SCHAUM'S ouTlines

DIGITAL PRINCIPLES

Third Edition

ROGER L. TOKHEIM

- Covers all course fundamentals and supplements any class text
- Teaches effective problem-solving
- 716 problems and worked solutions
- Hundreds more answered problems
- The perfect guide for self study

MORE THAN 30 MILLION SCHAUM'S OUTLINES SOLD

Use with these courses: ✓ Elements of Digital Systems

SCHAUM'S OUTLINE OF

THEORY AND PROBLEMS

OF

DIGITAL PRINCIPLES

Third Edition

•

ROGER L. TOKHEIM, M.S.

SCHAUM'S OUTLINE SERIES
McGraw-Hill
New York San Francisco Washington, D.C. Auckland Bogotá
Caracas Lisbon London Madrid Mexico City Milan
Montreal New Delhi San Juan Singapore
Sydney Tokyo Toronto

ROGER L. TOKHEIM holds B.S., M.S., and Ed.S. degrees from St. Cloud State University and the University of Wisconsin-Stout. He is the author of *Digital Electronics* and its companion *Activities Manual for Digital Electronics*, *Schaum's Outline of Microprocessor Fundamentals*, and numerous other instructional materials on science and technology. An experienced educator at the secondary and college levels, he is presently an instructor of Technology Education and Computer Science at Henry Sibley High School, Mendota Heights, Minnesota.

Schaum's Outline of Theory and Problems of
DIGITAL PRINCIPLES

7 8 9 10 11 12 13 14 15 16 17 18 19 20 BAW BAW 99

ISBN 0-07-065050-0

Sponsoring Editor: John Aliano
Production Supervisor: Denise Puryear
Editing Supervisor: Patty Andrews

Library of Congress Cataloging-in-Publication Data
Tokheim, Roger L.
Schaum's outline of theory and problems of digital prinicples/by Roger L. Tokheim—3rd ed.
p. cm.—(Schaum's outline series)
Includes index.
ISBN 0-07-065050-0
1. Digital electronics. I. Title. II. Series.
TK7868.D5T66 1994 93-64
621.3815—dc20 CIP

McGraw-Hill
A Division of The McGraw-Hill Companies

Preface

Digital electronics is a rapidly growing technology. Digital circuits are used in most new consumer products, industrial equipment and controls, and office, medical, military, and communications equipment. This expanding use of digital circuits is the result of the development of inexpensive integrated circuits and the application of display, memory, and computer technology.

Schaum's Outline of Digital Principles provides information necessary to lead the reader through the solution of those problems in digital electronics one might encounter as a student, technician, engineer, or hobbyist. While the principles of the subject are necessary, the Schaum's Outline philosophy is dedicated to showing the student how to apply the principles of digital electronics through practical solved problems. This new edition now contains over 1000 solved and supplementary problems.

The third edition of *Schaum's Outline of Digital Principles* contains many of the same topics which made the first two editions great successes. Slight changes have been made in many of the traditional topics to reflect the technological trend toward using more CMOS, NMOS, and PMOS integrated circuits. Several microprocessor/microcomputer-related topics have been included, reflecting the current practice of teaching a microprocessor course after or with digital electronics. A chapter detailing the characteristics of TTL and CMOS devices along with several interfacing topics has been added. Other display technologies such as liquid-crystal displays (LCDs) and vacuum fluorescent (VF) displays have been given expanded coverage. The chapter on microcomputer memory has been revised with added coverage of hard and optical disks. Sections on programmable logic arrays (PLA), magnitude comparators, demultiplexers, and Schmitt trigger devices have been added.

The topics outlined in this book were carefully selected to coincide with courses taught at the upper high school, vocational-technical school, technical college, and beginning college level. Several of the most widely used textbooks in digital electronics were analyzed. The topics and problems included in this Schaum's Outline reflect those encountered in standard textbooks.

Schaum's Outline of Digital Principles, Third Edition, begins with number systems and digital codes and continues with logic gates and combinational logic circuits. It then details the characteristics of both TTL and CMOS ICs, along with various interfacing topics. Next encoders, decoders, and display drivers are explored, along with LED, LCD, and VF seven-segment displays. Various arithmetic circuits are examined. It then covers flip-flops, other multivibrators, and sequential logic, followed by counters and shift registers. Next semiconductor and bulk storage memories are explored. Finally, multiplexers, demultiplexers, latches and buffers, digital data transmission, magnitude comparators, Schmitt trigger devices, and programmable logic arrays are investigated. The book stresses the use of industry-standard digital ICs (both TTL and CMOS) so that the reader becomes familiar with the practical hardware aspects of digital electronics. Most circuits in this Schaum's Outline can be wired using standard digital ICs.

I wish to thank my son Marshall for his many hours of typing, proofreading, and testing circuits to make this book as accurate as possible. Finally, I extend my appreciation to other family members Daniel and Carrie for their help and patience.

ROGER L. TOKHEIM

Contents

Chapter 1

Numbers Used in Digital Electronics

1-1 INTRODUCTION

The decimal number system is familiar to everyone. This system uses the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. The decimal system also has a place-value characteristic. Consider the decimal number 238. The 8 is in the 1s position or place. The 3 is in the 10s position, and therefore the three 10s stand for 30 units. The 2 is in the 100s position and means two 100s, or 200 units. Adding 200 + 30 + 8 gives the total decimal number of 238. The decimal number system is also called the *base 10 system*. It is referred to as base 10 because it has 10 different symbols. The base 10 system is also said to have a *radix* of 10. "Radix" and "base" are terms that mean exactly the same thing.

Binary numbers (base 2) are used extensively in digital electronics and computers. Both hexadecimal (base 16) and octal (base 8) numbers are used to represent groups of binary digits. Binary and hexadecimal numbers find wide use in modern microcomputers.

All the number systems mentioned (decimal, binary, octal, and hexadecimal) can be used for counting. All these number systems also have the place-value characteristic.

1-2 BINARY NUMBERS

The binary number system uses only two symbols (0, 1). It is said to have a radix of 2 and is commonly called the *base 2 number system*. Each *bi*nary digi*t* is called a *bit*.

Counting in binary is illustrated in Fig. 1-1. The binary number is shown on the right with its decimal equivalent. Notice that the *least significant bit* (LSB) is the 1s place. In other words, if a 1 appears in the right column, a 1 is added to the binary count. The second place over from the right is the 2s place. A 1 appearing in this column (as in decimal 2 row) means that 2 is added to the count. Three other binary place values also are shown in Fig. 1-1 (4s, 8s, and 16s places). Note that each larger place value is an added power of 2. The 1s place is really 2^0, the 2s place 2^1, the 4s place 2^2, the 8s place 2^3, and the 16s place 2^4. It is customary in digital electronics to memorize at least the binary counting sequence from 0000 to 1111 (say: one, one, one, one) or decimal 15.

Consider the number shown in Fig. 1-2*a*. This figure shows how to convert the binary 10011 (say: one, zero, zero, one, one) to its decimal equivalent. Note that, for each 1 bit in the binary number, the decimal equivalent for that place value is written below. The decimal numbers are then added (16 + 2 + 1 = 19) to yield the decimal equivalent. Binary 10011 then equals a decimal 19.

Consider the binary number 101110 in Fig. 1-2*b*. Using the same procedure, each 1 bit in the binary number generates a decimal equivalent for that place value. The *most significant bit* (MSB) of the binary number is equal to 32. Add 8 plus 4 plus 2 to the 32 for a total of 46. Binary 101110 then equals decimal 46. Figure 1-2*b* also identifies the binary point (similar to the decimal point in decimal numbers). It is customary to omit the binary point when working with whole binary numbers.

What is the value of the number 111? It could be one hundred and eleven in decimal or one, one, one in binary. Some books use the system shown in Fig. 1-2*c* to designate the base, or radix, of a number. In this case 10011_2 is a base 2 number as shown by the small subscript 2 after the number. The number 19_{10} is a base 10 number as shown by the subscript 10 after the number. Figure 1-2*c* is a summary of the binary-to-decimal conversions in Fig. 1-2*a* and *b*.

How about converting fractional numbers? Figure 1-3 illustrates the binary number 1110.101 being converted to its decimal equivalent. The place values are given across the top. Note the value of each position to the right of the binary point. The procedure for making the conversion is the same as with whole numbers. The place value of each 1 bit in the binary number is added to form the decimal number. In this problem 8 + 4 + 2 + 0.5 + 0.125 = 14.625 in decimal.

Decimal count	Binary count				
	16s	8s	4s	2s	1s
0					0
1					1
2				1	0
3				1	1
4			1	0	0
5			1	0	1
6			1	1	0
7			1	1	1
8		1	0	0	0
9		1	0	0	1
10		1	0	1	0
11		1	0	1	1
12		1	1	0	0
13		1	1	0	1
14		1	1	1	0
15		1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1
	2^4	2^3	2^2	2^1	2^0
	Powers of 2				

Fig. 1-1 Counting in binary and decimal

Powers of 2		2^4	2^3	2^2	2^1	2^0	
Place value		16s	8s	4s	2s	1s	
Binary		**1**	**0**	**0**	**1**	**1**	. ←—Binary point
Decimal		16	+		2	+ 1	= 19

(*a*) Binary-to-decimal conversion

Powers of 2	2^5	2^4	2^3	2^2	2^1	2^0	
Place value	32s	16s	8s	4s	2s	1s	
Binary	**1**	**0**	**1**	**1**	**1**	**0**	. ←—Binary point
Decimal	32	+	8	+ 4	+ 2		= 46

(*b*) Binary-to-decimal conversion

$10011_2 = 19_{10}$ $101110_2 = 46_{10}$

(*c*) Summary of conversions and use of small subscripts to indicate radix of number

Fig. 1-2

Powers of 2	2^3	2^2	2^1	2^0	$1/2^1$	$1/2^2$	$1/2^3$
Place value	8s	4s	2s	1s	0.5s	0.25s	0.125s
Binary	1	1	1	0 .	1	0	1
Decimal	8 +	4 +	2 +		0.5 +		0.125 = 14.625

Fig. 1-3 Binary-to-decimal conversion

Convert the decimal number 87 to a binary number. Figure 1-4 shows a convenient method for making this conversion. The decimal number 87 is first divided by 2, leaving 43 with a remainder of 1. The remainder is important and is recorded at the right. It becomes the LSB in the binary number. The quotient (43) then is transferred as shown by the arrow and becomes the dividend. The quotients are repeatedly divided by 2 until the quotient becomes 0 with a remainder of 1, as in the last line of Fig. 1-4. Near the bottom the figure shows that decimal 87 equals binary 1010111.

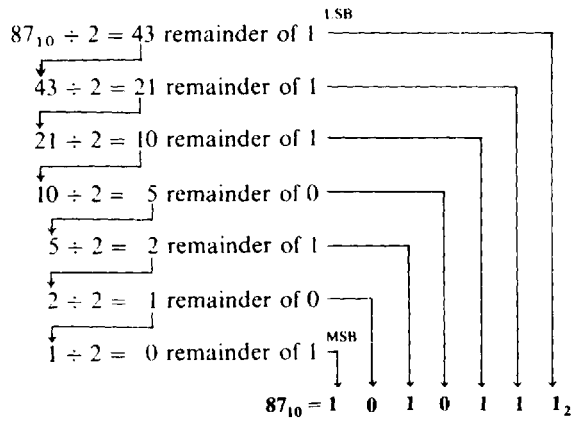


Fig. 1-4 Decimal-to-binary conversion

Convert the decimal number 0.375 to a binary number. Figure 1-5*a* illustrates one method of performing this task. Note that the decimal number (0.375) is being *multiplied* by 2. This leaves a product of 0.75. The 0 from the integer place (1s place) becomes the bit nearest the binary point. The 0.75 is then multiplied by 2, yielding 1.50. The carry of 1 to the integer (1s place) is the next bit in the binary number. The 0.50 is then multiplied by 2, yielding a product of 1.00. The carry of 1 in the integer place is the final 1 in the binary number. When the product is 1.00, the conversion process is complete. Figure 1-5*a* shows a decimal 0.375 being converted into a binary equivalent of 0.011.

Figure 1-5*b* shows the decimal number 0.84375 being converted into binary. Again note that 0.84375 is multiplied by 2. The integer of each product is placed below, forming the binary number. When the product reaches 1.00, the conversion is complete. This problem shows a decimal 0.84375 being converted to binary 0.11011.

Consider the decimal number 5.625. Converting this number to binary involves two processes. The integer part of the number (5) is processed by *repeated division* near the top in Fig. 1-6. Decimal 5 is converted to a binary 101. The fractional part of the decimal number (.625) is converted to binary .101 at the bottom in Fig. 1-6. The fractional part is converted to binary through the *repeated multiplication* process. The integer and fractional sections are then combined to show that decimal 5.625 equals binary 101.101.

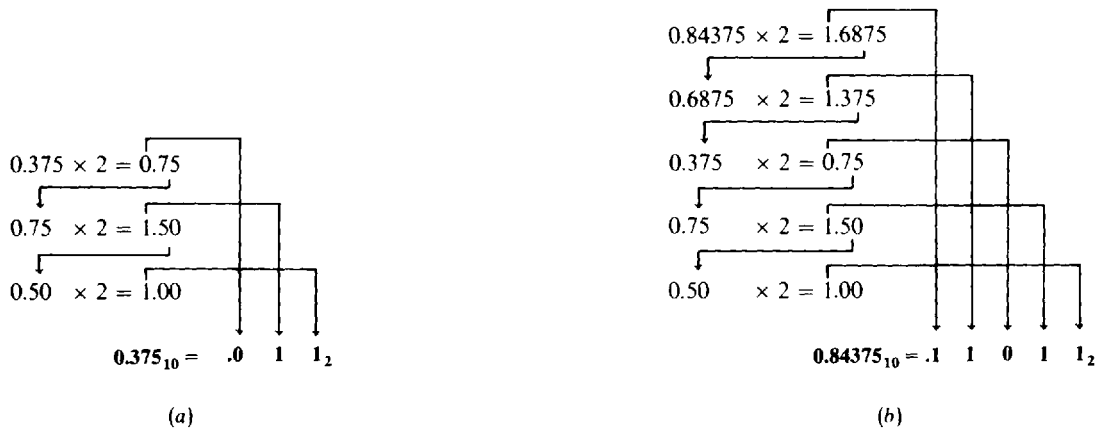


Fig. 1-5 Fractional decimal-to-binary conversions

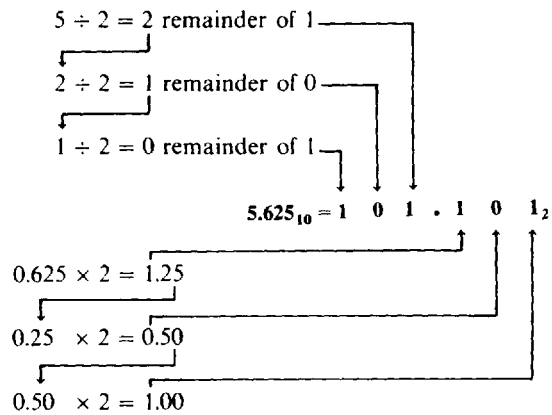


Fig. 1-6 Decimal-to-binary conversion

SOLVED PROBLEMS

1.1 The binary number system is the base _____ system and has a radix of _____.

Solution:

The binary number system is the base 2 system and has a radix of 2.

1.2 The term bit means _____ _____ when dealing with binary numbers.

Solution:

Bit means binary digit.

1.3 How would you say the number 1001 in (*a*) binary and (*b*) decimal?

Solution:

The number 1001 is pronounced as follows: (*a*) one, zero, zero, one; (*b*) one thousand and one.

1.4 The number 110_{10} is a base _____ number.

Solution:

The number 110_{10} is a base 10 number, as indicated by the small 10 after the number.

1.5 Write the base 2 number one, one, zero, zero, one.

Solution:

11001_2

1.6 Convert the following binary numbers to their decimal equivalents:
(*a*) 001100 (*b*) 000011 (*c*) 011100 (*d*) 111100 (*e*) 101010 (*f*) 111111
(*g*) 100001 (*h*) 111000

Solution:

Follow the procedure shown in Fig. 1-2. The decimal equivalents of the binary numbers are as follows:

(*a*) $001100_2 = 12_{10}$ (*c*) $011100_2 = 28_{10}$ (*e*) $101010_2 = 42_{10}$ (*g*) $100001_2 = 33_{10}$
(*b*) $000011_2 = 3_{10}$ (*d*) $111100_2 = 60_{10}$ (*f*) $111111_2 = 63_{10}$ (*h*) $111000_2 = 56_{10}$

1.7 $11110001111_2 =$ _____$_{10}$

Solution:

Follow the procedure shown in Fig. 1-2. $11110001111_2 = 1935_{10}$.

1.8 $11100.011_2 =$ _____$_{10}$

Solution:

Follow the procedure shown in Fig. 1-3. $11100.011_2 = 28.375_{10}$.

1.9 $110011.100\,11_2 =$ _____$_{10}$

Solution:

Follow the procedure shown in Fig. 1-3. $110011.10011_2 = 51.593\,75_{10}$.

1.10 $1010101010.1_2 =$ _____$_{10}$

Solution:

Follow the procedure shown in Fig. 1-3. $1010101010.1_2 = 682.5_{10}$.

1.11 Convert the following decimal numbers to their binary equivalents:
(*a*) 64, (*b*) 100, (*c*) 111, (*d*) 145, (*e*) 255, (*f*) 500.

Solution:

Follow the procedure shown in Fig. 1-4. The binary equivalents of the decimal numbers are as follows:

(*a*) $64_{10} = 1000000_2$ (*c*) $111_{10} = 1101111_2$ (*e*) $255_{10} = 11111111_2$
(*b*) $100_{10} = 1100100_2$ (*d*) $145_{10} = 10010001_2$ (*f*) $500_{10} = 111110100_2$

1.12 $34.75_{10} =$ _____$_2$

Solution:

Follow the procedure shown in Fig. 1-6. $34.75_{10} = 100010.11_2$.

1.13 $25.25_{10} =$ _____$_2$

Solution:

Follow the procedure shown in Fig. 1-6. $25.25_{10} = 11001.01_2$.

1.14 $27.1875_{10} =$ _____$_2$

Solution:

Follow the procedure shown in Fig. 1-6. $27.1875_{10} = 11011.0011_2$.

1-3 HEXADECIMAL NUMBERS

The hexadecimal number system has a radix of 16. It is referred to as the *base 16 number system*. It uses the symbols 0-9, A, B, C, D, E, and F as shown in the hexadecimal column of the table in Fig. 1-7. The letter A stands for a count of 10, B for 11, C for 12, D for 13, E for 14, and F for 15. The advantage of the hexadecimal system is its usefulness in converting directly from a 4-bit binary number. Note in the shaded section of Fig. 1-7 that each 4-bit binary number from 0000 to 1111 can be represented by a unique hexadecimal digit.

Decimal	Binary	Hexadecimal	Decimal	Binary	Hexadecimal
0	0000	0	16	10000	10
1	0001	1	17	10001	11
2	0010	2	18	10010	12
3	0011	3	19	10011	13
4	0100	4	20	10100	14
5	0101	5	21	10101	15
6	0110	6	22	10110	16
7	0111	7	23	10111	17
8	1000	8	24	11000	18
9	1001	9	25	11001	19
10	1010	A	26	11010	1A
11	1011	B	27	11011	1B
12	1100	C	28	11100	1C
13	1101	D	29	11101	1D
14	1110	E	30	11110	1E
15	1111	F	31	11111	1F

Fig. 1-7 Counting in decimal, binary, and hexadecimal number systems

Look at the line labeled 16 in the decimal column in Fig. 1-7. The hexadecimal equivalent is 10. This shows that the hexadecimal number system uses the place-value idea. The 1 (in 10_{16}) stands for 16 units, while the 0 stands for zero units.

Convert the hexadecimal number 2B6 into a decimal number. Figure 1-8*a* shows the familiar process. The 2 is in the 256s place so $2 \times 256 = 512$, which is written in the decimal line. The hexadecimal digit B appears in the 16s column. Note in Fig. 1-8 that hexadecimal B corresponds to decimal 11. This means that there are eleven 16s (16×11), yielding 176. The 176 is added into the decimal total near the bottom in Fig. 1-8*a*. The 1s column shows six 1s. The 6 is added into the decimal line. The decimal values are added ($512 + 176 + 6 = 694$), yielding 694_{10}. Figure 1-8*a* shows that $2B6_{16}$ equals 694_{10}.

Powers of 16	16^2	16^1	16^0
Place value	256s	16s	1s

Hexadecimal number	**2**		**B**		**6**		
	256		16		1		
	× 2		× 11		× 6		
Decimal	512	+	176	+	6	=	694_{10}

(*a*) Hexadecimal-to-decimal conversion

Powers of 16	16^2	16^1	16^0	$1\ 16^1$
Place value	256s	16s	1s	.0625s

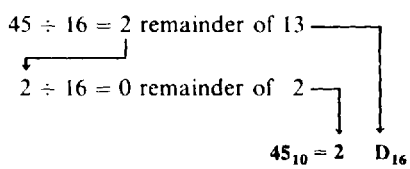
Hexadecimal number	**A**		**3**		**F**	·	**C**		
	256		16		1		.0625		
	× 10		× 3		× 15		× 12		
Decimal	2560	+	48	+	15	+	0.75	=	2623.75_{10}

(*b*) Fractional hexadecimal-to-decimal conversion

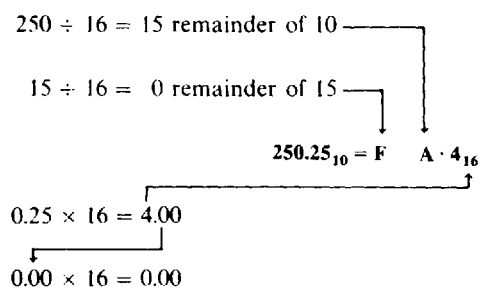
Fig. 1-8

Convert the hexadecimal number A3F.C to its decimal equivalent. Figure 1-8*b* details this problem. First consider the 256s column. The hexadecimal digit A means that 256 must be multiplied by 10, resulting in a product of 2560. The hexadecimal number shows that it contains three 16s, and therefore $16 \times 3 = 48$, which is added to the decimal line. The 1s column contains the hexadecimal digit F, which means $1 \times 15 = 15$. The 15 is added to the decimal line. The 0.0625s column contains the hexadecimal digit C, which means $12 \times 0.0625 = 0.75$. The 0.75 is added to the decimal line. Adding the contents of the decimal line ($2560 + 48 + 15 + 0.75 = 2623.75$) gives the decimal number 2623.75. Figure 1-8*b* converts $A3F.C_{16}$ to 2623.75_{10}.

Now reverse the process and convert the decimal number 45 to its hexadecimal equivalent. Figure 1-9*a* details the familiar repeated divide-by-16 process. The decimal number 45 is first divided by 16, resulting in a quotient of 2 with a remainder of 13. The remainder of 13 (D in hexadecimal) becomes the LSD of the hexadecimal number. The quotient (2) is transferred to the dividend position and divided by 16. This results in a quotient of 0 with a remainder of 2. The 2 becomes the next digit in the



(*a*) Decimal-to-hexadecimal conversion



(*b*) Fractional decimal-to-hexadecimal conversion

Fig. 1-9

hexadecimal number. The process is complete because the integer part of the quotient is 0. The process in Fig. 1-9*a* converts the decimal number 45 to the hexadecimal number 2D.

Convert the decimal number 250.25 to a hexadecimal number. The conversion must be done by using two processes as shown in Fig. 1-9*b*. The integer part of the decimal number (250) is converted to hexadecimal by using the repeated divide-by-16 process. The remainders of 10 (A in hexadecimal) and 15 (F in hexadecimal) form the hexadecimal whole number FA. The fractional part of the 250.25 is multiplied by 16 (0.25×16). The result is 4.00. The integer 4 is transferred to the position shown in Fig. 1-9*b*. The completed conversion shows the decimal number 250.25 equaling the hexadecimal number FA.4.

The prime advantage of the hexadecimal system is its easy conversion to binary. Figure 1-10*a* shows the hexadecimal number 3B9 being converted to binary. Note that each hexadecimal digit forms a group of four binary digits, or bits. The groups of bits are then combined to form the binary number. In this case $3B9_{16}$ equals 1110111001_2.

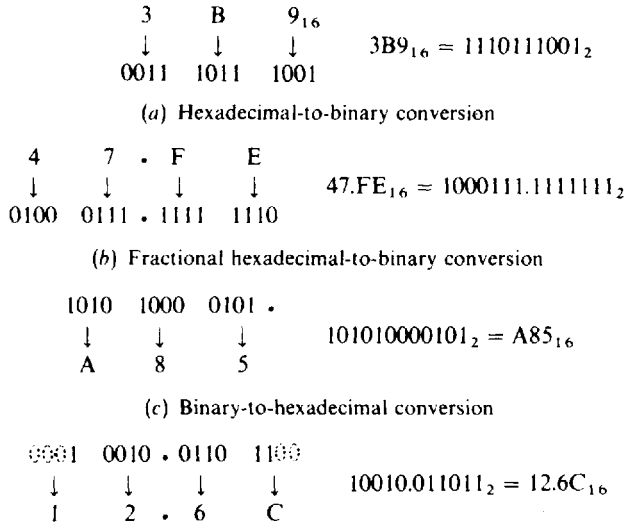


Fig. 1-10

Another hexadecimal-to-binary conversion is detailed in Fig. 1-10*b*. Again each hexadecimal digit forms a 4-bit group in the binary number. The hexadecimal point is dropped straight down to form the binary point. The hexadecimal number 47.FE is converted to the binary number 1000111.1111111. It is apparent that hexadecimal numbers, because of their compactness, are much easier to write down than the long strings of 1s and 0s in binary. The hexadecimal system can be thought of as a shorthand method of writing binary numbers.

Figure 1-10*c* shows the binary number 101010000101 being converted to hexadecimal. First divide the binary number into 4-bit groups *starting at the binary point*. Each group of four bits is then translated into an equivalent hexadecimal digit. Figure 1-10*c* shows that binary 101010000101 equals hexadecimal A85.

Another binary-to-hexadecimal conversion is illustrated in Fig. 1-10*d*. Here binary 10010.011011 is to be translated into hexadecimal. First the binary number is divided into groups of four bits, starting at the binary point. Three 0s are added in the leftmost group, forming 0001. Two 0s are added to the rightmost group, forming 1100. Each group now has 4 bits and is translated into a hexadecimal digit as shown in Fig. 1-10*d*. The binary number 10010.011011 then equals $12.6C_{16}$.

As a practical matter, many modern hand-held calculators perform number base conversions. Most can convert between decimal, hexadecimal, octal, and binary. These calculators can also perform arithmetic operations in various bases (such as hexadecimal).

SOLVED PROBLEMS

1.15 The hexadecimal number system is sometimes called the base _____ system.

Solution:

The hexadecimal number system is sometimes called the base 16 system.

1.16 List the 16 symbols used in the hexadecimal number system.

Solution:

Refer to Fig. 1-7. The 16 symbols used in the hexadecimal number system are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

1.17 Convert the following whole hexadecimal numbers to their decimal equivalents:
(*a*) C, (*b*) 9F, (*c*) D52, (*d*) 67E, (*e*) ABCD.

Solution:

Follow the procedure shown in Fig. 1-8*a*. Refer also to Fig. 1-7. The decimal equivalents of the hexadecimal numbers are as follows:

(*a*) $C_{16} = 12_{10}$ (*c*) $D52_{16} = 3410_{10}$ (*e*) $ABCD_{16} = 43981_{10}$
(*b*) $9F_{16} = 159_{10}$ (*d*) $67E_{16} = 1662_{10}$

1.18 Convert the following hexadecimal numbers to their decimal equivalents:
(*a*) F.4, (*b*) D3.E, (*c*) 1111.1, (*d*) 888.8, (*e*) EBA.C.

Solution:

Follow the procedure shown in Fig. 1-8*b*. Refer also to Fig. 1-7. The decimal equivalents of the hexadecimal numbers are as follows:

(*a*) $F.4_{16} = 15.25_{10}$ (*c*) $1111.1_{16} = 4369.0625_{10}$ (*e*) $EBA.C_{16} = 3770.75_{10}$
(*b*) $D3.E_{16} = 211.875_{10}$ (*d*) $888.8_{16} = 2184.5_{10}$

1.19 Convert the following whole decimal numbers to their hexadecimal equivalents:
(*a*) 8, (*b*) 10, (*c*) 14, (*d*) 16, (*e*) 80, (*f*) 2560, (*g*) 3000, (*h*) 62,500.

Solution:

Follow the procedure shown in Fig. 1-9*a*. Refer also to Fig. 1-7. The hexadecimal equivalents of the decimal numbers are as follows:

(*a*) $8_{10} = 8_{16}$ (*c*) $14_{10} = E_{16}$ (*e*) $80_{10} = 50_{16}$ (*g*) $3000_{10} = BB8_{16}$
(*b*) $10_{10} = A_{16}$ (*d*) $16_{10} = 10_{16}$ (*f*) $2560_{10} = A00_{16}$ (*h*) $62\,500_{10} = F424_{16}$

1.20 Convert the following decimal numbers to their hexadecimal equivalents:
(*a*) 204.125, (*b*) 255.875, (*c*) 631.25, (*d*) 10 000.003 906 25.

Solution:

Follow the procedure shown in Fig. 1-9*b*. Refer also to Fig. 1-7. The hexadecimal equivalents of the decimal numbers are as follows:

(*a*) $204.125_{10} = CC.2_{16}$ (*c*) $631.25_{10} = 277.4_{16}$
(*b*) $255.875_{10} = FF.E_{16}$ (*d*) $10\,000.003\,906\,25_{10} = 2710.01_{16}$

1.21 Convert the following hexadecimal numbers to their binary equivalents:
(*a*) B, (*b*) E, (*c*) 1C, (*d*) A64, (*e*) 1F.C, (*f*) 239.4

Solution:

Follow the procedure shown in Fig. 1-10*a* and *b*. Refer also to Fig. 1-7. The binary equivalents of the hexadecimal numbers are as follows:

(*a*) $B_{16} = 1011_2$ (*c*) $1C_{16} = 11100_2$ (*e*) $1F.C_{16} = 11111.11_2$

(*b*) $E_{16} = 1110_2$ (*d*) $A64_{16} = 101001100100_2$ (*f*) $239.4_{16} = 1000111001.01_2$

1.22 Convert the following binary numbers to their hexadecimal equivalents:

(*a*) 1001.1111 (*c*) 110101.011001 (*e*) 10100111.111011

(*b*) 10000001.1101 (*d*) 10000.1 (*f*) 1000000.0000111

Solution:

Follow the procedure shown in Fig. 1-10*c* and *d*. Refer also to Fig. 1-7. The hexadecimal equivalents of the binary numbers are as follows:

(*a*) $1001.1111_2 = 9.F_{16}$ (*c*) $110101.011001_2 = 35.64_{16}$ (*e*) $10100111.111011_2 = A7.EC_{16}$

(*b*) $10000001.1101_2 = 81.D_{16}$ (*d*) $10000.1_2 = 10.8_{16}$ (*f*) $1000000.0000111_2 = 40.0E_{16}$

1-4 2s COMPLEMENT NUMBERS

The 2s complement method of representing numbers is widely used in microprocessor-based equipment. Until now, we have assumed that all numbers are positive. However, microprocessors must process both positive and negative numbers. By using *2s complement representation*, the *sign as well as the magnitude* of a number can be determined.

Assume a microprocessor register 8 bits wide such as that shown in Fig. 1-11*a*. The most-significant bit (MSB) is the *sign bit*. If this bit is 0, then the number is (+) positive. However, if the sign bit is 1, then the number is (−) negative. The other 7 bits in this 8-bit register represent the magnitude of the number.

The table in Fig. 1-11*b* shows the 2s complement representations for some positive and negative numbers. For instance, a + 127 is represented by the 2s complement number 01111111. A decimal −128 is represented by the 2s complement number 10000000. Note that *the 2s complement representations for all positive values are the same as the binary equivalents* for that decimal number.

Convert the signed decimal −1 to a 2s complement number. Follow Fig. 1-12 as you make the conversion in the next five steps.

Step 1. Separate the sign and magnitude part of −1. The negative sign means the sign bit will be 1 in the 2s complement representation.

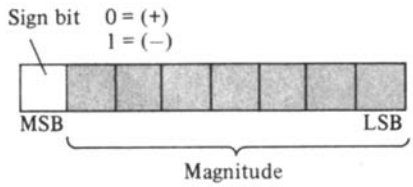
Step 2. Convert decimal 1 to its 7-bit binary equivalent. In this example decimal 1 equals 0000001 in binary.

Step 3. Convert binary 0000001 to its 1s complement form. In this example binary 0000001 equals 1111110 in 1s complement. Note that each 0 is changed to a 1 and each 1 to a 0.

Step 4. Convert the 1s complement to its 2s complement form. In this example 1s complement 1111110 equals 1111111 in 2s complement. Add +1 to the 1s complement to get the 2s complement number.

Step 5. The 7-bit 2s complement number (1111111 in this example) becomes the magnitude part of the entire 8-bit 2s complement number.

The result is that the signed decimal −1 equals 11111111 in 2s complement notation. The 2s complement number is shown in the register near the top of Fig. 1-12.



(*a*) The MSB of an 8-bit register is the sign bit

Signed decimal	8-bit 2s complement representation		
+127	0	111	1111
+126	0	111	1110
+125	0	111	1101
+124	0	111	1100
⋮	⋮	⋮	⋮
+5	0	000	0101
+4	0	000	0100
+3	0	000	0011
+2	0	000	0010
+1	0	000	0001
+0	0	000	0000
−1	1	111	1111
−2	1	111	1110
−3	1	111	1101
−4	1	111	1100
−5	1	111	1011
⋮	⋮	⋮	⋮
−125	1	000	0011
−126	1	000	0010
−127	1	000	0001
−128	1	000	0000
	Sign	Magnitude	

same as binary numbers

(*b*) 2s complement representations of positive and negative numbers

Fig. 1-11

Reverse the process and convert the 2s complement 11111000 to a signed decimal number. Follow Fig. 1-13 as the conversion is made in the following four steps.

Step 1. Separate the sign bit from the magnitude part of the 2s complement number. The MSB is a 1; therefore, the sign of decimal number will be (−) negative.

Step 2. Take the 1s complement of the magnitude part. The 7-bit magnitude 1111000 equals 0000111 in 1s complement notation.

Step 3. Add +1 to the 1s complement number. Adding 0000111 to 1 gives us 0001000. The 7-bit number 0001000 is now in binary.

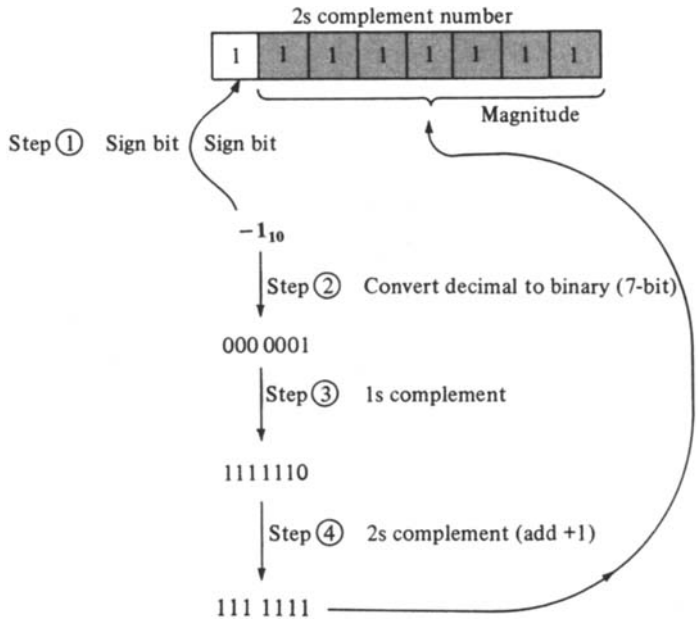


Fig. 1-12 Converting a signed decimal number to a 2s complement number

Step 4. Convert the binary number to its decimal equivalent. In this example, binary 0001000 equals 8 in decimal notation. The magnitude part of the number is 8.

The procedure in Fig. 1-13 shows how to convert 2s complement notation to negative signed decimal numbers. In this example, 2s complement 11111000 equals -8 in decimal notation.

Regular binary-to-decimal conversion (see Fig. 1-4) is used to convert 2s complements that equal positive decimal numbers. Remember that, for positive decimal numbers, the binary and 2s complement equivalents are the same.

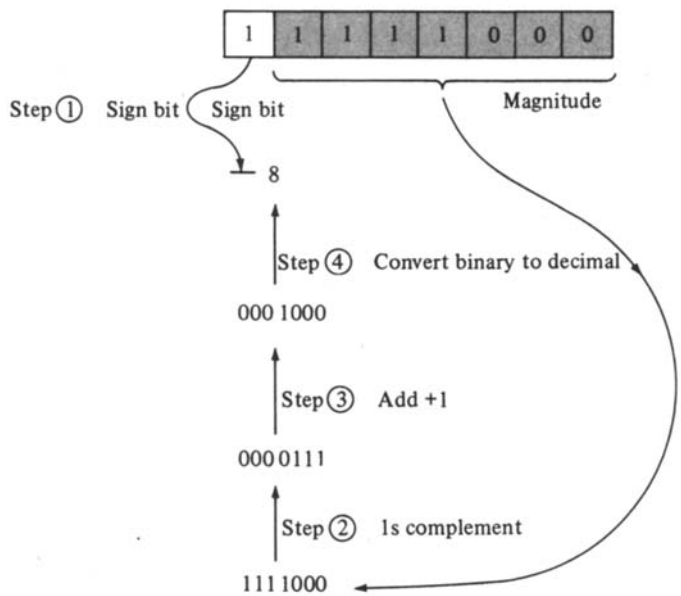


Fig. 1-13 Converting a 2s complement number to a signed decimal number

SOLVED PROBLEMS

1.23 The _____ (LSB, MSB) of a 2s complement number is the sign bit.

The MSB (most-significant bit) of a 2s complement number is the sign bit.

1.24 The 2s complement number 10000000 is equal to _____ in signed decimal.

Solution:

Follow the procedure shown in Fig. 1-13. The 2s complement number 10000000 equals −128 in decimal.

1.25 The number 01110000 is equal to _____ in signed decimal.

Solution:

The 0 in the MSB position means this is a positive number, and conversion to decimal follows the rules used in binary-to-decimal conversion. The number 01110000 is equal to +112 in signed decimal.

1.26 The signed decimal number +75 equals _____ in 8-bit 2s complement.

Solution:

Follow the procedure shown in Fig. 1-4. Decimal +75 equals 01001011 in 2s complement and binary.

1.27 The 2s complement number 11110001 is equal to _____ in signed decimal.

Solution:

Follow the procedure shown in Fig. 1-13. The 2s complement number 11110001 is equal to −15 in signed decimal.

1.28 The signed decimal number −35 equals _____ in 8-bit 2s complement.

Solution:

Follow the procedure shown in Fig. 1-12. Decimal −35 equals 11011101 in 2s complement.

1.29 The signed decimal number −100 equals _____ in 8-bit 2s complement.

Solution:

Follow the procedure shown in Fig. 1-12. Decimal −100 equals 10011100 in 2s complement.

1.40 The signed decimal number +20 equals _____ in 8-bit 2s complement.

Solution:

Follow the procedure shown in Fig. 1-4. Decimal +20 equals 00010100 in 2s complement and binary

Supplementary Problems

1.31 The number system with a radix of 2 is called the _____ number system. *Ans.* binary

1.32 The number system with a radix of 10 is called the _____ number system. *Ans.* decimal

1.33 The number system with a radix of 8 is called the _____ number system. *Ans.* octal

1.34 The number system with a radix of 16 is called the _____ number system. *Ans.* hexadecimal

1.35 A *binary digit* is sometimes shortened and called a(n) _____. *Ans.* bit

1.36 How would you pronounce the number 1101 in (*a*) binary and (*b*) decimal?
Ans. (*a*) one, one, zero, one (*b*) one thousand one hundred and one

1.37 The number 1010_2 is a base __(*a*)__ number and is pronounced __(*b*)__.
Ans. (*a*) 2 (*b*) one, zero, one, zero

1.38 Convert the following binary numbers to their decimal equivalents:
(*a*) 00001110, (*b*) 11100000, (*c*) 10000011, (*d*) 10011010.
Ans. (*a*) $00001110_2 = 14_{10}$ (*c*) $10000011_2 = 131_{10}$
(*b*) $11100000_2 = 224_{10}$ (*d*) $10011010_2 = 154_{10}$

1.39 $110011.11_2 =$ _____$_{10}$ *Ans.* 51.75

1.40 $11110000.0011_2 =$ _____$_{10}$ *Ans.* 240.1875

1.41 Convert the following decimal numbers to their binary equivalents:
(*a*) 32, (*b*) 200, (*c*) 170, (*d*) 258.
Ans. (*a*) $32_{10} = 100000_2$ (*c*) $170_{10} = 10101010_2$
(*b*) $200_{10} = 11001000_2$ (*d*) $258_{10} = 100000010_2$

1.42 $40.875_{10} =$ _____$_2$ *Ans.* 101000.111

1.43 $999.125_{10} =$ _____$_2$ *Ans.* 1111100111.001

1.44 Convert the following hexadecimal numbers to their decimal equivalents:
(*a*) 13AF, (*b*) 25E6, (*c*) B4.C9, (*d*) 78.D3.
Ans. (*a*) $13AF_{16} = 5039_{10}$ (*c*) $B4.C9_{16} = 180.78515_{10}$
(*b*) $25E6_{16} = 9702_{10}$ (*d*) $78.D3_{16} = 120.82421_{10}$

1.45 Convert the following decimal numbers to their hexadecimal equivalents:
(*a*) 3016, (*b*) 64881, (*c*) 17386.75, (*d*) 9817.625.
Ans. (*a*) $3016_{10} = BC8_{16}$ (*c*) $17386.75_{10} = 43EA.C_{16}$
(*b*) $64881_{10} = FD71_{16}$ (*d*) $9817.625_{10} = 2659.A_{16}$

1.46 Convert the following hexadecimal numbers to their binary equivalents:
(*a*) A6, (*b*) 19, (*c*) E5.04, (*d*) 1B.78.
Ans. (*a*) $A6_{16} = 10100110_2$ (*c*) $E5.04_{16} = 11100101.000001_2$
(*b*) $19_{16} = 11001_2$ (*d*) $1B.78_{16} = 11011.01111_2$

1.47 Convert the following binary numbers to their hexadecimal equivalents:
(*a*) 11110010, (*b*) 11011001, (*c*) 111110.000011, (*d*) 10001.11111.
Ans. (*a*) $11110010_2 = F2_{16}$ (*c*) $111110.000011_2 = 3E.0C_{16}$
(*b*) $11011001_2 = D9_{16}$ (*d*) $10001.11111_2 = 11.F8_{16}$

1.48 When 2s complement notation is used, the MSB is the _____ bit. *Ans.* sign

1.49 Convert the following signed decimal numbers to their 8-bit 2s complement equivalents:
(*a*) +13, (*b*) +110, (*c*) −25, (*d*) −90.
Ans. (*a*) 00001101 (*b*) 01101110 (*c*) 11100111 (*d*) 10100110

1.50 Convert the following 2s complement numbers to their signed decimal equivalents:
(*a*) 01110000, (*b*) 00011111, (*c*) 11011001, (*d*) 11001000.
Ans. (*a*) +112 (*b*) +31 (*c*) −39 (*d*) −56

Chapter 2

Binary Codes

2-1 INTRODUCTION

Digital systems process only codes consisting of 0s and 1s (binary codes). That is due to the bistable nature of digital electronic circuits. The straight binary code was discussed in Chap. 1. Several other, special binary codes have evolved over the years to perform specific functions in digital equipment. All those codes use 0s and 1s, but their meanings may vary. Several binary codes will be detailed here, along with the methods used to translate them into decimal form. In a digital system, electronic translators (called *encoders* and *decoders*) are used for converting from code to code. The following sections will detail the process of conversion from one code to another.

