CONTENTS

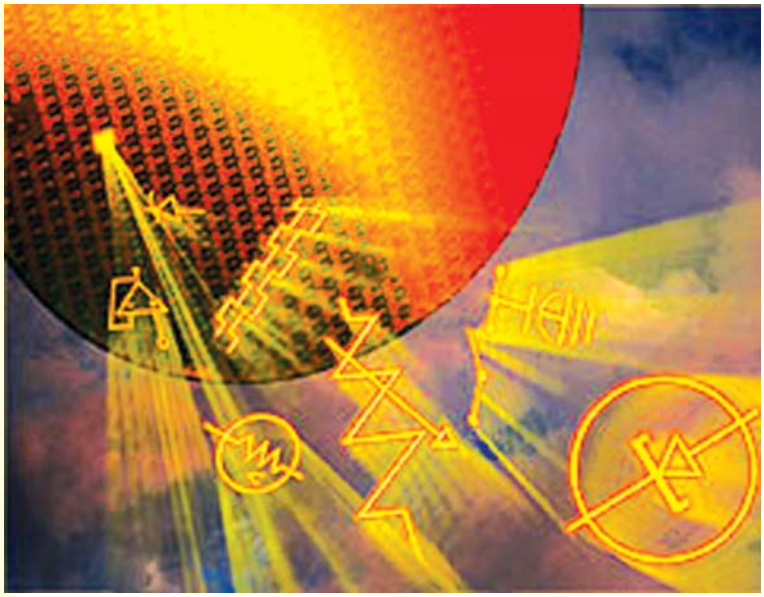
CHAPTER 71

Learning Objectives

- Unique Feature of Boolean Algebra
- Laws of Boolean Algebra
- Equivalent Switching Circuits
- DeMorgan's Theorem's
- The Sum-of-Products (SOP) Form
- The Standard SOP Form
- The Standard POS Form
- The Karnaugh Map
- The Four-variable Karnaugh Map
- Square Adjacency in Karnaugh Map
- Mapping Directly on Karnaugh Map from a Truth Table
- "Don't Care" Conditions
- Main Logic Families
- Saturated and Non-saturated Logic Circuits
- DC supply voltage
- Noise Immunity
- Noise Margin-Power Dissipation
- Power Dissipation versus Frequency
- Propagation Delay
- Speed-Power Product
- RTL Circuit
- DTL Circuit
- TTL Circuit
- TTL Sub-families
- ECL Circuit
- MOS family

CONTENTS

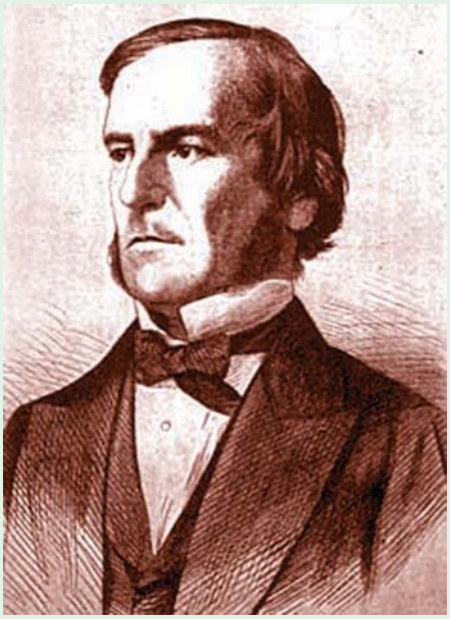
BOOLEAN ALGEBRA AND LOGIC FAMILIES



DNA Logic gates have been developed and are the first step towards creating a computer that has a structure of that of electronic PC

71.1. Introduction

Boolean algebra, named after its pioneer George Boole (1815-1864) is the algebra of logic presently applied to the opertion of computer devices. The rules of this algebra are based on human reasoning. It originated from the study of how we reason, what lines of reasoning are valid and what constitutes proof etc.


George Boole (1815–1864)

Starting with his investigation of the laws of thought, Boole developed in 1854 a mathematical system of logic in which he expressed truth functions *as* ***symbols*** and then manipulated these symbols to arrive at a conclusion. His new system was not the **ordinary numerical algebra** we know from our high school days but a totally new system called **logic algebra.** For example, in Boolean algebra $A + A = A$ and not $2A$ as is the case in ordinary algebra.

Boolean algebra remained in the realm of philosophy till 1938 when Claude *E*. Shannon used it to solve relay logic problems. As we know, all thinking and logic is concerned with finding answers to binary or two-valued questions like : is it good or bad, right or wrong, true or false etc. This binary nature of logic is exactly like the binary working of relay and switching circuits where relay is either energised or not, light is ON or OFF or pulse is present or not. Because of its very logical nature, Boolean algebra is ideal for the design and analysis of ligic circuits used in computers. Morever, it provides an economical and straight forward way of describing computer circuitry and complicated switching circuits. As compared to other mathematical tools of analysis and design, Boolean algebra has the advantages of simplicity, speed and accuracy.

71.2. Unique Feature of Boolean Algebra

As we know, the different variables used in ordinary algebra can have ***any value*** including plus and minus values. There is no restriction on the value they can assume. For example, in the equation $2x + 3y = z$, the variables x, y and z can take on any value available in the entire field of real numbers.

However, the variables used in Boolean algebra have a unique property *i.e.* they can assume ***only one of the two possible values of 0 and 1.*** Each of the variable used in a logical or Boolean equation can assume only the value 0 or 1. For example, in the logical equation $A + B = C$, **each** of the three variables A, B and C can have only the values of either 0 or 1. ***This point must be clearly taken note of by the reader for easy understanding of the laws of Boolean algebra.***

71.3. Laws of Boolean Algebra

As started earlier, Boolean algebra is a system of Mathematics based on ***logic.*** It has its own set of fundamental laws which are necessary for manipulating different Boolean expressions.

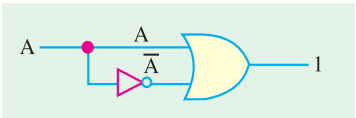


Fig. 71.1

1. OR Laws

These four laws have already been discussed in the previous chapter. These are

Law 1. $A + 0 = A$ Law 3. $A + A = A$

Law 2. $A + 1 = 1$ Law 4. $A + \overline{A} = 1$

The expression given in Law 4 can be understood with the help of Fig. 71.1. Consider the following two possibilities :

(*i*) When $A = 0, \bar{A} = 1$ $\therefore$ $A + \bar{A} = 0 + 1 = 1$

(*ii*) When $A = 1, \bar{A} = 0$ $\therefore$ $A + \bar{A} = 1 + 0 = 1$

2. AND Laws

Law 5. $A \,.\, 0 = 0$ Law 7. $A \,.\, A = A$

Law 6. $A \,.\, 1 = A$ Law 8. $A \,.\, \bar{A} = 0$

The expression for Law 8 can be easily understood with the help of the logic circuit of Fig. 71.2. Consider the following two possibilities :

(*i*) When $A = 0, \bar{A} = 1$ $\therefore$ $A \,.\, \bar{A} = 0.1 = 0$

(*ii*) When $A = 1, \bar{A} = 0$ $\therefore$ $A \,.\, \bar{A} = 1.0 = 0$

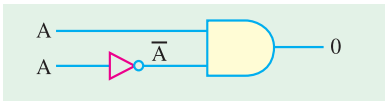


Fig. 71.2

3. Laws of Complementation

Law 9. $\bar{0} = 1$ Law 10. $\bar{1} = 0$

Law 11. if $A = 0$, then $\bar{A} = 1$ Law 12. if $A = 1$, then $\bar{A} = 0$

Law 13. $\bar{\bar{A}} = A$

4. Commutative Laws

These laws allow ***change in the position*** of variables in *OR* and *AND* expressions.

Law 14. $A + B = B + A$ Law 15. $A \,.\, B = B \,.\, A$

These two laws express the fact that the order in which a combination of terms is performed does not affect the final result of the combination.

5. Associative Laws

These laws allow removal of brackets from logical expression and regrouping of variables.

Law 16. $A + (B + C) = (A + B) + C$

Law 17. $(A + B) + (C + D) = A + B + C + D$ Law 18. $A \,.\, (B \,.\, C) = (A \,.\, B) \,.\, C$

6. Distributive Laws

These laws permit factoring or multiplying out of an expression.

Law 19. $A\,(B + C) = AB + AC$ Law 20. $A + BC = (A + B)\,(A + C)$

Law 21. $A + \bar{A} \,.\, B = A + B$

7. Absorptive Laws

These enable us to reduce a complicated logic expression to a simpler form by absorbing some of the terms into existing terms.

Law 22. $A + AB = A$ Law 23. $A \,.\, (A + B) = A$

Law 24. $A \,.\, (\bar{A} + B) = AB$

The above laws can be used to prove any given Boolean identiry and also for simplifying complicated expressions.

71.4. Equivalent Switching Circuits

The equivalent circuits to illustrate some of the *OR* and *AND* laws given above are shown in Fig. 71.3.

(*i*) Fig. 71.3 (*a*) illustrates $A + 1 = 1$. Here, lower switch is permanently closed representing 1. Hence, value of the *OR* function is 1 (*i.e.* it is *ON*) whatever the value of A.

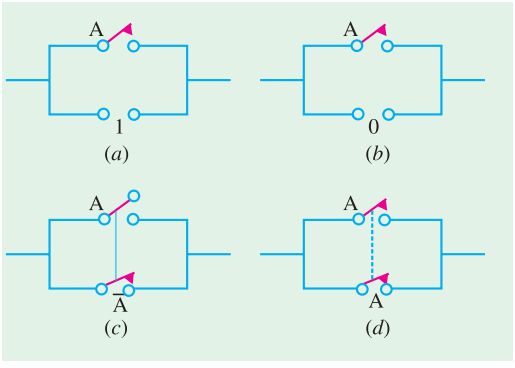


Fig. 71.3

(*ii*) Fig. 71.3 (*b*) represents $A + 0 = A$. Here, function value is determined by A alone.

(*iii*) In Fig. 71.3 (*c*) when A opens, A closes and *vice versa*. Obviously, whatever the position of A, the circuit would always be *ON* proving that $A + \bar{A} = 1$.

(*iv*) Fig. 71.3 (*d*) proves $A + A = A$. It shows that final result depends on the value of A alone. If $A = 0$, the two switches are open, hence circuit is *OFF*. If $A = 1$, both switches are closed. Hence, circuit is *ON*.

(*v*) In Fig. 71.4 (*a*), the circuit is permanently *OFF* irrespective of the value of A. It is due to ***permanent*** open (0) in the circuit. Hence, it proves $A \,.\, 0 = 0$.

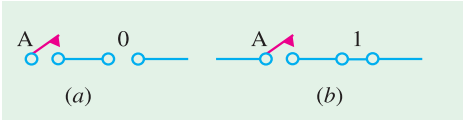


Fig. 71.4

(*vi*) Fig. 71.4 (*b*) shows that circuit conditions will depend solely on the position of the switch. If $A = 1$, circuit is *ON* (1) and when $A = 0$, circuit is *OFF* (0). It is all due to the presence of a ***permanent*** short (1) in the series circuit. Hence, everything depends on A.

Example 71.1. *Prove the following Boolean identity* : $AC + ABC = AC$

Solution. Taking the left hand side expression as y, we get

$$y = AC + ABC = AC(1 + B)$$

Now $1 + B = 1$ —Law 2

$\therefore$ $y = AC \,.\, 1 = AC$ —Law 6

$\therefore$ $AC + ABC = AC$

Example 71.2. *Determine the logic expression for the output Y, from the truth table shown in Fig. 71.5. Simplify and sketch the logic circuit for the simplified expression.*

Solution. There are two 1*s* in the output column of the given truth table. The corresponding binary values are 001 and 101. These values are converted into product terms as follows :

$$001 \rightarrow \bar{A}\ \bar{B}\ C \quad \text{and} \quad 101 \rightarrow A\ \bar{B}\ C.$$

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

Fig. 71.5

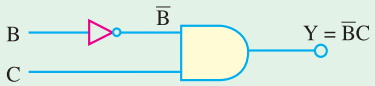


Fig. 71.6

The resulting expression for the output,

$$Y = \bar{A}\ \bar{B}\ C + A\ \bar{B}\ C$$
$$= (\bar{A} + A)\ \bar{B}\ C \quad \text{...(Law 19)}$$
$$= 1.\ \bar{B}\ C \quad \text{...(Law 4)}$$
$$= \bar{B}\ C \quad \text{...(Law 6)}$$

Fig. 71.6 shows the logic circuit to implement the simplified logic expression for the output. As seen it is formed by ANDing the variables $\bar{B}$ and $\bar{C}$. The $\bar{B}$ can be obtained by inverting B.

Example 71.3. *Prove the following Boolean identity* : $(A + B)(A + C) = A + BC$

Solution. Putting the left hand side expression equal to y, we get

$$\begin{aligned} Y &= (A+B)(A+C) \\ &= AA+AC+AB+BC && \text{—Law 19} \\ &= A+AC+AB+BC && \text{—Law 7} \\ &= A+AB+AC+BC \\ &= A(1+B)+AC+BC && \text{—Law 19} \\ &= A+AC+BC && \text{—Law 2} \\ &= A(1+C)+BC && \text{—Law 19} \\ &= A+BC && \text{—Law 2} \end{aligned}$$

$\therefore \quad (A+B)(A+C) = A+BC$

Example 71.4. *Prove the following identity* : $A+\bar{A}B = A+B$

Solution. Let the left-hand side expression be put equal to Y.

$$\begin{aligned} Y &= A+\bar{A}B \quad = A.1+\bar{A}B && \text{—Law 6} \\ &= A(1+B)+\bar{A}B && \text{—Law 12} \\ &= A.1+AB+\bar{A}B && \text{—Law 19} \\ &= A+BA+B\bar{A} && \text{—Law 6 and 15} \\ &= A+B(A+\bar{A}) && \text{—Law 19} \\ &= A+B.1 && \text{—Law 4} \\ &= A+B && \text{—Law 6} \end{aligned}$$

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

Example 71.5. *Prove the following Boolean identity* : $(A+B)(A+\bar{B})(\bar{A}+C) = AC$

(Digital Computations, Punjab Univ. 1990)

Solution. Left the left-hand side expressionbe represented by Y.

$$\begin{aligned} \therefore \quad Y &= (A+B)(A+\bar{B})(\bar{A}+C) = (AA+A\bar{B}+BA+B\bar{B})(\bar{A}+C) \\ &= (A+AB+A\bar{B})(\bar{A}+C) = [A(1+B)+A\bar{B}](\bar{A}+C) && \because B\bar{B}=0 \\ &= (A+A\bar{B})(\bar{A}+C) = A(1+\bar{B})(\bar{A}+C) \\ &= (A.1)(\bar{A}+C) = A(\bar{A}+C) = A\bar{A}+AC = AC && \because A\bar{A}=0 \end{aligned}$$

Example 71.6. *Prove the following Boolean identity* :

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

(Digital Electronic Systems-I, Kurukshetra Univ. 1991)

Solution. Equating the left-hand side expression to Y, we have

$$\begin{aligned} Y &= ABC+A\bar{B}C+AB\bar{C} = AC(B+\bar{B})+AB\bar{C} \\ &= AC+AB\bar{C} && \text{—Law 4} \\ &= A(C+B\bar{C}) \\ &= A(C+B) && \text{—Law 21} \\ &= A(B+C) && \text{—Law 14} \end{aligned}$$

Hence, it proves the given identity.

Example 71.7. *Simplify the following Boolean Expression* :

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

(Digital Computations, Punjab Univ. 1992)

Solution. Bringing together those terms which have *two* common letters, we get

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

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

$= AB + A\bar{B} + \bar{A}BC$ —Law 4

$= A(B + \bar{B}) + \bar{A}BC$ —Law 4

$= A + \bar{A}BC = A + BC$ —Law 21

Example 71.8. *Simplify the following expression and show the minimum gate implementation.*

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

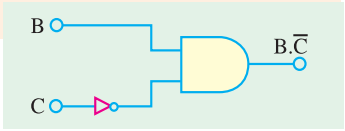


Fig. 71.7

Solution. As seen from *OR* and *AND* laws of Art 67.3, $A + \bar{A} = 1$ and $A \cdot 1 = A$

$\therefore \quad Y = B\bar{C}.\bar{D}(A + \bar{A}) + B\bar{C}.D = B\bar{C}.\bar{D}.1 + B\bar{C}.D.$

$= B.\bar{C}.\bar{D} + B.\bar{C}.D = B.\bar{C}.(D + \bar{D}) = B.\bar{C}.1 = B.\bar{C}.$

Minimum gate implementation is shown by the circuit of Fig. 71.7

Example 71.9. *Simplify the following Boolean expression and draw the logic circuits for the simplified expressions.*

(a) $Y = \bar{A}BC + A\bar{B}C + ABC + B\bar{C}$ ***(b)*** $Y = \bar{B}(A + C) + C(\bar{A} + B) + AC$

Solution. (*a*) $Y = \bar{A}BC + A\bar{B}C + ABC + B\bar{C} = BC(A + \bar{A}) + AC(\bar{B} + B) + B\bar{C}$

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

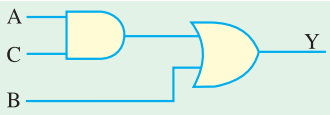


Fig. 71.8

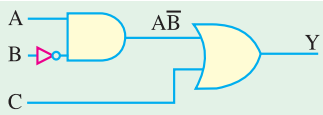


Fig. 71.9

(*b*) $Y = \bar{B}(A + C) + C(\bar{A} + B) + AC = A\bar{B} + \bar{B}C + \bar{A}C + BC + AC$

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

Example 71.10. *Using truth table, prove that* $A + \bar{A}B = A + B$ *and illustrate the equivalence with the help of a switching circuit.*

Solution. Since there are only two variables *A* and *B*, their number of possible combination is $2^2 = 4$ in terms of 0 and 1.

As seen, $\bar{A}$ is the negative of A. In the fourth column of Table 71.1, $\bar{A}$ has been *AND*ed with *B*. In the fifth column, *A* has been *OR*ed with $\bar{A}B$. The values in last column have been obtained by *OR*ing *A* with *B*. By comparing results of column 5 to 6, the equivalence between the two statements can be proved.

Table No. 71.1

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

Switching circuit of Fig. 71.10 (*a*) represents $(A + \bar{A}B)$. In this circuit, when *A* is open, $\bar{A}$ is closed and *vice versa*. It can be shown that his circuit becomes closed with either *A* or *B* is closed.

(*i*) when *A* is closed, then $\bar{A}$ opens. Circuit is completed *via* the upper branch.

(*ii*) keeping *A* open, when we close *B*, the circuit again becomes closed *via* lower branch because $\bar{A}$ is already closed (due to *A* being open).

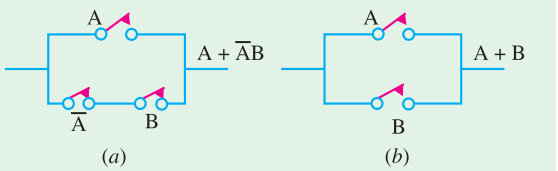


Fig. 71.10

Hence, all that we have to do for closing the circuit of Fig. 71.10 (*a*) is to close either switch *A* or *B*. It is exactly what circuit of Fig. 71.10 (*b*) does.

Example 71.11. *Simplify the expression* : $(AB + C)(AB + D)$

Solution. Let $Y = (AB+C)(AB+D)$

$= ABAB + ABD + ABC + CD$ —Law 19

$= AABB + ABD + ABC + CD$

$= AB + ABD + ABC + CD$ —Law 7

$= AB(1 + D) + ABC + CD = AB + ABC + CD$ —Law 2

$= AB(1 + C) + CD = AB + CD$

$\therefore \quad (AB + C)(AB + D) = AB + CD$

71.5. DE Morgan's Theorem

These two theorems (or rules) are a great help in *simplifying complicated logical expressions.* The theorems can be started as under :

Law 25. $\overline{A + B} = \bar{A} \cdot \bar{B}$ Law 26. $\overline{A \cdot B} = \bar{A} + \bar{B}$

The first statement says that ***the complement of a sum equals the product of complements.*** The second statement says that ***the complement of a product equals the sum of the complements.*** In fact, it allows transformation from a sum-of-products from to a product-of-sum from.

As seen from the above two laws, the procedure required for taking out an expression from under a *NOT* sign is as follows :

1. ***complement the given expression i.e., remove the overall NOT sign***
2. ***change all and ANDs to ORs and all the ORs to ANDs.***
3. ***complement or negate all individual variables.***

As an illustration, take the following example

$\overline{A + BC} = A + BC$ —step 1

$= A(B + C)$ —step 2

$= \bar{A}(\bar{B} + \bar{C})$ —step 3

Next, consider this example,

$\overline{(\bar{A} + B + \bar{C})(\bar{A} + B + C)} = (\bar{A} + B + \bar{C})(\bar{A} + B + C)$ —step 1

$= \bar{A}B\bar{C} + \bar{A}BC$ —step 2

$= \bar{\bar{A}}\bar{B}\bar{\bar{C}} + \bar{\bar{A}}\bar{B}\bar{C}$ —step 3

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

This process is called **demorganization.** It should, however, be noted that opposite procedure would be followed in bringing an expression under the *NOT* sign. Let us bring the expression $\bar{A} + \bar{B} + \bar{C}$ under the *NOT* sign.

