UNIT 11 FUNDAMENTALS OF BOOLEAN ALGEBRA AND FLIP FLOPS

Structure

11. INTRODUCTION

A digital circuit is designed for a desired application by a combination of several logic gates. This application involving several logic gates may be a simple or complex one. Different users may design digital circuits by using different combinations of logic gates for the same application. In selecting one of these digital circuits for that application, it is necessary to keep in mind that the chosen digital circuit should have a minimum number of logic gates. By seeing a digital circuit, it is not'obvious that a circuit is minimal or certain gales may be removed from the circuit without changing its operation. Boolean algebra provides a means by which logic circuitry may be expressed symbolically, manipulated and reduced.

In this Unit we shall learn about three basic logic gales: AND, OR, NOT and their various combinations. All digital (logic) circuits operate in the binary mode where all the inputs and outputs are predefined voltages representing binary digit either 1 or 0. It is this characteristics of the logic circuits that enables us to use boolean algebra for designing and analysing the digital systems. This area of digital circuitry is known as combinational logic where the relationship between the inputs and outputs can be precisely defined by the logic summarised in a truth table.

In the combinational logic circuits there is no memory, i.e. the output of the digital circuit does not depend upon the occurrence of a previous event. But it is very essential for more advanced digital circuits meant for storing and manipulating information to have memory. The basic memory element is a flipflop which is obtained by using NAND or NOR gates. In this Unit we shall learn about various kinds of flipflops and their operation. This area of digital circuitry is known as sequential circuits.

Objectives

After studying this unit, you should be able to

- describe the operation of AND, OR and NOT Gates and write their truth tables,
- describe the combination of gates and write the truth tables of NAND and NOR gates,
- explain as to how a timing diagram of the output of all the logic circuits is obtained,
- explain how the operation of three basic logic gates lead us to various theorems or rules used in the boolean algebra,
- write boolean theorems and use algebraic method for combinational logic,
- obtain a truth table from a give boolean expression,
- describe the operation of exclusive—OR and exclusive—NOR gates,
- design a half adder and describe its operation,
- design a full adder and describe its operation,
- design logic circuits using only NAND gates,
- describe the construction and explain the operation of the RS flipflop,
- describe the construction and explain the operation of clocked RS flipflop, D flipflop, and JK flipflop,
- obtain the timing diagrams of the outputs of flipflops.

11.2 LOGIC GATES

A logic gate is a digital circuit which has logical relationship between input and output voltages. There are three basic gates: AND, OR and NOT (also called inverter) gates. We shall now learn these gates one by one.

2 1 AND Gate

The AND gate can be understood by the circuit given in Fig. 11.1. In this circuit switch (s) is input and the bulb is output. Let us assign 0 to the event when the switch is open and 1 to the event when the switch is close. Similarly when the bulb does not glow we call it 0 and when the bulb glows we call it 1. With both the switches (A and B) off, the bulb (Y) does not glow.

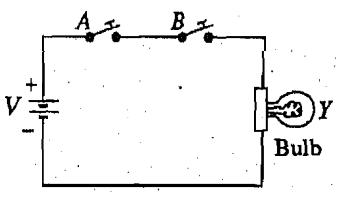


Fig. 11.1: AND gate using switches.

With one of the switches off and another switch on, once again the bulb (Y) does not glow., However, with bath the switches (A and B) on, the bulb (Y) glow. Thus there are four events which can be summarised in the form of a table which is called the truth table of this circuit. This is given in Table 11.1. The switches A and B, which control the input voltage are usually called input of the truth table and Y as the output.

Table **11.1:** Truth Table **of** AND Gate.

inputs		output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

From this table it is clear that the bulb glows (1) only when both the switches (A and B) are on (1). Stated in a different way, the output is 1 when both the inputs A and B are 1. This state of the circuit is distinct from other three states. This circuit is known as the AND gate. The symbol of AND gate is given in Fig. 11.2. It is clear from the Fig. 11.1 that if the circuit has any number of switches in series, then the output will be 1 if and only if all inputs are 1. Now for all times to come, you must remember that' **For** an **AND gate the** output is **1 if and only if all** the inputs are **1.**

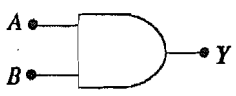


Fig. 11.2: Symbol **of AND gate.**

Electronically the AND gate can be realised by using two pn junction diodes as shown in the circuit of Fig. 11.3. The resistor R is used to control the current passing through the diodes. As stated above, a 0 bit is assigned 0V and a 1 bit is assigned 5V. However, such accurate values of voltage will not always be available at the output in electronic circuits. Therefore, a 0 bit is assigned a voltage range of 0 to 0.8V and a 1 bit is assigned to a voltage range of 2.8 to 5.0V. Quite often these voltage ranges are referred to a LOW and a HIGH respectivefy. The voltages greater than 0.8V and less than 2.8 V are indeterminate and hence not used.

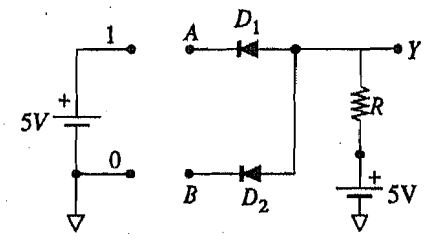


Fig. 11.3: Realisation of AND Gate using diodes.

In the circuit of Fig. 11.3 when the inputs A and B are 0, i.e. when they are connected to the 0V or ground terminal, both the diodes are forward biased with a voltage drop of 0.7V across each diode if the diodes are of Si or of 0.3V if the diodes are of Ge. Hence the output voltage is a LOW or a 0 bit. If the input A is 0 and B is 1 (i.e. 5V), the diode A is forward biased with 0.7V drop across it (assuming diode to be of Si) while the diode B is not biased (because both p and n sides of the diode are at the same voltage, 5V). Therefore the output voltage is 0.7V, i.e. a LOW or a 0 bit. Similarly, if the input A is 1 and input B is 0, the output is a 0. However, if both inputs are 1, i.e. connected to 5V, then both the sides of the diodes are at the same voltage and hence not conducting. Therefore, the output voltage is nothing but the battery voltage which is 5V, i.e. a HIGH or a 1 bit. These four cases satisfy the truth table of Table 11.1. For more input AND gate, the number of diodes may be more. The input output relationship of the AND gate is written as **A.B = Y** and is read as **A** AND **B** equal to **Y.**

Example 11.1

If the inputs **A and** B to the AND gate are as shown in Fig. 11.4, trace the output Y.

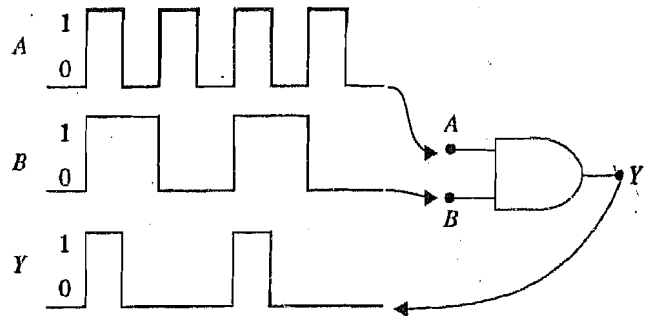


Fig. 11.4:

Solution

Recall that output of an AND gate is 1 when **all** the inputs are 1. If any of the inputs is 0, then the output is 0. With this understanding, the output comes out to be as shown in the trace for Y.

SAQ 1

Trace the output of an AND gate, if the inputs **A** and B are as shown in Fig. **11.5.**

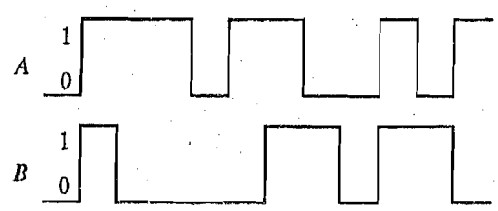


Fig. 11.5:

11.2.2 OR Gate

The OR gate operation can be understood by the circuit of Fig. 11.4. If both the switches are off, (0), the bulb **does** not glow (0). If one of the switches is on (1) and other is off (0), the bulb glows (1). And if both the switches **are** on (1), then also the bulb glows (1). These events are summarised in the truth table given in Table **11.2.**

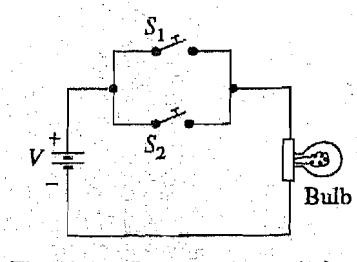


Fig. 11.6: OR gate using switches.

Table 11.2: Truth **Table of OR** Gate.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

t is clear from the truth table that the output of OR gate is 0 if hoth the inputs are 0 nd the output is 1 if any one of the inputs or both the inputs are 1. If a larger number of switches are used in parallel in the circuit, then the bulb does not glow if all the witches are off, and the bulb glows if any one of the switches is on. The symbol of)R gate is given in Fig. 11.7. The OR gate operation is expressed as A + B = Y and s read as **A** O R B = **Y.**

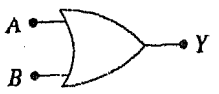


Fig. 11.7: Symbol of OR gate.

Electronically OR gate can be realised by using two pn junction diodes as shown in the circuit of Fig. 11.8. If both the inputs are 0, that is connected to the ground, then the diodes are not biased and hence no current flows through the diodes. The output is zero or a 0 bit. If the inputs to diode A is 0 and B is 1 (i.e. 5V), then the diode A is not biased and thus does not conduct, but the diode B is forward biased with a 0.7V drop across it and 4.3V drop across the resistor. Thus the output is a HIGH or a 1 bit. Similarly, if the inputs to the diode A is 1 and diode B is 0, the output is 1. When the inputs to both the diodes **A** and B are 1, both the diodes are forward biased, the voltage drop across the resistor R continues to be 4.3V. Hence, the output is a 1 hit. All these Four cases satisfy the truth table of OR gate. A more input OR gate is obtained by using more diodes in the circuit. Analysing the truth table of OR gate, we learn that the output is **0 if** both or **all** the **inputs are** 0, and the output is **1 if** at **least one** of **the** input **is 1.**

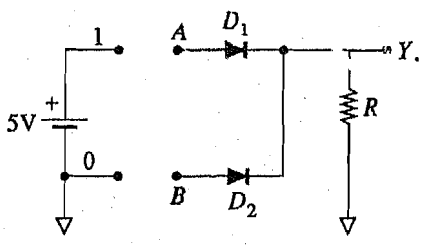


Fig. 11.8: Realisation of OR gate **using diodes.**

Example 11.2

I the inputs A and B to OR gate are as shown in Fig. 11.9, trace the output Y.

Recall that the output of an OR gate is **1** if **any** of the **input** is 1, and the output is 0 if all the inputs are 0. With this understanding, the output comes out to be **as** shown in the race for Y.

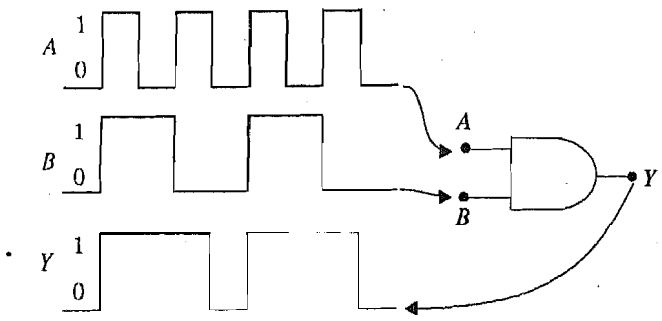


Fig. 11.9:

SAQ 2

Trace the output of an OK gate if the inputs **A** and B are as shown in Fie. 11.10

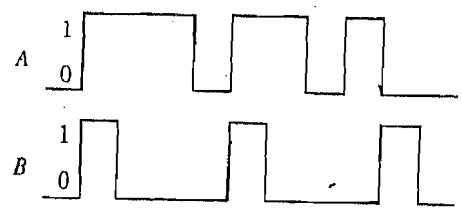


Fig. 11.10:

11.2.3 NOT Gate

The NOT gate can be understood by considering the electrical circuit shown in Fig. 11.11. Let us assign a 0 bit to the even when bulb does not glow and 1 bit to the event when bulb glows, and a 0 bit to switch off and 1 bit to switch closed. In Fig. 11.11, when switch is closed, no current will pass through the bulb and the bulb will not glow. This is because the current always flow through the least resistance path. Similarly, when the switch is open then the whole current will flow through the bulb making it glow.

