CHAPTER 2

Propositional Logic & Hardware

In This Chapter

2.1 Introduction

Studying logic is important because it provides us a way to support our claims to truth. It is tempting to say that logic arguments establish the truth of their conclusions. As a field of study, we can define logic in many ways. We may say that logic is all about *arguments*. It is a formal method of reasoning. In the context of logic, an argument is not a quarrel or dispute, but an example of reasoning where a statement offers support, justification, ground, reason or evidence for another statement.

As earlier said that *logic is a formal method for reasoning*. Logic can be symbolically represented in many ways. One such way of doing so is *propositional logic*. This chapter is dedicated to the study of *propositional logic, digital logic and hardware*. Let us begin with the discussion of propositional logic.

2.2 Propositional Logic

The propositional logic represents logic through *propositions* and *logical connectives*. We may define ***proposition*** as an elementary atomic sentence that may take either ***true*** value or ***false*** value but may not take any other value.

Consider the following examples :

It is raining.	[It is a proposition as it may either be *true* or *false*]
Australia have won ICC World Cup 2007.	[It is also a proposition as it is *true*]
India is a continent.	[It is a proposition as it is *false*]
What did you eat ?	[It is not a proposition as it does not result in *true* or *false*]
How are you ?	[Not a proposition for the similar reason as above]

Propositions are also called *sentences* or *statements*. After this introduction, let us now talk about terms and symbols used in propositional logic.

A **Proposition** is an elementary atomic sentence that may either be true or false but may take no other value.

2.2.1 Terms and Symbols

A ***simple proposition*** is one that does not contain any other proposition as a part. We will use the lower-case letters, p, q, r, ..., as symbols for simple statements or propositions.

A ***compound proposition*** is one with two or more simple propositions as parts or what we will call ***components.*** A component of a compound is any whole proposition that is part of a larger proposition ; components may themselves be compounds.

For example, following are compound propositions :

It is raining *and* wind is blowing.
Take it *or* leave it.
If you work hard *then* you will be rewarded.

An **operator** (or *connective*) joins simple propositions into compounds, and joins compounds into larger compounds. We will use the symbols, +, ., $\Rightarrow$, and $\Leftrightarrow$ to designate the sentential connectives. They are called *sentential* connectives because they join *sentences* (or what we are calling *statements* or *propositions*). *The symbol, ~, is the only operator that is not a connective* ; it affects single statements only, and does not join statements into compounds.

The symbols for statements and for operators comprise our notation or symbolic language. Parentheses serve as punctuation.

Different types of **connectives** (or operators) used in propositional logic are as given below :

1. **Disjunctive** (Also called **OR**). Represented by symbols + or $\vee$. Disjunction means one of the two arguments is *true* or both *e.g.*, $p + q$ (or $p \vee q$) means **p OR q.** Its meaning is **either p is true, or q is true, or both.**
2. **Conjunctive** (Also called **AND**). Represented by symbols . or & or $\wedge$. Conjunction means both arguments are true *e.g.*, $p \cdot q$ (or $p \,\&\, q$) means **p AND q.** Its meaning is **both p and q are true.**
3. **Conditional** (Also called **If.. Then** or **Implication**). Represented by symbols $\Rightarrow$ or $\rightarrow$ or $\supset$. Implication means **if one argument is true then other argument is true** *e.g.*, $p \Rightarrow q$ (or $p \supset q$ or $p \rightarrow q$) means If p then q. Its meaning is **if p is true, then q is true.**
4. **Bi-conditional** (Also called **If and only If** or **Equivalence**). Represented by symbols $\Leftrightarrow$ or $\equiv$.

Equivalence (or bi-conditional) means **either both arguments are true or both are false,** *e.g.*, $p \Leftrightarrow q$ (or $p \equiv q$) means **if and only if p is true then q is true.** Its meaning is **p and q are either both true or both false.**

5. Negation (Also called **NOT**). Not a connective actually, just an operator. Represented by ~ or ′ or ¯ (bar). It is an operator that affects a single statement only and does not join two or more statements *e.g.*, $\sim p$ (or p' or $\bar{p}$) means **NOT p.** Its meaning is **p is false.**

We can summarize the above discussion as follows :

Simple statements

p	"p is true"	assertion
$\sim p$	"p is false"	negation

Compounds and connectives

$p + q$	"either p is true, or q is true, or both"	disjunction
$p \cdot q$	"both p and q are true"	conjunction
$p \Rightarrow q$	"if p is true, then q is true"	implication
$p \Leftrightarrow q$	"p and q are either both true or both false"	equivalence

Implication statements ($p \Rightarrow q$) are sometimes called *conditionals, equivalence* statements ($p \Leftrightarrow q$) are sometimes called *biconditionals.*

Well-formed Formulae

As mentioned earlier, propositions are also called sentences or statements. Another term *formulae or well-formed formulae* also refer to the same. That is, we may also call ***well formed-formula*** (***wff***) to refer to a proposition.