2-2 WEIGHTED BINARY CODES

Straight binary numbers are somewhat difficult for people to understand. For instance, try to convert the binary number 10010110_2 to a decimal number. It turns out that $10010110_2 = 150_{10}$, but it takes quite a lot of time and effort to make this conversion without a calculator.

The *binary-coded decimal* (BCD) *code* makes conversion to decimals much easier. Figure 2-1 shows the 4-bit BCD code for the decimal digits 0-9. Note that the BCD code is a weighted code. The most significant bit has a weight of 8, and the least significant bit has a weight of only 1. This code is more precisely known as the *8421 BCD code*. The 8421 part of the name gives the weighting of each place in the 4-bit code. There are several other BCD codes that have other weights for the four place values. Because the 8421 BCD code is most popular, it is customary to refer to it simply as the BCD code.

Decimal	BCD
	8s 4s 2s 1s
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

Fig. 2-1 The 8421 BCD code

How is the decimal number 150 expressed as a BCD number? Figure 2-2*a* shows the very simple technique for converting decimal numbers to BCD (8421) numbers. Each decimal digit is converted to its 4-bit BCD equivalent (see Fig. 2-1). The decimal number 150 then equals the BCD number 000101010000.

Converting BCD numbers to decimal numbers also is quite simple. Figure 2-2*b* shows the technique. The BCD number 10010110 is first divided into groups of 4 bits starting at the binary point. Each group of 4 bits is then converted to its equivalent decimal digit, which is recorded below. The BCD number 10010110 then equals decimal 96.

Decimal	1	5	0
	↓	↓	↓
BCD	0001	0101	0000

(*a*) Decimal-to-BCD conversion

BCD	1001	0110	.
	↓	↓	
Decimal	9	6	.

(*b*) BCD-to-decimal conversion

Decimal	3	2	.	8	4
	↓	↓		↓	↓
BCD	0011	0010	.	1000	0100

(*c*) Fractional decimal-to-BCD conversion

BCD	0111	0001	.	0000	1000
	↓	↓		↓	↓
Decimal	7	1	.	0	8

(*d*) Fractional BCD-to-decimal conversion

Fig. 2-2

Figure 2-2*c* illustrates a fractional decimal number being converted to its BCD equivalent. Each decimal digit is converted to its BCD equivalent. The decimal point is dropped down and becomes the binary point. Figure 2-2*c* shows that decimal 32.84 equals the BCD number 00110010.10000100.

Convert the fractional BCD number 01110001.00001000 to its decimal equivalent. Figure 2-2*d* shows the procedure. The BCD number is first divided into groups of 4 bits starting at the binary point. Each group of four bits is then converted to its decimal equivalent. The binary point becomes the decimal point in the decimal number. Figure 2-2*d* shows the BCD number 01110001.00001000 being translated into its decimal equivalent of 71.08.

Consider converting a BCD number to its straight binary equivalent. Figure 2-3 shows the three-step procedure. Step 1 shows the BCD number being divided into 4-bit groups starting from the binary point. Each 4-bit group is translated into its decimal equivalent. Step 1 in Fig. 2-3 shows the BCD number 000100000011.0101 being translated into the decimal number 103.5.

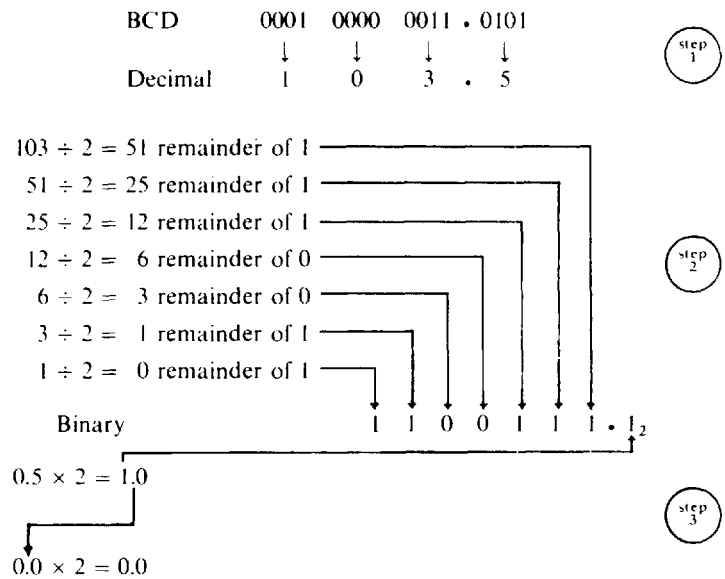


Fig. 2-3 BCD-to-binary conversion

Step 2 in Fig. 2-3 shows the integer part of the decimal number being translated into binary. The 103_{10} is converted into 1100111_2 in step 2 by the repeated divide-by-2 procedure.

Step 3 in Fig. 2-3 illustrates the fractional part of the decimal number being translated into binary. The 0.5_{10} is converted into 0.1_2 in step 3 by the repeated multiply-by-2 procedure. The integer and fractional parts of the binary number are joined. The BCD number 000100000011.0101 then equals the binary number 1100111.1.

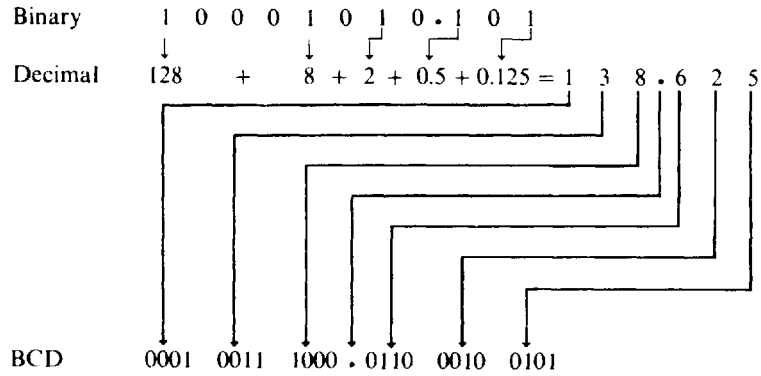


Fig. 2-4 Binary-to-BCD conversion

Note that it is usually more efficient to write down a figure in straight binary numbers than in BCD numbers. Binary numbers usually contain fewer 1s and 0s, as seen in the conversion in Fig. 2-3. Although longer, BCD numbers are used in digital systems when numbers must be easily converted to decimals.

Translate the binary number 10001010.101 into its BCD (8421) equivalent. The procedure is shown in Fig. 2-4. The binary number is first converted to its decimal equivalent. The binary number 10001010.101 then equals 138.625_{10}. Each decimal digit is then translated into its BCD equivalent. Figure 2-4 shows decimal 138.625 being converted into the BCD number 000100111000.011000100101. The entire conversion, then, translates binary 10001010.101_2 into the BCD number 000100111000.011000100101.

"Binary-coded decimal (BCD)" is a general term that may apply to any one of several codes. The most popular BCD code is the 8421 code. The numbers 8, 4, 2, and 1 stand for the weight of each bit in the 4-bit group. Examples of other weighted BCD 4-bit codes are shown in Fig. 2-5.

Decimal	8421	BCD	4221	BCD	5421	BCD
	8s 4s 2s 1s	8s 4s 2s 1s	4s 2s 2s 1s	4s 2s 2s 1s	5s 4s 2s 1s	5s 4s 2s 1s
0		0 0 0 0		0 0 0 0		0 0 0 0
1		0 0 0 1		0 0 0 1		0 0 0 1
2		0 0 1 0		0 0 1 0		0 0 1 0
3		0 0 1 1		0 0 1 1		0 0 1 1
4		0 1 0 0		1 0 0 0		0 1 0 0
5		0 1 0 1		0 1 1 1		1 0 0 0
6		0 1 1 0		1 1 0 0		1 0 0 1
7		0 1 1 1		1 1 0 1		1 0 1 0
8		1 0 0 0		1 1 1 0		1 0 1 1
9		1 0 0 1		1 1 1 1		1 1 0 0
10	0 0 0 1	0 0 0 0	0 0 0 1	0 0 0 0	0 0 0 1	0 0 0 0
11	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1	0 0 0 1
12	0 0 0 1	0 0 1 0	0 0 0 1	0 0 1 0	0 0 0 1	0 0 1 0
13	0 0 0 1	0 0 1 1	0 0 0 1	0 0 1 1	0 0 0 1	0 0 1 1

Fig. 2-5 Three weighted BCD codes

SOLVED PROBLEMS

2.1 The letters BCD stand for _____ - _____ _____.

Solution:

The letters BCD stand for binary-coded decimal.

2.2 Convert the following 8421 BCD numbers to their decimal equivalents:
(*a*) 1010 (*b*) 00010111 (*c*) 10000110 (*d*) 010101000011 (*e*) 00110010.10010100
(*f*) 0001000000000000.0101

Solution:

The decimal equivalents of the BCD numbers are as follows:

(*a*) 1010 = ERROR (no such BCD number)
(*b*) 00010111 = 17
(*c*) 10000110 = 86
(*d*) 010101000011 = 543
(*e*) 00110010.10010100 = 32.94
(*f*) 0001000000000000.0101 = 1000.5

2.3 Convert the following decimal numbers to their 8421 BCD equivalents:
(*a*) 6, (*b*) 13, (*c*) 99.9, (*d*) 872.8, (*e*) 145.6 (*f*) 21.001.

Solution:

The BCD equivalents of the decimal numbers are as follows:

(*a*) 6 = 0110
(*b*) 13 = 00010011
(*c*) 99.9 = 10011001.1001
(*d*) 872.8 = 100001110010.1000
(*e*) 145.6 = 000101000101.0110
(*f*) 21.001 = 00100001.000000000001

2.4 Convert the following binary numbers to their 8421 BCD equivalents:
(*a*) 10000, (*b*) 11100.1, (*c*) 101011.01, (*d*) 100111.11, (*e*) 1010.001,
(*f*) 1111110001.

Solution:

The BCD equivalents of the binary numbers are as follows:

(*a*) 10000 = 00010110
(*b*) 11100.1 = 00101000.0101
(*c*) 101011.01 = 01000011.00100101
(*d*) 100111.11 = 00111001.01110101
(*e*) 1010.001 = 00010000.000100100101
(*f*) 1111110001 = 0001000000001001

2.5 Convert the following 8421 BCD numbers to their binary equivalents:
(*a*) 00011000 (*b*) 01001001 (*c*) 0110.01110101 (*d*) 00110111.0101
(*e*) 01100000.00100101 (*f*) 0001.001101110101

Solution:

The binary equivalents of the BCD numbers are as follows:

(*a*) 00011000 = 10010
(*b*) 01001001 = 110001
(*c*) 0110.01110101 = 110.11
(*d*) 00110111.0101 = 100101.1
(*e*) 01100000.00100101 = 111100.01
(*f*) 0001.001101110101 = 1.011

2.6 List three weighted BCD codes.

Solution:

Three BCD codes are: (*a*) 8421 BCD code, (*b*) 4221 BCD code, (*c*) 5421 BCD code.

2.7 The 4221 BCD equivalent of decimal 98 is _____.

Solution:

The 4221 BCD equivalent of decimal 98 is 11111110.

2.8 The 5421 BCD equivalent of decimal 75 is _____.

Solution:

The 5421 BCD equivalent of decimal 75 is 10101000.

2.9 What kind of number (BCD or binary) would be easier for a worker to translate to decimal?

Solution:

BCD numbers are easiest to translate to their decimal equivalents.

2-3 NONWEIGHTED BINARY CODES

Some binary codes are nonweighted. Each bit therefore has no special weighting. Two such nonweighted codes are the excess-3 and Gray codes.

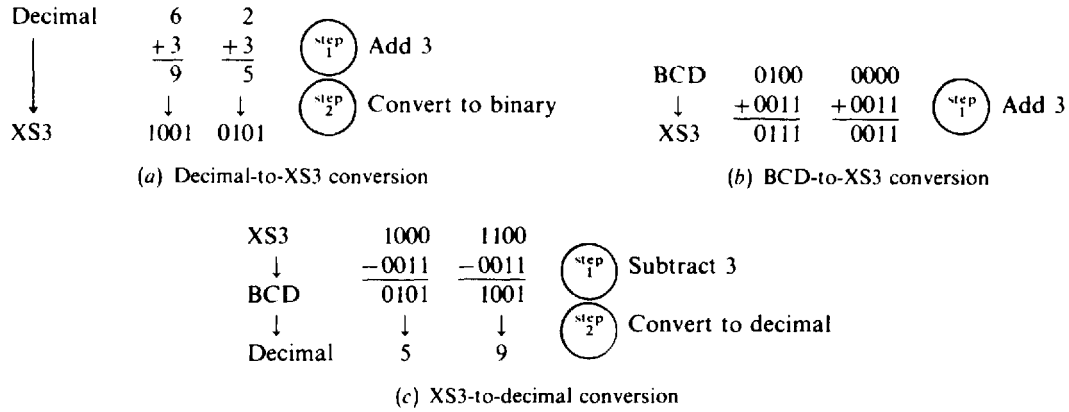
The *excess-3* (XS3) code is related to the 8421 BCD code because of its binary-coded-decimal nature. In other words, each 4-bit group in the XS3 code equals a specific decimal digit. Figure 2-6 shows the XS3 code along with its 8421 BCD and decimal equivalents. Note that the XS3 number is always *3 more* than the 8421 BCD number.

Decimal	8421 BCD		XS3 BCD	
	10s	1s	10s	1s
0		0000	0011	0011
1		0001	0011	0100
2		0010	0011	0101
3		0011	0011	0110
4		0100	0011	0111
5		0101	0011	1000
6		0110	0011	1001
7		0111	0011	1010
8		1000	0011	1011
9		1001	0011	1100
10	0001	0000	0100	0011
11	0001	0001	0100	0100

Fig. 2-6 The excess-3 (XS3) code

Consider changing the decimal number 62 to an equivalent XS3 number. Step 1 in Fig. 2-7*a* shows 3 being added to each decimal digit. Step 2 shows 9 and 5 being converted to their 8421 BCD equivalents. The decimal number 62 then equals the BCD XS3 number 10010101.

Convert the 8421 BCD number 01000000 to its XS3 equivalent. Figure 2-7*b* shows the simple procedure. The BCD number is divided into 4-bit groups starting at the binary point. Step 1 shows 3 (binary 0011) being added to each 4-bit group. The sum is the resulting XS3 number. Figure 2-7*b* shows the 8421 BCD number 01000000 being converted to its equivalent BCD XS3 number, which is 01110011.



(*a*) Decimal-to-XS3 conversion

(*b*) BCD-to-XS3 conversion

(*c*) XS3-to-decimal conversion

Fig. 2-7

Consider the conversion from XS3 code to decimal. Figure 2-7*c* shows the XS3 number 10001100 being converted to its decimal equivalent. The XS3 number is divided into 4-bit groups starting at the binary point. Step 1 shows 3 (binary 0011) being subtracted from each 4-bit group. An 8421 BCD number results. Step 2 shows each 4-bit group in the 8421 BCD number being translated into its decimal equivalent. The XS3 number 10001100 is equal to decimal 59 according to the procedure in Fig. 2-7*c*.

The XS3 code has significant value in arithmetic circuits. The value of the code lies in its ease of complementing. If each bit is complemented (0s to 1s and 1s to 0s), the resulting 4-bit word will be the 9s complement of the number. Adders can use 9s complement numbers to perform subtraction.

The *Gray code* is another nonweighted binary code. The Gray code is not a BCD-type code. Figure 2-8 compares the Gray code with equivalent binary and decimal numbers. Look carefully at the Gray code. Note that each increase in count (increment) is accompanied by *only 1 bit changing state*. Look at the change from the decimal 7 line to the decimal 8 line. In binary all four bits change state (from 0111 to 1000). In this same line the Gray code has only the left bit changing state (0100 to 1100). This change of a single bit in the code group per increment characteristic is important in some applications in digital electronics.

Decimal	Binary	Gray code	Decimal	Binary	Gray code
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

Fig. 2-8 The Gray code

Consider converting a binary number to its Gray code equivalent. Figure 2-9*a* shows the binary number 0010 being translated into its Gray code equivalent. Start at the MSB of the binary number. Transfer this to the left position in the Gray code as shown by the downward arrow. Now *add* the 8s bit to the next bit over (4s bit). The sum is 0 (0 + 0 = 0), which is transferred down and written as the second bit from the left in the Gray code. The 4s bit is now added to the 2s bit of the binary number. The sum is 1 (0 + 1 = 1) and is transferred down and written as the third bit from the left in the Gray

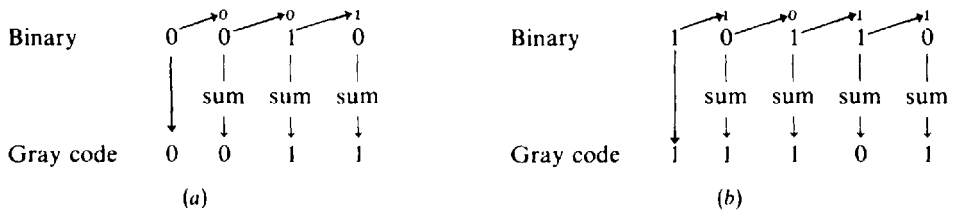


Fig. 2-9 Binary-to-Gray code conversion

code. The 2s bit is now added to the 1s bit of the binary number. The sum is 1 (1 + 0 = 1) and is transferred and written as the right bit in the Gray code. The binary 0010 is then equal to the Gray code number 0011. This can be verified in the decimal 2 line of the table in Fig. 2-8.

The rules for converting from any binary number to its equivalent Gray code number are as follows:

1. The left bit is the same in the Gray code as in the binary number.
2. Add the MSB to the bit on its immediate right and record the sum (neglect any carry) below in the Gray code line.
3. Continue adding bits to the bits on their right and recording sums until the LSB is reached.
4. The Gray code number will always have the same number of bits as the binary number.

Try these rules when converting binary 10110 to its Gray code equivalent. Figure 2-9*b* shows the MSB(1) in the binary number being transferred down and written as part of the Gray code number. The 16s bit is then added to the 8s bit of the binary number. The sum is 1 (1 + 0 = 1), which is recorded in the Gray code (second bit from left). Next the 8s bit is added to the 4s bit of the binary number. The sum is 1 (0 + 1 = 1), which is recorded in the Gray code (third bit from the left). Next the 4s bit is added to the 2s bit of the binary number. The sum is 0 (1 + 1 = 10) because the carry is dropped. The 0 is recorded in the second position from the right in the Gray code. Next the 2s bit is added to the 1s bit of the binary number. The sum is 1 (1 + 0 = 1), which is recorded in the Gray code (right bit). The process is complete. Figure 2-9*b* shows the binary number 10110 being translated into the Gray code number 11101.

Convert the Gray code number 1001 to its equivalent binary number. Figure 2-10*a* details the procedure. First the left bit (1) is transferred down to the binary line to form the 8s bit. The 8s bit of the binary number is transferred (see arrow) up above the next Gray code bit, and the two are added. The sum is 1 (1 + 0 = 1), which is written in the 4s bit place in the binary number. The 4s bit (1) is then added to the next Gray code bit. The sum is 1 (1 + 0 = 1). This 1 is written in the 2s place of the binary number. The binary 2s bit (1) is added to the right Gray code bit. The sum is 0 (1 + 1 = 10) because the carry is neglected. This 0 is written in the 1s place of the binary number. Figure 2-10*a* shows the Gray code number 1001 being translated into its equivalent binary number 1110. This conversion can be verified by looking at decimal line 14 in Fig. 2-8.

Convert the 6-bit Gray code number 011011 to its 6-bit binary equivalent. Start at the left and follow the arrows in Fig. 2-10*b*. Follow the procedure, remembering that a 1 + 1 = 10. The carry of 1 is neglected, and the 0 is recorded on the binary line. Figure 2-10*b* shows that the Gray code number 011011 is equal to the binary number 010010.

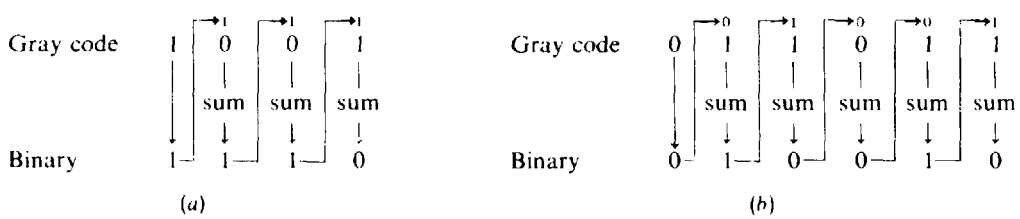


Fig. 2-10 Gray code-to-binary conversions

SOLVED PROBLEMS

2.10 The letters and numbers XS3 stand for _____ – _____ code.

Solution:

XS3 stands for the excess-3 code.

2.11 The _____ (8421, XS3) BCD code is an example of a nonweighted code.

Solution:

The XS3 BCD code is an example of a nonweighted code.

2.12 The _____ (Gray, XS3) code is a BCD code.

Solution:

The XS3 code is a BCD code.

2.13 Convert the following decimal numbers to their XS3 code equivalents:
(*a*) 9, (*b*) 18, (*c*) 37, (*d*) 42, (*e*) 650.

Solution:

The XS3 equivalents of the decimal numbers are as follows:
(*a*) 9 = 1100 (*c*) 37 = 01101010 (*e*) 650 = 100110000011
(*b*) 18 = 01001011 (*d*) 42 = 01110101

2.14 Convert the following 8421 BCD numbers to their XS3 code equivalents:
(*a*) 0001, (*b*) 0111, (*c*) 01100000, (*d*) 00101001, (*e*) 10000100.

Solution:

The XS3 equivalents of the 8421 BCD numbers are as follows:
(*a*) 0001 = 0100 (*c*) 01100000 = 10010011 (*e*) 10000100 = 10110111
(*b*) 0111 = 1010 (*d*) 00101001 = 01011100

2.15 Convert the following XS3 numbers to their decimal equivalents:
(*a*) 0011, (*b*) 01100100, (*c*) 11001011, (*d*) 10011010, (*e*) 10000101.

Solution:

The decimal equivalents of the XS3 numbers are as follows:
(*a*) 0011 = 0 (*c*) 11001011 = 98 (*e*) 10000101 = 52
(*b*) 01100100 = 31 (*d*) 10011010 = 67

2.16 The _____ (Gray, XS3) code is usually used in arithmetic applications in digital circuits.

Solution:

The XS3 code is usually used in arithmetic applications.

2.17 Convert the following straight binary numbers to their Gray code equivalents:
(*a*) 1010, (*b*) 10000, (*c*) 10001, (*d*) 10010, (*e*) 10011.

Solution:

The Gray code equivalents of the binary numbers are as follows:
(*a*) 1010 = 1111 (*c*) 10001 = 11001 (*e*) 10011 = 11010
(*b*) 10000 = 11000 (*d*) 10010 = 11011

2.18 Convert the following Gray code numbers to their straight binary equivalents:
(*a*) 0100, (*b*) 11111, (*c*) 10101, (*d*) 110011, (*e*) 011100.

Solution:

The binary equivalents of the Gray code numbers are as follows:
(*a*) 0100 = 0111 (*c*) 10101 = 11001 (*e*) 011100 = 010111
(*b*) 11111 = 10101 (*d*) 110011 = 100010

2.19 The Gray code's most important characteristic is that, when the count is incremented by 1, _____ (more than, only) 1 bit will change state.

Solution:

The Gray code's most important characteristic is that, when the count is incremented by 1, only 1 bit will change state.

2-4 ALPHANUMERIC CODES

Binary 0s and 1s have been used to represent various numbers to this point. Bits can also be coded to represent letters of the alphabet, numbers, and punctuation marks. One such 7-bit code is the *American Standard Code for Information Interchange* (ASCII, pronounced "ask-ee"), shown in Fig. 2-11. Note that the letter A is represented by 1000001, whereas B in the ASCII code is 1000010.

Character	ASCII	EBCDIC	Character	ASCII	EBCDIC
Space	010 0000	0100 0000	A	100 0001	1100 0001
!	010 0001	0101 1010	B	100 0010	1100 0010
"	010 0010	0111 1111	C	100 0011	1100 0011
#	010 0011	0111 1011	D	100 0100	1100 0100
$	010 0100	0101 1011	E	100 0101	1100 0101
%	010 0101	0110 1100	F	100 0110	1100 0110
&	010 0110	0101 0000	G	100 0111	1100 0111
'	010 0111	0111 1101	H	100 1000	1100 1000
(	010 1000	0100 1101	I	100 1001	1100 1001
)	010 1001	0101 1101	J	100 1010	1101 0001
*	010 1010	0101 1100	K	100 1011	1101 0010
+	010 1011	0100 1110	L	100 1100	1101 0011
,	010 1100	0110 1011	M	100 1101	1101 0100
-	010 1101	0110 0000	N	100 1110	1101 0101
.	010 1110	0100 1011	O	100 1111	1101 0110
/	010 1111	0110 0001	P	101 0000	1101 0111
0	011 0000	1111 0000	Q	101 0001	1101 1000
1	011 0001	1111 0001	R	101 0010	1101 1001
2	011 0010	1111 0010	S	101 0011	1110 0010
3	011 0011	1111 0011	T	101 0100	1110 0011
4	011 0100	1111 0100	U	101 0101	1110 0100
5	011 0101	1111 0101	V	101 0110	1110 0101
6	011 0110	1111 0110	W	101 0111	1110 0110
7	011 0111	1111 0111	X	101 1000	1110 0111
8	011 1000	1111 1000	Y	101 1001	1110 1000
9	011 1001	1111 1001	Z	101 1010	1110 1001

Fig. 2-11 Alphanumeric codes

The ASCII code is used extensively in small computer systems to translate from the keyboard characters to computer language. The chart in Fig. 2-11 is not a complete list of all the combinations in the ASCII code.

Codes that can represent both letters and numbers are called *alphanumeric codes*. Another alphanumeric code that is widely used is the *Extended Binary-Coded Decimal Interchange Code* (EBCDIC, pronounced "eb-si-dik"). Part of the EBCDIC code is shown in Fig. 2-11. Note that the EBCDIC code is an 8-bit code and therefore can have more variations and characters than the ASCII code can have. The EBCDIC code is used in many larger computer systems.

The alphanumeric ASCII code is the modern code for getting information into and out of microcomputers. ASCII is used when interfacing computer keyboards, printers, and video displays. ASCII has become the standard input/output code for microcomputers.

Other alphanumeric codes that you may encounter are:

1. 7-bit BCDIC (*Binary-Coded Decimal Interchange Code*).
2. 8-bit EBCDIC (*Extended Binary-Coded Decimal Interchange Code*). Used on some IBM equipment.
3. 7-bit Selectric. Used to control the spinning ball on IBM Selectric typewriters.
4. 12-bit Hollerith. Used on punched paper cards.

SOLVED PROBLEMS

2.20 Binary codes that can represent both numbers and letters are called _____ codes.

Solution:

Alphanumeric codes can represent both numbers and letters.

2.21 The following are abbreviations for what?
(*a*) ASCII (*b*) EBCDIC

Solution:

(*a*) ASCII = American Standard Code for Information Interchange
(*b*) EBCDIC = Extended Binary-Coded Decimal Interchange Code

2.22 Refer to Fig. 2-12. The ASCII keyboard-encoder output would be _____ if the K on the typewriter-like keyboard were pressed.

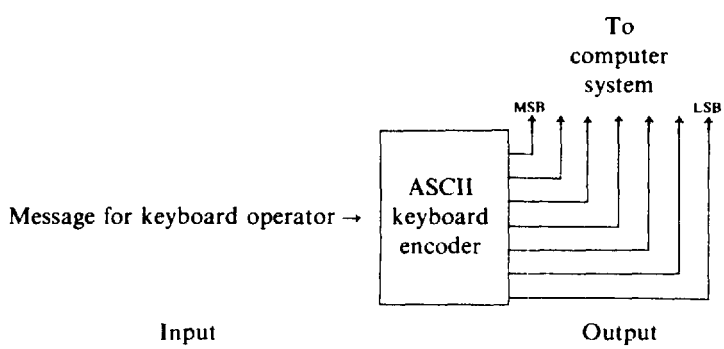


Fig. 2-12 ASCII keyboard-encoder system

Solution:

The ASCII output would be 1001011 if the K on the keyboard were pressed.

2.23 Refer to Fig. 2-12. List the 12 ASCII keyboard-encoder outputs for entering the message "pay $1000.00."

Solution:

The ASCII codes for the characters in the message are as follows:

(*a*)	P = 1010000	(*d*)	Space = 0100000	(*g*)	0 = 0110000	(*j*)	. = 0101110
(*b*)	A = 1000001	(*e*)	$ = 0100100	(*h*)	0 = 0110000	(*k*)	0 = 0110000
(*c*)	Y = 1011001	(*f*)	1 = 0110001	(*i*)	0 = 0110000	(*l*)	0 = 0110000

2.24 The _____ code is a 12-bit alphanumeric code used on punched paper cards.

Solution:

The 12-bit Hollerith code is used on punched cards.

2.25 The 7-bit _____ code is considered the industry standard for input/output on microcomputers.

Solution:

The 7-bit ASCII (American Standard Code for Information Interchange) code is considered the industry standard for microcomputer inputs and outputs.

Supplementary Problems

2.26 Electronic devices that translate from one code to another are called __(*a*)__ or __(*b*)__.
Ans. (*a*) encoders (*b*) decoders

2.27 Convert the following 8421 BCD numbers to their decimal equivalents:
(*a*) 10010000, (*b*) 11111111, (*c*) 0111.0011, (*d*) 01100001.00000101.
Ans. (*a*) 10010000 = 90 (*c*) 0111.0011 = 7.3
(*b*) 11111111 = ERROR (no such BCD number) (*d*) 01100001.00000101 = 61.05

2.28 Convert the following decimal numbers to their 8421 BCD equivalents:
(*a*) 10, (*b*) 342, (*c*) 679.8, (*d*) 500.6.
Ans. (*a*) 10 = 00010000 (*c*) 679.8 = 011001111001.1000
(*b*) 342 = 001101000010 (*d*) 500.6 = 010100000000.0110

2.29 Convert the following binary numbers to their 8421 BCD equivalents:
(*a*) 10100, (*b*) 11011.1, (*c*) 100000.01, (*d*) 111011.11.
Ans. (*a*) 10100 = 00100000 (*c*) 100000.01 = 00110010.00100101
(*b*) 11011.1 = 00100111.0101 (*d*) 111011.11 = 01011001.01110101

2.30 Convert the following 8421 BCD numbers to their binary equivalents:
(*a*) 01011000, (*b*) 000100000000, (*c*) 1001.01110101, (*d*) 0011.0000011000100101.
Ans. (*a*) 01011000 = 111010 (*c*) 1001.01110101 = 1001.11
(*b*) 000100000000 = 1100100 (*d*) 0011.0000011000100101 = 11.0001

2.31 The 4221 BCD equivalent of decimal 74 is _____. *Ans.* 11011000

2.32 The 5421 BCD equivalent of decimal 3210 is _____. *Ans.* 0011001000010000

2.33 The BCD code is convenient when translations must be made to _____ (binary, decimal) numbers.
Ans. decimal

2.34 "Excess-3" code is often shortened to _____. *Ans.* XS3

2.35 Convert the following decimal numbers to their XS3 code equivalents:
(*a*) 7, (*b*) 16, (*c*) 32, (*d*) 4089.
Ans. (*a*) 7 = 1010 (*c*) 32 = 01100101
(*b*) 16 = 01001001 (*d*) 4089 = 0111001110111100

2.36 Convert the following XS3 numbers to their decimal equivalents:
(*a*) 1100, (*b*) 10101000, (*c*) 100001110011, (*d*) 0100101101100101.
Ans. (*a*) 1100 = 9 (*c*) 100001110011 = 540
(*b*) 10101000 = 75 (*d*) 0100101101100101 = 1832

2.37 Convert the following straight binary numbers to their Gray code equivalents:
(*a*) 0110, (*b*) 10100, (*c*) 10101, (*d*) 10110.
Ans. (*a*) 0110 = 0101 (*b*) 10100 = 11110 (*c*) 10101 = 11111 (*d*) 10110 = 11101

2.38 Convert the following Gray code numbers to their straight binary equivalents:
(*a*) 0001, (*b*) 11100, (*c*) 10100, (*d*) 10101.
Ans. (*a*) 0001 = 0001 (*b*) 11100 = 10111 (*c*) 10100 = 11000 (*d*) 10101 = 11001

2.39 EBCDIC is a(n) _____-bit alphanumeric code used in some IBM equipment.
Ans. 8

2.40 The 7-bit alphanumeric _____ code serves as the industry standard for input/output on microcomputers.
Ans. ASCII

Chapter 3

Basic Logic Gates

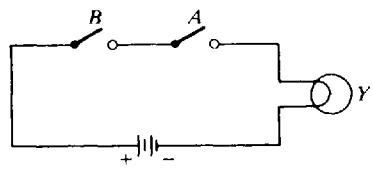
3-1 INTRODUCTION

The *logic gate* is the basic building block in digital systems. Logic gates operate with binary numbers. Gates are therefore referred to as binary logic gates. All voltages used with logic gates will be either HIGH or LOW. In this book, a HIGH voltage will mean a binary 1. A LOW voltage will mean a binary 0. Remember that logic gates are electronic circuits. These circuits will respond only to HIGH voltages (called 1s) or LOW (ground) voltages (called 0s).

All digital systems are constructed by using only three basic logic gates. These basic gates are called the AND gate, the OR gate, and the NOT gate. This chapter deals with these very important basic logic gates, or functions.

3-2 THE AND GATE

The AND gate is called the "all or nothing" gate. The schematic in Fig. 3-1*a* shows the idea of the AND gate. The lamp (Y) will light only when *both* input switches (A and B) are closed. All the possible combinations for switches A and B are shown in Fig. 3-1*b*. The table in this figure is called a *truth table*. The truth table shows that the output (Y) is *enabled* (lit) only when both inputs are closed.



(*a*) AND circuit using switches

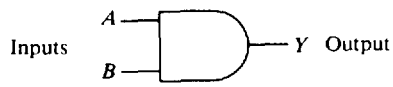
Input switches		Output light
B	*A*	*Y*
open	open	no
open	closed	no
closed	open	no
closed	closed	yes

(*b*) Truth table

Fig. 3-1

The standard *logic symbol* for the AND gate is drawn in Fig. 3-2*a*. This symbol shows the inputs as A and B. The output is shown as Y. This is the symbol for a 2-input AND gate. The truth table for the 2-input AND gate is shown in Fig. 3-2*b*. The inputs are shown as *bi*nary digi*ts* (bits). Note that only when both input A and input B are 1 will the output be 1. Binary 0 is defined as a LOW, or ground, voltage. Binary 1 is defined as a HIGH voltage. In this book, a HIGH voltage will mean about +5 volts (V) if the integrated circuits (ICs) being used are from the TTL family.

Boolean algebra is a form of symbolic logic that shows how logic gates operate. A *Boolean expression* is a "shorthand" method of showing what is happening in a logic circuit. The Boolean expression for the circuit in Fig. 3-2 is



(*a*) AND-gate symbol

Inputs		Output
B	*A*	*Y*
0	0	0
0	1	0
1	0	0
1	1	1

0 = low voltage
1 = high voltage

(*b*) AND truth table

Fig. 3-2

$$A \cdot B = Y$$

The Boolean expression is read as A AND ($\cdot$ means AND) B equals the output Y. The dot ($\cdot$) means the logic function AND in Boolean algebra, *not* multiply as in regular algebra.

Sometimes the dot ($\cdot$) is left out of the Boolean expression. The Boolean expression for the 2-input AND gate is then:

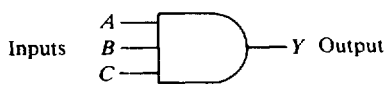
$$AB = Y$$

The Boolean expression reads A AND B equals the output Y.

A logic circuit will often have three *variables*. Figure 3-3*a* shows the Boolean expression for a 3-input AND gate. The input variables are A, B, and C. The output is shown as Y. The logic symbol for this 3-input AND expression is drawn in Fig. 3-3*b*. The three inputs (A, B, C) are on the left of the symbol. The single output (Y) is on the right of the symbol. The truth table in Fig. 3-3*c* shows the eight possible combinations of the variables A, B, and C. Note that the top line in the table is the binary count 000. The binary count then proceeds upward to 001, 010, 011, 100, 101, 110, and finally 111. Note that *only* when all inputs are 1 is the output of the AND gate enabled with a 1.

$$A \cdot B \cdot C = Y$$

(*a*) Three-variable Boolean expression



(*b*) 3-input AND gate symbol

Inputs			Output
C	*B*	*A*	*Y*
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

(*c*) Truth table with three variables

Fig. 3-3

Consider the AND truth tables shown in Figs. 3-2*b* and 3-3*c*. In each truth table the *unique output* from the AND gate is a HIGH only when *all* inputs are HIGH. Designers look at each gate's unique output when deciding which gate will perform a certain task.

The laws of Boolean algebra govern how AND gates operate. The formal laws for the *AND function* are:

$$A \cdot 0 = 0$$
$$A \cdot 1 = A$$
$$A \cdot A = A$$
$$A \cdot \overline{A} = 0$$

You can prove the accuracy of these laws by referring back to the truth table in Fig. 3-2. These are general statements that are always true about the AND function. AND gates must follow these laws. Note the bar over the variable in the last law. The bar over the variable means *not A*, or the opposite of *A*.

SOLVED PROBLEMS

3.1 Write the Boolean expression for a 4-input AND gate.

Solution:

$A \cdot B \cdot C \cdot D = Y$ or $ABCD = Y$

3.2 Draw the logic symbol for a 4-input AND gate.

Solution:

See Fig. 3-4.

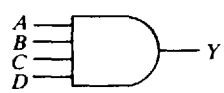


Fig. 3-4 Symbol for a 4-input AND gate

3.3 Draw a truth table for a 4-input AND gate.

Solution:

Inputs				Output	Inputs				Output
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	1

3.4 In Fig. 3-5, what would the output pulse train look like?

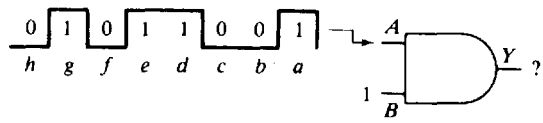


Fig. 3-5 Pulse-train problem

Solution:

In Fig. 3-5, the output waveform would look exactly like the input waveform at input A.

pulse $a = 1$ pulse $c = 0$ pulse $e = 1$ pulse $g = 1$
pulse $b = 0$ pulse $d = 1$ pulse $f = 0$ pulse $h = 0$

3.5 In Fig. 3-6, what would the output pulse train look like? Note that two pulse trains are being ANDed.

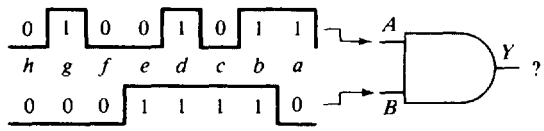


Fig. 3-6 Pulse-train problem

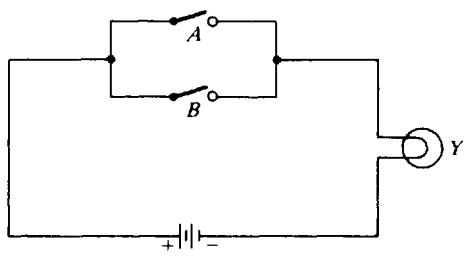
Solution:

In Fig. 3-6, the output pulses would be as follows:

pulse $a = 0$ pulse $c = 0$ pulse $e = 0$ pulse $g = 0$
pulse $b = 1$ pulse $d = 1$ pulse $f = 0$ pulse $h = 0$

3-3 THE OR GATE

The OR gate is called the "any or all" gate. The schematic in Fig. 3-7*a* shows the idea of the OR gate. The lamp (Y) will glow when either switch A *or* switch B is closed. The lamp will also glow when both switches A and B are closed. The lamp (Y) will *not* glow when both switches (A and B) are open. All the possible switch combinations are shown in Fig. 3-7*b*. The truth table details the *OR function* of the switch and lamp circuit. The output of the OR circuit will be enabled (lamp lit) when any or all input switches are closed.



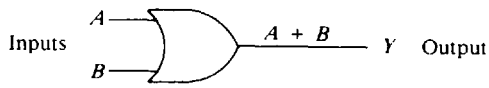
(*a*) OR circuit using switches

Input switches		Output light
B	A	Y
open	open	no
open	closed	yes
closed	open	yes
closed	closed	yes

(*b*) Truth table

Fig. 3-7

The standard logic symbol for an OR gate is drawn in Fig. 3-8*a*. Note the different shape of the OR gate. The OR gate has two inputs labeled A and B. The output is labeled Y. The shorthand Boolean expression for this OR function is given as $A + B = Y$. Note that the plus (+) symbol means OR in Boolean algebra. The expression ($A + B = Y$) is read as A OR (+ means OR) B equals output Y. You will note that the plus sign does *not* mean to add as it does in regular algebra.



