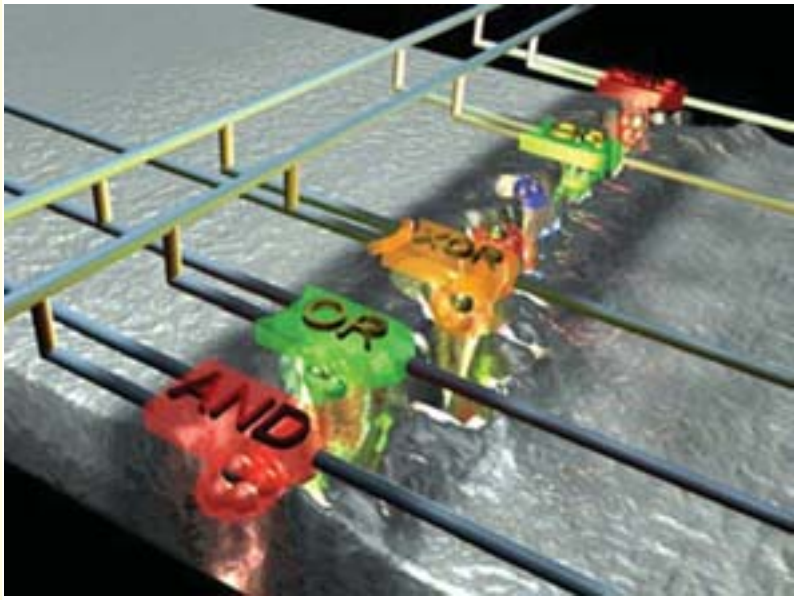
CONTENTS

CHAPTER 70

LOGIC GATES

Learning Objectives

- Definition of a Logic Gate
- Positive and Negative Logic
- The OR Gate
- Equivalent Relay Circuit of an OR Gate
- Diode OR Gate
- Transistor OR Gate
- OR Gate Symbolizes Logic Addition
- Three Input OR Gate
- Exclusive OR Gate
- The AND Gate
- Equivalent Relay Circuit of an AND Gate
- Diode AND Gate
- Transistor AND Circuit
- AND Gate Symbolizes Logic Multiplication
- The NOT Gate
- The NOT Operation
- Bubbled Gates
- The NOR Gate
- The NAND Gate
- The XNOR Gate
- Logic Gate at Glance
- Digital Signals Applied to Logic Gates
- Sequential Logic Circuits
- Adders and Subtractors
- Half Adder
- Full Adder
- Parallel Binary Adder
- Half Subtractor
- Full Subtractor



A logic gate is a circuit that has one or more input signals but only one output signal. All logic gates can be analysed by constructing a truth table

CONTENTS

70.1. Definition of a Logic Gate

A logic gate is an electronic circuit **which makes logic decisions**. It has one output and one or more inputs. The output signal appears only for certain combinations of input signals. Logic gates are the basic building blocks from which most of the digital systems are built up. They implement the hardware logic function based on the logical algebra developed by George Boole which is called Boolean algebra in his honour. A unique characteristic of the Boolean algebra is that variables used in it **can assume only one of the two values** *i.e.* either 0 or 1. Hence, every variable is either a 0 or a 1.

These gates are available today in the form of various IC families. The most popular families are: transistor-transistor logic (*TTL*), emitter-coupled logic (*ECL*), metal-oxide-semiconductor (*MOS*) and complementary metal-oxide-semiconductor (*CMOS*).

In this chapter, we will consider the *OR*, *AND*, *NOT*, *NOR*, *NAND*, exclusive *OR* (*XOR*) and exclusive *NOR* (*XNOR*) gates along with their truth tables.

70.2. Positive and Negative Logic

In computing systems, the number symbols 0 and 1 represent two possible states of a circuit or device. It makes no difference if these two states are referred to as *ON* and *OFF*, *CLOSED* and *OPEN*, *HIGH* and *LOW PLUS* and *MINUS* or *TRUE* and *FALSE* depending on the circumstances. Main point is that **they must be symbolized by two opposite conditions**.

In positive logic, a 1 represents

1. an *ON* circuit **2.** a *CLOSED* switch **3.** a *HIGH* voltage

4. a *PLUS* sign **5.** a *TRUE* statement

Consequently, a 0 represents

1. an *OFF* circuit **2.** an *OPEN* switch **3.** a *LOW* voltage **4.** a *MINUS* sign **5.** a *FALSE* statement.

In negative logic, just opposite conditions prevail.

Suppose, a digital system has two voltage levels of 0V and 5V. If we say that symbol 1 stands for 5V and symbol 0 for 0V, then we have positive logic system. If, on other hand, we decide that a 1 should represent 0 V and 0 should represent 5V, then we will get negative logic system.

Main point is that in ***positive logic***, **the more positive** of the two voltage levels represents the 1 while in negative logic, **the more negative** voltage represents the 1. Moreover, it is not essential that a 0 has to be represented by 0V although in some cases the two may coincide. Suppose, in a circuit, the two voltage levels are 2V and 10V. Then for positive logic, the 1 represents 10V and the 0 represents 2V (*i.e.* lesser of the two voltages). If the voltage levels are – 2V and – 8V, then, in positive logic, the 1 represents – 2V and the 0 represents – 8V (*i.e.* lesser of the two voltages).

Unless stated otherwise, we will be using only ***positive logic*** in this chapter.

70.3. The OR Gate

The electronic symbol for a two-input *OR* gate is shown in Fig. 70.1 (*a*) and its equivalent switching circuit in Fig. 70.1 (*b*). The two inputs have been marked as *A* and *B* and the output as *X*. It is worth reminding the reader that as per Boolean algebra, the three variables *A*, *B* and *X* can have only one of the two values *i.e.* either 0 or 1.

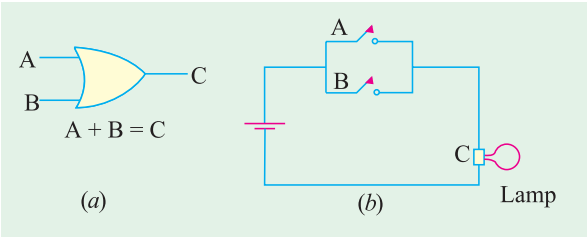


Fig. 70.1

Logic Operation

The *OR* gate has an **output of 1 when either A or B or both are 1**.

In other words, it is an **any-or-all** gate because an output occurs when any or all the inputs are present.

As seen from Fig. 70.1 (*b*), the lamp will light up (logic 1) when either switch *A* or *B* or both are closed.

Obviously, the output would be 0 **if and only if both its inputs are 0**. In terms of the switching conditions, it means that lamp would be *OFF* (logic 0) only when both switches *A* and *B* are *OFF*.

The *OR* gate represents the Boolean equation $A + B = X$

The meaning of this equation is that *X* is true when either *A* is true or *B* is true or both are true. Alternatively, it means that output *X* is 1 when either *A* or *B* or both are 1.

Table 70.1

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

Fig. 70.2

The above logic operation of the *OR* gate can be summarised with the help of the truth table given in Fig. 70.2. A truth table may be defined as a table which ***gives the output state for all possible input combinations.*** The *OR* Table 70.1 gives outputs for all possible *A B* inputs of 00, 01, 10 and 11.

We may interpret the truth table as follows:

When both inputs are 0 (switches are *OPEN*), output *X* is 0 (lamp is *OFF*). When *A* is in logic state 0 (switch *A* is *OPEN*) but *B* is in logic state 1 (switch *B* is *CLOSED*), the output *X* is logic state 1 (lamp is *ON*). Lamp would be also *ON* when *A* is *CLOSED* and *B* is *OPEN*. Of course, lamp would be *ON* when both switched are *CLOSED*. It is so because an *OR* gate is equivalent to a **parallel *circuit in its logic function.***

Another point worth remembering is that the above *OR* gate is called **inclusive** *OR* gate because **it includes the case when both inputs are true**.

70.4. Equivalent Relay Circuit of an OR Gate

In Fig. 70.3, the relay contacts have been wired ***in parallel.*** When +5V is applied to *A*, relay K_1 is energised and pulls *M* down thereby closing the contact. Hence, supply voltage of +5V appears at the output *X*.

Similarly, when +5V are applied to input *B*, K_2 is energised and pulls *N* down thereby bringing *X* in contact with *S*. Of course, when both *A* and *B* are at +5V, *X* is at +5V. Incidentally, when inputs at *A* and *B* are 0, *X* is also 0.

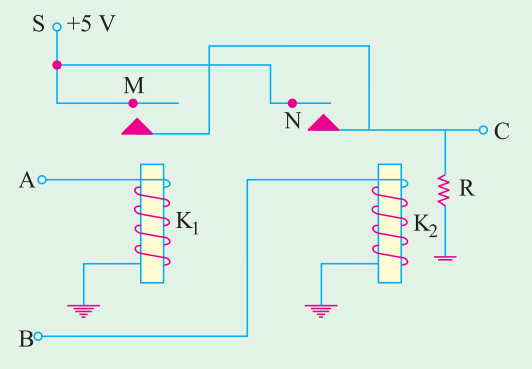


Fig. 70.3

70.5. Diode OR Gate

Fig. 70.4. shows the diode *OR* gate consisting of two ideal diodes D_1 and D_2 connected in parallel across the output *X*.

1. When *A* is at +5V, D_1 is forward-biased and hence conducts. The circuit current flows *via R* dropping 5V across it. In this way, point *X* achieves potential of +5 V.
2. When + 5V is applied to *B*, D_2 conducts causing point *X* to go to +5 V.
3. When both *A* and *B* are +5V, the drop across *R* is 5V because voltages of *A* and *B* ***are in parallel.***

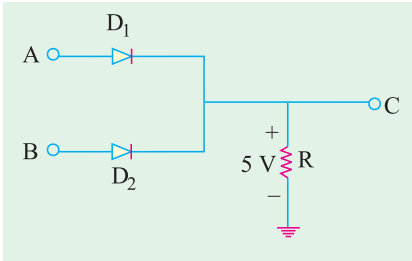


Fig. 70.4

Again, point X is driven to +5 V.

4. Obviously, when there is no voltage either at A or B, output X remains 0.

70.6. Transistor OR Gate

Fig. 70.5 illustrates a possible transistor *OR* gate consisting of three interconnected transistors Q_1, Q_2, and Q_3 supplied from a common supply V_{cc} = +5 V.

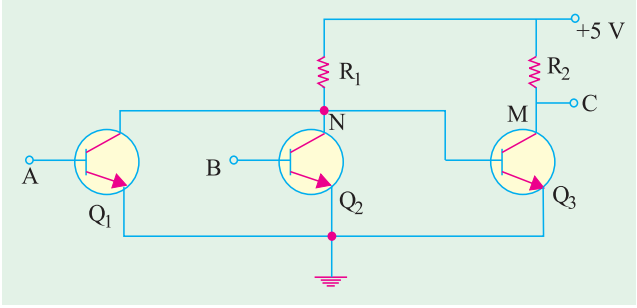


Fig. 70.5

1. When +5 V is applied to A, Q_1 is forward-biased and so it conducts. Assuming that Q_1 is saturated, entire V_{cc} = 5 V drops across R_1 thus causing N to go to ground. This, in turn, cuts off Q_3 thereby causing X to go to V_{cc} *i.e.* + 5V.

2. When +5 V is applied to B, Q_2 conducts thereby driving N to ground *i.e.* 0V. With no forward bias on its base, Q_3 is cut-off thus driving X again to V_{cc} *i.e.* +5 V.

3. If both inputs A and B are grounded, Q_1 and Q_2 are cut-off driving N to +5 V. As a result, Q_3 becomes forward-biased and conducts fully. In that case, entire V_{cc} drops across R_2 driving M and hence X to ground.

70.7. OR Gate Symbolizes Logic Addition

According to Boolean algebra, *OR* gate performs **logical addition**. Its truth table can be written as given below:

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

It must be clearly understood that '+' sign in Boolean algebra **does not stand for the addition** as understood in the ordinary or numerical algebra. In symbolic logic, the '+' sign indicates **OR** operation whose rules are given above. In logic algebra, $A + B = X$ means that if A is true *OR* B is true, then X will be true. ***It does not mean here that sum of A and B equals X.*** The other symbols used for '+' sign are U and V. Hence, the above equation could also be written as $A \cup B = X$ or $A \vee B = X$.

The meaning of the last three logic additions is that output is 1 when either input A or B or both are 1. The first addition implies that output is 0 ***only when both inputs are 0.***

The meaning of the '+' sign often becomes clear from the context as shown below:

$1 + 1 = 2$ — decimal addition
$1 + 1 = 10$ — binary addition
$1 + 1 = 1$ — *OR* addition

We can put the above *OR* laws in more general terms

$$A + 1 = 1$$
$$A + 0 = A$$
$$A + A = A \quad \text{— not } 2A$$

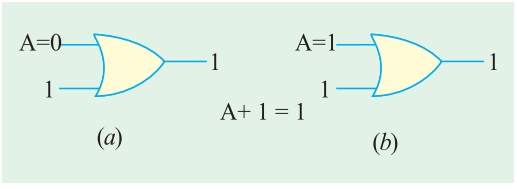


Fig. 70.6

(*i*) A + 1 = 1

As we know, A can have two values: 0 or 1. When A is 0, then we have 0 + 1 = 1 as shown in Fig. 70.6 (*a*).

When A = 1, then the above expression becomes : 1 + 1 = 1 as shown in Fig. 70.6 (*b*), Hence, we find that **irrespective of the value of A.**

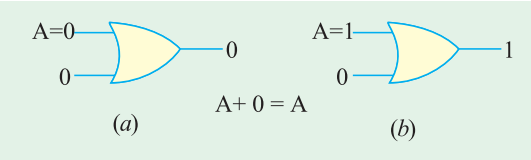


Fig. 70.7

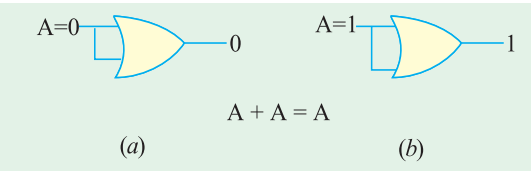


Fig. 70.8

$A + 1 = 1$

(*ii*) A + 0 = A

If $A = 0$, then $0 + 0 = 0$ *i.e.* output is 0 which is correct and is shown in Fig. 70. 7 (*a*). The output is what the value of A is.