2.2.2 Truth values and Wff

Every simple or compound proposition may take an either *true value* or *false value*. These *true* (also denoted by 1) or *false* (also denoted by 0) are also called ***truth values***. We may define *truth value* as *truth* or *falsity* of a proposition.

The ***truth value*** of a statement is its truth or falsity. All meaningful statements have truth values, whether they are **simple** or **compound, asserted** or **negated**. That is, p is either true or false, $\sim p$ is either true or false, $p + q$ is either true or false, and so on.

Truth value is defined as truth or falsity of a proposition.

A compound statement is ***truth-functional*** if its truth value as a whole can be figured out solely on the basis of the truth values of its parts or components. A connective is truth-functional if it makes only compounds that are truth-functional. For example, if we knew the truth values of p and of q, then we could figure out the truth value of the compound, $p + q$. Therefore the compound, $p + q$, is a truth-functional compound and disjunction is a truth-functional connective.

All four of the connectives we are studying (disjunction, conjunction, implication, and equivalence) are truth-functional. Negation is a truth-functional operator. With these four connectives and negation we can express ***all*** the truth-functional relations among statements. A truth table helps us express it.

A **Truth Table** is a complete list of possible truth values of a proposition.

Let us now learn to make truth tables for all the connectives, we have learnt so far.

(i) Negation (NOT)

The NOT operator works on single proposition, thus, it is also called *unary connective* sometimes. If p denotes a proposition, then its negation will be denoted by $\sim p$ or p' or $\bar{p}$. If p is 0 (false), then $\sim p$ is 1 (true) and if p is 1 (true) then $\sim p$ is 0 (false). The truth table for this operation is shown as follows :

Table 2.1 *Truth table for Negation (NOT)*

p	$\bar{p}$
0	1
1	0

Also note that

NOT (NOT p) results into p itself *i.e.*,

$$\bar{\bar{p}} = p$$

or $(p')' = p$

or $\sim(\sim p) = p$.

(ii) Disjunction (OR)

The OR connective works with more than one proposition. The compound $p + q$ has two (2) component propositions (p and q), each of which can be true or false. So there are four (2^2) possible combinations. The disjunction of p with q (denoted as $p + q$ or $p \vee q$) will be true whenever p is true or q is true or both are true. Consider the truth table given below :

Table 2.2 *Truth table for Disjunction (OR)*

p	q	$p + q$
0	0	0
0	1	1
1	0	1
1	1	1

Note If a compound has n distinct components, there will be 2^n rows in its truth table.

(iii) Conjunction (AND)

The AND connective also works with more than one proposition. The compound $p \cdot q$ (or $p \& q$) will be *true* whenever both p and q are true.

Table 2.3 *Truth table for Conjunction (AND)*

p	q	$p \cdot q$
0	0	0
0	1	0
1	0	0
1	1	1

(iv) Implication (If.. Then / Conditional)

In the conditional $p \Rightarrow q$, the first proposition (the if-clause) p here, is called the ***antecedent*** and the second proposition (then clause) q here, is called the ***consequent***. In more complex *conditionals*, the *antecedent* and the *consequent* could themselves be *compound propositions*. The conditional $p \Rightarrow q$ will be *false* when p is *true* and q is *false*. For all other input combinations, it will be *true*.

Table 2.4 *Truth table for If.. Then*

p	q	$p \Rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

The conditional $p \Rightarrow q$ may be expressed as follows :

$$p \Rightarrow q = p' + q$$

(v) Equivalence (If and only If / Bi-conditional)

A bi-conditional results into *false* when one of its component propositions is *true* and the other is *false*. That is, $p \Leftrightarrow q$ will be 0 (false) when p *is 0 and* q is 1 Or p is 1 and q is 0. For all other inputs, $p \Leftrightarrow q$ is 1.

Table 2.5 *Truth table for If and only if*

p	q	$p \Leftrightarrow q$
0	0	1
0	1	0
1	0	0
1	1	1

The bi-conditional $p \Leftrightarrow q$ may also be expressed as :

$$p \Leftrightarrow q = pq + p' \cdot q'$$

Some Related Terms

Contingencies — The propositions that have some combination of 1's and 0's in their truth table column, are called *contingencies*.