$\bar{A} + \bar{B} + \bar{C} = \bar{\bar{A}} + \bar{\bar{B}} + \bar{\bar{C}}$ —step 3

$= A + B + C$

$= ABC$ —step 2

$= \overline{ABC}$ —step 1

Fig. 71.11 shows the circuits to illustrate De Morgan's theorems.

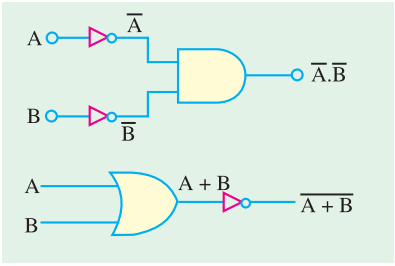


Fig. 71.11

As seen, basic logic function can be either an *OR* gate or an *AND* gate.

Example 71.12. *Demorganize the expression :* $\overline{(A+B)(C+D)}$

Solution. The procedure is as explained above.

$$\overline{(A+B)(C+D)} = (A+B)(C+D) \quad \text{—step 1}$$
$$= (AB)+(CD) \quad \text{—step 2}$$
$$= \overline{AB} + \overline{CD} \quad \text{—step 3}$$

Example 71.13. *Simplify each of the following expressions using De Morgan's theorems :*

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

(b) $\overline{(M+\overline{N})(\overline{M}+N)}$ **(c)** $\overline{\overline{AB}\,\overline{C}\,D}$

De Morgan (1806–1871)

Solution. Please note that there is more than one way of simplifying the expressions given in part (*a*), (*b*) and (*c*).

(a)
$$\overline{A\,\overline{(B+\overline{C})}D} = A\,\overline{(B+\overline{C})}\,D \quad \text{... step 1}$$
$$= A+\overline{(B+\overline{C})}+D \quad \text{... step 2}$$
$$= \overline{A}+\overline{\overline{B+\overline{C}}}+\overline{D} \quad \text{... step 3}$$
$$= \overline{A}+B+\overline{C}+\overline{D} \quad \text{...}(\because\ \overline{\overline{B+\overline{C}}} = B+\overline{C})$$

(b)
$$\overline{(M+\overline{N})(\overline{M}+N)} = (M+\overline{N})(\overline{M}+N) \quad \text{... step 1}$$
$$= (M\overline{N})+(\overline{M}N) \quad \text{... step 2}$$
$$= (\overline{M}\,\overline{\overline{N}})+\overline{\overline{M}}\,\overline{N} \quad \text{... step 3}$$
$$= \overline{M}N+M\overline{N} \quad \text{... step 4}$$

(c)
$$\overline{\overline{AB}\,\overline{C}\,D} = \overline{\overline{AB}\,\overline{C}}\,D \quad \text{... step 1}$$
$$= \overline{\overline{AB}\,\overline{C}}+D \quad \text{... step 2}$$
$$= \overline{\overline{AB}\,\overline{C}}+\overline{D} \quad \text{... step 3}$$
$$= \overline{AB}\,\overline{C}+\overline{D} \quad \text{...}(\because\ \overline{\overline{AB}\,\overline{C}} = \overline{AB}\,\overline{C})$$

The term $\overline{AB}\,\overline{C}$ can be simplified further by De Morganising the term $\overline{AB}$ as $\overline{A}+\overline{B}$. Thus

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

Example 71.14. *Find switching circuits for the following logic expressions :*

(i) $A\,.\,(B+C)$ **(ii)** $A\,\overline{B}+CD$ **(iii)** $(\overline{A}\,B+AC)\,\overline{C}$

(Industrial Electronics, City & Guilds, London)

Solution. (*i*) *A* . (*B* + *C*)

Here, switches *B* and *C* have been *OR*ed *i.e.* connected in parallel. This parallel circuit is connected in series with switch *A* because (*B* + *C*) has been *AND* ed with *A*. hence, the circuit becomes as shown in Fig. 71.12 (*a*). As seen, it is a series-parallel circuit.

(ii) $A\overline{B} + CD$

Here, $A\overline{B}$ has been *OR*ed with CD. We can easily make out that a 2-brached circuit is needed for this logic expression. One branch contains switch A in series with $\overline{B}$ (with one contact point shows raised to indicate negation) and the other branch contains two series-connected switches C and D as show in Fig. 71.12 (*b*).

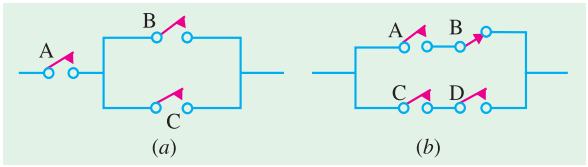


Fig. 71.12

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

From the look of it, we can make out that it consists of a series-parallel circuit as shown in Fig. 71.13. $\overline{A}\,B$ has been *OR*ed with AC *i.e.* $\overline{A}\,B$ and AC are in parallel. Of course, $\overline{A}$ and B are in series in one branch where as A and C are in series in the other branch. Both these parallel branches are in series with $\overline{C}$.

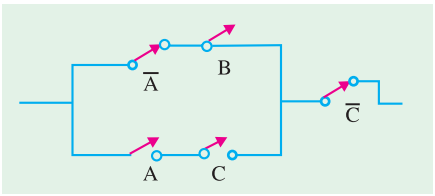


Fig. 71.13

Example 71.15. *Prove that 3-input NAND gate of Fig. 71.14 (a) is equivalent to the bubbled AND gate of Fig. 71.14 (b).*

Solution. The output of *NAND* gate is $\overline{ABC}$ and that of bubbled *OR* gate is $\overline{A} + \overline{B} + \overline{C}$. We have to show that the above two expressions are equivalent.

De Morgan's theorem can be used to prove the above equivalence,

$\overline{ABC} = ABC$ —step 1

$= A + B + C$ —step 2

$= \overline{A} + \overline{B} + \overline{C}$ —step 3

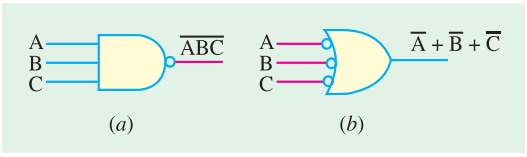


Fig. 71.14

71.6. Duals

Basic duality underlies all Boolean algebra. ***Each expression has its dual which is as true as the original expression.*** For getting the dual of a given Boolean expression, the procedure is to convert

1. all 1*s* to 0*s* and all 0*s* to 1*s*.
2. all *ANDs* to *ORs* and all *ORs* to *ANDs*.

The dual so obtained ***is also found to be true.***

Some of the Boolean relations and their duals are given in Table 71.2.

Table No. 71.2

Relation		*Dual Relation*	
$A \cdot 0$	$= 0$	$A + 1$	$= 1$
$A \cdot A$	$= A$	$A + A$	$= A$
$A \cdot \overline{A}$	$= 0$	$A + \overline{A}$	$= 1$
$A \cdot 1$	$= A$	$A + 0$	$= A$
$A \cdot (A + B)$	$= A$	$A + AB$	$= 1$
$A \cdot (\overline{A} + B)$	$= AB$	$A + \overline{A}B$	$= A + B$

Example 71.16. *Design a logic circuit whose output is HIGH* **only** *when a majority of the inputs A, B and C are HIGH.*

Solution. Since there are three inputs, A, B and C, therefore whenever two or more than two (*i.e.* a majority) inputs are HIGH, the output is HIGH. This situation can be represented in the form of a truth table as shown in Fig. 71.15.

A	B	C	Y	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\rightarrow \bar{A}BC$
1	0	0	0	
1	0	1	1	$\rightarrow A\bar{B}C$
1	1	0	1	$\rightarrow AB\bar{C}$
1	1	1	1	$\rightarrow ABC$

Fig. 71.15

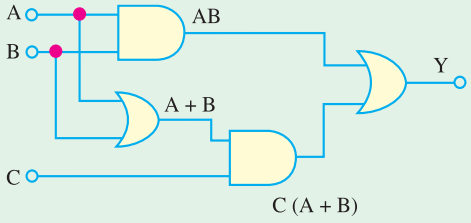


Fig. 71.16

There are four 1*s* in the output column of the truth table. The corresponding binary values are 011, 101, 110 and 111 respectively. Converting these values into product terms and summing up all the terms, we get

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

Adding the term ABC two times from our side in the Boolean expression for the output,

$$Y = AB\bar{C} + A\bar{B}C + \bar{A}BC + ABC + ABC + ABC \qquad (\because ABC + ABC + ABC = ABC)$$

Bringing together those terms which have two common letters, we get,

$$\begin{aligned} Y &= AB\bar{C} + ABC + A\bar{B}C + ABC + \bar{A}BC + ABC \\ &= AB(\bar{C} + C) + AC(\bar{B} + B) + BC(\bar{A} + A) \\ &= AB + AC + BC \qquad (\because \bar{C} + C = \bar{B} + B = \bar{A} + A = 1) \\ &= AB + C(A + B) \end{aligned}$$

The logic circuit that produces the output $Y = AB + C(A + B)$ is as shown in Fig. 71.16.

Alternatively :

You can also arrange the equation,

$$Y = AB + AC + BC$$

as $$Y = A(B + C) + BC \qquad \ldots (i)$$

or $$Y = (A + C)B + AC \qquad \ldots (ii)$$

If you implement equation **(*i*)** or **(*ii*)** using AND and OR logic gates, the number of logic gates is used will still be the same (4).

Example 71.17. *Determine the Boolean expression for the logic circuit shown in Fig. 71.17. Simplify the Boolean expression using Boolean Laws and De Morgan's theorem. Redraw the logic circuit using the simplified Boolean expression.*

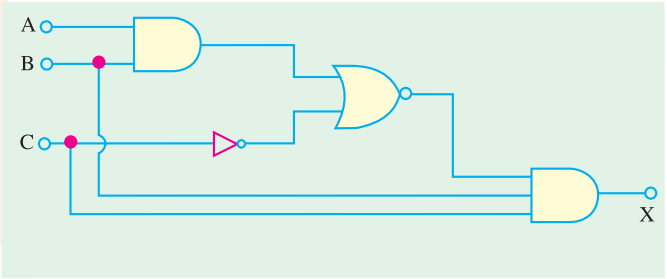


Fig. 71.17

Solution. The output of the given circuit can be obtained by determining the output of each logic gate while working from left to right.

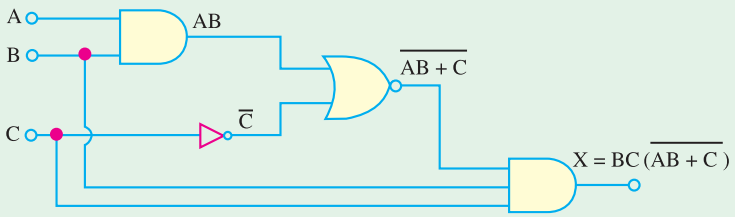


Fig. 71.18

As seen from the logic circuit shown in Fig. 71.18. The output of the circuit is,

$$X = BC\ \overline{(AB + \overline{C})}$$

The output, X can be simplified by De Morganizing the term $\overline{(AB + \overline{C})}$ as shown below.

$$\begin{aligned}
BC\ \overline{(AB + \overline{C})} &= BC\,(A\,B + \overline{C}) && \text{... step 1}\\
&= BC\,(A + B)\,.\,\overline{C} && \text{... step 2}\\
&= BC\,(\overline{A} + \overline{B})\,.\,\overline{\overline{C}} && \text{... step 3}\\
&= BC\,(\overline{A} + \overline{B})\,C && \text{... Law 13}\\
&= BC\,(\overline{A} + \overline{B}) && \text{... Law 7}\\
&= \overline{A}\,BC + BC\,\overline{B}\\
&= \overline{A}\,BC + 0 = && \text{... Law 8}\\
&= \overline{A}\,BC && \text{... Law 1}
\end{aligned}$$

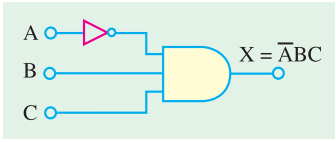


Fig. 71.19

The logic circuit with a simplified Boolean expression $X = \overline{A}\,BC$ is as shown in Fig. 71.19.

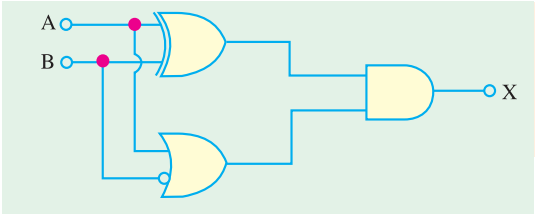


Fig. 71.20

Example 71.18. *Determine the output X of a logic circuit shown in Fig. 71.20. Simplify the output expression using Boolean Laws and theorems. Redraw the logic circuit with the simplified expression.*

Solution. The output of the given logic circuit can be obtained by determining the output of each logic gate while working from left to right.

As seen from Fig. 71.21, the output,

$$\begin{aligned}
X &= (\overline{A}B + A\overline{B})\,(A + \overline{B})\\
&= \overline{A}\,BA + A\,\overline{B}\,A + \overline{A}\,B\,\overline{B} + A\,\overline{B}\,\overline{B}\\
&= 0 + A\,\overline{B}\,A + 0 + A\,\overline{B}\,\overline{B} && \text{... Law 8}\\
&= A\,\overline{B} + A\,\overline{B} && \text{... Law and Law 1}\\
&= A\,\overline{B} && \text{... Law 3}
\end{aligned}$$

Using the simplified Boolean expression, the logic circuit is as shown in Fig. 71.22.

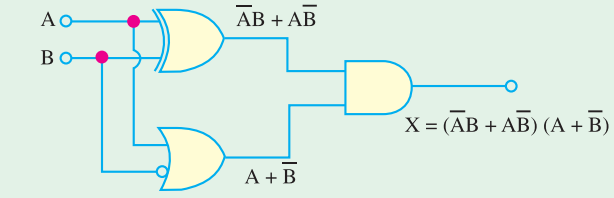


Fig. 71.21

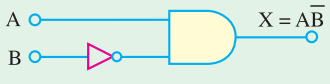


Fig. 71.22

Example 71.19. *Consider the logic circuit shown in Fig. 71.23. Determine the Boolean expression at the circuit output, simplify it. From the simplified Boolean expression, find which logic gate is redundant in the given logic circuit.*

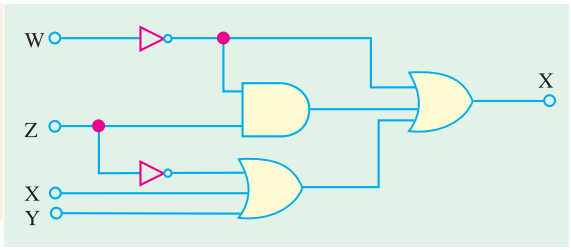


Fig. 71.23

Solution. As seen from Fig. 71.24, the logic circuit output,

$$X = \overline{W} + \overline{W}Z + XY\overline{Z}$$

The expression can be simplified as follows :

$$= \overline{W} + \overline{W}Z + XY\overline{Z}$$

$$= \overline{W}(1 + Z) + XY\overline{Z} \quad \text{—Law 19}$$

$$= \overline{W} + XY\overline{Z} \quad \text{—Law 2}$$

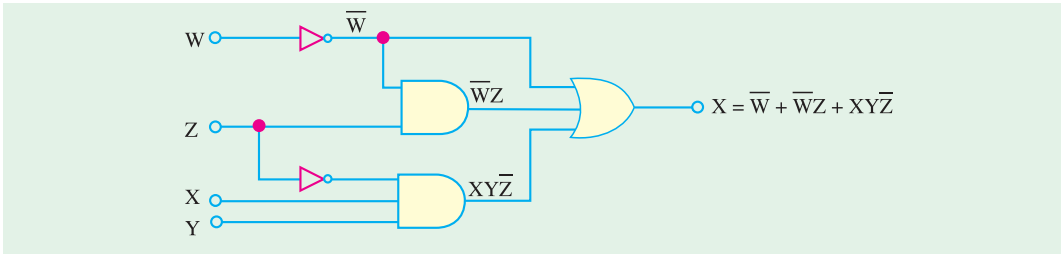


Fig. 71.24

From the simplified Boolean expression of the output $X = \overline{W} + XY\overline{Z}$, and the actual output $\overline{W} + \overline{W}Z + XY\overline{Z}$, we find that the two-input *AND* gate (producing the term $\overline{W}Z$) is redundant.

Example 71.20. *Determine the output of the logic circuit shown in Fig. 71.25. Simplify the output Boolean expression and sketch the logic circuit.*

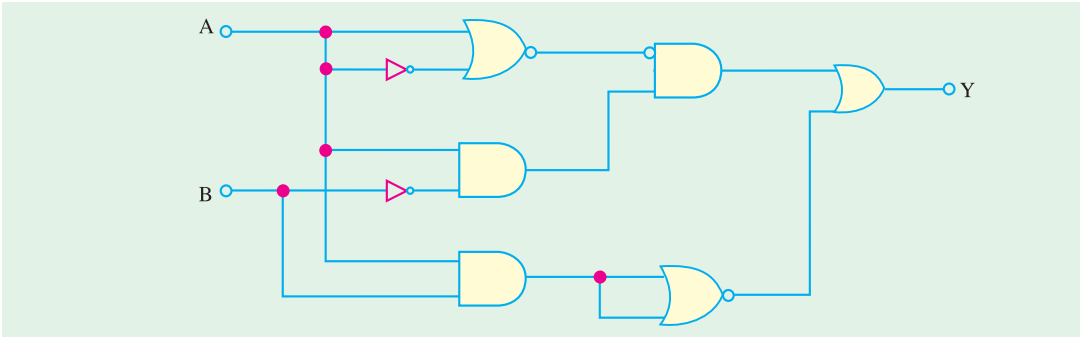


Fig. 71.25

Solution. The output of the circuit can be obtained by determining the output of each logic gate while working from left to right.

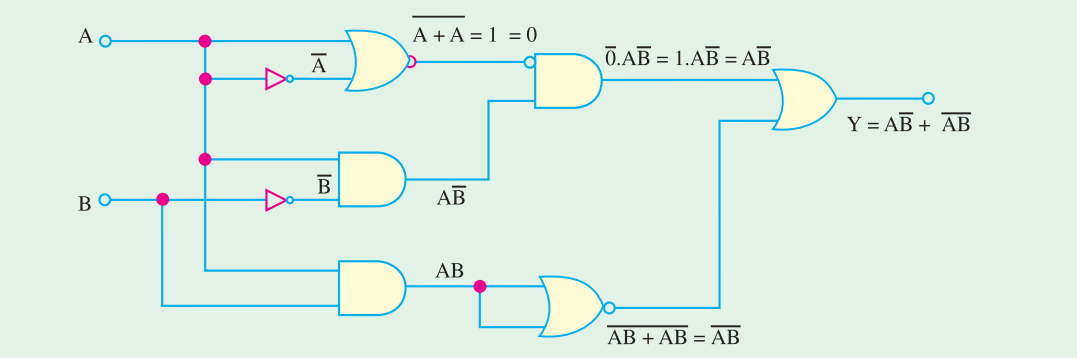


Fig. 71.26

As seen from the circuit shown in Fig. 71.26, we find that the output,

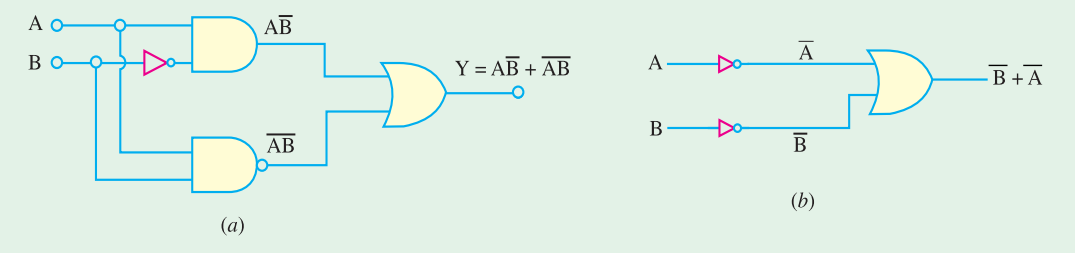


Fig. 71.27

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

The sketch of a logic circuit for the simplified Boolean expression is as shown in Fig. 71.27.

Alternatively :

$$\begin{aligned} y &= A\overline{B} + \overline{A}\overline{B} \\ &= A\overline{B} + (\overline{A} + \overline{B}) \\ &= (A\overline{B} + \overline{B}) + \overline{A} \\ &= (A+1)\overline{B} + \overline{A} \\ &= \overline{B} + \overline{A} \end{aligned}$$

The logic circuit to implement this logic equation is as shown in Fig. 71.27(*b*). Notice the difference in terms of the number and type of logic gates used in the circuits shown in Fig. 71.27. So when you are simplifying and designing logic circuits, it is always possible to have more than one solution.

