4 DIGITAL CIRCUITS

CHAPTER

LEARNING OBJECTIVES

After studying this chapter, the student should be able to:

- Differentiate between digital and analog circuits.
- Understand the application of number systems in digital circuits.
- Apply arithmetic rules for addition, subtraction, multiplication, and division of binary numbers.
- Relate the use of 1's complement and 2's complement methods to find actual differences with minor manipulations.
- Discuss Binary Coded Decimal (BCD).
- Explain Boolean algebra.
- Outline the role of logic gates in the make-up of a digital system.
- Relate the use of logic gates as the fundamental building blocks of combinational logic circuits.
- Practice Karnaugh-map technique for representing logical functions.
- List some common combinational circuits.
- Discuss sequential circuits.
- Explain the functionalities of (A/D) and (D/A) converters.
- Categorize different logical convertors.

4.1 INTRODUCTION

A digital circuit is different from analog circuits. The term *digital* is derived from the way in which circuits perform operations by counting digits. Applications of digital electronics are not only confined to computer systems but are applied in many diverse areas like telephony, data processing, radar navigation, military systems, medical instruments and consumer products. Digital technology has progressed from vacuum-tube circuits to integrated circuits and microprocessors. Digital circuits involve systems in which there are only two possible states. These states are typically represented by voltage levels. Other circuit conditions, such as current levels, open or closed switches, and ON or OFF lamps, can also represent the two states. In digital systems, the two states are used to represent numbers, symbols, alphabetic characters, and other types of information.

4.2 NUMBER SYSTEM

In our daily life, the decimal number system (0, 1, 2, …, 9) is commonly used even though there are many other number systems like binary, octal, hexadecimal, etc. For understanding the digital circuits and systems, one

FLASH CARD

What You Learn Here
Various number systems like binary, octal, hexadecimal, decimal and analyze various conversions like decimal to binary, octal to binary, hexadecimal to binary, and hexadecimal to octal.

must be familiar with these number systems also. The following sections are dedicated to binary, octal and hexadecimal number systems.

4.2.1 Binary Numbers

PROBE

What is the weight of each successive higher position in a binary number?

The binary number system is simple because it is composed of only two digits, i.e., 0 and 1. Just as the decimal system, with its ten digits, is a base-ten system, the binary system with its two digits is a base-two system. The position of 1 or 0 in a binary number indicates its "weight" within the number. The weight of each successively higher position (to the left) in a binary number is an increasing power of two.

For example, in the decimal system,

$$139_{10} = 1 \times 10^2 + 3 \times 10^1 + 9 \times 10^0$$

hundreds tens units Positional weights

Similarly, the binary numbers are also represented by positional weights. For example,

$$\begin{aligned} 139_{10} &= 10001011_2 \\ &= 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 128 + 0 + 0 + 0 + 8 + 0 + 2 + 1 \\ &= 139. \end{aligned}$$

In the digital system, each of the binary digits is called a *bit* and a group of bits having a significance is called a *byte* or *word*. The highest decimal number that can be represented by an n bits binary number is $2^n - 1$. Thus, with a 8-bit binary number the maximum decimal number that can be represented is $2^8 - 1 = 255$.

4.2.2 Decimal–Binary Conversion

PROBE

Define the double-dabble method.

One easier method of converting a decimal number into binary number is to divide progressively the decimal number by 2 until quotient of zero is obtained. Writing the remainders after each division in the reverse order, the binary number is obtained. This method is popularly known as *double-dabble* method. The procedure for converting the decimal to binary is explained below. Consider the decimal number 52; its binary equivalent can be obtained as follows.

52 divided by 2 = quotient 26 with a remainder of **0**
26 divided by 2 = quotient 13 with a remainder of **0**
13 divided by 2 = quotient 6 with a remainder of **1**
6 divided by 2 = quotient 3 with a remainder of **0**
3 divided by 2 = quotient 1 with a remainder of **1**
1 divided by 2 = quotient 0 with a remainder of **1**

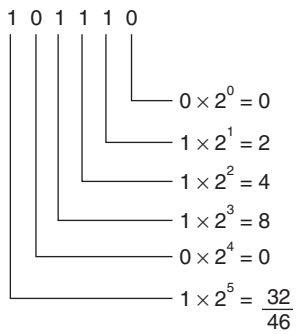
Reading the remainders from bottom to top gives the binary equivalent. Thus, $52_{10} = 110100_2$.

If the decimal number is a fraction, its binary equivalent is obtained by multiplying the number continuously by 2, recording each time a carry in the integer position. The carries in the forward order give the required binary number. For example, the decimal number 0.625_{10} can be expressed in binary as follows.

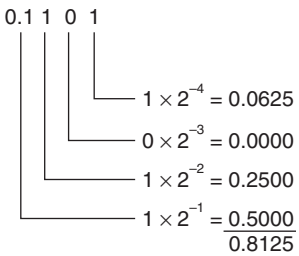
0.625 multiplied by 2 = **1**.250 : carry = **1**
0.250 × 2 = **0**.50 : carry = **0**
0.500 × 2 = **1**.00 : carry = **1**
0.000× 2 = **0**.00

As the product is zero, no further multiplication by two is possible. The binary equivalent is obtained by reading the carry terms from top to bottom. Thus, 0.625_{10} is 0.101_2. The binary equivalent of a decimal real number, which includes both integer and fractional parts, can be found using the same techniques, i.e., find the binary equivalent of the integer part and fractional part using respective methods and the combined number will give the binary equivalent. For example, $52.625_{10} = 110100.101_2$.

The conversion of a binary number into its decimal equivalent can also be carried out. In a binary number, digits from extreme right represent coefficient of the ascending powers of two, the starting power being zero. For example,



Therefore, $(101110)_2$ can be written as $(46)_{10}$. Conversion of fractions is also carried out in a similar manner. For example, the conversion of 0.1101 is illustrated below.



Thus, 0.1101_2 is equivalent to 0.8125_{10}.

4.2.3 Octal Numbers

The octal number system uses the digits 0, 1, 2, 3, 4, 5, 6, and 7. The base or radix of this system is eight. Each significant position in an octal number has a positional weight. The least significant position has a weight of 8^0, i.e., 1, and the higher significant positions are, respectively, given weightage as the ascending powers of eight, i.e., 8^1, 8^2, 8^3, etc. The octal equivalent of a decimal number can be obtained by dividing the decimal number by 8 repeatedly, until a quotient of 0 is obtained. The procedure is exactly same as the double-dabble method explained earlier. The conversion from decimal to octal number is explained hereunder.

EXAMPLE 4.1

Convert 12_{10} to an octal number.

Solution The procedure is as follows.

12 divided by 8 = quotient 1 with a remainder of **4**

1 divided by 8 = quotient 0 with a remainder of **1**

Reading the remainders from bottom to top, the decimal number 12_{10} is equivalent to octal 14_8.

The conversion from octal to decimal number can be carried out by multiplying each significant digit of the octal number by the respective weights and adding the products. The following example illustrates the conversion from octal to decimal.

EXAMPLE 4.2

Convert the following octal numbers to decimals: (a) 444_8 (b) 237_8 and (c) 120_8.

Solution

(a) $444_8 = 4 \times 8^2 + 4 \times 8^1 + 4 \times 8^0$
$= 4 \times 64 + 4 \times 8 + 4 \times 1$
$= 256 + 32 + 4$
$= 292_{10}$

(b) $237_8 = 2 \times 8^2 + 3 \times 8^1 + 7 \times 8^0$
$= 2 \times 64 + 3 \times 8 + 7 \times 1$
$= 128 + 24 + 7$
$= 159_{10}$

(c) $120_8 = 1 \times 8^2 + 2 \times 8^1 + 0 \times 8^0$
$= 1 \times 64 + 2 \times 8 + 0 \times 1$
$= 64 + 16 + 0$
$= 80_{10}$

4.2.4 Octal–Binary Conversions

PROBE

How can the conversion from octal to binary be carried out?

Conversion from octal to binary and vice-versa can be easily carried out. For obtaining the binary equivalent of an octal number, just replace each significant digit in the given number by its 3-bit binary equivalent. For example,

$276_8 = \quad 2 \quad 7 \quad 6$
$= \quad 010 \quad 111 \quad 110$

Thus, $276_8 = 010\ 111\ 110_2$. The reverse procedure is used for converting binary to octal, i.e., starting from the least significant bit, replace each group of 3 bits by their decimal equivalents. For example,

$11011010101_2 = \quad 11 \quad 011 \quad 010 \quad 101$

$= \quad 3 \quad 3 \quad 2 \quad 5$

Thus, $11011010101_2 = 3325_8$.

4.2.5 Hexadecimal Numbers

The hexadecimal number system has a radix of 16 and uses 16 symbols, namely, 1, 2, 3, 4, 5, 6, 7, 8, 9, *A*, *B*, *C*, *D*, *E*, and *F*. The symbols *A*, *B*, *C*, *D*, *E*, and *F* represent the decimals 10, 11, 12, 13, 14, and 15, respectively. Each significant position in hexadecimal number has a positional weight. The least significant position has a weight of 16^0, i.e., 1, and the higher significant positions are, respectively, given weightage as the ascending powers of sixteen, i.e., 16^1, 16^2, 16^3, etc. The hexadecimal equivalent of a decimal number can be obtained by dividing the decimal number by 16 repeatedly, until a quotient of 0 is obtained. The following example illustrates how the hexadecimal equivalent of a given decimal can be obtained.

PROBE

What is the weight of least signification positions?

EXAMPLE 4.3

Convert 112_{10} and 253_{10} to hexadecimal numbers.

Solution The procedure is as follows,

(a) 112 divided by 16 = quotient 7 with a remainder of **0**
7 divided by 16 = quotient 0 with a remainder of **7**
Reading the remainders from bottom to top, the decimal number 112_{10} is equivalent to hex 70_{16}. The hexadecimal 70_{16} can also be represented as 70 H.

(b) 253 divided by 16 = quotient 15 with a remainder of **13**, i.e., *D*
15 divided by 16 = quotient 0 with a remainder of **15**, i.e., *F*
Reading the remainders from bottom to top, the decimal number 253_{10} is equivalent to hex FD_{16}.

The conversion from hexadecimal to decimal number can be carried out by multiplying each significant digit of the hexadecimal number by the respective weights and adding the products. The following example illustrates the conversion from hexadecimal to decimal number.

EXAMPLE 4.4

Convert the following hexadecimal numbers to decimals. (a) 4AB H (b) 23F H

Solution

(a) $4AB_{16} = 4 \times 16^2 + A \times 16^1 + B \times 16^0$
$= 4 \times 16^2 + 10 \times 16^1 + 11 \times 16^0$
$= 4 \times 256 + 10 \times 16 + 11 \times 1$
$= 1024 + 160 + 11$
$= 1195_{10}$

(b) $23F_{16} = 2 \times 16^2 + 3 \times 16^1 + F \times 16^0$
$= 2 \times 256 + 3 \times 16 + 15 \times 1$
$= 512 + 48 + 15$
$= 575_{10}$

4.2.6 Hexadecimal–Binary Conversions

Conversion from hex to binary and vice versa can be easily carried out. For obtaining the binary equivalent of a hexadecimal number, replace each significant digit in the given number by its 4-bit binary equivalent. For example,

$$\begin{aligned} 2A6_{16} &= \quad 2 \quad A \quad 6 \\ &= \quad 0010 \quad 1010 \quad 0110 \end{aligned}$$

Thus, $2A6_{16} = 0010\ 1010\ 0110_2$. The reverse procedure is used for converting binary to hex, i.e., starting from the least significant bit, replace each group of 4 bits by their decimal equivalents. For example,

$$\begin{aligned} 11011010101_2 &= \quad 110 \quad 1101 \quad 0101 \\ &= \quad 6 \quad D \quad 5 \end{aligned}$$

Thus, $11011010101_2 = 6D5_{16}$

4.2.7 Hexadecimal–Octal Conversions

PROBE

Give the steps used in the conversion between hexadecimal and octal numbers.

Conversion between hexadecimal and octal numbers is sometimes required. To convert a hexadecimal number to octal, the following steps can be used.

1. Convert the given hexadecimal number to equivalent binary.
2. Starting from the LSB, form groups of 3 bits.
3. Write the equivalent octal number for each group of 3 bits. For example,

$$\begin{aligned} 24_{16} &= 0010\ 0100_2 \\ &= 00\ 100\ 100_2 \\ &= 044_8 \end{aligned}$$

Thus, 24 in hexadecimal is equivalent to 44 in octal number system.

To convert an octal number to hexadecimal the steps are as follows.

1. Convert the given octal number to equivalent binary.
2. Starting from the LSB, form groups of 4 bits.
3. Write the equivalent hexadecimal number for each group of 4 bits. For example,

$$\begin{aligned} 24_8 &= 010\ 100_2 \\ &= 01\ 0100_2 \\ &= 14_{16} \end{aligned}$$

Thus, 24 in octal is equivalent to 14 in hexadecimal number system.

4.3 BINARY ARITHMETIC

FLASH CARD

What You Learn Here
Analyze binary arithmetic by using binary addition, subtraction, multiplication, and division.

The arithmetic rules for addition, subtraction, multiplication, and division of binary numbers are given below:

	Addition	*Subtraction*	*Multiplication*	*Division*
1.	$0 + 0 = 0$	$0 - 0 = 0$	$0 \times 0 = 0$	$0 \div 1 = 0$
2.	$0 + 1 = 1$	$1 - 0 = 1$	$0 \times 1 = 0$	$1 \div 1 = 1$

3. $1 + 0 = 1$ $1 - 1 = 0$ $1 \times 0 = 0$ $0 \div 0 =$ not allowed
4. $1 + 1 = 10$ $10 - 1 = 1$ $1 \times 1 = 1$ $1 \div 0 =$ not allowed

4.3.1 Binary Addition

Two binary numbers can be added in the same way as two decimal number are added. The addition is carried out from the least significant bits and proceeded to higher significant bits, adding the carry resulting from previous addition each time. Consider the addition of the binary number 1101 and 1111.

MSB LSB	Decimal
1101	13
1111	15
11100	28

The addition carried out above can be explained as follows.

Step 1: The least significant bits are added, i.e., 1 + 1 = 0 with a carry 1.

Step 2: The carry in the previous step is added to the next higher significant bits, i.e., 1 + 0 + 1 = 0 with a carry 1.

Step 3: The carry in the previous step is added to the next higher significant bits, i.e., 1 + 1 + 1 = 1 with a carry 1.

Step 4: The carry in the previous step is added to the most significant bit, i.e., 1 + 1 + 1 = 1 with a carry 1.

Thus, the sum is 11100. The addition is also shown in decimal number system in order to compare the results.

4.3.2 Binary Subtraction

PROBE

Apply binary subtraction when a 1 is subtracted from a 0, and a 1 is borrowed from an immediate higher significant bit.

Binary subtraction is also carried out in the same way as decimal numbers are subtracted. The subtraction is carried out from the least significant bits and proceeded to higher significant bits. When a 1 is subtracted from a 0, a 1 is borrowed from immediate higher significant bit. The following problem explains the steps involved. Suppose that 1001 is subtracted from 1110.

MSB LSB	Decimal
1110	14
1001	9
0101	5

The steps are explained below.

Step 1: The least significant bits (1 column) are considered. 0 – 1 needs a borrow from the next higher significant bit (column 2 from the right). Thus, 0 – 1 results in a difference of **1** and borrow 1.

Step 2: The second column is taken. Since a 1 is given to the first column, now change the 1 here to 0. Thus, the subtraction to be performed is 0 – 0 = **0**

Step 3: In the third column, the difference is given by 1 – 0 = **1**

Step 4: In the fourth column (MSB), the difference is given by $1 - 1 = \mathbf{0}$

Thus, the difference between the two binary numbers is 0101.

4.3.3 Binary Multiplication

Binary multiplication is rather simpler than decimal multiplication. The procedure is same as that of decimal multiplication. The binary multiplication procedure is as shown.

Step 1: The least significant bit of the multiplier is taken. If the multiplier bit is 1, the multiplicand is copied as such and if the multiplier bit is 0, 0 is placed in all the bit positions.

Step 2: The next higher significant bit of the multiplier is taken and as in step 1, the product is written with a shift in left.

Step 3: Step 2 is repeated for all other higher significant bits and everytime a left shift is given.

Step 4: When all the bits in the multiplier have been taken into account, the product terms are added, which gives the actual product of the multiplier and the multiplicand. The following examples illustrates the multiplication procedure.

PROBE

Illustrate an example of binary multiplication.

EXAMPLE 4.5

Multiply the following binary numbers. (a) 1101 and 1100 (b) 1000 and 101 (c) 1111 and 1001

Solution

(a) 1101×1100

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ \times\ 1\ 1\ 0\ 0 \\ \hline 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0 \\ 1\ 1\ 0\ 1 \\ 1\ 1\ 0\ 1 \\ \hline 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0 \\ \hline \end{array}$$

(b) 1000×101

$$\begin{array}{r} 1\ 0\ 0\ 0 \\ \times\ \ 1\ 0\ 1 \\ \hline 1\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0 \\ \hline 1\ 0\ 1\ 0\ 0\ 0 \\ \hline \end{array}$$

(c) 1111×1001

$$\begin{array}{r} 1\ 1\ 1\ 1 \\ \times\ 1\ 0\ 0\ 1 \\ \hline 1\ 1\ 1\ 1 \\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0 \\ 1\ 1\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 0\ 1\ 1\ 1 \\ \hline \end{array}$$

4.3.4 Binary Division

Division in binary follows the same procedure as division in decimal. Division by 0 is meaningless. An example is given below.

EXAMPLE 4.6

Perform the following divisions: (a) 110 ÷ 10 (b) 1111 ÷ 110

Solution

(a) $110 \div 10$

$$\begin{array}{r} 11_2 \\ 10\overline{)110} \\ 10 \\ \hline 10 \\ 10 \\ \hline 00 \end{array} \qquad \begin{array}{r} 3_{10} \\ 2\overline{)6} \\ 6 \\ \hline 0 \end{array}$$

(b) $1111 \div 110$

$$\begin{array}{r} 10.1_2 \\ 110\overline{)1111.0} \\ 110 \\ \hline 110 \\ 110 \\ \hline 000 \end{array} \qquad \begin{array}{r} 2.5_{10} \\ 6\overline{)15.0} \\ 12 \\ \hline 30 \\ 30 \\ \hline 00 \end{array}$$

4.4 1'S AND 2'S COMPLEMENTS

FLASH CARD

What You Learn Here
Infer the application of 1's complement, 2's complement, 9's complement, and 10 's complement subtractions.

The usefulness of the complement numbers stems from the fact that subtraction of a number from another can be accomplished by adding the complement of the subtrahend to the minuend. The actual difference can be obtained with minor manipulations.

4.4.1 1's Complement Subtraction

PROBE

How can subtraction of binary numbers be accomplished by using the 1's complement method?

Subtraction of binary numbers can be accomplished by using the 1's complement method, which allows us to subtract using only addition. The 1's complement of a binary number is found by simply changing all 1s to 0s and all 0s to 1s. To subtract a smaller number from a larger number, the 1's complement method is as follows:

1. Determine the 1's complement of the smaller number.
2. Add the 1's complement to the larger number.
3. Remove the carry and add it to the result. This carry is called ***end-around-carry***.

EXAMPLE 4.7

Subtract 1010_2 from 1111_2 using 1's complement method. Show direct subtraction for comparison.

Solution

Direct subtraction		1's complement method
1 1 1 1		1 1 1 1
–1 0 1 0	1's comp. →	0 1 0 1
0 1 0 1	carry	**1** 0 1 0 0
	add carry	**1**
		0 1 0 1

To subtract a larger number from a smaller one, the 1's complement method is as follows:

1. Determine the 1's complement of the larger number.
2. Add the 1's complement to the smaller number.
3. The answer has an opposite sign and is the 1's complement of the result. There is no carry.

EXAMPLE 4.8

Subtract 1010_2 from 1000_2 using the 1's complement method. Show direct subtraction for comparison.

Solution

Direct subtraction		1's complement method
1 0 0 0		1 0 0 0
–1 0 1 0	1's comp.→	0 1 0 1
– 0 0 1 0		1 1 0 1

No carry results and the answer is the 1's complement of 1101 and opposite in sign, i.e., – 0010.

The 1's complement method is particularly useful in arithmetic logic circuits because subtraction can be accomplished with an adder.

4.4.2 2's Complement Subtraction

PROBE

How is the 2's complement method applied?

The 2's complement of a binary number is found by adding 1 to its 1's complement. To subtract a smaller number from a larger one, the 2's complement method is applied as follows:

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

EXAMPLE 4.9

Subtract 1010_2 from 1111_2 using the 2's complement method. Show direct subtraction for comparison.

Solution

Direct subtraction		2's complement method
1 1 1 1		1 1 1 1
–1 0 1 0	2's comp.→	0 1 1 0
0 1 0 1	carry	**1** 0 1 0 1

The carry is discarded. Thus, the answer is 0101_2.

To subtract a larger number from a smaller one, the 2's complement method is as follows:

1. Determine the 2's complement of the larger number.
2. Add the 2's complement to the smaller number.
3. There is no carry. The result is in 2's complement form and is negative.
4. To get an answer in true form, take the 2's complement and change the sign.

EXAMPLE 4.10

Subtract 1010_2 from 1000_2 using the 2's complement method. Show direct subtraction for comparison.

Solution

Direct subtraction		2's complement method
1 0 0 0		1 0 0 0
–1 0 1 0	2's comp.→	0 1 1 0
0 0 1 0	no carry	1 1 1 0

No carry results. Thus, the difference is negative and the answer is the 2's complement of 1110_2, i.e., 0010_2.

Both 1's and 2's complement methods of subtraction may seem more complex compared to direct subtraction. However, they both have distinct advantages when implemented with logic circuits because they allow subtraction to be done using addition. Both 1's and the 2's complements of a binary number are relatively easy to accomplish with logic circuits; and the 2's complement has an advantage over the 1's complement in that an end-around-carry operation does not have to be performed.

4.5 BINARY CODED DECIMAL

FLASH CARD

What You Learn Here
Expand Binary coded Decimal (BCD) and BCD addition.

When a computer is handling numbers in binary but in group of four digits, the number system is called Binary Coded Decimal (BCD). Combinations of binary digits that represent numbers, letters, or symbols are digital codes. The 8421 code is a type of binary coded decimal code. It has four bits and represents the decimal digits 0 through 9. The designation 8421 indicates the binary weights of the four bits. The case of conversion between 8421 code numbers and the familiar decimal numbers is the main advantage of this code. To express any decimal number in BCD, simply replace each decimal digit by the appropriate four-bit code. Table 4.1 gives the binary and BCD for the decimal numbers 0 through 15.

PROBE

Summarize binary and BCD for the decimal numbers.