Tautologies — The propositions having nothing but 1's in their truth table column, are called *tautologies*.

Contradictions — The propositions having nothing but 0's in their truth table column, are called *contradictions*.

Consistent Statements — Two statements are *consistent* if and only if their conjunction is *not a contradiction*.

Converse The converse of a conditional proposition is determined by interchanging the antecedent and consequent of given conditional. It results into a new conditional. *e.g.*, Converse of $p \Rightarrow q$ is $q \Rightarrow p$.

That is, if

p : It is raining.
q : Sky is not clear.

then, $p \Rightarrow q$ = *If it is raining then sky is not clear.*

Its converse will be new conditional as given below :

$q \Rightarrow p$ = *If sky is not clear then it is raining.*

Inverse The inverse of a conditional proposition is another conditional having negated *antecedent* and *consequent*. That is, the inverse of $p \Rightarrow q$ is $p' \Rightarrow q'$. *e.g.*, if

p : It is raining.
q : Sky is not clear.

then, $p \Rightarrow q$ = If it is raining then sky is not clear.

Its inverse will be a new conditional as given below :

$p' \Rightarrow q'$ = If it is not raining then sky is clear.

Contrapositive The contrapositive of a conditional is formed by creating another conditional that takes its *antecedent* as *negated consequent* of earlier conditional and *consequent* as *negated antecedent* of earlier conditional. That is, contrapositive of $p \Rightarrow q$ is $\sim q \Rightarrow \sim p$ or $\bar{q} \Rightarrow \bar{p}$ or $q' \Rightarrow p'$

2.2.3 Some Equivalence Propositional Laws

Two sentences are **equivalent** if they have the same truth value under every interpretation *i.e.*, both the sentences possess the same *truth set*.

In the following lines, we are giving some equivalence laws used in propositional logic. We are giving them without proofs, since their proofs are beyond the scope of this book.

Table 2.6 *Some Equivalence Laws*

1.	$0 + p = p$ $0 \cdot p = 0$	*Properties of 0*
2.	$1 + p = 1$ $1 \cdot p = p$	*Properties of 1*
3.	$p + pq = p$ $p + (p + q) = p$	*Absorption law*
4.	$\bar{\bar{p}} = p$	*Involution*
5.	$p + p = p$ $p \cdot p = p$	*Idempotence law*
6.	$p + \bar{p} = 1$ $p \cdot \bar{p} = 0$	*Complementarity law*
7.	$p + q = q + p$ $p \cdot q = q \cdot p$	*Commutative law*
8.	$(p + q) + r = p + (q + r)$ $(p \cdot q) \cdot r = p \cdot (q \cdot r)$	*Associative law*
9.	$p \cdot (q + r) = (p \cdot q) + (p \cdot r)$ $p + (q \cdot r) = (p + q) \cdot (p + r)$ $p + \bar{p}q = p + q$	*Distributive law*
10.	$\overline{p + q} = \bar{p} \cdot \bar{q}$ $\overline{p \cdot q} = \bar{p} + \bar{q}$	*De Morgan's law*
11.	$p \Rightarrow q = \bar{p} + q$	*Conditional elimination*
12.	$p \Leftrightarrow q = (p \Rightarrow q) \cdot (q \Rightarrow p)$	*Bi-conditional elimi.*

Let us now have a look at some examples.

Example 2.1. *Construct a truth table for the expression* $(A.(A+B))$. *What single term is the expression equivalent to ?*

Solution.

A	B	$A+B$	$(A\cdot(A+B))$
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Looking at the table, we find that columns $(A.(A+B))$ and A are identical. That is, they possess the same *truth set*. Hence the given expression $(A.(A+B))$ is equivalent to A.

Example 2.2. *Using truth table, prove that* $p \Rightarrow q$ *is equivalent to* $\sim q \Rightarrow \sim p$.

Solution.

p	q	$\sim q$	$\sim p$	$p \Rightarrow q$	$\sim q \Rightarrow \sim p$
0	0	1	1	1	1
0	1	0	1	1	1
1	0	1	0	0	0
1	1	0	0	1	1

From the above truth table it is obvious that columns $p \Rightarrow q$ and $\sim q \Rightarrow \sim p$ are identical *i.e.*, possessing same truth set (1, 1, 0, 1). Hence it is proved that

$$p \Rightarrow q = \sim q \Rightarrow \sim q.$$

This rule is also called *transposition*.

Example 2.3. *Prove that* $p \Rightarrow q = \bar{p} + q$.

