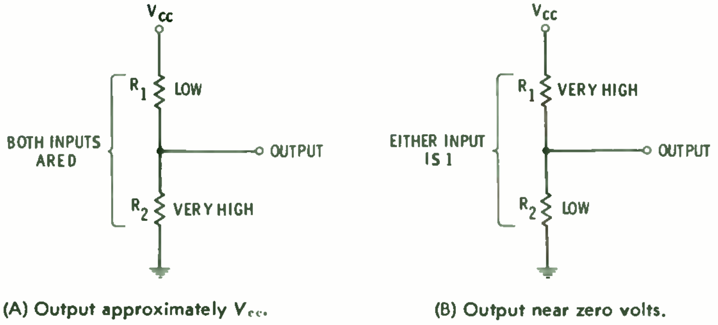


Fig. 6-10. Two-input complementary MOS gate.

Fig. 6-10 shows a two-input NOR gate, using complementary MOS transistors. Notice that there are no components other than the transistors themselves. The transistors are of two types, p-channel and n-channel, which is the reason the gate design is called "complementary." In the diagram, p-channel transistors are marked with a "P," and n-channel transistors are marked with an "N." The logical 0 signal range is from 0 volts to approximately 2 volts, and the logical 1 range is from about 7 volts to V_{cc}, which is typically 10 volts.



(A) Output approximately V_{cc}. (B) Output near zero volts.

Fig. 6-11. Dc equivalent circuit of complementary MOS gate.

In new designs, there is little reason not to begin with MOS logic, unless it is too slow. Although some MOS flip-flops will toggle up to 10 MHz., it may not be practical to use MOS in systems containing frequencies this high. The particular system must be examined to determine that each binary can toggle dependably, preferably without special selection, at the frequency or pulse rate at that point. Where a sequence of operations must occur within a certain period, the total propagation delay of all gates in the sequence must be less than the allotted period.

If MOS devices are too slow, then standard TTL, Schottky TTL, and ECL should be considered in that order (increasing speed). Mixing families—for example using part MOS and part higher speed devices—is practical and can save total system power, often simplifying power supplies. TTL and MOS devices can sometimes connect directly, without interface buffers or level translators.

Fig. 6-11 shows dc equivalent circuits of the gate for the two output states. In Fig. 6-11A, R_1 is relatively low while R_2 is very high (many megohms), and the output is accordingly approximately V_{cc}. In Fig. 6-11B, R_1 is very high whie R_2 is relatively low, and the output is near zero volts. In either case there is very little current drawn from the supply, since one of the two series resistors is high in value. At the same time, the output impedance for either output state is relatively low. Although a low output impedance is unnecessary for driving another MOS gate, since practically no dc driving power is required, the low impedance helps to charge or discharge capacitance at the output terminal, speeding up operation. The low output impedance also aids in improving noise immunity.

An example of a more complex MOS integrated circuit is the Type MM4620A 14-stage binary counter/frequency divider made by National Semiconductor. This device contains 14 binary flip-flops in one smal 16-pin package. It can count to, or divide by, $2^{14} = 16,384$. If a 60 Hz voltage were applied to its input, the output from the 14th flip-flop would complete a cycle in a little less than 5 minutes! Like most devices in this family, the MM4620A will operate over a supply voltage range of 3 to 15 volts, a five-to-one ratio. In contrast, many other logic families require that the power supply voltage be held to ±5%, requiring a reguated supply for dependable operation. The MM4620A will operate with input frequencies up to about 7 MHz.

Chapter 7

Using Off-the-Shelf Logic Elements

Whether your objective is to design new systems or to modify or service existing ones, you must frequently seek out data on IC logic devices. The problem of obtaining data is made difficult by the sheer number of models available. There are over a dozen manufacturers of "standard" digital IC's, and each makes literally hundreds and perhaps more than a thousand different kinds. The differences among some of these devices are relatively minor.

Sources of Data on Integrated Circuits

If you are to design a logic system you are faced with identifying the most suitable devices for your system out of the thousands that are on the market. Consideration of family characteristics, as discussed in the last chapter, will narrow the search. After this step, comparative or summary tables are more useful at first than the individual device data sheets. To go through hundreds of individual data sheets would be quite a formidable task. Tables summarizing general characteristics aid in selecting the most promising devices. Detailed data sheets can then be called on for the final selection and for use in detailed system design.

The primary sources of data on digital integrated circuits are the manufacturers of the devices. However, the larger electronic distributors' catalogs contain much information and should not be overlooked. In particular, they are a good source of comparative price information, although you must remember that prices are constantly changing. In the past several years it has generally been possible to purchase IC's at or below the prices listed in the distributors' catalogs. When prices are lowered, the distributors will often mail out announcements of this to those on their mailing lists.

The technical data in distributors' catalogs is usually dependable, often being supplied directly by the manufacturers, but it is seldom complete. A listing of a logic element is usually a good indication that it is really available and not being prematurely advertised. On the average, the information in these catalogs is one-half to a full year old, so that newer devices may not be listed.

Summary or comparative tables are published by each manufacturer for its products. These tables list a large number of devices, usually devoting one line to each device, with important characteristics given in a series of columns. Scanning down these columns, you can quickly select models meeting many of your requirements.

Detailed data sheets are useful for testing and troubleshooting, as well as being indispensable to the system designer. In addition to physical data (dimensions, pin numbers, etc.), the electrical characteristics will be defined as closely and as completely as the manufacturer wishes. Absence of information may mean the device is not tested for the characteristic in question, and if missing information is important, the manufacturer should be consulted. One data sheet may cover several models for which much of the data applies equally. An example of this would be a data sheet for DTL NAND gates which describes a quad two-input type, a triple three-input type, a dual four-input type, and a single eight-input type gate.

The minimum information on the detailed data sheet should include the following:

1. Description of logic function (NOR, NAND, etc.) and family (RLT, DTL, etc.).
2. Physical dimensions and package outline.
3. Identification of each pin (terminal) and its function.
4. Power supply voltage range and power requirements.
5. Temperature range.
6. Propagation delay (for gates) or maximum operating frequency (for flip-flops).
7. Loading or fan-out data; load that input imposes on a driving gate.

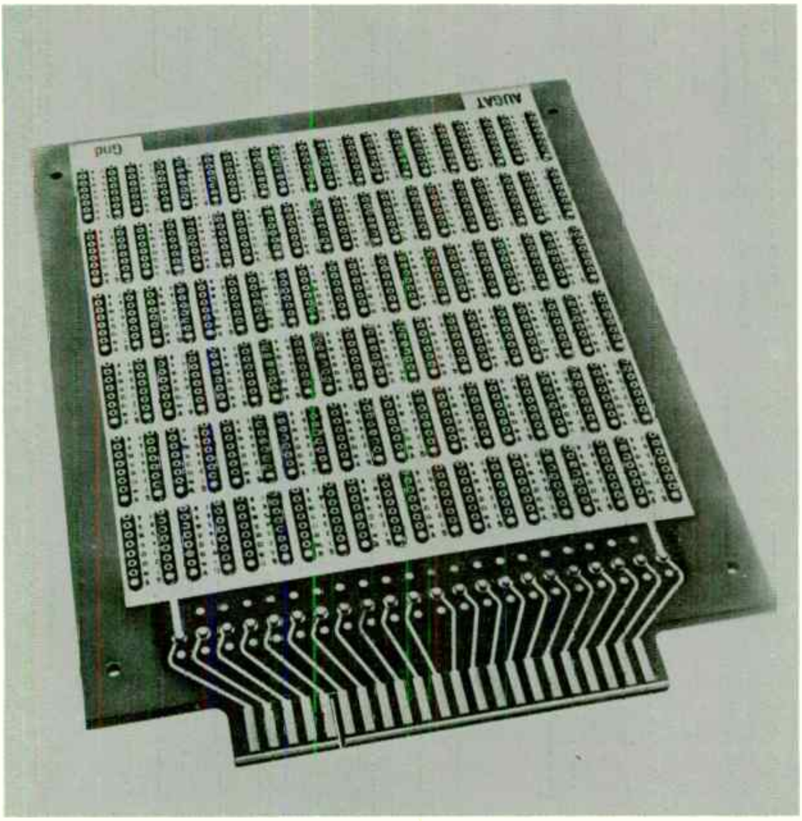
Also usually given are logic diagrams and equations, internal schematics, logic voltage levels, and curves of input, output, and transfer characteristics. Test procedures and limits are sometimes given, along with test circuits, to aid incoming inspection departments and minimize test discrepancies due to differences in test methods.

Almost all IC manufacturers publish applications notes to assist system designers, development engineers, and experimenters. The larger companies produce hundreds of these on a variety of subjects, varying in length from one to more than a dozen pages. These often discuss the operation of logic "subsystems" and are helpful to the service technician as well as to the designer. They can be obtained through local sales offices or representatives or through many electronic parts distributors. Most manufacturers publish a list of titles from which you can select those notes which are of most interest to you.

Breadboarding

The IC's in most finished systems are mounted onto one or more printed-circuit boards. As a designer, if you are planning on this construction, usually you will want to prove out your design, or at least major parts of it, before committing yourself to the fixed layout of a printed board. Breadboarding helps uncover design errors and oversights, timing problems, and other unexpected behavior. For simpler systems, experienced designers may be able to skip the breadboarding stage, especially if all loading is known to be light and speeds are well below the maximum speeds at which the devices are capable of operating. In high-speed systems, breadboarding is not the final answer, as layout details may affect the operating speed. Also, in a final design using a printed-circuit board, noise problems may be different from those in a breadboard using a much different physical layout.

For the convenience of designers and experimenters, a number of companies offer standardized breadboards (Fig. 7-1). These generally consist of a printed-circuit board with conductor areas and patterns specially designed to mount the common IC packages. In addition to the boards, the designer can choose among numerous sockets (Fig. 7-2) which accept the various IC packages. Using sockets permits the devices to be readily removed from one breadboard and used in another, as well as simplifying checking for component interchangeability in a system under study. Sockets may also be used in test circuits or equipment. For example, prior to permanent



Courtesy Augat, Inc.

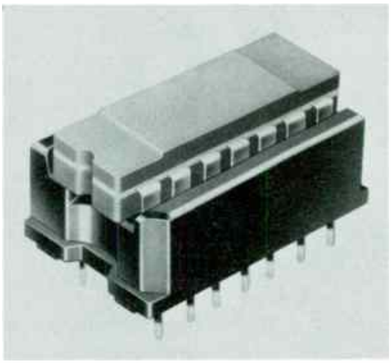
Fig. 7-1. Digital IC breadboard.

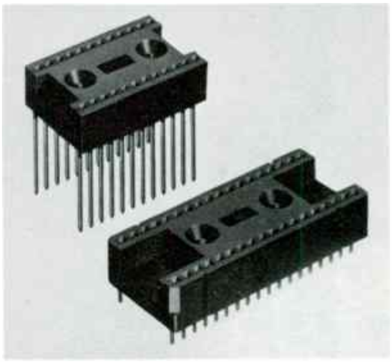
assembly an integrated circuit can be checked by plugging it into the proper socket in a duplicate of the circuit in which it is to be used.

