UNIT 3 BOOLEAN ALGEBRA AND CIRCUITS

Structure

3.0 INTRODUCTION

Fig. 1: Claude Shannon

This unit is very closely linked with Unit 1. It was C.E.Shannon, the founder of information theory, who observed an analogy between the functioning of switching circuits and certain operations of logical connectives. In 1938 he gave a technique based on this analogy to **express and manipulate** simple switching circuits algebraically. Later, the discovery of some new solid state devices (called **electronic switches** or **logic gates**) helped to modify these algebraic techniques and, thereby, paved a way to solve numerous problems related to digital systems algebraically.

In this unit, we shall discuss the symbolic logic techniques which are required for the algebraic understanding of circuits and computer logic. In Sec. 3.2, we shall introduce you to **Boolean algebras** with the help of certain examples based on objects you are already familiar with. You will see that such algebras are apt for describing operations of logical circuits used in computers.

In Sec. 3.3, we have discussed the linkages between **Boolean expressions** and logic circuits.

In Sec. 3.4, you will read about how to express the overall functioning of a circuit mathematically in terms of certain suitably defined functions called **Boolean functions**. In this section we shall also consider a simple **circuit design problem** to illustrate the applications of the relationship between Boolean functions and circuits.

Let us now consider the objectives of this unit.

3.1 OBJECTIVES

After reading this unit, you should be able to:

- define and give examples of Boolean algebras, expressions and functions;
- give algebraic representations of the functioning of logic gates;
- obtain and simplify the Boolean expression representing a circuit;
- construct a circuit for a Boolean expression;
- design and simplify some simple circuits using Boolean algebra techniques.

3.2 BOOLEAN ALGEBRAS

Let us start with some questions: Is it possible to design an electric/electronic circuit without actually using switches(or logic gates) and wires? Can a circuit be redesigned, to get a simpler circuit with the help of pen and paper only?

The answer to both these questions is `Yes'. What allows us to give this reply is the concept of **Boolean algebras**. Before we start a formal discussion on these types of algebras, let us take another look at the objects treated in Unit 1.

As before, let the letters p, q, r,. . . denote statements (or propositions). We write $\mathcal{S}$ for the set of all propositions. As you may recall, a tautology $\mathcal{T}$ (or a contradiction $\mathcal{F}$) is any proposition which is always true (or always false, respectively). By abuse of notation, we shall **let $\mathcal{T}$ denote the set of all tautologies and $\mathcal{F}$ denote the set of all contradictions.** Thus,

$$\mathcal{T} \leq \mathsf{S}, \mathcal{F} \leq \mathsf{S}.$$

You already know from Unit 1 that, given two propositions p and q, both p ^ q and p v q are again propositions. And so, by the definition of a binary operation, you can see that both $\wedge$ (**conjunction**) and $\vee$ (**disjunction**) are binary operations on the set S, where we are writing $\wedge$ (p, q) as $p \wedge q$ and v (p, q) as $p \vee q \; \forall \; p, q \in S$.

Again, since ~ p is also a proposition, the operation ~ **(negation)** defines a unary function $\sim: \mathsf{S} \to \mathsf{S}$. Thus, the set of propositions S, with these operations, acquires an algebraic structure.

As is clear from Sec.1.3, under these three operations, the elements of S satisfy **associative laws, commutative laws, distributive laws** and **complementation laws.**

Also, by E19 of Unit 1, you know that $p \vee \mathcal{F} = p$ and $p \wedge \mathcal{T} = p$, for any proposition p. These are called the **identity laws**. The set S with the three operations and properties listed above is a particular case of an algebraic structure which we shall now define.

In the NCERT textbook, '+' and '.' are used instead of '$\vee$' and '$\wedge$', respectively.

Definition: A Boolean algebra B is an algebraic structure which consists of a set X ($\neq \emptyset$) having two binary operations (denoted by $\vee$ and $\wedge$), one unary operation (denoted by ') and two specially defined elements **O** and **I** (say), which satisfy the following five laws for all x, y, z $\in$ **X**.

B1. Associative Laws: $x \vee (y \vee z) = (x \vee y) \vee z$, $x \wedge (y \wedge z) = (x \wedge y) \wedge z$

B2. Commutative Laws: $x \vee y = y \vee x$, $x \wedge y = y \wedge x$

B3. Distributive Laws: $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$, $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

B4. Identity Laws: $x \vee \mathbf{O} = x$, $x \wedge \mathbf{I} = x$

B5. Complementation Laws: $x \wedge x' = \mathbf{O}$, $x \vee x' = \mathbf{I}$.

We write this algebraic structure as **B** = (**X**, $\vee$, $\wedge$, ' , **O, I**), or simply **B**, if the context makes the meaning of the other terms clear. The two operations $\vee$ and $\wedge$ are called the **join operation** and **meet operation**, respectively. The unary operation ' is called the **complementation**.

From our discussion preceding the definition above, you would agree that the set S of propositions is a Boolean algebra, where $\mathcal{T}$ and $\mathcal{F}$ will do the job of **I** and **O**, respectively. Thus, (S, $\wedge$, $\vee$, ~, $\mathcal{F}$, $\mathcal{T}$) is an example of a Boolean algebra.

We give another example of a Boolean algebra below.

Example 1: Let X be a non-empty set, and $\mathcal{P}$(**X**) denote its power set, i.e., $\mathcal{P}$(**X**) is the set consisting of all the subsets of the set X. Show that $\mathcal{P}$(**X**) is a Boolean algebra.

Solution: We take the usual set-theoretic operations of intersection ($\cap$), union ($\cup$), and complementation (^{c}) in $\mathcal{P}(X)$ as the three required operations. Let ø and X play the roles of **O** and **I**, respectively. Then you can verify that all the conditions for ($\mathcal{P}$ (**X**), $\cup$, $\cap$, ^{c}, **Φ**, **X**) to be a Boolean algebra hold good.

For instance, the identity laws (B4) follow from two set-theoretic facts, namely, `the intersection of any subset with the whole set is the set itself' and `the union of any set with the empty set is the set itself'. On the other hand, the complementation laws (B5) follow from another set of facts from set theory, namely, `the intersection of any subset with its complement is the empty set' and `the union of any set with its complement is the whole set'.

***

Yet another example of a Boolean algebra is based on **switching circuits**. For this, we first need to elaborate on the functioning of ordinary switches in a mathematical way. In fact, we will present the basic idea which helped the American, C.E.Shannon, to detect the connection between the functioning of switches and Boole's symbolic logic.

You may be aware of the functioning of a simple on-off switch which is commonly used as an essential component in the electric (or electronic) networking systems. A switch is a device which allows the current to flow only when it is placed in the **ON** position, i.e., when the gap is **closed** by a conducting rod. Thus, the **ON** position of a switch is one state of a switch, called a **closed state**. The other state of a switch is the open state, when it is placed in the **OFF** position. So, a switch has two stable states.

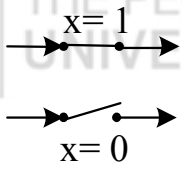


Fig. 2: OFF-ON position

There is another way to talk about the functioning of a switch. We can denote a switch by x, and use the values 0 and 1 to depict its two states, i.e., to convey that x is open we write x = 0, and to convey that x is closed we write x = 1 (see Fig.2).

These values which denote the state of a switch x are called the **state-values** (**s.v.,** in short) of that switch.

We shall also write x′ for a switch which is always in a state opposite to x. So that,

x **is open** → x' **is closed** and x **is closed** → x' **is open**.

The switch x' is called the invert of the switch x. For example, the switch a' shown in Fig.3 is an invert of the switch a.

Table 1: s.v. of x'

x	x'
0	1
1	0

Table 1 alongside gives the state value of x' for a given state value of the switch x. These values are derived from the definition of x' and our preceding discussion.

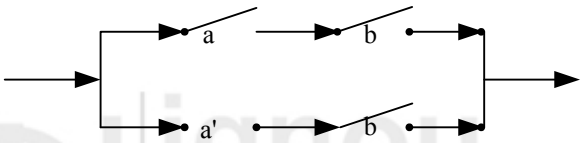
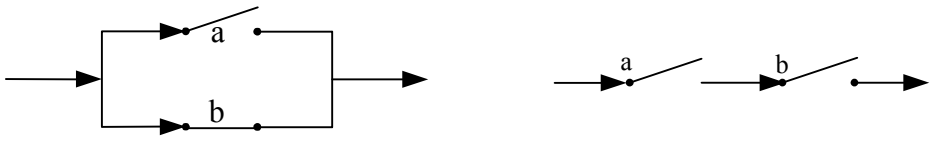


Fig. 3: a` is the invert of a.