Solution.

p	q	$\bar{p}$	$p \Rightarrow q$	$\bar{p} + q$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

From the above truth table, we find that columns $p \Rightarrow q$ and $\bar{p} + q$ are possessing same *truth set* (1, 1, 0, 1). Hence proved that

$$p \Rightarrow q \;=\; \bar{p} + q.$$

Example 2.4. *Prove that* $p \Leftrightarrow q = q \Leftrightarrow p$.

Solution.

p	q	$p \Leftrightarrow q$	$q \Leftrightarrow p$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	1	1

From the above truth table, we find that both propositions $p \Leftrightarrow q$ and $q \Leftrightarrow p$, possess the same truth set (1, 0, 0, 1). Hence proved that

$$p \Leftrightarrow q = q \Leftrightarrow p.$$

Example 2.5. *Prove that* $p \Leftrightarrow q = (p \Rightarrow q).(q \Rightarrow p)$.

Solution.

p	q	$p \Leftrightarrow q$	$p \Rightarrow q$	$q \Rightarrow p$	$(p \Rightarrow q).(q \Rightarrow p)$
0	0	1	1	1	1
0	1	0	1	0	0
1	0	0	0	1	0
1	1	1	1	1	1

Since the columns $p \Leftrightarrow q$ and $(p \Rightarrow q).(q \Rightarrow p)$ are identical, it is proved that

$$p \Leftrightarrow q = (p \Rightarrow q).(q \Rightarrow p)$$

Example 2.6. *Consider some simple propositions given below :*

A : It is raining. *B : Wind is blowing.* *C : I am not driving.*

From these, create the following compound proportions :

(*i*) $A \vee B$ (*ii*) $\sim B$ (*iii*) $\sim B.C$ (*iv*) $A.\sim C$ (*v*) $A + B.C$.

Solution.

(*i*) $A \vee B$: It is raining OR wind is blowing.

(*ii*) $\sim B$: Wind is NOT blowing.

(*iii*) $\sim A.C$: It is NOT raining AND I am not driving.

(*iv*) $A.\sim C$: It is raining AND I am driving.

(*v*) $A + B.C$: It is raining OR wind is blowing AND I am not driving.

Example 2.7. *Prove that* $X + 1$ *is a tautology.*

Solution.

X	1	$X + 1$
0	1	1
1	1	1

Since the column $X + 1$ has all trues (1's) in its column, it is a tautology.

Example 2.8. *Prove that* $X + X'$ *is a tautology and* $X.X'$ *is a contradiction.*

Solution.

X	X'	$X + X'$	$X.X'$
0	1	1	0
1	0	1	0

$X + X'$ has all 1's in its truth set, hence it is a tautology.

$X.X'$ has all 0's in its truth set, hence it is a contradiction.

2.2.4 Drawing Conclusions – Syllogism

While studying logic, many a times conclusions are drawn from given two or more logic statements. This process, rather logical process of drawing conclusions from given logic statements, is called ***syllogism***. The given statements or propositions are called ***premises***.

> The logical process of drawing conclusions from given propositions is called **syllogism.** The propositions used to draw conclusion are called **premises.**

To draw conclusions, we may use any of the *two* methods available for it :

- Truth Table Method
- Algebraic Method

1. Truth Table Method

In this method, a truth table (TT) is drawn for all the *given premises* and the *conclusion to be* drawn. Then a conditional is prepared having the *antecedent as conjunction of all given premises* and *consequent* as the *conclusion* to be drawn. If this conditional results into a tautology (all true's *i.e.*, 1's) then the given conclusion is established. For this consider the example given below :

Example 2.9. *From premises p and* $p \Rightarrow q$, *infer q.*

Solution. Given premises are :

$(P1)$: p

$(P2)$: $p \Rightarrow q$

Conclusion to be drawn $C : q$

As per the TT method, we have to prepare a truth table for given premises p, $p \Rightarrow q$, the conjunction of these given premises *i.e.*, $p.(p \Rightarrow q)$ and the conclusion to be drawn *i.e.*, q.

Also there should be a column for

$$P1 \;.\; P2 \Rightarrow C \;\; i.e., \; [p \;.\; (p \Rightarrow q)] \Rightarrow q$$

Our truth table will look like :

p	q	$p \Rightarrow q$	$p \;.\; (p \Rightarrow q)$	$[p \;.\; (p \Rightarrow q)] \Rightarrow q$
0	0	1	0	1
0	1	1	0	1
1	0	0	0	1
1	1	1	1	1

From the above truth table, it is clear that the conditional having *antecedent* as the conjunction of premises $[p.(p+q)]$ and *consequent* as the conclusion (q), is a tautology. Hence it is established that from given premises p and $p \Rightarrow q$ conclusion drawn is q.