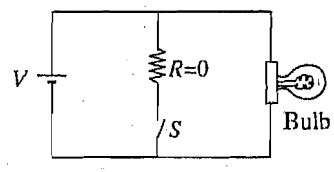


Fig. 11.11: NOT gate using a switch.

If input to the circuit is 1, the output is 0 and if the input is 0 then the output is 1. This is the NOT gate operation which is summarised in the truth table given in Table 11.3.

Table 11.3: Truth **table for** NOT gate.

A	Y
0	1
1	0

The NOT gate is also known as INVERTER. It has only one input. Its symbol is given in Fig. 11.12. The input-output relationship is expressed as A = Y.

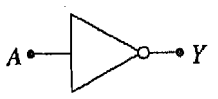


Fig. 11.12: Symbol of **NOT gate.**

The NOT gate can be realised using the circuit given in Fig. 11.13. The circuit uses the cutoff and saturation modes of the transistor. When the input to the circuit is **a** 0 hit, i.e. zero volt, no base current, I_B. Ilows. This means the collector current, I_C, is zero. This is cutoff mode of the transistor.

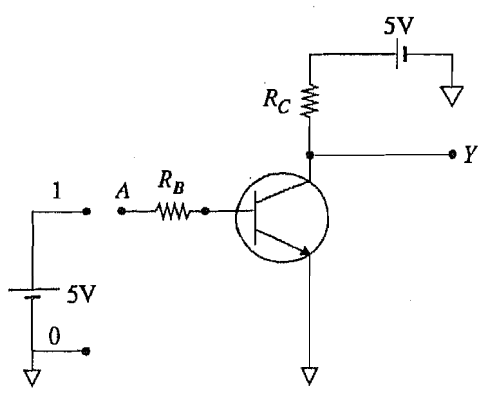
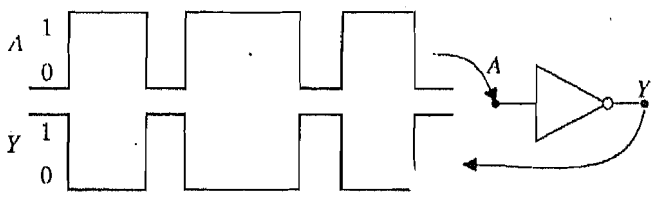


Fig. 11.13: Realisation of **NOT gate using a transistor.**

Therefore, the output voltage is the bias voltage of 5V indicating the output to be a 1 bit. When the input to the circuit is a 1 bit, i.e. 5V, very large I_B flows resulting in very large I_C, in fact $I_{C_{sat}}$. This is the saturation mode of the transistor. This indicates that most of the bias voltage is dropped across R_C with output to be a 0 bit.

Example 11.3

If the input **A** to NOT gate is as shown in Pig. 11.14, trace the output Y.



Pig. 11.14:

Solution

Recall that the output of a NOT gate is I if the input is 0, and the output is 0 if the input is I. With this understanding, the output comes out to be as shown in the trace for Y in Fig. 11.14

SAQ 3

Trace the output of a NOT gate if the input is as shown in Fig. 11.15.

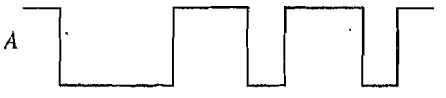


Fig. 11.15:

11.2.4 Combination of Logic Gates

The AND, OR and NOT gates are the fundamental gates for all digital circuits. These gates can be combined with each other for a particular application. However, two types of combinations are very important as you will learn now.

NAND Gate

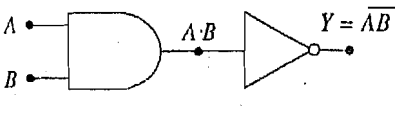


Fig. 11.16: Combination of AND and NOT gate.

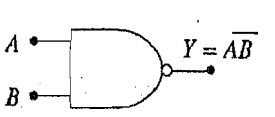


Fig. 11.17: Symbol of NAND gate.

If the output of an AND gate is given to the input of a NOT gate, as shown in Fig. 11.16, the resulting circuit is known as NAND gate the symbol for which is shown in Pig. 11.17. The truth table of this gate is obtained as follows:

A	B	Y' (AB)	Y
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Thus the truth table of NAND gate is shown in Table 11.4.

Table 11.4: Truth table for NAND gate.

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

The input-output relationship of a NAND gate is expressed as A·B = Y. The NAND gate is known as the building block for the digital circuits because using NAND gates, one can obtain AND, OR and NOT gates. This aspect will he explained later.

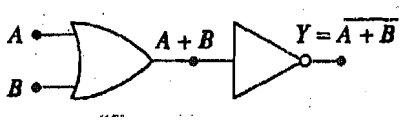


Fig. 11.18: Combination of OR and NOT gate.

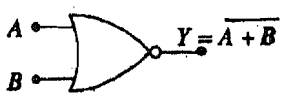


Fig. 11.19: Symbol of NOR gate.

If the output of an OR gate is given to the input of a NOT gate, as shown in Fig. 11.18, the resulting circuit is known as NOR gate the symbol for which is shown in Fig.' 11.19. The truth table of this gate is obtained as follows:

A	B	Y′ (A + B)	Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Thus the truth table of a NOR gate is shown in Tabla 11.5.

Table 11.5: Truth table for NOR gate.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

The input-output relationship of a NOR gate is expressed as $\overline{A + B} = Y$. The NOR gate is also known as the building block for the digital circuits because using NOR gates one can obtain AND, OR and NOT gates.

Example 11.4

If the inputs A and B to NAND gate are as shown in Fig. 11.20, trace the output Y.

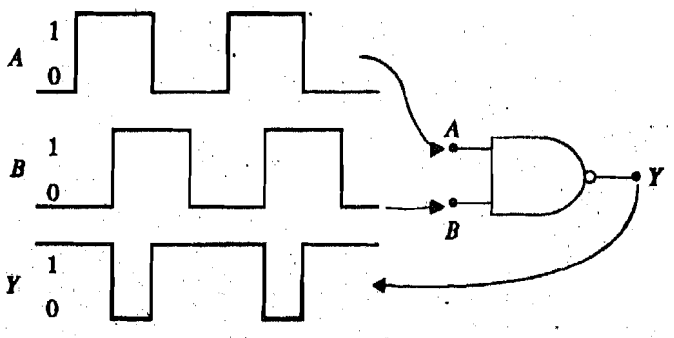


Fig. 11.20:

Solution

Recall that the output of a NAND gate is 0 only when all the inputs are 1, and its output is 1 if any or all of the inputs is/are 0. With this understanding, the output comes out to be as shown in the trace for Y.

SAQ 4

If the inputs A and B to a NOR gate are as shown in Fig. 11.21, trace its output Y. (Hint. Apply truth table 11.5).

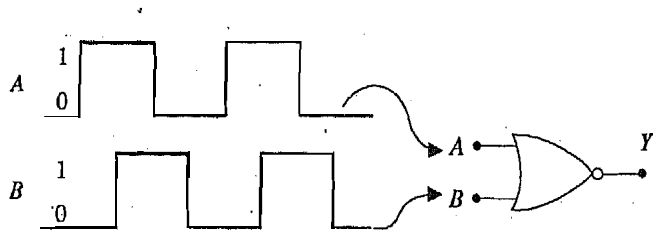


Fig. 11.21

11.3 BOOLEAN ALGEBRA

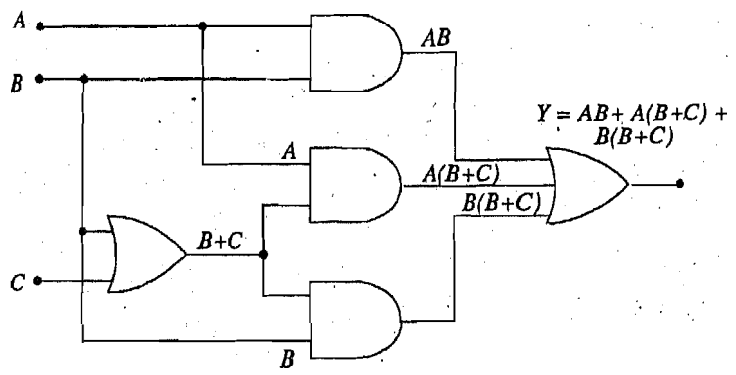
In this section we shall learn about the Boolean algebra which provides the methodology, for reducing a complex digital circuit into a simple one. This methodology includes the following:

1) The logic operations are written in the form of a Boolean expression.

2) From the given truth table, a boolean expression can be obtained which may not represent a simple circuit having minimum number of gates.

3) The boolean expression may then be simplified to get a digital circuit having minimum number of gates.

Consider the digital circuit given in Fig. 11.22. It has five logic gates of three types –



Pig. 11,22: Digital circuit using five gates.

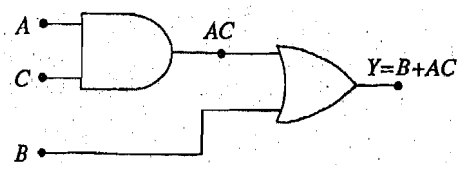


Fig. 11.23: Digital circuit having the same operation as that of the circuit given in Fig. 11.22.

three 2-input **AND** gates, one 2-input OR gate and one 3-input OR gate. Its logic table, is given in Table 11.6. This circuit can be reduced to the one shown in Fig. 11.23 which has only two logic gates and is considerably cheaper and simple. It fully satisfies the logic Table 11.6.

Table 11.6

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

The root of its initial assumptions, known as boolean postulates, lies in the truth tables of the logic .gates described in the previous section: Let us recall that the AND operation has been described by the sign of multiplication (·), that is logical multiplication. Most often we do not use this sign (.), e.g. $A \cdot B = AB$. Similarly the OR Operation has been described by the sign of addition (+), that is logical addition. And the NOT operation has been described as a bar (–) over the variable, that is logical inversion or complementation. These three operations are the basic Boolean operations based upon which we shall develop the Boolean algebra.

Since the number of bits used in binary system is only two, i.e. 0 and 1, there could be only four possible combinations of inputs A and B to 2-input AND and OR gates, and two possible inputs to NOT gate. The logical tables of AND, OR and NOT gates are rewritten in Table 11.7.

Table 11.7: Truth tables of AND, OR and NOT gates,

AND

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

NOT

X	Z
0	1
1	0

These logic tables lead to ten postulates of the boolean algebra, each of which describes the input-output relationship of the concerned logic gate in the form of boolean expression and is one of the truth table entries for AND, OR, NOT functions. These are:

Table 11.8: Boolean expression for AND, OR and NOT gates.

AND operation	OR operation	NOT operation
$0 \cdot 0 = 0$	$0 + 0 = 0$	$\overline{0} = 1$
$0 \cdot 1 = 0$	$0 + 1 = 1$	$\overline{1} = 0$
$1 \cdot 0 = 0$	$1 + 0 = 1$	
$1 \cdot 1 = 1$	$1 + 1 = 1$	

It is quite clear from these equations that all the four Boolean equations using AND operation satisfy the binary multiplication using bits 0 and 1. However, in the case of OR operation, while first three Boolean equation satisfy binary addition, but the last equation 1 + 1 = 1 does not. It is because in binary arithmetic 1 + 1 = 10. Despite this contradiction between Boolean and binary additions which will be settled later, the Boolean operations are very helpful in digital circuits. The Table 11.8 will lead us to various Boolean theorems which will be described in the following section.