As shown in Fig. 70.7 (*b*), when $A = 1$, output is 1 because $1 + 0 = 1$. Again, output is what the value of A is.

(*iii*) A + A = A

With A set to 0, the output is 0 because $0 + 0 = 0$ as shown in Fig. 70.8 (*a*).

With A set to 1, the output is 1 because $1 + 1 = 1$ as shown in Fig. 70.8 (*b*). Obviously, the output in both cases is A.

70.8. Three Input OR Gate

The electronic symbol for a 3-input (fan-in of 3) ***inclusive OR*** gate is shown in Fig. 70.9. As is usual in logic algebra, the inputs A, B, C as well as the output X can have only one of the two values *i.e.* 0 or 1.

Truth Table

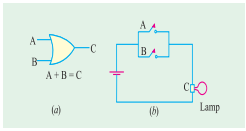


Fig. 70.9

It is shown in Table 70.2. Following points are worth noting:

1. The number of rows in the table is $2^3 = 8$ *i.e.* there are eight ways of combining the three inputs. In general, the number of horizontal rows is 2^n where n is the number of inputs.
2. In first column A, logic values alternate between 0 and 1 every four rows twice.
3. The second input column B alternates between 0 and 1 every two rows four times.
4. The third input column C alternates between 0 and 1 every other row eight times.

The truth table symbolizes the Boolean equation $A + B + C = X$ which means that output X is 1 when ***either A or B or C is 1 or all are 1.*** Alternatively, X is true when either A or B or C is true or all are true.

Table No. 70.2

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

vvhh

Its electronic symbol is shown in Fig. 70.10 (*a*) and its equivalent switching circuit in Fig. 70.10 (*b*).

In this gate, output is 1 if its either input **but not both**, is 1. In other words, it has an output 1 **when its inputs are different**. The output is 0 only when inputs **are the same**.

To put it a bit differently, this logic gate has output 0 **when inputs are either all 0 or all 1.**

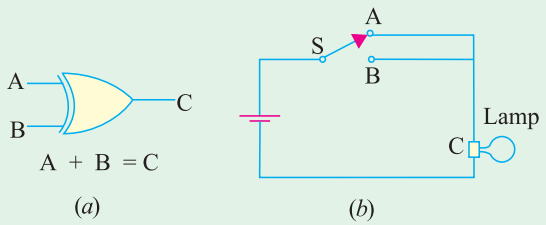


Fig. 70.10

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

Fig. 70.11

This gate works on the Boolean equation $A \oplus B = X$

The circle around plus (+) sign is worth noting.

The circuit is also called an ***inequality comparator*** or detector because it produces an output only **when the two inputs are different**.

Explanation

The ***inclusive*** *OR* gate exemplifies the everyday usage of the word *OR* which stands for one or the other or both. Take the following statement:

To qualify for a competition, you might have to subscribe to a magaine OR belong to a club.

Obviously, there is no bar on your doing both. But now take **exclusive** statement:

You can be rich OR you can be poor.

Obviously, you cannot be both at the same time.

The change-over switching ***circuit*** of Fig. 70.10 (*b*) simulates the ***exclusive OR (XOR)*** gate. Switch positions *A* and *B* will individually light up the lamp but a combination of *A* and *B* is not possible.

The truth table for a 2-input *XOR* gate is given in Table No.70.3. It is instructive to compare it with that for an *inclusive OR* gate (Table 70.1).

70.10. The AND Gate

The electronic (or logic) symbol for a 2-input *AND* gate is shown in Fig. 70.12 (*a*) and its equivalent switching circuit in Fig. 70.12 (*b*). It is worth reminding the readers once again that the three variables *A*, *B*, *C* can have a **value of either 0 or 1**.

Logic Operation

1. The *AND* gate gives an output only **when all its inputs are present**.
2. The *AND* gate has a 1 output when both *A* **and** *B* are 1. Hence, this gate is an **all-or-nothing** gate whose output occurs only when all its inputs are present.
3. In True/False terminology, the output of an *AND* gate will be **true** only if **all its inputs are true**. Its output would be false if **any *of its inputs is false.***

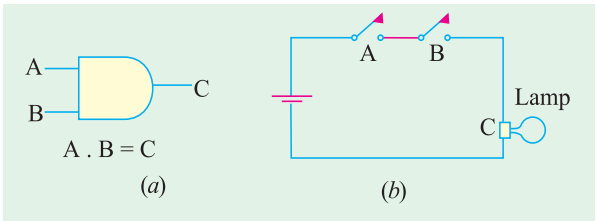


Fig. 70.12

The *AND* gate works on the Boolean algebra

$$A \times B = X \quad \text{or} \quad A \,.\, B = X \quad \text{or} \quad AB = X$$

It is a **logical** multiplication and is different from the **arithmetic** multiplication. Often the sign '×' is replaced by a dot which itself is generally omitted as shown above. The logical meaning of the above equation is that

1. output X is 1 only when both A *and* B are 1.
2. output X is true only when both A *and* B are true.

Table 70.4

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

Fig. 70.13

Table 70.5

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

ABC = X

Fig. 70.14

As seen from Fig. 70.12 (*b*), the lamp would be *ON* when both switches A and B are closed. Even when one switch is open, the lamp would be *OFF*. Obviously, an *AND* gate is equivalent to a **series switching circuit.**

Truth Table Fig. 70.13 shows truth table for a 2-input *AND* gate and Fig. 70.14 gives the same for a 3-input *AND* gate.

As seen, X is at logic 1 only when *all* inputs are at logic 1, not otherwise. The procedure for writing down the first three columns is the same as explained in Art. 70.8 earlier.

70.11. Equivalent Relay Circuit of an AND Gate

The *AND* gate can be physically realized with the help of relay circuit shown in Fig. 70.15. Here, the two relay contacts have been **wired in series**.

When +5 V is applied to both input circuits, relays K_1 and K_2 are energized thereby pulling M and N downwards which brings C in contact with the supply point S. Hence, X goes to +5 V.

It is obvious that energizing only ***one relay*** will not make X go to + 5 V.

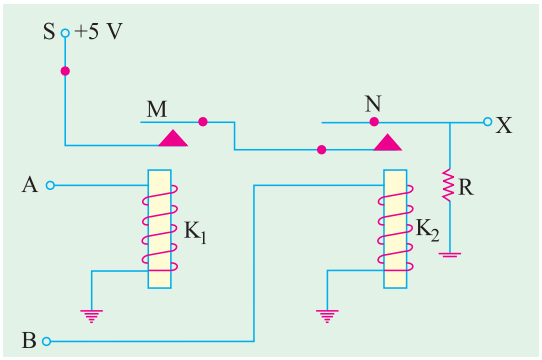


Fig. 70.15

70.12. Diode AND Gate

It is shown in Fig. 70.16. Its logical operation is as under :

1. When A is at 0 V, diode D_1 conducts and the supply voltage of +5 V drops across R. Consequently, point N and hence point X are driven to 0 V. Therefore, the output C is 0.
2. Similarly, when B is at 0 V, D_2 conducts thereby driving N and hence X to ground.
3. Obviously, when both A and B are at 0 V, both diodes conduct and, again, the output X is 0.
4. There is no supply current and hence no drop across R ***only when both A and B are at +5 V.*** Only in that case, the output X goes to supply voltage of +5 V.

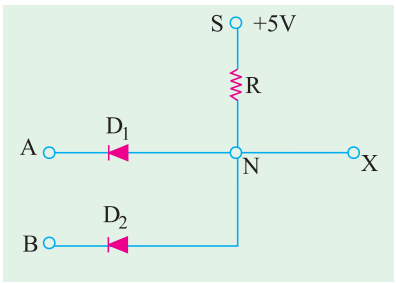


Fig. 70.16

70.13. Transistor AND Circuit

It is shown in Fig. 70.17. When both A and B are at +5 V, the two transistors Q_1 and Q_2 conduct. The current so produced drops the supply voltage of +5 V across R_1 thereby driving point N and hence base of Q_3 to ground or 0V. This cuts off Q_3 so that X goes to supply voltage of +5 V.

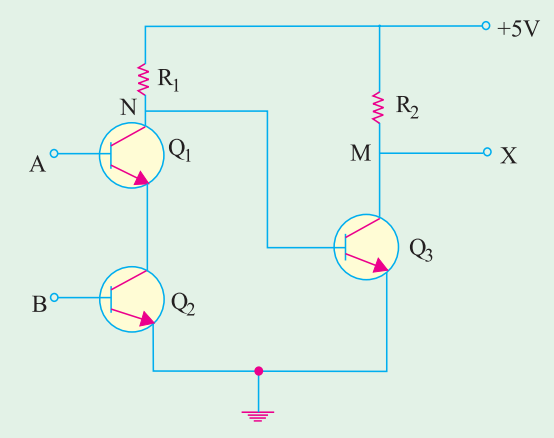


Fig. 70.17

Obivously, there is an output at X only when there is an input at A *and* B.

If either A or B is at 0 V, then Q_1 or Q_2 will be cut off and no drop will take place across R_1. Hence, point N will go to supply voltage of +5V. Consequently, Q_3 will conduct and whole of supply voltage will be dropped across R_2. As a result, point M and hence output X will go to 0 V.

70.14. AND Gate Symbolizes Logic Multiplication

According to Boolean algebra, the *AND* gate performs logical multiplication on its inputs as given below:

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

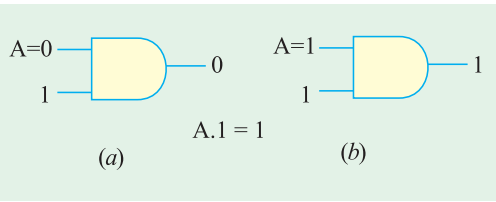


Fig. 70.18

In general, we can put the laws of Boolean multiplication in the following form:

$A.1 = A$ $\quad A.0 = 0$

$AA = A$ — not A^2

The above indentities can be verified by giving values of 0 and 1 to A.

1. A.1 = A When $A = 0$

then $0.1 = 0$ —Fig. 70.18 (*a*)

When $A = 1$

then $1.1 = 1$ —Fig. 70.18 (*b*)

It is seen that in each case, output has the same value as that of A.

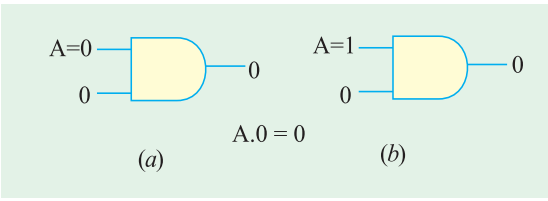


Fig. 70.19

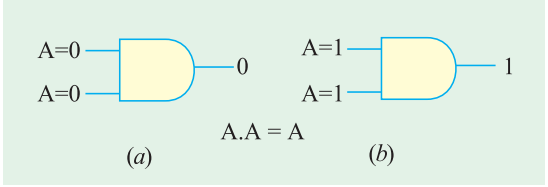


Fig. 70.20

2. A.0 = 0

When $A = 0$

then $0.0 = 0$ —Fig. 70.19 (*a*)

When $A = 1$

then $1.0 = 0$ — Fig. 70.19 (*b*)

It is seen that output is always 0 ***whatever the value of A.***

3. A.A = A

When $A = 0$, then $0.0 = 0$ — Fig 70.20 (*a*)

When $A = 1$, then $1.1 = 1$ — Fig. 70.20 (*b*)

It is seen that output always ***takes on the value of A.***

70.15. The NOT Gate

It is so called because *its output is NOT the* ***same as its input.*** It is also called an ***inverter*** because it inverts the input signal. It has **one** input and **one** output as shown in Fig. 70.21 (*a*). All it does is to invert (or complement) the input as seen from its truth table of Fig. 70.21 (*b*).

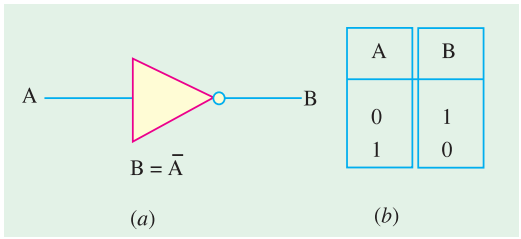


Fig. 70.21

The schematic symbol for inversion is a small circle as shown in Fig. 70.21 (*a*). The logical symbol for inversion or negation or complementation is a bar over the function to indicate the opposite state.

Sometimes, a prime is also used as A'. For example, $\bar{A}$ means not-A. Similarly, $\overline{(A + B)}$ means the complement of $(A + B)$.

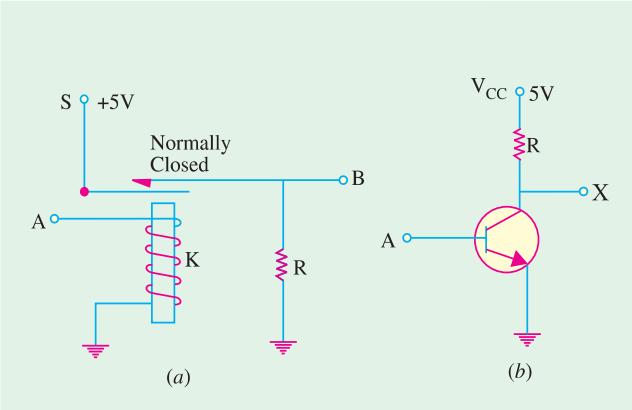


Fig. 70.22

70.16. Equivalent Circuits for a NOT Gate

The relay circuit of Fig. 70.22 (*a*) is a physical realization of the complementation operation of the Boolean algebra. When +5 V is applied to input *A*, *K* is energised and opens the ***normally-closed*** contact thereby driving output *X* to 0 V. Of course, when *A* is at 0V, *X* has the supply voltage of +5 V applied to it because the relay contact is normally closed.