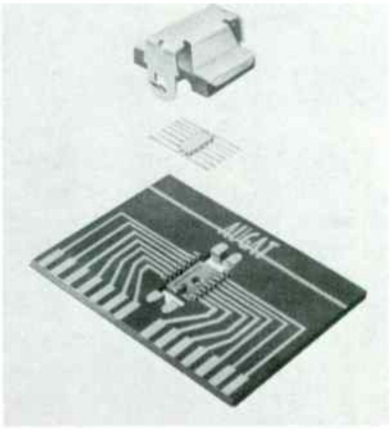
Fig. 7-3 shows a breadboard for a logic circuit. In this case, the integrated-circuit packages are soldered directly onto the same side of the board to which wire jumpers are soldered. It would also have been possible to mount the IC's on the other side of the board, with leads extending through small holes in the boards. This, of course, requires drilling several small holes. If sockets are used, they are mounted in this manner. Some experimenters will mount sockets in each position, on the back side of the breadboard, and then use the same breadboard over and over by rearranging the jumper wires and plugging in different sets of integrated circuits.

Two more breadboarding aids are shown in Fig. 7-4. The multiple socket in Fig. 7-4A accepts dual-inline IC's, capacitors, resistors, diodes, and transistors, all of which may be plugged in. Interconnecting wires may also be inserted directly into the individual sockets. Groups of individual sockets are permanently wired together internally. No soldering is needed to construct a circuit. In Fig. 7-4B, the multiple socket is combined with a panel and box containing a regulated 5-volt power supply, a clock oscillator, and monitoring lamps, providing a fast and simple way to breadboard digital circuits.

Noise problems due to power supply coupling are minimized if the breadboard and final system are operated from one or


(A) 14-pin dual in-line socket for PC mounting (IC shown in socket).


(B) 24-pin socket with wire-wrap terminals and 36-pin socket for printed boards.



(C) Test board and socket for 14-pin flat-pack integrated circuits.

Courtesy Augat, Inc.

Fig. 7-2. Sockets for digital integrated circuits.

more regulated power supplies. Low power supply impedances reduce noise coupled from transients in one part of the system to another part of the system. The breadboard power supply, including distribution busses, should be as similar as possible to the one that will be used in the final system.

Today it is practical to give each subpart of a system its own regulated power supply by using linear IC voltage regulators. At a cost of only a few dollars each, these regulators can supply up to 200 milliamperes of direct current, which will generally be enough to operate four to ten IC packages. Adding several external components, including a series control transistor, increases the current capability many times. Besides helping to avoid low-frequency noise, regulated supplies have the further advantage of reducing power supply voltage variations, and thus increase the dependability of the design for wide variations in the "raw" supply from the power lines or batteries.

Switching-type voltage regulators are suitable for many larger systems. Their high efficiency (up to 85 percent) results in less heat generation than occurs in series regulators. Switching regulators can be constructed with linear integrated circuits and a few external components to handle currents from 0.3 to 10 amperes.

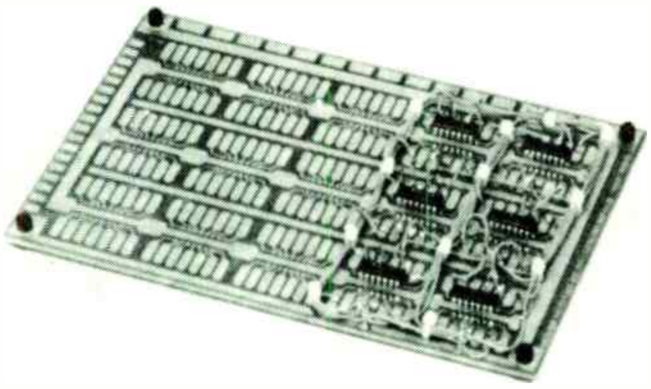
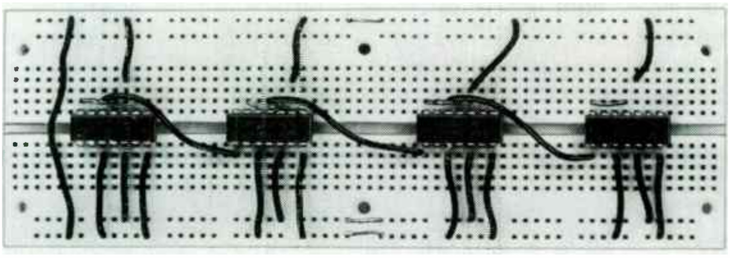
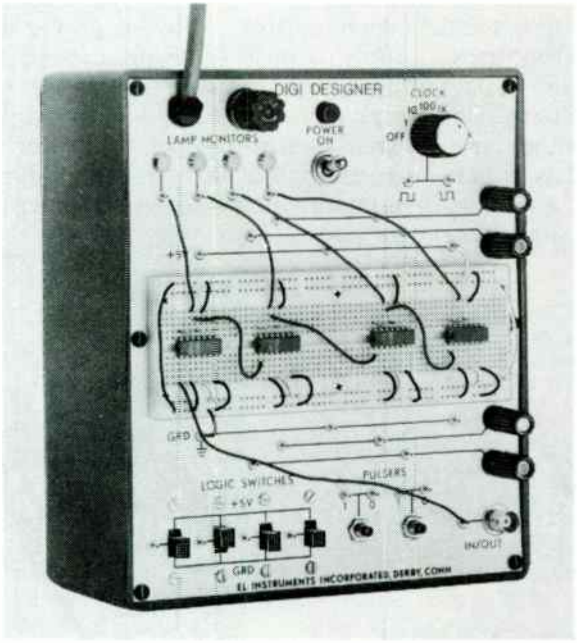

Courtesy Andresen Enterprises, Inc.

Fig. 7-3. Logic-circuit breadboard.

Fig. 7-5 shows both series-type and switching-type voltage regulators suitable for logic system power supplies.

In many systems, especially those containing flip-flops which can be triggered by very short noise pulses, it is necessary to use a number of bypass capacitors strategically located along


(A) Multiple socket.



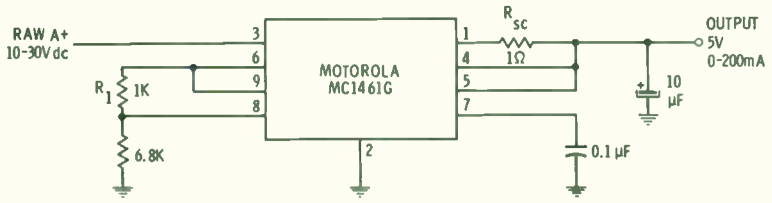
Courtesy EL Instruments Inc.

(B) Designers package.

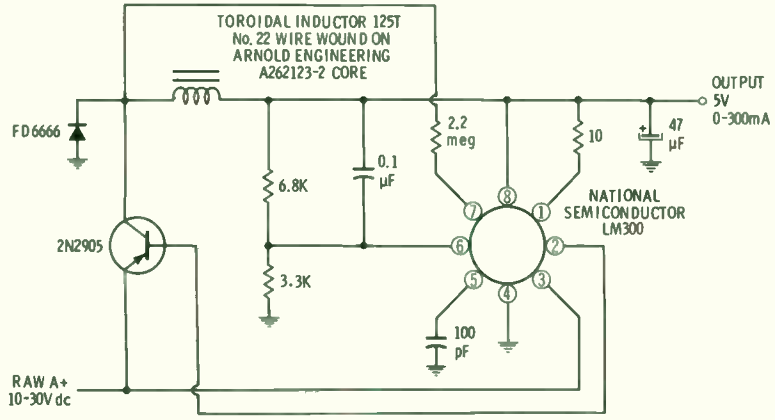
Fig. 7-4. Solderless breadboarding aids.

the supply conductors. It may even be necessary to use a capacitor for every two or three packages. Since these capacitors are intended to bypass short pulses, the same precautions should be observed as for hf or vhf bypassing: Keep capacitor leads as short as possible and run them directly between the supply and ground conductors near the IC's that are susceptible to noise pulses.

The pulses these capacitors are intended to eliminate are usually much too short to be captured on an oscilloscope, but can sometimes be detected by a test flip-flop whose trigger input is connected to points where noise pulses are suspected. Bypass capacitors can be added to see if they prevent the test flip-flop from triggering.



(A) Series regulator using linear integrated circuits.



(B) Switching-type voltage regulator.

Fig. 7-5. Logic-circuit regulated power supplies.

In breadboard and final design layout, ground and power supply conductors should be as large as practical, to minimize resistance and inductance.

In general, the faster a system operates, the more important it is to keep connecting conductors short. Slow systems can be physically spread out; faster systems require more compact layouts. However, even in slow systems, excessive "stray" capacitances and inductances can lengthen rise and fall times of pulses to the point that edge-triggered flip-flops will not trigger.

In complex breadboards where wires are soldered to terminals or printed-circuit pads, using teflon-insulated wire is less a luxury than one might suppose. Insulation on ordinary hookup wire is easily damaged by accidental contact with a soldering iron whereas teflon-insulated wire will withstand considerable abuse with less risk of accidental shorts and a neater appearance after debugging is completed.

As a final comment, careful breadboarding is recommended. Tracking down errors can be very time-consuming so that in the long run checking each connection as it is made is well worth the extra effort.

Testing

One way to classify tests performed on integrated-circuit logic elements is:

1. Tests performed before a device is installed in a system.
2. Tests performed after a device has been in use but is suspected of having failed.

In the first case the units to be tested can be plugged into test equipment which either measures important electrical parameters or checks its operation in a simulated system. In the second case unless the suspected unit can be readily removed from a plug-in socket, in-circuit tests may first be applied to determine whether the probability that the part is defective is sufficient to warrant removing it from the installation. Test procedures for the two cases will be discussed separately.

Some companies test logic elements as part of their standard incoming inspection procedure. The objective is to determine whether or not the devices meet the specifications in the purchase contract. Tests can be made on randomly selected samples and the entire lot either accepted or rejected on the basis of the results. If a certain percentage fail to meet specifications, then an additional sample may be taken. Sometimes devices are 100-percent tested before the shipment is accepted. However, since testing is expensive, many companies will not perform incoming inspection electrical tests unless they have had previous problems with a particular component or vendor.

Whether or not to perform incoming tests depends on how confident the company is that the devices will meet specifications, and on the seriousness of the problem created by installing a defective part. Modern digital IC's have proved themselves highly dependable and the percentage of as-received rejects is low, so that in many cases the cost of extensive incoming electrical testing is not justified. After all, the devices have been thoroughly tested in automated test equipment prior to shipment. On the other hand, when the IC's are to be installed in complex systems and soldered into printed-circuit boards, the expense of locating and replacing defective parts can be high. Should this occur several times, a company will probably decide that the cost of 100 percent incoming inspection is justified.