For the moment let us see how Boolean equations are written and used for a digital circuit. Consider the circuit of Fig. 11.24 in which A and B are the inputs to AND gate

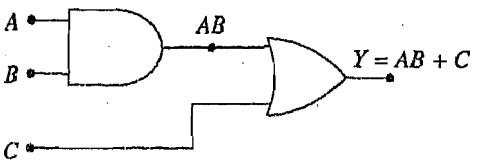


Fig. 11.24: Digital circuit for Y = A·B + C.

while. C is one of the inputs to OR gate. 'Another input to OR gate is the output of AND gate, i.e. AB. The output of this combination is Y which is

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

Let us find Y if, say, A = 0, B = 1, and C = 1.

$$Y = 0 \cdot 1 + 1$$

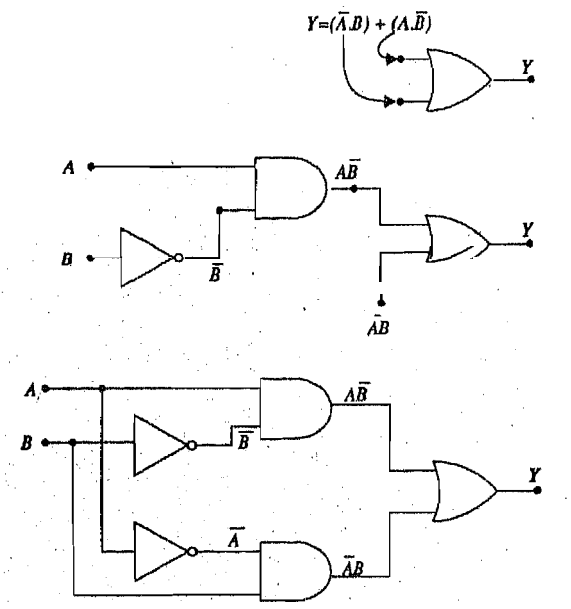
From Table 11.8, 0·1 = 0, so

$$Y = 0 + 1$$

From Table 11.8, 0 + 1 = 1. Hence,

$$Y = 1$$

Let us now convert a given Boolean expression into a logic circuit. Say, $Y = (\bar{A} \cdot B) + (A \cdot \bar{B})$. The equation means that Y is the output of a 2-input OR gate the inputs to which are $\bar{A} \cdot B$ and $A \cdot \bar{B}$ which in turn are the outputs of two AND gates. The inputs to these AND gates are $\bar{A}$ and B and A and $\bar{B}$ respectively. The whole of this exercise is summarised in the Fig, 11.25.



Rg. 11.25: Conversion of a boolean expression $Y = \bar{A}B + A\bar{B}$ into a digital circuit

11.3.1 Boolean Theorems

Recalling Table 11.8 we can now write several identities or, theorems which are used in Boolean algebra. It is also worthwhile to recall that

A. i) Output of an AND gate is 1 only when all the inputs are 1.

ii) Output of an AND gate is 0 when all or any of the inputs is 0.

B. i) Output of an OR gate is 0 when all the inputs are 0.

ii) Output of an OK gate is 1 when either of the inputs or all the inputs are 1.

C. Output of a NOT gate is inversion of its input.

From these conclusions and postulates, we derive the following properties or rules/law/ theorems:

From AND function,

1. $X \cdot 0 = 0$
2. $0 \cdot X = 0$
3. $X \cdot 1 = X$
4. $1 \cdot X = X$

From OR functions,

5. $X + 0 = X$
6. $0 + X = X$
7. $X + 1 = 1$
8. $1 + X = 1$

Combination variable with itself or its complement.

9. $X \cdot X = X$
10. $X \cdot \overline{X} = 0$
11. $X + X = X$
12. $X + \overline{X} = 1$

From double complementation.

13. $\overline{\overline{X}} = X$

Commutative laws for multiplication and addition. These laws show that the order in which two variables are ORed or ANDed together makes no difference.

14. $X \cdot Y = Y \cdot X$
15. $X + Y = Y + X$

Associative laws for addition and multiplication. These laws show while ORing or ANDing several variables, it makes no difference in what order the variables are grouped.

16. $X + (Y + Z) = (X + Y) + Z = X + Y + Z$
17. $X(YZ) = (XY)Z = XYZ$

Distributive laws.

18. $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$
19. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$
20. $(W + X) \cdot (Y + Z) = WY + XY + WZ + XZ$

Note here that commutative, associative and distributive laws are similar to ordinary algebra.

Absorption laws. These have no counterpart in ordinary algebra.

21. $X + X \cdot Y = X$

22. $X \cdot (X + Y) = X$

23. $X + \bar{X}Y = X + Y$

24. $X \cdot (\bar{X} + Y) = XY$

DeMorgan's theorems. First theorem says that the complement of a sum is equal to the product of complements:

25. $\overline{X + Y} = \bar{X} \cdot \bar{Y}$

Second theorem says that the complement of a product is equal to the sum of complements.

26, $\overline{X \cdot Y} = \bar{X} + \bar{Y}$

These theorems are valid even when the variables are expressions. There is no algebraic proof of these theorems. However, each theorem/law can be proved by putting the values (0 or 1) of variables and applying boolean postulates given in Table 11.8.

11.3.2 Algebraic Method for Combinational Logic

We have now know that a logic circuit can be expressed in the form of boolean expression which, in turn, can be simplified using boolean laws. We have also known that a boolean expression can also be transformed into an equivalent logic circuit.

Before we learn the simplification method and other techniques, let us understand the meaning of combinational logic. Whenever a logic circuit is explicitly defined by its truth table to provide a fixed, invariant relationship between input and output, the circuit is called the combinational circuit. A combinational circuit does not have a memory. It always operates in accordance with its truth table regardless of any prior input which may have been given to the circuit. This will be further understood after we have taken up some examples.

A boolean expression can be simplified in either of the two forms — (a) Sum of Product (SOP), and (b) Product of Sum (POS). We shall limit ourselves to only SOP form which is most commonly used. Object of simplification is to minimise the number of variables or occurrences of a variable in an expression. This means minimising operation symbols and hence the number of gates to be used in the circuit. Many a times we gel more than one simplified form of an expression, each being equivalent in number of gates and variables to be used. In final analysis, we shall use the Minimum Sum of Product (MSP) form which is written without brackets. Consider the reduced expression. A (B + C) which is written in MSP form AB + AC. While the reduced expression requires one AND gate and one OR gate, the MSP expression requires one AND gates and one OR gate. Thus in this case MSP expression is not the simplest. Fundamental rule is that the expression must be (a) reduced as much as possible, and (b) written without brackets. For the simplification of boolean expression, boolean operations should be carried out in the following order:

1) Inversion of single variables.

2) All operations with brackets.

3) AND operations before OR operations.

4) OR operatians.

5) If an expression is with a bar, then before inverting perform all operations.

Example 11.5

1) Find the MSP expression for

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

$= (\overline{A} + \overline{B})\,\overline{C} + (\overline{A} + \overline{B})$ Using DeMorgan's theorem, Th. 26

$= (\overline{A} + \overline{B})\,[\overline{C} + 1]$ Taking $(\overline{A} + \overline{B})$ common

$= (\overline{A} + \overline{B}) \cdot 1$ Using Th. 7

$= (\overline{A} + \overline{B})$ Using Th. 3

= MSP expression.

The logic circuits for the given and the MSP expressions are shown in Figs 11.26 and 11.27 respectively.

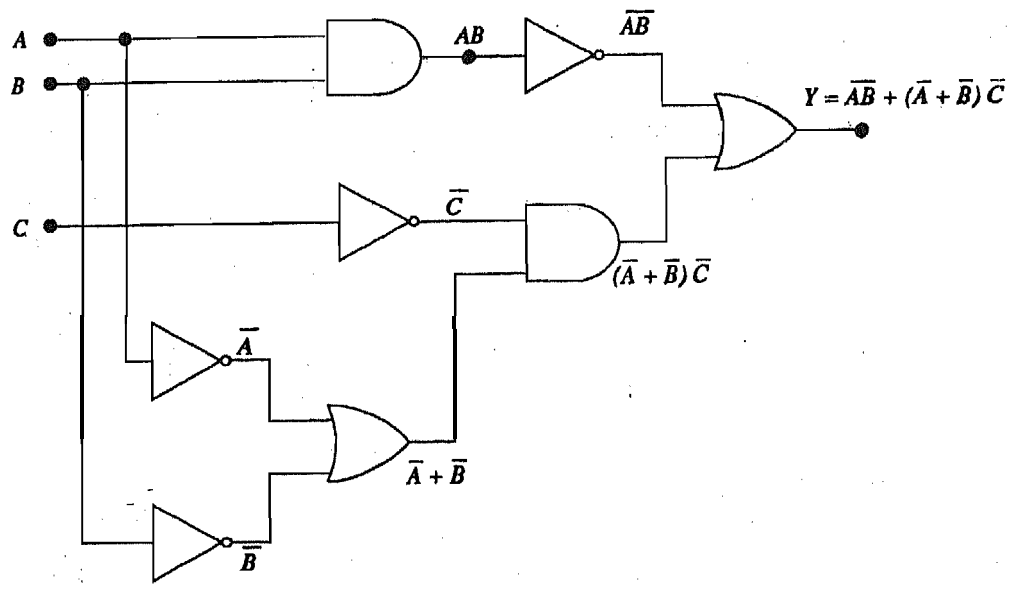


Fig. 11.26: Digital circuit for $Y = (\overline{A} + \overline{B})\,\overline{C} + \overline{A}\overline{B}$.

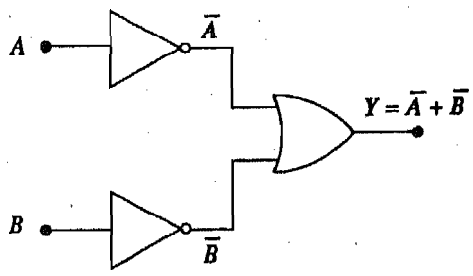


Fig. 11.27: Digital circuit for $Y = \overline{A} + \overline{B}$.

Example 11.6

Find the MSP expression for

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

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

$= \overline{A}C + A \cdot 0 + ABC$ Using Th. 10

$= \overline{A}C + ABC$ Using Th. 1

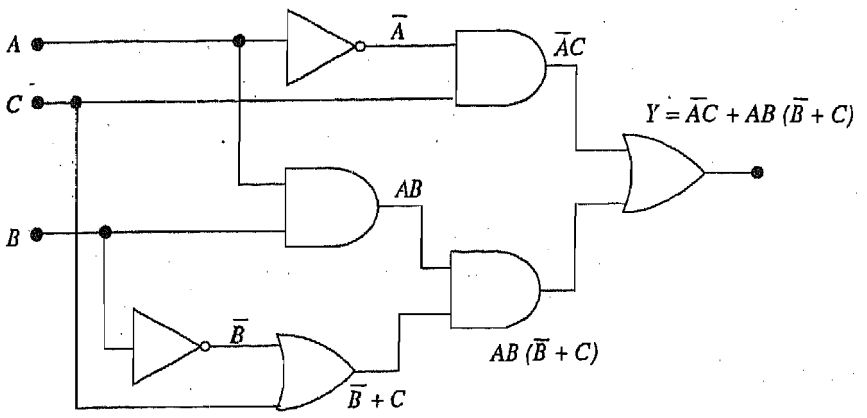
$= (\overline{A} + AB)\,C$ Taking C common

$= (\overline{A} + B)C$ Using Th. 23

$= \overline{A}C + BC$

= MSP expression

The logic circuits for the given and the MSP expressions are shown in Figs. 11.28 and 11.29 respectively.



Pig. 11.28: Digital circuit for Y = $\bar{A}$C + AB ($\bar{B}$ + C).

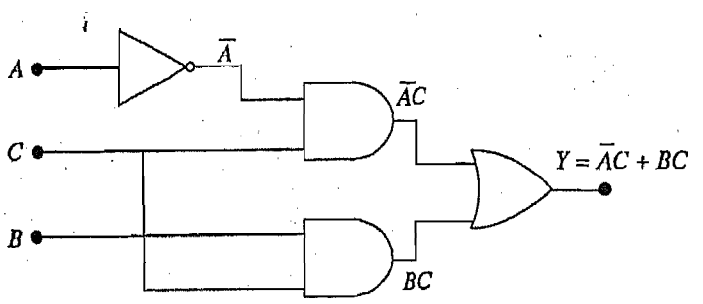


Fig. 11.29: Digital circuit for Y = $\bar{A}$C + BC.

Example 11.7

Find the MSP expression for

Y	=	AB + A (B + C) + B (B + C)	
	=	AB + AB +AC + BB + BC	
	=	AB + AB + AC + B + BC	**Using Th. 9**
	=	AB + AC + B + BC	**Using Th. 11**
	=	AB + AC + B(1 + C)	**Taking B** common
	=	AB + AC + B · 1	**Using Th. 8**
	=	AB + AC + B	**Using Th. 3**
	=	(A + 1) B + AC	
	=	1 · B + A C	**Using Th. 7**
	=	B + AC	**Using Th. 4**
	=	. **MSP expression.**	

The logic circuits for the given and the MSP expressions are shown in Figs. 11.22 and 11.23 respectively.

SAQ 5

Find the MSP expression for Y = A$\bar{B}\bar{C}$ + AB$\bar{C}$ + ABC.