In the equivalent transistor circuit of Fig. 70.22 (*b*) when +5V is applied to *A*, the transistor will be fully turned *ON*, drawing maximum ***collector*** current. Hence, whole of $V_{CC} = 5$ V will drop across *R* thereby sending *X* to 0 V. With 0 V applied at *A*, the transistor will be cut *OFF* and the output *X*, therefore, will go to V_{CC} *i.e.* + 5 V. Obviously, in each case, **output is the opposite of input**.

70.17. The NOT Operation

It is a ***complementation*** operation and its symbol is an **overbar**. It can be defined as under:

As stated earlier, $\bar{0}$ means taking the negation or complement of 0 which is 1.

$$\bar{0} = 1$$

$$\bar{1} = 0$$

It should also be noted that complement of a value can be taken repeatedly. For example,

$$\bar{\bar{1}} = \bar{0} = 1 \quad \text{or} \quad \bar{\bar{0}} = \bar{1} = 0$$

As seen double complementation gives the original value as shown in Fig. 70.23.

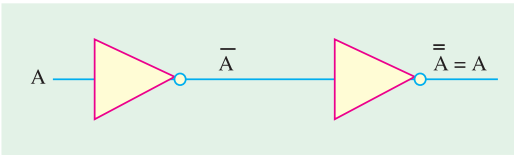


Fig. 70.23

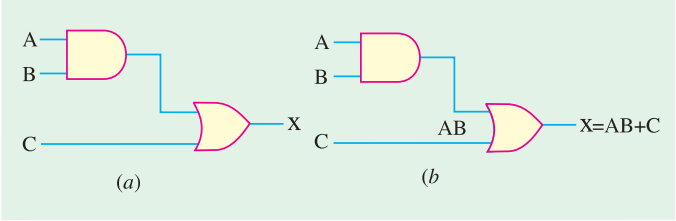


Fig. 70.24

Example 70.1. *Find the Boolean equation for the output X of Fig. 70.24 (a). Evaluate X when* ***(i)*** *A = 0, B = 1, C = 1* ***(ii)*** *A = 1, B = 1, C = 1.* **[Computer Engg., Pune Univ. 1991]**

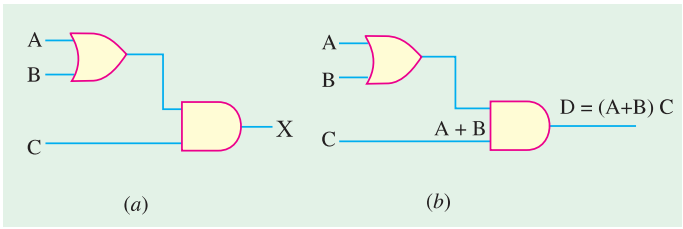


Fig. 70.25

Solution. The output of the *AND* gate is *AB*. It then becomes one of the inputs for the 2 input *OR* gate. When *AB* is ORed with *C*, we get (*AB* + C).

$\therefore \quad X = AB + C$ —Fig. 70.24 (*b*)

(i) $X = 0.1 + 1 = 0 + 1 = 1$ —Art 70.7

(ii) $X = 1.1 + 1 = 1 + 1 = \mathbf{1}$

Example 70.2. *Find the Boolean expression for the output of Fig. 70.25 (a) and evaluate it when* ***(i)*** *A = 0, B = 1, C = 1,* ***(ii)*** *A = 1, B = 1, C = 0.*

Solution. The output of the *OR* gate is (*A* + *B*). Afterwards, it becomes the input of the *AND* gate. When *AND*ed with *C*, it becomes (*A* + *B*).*C*.

$\therefore \quad X = (A + B).C$ —Fig. 70.25 (*b*)

(i) $X = (0 + 1).1 = 1.1 = 1$

(ii) $X = (1 + 1).0 = 1.0 = 0$

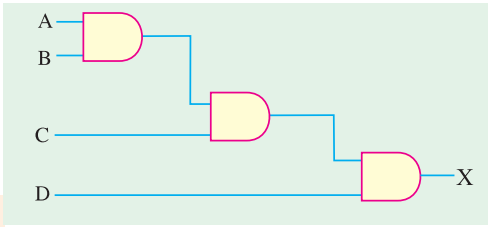


Fig. 70.26

Example 70.3. *Find the Boolean expression for the output of Fig. 70.26 and compute its value when A = B = C = 1 and X = 0.*

(Digital Computations, Punjab Univ. 1990)

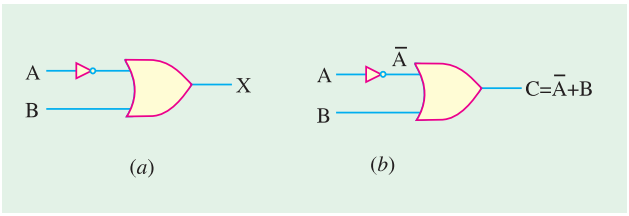


Fig. 70.27

Solution. The circuit is made up of three *AND* gates. Obviously, it is equivalent to a single 4-input *AND* gate *i.e.* an *AND* gate with a fan-in of four.

Output of the first gate is *AB*, that of the second is *ABC* and that of the third is *ABCD*. Hence, final output is *X* = *ABCD*.

Substituting the given values, we get

$X = 1.1.1.0 = 1.1.0 = 1.0 = \mathbf{0}$

Example 70.4. *Find the Boolean expression for the output X of Fig. 70.27 (a) and compute its value when*

(i) $A = 0, B = 1$

(ii) $A = 1, B = 0$

Solution. As seen, one of the inputs to the *OR* gate is inverted *i.e. A* becomes $\bar{A}$ as shown in Fig. 70.27 (*b*). Hence, output is given by $X = \bar{A} + B$

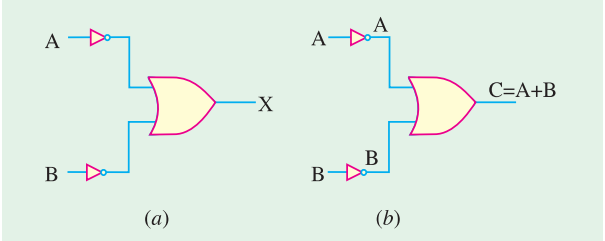


Fig. 70.28

(*i*) $X = \bar{0} + 1 = 1 + 1 = \mathbf{1}$ **(*ii*)** $X = \bar{1} + 0 = 0 + 0 = \mathbf{0}$

Example 70.5. *What is the Boolean expression for the output X of Fig. 70.28 (a) ? Compute the value of X when*

(i) $A = 0, B = 0$ *(ii)* $A = 1, B = 1$

Solution. As seen, in this case, both inputs to the OR gate have been inverted. Hence as shown in Fig. 70. 28 (*b*), the inputs become $\bar{A}$ and $\bar{B}$. Therefore, Boolean expression for the output becomes $X = \bar{A} + \bar{B}$.

(*i*) $X = \bar{0} + \bar{0} = 1 + 1 = \mathbf{1}$

(*ii*) $X = \bar{1} + \bar{1} = 0 + 0 = \mathbf{0}$

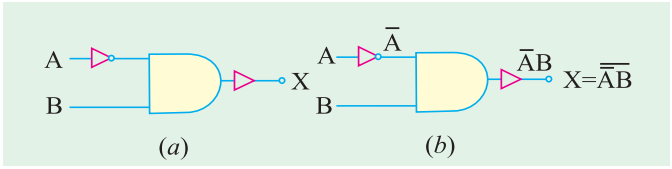


Fig. 70.29

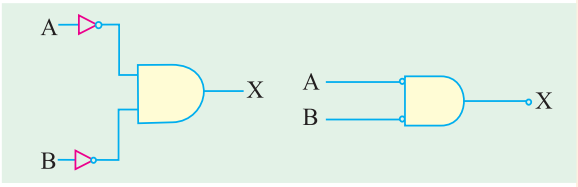


Fig. 70.30

Example 70.6. *Write down the Boolean equation for the output X of Fig. 70.29 (a). Compute its value when*

(*i*) $A = 0, B = 0$

(*ii*) $A = 0, B = 1$

(*iii*) $A = 1, B = 0$

(*iv*) $A = 1, B = 1$

Solution. As seen, inputs to the *AND* gate are A and B [Fig. 70.29 (*a*)]. The output of the *AND* gate is *A*.*B*. However, this output is inverted by the second inverter connected in the output. Hence, final output equation is

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

(*i*) $X = \overline{\bar{0}.0} = \overline{1.0} = \bar{0} = \mathbf{1}$

(*ii*) $X = \overline{\bar{0}.1} = \overline{1.1} = \bar{1} = \mathbf{0}$

(*iii*) $X = \overline{\bar{1}.0} = \overline{0.0} = \bar{0} = \mathbf{1}$

(*iv*) $X = \overline{\bar{1}.1} = \overline{0.1} = \bar{0} = \mathbf{1}$

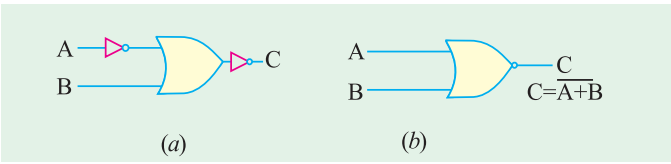


Fig. 70.31

While evaluating expressions of the above type, you must remember the following two points :

1. take the *NOT i.e.* inversion of the individual term first.
2. When a *NOT* or ***inversion*** **is** applied to more than one term (like 1.0), you should work out the *OR* (or *AND*) operation first and then take the *NOT* of the result so obtained.

70.18. Bubbled Gates

A bubbled gate is one whose inputs are *NOT*ed or inverted *i.e.* it is a negated gate. Fig. 70.30 (*a*) shows an *AND* gate whose both input are inverted.

In practice, instead of this logic symbol, the one shown in Fig. 70.30 (*b*) is widely used. As seen, the inverter triangles have been eliminated and the two small circles or bubbles have been moved to the inputs of the gate. Such a gate is called a ***bubbled AND*** gate, the bubbles acting as a reminder of the inversion or complementation that takes place before *AND*ing the inputs.

It would be shown later that a bubbled *AND* gate is equivalent to a *NOR* gate.

Table 70.6

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

Fig. 70.32

Similarly, a bubbled *OR* gate is equivalent to a *AND* gate.

70.19. The NOR Gate

In fact, it is a *NOT-OR* gate. It can be made out of an *OR* gate by connecting an inverter in its output as shown in Fig. 70.31 (*a*).

The output equation is given by

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

A *NOR* function is just the reverse of the *OR* function.

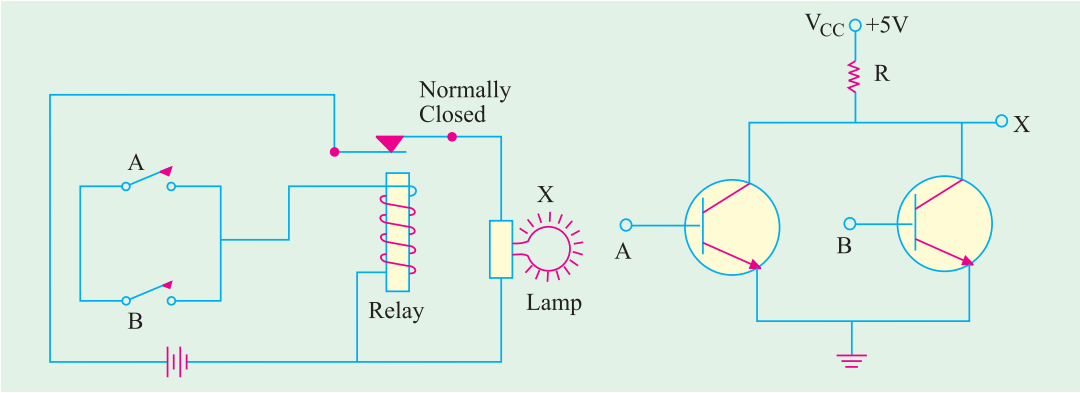


Fig. 70.33 Fig. 70.34

Logic Operation

A *NOR* gate will have an output of 1 **only when all its inputs are 0**. Obviously, if any input is **1**, the output will be 0. Alternatively, in a *NOR* gate, output is **true** only **when all inputs are false**.

The truth table for a 2-input *NOR* gate is shown in Fig. 70.32. It will be observed that the output *X* is just the reverse of that shown in Fig. 70.2.

The equivalent relay circuit for a *NOR* gate is shown in Fig. 70.33.

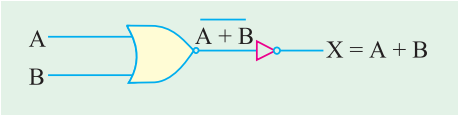


Fig. 70.35

It is seen that the lamp glows under 00 input condition only but not under 01, 10, 11 input conditions.

The transistor equivalent of the *NOR* gate is shown in Fig. 70.34. As seen, output *X* is 1 only when both transistors are cut-off *i.e.* when $A = 0$ and $B = 0$. For any other input condition like 01, 10 and 11, one or both transistors saturate forcing point *X* to go to ground.

70.20. NOR Gate is a Universal Gate

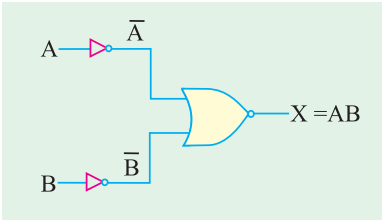


Fig. 70.36

It is interesting to note that a *NOR* gate can be used to realize the basic logic functions : *OR*, *AND* and *NOT*. That is why it is often referred to as a ***universal*** gate. Examples are given below:

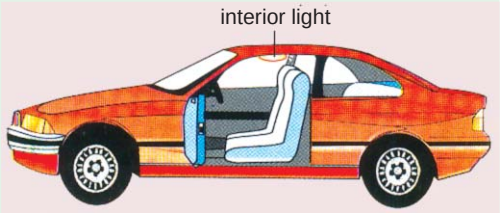
1. As OR Gate

As shown in Fig. 70.35, the output from *NOR* gate is $A + B$. By using another inverter in the output, the final output is inverted and is given by $X = A + B$ which is the logic function of a normal *OR* gate.