When a system is not operating properly and one of the IC elements is suspected of being defective, often the defective element can be identified with near certainty without removing it from the circuit. An IC that gets unusually hot may be defective, or may simply be reflecting a fault elsewhere. In-circuit voltage measurements at various gate input and output terminals often pinpoint an element which has failed.

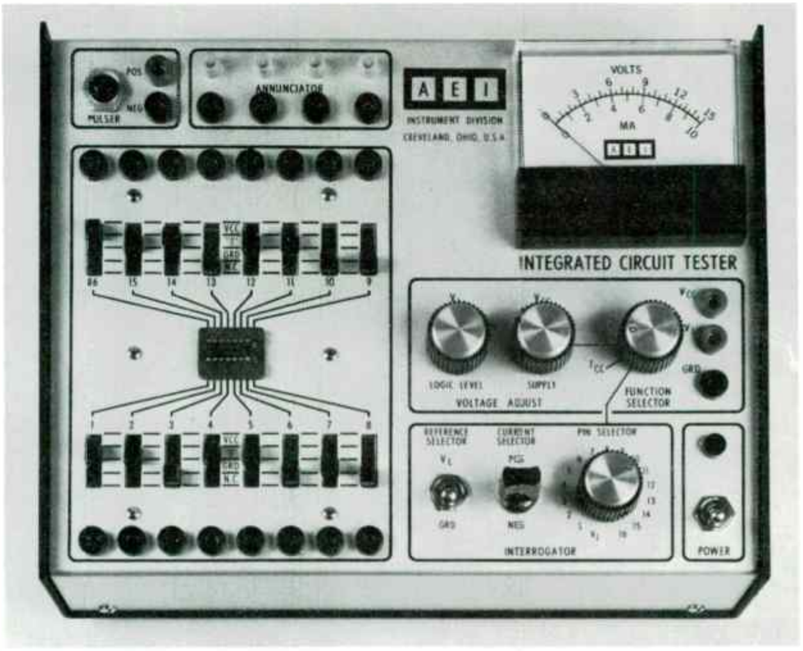
Equipment for testing digital integrated circuits varies widely in cost and complexity. The more testing there is to be done, the more elaborate and expensive the test equipment. Automatic testers, which step through a programmed series of tests and print out the test results, cost many thousands of dollars and are clearly beyond the range of the small individual user.

Low-cost testers are often quite adequate where the volume of testing is not too great. They require a moderate setup time and effort, which includes determining pin connections, connecting "patch" cables, adjusting voltages, and reading test results one-at-a-time on voltmeters and milliammeters.

Fig. 7-6 shows one low-cost integrated-circuit tester which can be used for 14- and 16-pin dual in-line packages (adaptor sockets available for other packages). By operating four-position slide switches, each terminal may be connected to ground, supply voltage, a 1 logic level set to any desired voltage, or left unconnected. A push-button pulser provides fast rise-time and fall-time pulses for triggering flip-flops. Patch cords can connect "annunciator" amplifier-lamp circuits to any terminal to indicate high or low logic levels. External resistive or capacitive loads may be connected.

Companies using only a few different types of IC's may prefer to construct their own test equipment exactly suited to their own needs. Instead of meter readings, lights may be used to indicate acceptable or not-acceptable performance, and various degrees of automatic operation may be built-in.

One method of testing logic devices is to have available a duplicate circuit with sockets into which each integrated circuit is inserted. A device that is suspected of being defective



Courtesy Andresen Enterprises, Inc.

Fig. 7-6. Digital integrated-circuit tester.

is substituted for the corresponding component in the properly operating duplicate circuit. If this causes the duplicate circuit to stop operating, the substituted part may be assumed to be defective.

Troubleshooting

A good troubleshooter is somewhat of an artist. While there are certain well-accepted and learnable techniques for troubleshooting, such as the principle of isolating the trouble to a small area of the system, some people have an almost uncanny knack of locating the problem very rapidly, while others will struggle along for hour after hour. The speed merchant is not aways lucky or the possessor of unusual intuition. Usually, behind his artistry lies experience and a practical understanding of electronic and logic theory.

While nothing can completely take the place of knowledge of how a system is supposed to work, the troubleshooter still needs other tools of his trade. These include an accurate schematic or circuit diagram, additional aids such as photographs or drawings to help in quickly identifying key components, and adequate test equipment.

The engineer or technician who must troubleshoot an experimental circuit faces complications beyond the problems of the service technician working on equipment known to have been working properly at one time. In the latter case, improper operation is nearly always due to failure of one or more components. The troubleshooter must identify these components and replace them. In the experimental circuit, however, there is the second possibility that the design is faulty. That these two possibilities more than double the troubleshooter's problems will be attested by many who have at one time or another tackled both "field failures" and problems in experimental systems.

When an experimental logic circuit does not operate as intended or expected, the source of trouble may be:

1. An outright wiring error—an omission or incorrect connection.
2. Careless wiring—frayed wire strands causing shorts, solder bridges, shorts due to burned or broken insulation.
3. A loading error, where a logic element is loaded too heavily.
4. An interfacing problem where different gate families meet, or where IC's drive or are driven by "discrete" circuits.
5. A timing problem, especially in systems containing flip-flops which operate in sequence. Sometimes delays must be added to prevent one or more gates from operating too soon.
6. One or more of the components may be defective.

In the case of a system which has been functioning dependably but suddenly develops trouble, the technician may assume that one of the components or a connection has failed. Sometimes failure of one part starts a chain reaction and other overstressed parts also fail, with the result that a number of parts must be replaced to get the system working again. These cases are often difficult; it is even possible to replace a part only to have it fail again immediately because trouble elsewhere has not yet been corrected.

Troubleshooting most logic systems does not require sophisticated test equipment. Many failures can be located using only a volt-ohm-milliammeter. First check power supply voltages. A low voltage does not necessarily mean the power supply has failed—it may simply be loaded beyond its capability by a defect in the system. Since this usually means a component somewhere is drawing an unusually high current, such a component can often be located by feeling its case temperature. Many digital IC's feel warm to the touch when operating properly, but not hot. A warm device indicates dissipation on the order of 100 mW. The device data sheet gives normal dissipation.

If power supply voltages are normal, logic levels at the various element inputs and outputs can be checked with a vom, provided the operation can be "stopped"—that is, all states maintained long enough to make the important measurements. In systems which operate sequentially this will require that the sequence be halted, then stepped one-at-a-time, making measurements for each step in the sequence.

In sequential systems an oscilloscope may be used to look for correct repetitive patterns in the pulses. The dc scope may also be used as a substitute for a voltmeter in static tests.

Other test equipment useful to the service technician are pulse generators, integrated-circuit testers, and specialized test equipment designed to be used with specific systems. Also available are test probes which indicate high and low logic levels with a lamp, or which detect pulses by the triggering of flip-flops inside the probe and indicate their state changes by lamps. The latter are especially helpful for detecting noise pulses. It is to be noted that J-K flip-flops are readily connected to detect which of two pulses occurs first—a useful tool in timing problems.

In the final analysis, the effectiveness of troubleshooting techniques is often more dependent on the ingenuity of the troubleshooter than on the sophistication of the test equipment he has available. In this connection, it is essential to understand thoroughly the operation of the logic system, and efforts to this end are nearly always well repaid.

Chapter 8

Experimenting With Logic Elements

This chapter is intended to provide a method for an easy transition from "book learning" to practical familiarity with integrated circuit logic elements. This is accomplished in two steps. First, an inexpensive logic demonstrator and breadboarding system which may be constructed with a few hours work is described. Then, experiments are presented which may be performed with the demonstrator-breadboarder using readily available and inexpensive digital integrated circuits. These experiments are designed to supplement material in earlier chapters of this book, both by reinforcing basic concepts and by providing additional practical insights.

Logic Demonstrator and Breadboarder

This piece of equipment has been used by the author as an educational aid. It is also a valuable aid in design work. It contains not only means for interconnecting logic elements quickly, but also a suitable power supply, signal generators, and simple but effective measuring "instruments." The complete system includes the following:

1. A 5 volt regulated power supply for TTL logic.
2. A set of five (or more) sockets into which 14- or 16-pin dual in-line IC packages may be inserted.
3. A set of four switches, with an output terminal for each, which may be used to generate four logic values. A 5-volt output is called a logical "1" (high) and a 0-volt or grounded output is called a logical "0" (low).
4. A set of four lamps, driven by transistors, which may be used to measure simultaneously the states of four logic lines. Each lamp is *off* if the measured voltage is less than 1 volt (low) and *on* if the voltage exceeds two volts (high).
5. A push-button "pulser," which generates a single "bounceless" 5-volt pulse.
6. A pulse generator which generates a repetitive signal having a rectangular waveform with fast rise and fall times, and whose repetition rate (frequency) may be varied over a wide range.

Fig. 8-1 shows a photograph of the equipment constructed by the author. The multiple socket, Model SK-10 manufactured by EL Instruments Inc., permits solderless wiring of bread-

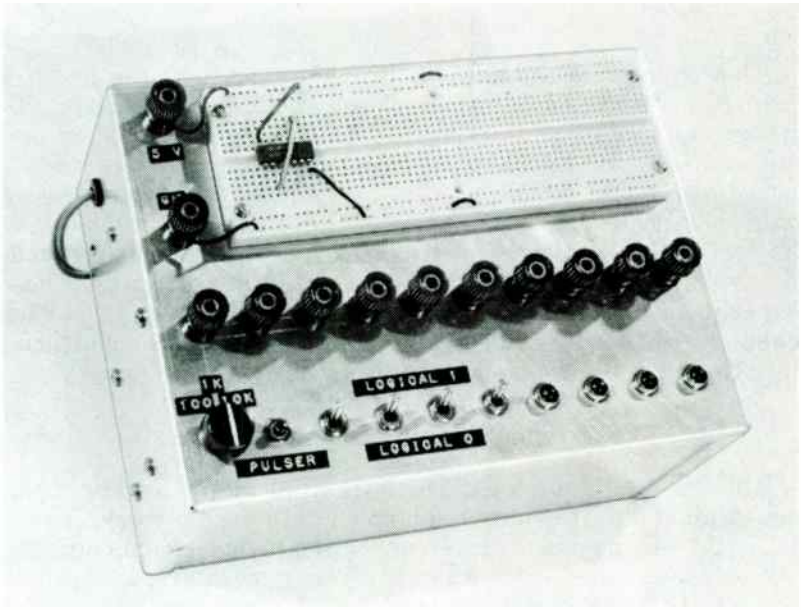


Fig. 8-1. Logic demonstrator/breadboarder.

board circuits, by pushing ends of No. 22 gauge solid wire into the holes in the socket. Dual in-line integrated circuits may also be inserted directly into the socket in a great variety of positions. The four toggle switches and four indicator lamps, and their associated output terminals, as well as the pulser push button, the pulse generator frequency control knob and their terminals, are all mounted along the lower edge of the aluminum chassis.