11.3.3 Obtaining a Truth Table from a Boolean Expression

A simplest method of obtaining the truth table from a boolean expression has already been mentioned. That is, substitute the values of variables in each possible combinations of values in the expression. Perform all the Iogic operations and get the result for each combination. For example,

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

In this expression, say, A = 1, B = 0, and C = 0, then

$$\begin{aligned} Y &= 1 \cdot 0 + 1 \cdot (0 + 0) + 0\,(0 + 0) \\ &= 0 + 1 \cdot 0 + 0 \cdot 0 \\ &= 0 + 0 + 0 \\ &= 0 \end{aligned}$$

Similarly, find Y for all combinations of values for A, B, and C, and complete the truth table which is given in Table 11.9.

Table 11.9: Truth table for $Y = AB + A(B + C) + B(B + C)$

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

The alternative method of obtaining a truth table from a boolean expression involves reasoning. Ask yourself:

When shall the output of the expression be 1. Consider the expression

$Y = A\overline{C} + BC$ = MSP expression

This expression is 1 so long as either $A\overline{C}$ or BC is 1. Therefore, put Y = 1 for all entries of $A\overline{C}$ = 1 (i.e. entries 5 and 7). Then put Y = 1 for all entries of BC = 1 (i.e. entries 4 and 8). Now, Y for all other entries is 0. Table 11.10 is thus the truth table for the given expression.

Table 11.10: Truth table for $Y = A\overline{C} + BC$.

	A	B	C	Y
1.	0	0	0	0
2.	0	0	1	0
3.	0	1	0	0
4.	0	1	1	1
5.	1	0	0	1
6.	1	0	1	0
7.	1	1	0	1
8.	1	1	1	1

Hence, it is better to use the method of reasoning for obtaining the truth table. This method involves just two steps:

1) Obtain the MSP form of the given boolean expression, and

2) Reason out which of the truth table entries should be 1 for each product in MSP form.

Example 11.8

Obtain the truth table for the boolean expression Y = A + AB + BCD.

$$\begin{aligned} Y &= A + AB + BCD \\ &= A(1 + B) + BCD \\ &= A \cdot 1 + BCD \\ &= A + BCD \\ &= \text{MSP expression} \end{aligned}$$

Reasoning out we find that Y = 1 whenever A = 1 or the product BCD = 1. Therefore, in the truth table for this expression, put Y = 1 for all entries of A = 1, (i.e. entries 9 to 16) and put Y = 1 for all entries of product BCD = 1 (i.e. entries 8 and 16). For all other entries put Y = 0 (i.e. entries 1 to 7). The complete truth table is given in Table 11.11.

Table 11.11: Truth table for Y = A + AB + BCD.

	A	B	C	D	Y
1.	0	0	0	0	0
2.	0	0	0	1	0
3.	0	0	1	0	0
4.	0	0	1	1	0
5.	0	1	0	0	0
6.	0	1	0	1	0
7.	0	1	1	0	0
8.	0	1	1	1	1
9.	1	0	0	0	1
10.	1	0	0	1	1
11.	1	0	1	0	1
12.	1	0	1	1	1
13.	1	1	0	0	1
14.	1	1	0	1	1
15.	1	1	1	0	1
16.	1	1	1	1	1

SAQ **6**

Obtain the truth table for Y = AB + BC + CA.

11.3.4 Obtaining a Boolean Expression from a Truth Table

Consider the truth table given in Table 11.12.

Table 11.12: Given truth table.

	A	B	C	Y
1.	0	0	0	0
2.	0	0	1	0
3.	0	1	0	0
4.	0	1	1	0
5.	1	0	0	1
6.	1	0	1	0
7.	1	1	0	1
8.	1	1	1	1

Note, that the entries 5, 7, and 8 contribute a logic 1 to the operation while all other entries give a logic 0. To obtain the boolean expression, we need only write a product term for each entry that contribute a logic 1, and then assemble the operations by connecting all the products with a logic OR. Do as follows.

Entry 5: $Y = 1$ for $A = 1, B = 0, C = 0$

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

because the output of an AND gate will be 1 only if all the inputs are 1. Similarly,

Entry 7: $Y = 1$ for $A = 1, B = 1, C = 0$

$= AB\overline{C}$

Entry 8: $Y = 1$ for $A = 1, B = 1, C = 1$

$= ABC.$

Now connect all the three products with an OR logic. Hence

$$Y = A\overline{B}\overline{C} + AB\overline{C} + ABC \text{ (Sum of Product)}$$

Which can be simplified as

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

The procedure can be summarised as follows:

1) Combine with an AND operation all the input variables for the entries that contribute a logic 1.

2) Select for each variable in the product an overbar or no overbar so that when the input values of the entries are substituted, the product gives a logic 1. These products are also known as fundamental products.

3) The products are assembled with an OR operation.

4) The sum of product expression thus obtained may not be minimal. Use boolean algebra to bring an **SP** expression in an **MSP** form.

SAQ 7

Obtain the boolean expression for the truth table given below:

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

11.3.5 Exclusive — OR (XOR) Gate

An XOR gate gives a high output (i.e. 1) when an odd number of inputs is high. A two-input exclusive — OR gate has its output 1 if one of the two inputs is 1 and the other is 0, and if both the inputs are same then the output is 0. The truth table of an **XOR** gate is given in Table 11.13.

Table 11.13: Truth table for XOR gate.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Its boolean expression is obtained from the entries 2 and 3, that is

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

This expression is in MSP form because it can not be simplified further. Thus Y is the output of an OR gate the inputs to which are $\bar{A}B$ and $A\bar{B}$, which in turn are the outputs of two AND gates. The circuit thus obtained for an XOR gate is given in Fig. 11.30 and it is represented by the symbol shown in Fig. 11.31. The XOR operation is expressed by $\oplus$.

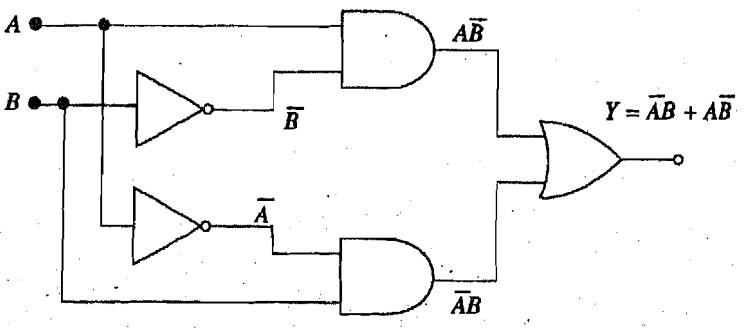


Fig. 1130: Exclusive — OR (XOR) gate.

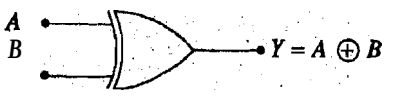


Fig. 1131: Symbol of XOR .gate.

11.3.6 Exclusive — NOR (XNOR) Gate

An exclusive — NOR gate has its output 1 if both the inputs are same, and if both the inputs are different then the output is 0. The truth table of an XNOR gate is given in Table 11.14.

Table 11.14: Truth **table** for **XNOR gate.**

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Its boolean expression is obtained from the entries 1 and 4, that is

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

This expression is in MSP form because it cannot be simplified further. Thus Y is the output of an OR gate the inputs to which arc $\overline{A}\,\overline{B}$ and AB, which in turn are the outputs of two AND gates. The circuit thus obtained for an ***XNOR*** gate is given in Fig. 11.32 and is represented by the symbol shown in Ag. 33.

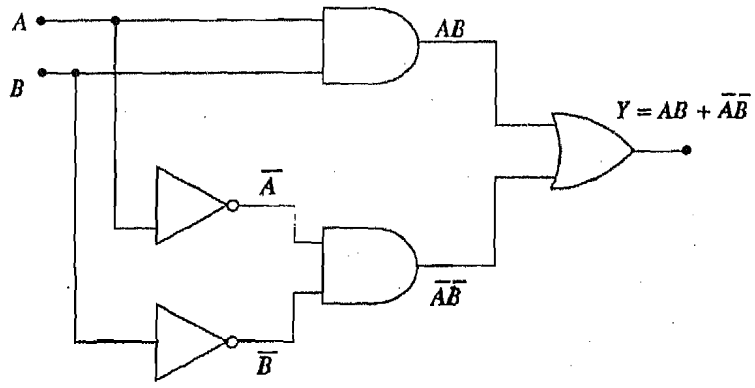


Fig. 11.32: Exclusive —NOR (XNOR) gate.

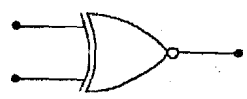


Fig. 1133: Symbol of XNOR gate.

11.3.7 Addition of Two One Bit Binary Numbers (Half Adder)

Recall the binary addition learnt in Unit 10. The binary addition of two single bit binary numbers is as follows.

0	0	1	1
+0	+1	+0	+1
00	01	01	10

In this example of addition, the bit on the right hand side is sum while the bit on the left hand side is carry. This can be put in a truth table as shown in Table 11.15.

Table **11.15:** Truth **table** for **half** adder.

A	**B**	Carry	**Sum**
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

This application has two outputs, one for 'sum' and another for 'carry'. Therefore, we have to obtain two boolean expressions for the two outputs.

The expression for carry is

$$\text{Carry} = AB$$

that is, it is the output of an AND gate.

The expression for sum is

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

that is, it is the output of an XOR gate described in the previous section. These two circuits are connected together as shown in Fig. 11.34. This circuit is known as half adder and its symbol is given in Fig. 11.35.

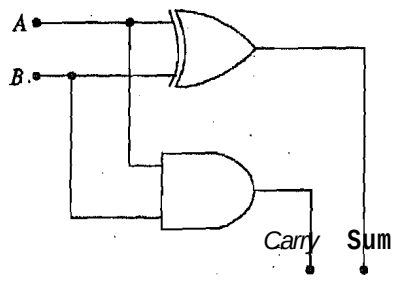


Fig. 1134: Half adder circuit.

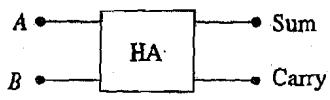


Fig. 1135: Symbol of half adder.

Recall the contradiction pointed out while describing addition by an OR gate. While it could justify addition in case of its first three entries of inputs, it could not give correct result of addition of binary numbers in its last entry of inputs, **i.e.** it gave $1 + 1 = 1$ (boolean addition) rather than $1 + 1 = 10$ (binary addition). This contradiction is now taken care of by the design of half adder. We can now say that the binary addition **should** be done using half adder or circuits described later in the Unit. But as far as boolean postulates, including based on OR gate, are concerned, they are helpful in designing circuits for **binary arithmetic.**

11.3.8 Addition of Three One Bit Binary Numbers (Full Adder)

The full adder can add three single-bit binary numbers. The binary addition three single-bit binary numbers is as follows:

0	0	0	0	1	1	1	1
+0	+0	+1	+1	+0	+0	+1	+1
+0	+1	+0	+1	+0	+1	+0	+1
00	01	01	10	01	10	10	11

The right hand bits of these additions represent the sum and the left hand bits represent the carry. These eight possible combinations of three single-bit binary numbers can be presented in the form of a truth table given in Table 11.16.

Table 11.16: Truth table for full adder.

A	B	C	Carry	Sum
0	0	0	**0**	**0**
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

In order to design the logic circuit for a full adder boolean expressions have to be written and simplified in **MSP** form for both sum and carry which are as follows:

$$\begin{aligned}
\text{Sum} &= \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC \\
&= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC) \\
&= \bar{A}(B \oplus C) + A(\overline{B \oplus C}) \\
&= \bar{A}X + A\bar{X} \qquad \text{where } X = B \oplus C \\
&= A \oplus X \\
&= A \oplus B \oplus C \\
&= \textbf{MSP expression.}
\end{aligned}$$

This is the output of a 3-input **XOR** gate.

$$\begin{aligned}
\text{Carry} &= ABC + A\bar{B}C + AB\bar{C} + ABC \\
&= BC(\bar{A} + A) + A\bar{B}C + AB\bar{C} \\
&= BC + A\bar{B}C + AB\bar{C} \\
&= C(B + A\bar{B}) + AB\bar{C} \\
&= C(B + A) + AB\bar{C} \\
&= BC + AC + AB\bar{C} \\
&= BC + A(C + B\bar{C}) \\
&= BC + A(C + B) \\
&= BC + AC + AB \\
&= \text{MSP expression.}
\end{aligned}$$

From these two MSP expressions, the logic circuit for a full adder can be obtained as described earlier. This circuit is given in Fig. 11.36 and its symbol is given in Fig. 11.37.