Note that the variable x that denotes a switch can only take on 2 values, 0 and 1. Such a variable (which can only take on two values) is called a **Boolean variable**. Thus, if x is a Boolean variable, so is x'. Now, in order to design a circuit consisting of several switches, there are two ways in which two switches can be connected: **parallel connections** and **series connections** (see Fig.4).

Do you see a connection between Table 1 above and Table 10, Unit 1 ?



(i) Parallel Connection (ii) Series Connection

Fig. 4: Two ways of connecting switches

From Fig.4(i) above, you can see that in case of a parallel connection of switches a and b (say), current will flow from the left to the right extreme if **at** least one of the two switches is closed. Note that 'parallel' **does not** mean that both the switches are in the same state.

On the other hand, current can flow in a series connection of switches only when **both** the switches a and b are closed (see Fig.4 (ii)).

Given two switches a and b, we write a **par** b and a **ser** b for these two types of connections, respectively.

In view of these definitions and the preceding discussion, you can see that the state values of the connections a **par** b and a **ser** b, for different pairs of state values of switches a and b, are as given in the tables below.

Table 2: State values of a par b and a ser b.

s.v. of a	**s.v.** of b	**s.v.** of a **par** b
0	0	0
0	1	1
1	0	1
1	1	1

s.v. of a	**s.v.** of b	**s.v.** of a **ser** b
0	0	0
0	1	0
1	0	0
1	1	1

We have now developed a sufficient background to give you the example of a Boolean algebra which is based on switching circuits.

Example 2: The set S = {0, 1} is a Boolean algebra.

Solution: Take **ser** and **par** in place of $\wedge$and $\vee$, respectively, and inversion(') instead of ~ as the three required operations in the definition of a Boolean algebra,. Also take 0 for the element **O** and 1 for the element **I** in this definition.
Now, using Tables 1 and 2, you can check that the five laws B1-B5 hold good. Thus, (**S, par, ser,**', 0, 1) is a Boolean algebra.

***

A Boolean algebra whose underlying set has only two elements is very important in the study of circuits. We call such an algebra a **two-element Boolean algebra**, and denote it by $\mathcal{B}$. From this Boolean algebra we can build many more, as in the following example.

Example 3: Let $\mathcal{B}^n = \mathcal{B} \times \mathcal{B} \times \cdots \times \mathcal{B} = \{(e_1, e_2, \ldots, e_n) \mid \text{each } e_i = 0 \text{ or } 1\}$, for $n \geq 1$, be the Cartesian product of n copies of $\mathcal{B}$. For $i_k, j_k \in \{0, 1\}$ $(1 \leq k \leq n)$, define

$$(i_1, i_2, \ldots, i_n) \wedge (j_1, j_2, \ldots, j_n) = (i_1 \wedge j_1, i_2 \wedge j_2, \ldots, i_n \wedge j_n),$$
$$(i_1, i_2, \ldots, i_n) \vee (j_1, j_2, \ldots, j_n) = (i_1 \vee j_1, i_2 \vee j_2, \ldots, i_n \vee j_n), \text{ and}$$
$$(i_1, i_2, \ldots, i_n)' = (i'_1, i'_2, \ldots, i'_n).$$

Then B^n is a Boolean algebra, for all $n \geq 1$.

Solution: Firstly, observe that the case n = 1 is the Boolean algebra $\mathcal{B}$.
Now, let us write 0 = (0, 0, . . . , 0) and **I** = (1, 1, . . . , 1), for the two elements of $\mathcal{B}^n$ consisting of n-tuples of 0's and 1's, respectively. Using the fact that $\mathcal{B}$ is a Boolean algebra, you can check that $\mathcal{B}^n$, with operations as defined above, is a Boolean algebra for every $n \geq 1$.

***

The Boolean algebras $\mathcal{B}^n$, $n \geq 1$, (called **switching algebras**) are very useful for the study of the hardware and software of digital computers.

We shall now state, without proof, some other properties of Boolean algebras, which can be deduced from the five laws (B1-B5).

Theorem 1: Let $\mathcal{B} = (\mathbf{S}, \vee, \wedge, ', \mathbf{O}, \mathbf{I})$ be a Boolean algebra. Then the following laws hold $\forall x, y \in S$.

a) **Idempotent laws** : $x \vee x = x$, $x \wedge x = x$.
b) **Absorption laws :** $x \vee (x \wedge y) = x$, $x \wedge (x \vee y) = x$.
c) **Involution law :** $(x')' = x$.
d) **De Morgan's laws :** $(x \vee y)' = x' \wedge y'$, $(x \wedge y)' = x' \vee y'$.

In fact, you have already come across some of these properties for the Boolean algebras of propositions in Unit 1. In the following exercise we ask you to verify them.

E1) a) Verify the identity laws and absorption laws for the Boolean algebra $(\mathbf{S}, \wedge, \vee, \sim, \mathcal{T}, \mathcal{F})$ of propositions.
b) Verify the absorption laws for the Boolean algebra $(\mathcal{P}(\mathbf{X}), \cup, \cap, ^c, \Phi, \mathbf{X})$.

In Theorem 1, you may have noticed that for each statement involving $\vee$ and $\wedge$, there is an analogous statement with $\wedge$ (instead of $\vee$) and $\vee$ (instead of $\wedge$). This is not a coincidence, as the following definition and result shows.

Definition : If p is a proposition involving $\sim$, $\wedge$ and $\vee$, the **dual** of p, denoted by p^d, is the proposition obtained by replacing each occurrence of $\wedge$ (and/or $\vee$) in p by $\vee$ (and/or $\wedge$, respectively) in p^d .
For example, $x \vee (x \wedge y) = x$ is the **dual** of $x \wedge (x \vee y) = x$.

Now, the following principle tells us that if a statement is proved true, then we have simultaneously proved that its dual is true.

Theorem 2 (The principle of duality): If s is a theorem about a Boolean algebra, then so is its dual s^d .
It is because of this principle that the statements in Theorem 1 look so similar.

Let us now see **how to apply Boolean algebra methods to circuit design.**

While expressing circuits mathematically, we identify each circuit in terms of some Boolean variables. Each of these variables represents either a simple switch or an input to some electronic switch.

Definition: Let $\mathcal{B} = (S, \vee, \wedge, ', \mathbf{O}, \mathbf{I})$ be a Boolean algebra. **A Boolean expression** in variables $x_1, x_2, \ldots, x_k$ (say), each taking their values in the set **S** is defined recursively as follows:

i) Each of the variables $x_1, x_2, \ldots, x_k$, as well as the elements **O** and **I** of the Boolean algebra $\mathcal{B}$ are Boolean expressions.

ii) If $\mathbf{X_1}$ and $\mathbf{X_2}$ are previously defined Boolean expressions, then $\mathbf{X_1 \wedge X_2}$, $\mathbf{X_1 \vee X_2}$ and $\mathbf{X'_1}$ are also Boolean expressions.

For instance, $x_1 \wedge x'_3$ is a Boolean expression because so are x_1 and x'_3 , Similarly, because $x_1 \wedge x_2$ is a Boolean expression, so is $(x_1 \wedge x_2) \wedge (x_1 \wedge x'_3)$.

If **X** is a Boolean expression in n variables $x_1, x_2, \ldots, x_n$ (say), we write this as $\mathbf{X = X(x_1, \ldots, x_n)}$.
In the context of simplifying circuits, we need to reduce Boolean expressions to

simpler ones. `Simple' means that the expression has fewer connectives, and all the literals involved are distinct. We illustrate this technique now.

Example 4: Reduce the following Boolean expressions to a simpler form.

(a) $X(x_1, x_2) = (x_1 \wedge x_2) \wedge (x_1 \wedge x'_2)$;
(b) $X(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_3)$.