Athough offered by several manufacturers, the multiple socket may not be readily available to some readers. There are many good alternate methods of construction. One excellent method is to use dual in-line sockets mounted on socket-adapter printed circuit boards, such as Radio Shack catalog numbers 276-030 and 276-024. Mount the socket on the *copper* side of the board, so that the copper pads are available for soldering wires for connections, and the IC's may be inserted from the same side. It is then a simple matter to change from one breadboard to another.

If interconnecting wires for breadboarded circuits are not soldered, the terminals must be of a type which provide very dependable contact. It is most frustrating trying to demonstrate or study a logic system containing one or more loose connections.

Fig. 8-2 shows a typical panel layout for the demonstrator, similar to that used by the author. Of course, any convenient arrangement could be used.

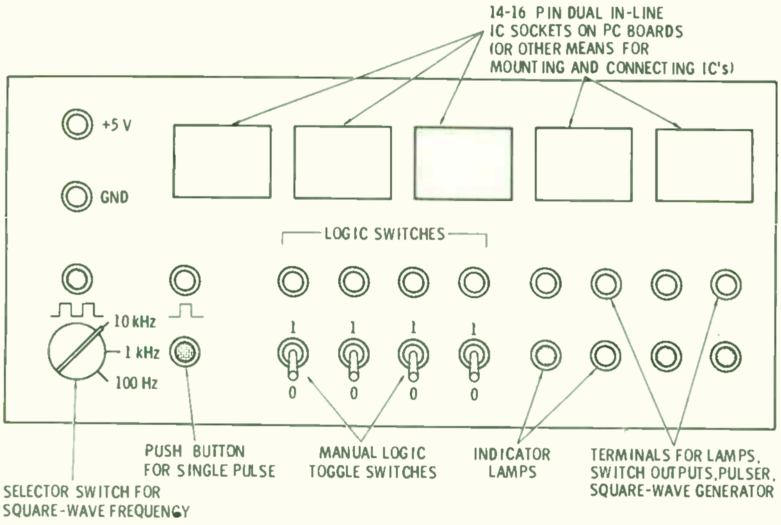

14-16 PIN DUAL IN-LINE IC SOCKETS ON PC BOARDS (OR OTHER MEANS FOR MOUNTING AND CONNECTING IC's)
+5 V
GND
LOGIC SWITCHES
10 kHz
1 kHz
100 Hz
1
0
PUSH BUTTON FOR SINGLE PULSE
MANUAL LOGIC TOGGLE SWITCHES
INDICATOR LAMPS
TERMINALS FOR LAMPS, SWITCH OUTPUTS, PULSER, SQUARE-WAVE GENERATOR
SELECTOR SWITCH FOR SQUARE-WAVE FREQUENCY

Fig. 8-2. Typical panel layout for logic demonstrator/breadboarder.

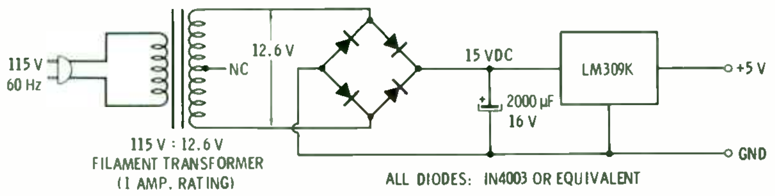
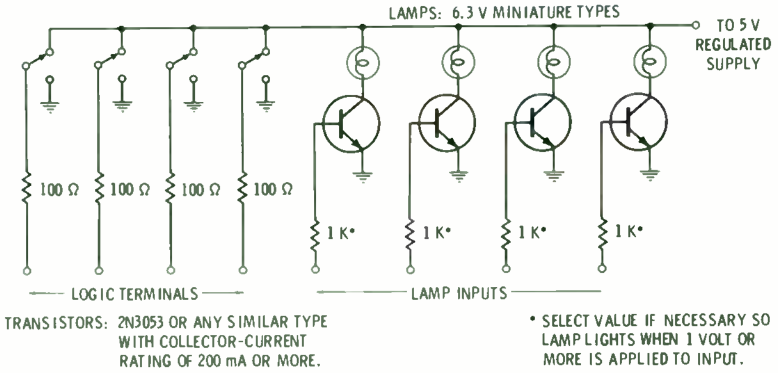


Fig. 8-3. Regulated 5-volt power supply circuit.

The circuit diagram for the 5-volt regulated power supply is given in Fig. 8-3. The center tap of the 12.6 V filament transformer is not used. It is necessary to have an unloaded input voltage to the LM309K of at least 12 V dc, so that regulation is maintained under load. This figure includes an allowance for ripple at the regulator input. The LM309K will supply up to one ampere at 5 volts output, and is short-circuit proof—a valuable feature when breadboarding.

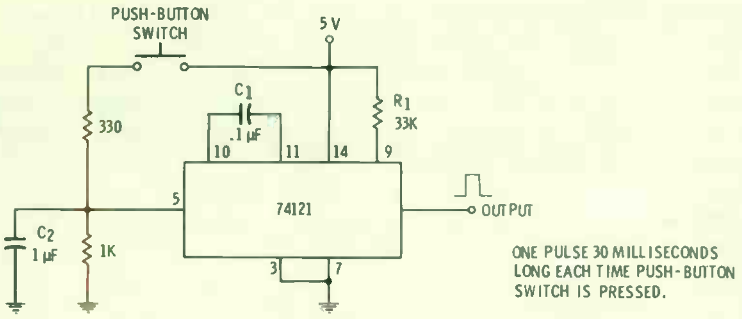
The circuit for the four logic switches and the four transistorized indicator lamps is shown in Fig. 8-4. While a wide variety of holders, lenses, and lamps are suitable, select lamps that require less than 150 milliamperes, in order to leave sufficient current capacity in the regulator to operate more complex circuits when breadboarding. Since the lamps work from only 5 volts, those types rated for 6.3 volts will draw considerably less than published current ratings. The lamps should light if the input exceeds one volt. To allow for different transistor gains, the base resistors may have to be selected.

In Fig. 8-5, circuits are shown for both the push-button pulser and the pulse generator. While there are many other ways to generate both single pulses and square waves or pulse trains, these circuits are simple and dependable, and produce

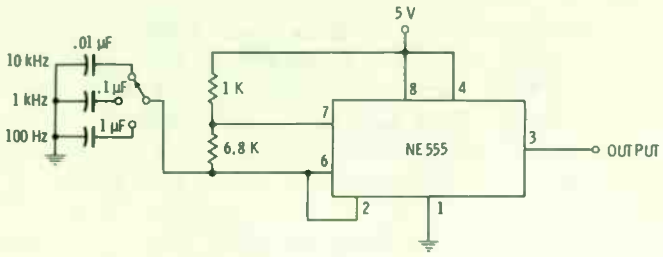


LAMPS: 6.3 V MINIATURE TYPES
TO 5 V REGULATED SUPPLY
100 Ω
100 Ω
100 Ω
100 Ω
1 K*
1 K*
1 K*
1 K*
LOGIC TERMINALS
LAMP INPUTS
TRANSISTORS: 2N3053 OR ANY SIMILAR TYPE WITH COLLECTOR-CURRENT RATING OF 200 mA OR MORE.
* SELECT VALUE IF NECESSARY SO LAMP LIGHTS WHEN 1 VOLT OR MORE IS APPLIED TO INPUT.

Fig. 8-4. Logic switch and lamp circuits.


PUSH-BUTTON SWITCH
5 V
C1
.1 µF
R1
33K
330
10
11
14
9
74121
5
OUTPUT
C2
1 µF
1K
3
7
ONE PULSE 30 MILLISECONDS LONG EACH TIME PUSH-BUTTON SWITCH IS PRESSED.

(A) Single-pulse generator.



(B) Square-wave generator.

Fig. 8-5. Single-pulse and square-wave generators.

excellent waveforms having fast rise and fall times. The NE555 timer or its equivalent is readily available at most electronics suppliers, as is the 74121 monostable multivibrator. As alternatives, a transistorized multivibrator may be used for the square-wave generator, and an R-S Flip-Flop can be substituted for the pulser (see Chapter 5).

In the square wave generator, frequencies of 10 kHz, 1 kHz, or 100 Hz are provided by selecting different values of capacity. Values other than those shown will give frequencies up to 100 kHz, or as low as 1 Hz. If desired, a variable resistor may be used in place of the 6.8K fixed resistor between terminals 6 and 7 of the NE555, permitting the frequency to be varied continuously between steps.

Each time the pulser push-button switch is pressed a single 30 millisecond positive-going pulse is generated. The pulse length is determined by capacitor C1 and resistor R1. When the push-button switch is closed, C2 charges toward 5 volts. The single pulse occurs when the trigger voltage for terminal 5 is reached. Without C2, bouncing of the switch contacts would cause multiple pulses to be produced.

In the logic switch circuits, the 100 ohm resistors provide protection against unintentionally shorting an output terminal, or against connecting the output to an IC terminal where a high current might damage the IC. As you construct breadboards, don't forget that TTL logic elements must have their inputs actually grounded in order to apply a logical "0." An open connection at a TTL input is equivalent to an applied logical "1"—in most cases the same as applying 5 volts.

After the demonstrator/breadboarder is constructed, it should be tested to make sure it operates properly. If an oscilloscope is available, examine the outputs of the pulser and square-wave generator. For the latter, the high and low periods should be nearly equal. Make sure the regulated supply voltage is between 4.75 and 5.25 volts, but use a voltmeter whose calibration you trust. Connect the four switched logic outputs to the four lamp inputs. All lamps should light when all logic switch outputs are high, and of course they should not light when the outputs are low. With all four lamps turned on, the regulated votage should not fall below 4.75 volts.

Getting Started

Whether your interest is that of servicing equipment, design, or as a hobbyist, a good starting point to gain practical experience is to experiment with individual logic elements. Once familiar with these, combinations can be tried in gradually increasing complexity.

The first requirement is a supply of digital integrated circuits. For this purpose, you can use almost any devices that you can obtain readily and economically, but for best compatibility with the demonstrator/breadboarder, TTL IC's are recommended. However, the breadboarder will operate satisfactorily with many RTL, DTL, and CMOS elements.

If you plan to buy an assortment, the following list is suggested. The listed devices are widely used, easily obtained, and can be used later in many projects.