Table 4.1 *Decimal numbers, equivalent binary, and BCD*

Decimal number	Binary number	Binary coded decimal (8421)
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111

8	1000	1000	
9	1001	1001	
10	1010	0001	0000
11	1011	0001	0001
12	1100	0001	0010
13	1101	0001	0011
14	1110	0001	0100
15	1111	0001	0101

4.5.1 BCD Addition

BCD is a numerical code, and many applications require that arithmetic operations be performed. Addition is the most important operation because the other three operations like subtraction, multiplication, and division can be accomplished using addition. The rule for adding two BCD numbers is given below.

PROBE

What are the rules for adding two BCD numbers?

1. Add the two numbers, using the rules for binary addition.
2. If a four-bit sum is equal to or less than 9, it is a valid BCD number.
3. If a four-bit sum is greater than 9, or if a carry-out of the group is generated, it is an invalid result. Add 6 (0110_2) to the four-bit sum in order to skip the six invalid states and return the code to 8421. If a carry results when 6 is added, simply add the carry to the next four-bit group.

EXAMPLE 4.11

Add the following BCD numbers:

Solution

(a)

```
                1 0 0 1                          9
              + 0 1 0 0                        + 4
              -------                          ----
                1 1 0 1   Invalid BCD number   13(10)
              + 0 1 1 0   Add 6
              -------
  0 0 0 1 0 0 1 1 1       Valid BCD number
      ↓       ↓
      1       3
```

(b)

```
  0 0 0 1 1 0 0 1                              19
+ 0 0 0 1 0 1 0 0                            + 14
-----------------                            ----
  0 0 1 0 1 1 0 1   Right group is invalid   33(10)
              + 0 1 1 0   Add 6
              -------
  0 0 1 1 0 0 1 1 1       Valid BCD number
      ↓       ↓
      3       3
```

4.6 BOOLEAN ALGEBRA

Boolean algebra is a set of rules, laws, and theorems by which logical operations can be expressed mathematically. It is a convenient and systematic way of expressing and analyzing the operation of digital circuits and systems. In Boolean algebra, a variable can be either a zero or a one. The binary digits are utilized to represent the two levels that occur within digital logic circuits. A binary 1 will represent a HIGH level and a binary 0 will represent a LOW level. The complement of a variable is represented by a 'bar' over the letter; for example, the complement of A is represented by $\overline{A}$.

FLASH CARD

What You Learn Here
Define and test Boolean algebra for the expression of logical operations using Boolean addition and multiplication, De Morgan's theorems, and algebraic simplification method.

4.6.1 Boolean Addition and Multiplication

PROBE

What are the basic rules of addition?

Boolean variables takes values of either a binary 1 or a 0. The basic rules for Boolean addition are as follows:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 1$$

Boolean addition is the same as the logical OR operation. The multiplication rules for Boolean algebra are the same as the binary multiplication rules, discussed in Section 4.3.3.

$$0.0 = 0$$
$$0.1 = 0$$
$$1.0 = 0$$
$$1.1 = 1$$

Boolean multiplication is the same as the logical AND operation. The three basic properties of Boolean algebra are commutativity, associativity, and distributivity.

- ***Commutative Property*** Boolean addition is commutative, i.e.,

$$A + B = B + A$$

PROBE

Compare commutative, associative, and distributive properties.

This says that the order in which the variables are ORed makes no difference. Boolean algebra is also commutative over multiplication, i.e.,

$$A \cdot B = B \cdot A$$

This states that the order in which the variables are ANDed makes no difference.

- ***Associative Property*** The associative property of addition is stated as follows:

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

The ORing of several variables results in the same regardless of the grouping of the variables. The associative law of multiplication is stated as follows:

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

This law tells us that it makes no difference in what order the variables are grouped when ANDing several variables.

- ***Distributive Property*** Boolean addition is distributive over Boolean multiplication. That is,

$$A + BC = (A + B)\,(A + C)$$

Also, Boolean multiplication is distributive over Boolean addition, i.e.,

$$A \cdot (B + C) = A.B + A.C$$

The first distributive property states that ANDing several variables and ORing the result with a single variable is equivalent to ORing the single variable with each of the several variables and then ANDing the sums. The second distributive property states that ORing several variables and ANDing the result with a single variable is equivalent to ANDing the single variable with each of the several variables and then ORing the products. The other basic laws of Boolean algebra are given in Table 4.2.

Table 4.2 *Laws of Boolean algebra*

Sl. No.	Boolean Law
1	$A + 0 = A$
2	$A + 1 = 1$
3	$A \,.\, 0 = 0$
4	$A \,.\, 1 = A$
5	$A + A = A$
6	$A + \overline{A} = 1$
7	$A \,.\, A = A$
8	$A \,.\, \overline{A} = 0$
9	$\overline{\overline{A}} = A$
10	$A + AB = A$
11	$A + \overline{A}B = A + B$
12	$AB + \overline{A}C + BC = AB + \overline{A}C$

The rules 1 to 9 can be very easily proved by perfect induction. The proofs for the rules 10, 11, and 12 are given as follows.

EXAMPLE 4.12

Prove the following laws of Boolean algebra. (a) $A + AB = A$ (b) $A + \overline{A}B = A + B$ (c) $AB + \overline{A}C + BC = AB + \overline{A}C$.

Solution

(a) $A + AB$ $= A\,(1 + B)$ distributive law
$= A \,.\, 1$ law 2
$= A$ law 4

(b) $A + \overline{A}B$ $= (A + \overline{A}) \,.\, (A + B)$ distributive law
$= 1 \,.\, (A + B)$ law 6
$= A + B$ law 4

(c) $AB + \bar{A}C + BC = AB + \bar{A}C + BC1$

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

$= AB + \bar{A}C + ABC + \bar{A}BC$

$= AB\,(1 + C) + \bar{A}C\,(1 + B)$

$= AB + \bar{A}C$

The above property, i.e., $AB + \bar{A}C + BC = AB + \bar{A}C$, is called *consensus theorem*.

4.6.2 De Morgan's Theorems

Two theorems that are important parts of Boolean algebra were proposed by De Morgan. The first theorem states that the complement of a product is equal to the sum of the complements. That is, if the variables are A and B, then

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

The second theorem states that the complement of a sum is equal to the product of the complements. In equation form, this can be written as

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

The complement of a Boolean logic function or a logic expression may be determined by using the following steps of De Morgan's theorem.

1. Replace the symbol (+) with symbol (.), the symbol (.) with symbol (+) given in the expression.
2. Complement each of the terms or variable in the given expression.

4.6.3 Algebraic Simplification of Logical Expressions

PROBE

How are the basic rules and laws of Boolean algebra used?

In the application of Boolean algebra, one has to reduce a particular expression to its simplest form or change its form to a more convenient one in order to implement the expression most efficiently. The basic rules and laws of Boolean algebra are used to manipulate and simplify an expression. The algebraic simplification method requires a thorough knowledge of Boolean algebra and considerable practice in its application. Several examples are given below to illustrate the technique.

EXAMPLE 4.13

Simplify the following expressions using Boolean algebra: (a) $A + AB + A\bar{B}C$ (b) $(\bar{A} + B)C + ABC$ (c) $A\bar{B}C\,(BD + CDE) + A\bar{C}$

Solution

(a) $A + AB + A\bar{B}C$

Step 1: Apply Rule 10 of Table 4.2, i.e., $A + AB = A$. The expression simplifies to

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

Step 2: Apply distributive property.

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

Step 3: Taking A as the common term,

$$A[1 \cdot (1 + \bar{B}C)]$$

Step 4: Apply Rule 2 of Table 4.2, i.e., $1 + \overline{B}C = 1$

$$A.1 = A$$

Thus, the simplified expression is A.

(b) $$(\overline{A} + B)C + ABC$$

Step 1: Apply distributive property

$$\overline{A}C + BC + ABC$$

Step 2: Taking BC as a common term,

$$\overline{A}C + BC(1 + A)$$

Step 3: Apply Rule 2.

$$\overline{A}C + BC \cdot 1$$

Step 4: Taking C as the common term,

$$C(\overline{A} + B)$$

Thus, the simplified expression is $C(\overline{A} + B)$.

(c) $$A\overline{B}C\,(BD + CDE) + A\overline{C}$$

Step 1: Apply distributive property.

$$A\overline{B}\ BCD + A\overline{B}CCDE + A\overline{C}$$

Step 2: Apply Rules 8 and 7 to the first and second terms, respectively.

$$0 + A\overline{B}CDE + A\overline{C}$$

Step 3: Taking A as the common term,

$$A\,(\overline{B}CDE + \overline{C})$$

Step 4: Apply Rule 11, i.e., $\overline{B}\ CDE + \overline{C} = \overline{B}DE + \overline{C}$

$$A\,(\overline{B}DE + \overline{C})$$

Thus, the simplified expression is $A\,(\overline{B}DE + \overline{C})$

4.7 LOGIC GATES

The basic elements that make up a digital system are called logic gates. The most common logic gates are OR, AND, NOT, NAND and NOR gates. NAND and NOR gates are called universal gates. Exclusive-OR gate is another logic circuit which can be constructed using AND, OR and NOT gates.

FLASH CARD

What You Learn Here

List the most common logic gates like OR, AND, NOT, NAND, and NOR and discuss their construction. Compare between Exclusive-OR and NOR gate.

4.7.1 OR Gate

PROBE

How many inputs and outputs does the OR gate have?

The OR gate performs logical addition, commonly known as OR function. The OR gate has two or more inputs and only one output. The operation of OR gate is such that a high (1) on the output is produced when any of the inputs is high (1). The output is low (0) only when all the inputs are low (0).

As shown in Fig. 4.1, A and B represent the inputs and Y, the output. Resistance R is the load resistance.

If $A = 0$ and $B = 0$ then $V_0 = 0$ and $Y = 0$.

If $A = 1$ and $B = 0$, diode D_1 will conduct and so the output $Y = 1$.

If $A = 0$ and $B = 1$, diode D_2 will conduct and the output $Y = 1$.

If $A = 1$ and $B = 1$, both the diodes will conduct and so the output $Y = 1$.

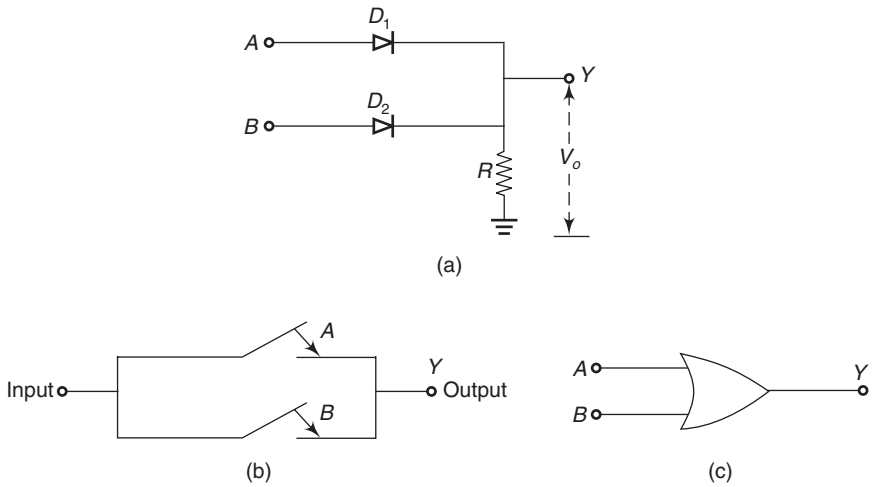


Fig. 4.1 *(a) Circuit diagram of an OR gate (b) Electrical equivalent of an OR gate (c) Logic symbol*

The electrical equivalent circuit of an OR gate is shown in Fig. 4.1(b) where switches A and B are connected in parallel with each other. If either A, B or both are closed, then the output will result. The logic symbol for the OR gate is shown in Fig. 4.1(c). The logic operation of the two-input OR gate is described in the truth table shown in Table 4.3.

Table 4.3 *Truth table for the two-input OR gate*

Input		Output
A	***B***	***Y***
0	0	0
0	1	1
1	0	1
1	1	1

4.7.2 AND Gate

The AND gate performs logical multiplication, commonly known as AND function. The AND gate has two or more inputs and a single output. The output of AND gate is high only when all the inputs are high. When any of the inputs is low, the output is low.

As shown in the Fig. 4.2(a), A and B represent the inputs and Y represents the output.

If $A = 0$ and $B = 0$, both diodes conduct as they are forward biased and the output $Y = 0$.

If $A = 0$ and $B = 1$, diode D_1 conducts and D_2 does not conduct, and again the output $Y = 0$.

If $A = 1$ and $B = 0$, diode D_1 does not conduct and D_2 conducts, and the output $Y = 0$.

If $A = 1$ and $B = 1$, both the diodes do not conduct as they are reverse based and so the output $Y = 1$.

The electrical equivalent circuit of an AND gate is shown in Fig. 4.2 (b) where two switches A and B are connected in series. If both A and B are closed, then only output will result. Logic symbol of the AND gate is shown in Fig. 4.2(c). The logic operation of the two-input AND gate is described in the truth table shown in Table 4.4.

PROBE

Create the logic symbol of the AND gate.

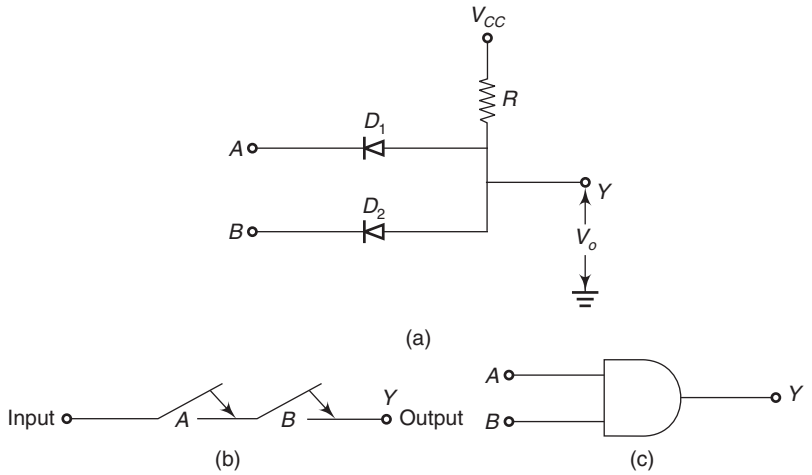


Fig. 4.2 *(a) Circuit diagram of an AND gate (b) Electrical equivalent of an AND gate (c) Logic symbol*

Table 4.4 *Truth table for a two-input AND gate*

Input		Output
A	*B*	*Y*
0	0	0
0	1	0
1	0	0
1	1	1

4.7.3 NOT Gate (Inverter)

The NOT gate performs a basic logic function called inversion or complementation. The purpose of the gate is to change one logic level to opposite level. It has one input and one output. When a high level is applied to an inverter input, a low level will appear at its output and vice

PROBE

What is the purpose of the NOT gate?

versa. The operation of the circuit can be explained as follows. When a high voltage is applied to the base of the transistor, base current increases and the transistor is saturated. The transistor now acts as a closed switch and conducts heavily. Thus, the output voltage is logic 0. On the other hand, when a low voltage is applied at the base, the transistor is cut-off due to very low or no base current. Now, the transistor can be considered as an open switch, with no current flowing through it. The output is now clamped to the supply voltage. The transistor when operated between cut-off and saturation will act as a switch.

As shown in Fig. 4.3(a), *A* represents the input and *y* represents the output. If the input is high, the transistor is in ON state and the output is low. If the input is low, the transistor is in OFF state and the output is high. The symbol for the inverter is shown in Fig. 4.3(b). The truth table is given in Table 4.5.

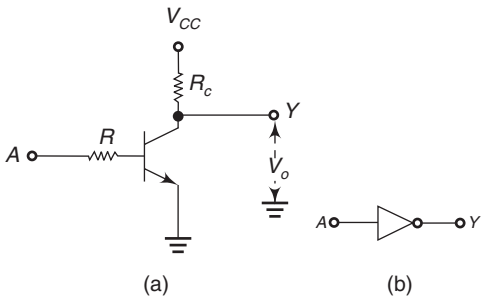


Fig. 4.3 *(a) Circuit diagram of an INVERTER GATE (b) Logic symbol*

Table 4.5 *Truth table for an INVERTER*

Input *A*	Output *Y*
0	1
1	0

4.7.4 NAND Gate

NAND is a contraction of NOT–AND. It has two or more inputs and only one output. When all the inputs are high, the output is low. If any of the inputs is low, the output is high. The logic symbol for the NAND gate is shown in Fig. 4.4.

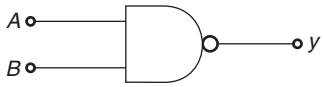


Fig. 4.4 *Logic symbol for the NAND gate*

The truth table for the NAND gate is shown in Table 4.6.

Table 4.6 *Truth table for NAND gate*

Input		Output
A	*B*	*Y*
0	0	1
0	1	1
1	0	1
1	1	0

The NAND gate is a very popular logic function because it is a universal function; that is, it can be used to construct an AND gate, an OR gate, and INVERTER or any combination of these functions. Figure 4.5 shows how NAND gates can be connected to realize various logic gates.

PROBE

What is the universal function of the NAND gate?

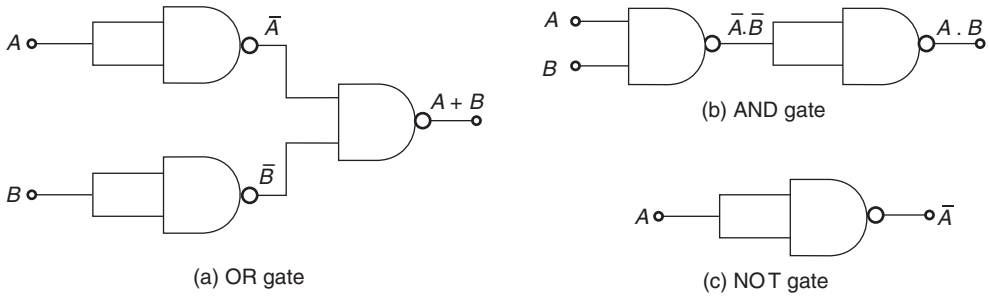


Fig. 4.5 *NAND gates connected to realize (a) OR, (b) AND, and (c) NOT*

4.7.5 NOR Gate

NOR is a contraction of NOT–OR. It has two or more inputs and only one output. Only when all the inputs are low, the output is high. If any of the inputs is high, the output is low. The logic symbol for the NOR gate is shown in Fig. 4.6.

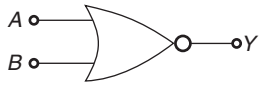


Fig. 4.6 *Logic symbol for NOR gate*

The truth table for the NOR gate is shown in Table 4.7.

Table 4.7 *Truth table for NOR gate*

Input		Output
A	***B***	***Y***
0	0	1
0	1	0
1	0	0
1	1	0

The NOR gate is also a very popular logic function because it is also a universal function; that is, it can be used to construct an AND gate, an OR gate, and INVERTER or any combination of these functions. Figure 4.7 shows how NOR gates can be connected to realize various logic gates.

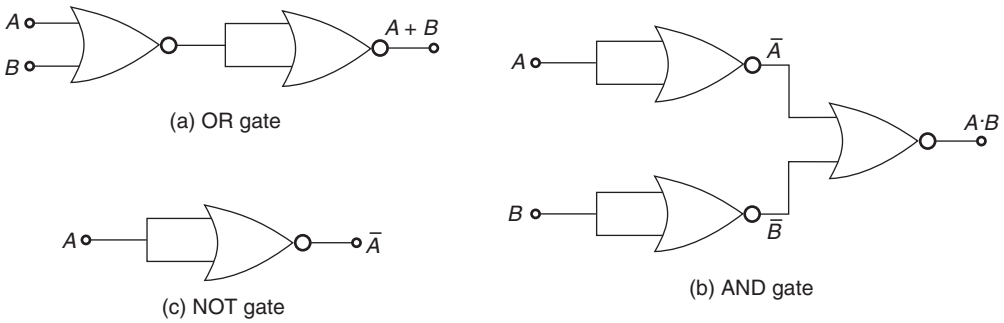


Fig. 4.7 *NOR gates connected to realize (a) OR, (b) AND, and (c) NOT*

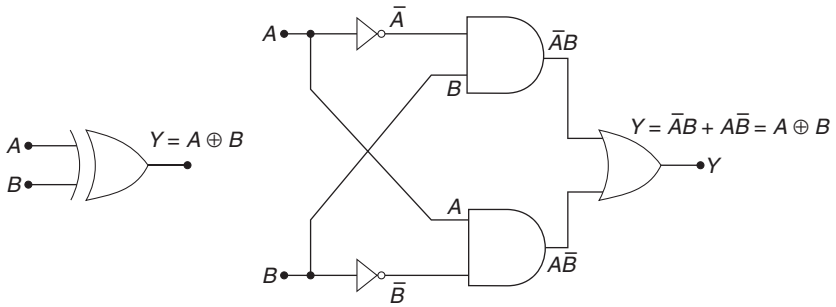
4.7.6 Exclusive-OR (Ex-OR) Gate

PROBE

What is an Exclusive-OR gate?

An Exclusive-OR gate is a gate with two or more inputs and one output. The output of a two-input Ex-OR gate assumes a HIGH state if one only one input assumes a HIGH state. This is equivalent to saying that the output is HIGH if either input A or input B is HIGH exclusively, and low when both are 1 or 0 simultaneously.

The logic symbol for the Ex-OR gate is shown in Fig. 4.8(a) and the truth table for the Ex-OR operation is given in Table 4.8.



(a) Logic symbol (b) Using AND-OR-NOT gates

Fig. 4.8 *Ex-OR Gate*

Table 4.8 *Truth table of a 2-input Ex-OR gate*

Input		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

The truth table of the Ex-OR gate shows that the output is HIGH when any one, but not all, of the inputs is at 1. This exclusive feature eliminates a similarity to the OR gate. The Ex-OR gate responds with a HIGH output only when an odd number of inputs is HIGH. When there is an even number of HIGH inputs, such as two or four, the output will always be LOW. From the truth table of a 2-input Ex-OR gate, the Ex-OR function can be written as $Y = \bar{A}B + A\bar{B} = A \oplus B$.

The above expression can be read as Y equals A Ex-OR B. Using the above expression, a 2-input

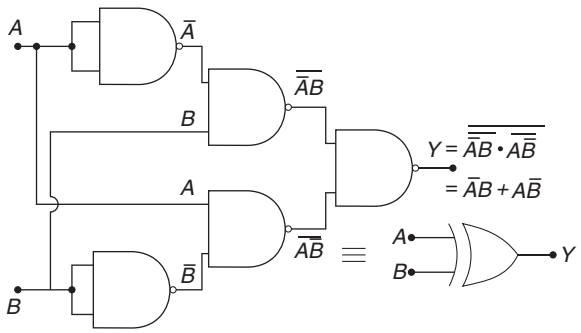


Fig. 4.9 *Ex-OR gates using NAND gates*

Ex-OR gate can be implemented using basic gates like AND, OR, and NOT gates as shown in Fig. 4.8(b). The 2-input Ex-OR gate can also be implemented using NAND gates as shown in Fig. 4.9.