Solution: (a) Here we can write

$(x_1 \wedge x_2) \wedge (x_1 \wedge x'_2) = ((x_1 \wedge x_2) \wedge x_1) \wedge x'_2$	(Associative law)
$= (x_1 \wedge x_2) \wedge x'_2$	(Absorption law)
$= x_1 \wedge (x_2 \wedge x'_2)$	(Associative law)
$= x_1 \wedge \mathbf{O}$	(Complementation law)
$= \mathbf{O}$.	(Identity law)

Thus, in its simplified form, the expression given in (a) above is **O**, i.e., a **null expression.**

(b) We can write

$(x_1 \wedge x_2) \vee (x_1 \wedge x'_2 \wedge x_3) \vee (x_1 \wedge x_3)$	
$= [x_1 \wedge \{x_2 \vee (x'_2 \wedge x_3\}] \wedge (x_1 \wedge x_3)$	(Distributive law)
$= [x_1 \wedge \{(x_2 \vee x'_2) \wedge (x_2 \vee x_3)\}] \wedge (x_1 \wedge x_3)$	(Distributive law)
$= [x_1 \wedge \{\mathbf{I} \wedge (x_2 \vee x_3)\}] \wedge (x_1 \wedge x_3)$	(Complementation law)
$= [x_1 \wedge (x_2 \vee x_3)] \wedge (x_1 \wedge x_3)$	(Identity law)
$= [(x_1 \wedge x_2) \vee (x_1 \wedge x_3)] \wedge (x_1 \wedge x_3)$	(Distributive law)
$= [(x_1 \wedge x_2) \wedge (x_1 \wedge x_3)] \vee [(x_1 \wedge x_3) \wedge (x_1 \wedge x_3)]$	(Distributive law)
$= (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_3)$	(Idemp.,& assoc. laws)
$= x_1 \wedge [(x_2 \wedge x_3) \vee x_3]$	(Distributive law)
$= x_1 \wedge x_3$	(Absorption law)

Thus, the simplified form of the expression given in (b) is $(x_1 \wedge x_3)$.

***

Now you should find it easy to solve the following exercise.

E2) Simplify the Boolean expression
$\mathbf{X}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee ((x_1 \wedge x_2) \wedge x_3) \vee (x_2 \wedge x_3)$.

With this we conclude this section. In the next section we shall give an important application of the concepts discussed here.

3.3 LOGIC CIRCUITS

If you look around, you would notice many electric or electronic appliances of daily use. Some of them need a simple switching circuit to control the auto-stop (such as in a stereo system). Some would use an auto-power off system used in transformers to control voltage fluctuations. Each circuit is usually a combination of on-off switches, wired together in some specific configuration. Nowadays certain types of **electronic blocks** (i.e., solid state devices such as transistors, resistors and capacitors) are more in use. We call these electronic blocks **logic gates,** or simply, **gates.** In Fig. 5 we have shown a box which consists of some electronic switches (or logic gates), wired together in a specific manner. Each line which is entering the box from the left represents an independent power source (called **input**), where all of them need not supply voltage to the box at a given moment. A single line coming out of the box gives the **final output** of the circuit box. The output depends on the type of input.

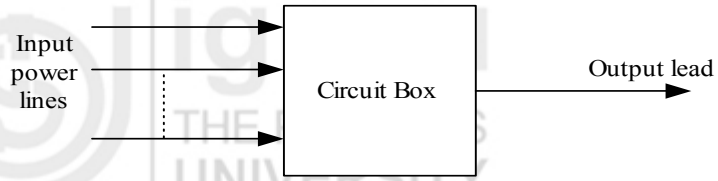


Fig. 5: A Logic circuit

This sort of arrangement of **input power lines, a circuit box** and **output lead** is basic to all electronic circuits. Throughout the unit, any such interconnected assemblage of logic gates is referred to as a **logic circuit.**

As you may know, computer hardwares are designed to handle only two levels of voltage, both as inputs as well as outputs. These two levels, denoted by 0 and 1, are called **bits** (an acronym for **binary digits**). When the bits are applied to the logic gates by means of **one** or **two** wires (input leads), the output is again in the form of voltages 0 and 1. Roughly speaking, **you may think of a gate to be on or off according to whether the output voltage is at level 1 or 0, respectively.**

Three basic types of logic gates are an **AND-gate,** an **OR-gate** and a **NOT-gate.** We shall now define them one by one.

Definition : Let the Boolean variables x_1 and x_2 represent any two bits. An **AND-gate** receives inputs x_1 and x_2 and produces the output, denoted by $x_1 \wedge x_2$, as given in Table 3 alongside.

Table 3:Outputs of AND-gate

x_1	x_2	$x_1 \wedge x_2$
0	0	0
1	1	0
0	1	0
1	1	1

The standard pictorial representation of an **AND**-gate is shown in Fig.6 below.

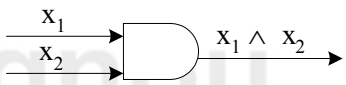


Fig. 6: Diagrammatic representation of an AND -gate

From the first three rows of Table 3, you can see that whenever the voltage in any one of the input wires of the **AND**-gate is at level 0, then the output voltage of the gate is also at level 0. You have already encountered such a situation in Unit 1. In the following exercise we ask you to draw an analogy between the two situations.

E3) Compare Table 3 with Table 2 of Unit 1. How would you relate $x_1 \wedge x_2$ with $p \wedge q$, where p and q denote propositions?

Let us now consider another elementary logic gate.

Definition : An **OR-gate** receives inputs x_1 and x_2 and produces the output, denoted by $x_1 \vee x_2$, as given in Table 4. The standard pictorial representation used for the **OR**-gate is as shown in Fig.7.

Table 4: Output of an OR-gate.

x_1	x_2	$x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

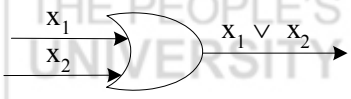


Fig. 7: Diagrammatic representation of an OR-gate

From Table 4 you can see that the situation is the other way around from that in Table 3, i.e., the output voltage of an **OR**-gate is at level 1 whenever the level of voltage in even one of the input wires is 1. What is the analogous situation in the context of propositions? The following exercise is about this.

E4) Compare Table 4 with Table 1 of Unit 1. How would you relate $x_1 \vee x_2$ with $p \vee q$, where p and q are propositions?

And now we will discuss an electronic realization of the invert of a simple switch about which you read in Sec. 3.2.

Definition : A NOT-gate receives bit x as input, and produces an output denoted by x', as given in Table 5. The standard pictorial representation of a **NOT-**gate is shown in Fig. 8 below.

Table 5: Output of a NOT-gate

x	x'
0	1
1	0

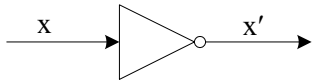


Fig. 8: Diagrammatic representation of NOT-gate

If you have solved E5 and E6, you would have noticed that Tables 3 and 4 are the same as the truth tables for the logic connectives $\wedge$ (conjunction) and $\vee$(disjunction). Also Table 3 of Unit 1, after replacing T by 1 and F by 0, gives Table 5. This is why the output tables for the three elementary gates are called **logic tables.** You may find it useful to remember these logic tables because they are needed very often for computing the logic tables of logic circuits.

Another important fact that these logic tables will help you prove is given in the following exercise.

E5) Let $\mathcal{B} = \{0, 1\}$ consist of the bits 0 and 1. Show that $\mathcal{B}$ is a Boolean algebra, i.e., that the bits 0 and 1 form a two-element Boolean algebra.

As said before, a logic circuit can be designed using elementary gates, where the output from an **AND-**gate, or an **OR**-gate, or a **NOT**-gate is used as an input to other such gates in the circuitry. The different levels of voltage in these circuits, starting from the input lines, move only in the direction of the arrows as shown in all the figures given below. For instance, one combination of the three elementary gates is shown in Fig.9.

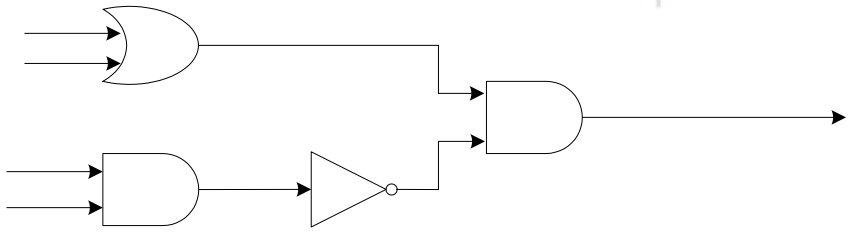


Fig. 9: A logic circuit of elementary gates.