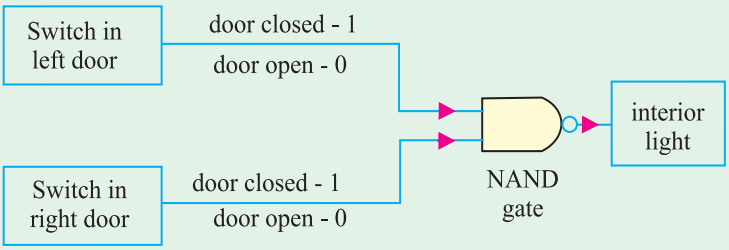
2. As AND Gate

Here, two inverters have been used, one for each input (Fig. 70.36). The inputs have, thus, been **inverted before they are applied** to the *NOR* gate. The output is $\overline{A} + \overline{B}$ which can be proved (with the help of De Morgan's theorem) to be equal to *AB*.

Incidentally, we could have used a bubbled *NOR* gate for the above purpose.



3. As NOT Gate



The NAND gate is used to design an interior lighting system of a car such that the light is switched off only when both doors are shut

The two inputs have been tied together as shown in Fig. 70.37 (*a*). The output is $\overline{A} + \overline{A}$ which can be proved to be equal to *A* with the help of De Morgan's theorem. Instead of the first symbol, the second symbol shown in Fig. 70.37 (*b*) is widely used where only single input has been used.

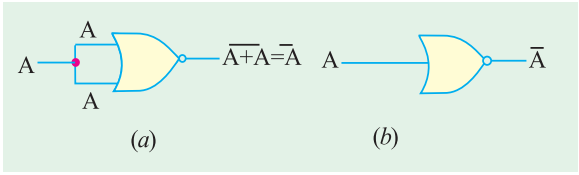


Fig. 70.37

Here is a different way of making *OR* and *AND* gates. Fig. 70.38 (*a*) shows how we can use *NOR* gates to produce an *OR* gate. Similarly, Fig. 70.38 (*b*) shows the formation of an *AND* gate from three *NOR* gates. Knowledge of De Morgan's theorem is needed to understand their logic operation.

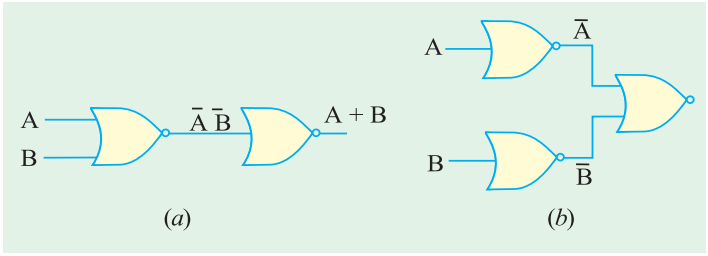


Fig. 70.38

70.21. The NAND Gate

It is, in fact, a *NOT-AND* gate. It can be obtained by connecting a NOT gate in the output of an AND gate as shown in Fig. 70.39. Its output is given by the Boolean equation.

This gate gives an output of 1 if its ***both inputs are* not 1**. In other words, it gives an output 1 if **either A or B or both are 0**.

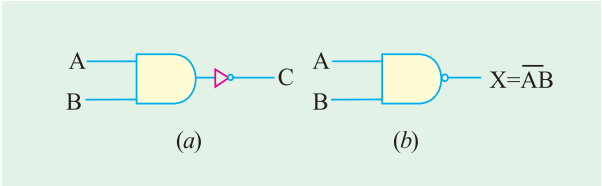


Fig. 70.39

The truth table for a 2-input *NAND* gate is given in Fig. 70.40. It is just the opposite of the truth for *AND* gate. It is so because *NAND* gate performs reverse function of an *AND* gate.

The equivalent relay circuit of a *NAND* gate is shown in Fig. 70.41.

It is seen that lamp glows under input conditions of 00, 01, 10 but not under 11 input condition

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

Fig. 70.40

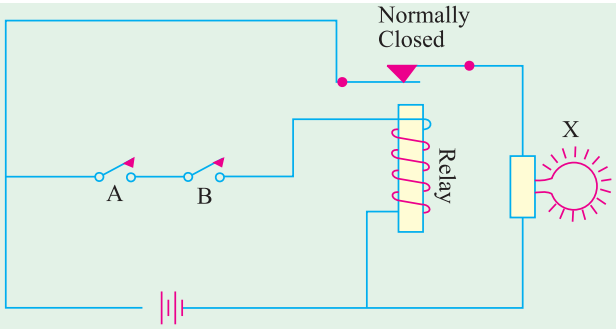


Fig. 70.41

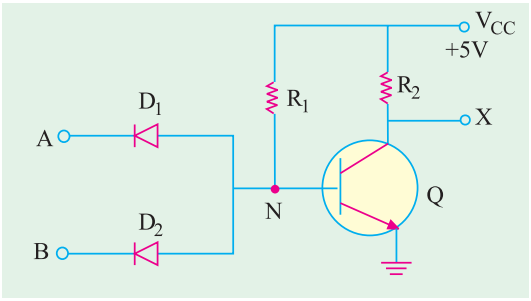


Fig. 70.42

when both switches A and B are ON. The diode-transistor equivalent of a $NAND$ gate is shown in Fig. 70.42.

It is seen that point N would be driven to ground when either D_1 or D_2 or both D_1 and D_2 conduct. It represents input conditions of 10, 01 and 11*. Under such conditions, Q is cut off and hence X goes to V_{CC} meaning logic 1 state. Only time X is 0 is when $A = 1$ and $B = 1$ (*i.e.* input voltages at A and B are at +5V) so that N is +5 V and Q is saturated.

70.22. NAND Gate is a Universal Gate

$NAND$ gate is also called universal gate because it can perform all the three logic functions of an OR gate, AND gate and inverter as shown below.

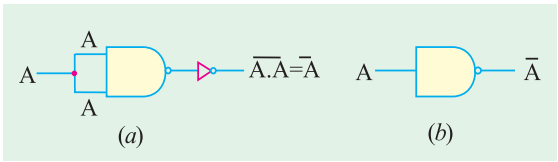


Fig. 70.43

As shown in Fig. 70.43 (*a*), a NOT gate can be made out of a $NAND$ gate by connecting its two inputs together. When a $NAND$ gate is used as a NOT gate, the logic symbol of Fig. 70.43 (*b*) is employed instead.

The use of two $NAND$ gates to produce an AND gate is shown in Fig. 70.44 (*a*).

Similarly, Fig. 70.44 (*b*) shows how OR gate can be made out of three $NAND$ gates. The OR function may not be clear from the figure because we need De Morgan's theorem to prove that $\overline{A}\,\overline{B} = A + B$.

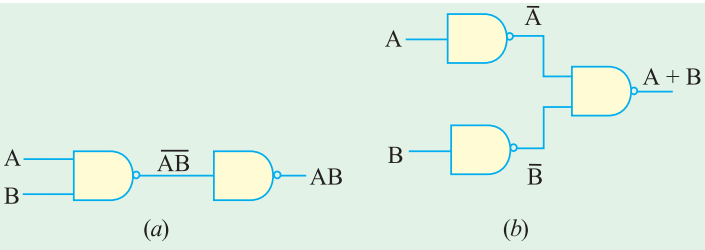
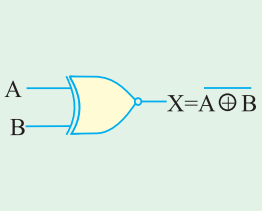


Fig. 70.44

* In this case, $V_A = V_B = 0$ V

70.23. The XNOR Gate

It is known as a not-*XOR* gate *i.e.* $\overline{XOR}$ gate. Its logic symbol and truth table are shown in Fig. 70.45.



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

Fig. 70.45

Its logic function and truth table are ***just the reverse of those for XOR gate*** (Art 70.9).

This gate has an output 1 if **its both inputs are either 0 or 1**. In other words, for getting an output, its both inputs should be ***at the* same** *logic level* of either 0 or 1. Obviously, it produces **no** output if its two inputs are at the **opposite** logic level.

70.24. Logic Gates at a Glance

In Fig. 70.46 is shown the summary of all the 2-output logic gates considered so far along with their truth tables.

Following points should prove helpful when writing these truth tables:

1. In first column *A*, logic values alternate between 0 and 1 every two rows
2. In second column *B*, logic values alternate every other row
3. Column *X* is filled up as per the logic function it performs

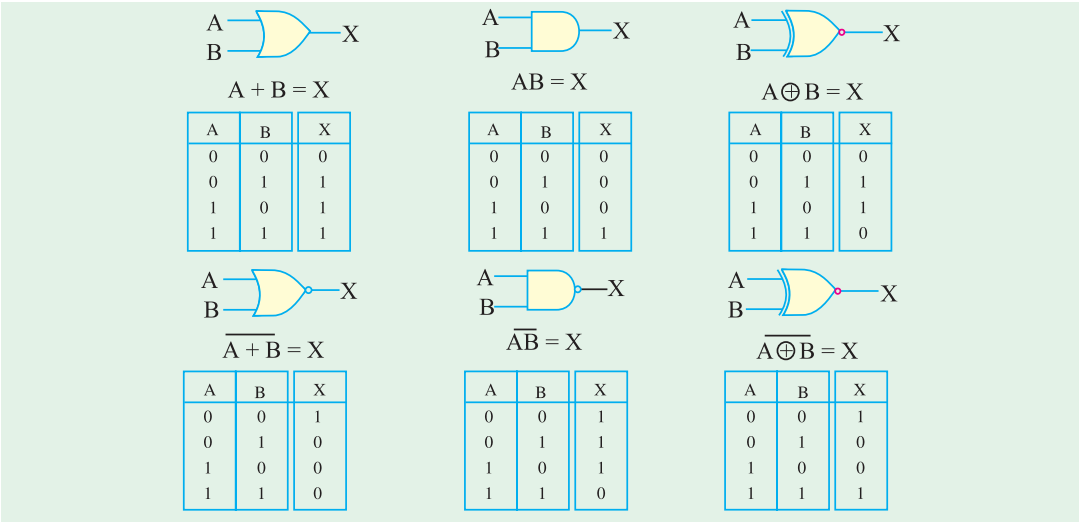


Fig. 70.46

4. Truth tables for *NOR*, *NAND* and *XNOR* (or $\overline{XOR}$) gates are ***just the opposite*** of those for *OR*, *AND* and *XOR* gates.

Example 70.7. *An electrical signal is expressed as 101011. Explain its meaning. If this signal is applied to a NOT gate, what would be the output signal ?*

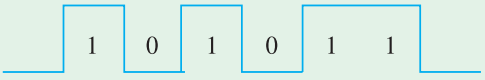


Fig. 70.47

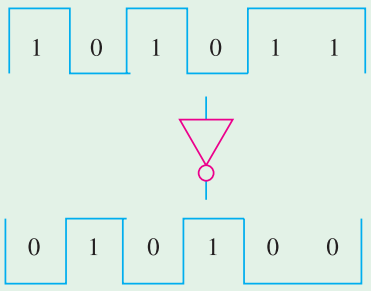


Fig. 70.48

Solution. The signal represents binary number 101011_2. It is electrically represented as a train of pulses. Taking positive logic, 1 will represent high voltage and 0 will represent low (or zero) voltage as shown in Fig. 70.47.

When such a signal is applied to a *NOT* gate, it would be inverted or complemented as shown in Fig. 70.48.

The *NOT* output will represent the binary number 010100_2.

Example 70.8. *Two electrical signals represented by A = 101101 and B = 110101 are applied to a 2-input AND gate. Sketch the output signal and the binary number it represents.*

Solution. The pulse trains corresponding to *A* and *B* are shown in Fig. 70.49.

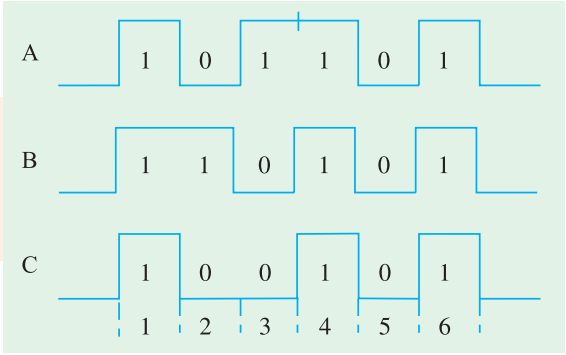


Fig. 70.49

Remember that in an *AND* gate, *C* is 1 only when both *A* and *B* are 1. It is an ***all-or-nothing gate***. The output can be found in different time intervals as under :

1. 1st interval : $1 + 1 = 1$
2. 2nd interval : $0 + 1 = 0$
3. 3rd interval : $1 + 0 = 0$
4. 4th interval : $1 + 1 = 1$
5. 5th interval : $0 + 0 = 0$
6. 6th interval : $1 + 1 = 1$

Hence, ouput of the *AND* gate is 100101_2. It is sketched in Fig. 70.49.

Example 70.9. *Convert the Boolean expression (AB + C) into a logic circuit using different logic gates.* **(Computer Engg. Pune Univ. 1992)**

Solution. In such cases, it is best to start with the ***output and work towards the input***. As seen, *C* has been *OR*ed with *AB*. Hence, the output gate must be a 2-input *OR* gate as shown in Fig. 70.50 (*a*).

Now, term *AB* is an *AND* function. Hence, we need an *AND* gate with inputs *A* and *B*. The complete logic circuit is shown in Fig. 70.50 (*b*).