The main characteristic property of an Ex-OR gate is that it can perform modulo-2 addition. It should be noted that the same Ex-OR truth table applies when adding two *binary digits* (bits). A 2-input Ex-OR circuit is, therefore, sometimes called a *module-2 adder* or a *half adder* without carry output. The name *half adder* refers to the fact that possible carry-bit, resulting from an addition of two preceding bits, has not been taken into account. A full addition is performed by a second Ex-OR circuit with the output signal of the first circuit and the carry as input signals, as shown in Fig. 4.10.

The configuration of Fig. 4.10 is a cascading of two Ex-OR circuits resulting in an Ex-OR operation of three variables A, B, and C. Consequently, the sum output of a full adder for two bits is an Ex-OR operation of the 2 bits to be added and the carry of the preceding adding stage.

The logic expression of the Ex-OR operation of three variables A, B, and C is given by

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

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

$$A \oplus B \oplus C = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C + ABC$$

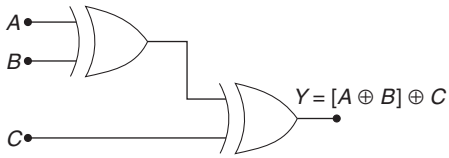


Fig. 4.10 *Cascading of two Ex-OR circuits*

In general, Ex-OR operation of n variables results in a logical 1 output if and *odd* number of the input variables are 1s. An Ex-OR operation of n variables can be obtained by cascading 2-input Ex-OR gates.

Another important property of an Ex-OR gate is that it can be used as a *controlled inverter*, i.e., by using an Ex-OR gate, a logic variable can be complemented or allowed to pass through it unchanged. This is done by using one Ex-OR input as a control input and the other as the logic variable input as shown in Fig. 4.11. When the control input is HIGH, the output $Y = \overline{A}$ and when the control input is LOW, the output $Y = A$.

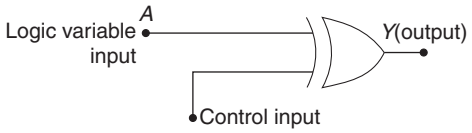


Fig. 4.11 *Ex-OR gate as a controlled inverter*

4.7.7 Exclusive-NOR (Ex-NOR) Gate

The exclusive-NOR gate, abbreviated Ex-NOR, is an Ex-OR gate, followed by an inverter. An exclusive-NOR gate has two or more inputs and one output. The output of a two-input Ex-NOR gate assumes a HIGH state if both the inputs assume the same logic state or have an even number of 1s, and its output is LOW when the inputs assume different logic states or have an odd number of 1s. The logic symbol of Ex-NOR gate is shown in Fig. 4.12 and its truth table is given in Table 4.9. From the truth table, it is clear that the Ex-NOR output is the complement of the Ex-NOR gate. The Boolean expression for the Ex-NOR gate is

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

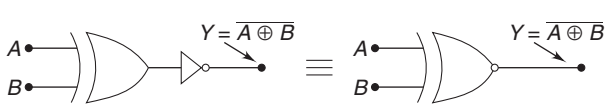


Fig. 4.12 *Logic symbol of 2-input Ex-NOR gate*

Table 4.9 *Truth table of 2-input Ex-OR gate*

Input		Output
A	*B*	$Y = \overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Read the above as "*Y* equals *A* Ex-NOR *B*". According to De Morgan's theorem,

$$\begin{aligned}\overline{A \oplus B} &= \overline{\overline{A}B + A\overline{B}} \\ &= \overline{\overline{A}B} \cdot \overline{A\overline{B}} \\ &= (A + \overline{B})(\overline{A} + B) \\ &= AB + \overline{A}\overline{B}\end{aligned}$$

An important property of the Ex-NOR gate is that it can be used for bit comparison. The output of an Ex-NOR gate is 1 if both the inputs are similar, i.e., both are 0 or 1; otherwise, its output is 0. Hence, it can be used as a one-bit comparator. It is also called a *coincidence circuit*.

PROBE

Write the different properties of the Ex- NOR gate.

Another property of the Ex-NOR gate is that it can be used as an *even-parity checker*. The output of the Ex-NOR gate is 1 if the number of 1s in its inputs is even; if the number of 1s is odd, the output is 0. Hence, it can be used as an even/odd parity checker. Hence, the 2-input Ex-NOR gate is immensely useful for bit comparison and parity checking.

EXAMPLE 4.14

Realize the logic expression $Y = \overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{C}$ using basic gates.

Solution In the given expression, there are 3 product terms, each with two variables, which can be implemented using three 2-input AND gates, and the product terms can be OR operated together using a 3-input OR gate. The complemented form of individual variable can be obtained by 3 NOT gates. Thus, the circuit for the given expression is realized as shown in Fig. 4.13.

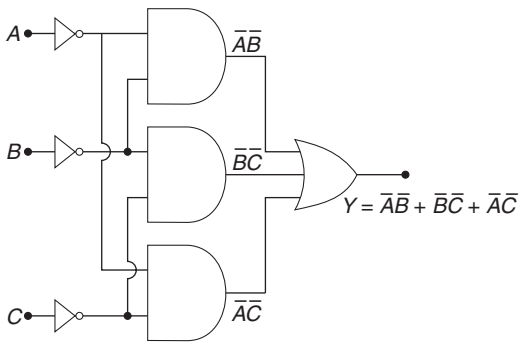


Fig. 4.13

EXAMPLE 4.15

Realize the logic expression $Y = (A + B)(\overline{A} + C)(B + D)$ using basic gates.

Solution In the given expression, there are 3 sum terms which can be implemented using three 2-input OR gates and their outputs are AND operated together by a 3-input AND gate. A NOT gate can be used to obtain the inverse of A. Now, the realized circuit is shown in Fig. 4.14.

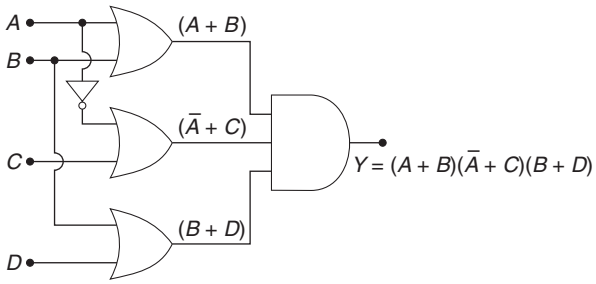


Fig. 4.14

EXAMPLE 4.16

Implement $Y = \overline{AB} + A + (\overline{B + C})$ using NAND gates only.

Solution The implementation of the given function is shown in Fig. 4.15.

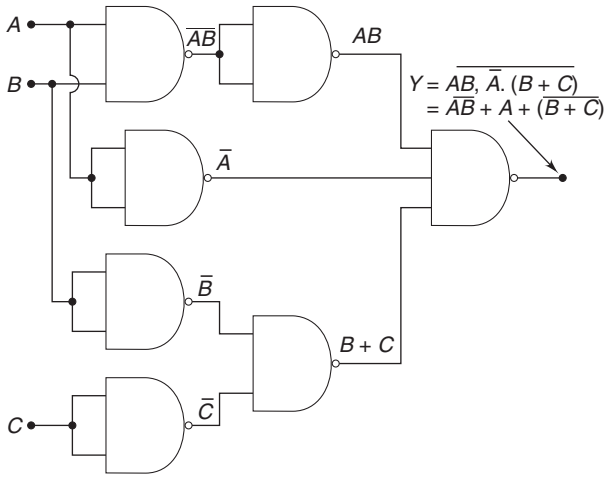


Fig. 4.15

EXAMPLE 4.17

Simplify the logic circuit shown in Fig. 4.16(a)

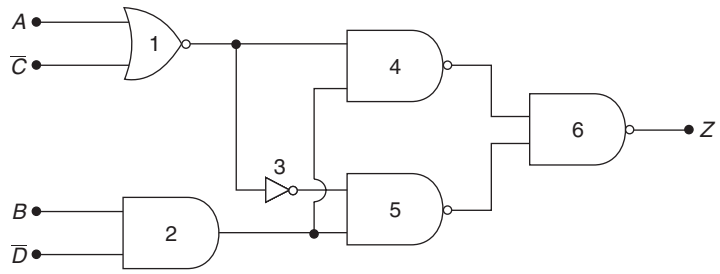


Fig. 4.16(a)

Solution From the given logic circuit, the expression for Z can be written as

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

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

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

$$= \overline{B\overline{D}} + (A + \overline{C})(\overline{A + \overline{C}}) \qquad [\because (A + B)(A + C) = A + BC]$$

$$= \overline{\overline{B\overline{D}}} \qquad [\because A\overline{A} = 0]$$

$$= B\overline{D}$$

Therefore, the above logic circuit can be simplified as shown in Fig. 4.16 (b).

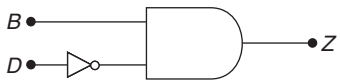


Fig. 4.16(b)

Instead of using Boolean algebra, the logic circuit can be simplified directly as shown below. In the given logic circuit shown in Fig. 4.16 (a) the NAND gate (6) can be replaced by an OR gate with a bubble at its inputs as shown in Fig. 4.16 (c).

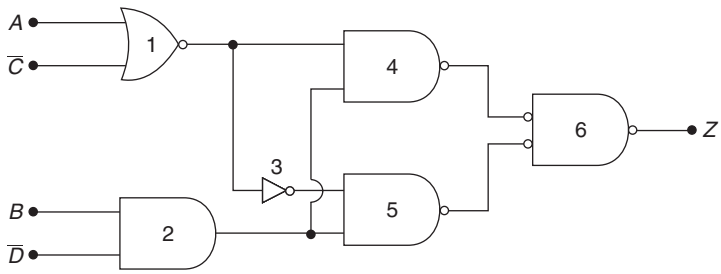


Fig. 4.16(c)

Now, using $\overline{\overline{A}} = A$, the bubble at the outputs of gates 4 and 5 get cancelled with the bubble at the inputs of gate 6 as shown in Fig. 4.16 (d).

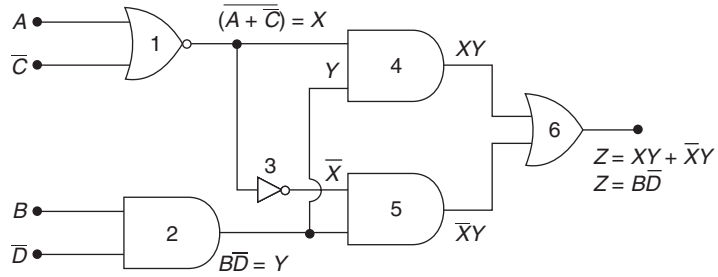


Fig. 4.16(d)

In the above figure, if we assume the output of gate 1 $(\overline{A + \overline{C}}) = X$ and the output of gate 2 $B\overline{D} = Y$, then the output of gate 4 is XY and the output of gate 5 is $\overline{X}Y$. If XY and $\overline{X}Y$ are OR operated in gate 6, then $XY = \overline{X}Y = Y(X + \overline{X}) = Y = B\overline{D}$. Therefore, the above circuit can be simplified as shown in Fig. 4.16 (e).

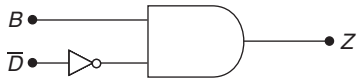


Fig. 4.16(e)

EXAMPLE 4.18

Realise (a) $Y = A + BC\overline{D}$ using NAND gates and (b) $Y = (A + C)\,(A + \overline{D})\,(A + B + \overline{C})$ using NOR gates.

Solution

(a) Using NAND gates

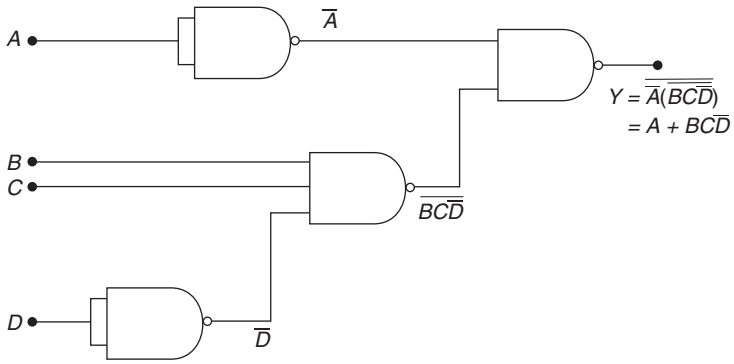


Fig. 4.17(a)

(b) Using NOR gates

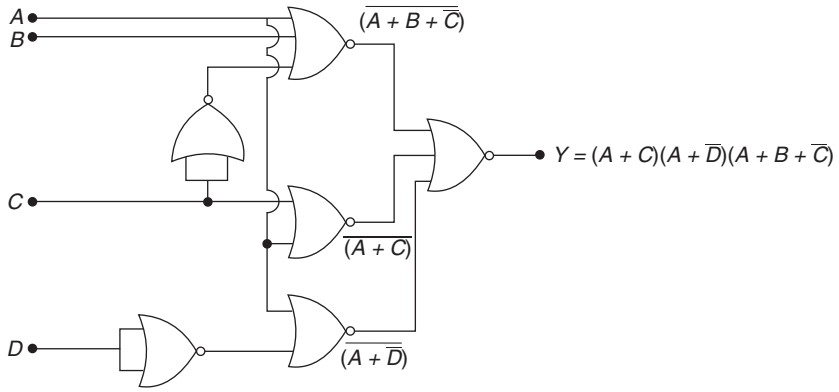


Fig. 4.17(b)

4.8 COMBINATIONAL LOGIC DESIGN

FLASH CARD

What You Learn Here
Define combinational logic circuit.

The logic gates are the fundamental building blocks of the combinational logic circuit. When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting network is called combinational logic. In combinational logic, the output level is at all times dependent on the combination of input levels. Basically, digital circuits are divided into (i) combinational circuits, and (ii) sequential circuits.

In combinational circuits, the outputs at any instant of time depend upon the inputs present at that instant of time. This means there is no memory in these circuits. There are other types of circuits in which the output at any instant of time depends upon the present inputs as well as past outputs. This means that there are elements used to store past information. Such circuits are known as sequential circuits.

Logical functions are expressed in terms of logical variables. Boolean algebraic theorems are used for the manipulation of logical expressions. A logical expression can be realized using the logical gates. The values assumed by the logical functions as well as the logical variables are in the binary form. Any arbitrary logical function can be expressed in the following forms.

1. Sum-of-Products Form (SOP)
2. Product-of-Sums Form (POS)

For example, the logical expression, $Y = AB + A'C + BC$ is a sum of products expression and $Y = (A + B)(B + C')$ is in product of sums form. The sum-of-product form of logical expressions can be realized using AND–OR combination as shown in Fig. 4.18. This realization is known as two level realization. The first level consists of AND gates and the second level consists of OR gates.

Consider the expression,

$$\begin{aligned} Y &= (A + BC)(B + C'A) \\ &= (A + B)(A + C)(B + C')(B + A) \\ &= (A + B)(A + C)(B + C') \end{aligned}$$

The above equation can be realized using OR–AND combination as shown in Fig. 4.19.

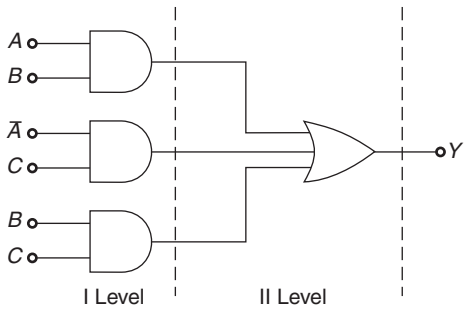


Fig. 4.18 *AND-OR realization of* $Y = AB + A'C + BC$

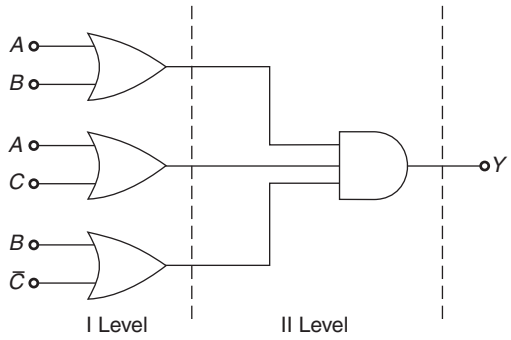


Fig. 4.19 *OR-AND realization of* $Y = (A + B)(A + C)(B + C')$

Each individual term in the standard SOP form is called minterm and the standard POS form as maxterm. An important characteristic of sum-of-products and product-of-sums forms is that the corresponding implementation is always a two-level gate network; hence, the maximum number of gates through which a signal must pass in going from an input to the output is two, excluding inversions.

4.9 KARNAUGH MAP REPRESENTATION OF LOGICAL FUNCTIONS

FLASH CARD

What You Learn Here
Discuss the Karnaugh-map technique for simplifying and manipulating Boolean expressions.

The Karnaugh-map technique provides a systematic method for simplifying and manipulating Boolean expressions. In this technique, the information contained in a truth table or available in POS or SOP form is represented on Karnaugh map (K-map). In an n-variable K-map there are 2^n cells. Each cell corresponds to one of the combinations of n variables. Therefore, we see that for each row of the truth table, i.e., for each minterm and for each maxterm, there is one specific cell in the K-map. The variables have been designated as A, B, C, and D, and the binary numbers formed by them are taken as AB, ABC, and $ABCD$ for two, three, and four variables, respectively. The K-maps for two, three, and four variables are shown in Fig. 4.20.

The entries in a truth table can be represented in a K-map as discussed below.

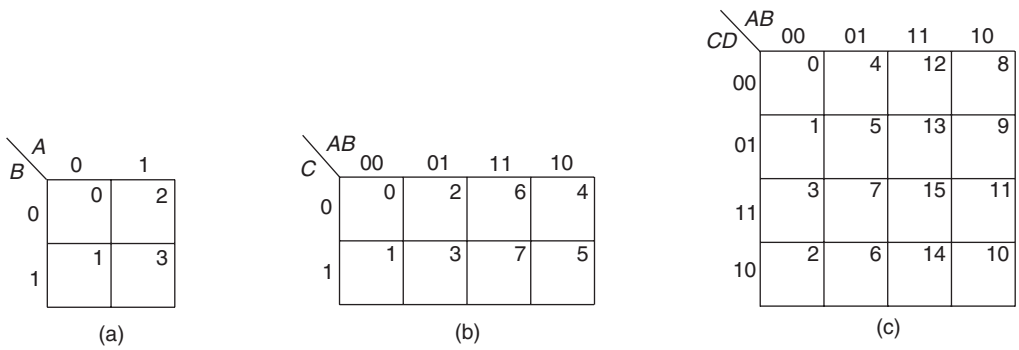


Fig. 4.20 *Karnaugh maps: (a) Two-variable (b) Three-variable (c) Four-variable*

Consider the truth table shown in Table 4.10.

Table 4.10 *Truth table of a digital system*

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

The output Y can be written as,

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

The K-map for the above expression is shown below.

Variables		$\overline{A}\,\overline{B}$	$\overline{A}B$	AB	$A\overline{B}$
		00	**01**	**11**	**10**
$\overline{C}$	**0**		1		1
C	**1**	1		1	

The value of the output variable Y (0 or 1) in each cell can be entered corresponding to its decimal or minterm or maxterm identification.

Simplification is based on the principle of combining terms in adjacent cells. One can group 1s that are in adjacent cells according to the following rules by drawing a loop around those cells:

PROBE

What are the various rules on which simplification is based?

1. Adjacent cells are cells that differ by only a single variable. For example, the cells numbered 000 and 001 are adjacent to each other because they differ by only one bit. Similarly, the cells 011 and 111, 000 and 010, 100 and 110 are all adjacent cells.
2. The 1s in adjacent cells must be combined in groups of 1, 2, 4, 8, and so on.
3. Each group of 1s should be maximized to include the largest number of adjacent cells as possible in accordance with rule 2.
4. Every 1 on the map must be included in at least one group. There can be overlapping groups if they include common 1s.

Grouping is illustrated by the following examples.

EXAMPLE 4.19

Simplify the K-map shown below.

Variables		$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
		00	01	11	10
$\bar{C}$	0	1			1
C	1		1	1	

Solution The adjacent cells that can be combined together are the cells 000 and 100 and the cell 011 and 111.

By combining the adjacent cells, we get

$$Y = (A' + A)\, B'C' + (A' + A)\, BC$$

$$= B'C' + BC$$

The above equation can be obtained from the K-map by using the following procedure.

(a) Identify adjacent ones, then see the values of the variables associated with these cells. Only one variable will be different and gets eliminated. Other variables will appear in ANDed form in the term; it will be in the uncomplemented form if it is 1 and in the complemented form if it is 0.

(b) Determine the term corresponding to each group of adjacent ones. These terms are ORed to get simplified equation in SOP form.

EXAMPLE 4.20

Simplify the K-map shown below.

Variables		$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
		00	01	11	10
$\bar{C}\,\bar{D}$	00	1			1
$\bar{C}D$	00		1	1	
CD	00				
$C\bar{D}$	00				

Solution In the above K-map, the following adjacent cells can be combined to form two pairs of adjacent 1s. Thus, the cell pairs are $\bar{B}\bar{C}\bar{D}$ and $B\bar{C}D$. The simplified functions is $Y = \bar{B}\bar{C}\bar{D} + B\bar{C}D$.

4.10 SOME COMMON COMBINATIONAL CIRCUITS

FLASH CARD

What You Learn Here
List some common combinational circuits like half adder, full adder, half subtractor, full subtractor, and multiplexer, demultiplexer, encoder and decoder and their applications.

In this section, several types of MSI combinational logic functions are studied, including adders, multiplexers, demultiplexers, and code converters.

4.10.1 Half Adder

PROBE

How many binary digits are accepted as inputs by the half adder?

Adder circuits form one of the main applications of the combinational logic circuits. Half adders and full adders are discussed in this section. A logic circuit for the addition of two one-bit numbers is referred to as a half adder. The half adder accepts two binary digits on its inputs and produces two binary digits on its outputs, a sum bit and a carry bit. The logical operation of the half adder is explained in Table 4.11.

Table 4.11 *Truth table for half adder*

Inputs		Outputs	
X	*Y*	Sum(S)	Carry(C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From the truth table, the logical expressions for the sum (S) and carry (C) outputs are given by

$$S = X'Y + XY' = X \oplus Y$$

$$C = XY$$

The implementation required for the half adder is apparent from the above two logical expressions. The sum output is generated with an exclusive-OR gate and the carry output is produced with an AND gate with X and Y on the inputs. The realization of an half adder using gates is shown in Fig. 4.21.

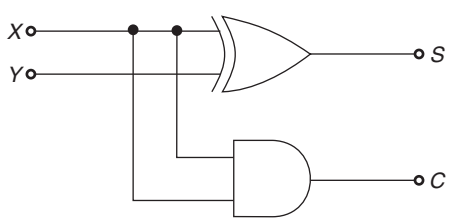


Fig. 4.21 *Realization of a half adder*

4.10.2 Full Adder

PROBE

Tabulate the operation of a full adder.