(*a*) OR-gate symbol

Inputs		Output
B	A	Y
0	0	0
0	1	1
1	0	1
1	1	1

0 = low voltage
1 = high voltage

(*b*) OR truth table

Fig. 3-8

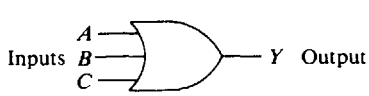
The truth table for the 2-input OR gate is drawn in Fig. 3-8*b*. The input *variables* (A and B) are given on the left. The resulting output (Y) is shown in the right column of the table. The OR gate is *enabled* (output is 1) anytime a 1 appears at any or all of the inputs. As before, a 0 is defined as a LOW (ground) voltage. A 1 in the truth table represents a HIGH (+5 V) voltage.

The Boolean expression for a 3-input OR gate is written in Fig. 3-9*a*. The expression reads A OR B OR C equals output Y. The plus sign again signifies the OR function.

A logic symbol for the 3-input OR gate is drawn in Fig. 3-9*b*. Inputs A, B, and C are shown on the left of the symbol. Output Y is shown on the right of the OR symbol. This symbol represents some circuit that will perform the OR function.

$$A + B + C = Y$$

(*a*) Three-variable Boolean expression



(*b*) 3-input OR gate symbol

Inputs			Output
C	B	A	Y
0	0	0	0
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	1

(*c*) Truth table with three variables

Fig. 3-9

A truth table for the 3-input OR gate is shown in Fig. 3-9*c*. The variables (A, B, and C) are shown on the left side of the table. The output (Y) is listed in the right column. Anytime a 1 appears at any input, the output will be 1.

Consider the OR truth tables in Figs. 3-8*b* and 3-9*c*. In each truth table the *unique output* from the OR gate is a LOW only when *all* inputs are LOW. Designers look at each gate's unique output when deciding which gate will perform a certain task.

The laws of Boolean algebra govern how an OR gate will operate. The formal laws for the OR function are:

$$A + 0 = A$$
$$A + 1 = 1$$
$$A + A = A$$
$$A + \overline{A} = 1$$

Looking at the truth table in Fig. 3-8 will help you check these laws. These general statements are always true of the OR function. The bar over the last variable means *not* A, or the opposite of A.

SOLVED PROBLEMS

3.6 Write the Boolean expression for a 4-input OR gate.

Solution:

$A + B + C + D = Y$

3.7 Draw the logic symbol for a 4-input OR gate.

Solution:

See Fig. 3-10.

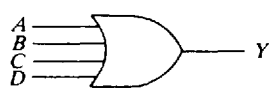


Fig. 3-10 Symbol used for a 4-input OR gate

3.8 Draw a truth table for a 4-input OR gate.

Solution:

Inputs				Output	Inputs				Output
D	C	B	A	Y	D	C	B	A	Y
0	0	0	0	0	1	0	0	0	1
0	0	0	1	1	1	0	0	1	1
0	0	1	0	1	1	0	1	0	1
0	0	1	1	1	1	0	1	1	1
0	1	0	0	1	1	1	0	0	1
0	1	0	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1

3.9 In Fig. 3-11, what would the output pulse train look like?

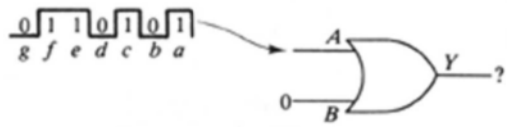


Fig. 3-11 Pulse-train problem

Solution:

In Fig. 3-11, the output waveform would look exactly like the input waveform at input A.

pulse $a = 1$ pulse $c = 1$ pulse $e = 1$ pulse $g = 0$
pulse $b = 0$ pulse $d = 0$ pulse $f = 1$

3.10 In Fig. 3-12, what would the output pulse train look like? Note that two pulse trains are being ORed together.

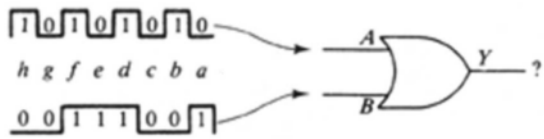


Fig. 3-12 Pulse-train problem

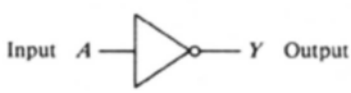
Solution:

In Fig. 3-12, the output pulses would be as follows:

pulse $a = 1$ pulse $c = 0$ pulse $e = 1$ pulse $g = 0$
pulse $b = 1$ pulse $d = 1$ pulse $f = 1$ pulse $h = 1$

3-4 THE NOT GATE

A NOT gate is also called an *inverter*. A NOT gate, or inverter, is an unusual gate. The NOT gate has only one input and one output. Figure 3-13*a* illustrates the logic symbol for the inverter, or NOT gate.



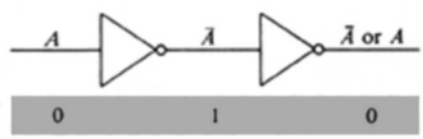
(*a*) NOT-gate symbol

Input	Output
A	Y
0	1
1	0

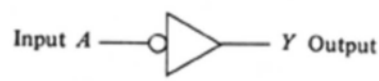
(*b*) NOT-gate truth table

$A = \bar{\bar{A}}$

(*c*) NOT Boolean expression



(*d*) Double inversion



(*e*) Alternative inverter symbol

Fig. 3-13

The process of inverting is simple. Figure 3-13*b* is the truth table for the NOT gate. The input is always changed to its opposite. If the input is 0, the NOT gate will give its *complement*, or opposite, which is 1. If the input to the NOT gate is a 1, the circuit will complement it to give a 0. This inverting is also called *complementing* or *negating*. The terms negating, complementing, and inverting all mean the same thing.

The Boolean expression for inverting is shown in Fig. 3-13*c*. The expression $A = \overline{A}$ reads as A equals the output *not* A. The bar over the A means to complement A. Figure 3-13*d* illustrates what would happen if two inverters were used. The Boolean expressions are written above the lines between the inverters. The input A is inverted to $\overline{A}$ (not A). The $\overline{A}$ is then inverted again to form $\overline{\overline{A}}$ (not not A). The double inverted A ($\overline{\overline{A}}$) is equal to the original A, as shown in Fig. 3-13*d*. In the shaded section below the inverters, a 0 bit is the input. The 0 bit is complemented to a 1. The 1 bit is complemented again back to a 0. After a digital signal goes through two inverters, it is restored to its original form.

An alternative logic symbol for the NOT gate or inverter is shown in Fig. 3-13*e*. The *invert bubble* may be on either the input or the output side of the triangular symbol. When the invert bubble appears on the input side of the NOT symbol (as in Fig. 3-13*e*), the designer is usually trying to suggest that this is an *active LOW* input. An active LOW input requires a LOW to activate some function in the logic circuit. The alternative NOT gate symbol is commonly used in manufacturer's logic diagrams.

The laws of Boolean algebra govern the action of the inverter, or NOT gate. The formal Boolean algebra laws for the NOT gate are as follows:

$$\overline{0} = 1 \qquad \overline{1} = 0$$

$$\text{If } A = 1, \text{ then } \overline{A} = 0$$

$$\text{If } A = 0, \text{ then } \overline{A} = 1$$

$$\overline{\overline{A}} = A$$

You can check these general statements against the truth table and diagrams in Fig. 3-13.

SOLVED PROBLEMS

3.11 In Fig. 3-14, what is the output at point (*e*) if the input at point (*a*) is a 0 bit?

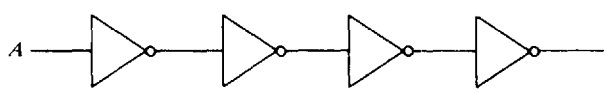


Fig. 3-14 Inverter problem

Solution:

The output at point (*e*) is a 0 bit.

3.12 What is the Boolean expression at point (*b*) in Fig. 3-14?

Solution:

The Boolean expression at point (*b*) is $\overline{A}$ (not A).

3.13 What is the Boolean expression at point (*c*) in Fig. 3-14?

Solution:

The Boolean expression at point (*c*) is $\overline{\overline{A}}$ (not not A). $\overline{\overline{A}}$ equals A according to the laws of Boolean algebra.

3.14 What is the Boolean expression at point (d) in Fig. 3-14?

Solution:

The Boolean expression at point (d) is $\overline{\overline{\overline{A}}}$ (not not not A). $\overline{\overline{\overline{A}}}$ equals $\overline{A}$ (not A).

3.15 What is the output at point (d) in Fig. 3-14 if the input at point (a) is a 1 bit?

Solution:

The output at point (d) is a 0 bit.

3.16 The NOT gate is said to *invert* its input. List two other words we can use instead of "invert."

Solution:

The words *complement* and *negate* also mean to invert.

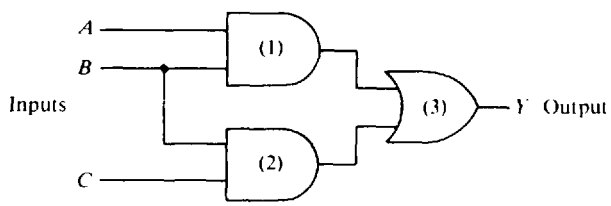
3.17 The NOT gate can have _____ (one, many) input variable(s).

Solution:

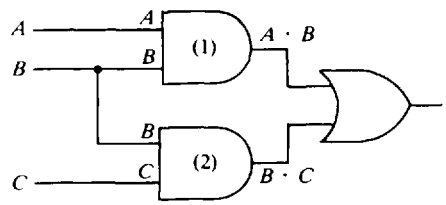
The NOT gate can have one input variable.

3-5 COMBINING LOGIC GATES

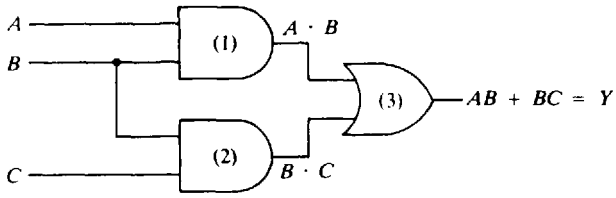
Many everyday digital logic problems use several logic gates. The most common *pattern* of gates is shown in Fig. 3-15*a*. This pattern is called the AND-OR pattern. The outputs of the AND gates (1 and 2) are feeding the inputs of the OR gate (3). You will note that this logic circuit has three inputs (A, B, and C). The output of the entire circuit is labeled Y.



(a) AND-OR logic circuit



(b) Boolean expressions at the outputs of the AND gates



(c) Boolean expression at the output of the OR gate

Fig. 3-15

Let us first determine the Boolean expression that will describe this logic circuit. Begin the examination at gate (1). This is a 2-input AND gate. The output of this gate will be $A \cdot B$ (A AND B). This expression is written at the output of gate (1) in Fig. 3-15*b*. Gate (2) is also a 2-input AND gate. The output of this gate will be $B \cdot C$ (B AND C). This expression is written at the output of gate (2). Next the outputs of gates (1) and (2) are ORed together by gate (3). Figure 3-15*c* shows AB being ORed with BC. The resulting Boolean expression is $AB + BC = Y$. The Boolean expression $AB + BC = Y$ is read as (A AND B) OR (B AND C) will equal a 1 at output Y. You will note that the ANDing is done first, and finally the ORing is done.

The next question arises. What is the truth table for the AND-OR logic diagram in Fig. 3-15? Figure 3-16 will help us determine the truth table for the Boolean expression $AB + BC = Y$. The Boolean expression tells us that if *both* variables A AND B are 1, the output will be **1**. Figure 3-16 illustrates that the last two lines of the truth table have 1s in *both* the A and B positions. Therefore an output **1** is placed under the Y column.

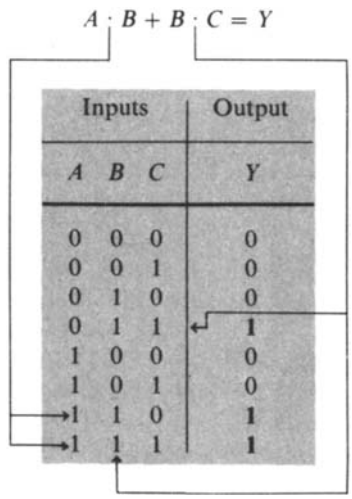


Fig. 3-16 Complementing output column of truth table from a Boolean expression

The Boolean expression then goes on to say that another condition will also generate an output of **1**. The expression says that B AND C will also generate a **1** output. Looking at the truth table, it is found that the fifth line from the bottom has 1s in *both* the B AND C positions. The bottom line also has 1s in *both* the B AND C positions. Both of these lines will generate a **1** output. The bottom line already has a **1** under the output column (Y). The fifth line up will also get a **1** in the output column (Y). These are the only combinations that will generate a **1** output. The rest of the combinations are then listed as 0 outputs under column Y.

SOLVED PROBLEMS

3.18 What is the Boolean expression for the AND-OR logic diagram in Fig. 3-17?

Solution:

The Boolean expression for the logic circuit shown in Fig. 3-17 is

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

The expression is read as (not A AND B) OR (A AND C) equals output Y.

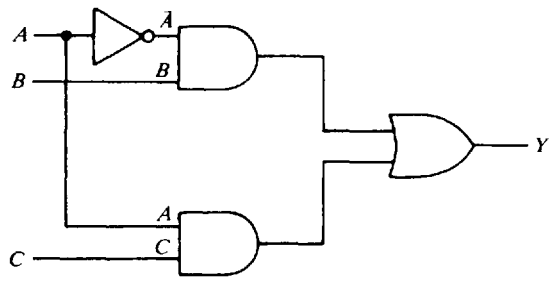


Fig. 3-17 AND-OR logic-circuit problem

3.19 What is the truth table for the logic diagram in Fig. 3-17?

Solution:

Inputs			Output	Inputs			Output
A	B	C	Y	A	B	C	Y
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	1	1	1	1	1

3.20 What is the Boolean expression for the AND-OR logic diagram in Fig. 3-18?

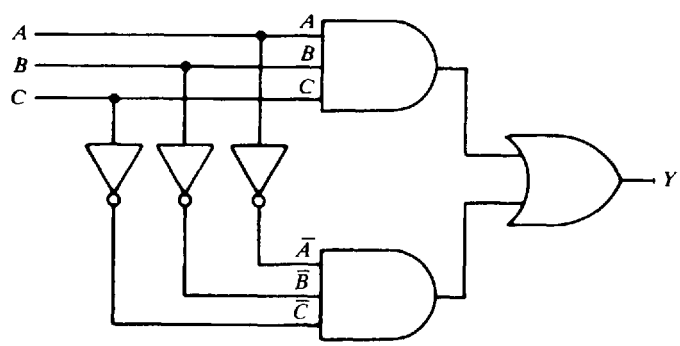


Fig. 3-18 AND-OR logic-circuit problem

Solution:

The Boolean expression for the logic circuit shown in Fig. 3-18 is

$$ABC + \bar{A}\bar{B}\bar{C} = Y$$

The expression reads as (A AND B AND C) OR (not A AND not B AND not C) equals output Y.

3.21 What is the truth table for the logic diagram in Fig. 3-18?

Solution:

Inputs			Output	Inputs			Output
A	B	C	Y	A	B	C	Y
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0
0	1	1	0	1	1	1	1

3.22 What is the Boolean expression for the AND-OR logic diagram in Fig. 3-19?

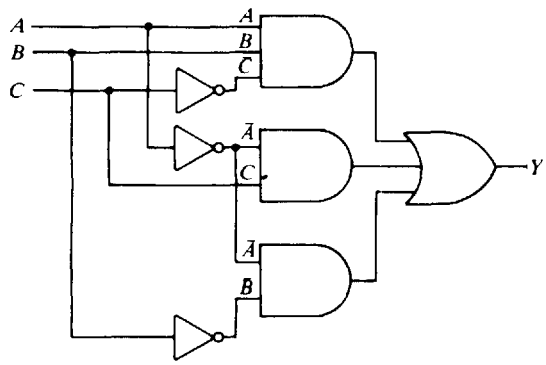


Fig. 3-19 AND-OR logic-circuit problem

Solution:

The Boolean expression for the logic circuit shown in Fig. 3-19 is $AB\overline{C} + \overline{A}C + \overline{A}\,\overline{B} = Y$. The expression reads as (A AND B AND not C) OR (not A AND C) OR (not A AND not B) equals output Y.

3.23 What is the truth table for the logic diagram in Fig. 3-19?

Solution:

Inputs			Output	Inputs			Output
A	B	C	Y	A	B	C	Y
0	0	0	1	1	0	0	0
0	0	1	1	1	0	1	0
0	1	0	0	1	1	0	1
0	1	1	1	1	1	1	0

3-6 USING PRACTICAL LOGIC GATES

Logic functions can be implemented in several ways. In the past, vacuum-tube and relay circuits performed logic functions. Presently tiny *integrated circuits* (ICs) perform as logic gates. These ICs contain the equivalent of miniature resistors, diodes, and transistors.

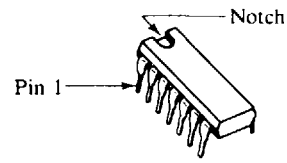


Fig. 3-20 14-pin DIP integrated circuit

A popular type of IC is illustrated in Fig. 3-20. This case style is referred to as a *dual-in-line package* (DIP) by IC manufacturers. This particular IC is called a 14-pin DIP integrated circuit.

Note that immediately *counterclockwise* from the notch on the IC shown in Fig. 3-20 is pin number 1. The pins are numbered counterclockwise from 1 to 14 *when viewed from the top* of the IC. Manufacturers of ICs provide pin diagrams similar to the one in Fig. 3-21 for a 7408 IC. Note that this IC contains four 2-input AND gates; thus it is called a *quadruple 2-input AND gate*. Figure 3-21 shows the IC pins numbered from 1 through 14 in a counterclockwise direction from the notch. The *power connections* to the IC are the GND (pin 7) and V_{CC} (pin 14) pins. All other pins are the inputs and outputs to the four AND gates. The 7408 IC is part of a family of logic devices. It is one of many

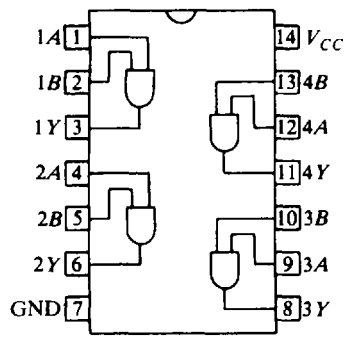
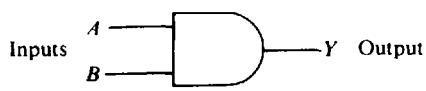
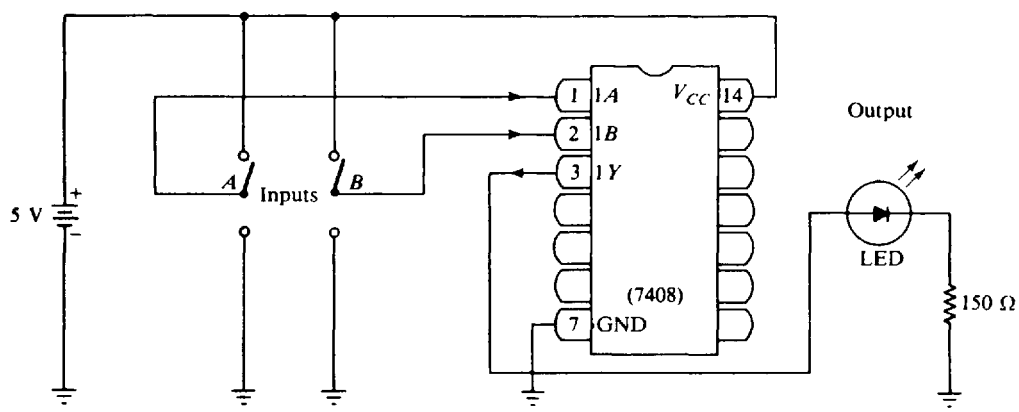


Fig. 3-21 Pin diagram for a 7408 IC



(*a*) AND-gate logic symbol



(*b*) Wiring an AND gate using a 7408 IC

Fig. 3-22

devices in the *transistor-transistor logic* (TTL) family of logic circuits. TTL devices are currently among the most popular logic devices.

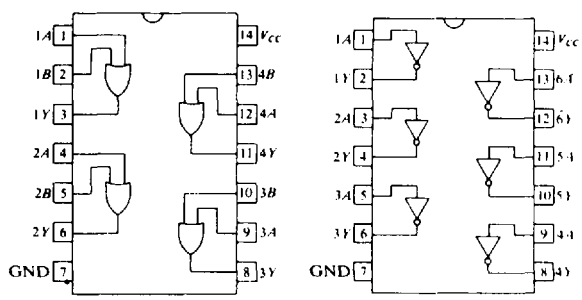
Given the logic diagram in Fig. 3-22*a*, wire a circuit using a 7408 IC. A wiring diagram for the circuit is shown in Fig. 3-22*b*. A 5-V power supply is used with all TTL devices. The positive (V_{CC}) and negative (GND) power connections are made to pins 14 and 7. Input switches (*A* and *B*) are wired to pins 1 and 2 of the 7408 IC. Note that, if a switch is in the *up position*, a logical 1 (+5 V) is applied to the input of the AND gate. At the right, a light-emitting diode (LED) and 150-ohm (Ω) limiting resistor are connected to ground. If the output at pin 3 is HIGH (+5 V), current will flow through the LED. Lighting the LED indicates a HIGH, or a binary 1, at the output of the AND gate.

The truth table in Fig. 3-23 shows the results of operating the 2-input AND circuit. The LED in Fig. 3-22*b* lights only when *both* input switches (*A* and *B*) are at +5 V.

Inputs		Output	
A	*B*		
Voltage	Voltage	Voltage	LED lights?
GND	GND	GND	no
GND	+5 V	GND	no
+5 V	GND	GND	no
+5 V	+5 V	near +5 V	yes

Fig. 3-23 Truth table for a TTL-type AND gate

Manufacturers of integrated circuits also produce other logic functions. Figure 3-24 illustrates pin diagrams for two basic TTL ICs. Figure 3-24*a* is the pin diagram for a *quadruple 2-input OR gate*. In other words, the 7432 IC contains four 2-input OR gates. It could be wired and tested in a manner similar to the testing of the AND gate shown in Fig. 3-22*b*.

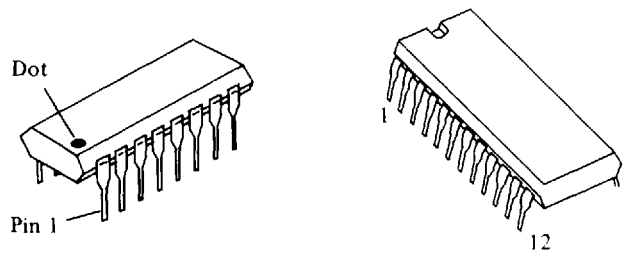


(*a*) Pin diagram for a 7432 IC (*b*) Pin diagram for a 7404 IC

Fig. 3-24

The 7404 IC shown in Fig. 3-24*b* is also a TTL device. The 7404 IC contains six NOT gates, or inverters. The 7404 is described by the manufacturer as a *hex inverter* IC. Note that each IC has its power connections (V_{CC} and GND). A 5-V dc power supply is always used with TTL logic circuits.

Two variations of DIP ICs are illustrated in Fig. 3-25. The integrated circuit shown in Fig. 3-25*a* has 16 pins with pin 1 identified by using a dot instead of a notch. The IC shown in Fig. 3-25*b* is a 24-pin DIP integrated circuit with pin 1 located immediately counterclockwise (when viewed from the top) from the notch.



(*a*) 16-pin DIP integrated circuit (*b*) 24-pin DIP integrated circuit

Fig. 3-25

The 7408, 7432, and 7404 ICs you studied in this section were all from the TTL logic family. The newer *complementary metal oxide semiconductor* (CMOS) family of ICs has been gaining popularity because of its low power requirements. Logic gates (AND, OR, and NOT) also are available in DIP IC form in the CMOS family. Typical DIP ICs might be the CMOS 74C08 quad 2-input AND gate, 74C04 hex inverter, or the 74C32 quad 2-input OR gate. The 74CXX series of CMOS gates is *not* directly compatible with the TTL 7400 series of integrated circuits.

SOLVED PROBLEMS

3.24 What logic function is performed by the circuit illustrated in Fig. 3-26?

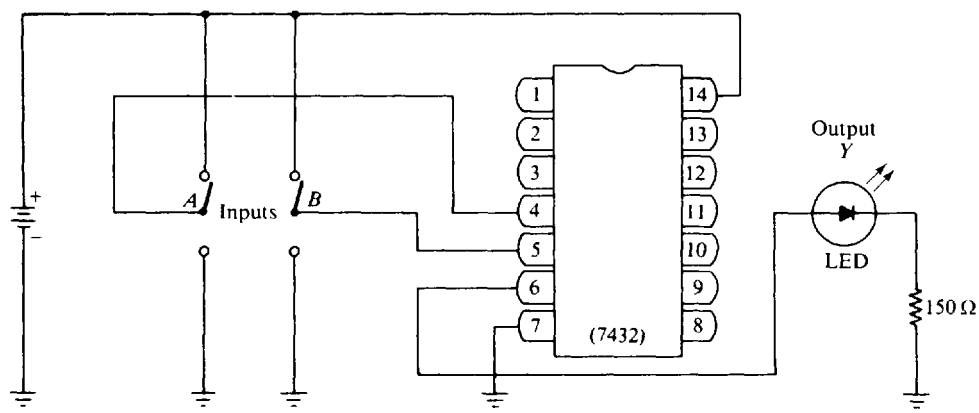


Fig. 3-26 Logic-circuit problem

Solution:

The 7432 IC performs as a 2-input OR gate when wired as shown in Fig. 3-26.

3.25 Write the Boolean expression for the circuit shown in Fig. 3-26.

Solution:

The Boolean expression for the 2-input OR function (Fig. 3-26) is $A + B = Y$.

3.26 What is the *voltage* of the power supply at the left in Fig. 3-26? The 7432 IC is a TTL device.

Solution:

TTL devices use a 5-V dc power supply.

3.27 If both switches A and B in Fig. 3-26 are in the *down position* (toward ground) the output LED will be _____ (lit, not lit).

Solution:

When both inputs are 0, the output of the OR gate will be 0 and the output LED will be not lit.

3.28 In Fig. 3-26, if switch A is up and switch B is down, the output LED will be _____ (lit, not lit).

Solution:

When input A is 1 and input B is 0 (Fig. 3-26), the output of the OR gate will be 1 and the output LED will be lit.

3.29 Pins 7 and 14 of the 7432 IC are _____ (input, output, power) connections.

Solution:

Pins 7 and 14 of the 7432 IC are power connections.

3.30 A _____ (+5 V, GND) voltage at pin 4 of the 7432 IC will cause pin 6 to go to a HIGH logic level.

Solution:

The output (pin 6) goes HIGH anytime as input (such as pin 4) is HIGH (near +5 V).

3.31 The letters TTL stand for _____.

Solution:

The letters TTL stand for the popular transistor-transistor logic family of integrated circuits.

3.32 The _____ (CMOS, TTL) logic family is characterized by its very low power consumption.

Solution:

The CMOS logic family is noted for its very low power consumption.

3.33 Integrated circuits from the TTL and CMOS families _____ (can, cannot) be interchanged in a digital circuit.

Solution:

TTL and CMOS ICs cannot be interchanged in a digital circuit. They may have the same logic function or even have the same pin diagram, but their input and output characteristics are quite different.

Supplementary Problems

3.34 Draw the logic symbol for a 6-input AND gate. Label the inputs A, B, C, D, E, and F. Label the output Y.

Ans. See Fig. 3-27.

3.35 Draw the logic symbol for a 7-input OR gate. Label the inputs A, B, C, D, E, F, and G. Label the output Y.

Ans. See Fig. 3-28.

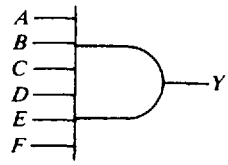


Fig. 3-27 A 6-input AND gate

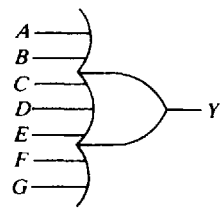


Fig. 3-28 A 7-input OR gate

3.36 Describe the pulse train at output Y of the AND gate shown in Fig. 3-29, if input B is 0.
Ans. A 0 will disable the AND gate, and the output will be 0.

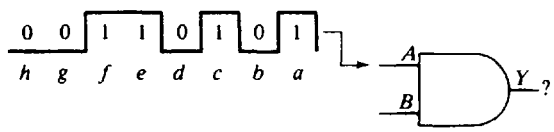


Fig. 3-29 Pulse-train problem

3.37 Describe the pulse train at output Y of the AND gate shown in Fig. 3-29 if input B is 1.
Ans. The output waveform will look exactly like the input waveform at input A (Fig. 3-29).

3.38 Describe the pulse train at output Y of the OR gate shown in Fig. 3-30 if input B is 0.
Ans. The output waveform will look exactly like the input waveform at input A (Fig. 3-30).

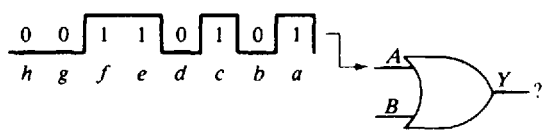


Fig. 3-30 Pulse-train problem

3.39 Describe the pulse train at output Y of the OR gate shown in Fig. 3-30 if input B is 1.
Ans. The output will always be 1.

3.40 Write the Boolean expression for the logic circuit shown in Fig. 3-31.
Ans. $A \cdot \overline{B} + \overline{B} \cdot C = Y$ or $A\overline{B} + \overline{B}C = Y$

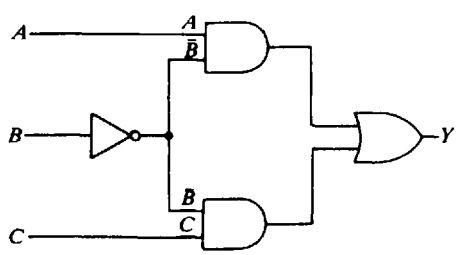


Fig. 3-31 AND-OR logic-circuit problem

3.41 Draw the truth table for the logic circuit shown in Fig. 3-31.
Ans.

Inputs			Output	Inputs			Output
C	B	A	Y	C	B	A	Y
0	0	0	0	1	0	0	1
0	0	1	1	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	0	1	1	1	0

3.42 Write the Boolean expression for the logic circuit shown in Fig. 3-32.
Ans. $\overline{A} \cdot B \cdot \overline{C} + \overline{B} \cdot \overline{C} = Y$ or $\overline{A}B\overline{C} + \overline{B}\overline{C} = Y$

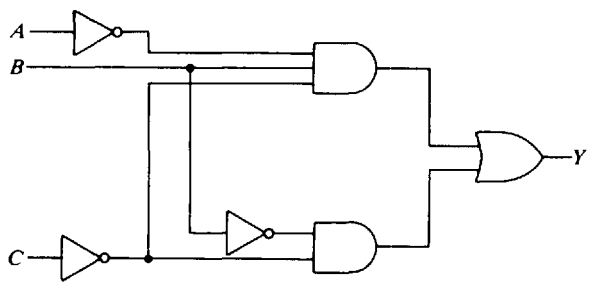


Fig. 3-32 AND-OR logic-circuit problem

3.43 Draw the truth table for the logic circuit shown in Fig. 3-32.
Ans.

Inputs			Output	Inputs			Output
C	B	A	Y	C	B	A	Y
0	0	0	1	1	0	0	0
0	0	1	1	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	1	1	1	0

3.44 Write the Boolean expression for the logic circuit shown in Fig. 3-33.
Ans. $\overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} = Y$ or $\overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} = Y$

3.45 Draw the truth table for the logic circuit shown in Fig. 3-33.
Ans.

Inputs			Output	Inputs			Output
C	B	A	Y	C	B	A	Y
0	0	0	0	1	0	0	1
0	0	1	1	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	1	1	1	0

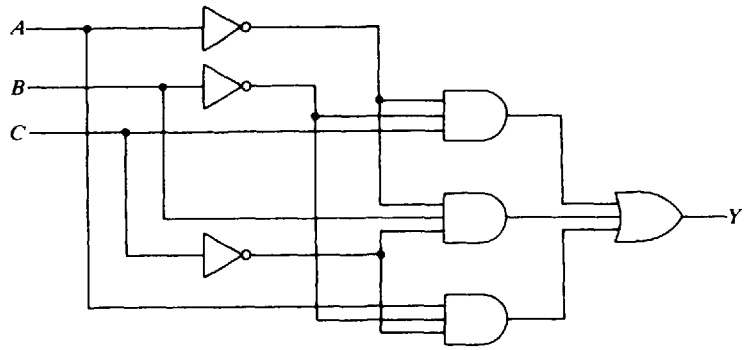


Fig. 3-33 AND-OR logic-circuit problem

3.46 Describe the pulse train at output Y of the AND gate shown in Fig. 3-34.

Ans.

pulse $a = 0$ pulse $c = 0$ pulse $e = 0$ pulse $g = 0$
pulse $b = 1$ pulse $d = 0$ pulse $f = 1$ pulse $h = 1$

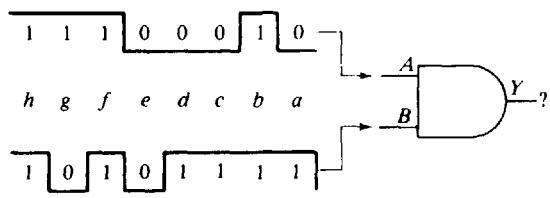


Fig. 3-34 Pulse-train problem

3.47 Describe the pulse train at output Y of the OR gate shown in Fig. 3-35.

Ans.

pulse $a = 0$ pulse $c = 1$ pulse $e = 1$ pulse $g = 1$
pulse $b = 1$ pulse $d = 1$ pulse $f = 1$ pulse $h = 0$

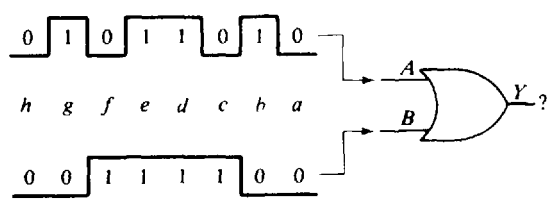


Fig. 3-35 Pulse-train problem

3.48 Write the Boolean expression for the logic circuit shown in Fig. 3-36.

Ans. $A \cdot B \cdot C \cdot D + \overline{A} \cdot \overline{C} = Y$ or $ABCD + \overline{A}\,\overline{C} = Y$

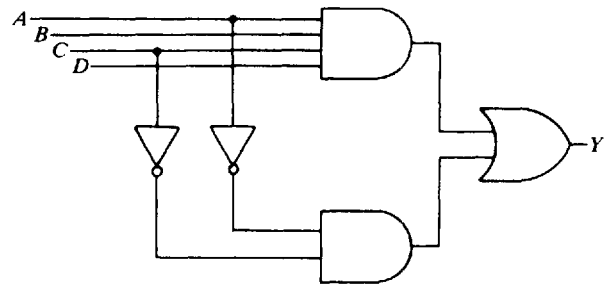


Fig. 3-36 AND-OR logic-circuit problem

3.49 Draw the truth table for the logic circuit shown in Fig. 3-36. Note that the circuit has four input variables. The truth table will have 16 possible combinations.
Ans.

Inputs				Output	Inputs				Output
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0	1	0
0	0	1	0	1	1	0	1	0	1
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	1

3.50 The part number 74C08 is a quad 2-input AND gate from the _____ (CMOS, TTL) family of ICs.
Ans. The 74C08 is a part number from the CMOS family of ICs. The C in the center of the part number means the part is a CMOS-type IC.

Chapter 4

Other Logic Gates

4-1 INTRODUCTION

The most complex digital systems, such as large computers, are constructed from basic logic gates. The AND, OR, and NOT gates are the most fundamental. Four other useful logic gates can be made from these fundamental devices. The other gates are called the NAND gate, the NOR gate, the exclusive-OR gate, and the exclusive-NOR gate. At the end of this chapter, you will know the logic symbol, truth table, and Boolean expression for each of the *seven logic gates* used in digital systems.

4-2 THE NAND GATE

Consider the logic diagram at the top of Fig. 4-1. An AND gate is connected to an inverter. Inputs A and B are ANDed to form the Boolean expression $A \cdot B$. The $A \cdot B$ is then inverted by the NOT gate. On the right side of the inverter, the overbar is added to the Boolean expression. The Boolean expression for the entire circuit is $\overline{A \cdot B} = Y$. It is said that this is a *not-AND* or NAND circuit.

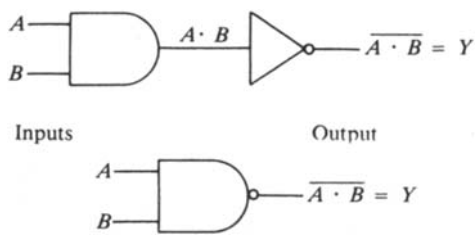


Fig. 4-1 The NAND gate

The standard logic symbol for the NAND gate is shown in the bottom diagram in Fig. 4-1. Note that the NAND symbol is an AND symbol with a small bubble at the output. The bubble is sometimes called an *invert bubble*. The invert bubble provides a simplified method of representing the NOT gate shown in the top diagram in Fig. 4-1.

The truth table describes the exact operation of a logic gate. The *truth table* for the NAND gate is illustrated in the unshaded columns of Fig. 4-2. The AND-gate truth table is also given to show how each output is inverted to give the NAND output. Some students find it useful to think of the NAND gate as an AND gate that puts out a 0 when it is enabled (when both inputs are 1).

Inputs		Output	
B	A	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Fig. 4-2 The AND- and NAND-gate truth tables

The NAND function has traditionally been the *universal gate* in digital circuits. The NAND gate is widely used in most digital systems.

Consider the NAND gate truth table in Fig. 4-2. The *unique output* from the NAND gate is a LOW when all inputs are HIGH.

SOLVED PROBLEMS

4.1 Write the Boolean expression for a 3-input NAND gate.

Solution:

$\overline{A \cdot B \cdot C} = Y$ or $\overline{ABC} = Y$

4.2 Draw the logic symbol for a 3-input NAND gate.

Solution:

See Fig. 4-3.

Fig. 4-3 A 3-input NAND gate

4.3 Draw the truth table for a 3-input NAND gate.

Solution:

Inputs			Output	Inputs			Output
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	1	1	0	0	1
0	0	1	1	1	0	1	1
0	1	0	1	1	1	0	1
0	1	1	1	1	1	1	0

4.4 What would the output pulse train shown in Fig. 4-4 look like if input *B* were 0?

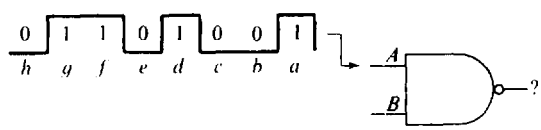


Fig. 4-4 Pulse-train problem

Solution:

The output of the NAND gate shown in Fig. 4-4 would always be 1.

4.5 What would the output pulse train shown in Fig. 4-4 look like if input B were 1?

Solution:

The output would be an inverted copy of the waveform at input A (Fig. 4-4). The output pulses would be as follows:

pulse $a = 0$ pulse $c = 1$ pulse $e = 1$ pulse $g = 0$
pulse $b = 1$ pulse $d = 0$ pulse $f = 0$ pulse $h = 1$

4.6 Draw a logic diagram of how you could connect a 2-input NAND gate to perform as an inverter. Label inverter input as A. Label inverter output as $\bar{A}$.

Solution:

See Fig. 4-5. There are two possibilities.

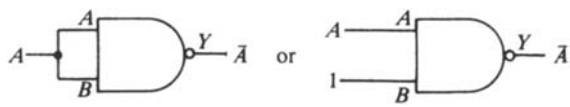


Fig. 4-5 Wiring the NAND gate as an inverter

4-3 THE NOR GATE

Consider the logic diagram in Fig. 4-6. An inverter has been connected to the output of an OR gate. The Boolean expression at the input to the inverter is $A + B$. The inverter then complements the ORed terms, which are shown in the Boolean expression with an overbar. Adding the overbar produces the Boolean expression $\overline{A + B} = Y$. This is a *not-OR* function. The not-OR function can be drawn as a single logic symbol called a *NOR gate*. The standard symbol for the NOR gate is illustrated in the bottom diagram of Fig. 4-6. Note that a small invert bubble has been added to the OR symbol to form the NOR symbol.

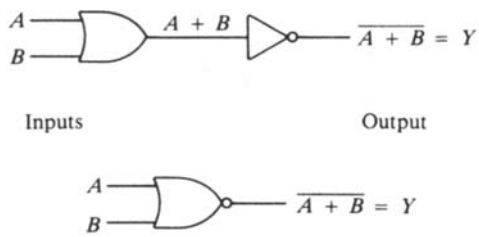


Fig. 4-6 The NOR gate

The truth table in Fig. 4-7 details the operation of the NOR gate. Note that the output column of the NOR gate is the complement (has been inverted) of the shaded OR column. In other words, the

Inputs		Output	
B	A	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Fig. 4-7 The OR- and NOR-gate truth tables

NOR gate puts out a 0 where the OR gate would produce a 1. The small invert bubble at the output of the NOR symbol serves as a reminder of the 0 output idea.

Consider the NOR gate truth table in Fig. 4-7. The *unique output* from the NOR gate is a HIGH when all inputs are LOW.

SOLVED PROBLEMS

4.7 Write the Boolean expression for a 3-input NOR gate.

Solution:

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

4.8 Draw the logic symbol for a 3-input NOR gate.

Solution:

See Fig. 4-8.

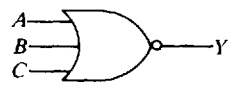


Fig. 4-8 A 3-input NOR gate

4.9 What is the truth table for a 3-input NOR gate?

Solution:

Inputs			Output	Inputs			Output
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0
0	1	1	0	1	1	1	0

4.10 What would the output pulse train shown in Fig. 4-9 look like if input *B* were 1?

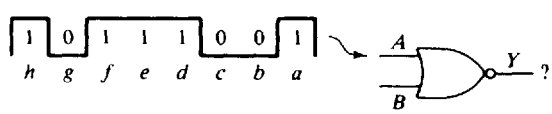


Fig. 4-9 Pulse-train problem

Solution:

The output of the NOR gate in Fig. 4-9 would always be 0.