The circuit shown in Fig. 71.27 shows that it makes use of one inverter (or NOT gate), one AND gate, one NAND gate and one OR gate. In other words, there are four different types of logic gates. However the logic circuit of Fig. 71.27 (*b*) makes use of only three logic gates (two inverters and one OR gate).

71.7. Standard Forms of Boolean Expressions

All Boolean expressions, regardless of their form, can be converted into either of the two following standard forms :

1. Sum-of-products (*SOP*) form and **2.** Product-of-sums (*POS*) form.

The standardization of Boolean expressions makes their evaluation, simplification and implementation much more systematic and easier. Now we shall discuss these two standard forms in more detail.

71.8. The Sum-of-Products (SOP) Form

A product term is a term consisting of the product (or Boolean multiplication) of literals (*i.e.* variables or their complements). When we add two or more product terms, the resulting expression is called, sum-of-products (*SOP*) expression. Some examples of sum-of-products expressions are $A\overline{B} + \overline{A}BC$, $\overline{A}BC + A\overline{C}D + A\overline{B}CD$, $\overline{A}B + AC + \overline{A}\,\overline{B}\,C$

Sometimes, it is convenient to define the set of variables contained in the expression (in either complemented or uncomplemented form) as a ***domain***. For example, domain of the expression $A\,\overline{B}$ + AB is the set of variables A and B. Similarly the domain of the expression $\overline{A}\,BC + ABC + A\,\overline{B}\,CD$ is the set of variables A, B, C and D.

Any logic expression can be changed into *SOP* form by applying Boolean algebra laws and theorems. For example, the expression $A\ (BC + D)$ can be converted to *SOP* form by applying the distributive law :

$$A\,(BC + D) \ = ABC + AD$$

71.9. The Standard SOP Form

So far, we have seen *SOP* (sum-of-products) expressions in which some of the product terms do not contain all the variables in the domain of the expression. For example, the expression, $\overline{A}\,BC + A\,\overline{C}\,D\ + A\,\overline{B}\,CD$ has a domain made up of the variables A, B, C and D. But notice that the first two terms contains only three variables, *i.e.* D or $\overline{D}$ is missing from the first term and B or $\overline{B}$ is missing from the second term.

A standard *SOP* expression is defined as an expression in which all the variables in the domain apper in each product term. For example $\overline{A}\,BCD + AB\,\overline{C}\,D + A\,\overline{B}\,CD$ is a standard *SOP* expression. Standard *SOP* expressions are important in constructing truth tables and in karnaugh map simplification method.

It is very straightforward to convert non-standard product term to a standard *SOP* using Boolean algebra. Each product term in the *SOP* expression that does not contain all the variables in the domain has to be expanded to standard form to include all the variables in the domain and their complements. As stated below, a nonstandard *SOP* expression is converted into standard form using Boolean algebra law 4 : $A\ +\ \overline{A}\ = 1$ *i.e.* a variable added to its complement equals 1.

Step 1. Multiply each nonstandard product term by a term made up of the sum of a missing variable and its complement. This results in two product terms. It is possible because we know that we can multiply anything by 1 without changing its value.

Step 2. Repeat step 1 until all resulting product terms contain all variables in the domain in either complemented or uncomplemented form. Note that in converting a product term to a standard form, the number of product terms is doubled for each missing variable.

For example, suppose we want to convert the Boolean expression $\overline{A}\,BC + A\,\overline{C}\,D + A\,\overline{B}\,CD$ to a standard *SOP* form. Then following the above procedure we proceed as below :

$$\overline{A}\,BC\,(D + \overline{D}\,) + A\,(B + \overline{B}\,)\,\overline{C}\,D + A\,\overline{B}\,CD$$

$$= \overline{A}\,BCD + \overline{A}\,BC\,\overline{D}\ + AB\,\overline{C}\,D + A\,\overline{B}\ \overline{C}\,\mathrm{D} + A\,\overline{B}\,CD$$

The expression given above is a standard *SOP* expression.

71.10. The Product-of-sums (POS) Form

The sum term is a term consisting of the sum (or Boolean addition) of literals (*i.e.* variables or their complements). When we multiply two or more sum terms, the resulting expression is called product-of-sums (*POS*). Some examples of *POS* form are (A + $\overline{B}$) ($\overline{A}$ + B + C), ($\overline{A}$ + B + C) (A + $\overline{C}$ + D) (A + $\overline{B}$ + C + D) and ($\overline{A}$ + $\overline{B}$) (*A* + *C*) ($\overline{A}$ + $\overline{B}$ + *C*) .

It may be carefully noted that a *POS* expression, can contain a single-variable term as in $\overline{A}$ (*A* + $\overline{B}$ + *C*) (*B* + $\overline{C}$ + $\overline{D}$). In *POS* expression, a single overbar cannot extend over more than one variable, although more than one variable in a term can have an overbar.

71.11. The Standard POS Form

So far, we have seen *POS* (product-of-sums) expressions in which some of the sum terms do not contain all the variables in the domain of the expression. For example, the expression, ($\overline{A}$ + *B* + *C*), (*A* + $\overline{C}$ + *D*) (*A* + $\overline{B}$ + *C* + *D*) has a domain made up of variables, *A*, *B*, *C* and *D*. Notice that the complete set of variables in the domain is not represented in the first two terms of the expression, *i.e.* *D* or $\overline{D}$ is missing in the first term and *B* or $\overline{B}$ is missing in the second term.

A standard *POS* expression is defined as an expression in which all the variables in the domain appear in each sum term. For example ($\overline{A}$ + *B* + *C* + *D*) (*A* + *B* + $\overline{C}$ + *D*) (*A* + $\overline{B}$ + *C* + *D*) is a standard *POS* form. Any nonstandard *POS* expression can be converted to a standard form using Boolean algebra.

Each sum term in an *POS* expression that does not contain all the variables in the domain can be expanded to standard form to include all variables in the domain and their complements. As stated below : a nonstandard *POS* expression is converted into a standard form using Boolean algebra Law 8 : *A* . $\overline{A}$ = 0, *i.e.* a variable multiplied by its complemented equals 0.

Step 1. Add to each nonstandard product term a term made up of the ptoduct of a missing variable and its complement. This results in two sum terms. This is possible because we know that we can add 0 to anything without changing its value.

Step 2. Apply law 20, *i.e.* *A* + *BC* = (*A* + *B*) (*A* + *C*)

Step 3. Repeat Step 1 until all resulting sum terms contain all variables in the domain in either complemented or uncomplemented form.

For example we want to convert the Boolean expression,

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

into a standard *POS* form. Then following the above prodcedure, we proceed as below :

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

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

The expression given above is a standard *POS* expression

71.12. The Karnaugh Map

The Karnaugh map (or simply a *K*-map) is similar to a truth table because it presents all the possible values of input variables and the resulting output for each value. However, instead of being organised into columns and rows like a truth table, the Karnaugh map is an array of squares (or cells) in which each square represents a binary value of the input variables. The squares are arranged in a way so that simplification of given expression is simply a matter of grouping the squares. Karnaugh maps can be used for expression with two-three, four, and five variable Karnaugh maps to illustrate the principles. Karnaugh map with five-variables is beyond the scope of this book. For higher number of

variables, a Quine-McClusky method can be used. This method is also beyond the scope of this book.

The number of squares in a Karnaugh map is equal to the total number of possible input variable combinations (as is the number of rows in a truth table). For two variables, the number of square is $2^2 = 4$, for three variables, the number of squares is $2^3 = 8$ and for four variables, the number of squares is $2^4 = 16$.

71.13. The Two-variable Karnaugh Map

Fig. 71.28 (*a*) shows a two-variable Karnaugh map. As seen, it is an array of four squares. In this cass, A and B are used for two variables although any other two letters could be used. The binary values of A (*i.e.* 0 and 1) are indicated along the left side as $\overline{A}$ and A (notice the sequence) and the binary values of B are indicated across the top as $\overline{B}$ and B. The value of a given square is the value of A at the left in the same row combined with the value of B at the top in the same column. For example, a square in the upper left corner has a value of $\overline{A}\,\overline{B}$ and a square in the lower right corner has a value of $A\,B$. Fig. 71.28 (*b*) shows the standard product terms represented by each square in the Karnaugh map.

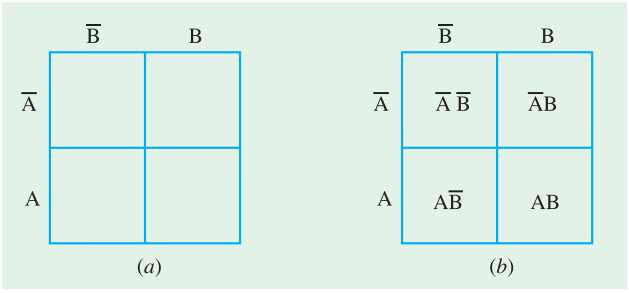


Fig. 71.28

71.14. The Three-variable Karnaugh Map

Fig. 71.29 (*a*) shows a three-variable Karnaugh map. As seen it is an array of eight squares. In this case, A, B and C are used for the variables although any other three letters could be used. The value of A and B are along the left side (notice the sequence carefully) and the values of C are across the top.

The value of a given square is the values of A and B at the left in the same row combined with the value of C at the top in the same column. For example, a square in the upper left corner has a value of $\overline{A}\,\overline{B}\,\overline{C}$ and a square in the bottom right corner has a value of $A\,\overline{B}\,C$. Fig. 71.29 (*b*) shows the product terms that are represented by each square in the Karnaugh map.

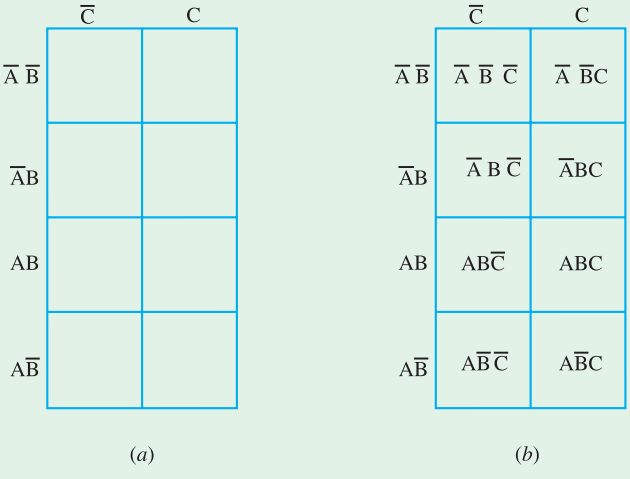


Fig. 71.29

71.15. The Four-variable Karnaugh Map

Fig. 71.30 (*a*) shows a four-variable Karnaugh map. As seen, it is an array of sixteen squares. In this case *A*, *B*, *C* and *D* are used for the variables. The values of *A* and *B* are along the left side, and the values of *C* and *D* are across the top. The sequence of the variable values may be noted carefully.

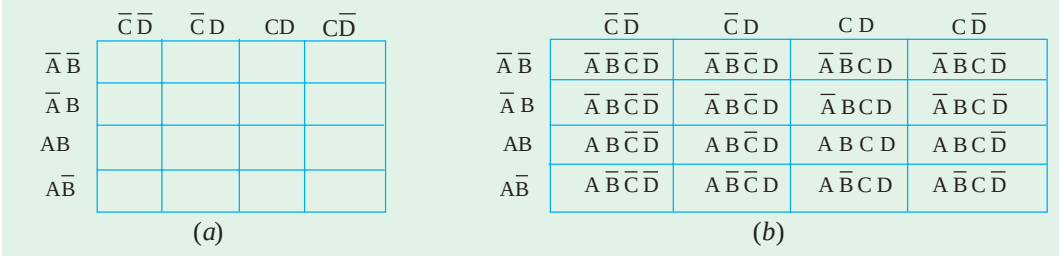


Fig. 71.30

The value of a given square is the values of *A* and *B* at the left in the same row combined with the values of *C* and *D* at the top in the same column. For example, a square in the upper right corner has a value $\overline{A}\,\overline{B}\,C\,\overline{D}$ and a square in the lower left corner has a value $A\,\overline{B}\,C\,\overline{D}$. Fig. 71.30 (*b*) shows the standard product terms that are represented by each square in the four-variable Karnaugh map.

71.16. Square Adjacency in Karnaugh Map

We have already discussed a two-variable Karnaugh map, a three-variable Karnaugh map and a four-variable Karnaugh map. Now we shall discuss the concept of square adjacency in a Karnaugh map.

It will be interesting to know that the squares in a Karnaugh map are arranged in such a way that there is only a single-variable change between adjacent squares. Adjacency is defined as a single-variable change. It means the squares that differ by only one variable are adjacent. For example, in a three-variable Karnaugh map shown in Fig. 71.29 (*b*), the $\overline{A}\,B\,\overline{C}$ square is adjacent to $\overline{A}\,\overline{B}\,\overline{C}$ square, the $\overline{A}BC$ square and the $AB\overline{C}$ square. It may be carefully noted that square with values that differ by more than one variable are not adjacent. For example, the $\overline{A}\,B\,\overline{C}$ square is not adjacent to the $\overline{A}\,\overline{B}\,C$ square, the ABC square, the $A\,\overline{B}\,\overline{C}$ square or the $A\,\overline{B}\,C$ square. In other words, each square is adjacent to the squares that are immediately next to it on any of its four sides. However, a square is not adjacent to the squares that diagonally touch any of its corners.

It may also be noted that squares in the top row are adjacent to the corresponding squares in the bottom row and squares in the outerleft column are adjacent to the corresponding squares in the outer right column. This is called "**wrap around**" adjacency because we can think of the map as wrapping around from top to bottom to form a cylinder or from left to right to form a cylinder. Fig. 71.31 (*a*) and (*b*) shows the square adjacencies with a three-variable and a four-variable Karnaugh maps respectively.

Notice the square adjacencies in a four variable Karnaugh map :

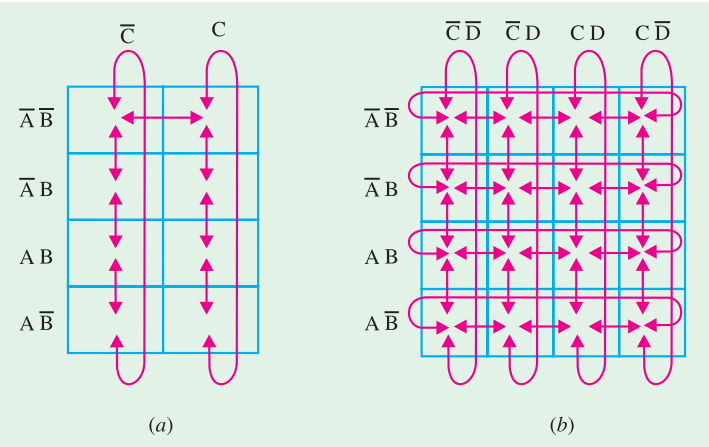


Fig. 71.31

Here for example, the square $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ is adjacent to $\overline{A}\,\overline{B}\,\overline{C}\,D$ square, $\overline{A}\,B\,\overline{C}\,\overline{D}$ square, $A\,\overline{B}\,\overline{C}\,\overline{D}$ square and $A\,\overline{B}C\,\overline{D}$ square. Similarly $\overline{A}\,B\,\overline{C}\,D$ square is adjacent to $\overline{A}\,\overline{B}\,\overline{C}\,D$ square, $\overline{A}\,B\,\overline{C}\,\overline{D}$ square, $\overline{A}BCD$ square and $AB\overline{C}D$ square.

71.17. Mapping a Standard SOP Expression on the Karnaugh Map

Consider a *SOP* (sum-of-products) expression. $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} + A\overline{B}C + A\,\overline{B}\,\overline{C}$. In order to map this expression on the Karnaugh map, we need a three variable Karnaugh-map because the given expression has three variables *A*, *B*, and *C*. Then select the first product term $\overline{A}\,\overline{B}\,\overline{C}$ and enter 1 in the corresponding square (*i.e.* the first row and the first column) as shown in Fig. 71.32.

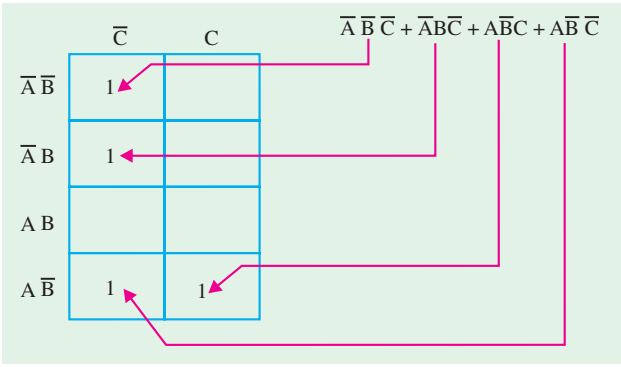


Fig. 71.32

Similarly, for the second product term, $\overline{A}\,B\,\overline{C}$ place a 1 in the second row and first column. Repeat this process for the other two product terms, *i.e.* $A\,\overline{B}\,C$ and $A\,\overline{B}\,\overline{C}$. The squares that do not have 1 are the squares for which the expression is 0. Usually when working with sum-of-products expressions, the 0*s* are left off the map.

Example 71.21. *Map the following SOP expression on the Karnaugh map :* $\overline{A}\,\overline{B}\,C + \overline{A}BC + AB\overline{C} + ABC$.

Solution. Sketch a three variable Karnaugh map as shown in Fig. 71.33. Select the first product term $\overline{A}\,\overline{B}\,C$ and enter 1 in the corresponding square. Similarly enter 1 for the other product terms in the given *SOP* expression. Check that number of 1*s* in the Karnaugh map is equal to the number of product terms in the given *SOP* expression.

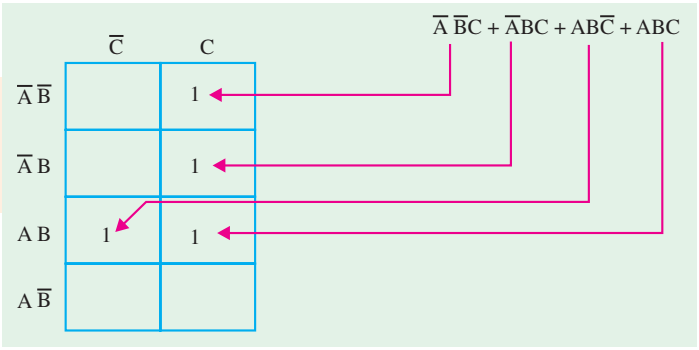


Fig. 71.33

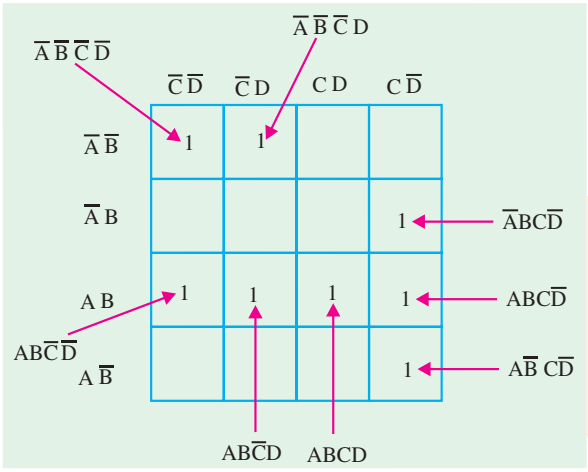


Fig. 71.34

Example 71.22. *Map the following standard sum-of-products (SOP) expression on a Karnaugh map :*

$$\overline{A}\,B\,C\,\overline{D} + AB\,C\,\overline{D} + A\,B\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + A\,B\,\overline{C}D + \overline{A}\,\overline{B}\,\overline{C}\,D + A\,\overline{B}C\,\overline{D} + ABCD$$

Solution. Sketch a four variable Karnaugh map. Select the first product term $\overline{A}\,B\,C\,\overline{D}$ from the given *SOP* expression and enter 1 in the corresponding square as shown in Fig. 71.34. Similarly enter 1 for the other product terms in the given *SOP* expression. Check that number of 1*s* in the Karnaugh Map is equal to the number of product terms in the given *SOP* expression.

71.18. Mapping a Nonstandard SOP Expression on the Karnaugh Map

A nonstandard sum-of-products (*SOP*) expression is a one that has product terms with one or more missing variables. In such a case, Boolean expression must first be converted to a standard form by a procedure explained in Art. 71.9.

Let us consider an example to illustrate the procedure for mapping a nonstandard *SOP* expression on the Karnaugh map. Suppose we have the *SOP* expression :

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

As seen, this expression is obviously not in standard form because each product term does not have three variables. The first term is missing two variables, the second term is missing one variable and the third term is standard. In order to convert the given nonstandard *SOP* expression to a standard form, we multiply the first product term by $B + \overline{B}$ and $C + \overline{C}$, and the second term by $C + \overline{C}$. Expanding the resulting expression, we get,

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

$$= \overline{A}BC + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C + A\,\overline{B}\,\overline{C} + AB\overline{C}$$

Rearranging the expression for our convenience, we get

$$\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}BC + AB\overline{C} + A\,\overline{B}\,\overline{C} + A\overline{B}C$$

This expression can be mapped on a three-variable Karnaugh map as shown in Fig. 71.35.