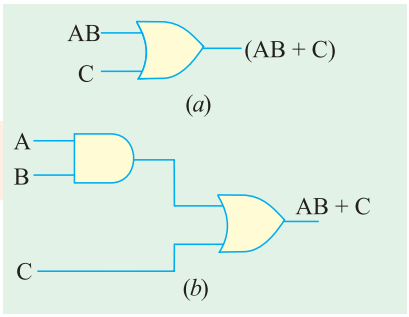


Fig. 70.50

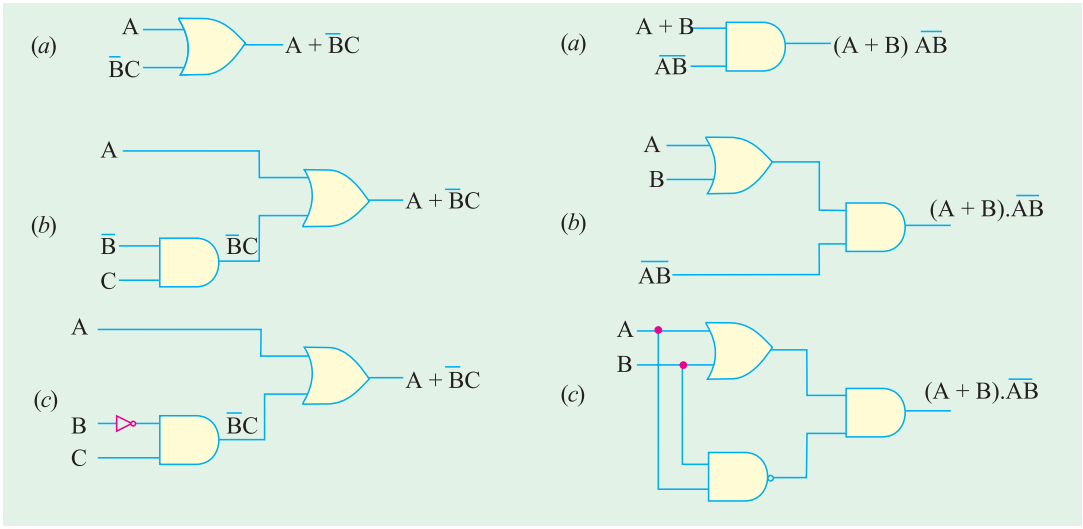
Example 70.10. *Design logic hardware based on the Boolean expression* $(A + \overline{B}C)$.

Solution. We will work from ***output to input.*** It is seen that the last gate is a 2-input *OR* gate with inputs of *A* and $\overline{B}C$. It is shown in Fig. 70.51 (*a*).

Since $\overline{B}$ has been *AND*ed with *C*, it requires an *AND* gate as shown in Fig. 70.51 (*b*). For inversion of *B*, a *NOT* gate has been used as shown in Fig. 70.51 (*c*).

Example 70.11. *Design a logic circuit whose output is given by the Boolean expression* $(A + B).\overline{AB}$. **(Computer Science, Allahabad Univ. 1992)**

Solution. Working from output to input, we find that the output gate has to be a 2-input *AND* gate with inputs of $(A + B)$ and $\overline{AB}$. The first step of the circuit design is shown in Fig. 70.52 (*a*). It is also seen that the input to the entire circuit consists of *A* and *B* only.



Fi.g 70.51

Fig. 70.52

The input of $(A + B)$ has been obtained with the help of an *OR* gate as shown in Fig. 70.52 (*b*).

Finally, a *NAND* gate is connected **in parallel** with the *OR* gate for getting its inputs of *A* and *B* and thereafter for supplying an output of $\overline{AB}$. The complete circuit is shown in Fig. 70.52 (*c*).

70.25. Digital Signals Applied to Logic Gates

A binary digital signal applied to a logic gate is nothing else but the application of a time sequence of 1's and 0's. The response of *AND* and *OR* gates to various periodic digital signals is shown in Fig. 70.53.

Fig. 70.54 shows how two waveforms can be *OR*ed by using two input gates.

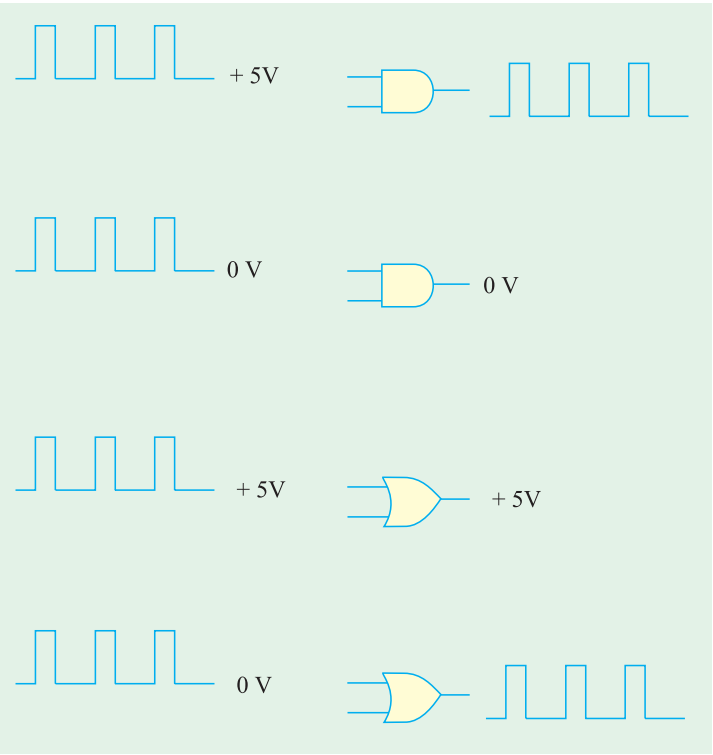


Fig. 70.53

70.26. Applications of Logic Gates

Range of application of the logic gates is very wide but the main headings would include.

1. to build more complex devices like binary counters etc.,
2. for decision making in automatic control of machines and various industrial processes,
3. in calculators and computers,
4. in digital measuring techniques,

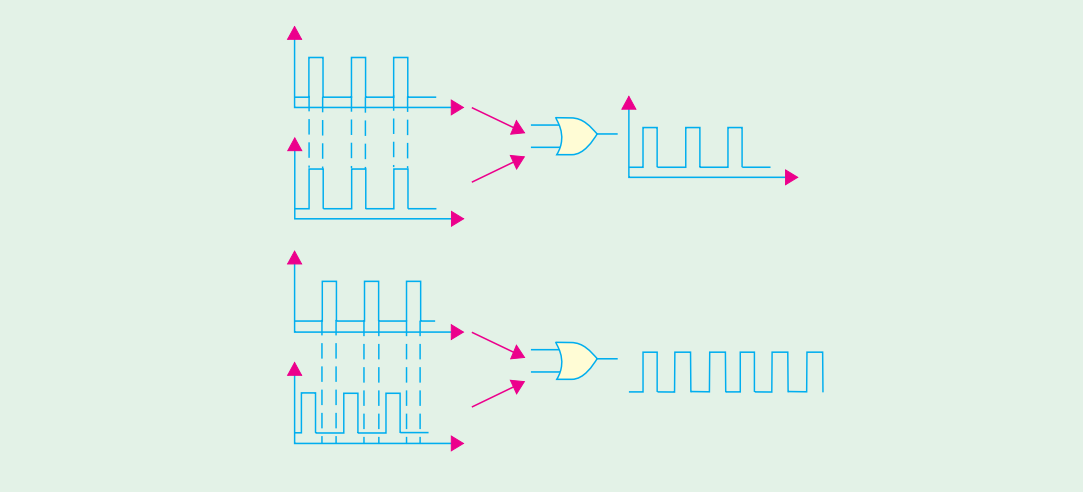


Fig. 70.54

5. in digital processing of communications,
6. in musical instruments, games and domestic appliances etc.

Sometimes it is more convenient to show the digital signals or pulse waveforms without the *x*- and *y*-axes. The examples below will follow this practice.

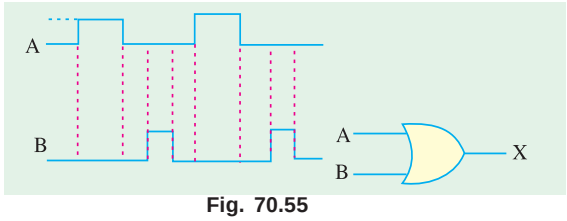


Fig. 70.55

Example 70.12. *Fig. 70.55 shows an OR gate with two input waveforms. What is the resulting output waveform ?*

Solution. Remember that the output of an OR gate is 1 when either or both inputs are 1. Therefore, we can sketch the output wave form, *C* as shown in Fig. 70.56.

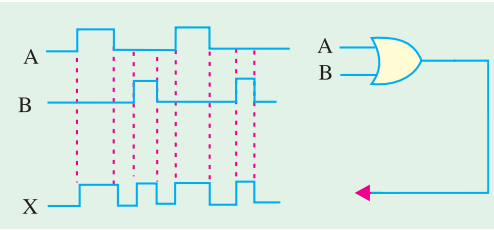


Fig. 70.56

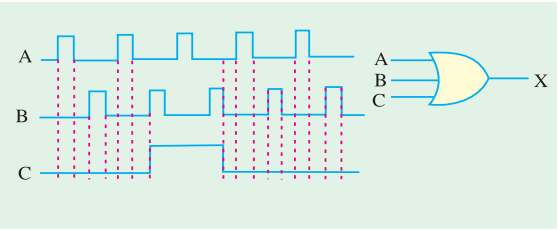


Fig. 70.57

Example 70.13. *For a three-input OR gate shown in Fig. 70.57 determine the output waveform.*

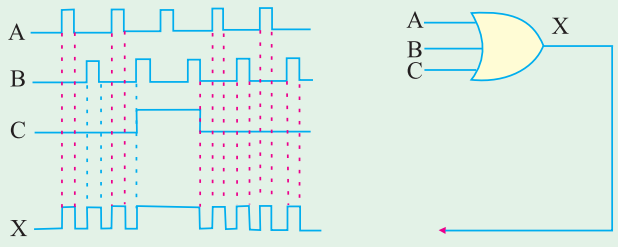


Fig. 70.58

Solution. Remember the output of a three-input *OR* gate is 1 when one or more of the inputs are

1. Therefore, we can sketch the output waveform, *D* as shown in Fig. 70.58.

Example 70.14. *Fig. 70.59 shows a 2-input AND gate with waveforms A and B. Sketch the resulting output waveform.*

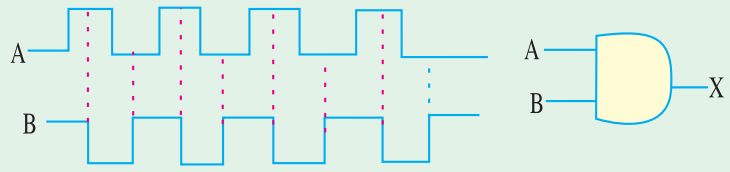


Fig. 70.59

Solution. Remember the *AND* gate produces an output 1 only when all its inputs are present. Thus the output of *AND* gate is 1 when both *A* and *B* are 1. Its output is 0 when any of its inputs is 0. Using this concept, we can sketch the output waveform as shown in Fig. 70.60.

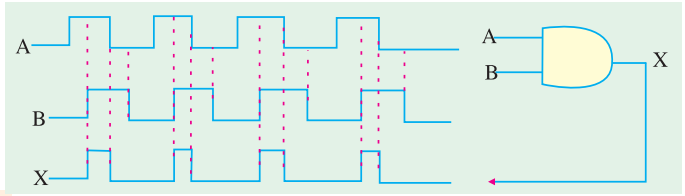


Fig. 70.60

Example 70.15. *For a 3-input AND gate with waveforms A, B and C at its inputs as shown in Fig. 70.61, determine the resulting output waveform.*

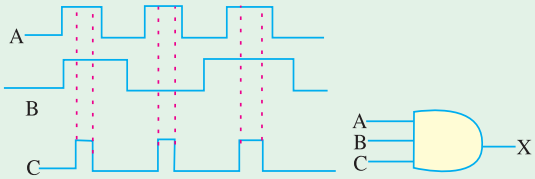


Fig. 70.61

Solution. A 3-input AND gate produces an output 1 only when all the three inputs *A*, *B* and *C* are 1. Its output is 0 when any one of three inputs is 0. Using this concept, the resulting output waveform is as shown in Fig. 70.62.

Example 70.16. *Fig. 70.63 shows the two waveforms applied to the NOR gate inputs. Sketch the resulting output waveform.*

Solution. Remember, a *NOR* gate will have an output of 1 only when all its inputs are 0. Obviously, if any input is 1, the output will be 0. Using this concept, we can sketch the output waveform as shown in Fig. 70.64.

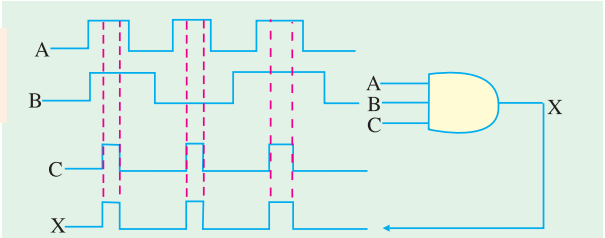


Fig. 70.62

Example 70.17. *Sketch the output waveform for a 3-input NOR gate shown in Fig. 70.65. Showing the proper time relationship to the inputs.*

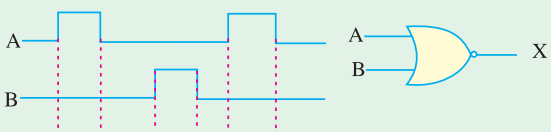


Fig. 70.63

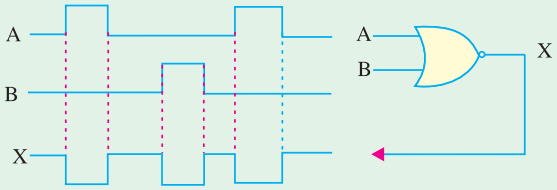


Fig. 70.64

Solution. Remember, a 3-input *NOR* gate will have an output of 1 only when all the three inputs *A*, *B* and *C* are 0. Obviously if any one of the three inputs *A*, *B* and *C* or all are 1, the output will be 0.