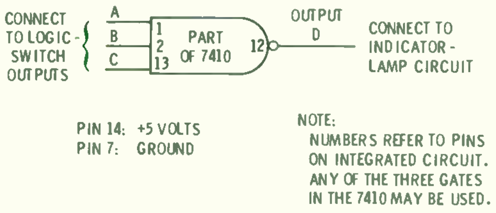
Type	*Function*	*Quantity*
7400	contains four 2-input NAND gates	4
7404	contains six inverters	1
7410	contains three 3-input NAND gates	1

7473	contains two J-K flip-flops	1
7490	decade counter	1
7448	converts BCD code to operate 7-segment numerical readout	1
MAN-4 or equiv.	light-emitting-diode Type 7-segment numerical readout	1

Experiment 8-1: Operation of NAND Gate

Part A—Basic Operation

Install a 7410 integrated circuit in one of the sockets of the demonstrator/breadboard system. Connect pin 14 to +5 volts, and pin 7 to ground. Select one of the three gates, and connect its output to one of the indicator lamp circuits. For the same gate, connect each of the three inputs to one of the logic switches. (See Fig. 8-6.) Verify the truth table experimentally, for all eight possible input combinations.



CONNECT TO LOGIC-SWITCH OUTPUTS
A
B
C
1
2
13
PART OF 7410
12
OUTPUT D
CONNECT TO INDICATOR-LAMP CIRCUIT
PIN 14: +5 VOLTS
PIN 7: GROUND
NOTE: NUMBERS REFER TO PINS ON INTEGRATED CIRCUIT. ANY OF THE THREE GATES IN THE 7410 MAY BE USED.

(A) Diagram.

INPUTS			OUTPUT
A	B	C	D
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(B) Truth table.

Fig. 8-6. Connections and truth table for NAND gate.

Part B—NAND Gate as Enable/Disable Gate

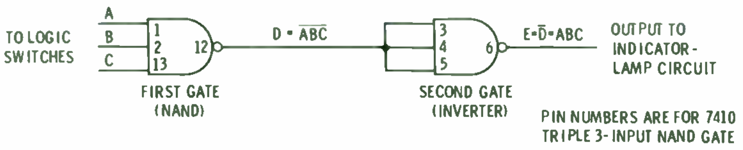
Place the switch for input "A" in the logical "1" position. Now refer to the truth table: if the switch for input "B" is in the "0" position, then the output is high regardless of the signal on input "C." This is the *disable* condition. But if the switch for input "B" is in the "1" position, the signal at input "C" passes through, although it is inverted as it does so. This is the *enable* condition. Verify these facts experimentally.

Part C—NAND Gate Connected as an Inverter

Connect gate inputs A, B, and C together, and then to one of the logic switch terminals. Using a lamp indicator to show the state of the gate output, observe what happens as the input is switched between "0" and "1." Any NAND or NOR gate acts as an *inverter* if the inputs are tied together.

Part D—AND Gate From NAND Gate Plus Inverter

Connect one of the NAND gates in the 7410 as an inverter, following another NAND gate, as in Fig. 8-7. The inputs to the first NAND gate are connected to logic switch outputs. The inverter output is connected to an indicator lamp circuit.



(A) Diagram.

INPUTS			OUTPUTS	
A	B	C	D	E
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

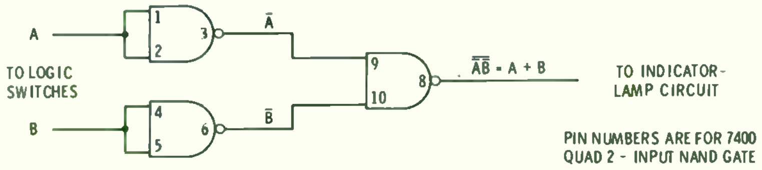
(B) Truth table.

Fig. 8-7. An AND gate from NAND gates.

Show experimentally that this combination does function as a three-input AND gate. Also observe that this combination may be used for enabling or disabling a signal, except that in this case the signal passes through inverted *twice*, which is equivalent to not being inverted at all.

Part E—OR Gate Made From NAND Gates

Change the integrated circuit from a 7410 to a Type 7400, a quad 2-input NAND gate. To construct a gate combination which functions as an OR gate requires using three of the four gates available in the 7400. The connections are shown in Fig. 8-8. Note that two of the gates are used as inverters. Examine the Boolean expressions, referring to De Morgan's Laws to develop the output expression A + B. Using switches and lamps as in previous experiments, confirm the OR truth table with this combination of gates.


A
B
TO LOGIC SWITCHES
$\bar{A}$
$\bar{B}$
$\overline{\bar{A}\bar{B}} = A + B$
TO INDICATOR-LAMP CIRCUIT
PIN NUMBERS ARE FOR 7400 QUAD 2 - INPUT NAND GATE

(A) Diagram.

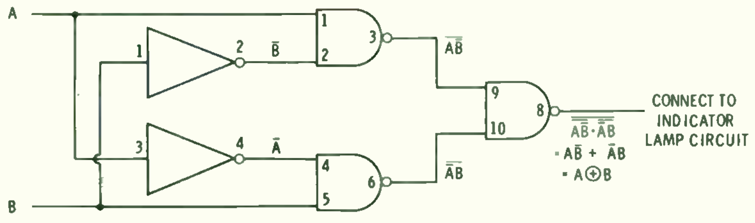
A	B	$\bar{A}$	$\bar{B}$	$\overline{\bar{A}\cdot\bar{B}}$
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

(B) Truth table.

Fig. 8-8. An OR gate constructed from NAND gates.

Part F—Exclusive-OR Function from NAND Gates

The truth table for the exclusive-OR function, along with the correct gate interconnections, are presented in Fig. 8-9. Observe the difference between the OR function and the exclusive-



(A) Diagram.

A	B	OUTPUT = $A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

NOTE: $\oplus$ IS SYMBOL FOR EXCLUSIVE - OR.
PIN NUMBERS REFER TO:
7400 QUAD 2 - INPUT NAND GATE
7404 HEX INVERTER

(B) Truth table.

Fig. 8-9. The exclusive-OR function from NAND gates and inverters.

OR function. In the latter case, the output is high if one or the other input is high, but not if both inputs are high. Construct the circuit as shown and verify its operation.

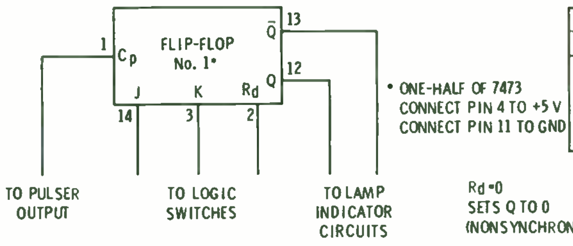
The exclusive-OR function is an important part of circuits which add binary numbers.

Experiment 8-2: The J-K Flip-Flop

Part A—Single Stage Operation

Install a 7473 dual J-K flip-flop in a test socket and connect as in Fig. 8-10. Set the reset (R_d) logic switch to the high position.

Observe the behavior of the flip-flop for the four possible combinations of the J and K inputs, and compare your observations with the truth table in Fig. 8-10. In the table, t_n refers to the status of the inputs just before the clock pulse is applied; the last column gives the state of the output *Q* just after the clock pulse. The symbol t_{n+1} in Fig. 8-10B means one clock pulse after t_n.



(A) Diagram.

t_n		t_n+1
J	K	Q
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

Rd = 0
SETS Q TO 0
(NONSYNCHRONOUS)

(B) Truth table.

Fig. 8-10. Connections for studying the J-K flip-flop.

Next set R_d to the low state. Output *Q* will now go to or remain in the low state, as the indicator lamp will show, regardless, of the J, K, or clock inputs. The reset input is the only *nonsynchronous* input; that is, if *Q* is not already low, it will go low immediately upon setting the reset input low, without waiting for the next clock pulse. The J and K inputs are *synchronous* inputs—changes in *Q*, as controlled by these inputs, occur only at the next clock pulse.

Toggle operation occurs with J, K, and R_d all high. The output changes each time a clock pulse is applied. Note that for the 7473 flip-flop, an output for Q is provided.

It takes two operations of the pulser for the flip-flop to change state and back again. If a repetitive rectangular wave is applied to the clock input, the frequency of the flip-flop output will be half the frequency of the input signal. For this reason, the flip-flop may be called a *frequency divider* when operated in the toggling mode. Even if the input signal consists of repetitive short pulses, the output will have the form of a square wave.

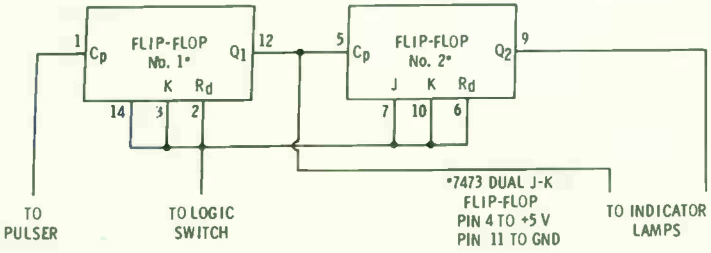
Part B—Ripple-Through Counter

Connect the 7473 dual J-K Flip-Flop as in Fig. 8-11. Output Q_1 of the first flip-flop is connected to an indicator lamp and also to input C_p of the second flip-flop. The output of the second flip-flop (Q_2) is connected to a second indicator lamp. In order for toggling to occur, the logic switch must be in the high position.

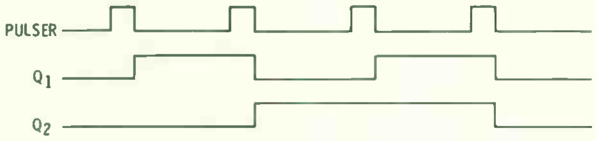
To "reset" the flip-flops, move the logic switch to the low position. Both flip-flop outputs will go to their low states, if they are not already there. Then place the logic switch in the high position, which allows the pulser to toggle the first flip-flop. Now observe the progression of the two outputs as the pulser button is pressed repeatedly. This progression will be:

	Q_1	Q_2
start	0	0
after pulse 1	1	0
after pulse 2	0	1
after pulse 3	1	1
after pulse 4	0	0

Every fourth pulse returns the outputs to their original states. The combination of the two flip-flops forms a simple *counter*, which can "keep track" of the number of times the pulser button was pressed after resetting—up to the capacity of the counter, which is three in this case. This circuit can also be



(A) Diagram.



(B) Timing.

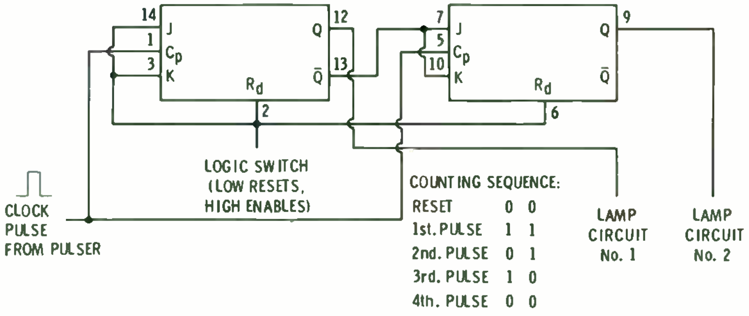
Fig. 8-11. Two-step ripple-through counter.

used as a frequency divider; if a periodic rectangular waveform is applied to C_p of the first flip-flop, square waves of one-half and one-quarter the input frequency respectively will appear at Q_1 and Q_2. Because the output of the first flip-flop is used to trigger the second flip-flop, the combination of Fig. 8-11 is called a ripple-through counter. Note that when counting the two lamps display the binary numbers for 0, 1, 2, and 3 in sequence.