An half adder has only two-inputs and there is no provision to add a carry coming from the lower order bits when multibit addition is performed. To include the carry, a third input terminal is added and this circuit is used to add X_n, Y_n, and C_{n-1}, where X_n and Y_n are the nth order bits of the numbers X and Y, respectively, and C_{n-1} is the carry generated from the addition of (n–1) order bits. This circuit is called the full adder circuit. The operation of a full adder is shown in Table 4.12.

Table 4.12 *Truth table for full adder*

Inputs			Outputs	
X_n	Y_n	C_{n-1}	S_n	C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

From the truth table, the logical expressions for sum (S_n) and carry (C_n) outputs are given by

$$\begin{aligned} S_n &= \overline{X}_n \overline{Y}_n C_{n-1} + \overline{X}_n Y_n \overline{C_{n-1}} + X_n \overline{Y}_n \overline{C}_{n-1} + X_n Y_n C_{n-1} \\ &= X_n (\overline{Y}_n \overline{C}_{n-1} + Y_n \overline{C}_{n-1}) + \overline{X}_n (Y_n \overline{C}_{n-1} + \overline{Y}_n C_{n-1}) \\ &= X_n \oplus Y_n \oplus C_{n-1} \\ C_n &= \overline{X}_n Y_n C_{n-1} + X_n \overline{Y}_n C_{n-1} + X_n Y_n \overline{C}_{n-1} + X_n Y_n C_{n-1} \\ &= Y_n C_{n-1} + X_n C_{n-1} + X_n Y_n \end{aligned}$$

The realization of a full-adder circuit is shown in Fig. 4.22.

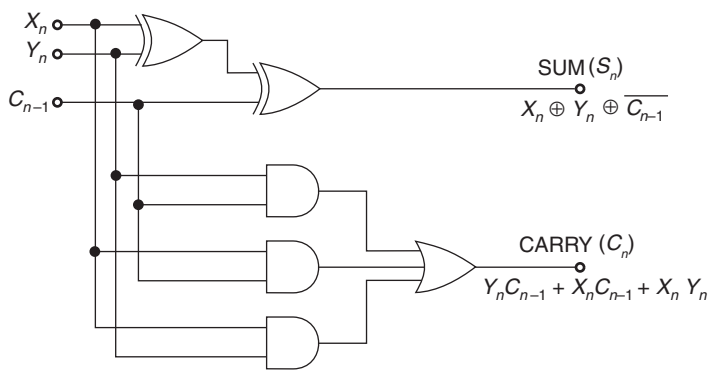


Fig. 4.22 *Realization of a full adder circuit*

4.10.3 Half Subtractor

A logic circuit for subtracting two bits is referred to as a half subtractor. The logical operation of a half subtractor can be understood from Table 4.13.

Table 4.13 *Truth table for half subtractor*

Inputs		Outputs	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Here, X (minuend) and Y (subtrahend) are the two-inputs, and D (difference) and B (borrow) are the two outputs. From the truth table, the logical expressions for D and B are obtained as

$$D = X'Y + XY' = X \oplus Y$$

$$B = X'Y$$

Figure 4.23 shows the realization of a half subtractor using logic gates.

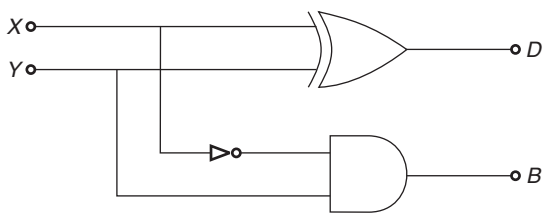


Fig. 4.23 *Realization of a half subtractor*

4.10.4 Full Subtractor

A full subtractor circuit performs three bit subtraction where a borrow from the previous bit position may be included. A full subtractor has three inputs, X_n (minuend), Y_n (subtrahend), and B_{n-1} (borrow from the previous stage), and two outputs, D_n (difference) and B_n (borrow).

PROBE

How does a full subtractor perform?

Table 4.14 *Truth table for full subtractor*

Input			Output	
X_n	Y_n	B_{n-1}	D_n	B_n
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

From the truth table, the output logical expressions can be written as

$$D_n = \overline{X}_n \overline{Y}_n B_{n-1} + \overline{X}_n Y_n \overline{B}_{n-1} + X_n \overline{Y}_n \overline{B}_{n-1} + X_n Y_n B_{n-1}$$

$$= X_n \oplus Y_n \oplus B_{n-1}$$

$$
\begin{aligned}
B_n &= \bar{X}_n \bar{Y}_n B_{n-1} + \bar{X}_n Y_n \bar{B}_{n-1} + \bar{X}_n Y_n B_{n-1} + X_n Y_n B_{n-1} \\
&= \bar{X}_n \bar{Y}_n B_{n-1} + \bar{X}_n Y_n \bar{B}_{n-1} + Y_n B_{n-1} (X_n + \bar{X}_n) \\
&= \bar{X}_n (\bar{Y}_n B_{n-1} + Y_n \bar{B}_{n-1}) + Y_n B_{n-1} \\
&= \bar{X}_n (Y_n \oplus B_{n-1}) + Y_n B_{n-1}
\end{aligned}
$$

Figure 4.24 shows the realization of a full subtractor circuit.

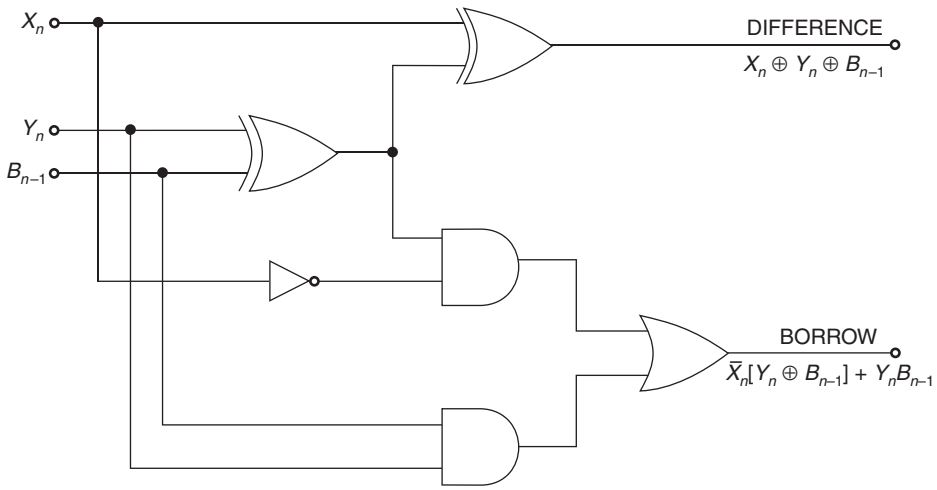


Fig. 4.24 *Realization of a full subtractor*

4.10.5 Multiplexer

PROBE

Define multiplexer.

The multiplexer (or data selector) is a logical circuit that gates one out of several inputs to a single output. The input selected is controlled by a set of select inputs. Figure 4.25 shows the block diagram of a multiplexer with n input lines and one output line. For selecting one out of n inputs for connection to the output, a set of m ($2^m = n$) select signals is required.

Depending upon the digital codes applied to the select inputs, one out of n data sources is selected and transmitted to a single output channel, provided that the system is enabled. The truth table for 4–1 multiplexer is given in Table 4.15.

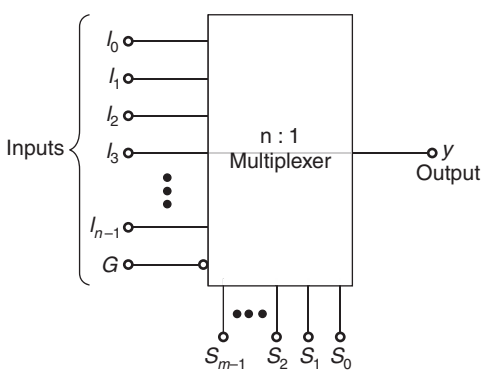


Fig. 4.25 *Block diagram of a digital multiplexer*

Table 4.15 *Truth table for 4–1 line multiplexer*

Select inputs		Output
S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

The logical output Y can be expressed as

$$Y = S_0'S_1'I_0 + S_0'S_1I_1 + S_0S_1'I_2 + S_0S_1I_3$$

The above equation can be realized using gates and is shown in Fig. 4.26.

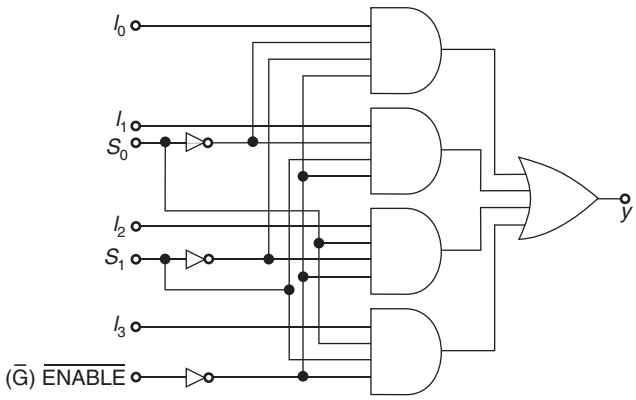


Fig. 4.26 *Realization of a full-adder circuit*

4.10.6 Demultiplexer

The demultiplexer does the reverse operation of a multiplexer. It can be used to separate the multiplexed signals into individual signals. The block diagram of a demultiplexer is shown in Fig. 4.27. The select input code determines to which output the data input will be transmitted.

The number of output lines is n and the number of select lines is m, where $n = 2^m$. Figure 4.28 shows a 1–16 demultiplexer. The input data D is transmitted to one of the outputs Y_i by means of the select signals S_1, S_2, S_3, and S_4. When $S_4 = 0$, $S_3 = 0$, $S_2 = 0$, and $S_1 = 0$, data D is available at output Y_0, and for $S_4 = 1$, $S_3 = 1$, $S_2 = 1$ and $S_1 = 1$ data D is available at the output Y_{15}. For other combinations of select signal lines, data D is available on the respective output lines.

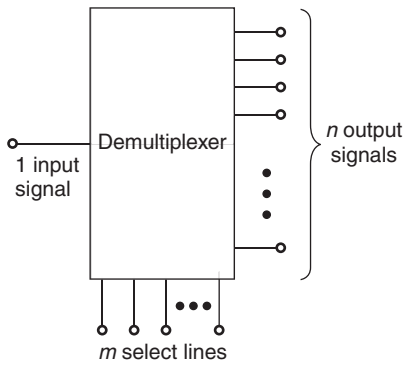


Fig. 4.27 *Block diagram of a digital demultiplexer*

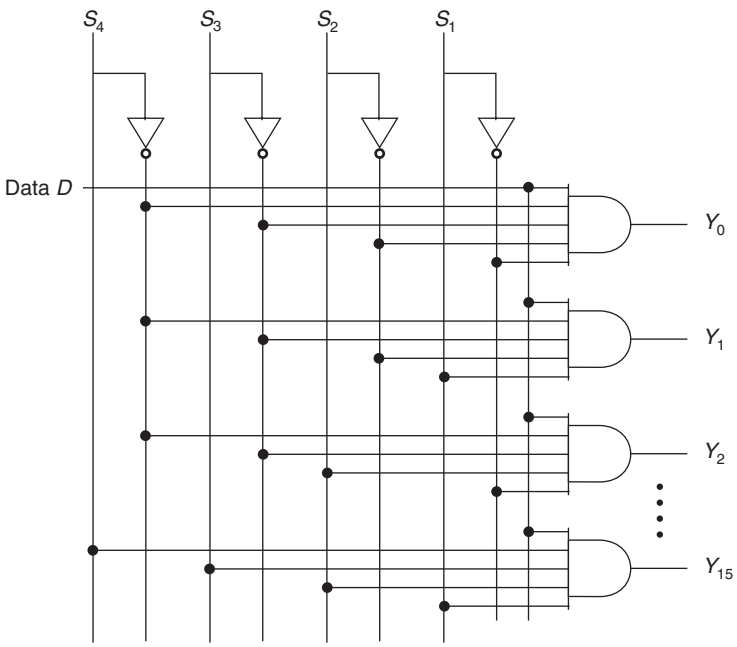


Fig. 4.28 *1-to-16 demultiplexer*

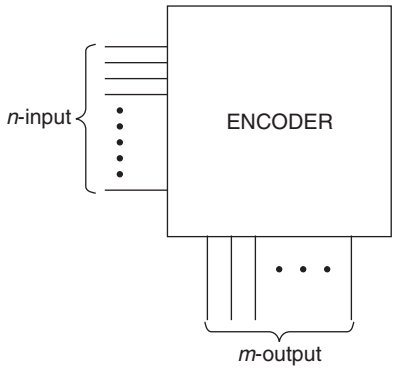


Fig. 4.29 *Block diagram of an encoder*

4.10.7 Encoder

An encoder converts an active input signal into a coded output signal. Figure 4.29 shows the block diagram of an encoder. There are n input lines, only one of which is active.

Internal logic within the encoder converts this active input to a coded binary output with m bits. Figure 4.30 shows a common-type encoder, the decimal to *BCD* encoder. For example, when the button 5 is pressed, the B and D OR gates have high inputs, therefore the output is $ABCD = 0101$.

4.10.8 Decoder

PROBE

How is a decoder different from a demultiplexer?

A decoder is similar to a demultiplexer, with one exception—there is no data input. The only inputs are the control bits *ABCD* as shown in

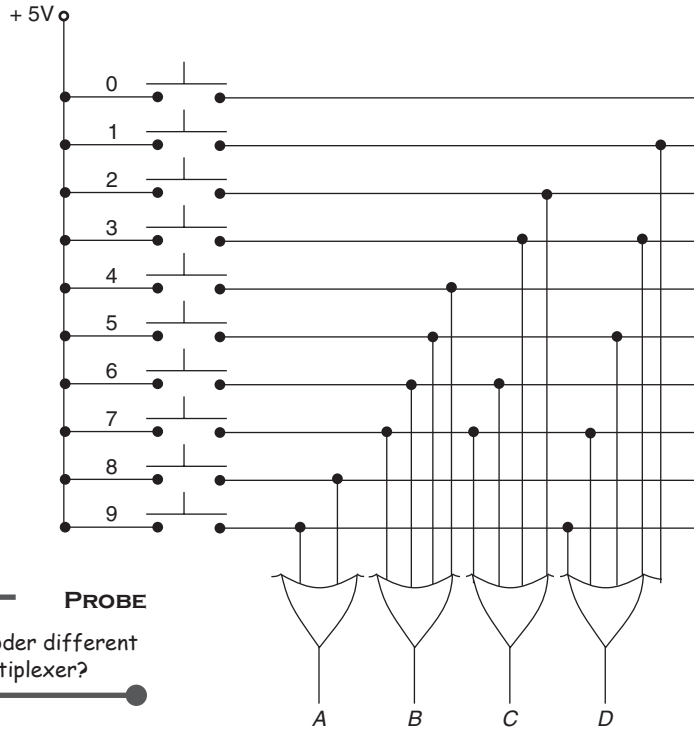


Fig. 4.30 *Decimal-to-BCD encoder*

Fig. 4.31. This logic circuit is called 1-of-16 decoder stance, when *ABCD* is 0001, only the Y_1 AND gate has all inputs high, and the outputs of all other AND gates are low.

4.10.9 Code Converters

▪ ***Gray-code*** The Gray-code is an unweighted code, which means that there are no specific weights assigned to the bit positions. The Gray-code exhibits only a single bit change from one code number to the other.

▪ ***Binary to Gray Code Conversion*** Table 4.16 gives a set of four bit binary numbers and their equivalent Gray-codes.

Table 4.16 *Truth table for binary-to-Gray-code conversion*

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

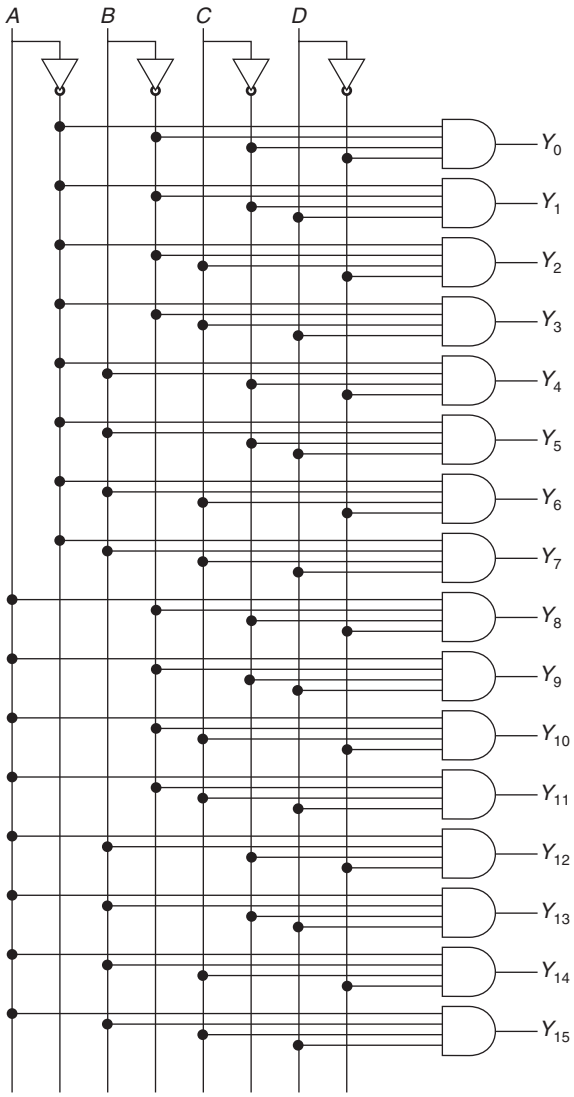


Fig. 4.31 *1-of-16 decoder*

The most significant digit (leftmost) in the Gray-code is the same as the corresponding digit in the binary number. Going from left to right, add each adjacent pair of binary digits to get the next Gray-code digit. Neglect carries.

For example, consider the binary number 10110 to Gray-code conversion.

1. The leftmost Gray digit is the same as the leftmost binary digit.

   10110 Binary
   1 Gray

2. Add the leftmost binary digit to the adjacent one.
   10110 Binary
   11 Gray
3. Add the next adjacent pair.
   10110 Binary
   111 Gray
4. Add the next adjacent pair and discard carry.
   10110 Binary
   1110 Gray
5. Add the last adjacent pair.
   10110 Binary
   11101 Gray

The conversion is now complete and the Gray-code is 11101. The logical expression for each of the Gray-code bits can be obtained from the truth table using algebraic simplification. The logical expressions for each bit of the 4-bit Gray-code ($G_3G_2G_1G_0$) are given below.

$$G_3 = B_3$$
$$G_2 = B_3 \oplus B_2$$
$$G_1 = B_2 \oplus B_1$$
$$G_0 = B_1 \oplus B_0$$

where $B_3B_2B_1B_0$ is the given binary number. A binary-to-Gray-code convertor is shown in Fig. 4.32(a).

▪ ***Gray-to-Binary-Code Conversion*** The most significant digit in the binary code is the same as the corresponding digit in the Gray-code. Add each binary digit generated to the Gray digit in the next adjacent position. Neglect carries.

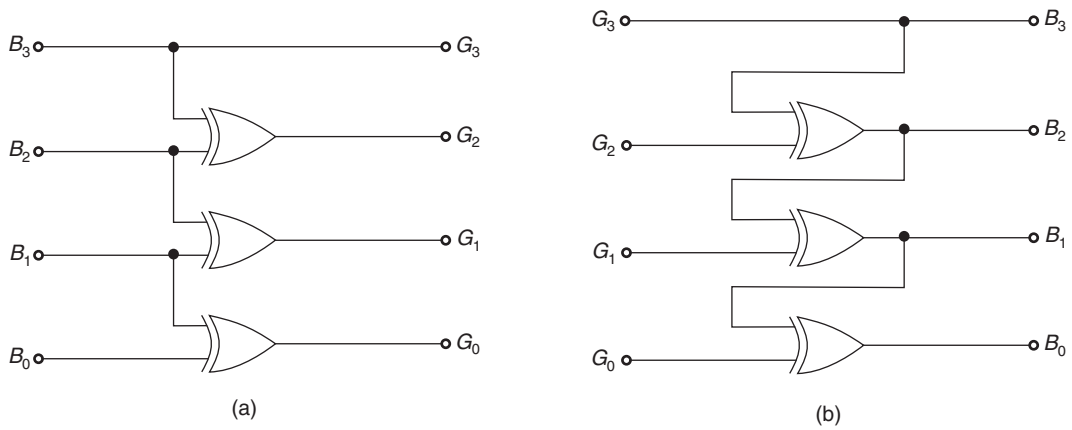


Fig. 4.32 *(a) Binary-to-Gray-code convertor (b) Gray-to-Binary-code convertor*

The conversion of the Gray-code number 11011 to binary is given below.

1. The leftmost digits are the same.
   11011 Gray
   1 Binary

2. Add the last binary digit just generated to the Gray digit in the next position. Disregard carry.
   11011 Gray
   10 Binary
3. Add the last binary digit generated to the next Gray digit.
   11011 Gray
   100 Binary
4. Add the last binary digit generated to the next Gray digit.
   11011 Gray
   1001 Binary
5. Add the last binary digit generated to the next Gray digit. Disregard carry.
   11011 Gray
   10010 Binary

The final binary number is 10010. The logical expression for each of the binary bits can be obtained from the truth table using algebraic simplification. The logical expressions for each bit of the 4-bit Binary ($B_3B_2B_1B_0$) are given as follows.

$$B_3 = G_3$$
$$B_2 = G_3 \oplus G_2$$
$$B_1 = G_3 \oplus G_2 \oplus G_1$$
$$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$$

where $G_3G_2G_1G_0$ is the given binary number. A Gray-to-binary-code convertor is shown in Fig. 4.32 (b).

▪ ***The Excess-3 Code*** The Excess-3 code is got by adding 3 to each decimal digit and then converting the result to 4-bit binary. Excess-3 code is an unweighted code. Table 4.17 shows the conversion between decimal, *BCD,* and Excess-3 codes.

Table 4.17 *Conversion between decimal, BCD, and Excess-3*

Decimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

4.11 SEQUENTIAL CIRCUITS

FLASH CARD

What You Learn Here
Recognize the functionality of sequential circuits.

PROBE

Differentiate a combinational circuit from a sequential circuit.

A combinational circuit can be defined from the behavioural point of view as a circuit whose output is dependent on the inputs at that time instant, or can be defined from the constructional point of view as a circuit that does not contain any memory element.

A sequential circuit, on the other hand, can be defined from the behavioural point of view as a circuit whose output depends not only on the present inputs, but also on the past outputs, or can be defined from the constructional point of view as a circuit that contains at least one memory element. The block diagram of a sequential circuit is shown in Fig. 4.33.