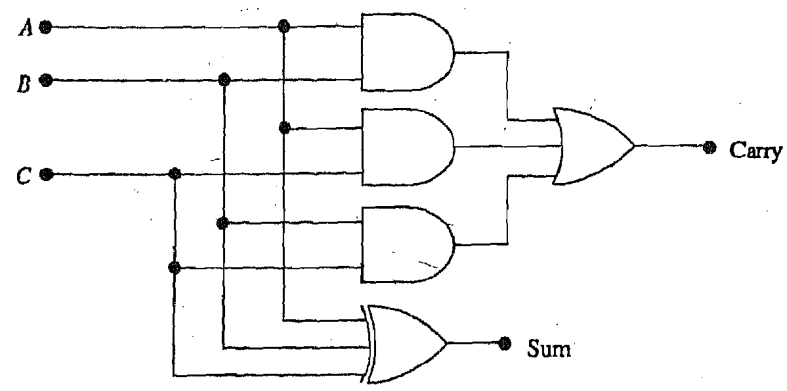


Fig. 1136: Full adder **circuit.**

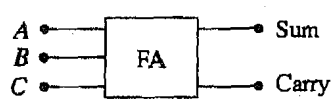


Fig. 11.37: Symbol of full adder.

You would recall that a computer or a digital circuit can add only two binary **numbes** at a time. If a digital circuit has to add more than two binary numbers, as would **mostly** be the case, the circuit will add first two binary numbers and to the sum of these two numbers it will add the third binary number, and so on. But while adding two bits a carry is likely to appear as shown above. Therefore if the two binary numbers to be added are having more than one bit, then after the addition of first bits of the numbers the addition of second bits will also require the addition of any carry which appears from the addition of first bits. Thus the **addition** of first bits can be carried out by the **half** adder which has two inputs, but the addition of second bits require a 3-input adder which is realised by the full adder. There are eight entries to the truth table of a full adder, half of which are satisfied by the truth table of half adder ignoring carry bit (because the addition of first bits of two numbers **do** not have a carry to be added). For this reason, the adder described in the previous section is called the half adder and the described in this section is called the full adder.

Example 11.8

Addition of two 4-bit binary numbers. Let us say the numbers are $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$. This addition requires one half adder to add A_0 and B_0 and three full adders to add rest of the bits as shown in the circuit of Fig. 11.30. The outputs of the half adder are sum (S_0) and **carry.** The carry output of the half adder is given as the third input to the first full adder which has a carry output and a sum (S,) output. The carry output of the first full adder is given to the second full adder, and so on. Thus for addition of two 4-bit binary numbers, **we** require one half adder and three full adders. For each additional bit in the numbers to be added, we require one more full adder.

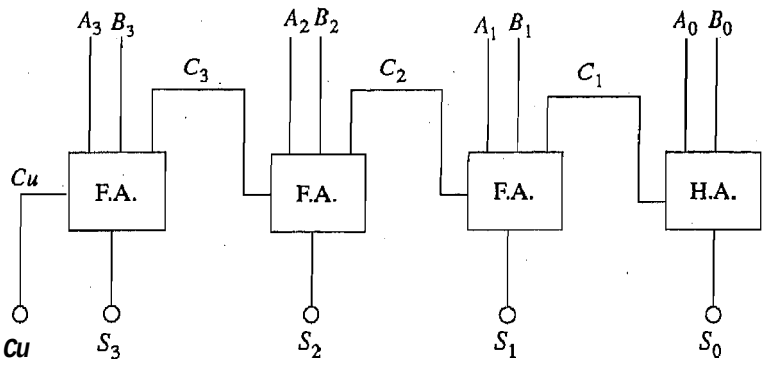


Fig. 11.38: A 4-bit binary adder.

SAQ 8

Draw a digital circuit for a 2-bit binary adder.

11.3.9 Designing Circuits Using NAND Gates Only

Quite often it is required that only **NAND** gates should be used in designing digital circuits. The **NAND** gate being universal can be used to realise **AND,** OR and **NOT** gates. Therefore, wherever these gates are appearing, the equivalent **NAND** circuit is used. The realisation of **AND, OR** and **NOT** gates from **NAND** gates is shown in Fig. 11.39.

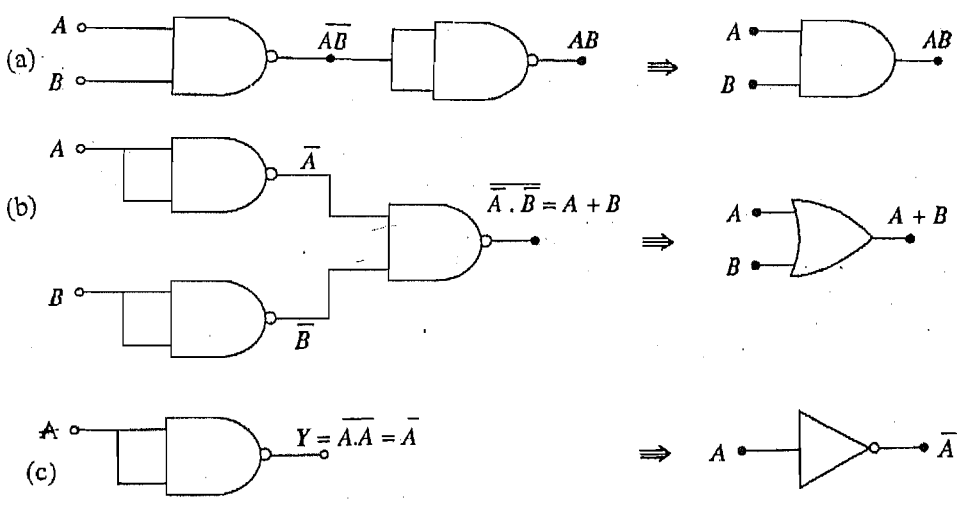


Fig. 11.39: Realisation of (a) AND, (b) OR, and (c) NOT gates using NAND gates.

Example 11.9

Design a circuit for Y = AB + **CD** using **NAND** gates only.

The circuit for Y = **AB + CD** using AND and OR gates is shown in Fip. 11.40.

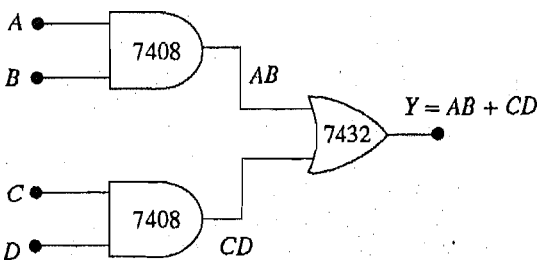


Fig. 11.40: Digital circuit for Y = AB + CD.

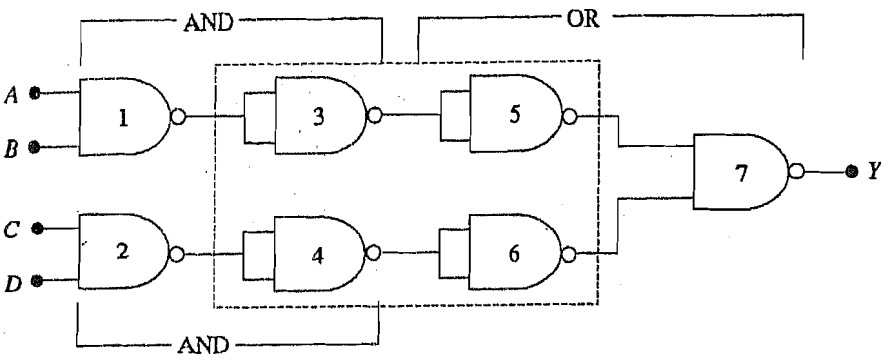
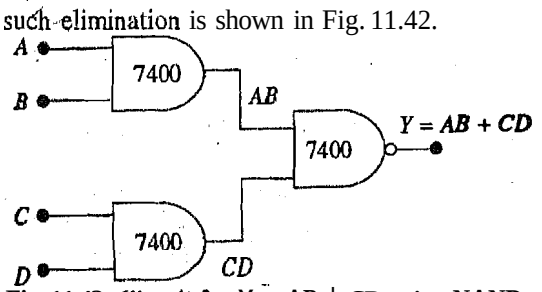


Fig. 11.41: AND and OR gates in the circuit given in Fig. 11.40 replaced *by* their equivalents.

The **AND** gates and **OR** gate in Fig. 11.40 are replaced by equivalent NAND gate circuits from Fig. 11.39 as shown in Rg. 11.41. It requires two NAND **ICs.** Since the input and output of a combination shown as dotted of a NOT gate followed by another **NOT** gate are same, therefore such a combination is useless and hence eliminate it. The final circuit after such elimination is shown in Fig. 11.42.



Fig, 11.42: Circuit for Y = AB + CD using NAND *gates*.

Another method involves the use of Demorgan's theorems. Consider the example of XOR gate. It requires one NOT IC, one AND IC and one OR IC, i.e. three ICs in total.

The MSP equation for XOR gate is $Y = \bar{A}B + A\bar{B}$. Double complement the right hand side and solve using DeMorgan's theorem.

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

The right hand side is the output of a **NAND** gate the inputs to which are the outputs of two NAND gates, i.e. $\overline{(\bar{A}B)}$ and $\overline{(A\bar{B})}$. The final circuit for XOR gate using NAND gates only is shown in **Fig.** 11.43. It requires two **NAND** ICs.

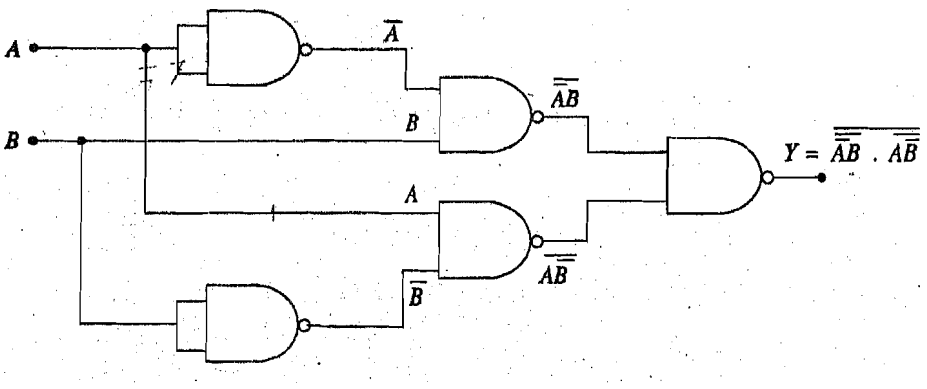


Fig. 11.43: Circuit for XOR *gate* using *NAND* gates only.

SAQ 9

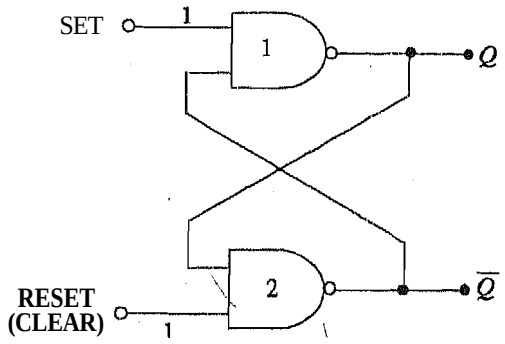
Design a digital circuit for Y = A + BC using NAND gates only.

1.4 FLIPFLOPS

We have learnt combinational logic circuits in the previous section. The combinational logic circuits operate strictly in accordance with their truth table. However, there are logic circuits which hnvc feedback path and the operation of which is not strictly defined by their truth tables. Such circuits operate differently for a given input condition depending upon the prior input sequence applied to the circuit. Such circuits arc known as sequential logic circuits. These circuits have memory element also. In addition to the logic gates, a computer requires memory element. The simplest memory element is a flipflop. It has two stable states and remains in any one of these two stable states until triggered into the other state. Quite often the flipflop is also known as a latch.

11.4.1 RS Flipflop

The most basic flipflop circuit is constructed using two NAND gates or two **NOR** *gates.* In NAND gate flipflop, two NAND gates are cross-coupled as shown in Fig. 1 1.44. **It** has two latched outputs Q and $\overline{Q}$. It has two inputs: SET (S) arid RESET (R) or CLEAR (C). The input names signify their actions as well. For the input names such a flipflop is known as RS flipflop.



Pig. 11.44: RS flipflop.

Let us now understand the working of a RS flipflop. Both the inputs, SET and RESET, are kept HIGH, i.e. at logic 1. In the beginning, let us say S = R = 1. With the outputs Q = 0 and $\overline{Q}$ = 1, NAND-1 has the inputs 1 and 1 hence Q = 0, and **NAND-2** has inputs 1 and 0, hence $\overline{Q}$ = I. These outputs are latched or stuck with each other and continue to be latched until input conditions are changed.