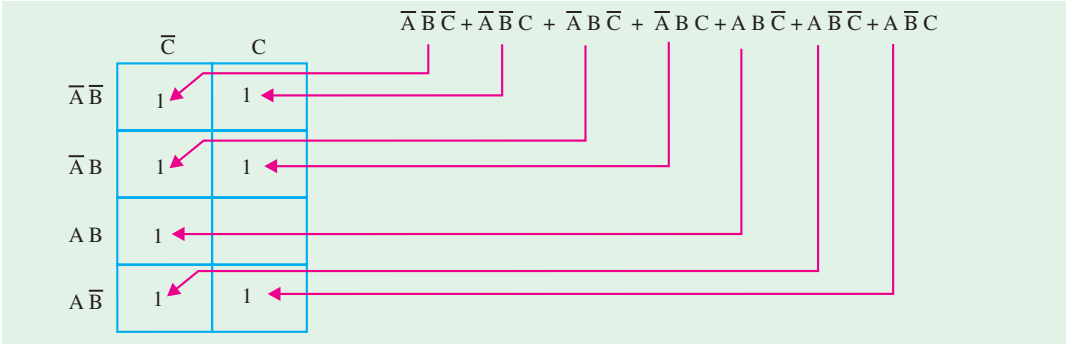


Fig. 71.35

71.19. Simplification of Boolean Expression Using Karnaugh Map

The process that results in an expression containing the minimum number of possible terms with the minimum number of variables is called **simplification** or **minimization.** After the *SOP* (or the Boolean) expression has been mapped on the Karnaugh map, there are three steps in the process of obtaining a minimum *SOP* expression. The three steps are : **(*a*)** grouping the 1*s*, **(*b*)** determining the product term for each group and **(*c*)** summing the resulting product terms.

(*a*) *Grouping the 1s* : We can group the 1*s* on the Karnaugh map according to the following rules by enclosing those adjacent squares containing 1*s*. The objective is to maximize the size of the groups and to minimize the number of groups.

1. A group must contain either 1, 2, 4, 8 or 16 squares. In the case of two-variable Karnaugh map, 4 squares is the maximum group, for three-variable map, 8 squares are the maximum group and so on.
2. Each square in the group must be adjacent to one or more squares in that same group but all squares in the same group do not have to be adjacent to each other.
3. Always include the largest possible number of 1*s* in a group in accordance with rule 1.
4. Each 1 on the Karnaugh map must be included in at least one group. The 1*s* already in a group can be included in another group as long as the overlapping groups include non-common 1*s*.

(*b*) *Determining the Product Term for each group* : Following are the rules that are applied to find the minimum product terms and the minimum sum-of-products expression :

1. ***Group the squares that have 1s* :** Each group of squares containing 1*s* creates one product term composed of all variables that occur in only one form (either uncomplemented or complemented) within the group. Variables that occur both uncomplemented and complemented within the group are eliminated. These are known as contradictory variables.
2. In order to determine the minimum product term for each group, we need to look at the standard methodology for three-variable and four-variable Karnaugh map respectively.
   - **(*i*) *For a three-variable K-map* :** **(1)** for 1-square, group we get a three-variable product term, **(2)** for a 2-square group, we get a two-variable product term, **(3)** for a 4-square product term, we get a one-variable product term.
   - **(*ii*) *For a four-variable K-map* : (1)** For a 1-square group, we get a four-variable product term, **(2)** for a 2-square group, we get a three-variable product term, **(3)** for a 4-square group, we get a two-variable product term and **(4)** for a 8-square group, we get a one-variable product term.

(*c*) *Summing the resulting product terms* : When all the minimum product terms are derived from the Karnaugh map, these are summed to form the minimum sum-of-products expression.

Note : In some cases, there may be more than one way to group the 1*s* to form the product terms. Whatever be the way, the minimal expression must have the same number of product terms and each product term, the same number of Boolean variables.

The examples given below will help you to understand and apply the simplification of the *SOP* expression using Karnaugh map.

Example 71.23. *Simplify the following Boolean expression using the Karnaugh mapping technique* :

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

Solution. The first step is to map the given Boolean expression on the Karnaugh map. Notice that there are three variables *A*, *B* and *C* in the Boolean expression, therefore we need a three-variable Karnaugh map.

The Boolean expression to be mapped is,

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

Note that the given Boolean expression is a nonstandard *SOP* expression because the first product term $\overline{A}\,B$ has the variable *C* missing in it. This can be converted into a standard *SOP* form by modifying the expression as below.

$$\begin{aligned} X &= \overline{A}\,B.1 + \overline{A}\,\overline{B}C + A\,B\,\overline{C} + A\,\overline{B}\,\overline{C} \\ &= \overline{A}\,B\,(C + \overline{C}) + \overline{A}\,\overline{B}C + A\,B\,\overline{C} + A\,\overline{B}\,\overline{C} && \ldots(C + \overline{C} = 1) \\ &= \overline{A}\,BC + \overline{A}\,B\,\overline{C} + \overline{A}\,\overline{B}C + A\,B\,\overline{C} + A\,\overline{B}\,\overline{C} && \ldots(1) \end{aligned}$$

Equation (1) can be mapped on the Karnaugh map as shown in Fig. 71.36. In order to simplify the given expression, the 1*s* can be grouped together as shown by the loop around the 1*s*. The four 1*s* in the first column are grouped together and the term we get is $\overline{C}$. This is because of the fact that the squares within this group contain both A and $\overline{A}$ and B and $\overline{B}$, so these variables are eliminated. Similarly, the two 1*s* in the second row are grouped together and the term we get is $\overline{A}B$. This is because of the fact that squares in this group contain both C and $\overline{C}$ which is eliminated. Summing up the two-product terms, the simplified expression is,

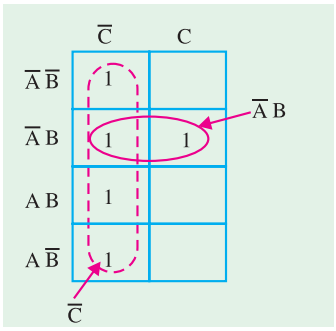


Fig. 71.36

$$X = \overline{A}B + \overline{C}.$$

Example 71.24. *Simplify the following SOP expression using the Karnaugh mapping procedure :*

$$X = \overline{A}\,B\overline{C}\,D + A\,\overline{B}\,\overline{C}\,D + \overline{A}\,\overline{B}\,C\,D + A\,B\,\overline{C}\,D + A\,B\,\overline{C}\,\overline{D} + ABCD$$

Solution. First of all, notice that the given *SOP* expression is already in the standard from *i.e.* all the product terms in the given expression have all the four variables *A*, *B*, *C* and *D*.

Next sketch a four-variable Karnaugh map. Select the first product term ($\overline{A}\,B\overline{C}\,D$) from the given expression and enter 1 in the corresponding square as shown in Fig. 71.37. Similarly enter 1 for the other product terms in the given *SOP* expression to complete the mapping. In order to simplify the given *SOP* expression, the 1*s* can be grouped together as shown by the loop around the 1*s*. The four 1*s* in the second column are grouped together and the product term we get is $\overline{C}\,D$. This is because of the fact that squares within this group contain both A and $\overline{A}$ and B and $\overline{B}$, so these variables are eliminated.

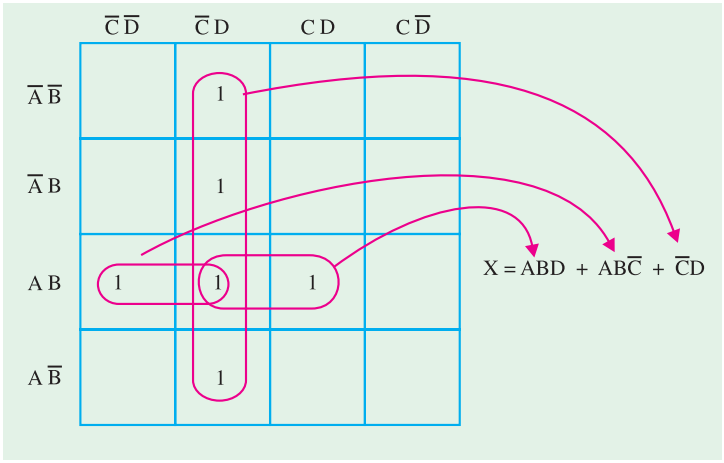


Fig. 71.37

The two 1*s* in the first and second columns can be grouped together. This group contains both D and $\overline{D}$, so this variable is eliminated and the resulting product term is $AB\overline{C}$. Similarly the two 1*s* in the second and third columns can be grouped together. This group contains both C and $\overline{C}$, so this variable is eliminated and the resulting product term is ABD.

The resulting minimal or simplified *SOP* expression is obtained by summing up the three product terms $\overline{C}\,D$, $AB\overline{C}$ and ABD as shown below :

$$X = ABD + AB\overline{C} + \overline{C}\,D.$$

Example 71.25. *Simplify the following SOP expression using the Karnaugh mapping technique.*

$$X = B\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D + AB\overline{C}D + \overline{A}BCD + ABCD$$

Solution. First of all, notice that the given *SOP* expression is in the nonstandard *SOP* form because the first product term ($B\overline{C}\,\overline{D}$) has a variable A or $\overline{A}$ missing in it. Let us convert the given *SOP* expression into a standard *SOP* form as shown below :

$$X = 1.B\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D + AB\overline{C}D + \overline{A}BCD\;ABCD$$
$$= (\overline{A} + A)\,B\,\overline{C}\,\overline{D} + \overline{A}\,B\,\overline{C}\,D + AB\overline{C}D + \overline{A}BCD + ABCD \quad ...(\because \overline{A} + A = 1)$$
$$= \overline{A}B\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}D + AB\overline{C}D + \overline{A}BCD + ABCD$$

This expression can be mapped on to the four-variable Karnaugh by entering 1 for each product term in the corresponding square as shown in Fig. 71.38.

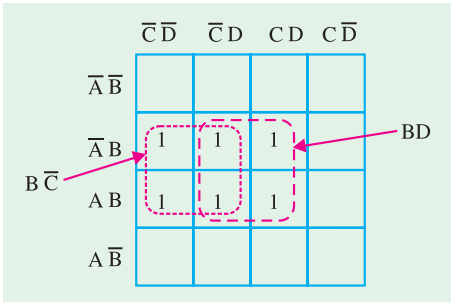


Fig. 71.38

In order to simplify the *SOP* expression, the 1*s* can be grouped together as shown by the loop around the 1*s*. The four 1*s* looped together form the first and second cloumns, contain both A and $\overline{A}$ and D and $\overline{D}$, so these variables are eliminated and the resulting product term is $B\overline{C}$. Similarly, the four 1*s* looped together form the second and the third column contain both A and $\overline{A}$ and C and $\overline{A}$, so these variables are eliminated and the resulting product term is BD. The resulting simplified *SOP* expression is the sum of the product terms $B\overline{C}$ and BD, *i.e.*,

$$X = B\overline{C} + BD$$

Example 71.26. *Fig. 71.39 shows a Karnaugh map of a sum-of-products (SOP) function. Determine the simplified SOP function.* **(UPSC Civil Services 2000)**

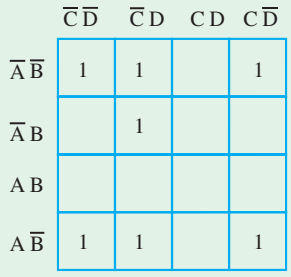


Fig. 71.39

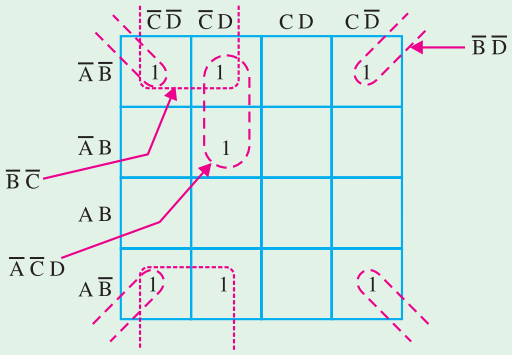


Fig. 71.40

Solution. The grouping of 1*s* is as shown in the Fig. 71.40. Notice the "wrap around" four-square group that includes the 1*s* on four corners of the Karnaugh map. This group produces a product term $\overline{B}\,\overline{D}$. This is determined by observing that the group contains both A and $\overline{A}$ and C and $\overline{C}$, so these variables are eliminated.

Another group of four with "wrap-around" adjacency is formed form the top and the bottom rows of the Karnaugh map. This group overlaps with the previous group and produces a product term $\overline{B}\,\overline{C}$. This is determined by observing that this group contains both A and $\overline{A}$ and D and $\overline{D}$, so these variables are eliminated.

The remaining 1 is absorbed in a overlapping group of two squares. This group produces a three-variable term $\overline{A}C\overline{D}$. This is determined by observing that this group contains both B and $\overline{B}$, so this variable is eliminated. This resulting simplified *SOP* function is the sum of the product terms $\overline{B}\,\overline{D}$ + $\overline{B}\,\overline{C}$ and $\overline{A}C\overline{D}$, *i.e.*

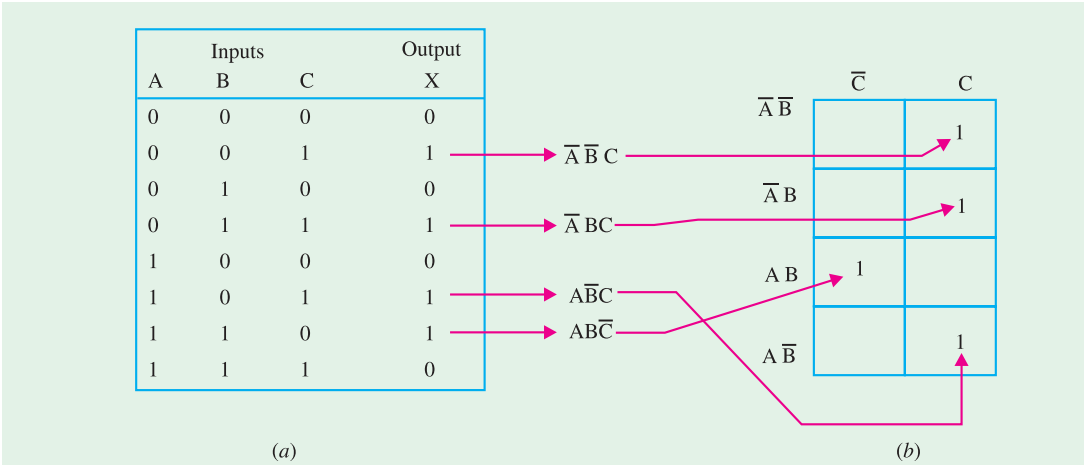
$$X = \overline{B}\,\overline{D} + \overline{B}\,\overline{C} + \overline{A}C\overline{D}$$

71.20. Mapping Directly on Karnaugh Map from a Truth Table

It is possible to map directly on Karnaugh map from a truth table. Recall that a truth table gives the output of a Boolean expression for all possible input variable combinations. Let us illustrate direct mapping through an example of a Boolean expression and its truth table representation.

Let $X = \overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}C + AB\overline{C}$. Then its truth table can be indicated as shown in Fig. 71.41 (*a*) and its Karnaugh mapping is shown in Fig. 71.41 (*b*). Notice in the truth table that the output X is 1 for four different input variable combinations.

It is evident from the Fig. 71.41 (*a*) and (*b*) that truth table and Karnaugh map are simply different ways to represent a logic function.



Inputs			Output
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Fig. 71.41

Example 71.27. *Implement the following Boolean expression using minimum number of 3-input NAND gates.*

$$f(A, B, C, D) = \Sigma(1, 2, 3, 4, 7, 9, 10, 12)$$ **(UPSC Engg. Services 1991)**

Solution. The given Boolean function indicates that its output is 1 corresponding to the terms indicated within the expression *i.e.*, 1, 2, 3, 4, 7, 9, 10 and 12. This is shown in Fig. 71.42 (*a*). We can map these values directly on to the four-variable Karnaugh map as shown in Fig. 71.42 (*b*). In order to simplify the Boolean expression represented on the Karnaugh map, group the 1*s* as shown in the Fig. 71.42 (*b*). The group of two squares in the first coloum produces a product term $B\,\overline{C}\,\overline{D}$. This is determined by observing that the group contains both A and $\overline{A}$, so this variable is eliminated. Another group of two squares in the second column produces term $\overline{B}\,CD$. The variable A is eliminated because the group contains both A and $\overline{A}$.

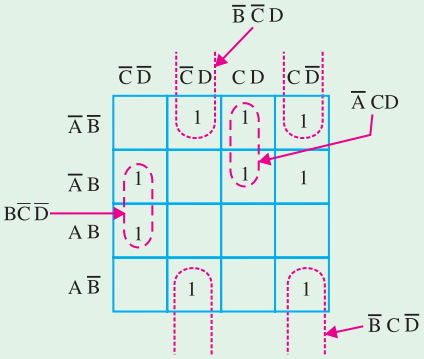
Another group of two squares in the third column produces the term $\overline{A}\,CD$. The variable B is eliminated because the group contains both B and $\overline{B}$. Still another group of two squares in the fourth column produces the term $\overline{B}C\,\overline{D}$. The variable A is eliminated because the group contains both A and $\overline{A}$. Thus the resulting simplified expression,

$$f(A, B, C, D) = B\overline{C}\,\overline{D} + \overline{B}\overline{C}\,D + \overline{A}C\,D + \overline{B}C\,\overline{D}$$

This can be implemented using 3-input NAND gate as shown in Fig. 71.43.

Decimal Number	Inputs A	B	C	D	Output f	
0	0	0	0	0	0	
1	0	0	0	1	1	$\bar{A}\,\bar{B}\,\bar{C}\,D$
2	0	0	1	0	1	$\bar{A}\,\bar{B}\,C\,\bar{D}$
3	0	0	1	1	1	$\bar{A}\,\bar{B}\,C\,D$
4	0	1	0	0	1	$\bar{A}\,B\,\bar{C}\,\bar{D}$
5	0	1	0	1	0	
6	0	1	1	0	0	
7	0	1	1	1	1	$\bar{A}\,B\,C\,D$
8	1	0	0	0	0	
9	1	0	0	1	1	$A\,\bar{B}\,\bar{C}\,D$
10	1	0	1	0	1	$A\,\bar{B}\,C\,\bar{D}$
11	1	0	1	1	0	
12	1	1	0	0	1	$A\,B\,\bar{C}\,\bar{D}$
13	1	1	0	1	0	
14	1	1	1	0	0	
15	1	1	1	1	0	

(*a*)



(*b*)

Fig. 71.42

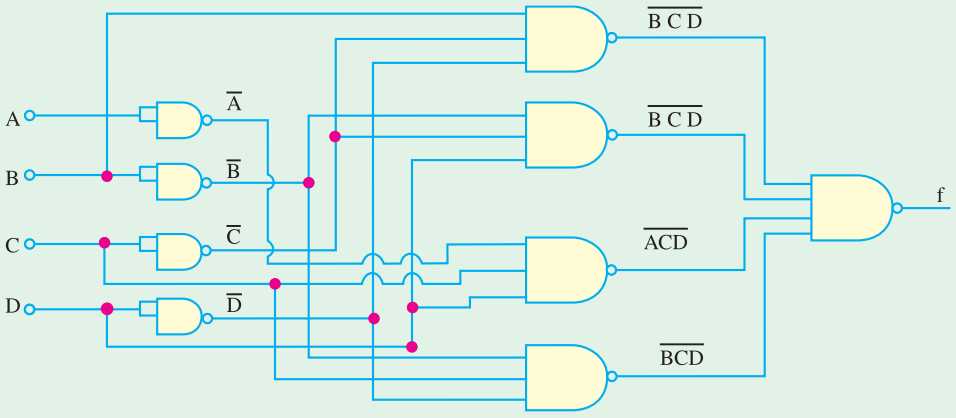


Fig. 71.43

71.21. "Don't Care" conditions

In digital systems design sometimes a situation arises in which some input variable conditions are not allowed. For example in a *BCD* (binary coded decimal) code, there are six invalid combinations : 1010, 1011, 1100, 1101, 1110 and 1111. Since these unallowed states will never occur in an application involving the *BCD* code, they can be treated as "don't care" terms with respect to their effect on the output. That is, for these "don't care" terms either 1 or 0 may be assigned to the output.

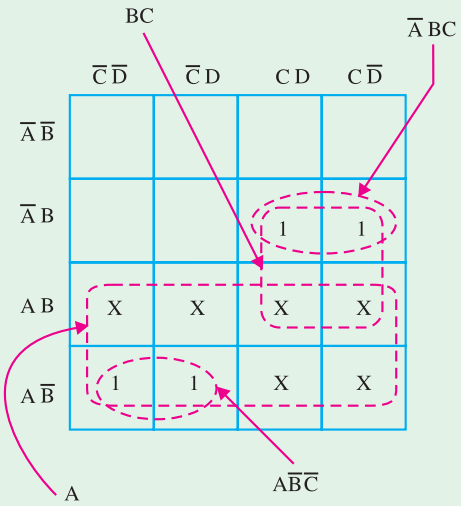
Now, we shall discuss as how the "don't care" terms can be used to advantage on the Karnaugh map for simplifying the Logic equations.