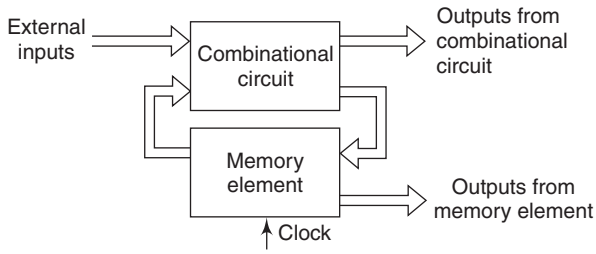


Fig. 4.33 *Block diagram of a sequential circuit*

PROBE

Classify sequential circuits.

Sequential circuits are classified into (i) asynchronous, and (ii) synchronous circuits depending on the timing signals. A sequential circuit whose behaviour depends upon the sequence in which the input signals change is referred to as an asynchronous sequential circuit. The outputs will be affected whenever the inputs change.

A sequential circuit whose behaviour can be defined from the knowledge of its signals at discrete instants of time is referred to as a synchronous sequential circuit. In these circuits, the memory elements are affected only at discrete instants of time. The synchronization is achieved by a timing signal known as system clock. The outputs are affected only with the application of a clock pulse.

4.11.1 Flip-Flops

FLASH CARD

What You Learn Here
Discuss various flip-flops and counters with their applications.

A device that exhibits two stable states is extremely useful as a memory element in a binary system. Any electrical circuit that has this characteristic falls into the category of the devices commonly known as flip-flop (or bistable multivibrators).

▪ ***RS Flip-Flops*** The most basic type of flip-flop is the reset/set flip-flop. This can be built using either two NOR gates or two NAND gates. Each flip-flop has two outputs, Q and Q'. When $Q = 1$ and $Q' = 0$, the flip-flop is said to be set. When $Q = 0$ and $Q' = 1$, the flip-flop is said to be reset. The truth table for the *RS* flip-flop is given in Table 4.18.

Table 4.18 *Truth table for RS flip-flop*

R	*S*	*Q*
0	0	Last value
0	1	1 (set)
1	0	0 (reset)
1	1	Illegal

Consider the NOR gate flip-flop circuit shown in Fig. 4.34(a). When $R = 1$ and $S = 0$, the outputs $Q = 0$ and $Q' = 1$, therefore, the flip-flop is reset. When $R = 0$ and $S = 1$, the outputs $Q = 1$ and $Q' = 0$, therefore, the flip-flop is set. When the first condition is applied, i.e., $R = 0$ and $S = 0$, the flip-flop remains in its present state, i.e., Q remains unchanged. The condition $R = 1$ and $S = 1$ is forbidden as it violates the basic definition of a flip-flop taking both Q and Q' to the low state.

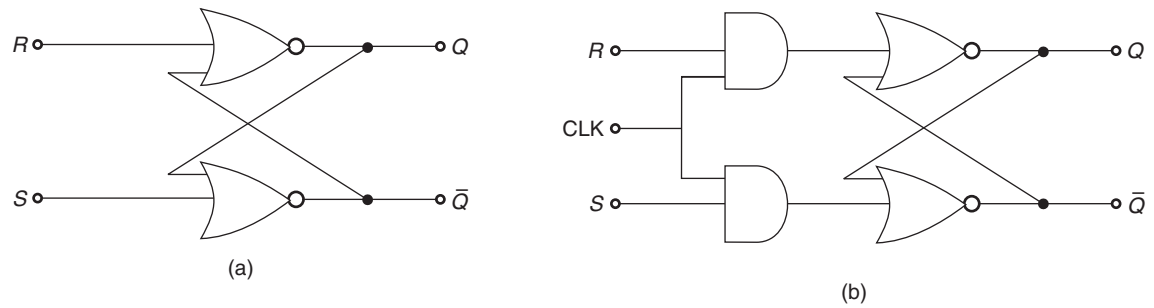


Fig. 4.34 *(a) Realization of RS flip-flop using NOR gates (b) Clocked RS flip-flop*

▪ ***Clocked RS Flip-Flops*** Figure 4.34(b) shows the *RS* flip-flop that has a clock (CLK) input (square wave). When the clock is low, the outputs will not change regardless of the conditions of the *R* and *S* inputs. When the clock input is high, the flip-flop will set if $R = 0$ and $S = 1$. When the clock input is high and $R = 1$ and $S = 0$, the flip-flop will reset. The truth table of the clocked *RS* flip-flop is shown below in Table 4.19.

Table 4.19 *Truth table for clocked RS flip-flop*

CLK	*R*	*S*	*Q*
0	0	0	No change
0	0	1	No change
0	1	0	No change
0	1	1	No change
1	0	0	No change
1	0	1	1
1	1	0	0
1	1	1	Illegal

▪ ***D Flip-Flops*** Figure 4.35 shows the logic symbol and the realization of an edge-triggered *D* flip-flop using NAND gates. The presence of a small triangle on the clock input indicates that the flip-flop is edge triggered.

The *x's* in the truth table (Table 4.20) are called "don't cares" because if the clock is low, high or on its negative edge, the flip-flop is inactive. The outputs Q and Q' change only on the positive going edge of the incoming clock pulse. If $D = 0$ when the positive going clock appears, then $Q = 0$ and $Q' = 1$. If $D = 1$ when the positive going clock edge appears then $Q = 1$ and $Q' = 0$. The data input and output are the same after the

positive going pulse, i.e., the input data *D* is stored only on the positive going edge of the incoming clock pulse.

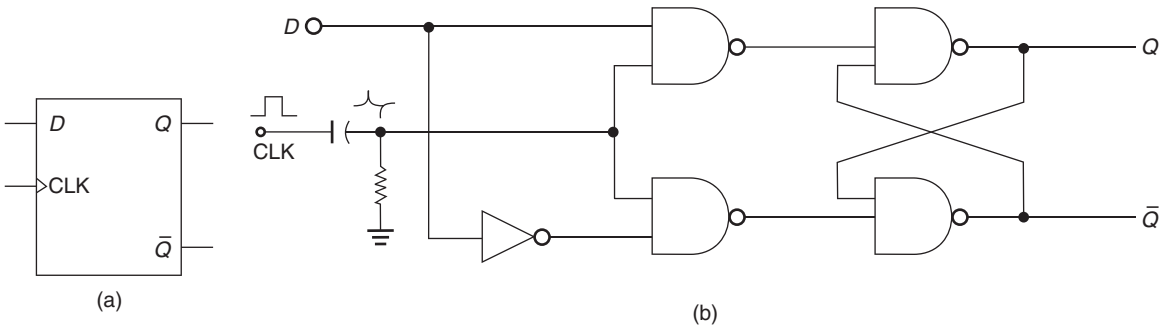


Fig. 4.35 *(a) Logical symbol (b) NAND gates realization of an edge-triggered D flip-flop*

Table 4.20 *Truth table for D flip-flop*

Clock	*D*	*Q*
x	0	No change
x	1	No change
↑	0	0
↑	1	1

▪ ***JK Flip-Flop*** Figure 4.36 shows the negative edge-triggered *JK* flip-flop. The flip-flop is inactive when the clock is low, high, or on its positive going edge.

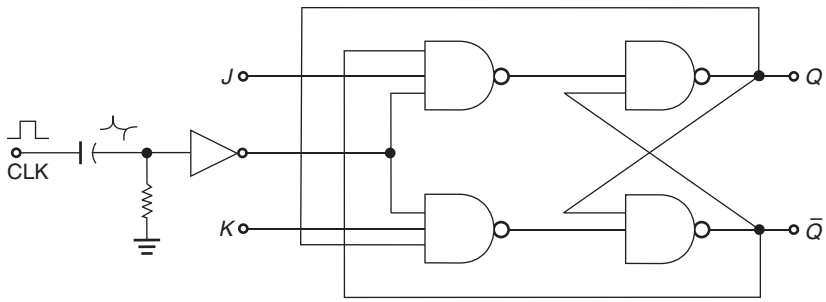


Fig. 4.36 *Negative edge-triggered JK flip-flop*

When $J = 0$ and $K = 0$, the circuit is inactive. When $J = 0$ and $K = 1$, the negative-going edge of the clock pulse puts the outputs at $Q = 0$ and $Q' = 1$. When $J = 1$ and $K = 0$, the negative-going edge of the clock pulse puts the Q outputs at $Q = 1$ and $Q' = 0$. When $J = 1$ and $K = 1$, the outputs Q and Q' toggle or alternate with each negative-going clock edge. (See Table 4.21).

Table 4.21 *Truth table for JK flip-flop*

Clock	J	K	Q
x	0	0	No change
x	0	1	No change
x	1	0	No change
x	1	1	No change
↓	0	0	No change
↓	0	1	0
↓	1	0	1
↓	1	1	$\overline{Q}$

▪ ***T Flip-Flop*** This changes state with each clock pulse and hence, it acts as a toggle switch. If $J = K = 1$, then output is the complement of the previous state, so that the JK flip-flop is converted into a T flip-flop. The realization of a T flip-flop from a JK flip-flop is shown in Fig. 4.37.

The truth table for a positive edge-triggered T flip-flop is shown in Table 4.22.

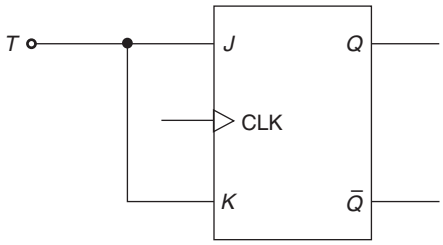


Fig. 4.37 *Realization of a T flip-flop from a JK flip-flop*

Table 4.22 *Truth table for positive edge-triggered T flip-flop*

Clock	T	Q_N
x	0	No change
x	1	No change
↑	0	No change
↑	1	$\overline{Q}_{N-1}$

4.11.2 Shift Registers

A register is a group of flip-flops that can be used to store a binary number. Registers find a variety of applications in digital systems including microprocessors. If the output of each flip-flop is connected to the input of the adjacent flip-flop, then the circuit is called a shift register. The name comes from the fact that each successive clock pulse moves or shifts the data bits one flip-flop to the left or to the right, depending on how the flip-flops are connected.

Registers are classified depending upon the way in which data is entered and retrieved. They are

1. Serial-In Serial-Out (SISO)
2. Serial-In Parallel-Out (SIPO)
3. Parallel-In Serial-Out (PISO)
4. Parallel-In Parallel-Out (PIPO)

PROBE

Describe different registers.

▪ ***Serial-In Serial-Out (SISO)*** Input data is applied one bit at a time to the first flip-flop in a chain and read out from the last flip-flop in a chain one bit at a time. Figure 4.38 shows four *D* flip-flops connected in serial-in serial-out fashion. The output *Q* of one flip-flop is connected to the *D* input of the next flip-flop. *CLK*, $\overline{PRE}$, and $\overline{CLR}$ signals are connected in parallel to all four flip-flops, so that they are all clocked, all set, or all reset at the same time. This shift register is known as the *shift right register* because when the clock pulse is given the data are shifted to the right.

When an active low signal, as shown in Fig. 4.38(a), is given to the $\overline{CLR}$ input, all the *Q* outputs are 0s. Next, assume that the data signal '1' is given as the input to the flip-flop *A*. When the next positive edge of the clock pulse occurs, the 1 on the *D* inputs of the *A* flip-flop will be transferred to the Q_A output. On the next rising edge of the clock signal, the 1 at Q_A is transferred to Q_B output. The *D* input for the flip-flop *A* is '0' and hence, the output Q_A is '0'. On the third positive edge of the clock pulse Q_C becomes high and for the fourth positive edge of the clock pulse Q_D becomes high. The corresponding waveforms are shown in Fig. 4.38(b).

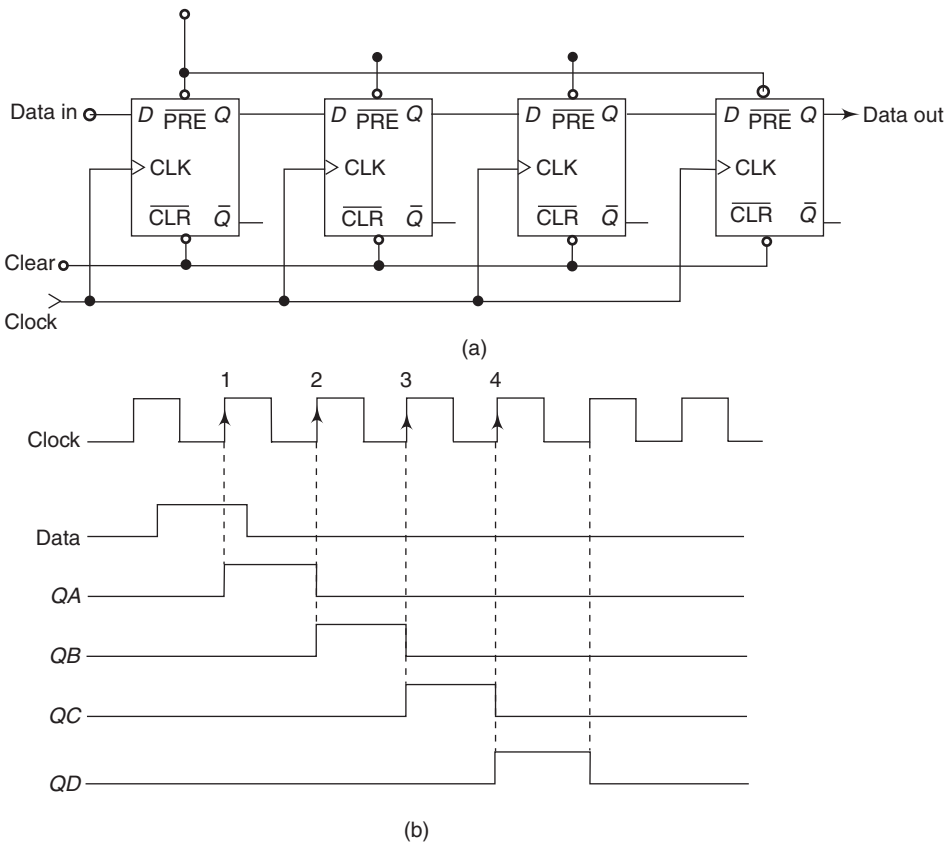


Fig. 4.38 *(a) D flip-flops connected as a shift register (b) Waveforms for shift register*

▪ ***Serial-In Parallel-Out (SIPO)*** Input data is applied one bit at a time to the *D* input of the first flip-flop in a chain and read out from the *Q* outputs in parallel after a data word is all shifted in. A serial in parallel out shift register is shown in Fig. 4.39.

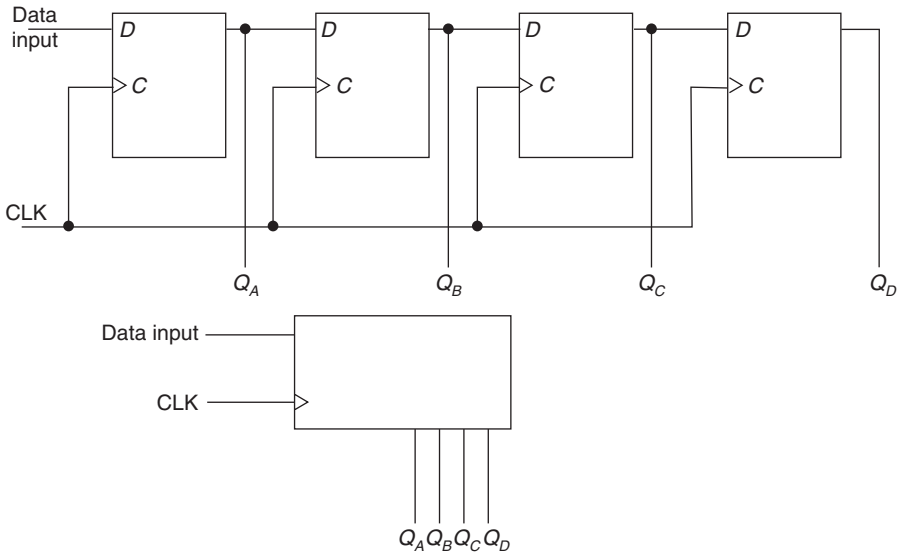


Fig. 4.39 *A serial-in parallel-out shift register*

▪ ***Parallel-In Serial-Out (PISO)*** In a parallel-in serial-out shift register, the bits are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis on one line as with serial data inputs. Figure 4.40 shows the PISO shift register. There are four input data lines, *A*, *B*, *C,* and *D*, and a SHIFT/$\overline{\text{LOAD}}$ input that allows four bits for data to be entered into the shift register in a parallel fashion.

When SHIFT/$\overline{\text{LOAD}}$ is low, gates G_1, G_2 and G_3 are enabled, allowing each data bit to be applied to the *D* input of the respective flip-flops. When a clock pulse is applied, the flip-flops with $D = 1$ will SET and those with $D = 0$ will RESET, thereby storing all 4 bits simultaneously. When SHIFT/$\overline{\text{LOAD}}$ is high, gates G_4, G_5, and G_6 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates allow either the normal shifting operation or the parallel data-entry operation, depending on which AND gates are enabled by the SHIFT/$\overline{\text{LOAD}}$ control signal.

▪ ***Parallel-In Parallel-Out (PIPO)*** Data is entered into the PIPO shift register as in the previous PISO case. The difference is, the outputs are taken only from the *Qs* of all the flip-flops simultaneously.

4.11.3 Counters

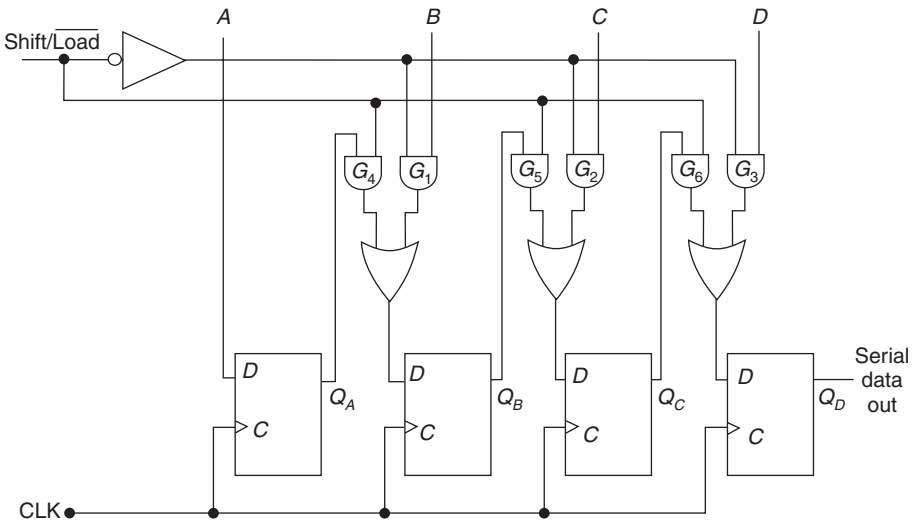
PROBE

What are counters?

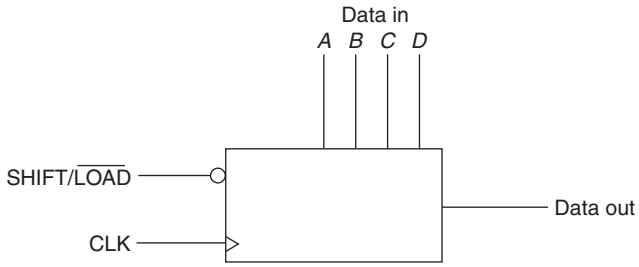
A counter is one of the most important subsystems in a digital system. A counter circuit activated by a clock can be used to count the number of clock cycles. There are two types of counters:

1. Synchronous counters
2. Asynchronous counters

The ripple counter is simple and requires less hardware. However, it has speed limitation. Each flip-flop is triggered by the previous flip-flop, and thus, the counter has a cumulative settling time. These counters are called serial or asynchronous.



(a) Logic diagram



(b) Logic symbol

Fig. 4.40 *A four-bit parallel-in serial-out shift register*

The speed of operation can be increased by using a parallel or synchronous counter but the hardware cost increases because of the extra circuitry introduced. In this type, every flip-flop is triggered by the clock in synchronism and the settling time is equal to the delay time of a single flip-flop.

4.11.4 Asynchronous Counters

FLASH CARD

What You Learn Here
Distinguish between asynchronous counter, synchronous counter, and decade counter.

▪ ***Binary Ripple Counter*** Figure 4.41 shows the connection of *JK* flip-flops to act as a binary counter. For each flip-flop, the *J* and *K* inputs are connected to +5 V. This means that each flip-flop will toggle when its clock input receives a negative-going clock pulse. The MSB of the counter is Q_3 and the LSB of the counter is Q_0. Initially, the $\overline{CLR}$ input of all the flip-flops are tied to ground and hence all the *Q* outputs will be '0'. When the flip-flops are counting, the $\overline{CLR}$ input is tied to +5 V, its inactive state. Figure 4.41 shows how the *Q* outputs of each flip-flop respond to each negative-going clock edge. At each negative-going clock edge, the count increases by 1.