Using this concept, we can sketch the output waveform as shown in Fig. 70.66.

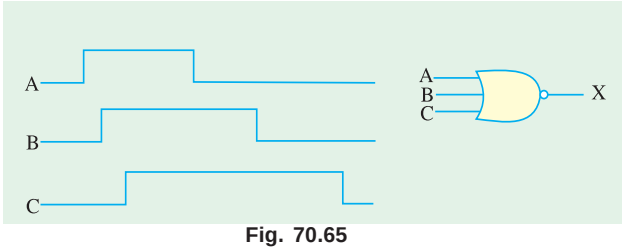


Fig. 70.65

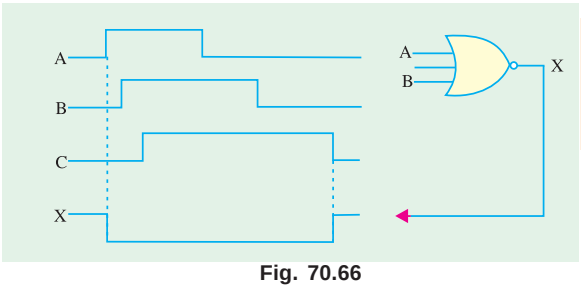


Fig. 70.66

Example 70.18. *Fig. 70.67 shows the two waveforms A and B applied to the NAND gate inputs. Determine the resulting output waveform.*

Solution. Remember, the output of *NAND* gate is 1 if either *A* or *B* or both are 0. Using this concept, we can sketch the output waveform as shown in Fig. 70.68.

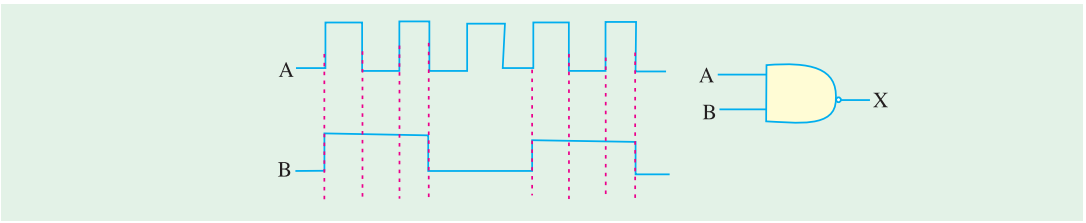


Fig. 70.67

Example 70.19. *Sketch the output waveform for a 3-input NAND gate shown in Fig. 70.69*

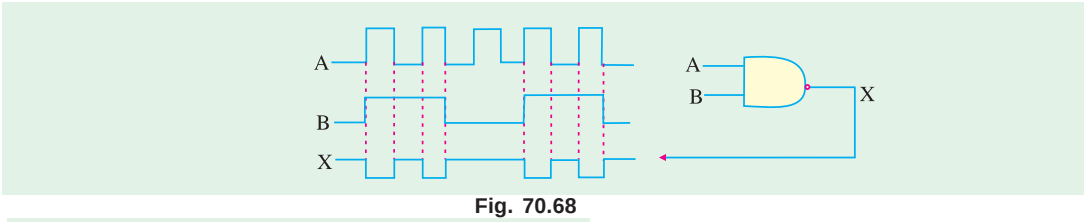


Fig. 70.68

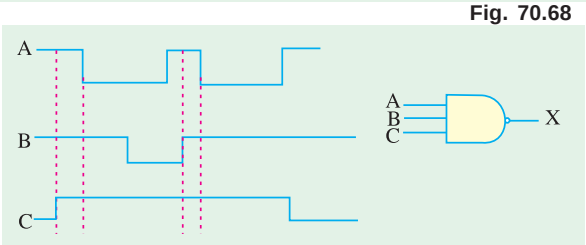


Fig. 70.69

Solution. Remember, the output of a 3-input NAND gate is 1 only if either any one of the inputs A, B and C or all are 0. Using this concept, we can sketch the output waveform as shown in Fig. 70.70.

Example 70.20. *The waveforms A and B are applied as an input to the XOR and XNOR gate as shown in Fig. 70.71. Determine the output waveforms of these logic gates.*

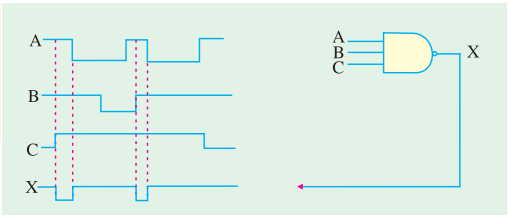


Fig. 70.70

Solution. In exclusive–*OR* (*XOR*) gate, output is 1 if its either input but not both, is 1. In other words, it has an output 1 when its inputs are different. The output is 0 only when inputs are the same. Using this concept, we can sketch the *XOR* output waveform as shown in Fig. 70.72.

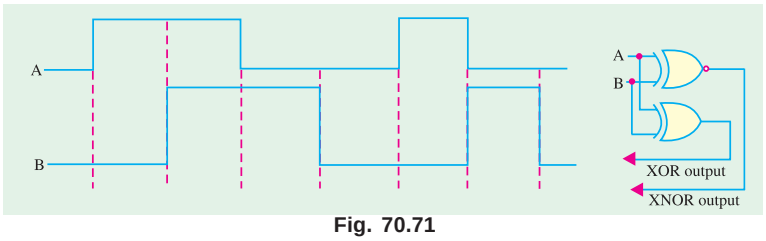


Fig. 70.71

In *XNOR* gate, output is 1 if its both inputs are either 0 or 1. It produces 0 output if its two inputs are at the opposite logic level. Using this concept, we can sketch the *XNOR* output waveform as shown in Fig. 70.72

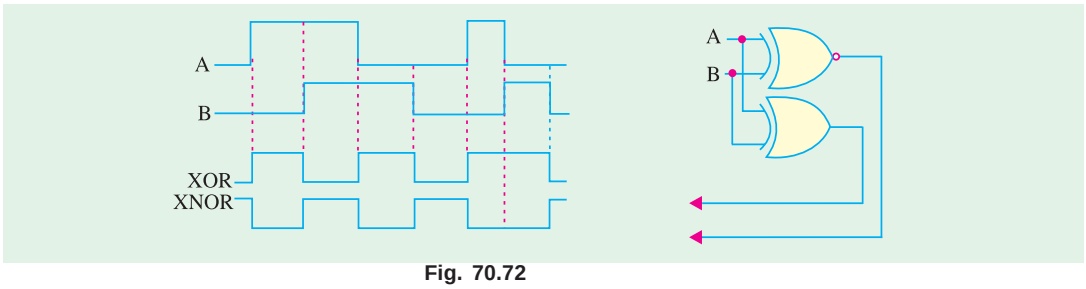


Fig. 70.72

70.27. Combinational Logic Circuit

It is a circuit built from various logic gate combinations. The circuit possesses a set of inputs, a ***memoryless*** logic network to operate on the inputs and a set of outputs as shown in Fig. 70.73 (*a*). The output from a combinational logic circuit depends solely on the ***present*** input values and not on the previous ones. Moreover, output combinational networks are used to make logical decisions and control the operation of different circuits in digital electronic systems. For a given set of input conditions, the output of such a circuit is the same. Consequently, a truth table can fully describe the

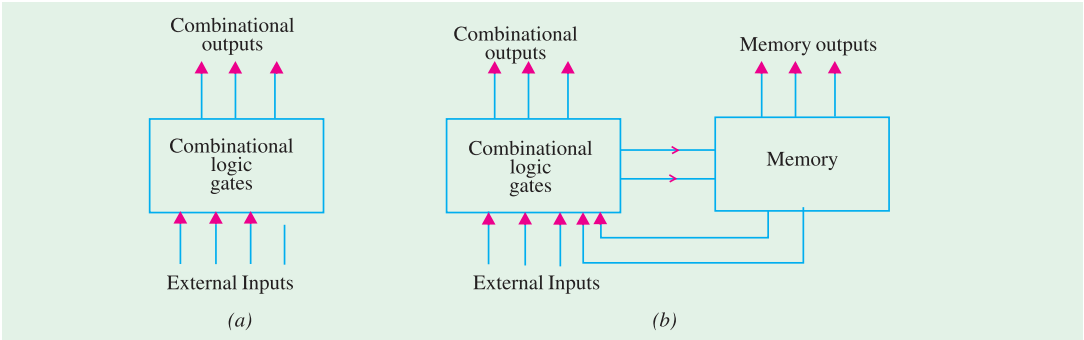


Fig. 70.73

operation of such a circuit. Examples of such a circuit are : decoders, adders, multiplexer and demultiplexers etc.

70.28. Sequential Logic Circuits

Such circuits have inputs, logic network, outputs and a memory as shown in Fig. 70.73 (*b*). Their present output depends not only on their present inputs but also on the previous logic states of the outputs.

Examples of such circuits are a variety of latches and flip-flops. Sequential logic circuits may be either synchronous or asynchronous. The synchronous sequential circuits are built to operate at a clocked rate whereas asynchronous ones are without clocking.

70.29. Adders and Subtractors

The logical gates discussed so far can be used for performing arithmetical functions like addition, subtraction, multiplication and division in electronic calculators and digital instruments. In the central processing unit (*CPU*) of a computer, these arithmetic functions are carried out by the arithmetic and logic unit (*ALU*). The logic functions used generally are *XOR*, *OR* and *AND*. We will consider the following:

1. Half Adder—it is 1-bit adder and carries out binary addition with the help of *XOR* and *AND* gates. It has **two** inputs and **two** outputs.
2. Full Adder—It has **three** inputs and can add three bits at a time. It is made up of ***two half adders and one OR gate.***
   These adders can also perform subtraction by the method of 1's and 2's complements.
3. Half Subtractor—it uses one *XOR* and one *AND* gate.
4. Full Subtractor—it employs two half subtractors and one *OR* gate.

70.30. Half Adder

It can add 2 binary digits **at a time** and produce a 2-bit data *i.e.* sum and carry according to the binary addition rules (Art. 70.10).

Block Diagram

It is shown in Fig. 70.74. As can be seen, it has two inputs for applying the two binary digits to be added. As is well known, binary addition of two bits always produces 2-bit output data *i.e.* one *SUM* and one *CARRY*. For example, (1 + 1) gives a sum of 0 and a carry of 1. Also, (0 + 0) gives the sum 0 and carry 0. That is why the adder has two outputs : one for *SUM* and the other for *CARRY*.

Table 70.7

Input		Output	
A	*B*	*S*	*C*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

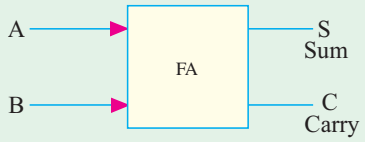


Fig. 70.74

Truth Table 70.7 lists the two columns of input, one of *SUM* and one of *CARRY*. The *SUM* output has the same logic pattern as when *A* is *XOR*ed with *B*. In fact, ***to add it to XOR.*** Also, the *CARRY* output has the same logic pattern as when *A* is *AND*ed with *B*. That is why a half-adder can be formed from a combination of one *XOR* gate and one *AND* gate as shown in Fig. 70.75.

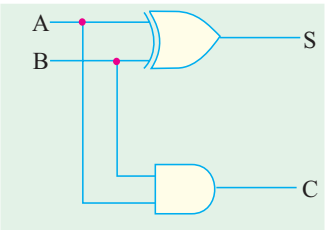
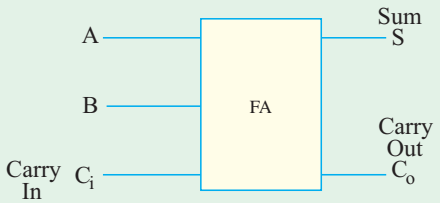


Fig. 70.75

The circuit is called ***half-adder*** because it cannot accept a carry-in from previous additions. For that purpose, we need a 3-*input* adder called full-adder.

Incidentally, the logical equations for the *SUM* and *CARRY* are S = A ⊕ B and $C = A.B$



Input			Output	
A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig. 70.76

70.31. Full Adder

As shown in the block diagram of Fig. 70.76, it has ***three inputs and two outputs.*** It can add 3 digits (or bits) **at a time**. The bits *A* and *B* which are to be added come from the two registers and the third input comes from the carry generated by the previous addition. It produces two outputs; *SUM* and *CARRY-OUT*.

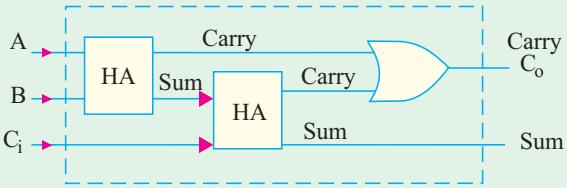


Fig. 70.77

The truth table 70.9 gives all possible input/output relationships for the full adder. *A* and *B* are the inputs from the respective digits of the registers to be added and C_i is the input for any carry generated by the previous stage. The *SUM* output gives binary addition of *A*, *B* and C_i. The other output generates the carry C_0 to be added to the next stage.

The full-adder can be constructed from two half-adders and one *OR* gate (Fig. 70.77).

Working. Let us illustrate, with the help of two examples, how this full adder adds three bits.

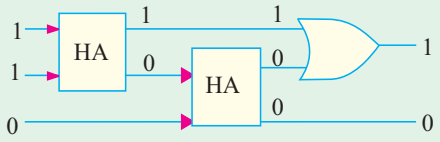


Fig. 70.78

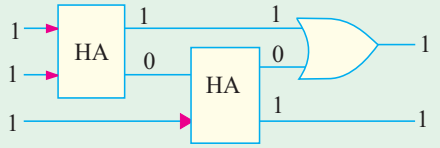


Fig. 70.79

(*i*) A = 1, B = 1, Ci = 0

The full adder with these three inputs is shown in Fig. 70.78. First half adder gives a sum of 0 and a carry of 1. The second *HA* gives a sum of 0 with a carry of 0. The final output is : *SUM* 0, *CARRY* **1.** As we know from the rules of binary addition, $1 + 1 + 0 = 10_2$ (*i.e* decimal 2).