Consider for example, a combinational circuit which products a '1' output corresponding to a

BCD input equal and greater than 6. The output is 0 corresponding to a *BCD* input less than 6. The truth table for this situation is as shown in Fig. 71.44 (*a*).

Inputs				Output	
A	*B*	*C*	*D*	*X*	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	0	
0	1	1	0	1	→ $\overline{A}BC\overline{D}$
0	1	1	1	1	→ $\overline{A}BCD$
1	0	0	0	1	→ $A\overline{B}\overline{C}\overline{D}$
1	0	0	1	1	→ $A\overline{B}\overline{C}D$
1	0	1	0	*X*	
1	0	1	1	*X*	
1	1	0	0	*X*	
1	1	0	1	*X*	
1	1	1	0	*X*	
1	1	1	1	*X*	

(*a*)



(*b*)

Fig. 71.44

We know that we can place 1*s* directly from the truth table on the Karnaugh map. Similarly we can place *X* for the "don't care" enteries directly on the Karnaugh map as shown in Fig. 71.44 (*b*). When grouping the 1*s*, *X*s can be treated as 1*s* make a larger grouping or as 0*s*, if they cannot be used to advantage Recall the larger the group of 1*s* the simpler the resulting term will be. Taking advantage of the "don't care" and using them as 1*s*, the resulting expression for the output is $A + BC$. However, if the "don't cares" are not used as 1*s*, the resulting expression is $A\overline{B}\,\overline{C} + \overline{A}BC$. Thus we can see the advantage of using "dont care" terms to get the simplest logic expression.

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1		*X*	*X*
$\overline{A}B$		1	*X*	1
AB		*X*	1	
$A\overline{B}$	1	*X*		*X*

Fig. 71.45

Example 71.28. *Consider the Karnaugh map shown in Fig. 71.45. Determine the logic function represented by the map and simplify it in the minimal form.* **(UPSC Engg. Services 1997)**

Solution. We know that when grouping the 1*s*, X*s* can be treated as 1*s* to make a larger grouping or as 0*s* if they cannot be used to advantage. Recall, the larger the group of 1*s*, the simpler the resulting term will be. Taking advantage of the *X*s and using them as 1*s*, the grouping of 1*s* and *X*s is as shown in Fig. 71.46.

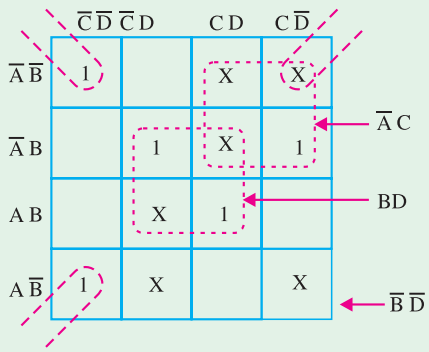


Fig. 71.46

Notice the "wrap-around" four-square group that includes the 1*s* and *X*s on fours corners of the Karnaugh map. This group produces a product term $\overline{B}\,\overline{D}$. This is determined by observing that the group contains both A and $\overline{A}$ and C and $\overline{C}$, so these variables are eliminated. Another group of four squares containing 1*s* and *X*s around the centre of Karnaugh map is formed.

This group produces a product term BD. This is determined by observing that the group contains both A and $\overline{A}$ and C and $\overline{C}$, so these variables are eliminated.

Another group of four containing 1*s* and *X*s is formed near the top right corner of Karnaugh map. This group overlaps with the previous group and produces a product term $\overline{A}\,C$. This is determined by observing that the group contains both B and $\overline{B}$ and D and $\overline{D}$, so these variables are eliminated.

The resulting simplified logic function is the sum of the three product terms : $\overline{B}\,\overline{D}$, BD and $\overline{A}\,C$, *i.e.*,

$$X = \overline{B}\,\overline{D} + BD + \overline{A}\,C$$

71.22. Main Logic Families

Most digital systems are designed by combining various logic functions discussed in Chapter 19. All these logic circuits are available in *IC* modules and are divided into many 'families'. Each family is classified by abbreviations which indicate the type of logic circuit used. For example, *RTL* means resistor-transistor logic. We will discuss the following seven transistor logic families although the first two are, at present, of historic interest only.

1. **Resistance-transistor logic (*RTL*) :** it was the first family group of logic circuits to be developed and packaged in *IC* form in *early* 1960*s*;
2. **Diode-transistor logic (*DLT*) :** It followed *RTL* in *late* 1960*s*;
3. **Transistor-transistor logic (*TTL*) *OR* (T^2L) :** was introduced in the early 1970 *s*;
4. **Schottky *TTL* :** was introduced to improve the speed of *TTL*;
5. **Emitter-coupled logic (*ECL*) :** It is fastest logic line currently available;
6. **Integrated-injection logic (I^2L) :** It is one of the latest of the bipolar types of logic;
7. **Complementary metal-oxide semiconductor (*CMOS*) :** It has the lowest power dissipation of the currently-available logic circuits.

The various logic families discussed above posses different characteristics as detailed below.

71.23. Saturated and Non-saturated Logic Circuits

Those logic circuits in which transistors are driven into saturation are called **saturated** logic circuits or simply **saturated logic.** Those circuits which avoid saturation of their transistors are designed **non-saturated logic.**

The disadvantage of saturated logic is the delay that occurs when the transistors is brought out of saturation. When a transistor is saturated, its base is flooded with carriers. Even when base voltage is switched off, the base remains flooded for some time till all carriers leave it. The time required by the carriers to leave the base is called **saturation delay time** (t_s). Obviously, saturated logic circuits have low switching speeds whereas non-saturated type are much faster. *TTL* is the example of a saturated logic whereas *ECL* represents a non-saturated logic.

71.24. Basic Operating Characteristics and Parameters of Logic Families

When we work with digital ICs from different logic families, we should be familiar with, not only their logical operation but also with the basic operational properties. Following are the important basic operational properties important from the subject point of view.

1. *DC* supply voltage.
2. *TTL* and *CMOS* logic levels
3. Noise immunity
4. Noise margin.
5. Power dissipation.
6. Propagation delay.
7. Speed-power product.
8. Loading and fan-out.

Now we will describe all the above operational characteristics one by one in the following pages.

71.25. DC Supply Voltage

The standard value of the dc supply voltage for *TTL* (*i.e.*, transistor-transistor logic) and CMOS (*i.e.*, complementary metal-oxide semiconductor) device is + 5*V*. For simplicity, the dc supply voltage is usually omitted from the logic circuits. But in practice, it is connected to the V_{CC} or V_{DD} pin of an *IC*

package and the ground is connected to the *GND* pin of an *IC* package. Both the voltage and ground are distributed internally to all the logic gates with the package as shown in Fig. 71.47 (*a*). The connections for the single logic gate are as shown in Fig. 71.47 (*b*).

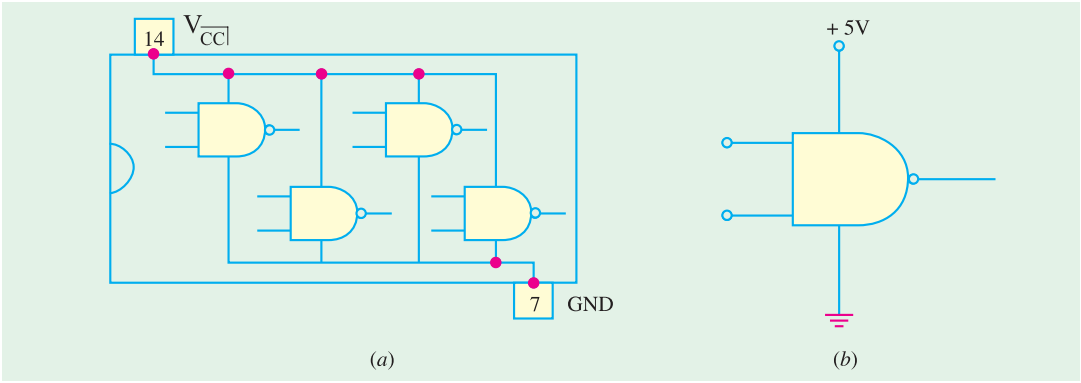


Fig. 71.47

71.26. TTL and CMOS Logic Levels

For TTL circuits, the range of input voltages, that can represent a valid *LOW* (logic 0) and a valid *HIGH* (logic 1) is as shown in Fig. 71.48 (*a*). As seen from this diagram, the range of input voltages that can represent a valid LOW (logic 0) is from 0 to 0.8 V. The LOW input voltage is indicated by the symbol V_{IL}. The lower limit for V_{IL} is represented by $V_{IL(min)}$ and the higher limit for V_{IL}, by $V_{IL(max)}$. The range of input voltages that can represent a valid HIGH (logic 1) is from 2 V to V_{CC} (usually 5 V). The HIGH input voltage is indicated by symbol V_{IH}. The lower limit for V_{IH} is represented by $V_{IH(min)}$ and the higher limit for V_{IH} by $V_{IH(max)}$. Note that the range of values between 0.8 V and 2 V is called ***indeterminate.*** This means that when a input voltage is in this range, it can be interpreted as a HIGH or LOW by the logic circuit. Therefore TTL logic gates cannot be operated reliably when input voltages are in this range.

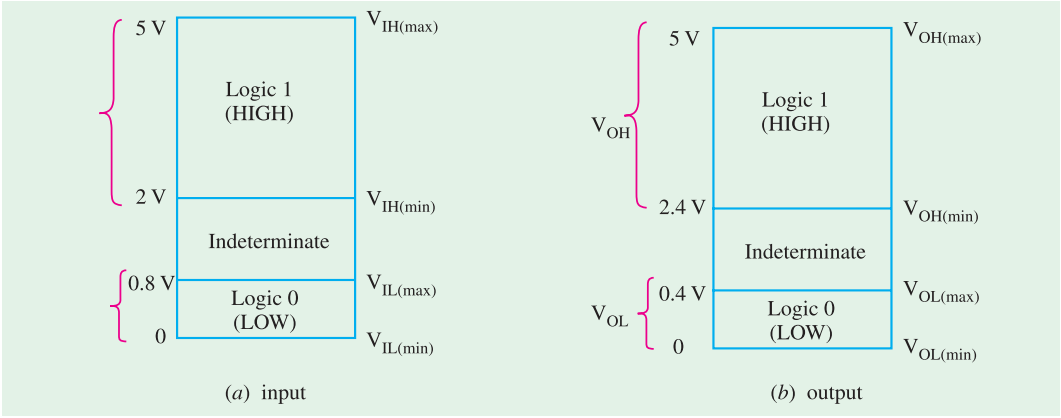


Fig. 71.48

Fig. 71.48 (*b*) shows the range of *TTL* output voltages that can represent valid HIGH (logic 1) and valid *LOW* (logic 0). As seen the range for logic 1 output ifs from 2.4 V to 5 V and logic 0 output is from 0 to 0.4 V. Note again that the range of values between 0.4 V and 2.4 V is indeterminate. Also note that the output voltage for logic 1 is indicated by the symbol V_{OH}. The lower limit for V_{OH} is represented by $V_{OH(min)}$ and the higher limit for V_{OH} by $V_{OH(max)}$. Similarly the output voltage for logic 0 is indicated by the symbol V_{OL}. The lower limit for V_{OL} is represented by $V_{OL(min)}$ and the higher limit for V_{OL} by $V_{OL(max)}$.

It may be noted from Fig. 71.48 (*a*) and (*b*) that the minimum HIGH output voltage, $V_{OH\,(min)}$ is greater than the minimum HIGH in put voltage $V_{IH\,(min)}$. On the other hand, the maximum LOW output voltage, $V_{OL\,(min)}$ is less than the maximum LOW input voltage, $V_{IL(max)}$.

The input and output voltages for a device from HCMOS (*i.e.* High-speed CMOS) logic family for V_{DD} = 5 V as shown in Fig. 71.49 (*a*) and (*b*) respectively.

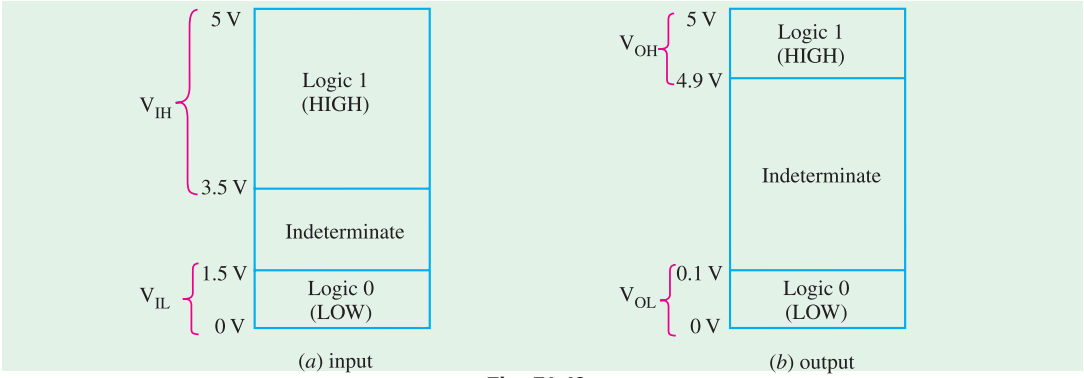


Fig. 71.49

Notice that the input and output voltage values in the HCMOS device are different from that of a TTL device. In a HCMOS device, $V_{IL(max)}$ is 1.5 V (its value is 0.8 V in TTL), $V_{IH\,(min)}$ is 3.5 V (its value is 2 V in TTL), $V_{OL\,(max)}$ is 0.1 V (its value is 0.4 V is TTL), $V_{OH\,(min)}$ is 4.9 V (its value is 2.4 V in TTL).

71.27. Noise Immunity

The noise immunity of a logic circuit refers to the circuit's ability to tolerate noise without causing a false change in its output voltage. The noise voltage is produced by stray electric and magnetic fields on the connecting wires between logic circuits. Sometimes, too much noise voltage cause the voltage at the input of the logic circuit to drop below $V_{IH\,(min)}$ or rise above $V_{IL\,(max)}$. This could produce unpredictable operation in a logic circuit.

71.28. Noise Margin

A quantitative measure of a circuit's noise immunity is called noise margin. It is expressed in volts. There are two values of noise margin specified for a given logic circuit as described below.

1. The high level noise margin (V_{NH})
2. The low level noise margin (V_{NL})

These parameters are shown in Fig. 71.50 and are given by the equations,

$$V_{NH} = V_{OH\,(min)} - V_{IH\,(min)}$$

$$V_{NL} = V_{IL\,(max)} - V_{OL\,(max)}$$

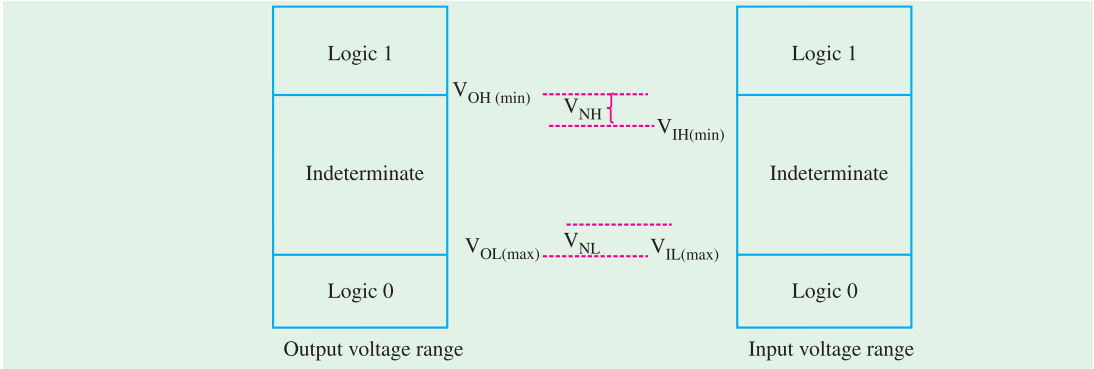


Fig. 71.50

Example 71.29. *Table shows the input/output voltage specifications for the standard TT family.*

Parameter	*Min (V)*	*Typical (V)*	*Max (V)*
V_{OH}	2.4	3.4	—
V_{OL}	—	0.2	0.4
V_{IH}	2.0	—	—
V_{IL}	—	—	0.8

Using these values, find ***(a)*** *the maximum value of noise spike that can be tolerated when a HIGH output is driving an input,* ***(b)*** *the maximum value of noise spike when a LOW output is driving an input.*

Solution. The maximum value of the noise spike, when driven by a HIGH output,

$$V_{NH} = V_{OH\,(\min)} - V_{IH\,(\min)} = 2.4 - 2.0 = 0.4\ \text{V}$$

and the maximum value of the noise spike, when driven by a LOW output,

$$V_{NL} = V_{IL\,(\max)} - V_{OL\,(\max)} = 0.8 - 0.4 = 0.4\ \text{V}$$

It is observed that a TTL gate is immunte to 0.4 V of poise for both the HIGH and LOW input states.

71.29. Power Dissipation

As a matter of fact, all logic gates draw current from the dc supply voltage for its normal operation. When the logic gate is in the HIGH output state, it draws an amount of current, I_{CCH}, as shown in Fig. 71.51 (*a*) and when in LOW output state, it draws an amount of current I_{CCL} as shown in Fig. 71.51 (*b*).

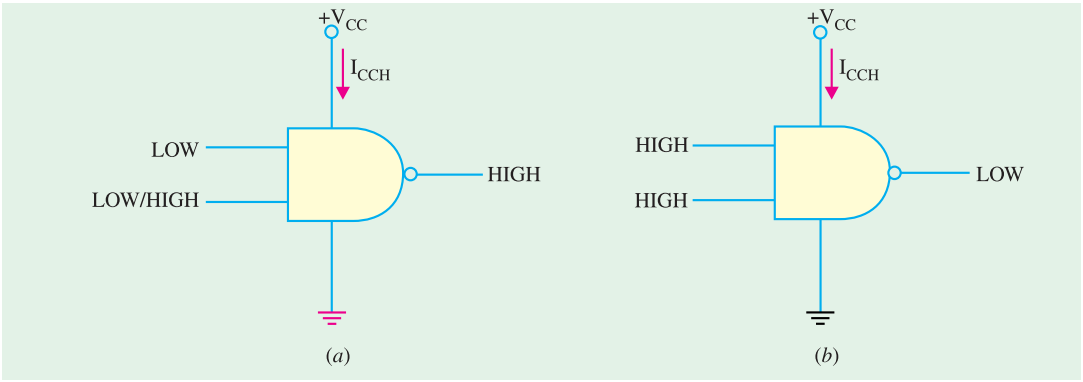


Fig. 71.51

The power dissipation of a logic gate is given by the product of the dc supply votage (V_{CC}) and the amount of current drawn from the suppy (*i.e.*, I_{CCH} or I_{CCL}). Thus power dissipation is given by :

$$P_D = V_{CC} \cdot I_{CCH} \text{ or } V_{CC} \cdot I_{CCL}$$

For example, if I_{CCH} is 2.5 *m A* when V_{CC} *is* 5 V, the power dissipation,

$$P_D = V_{CC} \cdot I_{CCH} = 5\ \text{V} \times 2.5\ \text{mA} = 12.5\ \text{mW}.$$

Usually, the logic gate operates with the inputs, which keep on changing with time (*i.e.*, the input is pulsed). Accordingly the output of a logic gate also switches back and forth between HIGH and LOW. Because of this, the amount of current drawn from the dc supply also varies between I_{CCH} and I_{CCL}. In such a situation, we calculate the average power dissipation. The average power dissipation depends upon the duty cycle and is usually specified for a duty cycle of 50%, the output is HIGH half the time and LOW the other half. Therefore average supply current is :

$$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2}$$

and the average power dissipation is,

$$P_D = V_{CC} . I_{CC}$$

Example 71.30. *A TTL logic gate draws 2 mA when its output is HIGH and 3.5 mA when its output is LOW. Calculate the average power dissipation if the supply voltage is 5 V and the logic gate is operated on 50% duty cycle.*

Solution. The average supply current,

$$I_{CC} = \frac{I_{CCH} + I_{CCL}}{2} = \frac{2\text{mA} + 3.5\text{mA}}{2} = 2.75\text{ mA}$$

∴ The average power dissipation,

$$P_D = V_{CC} . I_{CC} = 5\ r \times 2.75\text{ mA} = \mathbf{13.7\ mW}$$

71.30. Power Dissipation versus Frequency

Fig. 71.52 shows a graph of power dissipation versus frequency for a *TTL* and a *CMOS* logic gate. As seen from this graph, the power dissipation in a *TTL* circuit is essentially constant over its range of operating frequencies. However, the power dissipation in a *CMOS* circuit is frequency dependent. That is, it is extremely low under zero frequency (or dc) conditions. But the power dissipation increases as the frequency increases. For example, the power dissipation of a typical TTL logic gate is a constant 2 mW. On the other hand, power dissipation of typical *CMOS* logic gate is 0.0025 mW under static (or dc) conditions and 0.17 mW at 100 kHz.