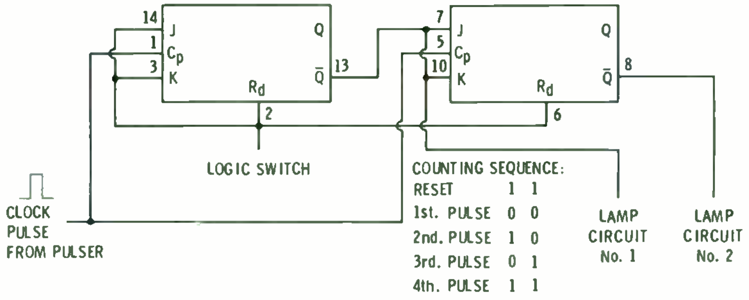
Part C—The Synchronous Counter

The synchronous counter differs from the ripple-through counter in that in the former the clock pulse is applied simultaneously to all flip-flops. Whether or not each flip-flop toggles depends on the logic signals applied to the J, K, and reset inputs at the time the clock pulse is applied.

Fig. 8-12 shows how the 7473 dual J-K Flip-Flop may be connected in two different ways to demonstrate synchronous counting. In Fig. 8-12A, the lamps connect to the Q outputs. Momentarily moving the logic switch to low resets both outputs



(A) Outputs connected to Q.



(B) Outputs connected to $\overline{Q}$.

Fig. 8-12. Synchronous counters from the 7473.

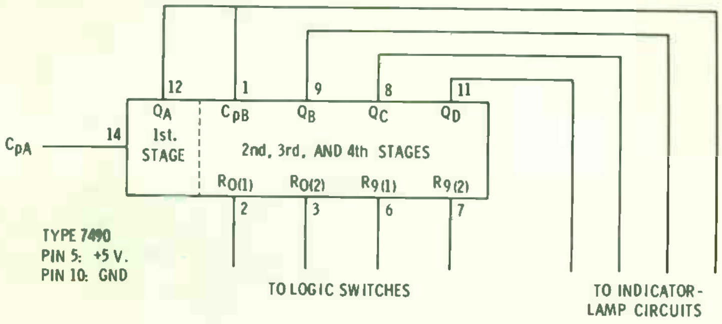
to "0," but the counting sequence is different than for the ripple-through counter—in fact, it could be described as "counting down." The first counter toggles on each clock pulse. The second counter toggles only when both J and K inputs are high, which occurs only when the output of the first counter is already low.

In Fig. 8-12B, the outputs are taken from $\overline{Q}$ instead of Q, for both flip-flops. This gives a normal "up" counting sequence, but operating the reset returns both outputs to their high states instead of low.

The advantage of the synchronous counter is that in certain applications it can operate at higher frequencies than the ripple-through counter. This results from avoiding the time delays of previous stages before toggling of later stages can occur.

Part D—The Decade Counter

Install a Type 7490 decade counter in a test socket and connect as shown in Fig. 8-13. Note that this counter is divided into two sections. The first section is a single J-K flip-flop,



(A) Diagram.

RO (1)	RO (2)	RQ (1)	RQ (2)	OUTPUT
1	1	0	x	0000
1	1	x	0	0000
x	x	1	1	1001
x	0	x	0	COUNT
0	x	0	x	COUNT
0	x	x	0	COUNT
x	0	0	x	COUNT

(B) Logic control table.

0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1

(C) Counting sequence.

Fig. 8-13. Decade counter connections.

while the second is a 3-stage counter in which the flip-flops are interconnected so that in combination they count or divide by five. This permits two modes of operation. If the signal is applied first to the single flip-flop, whose output is connected to the count-by-five stages, the complete circuit functions as a decade counter. The output sequence is the set of binary numbers from 0000 to 1001 (0 to 9), after which the sequence repeats. This connection also performs as a divide-by-ten frequency divider, but the output, taken from the fourth flip-flop at Q_D, is in the form of relatively short pulses. If the signal is applied first to the divide-by-five section (via terminal C_{pB}), and Q_D is in turn connected to the clock input of the single flip-flop, the circuit still acts as a divide-by-ten frequency divider, but in this case the output is a square wave. Of course, the counting sequence is no longer the BCD numbers from 0 to 9.

Observe the significance of the four logic inputs for the connections in Fig. 8-13. The counter has three modes of operation: reset to "0," reset to "9," and count. While there are several combinations of logic settings that reset to "0" or permit counting, resetting to "9" occurs only with $R_{9(1)}$ and $R_{9(2)}$ high. Therefore, a convenient setting of the logic inputs is:

$$\begin{aligned} R_{0(1)} &= \text{high} \\ R_{0(2)} &= \text{low} \\ R_{9(1)} &= \text{high} \\ R_{9(2)} &= \text{low} \end{aligned}$$

Now if a short high pulse is applied to $R_{0(2)}$ the outputs are reset to "0." If a short high pulse is applied to $R_{9(2)}$ the outputs are set to "9." Otherwise, the four outputs sequence through the binary numbers for 0 to 9, advancing once for each clock pulse.

The 7490 can be connected to divide by other numbers than ten. In fact, without additional gates, it can readily divide by any number from two to ten. This is accomplished by connecting different outputs back to one or more of the four logic control inputs.

An example of this is shown in Fig. 8-14, the operation of which may be verified with your demonstrator-breadboard. To understand the operation, refer to the first line of the logic control table (Fig. 8-13). Note that if $R_{9(1)}$ is set to "0," then a high logic signal applied to both R_0 inputs will cause the counter to reset to 0000. If Q_B and Q_C are connected to $R_{0(1)}$ and $R_{0(2)}$, this reset condition occurs upon the sixth pulse. The net effect is for the counter to count from 0000 to 1010, and then on the next clock pulse to reset to 0000. The output should be taken at Q_C; it is not a square wave, but rather a series of pulses with an "up" duty cycle of 33%.

The divide-by-six counter is of special interest in clocks and timing devices because it is frequently necessary to divide the 60-Hz power-line frequency down to one pulse per second, or one per minute. Division by sixty can be achieved with one 7490 used to divide by six, followed by a second unit dividing by ten.

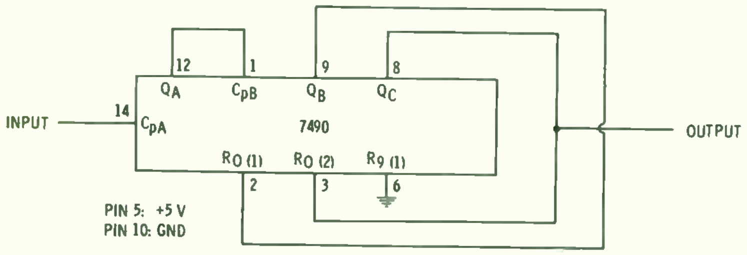


Fig. 8-14. The 7490 connected to divide-by-six.

Experiment 8-3: The Seven-Segment Numerical Readout

There are a great variety of numerical displays (readouts), the most common being these:

1. *LED (Light-Emitting-Diode) Types.* LED's are usually red in color, but may be yellow or green. They are *polar* devices; that is, the correct polarity of voltage must be applied to each segment. It is helpful to visualize each segment electrically as consisting of a diode or two diodes in series. The current must be limited, usually by a resistor, or the diode may be destroyed. Most LED readouts operate with about 15-20 milliamperes of current per segment.
2. *Cold Cathode or Glow Discharge Types.* The color is orange or red. They require a relatively high voltage—up to 200 volts—but very little current.
3. *Incandescent Types.* Incandescent displays contain tiny filaments, like electric lamps. They are nonpolar (current may be in either direction or ac), and can be made any color by covering with transparent color filters. External resistors are usually not required.
4. *Liquid Crystals.* Liquid crystal displays do not create their own light, but either use front incident light or may

be illuminated from the rear. They require very low power (ac).

The seven segment configuration has become widely used, because it represents the simplest arrangement that forms readily identifiable figures for the decimal digits 0 through 9. Such readouts have eight leads—one for each segment plus a common terminal. Some may also have a terminal for a decimal point.

The method of connecting to the display depends on the particular model to be used. Some models plug directly into a dual in-line socket. If you experiment with a readout which does not do this, you may have to solder wires to the leads.

Part A. Tests on LED 7-Segment Readout

The following discussion assumes that you are testing a MAN-4 (or equivalent) LED. Insert the readout in a socket of the demonstrator/breadboarder. Connect pin 4 (common cathode) to ground. Connect any anode (such as anode F, pin 1) to a 150-ohm resistor and connect the other end of the resistor to +5 volts (see Fig. 8-15). Observe the segment light up and note its brightness. Since the voltage across the segment is approximately 1.7 volts, the voltage across the 150-ohm resistor is $5 - 1.7 = 3.3$ volts. The current through the diodes and resistor is

$$\frac{3.3 \text{ volts}}{150 \text{ ohms}} = 22 \text{ mA}.$$

If you have a small piece of transparent red plastic, cover the face of the readout and observe the improved contrast between the illuminated segment and the background.

Using additional 150-ohm resistors (one for each segment), illuminate other segments to form numbers or letters.

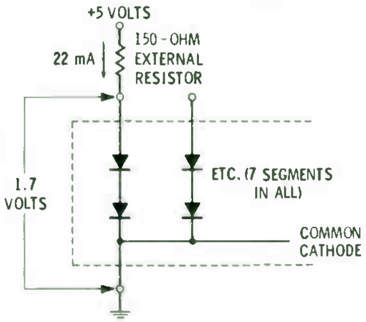


Fig. 8-15. Representation of an LED segment.

Part B. Driving Display From BCD/7-Segment Decoder

The decoder has four inputs, to accept a BCD code corresponding to the number 0-9. It has seven outputs, one per segment of the readout. Most decoders also have one or more control inputs, such as a blanking input which can disable all seven output simultaneously.

Decoders vary in the following ways:

1. They act as current *sources* or current *sinks*. In a current source, electrons flow *into* the output terminal of the decoder. In a sink, electrons flow outward.
2. Internal current limiting resistors may or may not be provided.
3. Current capacity—some decoders can't drive displays without additional external transistors.

If you are using a MAN-4 readout (common cathode), a suitable BCD/7-segment decoder is the Type 7448, which is of the current source type. For common anode readouts, use the 7447. If you have an incandescent readout, either the 7447 or 7448 can be used. For the 7447 you must connect the common terminal to the positive side of the supply; for the 7448 the common terminal of the readout is grounded. For 5-volt incandescent readouts, no current limiting resistors are required, although they are sometimes used to extend readout life.