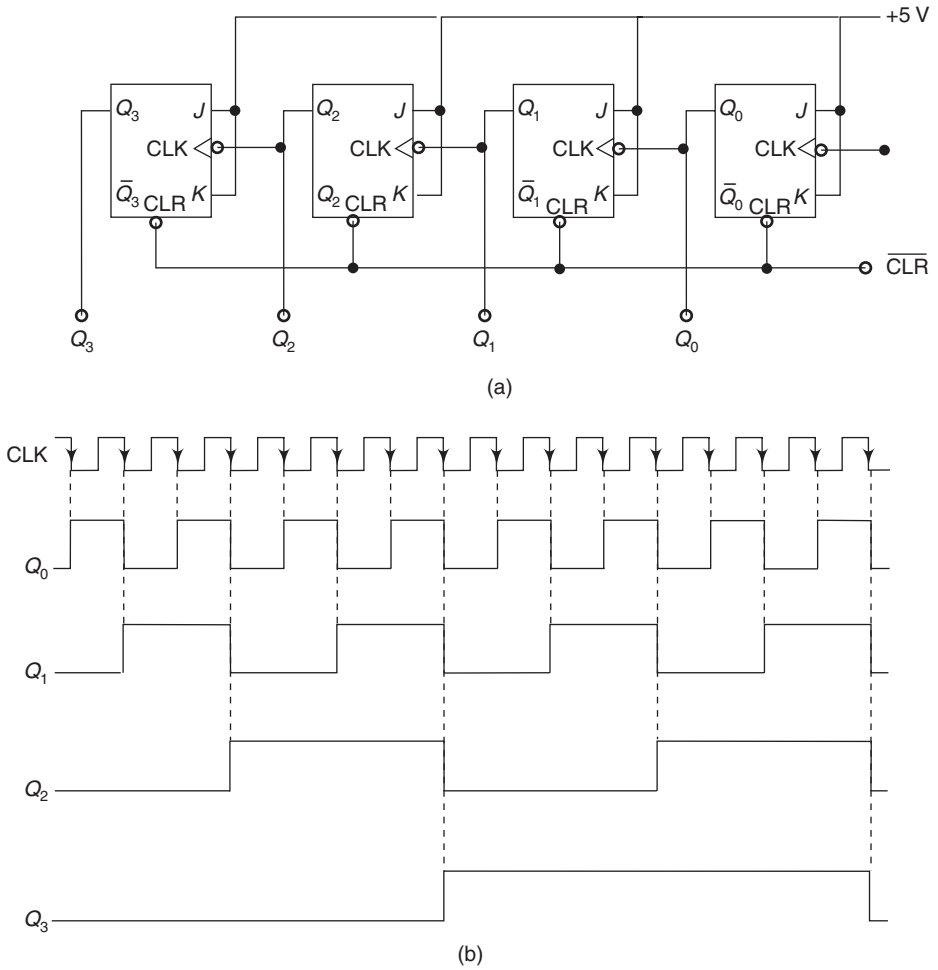


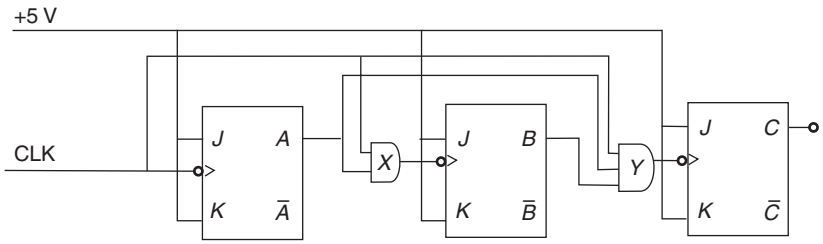
Fig. 4.41 *(a) 4-bit ripple counter (b) Output waveforms at each flip-flop*

Initially, the count is 0000. For the first negative-going edge of the clock input, the LSB flip-flop sets, increasing the count to 0001. The Q_0 output is connected to the clock input of the next most significant flip-flop. This high clock pulse does not cause the Q_1 output to change. However, the second negative-going clock edge applied to the LSB flip-flop causes the Q_0 output to toggle from 1 to 0. This negative-going clock edge causes the Q_1 output to go from 0 to 1, changing the count to 0010. The count continues until 1111 is reached. Then on the next negative-going clock edge, all flip-flop outputs toggle back to 0 for a count of 0000.

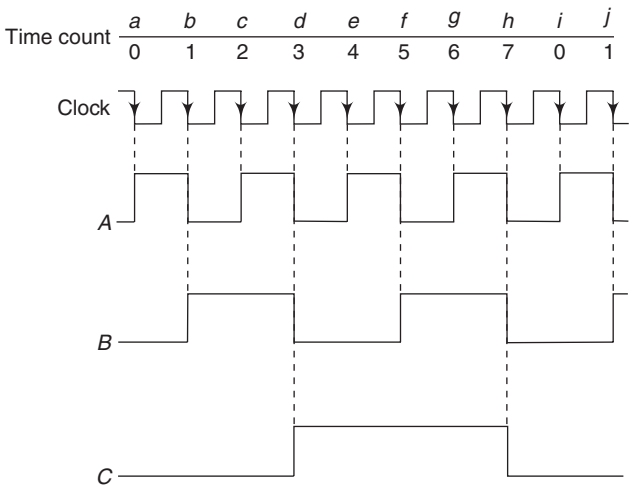
Each flip-flop divides the incoming clock pulse frequency by a factor of 2. The frequency of the Q_0 output is one-half that of the clock input. Similarly, Q_1 output is one-fourth the frequency of the clock input, Q_2 output is one-eighth the frequency of the clock input and the Q_3 output is one-sixteenth the frequency of the clock input. The counter is called a *ripple counter* because the output of one flip-flop is fed to the clock input of another.

4.11.5 Synchronous Counters

The delay time and hence, the settling time increase in the ripple counters are overcome in the synchronous counters. Figure 4.42 shows the circuit of a parallel (synchronous) binary counter. The *J* and *K* inputs of each flip-flop is tied to +5 V, such that the flip-flop toggles for negative clock transition at its clock input. The AND gates are used to gate every second clock to flip-flop *B*, every fourth clock to flip-flop *C*, and so on. The clock is directly applied to the flip-flop *A*. Since *J* and *K* inputs are tied to +5 V, the flip-flop *A* will change state for each negative clock transition. When *A* is high, AND gate X is enabled and the clock pulse is passed through the gate to the clock input of flip-flop *B*. Similarly, the AND gate *Y* is enabled and the clock pulse is passed through the gate to the clock input of the flip-flop *C* when *A* and *B* are high. The counter counts from 000 to 111 in a synchronous manner.



(a)



(b)

C	B	A	Count
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7
0	0	0	0

(c)

Fig. 4.42 *(a) Mod-8 parallel binary counter (b) Waveform (c) Truth table*

4.11.6 Decade Counters

A decade counter requires four flip-flops. This is achieved by recycling after the count of 9 (1001) is reached. To decode the count 1010, a NAND gate is used and the output of the gate is connected to the clear $(\overline{CLR})$ inputs of the flip-flop as shown in the Fig. 4.43(a).

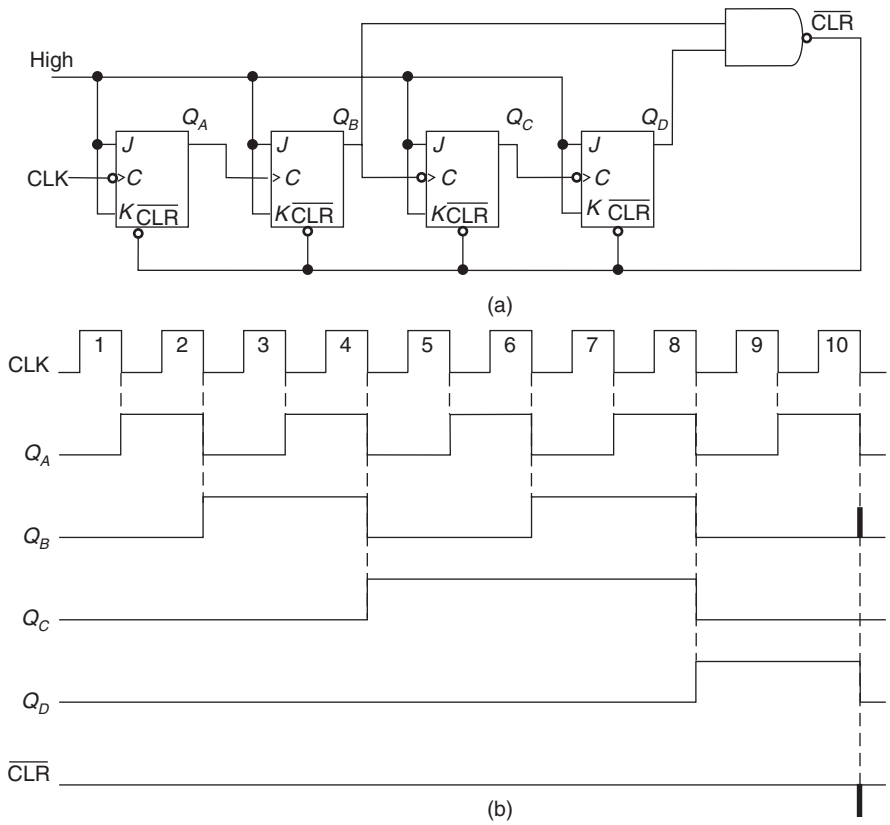


Fig. 4.43 *(a) An asynchronously clocked decade counter with asynchronous recycling (b) Waveform*

Only Q_B and Q_D are connected to the NAND gate inputs. This is an example of partial decoding, in which the two unique states ($Q_B = 1$ and $Q_D = 1$) are sufficient to decode the count 10_{10} because none of the other states have both Q_B and Q_D high at the same time. When the counter goes into count 1010, the decoding gate output goes low and asynchronously RESETS all the flip-flops.

4.12 A/D AND D/A CONVERTER CIRCUITS

FLASH CARD

What You Learn Here

Differentiate between A/D and D/A convertor circuits by describing a binary-weighted resistor D/A convertor, ladder-type D/A convertor, and analog-digital convertor.

Most physical quantities such as pressure, temperature, and flow are analog in nature. There are usually several steps in producing electrical signals which represent the values of these variables and in converting the electrical signals to a digital form that can be used for example, to drive an LED display or be stored in the memory of a microcomputer.

The first step involves a sensor which produces a current or voltage signal that is proportional to the amount of the physical pressure, temperature, or other variable. The signals from most sensors are

quite small, so they must be amplified and perhaps filtered to remove unwanted noise. Amplification is usually done with some type of operational amplifier circuit. The final step is to convert the signal to a proportional binary word with an analog-to-digital (A/D) converter.

4.12.1 Digital-to-Analog Converters

Many systems accept a digital word as an input signal and translate or convert it to an analog voltage or current. These systems are called digital-to-analog converters. The digital word is presented in a variety of codes, the most common being pure binary or binary-coded-decimal.

▪ ***A Binary-Weighted Resistor D/A Converter*** Figure 4.44 shows a circuit which will produce an output voltage proportional to the binary word applied to its inputs by the four switches. Since this converter has four data inputs, it is called a 4-bit converter. The circuit in Figure 4.44 is just a 4-input op-amp adder circuit. The non-inverting input of the op-amp is tied to ground, so that input will be at 0 V. There is feedback from the output of the op-amp to the inverting input, so the op-amp will work continually to hold this input at 0 V.

PROBE

How many data inputs does a 4-bit convertor have?

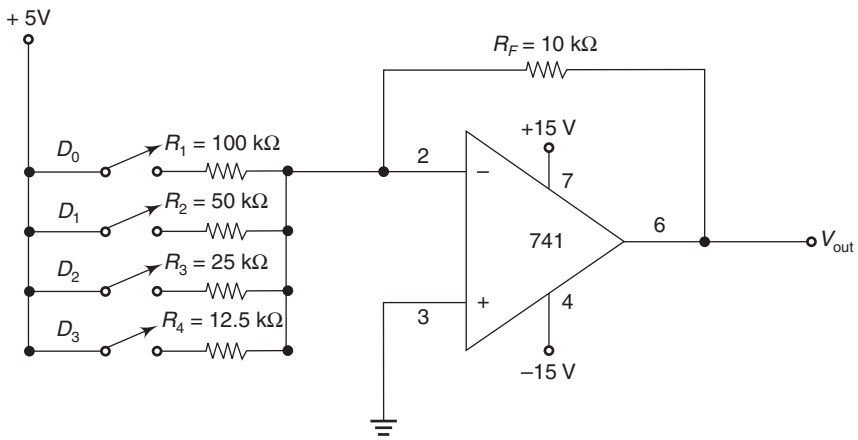


Fig. 4.44 *Binary-weighted resistor D/A converter*

If none of the data switches are closed, there will be no current through any of the input resistors and no current through R_F. The output of the op-amp, then, will be at the same voltage as the inverting input, 0 V.

Now, suppose the D_0 data switch is closed. This will apply a voltage of +5 V to one end of R_1. The other end of R_1 will be held at 0 V by the op-amp, so R_1 has a voltage of 5 V across it and the current through R_1 is 0.05 mA. In order to hold the inverting input at 0 V, the op-amp pulls this current through R_F. The voltage across R_F produced by this current will be 0.05 mA × 10 kΩ = 0.5 V. Since one end of R_F is at 0 V, the output of the op-amp must be at –0.5 V in order to keep the current flowing from +5 V through R_1 to 0 V and on through R_F. Thus, closing D_0 produces a voltage of –0.5 V on the output of the converter circuit.

Suppose the D_1 data switch alone is closed. Since R_2 is only half the value of R_1, twice as much current, or 0.1 mA, will flow through R_2 to the summing point and on through R_F into the output of the op-amp. In order to pull a current of 0.1 mA through R_F, the op-amp will assert a voltage of –1 V on its output. The D_1 data switch then produces an output voltage of –1 V, or twice as much as that produced by the D_0 switch.

If D_2 alone is closed, a current of 0.2 mA, will flow through R_3 to the summing point and on through R_F into the output of the op-amp. This current will produce a voltage of –2 V on the output of the op-amp. Likewise, if the switch D_3 is closed, a current of 0.4 mA, will flow into the summing point and on through R_F into the output of the op-amp. This current will produce a voltage of –4 V on the output of the op-amp.

Now, suppose that switches D_2 and D_3 are both closed. The 0.2 mA current through R_3 will combine with the 0.4 mA current through R_4 at the summing point to produce a total current of 0.6 mA. The output voltage is proportional to the sum of the currents produced by the closed switches and is –6 V. The value of the four currents are related to each other in the same way that the weights of binary digits are. Therefore, the output voltage will be proportional to the binary word applied to the data inputs. D_0 represents the Least Significant Bit (LSB) because it produces the smallest current, and D_3 represents the Most Significant Bit (MSB) because it produces the largest current.

Since there are four inputs, there are 16 possible input words, 0000 to 1111. These words produce output voltages ranging from 0 V for an input word of 0000 to –7.5 for an input word of 1111.

▪ ***Ladder-type D/A Converter*** Figure 4.45 shows D/A converter using *R*–2*R* ladder network. The ladder used in this circuit is a current-splitting device, and thus, the ratio of the resistors is more critical than their absolute value. It can be observed from the figure that at any of the ladder nodes the resistance is 2*R* looking to the left or the right or towards the switch. Hence, the current will split equally toward the left and right, and this happens at every node. Considering the node $N-1$ and assuming the MSB turned ON, the voltage at that node will be $-V_R/3$. Since the gain of the operational amplifier to node N–1 is $-3R/2R$, the weight of the MSB becomes

PROBE

How does a ladder-type D/A convertor work?

$$V_o = -\left(\frac{V_R}{3}\right)\left(\frac{-3R}{2R}\right) = \frac{V_R}{2}$$

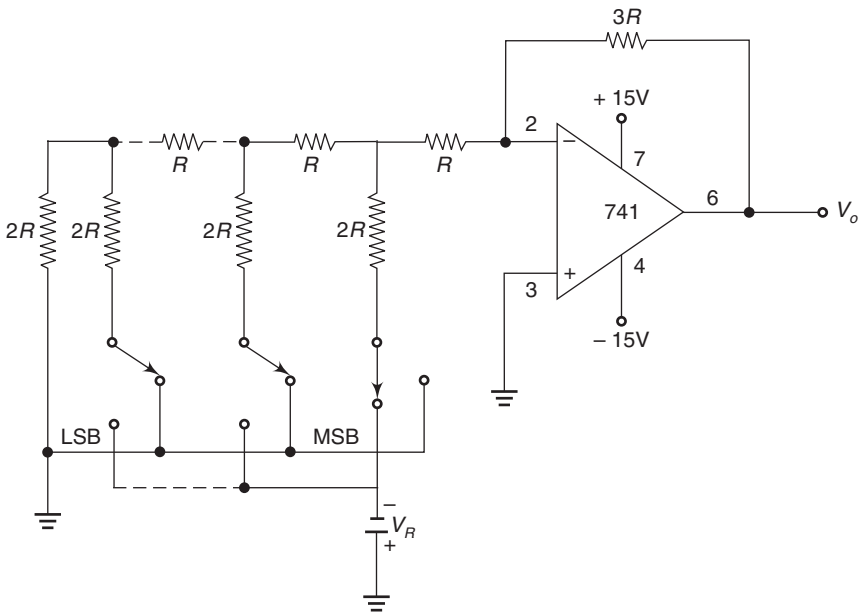


Fig. 4.45 *D/A converter using R–2R ladder network*

Similarly, when the second most significant bit is ON and all others are OFF, the output will $V_o = +\frac{V_R}{4}$, the third bit gives $+\frac{V_R}{8}$, and the LSB gives $+\frac{V_R}{2^N}$. The circuit uses a negative reference voltage and gives a positive analog output voltage. If negative binary numbers are to be converted, the sign-magnitude approach is used; an extra bit is added to the binary word to represent the sign, and this bit can be used to select the polarity of the reference voltage.

4.12.2 Analog-to-Digital Converters

It is often required that data taken in a physical system be converted into digital form. Such data would normally appear in electrical analog form. For example, a temperature difference would be represented by the output of a thermocouple, the strain of a mechanical member would be represented by the electrical unbalance of a strain-gauge bridge, etc. The need, therefore, arises for a device that converts analog information into digital form. Two major characteristics of an A/D converter are *resolution and conversion time*.

PROBE

How is resolution of an A/D convertor determined?

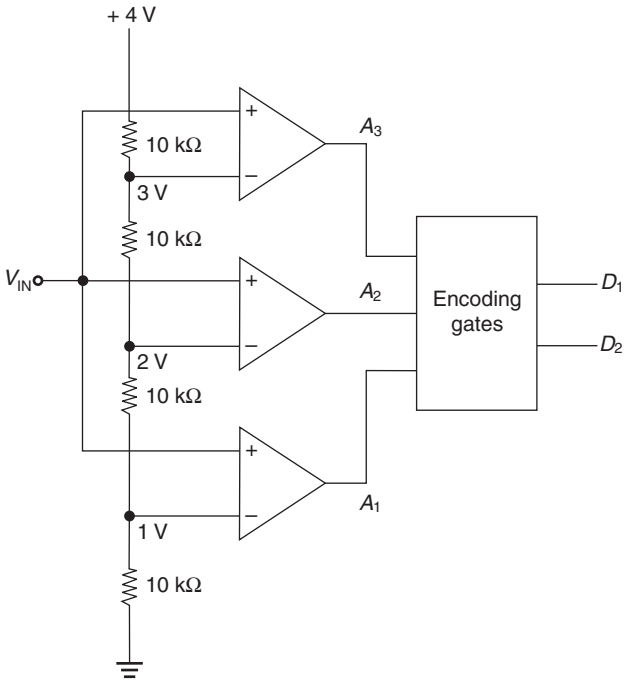
The resolution of an A/D converter is determined by the number of bits in the output word. An 8-bit A/D converter, for example, will represent the value of an input voltage with one of 256 possible words. Another way of putting this is to say that it will resolve an input voltage to the nearest one of 256 levels.

The conversion time for an A/D converter is simply the time it takes the converter to produce a valid output word after it is given a 'start conversion' signal. When we refer to an A/D converter as 'high speed' or 'fast', we mean that it has a short conversion time.

There are many different ways to do an analog-to-digital conversion. The method chosen depends on the resolution, speed, and type of interfacing needed for a given application.

▪ ***Parallel Comparator or Flash Converter*** The simplest in concept and the fastest type of A/D converter is the parallel converter, or flash type shown in Fig. 4.46(a). The resistor voltage divider sets a sequence of voltages on the reference inputs of the comparators as shown. If the voltage on the '+' input of a comparator is greater than the reference voltage on its '–' input, the output of the comparator will be high. In the circuit, the voltage to be converted is applied to the + inputs of all the comparators in parallel, so the number of comparators that have high on their outputs indicates the amplitude of the input voltage. An input voltage of between 0 and 1 V, for example, is less than the reference voltage on any of the comparators, so none of the comparator outputs will be high. If the input voltage is between 1.002 and 2 V, only the A_1 output will be high. For an input voltage between 2.001 and 3 V, both the A_1 and A_2 outputs will be high. Finally, for an input voltage greater than 3 V, all the comparator outputs will be high. Figure 4.46(b) summarizes the comparator output code that will be produced by input voltages between 0 and 4 V. The code produced on the outputs of the comparators is not binary, but with a simple gate circuit, it can be converted to the binary equivalents shown in the rightmost column of Fig. 4.46(b).

To increase the resolution, more comparators can be added. Seven comparators are required for 3-bit resolution and 15 comparators are required for 4-bit resolution. An *N*-bit converter requires $2^N - 1$ comparators. The number of comparators needed increases rapidly as the desired number of bits increases. The main advantage of a parallel comparator type A/D converter is its speed. The binary word is present on the outputs after just the propagation delay time of the comparators plus the delay time of the encoding gates. This is why a parallel comparator type A/D converter is often called a flash converter.



(a)

V_{IN} Volts	Comparator outputs			Binary outputs	
	A_3	A_2	A_1	D_1	D_0
0 to 1	0	0	0	0	0
1.001 to 2	1	1	1	1	1
2.001 to 3	0	0	0	0	0
3.001 to 4	1	1	1	1	1

(b)

Fig. 4.46 *(a) Circuit for flash-type A/D converter (b) Comparator output codes*

▪ ***Counter-type A/D Converter*** Figure 4.47 shows the block diagram of a counter-type A/D converter. In this system, a continuous sequence of equally spaced pulses is passed through a gate. At the start of a conversion cycle, the counters are reset to 0, so the output of the D/A is at 0 V. A positive unknown voltage applied to the input of the converter will cause the output of the comparator to go high and enable the AND gate. This will let the clock pulses into the counter. Each clock pulse increments the counter by 1 and increases the voltage on the output of the D/A converter by one step. When the voltage on the output of the D/A converter passes the voltage on the unknown *V* input, the output of the comparator will go low and shut off the clock pulses to the counters. The count accumulated on the counters is proportional to the input voltage. The control circuitry then strobes the latches to transfer the count to the output and resets the counters to start another conversion cycle.

The counter method is slower than the flash-type converter. The drawback of this type is that it requires a precision D/A converter. Another drawback of this type is that the counter has to start at 0 and count up until the D/A output passes the input voltage. For an 8-bit converter, then, conversion, may take as long as 255 clock cycles and for a 10-bit converter, a conversion may take as many as 1024 clock pulses.

▪ ***Successive Approximation A/D Converters*** The successive approximation technique is another method to implement an A/D converter. Instead of a binary counter as shown in Fig. 4.47, a programmer is used. The programmer sets the most significant bit (MSB) to 1, with all other bits to 0, and the comparator compares the D/A output with the analog signal. If the D/A output is

PROBE

Define successive approximation technique.

larger, the 1 is removed from the MSB, and it is tried in the next most significant bit. If the analog input is larger, the 1 remains in that bit. Thus, a 1 is tried in each bit of the D/A decoder until, at the end of the process, the binary equivalent of the analog signal is obtained.

The successive-approximation-type A/D converter has the disadvantage that it requires a D/A converter, but it has the advantages of good resolution and relatively high speed.