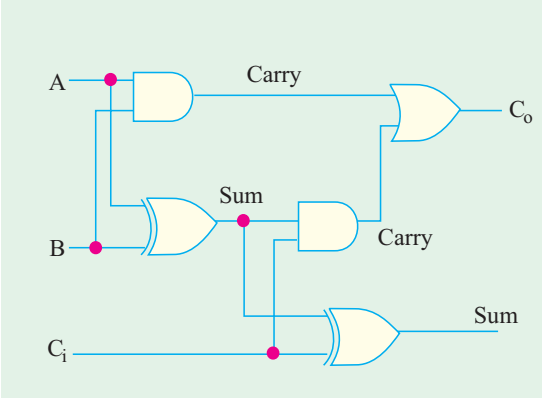


Fig. 70.80

(ii) **A = 1, B =1, C = 1**

As detailed in Fig. 70.79, we get a final *SUM* 1 with a *CARRY* 1. The result conforms to the binary addition : $1 + 1 + 1 = 11_2$ (*i.e.* decimal 3)

Detailed Circuit

Fig. 70.80 shows more circuit details of a full adder. The two half adders have been replaced by their *XOR* and *AND* gates. The final carry is given by the *OR* gate and final sum by the *XOR* gate of the second adder.

70.32. Parallel Binary Adder

For adding two 4-bit numbers, we need 4 full adders *connected in parallel* as shown in Fig. 70.81. The two numbers being added are $A_3\,A_2\,A_1\,A_0$ and $B_3\,B_2\,B_1\,B_0$ and their sum is $S_4\,S_3\,S_2\,S_1\,S_0$.

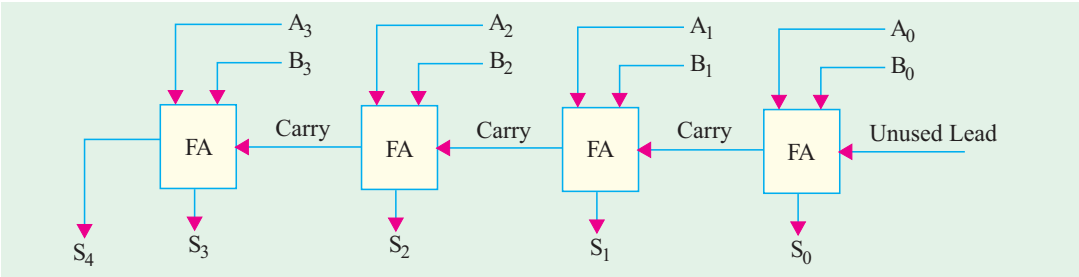


Fig. 70.81

The first adder could be a half adder though we may use a full adder but leave its *CARRY*-IN lead unconnected. As seen, different bits are fed to the four adders from two parallel registers which hold these bits. The final *SUM* appears as a 5-digit display.

Operation

The actual operation may be better understood with the help of the diagram of Fig. 70.82. Suppose, we want to add the following two 4-bit numbers.

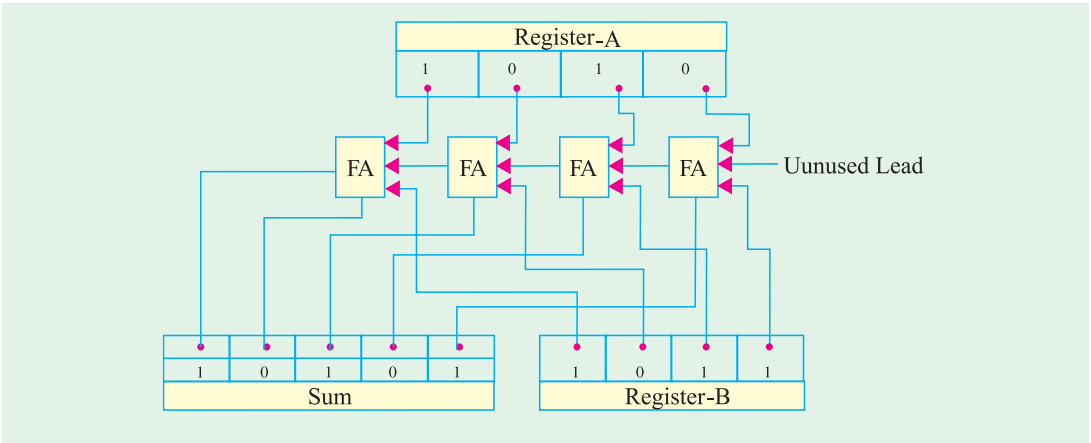


Fig. 70.82

$$\begin{array}{r} 1010 \\ +1011 \\ \hline 10101 \end{array} \qquad \begin{array}{r} 10 \\ +11 \\ \hline 21 \end{array}$$

The first adder performs 0 + 1 binary addition, giving a sum of 1 and a carry of 0. The two bits 0 and 1 are supplied ***simultaneously*** from the two registers A and B. The sum 1 appears on the display panel and carry 0 is passed on to the next full adder.

The next adder adds 1 + 1 +0 carry = sum 1 with a carry 1. The third adder performs 0 + 0 + 1 carry = 1 with carry 0. The fourth adder adds 1 + 1 + 0 carry = sum 0 with carry 1 both of which appear on the display unit. Hence, the final addition of the two numbers appears as 10101.

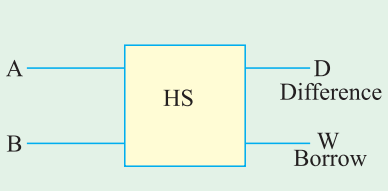


Table 70.10

Input		Output	
A	B	W	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Fig. 70.83

It may be noted that the largest binary numbers that can be added by this parallel adder are 1111 and 1111 which give a sum of 11110_2 (decimal 30). To increase its capacity, more full adders may be connected at the left end of Fig. 70.82. For example, for adding 6-bit numbers, we will have to add two more adders thus making a total of **six**.

70.33. Half Subtractor

It can subtract only two binary digits ***at a time*** and produce an output of a difference and a borrow. As shown in the block diagram of Fig. 70.83, it has two inputs and two outputs.

The operation of a half subtractor is based on the rules of binary subtraction illustrated in the truth Table 70.10 for all possible input/output combinations. The difference output in the fourth column has the same logic pattern as when A is XORed with B (same was the case for SUM in Art. 70.30). Hence, we can use an XOR gate to get the difference of two bits. The borrow output in the third column can be obtained by ANDing $\overline{A}$ with B.

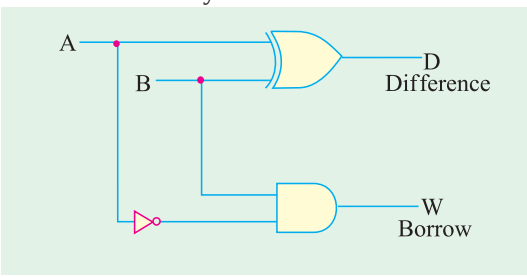


Fig. 70.84

The circuit for a half subtractor is shown in Fig. 70.84.

As mentioned earlier, the logical equations for the difference and borrow are given by

$D = A \oplus B$ and $W = \overline{A}\,B$

70.34. Full Subtractor

As shown in the block diagram of Fig. 70.85, it has three inputs and two outputs. As explained above, half subtractor can handle only 2 bits at a time and can be used for the least significant column of a subtraction problem. A full subtractor can, however, take care of higher-order columns.

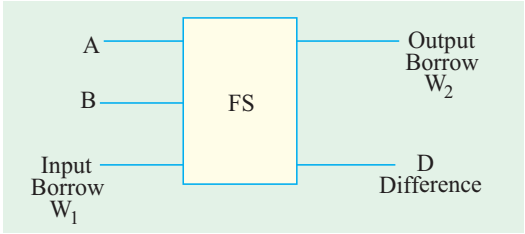


Fig. 70.85

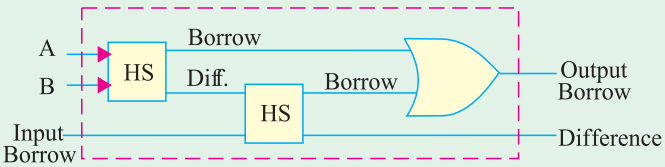


Fig. 70.86

As shown in Fig. 70.86, a full subtractor consists of two half subtractors and one *OR* gate.

It may be remarked here that by cascading 4 full subtractors, we can directly subtract 4-bit numbers *i.e.* we can subtract $B_3\ B_2\ B_1\ B_0$ from $A_3 A_2 A_1 A_0$.

Tutorial Problems No. 70.1

1. Find the Boolean equation for the output of the logic circuit shown in Fig. 70.87. What would be the output if $A = 1$, $B = 0$, $C = 1$, $D = 1$ ***[X = AB + CD; 1]***

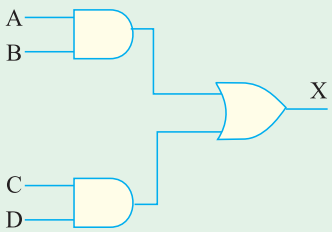


Fig. 70.87

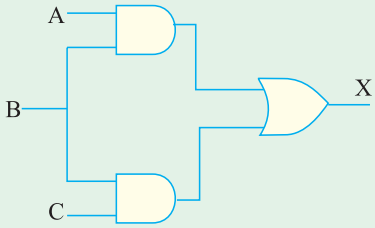


Fig. 70.88

2. What is the output equation of the logic circuit shown in Fig. 70.88 ? Evaluate the output if $A = 1$, $B = 1$, $C = 0$. ***[X= AB + (B + C); 1]***

3. After finding the Boolean equation for the circuit shown in Fig. 70.89, compute the output if $A = 1$, $B = 0$, $C = 1$, $D = 0$. ***[X = (A + B) (C + D) ; 1]***

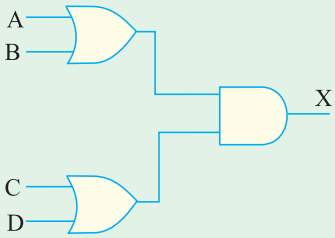


Fig. 70.89

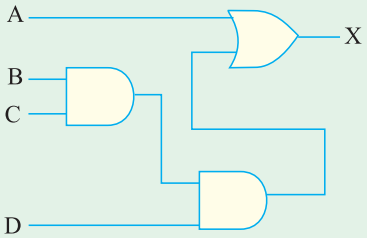


Fig. 70.90

4. Translate the hardware shown in Fig. 70.90 into a Boolean expression. Compute the value of the output if $A = 0$, $B = 1$, $C = 0$, $D = 1$. ***[X = A + BCD ; 0]***

5. What is the Boolean expression for the logic diagram shown in Fig. 70.91? Evaluate its output if $A = 1$, $B = 1$, and $C = 1$. ***[X= AB + C; 1]***

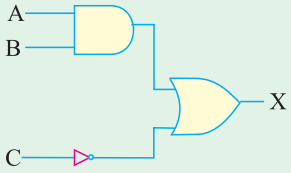


Fig. 70.91

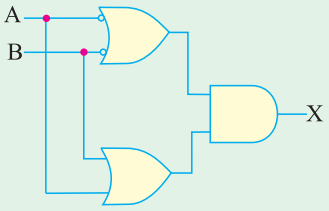


Fig. 70.92

6. Find Boolean expression for the logic circuit of Fig. 70.92. What is the output if $A = 1$, $B = 1$?

[$X = (\overline{A} + \overline{B}).\ (A + B)\ ;\ 0$]

7. Give the logic functions performed by the circuits shown in Fig. 70.93 (*a*), (*b*) and (*c*).

[(*a*) $X = \overline{AB}.\overline{(C + D)}$ (*b*) $X = \overline{AB.C}$ (*c*) $\overline{(A + B)CD}$]

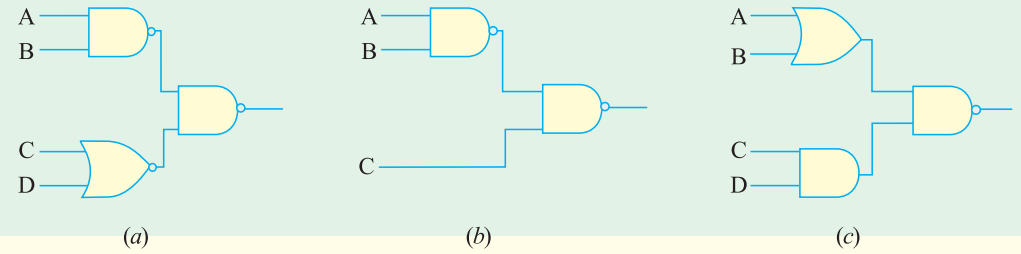


Fig. 70.93

8. State the logic functions of the circuits shown in Fig. 70.94 (*a*), (*b*) and (*c*).

[(*a*) (A + B).(CD) (*b*) AB + CD (*c*) (A + B). C.D.E]

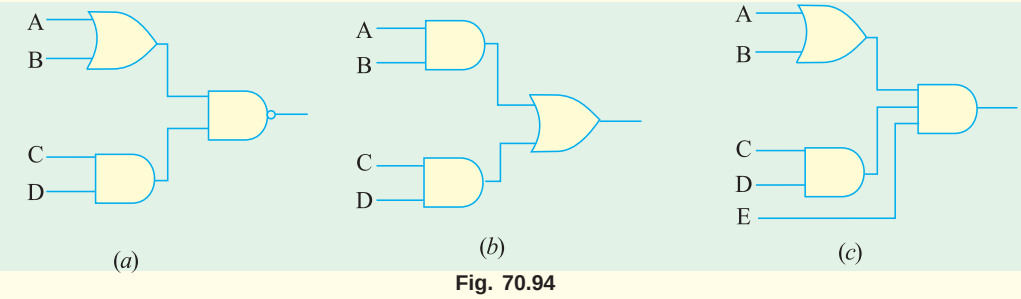


Fig. 70.94

9. State the logic functions performed by the circuits of Fig. 70.95 (*a*), (*b*) and (*c*).

[(*a*) $\overline{AB}.\overline{(C + D)}$ (*b*) $\overline{AB + C}$ (*c*) A + B]