Now let us try to see the connection between logic circuits and Boolean expressions. We first consider the elementary gates. For a given pair of inputs x_1 and x_2 , the output in the case of each of these gates is an expression of the form $x_1 \wedge x_2$ or $x_1 \vee x_2$ or x' .

Next, let us look at larger circuits. Is it possible to find an expression associated with a logic circuit, using the symbols $\wedge$, $\vee$ and ' ? Yes, it is. We will illustrate the technique of finding a Boolean expression for a given logic circuit with the help of some examples. But first, note that the output of a gate in a circuit may serve as an input to some other gate in the circuit, as in Fig. 9. So, to get an expression for a logic circuit the process always moves in the direction of the arrows in the circuitry. With this in mind, let us consider some circuits.

Example 5: Find the Boolean expression for the logic circuit given in Fig.9 above.

Solution: In Fig.9, there are four input terminals. Let us call them x_1 , x_2 , x_3 and x_4 . So, x_1 and x_2 are inputs to an **OR**-gate, which gives $x_1 \vee x_2$ as an output expression (see Fig. 9(a)).

Similarly, the other two inputs x_3 and x_4 , are inputs to an **AND**-gate. They will give $x_3 \wedge x_4$ as an output expression. This is, in turn, an input for a **NOT**-gate in the circuit. So, this yields $(x_3 \wedge x_4)'$ as the output expression. Now, both the expressions $x_1 \vee x_2$ and $(x_3 \wedge x_4)'$ are inputs to the extreme right **AND**-gate in the circuit. So, they give $(x_1 \vee x_2) \wedge (x_3 \wedge x_4)'$ as the final output expression, which represents the logic circuit.

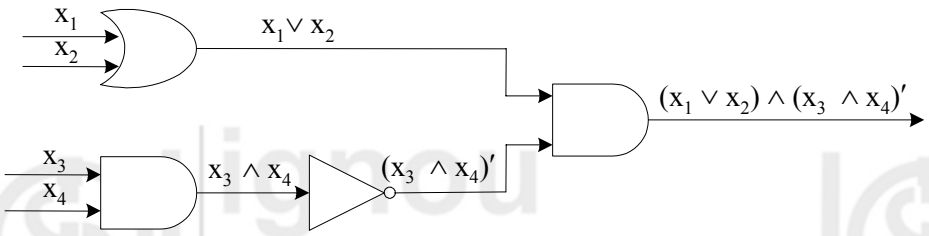


Fig. 9 (a)

***

You have just seen how to find a Boolean expression for a logic circuit. For more practice, let us find it for another logic circuit.

Example 6: Find the Boolean expression C for the logic circuit given in Fig. 10.

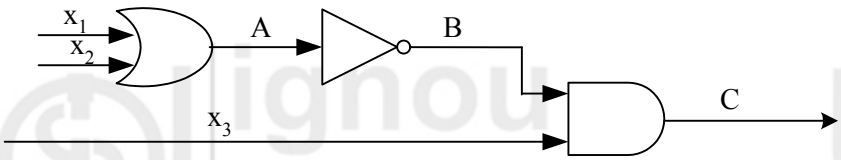


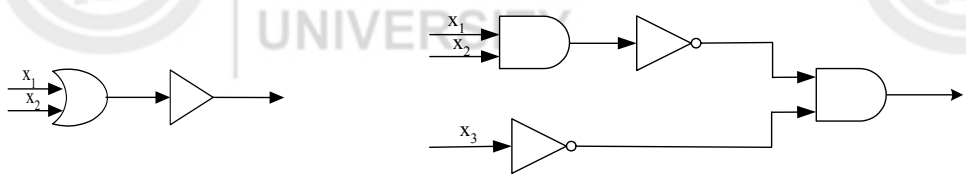
Fig. 10

Solution: Here the first output is from an **OR**-gate, i.e., A is $x_1 \vee x_2$. This, in turn, serves as the input to a **NOT**-gate attached to it from the right. The resulting bit B is $(x_1 \vee x_2)'$. This, and x_3 , serve as inputs to the extreme right **AND**-gate in the circuit given above. This yields an output expression $(x_1 \vee x_2)' \wedge x_3$, which is C, the required expression for the circuit given in Fig.10.

***

Why don't you try to find the Boolean expressions for some more logic circuits now?

E6) Find the Boolean expression for the output of the logic circuits given below.



(a) **(b)**

So far, you have seen how to obtain a Boolean expression that represents a given circuit. Can you do the converse? That is, can you construct a logic circuit corresponding to a given Boolean expression? In fact, this is done when a circuit

designing problem has to be solved. The procedure is quite simple. We illustrate it with the help of some examples.

Example 7: Construct the logic circuit represented by the Boolean expression $(x'_1 \wedge x_2) \vee (x_1 \vee x_3)$, where $x_i (1 \leq i \leq 3)$ are assumed to be inputs to that circuitry.
Solution: Let us first see what the portion $(x'_1 \wedge x_2)$ of the given expression contributes to the complete circuit. In this expression the literals x'_1 and x_2 are connected by the connective $\wedge$ (**AND**). Thus the circuit corresponding to it is as shown in Fig.11(a) below, by the definitions of **NOT**-gate and **AND**-gate.

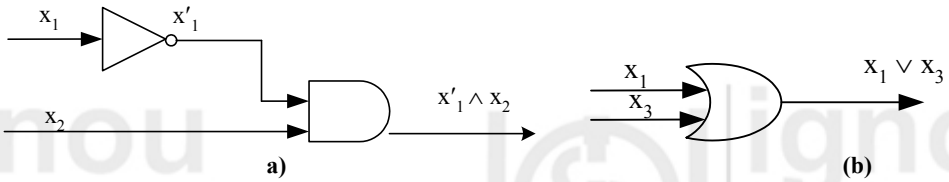


Fig. 11: Logic circuits for the expressions $x'_1 \wedge x_2$ and $x_1 \vee x_3$.

Similarly, the gate corresponding to the expression $x_1 \vee x_3$ is as shown in Fig.11(b) above. Finally, note that the given expression has two parts, namely, $x'_1 \wedge x_2$ and $x_1 \vee x_3$, which are connected by the connective $\vee$ (**OR**). So, the two logic circuits given in Fig.11 above, when connected by an **OR**-gate, will give us the circuit shown in Fig. 12 below.

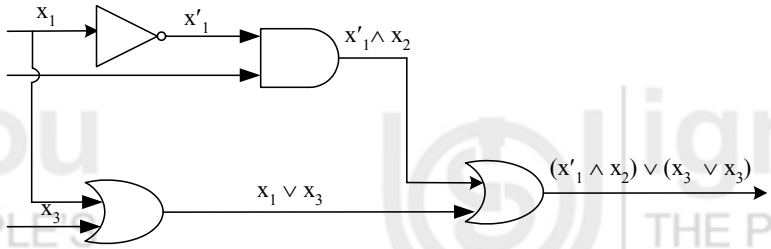


Fig.12: Circuitry for the expression $(x'_1 \wedge x_2) \vee (x_1 \vee x_3)$

This is the required logic circuit which is represented by the given expression.

***

Example 8: Given the expression $(x'_1 \vee (x_2 \wedge x'_3)) \wedge (x_2 \vee x'_4)$, find the corresponding circuit, where x_i $(1 \leq i \leq 4)$ are assumed to be inputs to the circuitry.

Solution: We first consider the circuits representing the expressions $x_2 \wedge x'_3$ and $x_2 \vee x'_4$. They are as shown in Fig.13(a).

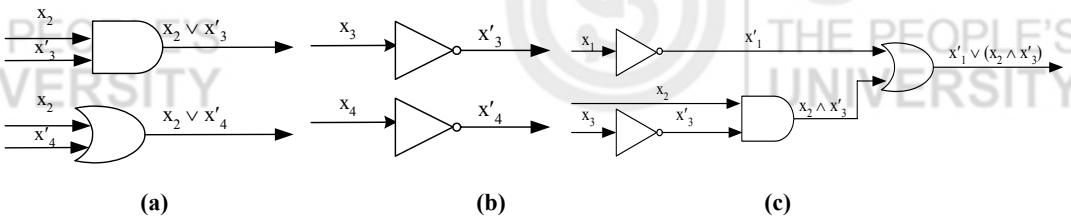


Fig. 13: Construction of a logic circuitry.