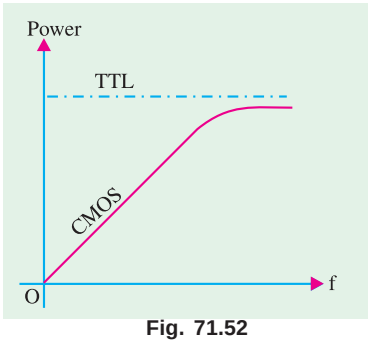


Fig. 71.52

71.31. Propagation Delay

When a signal passes (*i.e.*, propagates) through a logic circuit, it always experiences a finite time delay as shown in Fig. 71.53. It shows that the change in output level occurs after a short time, (called the propagation delay time), later than the change in input level that caused it. There are two propagation delay times specified for logic gates.

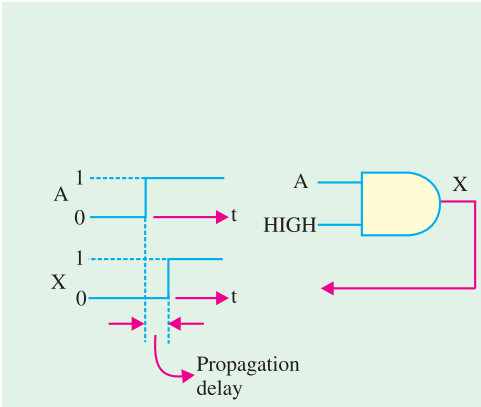


Fig. 71.53

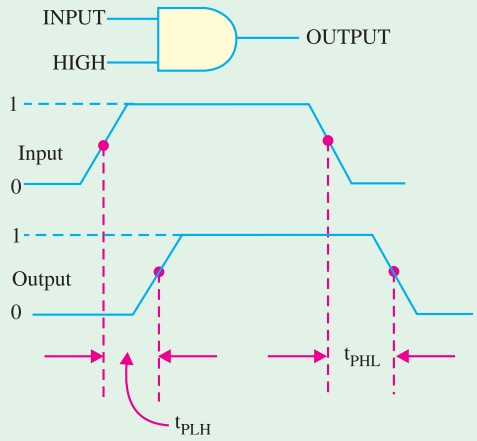


Fig. 71.54

1. t_{PLH} – It is the time interval between a designated point on the input pulse and the corresponding point on the output pulse when the output is changing from LOW to HIGH (or 0 to 1) as shown in Fig. 71.54.

2. t_{PHL} – It is the time interval between a designated point on the input pulse and the corresponding pont on the output pulse when the output is changing from HIGH to LOW (or 1 to 0) as shown in Fig. 71.54.

It may be noted that the propagation delay times are indicated in Fig. 71.54 with 50% points on the pulse edges used as references.

The propagation delay time of a logic gate limits its maximum operating frequency. The greater the propagation delay time of a logic gate, the lower is its maximum operating frequency. This means a high speed logic gate is a one that has a small propagation delay time. For example, a logic gate with a delay time of 2 ns is faster than a logic gate that has a delay time of 10 ns.

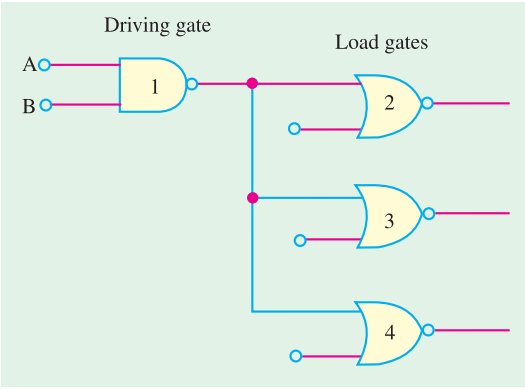


Fig. 71.55

71.32. Speed-Power Product

It provides a basis for comparison of logic circuits when the propagation delay and power dissipation are important considerations in the selection of the type of logic family to be used in certain application. The speed power product is exdpressed in picojule (pJ). It may be noted carefully that the lower the speed-product, the better is the value. Typically the *CMOS* family has much lower value of speed-power product as compared to the *TTL* logic family. For example, a typical *CMOS* logic family has a speed-power product of 1.5 pJ at 100 Hz while a typical *TTL* has speed-power product of 20 pJ.

71.33. Loading and Fan-out

When the output of any logic gate is connected to one or more inputs of other logic gates, a load on the driving gate is created. This is shown in Fig. 71.55. Here the output of a logic gate (labeled as 1) is connected to the inputs of 3 other logic gates (labeled as 2, 3, and 4). Note that the logic gate labeled 1 is called a ***driving gate*** while the logic gates labeled as 2, 3 and 4 are called ***load gates***.

In any logic family, there is a limit to the number of load gate inputs that a given logic gate can drive. This limit is called the ***fan-out*** of the logic gate. Now we will study the loading and fan-out in *TTL* and *CMOS* logic families in more detail.

Loading and fan-out in TTL family : A *TTL* gate that acts as a driving gate ***sources*** (*i.e.*, supplies) current to a load gate input in the logic HIGH state and *sinks* (*i.e.*, receives) current from the load gate in the logic LOW state. Fig. 71.56 **(*a*)** illustrates the current sourcing and **(*b*)** shows current sinking in the logic gates.

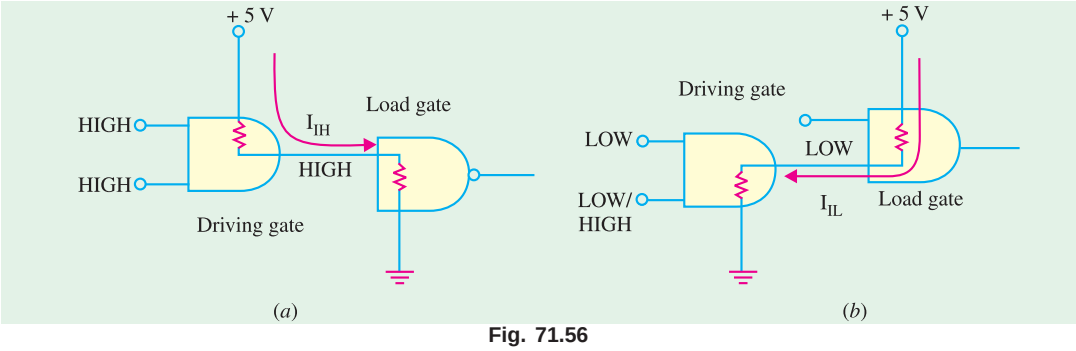


Fig. 71.56

Note that I_{IH} is the current supplied by the driving gate to the load gate when the input of the load gate is HIGH similarly. I_{IL} is the current received by the driving gate from the load gate when the input of the load gate is *LOW*.

As more and more number of load gates are connected to the driving gate, the loading on the driving gate increases. The total source current increases with each load gate input that is added as illustrated in Fig. 71.57. As the source current increase, the internal voltage drop of the driving gate increases V_{OH}. This causes the voltage drop V_{OH} to decreases. If a large number of load gate inputs are connected, drops below $V_{OH\,(min)}$. As a result of this, HIGH level noise margin is reduced, thus affecting the specified operating characteristics of the gate. Moreover, as the total source current increases, the power dissipation of the driving gate increases.

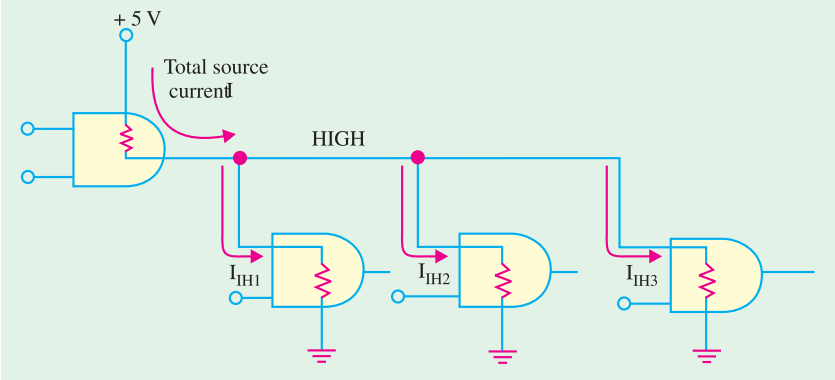


Fig. 71.57

The maximum number of load gate inputs that can be connected without affecting the specified operational characteristics of the driving gate is called ***fan-out.*** Its value is important for designing logic circuits. For example, the standard *TTL* has a fan-out of 10. One input of the same logic family as the driving gate is referred to as a ***unit load.*** This we can also say that a standard *TTL* has a fan-out of 10 unit loads.

The total sink current also increases with each load gate input that is added as shown in Fig. 71.58. As this current increases, the internal voltage drop of the driving gate increases, causing V_{OL} to increase. If a large number of loads are added, V_{OL} exceeds $V_{OL\,(max)}$ and the LOW-level margin is reduced.

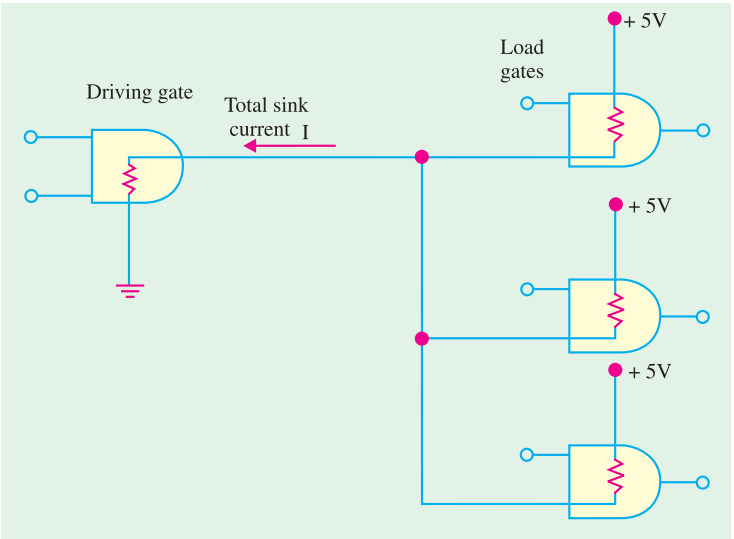


Fig. 71.58

As a matter of fact, in *TTL*, the current sinking capability (in the *LOW* state) is the limiting factor in determining the fan-out.

Loading and fan-out in CMOS: Loading in *CMOS* logic family differs from that in *TTL*. It is because of the fact that the field-effect transistors used in *CMOS* logic family present a predominantly capacitive load to the driving gate as shown in Fig. 71.59. In this case, the limitations are the charging and discharging times associated with the output resistance of the driving gate and the input capacitance of the load gates.

When the output of the driving gate is HIGH, the input capacitance of the load gate is charging through the output resistance of the driving gate. When the output of the driving gate is LOW, the

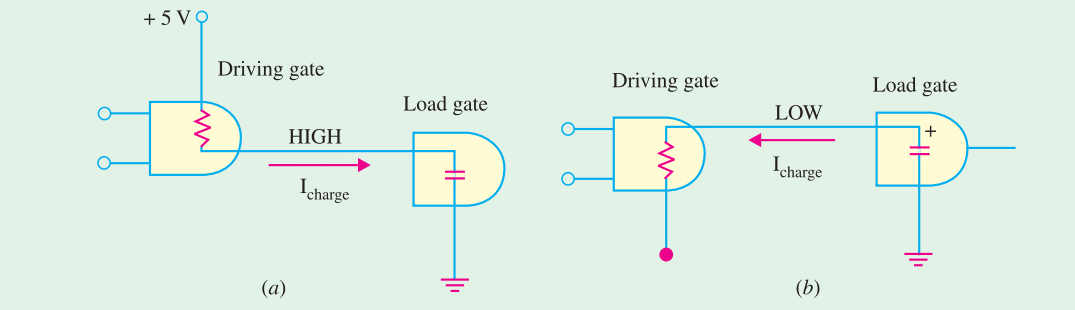


Fig. 71.59

capacitance is discharging. When more and more number of load gate inputs are added to the driving gate output, the total capacitance increases because the input capacitances effectively appear in parallel. This increase in capacitance increases the charging and discharging times. As a result, the maximum operating frequency of the logic gate is reduced. Thus fan-out in a *CMOS* logic family depends upon the operating frequency. The smaller the number of load gate inputs, the greater the maximum operating frequency.

71.34. RTL Circuit

It is a ***saturated* logic.** It uses only transistors and resistors as circuit elements and also resistances in the input to each base. This family is based on the *NOR* circuit shown in Fig. 71.60. All other members of the family are made up of *NOR* cells or variations on them.

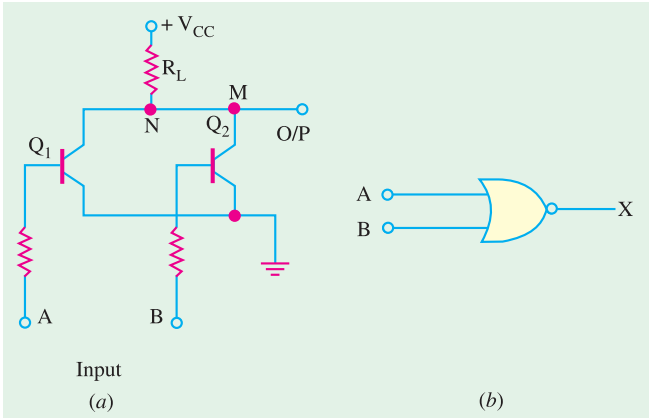


Fig. 71.60

Circuit Operation

We will assume ideal transistors. When both inputs *A* and *B* are 0 V (or logic 0) both transistors are turned *OFF*, hence point *M* goes to + V_{CC} so that output is logic 1.

If either or both input terminals are at + V_{CC} *i.e.* are high (or logic 1), one or both transistors would be fully turned *ON* (*i.e.* saturate) thereby reducing the voltage of point *N* to almost 0 V. Hence, output would be at logic 0.

It is seen that the output is at logic 1 only when ***both inputs are at logic 0*** — the *NOR* logic functions as shown in Fig. 71.60 (*b*).

The *RTL* family has the following characteristics :

1. relatively slow speed,

2. low fan-out of 6 and a fan-in of 4,

3. poor noise immunity,

4. expensive since resistors are required to be fabricated,

5. cannot operate at speed above 4 MHz.

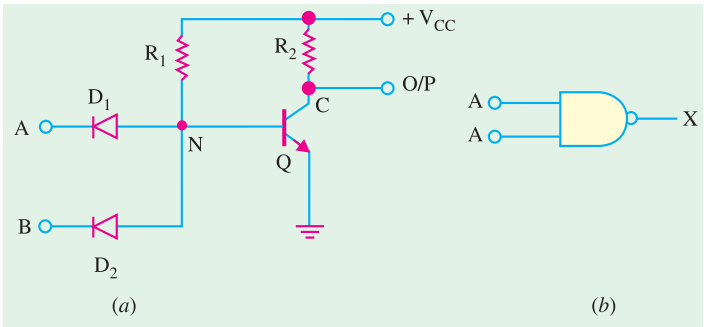


Fig. 71.61

71.35. DTL Circuit

It is a ***saturated*** logic because transistors between cut-off and saturation. It was the next family to be introduced after *RTL*. It consists of diodes, resistors and transistors. The basic gate of this family performs *NAND* function. As shown in Fig. 71.61 (*a*) the circuit basically consists of a diode *AND* gate followed by a transistor inverter which leads to a *NAND* gate.

Circuit Operation

1. When both D_1 and D_2 have positive voltage applied to them (logic 1), neither conducts and Q is turned *ON* by the current provided by V_{CC} through R_1. Since Q becomes saturated, point C is brought to 0V (logic 0). Hence output goes logic 0.
2. If either or both inputs are at 0V (logic 0), the associated diode will conduct driving point N to ground *i.e.* 0V. Since there is no base voltage for Q, it will be cut *OFF* thereby driving point C and hence output to V_{CC} *i.e.* logic 1.

It is seen that output is low (a logic 0) only when all inputs are high—the condition for a *NAND* gate.

The *DTL* family is characterised by

1. relatively lower speed,
2. Comparatively better noise immunity,
3. propagation delay of 30 ns,
4. a fan-out of 5.

71.36. TTL Circuit

It is a ***saturated* logic.** It is the most widely used circuit line since early 1970*s* because of its speed, good fan-out and easy interface with other digital circuitry. The unique feature of this circuit is that it used ***multiple-emitter transistor*** at input which replaces the input diodes of the *DTL*. Th number of emitters is equal to the number of inputs of the logic circuit (limited to 8). Since a multi-emitter transistor is small in area than the diodes it replaces, the yield from a wafer is increased. Moreover, smaller area resultes in ***lower capacitance*** to the substrate, thereby wide selection of circuit modules ranging from simple gates and flip-flops in *SSI* circuit series through various registers in computers in *MSI* circuit series to micro-processor bit-slice chips in the *LSI* series.

Basic Circuit

The basic circuit of the *TTL* familyis the *NAND* gate cell shown in Fig. 71.62. However, at present *NOR*, *OR* and *AND* gate configurations have also been added to the series.

Circuit Operation

1. If both inptus A and B are high (logic 1), E/B junction of Q_1 is *reverse-biased* so that it has no emitter current. Hence Q_1 is *OFF*. However, its C/B junction is ***forward-baised*** supplying base current to Q_2 from V_{CC} *via* R_1. As a result transistor Q_2 is turned fully *ON* (*i.e.* it becomes saturated) driving point N to $0V$. Hence, output is a logic 0.
2. When either or both inputs are at $0V$ (logic 0), the associated E/B junction becomes forward-biased. The value of R_1 is so selected as to ensure that Q is turned fully *ON*. The voltage at point M falls to $0V$ with the result the base current for Q_2 is reduced to zero. Hence, Q_2 is cut *OFF* driving point N and the output to logic 1.

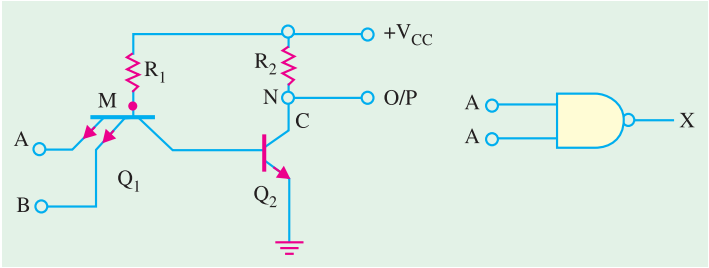


Fig. 71.62

Totem Pole Output

The basic circuited of Fig. 71.62 is never used in practice. Its modified version with an added output stage is in common use (Fig. 71.63). This extra output stage is often known as *totem-pole* stage because the three components Q_3, Q_4 and D are stacked one on top of the other in the manner of a totem-pole. The circuit action is as follows :

1. When Input is High

In this case, the two input terminals have ***positive*** voltage (logic 1). the E/B junction is reverse-biased because of which there is no emitter current. Hence, Q_1 is OFF. Since C/B junction of Q_1 is forward-biased, base current of Q_2 flows from V_{CC} through R_1. Hence, Q is turned ON. As a result, potential of point N falls so much that Q_2 is turned OFF. At the same time Q_4 is turned ON by the voltage drop across R_3. Now, when Q_4 is ON, its collector potential (*i.e.* potential of point C) is nearly that of its emitter. Hence, output is low *i.e.* at logic 0.

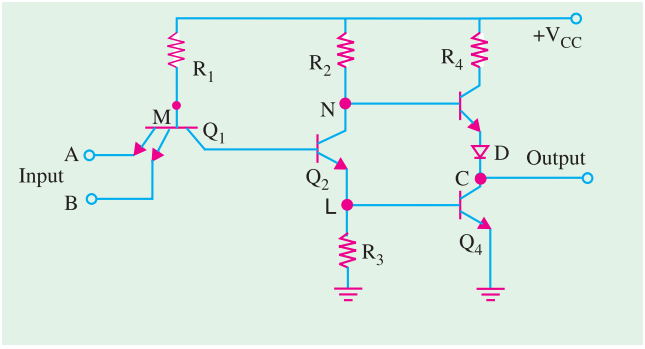


Fig. 71.63

In short, when inputs are at logic 1, Q_1 is OFF, Q_2 is ON, Q_3 is OFF and Q_4 is ON because of which output because logic 0.

2. When Input is Low

If any of the two inputs or both are low (logic 0), Q_1 turns ON and potential of its collector (point M) falls. Hence, Q_2 is turned OFF, grounding its emitter and the base of Q_4 so that Q_4 is also turned OFF.

Since N is at V_{CC}, it turns Q_3 ON. The potential of point C is V_{CC} minus drop in R_4, Q_3 and D. Since these drops do not amount to much, output is at logic 1.