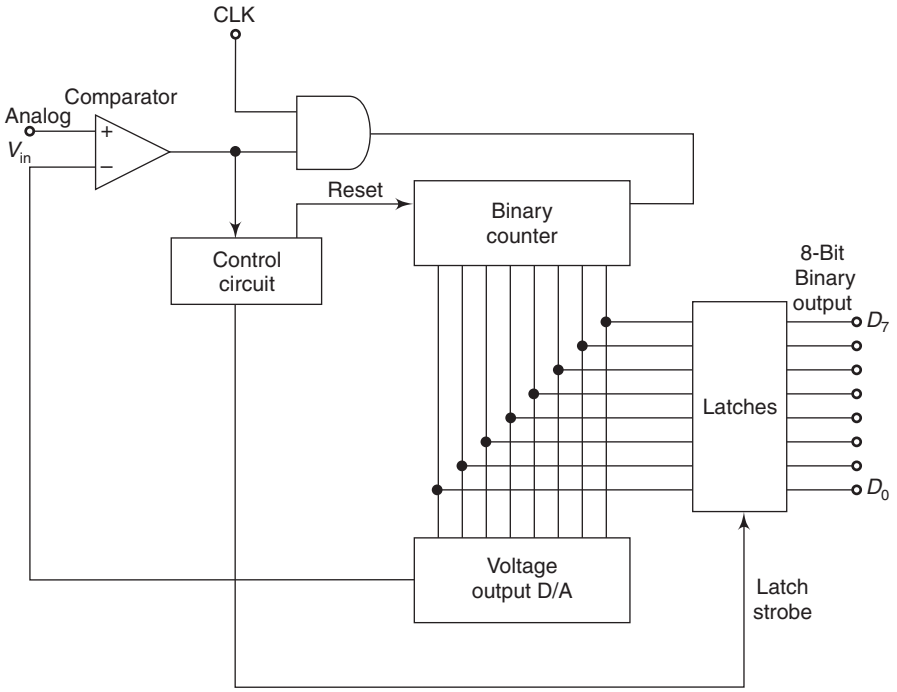


Fig. 4.47 *Counter-type A/D converter*

4.13 LOGIC FAMILIES

FLASH CARD

What You Learn Here

List various categories of logic families like DLT, TTL, TTL subfamilies, ECL, IIL, and MOS.

Digital integrated circuits are classified as Small-Scale Integration (SSI) with less than 12 gates on the same chip, Medium-Scale Integration (MSI) from 12 to 100 gates per chip, Large-Scale Integration (LSI) with more than 100 gates per chip, and very Large-Scale Integration (VLSI) with more than 1000 gates per chip.

ICs can be manufactured using two basic techniques, namely, bipolar and metal-oxide semiconductor (MOS) technologies. Bipolar technology is preferred for SSI and MSI because it is faster. MOS technology dominates the LSI field because of increased density of MOSFETs in the same chip area. A digital family is a group of compatible devices with the same logic levels and supply voltages.

The major categories of the bipolar family are Diode–Transistor Logic (DTL), Transistor–Transistor Logic (TTL) and Emitter-Coupled Logic (ECL). The MOS category consists of PMOS–*p* channel MOSFET, NMOS–n channel MOSFET and CMOS–Complementary MOSFET families.

PROBE

Give various categories of the major categories of the bipolar family.

4.13.1 Diode–Transistor Logic (DTL)

PROBE

Discuss the functioning of DTL.

DTL uses diodes and transistors. A DTL NAND gate may be implemented as shown in Fig. 4.48.

The operation of this positive NAND gate can be understood easily. If at least one input is low, diode *D* connected to this input conducts and the voltage V_A at point *A* is low. Therefore, diodes D_1 and D_2 do not conduct, $I_B = 0$ and the transistor is off. This makes the output of transistor *Q* high and *Y* is in logic 1 state. When all the inputs are high (1) so that all input diodes *D* are cutoff, then V_A rise towards V_{CC}, and a base current I_B results. When I_B is sufficiently large, *Q* is driven into saturation and the output *Y* falls to its low (0) state. DTL has a fan-out of about 12 and can be further increased by replacing D_1 by a transistor. DTL gates are obsolete and are seldom used these days.

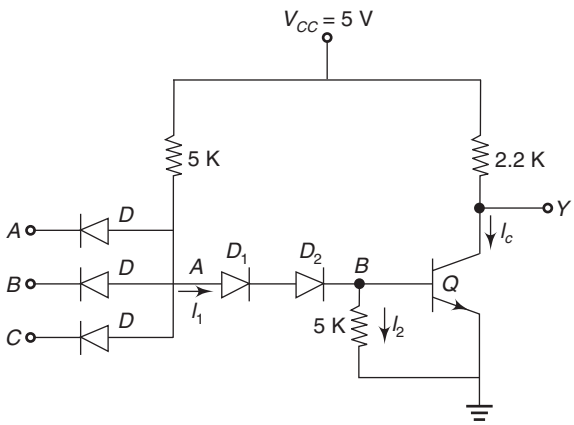


Fig. 4.48 *An integrated positive DTL NAND gate*

4.13.2 Transistor–Transistor Logic (TTL)

The fastest saturating logic circuit is the transistor–transistor logic shown in Fig. 4.49. TTL is fast, inexpensive, and easy to use. This switch uses a multiple-emitter transistor which can be easily and economically fabricated. The TTL circuit has the topology of the DTL circuit with the emitter junctions of Q_1 acting as the input diodes *D* of the DTL gate and the collector junction of Q_1 replacing diode D_1. The base-to-emitter diode of Q_2 is used in place of diode D_2 of the DTL gate, and both circuits have an output transistor.

Transistor Q_1 and 4 kΩ resistor act like a 2-input AND gate. The rest of the circuit inverts the signal so that the overall circuit acts like a 2-input NAND gate. The output transistors (Q_3 and Q_4) form a totem-pole connection; this kind of output stage is typical of most TTL devices. With a totem-pole output stage, either the upper or lower transistor is on. When Q_3 is on, the output is high and when Q_4 is on, the output is low.

The input voltages *A* and *B* are either low (ground) or high (+5 V). If *A* or *B* is low, the base of Q_1 is pulled down to approximately

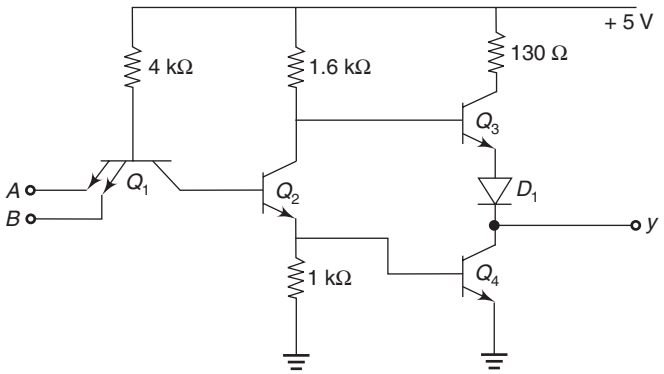


Fig. 4.49 *Two-input TTL NAND gate*

0.7 V. This reduces the base voltage of Q_2 to almost zero. Therefore, Q_2 is cut-off. With Q_2 open, Q_4 goes into cut-off, and the base of Q_3 is pulled high. Since Q_3 acts as an emitter follower, the *Y* output is pulled up to a high voltage.

On the other hand, when *A* and *B* are both high voltages, the emitter diodes of Q_1 stop conducting, and the collector diode goes into forward conduction. This forces Q_2 base to go high. In turn, Q_4 goes into saturation, producing a low output.

Without diode D_1 in the circuit, Q_3 will conduct slightly when the output is low. To prevent this, the diode is inserted; its voltage drop keeps the base–emitter diode of Q_3 reverse-biased. In this way, only Q_4 conducts when the output is low. Totem-pole transistors are used because they produce a low output impedance. Either Q_3 acts as an emitter follower (high output), or Q_4 is saturated (low output). The output voltage can change quickly from one state to the other because any stray output capacitance is rapidly charged or discharged through the low output impedance.

4.13.3 TTL Sub-Families

PROBE

Elaborate on TTL sub-families.

▪ ***High-Speed TTL*** The circuit shown in Fig. 4.49 is called standard TTL. The internal time constants of the circuit can be lowered by decreasing the resistances. This decreases the propagation delay time; however, the small resistances increase the power dissipation. This design variation is known as high-speed TTL. A high speed TTL gate has a power dissipation of around 22 mW and a propagation delay time of approximately 6 nS, whereas a standard TTL gate has a power dissipation of about 10 mW and propagation delay time of approximately 10 ns.

▪ ***Low-Power TTL*** By increasing the internal resistances, the power dissipation of TTL gates can be reduced. Devices of this type are called low-power TTL. These devices are slower than standard TTL because of the larger internal time constants. A low-power TTL gate has a power dissipation of 1 mW and a propagation delay time of about 35 ns.

▪ ***Schottky TTL*** When a transistor is switched from saturation to cut-off, one has to wait for the extra carriers to flow out of the base. The delay is known as saturation delay time. This delay can be reduced by using Schottky TTL. A Schottky diode is fabricated along with each bipolar transistor of a TTL circuit, as shown in Fig. 4.50.

Because the Schottky diode has a forward voltage of only 0.25 to 0.4 V, it prevents the transistor from saturating fully. This eliminates saturation delay time and increases switching speed. Schottky TTL devices are very fast, capable of operating reliably at 100 MHz. The power dissipation is around 20 mW per gate and the propagation delay time is approximately 3 ns.

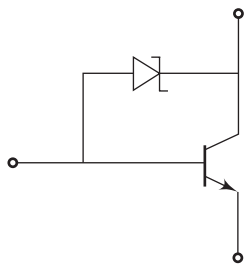


Fig.4.50 *Schottky diode prevents transistor saturation*

▪ ***Low-Power Schottky TTL*** By increasing internal resistances as well as using Schottky diodes, a compromise has been made between low power and high speed, which is referred to as low-power Schottky TTL. A low-power Schottky gate has a power dissipation of around 2 mW and a propagation delay time of approximately 10 ns.

4.13.4 Emitter-Coupled Logic (ECL)

PROBE

What are several other names of ECL?

Emitter-coupled logic has several other common names—current-mode logic (CML), current-steering logic, and non-saturating logic. The last term is the key to this type of circuit. The propagation delay time can be eliminated by operating transistors only in either the active or the cut-off regions rather than operating them in saturation or cut-off regions. The ECL devices are the fastest currently available. ECL has not proved as popular as TTL, primarily because it is more expensive. Superfast computers use ECL as do a number of the higher speed special-purpose computers.

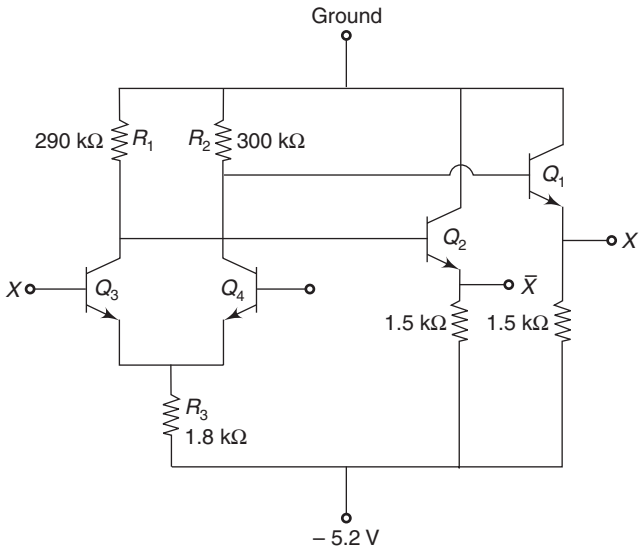


Fig. 4.51 *Basic circuit of ECL inverter gate*

The basic ECL configuration can be best understood by examining a particular inverter. Figure 4.51 shows an ECL inverter. The logic levels in this system are as follows: Binary 0 is represented by –1.55 V and binary 1 by –0.75 V. This is 'positive logic' since the more positive level, –0.75 V, is the binary 1.

The circuit's operation is based on a differential amplifier consisting of Q_3 and Q_4. When the input to Q_3 is at –1.55 V, Q_3 will be off and current will flow through R_3 and R_2. There will be a drop of about 0.8 V across R_2. So, figuring a base-emitter drop of 0.75 V for Q_1, the X output will be at –1.55 V.

Since Q_2 is cut-off by the –1.55 V input, very little current will flow through R_1, and the output X will be at the base-emitter drop voltage across Q_2. So the output will be at –0.75 V. When the input is at –0.75 V, transistor Q_3 will be on, Q_4 will be off, the X output will be at –0.75 V, and the X output will be at –1.55 V. The transistors are never saturated; they are either in their active region or off.

A three-input ECL gate is shown in Fig. 4.52. This is a combined NOR and OR gate, depending on which output connection is used. Several generations of ECL circuits have evolved. In general, the circuits have become faster and require more power with each generation.

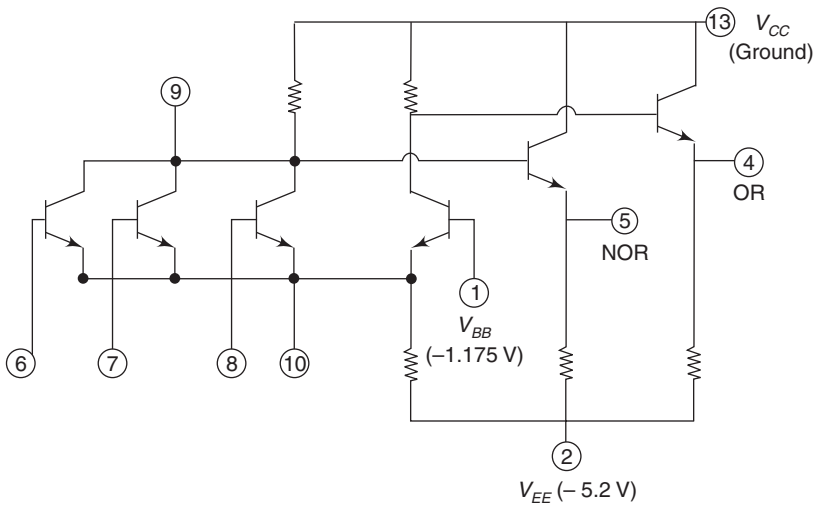
4.13.5 Integrated Injection Logic (IIL) Circuits

PROBE

What is merging?

Integrated injection logic circuits represent an attempt to attain packing densities comparable to MOS circuits while using bipolar junction transistor technology. Standard bipolar technology is the fastest, however, it occupies larger space when compared to MOS circuits. This is due to the reasons that bipolar circuits require resistors, the transistors are larger and an isolation diffusion has to be provided. The problem can be overcome by the technique called **merging**, where the same transistor region is used as part of two or more devices. IIL is the most popular and commonly used merged technology.

The basic IIL gate and a possible semiconductor layout is shown in Fig. 4.53. Each gate has an injector transistor to feed current into the base. This logic circuit has single input and multiple outputs. As no standard symbol exists, the logic gate is represented by a rectangular box, with arrow to represent input and outputs.



Inputs			Outputs	
8	7	6	5	4
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

Fig. 4.52 *Three-input ECL gate and its truth table*

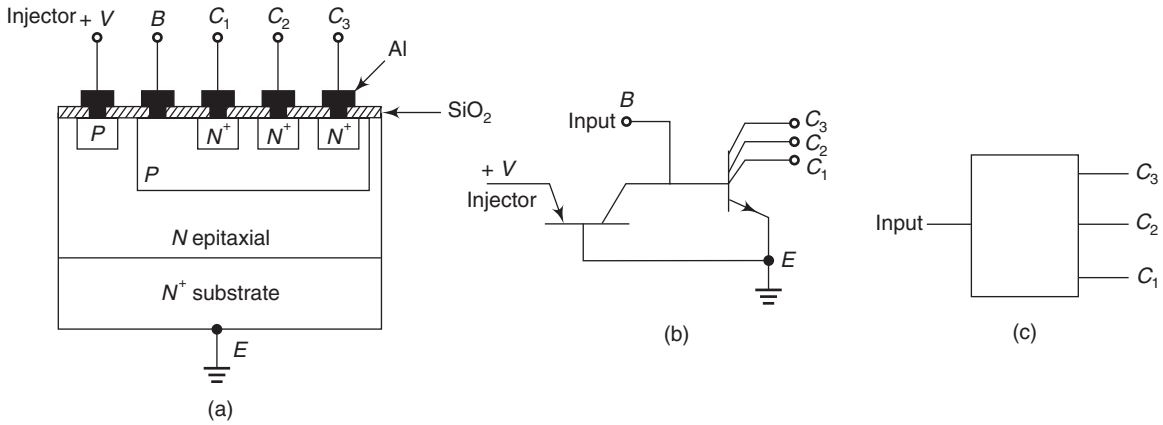


Fig. 4.53 *(a) Structure of an IIL gate (b) Layout (c) Logic symbol*

The configuration shown in Fig. 4.53 is actually an INVERTER with multiple outputs. Various other gates can be formed from this configuration. In Fig. 4.54, realizations for different logic gates are shown.

The IIL outputs are open-collector outputs, and hence connecting them together forms a wired AND. Because the basic IIL gate has a single input and multiple outputs, design does not proceed along regular lines. The advantages of this technology has overcome this problem. IIL doesn't lend itself to chip interconnections as TTL, and seems primarily suited for large-scale integration.

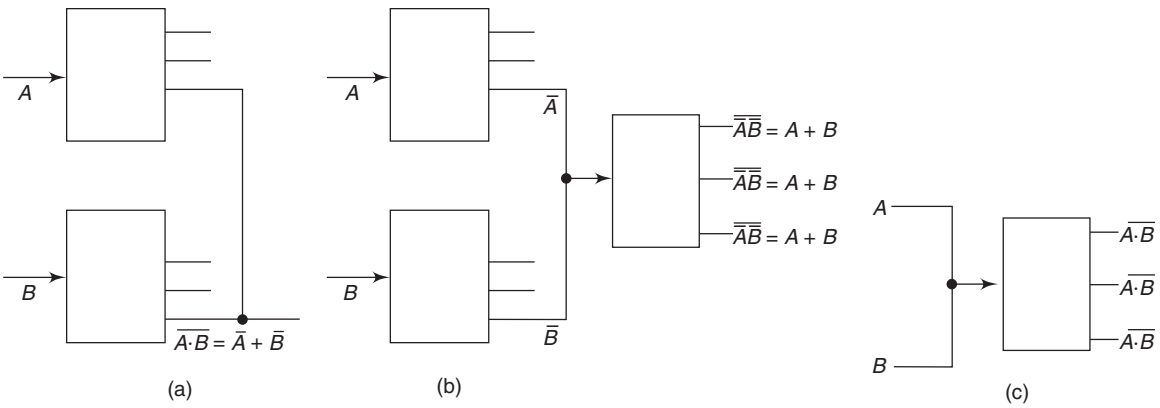


Fig. 4.54 *IIL gates: (a) NOR, (b) OR, (c) NAND*

4.13.6 Metal Oxide Semiconductor (MOS) Circuits

The circuits described so far are all termed bipolar circuits and use conventional transistors. For large-scale integration, quite often a field-effect transistor (FET) is used. The FET devices are easier to manufacture, smaller in size and have small power dissipation.

As switching circuit, a FET can be used to form an inverter. A MOS INVERTER is shown in Fig. 4.55. With a logic 0, or ground input, the output of the circuit will have a –5 V output and with a –2 V or more negative input, the output will go to about 0 V. The FET on the top acts as a resistor.

When a *P*-type substrate with *N*-type doping for the source and drain is used, the *N*-channel MOSFET is formed. This type is called a NMOS device. Figure 4.56 shows a NMOS NOR gate.

The logic levels for this circuit are 0 to 1 V for a binary 0 and greater than +1.5 V for a binary 1. If any of the inputs *A*, *B,* or *C* is a 1, the corresponding FET will conduct, causing the output to go to about +0.8 V or less. If all inputs are at +0.8 V or less, all the FETs will be off and the output will be at +5 V. Figure 4.57 shows a NMOS NAND gate.

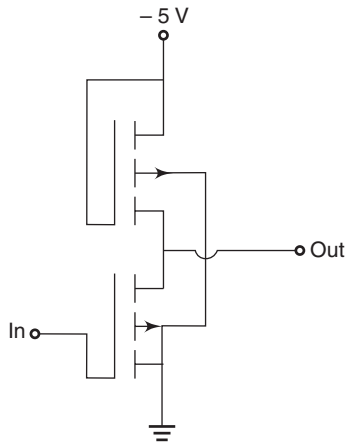


Fig. 4.55 *A PMOS inverter*

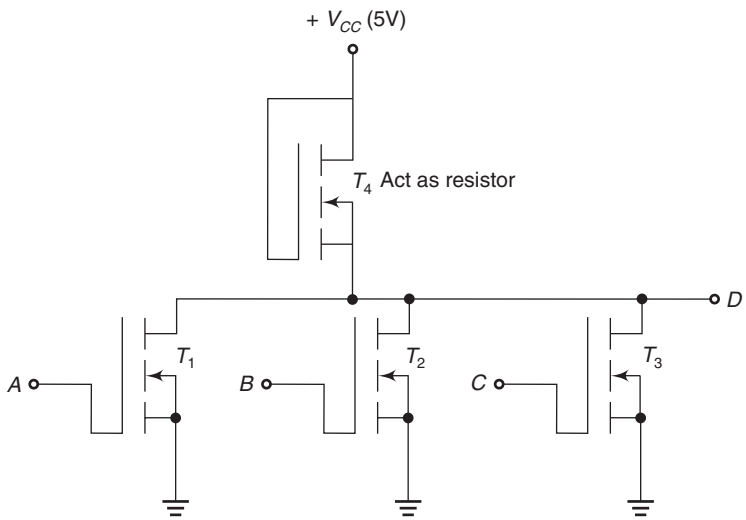


Fig. 4.56 *Three-input NOR gate*

▪ ***CMOS Logic Circuits*** The complementary MOS (CMOS) circuits have very low power consumption and considerable resistance to noise. They are, however, slow. But large numbers of circuits can be placed on a single chip, the power supply voltage can vary over a large range, and the circuits are relatively economical to manufacture. The newest CMOS circuits have become relatively fast and are widely used for everything from electronic watches and calculators to microprocessors.

In CMOS circuits, both *N*- and *P*-channel field effect transistors are fabricated on the same substrate. The simplest form of CMOS integrated circuit consists of one *N*-channel and one *P*-channel MOS transistor, with both gate contacts tied together to form the input and both drain contacts tied together to form the output. Figure 4.58 shows the basic CMOS INVERTER.

PROBE

Describe the simplest form of CMOS.