4.11 What would the output pulse train shown in Fig. 4-9 look like if input B were 0?

Solution:

The output pulse would be the one shown in Fig. 4-9 but inverted. The pulses would be as follows:

pulse $a = 0$ pulse $c = 1$ pulse $e = 0$ pulse $g = 1$
pulse $b = 1$ pulse $d = 0$ pulse $f = 0$ pulse $h = 0$

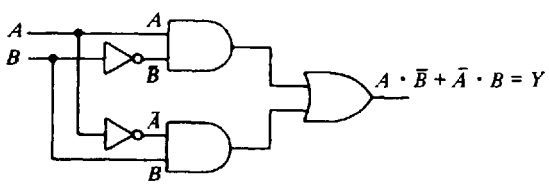
4-4 THE EXCLUSIVE-OR GATE

The *exclusive-OR gate* is referred to as the "any but not all" gate. The exclusive-OR term is often shortened to read as *XOR*. A truth table for the XOR function is shown in Fig. 4-10. Careful examination shows that this truth table is similar to the OR truth table except that, when both inputs are 1, the XOR gate generates a 0. The XOR gate is enabled *only when an odd number of 1s appear at the inputs*. Lines 2 and 3 of the truth table have odd numbers of 1s, and therefore the output is enabled with a 1. Lines 1 and 4 of the truth table contain even numbers (0, 2) of 1s, and therefore the XOR gate is disabled and a 0 appears at the output. The XOR gate could be referred to as an odd-bits check circuit.

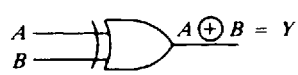
Inputs		Output
B	A	Y
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 4-10 The exclusive-OR-gate truth table

A Boolean expression for the XOR gate can be developed from the truth table in Fig. 4-10. The expression is $A \cdot \bar{B} + \bar{A} \cdot B = Y$. With this Boolean expression, a logic circuit can be developed by using AND gates, OR gates, and inverters. Such a logic circuit is drawn in Fig. 4-11*a*. This logic circuit will perform the XOR logic function.



(*a*) Logic circuit that performs the XOR function



(*b*) Standard logic symbol for the XOR gate

Fig. 4-11

The standard logic symbol for the XOR gate is shown in Fig. 4-11*b*. Both logic symbol diagrams in Fig. 4-11 would produce the same truth table (XOR). The Boolean expression at the right in Fig. 4-11*b* is a *simplified XOR expression*. The $\oplus$ symbol signifies the XOR function in Boolean algebra. It is said that inputs A and B in Fig. 4-11*b* are exclusively ORed together.

SOLVED PROBLEMS

4.12 Write the Boolean expression (simplified form) for a 3-input XOR gate.

Solution:

$A \oplus B \oplus C = Y$

4.13 Draw the logic symbol for a 3-input XOR gate.

Solution:

See Fig. 4-12.

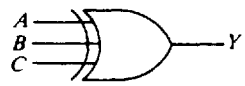


Fig. 4-12 A 3-input XOR gate

4.14 What is the truth table for a 3-input XOR gate? Remember that an odd number of 1s generates a 1 output.

Solution:

Inputs			Output	Inputs			Output
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	0	1	0	0	1
0	0	1	1	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	1	1	1	1

4.15 The XOR gate might be considered an _____ (even-, odd-) number-of-1s detector.

Solution:

The XOR gate generates a 1 when an *odd* number of 1 bits are present. For this reason it might be considered as an odd-number-of-1s detector.

4.16 What will the pulse train at the output of the XOR gate shown in Fig. 4-13 look like?

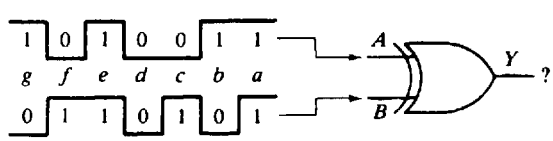


Fig. 4-13 Pulse-train problem

Solution:

The output pulses from the XOR gate shown in Fig. 4-13 will be as follows:

pulse $a = 0$ pulse $c = 1$ pulse $e = 0$ pulse $g = 1$
pulse $b = 1$ pulse $d = 0$ pulse $f = 1$

4-5 THE EXCLUSIVE-NOR GATE

The output of an XOR gate is shown inverted in Fig. 4-14. The output of the inverter on the right side is called the *exclusive-NOR* (XNOR) function. The XOR gate produces the expression $A \oplus B$. When this is inverted, it forms the Boolean expression for the XNOR gate, $\overline{A \oplus B} = Y$. The standard logic symbol for the XNOR gate is shown in the bottom diagram of Fig. 4-14. Note that the symbol is an XOR symbol with an invert bubble attached to the output.

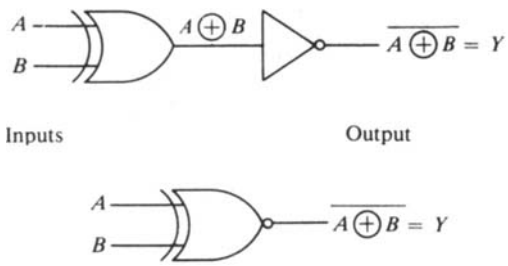


Fig. 4-14 The XNOR gate

The right-hand column of the truth table in Fig. 4-15 details the operation of the XNOR gate. Note that all outputs of the XNOR gate are the complements of the XOR-gate outputs. While the XOR gate is an odd-number-of-1s detector, the XNOR gate detects *even numbers of 1s*. The XNOR gate will produce a 1 output when an *even number of 1s* appear at the inputs.

Inputs		Output	
B	A	XOR	XNOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Fig. 4-15 The XOR- and XNOR-gate truth tables

SOLVED PROBLEMS

4.17 Write the Boolean expression for a 3-input XNOR gate.

Solution:

$\overline{A \oplus B \oplus C} = Y$

4.18 Draw the logic symbol for a 3-input XNOR gate.

Solution:

See Fig. 4-16.

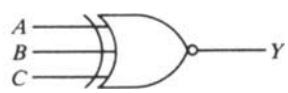


Fig. 4-16 A 3-input XNOR gate

4.19 Construct a truth table for a 3-input XNOR gate. Remember that an even number of 1s generates a 1 output.

Solution:

Inputs			Output	Inputs			Output
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	1
0	1	1	1	1	1	1	0

4.20 What will the pulse train at the output of the XNOR gate shown in Fig. 4-17 look like?

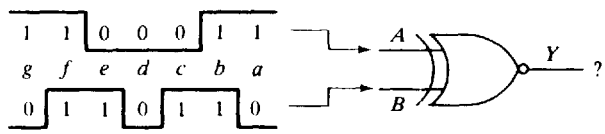


Fig. 4-17 Pulse-train problem

Solution:

The output pulses from the XNOR gate shown in Fig. 4-17 will be as follows:

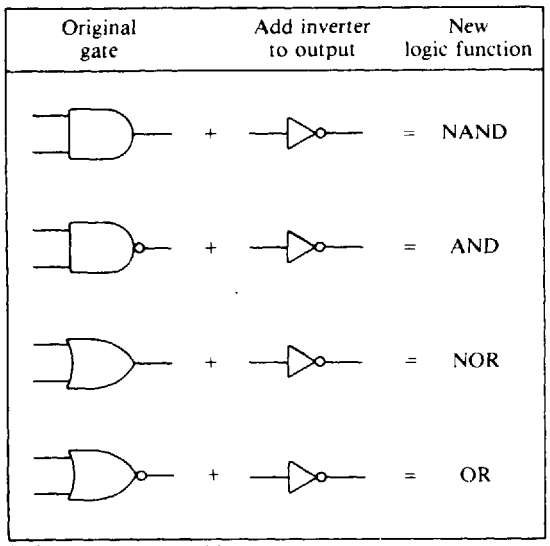
pulse $a = 0$ pulse $c = 0$ pulse $e = 0$ pulse $g = 0$
pulse $b = 1$ pulse $d = 1$ pulse $f = 1$

4-6 CONVERTING GATES WHEN USING INVERTERS

When using logic gates, the need will arise to convert to another logic function. An easy method of converting is to use inverters placed at the outputs or inputs of gates. It has been shown that an inverter connected at the output of an AND gate produces the NAND function. Also, an inverter placed at the output of an OR gate produces the NOR function. The chart in Fig. 4-18 illustrates these and other conversions.

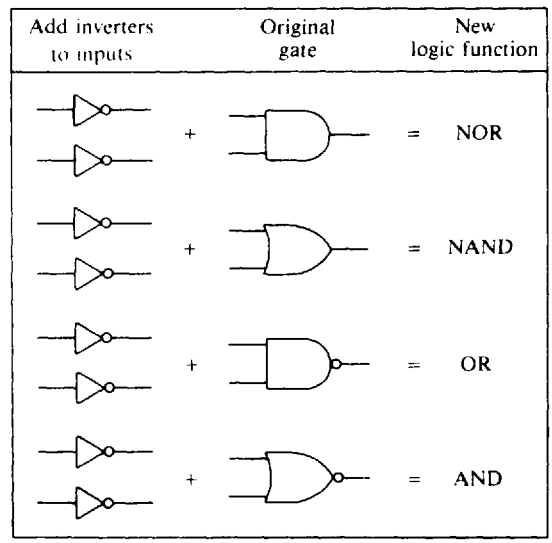
Placing inverters at all the inputs of a logic gate produces the results illustrated in Fig. 4-19. In the first line the inputs to an AND gate are being inverted (the plus symbol indicates addition in this figure). This produces the NOR function at the output of the AND gate. The second line of Fig. 4-19 shows the inputs of an OR gate being inverted. This produces the NAND function. The first two examples suggest new symbols for the NOR and NAND functions. Figure 4-20 illustrates two logic symbols sometimes used for the NOR and NAND functions. Figure 4-20*a* is an *alternative logic symbol* for a NOR gate. Figure 4-20*b* is an *alternative logic symbol* for a NAND gate. These symbols are encountered in some manufacturers' literature.

The effect of inverting both the inputs and output of a logic gate is shown in Fig. 4-21. Again the plus symbol stands for addition. This technique is probably not used often because of the large number of gates needed. Note that this is the method of converting from the AND to the OR to the AND function. This is also the method of converting from the NAND to the NOR to the NAND function.



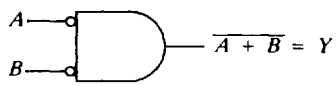
(+) symbol means adding on this chart

Fig. 4-18 Effect of inverting outputs of gates

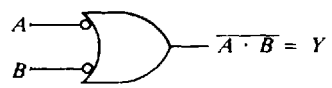


(+) symbol means adding on this chart

Fig. 4-19 Effect of inverting inputs of gates

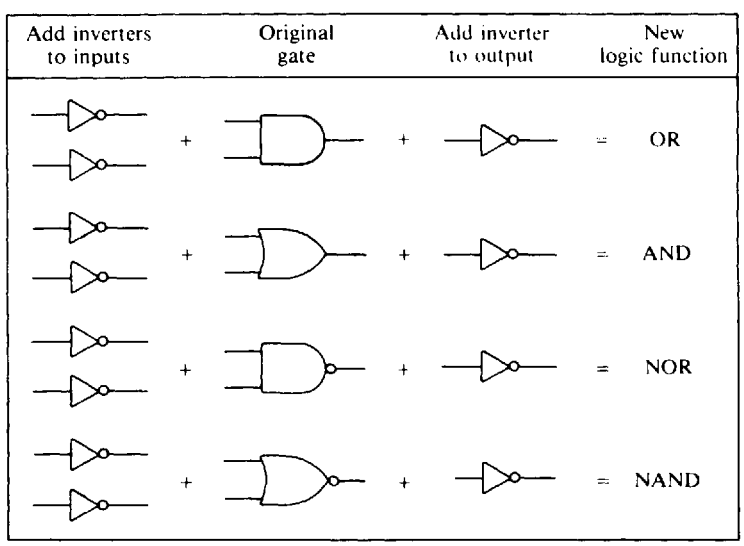


(*a*) NOR gate symbol



(*b*) NAND gate symbol

Fig. 4-20 Alternative logic symbols



(+) symbol means adding on this chart

Fig. 4-21 Effect of inverting both inputs and outputs of gates

SOLVED PROBLEMS

4.21 Given an OR gate and inverters, draw a logic diagram that will perform the 2-input NAND function.

Solution:

See Fig. 4-22.

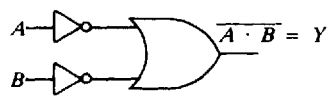


Fig. 4-22 2-input NAND function

4.22 Given an OR gate and inverters, draw a logic diagram that will perform the 3-input AND function.

Solution:

See Fig. 4-23.

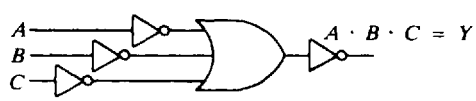


Fig. 4-23 3-input AND function

4.23 Given a NAND gate and inverters, draw a logic diagram that will perform the 2-input OR function.

Solution:

See Fig. 4-24.

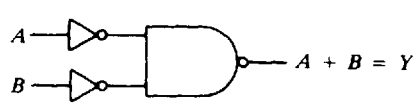


Fig. 4-24 2-input OR function

4.24 Given a NAND gate and inverters, draw a logic diagram that will perform the 3-input AND function.

Solution:

See Fig. 4-25.

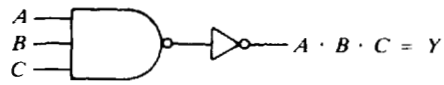


Fig. 4-25 3-input AND function

4.25 Given an AND gate and inverters, draw a logic diagram that will perform the 2-input NOR function.

Solution:

See Fig. 4-26.

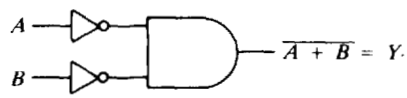
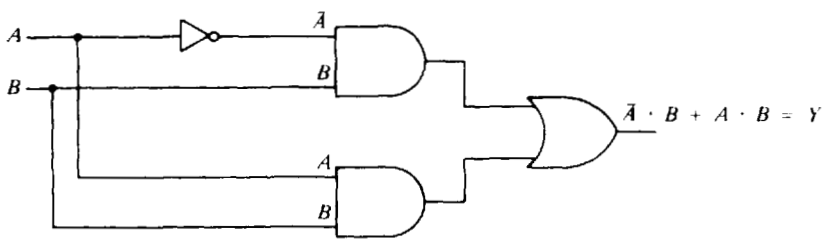


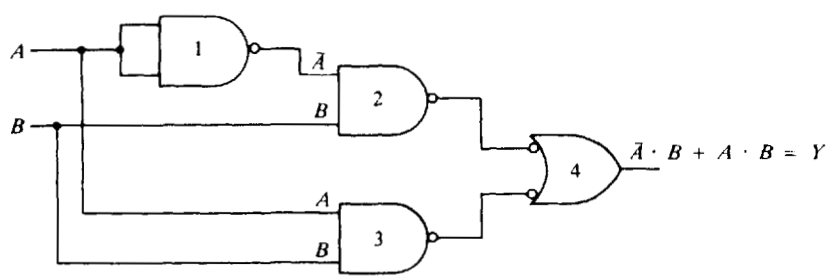
Fig. 4-26 2-input NOR function

4-7 NAND AS A UNIVERSAL GATE

Consider the logic circuit shown in Fig. 4-27*a*. This is referred to as an *AND-OR pattern of gates*. The AND gates feed into the final OR gate. The Boolean expression for this circuit is shown at the right as $\bar{A} \cdot B + A \cdot B = Y$. In constructing the circuit, you need three different types of gates (AND gates, an OR gate, and an inverter). From a manufacturer's catalog, you would find that three different ICs would be needed to implement the circuit shown in Fig. 4-27*a*.



(*a*) An AND-OR logic circuit



(*b*) An equivalent NAND logic circuit

Fig. 4-27

It was mentioned earlier that the NAND gate is considered a universal gate. Figure 4-27*b* shows NAND gates being used to implement the logic $\bar{A} \cdot B + A \cdot \bar{B} = Y$. This logic is the same as that performed by the AND-OR circuit shown in Fig. 4-27*a*. Remember that the OR-looking gate (gate 4) with the invert bubbles at the inputs is a NAND gate. The circuit shown in Fig. 4-27*b* is simpler because all the gates are NAND gates. It is found that only one IC (a quadruple 2-input NAND gate) is needed to implement the NAND logic of Fig. 4-27*b*. Fewer ICs are needed to implement the NAND logic circuit than the AND-OR pattern of logic gates.

It is customary when converting from AND-OR logic to NAND logic to draw the AND-OR pattern first. This can be done from the Boolean expression. The AND-OR logic diagram would be similar to that in Fig. 4-27*a*. A NAND gate is then substituted for each inverter, AND gate, and OR gate. The NAND logic pattern will then be similar to the circuit shown in Fig. 4-27*b*.

A clue to *why* the AND-OR logic can be replaced by NAND logic is shown in Fig. 4-27*b*. Note the two invert bubbles between the output of gate 2 and the input of gate 4. *Two invert bubbles cancel one another*. That leaves the AND-OR symbols just as in Fig. 4-27*a*. The *double inversion* also takes place in Fig. 4-27*b* between gates 3 and 4. This leaves AND gate 3 feeding OR gate 4. NAND gate 1 acts as an inverter when its inputs are tied together as shown in Fig. 4-27*b*.

SOLVED PROBLEMS

4.26 Redraw the AND-OR circuit shown in Fig. 4-11*a* by using five 2-input NAND gates. The NAND logic circuit should perform the logic $A \cdot \bar{B} + \bar{A} \cdot B = Y$.

Solution:

See Fig. 4-28.

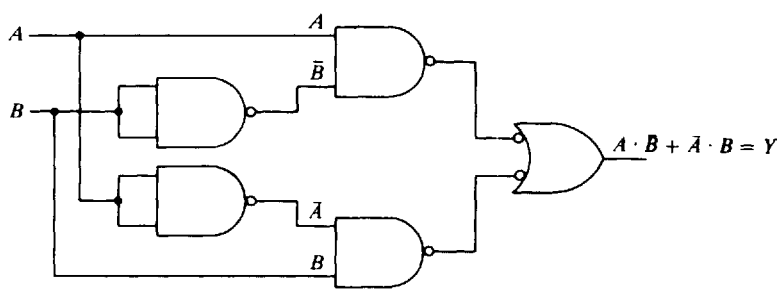


Fig. 4-28 Solution using NAND logic

4.27 Draw a logic diagram for the Boolean expression $\bar{A} \cdot \bar{B} + A \cdot B = Y$. Use inverters, AND gates, and OR gates.

Solution:

See Fig. 4-29.

4.28 Redraw the logic diagram of Prob. 4.27 by using only five 2-input NAND gates. The circuit should perform the logic $\bar{A} \cdot \bar{B} + A \cdot B = Y$.

Solution:

See Fig. 4-30.

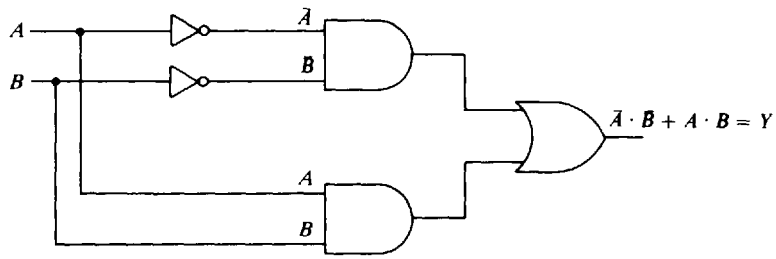


Fig. 4-29 AND-OR logic circuit

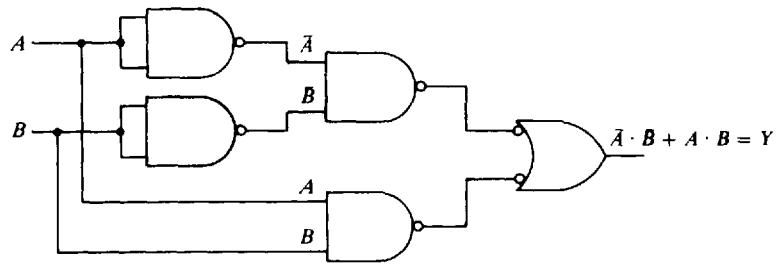
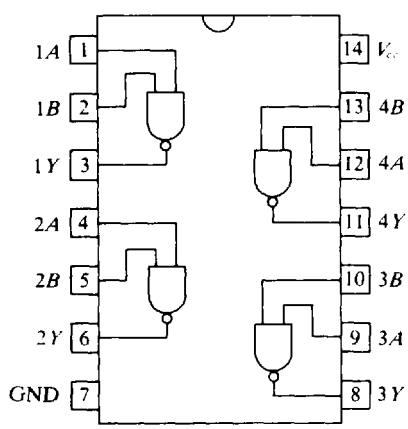


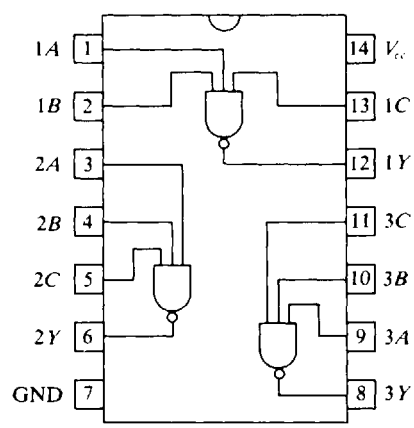
Fig. 4-30 Equivalent NAND logic circuit

4-8 USING PRACTICAL LOGIC GATES

The most useful logic gates are packaged as integrated circuits. Figure 4-31 illustrates two TTL logic gates that can be purchased in IC form. A pin diagram of the 7400 IC is shown in Fig. 4-31*a*. The 7400 is described by the manufacturer as a *quadruple 2-input NAND gate* IC. Note that the 7400 IC does have the customary power connections (V_{CC} and GND). All other pins are the inputs and outputs from the four 2-input NAND gates.



(*a*) Pin diagram for a 7400 IC



(*b*) Pin diagram for a 7410 IC

Fig. 4-31

Three 3-input NAND gates are housed in the 7410 TTL IC. The pin diagram for the 7410 IC is shown in Fig. 4-31*b*. This device is described by the manufacturer as a *triple 3-input NAND gate* IC. NAND gates with more than three inputs also are available.

The 7400 and 7410 ICs were from the common TTL logic family. Manufacturers also produce a variety of NAND, NOR, and XOR gates in CMOS-type ICs. Typical NAND gates might be the CMOS 74C00 quad 2-input NAND gate, 74C30 8-input NAND gate, and 4012 dual 4-input NAND gate DIP ICs. Some CMOS NOR gates in DIP IC form are the 74C02 quad 2-input NOR gate and the 4002 dual 4-input NOR gate. Several exclusive-OR gates are produced in CMOS; examples are the 74C86 quad 2-input XOR gate and the 4030 quad 2-input XOR gate. Note that CMOS ICs come in both a 74C00 series and a 4000 series. It must be remembered that without special interfacing, *TTL and CMOS ICs are not compatible.*

SOLVED PROBLEMS

4.29 Construct the truth table for the circuit shown in Fig. 4-32.

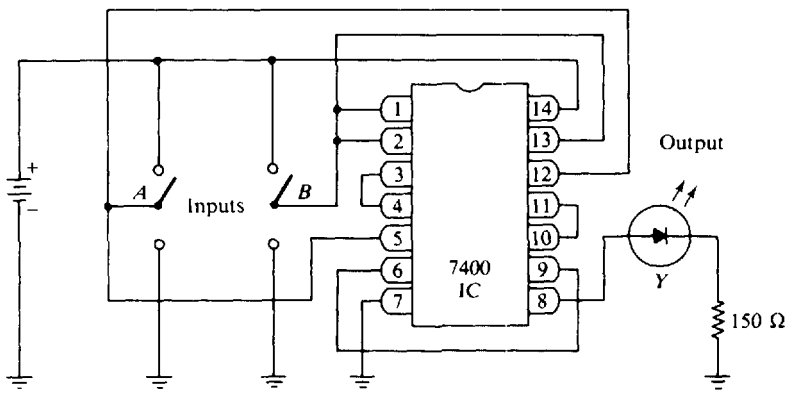


Fig. 4-32 Wiring diagram of a logic-circuit problem

Solution:

Inputs		Output
B	*A*	*Y*
0	0	0
0	1	1
1	0	0
1	1	1

4.30 What is the voltage of the power supply at the left in Fig. 4-32? The 7400 IC is a TTL device.

Solution:

A TTL device uses a 5-V dc power supply.

4.31 If both switches (*A* and *B*) shown in Fig. 4-32 are in the *up* position (at logical 1), the output LED will be _____ (lit, not lit).

Solution:

When both inputs are 1, the output of the circuit will be 1 and the output LED will be lit.

4.32 The 7400 IC is described by the manufacturer as a quadruple _____ _____ _____.

Solution:

The 7400 IC is a quadruple 2-input NAND gate.

4.33 The circuit shown in Fig. 4-32 could be described as a(n) _____ (AND-OR, NAND) logic circuit.

Solution:

The circuit shown in Fig. 4-32 uses NAND logic.

4.34 The 4012 is a dual 4-input NAND gate IC using the _____ (CMOS, TTL) technology.

Solution:

The 4000 series part numbers designate CMOS digital ICs.

Supplementary Problems

4.35 Write the Boolean expression for a 4-input NAND gate.
Ans. $\overline{A \cdot B \cdot C \cdot D} = Y$ or $\overline{ABCD} = Y$

4.36 Draw the logic symbol for a 4-input NAND gate. *Ans.* See Fig. 4-33.

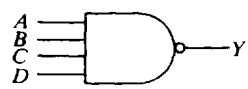


Fig. 4-33 A 4-input NAND gate

4.37 Construct the truth table for a 4-input NAND gate.
Ans.

Inputs				Output	Inputs				Output
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	1	1	0	0	0	1
0	0	0	1	1	1	0	0	1	1
0	0	1	0	1	1	0	1	0	1
0	0	1	1	1	1	0	1	1	1
0	1	0	0	1	1	1	0	0	1
0	1	0	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0

4.38 What would the output pulse train shown in Fig. 4-34 look like if input *C* were 0?
Ans. The output of the NAND gate would be 1 at all times.

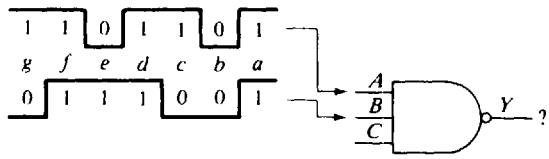


Fig. 4-34 Pulse-train problem

4.39 What would the output pulse train shown in Fig. 4-34 look like if input C were 1?
Ans. pulse $a = 0$ pulse $c = 1$ pulse $e = 1$ pulse $g = 1$
pulse $b = 1$ pulse $d = 0$ pulse $f = 0$

4.40 Write the Boolean expression for a 4-input NOR gate. *Ans.* $\overline{A + B + C + D} = Y$

4.41 Draw the logic symbol for a 4-input NOR gate. *Ans.* See Fig. 4-35.

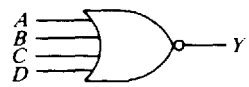


Fig. 4-35 A 4-input NOR gate

4.42 Construct the truth table for a 4-input NOR gate.
Ans.

Inputs				Output	Inputs				Output
D	C	B	A	Y	D	C	B	A	Y
0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	1	0

4.43 What would the output pulse train shown in Fig. 4-36 look like if input C were 1?
Ans. The output of the NOR gate would always be 0.

4.44 What would the output pulse train shown in Fig. 4-36 look like if input C were 0?
Ans. pulse $a = 0$ pulse $c = 1$ pulse $e = 0$ pulse $g = 1$
pulse $b = 1$ pulse $d = 0$ pulse $f = 0$

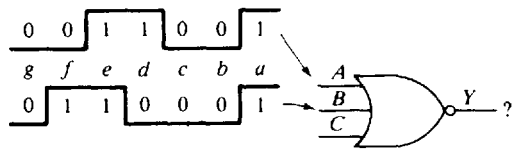


Fig. 4-36 Pulse-train problem

4.45 Write the Boolean expression for a 4-input XOR gate.
Ans. $A \oplus B \oplus C \oplus D = Y$

4.46 Draw the logic symbol for a 4-input XOR gate. *Ans.* See Fig. 4-37.

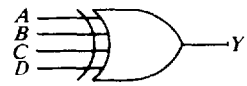


Fig. 4-37 A 4-input XOR gate

4.47 Construct the truth table for a 4-input XOR gate.
Ans.

Inputs				Output	Inputs				Output
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	0	1	0	0	0	1
0	0	0	1	1	1	0	0	1	0
0	0	1	0	1	1	0	1	0	0
0	0	1	1	0	1	0	1	1	1
0	1	0	0	1	1	1	0	0	0
0	1	0	1	0	1	1	0	1	1
0	1	1	0	0	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0

4.48 What would the pulse train at the output of the XOR gate shown in Fig. 4-38 look like?
Ans. pulse $a = 0$ pulse $c = 1$ pulse $e = 0$ pulse $g = 0$
pulse $b = 1$ pulse $d = 1$ pulse $f = 0$ pulse $h = 1$

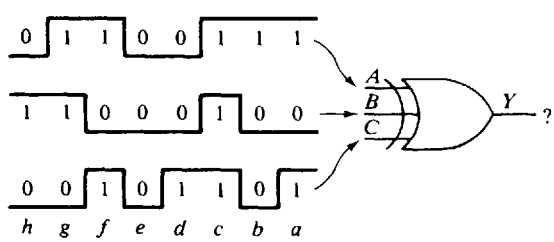


Fig. 4-38 Pulse-train problem

4.49 Write the Boolean expression for a 4-input XNOR gate.
Ans. $\overline{A \oplus B \oplus C \oplus D} = Y$

4.50 Draw the logic symbol for a 4-input XNOR gate. *Ans.* See Fig. 4-39.

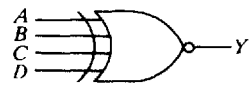


Fig. 4-39 A 4-input XNOR gate

4.51 Construct a truth table for a 4-input XNOR gate.
Ans.

Inputs				Output	Inputs				Output
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	0	1	0	1
0	0	1	1	1	1	0	1	1	0
0	1	0	0	0	1	1	0	0	1
0	1	0	1	1	1	1	0	1	0
0	1	1	0	1	1	1	1	0	0
0	1	1	1	0	1	1	1	1	1

4.52 What would the pulse train at the output of the XNOR gate shown in Fig. 4-40 look like?
Ans. pulse $a = 1$ pulse $c = 0$ pulse $e = 0$ pulse $g = 1$
pulse $b = 0$ pulse $d = 1$ pulse $f = 1$ pulse $h = 0$

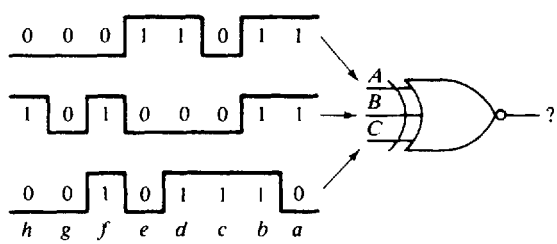


Fig. 4-40 Pulse-train problem

4.53 Given an OR gate and inverters, draw a logic diagram that will perform the 3-input NAND function.
Ans. See Fig. 4-41.

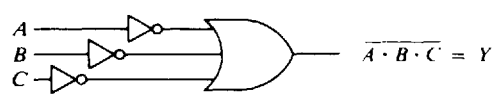


Fig. 4-41 3-input NAND function

4.54 Given a NOR gate and inverters, draw a logic diagram that will perform the 3-input AND function.
Ans. See Fig. 4-42.

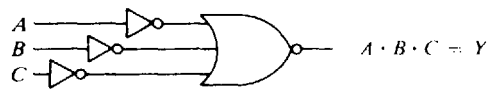


Fig. 4-42 3-input AND function

4.55 Given a NOR gate and inverters, draw a logic diagram that will perform the 5-input OR function.
Ans. See Fig. 4-43.

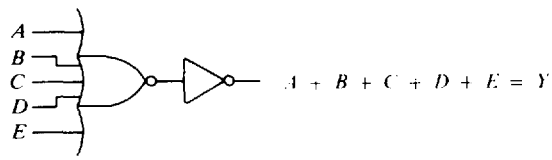


Fig. 4-43 5-input OR function

4.56 Draw a logic diagram for the Boolean expression $\bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot C + A \cdot B \cdot \bar{C} = Y$. Use inverters, AND gates, and an OR gate. *Ans.* See Fig. 4-44.

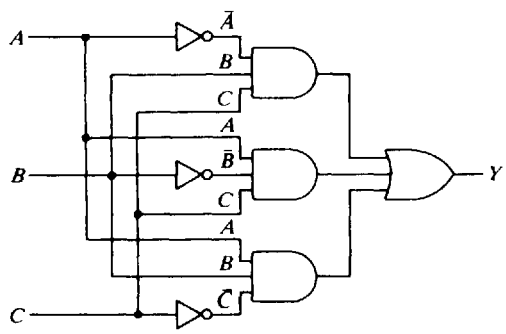


Fig. 4-44 An AND-OR logic circuit

4.57 Redraw the logic diagram for Prob. 4.56 by using three 2-input NAND gates and four 3-input NAND gates. *Ans.* See Fig. 4-45.

4.58 Write the Boolean expression for the circuit shown in Fig. 4-46.
Ans. $A \cdot B + \bar{A} \cdot \bar{B} \cdot C = Y$

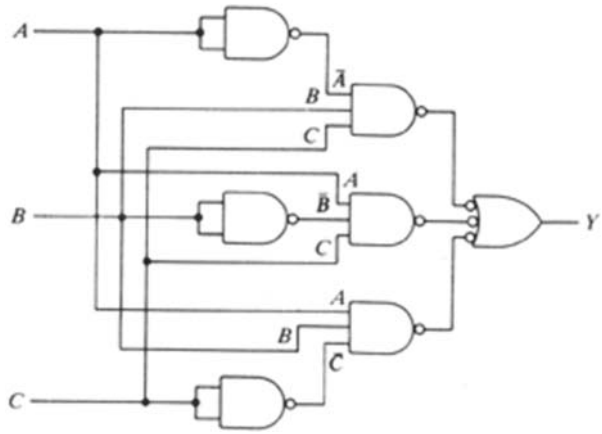


Fig. 4-45 An equivalent NAND logic circuit

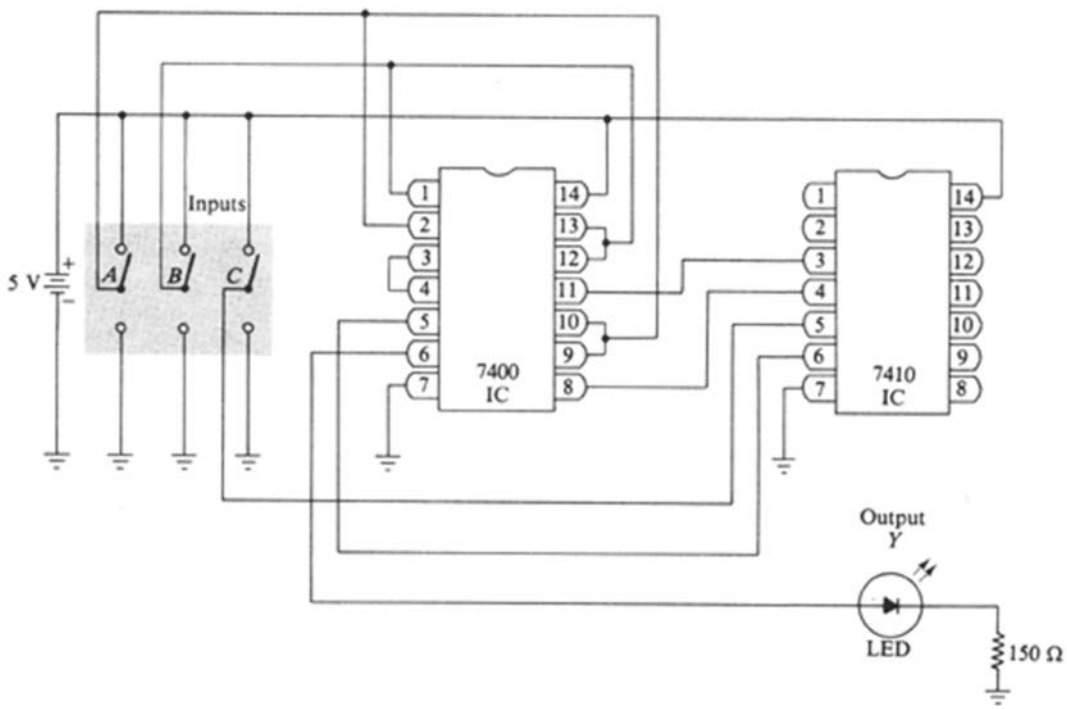


Fig. 4-46 Wiring diagram of a logic-circuit problem

4.59 Construct the truth table for the circuit shown in Fig. 4-46.

Ans

Inputs			Output	Inputs			Output
C	B	A	Y	C	B	A	Y
0	0	0	0	1	0	0	1
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	1

4.60 If switches A, B, and C shown in Fig. 4-46 are in the *up* position (logical 1), the output LED will be _____ (lit, not lit).
Ans. When all inputs are 1, the output of the circuit will be 1 according to the truth table and the output LED will be lit.

4.61 The unique output for the _____ logic gate is a LOW when all inputs are HIGH.
Ans. NAND

4.62 The unique output for the _____ logic gate is a HIGH when all inputs are LOW.
Ans. NOR

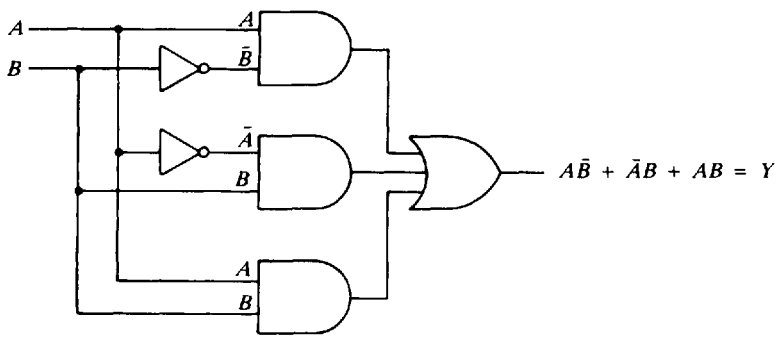
4.63 The _____ logic gate generates a HIGH output when an *odd number* of inputs are HIGH.
Ans. exclusive-OR or XOR.

Chapter 5

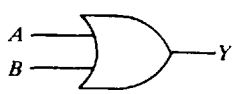
Simplifying Logic Circuits: Mapping

5-1 INTRODUCTION

Consider the Boolean expression $A \cdot \bar{B} + \bar{A} \cdot B + A \cdot B = Y$, a logic diagram for which is in Fig. 5-1*a*. Note that six gates must be used to implement this logic circuit, which performs the logic detailed in the truth table (Fig. 5-1*c*). From examination of the truth table, it is determined that a *single 2-input OR gate* will perform the function. It is found that the OR gate shown in Fig. 5-1*b* will be the simplest method of performing this logic. The logic circuits in Fig. 5-1*a* and *b* perform exactly the same logic function. Obviously a designer would choose the simplest, least expensive circuit, shown in Fig. 5-1*b*. It has been shown that the unsimplified Boolean expression ($A \cdot \bar{B} + \bar{A} \cdot B + A \cdot B = Y$) could be simplified to $A + B = Y$. The simplification was done by simple examination of the truth table and recognizing the OR pattern. Many Boolean expressions can be greatly simplified. Several systematic methods of simplification will be examined in this chapter.



(*a*) Unsimplified logic circuit



(*b*) Simplified logic circuit

Inputs		Output
B	*A*	*Y*
0	0	0
0	1	1
1	0	1
1	1	1

(*c*) Truth table for OR function

Fig. 5-1

In this chapter simple logic gates will be used to implement combinational logic. Other techniques also are commonly used for simplifying more complex logic problems. They include the use of *data selectors* (multiplexers), *decoders*, *PLAs* (programmable logic arrays), *ROMs* (read-only memories) and *PROMs* (programmable read-only memories).

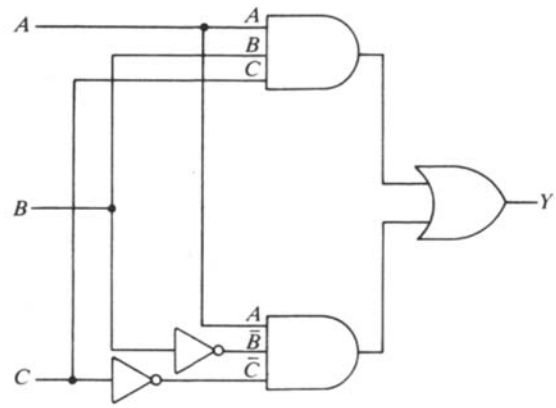
5-2 SUM-OF-PRODUCTS BOOLEAN EXPRESSIONS

It is customary when starting a logic-design problem to first construct a truth table. The table details the exact operation of the digital circuit. Consider the truth table in Fig. 5-2*a*. It contains the three variables *C*, *B*, and *A*. Note that only two combinations of variables will generate a 1 output. These combinations are shown in the shaded second and eighth lines of the truth table. From line 2, we say that "a not *C* AND a not *B* AND an *A* input will generate a 1 output." This is shown near the right side of line 2 as the Boolean expression $\bar{C} \cdot \bar{B} \cdot A$. The other combination of variables that will generate a 1 is shown in line 8 of the truth table. Line 8 reads as "a *C* AND a *B* AND an *A* input will generate a 1 output." The Boolean expression for line 8 is shown at the right as $C \cdot B \cdot A$. These two

(a)

Inputs			Output	
C	B	A	Y	
0	0	0	0	
0	0	1	1	$\bar{C} \cdot \bar{B} \cdot A$
0	1	0	0	
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	$C \cdot B \cdot A$

(b) Boolean expression: $C \cdot B \cdot A + \bar{C} \cdot \bar{B} \cdot A = Y$



(c) Equivalent AND-OR logic circuit

Fig. 5-2

possible combinations are then ORed together to form the complete Boolean expression for the truth table. The complete Boolean expression is shown in Fig. 5-2*b* as $C \cdot B \cdot A + \bar{C} \cdot \bar{B} \cdot A = Y$. The $C \cdot B \cdot A + \bar{C} \cdot \bar{B} \cdot A = Y$ in Fig. 5-2*b* is sometimes called a *sum-of-products* form of a Boolean expression. Engineers also call this form of expression the *minterm form*. Note that this expression can be translated into a familiar AND-OR pattern of logic gates. The logic diagram in Fig. 5-2*c* performs the logic described by the minterm Boolean expression $C \cdot B \cdot A + \bar{C} \cdot \bar{B} \cdot A = Y$ and will generate the truth table in Fig. 5-2*a*.