Also you know that the literals x'_3 and x'_4 are outputs of the **NOT**-gate. So, these can be represented by logic gates as shown in Fig.13(b). Then the circuit for the part $x'_1 \vee (x_2 \wedge x'_3)$ of the given expression is as shown in Fig.13(c). You already know how to construct a logic circuit for the expression $x_2 \vee x'_4$.

Finally, the two expressions $(x'_1 \vee (x_2 \wedge x'_3))$ and $(x_2 \vee x'_4)$ being connected by the connective $\wedge$ (**AND**), give the required circuit for the given expression as shown in Fig.14.

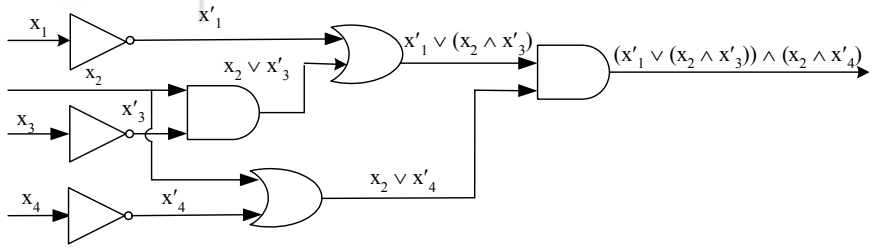


Fig. 14: Circuitry for the expression $(x'_1 \vee (x_2 \wedge x'_3)) \wedge (x_2 \vee x'_4)$.

***

Why don't you try to solve some exercises now?

E7) Find the logic circuit corresponding to the expression $x'_1 \wedge (x_2 \vee x'_3)$.

E8) Construct the logic circuit and obtain the logic table for the expression $x_1 \vee (x'_2 \wedge x_3)$.

So far we have established a one-to-one correspondence between logic circuits and Boolean expressions. You may wonder about the utility of this. The mathematical view of a circuit can help us understand the **overall functioning** of the circuit. To understand how, consider the circuit given in Fig.10 earlier.

You may think of the inputs bits x_1 , x_2 , and x_3 as three variables, each one of which is known to have two values only, namely, 0 or 1, depending upon the level of voltage these inputs have at any moment of time. Then the idea is to evaluate the expression $(x_1 \vee x_2)' \wedge x_3$, which corresponds to this circuit, for different values of the 3-tuple (x_1 , x_2 , x_3).

How does this evaluation help us to understand the functioning of the circuit? To see this, consider a situation in which the settings of x_1 , x_2 and x_3 at a certain stage of the process are $x_1 = x_3 = 0$ and $x_2 = 1$. Then we know that $x_1 \vee x_2 = 0 \vee 1 = 1$ (see the second row of Table 3 given earlier). Further, using the logic table of a **NOT**-gate, we get $(x_1 \vee x_2)' = 1' = 0$. Finally, from Table 3, we get $(x_1 \vee x_2)' \wedge x_3 = 0 \wedge 1 = 0$. Thus, the expression $(x_1 \vee x_2)' \wedge x_3$ has value 0 for the set of values (0, 1, 0) of input bits (x_1 , x_2 , x_3). **Thus, if x 1 and x 3 are closed, while x_2 is open, the circuit remains closed.**

Using similar arguments, you can very easily calculate the other values of the expression $(x_1 \vee x_2)' \wedge x_3$ in the set

$$\{0,1\}^3 = \{(x_1 , x_2 , x_3) \mid x_i = 0 \text{ or } 1,\ 1 \leq i \leq 3\}$$

of values of input bits. We have recorded them in Table 6.

Observe that the row entries in the first three columns of Table 6 represent the different values which the input bits (x_1 , x_2 , x_3) may take. Each entry in the last column of the table gives the output of the circuit represented by the expression $(x_1 \vee x_2)' \wedge x_3$ for the corresponding set of values of (x_1 , x_2 , x_3). For example, if (x_1 , x_2 , x_3) is (0,1,0), then the level of voltage in the output lead is at a level 0 (see the third row of Table 6).

You should verify that the values in the other rows are correct.

Table 6: Logic table for the expression $(x_1 \vee x_2)' \wedge x_3$.

x_1	x_2	x_3	$x_1 \vee x_2$	$(x_1 \vee x_2)$	$(x_1 \vee x_2)' \wedge x_3$
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
1	0	0	1	0	0
0	1	1	1	0	0
1	1	0	1	0	0
1	0	1	1	0	0
1	1	1	1	0	0

Table 6 is the **logic table** for the circuit given in Fig. 10.

Why don't you try an exercise now?

E9) Compute the logic table for the circuit given in E6(b) above.

You have seen how the logic table of an expression representing a circuit provides a functional relationship between the state (or level) of voltage in the input terminals and that in the output lead of that logic circuitry. This leads us the concept of Boolean functions, which we will now discuss.

3.4 BOOLEAN FUNCTIONS

In the last section you studied that an output expression is not merely a device for representing an interconnection of gates. It also defines output values as a function of input bits. This provides information about the overall functioning of the corresponding logic circuit. So, this function gives us a relation between **the inputs to the circuit** and its **final output** .

This is what helps us to understand control over the functioning of logic circuits from a mathematical point of view. To explain what this means, let us reformulate the logic tables in terms of functions of the input bits.

Let us first consider the Boolean expression

$$X(x_1 , x_2) = x_1 \wedge x'_2,$$

where x_1 and x_2 take values in $\mathcal{B} = \{0, 1\}$. You know that all the values of this expression, for different pairs of values of the variables x_1 and x_2 , can be calculated by using properties of the Boolean algebra $\mathcal{B}$. For example,

$$0 \wedge 1' = 0 \wedge 0 = 0 \Rightarrow X(0, 1) = 0.$$

Similarly, you can calculate the other values of $X(x_1, x_2) = x_1 \wedge x'_2$ over $\mathcal{B}$.

In this way we have obtained a function $f : \mathcal{B}_2 \to B$, defined as follows:

$$f(e_1 , e_2) = X(e_1 , e_2) = e_1 \wedge e'_2 , \text{ where } e_1 , e_2 \in \{0, 1\}.$$

So f is obtained by replacing x_i with e_i in the expression $X(x_1 , x_2)$. For example, when $e_1 = 1$, $e_2 = 0$, we get $f(1, 0) = 1 \wedge 0' = 1$.

More generally, each Boolean expression $X(x_1 , x_2 , \ldots , x_k)$ in k variables, where

each variable can take values from the two-element Boolean algebra $\mathcal{B}$, defines a function $f : B^k \to B : f(e_1, \ldots, e_k) = X(e_1, \ldots, e_k)$.

Any such function is called a **Boolean function.**
Thus, each Boolean expression over $\mathcal{B} = \{0, 1\}$ gives rise to a Boolean function.
In particular, corresponding to each circuit, we get a Boolean function.
Therefore, the logic table of a circuit is just another way of representing the Boolean function corresponding to it.

For example, the logic table of an **AND**-gate can be obtained using the function $\wedge$: $\mathcal{B}^2 \to B : \wedge(e_1, e_2) = e_1 \wedge e_2$.

To make matters more clear, let us work out an example.

Example 9: Let $f : \mathcal{B}^2 \to \mathcal{B}$ denote the function which is defined by the Boolean expression $X(x_1, x_2) = x'_1 \wedge x'_2$. Write the values of f in tabular form.

Solution: f is defined by $f(e_1, e_2) = e'_1 \wedge e'_2$ for $e_1, e_2 \in \{0, 1\}$. Using Tables 3, 4 and 5, we have

$f(0, 0) = 0' \wedge 0' = 1 \wedge 1 = 1$, $f(0, 1) = 0' \wedge 1' = 1 \wedge 0 = 0$,
$f(1, 0) = 1' \wedge 0' = 0 \wedge 1 = 0$, $f(1, 1) = 1' \wedge 1' = 0 \wedge 0 = 0$.

We write this information in Table 7.

Table 7: Boolean function for the expression $x'_1 \wedge x'_2$.

e_1	e_2	e'_1	e'_2	$f(e_1, e_) = e'_1 \wedge e'_2$
0	0	1	1	$1 \wedge 1 = 1$
0	1	1	0	$1 \wedge 0 = 0$
1	0	0	1	$0 \wedge 1 = 0$
1	1	0	0	$0 \wedge 0 = 0$

***

Why don't you try an exercise now?