Second possibility with S = R = 1 is when Q = 1 and $\overline{Q}$ = 0. The **NAND-1** will have 1 and 0 inputs giving Q = 1. Likewise the **NAND-2** will have 1 and 1 inputs giving $\overline{Q}$ = 0. Once again the two outputs are latched together and they will continue to be latched until input conditions are changed. S and R both high means the two sets of possible outputs remains in its last state indefinitely because of the internal latching action. Thus, a high S and a high R gives us the inactive state; the circuit stores or remembers. When we want to change the flipflop output one of the inputs will be pulsed LOW (i.e. logic 0).

Setting the Flipflop

Let us say that the SET is mometarily pulsed LOW (i.e. S = 0 for a moment) while RESET continues to be 1. Now if Q = 0 and $\overline{Q}$ = 1 prior to the occurrence of a LOW pulse at SET, Q goes 1 which in turn forces $\overline{Q}$ to a 0. Thus when SET returns to 1, the NAND-I output remains HIGH which in turn keeps the NAND-2 output at 0.

If prior to the application of SET pulse, Q = 1 and $\overline{Q}$ = 0, then a LOW pulse at **SET** will not change anything because $\overline{Q}$ = 0 is already keeping the **NAND-1** output to 1. Thus when SET returns to 1, the outputs are still Q = 1 and $\overline{Q}$ = 0.

Thus a LOW on the SET input will always cause the flipflop to end up in Q = 1 state. Hence, this operation is called setting the flipflop, and Q = 1 state is known as SET state.

Resetting or Clearing **the Flipflop**

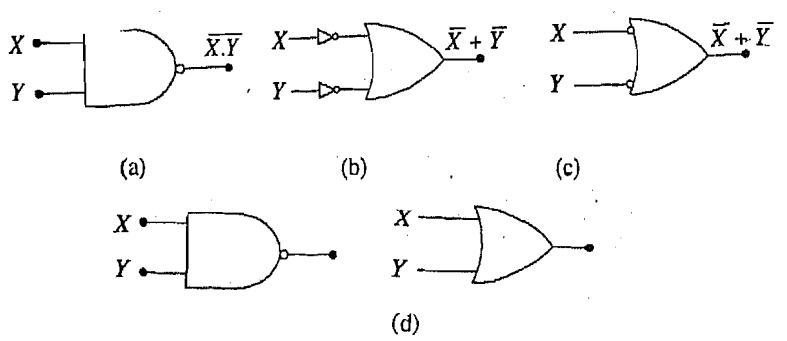
The SET is kept at 1 and RESET is momentarily pulsed LOW (i.e. 0). Let us say that prior to the pulse, Q = 0 and $\overline{Q}$ = 1. Since Q = 0 is already keeping the NAND-2 output at 1, therefore the application of a LOW pulse at RESET will not change the situation. However, if prior to the application of a LOW pulse, Q = 1 and $\overline{Q}$ = 0 , then a LOW pulse at RESET will give NAND-2 output as 1, which in turn forces the NAND-1 output to a 0. Thus a LOW at RESET always ends up in Q = 0. This operation is called clearing or resetting operation. And Q = 0 state is known as CLEAR (or RESET) state.

When SET and CLEAR are simultaneously pulsed LOW, it produces 1 at both the outputs. There is a race to come to a 1 state. This is an undesired state because Q and $\overline{Q}$ are inverse of each other. When R and S return to 1, race among the two will give unpredictable results. Therefore, R = S = 0 is not used. However, as described above, R = S = 1 produces no change in the outputs. The entire operation of the RS flipflop is summarised in the truth table given in table 11.17.

Table **11.17: Truth** table **for** RS **flipflop.**

S	R	Output
1	1	NC (No change)
0	1	Set (Q = 1)
I	0	Reset (Q = 0)
0	0	*(Race and invalid)

The DeMorgan equivalent of NAND gate is given in Fig. 11.45. Fig. 11.45 (a) represent the left side of DeMorgan's theorem. The right side of the theorem implies that the inputs are inverted before reaching an OR gate (see Fig. 11.45b). This combination is used so often that the abbreviated symbol shown in Fig. 11.45c has 'come into use: This symbol is called a bubbled OR gate. Fig. 11.45d is a graphic summary of DeMorgan's theorem which shows that a NAND gate and a bubbled OR gate are equivalent. Therefore, we can replace one with the other whenever desired.



Pig. 11.45: DeMorgan equivalent of NAND gate.

Using DeMorgan equivalent of a NAND gate, the NAND gate flipflop can also be represented by the circuit shown in Fig. 11.46. Tho symbol of this flipflop is shown in Fig. 11.47. The bubble at the S and R inputs indicate that the flipflop can be set or reset by living a LOW pulse.

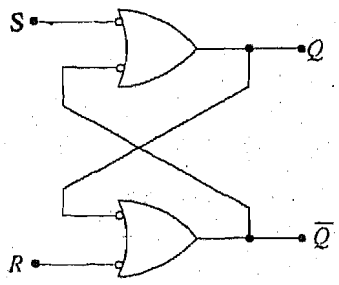


Fig. 11.46: DeMorgan equivalent **of** NAND gate **RS flipflop.**

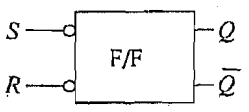


Fig. 11.47: Symbol of RS flipflop.

Example 11.10

If the train of pulses given to the S and R inputs of SR flipflop are as shown in Fig. 11.48(a) and (b) respectively, then trace its Q output. Initial value of Q is given to be 0.

Solution

Using Table 11.17, the Q output of the RS flipflop is as shown in Fig. 11.48(c).

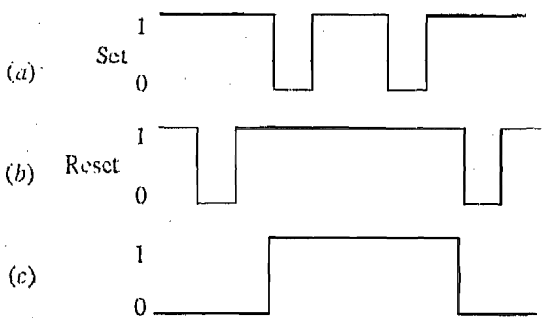


Fig. 11.48: Set/Reset pulses and the output.

SAQ 10

What is the shape of the Q output of RS flipflop if the S and R inputs me as shown in Fig. 11.49? Initial value of Q is given to be 1.

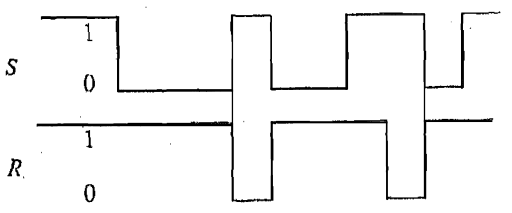


Fig. 11.49:

11.4.2 Clocked RS Flipflop

Computers use thousand of flipflops. To coordinate the overall action, a square wave signal called the **clock** is sent to each flipflop. The clock is applied to all flipflops simultaneously; this ensures that. they all change stales in unism. This synchronization is essential in many digital systems.

In most of the synchronous systems the output can change only wiien the clock signal is making a transition from 0 to I, i.e. positive going transition **(PGT)** or 1 to 0, i.e. negative going transition (NGT), These systems are known as edge triggered. The PGT and NGT are shown in Fig. 11.50. The symbols of edge triggered RS flipflop which work with PGT and NGT are shown symbolically in Figs. 11.51(a) and (b) respectively.

Note the difference in symbol of clock activated by a PGT and NGT. The change in the control inputs R and S to the flipflop will not effect a change in the Q output until an active clock (CLK) transition, i.e. a PGT in case of Fig. 11.51(a) and a NCT in case of Fig. 11.51(b), occurs. The control inputs keep the flipflop ready to change and the active clock transition at the CLK input actually triggers the change. To ensure that a clocked flipflop responds properly when the active clock transition occurs, the inputs must be stable, i.e. unchanging.

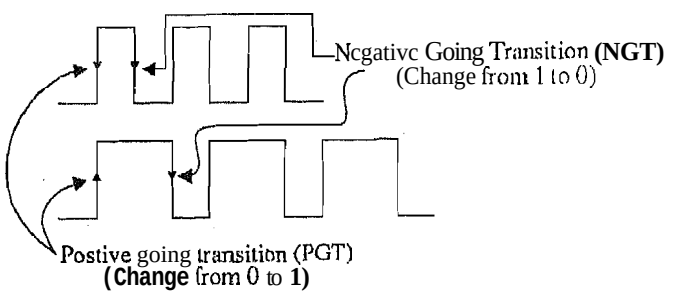


Fig. 11.50: Positive and negative going transitions.

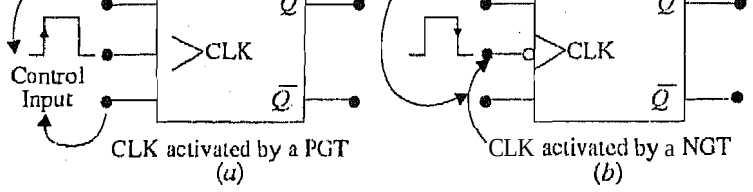


Fig. 11.51: Symbol of edge triggered flipflop activated by a (a) PGT, and (b) NGT.

Consider the circuit given in Fig. 11.52 in which two additional NAND gates are used as the clock pulse steering circuit and is triggered by a PGT. A LOW (i.e. 0) clock CLK prevents S and R from controlling the fliptlop, because with whatever values of

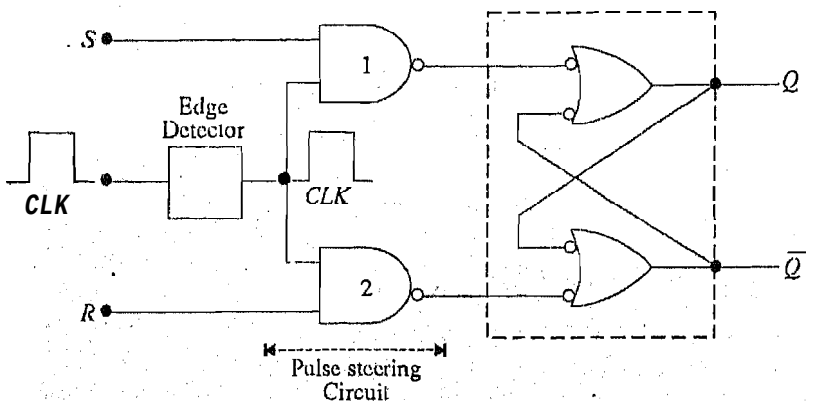


Fig. 11.52: Circuit of edge triggered RS flipflop.

S and R the outputs of the NAND-1 and NAND-2 will be 1 which will not produce any change in the **Q** output of the flipflop. However, when the CLK is HIGH (i.e. 1) and S = R = 0, the: outputs of the two NAND gates will be 1 and there would be no change in the Q output.

Table 11.18 shows the truth table for a positive edge triggered RS flipflop. The $Q = Q_0$ is output level before the arrival of the PGT of the CLK. The arrow directed upward (↑) indicates that a PGT is required at the CLK.

Table **11.18:** Truth table **for** a positive edge triggered RS flipflop.

Inputs			Output
R	S	CLK	Q
0	0	↑	Q_0 (No change)
0	1	↑	1
1	0	↑	0
1	1	↑	*Race

The inputs S and R, and corresponding Q output, assuming the initial value of Q, i.e. Q_0, equal to 0, are as shown in Fig. 11.53. It is clear that at the arrival of first clock transition both R and S are 0, therefore there is no change in the Q output which continues to be 0. But at the arrival of the second cluck transition S is 1 and R is 0, this sets the flipflop with Q = 1 which does not change till third clock transition. At the time of the third clock transition R is 1 and S = 0 which resets the flipflop with Q = 0. This is how the Q output is traced. Note that between two PGTs of the CLK, the Q output does not change. It must be remembered that whenever tracing a Q output corresponding to the inputs, you have to look for the active clock, note the values of inputs nand then decide the value of the Q output.