It is typical procedure in logic-design work to *first* construct a truth table. *Second*, a minterm Boolean expression is then determined from the truth table. *Finally*, the AND-OR logic circuit is drawn from the minterm Boolean expression. This procedure is outlined in the sample problem in Fig. 5-2.

SOLVED PROBLEMS

5.1 Write a *minterm* Boolean expression for the truth table in Fig. 5-3.

Inputs			Output	Inputs			Output
C	B	A	Y	C	B	A	Y
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	1	0	1
0	1	1	1	1	1	1	0

Fig. 5-3

Solution:

$\bar{C} \cdot B \cdot A + C \cdot B \cdot \bar{A} = Y$ or $\bar{C}BA + CB\bar{A} = Y$

5.2 The Boolean expression developed in Prob. 5.1 was a _____ (maxterm, minterm) expression. This type of expression is also called the _____ (product-of-sums, sum-of-products) form.

Solution:

This type of Boolean expression ($\bar{C} \cdot B \cdot A + C \cdot B \cdot \bar{A} = Y$) is called the minterm form or the sum-of-products form.

5.3 Diagram a logic circuit that will perform the logic in the truth table in Fig. 5-3.

Solution:

See Fig. 5-4.

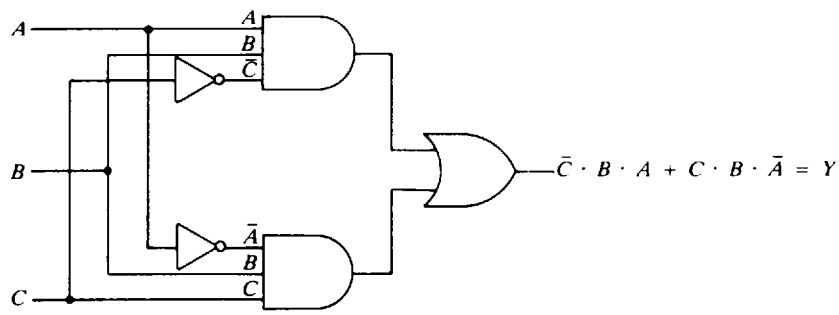


Fig. 5-4 Logic-diagram solution

5.4 Write a sum-of-products Boolean expression for the truth table in Fig. 5-5.

Solution:

$\bar{C} \cdot \bar{B} \cdot \bar{A} + \bar{C} \cdot B \cdot A + C \cdot \bar{B} \cdot A = Y$

Inputs			Output	Inputs			Output
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	0	1	1	0	0
0	1	1	1	1	1	1	0

Fig. 5-5

5.5 Diagram a logic circuit that will perform the logic in the truth table in Fig. 5-5.

Solution:

See Fig. 5-6.

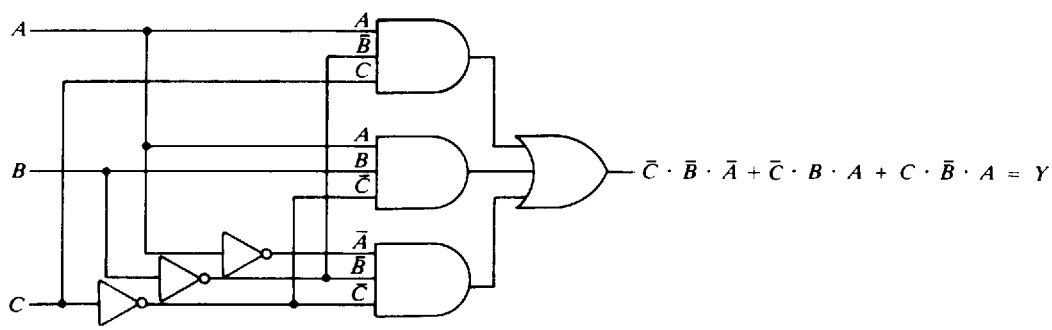


Fig. 5-6 Logic-diagram solution

5-3 PRODUCT-OF-SUMS BOOLEAN EXPRESSIONS

Consider the OR truth table in Fig. 5-7*b*. The Boolean expression for this truth table can be written in two forms, as was observed in the introductory section. The minterm Boolean expression is developed from the output 1s in the truth table. Each 1 in the output column becomes a term to be ORed in the minterm expression. The minterm expression for this truth table is given in Fig. 5-7*c* as

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

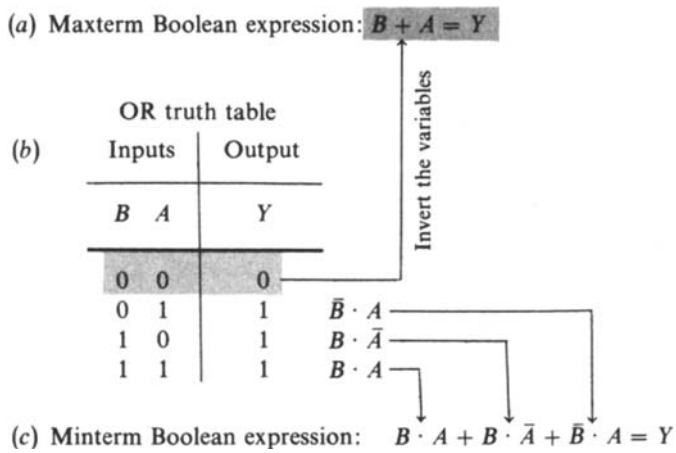


Fig. 5-7

The truth table in Fig. 5-7 can also be described by using a *maxterm form* of Boolean expression. This type of expression is developed from the 0s in the output column of the truth table. For each 0 in the output column, an ORed term is developed. Note that the *input variables are inverted and then ORed*. The maxterm Boolean expression for this truth table is given in Fig. 5-7*a*. The maxterm expression for the OR truth table is shown as $B + A = Y$. This means the same thing as the familiar OR expression $A + B = Y$. For the truth table in Fig. 5-7, the maxterm Boolean expression turns out to be the simplest. Both the minterm and maxterm expressions accurately describe the logic of the truth table in Fig. 5-7.

Consider the truth table in Fig. 5-8*a*. The minterm expression for this truth table would be rather long. The *maxterm* Boolean expression is developed from the variables in lines 5 and 8. Each of these

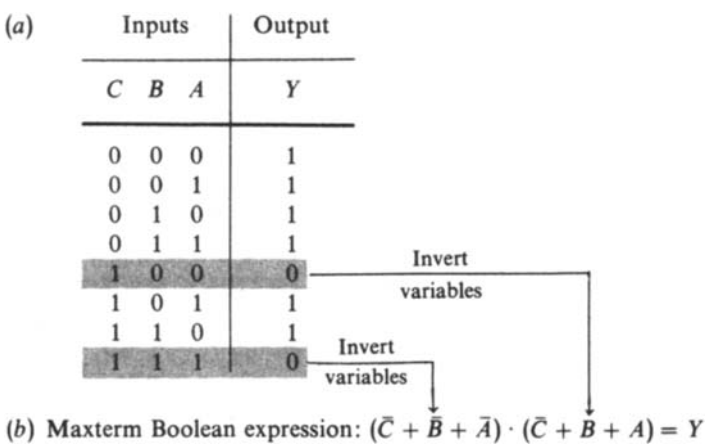


Fig. 5-8 Developing a maxterm expression

lines has a 0 in the output column. The variables are inverted and ORed with parentheses around them. The terms are then ANDed. The complete maxterm Boolean expression is given in Fig. 5-8*b*. The maxterm expression is also called the *product-of-sums form* of a Boolean expression. The product-of-sums term comes from the arrangement of the sum (+) and product (·) symbols.

A maxterm Boolean expression would be implemented by using an OR-AND pattern of logic gates illustrated in Fig. 5-9. Note that the outputs of the two OR gates are feeding into an AND gate. The maxterm expression $(\overline{C}+\overline{B}+\overline{A})\cdot(\overline{C}+B+A)=Y$ is implemented by using the OR-AND pattern of gates in Fig. 5-9.

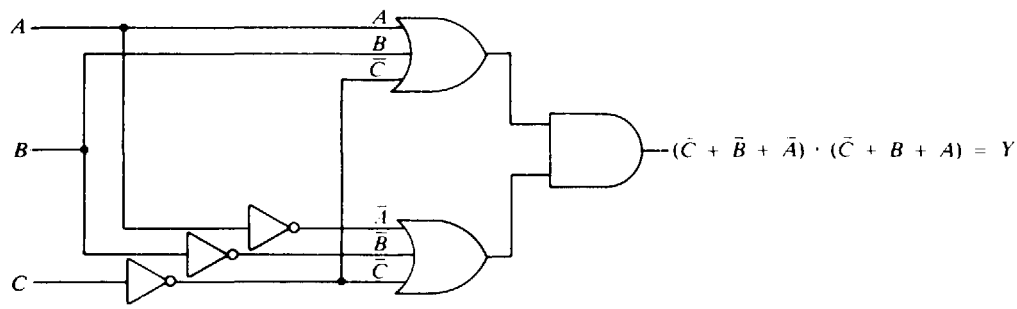


Fig. 5-9 Maxterm expression implemented with OR-AND circuit

SOLVED PROBLEMS

5.6 Write a *maxterm* Boolean expression for the truth table in Fig. 5-10.

Inputs			Output	Inputs			Output
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	1	1	0	0	1
0	0	1	0	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	1	1	1	1	1

Fig. 5-10

Solution:

$(C+B+\overline{A})\cdot(\overline{C}+\overline{B}+A)=Y$

5.7 The Boolean expression developed in Prob. 5.6 is a _____ (maxterm, minterm) expression. This type of expression is also called the _____ (product-of-sums, sum-of-products) form.

Solution:

The type of Boolean expression developed in Prob. 5-6 is called the maxterm form or the product-of-sums form.

5.8 Diagram a logic circuit that will perform the logic in the truth table in Fig. 5-10.

Solution:

See Fig. 5-11.

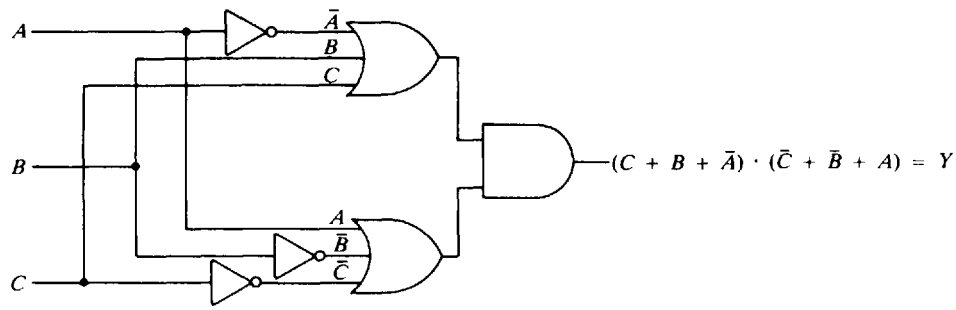


Fig. 5-11 Maxterm expression implemented with OR-AND circuit

5.9 The logic diagram of Prob. 5.8 is called the _____ (AND-OR, OR-AND) pattern of logic gates.

Solution:

The pattern of gates shown in Fig. 5-11 is called the OR-AND pattern.

5.10 Write the product-of-sums Boolean expression for the truth table in Fig. 5-12.

Inputs			Output	Inputs			Output
C	*B*	*A*	*Y*	*C*	*B*	*A*	*Y*
0	0	0	0	1	0	0	1
0	0	1	1	1	0	1	0
0	1	0	1	1	1	0	1
0	1	1	0	1	1	1	1

Fig. 5-12

Solution:

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

5.11 Diagram a logic circuit that will perform the logic in the truth table in Fig. 5-12.

Solution:

See Fig. 5-13.

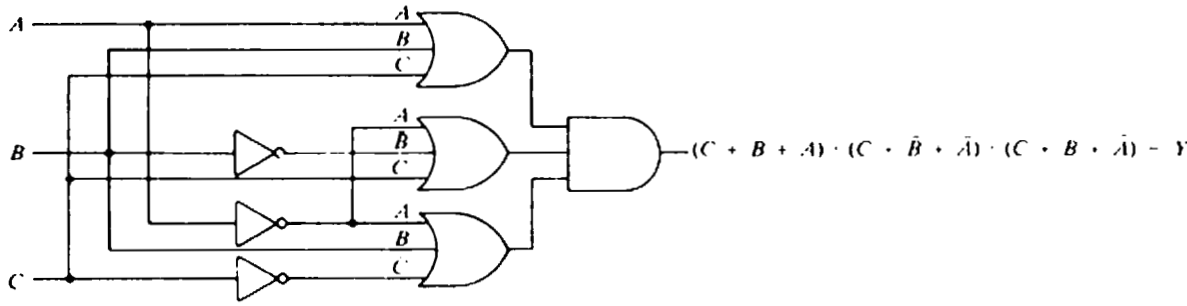


Fig. 5-13 Maxterm expression implemented with OR-AND circuit

5-4 USING DE MORGAN'S THEOREMS

Boolean algebra, the algebra of logic circuits, has many laws or theorems. *De Morgan's theorems* are very useful. They allow for easy transfer back and forth from the minterm to the maxterm form, of Boolean expression. They also allow for elimination of overbars that are over several variables.

De Morgan's theorems can be stated as follows:

$$\text{first theorem } \overline{A + B} = \bar{A} \cdot \bar{B} \qquad \text{second theorem } \overline{A \cdot B} = \bar{A} + \bar{B}$$

The first theorem changes the basic OR situation to an AND situation. A practical example of the first theorem is illustrated in Fig. 5-14*a*. The NOR gate on the left is equal in function to the AND gate (with inverted inputs) on the right. Note that the conversion is from the basic OR situation to the basic AND situation as shown by the gates in Fig. 5-14(*a*). This conversion is useful in getting rid of the long overbar on the NOR and can be used in converting from a minterm to a maxterm expression. The AND-looking symbol at the right in Fig. 5-14*a* would produce a NOR truth table.

Fig. 5-14 Applications of De Morgan's theorems

The second theorem changes the basic AND situation to an OR situation. A practical example of the second theorem is illustrated in Fig. 5-14*b*. The NAND gate on the left is equal in function to the OR gate (with inverted inputs) on the right. Again the long overbar is eliminated, and conversions can be done from maxterm to minterm forms of Boolean expressions. The OR-looking symbol at the right in Fig. 5-14*b* would produce a NAND truth table.

The symbols at the right in Fig. 5-14 are the alternate symbols used for the NOR and NAND logic functions. Figure 5-14 illustrates but one use of De Morgan's theorems.

Four steps are needed to transform a basic AND situation to an OR situation (or an OR to an AND situation). The four steps, based on De Morgan's theorems, are as follows:

1. Change all ORs to ANDs and all ANDs to ORs.
2. Complement each individual variable (add overbar to each).
3. Complement the entire function (add overbar to entire function).
4. Eliminate all groups of double overbars.

Consider the maxterm expression in Fig. 5-15*a*. By using the above procedure, transform this maxterm expression into a minterm expression. The *first step* (Fig. 5-15*b*) is to change all ORs to ANDs and ANDs to ORs. The *second step* (Fig. 5-15*c*) is to add an overbar to each individual variable. The *third step* (Fig. 5-15*d*) is to add an overbar to the entire function. The *fourth step* is to eliminate all double overbars and rewrite the final minterm expression. The five groups of double overbars which will be eliminated are shown in the shaded areas in Fig. 5-15*e*. The final minterm expression appears in Fig. 5-15*f*. The maxterm expression in Fig. 5-15*a* and the minterm expression in Fig. 5-15*f* will produce the same truth table.

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

(*a*) Maxterm expression

$$\overline{\bar{A}\cdot\bar{B}\cdot\bar{C}+A\cdot B\cdot\bar{C}}$$

(*b*) First step

$$\overline{\bar{A}\cdot\bar{B}\cdot\bar{C}+\bar{A}\cdot\bar{B}\cdot\bar{C}}$$

(*c*) Second step

$$\overline{\overline{\bar{A}\cdot\bar{B}\cdot\bar{C}+\bar{A}\cdot\bar{B}\cdot\bar{C}}}$$

(*d*) Third step

$$\overline{\overline{\bar{A}\cdot\bar{B}\cdot\bar{C}+\bar{A}\cdot\bar{B}\cdot\bar{C}}}$$

(*e*) Fourth step

$$A\cdot B\cdot C+\bar{A}\cdot\bar{B}\cdot C=Y$$

(*f*) Minterm expression

Fig. 5-15 From maxterm to minterm expressions using De Morgan's theorems

SOLVED PROBLEMS

5.12 Convert the Boolean expression $\overline{(A+\bar{B}+\bar{C})\cdot(\bar{A}+B+\bar{C})}=Y$ to its minterm form. Show each step as in Fig. 5-15.

Solution:

Maxterm expression	$\overline{(A+\bar{B}+\bar{C})\cdot(\bar{A}+B+\bar{C})}=Y$
First step	$\overline{A\cdot\bar{B}\cdot\bar{C}+\bar{A}\cdot B\cdot\bar{C}}$
Second step	$\overline{\bar{A}\cdot\bar{\bar{B}}\cdot\bar{\bar{C}}+\bar{\bar{A}}\cdot\bar{B}\cdot\bar{\bar{C}}}$
Third step	$\bar{A}\cdot\bar{\bar{B}}\cdot\bar{\bar{C}}+\bar{\bar{A}}\cdot\bar{B}\cdot\bar{\bar{C}}$
Fourth step	eliminate double overbars
Minterm expression	$\bar{A}\cdot B\cdot C+A\cdot\bar{B}\cdot C=Y$

5.13 Convert the Boolean expression $\overline{\bar{C}\cdot\bar{B}\cdot\bar{A}+C\cdot B\cdot\bar{A}}=Y$ to its maxterm form. Show each step in the procedure.

Solution:

Minterm expression	$\overline{\bar{C}\cdot\bar{B}\cdot\bar{A}+C\cdot B\cdot\bar{A}}=Y$
First step	$\overline{(\bar{C}+\bar{B}+\bar{A})\cdot(C+B+\bar{A})}$
Second step	$\overline{(\bar{\bar{C}}+\bar{\bar{B}}+\bar{A})\cdot(\bar{C}+\bar{B}+\bar{\bar{A}})}$
Third step	$(\bar{\bar{C}}+\bar{\bar{B}}+\bar{A})\cdot(\bar{C}+\bar{B}+\bar{\bar{A}})$
Fourth step	eliminate double overbars
Maxterm expression	$(C+B+A)\cdot(\bar{C}+\bar{B}+A)=Y$

5.14 Convert the Boolean expression $\overline{\overline{A} \cdot \overline{B}} = Y$ to a sum-of-products form.

Solution:

$A + B = Y$

5.15 Convert the Boolean expression $\overline{\overline{A} + \overline{B}} = Y$ to a product-of-sums form.

Solution:

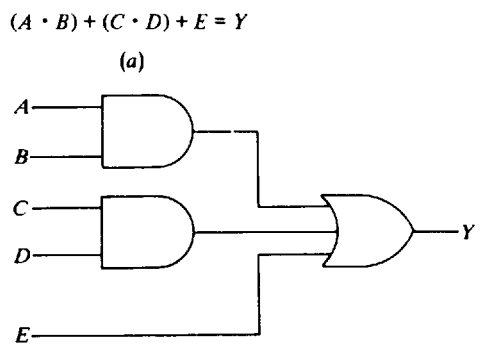
$A \cdot B = Y$

5-5 USING NAND LOGIC

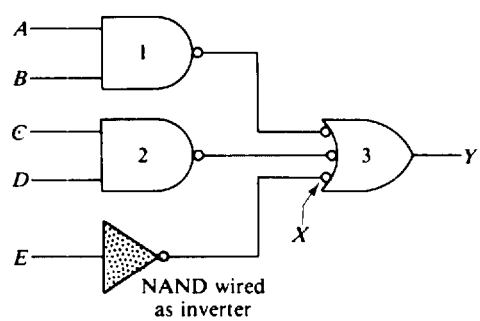
All digital systems can be constructed from the fundamental AND, OR, and NOT gates. Because of their low cost and availability, NAND gates are widely used to replace AND, OR, and NOT gates. There are several steps in converting from AND-OR logic to NAND logic:

1. Draw an AND-OR logic circuit.
2. Place a bubble at the output of each AND gate.
3. Place a bubble at each input to the OR gate.
4. Check the logic levels on lines coming from the inputs and going to the outputs.

Consider the minterm Boolean expression in Fig. 5-16*a*. To implement this expression by using NAND logic, the steps outlined above will be followed. The *first step* (Fig. 5-16*b*) is to diagram an AND-OR logic circuit. The *second step* is to place a bubble (small circle) at the output of each AND gate. This changes the AND gates to NAND gates. Figure 5-16*c* shows bubbles added to gates 1 and 2. The *third step* is to place a bubble (small circle) at each input of the OR gate. This will convert the OR gate to a NAND gate. Figure 5-16*c* shows three bubbles added to the inputs of gate 3. The *fourth step* involves examination of the input and output lines from the AND and OR symbols to see if any of the logic levels have been changed by the addition of bubbles. On examination of the circuit shown in Fig. 5-16*c*, it is found that the added bubble at point X has changed the input logic level on OR symbol 3. The AND-OR diagram in Fig. 5-16*b* shows that a HIGH logic level is connected from input E to the OR gate. The HIGH, or 1, activates the OR gate. A HIGH must also arrive at the input of symbol 3 in Fig. 5-16*c*. This is accomplished by adding the shaded inverter in input line E. In actual practice, a NAND gate is used as the inverter. The double inversion will deliver the HIGH logic level to the OR symbol to activate the OR. The invert bubbles between gates 1 and 3 cancel one another. Likewise, the invert bubbles between gates 2 and 3 cancel. The NAND logic circuit shown in Fig. 5-16*c* will produce the same truth table as that for the AND-OR circuit.



(*b*) Equivalnet AND-OR logic circuit



(*c*) Equivalent NAND logic circuit

Fig. 5-16

Using NAND logic does not always simplify a circuit. The example shown in Fig. 5-16 shows that the AND-OR circuit would probably be preferred over the NAND circuit because of the fewer gates used. Most manufacturers of ICs do produce a good variety of all types of gates. The logic designer can usually select the logic that produces the simplest circuitry.

SOLVED PROBLEMS

5.16 Diagram an AND-OR logic circuit for the Boolean expression $A \cdot B + \overline{C} + D \cdot E = Y$.

Solution:

See Fig. 5-17.

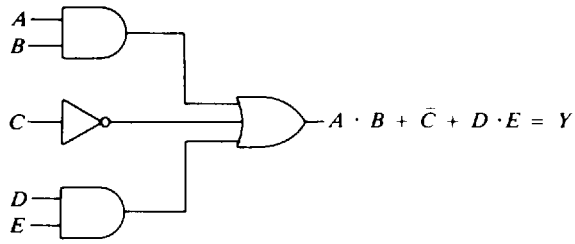


Fig. 5-17 AND-OR logic-circuit solution

5.17 Diagram a NAND logic circuit from the AND-OR circuit in Prob. 5.16. The NAND circuit should perform the logic in the expression $A \cdot B + \overline{C} + D \cdot E = Y$.

Solution:

See Fig. 5-18.

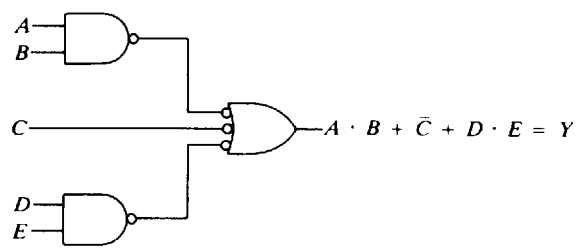


Fig. 5-18 NAND logic-circuit solution

5.18 Diagram an AND-OR logic circuit for the Boolean expression $A + (B \cdot C) + \overline{D} = Y$.

Solution:

See Fig. 5-19.

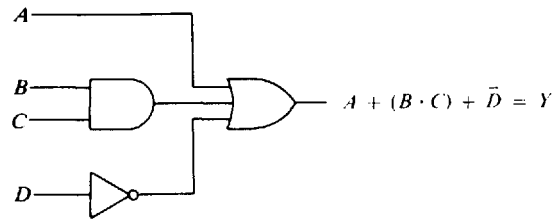


Fig. 5-19 AND-OR logic-circuit solution

5.19 Diagram a NAND logic circuit from the AND-OR circuit in Prob. 5.18. The NAND circuit should perform the logic in the expression $A + (B \cdot C) + \overline{D} = Y$.

Solution:

See Fig. 5-20.

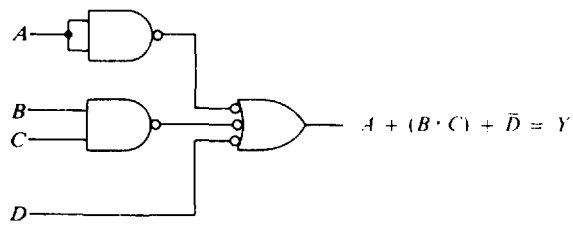
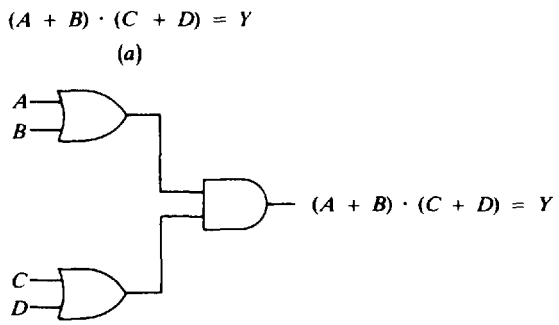


Fig. 5-20 NAND logic-circuit solution

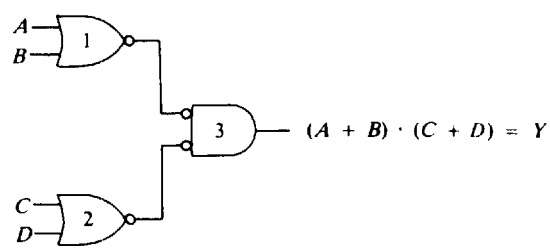
5-6 USING NOR LOGIC

The NAND gate was the "universal gate" used for substituting in an AND-OR logic pattern. When a maxterm Boolean expression forms an OR-AND gate pattern, the NAND gate does not work well. The NOR gate becomes the "universal gate" when substituting in OR-AND logic patterns. The NOR gate is not as widely used as the NAND gate.

Consider the maxterm Boolean expression written in Fig. 5-21*a*. The expression is drawn as an OR-AND logic diagram in Fig. 5-21*b*. The OR-AND pattern is redrawn with NOR gates in Fig. 5-21*c*. Each OR gate and AND gate is replaced by a NOR gate. Gates 1 and 2 in Fig. 5-21*c* are shown as the



(*b*) Equivalent OR-AND logic circuit



(*c*) Equivalent NOR logic circuit

Fig. 5-21

standard NOR symbols. Gate 3 is the alternate NOR symbol. The substitution works because the two invert bubbles between gates 1 and 3 cancel each other. Likewise, the two invert bubbles between gates 2 and 3 cancel. This leaves the two OR symbols (1 and 2) driving an AND symbol (3). This is the pattern used in the original OR-AND logic diagram in Fig. 5-21*b*.

The procedure for converting from a maxterm Boolean expression to a NOR logic circuit is similar to that used in NAND logic. The steps for converting the NOR logic are as follows:

1. Draw an OR-AND logic circuit.
2. Place a bubble at each input to the AND gate.
3. Place a bubble at the output of each OR gate.
4. Check the logic levels on lines coming from the inputs and going to the output.

Consider the maxterm Boolean expression in Fig. 5-22*a*. To implement this expression by using NOR logic, the four steps outlined above will be followed. The *first step* (Fig. 5-22*b*) is to draw an OR-AND logic circuit. The *second step* is to place a bubble (small circle) at each input to the AND gate. This changes it to a NOR gate. The "AND looking" symbol with the three bubbles at the inputs is then a NOR gate (Fig. 5-22*c*). The *third step* is to place a bubble (small circle) at the output of each OR gate. The bubbles are added to gates 1 and 2 in Fig. 5-22*c*. The *fourth step* is to examine the input and output lines for changes in logic levels due to added bubbles. The bubble added at point Z in Fig. 5-22*c* is a change from the original OR-AND pattern. The inverting effect of bubble Z is canceled by adding the shaded inverter 4. The double inversion (inverter 4 and invert bubble Z) cancels in input line E. In actual practice, inverter 4 would probably be a NOR gate. By shorting all inputs together, a NOR gate becomes an inverter. The NOR and the OR-AND circuits pictured in Fig. 5-22 perform the same logic function.

The NOR gate was used as a "universal gate" in the previous example. Using NOR logic may or may not simplify the circuit. In this case the OR-AND circuit might be preferred.

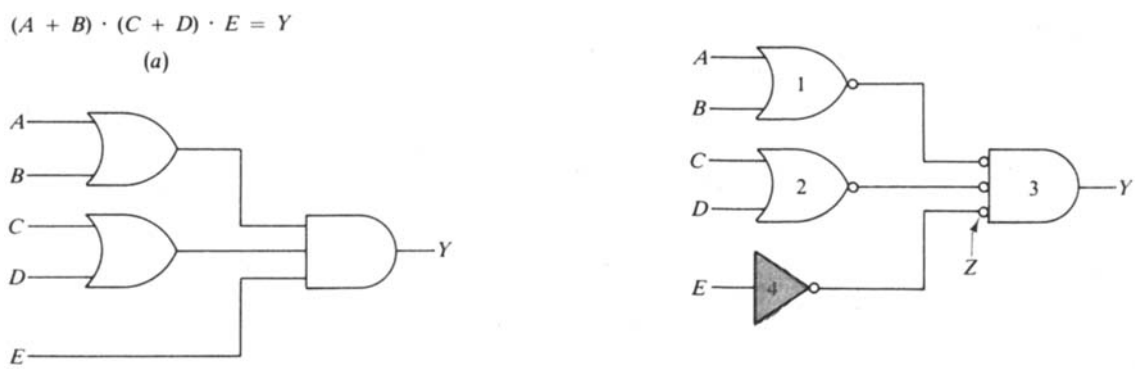


Fig. 5-22

SOLVED PROBLEMS

5.20 Diagram an OR-AND logic circuit for the Boolean expression $(A + B) \cdot \overline{C} \cdot (D + E) = Y$.

Solution:

See Fig. 5-23.

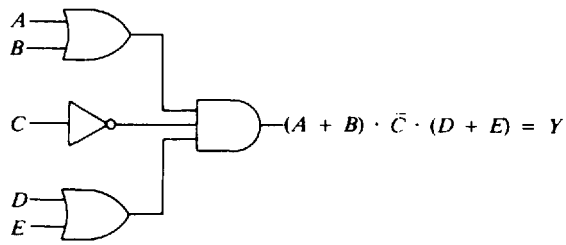


Fig. 5-23 OR-AND logic-circuit solution

5.21 Diagram a NOR logic circuit from the OR-AND circuit in Prob. 5.20. The NOR circuit should perform the logic in the Boolean expression $(A + B) \cdot \overline{C} \cdot (D + E) = Y$.

Solution:

See Fig. 5-24.

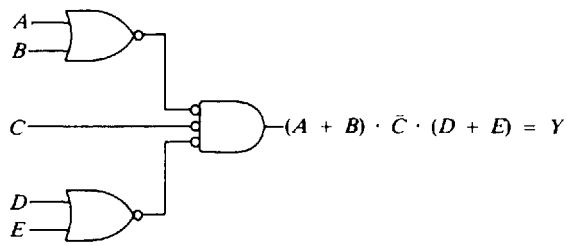


Fig. 5-24 NOR logic-circuit problem

5.22 Diagram an OR-AND logic circuit for the Boolean expression $\overline{A} \cdot (B + C) \cdot D = Y$.

Solution:

See Fig. 5-25.

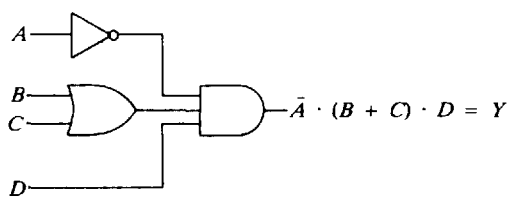


Fig. 5-25 OR-AND logic-circuit solution

5.23 Diagram a NOR logic circuit from the OR-AND circuit in Prob. 5.22. The NOR circuit should perform the logic in the Boolean expression $\overline{A} \cdot (B + C) \cdot D = Y$.

Solution:

See Fig. 5-26.

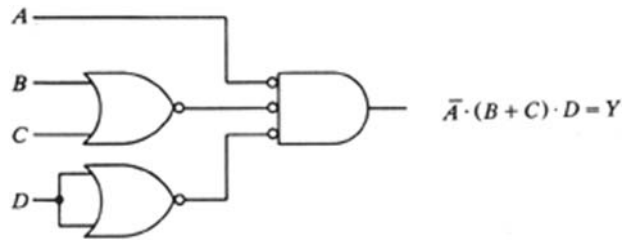


Fig. 5-26 NOR logic-circuit solution

5-7 KARNAUGH MAPS

Boolean algebra is the basis for any simplification of logic circuits. One of the easiest ways to simplify logic circuits is to use the *Karnaugh map* method. This graphic method is based on Boolean theorems. It is only one of several methods used by logic designers to simplify logic circuits. Karnaugh maps are sometimes referred to as *K maps*.

The *first step* in the Karnaugh mapping procedure is to develop a minterm Boolean expression from a truth table. Consider the familiar truth table in Fig. 5-27*a*. Each 1 in the *Y* column of the truth table produces two variables ANDed together. These ANDed groups are then ORed to form a sum-of-products (minterm) type of Boolean expression (Fig. 5-27*b*). This expression will be referred to as the *unsimplified* Boolean expression. The *second step* in the mapping procedure is to plot 1s in the Karnaugh map in Fig. 5-27*c*. Each ANDed set of variables from the minterm expression is placed in

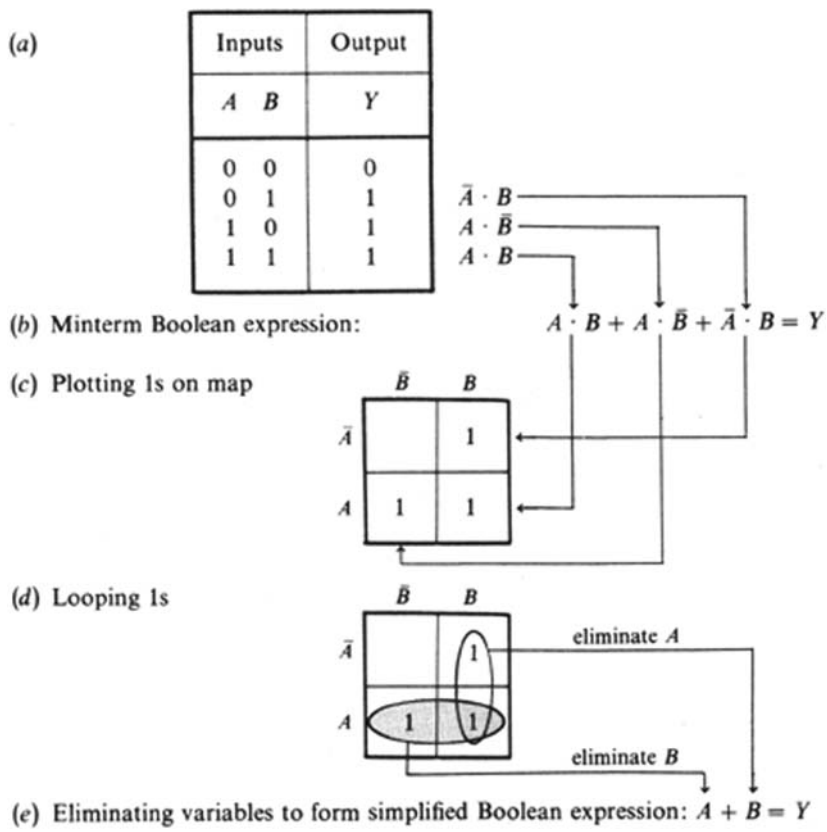


Fig. 5-27 Using a map

the *appropriate* square of the map. The map is just a very special output column of the truth table. The *third step* is to *loop* adjacent groups of two, four, or eight 1s together. Figure 5-27*d* shows two loops drawn on the map. Each loop contains two 1s. The *fourth step* is to eliminate variables. Consider first the shaded loop in Fig. 5-27*d*. Note that a B and a $\bar{B}$ (not B) are contained within the shaded loop. *When a variable and its complement are within a loop, that variable is eliminated.* From the shaded loop, the B and $\bar{B}$ terms are eliminated, leaving the A variable (Fig. 5-27*e*). Next consider the unshaded loop in Fig. 5-27*d*. It contains an A and a $\bar{A}$ (not A). The A and $\bar{A}$ terms are eliminated, leaving only the B variable (Fig. 5-27*e*). The *fifth step* is to OR the remaining variables. The final *simplified* Boolean expression is $A + B = Y$ (Fig. 5-27*e*). The simplified expression is that of a 2-input OR gate.

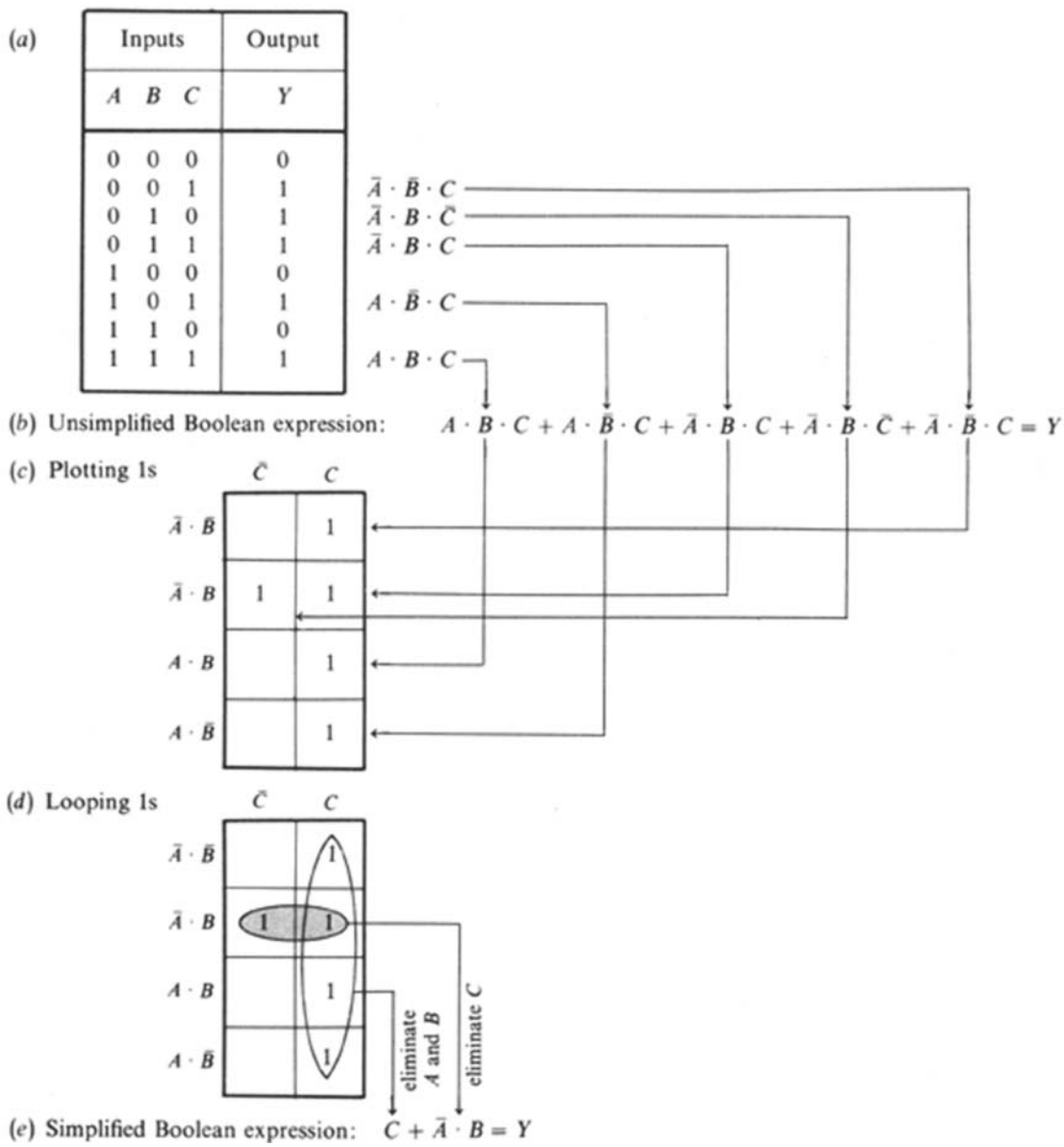


Fig. 5-28 Using a three-variable map

In summary, the steps in simplifying a logic expression using a Karnaugh map are as follows:

1. Write a minterm Boolean expression from the truth table.
2. Plot a 1 on the map for each ANDed group of variables. (The number of 1s in the Y column of the truth table will equal the number of 1s on the map.)
3. Draw loops around adjacent groups of two, four, or eight 1s on the map. (The loops may overlap.)
4. Eliminate the variable(s) that appear(s) with its (their) complement(s) within a loop, and save the variable(s) that is (are) left.
5. Logically OR the groups that remain to form the simplified minterm expression.