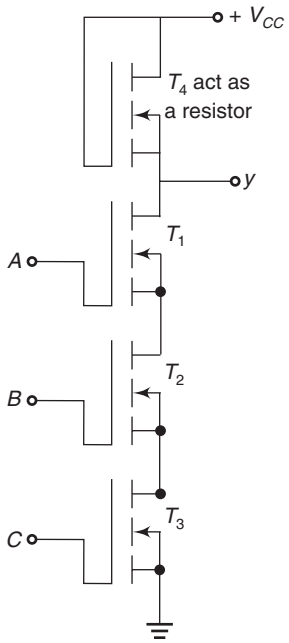


Fig. 4.57 *Three-input NAND gate*

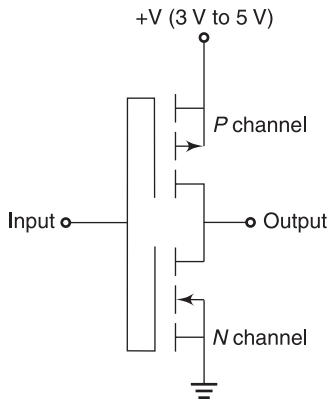


Fig. 4.58 *CMOS inverter*

When the voltage at the input is near ground level, the gate-to-source voltage of the *P*-channel transistor approaches the value of the supply voltage +*V*, and *P*-channel FET is turned on. A low-resistance path is created between the supply voltage and the output, while a high-resistance path exists between the output and ground because the *N*-channel FET is off. The output voltage will approach that of the supply voltage +*V*. When the input voltage is near +*V*, the *P*-channel FET is turned off and the *N*-channel FET is turned on. This makes the output voltage approach the ground value.

As one FET is always off and since both *N*-channel and *P*-channel FETs allow very low leakage current when off, the power consumption is very low in either state. A two-input NOR gate can be constructed using CMOS circuits as shown in Fig. 4.59. It can be seen that each additional input requires an additional *P*- and *N*-channel pair of MOSFETs.

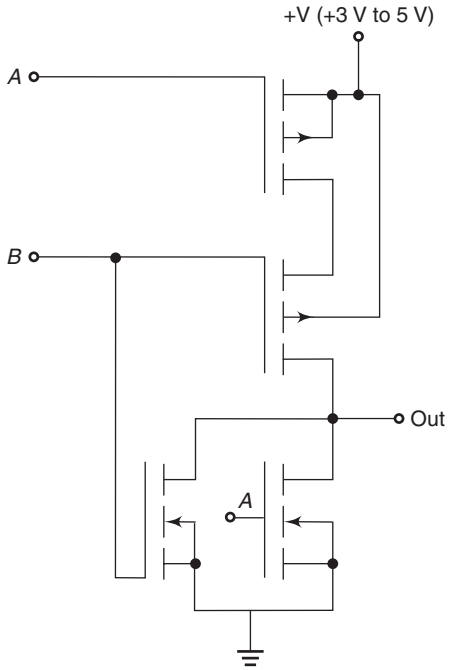


Fig. 4.59 *CMOS NOR gate*

4.13.7 Comparison of Logic Families

PROBE

What are the different parameters of comparing logic families?

Logic families are normally compared based on the following parameters.

- **Fan-out** It is the maximum number of loads that the device can reliably drive. Here, the load and device refer to the logic gates of the same family.
- **Power Dissipation** This is the power dissipated per gate.
- **Noise Immunity** It is defined as the amount of noise voltage that causes unreliable operation. If noise immunity is 0.3 V, then as long as the noise voltages induced on connecting lines are less than 0.3 V, the device will work reliably.
- **Propagation Delay** The time delay encountered when a transistor tries to come out of the saturation condition. When the base drive switches from high to low, a transistor cannot instantaneously come out of hard saturation; extra carriers must first flow out of the base region. The delay incurred in this process is called as the propagation delay.
- **Package Density** It is the number of devices that can be fabricated per unit area of the chip.

Between the bipolar family and MOS family, in general, the bipolar circuits are faster and the MOS circuits provide high package density. Table 4.23 gives the values of the parameters discussed above, for different logic families.

Table 4.23 *Comparison of major IC logic families*

Parameter	DTL	TTL	ECL	MOS	CMOS
Basic gates	NAND	NAND	OR-NOR	NAND	NOR or NAND
Fan-out	8	10	25	20	> 50
Power dissipated per gate, mW	8 – 12	12 – 22	40 – 55	0.2 – 10	0.01
Noise immunity	Good	Very good	Good	Nominal	Very good
Propagation delay per gate, ns	30	12 – 6	4 – 1	300	70

Key Terms

1's Complement
10's Complement
2's Complement
9's Complement
Analog-to-Digital converters
AND gate
Associative property
Asynchronous counters
BCD addition
Binary addition
Binary arithmetic
Binary coded decimal
Binary division
Binary multiplication
Binary numbers
Binary ripple counter
Binary subtraction
Binary-weighted resistor D/A converter
Bistable multivibrators
Boolean addition
Boolean algebra
Boolean multiplication
Clocked RS flip-flops
CMOS logic circuits
Code converters gray-code
Coincidence circuit
Combinational circuits
Combinational logic design
Commutative property
Controlled inverter
Conversion time
Counter-type A/D converter
Counters
D flip-flops
De Morgan's theorems
Decade counters
Decimal–binary conversion
Decoder
Demultiplexer
Digital-to-analog converters
Diode–Transistor Logic (DTL)
Distributive property
Double-dabble method
Emitter-Coupled Logic (ECL)
Encoder
End-arround-carry
Even-parity checker
Excess-3 code
Exclusive-NOR (Ex-NOR) gate
Exclusive-OR (Ex-OR) gate
Fan-out
Flash converter
Flip-flops
Full adder
Full subtractor
Half adder
Half subtractor
Hexadecimal numbers
Hexadecimal–binary conversions
Hexadecimal–octal conversions
High-Speed TTL
Integrated Injection Logic (IIL) circuits
JK flip-flop
Karnaugh map
Ladder-type D/A converter
Least significant bits
Logic families
Logic gates
Low-power Schottky TTL
Low-power TTL
Maxterm
Merging
Metal Oxide Semiconductor (MOS) circuits
Minterm
Most significant bit
Multiplexer
NAND gate
Noise immunity
NOR gate
NOT gate (Inverter)
Number system
Octal numbers
Octal-binary conversions
OR gate
Package density
Parallel comparator
Parallel-In Parallel-Out (PIPO)
Parallel-In Serial-Out (PISO)
Power dissipation
Product-of-Sums form (POS)
Propagation delay
Resolution time
RS flip-flops
Saturation delay time
Schottky TTL
Sequential circuits
Serial-In Parallel-Out (SIPO)
Serial-In Serial-Out (SISO)
Shift registers
Successive approximation A/D converters
Sum-of-Products form (SOP)
Synchronous counters
T flip-flop
Transistor–Transistor Logic (TTL)
TTL sub-families

Review Questions

1. What is the base of the decimal number system? ○○●

2. Find the binary equivalent of each decimal number. ○●●
(i) 238.112 (ii) 36.002 (iii) 1647 (iv) 48.442

3. Find the decimal equivalent of each binary number. ○●●
(i) 1101 (ii) 100101 (iii) 1011.011 (iv) 1110.0011

4. Convert each decimal number to octal. ○●●
(i) 16 (ii) 45 (iii) 270

5. Express each octal number in binary. ○●●
(i) 7013 (ii) 1234 (iii) 72.66

6. What is the hexadecimal representation for 10000100001010010111110011000 10? ●●●

7. What binary number does $A4D6F_{16}$ represent? ●●●

8. Perform the indicated binary operations. ●●●
(i) 110 + 011 (ii) 111011.01 + 110001.11 (iii) 110 – 010 (iv) 11101 – 10101
(v) 101 × 111 (vi) 1110 ÷ 110

9. What is meant by 1's and 2's complement of a binary number? ○○●

10. Explain the rules for binary subtraction using 1's and 2's complement methods. ○○●

11. Subtract 1110 from 1001 using 1's complement and 2's complement methods. ○○●

12. What is a binary coded decimal? ○○●

13. What decimal number does the BCD sequence 0110 1110 1100 0010 1101 1011 0001 represent? ●●●

14. Explain how BCD addition is carried out. ○○●

15. What are the basic rules and properties of Boolean algebra? ○○●

16. State and explain De Morgan's theorems. ○○●

17. Determine whether or not the following equalities are correct: ●●●
(i) $ABC + \overline{ABC} = A$ (ii) $A + BC + \overline{A}C = BC$ (iii) $A(\overline{A}BC + ABCD) = ABCD$

18. Using Boolean algebra techniques, simplify the following expressions as much as possible: ●●●
(i) $A\,(A + B)$ (ii) $A(\overline{A} + AB)$ (iii) $A\overline{B}C + \overline{A}BC + \overline{AB}C$

19. What are logic gates? What are the different types of logic gates? ○○●

20. What are called universal gates? Why they are called so? ○○●

21. Draw the logic diagram of an Ex-OR Gate and discuss its operation. ○○●

22. Give the action of the two input Ex-OR gate and construct it using NAND Gates. ○○●

23. What is an Ex-NOR Gate? Write its truth table. ○○●

24. Verify that the following operations are commutative and associative ○●●
(a) AND (b) OR (C) Ex-OR

25. Design a logic circuit with four input variables that will produce a 1 output when any three input variables are 1s. Use K-map for simplifying the logical expression. ●●●

26. Implement the following logic expressions with logic gates: ●●●
$Y = ABC + AB + BC$
$Y = ABC\,(D + EF)$

27. What is a half adder? How it is realized using logic gates? ○○●

28. Design a full adder circuit using NAND gates. ○○●

29. What is a half subtracter? Realise a full subtracter using NAND gates only. ○○●

30. Determine the logic required to decode the binary number 1110 by producing a 'high' indication on the output. ●●●

31. What is a multiplexer? Explain the operation of 4-to-1 line multiplexer. ○○●

32. Explain how a digital demultiplexer can be realized using logic gates. ○●●
33. Draw a decimal-to-BCD encoder and explain its operation. ○●●
34. What is a decoder and how it is different from a demultiplexer? ○●●
35. What are the key features of Gray-code and Excess-3 code? ○●●
36. Design a combinational logic circuit for binary to Gray-code and Gray to binary code conversion. ●●●
37. Convert the following binary numbers to Gray-codes. ○●●
(i) 11001 (ii) 1110101
38. Convert the following Gray-codes to binary numbers. ○●●
(i) 10010 (ii) 111101
39. Design a binary to excess-3 code convertor using logic gates. ●●●
40. What is a sequential circuit? What is the main difference between combinational circuits and sequential circuits? ○○●
41. Show how an *SR* flip-flop can be constructed using NOR gates? Explain the different states of the SR flip-flop. ●●●
42. Explain the operations of *D* flip-flop, *JK* flip-flop, and *T* flip-flop. ○○●
43. What are shift registers? What are the different types of shift registers? ○○●
44. What are binary counters? What are the two types of binary counters? ○○●
45. Implement an up-down counter using *JK* flip-flop, the up or down count being done based on a control signal. ●●●
46. What is a decade counter? Explain its operation using clock signal waveform. ○●●
47. Why A/D and D/A converters are needed? ○○●
48. What are the different types of D/A converters? ○○●
49. Explain the working of binary weighted resistor type D/A converter. ○○●
50. Explain the working of $R - 2R$ ladder type D/A converter. ○●●
51. What is meant by *resolution* of an A/D converter? ○○●
52. Explain the term *conversion time* of an A/D converter. ○●●
53. Explain the working of a *flash-type* A/D converter. ○●●
54. Explain the working of a *counter-type* A/D converter. ○●●
55. Explain the function of *successive approximation register* in A/D converter. ○●●
56. What are the different kinds of logic families available? ○○●
57. Explain how a DTL NAND gate works, with a suitable diagram. ○○●
58. What are the advantages of TTL circuits? ○○●
59. Draw a two-input TTL NAND gate and explain its operation. ○●●
60. What is a totem-pole arrangement and what is its advantage? ○○●
61. How can a high-speed TTL can be realized? ○○●
62. Explain in brief the different sub-families of TTL circuits. ○○●
63. What is a Schottky TTL circuit? What is its advantage? ○○●
64. What are the advantages of ECL circuit? ○○●
65. Draw the circuit of the basic ECL inverter circuit and explain its operation. ○○●
66. With a truth table, explain how an ECL NOR/OR gate works. ○●●
67. What is an integrated injection logic? ○●●
68. What is the main advantage of IIL circuits? ○○●
69. Show how different logic gates can be realized using the basic IIL inverter. ○●●
70. What are called MOS circuits? What are the advantages of this family of circuits? ○○●
71. Explain how an inverter can be realized in MOS circuits. ○●●
72. Draw the NMOS circuits of NOR and NAND gates. ○○●
73. What is a CMOS circuit? Explain how a CMOS NOR gate functions, using a neat sketch. ○●●

74. What is meant by fan-out of a logic circuit? ○○●

75. What is meant by noise immunity of a logic circuit? ○○●

76. Compare the performance of different logic circuits, based on fan-out, noise immunity, package density, etc. ○●●

Note: ○○● Level 1 & Level 2 category
○●● Level 3 & Level 4 category
●●● Level 5 & Level 6 category

Objective-Type Questions

1. The subtraction of a binary number Y from another binary number X, done by adding the 2's complement of Y to X, results in a binary number without overflow. This implies that the result is
(a) negative and is in normal form
(b) negative and in 2's complement form
(c) positive and is in normal form
(d) positive and is in 2's complement form **[GATE 1987]**

2. The 2's complement representation of –17 is
(a) 101110 (b) 101111 (c) 111110 (d) 110001 **[GATE 2001]**

3. 4-bit 2's complement representation of a decimal number is 1000. The number is
(a) +8 (b) 0 (c) –7 (d) –8 **[GATE 2002]**

4. Decimal 43 in hexadecimal and BCD number system is respectively
(a) B2, 01000011 (b) 2B, 01000011 (c) 2B, 00110100 (d) B2, 01000100 **[GATE 2005]**

5. Two numbers represented in signed 2's complement form are $P = 11101101$ and $Q = 11100110$. If Q is subtracted from P, the value obtained in signed 2's complement form is
(a) 100000111 (b) 00000111 (c) 11111001 (d) 111111001 **[GATE 2008]**

6. The logical expression $y = A + \overline{A}B$ is equivalent to
(a) $y = AB$ (b) $y = \overline{A}B$ (c) $y = \overline{A} + B$ (d) $y = A + B$ **[GATE 1999]**

7. The minimized form of the logical expression $(\overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C})$ is
(a) $\overline{A}\overline{C} + B\overline{C} + \overline{A}B$ (b) $A\overline{C} + \overline{B}C + \overline{A}B$
(c) $\overline{A}C + \overline{B}C + \overline{A}B$ (d) $A\overline{C} + \overline{B}C + A\overline{B}$ **[GATE 1999]**

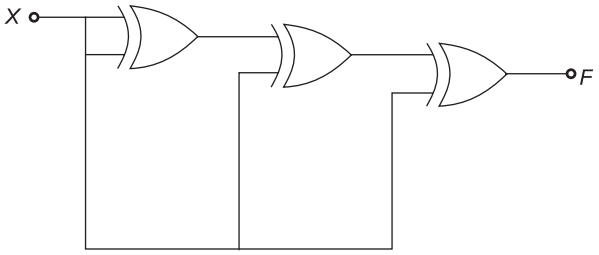
8. The Boolean expression $AC + B\overline{C}$ is equivalent to
(a) $\overline{A}C + B\overline{C} + AC$ (b) $\overline{B}C + AC + B\overline{C} + \overline{A}C\overline{B}$
(c) $AC + B\overline{C} + \overline{B}C + ABC$ (d) $ABC + \overline{A}B\overline{C} + AB\overline{C} + A\overline{B}C$ **[GATE 2004]**

9. The Boolean function $Y = AB + CD$ is to be realized using only 2-input NAND gates. The minimum number of gates required is
(a) 2 (b) 3 (c) 4 (d) 5 **[GATE 2007]**

10. The Boolean expression $Y = \overline{A}\overline{B}\overline{C}D + A\overline{B}\overline{C}D + A\overline{B}\overline{C}\overline{D} + AB\overline{C}\overline{D}$ can be minimized to
(a) $Y = \overline{A}\overline{B}\overline{C}D + \overline{A}B\overline{C} + A\overline{C}D$ (b) $Y = \overline{A}\overline{B}\overline{C}D + BC\overline{D} + A\overline{B}\overline{C}D$
(c) $Y = \overline{A}BC\overline{D} + \overline{B}\overline{C}D + A\overline{B}\overline{C}D$ (d) $Y = \overline{A}BC\overline{D} + \overline{B}\overline{C}D + AB\overline{C}\overline{D}$ **[GATE 2007]**

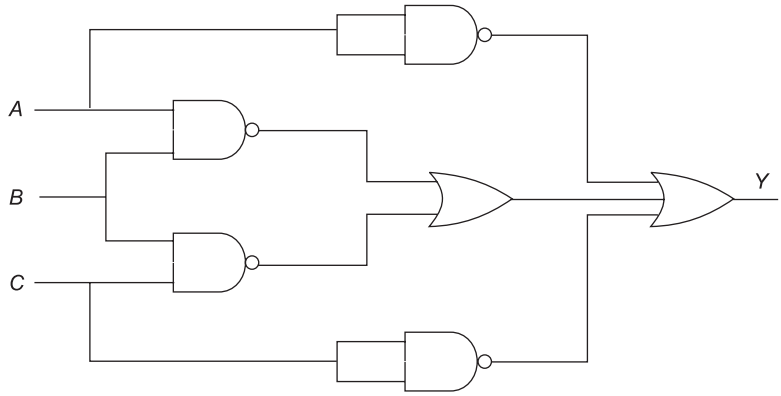
11. The Boolean expression $(X + Y)(X + \overline{Y}) + \overline{(\overline{X}\overline{Y} + \overline{X})}$ simplifies to
(a) X (b) Y (c) XY (d) $X + Y$ **[GATE 2014]**

12. For the circuit shown below, the output F is given by



(a) $F = 1$ (b) $F = 0$ (c) $F = X$ (d) $F = \overline{X}$ **[GATE 1988]**

13. For the logic circuit shown, the output is equal to



(a) $\overline{ABC}$ (b) $\overline{A} + \overline{B} + \overline{C}$

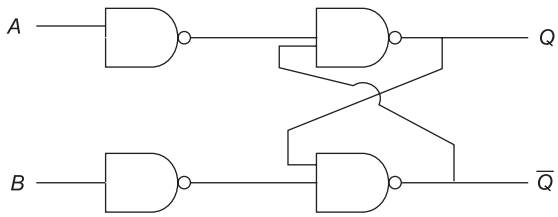
(c) $\overline{AB} + \overline{BC} + A + \overline{C}$ (d) $\overline{AB} + \overline{BC}$ **[GATE 1993]**

14. The number of product terms in the minimized sum-of-product expression obtained through the following K-map is (where, "d" denotes don't care states)

1	0	0	1
0	d	0	0
0	0	d	1
1	0	0	1

(a) 2 (b) 3 (c) 4 (d) 5 **[GATE 2005]**

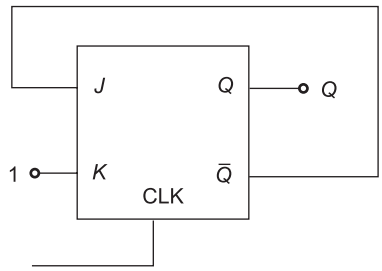
15. The circuit given below is a



(a) *J-K* flip-flop
(b) Johnson's counter
(c) *R-S* latch
(d) none of above **[GATE 1990]**

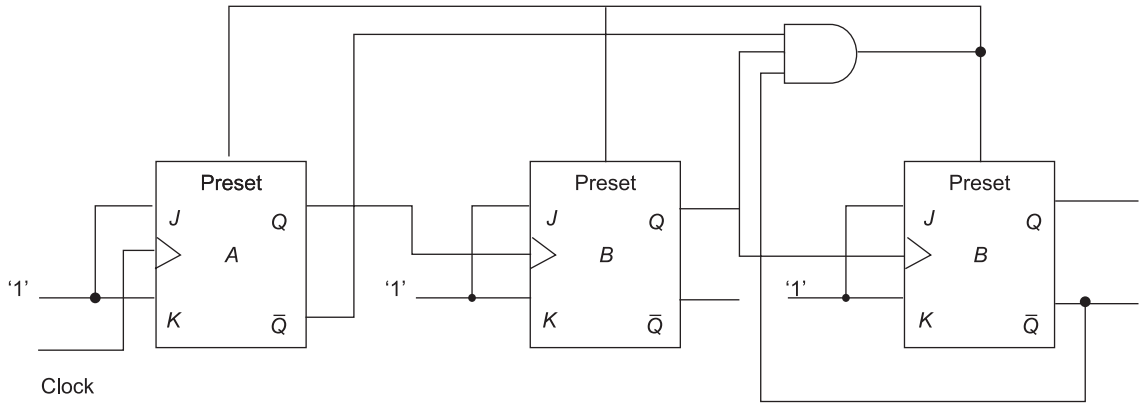
16. An *R-S* latch is a
(a) combinatorial circuit
(b) synchronous sequential circuit
(c) one-bit memory element
(d) one-clock delay element **[GATE 1995]**

17. In a *J-K* flip-flop, we have $J = \overline{Q}$ and $K = 1$ (see figure) Assuming the flip-flop was initially cleared and then clocked for 6 pulses, the sequence at the Q output will be



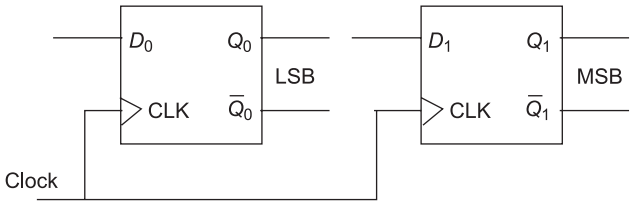
(a) 010000 (b) 011001 (c) 010010 (d) 010101 **[GATE 1997]**

18. The ripple counter shown in the figure works as a



(a) Mod-3 up counter
(b) Mod-5 up counter
(c) Mod-3 down counter
(d) Mod-5 down counter **[GATE 1999]**

19. Two *D* flip-flops, as shown below, are to be connected as a synchronous counter that goes through the following Q_1Q_0 sequence $00 \rightarrow 01 \rightarrow 11 \rightarrow 10 \rightarrow 00 \rightarrow \cdots$. The input D_0 and D_1respectively should be connected as



(a) $\overline{Q}_1$ and Q_0
(b) $\overline{Q}_0$ and Q_1

(c) $\bar{Q}_1 Q_0$ and $\bar{Q}_1 Q_0$ (d) $\bar{Q}_1 \bar{Q}_0$ and $Q_1 Q_0$ **[GATE 2006]**

20. In standard TTL, the 'totem pole' stage refers to
(a) the multi-emitter input stage (b) the phase splitter
(c) the output buffer (d) open collector output stage **[GATE 1997]**

State whether the following statements are TRUE or FALSE.

21. $A + A'B = A + B$

22. $A \cdot (A' + B) = BA$

23. In combinational circuits, the output at any instant of time is dependent only on the inputs present at the instant of time.

24. The output of a sequential circuit at any instant of time is dependent on the present inputs as well as the past inputs/outputs.

25. Each individual term in the standard SOP form is called minterm.

For **Interactive Quizzes**, scan the QR Code given here

OR visit **http://qrcode.flipick.com/index.php/302**