E10) Find all the values of the Boolean function $f : \mathcal{B}_2 \to \mathcal{B}$ defined by the Boolean expression $(x_1 \wedge x_2) \vee (x_1 \wedge x'_3)$.

Let us now consider the Boolean function $g : \mathcal{B}_2 \to \mathcal{B}$, defined by the expression $X(x_1, x_2) = (x_1 \vee x_2)'$.
Then $g(e_1, e_2) = (e_1 \vee e_2)'$, $e_1, e_2 \varepsilon \mathcal{B}$.

So, the different values that g will take are

$g(0, 0) = (0 \vee 0)' = 0' = 1$, $g(0, 1) = (0 \vee 1)' = 1' = 0$,
$g(1, 0) = (1 \vee 0)' = 1' = 0$, $g(1, 1) = (1 \vee 1)' = 1' = 0$.

In tabular form, the values of g can be presented as in Table 8.

Table 8: Boolean function of the expression $(x_1 \vee x_2)'$.

e_1	e_2	$e_1 \vee e_2$	$g(e_1, e_2) = (e_1 \vee e_2)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

By comparing Tables 7 and 8, you can see that $f(e_1, e_2) = g(e_1, e_2)$ for all

$(e_1 , e_2) \in \mathcal{B}^2$. So f and g are the same function.

What you have just seen is that **two (seemingly) different Boolean expressions can have the same Boolean function specifying them.** Note that if we replace the input bits by propositions in the two expressions involved, then we get logically equivalent statements. This may give you some idea of how the two Boolean expressions are related. We give a formal definition below.

Definition : Let $X = X(x_1 , x_2 , \ldots , x_k)$ and $Y = Y(x_1 , x_2 , \ldots , x_k)$ be two Boolean expressions in the k variables $x_1 , \ldots , x_k$. We say **X is equivalent to Y** over the Boolean algebra $\mathcal{B}$, and write **$X \equiv Y$,** if both the expressions X and Y define the same Boolean function over $\mathcal{B}$, i.e.,

$$X(e_1 , e_2 , \ldots , e_k) = Y(e_1 , e_2 , \ldots , e_k), \text{ for all } e_i \in \{0, 1\}.$$

So, the expressions to which f and g (given by Tables 7 and 8) correspond are equivalent.

Why don't you try an exercise now?

E11) Show that the Boolean expressions

$$X = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \text{ and } Y = x_1 \wedge (x_2 \vee x'_3)$$

are equivalent over the two-element Boolean algebra $\mathcal{B} = \{0, 1\}$.

So far you have seen that given a circuit, we can define a Boolean function corresponding to it. You also know that given a Boolean expression over $\mathcal{B}$, there is a circuit corresponding to it. Now, you may ask:

Given a Boolean function $f : \mathcal{B}^n \to \mathcal{B}$, is it always possible to get a Boolean expression which will specify f over B? The answer is `yes', i.e., for every function $f : \mathcal{B}^n \to \mathcal{B}$ ($n \geq 2$) there is a Boolean expression (in n variables) whose Boolean function is f itself.

To help you understand the underlying procedure, consider the following examples.

Example 10: Let $f : \mathcal{B}^2 \to \mathcal{B}$ be a function which is defined by

$$f(0, 0) = 1, f(1, 0) = 0, f(0, 1) = 1, f(1, 1) = 1.$$

Find the Boolean expression specifying the function f.

Solution: f can be represented by the following table.

Input		Output
x_1	x_2	$f(x_1, x_2)$
0	0	1
1	0	0
0	1	1
1	1	1

We find the Boolean expression according to the following algorithm:

In Boolean algebra terminology this is known as the 'disjunctive normal form' (DNF) of the expression.

Step 1: Identify all rows of the table where the output is 1: these are the 1st, 3rd and 4th rows.

Step 2: Combine the variables in each of the rows identified in Step 1 with `and'. Simultaneously, apply `not' to the variables with value zero in these rows. So, for the

1st row: $x'_1 \wedge x'_2$,

3rd row: $x'_1 \wedge x_2$,
4th row: $x_1 \wedge x_2$.

Step 3: Combine the Boolean expressions obtained in Step 2 with `or' to get the compound expression representing f:
So, $f(x_1 , x_2) = (x'_1 \wedge x'_2) \vee (x'_1 \wedge x_2) \vee (x_1 \wedge x_2)$.

***

You can complete Example 10, by doing the following exercise.

E12) In the previous example, show that $X(e_1 , e_2) = f(e_1 , e_2) \; \forall e_1 , e_2 \in \mathcal{B}$.

E13) By Theorem 2, we could also have obtained the expression of f in Example 10 in `conjunctive normal form' (CNF). Please do so.

An important remark: To get a Boolean expression for a Boolean function h (say), we should first see how many points v_i there are at which $h(v_i) = 0$, and how many points v_i there are at which $h(v_i) = 1$. **If the number of values for which the function h is 0 is less than the number of values at which h is 1, then we shall choose to obtain the expression in CNF, and not in DNF.** This will give us a shorter Boolean expression, and hence, a simpler circuit. For similar reasons, we will prefer DNF if the number of values at which h is 0 is more.

Why don't you apply this remark now?

E14) Find the Boolean expressions, in DNF or in CNF (keeping in mind the remark made above), for the functions defined in tabular form below.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	1

(a)

x_1	x_2	x_3	$g(x_1, x_2, x_3)$
1	1	1	1
1	1	0	1
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	1
0	0	0	1

(b)

Boolean functions tell us about the functioning of the corresponding circuit. Therefore, circuits represented by two equivalent expressions should essentially do the same job. We use this fact while redesigning a circuit to create a simpler one. In fact, in such a simplification process of a circuit, we write an expression for the circuit and then evaluate the same (over two-element Boolean algebra $\mathcal{B}$) to get the Boolean function. Next, we proceed to get an equivalent, simpler expression. Finally, the process terminates with the construction of the circuit for this simpler expression. Note that, **as the two expressions are equivalent, the circuit represented by the simpler expression will do exactly the same job as the circuit represented by the original expression.**

Let us illustrate this process by an example in some detail.

Example 11: Design a logic circuit capable of operating a central light bulb in a hall by three switches x_1 , x_2 , x_3 (say) placed at the three entrances to that hall.

Solution: Let us consider the procedure stepwise.
Step 1: To obtain the function corresponding to the unspecified circuit.

To start with, we may assume that the bulb is off when all the switches are off. Mathematically, this demands a situation where $x_1 = x_2 = x_3 = 0$ implies $f(0, 0, 0)= 0$, where f is the function which depicts the functional utility of the circuit to be designed.
Let us now see how to obtain the other values of f. Note that every change in the state of a switch should alternately put the light bulb on or off. Using this fact repeatedly, we obtain the other values of the function f.

Now, if we assign the value (1,0,0) to (x_1 , x_2 , x_3), it brings a single change in the state of the switch x_1 only. So, the light bulb must be on. This can be written mathematically in the form $f(1, 0, 0) = 1$. Here the value 1 of f stands for the on state of the light bulb.
Then, we must have $f(1, 1, 0) = 0$, because there is yet another change, now in the state of switch x_2 .

You can verify that the other values of $f(x_1 , x_2 , x_3)$ are given as in Table 9.

Table 9: Function of a circuitry for a three-point functional bulb.

x_1	x_2	X_3	$f(x_1, x_2, x_3)$
0	0	0	0
1	0	0	1
1	1	0	0
1	1	1	1
0	1	0	1
0	1	1	0
0	0	1	1
1	0	1	0

Step 2: To obtain a Boolean expression which will specify the function f. Firstly, note that the number of 1's in the last column of Table 9 are fewer than the number of 0's. So we shall obtain the expression in DNF (instead of CNF).

By following the stepwise procedure of Example 10, you can see that the required Boolean expression is given by

$$X(x_1 , x_2 , x_3) = (x_1 \wedge x'_2 \wedge x_3) \vee (x'_1 \wedge x_2 \wedge x'_3) \vee (x'_1 \wedge x'_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3)$$

At this stage we can directly jump into the construction of the circuit for this expression (using methods discussed in Sec.3.3). But why not try to get a simpler circuit?