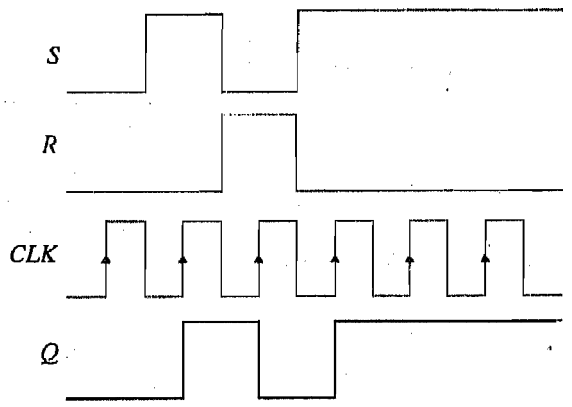


Fig. **11.53:** Inputs and **output** of a clocked RS **flipflop.**

The truth table of a RS flipflop triggered by a NGT is shown in Table 11.19.

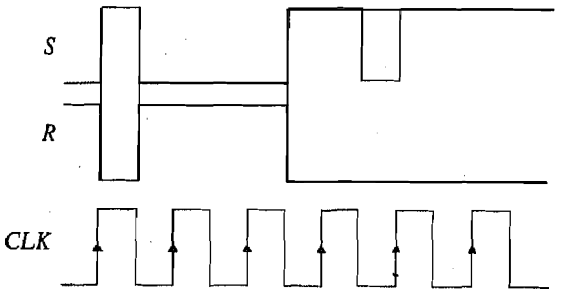
Table **11.19:** Truth table for a negalive edge triggered RS **flipflop.**

Inputs			Output
R	S	CLK	Q
0	0	↓	Q_0 (No change)
0	1	↓	1
1	0	↓	0
1	1	↓	*Race

The PGT or NGT can be obtained by using a combination of gates or a differentiating circuit consisting of a capacitor and a resistor.

SAQ 11

If the train of pulses to S and R inputs of a clocked RS flipflop are as shown in Fig. 11.54, and if the initial value of Q is 0, trace its Q output.



Pig. 11.54:

11.4.3 Clocked D Flipflop

The RS flipflop has two inputs S and R. Generating two signals to drive a flipflop is a disadvantage in many applications. Furthermore, the race condition of both S and R low may occur inadvertently. In order to eliminate the possibility of a race condition a new kind of flipflop is designed. This is called a D flipflop. The letter D stands for the data. The data input is given to S-input of the RS flipflop while the same input goes to its R-input through an inverter as shown in Fig. 11.55. The symbol of the edge triggered D flipflop activated by a PGT is shown in Fig. 11.56. Its truth table is given in Table 11.20 which shows that the Q output of D flipflop follows the input data D. The D input and corresponding Q output, assuming initial Q to be 1, are shown in Fig. 11.57.

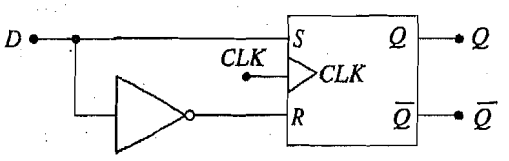


Fig. 11.55: Circuit for D flipflop.

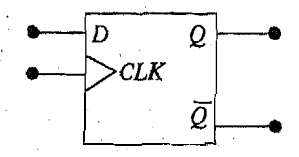


Fig. 11.56: Symbol of D flipflop.

Table 11.20: Truth table fur a positive edge triggered D flipflop.

D	CLK	Q
0	↑	0
1	↑	1

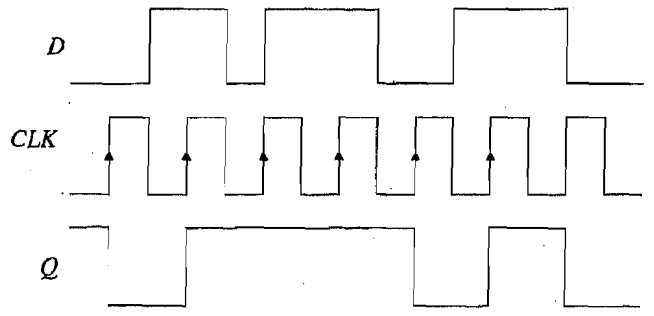


Fig. 11.57: Input and output of a D flipflop.

D Latch

Sometimes edge trigger detecting circuit (like RC combination) for D flipflop is not used. In this case the D flipflop functions slightly differently and is known as D latch. Instead of edge triggering, level clock or an ENABLE (abbreviated as EN) signal is used as shown in Fig. 11.58. When ENICLK is 1, D will produce a 0 at either SET or CLEAR inputs of the NAND latch to give a Q output to be at the same level of D.

Fig.11.58: Circuit for D latch.

When EN/CLK is 1, if D changes, Q will follow changes exactly like D as the Q output does not have to wait for the clock transition to respond to changes in D. The D latch is thus 'transparent' to the input in this mode. When EN/CLK is at 0, D is inhibited from affecting NAND latch because the outputs of both steering NAND gates will be 1. Thus Q and Q continue to stay wherever they were before ENICLK became 0. In other words, the outputs are latched to their current level and cannot change during the period ENICLK is 0, even if D changes. The truth table of D latch is given in Table 11.21.

Table 11.21: Truth table for D latch.

D	EN/CLK	Q
X	0	NC
0	1	0
1	1	1

Quite often two AND gates are introduced between the pulse steering circuit and the NAND latch as shown in Fig. 11.59. One input each of these AND gates are known as RESET (direct SET) and CLEAR (direct RESET) and are kept at 1 so as to allow the

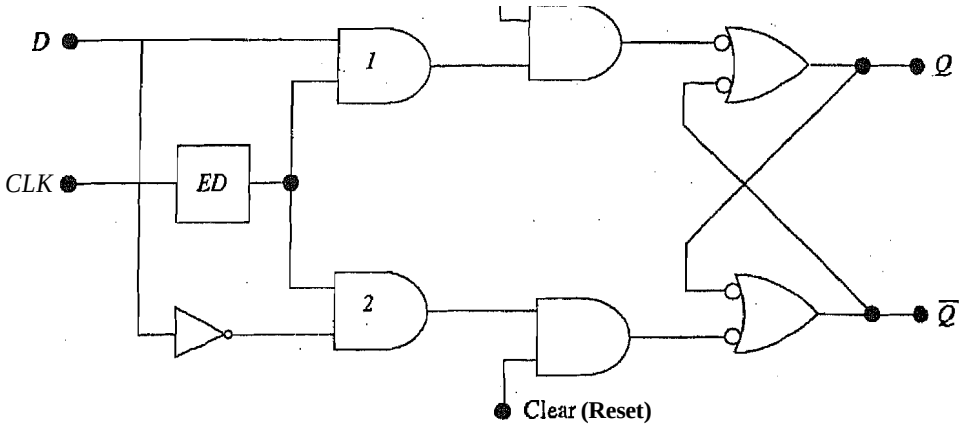


Fig. 11.59: Edge triggered D flipflop with preset and clear.

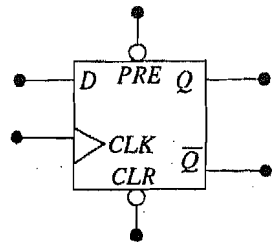


Fig. 11.60: Symbol of edge triggered D flipflop with preset and clear.

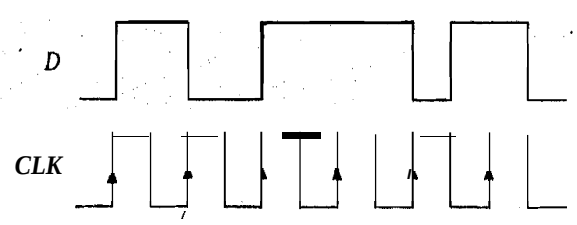
output of pulse steering circuit to pass through. However, if we want to set the flipflop irrespective of the value of D input, then' give a 0 to PRESET which will set the flipflop. Similarly, by giving a 0 to clear will directly reset the flipflop, The symbol for D flipflop with PRESET and CLEAR is shown in Fig. 11.60 and its truth table, is given in Table 11.22.

Table 11.22: Truth table for clocked D flipflop with preset and clear.

Preset	Clear	CLK	D	Q
0	0	X	X	*Race
0	1	X	X	1
1	0	X	X	0
1	1	0	X	NC
1	1	1	X	NC
1	1	↑	X	NC
1	1	↑	0	0
1	1	↑	1	1

SAQ 12

The D input to a positive edge triggered D flipflop is as shown in Fig. 11.61. Trace the Q output.



Fig, 11.61:

11.4.4 Clocked JK Flipflop

In the next unit, we show you how to build a counter, a circuit that counts the number of positive or negative clock edges driving its clock input. When it comes to circuits that count, JK flipflop is the ideal element to use. Therefore before ending this unit we will study about JK flipflop.

The circuit for an edge triggered JK flipflop is shown in Fig. 11.62 and its symbol is shown in Fig. 11.63. The working of JK flipflop is same as that of RS flipflop

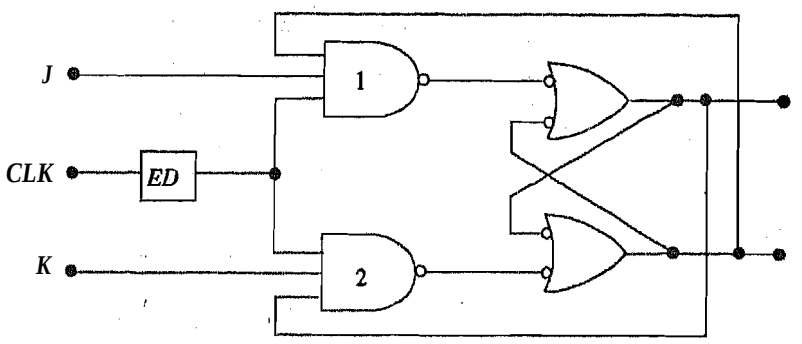


Fig. 11.62: Circuit for edge triggered JK flipflop.

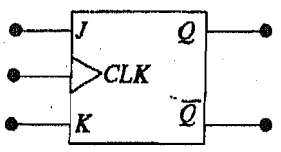


Fig. 11.63: Symbol of edge triggered JK flipflop.

except that race condition is not there. That is, there is no ambiguous result. The outputs Q and $\bar{Q}$ of the NAND latch are fedback to NAND-2 and NAND-1 respectively of the pulse steering circuit which gives toggle operation. With J = K = 1, assume that Q is 0. When clock transition arrives. With Q = 0 and Q = 1, NAND-1 will steer PGT to set the NAND latch to give Q = 1. If we assume Q = 1 when PGT of the clock appears, NAND-2 will steer PGT to clear the NAND latch to produce Q = 0. Thus Q always ends up in opposite state. This is known as the toggle mode of operation. If both J and K are left to a state of 1, the flipflop will change state for each clock transition. The Q output equal to Q_0 means that the new value of Q will be inverse of the value it had prior to the PGT. The truth table of this flipflop is given in Table; 11.23. Fig. 11.64 shows J and K inputs and the corresponding Q output.

Table 11.23: Truth table for a positive edge triggered JK flipflop.

J	K	CLK	Q
0	0	↑	Q_0 (NO change)
1	0	↑	1
0	1	↑	0
1	1	↑	Q_0 (Toggle)

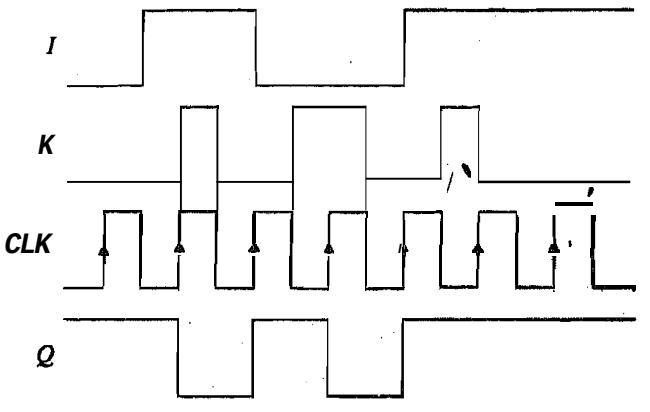


Fig. 11.64: Inputs and output of JK flipflop.

The symbol for edge triggered JK flipflop which is activated by a NGT of the clock is shown in Fig. 11.65 and its truth table is given in Table 11.24.

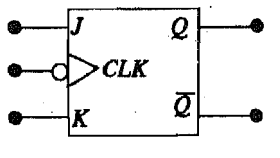


Fig. 11.65: Symbol of edge triggered JK flipflop activated by a NGT.

Table 11.24: Truth table for a negative edge triggered JK flipflop.