Connect your decoder and readout according to the foregoing guidelines and individual instructions accompanying each. Don't forget to use current limiting resistors (between 100 and 200 ohms) for each LED segment. The logic switches on the demonstrator/breadboard may be used to generate BCD inputs to the decoder. Another interesting demonstration is to connect the decade counter outputs to the BCD inputs of the decoder. As pulses are applied to the counter, the display will step through the digits 0 through 9.

Appendix 1

Decimal–Binary–Binary-Coded-Decimal Conversion Table

Decimal	Binary	BCD
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101
16	10000	0001 0110
17	10001	0001 0111
18	10010	0001 1000
19	10011	0001 1001
20	10100	0010 0000
21	10101	0010 0001
22	10110	0010 0010
23	10111	0010 0011
24	11000	0010 0100
25	11001	0010 0101
26	11010	0010 0110
27	11011	0010 0111
28	11100	0010 1000
29	11101	0010 1001
30	11110	0011 0000
31	11111	0011 0001
32	100000	0011 0010

Appendix 2

Glossary

active element—A transistor or diode; in a circuit, an element which amplifies or directs a current.

adder—A logic circuit which adds binary digits, having three inputs (addend, augend, carry-in) and two outputs (sum, carry-out).

address—A binary code applied to a memory circuit input to obtain at the memory output, part of the information stored in the memory.

analog computer—A class of computers in which variables, usually functions of time, are represented by electric currents or voltages which can have an infinite or very large number of values.

AND—A logic operation such that all inputs must have the same logic value (true or false, 0 or 1, etc.) to make the output equal to any of the inputs.

asynchronous—Not synchronized or not occurring simultaneously with clock pulses.

asynchronous inputs—Inputs to a flip-flop which affect the output independently of the clock input.

binary arithmetic—Addition, subtraction, multiplication, and division and the rules and methods of performing these operations on binary numbers.

binary-coded decimal—Any of a set of binary codes used to represent decimal numbers by groups of four binary digits.

binary digit—Either of the two symbols used to form binary numbers, usually 0 or 1.

binary logic—A mathematical system, originally formulated to define rules for logical reasoning and thought, where all values have only two possible states (true or false, 0 or 1, etc.); also, two-state electronic circuits or devices whose operation can be symbolized by this mathematical system.

binary number—Numbers containing only binary digits.

binary operation—In mathematics, the determination of a third number, given two other numbers; example: addition of two numbers.

binary signal—A voltage or current which carries information by varying between two possible values.

bistable element—An electronic circuit having two stable states with the ability to change from one to the other when the proper input signal is applied, and which will remain in the new state even after the signal is removed.

bit—Abbreviation of *bi*nary digi*t*; also, the smallest unit of information, representable by a binary digit or the value of any two-state quantity.

black box—An electronic circuit considered only in terms of its input/output relationships without regard to the circuit details.

Boolean algebra—A self-contained mathematical system in which all quantities can have only one of two values, originally formulated to describe logical reasoning.

buffer—A circuit used to permit a signal source to drive a larger load than it otherwise could; also, a circuit used to convert binary signals from one pair of voltage ranges to another so that otherwise incompatible circuits can be interconnected.

chip—The piece of silicon upon which integrated circuits are formed.

clear—To place a flip-flop output in the zero state by applying the proper signal to the clear terminal; also, the name of the asynchronous input terminal used for this purpose.

clock—A pulse generator whose signals control the timing and switching rate of logic circuits; also, the name of the input terminal of a logic element to which clock pulses are applied.

CML—Abbreviation for current-mode logic.

code—An ordered collection of symbols having a specific meaning; more specifically, one or more ordered binary digits which carry instructions to a logic circuit; also, any of various systems for representing decimal numbers by a series of binary digits.

comparator—A two-input logic circuit whose output indicates whether the binary inputs are the same.

compatible family—A set of logic elements having the same ranges of voltages for the 0 and 1 states, and which may be interconnected without buffers.

complement—To change a binary digit or value from its present state to the other possible state; also, the opposite state from the present one.

complementary-transistor logic—Logic circuits using both pnp and npn transistors.

counter—A logic circuit containing one or more flip-flops, with an output accessible for each flip-flop, wherein application of input pulses causes the combination of output states to change in a specific sequence.

counter, parallel—A counter in which all flip-flops can change state at the same time, and which accordingly requires a relatively short time to respond to an input pulse.

counter, ring or shift—A particular kind of parallel counter in which binary values are shifted from each flip-flop to the following one, and from the last flip-flop back to the first one.

counter, ripple-through or serial—A counter in which the second flip-flop cannot change state until after the first flip-flop has changed, and the third can only change after the second has changed, etc., so that relatively long delays can occur before all counters have reached their final states following an input pulse to the counter.

CTL—Abbreviation for complementary-transistor logic.

current-mode logic—A kind of logic circuit in which input signals shift currents from one path to another, the total current remaining nearly constant, and in which high speed is obtained by not allowing the transistors to saturate.

data—A set of symbols carrying quantitative information about something; in digital systems, a string of 0's and 1's representing such information.

decimal digit—Any of the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9.

decimal number—A number using one or more decimal digits.

decoder—A logic circuit which converts one code into another one, usually a binary into a decimal code.

delay—The time between application of a signal at the input of a circuit and the appearance of the resulting signal at the output.

De Morgan's laws—Two useful Boolean algebra equations relating AND, OR, and NOT functions.

digital circuit—An electronic circuit where signals are intended to have either of two distinct voltages or currents instead of a continuous range of values.

digital computer—An electronic system containing an input or read-in means, an output means, a relatively extensive electronic memory, and binary logic circuits, and capable of making logical decisions and performing numerical computations at great speed and with high accuracy.

diode logic—An electronic circuit using current-steering diodes such that the relations between input and output voltages correspond to AND or OR logic functions.

diode matrix—A set of diodes connected in an orderly array or pattern.

diode-transistor logic—An electronic circuit, containing both diodes and transistors, whose input-output relations correspond to a binary logic function.

direct-coupled amplifier—An amplifier containing no coupling capacitors, capable of amplifying dc as well as ac signals.

disable—To prevent passage of binary signals by applying the proper signal to the disable control terminal.

discrete—Containing complete individual components as opposed to integrated circuitry.

divider—A logic circuit containing flip-flops which produces one pulse output after a certain number of input pulses have been applied. The circuit is similar to a counter, except that only one flip-flop output may be available.

dot AND—Two or more gates whose outputs are connected together so that the combined output is a 1 only if both individual gate outputs are 1.

dot OR—Two or more gates whose outputs are connected so that the combined output is a 1 if either individual gate output is 1.

driver—An amplifier or gate designed to supply larger-than-usual output currents (high fan-out).

DTL—Abbreviation of diode-transistor logic.

dual-in-line—A type of case or "package" for integrated circuits in which the terminals are arranged in two parallel rows, usually seven or eight terminals per row (14 or 16 terminals in all).

ECL—Abbreviation for emitter-coupled logic.

edge-triggered flip-flop—A kind of flip-flop having as one condition for an output change that the clock-signal rate of change be at least some minimum amount.

electronic gate—A device whose input-output relations correspond to a Boolean algebra function (AND, OR, etc.), and which accomplishes this using diodes and/or transistors rather than relays or mechanical means.

emitter-coupled logic—A fast electronic gate or flip-flop whose design is such that the transistors are never saturated, thereby avoiding the storage time delay.

enable—To allow passage of binary signals by applying the proper signal to the enable control terminal of a gate or flip-flop.

encoder—A logic circuit which converts binary data from one code to another (often decimal to binary or BCD).

exclusive-OR gate—A two-input logic circuit whose output is 1 only if one or the other input (but not both) is 1.

expandable gate—A gate whose number of inputs may be easily increased by adding external circuitry.

expander—The circuit which is added to increase the number of inputs to an expandable gate.

fall time—The time required for a signal to change from a 1 to a 0; often, more precisely, the time for the signal to fall from 90% to 10% of the nominal 1 voltage.

fan-in—The number of input terminals to a gate.

fan-out—The number of gate inputs that a particular gate can drive as loads.

field-effect transistor or FET—A three-terminal semiconductor amplifying device having extremely high input resistance.

flat pack—One kind of package or case used to enclose an integrated-circuit chip.

flip-flop—Same as bistable element.

flip-flop, D—A flip-flop with data, clock, and output terminals, designed such that the binary signal at the data terminal is transferred to the output terminal upon application of the proper clock pulse.

flip-flop, J-K—A flip-flop with two synchronous input terminals called J and K, and whose output takes on a 0 or 1, depending on the combination of binary signals applied to the J and K terminals, when the proper clock pulse is applied.

flip-flop, R-S—A flip-flop with two asynchronous input terminals called R and S (reset and set). There is one input combination (both 0's or sometimes both 1's) that has no effect on the output, permitting the flip-flop to "remember" how it responded to the previous input conditions.

flip-flop, R-S-T—An R-S flip-flop with an additional T or trigger input terminal. A pulse on the T input causes the output to change state, permitting the device to be used as a simple counter or divider.

flip-flop, T—A flip-flop with only a trigger input, useful for simple counting or frequency division.

frequency counter—An instrument which measures frequency by counting the number of cycles (pulses) occurring during a certain time interval.

frequency divider—See Divider.

gate—See Electronic gate.

gate, AND—A gate whose output is 1 only if all the inputs are 1's.

gate, NAND—A gate whose output is 0 only if all inputs are 1's.

gate, NOR—A gate whose output is 1 only if all inputs are 0's.

gate, OR—A gate whose output is 0 only when all inputs are 0's.

high state—One of the two binary states, usually the state also called the "1" state, and often represented by the more positive voltage.

hybrid—A combination of integrated circuits and discrete components, or integrated circuits and components made by another process.

impurities—The chemical substances added to pure germanium or silicon to produce semiconductors.

inhibit—See Disable.

instructions—In logic systems, binary codes applied to a logic circuit so as to affect its mode of operation.

integrated circuit—"The physical realization of a number of electrical elements inseparably associated on or within a continuous body of semiconductor material to perform the functions of a circuit." (EIA definition)

interface—The connection between, or circuit connecting between, two logic elements, often elements belonging to two different "families."

inverter—A one-input logic circuit whose binary output is always the opposite from its input.

junction—In diodes and transistors, the area dividing p-type and n-type semiconducting materials.

large-scale integration—Making integrated circuits of great complexity on one chip; the criterion for large-scale integration is arbitrary, but some manufacturers define a digital circuit with more than 25 gates or their equivalents as the minimum complexity, while others set the criterion as high as 100 gates.

latch—An R-S flip-flop.

least significant digit—In a number consisting of more than one digit, that digit having the lowest place value (customarily the right-hand digit).

level translator—A circuit which accepts digital input signals having one pair of voltage levels, and whose output signals are of a different pair of voltage levels.

level-triggered flip-flop—A flip-flop which responds to the voltage level, rather than rate of change, of a signal applied to the proper input terminal.

linear circuit—A circuit whose output voltage is approximately directly proportional to the input voltage. This relationship generally holds only over a limited signal voltage range and often for a limited frequency range.

loading factor—A number representing the relative capacity of a gate to drive other gates, or the relative load that a gate presents to the driving gate.

logic circuit—An electronic circuit whose input-output relationship corresponds to a Boolean algebra logic function.

logic diagram—A pictorial representation of interconnected logic elements using standard symbols.

logic element—A gate or flip-flop or in some cases combinations of these considered as a single entity.

logic function—The Boolean algebra functions of AND, OR, and NOT or combinations of these.

logic swing—The voltage difference between the 1 and 0 voltage levels in a binary logic circuit.

logic system—A group of interconnected logic elements which act together to perform a relatively complex logic function.

logic table—See Truth table.

logical addition—The OR operation.

logical multiplication—The AND operation.

logical 0—A name for one of the two values of a binary digit or signal. If the signal is an electric voltage, this name is usually assigned to the lower of the two voltage levels.

logical 1—The opposite of logical 0.

logically equivalent circuits—Logic circuits which perform the same logic function, even though the circuit details differ.

low—The opposite of the high logic state.