Step 3 : To simplify the expression $X(x_1 , x_2 , x_3)$ given above. Firstly, observe that

$$\begin{aligned}(x_1 \wedge x'_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3) &= x_1 \wedge [(x'_2 \wedge x_3) \vee (x_2 \wedge x_3)] \\ &= x_1 \wedge [(x'_2 \vee x_2) \wedge x_3] \\ &= x_1 \wedge (1 \wedge x_3) \\ &= x_1 \wedge x_3 ,\end{aligned}$$

by using distributive, complementation and identity laws (in that order).
Similarly, you can see that

$$(x'_1 \wedge x'_2 \wedge x_3) \vee (x_1 \wedge x_3) = (x'_2 \vee x_1) \wedge x_3 .$$

We thus have obtained a simpler (and equivalent) expression, namely,

$$X(x_1 , x_2 , x_3) = (x'_1 \wedge x_2 \wedge x'_3) \vee [(x'_2 \vee x_1) \wedge x_3] ,$$

whose Boolean function is same as the function f. (Verify this!)

Step 4: To design a circuit for the expression obtained in Step 3.

Now, the logic circuit corresponding to the simpler (and equivalent) expression

obtained in Step 3 is as shown in Fig.15.

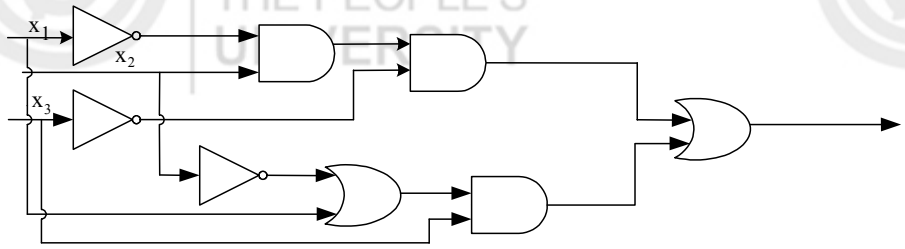


Fig. 15: A circuit for the expression $(x'_1 \wedge x_2 \wedge x'_3) \vee ((x'_2 \vee x_1) \wedge x_3)$

So, in 4 steps we have designed a 3-switch circuit for the hall.

***

We can't claim that the circuit designed in the example above is the simplest circuit. How to get that is a different story and is beyond the scope of the present course.

Why don't you try an exercise now?

E15) Design a logic circuit to operate a light bulb by two switches, x_1 and x_2 (say).

We have now come to the end of our discussion on applications of logic. Let us briefly recapitulate what we have discussed here.

3.5 SUMMARY

In this unit, we have considered the following points.

1. The definition and examples of a Boolean algebra. In particular, we have discussed the two-element Boolean algebra $\mathcal{B} = \{0, 1\}$, and the switching algebras B_n, $n \geq 2$.
2. The definition and examples of a Boolean expression.
3. The three elementary logic gates, namely, **AND**-gate, **OR**-gate and **NOT**-gate; and the analogy between their functioning and operations of logical connectives.
4. The method of construction of a logic circuit corresponding to a given Boolean expression, and vice-versa.
5. How to obtain the logic table of a Boolean expression, and its utility in the understanding of the overall functioning of a circuit.
6. The method of simplifying a Boolean expression.
7. The method of construction of a Boolean function $f : \mathcal{B}^n \rightarrow B$, corresponding to a Boolean expression, and the concept of **equivalent** Boolean expressions.
8. Examples of the use of Boolean algebra techniques for constructing a logic circuit which can function in a specified manner.

3.6 SOLUTIONS/ ANSWERS

E1) a) In E19 of Unit 1, you have already verified the Identity laws. Let us proceed to show that the propositions $p \vee (p \wedge q)$ and p are logically equivalent. It suffices to show that the truth tables of both these propositions are the same. This follows from the first and last columns of the following table.

p	q	$p \wedge q$	$p \vee (p \wedge q)$
F	F	F	F
F	T	F	F
T	F	F	T
T	T	T	T

Similarly, you can see that the propositions $p \wedge (p \vee q)$ and p are equivalent propositions. This establishes the absorption laws for the Boolean algebra $(S, \wedge, \vee', \mathcal{T}, \mathcal{F})$.

b) Let A and B be two subsets of the set X. Since $A \cap B \subseteq A$, $(A \cap B) \cup A = A$. Similarly, as $A \subseteq A \cup B$, we have $(A \cup B) \cap A = A$. Thus, both the forms of the absorption laws hold good for the Boolean algebra $(\mathcal{P}(X), \cup, \cap, ^c, X,)$.

E2) We can write

$$X(x_1, x_2, x_3) = ((x_1 \wedge x_2) \vee ((x_1 \wedge x_2) \wedge x_3)) \vee (x_2 \wedge x_3)$$
$$= (x_1 \wedge x_2) \vee (x_2 \wedge x_3) \quad \text{(by Absorption law)}$$
$$= x_2 \wedge (x_1 \vee x_3) \quad \text{(by Distributive law)}$$

This is the simplest form of the given expression.

E3) Take the propositions p and q in place of the bits x_1 and x_2, respectively. Then, when 1 and 0 are replaced by T and F in Table 3 here, we get the truth table for the proposition $p \wedge q$ (see Table 2 of Unit 1).
This establishes the analogy between the functioning of the **AND**-gate and the conjunction operation on the set of propositions.

E4) Take the propositions p and q in place of the bits x_1 and x_2, respectively. Then, when 1 and 0 are replaced by T and F in Table 4 here, we get the truth table for the proposition $p \vee q$ (see Table 1 of Unit 1).
This establishes the analogy between the functioning of the **OR**-gate and the disjunction operation on the set of propositions.

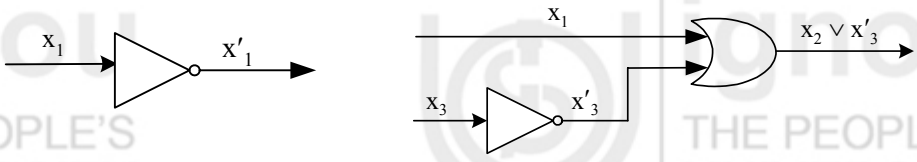
E5) Firstly, observe that the information about the outputs of the three elementary gates, for different values of inputs, can also be written as follows:

$0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0$, $1 \wedge 1 = 1$; (see Table 3)
$0 \vee 0 = 0$, $0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$; and (see Table 4)
$0' = 1$, $1' = 0$. (see Table 5)

Clearly, then both the operations # and # are the binary operations on $\mathcal{B}$ and $' : \mathcal{B} \to \mathcal{B}$ is a unary operation. Also, we may take 0 for **O** and 1 for **I** in the definition of a Boolean algebra.
Now, by looking at the logic tables of the three elementary gates, you can see that all the five laws B1-B5 are satisfied. Thus, $\mathcal{B}$ is a Boolean algebra.

E6) a) Here x_1 and x_2 are inputs to an **OR**-gate, and so, we take $x_1 \vee x_2$ as input to the **NOT**-gate next in the chain which, in turn, yields $(x_1 \vee x_2)'$ as the required output expression for the circuit given in (a).
b) Here x_1 and x_2 are the inputs to an **AND**-gate. So, the expression $x_1 \wedge x_2$ serves as an input to the **NOT**-gate, being next in the chain.
This gives the expression $(x_1 \wedge x_2)'$ which serves as one input to the extreme right **AND**-gate. Also, since x'_3 is another input to this **AND**-gate (coming out of a **NOT**-gate), we get the expression $(x_1 \wedge x_2)' \wedge x'_3$ as the final output expression which represents the circuit given in (b).

E7) You know that the circuit representing expressions x_1 and $x_2 \vee x'_3$ are as shown in Fig.16 (a) and (b) below.



(a) (b)

Fig. 16

Thus, the expression $x'_1 \vee (x_2 \vee x'_3)$, being connected by the symbol $\wedge$, gives the circuit corresponding to it as given in Fig.17 below.

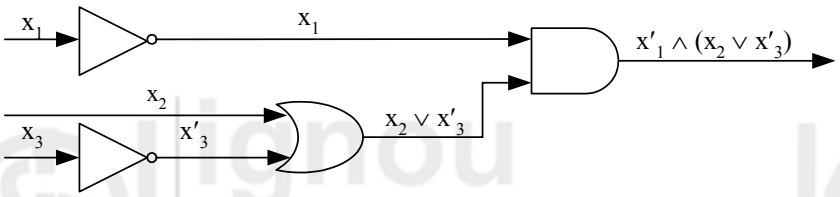


Fig. 17: A logic circuit for the expression $x'_1 \wedge (x_2 \vee x'_3)$

E8) You can easily see, by following the arguments given in E9, that the circuit represented by the expression $x_1 \vee (x'_2 \wedge x_3)$ is as given in Fig.18.