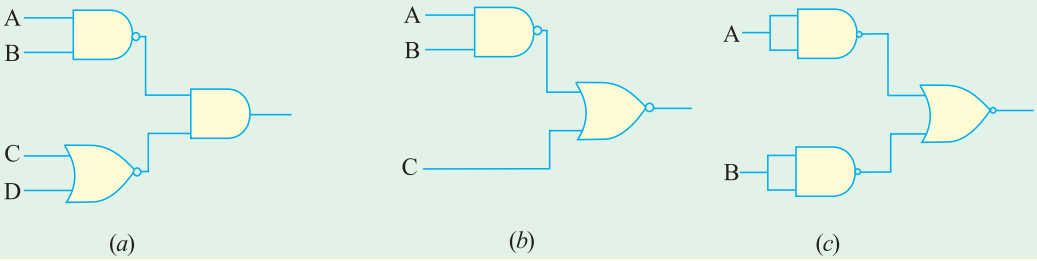


Fig. 70.95

10. Write the logic functions for the circuits shown in Fig. 70.96 (*a*), (*b*) and (*c*).

[(*a*) A(B + C) + CD (*b*) $\overline{A + B.A.C}$ (*c*) $A\overline{B} + \overline{C}D$]

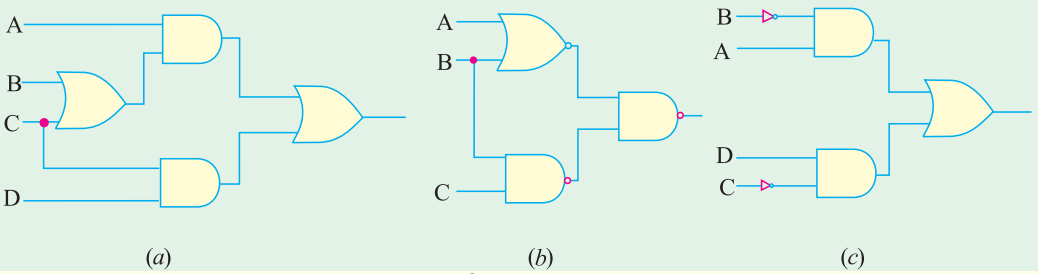


Fig. 70.96

11. What would be the output signal if two signals $A = 101011_2$ and $B = 110101_2$ are applied to the inputs of an *AND* gate? **[100 001]2**

12. What would be the output signal if two input binary signals given by $A = 100101$ and $B = 110110$ are applied to (*a*) *OR* gate (*b*) *NAND* gate and (*c*) *XNOR* gate?

[(*a*) 1101112 (*b*) 0110112 (*c*) 1011002]

13. Sketch the output waveform at *C*-for a 2-input *OR* gate shown in Fig. 70.97 and with the given *A* and *B* input waveforms.

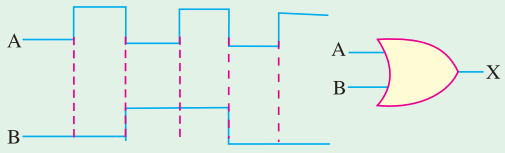


Fig. 70.97

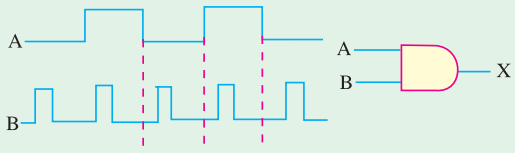


Fig. 70.98

14. Sketch the output waveform of a 2-input *AND* gate shown in Fig. 70.98 with the input waveforms *A* and *B*.

15. Sketch the output waveform at *D* for a 3-input *AND* gate shown in Fig. 70.99, with the given *A*, *B* and *C* input waveforms.

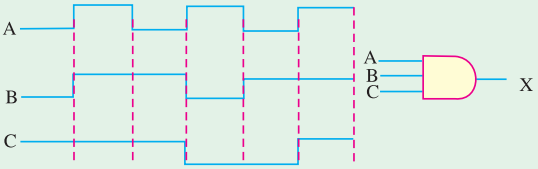


Fig. 70.99

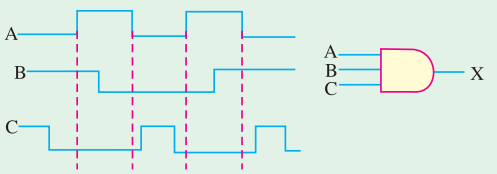


Fig. 70.100

16. Sketch the output waveform at *D* for a 3-input *AND* gate shown in Fig. 70.100, with the given *A*, *B* and *C* input waveforms.

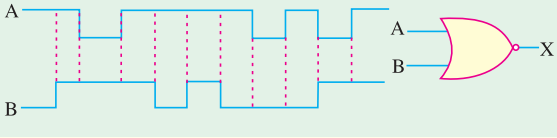


Fig. 70.101

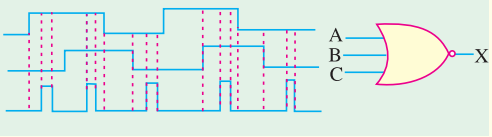


Fig. 70.102

17. Sketch the output waveform at *C* for a 2-input *NOR* gate shown in Fig. 70.101, with the given *A* and *B* input waveforms.

18. Sketch the output waveform at *D* for a 3-input *NOR* gate shown in Fig. 70.102, with the given *A*, *B* and *C* input waveforms.

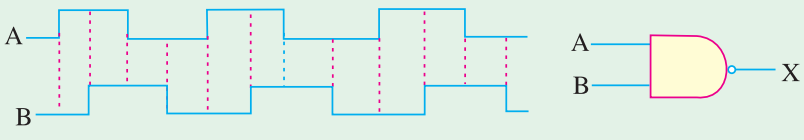


Fig. 70.103

19. Sketch the output waveform at *C* for a 2-input *NAND* gate shown in Fig. 70.103, with the given *A* and *B* input waveforms.

20. Sketch the output waveform at *D* for a 3-input *NAND* gate shown in Fig. 70.104, with the given *A*, *B* and *C* input waveforms.

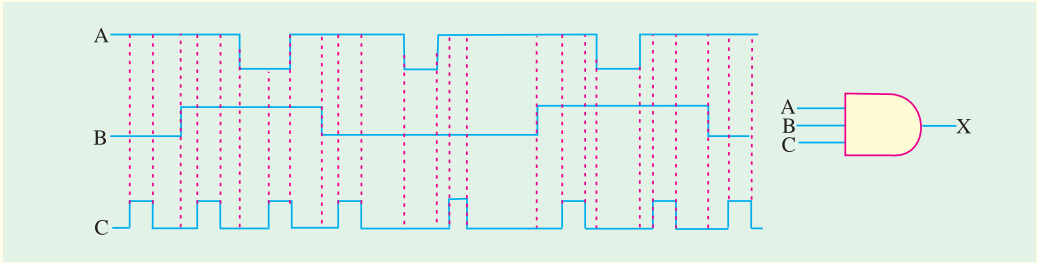


Fig. 70.104

OBJECTIVE TESTS – 70

1. A logic gate is an electronic circuit which
(*a*) makes logic decision
(*b*) allows electron flow only in one direction
(*c*) works on binary algebra
(*d*) alternates between 0 and 1 values.

2. In positive logic, logic state 1 corresponds to
(*a*) positive voltage
(*b*) higher voltage level
(*c*) zero voltage level
(*d*) lower voltage level.

3. In negative logic, the logic state 1 corresponds to
(*a*) negative voltage
(*b*) zero voltage
(*c*) more negative voltage
(*d*) lower voltage level.

4. The voltage levels of a negaive logic system
(*a*) must necessarily be negative
(*b*) may be negative orpositive
(*c*) must necessarily be positive
(*d*) must necessarily be 0V and –5V

5. The output of a 2-input *OR* gate is zero only when its
(*a*) both inputs are 0
(*b*) either input is 1
(*c*) both inputs are 1
(*d*) either input is 0.

6. An *XOR* gate produces an output only when its two inputs are
(*a*) high (*b*) low
(*c*) different (*d*) same.

7. An AND gate
(*a*) implements logic addition
(*b*) is equivalent to a series switching circuit
(*c*) is an any-or-all gate
(*d*) is equivalent to a parallel switching circuit.

8. When an input electrical signal *A* = 10100 is applied to a NOT gate, its output signal is
(*a*) 01011 (*b*) 10101
(*c*) 10100 (*d*) 00101.

9. The only function of a NOT gate is to
(*a*) stop a signal
(*b*) recomplement a signal
(*c*) invert an input signal
(*d*) act as a universal gate.

10. A NOR gate is ON only when all its inputs are
(*a*) ON (*b*) positive
(*c*) high (*d*) OFF.

11. For getting an output from an XNOR gate, its both inputs must be
(*a*) high (*b*) low
(*c*) at the same logic level
(*d*) at the opposite logic levels.

12. In a certain 2-input logic gate, when *A* = 0, *B* = 0, then *C* = 1 and when *A* = 0, *B* = 1, then again *C* = 1. It must be ..... gate.
(*a*) XOR (*b*) AND
(*c*) NAND (*d*) NOR

13. The logic symbol shown in Fig. 70.105 represents
(*a*) single-output AND gate
(*b*) NAND gate
(*c*) NAND gate used as NOT gate
(*d*) NOR gate.

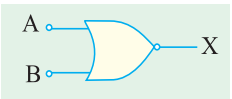


Fig. 70.105

14. The output from the logic gate shown in Fig. 70.106 will be available when inputs ..... are present.

(*a*) *A* and *C*

(*b*) *B* and *C*

(*c*) *A*, *B* and *C*

(*d*) *A* and *B*

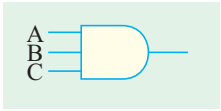


Fig. 70.106

15. To get an output 1 from circuit of Fig. 70.107, the input must be *A B C*

(*a*) 0 1 0 (*b*) 1 0 0

(*c*) 1 0 1 (*d*) 1 1 0

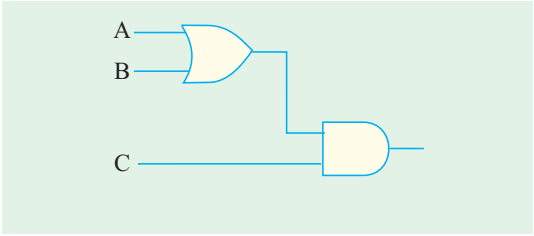


Fig. 70.107

16. Which of the following logic gates in Fig. 70.108 will have an output of 1 ?

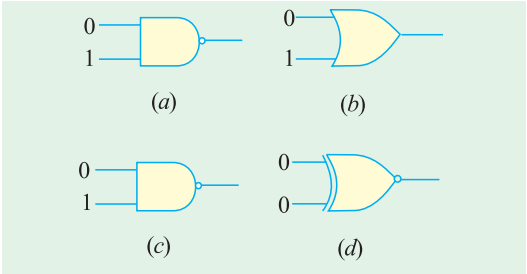


Fig. 70.108

17. A half-adder can be constructed from

(*a*) two XNOR gates only

(*b*) one XOR and one OR gate with their outputs connected in parallel

(*c*) one XOR and one OR gate with their inputs connected in parallel

(*d*) one XOR gate and one AND gate

18. The digital equivalent of an electric series circuit is the ..... gate.

(*a*) NOR (*b*) NAND

(*c*) OR (*d*) AND

19. Which of the following represents analog data ?

(*a*) ON and OFF states (*b*) 0 and 1

(*c*) 0V and 5V (*d*) 1.5, 3.2, 4 and 5V

20. The logic gate which produces a 0 or low-level output when one or both of the inputs are 1 is called ..... gate.

(*a*) AND (*b*) OR

(*c*) NOR (*d*) NAND

21. The output X of the gated network shown in Fig. 70.109 is

(*a*) $\overline{AB} \cdot \overline{CD} \cdot \overline{EF}$

(*b*) $\overline{AB} + \overline{CD} + \overline{EF}$

(*c*) AB + CD + EF

(*d*) (A + B) (C + D) (E + F)

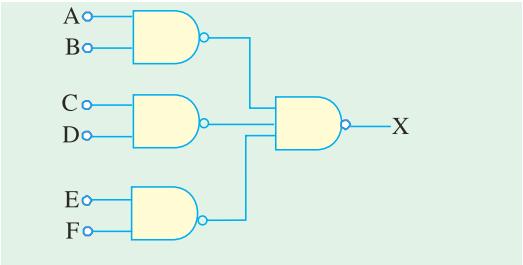


Fig. 70.109

22. The digital circuit shown in Fig 70.110 generates a modified clock pulse at the output. Choose the correct output waveform from the options given below. ***(GATE; 2004)***

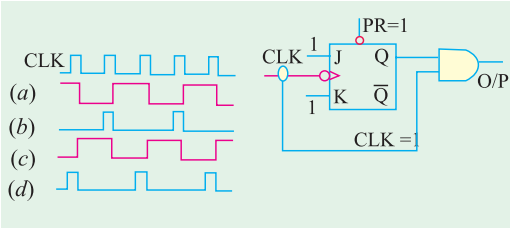


Fig. 70.110

ANSWERS

1. (*a*) **2.** (*b*) **3.** (*d*) **4.** (*b*) **5.** (*a*) **6.** (*c*) **7.** (*b*) **8.** (*a*) **9.** (*c*) **10.** (*d*)
11. (*c*) **12.** (*c*) **13.** (*d*) **14.** (*c*) **15.** (*c*) **16.** (*d*) **17.** (*c*) **18.** (*d*) **19.** (*d*) **20.** (*c*)
21. (*c*) **22.** (*d*)