It may be noted that when ***even-numbered transistors are ON, the odd-numbered ones are OFF and vice-versa.***

The function of diode D in Fig. 71.63 is to prevent both Q_3 and Q_4 from being turned ON simultaneously. If both were to be *On* at the same time, they would offer low immedance to teh suppy which will draw excessive current and produce large noise 'spikes' in the output. It may also be noted that the addition of a pair of totem pole transistor increases the operating speed and output current capability of this circuit. The standard TTL-family has

1. greater speed then DTL,

2. less noise immunity (0.4 V),

3. average propagation delay per gate of 9 ns,

4. average power dissipation of 10 mW,

5. a fan-out of 10 meaning one output can drive 10 other TTL inputs,

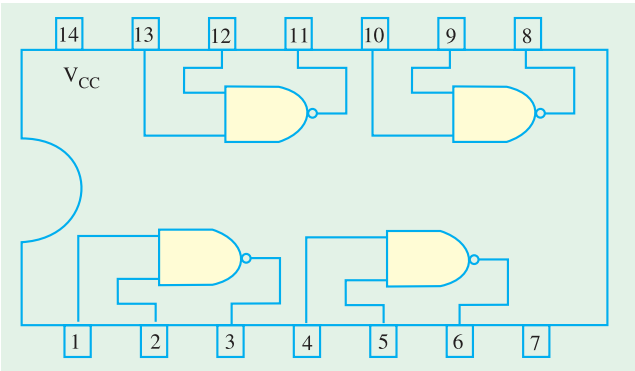


Fig. 71.64

Fig. 71.64 shows a pin-out of a BEL 7400 IC referred to as 'quad (quadrupole). 2-input $NAND$ gate chip'. As seen, there are four separate

2-input *NAND* gates of which any or all may be used at any point in time. It is manufactured by Bharat Electronics LTD. (BEL) Bangalore, India.

71.37. TTL Sub-families

TTL has several ***sub-families*** having different speed and lower dissipation charcteristics as detailed below :

1. **74L00** series—the letter *L* standing for low power consumption. It has an average power dissipation of 1 mW per gate but an average propagation delay of 33 ns.
2. **74H00** series—the letter *H* standing for higher speed. It has a propagation delay of 6 ns but averge power dissipation of 23 mW/gate.
3. **74S00**—the letter *S* representing Schottky. It has the highest speed because its average propagation delay is just 3 ns per gate. However, its average power dissipation is 23 mW/gate.
4. **74LS00**—It is called low-power Schottky *TTL*. It has an average propagation delay of 9.5 ns and an average power dissipationof 2 mW.
5. **74AS00** series—The letter *A* representing Advanced and *S* stnading for schottky. It is called advanced schottky *TTL* series. It is the fastest *TTL* series.
6. **74ALS00** series—It is called Advanced Low-power Schottky *TTL* series. The 74*ALS* series has the lower speed-power product and the lowest gate power dissipation of all the *TTL* series.
7. **74F00** series —The letter *F* standing for fast. This logic family uses a new *IC* fabrication technique to reduce interdevice capacitances to achieve reduced propagation delays. It has a propagation delay of 3 ns and a power consumption of 6 mW.

Table 71.3 summarizes the important characteristics of the each of the *TTL* sub-families.

Table 71.3

Characteristic	*TTL* sub-family					
	74	74S	74LS	74AS	74ALS	74F
Propagation delay (ns)	9	3	9.5	1.7	4	3
Power Dissipation (mW)	10	20	2	8	1.2	6
Speed-power product (pJ)	90	60	19	13.6	4.8	18
Fan-out	10	20	20	40	20	33

71.38. ECL Circuit

The *ECL* also called current-mode logic (*CML*), has the **highest speed** of any of the currently-available logic circuits. It is primarily due to the fact that transistors never operate fully saturated or cut-off. That is why *ECL* is known as ***non-saturated* logic.** The latest *ECL* series has propagation delay time varying from 0.1ns to 0.8 ns. However, power dissipation is increased since one transistor is always in the active region.

Another feature of *ECL* is that it provides two outputs which are always complement of each other (Fig. 71.65). Itis so because the circuit operation is based on a differential amplifier.

This family is particularly suited to monolithic fabrication techniques because logic levels are function of resistor ratios.

Circuit Operation

The basic circuit shown in Fig. 71.65 is combined *OR/NOR* circuit and is operated from a $V_{EE} = -5.2\ V$ supply. A built-in constant-current source provides current to the emitters. Strictly speaking,

logic 1 is represented by – 0.9 *V* (less negative) and logic 0 by – 1.75 *V* (more negative). Please note that it is a *positive* logic. In negative logic, the functions would be *AND/NAND*. A reference voltage of – 1.29 *V* is applied to the base of Q_3 from a built-in temperature-compensated reference voltage source (*RVS*).

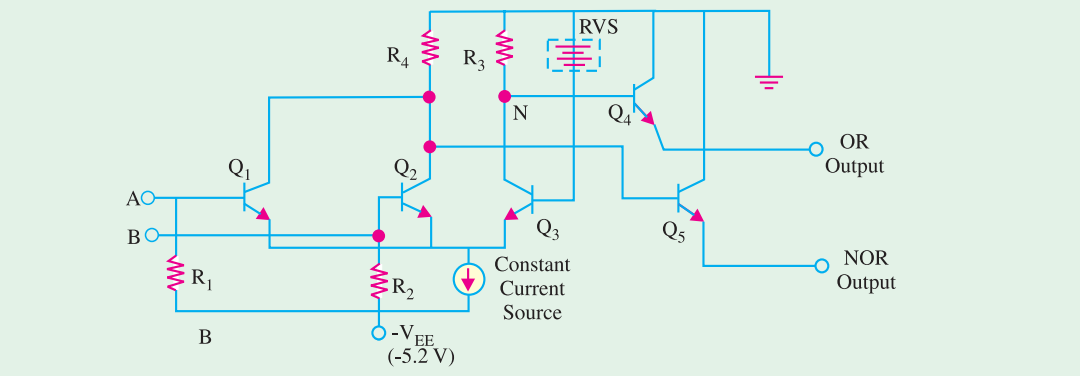


Fig. 71.65

1. When both inptus are logical 0 *i.e.* –1.75 V

In this case, base potential of Q_3 is less negative (or more positive) than the base potential of either Q_1 or Q_2. Hence, Q_3 conducts whilst Q_1 and Q_2 do not. Only enough base current is drawn by Q_3 from RVS so as to remain out of saturation. The collector current of Q_3 develops a voltage of – 1.0 V across R_3 which makes Q_4 to conduct. Transistor Q_4 gives an output voltage at the emitter of about –1.0 – 0.7 = –1.7 V which represents logic 0. Since collector potentials of Q_1 and Q_2 are nearly zero (because they are cut off), the output voltage at the emitter of Q_5 is 0 – 0.7 = –0.7 V which is a logic 1. Obviously, the two outputs are complements of each other.

2. When either input A or B is at logical 1 *i.e.* –0.9 V

In that case, the associated transistor (either Q_1 or Q_2) is turned *ON* while Q_3 is turned *OFF*. The collector potentials of Q_1/Q_2 and are opposite of the previous case. Hence, now Q_5 output is at logical 1 and Q_4 output is at logical 0.

Typical characteristics of an *ECL* family are :

1. propagation delay time per gate of 0.3 ns (meaning extremely fast speed),
2. power dissipation of 25 mW,
3. fan-out of 25 to 50,
4. noise margin from about 0.2 to 0.25 V.

It will be interesting to know that *ECL* family of *ICs* does not include a wide range of general purpose logic devices as do the *TTL* and *CMOS* logic families. *ECL* does include complex, special purpose *ICs* used in applications such as high-speed data transmission, high-speed memories and high-speed arithmetic units. The relatively low noise margins and high power drain of *ECL* are disadvantages compared with *TTL* and *CMOS*. Another drawback is its negative power supply voltage and logic levels which are not compatible with those of other logic families. This makes it difficult to use *ECL* devices in conjunction with *TTL* and/or *CMOS ICs*. Special level-shifting (also called interface) circuits must be connected between *ECL* devices and the *TTL* (or *CMOS*) devices on both input and output.

71.39. I^2L Circuit

It is the latest entry into the bipolar ***saurated*** logic field. It uses no biasing and loading resistors at all! Resistors require lot of power and space on an *IC* chip. Hence, their elimination results in ***higher density circuits operating at much reduce power.*** Because of its **high speed** and **less power dissipa-**

tion, it is used in large computers. Such circuits are also used where high packing density is of prime consideration as in digital wrist watches. I^2L chips are capable of microwatt power dissipation yet can provide high currents when necessary to drive *LED* displays. Another feature of I^2L (integrated injection logic) is that it is easy to fabricate.

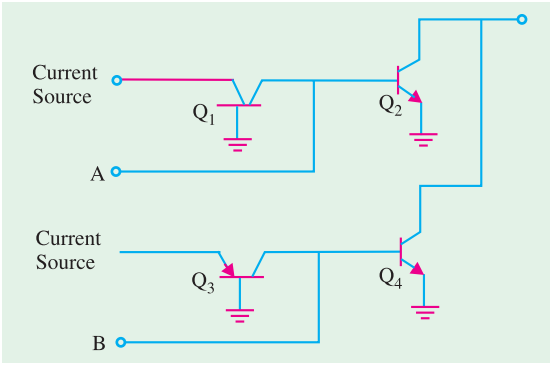


Fig. 71.66

Circuit Operation

I^2L *NOR* logic circuit is shown in Fig. 71.66. Here, transistors Q_1 and Q_2 act as current sources to the bases of Q_3 and Q_4 respectively.

If *A* input goes low, the current to the base of Q_2 will be shorted to ground which will result in Q_2 being turned *OFF*. The input *B* controls Q_4 in a similar way.

If *A* is high, base current flowing to Q_2 will turn in *ON*, making output *C* low. Same would be the case when *B* is high.

It is obvious that output would be high only when ***both inputs*** *A* and *B* are low.

The output would be low when either *A* or *B* or both are high *i.e. NOR* logic function.

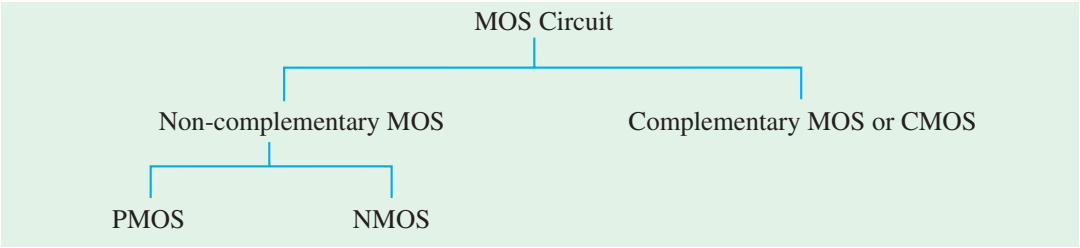
71.40. MOS Family

It does not use bipolar transistors but just Enhancement-only *MOSFETs* (Art. 71.1). There are two kinds of digital *MOS* circuits.

(a) one which uses *MOSFETs* of one polarity either all of *N-type* called *NMOS* or all of *P*-type called *PMOS* but not both on the same chip,

(b) the other which employs both *N*-type and *P*-type *MOSFETSs* on the same chip. It is called comlementary *MOS* (*CMOS*).

The *MOS* family may be subdivided as under :



Since a *FET* requires small area, it is possible to fabricate a large number of *MOS* ciruits on a single small chip. Gating arrays with thousands of gates and flip-flops are manufactured in standard containers and are often used in *IC* memories and microprocessors.

Followings are same of the advantages of *MOS ICs* over the bipolar *ICs* (i.e. *TTL*, *ECL* etc.)

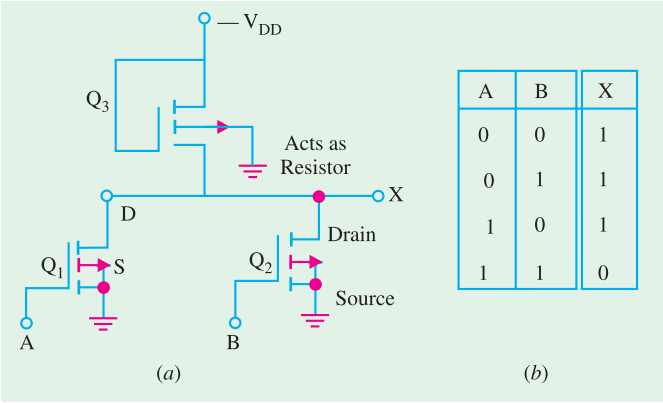
1. The *MOS IC* is relatively simple and inexpensive to fabricate.
2. The *MOS* device size is small and it consumers less power. Because of the small size, the *MOS ICs* can accommodate a much larger number of circuit elements on a single chip than bipolar *IC* in the area of large scale integration. This makes them especially well-suited for complex *ICs* such as microprocessor, memory chips etc.
3. *MOS* digital *ICs* normally do not use the *IC* resistor elements that take up so much of the chip area or bipolar *ICs*.

The continuous improvement in *MOS IC* technology has led to the device that are faster than *TTL* devices. Consequently *MOS* devices (especially *CMOS*) have become dominant in the *SSI* (small-scale integration) and *MSI* (medium-scale integration) market.

The major disadvantage of *MOS* devices is that they are susceptible to static-electricity damage. Although we can minimize it by adopting proper handling procedures, yet *TTL* is still more durable for laboratory experimentation. As a result, we are likely to see *TTL* devices used in education as long as they are available.

71.41. PMOS Circuit

A *PMOS NAND* gate is shown in Fig. 71.67 (*a*). As seen, there are no resistors in the circuit. The gate consists of two *E* only *MOSEFETs* Q_1 and Q_2 as logic elements and the third one Q_3 as load resistors. When $-V_{DD}$ (say −12 V) is applied, *MOSFETs* will be turned *ON* and when 0 V is applied they would be turned *OFF*. Hence, with positive logic, 0 V would be 1 and – 12 V would be 0 since 1 is assigned to the most positive voltage.



A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Fig. 71.67

Circuit Operation

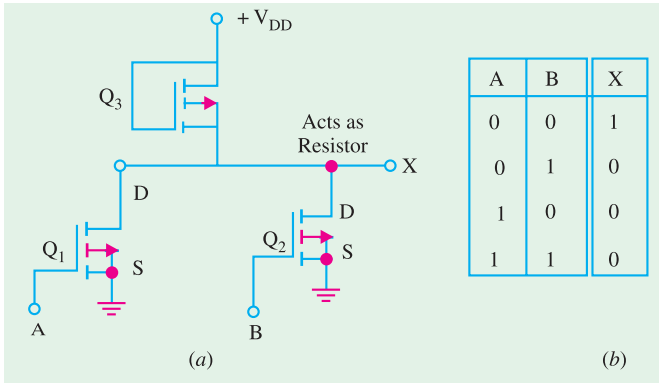
1. If any of the two inputs *A* or *B* is at logic 0 (*i.e.* – 12 V), the concerned *MOSFET* would turn *ON* thereby offering a low resistance from drain to source thus causing the output to be nearly 0*V i.e.* a logic 1.
2. The output can be at – 12 V *i.e.* logic 0 only when both inputs *A* and *B* are at 0 V *i.e.* logic 1.

This is the *NAND* function as shown by the truth table of Fig. 71.67 (*b*).

Incidentally, the positive logic *NAND* gate of Fig. 71.67 (*a*) would be the ***negative logic NOR*** gate since the two are identical.

71.42. NMOS Circuit

In Fig. 71.68 (*a*) is shown a two-input *NOR* gate circuit consisting of two *MOSFETs* Q_1 and Q_2 acting as logic elements and Q_3 as a load resistor. For positive logic, 0*V* will be logic 0 and positive voltge ($+V_{DD}$) will be logic 1.



A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Fig. 71.68

Circuit Operation

1. If any of the inputs *A* or *B* is at logic 1, the corresponding *MOSFET* will conduct causing the output to go low *i.e.* logic 0.
2. If both inpouts *A* and *B* are at logic 0, then both *MOSFETs* will be *OFF* driving the output to logic 1.

This is *NOR* function as shown by the trugh table of Fig. 71.68 (*b*).

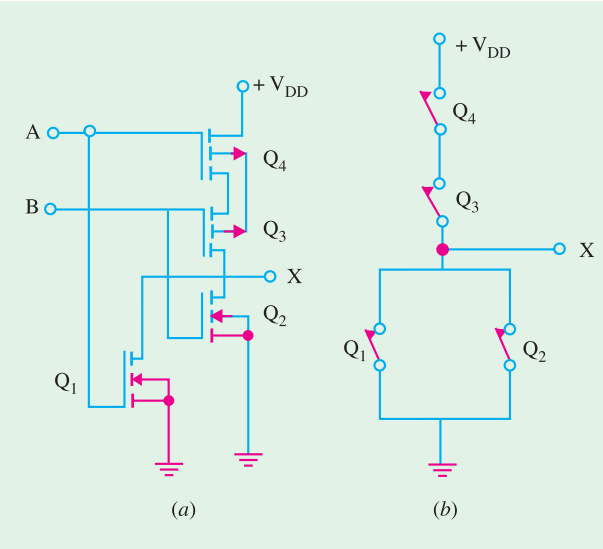


Fig. 71.69

71.43. CMOS Circuit

CMOS logic circuits use both *PMOS* and *NMOS* devices in the same circuit. It gives the advnatage of ***drastic decrease in power dissipation*** (12 nW per gate) and ***increase in speed of operation*****.** In fact, it has the **lowest power dissipation** amongest different logic families. It has very high packing density *i.e.* larger number of circuits can be placed on a single chip. As a result, it is extensively used in *VLSI* circuits such as on-chip computers and memory systems. The recent silicon-on-saphire *MOS* (*SOSMOS*) is 2 to 4 times faster than the standard *CMOS*. Hence, they are being widely used for everything from electronic watches and calculators to micro processors.

Fig. 71.69 (*a*) shows a *CMOS NOR* circuit which has two *N*-channel *MOSFETs* Q_1 and Q_2 and tow *P*-channel *MOSFETs* Q_3 and Q_4. The two inputs *A* and *B* switch between + V_{DD} (logic 1) and ground (logic 0).

Circuit Operation

1. Let $A = B$ = logic *i.e.* have positive voltage. In that case, Q_1 and Q_2 are *ON* (closed switches) but Q_3 and Q_4 are *OFF* and act as open switches as shown in Fig. 71.79 (*b*). Henc, output *C* is logic 0.
2. If either *A* or *b* is at logic 1, then the associated *N*-channel *MOSFET* (Q_1 or Q_2) is turned *ON* but the associated *P*-channel *MOSFET* (Q_3 or Q_4) is turned *OFF*. Since either Q_3 or Q_4 would be *OFF* [Fig. 71.69 (*b*)], output *C* would be at logic 0.
3. When both *A* and *B* are at logic 0, Q_3 and Q_4 would be *ON* but Q_1 and Q_2 would be *OFF*—just the opposite of that in Fig. 71.69 (*b*). Hence output *C* would be at logic 1 (remember, voltage across an open equals the supply voltage—Art. 1.21.

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

Fig. 71.70

The above logic represents a *NOR* function as shown in the truth table of Fig. 71.70. It would be observed from Fig. 71.69 that in each combination of *A* and *B*, there is at least ***one open switch*** between + V_{DD} and ground. Hence, the gate draws ***only leakage current from supply for any static state.*** However, when the gate switches from one level to another, some power is consumed becasue two *MOSFETs* are ***partly ON*** at the same time. Because of this reason, power dissipated by *CMOS* circuit *is a* ***function of input signal frequency.*** Higher the frequency, greater the power dissipation.

71.44. CMOS Sub-families and their characteristics

The *CMOS* family of *ICs* competes directly with *TTL* in the small and medium-scale integration. As *CMOS* technology has produced better and better performance characteristics, it has gradually taken over the field that has been dominated by *TTL* for so long. As a matter of fact, *TTL* devices will be around for a long time, but more and more new equipment is using *CMOS* logic circuits. These days, the *CMOS ICs* provide all the logic-functions that are available in *TTL* Some special-purpose functions provided in *CMOS ICs* are not provided even by *TTL*. Before we look at the various *CMOS*

subfamilies, let us define few terms that are used when *ICs* from different families or series are to be used together or as replacements for one another.

(*a*) Pin-to-pin Compatible : Two *ICs* are said to be pin-to-pin compatible when three pin configurations are the same. For example, pin 14 on both *ICs* is V_{CC} supply. Pin 7 on both is ground etc.

(*b*) Functionally Equivalent : Two *ICs* are said to be functionally equivalent when the logic function they perform are exactly the same. For example, both contain four two-input *AND* gates, or two-input *NAND* gates etc.

(*c*) Electrically compatible : Two *ICs* are said to be electricaly compatible when they can be connected directly to each other without taking any special measures to ensure proper operation.

Let us now study the different *CMOS* subfamilies.