Consider the truth table in Fig. 5-28*a*. The *first step* in using the Karnaugh map is to write the minterm Boolean expression for the truth table. Figure 5-28*b* illustrates the unsimplified minterm expression for the truth table. The *second step* is plotting 1s on the map. Five 1s are plotted on the map in Fig. 5-28*c*. Each 1 corresponds to an ANDed group of variables (such as $A \cdot B \cdot C$). The *third step* is to loop adjacent groups of 1s on the map. Loops are placed around groups of eight, four, or two 1s. Two loops are drawn on the map in Fig. 5-28*d*. The shaded loop contains two 1s. The larger loop contains four 1s. The *fourth step* is to eliminate variables. The shaded loop in Fig. 5-28*d* contains both the C and $\overline{C}$ terms. The C variable can thus be eliminated, leaving the $\overline{A} \cdot B$ term. The large loop contains the A and $\overline{A}$ as well as the B and $\overline{B}$ terms. These can be eliminated, leaving only the C variable. The *fifth step* is to OR the remaining terms. The C and $\overline{A} \cdot B$ terms are ORed in Fig. 5-28*e*. The final simplified Boolean expression is then $C + \overline{A} \cdot B = Y$. This is much easier to implement with ICs than the unsimplified version of Fig. 5-28*b*. The simplified expression will generate the truth table in Fig. 5-28*a*.

SOLVED PROBLEMS

5.24 Write the unsimplified minterm Boolean expression for the truth table in Fig. 5-29.

Inputs			Output	Inputs			Output
A	B	C	Y	A	B	C	Y
0	0	0	1	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	0	1	1	1	1

Fig. 5-29

Solution:

$$\overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + A \cdot B \cdot C = Y$$

5.25 Draw a 3-variable Karnaugh map. Plot four 1s on the map from the Boolean expression developed in Prob. 5.24. Draw the appropriate loops around groups of 1s on the map.

Solution:

See Fig. 5-30.

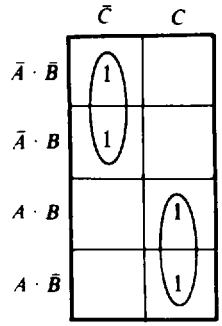


Fig. 5-30 Karnaugh map solution

5.26 Write the simplified Boolean expression based on the Karnaugh map from Prob. 5.25.

Solution:

$$\bar{A} \cdot \bar{C} + A \cdot C = Y$$

5-8 KARNAUGH MAPS WITH FOUR VARIABLES

Consider the truth table with four variables in Fig. 5-31*a*. The *first step* in simplification by using a Karnaugh map is to write the minterm Boolean expression. The lengthy unsimplified minterm expression appears in Fig. 5-31*b*. An ANDed group for four variables is written for each 1 in the Y column of the truth table. The *second step* is to plot 1s on the Karnaugh map. Nine 1s are plotted on the map in Fig. 5-31*c*. Each 1 on the map represents an ANDed group of terms from the unsimplified expression. The *third step* is to loop adjacent groups of 1s. Adjacent groups of eight, four, or two 1s are looped. Larger loops provide more simplification. Two loops have been drawn in Fig. 5-31*c*. The larger loop contains eight 1s. The *fourth step* is to eliminate variables. The large loop in Fig. 5-31*c* eliminates the A, B, and C variables. This leaves the D term. The small loop contains two 1s and

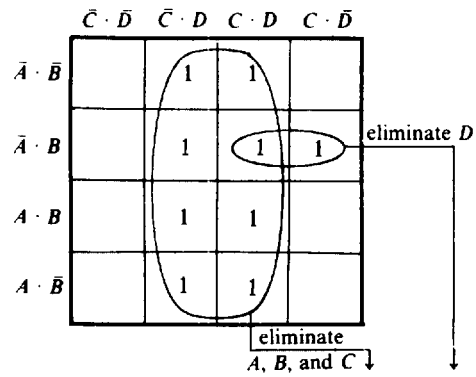
(*a*)

Inputs				Output
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

(*b*) Unsimplified minterm expression

$$\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D + \bar{A} \cdot \bar{B} \cdot C \cdot D + \bar{A} \cdot B \cdot \bar{C} \cdot D + \bar{A} \cdot B \cdot C \cdot \bar{D} + \bar{A} \cdot B \cdot C \cdot D + A \cdot \bar{B} \cdot \bar{C} \cdot D + A \cdot \bar{B} \cdot C \cdot D + A \cdot B \cdot \bar{C} \cdot D + A \cdot B \cdot C \cdot D = Y$$

(*c*) Plotting and looping 1s on map



(*d*) Simplified Boolean expression: $D + \bar{A} \cdot B \cdot C = Y$

Fig. 5-31 Using a four-variable map

eliminates the D variable. That leaves the $\bar{A} \cdot B \cdot C$ term. The *fifth step* is to logically OR the remaining terms. Figure 5-31*d* shows the remaining groups ORed to form the simplified minterm expression $D + \bar{A} \cdot B \cdot C = Y$. The amount of simplification in this example is obvious when the two Boolean expressions in Fig. 5-31 are compared.

Consider the 3-variable Karnaugh map in Fig. 5-32*a*. The letters have been omitted from the edges of the map to simplify the illustration. How many loops can be drawn on this map? There are *no adjacent groups of 1s*, and therefore no loops are drawn in Fig. 5-32*a*. No simplification is possible in the example shown in Fig. 5-32*a*.

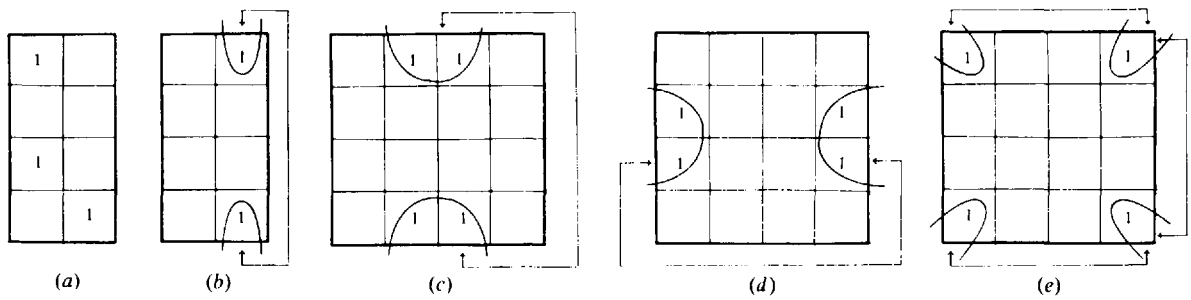


Fig. 5-32 Some unusual looping variations

The 3-variable Karnaugh map in Fig. 5-32*b* contains two 1s. Think of the top and bottom edges of the map as being connected as if rolled into a tube. The 1s can then be looped into a group of two, as shown in Fig. 5-32*b*. One variable can thus be eliminated.

Consider the 4-variable Karnaugh maps in Fig. 5-32*c* and *d*. The top and bottom edges of the map are considered connected for looping purposes in Fig. 5-32*c*. The 1s can then be looped into a group of four 1s, and two terms can be eliminated. In Fig. 5-32*d* the right edge of the map is considered connected to the left edge. The four 1s are looped into a single loop. Two variables are thus eliminated.

Another looping variation is illustrated in Fig. 5-32*e*. The corners of the map are considered connected as if the map were wrapped around a ball. The four 1s in the corners of the map are then looped into a single loop. The single loop of four 1s thereby eliminates two variables.

SOLVED PROBLEMS

5.27 Write the unsimplified minterm Boolean expression for the truth table in Fig. 5-33.

Inputs				Output	Inputs				Output
A	*B*	*C*	*D*	*Y*	*A*	*B*	*C*	*D*	*Y*
0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	1	0	1
0	0	1	1	0	1	0	1	1	1
0	1	0	0	1	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	1	0	1
0	1	1	1	0	1	1	1	1	1

Fig. 5-33

Solution:

$$\bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\bar{D}+\bar{A}\cdot B\cdot\bar{C}\cdot\bar{D}+A\cdot\bar{B}\cdot C\cdot\bar{D}+A\cdot\bar{B}\cdot C\cdot D+A\cdot B\cdot C\cdot\bar{D}+A\cdot B\cdot C\cdot D=Y$$

5.28 Draw a 4-variable Karnaugh map. Plot six 1s on the map from the Boolean expression developed in Prob. 5.27. Draw the appropriate loops around groups of 1s on the map.

Solution:

See Fig. 5-34.

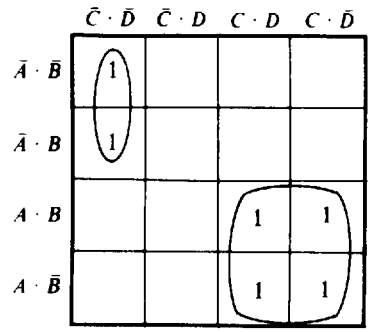


Fig. 5-34 Karnaugh map solution

5.29 Write the simplified Boolean expression based on the Karnaugh map from Prob. 5.28.

Solution:

$$A\cdot C+\bar{A}\cdot\bar{C}\cdot\bar{D}=Y$$

5.30 Write the unsimplified sum-of-products Boolean expression for the truth table in Fig. 5-35.

Inputs				Output	Inputs				Output
A	B	C	D	Y	A	B	C	D	Y
0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	1	1	0	1	0	0
0	0	1	1	0	1	0	1	1	0
0	1	0	0	1	1	1	0	0	0
0	1	0	1	0	1	1	0	1	0
0	1	1	0	1	1	1	1	0	0
0	1	1	1	0	1	1	1	1	1

Fig. 5-35

Solution:

$$\bar{A}\cdot\bar{B}\cdot\bar{C}\cdot\bar{D}+\bar{A}\cdot\bar{B}\cdot C\cdot\bar{D}+\bar{A}\cdot B\cdot\bar{C}\cdot\bar{D}+\bar{A}\cdot B\cdot C\cdot\bar{D}+A\cdot B\cdot C\cdot D=Y$$

5.31 Draw a 4-variable Karnaugh map. Plot five 1s on the map from the Boolean expression developed in Prob. 5.30. Draw the appropriate loops around groups of 1s on the map.

Solution:

See Fig. 5-36.

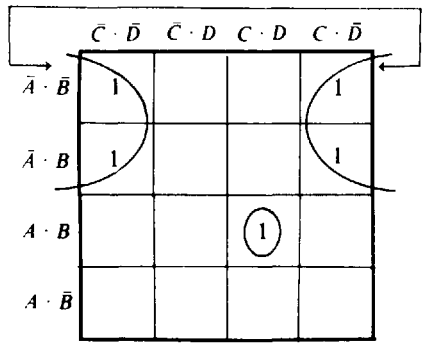


Fig. 5-36 Karnaugh map solution

5.32 Write the simplified Boolean expression based on the Karnaugh map from Prob. 5.31.

Solution:

$\bar{A} \cdot \bar{D} + A \cdot B \cdot C \cdot D = Y$

5-9 USING MAPS WITH MAXTERM EXPRESSIONS

A different form of the Karnaugh map is used with maxterm Boolean expressions. The steps for simplifying maxterm expressions are as follows:

1. Write a maxterm Boolean expression from the truth table. (Note the inverted form in Fig. 5-37*a*.)
2. Plot a 1 on the map for each ORed group of variables. The number of 0s in the Y column of the truth table will equal the number of 1s on the map.
3. Draw loops around adjacent groups of two, four, or eight 1s on the map.
4. Eliminate the variable(s) that appear(s) with its (their) complement(s) within a loop, and save the variable(s) that is (are) left.
5. Logically AND the groups that remain to form the simplified maxterm expression.

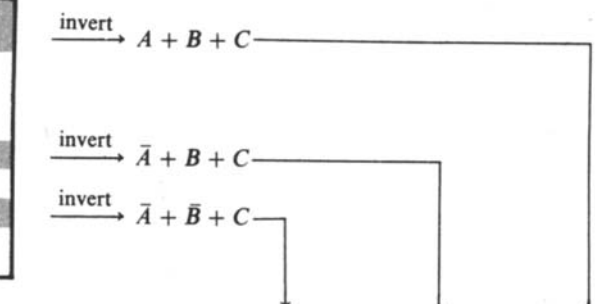
Consider the truth table in Fig. 5-37*a*. The *first step* in simplifying a maxterm expression by using a Karnaugh map is to write the expression in unsimplified form. Figure 5-37*a* illustrates how a maxterm is written for *each 0 in the Y column* of the truth table. The terms of the ORed group are *inverted* from the way they appear in the truth table. The ORed groups are then ANDed to form the unsimplified maxterm Boolean expression in Fig. 5-37*b*. The *second step* is to plot 1s on the map for each ORed group. The three maxterms in the unsimplified expression are placed as three 1s on the *revised* Karnaugh map (Fig. 5-37*c*). The *third step* is to loop adjacent groups of eight, four, or two 1s on the map. Two loops have been drawn on the map in Fig. 5-37*c*. Each loop contains two 1s. The *fourth step* is to eliminate variables. The shaded loop in Fig. 5-37*c* is shown to eliminate the A variable. This leaves the maxterm $(B + C)$. The partially unshaded loop is shown to eliminate the B variable. This leaves the maxterm $(\bar{A} + C)$. The *fifth step* is the ANDing of the remaining terms. Figure 5-37*d* shows the two maxterms being ANDed to form the simplified maxterm Boolean expression $(B + C) \cdot (\bar{A} + C) = Y$. Compare this simplified maxterm expression with the simplified minterm expression from Fig. 5-28*e*. These two expressions were developed from the *same* truth table. The minterm expression $(C + \bar{A} \cdot B = Y)$ is slightly easier to implement by using logic gates.

The maxterm mapping procedure and Karnaugh map are different from those used for minterm expressions. Both techniques should be tried on a truth table to find the less costly logic circuit.

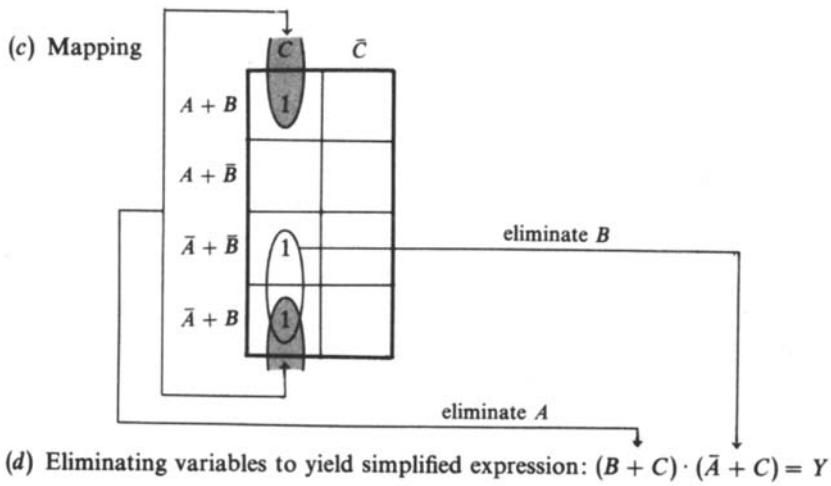
A 4-variable Karnaugh map *for maxterm expressions* is illustrated in Fig. 5-38. Note the special pattern of letters on the left and top edges of the map. *Care must always be used to position all the terms correctly when drawing maps.*

(*a*)

Inputs			Output
A	*B*	*C*	*Y*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



(*b*) Developing unsimplified maxterm expression: $(\bar{A} + \bar{B} + C) \cdot (\bar{A} + B + C) \cdot (A + \bar{B} + C) = Y$



(*d*) Eliminating variables to yield simplified expression: $(B + C) \cdot (\bar{A} + C) = Y$

Fig. 5-37 Mapping with maxterm expressions

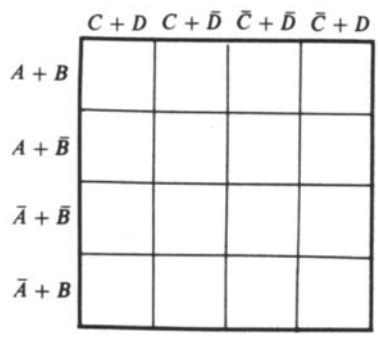


Fig. 5-38 A four-variable maxterm Karnaugh map

SOLVED PROBLEMS

5.33 Write the unsimplified maxterm Boolean expression for the truth table in Fig. 5-39. (Be sure to note the inverted form.)

Solution:

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

Inputs			Output	Inputs			Output
A	B	C	Y	A	B	C	Y
0	0	0	1	1	0	0	1
0	0	1	0	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	1	1	1	1	0

Fig. 5-39

5.34 Draw a 3-variable Karnaugh map for maxterm expressions. Plot four 1s on the map for the maxterm Boolean expression developed in Prob. 5.33. Draw the appropriate loops around groups of 1s on the map.

Solution:

See Fig. 5-40.

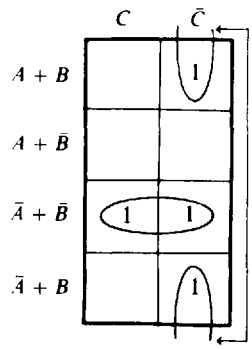


Fig. 5-40 Maxterm map solution

5.35 Write the simplified Boolean expression based on the Karnaugh map from Prob. 5.34.

Solution:

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

5.36 Write the unsimplified product-of-sums Boolean expression for the truth table in Fig. 5-41.

Inputs				Output	Inputs				Output
A	B	C	D	Y	A	B	C	D	Y
0	0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	0	1	0	0
0	0	1	1	1	1	0	1	1	1
0	1	0	0	1	1	1	0	0	1
0	1	0	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1

Fig. 5-41

Solution:

$(A + B + C + D) \cdot (A + B + C + \bar{D}) \cdot (A + B + \bar{C} + D) \cdot (\bar{A} + B + C + D) \cdot (\bar{A} + B + \bar{C} + D) = Y$

5.37 Draw a 4-variable product-of-sums Karnaugh map. Plot five 1s on the map for the Boolean expression developed in Prob. 5.36. Draw the appropriate loops around groups of 1s on the map.

Solution:

See Fig. 5-42.

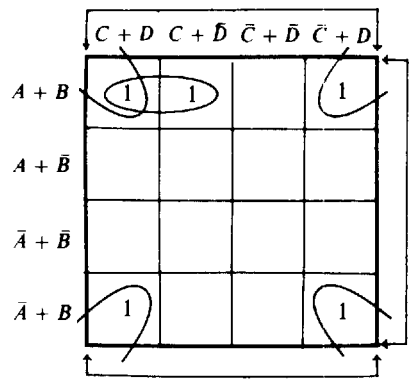


Fig. 5-42 Maxterm Karnaugh map solution

5.38 Write the simplified product-of-sums Boolean expression based on the Karnaugh map from Prob. 5.37.

Solution:

$(A + B + C) \cdot (B + D) = Y$

5-10 DON'T CARES ON KARNAUGH MAPS

Consider the table for BCD (8421) numbers given in Fig. 5-43. Note that the binary numbers 0000 to 1001 on the table are used to specify decimal numbers from 0 to 9. For convenience, the table is completed in the shaded section, which shows other possible combinations of the variables D, C, B, and A. These six combinations (1010, 1011, 1100, 1101, 1110, and 1111) are not used by the BCD code. These combinations are called *don't cares* when plotted on a Karnaugh map. The don't cares may have some effect on simplifying any logic diagram that might be constructed.

Suppose a problem specifying that a warning light would come ON when the BCD count reached 1001 (decimal 9); see the truth table in Fig. 5-44. See 1 is placed in the output column (Y) of the truth table after the input 1001. The Boolean expression for this table (above the shaded section) is $D \cdot \overline{C} \cdot \overline{B} \cdot A = Y$. This is shown to the right of the table. The "not used" combinations in the shaded section of the truth table might have some effect on this problem. A Karnaugh map is drawn in Fig. 5-45*b*. The 1 for the $D \cdot \overline{C} \cdot \overline{B} \cdot A$ term is plotted on the map. The six *don't cares* (X's from the truth table) are plotted as X's on the map. An X on the map means that square can be either a 1 or a 0. A loop is drawn around adjacent 1s. The X's on the map can be considered 1s, so the single loop is drawn around the 1 and three X's. Remember that only groups of two, four, or eight adjacent 1s and X's are looped together. The loop contains four squares, which will eliminate two variables. The B and C variables are eliminated, leaving the simplified Boolean expression $D \cdot A = Y$ in Fig. 5-45*c*.

As was said earlier, unused combinations from a truth table are called don't cares. They are shown as X's on a Karnaugh map. Including don't cares (X's) in loops on a map helps to further simplify Boolean expressions.

BCD (8421) number				Decimal equivalent
D	C	B	A	
8s	4s	2s	1s	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	not used
1	0	1	1	not used
1	1	0	0	not used
1	1	0	1	not used
1	1	1	0	not used
1	1	1	1	not used

Fig. 5-43 Table of BCD numbers

Inputs				Output
D	C	B	A	
8s	4s	2s	1s	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

$D \cdot \bar{C} \cdot \bar{B} \cdot A$

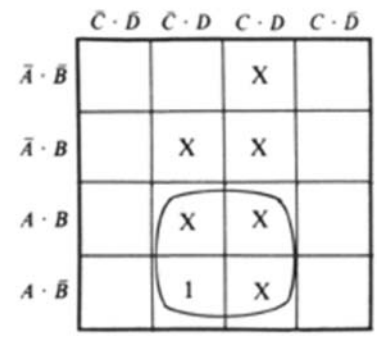
Fig. 5-44

$D \cdot \bar{C} \cdot \bar{B} \cdot A = Y$

(*a*) Unsimplified Boolean expression

$D \cdot A = Y$

(*c*) Simplified Boolean expression



(*b*) Map

Fig. 5-45 Using a map

SOLVED PROBLEMS

5.39 Write the unsimplified minterm Boolean expression for the BCD truth table in Fig. 5-46.

Solution:

$$D \cdot \bar{C} \cdot \bar{B} \cdot \bar{A} + D \cdot \bar{C} \cdot \bar{B} \cdot A = Y$$

Inputs				Output	Inputs				Output
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
8s	4s	2s	1s		8s	4s	2s	1s	
0	0	0	0	0	0	1	0	1	0
0	0	0	1	0	0	1	1	0	0
0	0	1	0	0	0	1	1	1	0
0	0	1	1	0	1	0	0	0	1
0	1	0	0	0	1	0	0	1	1

Fig. 5-46

5.40 Draw a 4-variable minterm Karnaugh map. Plot two 1s and six X's (for the don't cares) on the map based on the truth table in Fig. 5-46. Draw the appropriate loops around groups of 1s and X's on the map.

Solution:

See Fig. 5-47.

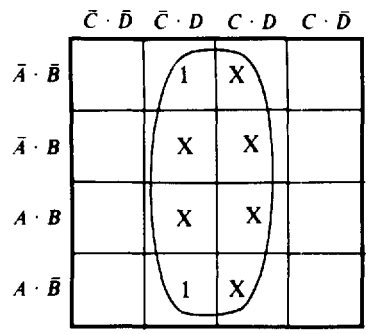


Fig. 5-47 Karnaugh map solution

5.41 Write the simplified Boolean expression based on the Karnaugh map from Prob. 5.40.

Solution:

$D = Y$

5-11 KARNAUGH MAPS WITH FIVE VARIABLES

A three-dimensional Karnaugh map can be used to solve logic problems with five variables. The map used to simplify 5-variable minterm Boolean expressions is shown in Fig. 5-48*c*. Notice that both the top (E) plane and bottom ($\bar{E}$) plane are duplicates of the 4-variable minterm map used in Sec. 5-8. The procedure for simplifying a minterm logic expression using a 5-variable Karnaugh map is like those used previously.

Consider the truth table with five variables in Fig. 5-48*a*. The *first step* in simplification is to write the minterm Boolean expression. The lengthy unsimplified minterm Boolean expression appears in Fig. 5-48*b*. An ANDed group of five variables is written for each 1 in the Y column of the truth table. The *second step* is to plot 1s on the 5-variable map. Seven 1s are plotted on the map in Fig. 5-48*c*.

(*a*)

Inputs					Output	Inputs					Output
A	*B*	*C*	*D*	*E*	*Y*	*A*	*B*	*C*	*D*	*E*	*Y*
0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	1	0	1	0	0	0	1	1
0	0	0	1	0	1	1	0	0	1	0	0
0	0	0	1	1	1	1	0	0	1	1	0
0	0	1	0	0	0	1	0	1	0	0	0
0	0	1	0	1	0	1	0	1	0	1	0
0	0	1	1	0	0	1	0	1	1	0	1
0	0	1	1	1	0	1	0	1	1	1	0
0	1	0	0	0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	1	0	0	1	0
0	1	0	1	0	1	1	1	0	1	0	0
0	1	0	1	1	1	1	1	0	1	1	0
0	1	1	0	0	0	1	1	1	0	0	0
0	1	1	0	1	0	1	1	1	0	1	0
0	1	1	1	0	0	1	1	1	1	0	0
0	1	1	1	1	0	1	1	1	1	1	0

(*b*) Unsimplified minterm expression

$$\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D\cdot\overline{E}+\overline{A}\cdot\overline{B}\cdot\overline{C}\cdot D\cdot E+\overline{A}\cdot B\cdot\overline{C}\cdot D\cdot\overline{E}+\overline{A}\cdot B\cdot\overline{C}\cdot D\cdot E+A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}\cdot\overline{E}$$
$$+A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}\cdot E+A\cdot\overline{B}\cdot C\cdot D\cdot\overline{E}=Y$$

(*c*) Plotting and looping 1s on map

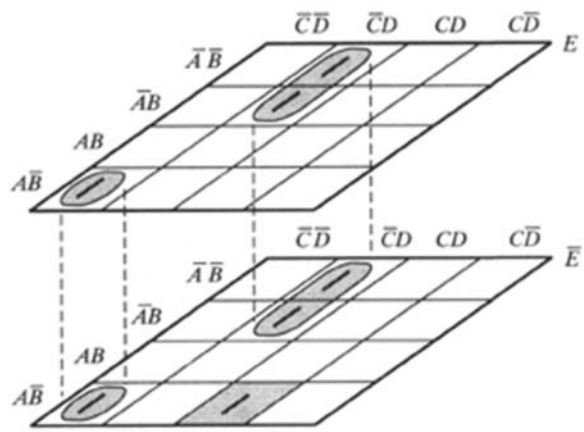


Fig. 5-48 Karnaugh map solution—5 variable

(*d*) Simplified minterm expression

$$A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}+\overline{A}\cdot\overline{C}\cdot D+A\cdot\overline{B}\cdot C\cdot D\cdot\overline{E}=Y$$

Each 1 on the map represents an ANDed group of terms from the unsimplified minterm expression. The *third step* is to loop adjacent groups of 1s. Adjacent groups of eight, four, or two 1s are looped. Two loops have been drawn in Fig. 5-48*c*. The larger loop contains four 1s and forms a cylinder between the top and bottom planes of the map. The smaller loop contains two 1s and forms the cylinder at the lower left in Fig. 5-48*c*. The single 1 near the bottom of the map does not have any 1s adjacent to it on either the E or $\overline{E}$ plane. The *fourth step* is to eliminate variables. The large loop (cylinder) in Fig. 5-48*c* eliminates the B and E variables leaving the term $\overline{A}\cdot\overline{C}\cdot D$. The smaller loop (cylinder) contains two 1s and eliminates the E term leaving the term $A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}$. The single 1 near the bottom is not looped and allows no simplification. The *fifth step* is to logically OR the remaining

terms. Figure 5-48*d* shows the remaining groups ORed, yielding the simplified minterm expression of $A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot \overline{C} \cdot D + A \cdot \overline{B} \cdot C \cdot D \cdot \overline{E} = Y$. The amount of simplification in this example is obvious when the two Boolean expressions in Fig. 5-48 are compared.

SOLVED PROBLEMS

5.42 Write the unsimplified minterm Boolean expression for the truth table in Fig. 5-49.

Inputs					Output	Inputs					Output
A	*B*	*C*	*D*	*E*	*Y*	*A*	*B*	*C*	*D*	*E*	*Y*
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	1	1
0	0	0	1	0	1	1	0	0	1	0	0
0	0	0	1	1	1	1	0	0	1	1	0
0	0	1	0	0	0	1	0	1	0	0	0
0	0	1	0	1	0	1	0	1	0	1	0
0	0	1	1	0	1	1	0	1	1	0	0
0	0	1	1	1	1	1	0	1	1	1	0
0	1	0	0	0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	1	0	0	1	1
0	1	0	1	0	1	1	1	0	1	0	0
0	1	0	1	1	1	1	1	0	1	1	0
0	1	1	0	0	0	1	1	1	0	0	0
0	1	1	0	1	0	1	1	1	0	1	0
0	1	1	1	0	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	1	1	1	0

Fig. 5-49

Solution:

$$\overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D \cdot \overline{E} + \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D \cdot E + \overline{A} \cdot \overline{B} \cdot C \cdot D \cdot \overline{E} + \overline{A} \cdot \overline{B} \cdot C \cdot D \cdot E +$$
$$\overline{A} \cdot B \cdot \overline{C} \cdot D \cdot \overline{E} + \overline{A} \cdot B \cdot \overline{C} \cdot D \cdot E + \overline{A} \cdot B \cdot C \cdot D \cdot \overline{E} + \overline{A} \cdot B \cdot C \cdot D \cdot E +$$
$$A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} \cdot E + A \cdot B \cdot \overline{C} \cdot \overline{D} \cdot E = Y$$

5.43 Draw a 5-variable Karnaugh map. Plot ten 1s on the map from the Boolean expression developed in Prob. 5.42. Draw the appropriate loops around groups of 1s on the map.

Solution:

See Fig. 5-50.

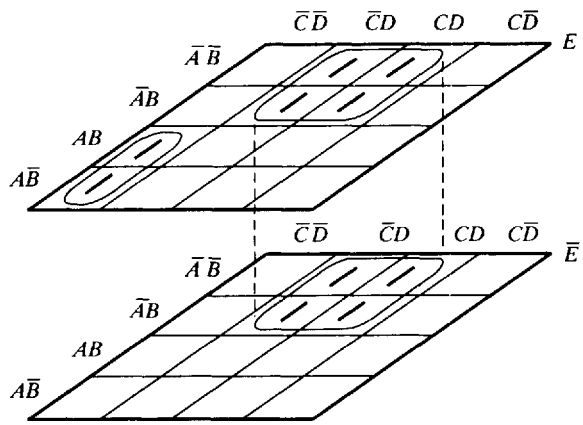


Fig. 5-50

5.44 Write the simplified Boolean expression based on the Karnaugh map from Prob. 5.43.

Solution:

$A \cdot \overline{C} \cdot \overline{D} \cdot E + \overline{A} \cdot D = Y$

Supplementary Problems

5.45 Write a minterm Boolean expression for the truth table in Fig. 5-51.
Ans. $\overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C = Y$

Inputs			Output	Inputs			Output
A	*B*	*C*	*Y*	*A*	*B*	*C*	*Y*
0	0	0	0	1	0	0	1
0	0	1	1	1	0	1	0
0	1	0	1	1	1	0	0
0	1	1	0	1	1	1	1

Fig. 5-51

5.46 Draw an AND-OR logic diagram that will perform the logic specified by the Boolean expression developed in Prob. 5.45. *Ans.* See Fig. 5-52.

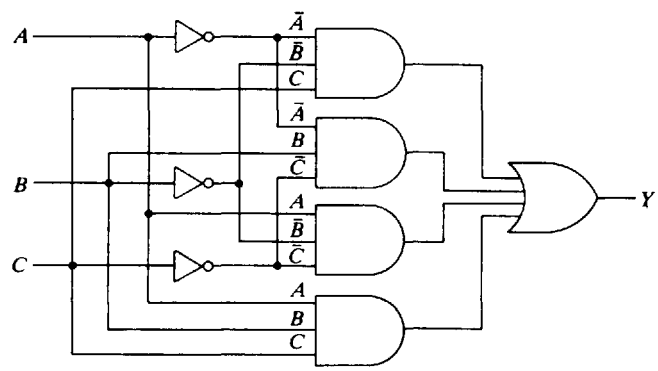


Fig. 5-52 AND-OR logic circuit

5.47 Write the maxterm Boolean expression for the truth table in Fig. 5-51.
Ans. $(A + B + C) \cdot (A + \overline{B} + \overline{C}) \cdot (\overline{A} + B + \overline{C}) \cdot (\overline{A} + \overline{B} + C) = Y$

5.48 Draw an OR-AND logic diagram that will perform the logic specified by the Boolean expression developed in Prob. 5.47. *Ans.* See Fig. 5-53.

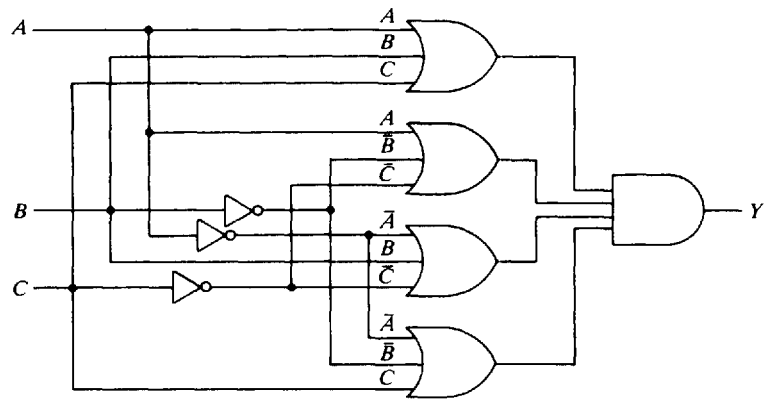


Fig. 5-53 OR-AND logic circuit

5.49 Use De Morgan's theorem to convert the Boolean expression

$$\overline{(A+\overline{B}+C+D)\cdot(A+\overline{B}+\overline{C}+D)}=Y$$

to its minterm form. Show each step as in Fig. 5-15.

Ans.

Maxterm expression	$\overline{(A+\overline{B}+C+D)\cdot(A+\overline{B}+\overline{C}+D)}=Y$
First step	$\overline{A\cdot\overline{B}\cdot C\cdot D+A\cdot\overline{B}\cdot\overline{C}\cdot D}$
Second step	$\overline{\overline{A}\cdot\overline{\overline{B}}\cdot\overline{C}\cdot\overline{D}+\overline{A}\cdot\overline{\overline{B}}\cdot\overline{\overline{C}}\cdot\overline{D}}$
Third step	$\overline{A}\cdot\overline{\overline{B}}\cdot\overline{C}\cdot\overline{D}+\overline{A}\cdot\overline{\overline{B}}\cdot\overline{\overline{C}}\cdot\overline{D}$
Fourth step	eliminate double overbars
Minterm expression	$\overline{A}\cdot B\cdot\overline{C}\cdot\overline{D}+\overline{A}\cdot B\cdot C\cdot\overline{D}=Y$

5.50 Draw a 4-variable minterm Karnaugh map. Plot two 1s on the map for the terms in the minterm expression developed in Prob. 5.49. Draw the appropriate loops around groups of 1s on the map.

Ans. See Fig. 5-54.

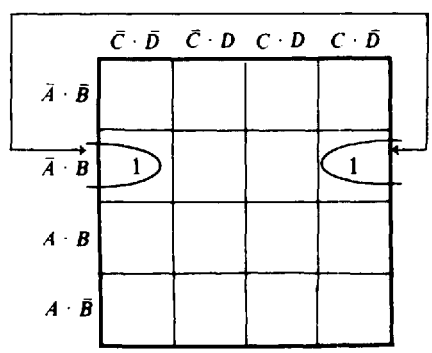


Fig. 5-54 Completed minterm Karnaugh map

5.51 Write the simplified minterm Boolean expression based on the Karnaugh map from Prob. 5.50.

Ans. $\overline{A}\cdot B\cdot\overline{D}=Y$

5.52 Use De Morgan's theorem to convert the Boolean expression $\overline{A\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}+\overline{A}\cdot B\cdot\overline{C}\cdot\overline{D}}=Y$ to its maxterm form. Show each step as in Fig. 5-15.

Ans.

Minterm expression	$\overline{\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot B \cdot \bar{C} \cdot \bar{D}} = Y$
First step	$\overline{(\bar{A} + \bar{B} + \bar{C} + \bar{D}) \cdot (\bar{A} + B + \bar{C} + \bar{D})}$
Second step	$\overline{\left(\bar{\bar{A}} + \bar{\bar{B}} + \bar{\bar{C}} + \bar{\bar{D}}\right) \cdot \left(\bar{\bar{A}} + \bar{B} + \bar{\bar{C}} + \bar{\bar{D}}\right)}$
Third step	$\left(\bar{\bar{A}} + \bar{\bar{B}} + \bar{\bar{C}} + \bar{\bar{D}}\right) \cdot \left(\bar{\bar{A}} + \bar{B} + \bar{\bar{C}} + \bar{\bar{D}}\right)$
Fourth step	eliminate double overbars
Maxterm expression	$(A + B + C + D) \cdot (A + \bar{B} + C + D) = Y$

5.53 Draw a 4-variable Karnaugh map. Plot two 1s on the map for the terms in the maxterm expression developed in Prob. 5.52. Draw the appropriate loops around groups of 1s on the map.
Ans. See Fig. 5-55.

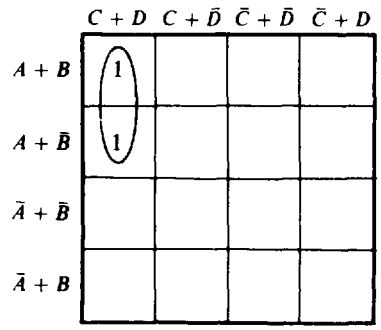


Fig. 5-55 Completed maxterm map

5.54 Write the simplified maxterm Boolean expression based on the Karnaugh map from Prob. 5.53.
Ans. $A + C + D = Y$

5.55 Draw an AND-OR logic circuit from the Boolean expression $A \cdot B + \bar{C} \cdot D + \bar{E} + \bar{F} = Y$.
Ans. See Fig. 5-56.

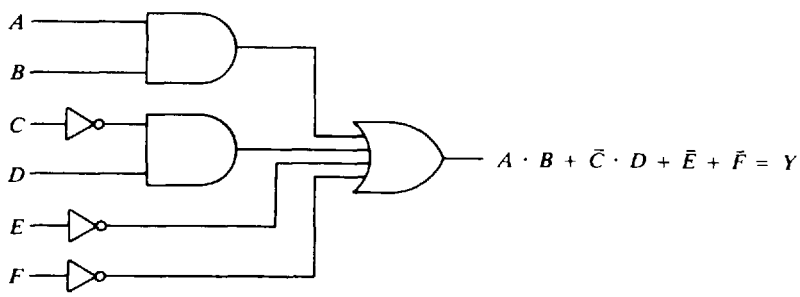


Fig. 5-56 AND-OR logic circuit

5.56 Draw a NAND logic circuit for the AND-OR circuit in Prob. 5.55. The NAND logic circuit should perform the logic in the expression $A \cdot B + \bar{C} \cdot D + \bar{E} + \bar{F} = Y$. *Ans.* See Fig. 5-57.

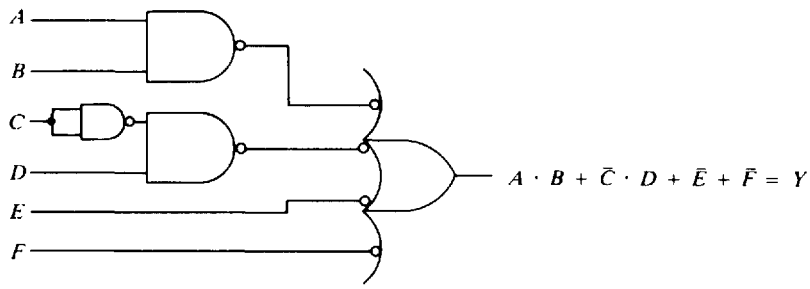


Fig. 5-57 NAND logic circuit

5.57 Draw an OR-AND logic circuit for the Boolean expression $\bar{A} \cdot (\bar{B} + C) \cdot \bar{D} \cdot E = Y$.
Ans. See Fig. 5-58.

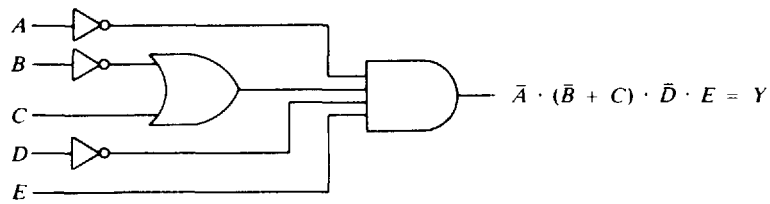


Fig. 5-58 OR-AND logic circuit

5-58 Draw a NOR logic circuit for the OR-AND circuit in Prob. 5.57. The NOR circuit should perform the logic in the expression $\bar{A} \cdot (\bar{B} + C) \cdot \bar{D} \cdot E = Y$. *Ans.* See Fig. 5-59.

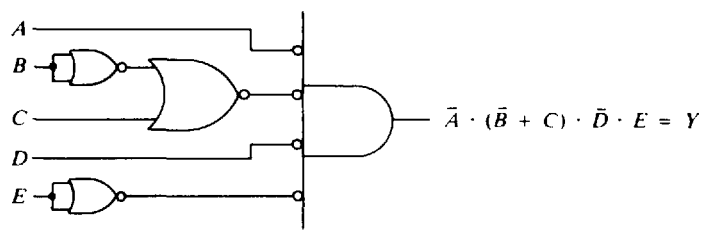


Fig. 5-59 NOR logic circuit

5.59 NOR logic can be easily substituted in an _____ (AND-OR, OR-AND) circuit.
Ans. NOR logic can be substituted for OR-AND circuits.

5.60 Write the unsimplified sum-of-products Boolean expression for the truth table in Fig. 5-60.
Ans. $\bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot \bar{B} \cdot C \cdot \bar{D} + \bar{A} \cdot B \cdot C \cdot \bar{D} + A \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} + A \cdot \bar{B} \cdot C \cdot \bar{D} + A \cdot B \cdot \bar{C} \cdot \bar{D} + A \cdot B \cdot C \cdot \bar{D} = Y$

Inputs				Output	Inputs				Output
A	B	C	D	Y	A	B	C	D	Y
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0	1	0
0	0	1	0	1	1	0	1	0	1
0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	0	1
0	1	0	1	0	1	1	0	1	0
0	1	1	0	1	1	1	1	0	1
0	1	1	1	0	1	1	1	1	0

Fig. 5-60

5.61 Draw a 4-variable minterm Karnaugh map. Plot seven 1s on the map from the Boolean expression developed in Prob. 5.60. Draw the appropriate loops around groups of 1s on the map.
Ans. See Fig. 5-61.