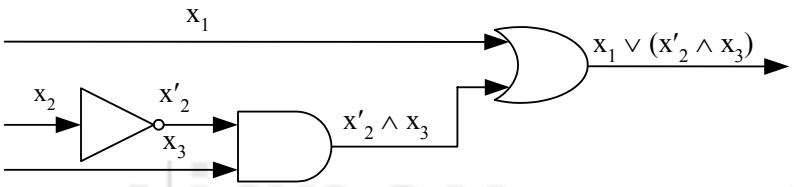


Fig. 18

The logic table of this expression is as given below.

x_1	x_2	x_3	x'_2	$x'_2 \wedge x_3$	$X_1 \vee (x'_2 \wedge x_3)$
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	0	0	0
1	0	0	1	0	1
0	1	1	0	0	0
1	1	0	0	0	1
1	0	1	1	1	1
1	1	1	0	0	1

E9) Since the output expression representing the circuit given in E8(b) is found to be $(x_1 \wedge x_2)' \wedge x'_3$, the logic table for this circuit is as given below.

x_1	x_2	x_3	$x_1 \wedge x_2$	$(x_1 \wedge x_2)'$	x'_3	$(x_1 \wedge x_2)' \wedge x'_3$
0	0	0	0	1	1	1
0	0	1	0	1	0	0
0	1	0	0	1	1	1
1	0	0	0	1	1	1
0	1	1	0	1	0	0
1	1	0	1	0	1	0
1	0	1	0	1	0	0
1	1	1	1	0	0	0

E10) Because the expression $(x_1 \wedge x_2) \vee (x_1 \wedge x'_3)$ involves three variables, the

corresponding Boolean function, f (say) is a three variable function, i.e. f : $B_3 \to B$. It is defined by

$f(e_1, e_2, e_3) = (e_1 \wedge e_2) \vee (e_1 \wedge e'_3)$, e_1, e_2 and $e_3 \in B$.

Now, you can verify that the values of f in tabular form are as given in the following table.

e_1	e_2	E_3	$e_1 \wedge e_2$	e'_3	$e_1 \wedge e'_3$	$f(e_1, e_2, e_3) = (e_1 \wedge e_2) \vee (e_1 \wedge e'_3)$
0	0	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	0	1	0	0
1	0	0	0	1	1	1
0	1	1	0	0	0	0
1	1	0	1	1	1	1
1	0	1	0	0	0	0
1	1	1	1	0	0	1

E11) To show that the Boolean expressions X and Y are equivalent over the two-element Boolean algebra $B = \{0, 1\}$, it suffices to show that the Boolean functions f and g (say) corresponding to the expressions X and Y, respectively, are the same. As you can see, the function f for the expression X is calculated in E10 above.
Similarly, you can see that the Boolean function g for the expression Y in tabular form is as given below.

x_1	x_2	x_3	x'_3	$x_2 \vee x'_3$	$G(x_1, x_2, x_3) = X_1 \wedge (x_2 \vee x'_3)$
0	0	0	1	1	0
0	0	1	0	0	0
0	1	0	1	1	0
1	0	0	1	1	1
0	1	1	0	1	0
1	1	0	1	1	1
1	0	1	0	0	0
1	1	1	0	1	1

Comparing the last columns of this table and the one given in E10 above, you can see that $f(e_1, e_2, e_3) = g(e_1, e_2, e_3)$ $\forall$ $e_1, e_2, e_3 \in B = \{0, 1\}$. Thus, **X** and **Y** are equivalent.

E12) Firstly, let us evaluate the given expression $X(x_1, x_2)$ over the two-element Boolean algebra $B = \{0, 1\}$ as follows:

$X(0, 0) = (0' \wedge 0') \vee (0' \wedge 0) \vee (0 \wedge 0)$
$= (1 \wedge 1) \vee (1 \wedge 0) \vee (0 \wedge 0)$
$= 1 \vee 0 \vee 0 = 1 = f(0, 0);$

$X(1, 0) = (1' \wedge 0') \vee (1' \wedge 0) \vee (1 \wedge 0)$
$= (0 \wedge 1) \vee (0 \wedge 0) \vee (1 \wedge 0)$
$= 0 \vee 0 \vee 0 = 0 = f(1, 0);$

$X(0, 1) = (0' \wedge 1') \vee (0' \wedge 1) \vee (0 \wedge 1)$
$= (1 \wedge 0) \vee (1 \wedge 1) \vee (0 \wedge 1)$
$= 0 \vee 1 \vee 0 = 1 = f(0, 1);$

$X(1, 1) = (1' \wedge 1') \vee (1' \wedge 1) \vee (1 \wedge 1)$
$= (0 \wedge 0) \vee (0 \wedge 1) \vee (1 \wedge 1)$

$= \quad 0 \vee 0 \vee 1 = 1 = f(1, 1).$

It thus follows that $\mathbf{X}(e_1, e_2) = f(e_1, e_2) \; \forall \; e_1, e_2 \in \boldsymbol{B} = \{0, 1\}$.

E13) **Step 1:** Identify all rows of the table where output is 0: This is the 2nd row.
Step 2: Combine x_1 and x_2 with `or' in these rows, simultaneously applying `not' to x_i if its value is 0 in the row: So, for the 2nd row the expression we have is $x_1 \vee x_2$.
Step 3: Combine all the expressions obtained in Step 2 with `and' to get the **CNF** form representing f: In this case there is only 1 expression.
So f is represented by $x_1 \vee x_2$ in CNF.

E14) a) Observe from the given table that, among the two values 0 and 1 of the function $f(x_1, x_2, x_3)$, the value 1 occurs the least number of times. Therefore, by the remark made after **E 13**, we would prefer to obtain the Boolean expression in **DNF**. To get this we will use the stepwise procedure adopted in Example 10.
Accordingly, the required Boolean expression in DNF is given by

$$X(x_1, x_2, x_3) = (x_1 \wedge x_2 \wedge x_3) \wedge (x_1 \wedge x'_2 \wedge x'_3) \vee (x'_1 \wedge x'_2 \wedge x'_3).$$

b) By the given table, among the two values 0 and 1 of the function the points v i at which $g(v_i) = 0$ are fewer than the points v_i at which $g(v_i) = 1$. So we would prefer to obtain the corresponding Boolean expression in CNF. Applying the stepwise procedure in the solution to E13, the required Boolean expression (in CNF) is given by

$$X(x_1, x_2, x_3) = (x'_1 \vee x_2 \vee x'_3) \wedge (x_1 \vee x'_2 \vee x'_3) \wedge (x_1 \vee x'_2 \vee x_3).$$

E15) Let g denote the function which depicts the functional utility of the circuit to be designed. We may assume that the light bulb is off when both the switches x_1 and x_2 are off, i.e., we write $g(0, 0) = 0$.
Now, by arguments used while calculating the entries of Table 9, you can easily see that all the values of the function g are as given below:

$$g(0, 0) = 0, \; g(0, 1) = 1, \; g(1, 0) = 1, \; g(1, 1) = 0.$$

Thus, proceeding as in the previous exercise, it can be seen that the Boolean expression (in DNF), which yields g as its Boolean function, is given by the expression

$$X(x_1, x_2) = (x'_1 \wedge x_2) \vee (x_1 \wedge x'_2),$$

because $g(0, 1) = 1$ and $g(1, 0) = 1$.

Finally, the logic circuit corresponding to this Boolean expression is shown in Fig. 19.

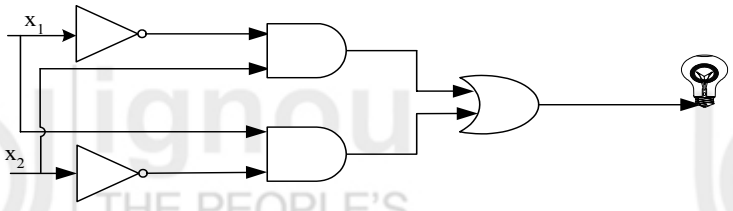


Fig. 19