1. **5000/14000 series :** The *CMOS* 4000 series or the improved 4000 *B* is the oldest series and was first introduced by *RCA*. This series is functionally equivalent to 14000 series from Motorola. The *CMOS* devices in 4000/14000 series have a very low power dissipation and can operate over a wide range of supply voltage (*i.e.* from 3 to 15 V). These devices are very slow as compared to *TTL* and other *CMOS* subfamilies. The 40H00 series was designed to be faster than 4000 series. It did overcome some of the speed limitations, but it is still much slower than *LSTTL* series. The 4000 series have very low output current capability. The devices in 4000 series are not pin-compatible or electrically compatible with any *TTL* series.
2. **74C00 series :** This is pin-compatible as well as functionally equivalent to *TTL* devices having the same device number. For example, 74C04 is a HEX inverter that has the same pin configuration as the *TTL* 7404 HEX inverter *IC*. The performance characteristics of 74C00 series are the same as those of 4000 series.
3. **74HC/HCT00 series :** The letter *H/HC* is standing for high-speed *CMOS* series. This series has a speed which is 10 times faster than of 74*LS*00 series devices. It has also a higher current capability than that of 74*C*00 series. The 74*HC/HCT00* series *ICs* are pin-compatible with and functionally equivalent to *TTL ICs* with the same device number. This series has become the most widely used series.
4. **74AC/ACT 00 series :** This series is referred as advanced *CMOS* logic. It is functionally equivalent to the *TTL* series but not pin-compatible.
5. **74AHC00 series :** This series is referred to as advanced high-speed *CMOS* logic. It is faster, and haslower-power dissipation. The devices in this series are 3 times faster and can be used as direct replacements for *HC* series devices.
6. **74-BiCMOS series :** These days, the *IC* manufacturers have developed a new logic series-called *BiCMOS* logic. This series combines the best features of bipolar and *CMOS* logic *i.e.* low power characteristics of *CMOS* and high speed characteristics of bipolar circuits. *BiCMOS ICs* are available only in those functions that are used in microprocessor interfacing and memory applications such as latches, buffers, drivers and transceivers.
7. **74-Low Voltage series :** A new series of logic using a nominal supply voltage of 3.3 V has been developed to meet the extremely low power design requirements of battery powered and hand held devices. These *ICs* are being designed into the circuits of notebook computers, mobile radios, hand-held video games, telecommunication equipment, personal digital assistant (*PDA*) and high performance work station computers.

   Table 71.4 shows some of the common Low-voltage families identified by the suffixes as indicated :

Table 71.4

Suffixes	*Low-voltage series*
LV	Low-voltage *HCMOS*
LVC	Low-voltage *CMOS*
LVT	Low-voltage technology
ALVC	Advanced Low-voltage *CMOS*
HLL	High speed Low-power Low-voltage

The power consumption of *CMOS* logic *ICs* decreases approximately with the square of the supply voltage. On the other hand, the propagation delay increases slightly at this reduced voltage. However the speed is restored and even increased by using finer geometry and sub-micron (<1μm) *CMOS* technology that is tailored for low-power and low-voltage applications.

8. **74 AHC/AHCT series :** This is an enchanced version of 74 *HC/HCT*00 series. It provides superior speed and low power consumption. The 74*AHC* series devices have half the static power consumption, one-third the propagation delay, high output drive current, and can operate at V_{CC} of 3.3 to 5 V.

Table 71.5 shows a comparison of the important characteristics of the *CMOS* logic families. It is evident from this table that the devices from 74*AHC*/*AHCT* series has the lowest propagation delay and the smallest value of power dissipation.

Table 71.5

S. No.	*Characteristic*	*CMOS logic family*			
		4000 B	*74 HC/HCT*	*74 BiCMOS*	*74 AHC/AHCT*
1.	Propagation delay (ns)	50	8	2.9	3.7
2.	Power dissipation static (mW)	0.001	0.0025	0.0003 – 7.5	0.000009
3.	Speed-power product (pw-s)	105	15	0.00087 to 22	—

Tutorial Problems No. 71.1

1. Prove the following identities :

   (*i*) $AB + A\overline{B} = A$ (*ii*) $A + A\overline{B} = A + B$ (*iii*) $(A + B)(A + \overline{B}) = A$

2. Simplify the following Boolean expressions :

   (*i*) $A\,\overline{A}\,C$ (*ii*) $ABCD + ABD$ (*iii*) $ABCD + A\,\overline{B}\,C\,D$

   [(*i*) 0, (*ii*) *ABD*, (*iii*) *ACD*]

3. Simplify the following Boolean functions :

   (*i*) $A + \overline{A}\,B + AB$ (*ii*) $\overline{A}\,B + \overline{A}\,B + AB\,\overline{A}\,B$ (*iii*) $(AB + C)(AB + D)$

   (*iv*) $A + \overline{B}\,C\,(A + B\,\overline{C})$ (*v*) $A[B + C\,(\overline{AB + AC})]$ (*vi*) $(A + B).(A + C)$

   [(*i*) $A + B$ (*ii*) 1 (*iii*) $AB + CD$ (*iv*) A (*v*) AB (*vi*) $A + B.C$ (*vii*) $\overline{B}$]

4. Simplify the following Boolean expressions :

(*i*) $A\,\overline{B}\,C + A\,B\,\overline{C} + A\,B\,C$ (*ii*) $\overline{A}\,\overline{C} + B\,\overline{C} + \overline{A}\,B\,C + ABC$

(*iii*) $AD + A\,\overline{B}\,\overline{C} + \overline{B}\,C\,\overline{D} + \overline{A}\,C\,D + \overline{A}\,B\,C\,\overline{D}$

(*iv*) $A\,\overline{B}\,\overline{D} + A\,B\,\overline{C}\,\overline{D} + A\,B\,C\,\overline{D} + \overline{A}\,B\,C\,\overline{D}$

[(*i*) $A\,(B + C)$ (*ii*) $B + \overline{A}\,\overline{C}$ (*iii*) $AD + \overline{A}\,\overline{B} + \overline{A}\,\overline{C}$ (*iv*) $A\,\overline{B}\,\overline{D} + A\,\overline{D} + C\,\overline{D}$]

5. Use truth tables to verify the following identities :

(*i*) $\overline{A} + B = \overline{A}\,.\,B$ (*ii*) $A + BC = (A + B)\,(A + C)$ (*iii*) $A\,(\overline{A} + \overline{B}) = AB$

6. Write the logic equation for the switching circuit of Fig. 71.71. **[$A\,(B + C + D + E$]**

7. Write down the logical expression which describes the working of the circuit of Fig. 71.72.

[$AB\,(CD)$]

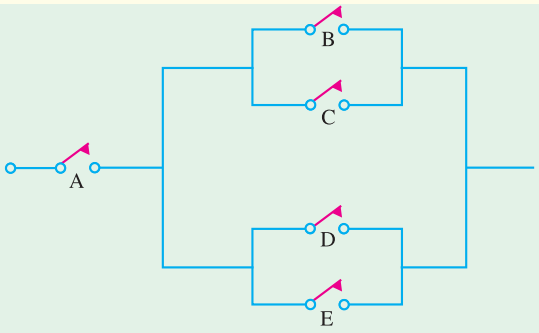


Fig. 71.71

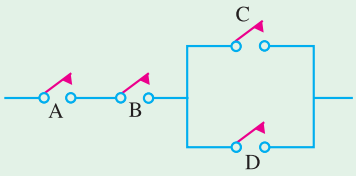


Fig. 71.72

8. Simplify the following Boolean expressions :

(*i*) $y = AB\overline{C} + A\overline{B}\,\overline{C} + A\overline{B}\,\overline{C} + \overline{A}\,\overline{C}\,\overline{B} + A\overline{B}$

(*ii*) $y = AB\overline{C}\,\overline{D} + AB\overline{C}D + ABC + ABD + CD$ **[(*i*) $A\overline{C} + \overline{B}$ (*ii*) $AB + CD$]**

9. Design a logic circuit whose output is HIGH only when majority of inputs *A*, *B* and *C* are LOW.

10. Determine the Boolean expression for the logic circuit shown in Fig. 71.73. Simplify the Boolean expression using Boolean Laws and De Morgan's theorem. Redraw the logic circuit using the simplified Boolean expression.

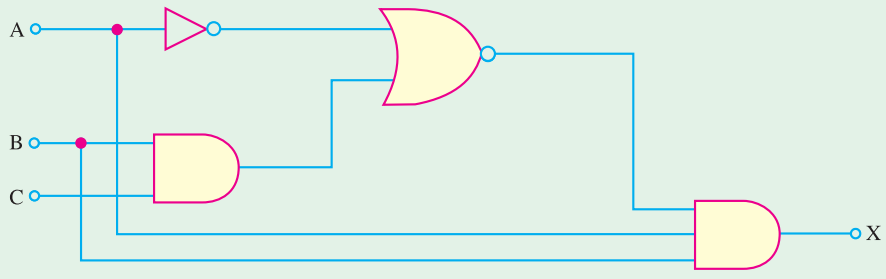


Fig. 71.73

11. Determine the output, *X* of a logic circuit shown in Fig. 71.74. Simplify the output expression using Boolean Laws and theorems. Redraw the logic circuit with the simplified expression.

($\overline{B}\overline{C}D + AC$)

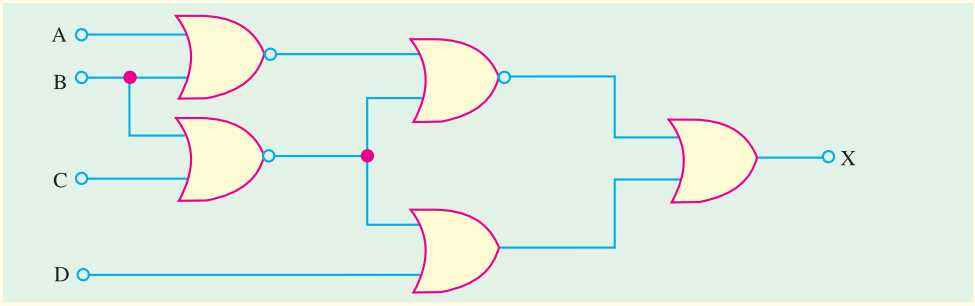


Fig. 71.74

12. Consider the logic circuit shown in Fig. 71.75. Determine the Boolean expression at the circuit output X, simplify it. Using the simplified expression, redraw the logic circuit.

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

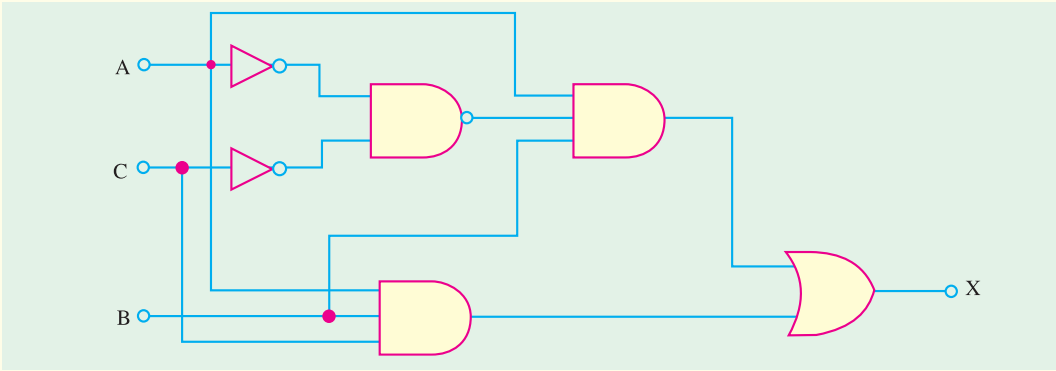


Fig. 71.75

13. Fig. 71.76 (*a*) shows a three-variable Karnaugh map. Group the 1*s* and hence obtain the minimized Boolean expression. ***(AB + BC + $\overline{A}$ $\overline{B}$ $\overline{C}$)***

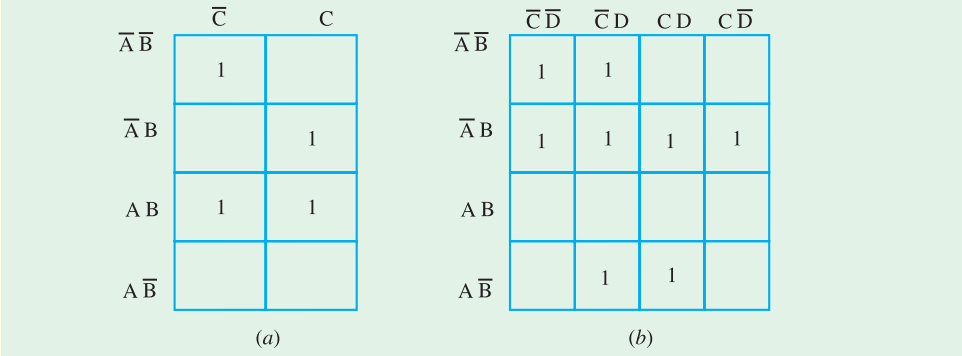


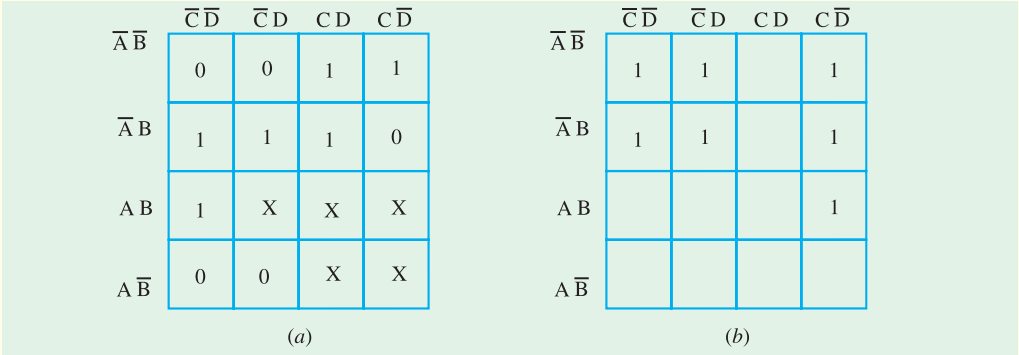
Fig. 71.76

14. Fig. 71.76 (*b*) shows a four-variable Kamaugh-map. Group the 1*s* and hence obtain the minimized Boolean expression. ***($\overline{A}$ B + $\overline{A}$ $\overline{C}$ + A$\overline{B}$D)***

15. A truth table has output 1*s* for these inputs : $ABCD = 0011$, $ABCD = 0110$, $ABCD = 1000$ and $ABCD = 1100$, and 0*s* for the other inputs. Draw the Karnaugh map and find the simplified Boolean equation for the truth table. ***(A$\overline{C}$ $\overline{D}$ + $\overline{A}$ $\overline{B}$ CD + $\overline{A}$BC$\overline{D}$)***

16. Determine the minimized expression for the Karnaugh map shown in Fig. 71.77 (*a*).

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



	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	0	0	1	1
$\overline{A}B$	1	1	1	0
AB	1	X	X	X
$A\overline{B}$	0	0	X	X

(*a*)

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1		1
$\overline{A}B$	1	1		1
AB				1
$A\overline{B}$				

(*b*)

Fig. 71.77

17. Determine the simplified Boolean expression for the Karnaugh map shown in Fig. 71.77 (*b*).

(UPSC Engg. Services 1995) $(\overline{A}\,\overline{C} + \overline{A}\,\overline{D} + ABC)$

OBJECTIVE TESTS – 71

1. Boolean algebra is essential based on
(*a*) symbols (*b*) logic
(*c*) truth (*d*) numbers

2. The first person who used Boolean algebra for the design of relay switching circuits was
(*a*) Aristotle (*b*) Boole
(*c*) Shannon (*d*) Ramanujam

3. Different variables used in Boolean algebra can hvae values of
(*a*) 0 or 1 (*b*) low or high
(*c*) true or false (*d*) On or OFF.

4. According to the algebra of logic, $(A + \overline{A})$ equals
(*a*) A (*b*) 1
(*c*) 0 (*d*) $\overline{AA}$.

5. According to the absorptive Laws of Boolean algebra , expressioin $(A + AB)$ equals
(*a*) A (*b*) B
(*c*) AB (*d*) A

6. When we demorganize $\overline{A}\,\overline{B}$, we get
(*a*) $\overline{AB}$ (*b*) $\overline{A} + \overline{B}$
(*c*) $\overline{A + B}$ (*d*) $\overline{AB}$.

7. The dual of the statement $(A + 1) = 1$ is
(*a*) $A \,.\, 1 = A$ (*b*) $A \,.\, 0 = 0$
(*c*) $A + A = A$ (*d*) $A \,.\, A = 1$.

8. The expression $\overline{ABC}$ can be simplified to
(*a*) $\overline{A} \,.\, B \,.\, \overline{C}$ (*b*) $AB + BC + CA$
(*c*) $AB + \overline{C}$ (*d*) $\overline{A} + \overline{B} + \overline{C}$.

9. While obtaining minimal sum-of-products expression,
(*a*) all don't cares are ignored
(*b*) all don't cares are treated as logic 1*s*
(*c*) all don't cares are treated as logic 0*s*
(*d*) only such don't cares that aid minimization are treated as logic 1*s*.

10. In a saturated bipolar logic circuit, transistor operate
(*a*) in deep cut-off (*b*) over active region
(*c*) in saturation
(*d*) just short of saturation

11. Saturated logic circuits have inherently
(*a*) short saturation delay time
(*b*) low switching speed
(*c*) higher power dissipation
(*d*) lower noise immunity.

12. Noise margin is expressed in
(*a*) decibel (*b*) watt
(*c*) volt (*d*) phon

13. DTL family employs
(*a*) resistors and transistors
(*b*) diode and resistor
(*c*) diode and transistors
(*d*) diodes, resistors and transistors.

14. The chief advantage of Schotty *TTL* logic family is its least
(*a*) power dissipation (*b*) propagation delay
(*c*) fan-in (*d*) noise immunity.

15. The main advantage claimed for *ECL* family oflogic gates is its
(*a*) very large fan-in
(*b*) use of negative power supply voltage
(*c*) extremely low propagation times
(*d*) least power dissipation.

16. Special feature of an I^2L logic circuit is that it
(*a*) uses ony high-value resistors

(*b*) dissipates negligible power
(*c*) is a bipolar saturated logic
(*d*) is easy to fabricate
(*e*) uses no biasing and loading resistors.

17. A unique advantages feature of *CMOS* logic family is its
(*a*) use of *NMOS* circuits
(*b*) power dissipation in nanowatt range
(*c*) speed
(*d*) dependence on frequency for power dissipation.

18. CMOS circuits are extensively used for on-chip computers mainly because of their extremely
(*a*) low power dissipation
(*b*) large packing density
(*c*) high noise immunity
(*d*) low cost.

19. The main advantage of a CMOS logic family over the TTL family is its
(*a*) much reduced power
(*b*) increased speed of operation
(*d*) extremely low cost
(*d*) series base resistor

20. CMOS logic family uses only
(*a*) MOSFETs and resistors
(*b*) NMOS circuits
(*c*) MOSFETs (*d*) bipolar transistors.

21. Power is drawn by a CMOS circuit only when
(*a*) its output is high (*b*) its output is low
(*c*) it switches logic levels
(*d*) in static state.

22. The most obvious identifying feature of a *TTL* gate is its
(*a*) large fan-out (*b*) high power dissipation
(*c*) interconnected transistors
(*d*) multiemetter input transistor.

23. In digital circuits shottky transistors are preferred over normal transistor because of their
(*a*) lower propagation delay
(*b*) higher propagation delay
(*c*) lower power dissipation
(*d*) higher power dissipation

24. A unique operting feature of *ECL* circuit is its
(*a*) very high speed
(*b*) high power dissipation
(*c*) series base resistor
(*d*) compatibility with other logic series.

25. The fan-in a logic gate refers to the number of
(*a*) input devices that can be connected
(*b*) input terminals (*c*) output terminals
(*d*) circuits output can drive

26. Which of the following statements regarding *ICs* is not correct,
(*a*) *ECL* has the least propagation delay
(*b*) *TTL* has the largest fan out
(*c*) *CMOS* has the biggest noise margin
(*d*) *TTL* has the lowest power consumption.

27. The Boolean equation for *NOR* gate is
(*a*) $\overline{AB} = C$ (*b*) $\overline{A} + \overline{B} = C$
(*c*) $A + B = C$ (*d*) $\overline{A + B} = C$.

28. The Boolean equation for a NAND gate is
(*a*) $\overline{A + B} = C$ (*b*) $\overline{\overline{AB}} = C$
(*c*) $A + B = C$ (*d*) $\overline{A} + \overline{B} = C$

29. The logic function of an invertor is
(*a*) $B = A$ (*b*) $B = \overline{A}$
(*c*) $B = \overline{\overline{A}}$ (*d*) None of these

30. The simplified form of the Boolean expression $Y = (\overline{A}.BC + D)\ (\overline{A}.D + \overline{B}.\overline{C})$ can be written as
(*a*) $\overline{A}.D + \overline{B}.\overline{C}.D$ (*b*) $AD + B.\overline{C}.D$
(*c*) $(\overline{A} + D)(\overline{B}.C + \overline{D})$ (*d*) $A.\overline{D} + BC.\overline{D}$

(GATE; 2004)

ANSWERS

1. (*b*) **2.** (*c*) **3.** (*a*) **4.** (*b*) **5.** (*a*) **6.** (*c*) **7.** (*b*) **8.** (*d*) **9.** (*d*) **10.** (*c*) **11.** (*b*) **12.** (*c*)
13. (*d*) **14.** (*b*) **15.** (*c*) **16.** (*e*) **17.** (*b*) **18.** (*b*) **19.** (*a*) **20.** (*c*) **21.** (*c*) **22.** (*d*) **23.** (*a*) **24.** (*a*)
25. (*b*) **26.** (*d*) **27.** (*d*) **28.** (*d*) **29.** (*b*) **30.** (*a*)

GO To FIRST