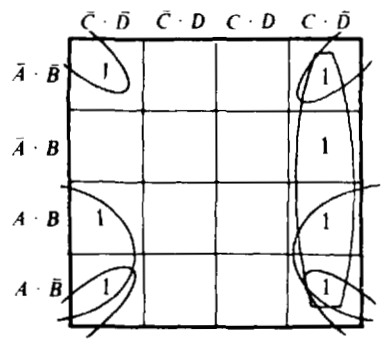


Fig. 5-61 Completed minterm map

5.62 Write the simplified minterm Boolean expression based on the Karnaugh map from Prob. 5.61.
Ans. $C\cdot\overline{D}+A\cdot\overline{D}+\overline{B}\cdot\overline{D}=Y$

5.63 Write the unsimplified product-of-sums Boolean expression for the truth table in Fig. 5-60.
Ans. $(A+B+C+\overline{D})\cdot(A+B+\overline{C}+\overline{D})\cdot(A+\overline{B}+C+D)\cdot(A+\overline{B}+C+\overline{D})\cdot(A+\overline{B}+\overline{C}+\overline{D})$
$\cdot(\overline{A}+B+C+\overline{D})\cdot(\overline{A}+B+\overline{C}+\overline{D})\cdot(\overline{A}+\overline{B}+C+\overline{D})\cdot(\overline{A}+\overline{B}+\overline{C}+\overline{D})=Y$

5.64 Draw a 4-variable maxterm Karnaugh map. Plot nine 1s on the map from the Boolean expression developed in Prob. 5.63. Draw the appropriate loops around groups of 1s on the map.
Ans. See Fig. 5-62.

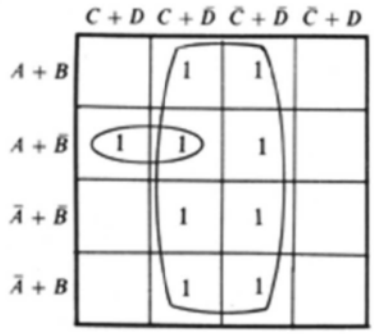


Fig. 5-62 Completed maxterm map

5.65 Write the simplified maxterm Boolean expression based on the Karnaugh map from Prob. 5.64.
Ans. $(A + \overline{B} + C) \cdot \overline{D} = Y$

5.66 The simplified _____ (maxterm, minterm) form of Boolean expression is the easiest circuit to implement for the truth table in Fig. 5-60.
Ans. The *maxterm* expression $(A + \overline{B} + C) \cdot \overline{D} = Y$ appears to be simpler to implement with logic gates than the minterm expression $C \cdot \overline{D} + A \cdot \overline{D} + \overline{B} \cdot \overline{D} = Y$.

5.67 Design a logic circuit that will respond with a 1 when even numbers (decimals 0, 2, 4, 6, 8) appear at the inputs. Figure 5-63 is the BCD (8421) truth table you will use in this problem. *Write* the unsimplified *minterm* Boolean expression for the truth table.
Ans. $\overline{D} \cdot \overline{C} \cdot \overline{B} \cdot \overline{A} + \overline{D} \cdot \overline{C} \cdot B \cdot \overline{A} + \overline{D} \cdot C \cdot \overline{B} \cdot \overline{A} + \overline{D} \cdot C \cdot B \cdot \overline{A} + D \cdot \overline{C} \cdot \overline{B} \cdot \overline{A} = Y$. The expression represents the 1s in the Y column of the truth table. Six other groups of don't cares (X's) might also be considered and will be plotted on the map.

Inputs				Output	Inputs				Output
D	*C*	*B*	*A*	*Y*	*D*	*C*	*B*	*A*	*Y*
8s	4s	2s	1s		8s	4s	2s	1s	
0	0	0	0	1	1	0	0	0	1
0	0	0	1	0	1	0	0	1	0
0	0	1	0	1	1	0	1	0	X
0	0	1	1	0	1	0	1	1	X
0	1	0	0	1	1	1	0	0	X
0	1	0	1	0	1	1	0	1	X
0	1	1	0	1	1	1	1	0	X
0	1	1	1	0	1	1	1	1	X

Fig. 5-63 Truth table with don't cares

5.68 Draw a 4-variable minterm Karnaugh map. Plot five 1s and six X's (for the don't cares) on the map based on the truth table in Fig. 5-63. Draw the appropriate loops around groups of 1s and X's on the map.
Ans. See Fig. 5-64.

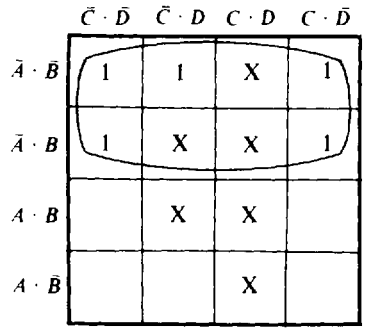


Fig. 5-64 Completed minterm maps using don't cares

5.69 Write the simplified Boolean expression based on the Karnaugh map from Prob. 5.68.
Ans. $\bar{A} = Y$

5.70 Write the simplified Boolean expression based on the Karnaugh map from Prob. 5.68 *without using the don't cares* for simplification.
Ans. $\bar{A}\cdot\bar{D} + \bar{A}\cdot\bar{B}\cdot\bar{C} = Y$. The use of the don't cares greatly aids simplification in this problem because using them reduces the expression to $\bar{A} = Y$.

5.71 In this chapter, individual logic gates were used to simplify combinational logic problems. List several more complex ICs used for logic circuit simplification.
Ans. Several ICs used to simplify combinational logic problems are data selectors (multiplexers), decoders, PLAs, ROMs, and PROMs.

5.72 Write the unsimplified minterm Boolean expression for the truth table in Fig. 5-65.
Ans. $\bar{A}\cdot\bar{B}\cdot C\cdot\bar{D}\cdot\bar{E} + \bar{A}\cdot\bar{B}\cdot C\cdot D\cdot E + \bar{A}\cdot B\cdot C\cdot\bar{D}\cdot\bar{E} + \bar{A}\cdot B\cdot C\cdot D\cdot E +$
$A\cdot B\cdot\bar{C}\cdot D\cdot\bar{E} + A\cdot B\cdot\bar{C}\cdot D\cdot E + A\cdot B\cdot C\cdot D\cdot\bar{E} + A\cdot B\cdot C\cdot D\cdot E = Y$

Inputs					Output	Inputs					Output
A	B	C	D	E	Y	A	B	C	D	E	Y
0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	1	0
0	0	0	1	0	0	1	0	0	1	0	0
0	0	0	1	1	0	1	0	0	1	1	0
0	0	1	0	0	0	1	0	1	0	0	0
0	0	1	0	1	1	1	0	1	0	1	0
0	0	1	1	0	0	1	0	1	1	0	0
0	0	1	1	1	1	1	0	1	1	1	0
0	1	0	0	0	0	1	1	0	0	0	0
0	1	0	0	1	0	1	1	0	0	1	0
0	1	0	1	0	0	1	1	0	1	0	1
0	1	0	1	1	0	1	1	0	1	1	1
0	1	1	0	0	0	1	1	1	0	0	0
0	1	1	0	1	1	1	1	1	0	1	0
0	1	1	1	0	0	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1

Fig. 5-65

5.73 Draw a 5-variable minterm Karnaugh map. Plot eight 1s on the map from the Boolean expression developed in Prob. 5.72. Draw the appropriate loops around groups of 1s on the map.
Ans. See Fig. 5-66.

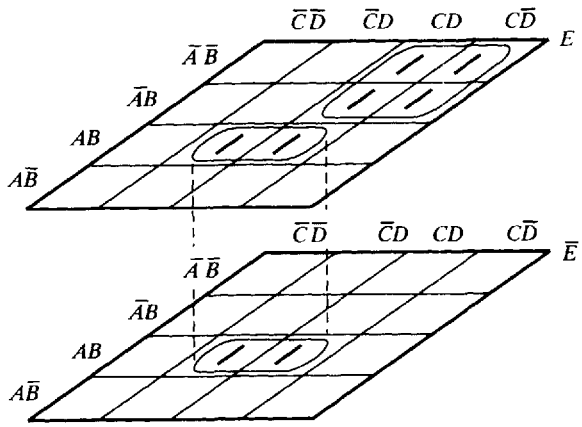


Fig. 5-66

5.74 Write the simplified minterm Boolean expression based on the Karnaugh map from Prob. 5.73.
Ans. $A \cdot B \cdot D + \overline{A} \cdot C \cdot E = Y$

Chapter 9

Flip-Flops and Other Multivibrators

9-1 INTRODUCTION

Logic circuits are classified in two broad categories. The groups of gates described thus far have been wired as *combinational logic circuits*. In this chapter a valuable type of circuit will be introduced: the *sequential logic circuit*. The basic building block of combinational logic is the logic gate. The basic building block of the sequential logic circuit is the *flip-flop* circuit. Sequential logic circuits are extremely valuable because of their *memory characteristic*.

Several types of flip-flops will be detailed in this chapter. Flip-flops are also called "latches," "bistable multivibrators," or "binaries." The term "flip-flop" will be used in this book. Useful flip-flops can be wired from logic gates, such as NAND gates, or bought in IC form. Flip-flops are interconnected to form sequential logic circuits for data storage, timing, counting, and sequencing.

Besides the bistable multivibrator (flip-flop), two other types of multivibrators (MVs) are introduced in this chapter. The *astable multivibrator* is also called a *free-running MV*. The astable MV produces a continuous series of square-wave pulses and is commonly used as a clock in a digital system. The *monostable multivibrator* is also called a *one-shot MV* in that it produces a single pulse when triggered by an external source.

9-2 *RS* FLIP-FLOP

The most basic flip-flop is called the *RS flip-flop*. A block logic symbol for the *RS* flip-flop is shown in Fig. 9-1. The logic symbol shows two inputs, labeled *set* (S) and *reset* (R), on the left. The *RS* flip-flop in this symbol has active LOW inputs, shown as small bubbles at the S and R inputs. Unlike logic gates, flip-flops have two complementary outputs. The outputs are typically labeled Q and $\overline{Q}$ (say "not Q or Q not"). The Q output is considered the "normal" output and is the one most used. The other output ($\overline{Q}$) is simply the complement of output Q, and it is referred to as the complementary output. Under normal conditions these outputs are always complementary. Hence, if $Q = 1$, then $\overline{Q} = 0$; or if $Q = 0$, then $\overline{Q} = 1$.

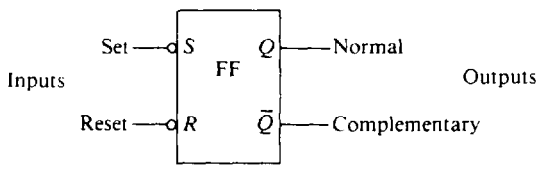
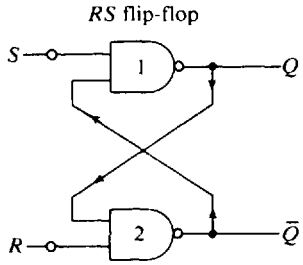


Fig. 9-1 Logic symbol for *RS* flip-flop

The *RS* flip-flop can be constructed from logic gates. An *RS* flip-flop is shown wired from two NAND gates in Fig. 9-2*a*. Note the characteristic feedback from the output of one NAND gate into the input of the other gate. As with logic gates, a truth table defines the operation of the flip-flop. Line 1 of the truth table in Fig. 9-2*b* is called the *prohibited state* in that it drives both outputs to 1, or HIGH. This condition is not used on the *RS* flip-flop. Line 2 of the truth table shows the *set* condition of the flip-flop. Here a LOW, or logical 0, activates the set (S) input. That sets the normal Q output to HIGH, or 1, as shown in the truth table. This set condition works out if the NAND circuit shown in Fig. 9-2*a* is analyzed. A 0 at gate 1 generates a 1 at output Q. This 1 is fed back to gate 2. Gate 2 now has two 1s applied to its inputs, which forces the output to 0. Output $\overline{Q}$ is therefore 0, or LOW. Line 3 in Fig. 9-2*b* is the *reset* condition. The LOW, or 0, activates the reset input. This clears (or resets) the normal Q output to 0. The fourth line of the table shows the disabled, or *hold*, condition of the *RS*



(*a*) Wired using NAND gates

Mode of operation	Inputs		Outputs	
	S	*R*	*Q*	$\bar{Q}$
Prohibited	0	0	1	1
Set	0	1	1	0
Reset	1	0	0	1
Hold	1	1	no change	

(*b*) Truth table

Fig. 9-2 *RS* flip-flop

flip-flop. The outputs remain *as they were* before the hold condition existed. There is no change in the outputs from their previous states.

Note that, when the truth table in Fig. 9-2*b* refers to the *set* condition, it means setting output Q to 1. Likewise, the *reset* condition means resetting (clearing) output Q to 0. The operating conditions therefore refer to the normal output. Note that the complementary output ($\bar{Q}$) is exactly the opposite. Because of its typical function of holding data temporarily, the *RS* flip-flop is often called the *RS latch*. *RS* latches can be wired from gates or purchased in IC form. Think of the *RS* flip-flop as a memory device that will hold a single bit of data.

SOLVED PROBLEMS

9.1 Refer to Fig. 9-1. This *RS* flip-flop has active _____ (HIGH, LOW) inputs.

Solution:

As indicated by the small bubbles at the inputs of the logic symbol in Fig. 9-1, the *RS* flip-flop has active LOW inputs.

9.2 If the normal output of the *RS* flip-flop is HIGH, then output Q = _____ (0, 1) and $\bar{Q}$ = _____ (0, 1).

Solution:

If the normal output of the *RS* flip-flop is HIGH, then output $Q = 1$ and $\bar{Q} = 0$.

9.3 Activating the reset input with a _____ (HIGH, LOW) effectively _____ (clears, sets) output Q to a logical _____ (0, 1).

Solution:

Activating the reset input with a LOW clears output Q to 0.

9.4 List the *binary* outputs at the normal output (Q) of the *RS* flip-flop shown in Fig. 9-3.

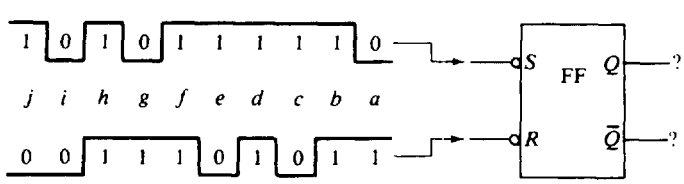


Fig. 9-3 *RS* flip-flop pulse-train problem

Solution:

The binary outputs at output Q shown in Fig. 9-3 are as follows:

pulse $a = 1$	pulse $d = 0$	pulse $g = 1$	pulse $j = 0$
pulse $b = 1$	pulse $e = 0$	pulse $h = 1$	
pulse $c = 0$	pulse $f = 0$	pulse $i = 1$ (prohibited state)	

9.5 List the *binary* outputs at output $\overline{Q}$ of the *RS* flip-flop shown in Fig. 9-3.

Solution:

The binary outputs at output $\overline{Q}$ (Fig. 9-3) are as follows:

pulse $a = 0$	pulse $d = 1$	pulse $g = 0$	pulse $j = 1$
pulse $b = 0$	pulse $e = 1$	pulse $h = 0$	
pulse $c = 1$	pulse $f = 1$	pulse $i = 1$ (prohibited state)	

9.6 List the mode of operation of the *RS* flip-flop for each input pulse shown in Fig. 9-3.

Solution:

The modes of operation of the *RS* flip-flop (Fig. 9-3) are as follows:

pulse a = set	pulse d = hold	pulse g = set	pulse j = reset
pulse b = hold	pulse e = reset	pulse h = hold	
pulse c = reset	pulse f = hold	pulse i = prohibited	

9-3 CLOCKED *RS* FLIP-FLOP

The basic *RS* latch is an *asynchronous* device. It does *not* operate in step with a clock or timing device. When an input (such as the set input) is activated, the normal output is immediately activated just as in combinational logic circuits. Gating circuits and *RS* latches operate asynchronously.

The clocked *RS flip-flop* adds a valuable *synchronous* feature to the *RS* latch. The clocked *RS* flip-flop *operates in step with the clock* or timing device. In other words, it operates synchronously. A logic symbol for the clocked *RS* flip-flop is shown in Fig. 9-4. It has the set (S) and reset (R) inputs and the added clock (CLK) input. The clocked *RS* flip-flop has the customary normal output (Q) and complementary output ($\overline{Q}$).

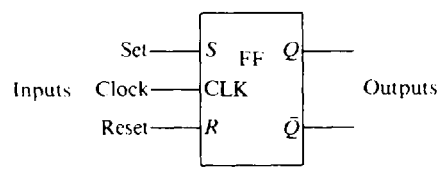
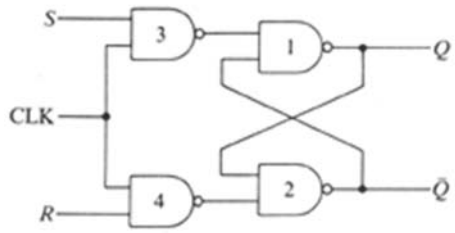


Fig. 9-4 Logic symbol for clocked *RS* flip-flop

The clocked *RS* flip-flop can be implemented with NAND gates. Figure 9-5*a* illustrates two NAND gates being added to the *RS* latch (flip-flop) to form the clocked *RS* flip-flop. NAND gates 3 and 4 add the clocked feature to the *RS* latch. Note that just gates 1 and 2 form the *RS* latch, or flip-flop. Note also that, because of the inverting effect of gates 3 and 4, the set (S) and reset (R) inputs are now active HIGH inputs. The clock (CLK) input triggers the flip-flop (enables the flip-flop) when the clock pulse goes HIGH. The clocked *RS* flip-flop is said to be a *level-triggered* device. Anytime the clock pulse is HIGH, the information at the data inputs (R and S) will be transferred to the outputs. It should be emphasized that the S and R inputs are active during the entire time the clock pulse level is HIGH. The HIGH of the clock pulse may be thought of as an enabling pulse.



(*a*) Wired using NAND gates

Mode of operation	Inputs CLK	S	R	Outputs Q	$\bar{Q}$
Hold	⎍	0	0	no change	
Reset	⎍	0	1	0	1
Set	⎍	1	0	1	0
Prohibited	⎍	1	1	1	1

⎍ = positive clock pulse

(*b*) Truth table

Fig. 9-5 Clocked *RS* flip-flop

The truth table in Fig. 9-5*b* details the operation of the clocked *RS* flip-flop. The hold mode of operation is described in line 1 of the truth table. When a clock pulse arrives at the CLK input (with 0s at the *S* and *R* inputs), the outputs *do not change*. The outputs stay the same as they were before the clock pulse. This mode might also be described as the *disabled* condition of the flip-flop. Line 2 is the reset mode. The normal output (Q) will be cleared or reset to 0 when a HIGH activates the *R* input and a clock pulse arrives at the CLK input. It will be noted that just placing $R = 1$ and $S = 0$ does not immediately reset the flip-flop. The flip-flop waits until the clock pulse goes from LOW to HIGH, and then the flip-flop resets. This unit operates synchronously, or in step with the clock. Line 3 of the truth table describes the set condition of the flip-flop. A HIGH activates the *S* input (with $R = 0$ and a HIGH clock pulse), setting the Q output to 1. Line 4 of the truth table is a prohibited combination (all inputs 1) and is not used because it drives both outputs HIGH.

Waveforms, or *timing diagrams*, are widely used and are quite useful for working with flip-flops and sequential logic circuits. Figure 9-6 is a timing diagram for the clocked *RS* flip-flop. The top three lines represent the binary signals at the clock, set, and reset inputs. Only a single output (Q) is shown across the bottom. Beginning at the left, clock pulse 1 arrives but has no effect on Q because inputs *S* and *R* are in the hold mode. Output Q therefore stays at 0. At point *a* on the timing diagram, the set input is activated to a HIGH. After a time at point *b*, output Q is set to 1. Note that the flip-flop waited until clock pulse 2 started to go from LOW to HIGH before output Q was set. Pulse 3 senses the inputs (*R* and *S*) in the hold mode, and therefore the output does not change. At point *c* the reset input is activated with a HIGH. A short time later at point *d*, output Q is cleared, or reset to 0. Again this happens on the LOW-to-HIGH transition of the clock pulse. Point *e* senses the set input activated, which sets output Q to 1 at point *f* on the timing diagram. Input *S* is deactivated and *R* is activated before pulse 6, which causes output Q to go to LOW, or to the reset condition. Pulse 7 shows that output Q follows inputs *S* and *R* the entire time the clock is HIGH. At point *g* on the timing diagram in Fig. 9-6, the set input (*S*) goes HIGH and the Q output follows by going HIGH. Input *S* then goes LOW. Next the reset input (*R*) is activated by a HIGH at point *h*. That causes

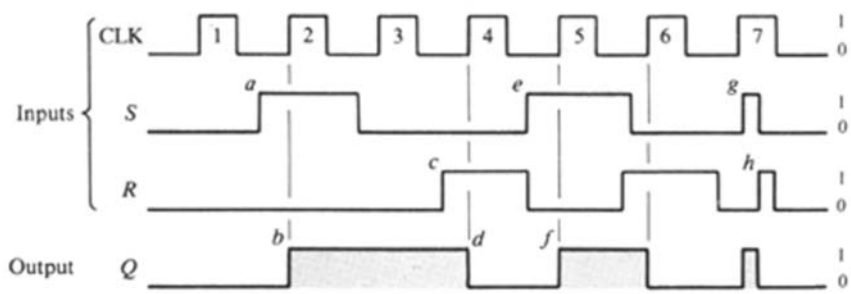


Fig. 9-6 Waveform diagram for clocked *RS* flip-flop

output Q to reset, or go LOW. Input R then returns to LOW, and finally clock pulse 7 ends with a HIGH-to-LOW transition. During clock pulse 7, the output was *set* to HIGH and then *reset* to LOW. Note that between pulses 5 and 6 it appears that both inputs S and R are at 1. The condition of inputs R and S both being HIGH would normally be considered the prohibited state for the flip-flop. In this case it is acceptable for both R and S to be HIGH because the clock pulse is LOW and the flip-flop is not activated.

SOLVED PROBLEMS

9.7 Refer to Fig. 9-4. The reset and set inputs on the clocked *RS* flip-flop are said to be active _____ (HIGH, LOW) inputs.

Solution:

The R and S inputs are active HIGH inputs on the clocked *RS* flip-flop shown in Fig. 9-4.

9.8 A flip-flop that operates in step with the clock is said to operate _____ (asynchronously, synchronously).

Solution:

A flip-flop that operates in step with the clock operates synchronously.

9.9 The *RS* latch operates _____ (asynchronously, synchronously).

Solution:

The *RS* latch operates asynchronously.

9.10 The clocked *RS* flip-flop operates _____ (asynchronously, synchronously).

Solution:

The clocked *RS* flip-flop operates synchronously.

9.11 Draw the logic symbol of a clocked *RS* flip-flop by using NAND gates.

Solution:

See Fig. 9-5*a*.

9.12 List the *binary* output at $\overline{Q}$ for the clocked *RS* flip-flop shown in Fig. 9-6 during the input clock pulses.

Solution:

The binary outputs at $\overline{Q}$ in this flip-flop are the opposite of those at the Q output. They are as follows:

pulse 1 = 1 pulse 3 = 0 pulse 5 = 0 pulse 7 = 1, then 0, and then 1
pulse 2 = 0 pulse 4 = 1 pulse 6 = 1

9.13 List the *binary* output at Q for the flip-flop of Fig. 9-7 during the eight clock pulses.

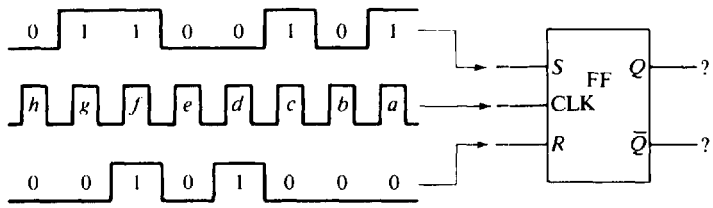


Fig. 9-7 Clocked *RS* flip-flop pulse-train problem

Solution:

The binary outputs at Q for the clocked *RS* flip-flop of Fig. 9-7 are as follows:

pulse $a = 1$	pulse $c = 1$	pulse $e = 0$	pulse $g = 1$
pulse $b = 1$	pulse $d = 0$	pulse $f = 1$ (prohibited condition)	pulse $h = 1$

9.14 List the mode of operation of the flip-flop of Fig. 9-7 during the eight clock pulses (use terms: hold, reset, set, prohibited).

Solution:

The operational modes for the clocked *RS* flip-flop of Fig. 9-7 are as follows:

pulse a = set	pulse c = set	pulse e = hold	pulse g = set
pulse b = hold	pulse d = reset	pulse f = prohibited	pulse h = hold

9.15 Refer to Fig. 9-6. The clocked *RS* flip-flop is level-triggered, which means the unit is enabled during the entire _____ (HIGH, LOW) portion of the clock pulse.

Solution:

The flip-flop of Fig. 9-6 is level-triggered, which means it is enabled during the entire HIGH portion of the clock pulse.

9-4 *D* FLIP-FLOP

The logic symbol for a common type of flip-flop is shown in Fig. 9-8. The *D flip-flop* has only a single *data* input (D) and a clock input (CLK). The customary Q and $\bar{Q}$ outputs are shown on the right side of the symbol. The D flip-flop is often called a *delay flip-flop*. This name accurately describes the unit's operation. Whatever the input at the data (D) point, it is *delayed* from getting to the normal output (Q) *by one clock pulse*. Data is transferred to the output on the LOW-to-HIGH transition of the clock pulse.

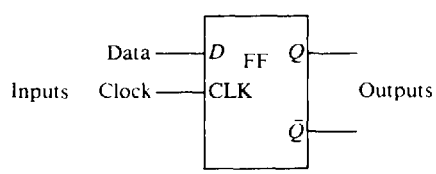
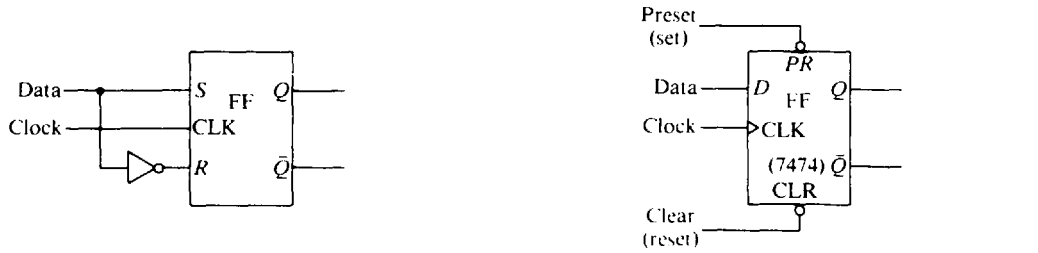


Fig. 9-8 Logic symbol for D flip-flop

The clocked *RS* flip-flop can be converted to a D flip-flop by adding an inverter. That conversion is shown in the diagram in Fig. 9-9*a*. Note that the R input to the clocked *RS* flip-flop has been inverted.



(*a*) *D* flip-flop wired from clocked *RS* flip-flop (*b*) Logic symbol for 7474 *D* flip-flop with asynchronous inputs

Fig. 9-9

A commercial *D* flip-flop is shown in Fig. 9-9*b*. The *D* flip-flop in Fig. 9-9*b* is a TTL device described by the manufacturers as a 7474 IC. The logic symbol for the 7474 *D* flip-flop shows the regular *D* and CLK inputs. Those inputs are called the *synchronous* inputs, for they operate in step with the clock. The extra two inputs are the *asynchronous* inputs, and they operate just as in the *RS* flip-flop discussed previously. The asynchronous inputs are labeled preset (*PR*) and clear (CLR). The preset (*PR*) input can be activated by a LOW, as shown by the small bubble on the logic symbol. When the preset (*PR*) is activated, it *sets* the flip-flop. In other words, it places a 1 at the normal output (*Q*). It presets *Q* to 1. The clear (CLR) input can be activated by a LOW, as shown by the small bubble on the logic symbol. When the clear (CLR) input to the *D* flip-flop is activated, the *Q* output is reset, or cleared to 0. The *asynchronous inputs override the synchronous inputs* on this *D* flip-flop.

A truth table for the 7474 *D* flip-flop is in Fig. 9-10. The modes of operation are given on the left and the truth table on the right. The first three lines are for asynchronous operation (preset and clear inputs). Line 1 shows the preset (*PR*) input activated with a LOW. That sets the *Q* output to 1. Note the X's under the synchronous inputs (CLK and *D*). The X's mean that these inputs are irrelevant because the asynchronous inputs override them. Line 2 shows the clear (CLR) input activated with a LOW. This results in output *Q* being reset, or cleared to 0. Line 3 shows the prohibited asynchronous input (both *PR* and CLR at 0). The synchronous inputs (*D* and CLK) will operate when both asynchronous inputs are disabled (*PR* = 1, CLR = 1). Line 4 shows a 1 at the data (*D*) input and a rising clock pulse (shown with the upward arrow). The 1 at input *D* is transferred to output *Q* on the clock pulse. Line 5 shows a 0 at the data (*D*) input being transferred to output *Q* on the LOW-to-HIGH clock transition.

Mode of operation	Inputs				Outputs	
	Asynchronous		Synchronous			
	PR	CLR	CLK	*D*	*Q*	$\bar{Q}$
Asynchronous set	0	1	X	X	1	0
Asynchronous reset	1	0	X	X	0	1
Prohibited	0	0	X	X	1	1
Set	1	1	↑	1	1	0
Reset	1	1	↑	0	0	1

0 = LOW, 1 = HIGH, X = irrelevant, ↑ = LOW-to-HIGH transition of the clock pulse.

Fig. 9-10 Truth table for 7474 *D* flip-flop

Only the bottom two lines of the truth table in Fig. 9-10 are needed if the D flip-flop does not have the asynchronous inputs. D flip-flops are widely used in data storage. Because of this use, it is sometimes also called a *data flip-flop*.

Look at the D flip-flop symbols shown in Figs. 9-8 and 9-9*b*. Note that the clock (CLK) input in Fig. 9-9*b* has a small $>$ inside the symbol, meaning that this is an *edge-triggered* device. This edge-triggered flip-flop transfers data from input D to output Q on the LOW-to-HIGH transition of the clock pulse. In edge triggering, it is the *change of the clock* from LOW to HIGH (or H to L) that transfers data. Once the clock pulse is HIGH on the edge-triggered flip-flop, a change in input D will have no effect on the outputs.

Figures 9-8 and 9-9*a* show a D flip-flop that is *level-triggered* (as opposed to edge-triggered). The absence of the small $>$ inside the symbol at the clock input indicates a level-triggered device. On a level-triggered flip-flop, a certain voltage level will cause the data at input D to be transferred to output Q. The problem with the level-triggered device is that the output will follow the input if the input changes while the clock pulse is HIGH. Level triggering, or clocking, can be a problem if input data changes while the clock is HIGH.

SOLVED PROBLEMS

9.16 What two other names are given to the D flip-flop?

Solution:

The D flip-flop is also called the *delay* and the *data* flip-flop.

9.17 Draw a logic diagram of a clocked RS flip-flop and inverter wired as a D flip-flop.

Solution:

See Fig. 9-9*a*.

9.18 Draw the logic symbol for a D flip-flop. Label inputs as D, CLK, PR, and CLR. Label outputs as Q and $\overline{Q}$.

Solution:

See Fig. 9-9*b*.

9.19 The data bit at the D input of the 7474 D flip-flop is transferred to output _____ (Q, $\overline{Q}$) on the _____ (H-to-L, L-to-H) transition of the clock pulse.

Solution:

The data at the D input of a D flip-flop is transferred to output Q on the L-to-H transition of the clock pulse.

9.20 Refer to Fig. 9-10. An X in the truth table stands for an _____ (extra, irrelevant) input.

Solution:

An X in the truth table stands for an irrelevant input. An X input can be either 0 or 1 and has no effect on the output.

9.21 List the binary outputs at the complementary output ($\overline{Q}$) of the D flip-flop of Fig. 9-11 after each of the clock pulses.

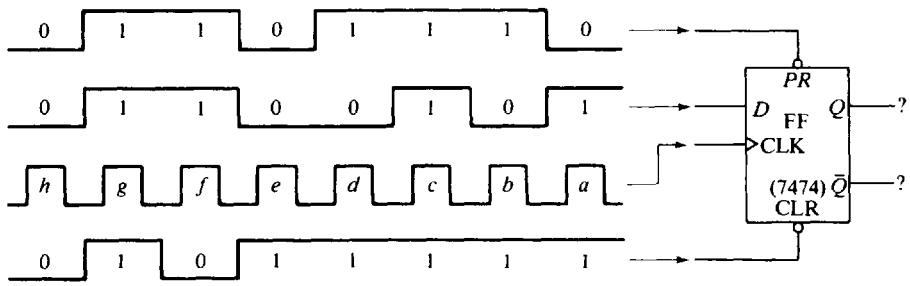


Fig. 9-11 *D* flip-flop pulse-train problem

Solution:

Refer to the truth table in Fig. 9-10. The binary outputs at $\overline{Q}$ of the *D* flip-flop (Fig. 9-11) are as follows:

pulse $a = 0$ pulse $c = 0$ pulse $e = 0$ pulse $g = 0$
pulse $b = 1$ pulse $d = 1$ pulse $f = 1$ pulse $h = 1$ (prohibited state)

9.22 Refer to Fig. 9-11. Which input has control of the flip-flop during pulse *a*?

Solution:

The preset (*PR*) input is activated during pulse *a* and overrides all other inputs. It sets the *Q* output to 1.

9.23 Refer to Fig. 9-11. Just before pulse *b*, output *Q* is _____ (HIGH, LOW); during pulse *b*, output *Q* is _____ (HIGH, LOW); on the H-to-L clock-pulse transition, output *Q* is _____ (HIGH, LOW).

Solution:

Just before pulse *b*, output *Q* is HIGH; during pulse *b*, output *Q* is LOW; on the H-to-L clock-pulse transition, output *Q* is LOW.

9-5 *JK* FLIP-FLOP

The logic symbol for a *JK* flip-flop is shown in Fig. 9-12. This device might be considered the universal flip-flop; other types can be made from it. The logic symbol shown in Fig. 9-12 has three synchronous inputs (*J*, *K*, and CLK). The *J* and *K* inputs are data inputs, and the clock input transfers data from the inputs to the outputs. The logic symbol shown in Fig. 9-12 also has the customary normal output (*Q*) and complementary output ($\overline{Q}$).

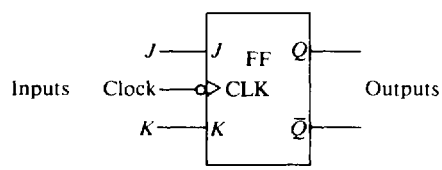


Fig. 9-12 Logic symbol for *JK* flip-flop

A truth table for the *JK* flip-flop is in Fig. 9-13. The modes of operation are given on the left and the truth table is on the right. Line 1 of the truth table shows the hold, or disabled, condition. Note that both data inputs (*J* and *K*) are LOW. The reset, or clear, condition of the flip-flop is shown in

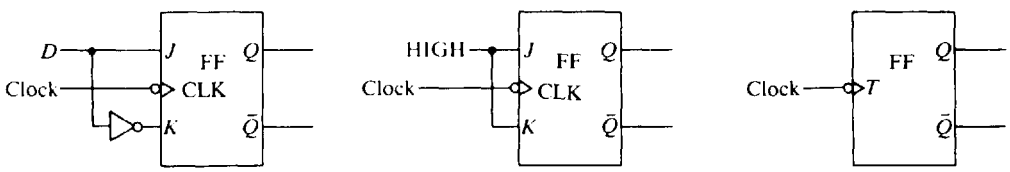
Mode of operation	Inputs			Outputs	
	CLK	J	K	Q	$\bar{Q}$
Hold	⎍	0	0	no change	
Reset	⎍	0	1	0	1
Set	⎍	1	0	1	0
Toggle	⎍	1	1	opposite state	

Fig. 9-13 Truth table for pulse-triggered *JK* flip-flop

line 2 of the truth table. When $J = 0$ and $K = 1$ and a clock pulse arrives at the CLK input, the flip-flop is reset ($Q = 0$). Line 3 shows the set condition of the *JK* flip-flop. When $J = 1$, $K = 0$, and a clock pulse is present, output Q is set to 1. Line 4 illustrates a very useful condition of the *JK* flip-flop that is called the *toggle* position. When both inputs J and K are HIGH, the output will go to the opposite state when a pulse arrives at the CLK input. With repeated clock pulses, the Q output might go LOW, HIGH, LOW, HIGH, LOW, and so forth. This LOW-HIGH-LOW-HIGH idea is called *toggling*. The term "toggling" comes from the ON-OFF nature of a toggle switch.

Note in the truth table in Fig. 9-13 that an entire clock pulse is shown under the clock (CLK) input heading. Many *JK* flip-flops are *pulse-triggered*. It *takes the entire pulse to transfer data* from the input to the outputs of the flip-flop. With the clock input in the truth table, it is evident that the *JK* flip-flop is a synchronous flip-flop.

The *JK* is considered the universal flip-flop. Figure 9-14*a* shows how a *JK* flip-flop and an inverter would be wired to form a *D* flip-flop. Note the single *D* input at the far left and the clock input. This wired *D* flip-flop would trigger on the HIGH-to-LOW transition of the clock pulse, as shown by the bubble at the CLK input.



(*a*) Wiring the *JK* flip-flop as a *D* flip-flop

(*b*) Wiring the *JK* flip-flop as a *T* flip-flop

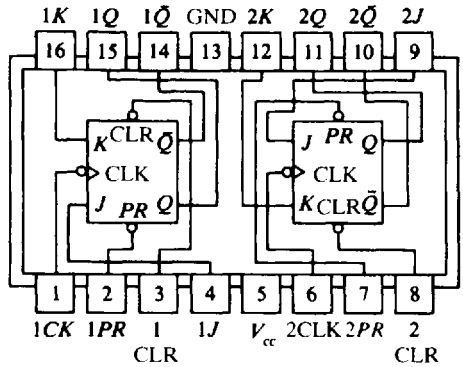
(*c*) Logic symbol for a *T* flip-flop

Fig. 9-14

A useful *toggle flip-flop* (*T*-type flip-flop) is shown wired in Fig. 9-14*b*. A *JK* flip-flop is shown being used in its toggle mode. Note that the J and K inputs are simply tied to a HIGH, and the clock is fed into the CLK input. As the repeated clock pulses feed into the CLK input, the outputs will simply toggle.

The toggle operation is widely used in sequential logic circuits. Because of its wide use, a special symbol is sometimes used for the toggle (*T*-type) flip-flop. Figure 9-14*c* shows the logic symbol for the toggle flip-flop. The single input (labeled T) is the clock input. The customary Q and $\bar{Q}$ outputs are shown on the right of the symbol. The T flip-flop has only the toggle mode of operation.

One commercial *JK* flip-flop is detailed in Fig. 9-15. This is described by the manufacturer as a *7476 TTL dual JK flip-flop*. A pin diagram of the 7476 IC is reproduced in Fig. 9-15*a*. Note that the IC contains two separate *JK* flip-flops. Each flip-flop has asynchronous preset (*PR*) and clear (CLR) inputs. The synchronous inputs are shown as J, K, and CLK (clock). The customary normal (Q) and



(*a*) Pin diagram (*Reprinted by permission of Texas Instruments, Inc.*)

Mode of operation	Inputs					Outputs	
	PR	CLR	CLK	*J*	*K*	*Q*	$\bar{Q}$
Asynchronous set	0	1	X	X	X	1	0
Asynchronous clear	1	0	X	X	X	0	1
Prohibited	0	0	X	X	X	1	1
Hold	1	1	⎍	0	0	no change	
Reset	1	1	⎍	0	1	0	1
Set	1	1	⎍	1	0	1	0
Toggle	1	1	⎍	1	1	opposite state	

X = irrelevant ⎍ = positive clock pulse

(*b*) Mode-select truth table

Fig. 9-15 The 7476 *JK* flip-flop IC

complementary ($\bar{Q}$) outputs are available. Pins 5 and 13 are the +5-V (V_{cc}) and GND power connections on this IC.

A truth table for the 7476 *JK* flip-flop is shown in Fig. 9-15*b*. The top three lines detail the operation of the asynchronous inputs preset (*PR*) and clear (CLR). Line 3 of the truth table shows the prohibited state of the asynchronous inputs. Lines 4 through 7 detail the conditions of the synchronous inputs for the hold, reset, set, and toggle modes of the 7476 *JK* flip-flop. The manufacturer describes the 7476 as a *master-slave JK flip-flop* using positive-pulse triggering. The data at the outputs changes on the H-to-L transition of the clock pulse, as symbolized by the small bubble and > symbol at the CLK input on the flip-flop logic diagram in Fig. 9-15*a*.

Most commercial *JK* flip-flops have asynchronous input features (such as *PR* and CLR). Most *JK* flip-flops are pulse-triggered devices like the 7476 IC, but they can also be purchased as edge-triggered units.

Flip-flops are the fundamental building blocks of sequential logic circuits. Therefore, IC manufacturers produce a variety of flip-flops using both the TTL and CMOS technologies. Typical TTL flip-flops are the 7476 *JK* flip-flop with preset and clear, 7474 dual positive-edge-triggered *D* flip-flop with preset and clear, and the 7475 4-bit bistable latch. Typical CMOS flip-flops include the 4724 8-bit addressable latch, 40175 quad *D* flip-flop, and the 74C76 *JK* flip-flop with preset and clear.

SOLVED PROBLEMS

9.24 Draw the logic symbol for a *JK* flip-flop with pulse triggering. Label inputs as *J*, *K*, and CLK. Label outputs as Q and $\overline{Q}$.

Solution:

See Fig. 9-12.

9.25 List the four synchronous modes of operation of the *JK* flip-flop.

Solution:

The synchronous modes of operation for the *JK* flip-flop are hold, reset, set, and toggle.

9.26 When the output of a flip-flop goes LOW, HIGH, LOW, HIGH on repeated clock pulses, it is in what mode of operation?

Solution:

If a flip-flop's output alternates states (LOW, HIGH, LOW) on repeated clock pulses, it is in the toggle mode.