LSI—Abbreviation of large-scale integration.

medium-scale integration—Integrated circuits with a number of gates or their equivalents, but not enough to meet the complexity criterion for large-scale integration.

memory—A circuit or device which maintains its state following the conclusion of the event causing that state.

microcircuit—Another name for integrated circuit.

microsecond—One millionth of a second.

millisecond—One thousandth of a second.

monolithic—From the Greek, literally, "one stone"; made from one silicon chip.

MOS—Metal-oxide semiconductor; a kind of field-effect transistor.

MSI—Abbreviation of medium-scale integration.

multivibrator—A nonsinusoidal oscillator made from a two-stage amplifier, each stage which reverses the signal phase, and with feedback from output to input.

multivibrator, astable—A multivibrator which, once started, will continue to produce oscillations without application of an external signal.

multivibrator, bistable—Another name for a flip-flop.

multivibrator, free-running—Same as an astable multivibrator.

multivibrator, monostable—A multivibrator with only one stable state. When triggered into its unstable state by an external pulse, it remains in that state for a limited time and then returns to the stable state. It is used to generate pulses of a controlled duration. (Also called a "one-shot" or "single-shot.")

NAND **gate**—A gate whose output is a logical 1 unless all inputs are logical 1's.

nanosecond—One thousandth of one millionth of a second.

negation—The NOT operation in Boolean algebra.

negative logic—The assignment of the symbol 0 to the more positive voltage in binary signals; using negative logic has the effect of changing a gate from one function to another. (Example: a NAND gate, using positive logic, becomes a NOR gate when using negative logic.)

Nixie **tube**—A glow tube which displays any one of the digits 0 through 9 depending on which of ten terminals has an applied energizing voltage.

noise—Unwanted variations in signal voltage level or state which interfere with signal processing.

noise immunity—The relative insensitivity of a gate or flip-flop to respond to noise.

noise margin—The largest noise pulse amplitude which will not cause operation of a particular gate or flip-flop.

nondestructive readout—A type of memory operation whereby determining its state ("reading out") does not cause the memory to lose its stored data.

nonsaturated logic—A type of logic circuit in which short delay times are achieved by not allowing transistors to saturate.

NOR **gate**—A gate whose output is a 1 only if all inputs are logical 0's.

NOT **gate**—A single input inverting amplifier whose binary output state is always opposite to its input state.

numerical control—Automatic operation of machinery from binary data, usually stored on punched paper tape.

off-the-shelf—Available for fast delivery from stock, as opposed to requiring custom design or manufacturing to order.

one-shot—See Multivibrator, monostable.

operation—In mathematics, determining a resulting quantity according to some rule when given two or more other quantities; examples: addition and multiplication in conventional arithmetic, and the Boolean AND and OR operations in Boolean algebra.

OR gate—A gate whose output is a logical 1 if any one or more of the inputs have a logical 1 signal applied, corresponding to the Boolean OR function.

package—An enclosure for an integrated circuit, usually either a metal case or a ceramic or plastic molded enclosure.

parallel operation—Connecting logic circuits so that one or more signals are applied simultaneously to several gates or flip-flops, as opposed to serial operation where the signals must pass through each logic element in order. Parallel operation is faster.

passive elements—Resistors, capacitors, inductors: those elements not capable of amplification.

positive logic—The assignment of the symbol 1 to the more positive voltage in binary signals; see also Negative logic.

preset—An asynchronous flip-flop input, sometimes another name for the set input.

propagation time (or **propagation delay**)—See Delay.

pull-up, active—In the output circuit of a gate, the use of a transistor to provide a low-impedance source to charge capacitance in the load when the output changes from a low to high voltage.

pull-up, passive—A gate output circuit in which the current for charging load capacitance is supplied through a resistor.

pulse—A sequence of voltage or current changes characterized by a rapid change from one level to another, a dwell period at the latter level, followed by a rapid return to the first level.

Q output—The output terminal of a flip-flop which goes to a high-voltage level when the flip-flop is "set."

radix—The number of different symbols used in a number system. The radix of our common decimal number system is ten; the radix of the binary number system is two.

read-only memory—A logic subsystem in which different binary signal input combinations (called "addresses") cause defined output combinations (the "stored data"). The relationships between address words and output words are determined by the internal wiring, and cannot be changed electrically (by "reading in" data).

readout—The means of converting electrical signals into visual displays for human interpretation.

register—Two or more flip-flops connected for storing binary words.

reset—The asynchronous input to a flip-flop used to force the Q output to its low state; sometimes called the "clear" input.

resistor-transistor logic—A relatively simple kind of logic circuit using only resistors and transistors, with one transistor for each input and all transistor collectors connected together for the output.

ripple counter—See Counter, ripple-through.

rise time—The time required for a binary signal to change from a low state to a high state; often, more precisely, the time for the signal to rise from 10% to 90% of the way between the two voltage levels.

ROM—See Read-only memory.

R-S flip-flop—See Flip-flop, R-S.

RTL—Abbreviation of Resistor-transistor logic.

saturated—The operating state of a transistor where no further increase in collector current results from increasing the base current; in this state the collector-emitter voltage is low, typically less than 1 volt.

saturated logic—A type of logic circuit in which transistors operate in the saturated state part of the time; such a circuit has inherently longer delays than nonsaturated logic.

saturation voltage—The collector-emitter voltage of a saturated transistor.

Schmitt trigger—A two-stage direct-coupled amplifier with positive feedback due to a common emitter resistor, whose output switches abruptly between low and high voltage levels when the input signal passes through a threshold value.

serial operation—A method of transferring binary data from one part of a system to another, in which individual bits are sent one at a time along one wire.

set input—An asynchronous input to a flip-flop, used to force the Q output to its high sate.

shift register—A set of flip-flops connected in an order such that on every clock pulse the binary output of each flip-flop takes on the value of the one next in order. The effect is to shift the digits in a binary word right or left one place with each clock pulse.

significant digit, least—In a binary number, that digit having the lowest place value.

significant digit, most—In a binary number, that digit having the highest place value.

simulation—The study of the behavior of one system (such as a rocket in flight, a chemical processing plant, etc.) by observing the behavior of another system which is mathematically similar.

single shot—See Multivibrator, monostable.

skewing—An offset in time between two related pulses.

software—Prepared "standard" programs that make a computer easier to use.

state—A condition or set of conditions considered together, especially one of the two normal sets of operating conditions of a gate or flip-flop.

synchronous inputs—Inputs to a flip-flop which affect the output only at the time the clock signal changes from a particular level to the other level.

thick film—A manufacturing technique used for microcircuits where electronic components are formed by depositing layers of materials on an insulating base (substrate) such as ceramic.

threshold-triggered flip-flop—A kind of flip-flop which changes state when the actuating signal passes through a certain voltage level, irrespective of the rate of change of voltage.

toggle—To change abruptly from one state to another in such a way that once the process reaches a critical point, it goes to completion even if the actuating cause is removed; in logic circuits the term is applied to a change of state of a flip-flop or Schmitt trigger.

transfer circuit—A circuit equivalent to a two- or more-pole switch, allowing signals to be switched from one line to another.

transistor-transistor logic—An electronic logic circuit using transistors and resistors in which a number of inputs are obtained by using a multiple-emitter transistor.

translate—To change one binary word to another, or to change from one pair of binary signal levels to another, in order to achieve compatibility between parts of a system.

trigger—To initiate a change of state of a flip-flop.

truth table—A table defining the outputs of a Boolean algebra function or a logic system, for various input combinations.

TTL—Abbreviation of transistor-transistor logic.

turn-off time—The propagation delay of a gate or flip-flop when the output changes from high to low.

turn-on time—The propagation delay of a gate or flip-flop when the output changes from low to high.

two-state device—A mechanical or electronic device which except during transition periods is intended to be operated in either of two states or conditions.

wired OR—Two or more outputs of certain gates connected together so that the voltage at the connection point will be low if any of the individual gate outputs are low; despite the name, it is equivalent to the dot AND and performs the Boolean AND function.

word—An ordered set of two or more binary digits which taken together have a definite meaning in a logic system.

Index

How to Use Integrated-Circuit Logic Elements

SECOND EDITION

by Jack W. Streater

Even with 17 years of engineering experience behind him, the author found himself lost when he first entered the field of integrated-circuit logic electronics. Any references he found either assumed he had wide experience in the computer field and had lived with gates and flip-flops for a long time, or they were devoted to Boolean Algebra and the mathematics of logic circuits. Finally, after much frustration, he mastered the subject and decided to write a book to help others faced with the same problem.

This book is designed to help the engineer or technician who has not previously used or designed digital logic circuits to meet the challenge of digital integrated circuits in electronics. Here, the practical problems and limitations of connecting integrated-circuit logic elements into logic systems in order to accomplish the required result are thoroughly covered.

The first two chapters cover binary, BCD, and decimal number systems, and Boolean algebra with its applications to simple switching circuits. The next two chapters deal with gates and gate combinations, and the following chapter treats bistable elements. Then the logic families (RTL, DTL, TTL, ECL, CTL or CML, MOS, and diode logics) are discussed and compared. The next chapter deals with off-the-shelf logic elements—breadboarding, testing, troubleshooting, and locating sources of data on integrated circuits. The final chapter includes experiments to aid in understanding the operation of logic circuits. A glossary of digital terms has been included as an appendix.

ABOUT THE AUTHOR

Jack Streater received B.S.E.E. and M.B.A. degrees from the Drexel Institute of Technology. He holds two patents and is the author of several technical papers and articles. He has spent many years in the design, research, and development of data recording, logging, and controlling systems. Currently, he is Chief Engineer for Mentor Radio Company, designing aircraft communications and navigation systems, including frequency synthesizers using digital techniques.

HOWARD W. SAMS & CO., INC.
THE BOBBS-MERRILL CO., INC.

21081
$4.50 (In Canada $5.40)