J	K	CLK	Q
0	0	↑	Q_0 (No change)
1	0	↑	1
0	1	↑	0
1	1	↑	Q_0 (Toggle)

SAQ 13

The J and K inputs to a JK flipflop are as shown in Fig. 11.66. If the initial value of Q output is 0, trace the Q output.

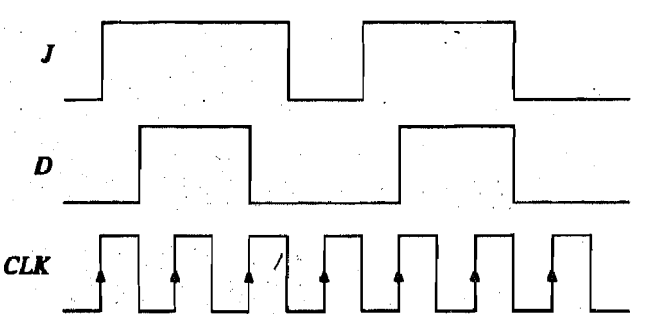


Fig. 11.66:

11.5 SUMMARY

- There are three basic logic gates — AND, OR and NOT. The output of an **AND** gate is 1 only when all the inputs are 1. The output of an OR gate is 0 only when all the inputs are 0. The output of a NOT gate is complement of the input.
- The **AND** and NOT gates are combined to get the NAND gate and OR and NOT gates are combined to get NOR gate. The NAND and NOR gates are known as building blocks in digital circuitry because AND, OR and NOT gates can be obtained using NAND and NOR gates only.
- **All** logic gates and circuits work in binary mode, that is the inputs and outputs can have values either 1 or 0. Therefore, the boolean algebra is used to describe their input-output relationships, the basic boolean rules or theorems are obtained from the truth tables of three basic gales.
- **A** digital circuit can be expressed as a boolean expression and likewise a logic circuit can be obtained from a boolean expression. **A** boolean expression can be simplified which gives us a simplified digital circuit. In all applications, first a boolean expression is simplified to give a simpler circuit.
- **A** boolean expression can also be obtained from a truth table. And a truth table can be obtained from a boolean expres'sion without reference to its logic circuit. The boolean expression is written in the Sum-of-the-Product (SOP) form which is simplified to get the Minimum-Sum-of-the-Product (MSP) form. The **MSP** expression is used to write the final digital circuit.
- Exclusive — OR and exclusive — NOR gates are obtained by the combinations of three basic gates. The output of the XOR gate is 0 if both the inputs are same and is 1 if both the inputs are different. The output of the XNOR gate is 1 if both the inputs are same and is 0 if both the inputs are different.
- **A** half adder adds two bits binary numbers while a full adder adds three bits. The half and full adders are combined to add two multi-bit binary numbers.
- The combinational logic circuits do no have memory, that is output of such circuits do not depend on the previous occurrence of an event, The input-output relationship of these circuits is precisely defined by its truth table.
- The RS flipflop is the basic element which has memory, that is its output depends on the previous occurrences of an event. The input of a RS flipflop can also be triggered by a clock by using a pulse steering circuit. The other flipflops are D and JK flipflops. The output of the **D** flipflop follows the input. The race condition of RS flipflop is avoided in JK flipflops.
- The RS, D and JK flipflop can be triggered by a positive going transition (PGT) or a negative going transition (NGT). These flipflops are used as' memory devices.

11.6 TERMINAL QUESTIONS

1. Simplify the expression $Y = A\bar{B}D + A\bar{B}\bar{D}$.
2. Simplify the expression $Y = BCD + A\bar{B}CD$ and find its MSP form.
3. Simplify the expression $Y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}BCD$.
4. Simplify the expression $Y = \overline{(A + BC) \cdot (D + FG)}$
5. Write boolean expression for the truth table given in Table 11.25.

Table 11.25:

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

6. Write boolean expression for the truth table given in Table 11.26.

Table 11.26:

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

7. Write boolean expression for the truth table given in Table 11.27.

Table 11.27:

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

8. Write boolean expression for the truth table given in table 11.28.

Table 11.28:

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

9. Write the truth table for the expression obtained in question No. 2 above.
10. Write the truth table for the expression obtained in question No. 3 above.
11. Write the truth table for the expression obtained in question No. 5 above.
12. Write the truth table for the expression obtained in question No. 8 above.
13. Draw a digital circuit for a 5-bit binary adder.
14. Design a digital circuit for $Y = \overline{A}C + A\overline{D}$.
15. Design a digital circuit for expression of question No. 14 using NAND gates only.

15.7 SOLUTIONS AND ANSWERS

Fig. 11.67:

2.

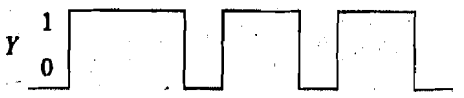


Fig. 11.68:

3.

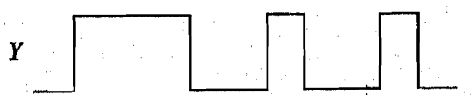


Fig. 11.69:

4.

Fig. 11.70:

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

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

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

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

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

$= AB + A\bar{C}.$

6. Using the reasoning method, Y = 1 when either or all of **AB. BC, and** CA/are 1. Thus we get the truth table as follows:

Table 11.29:

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

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

8.

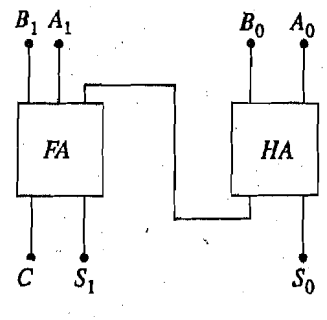


Fig. 11.71: A 2-bit binary adder.

9. **Digital circuit for Y = A + BC is as shown in Fig. 11.72.**

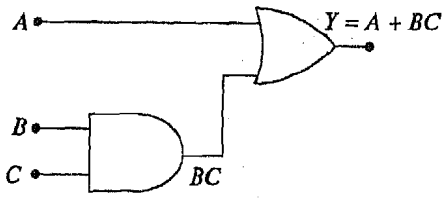


Fig. 11.72:

Now replace OR and AND gates by their NAND equivalents as shown in Fig. 11.73.

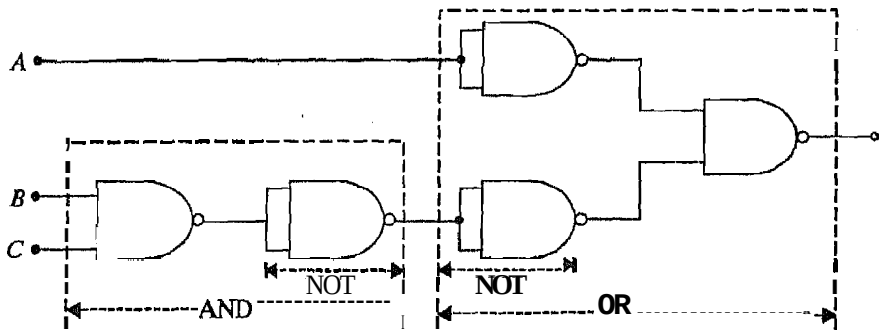
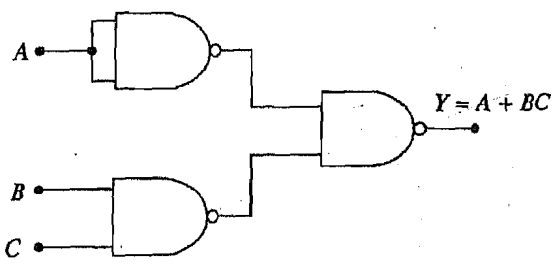


Fig. 11.73:

Removing the combination of a NOT gate followed by a NOT gate, we get the circuit as shown in Fig. 11.74.



Pig. 11.74:

Alternatively, simplify the expression using DeMorgan's theorem as follows:

$$Y = \overline{\overline{A + BC}}$$

$$= \overline{A \cdot B\,C}$$

This equation gives the circuit already obtained in Fig. 11.74.

10.

Ng. 11.75:

11.

Fig. 11.76:

12.

Fig. 11.77:

13.

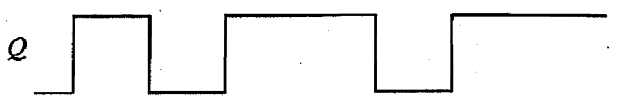


Fig. 11.78:

TQs

1. $Y = A\overline{B}D + A\overline{B}\overline{D}$
   $= A\overline{B}(D + \overline{D})$
   $= A\overline{B} \cdot 1$
   $= A\overline{B}.$

2. $Y = BCD + A\overline{B}CD$
   $= CD(B + A\overline{B})$
   $= CD(B + A)$
   $= CDB + CDA.$

3. $Y = \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}\overline{D}$
   $= \overline{A}\overline{B}\overline{D}(C + \overline{C})$
   $= \overline{A}\overline{B}\overline{D}.$

4. $Y = \overline{(A + BC) \cdot (D + FG)}$
   $= \overline{A} + \overline{BC} + \overline{D + FG}$
   $= \overline{A} \cdot \overline{BC} + \overline{D} \cdot \overline{FG}$
   $= A.(\overline{B} + \overline{C}) + \overline{D} \cdot (\overline{F} + \overline{G})$
   $= A\overline{B} + A\overline{C} + \overline{D}\overline{F} + \overline{D}\overline{G}.$

5. $Y = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + ABC$
   $= \overline{A}\overline{C}(\overline{B} + B) + AC(\overline{B} + B)$
   $= \overline{A}\overline{C} + AC.$

6. $Y = \overline{A}BC + A\overline{B}\overline{C}.$

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

8. $Y = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + \overline{A}B\overline{C} + \overline{A}BC$
   $= \overline{A}\overline{B}(\overline{C} + C) + \overline{A}B(\overline{C} + C)$
   $= \overline{A}\overline{B} + \overline{A}B$
   $= \overline{A}(\overline{B} + B)$
   $= \overline{A}.$

9.

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

10.

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

11.

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

12.

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

13.

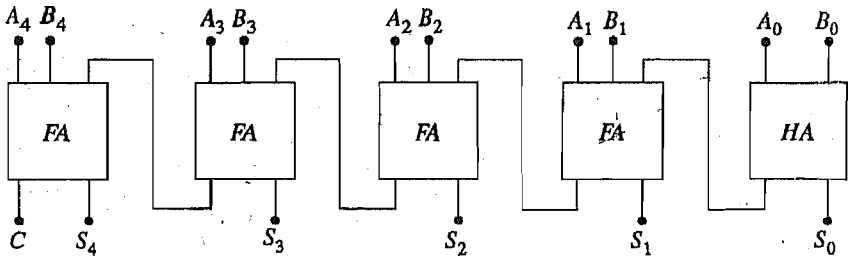


Fig. 11.79: A 5-bit binary adder.

14.

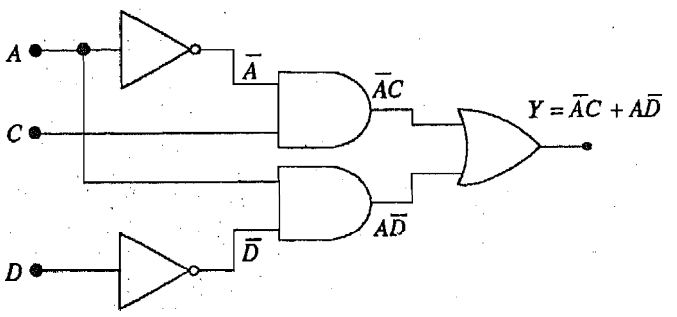


Fig. 11.80; Digital Circuit for Y = $\bar{A}$C + A$\bar{D}$.

15.

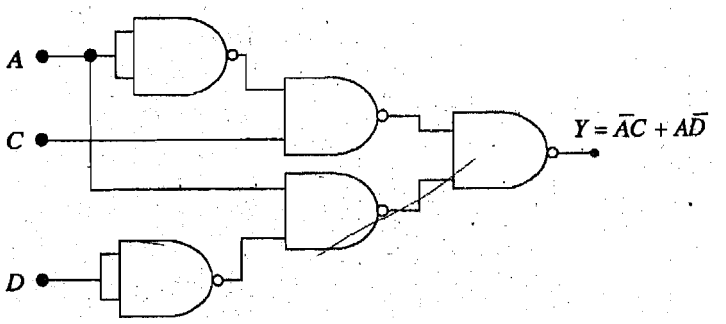


Fig. 11.81: Digital circuit for Y = $\bar{A}$C + A$\bar{D}$ using NAND gates only.