9.27 List the *binary* output at output Q of the *JK* flip-flop of Fig. 9-16 after each of the eight clock pulses.

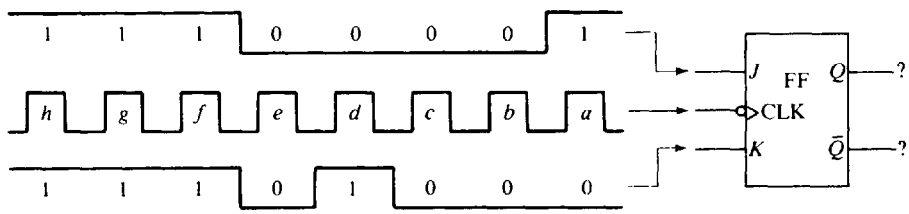


Fig. 9-16 *JK* flip-flop pulse-train problem

Solution:

Refer to the truth table in Fig. 9-13. Based on the truth table, the binary output (at Q) (Fig. 9-16) after each clock pulse is as follows:

pulse $a = 1$ pulse $c = 1$ pulse $e = 0$ pulse $g = 0$
pulse $b = 1$ pulse $d = 0$ pulse $f = 1$ pulse $h = 1$

9.28 List the mode of operation of the *JK* flip-flop during each of the eight clock pulses shown in Fig. 9-16.

Solution:

Refer to the mode-select truth table in Fig. 9-13. Based on the table, the mode of the *JK* flip-flop during each clock pulse shown in Fig. 9-16 is as follows:

pulse a = set pulse c = hold pulse e = hold pulse g = toggle
pulse b = hold pulse d = reset pulse f = toggle pulse h = toggle

9.29 List the asynchronous inputs to the 7476 *JK* flip-flop.

Solution:

The asynchronous inputs to the 7476 *JK* flip-flop are preset (*PR*) and clear (CLR).

9.30 The asynchronous inputs to the 7476 *JK* flip-flop have active _____ (HIGH, LOW) inputs.

Solution:

The asynchronous inputs to the 7476 *JK* flip-flop have active LOW inputs.

9.31 Both asynchronous inputs to the 7476 IC must be _____ (HIGH, LOW); the *J* and *K* inputs must be _____ (HIGH, LOW); and a clock pulse must be present for the flip-flop to toggle.

Solution:

Both asynchronous inputs to the 7476 IC must be HIGH; the *J* and *K* inputs must be HIGH, and a clock pulse must be present for the flip-flop to toggle.

9.32 List the mode of operation of the 7476 *JK* flip-flop during each of the seven clock pulses shown in Fig. 9-17.

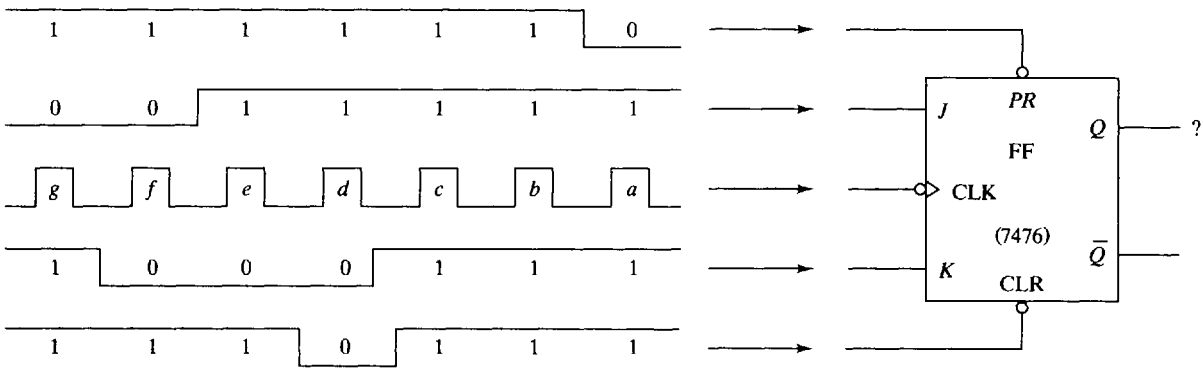


Fig. 9-17

Solution:

Refer to the mode-select truth table in Fig. 9-15*b*. Based on the table, the mode of the *JK* flip-flop during each clock pulse shown in Fig. 9-17 is as follows:
pulse a = asynchronous set
pulse b = toggle
pulse c = toggle
pulse d = asynchronous clear (reset)
pulse e = set
pulse f = hold
pulse g = reset

9.33 Refer to Fig. 9-17. List the *binary* output at Q of the *JK* flip-flop after each of the seven clock pulses.

Solution:

Refer to the truth table in Fig. 9-15*b*. Based on the table, the binary output (at Q) after each clock pulse is as follows:
pulse $a = 1$
pulse $b = 0$
pulse $c = 1$
pulse $d = 0$
pulse $e = 1$
pulse $f = 1$
pulse $g = 0$

9-6 TRIGGERING OF FLIP-FLOPS

Most complicated digital equipment operates as a synchronous sequential system. This suggests that a master clock signal is sent to all parts of the system to coordinate system operation. A typical clock pulse train is shown in Fig. 9-18. Remember that the horizontal distance on the waveform is *time* and the vertical distance is *voltage*. The clock pulses shown in the figure are for a TTL device because of the +5-V and GND voltages. Other digital circuits use clocks, but the voltages might be different.

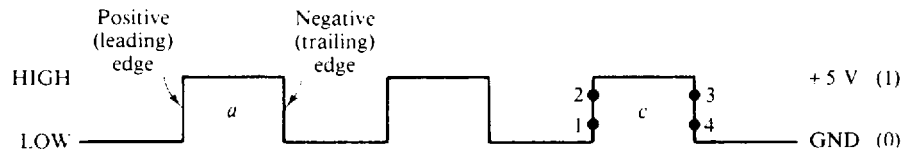


Fig. 9-18 Clock pulses

Start at the left of the waveform in Fig. 9-18. The voltage is at first LOW, or at GND. That is also called a logical 0. Pulse *a* shows the *leading edge* (also called the *positive edge*) of the waveform going from GND voltage to +5 V. This edge of the waveform may also be called the LOW-to-HIGH (L-to-H) edge of the waveform. On the right side of pulse *a*, the waveform drops from +5 V to GND voltage. This edge is called the HIGH-to-LOW (H-to-L) edge of the clock pulse. It is also called the *negative-going* edge or the *trailing* edge of the clock pulse.

Some flip-flops transfer data from input to output on the positive (leading) edge of the clock pulse. These flip-flops are referred to as *positive-edge-triggered flip-flops*. An example of such a flip-flop is shown in Fig. 9-19. The clock input is shown by the middle waveform. The top waveform shows the Q output when the positive-edge-triggered flip-flop is in its toggle mode. Note that each leading edge (positive-going edge) of the clock toggles the flip-flop.

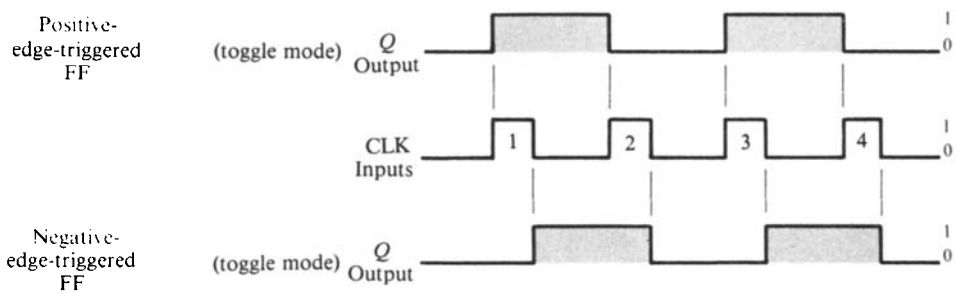


Fig. 9-19 Triggering of positive- and negative-edge flip-flops

Other flip-flops are classed as *negative-edge-triggered flip-flops*. The operation of a negative-edge-triggered flip-flop is shown by the bottom two waveforms in Fig. 9-19. The middle waveform is the clock input. The bottom waveform is the Q output when the flip-flop is in its toggle mode. Note that this flip-flop toggles to its opposite state only on the trailing edge (negative-going edge) of the clock pulse. It is important to note the difference in timing of the positive- and negative-edge-triggered flip-flops shown in Fig. 9-19. The timing difference is of great importance in some applications.

Many *JK* flip-flops are *pulse-triggered* units. These pulse-triggered devices are *master-slave JK flip-flops*. A master-slave *JK* flip-flop is actually several gates and flip-flops wired together to use the entire clock pulse to transfer data from input to output. In Fig. 9-18, pulse *c* will be used to help explain how pulse triggering works with a master-slave *JK* flip-flop. The following events happen,

during the pulse-triggering sequence, at the numbered points in Fig. 9-18:

1. The input and output of the flip-flop are isolated.
2. Data is entered from the J and K inputs, but it is not transferred to the output.
3. The J and K inputs are disabled.
4. Previously entered data from J and K is transferred to the output.

Note that the data actually appears at the outputs at point 4 (trailing edge) of the waveform in Fig. 9-18. The logic symbol for a pulse-triggered flip-flop has a small bubble attached to the clock (CLK) input (see Fig. 9-15*a*) to show that the actual transfer of data to the output takes place on the H-to-L transition of the clock pulse.

The waveforms in Fig. 9-20 will aid understanding of the operation of the master-slave JK flip-flop and pulse triggering. Start at the left of the waveform diagram. The top three waveforms are the synchronous inputs J, K, and CLK. The top line describes the mode of operation during the clock pulse. The bottom line is the resultant output of the JK flip-flop at output Q.

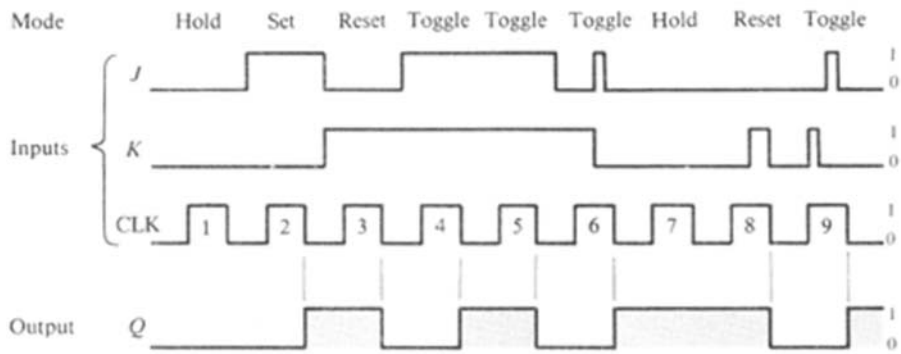


Fig. 9-20 Waveform diagram for a master-slave JK flip-flop

Look at clock (CLK) pulse 1 shown in Fig. 9-20. Both J and K inputs are LOW. This is the hold condition, and so the Q output stays at 0, as it was before pulse 1. Look at clock (CLK) pulse 2. Inputs J and K are in the set mode ($J = 1, K = 0$). On the trailing edge of pulse 2, output Q goes to a logical 1, or HIGH. Pulse 3 sees the inputs in the reset mode ($J = 0, K = 1$). On the trailing edge of clock pulse 3, output Q is reset, or cleared to 0. Pulse 4 sees the inputs in the toggle mode ($J = 1, K = 1$). On the trailing edge of clock pulse 4, output Q toggles to a logical 1, or HIGH. Pulse 5 sees the inputs in the toggle mode again. On the trailing edge of pulse 5, output Q toggles to a logical 0 or LOW.

Pulse 6 (Fig. 9-20) will show an unusual characteristic of the master-slave JK flip-flop. Note that, on the leading edge of clock pulse 6, input $K = 1$ and $J = 0$. When clock pulse 6 is HIGH, input K goes from 1 to 0 while J goes from 0 to 1 to 0. On the trailing edge of clock pulse 6, both inputs (J and K) are LOW. However, as strange as it may seem, the flip-flop still *toggles* to a HIGH. The master-slave JK flip-flop *remembers any or all HIGH inputs while the clock pulse is HIGH*. During pulse 6, both input J and K were HIGH for a short time when the clock input was HIGH. The flip-flop therefore regarded this as the toggle condition.

Next refer to clock pulse 7 (Fig. 9-20). Pulse 7 sees inputs J and K in the hold mode ($J = 0, K = 0$). Output Q remains in its present state (stays at 1). Pulse 8 sees input K at 1 for a short time and input J at 0. The flip-flop interprets this as the reset mode. Output Q therefore resets output Q to 0 on the trailing edge of clock pulse 8.

Refer to clock pulse 9 (Fig. 9-20). The master-slave JK flip-flop sees both J and K inputs at a LOW on the positive edge of clock pulse 9. When the pulse is HIGH, input K goes HIGH for a short time and then input J goes HIGH for a short time. Inputs J and K are *not* HIGH at the same time,

however. On the trailing edge of clock pulse 9, both inputs (J and K) are LOW. The flip-flop interprets this as the toggle mode. Output Q changes states and goes from a 0 to a 1.

It should be noted that not all JK flip-flops are of the master-slave type. Some JK flip-flops are edge-triggered. Manufacturers' data manuals will specify if the flip-flop is edge-triggered or pulse-triggered.

SOLVED PROBLEMS

9.34 Flip-flops are classified as either edge-triggered or _____-triggered units.

Solution:

Flip-flops are classified as either edge-triggered or pulse-triggered units.

9.35 A positive-edge-triggered flip-flop transfers data from input to output on the _____ (leading, trailing) edge of the clock pulse.

Solution:

A positive-edge-triggered flip-flop transfers data from input to output on the leading edge of the clock pulse.

9.36 A negative-edge-triggered flip-flop transfers data from input to output on the _____ (H-to-L, L-to-H) transition of the clock pulse.

Solution:

A negative-edge-triggered flip-flop transfers data from input to output on the H-to-L transition of the clock pulse.

9.37 The master-slave JK flip-flop is an example of a _____ (positive-edge-, pulse-) triggered unit.

Solution:

The master-slave JK flip-flop is an example of a pulse-triggered unit.

9.38 Refer to Fig. 9-20. List the *binary* output (at $\overline{Q}$) *after* each of the nine clock pulses.

Solution:

The $\overline{Q}$ output is always the complement of the Q output on a flip-flop. Therefore the binary outputs (at $\overline{Q}$) in Fig. 9-20 after each clock pulse are as follows:

pulse 1 = 1	pulse 3 = 1	pulse 5 = 1	pulse 7 = 0	pulse 9 = 0
pulse 2 = 0	pulse 4 = 0	pulse 6 = 0	pulse 8 = 1	

9.39 List the *binary* output (at Q) of the master-slave JK flip-flop of Fig. 9-21 *after* each of the eight clock pulses.

Solution:

Refer to the truth table in Fig. 9-13. According to this table, the binary output (at Q) of the master-slave JK flip-flop (Fig. 9-21) after each clock pulse is as follows:

pulse $a = 1$	pulse $c = 1$	pulse $e = 0$	pulse $g = 0$
pulse $b = 0$	pulse $d = 0$	pulse $f = 1$	pulse $h = 1$

9.40 List the mode of operation for the master-slave *JK* flip-flop of Fig. 9-21 for each clock pulse.

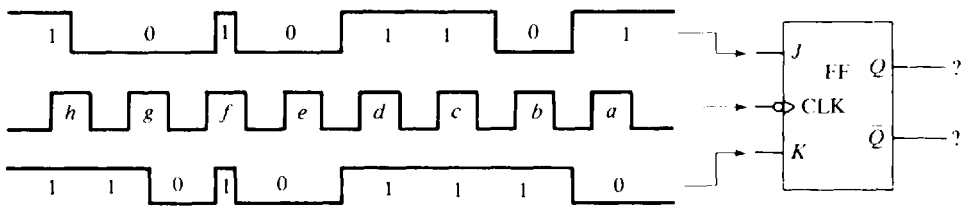


Fig. 9-21 *JK* flip-flop pulse-train problem

Solution:

Refer to the mode-select truth table in Fig. 9-13. According to the table, the modes of operation for the master-slave *JK* flip-flop (Fig. 9-21) for each clock pulse are as follows:

pulse a = set	pulse c = toggle	pulse e = hold	pulse g = reset
pulse b = reset	pulse d = toggle	pulse f = toggle	pulse h = toggle

9.41 Refer to Fig. 9-21. Assume the *JK* flip-flop is a negative-edge-triggered unit. List the *binary* output (at Q) of the edge-triggered flip-flop after each of the eight clock pulses.

Solution:

Refer to the truth table in Fig. 9-13, but remember that this is a negative-edge-triggered *JK* flip-flop (it triggers on the H-to-L transition of the clock pulse). The binary output (at Q) for the negative-edge-triggered *JK* flip-flop after each clock pulse is as follows:

pulse $a = 1$	pulse $c = 1$	pulse $e = 0$	pulse $g = 0$
pulse $b = 0$	pulse $d = 0$	pulse $f = 0$	pulse $h = 1$

9-7 ASTABLE MULTIVIBRATORS—CLOCKS

Introduction

A multivibrator (MV) is a pulse generator circuit which produces a rectangular-wave output. Multivibrators are classified as *astable*, *bistable*, or *monostable*.

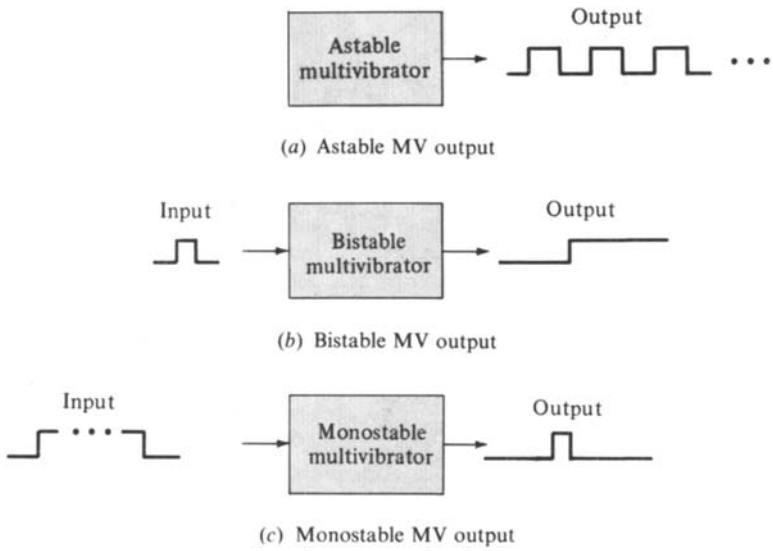
An astable MV is also called a *free-running multivibrator*. The astable MV generates a continuous flow of pulses as depicted in Fig. 9-22*a*.

A bistable multivibrator is also called a *flip-flop*. The bistable MV is always in one of two stable states (set or reset). The basic idea of a bistable MV is diagrammed in Fig. 9-22*b*, where the input pulse triggers a change in output from LOW to HIGH.

A monostable MV is also called a *one-shot multivibrator*. When the one-shot is triggered, as shown in Fig. 9-22*c*, the MV generates a single short pulse.

Astable Multivibrator

The versatile 555 Timer IC can be used to implement an astable, bistable, or monostable multivibrator. The 555 timer is shown wired as a free-running (astable) multivibrator in Fig. 9-23*a*. If



(*a*) Astable MV output

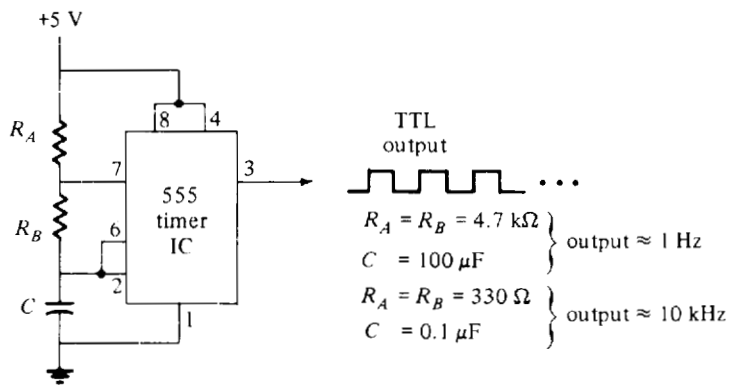
(*b*) Bistable MV output

(*c*) Monostable MV output

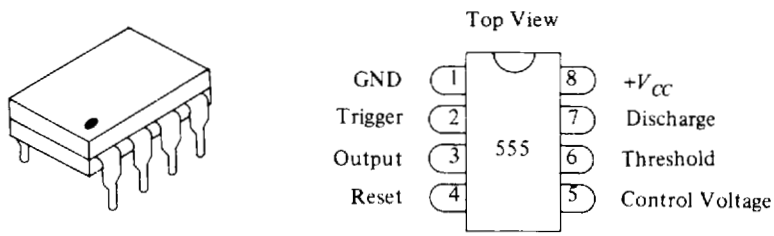
Fig. 9-22

both resistors (R_A and R_B) = 4.7 kΩ (kilohms) and C = 100 μF, the output will be a string of TTL level pulses at a frequency of 1 Hz.

The output frequency of the MV shown in Fig. 9-23*a* can be increased by *decreasing* the value of the resistors and/or capacitor. For example, if the resistors (R_A and R_B) = 330 Ω and C = 0.1 μF, then the output frequency will rise to about 10 kHz.



(*a*) 555 timer IC wired as an astable MV



(*b*) 555 timer DIP IC

(*c*) 555 timer pin diagram

Fig. 9-23

The 555 timer is commonly sold in an 8-pin DIP IC like that pictured in Fig. 9-23*b*. The pin functions for the 555 timer IC are shown in Fig. 9-23*c*.

Another astable multivibrator circuit is shown in Fig. 9-24. This free-running MV uses two CMOS inverters from the 4069 hex inverter IC. Note the use of a 10-V dc power source, which is common (but not standard) in CMOS circuits. The frequency of the output is about 10 kHz. The output frequency can be varied by changing the value(s) of the resistors and capacitor in the circuit.

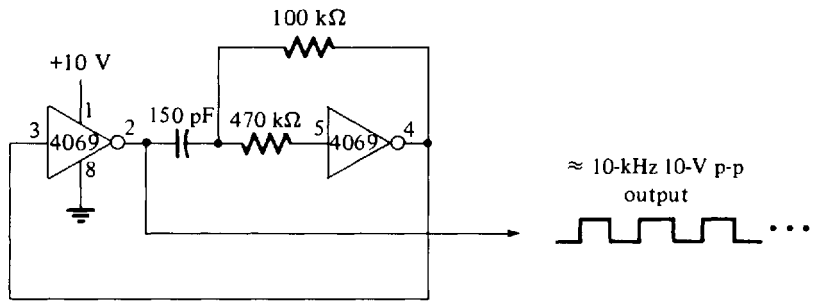


Fig. 9-24 An astable MV using CMOS inverters

Another astable multivibrator circuit using CMOS inverters is diagrammed in Fig. 9-25. This free-running MV contains a crystal-controlled oscillator (4049*a* and 4049*b*) with inverters (4049*c* and 4049*d*) used to square up the waveform. The output frequency is controlled by the natural frequency of the crystal, which is 100 kHz in this circuit. The frequency is very stable. The square-wave output is at CMOS voltage levels (about 10 V p-p).

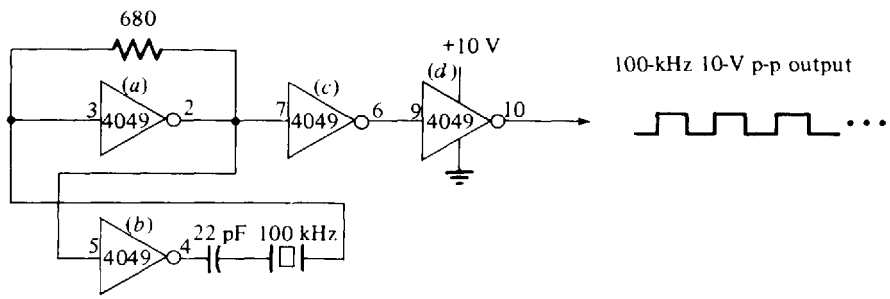


Fig. 9-25 Crystal-controlled astable MV

Astable multivibrators are often called *clocks* when they are used in digital systems. A system clock is used in all synchronous digital and microprocessor-based systems. Some important characteristics of a clock in a digital system are *frequency*, *clock cycle time*, *frequency stability*, *voltage stability*, and *shape* of the waveform. The clock cycle time is calculated by using the formula

$$T = \frac{1}{f}$$

where T = time, s
f = frequency, Hz
Clocks require square-wave pulses with fast rise times and fast fall times.

SOLVED PROBLEMS

9.42 List three classes of multivibrators.

Solution:

Multivibrators are classified as astable, bistable, or monostable.

9.43 Another name for an astable multivibrator is _____.

Solution:

An astable MV is also called a free-running multivibrator.

9.44 Another name for a bistable multivibrator is _____.

Solution:

A bistable MV is also called a flip-flop.

9.45 Another name for a monostable multivibrator is _____.

Solution:

A monostable MV is also called a one-shot multivibrator.

9.46 Increasing the value of both resistors and the capacitor in the MV circuit shown in Fig. 9-23*a* will _____ (decrease, increase) the output frequency.

Solution:

Increasing the value of the resistors and capacitor in the free-running MV of Fig. 9-23*a* will decrease the output frequency.

9.47 Pin number _____ is next to the dot on the 8-pin DIP IC shown in Fig. 9-23*b*.

Solution:

Pin 1 is located next to the dot on top of the 8-pin DIP IC of Fig. 9-23*b*.

9.48 The clock pulses from the MV shown in Fig. 9-23*a* _____ (are, are not) compatible with TTL.

Solution:

The clock pulses from the 555 timer shown in Fig. 9-23*a* are at TTL voltage levels. (LOW = 0 V and HIGH = about +4.5 V.)

9.49 The 4069 inverters used in the MV shown in Fig. 9-24 are _____ (CMOS, TTL) ICs.

Solution:

The 4069 is a CMOS hex inverter IC.

9.50 The astable MV shown in Fig. _____ (9-24, 9-25) would have the greatest frequency stability.

Solution:

The free-running MV shown in Fig. 9-25 would have great frequency stability. The oscillator's frequency is crystal-controlled.

9.51 The output from the crystal-controlled free-running MV shown in Fig. 9-25 is compatible with a _____ (CMOS, TTL) circuit.

Solution:

The 10-V output from the MV shown in Fig. 9-25 means it is compatible with a CMOS circuit. TTL voltage levels must range from 0 to +5.5 V only.

9.52 The clock cycle time for the MV shown in Fig. 9-25 is _____ s.

Solution:

The formula is $T = 1/f$, so $T = 1/100{,}000 = 0.00001$. The clock cycle time for the MV shown in Fig. 9-25 is 0.00001 s, or 10 μs.

9-8 MONOSTABLE MULTIVIBRATORS

The *one-shot* or *monostable multivibrator* generates an output pulse of fixed duration each time its input is triggered. The basic idea of the monostable MV is shown graphically in Fig. 9-22*c*. The input trigger may be an entire pulse, an L-to-H transition of the clock, or an H-to-L transition of the trigger pulse depending on the one-shot. The output pulse may be either a positive or a negative pulse. The designer can adjust the time duration of the output pulse by using different resistor-capacitor combinations.

The adaptable 555 timer IC is shown wired as a one-shot MV in Fig. 9-26. A short negative input pulse causes the longer positive output pulse. The time duration t of the output pulse is calculated by using the formula

$$t = 1.1R_AC$$

where R_A is equal to the value of the resistor in ohms, C is equal to the value of the capacitor in farads, and t is equal to the time duration of the output pulse in seconds. By calculating the output pulse time duration t for the one-shot shown in Fig. 9-26, we have

$$t = 1.1 \times 10{,}000 \times 0.0001 = 1.1 \text{ s}$$

The calculated time duration t of the output pulse for the one-shot MV shown in Fig. 9-26 is 1.1 s.

The one-shot MV shown in Fig. 9-26 is *nonretriggerable*. This means that when the one-shot's output is HIGH, it will disregard any input pulse. Retriggerable monostable MVs also are available.

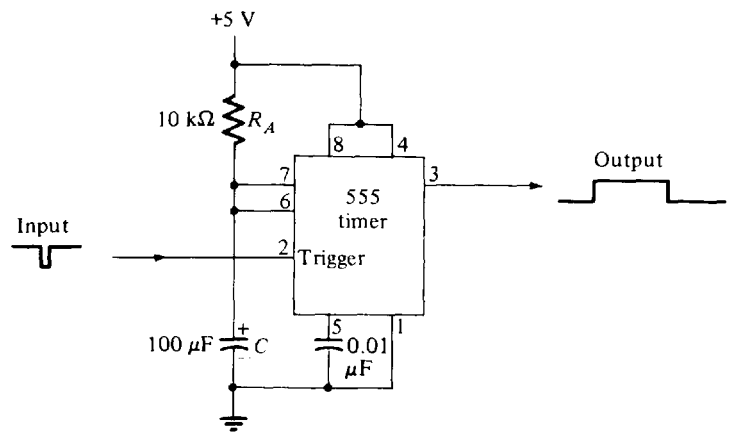


Fig. 9-26 555 timer IC wired as a monostable MV

In Fig. 9-27 the TTL 74121 one-shot IC is shown being used to generate *single* TTL level pulses when a mechanical switch is pressed. Many digital trainers used in technical education and design work use circuits of this type to generate single clock pulses. Both positive and negative clock pulses are available from the normal (Q) and complementary ($\overline{Q}$) outputs of the 74121 one-shot IC.

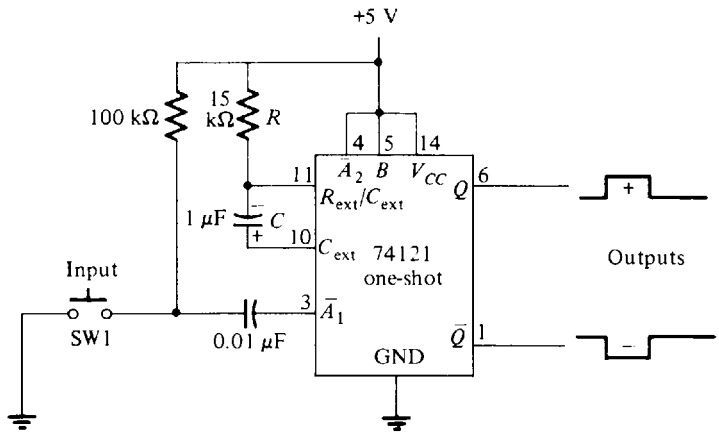


Fig. 9-27 74121 IC wired to generate single clock pulses

The duration of the output pulse can be adjusted by varying the values of the resistor R and capacitor C. To calculate the time duration of the output pulse, use the formula

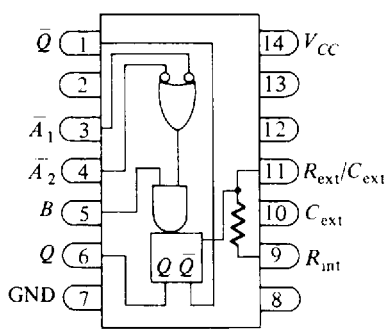
$$t = 0.7RC$$

where R equals the value of the resistor in ohms, C equals the value of the capacitor in farads, and t is the duration of the output pulse in seconds. By calculating the duration of the output pulse in the one-shot circuit in Fig. 9-27, we have

$$t = 0.7 \times 15{,}000 \times 0.000001 = 0.0105 \text{ s}$$

The calculated time duration t for the output pulse from the one-shot shown in Fig. 9-27 is 0.0105 s, or about 10 ms.

A pin diagram and truth table for the 74121 one-shot IC are reproduced in Fig. 9-28. Note that the 74121 one-shot has three separate trigger inputs ($\overline{A}_1$, $\overline{A}_2$, and B). Typically, only a single input



(*a*) Pin diagram

Inputs			Outputs	
$\overline{A}_1$	$\overline{A}_2$	B	Q	$\overline{Q}$
L	X	H	L	H
X	L	H	L	H
X	X	L	L	H
H	H	X	L	H
H	↓	H	⎍	⎎
↓	H	H	⎍	⎎
↓	↓	H	⎍	⎎
L	X	↑	⎍	⎎
X	L	↑	⎍	⎎

H = HIGH voltage level
L = LOW voltage level
X = don't care
↑ = LOW-to-HIGH transition
↓ = HIGH-to-LOW transition

(*b*) Truth table (*Courtesy of Signetics Corporation*)

Fig. 9-28

would be used at a time. In the application shown in Fig. 9-27, input $\overline{A}_1$ (pin 3) serves as the trigger input. This matches the situation in line 6 of the truth table (Fig. 9-28*b*). Inputs $\overline{A}_2$ and B are HIGH, and trigger input $\overline{A}_1$ reacts to a HIGH-to-LOW transition of the trigger pulse.

Monostable multivibrators are useful for timing applications when precision is not critical. One-shots are also used to introduce delays in digital systems.

SOLVED PROBLEMS

9.53 A _____ multivibrator is also referred to as a one-shot.

Solution:

A monostable MV is also called a one-shot.

9.54 The one-shot circuit shown in Fig. 9-26 generates a _____ (negative, positive) output pulse when triggered by a negative input pulse.

Solution:

The one-shot shown in Fig. 9-26 generates a positive output pulse when triggered by a negative input pulse.

9.55 What is the calculated time duration t of the output pulse from the one-shot shown in Fig. 9-26 if $R_A = 9.1\ \text{k}\Omega$ and $C = 10\ \mu\text{F}$?

Solution:

The formula is $t = 1.1R_AC$; then

$$t = 1.1 \times 9100 \times 0.00001$$

The calculated time duration of the output pulse from the one-shot would be 0.1 s.

9.56 An _____ (H-to-L, L-to-H) transition of the trigger pulse shown in Fig. 9-27 causes the one-shot to generate an output pulse.

Solution:

In Fig. 9-27, it is the HIGH-to-LOW transition of the trigger pulse that causes the one-shot to generate an output pulse.

9.57 In Fig. 9-27, what will be the time duration of the output pulse if $R = 30\ \text{k}\Omega$ and $C = 100\ \mu\text{F}$?

Solution:

The formula is

$$t = 0.7RC$$

so

$$t = 0.7 \times 30{,}000 \times 0.0001$$

The time duration of the output pulse from the 74121 IC will be 2.1 s.

9.58 Pressing the switch SW1 shown in Fig. 9-27 activates the trigger input and generates a _____ (negative, positive) pulse from the complementary ($\overline{Q}$) output.

Solution:

Pressing SW1 in Fig. 9-27 triggers the 74121 one-shot and generates a negative pulse from the complementary ($\overline{Q}$) output. Also see truth table in Fig. 9-28*b*.

Supplementary Problems

9.59 If it is said that "the flip-flop is set," then output Q is _____ (HIGH, LOW). *Ans.* HIGH

9.60 A(n) _____ (clocked *RS*, *RS*) flip-flop is an example of a synchronously operated device.
Ans. clocked *RS*

9.61 A(n) _____ (*D*, *RS*) flip-flop does *not* have a clock input. *Ans.* *RS*

9.62 Combinational logic circuits and the *RS* latch operate _____ (asynchronously, synchronously).
Ans. asynchronously

9.63 The normal output of a flip-flop is the _____ (Q, $\overline{Q}$) output. *Ans.* Q

9.64 List the *binary* output (at Q) of the *RS* latch shown in Fig. 9-29 for each of the eight pulses.
Ans. pulse $a = 0$ pulse $c = 1$ pulse $e = 1$ pulse $g = 1$
pulse $b = 0$ pulse $d = 0$ pulse $f = 1$ (prohibited) pulse $h = 1$

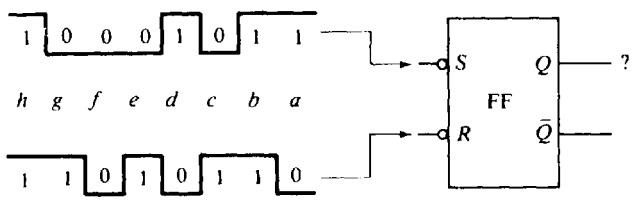


Fig. 9-29 *RS* flip-flop pulse-train problem

9.65 List the mode of operation of the *RS* flip-flop shown in Fig. 9-29 for each of the eight pulses.
Ans. pulse a = reset pulse c = set pulse e = set pulse g = set
pulse b = hold pulse d = reset pulse f = prohibited condition pulse h = hold

9.66 List the binary output at $\overline{Q}$ for the clocked *RS* flip-flop shown in Fig. 9-7 for each of the eight clock pulses.
Ans. pulse $a = 0$ pulse $c = 0$ pulse $e = 1$ pulse $g = 0$
pulse $b = 0$ pulse $d = 1$ pulse $f = 1$ pulse $h = 0$

9.67 Refer to Fig. 9-30. The clocked *RS* flip-flop is triggered by the _____ (leading, trailing) edge of the clock pulse. *Ans.* leading

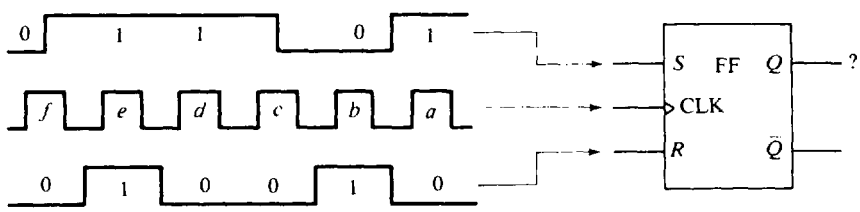


Fig. 9-30 Clocked *RS* flip-flop pulse-train problem

9.68 List the *binary* output (at Q) of the clocked RS flip-flop shown in Fig. 9-30 for each of the six clock pulses.

Ans. pulse $a = 1$ pulse $c = 0$ pulse $e = 1$ (prohibited condition)
pulse $b = 0$ pulse $d = 1$ pulse $f = 1$

9.69 List the mode of operation for the clocked RS flip-flop shown in Fig. 9-30 as each pulse triggers the unit.

Ans. pulse a = set
pulse b = reset
pulse c = hold (S and R are both 0 on leading edge)
pulse d = set
pulse e = prohibited condition
pulse f = set ($S = 1$, $R = 0$ on the leading edge)

9.70 List the *binary* output (at Q) for the D flip-flop shown in Fig. 9-11 after each of the eight clock pulses.

Ans. pulse $a = 1$ pulse $c = 1$ pulse $e = 1$ pulse $g = 1$
pulse $b = 0$ pulse $d = 0$ pulse $f = 0$ pulse $h = 1$ (prohibited condition)

9.71 Refer to Fig. 9-11. Which input has control of the flip-flop during pulse e?

Ans. The preset (PS) input is activated during pulse e and overrides all other inputs. It sets the Q output to 1.

9.72 Refer to Fig. 9-11. Which input has control of the flip-flop during pulse f?

Ans. The clear (CLR) input is activated during pulse f and overrides all other inputs. It resets the Q output to 0.

9.73 A delay flip-flop is also called a _____ (D, T)-type flip-flop. *Ans.* D

9.74 On a D flip-flop, the data bit at input D is delayed _____ (0, 1, 2, 3, 4) clock pulse(s) from getting to output _____ (Q, $\overline{Q}$). *Ans.* 1; Q

9.75 A T-type flip-flop is also called a _____ (toggle, truth-table) flip-flop. *Ans.* toggle

9.76 Draw a logic diagram showing how to wire a JK flip-flop as a T flip-flop. *Ans.* See Fig. 9-14*b*.

9.77 Draw a logic diagram showing how to wire a JK flip-flop and an inverter as a D flip-flop.

Ans. See Fig. 9-14*a*.

9.78 List the *binary* output (at $\overline{Q}$) fo the JK flip-flop shown in Fig. 9-16 after each of the eight clock pulses.

Ans. pulse $a = 0$ pulse $c = 0$ pulse $e = 1$ pulse $g = 1$
pulse $b = 0$ pulse $d = 1$ pulse $f = 0$ pulse $h = 0$

9.79 Refer to Fig. 9-16. The _____ (asynchronous, synchronous) inputs to the JK flip-flop are shown being used on this unit.

Ans. The J, K, and CLK inputs are synchronous inputs.

9.80 Refer to Fig. 9-17. Which input has control of the JK flip-flop during pulse a?

Ans. PR (Preset input activated with LOW sets output Q to 1.)

9.81 Refer to Fig. 9-17. Which input has control of the JK flip-flop during pulse d?

Ans. CLR (Clear input activated with LOW resets output Q to 0.)

9.82 Refer to Fig. 9-17. List the *binary* output at $\overline{Q}$ (complementary output) of the JK flip-flop *after* each of the seven clock pulses.

Ans. pulse $a = 0$
pulse $b = 1$
pulse $c = 0$
pulse $d = 1$
pulse $e = 0$
pulse $f = 0$
pulse $g = 1$

9.83 A negative-edge-triggered flip-flop transfers data from input to outputs on the _____ (leading, trailing) edge of the clock pulse. *Ans.* trailing

9.84 A positive-edge-triggered flip-flop transfers data from input to outputs on the _____ (H-to-L, L-to-H) transition of the clock pulse. *Ans.* L-to-H

9.85 Refer to Fig. 9-21. List the *binary* output (at $\bar{Q}$) of the master-slave *JK* flip-flop *after* each of the eight clock pulses.

Ans. pulse $a = 0$ pulse $c = 0$ pulse $e = 1$ pulse $g = 1$
pulse $b = 1$ pulse $d = 1$ pulse $f = 0$ pulse $h = 0$

9.86 Refer to Fig. 9-21. List the mode of operation of the *negative-edge-triggered JK* flip-flop for each of the clock pulses.

Ans. pulse a = set; pulse b = reset; pulse c = toggle; pulse d = toggle; pulse e = hold; pulse f = hold (J and $K = 0$ during H-to-L pulse); pulse g = reset; pulse h = toggle

9.87 A flip-flop is also referred to as a(n) _____ multivibrator. *Ans.* bistable

9.88 A free-running clock is also referred to as a(n) _____ multivibrator. *Ans.* astable

9.89 One-shots are also called _____. *Ans.* monostable multivibrators

9.90 The main advantage of a crystal-controlled clock is its _____ stability. *Ans.* frequency

9.91 If a free-running MV has a clock cycle time of 0.000001 s, the frequency of the clock is _____. *Ans.* 1 MHz

9.92 The output of the astable MV shown in Fig. 9-25 _____ (is, is not) TTL-compatible. *Ans.* is not (voltage is too high.)