That is

$$\frac{\begin{array}{l} p \\ p \Rightarrow q \end{array}}{q}$$

Example 2.10. From $p \Rightarrow q$ and $q \Rightarrow r$, infer $p \Rightarrow r$.

Solution. Given premises are :

(P1) : $p \Rightarrow q$

(P2) : $q \Rightarrow r$

Conclusion (C) to be drawn : $p \Rightarrow r$ as per TT method.

We shall prepare a truth table having column for

$P1$ (*i.e.*, $p \Rightarrow q$), $P2$ (*i.e.*, $q \Rightarrow r$), $P1 . P2$ *i.e.*, $(p \Rightarrow q) . (q \Rightarrow r)$,

C (*i.e.*, $p \Rightarrow r$) and $P1 . P2 \Rightarrow C$ *i.e.*, $[(p \Rightarrow q) . (q \Rightarrow r)] \Rightarrow (p \Rightarrow r)$

p	q	r	$p \Rightarrow q$	$q \Rightarrow r$	$(p \Rightarrow q) . (q \Rightarrow r)$	$p \Rightarrow r$	$[(p \Rightarrow q) . (q \Rightarrow r)] \Rightarrow (p \Rightarrow r)$
0	0	0	1	1	1	1	1
0	0	1	1	1	1	1	1
0	1	0	1	0	0	1	1
0	1	1	1	1	1	1	1
1	0	0	0	1	0	0	1
1	0	1	0	1	0	1	1
1	1	0	1	0	0	0	1
1	1	1	1	1	1	1	1

From the table, we derive that

$P1 . P2 \Rightarrow C$ is a tautology *i.e.*, having all 1's in its truth set. Hence concluded that

$$\frac{\begin{array}{l} p \Rightarrow q \\ q \Rightarrow r \end{array}}{p \Rightarrow r}$$

(*ii*) Algebraic Method

In this method, to draw a conclusion from given premises, **conditional elimination** is carried out. That is, in place of a conditional $p \Rightarrow q$ its equivalent $\bar{p} + q (\sim p + q)$ is substituted and then it is checked whether the conditional having *antecedent* as the *conjunction-of-all premises* and *consequent* as the *conclusion-to-be-drawn*, is a tautology or not. To understand this, consider the example given below.

Example 2.11. *From p and* $p \Rightarrow q$, *infer q.*

Solution. Given premises are

(P1): p

(P2): $p \Rightarrow q$

Conclusion (C) to be drawn : q.

Let us compute $P1 . P2 \Rightarrow C$

i.e., $[p . (p \Rightarrow q)] \Rightarrow q$

Carrying out conditional elimination *i.e.*, substituting $p \Rightarrow q$ with $\bar{p} + q$, we get

$= [p . (\bar{p} + q)] \Rightarrow q$

$= (p . \bar{p} + p . q) \Rightarrow q$

$= 0 + (p . q) \Rightarrow q$

$(\because p \bar{p} = 0)$

Carrying out conditional elimination once again, we get

$$\begin{aligned}
&\overline{(p.q)}+q \\
&=\bar{p}+\bar{q}+q && (\because \overline{pq}=\bar{p}+\bar{q}, \text{ Rule 10, table 2.6}) \\
&=\bar{p}+1 && (\because \bar{q}+q=1, \text{ Rule 6, table 2.6}) \\
&=\bar{p}+1 && (\because \bar{p}+1=\bar{p}, \text{ Rule 2, table 2.6}) \\
&=1
\end{aligned}$$

Hence the result is established.

Example 2.12. *From $p \Rightarrow q$ and $q \Rightarrow r$, infer $p \Rightarrow r$.*

Solution. Given premises are : $p \Rightarrow q$, $q \Rightarrow r$

and conclusion to be drawn is $p \Rightarrow r$.

Thus we have to establish that

$$[(p \Rightarrow q).(q \Rightarrow r)] \Rightarrow (p \Rightarrow r)$$

Carrying out conditional elimination, we get

$$[(\bar{p}+q).(\bar{q}+r)] \Rightarrow (\bar{p}+r)$$

Carrying out conditional elimination once again, we get

$$\begin{aligned}
&=\overline{(\bar{p}+q).(\bar{q}+r)}+(\bar{p}+r) \\
&=\overline{(\bar{p}+q)}+\overline{(\bar{q}+r)}+(\bar{p}+r) && \text{(De Morgan's Law)} \\
&=(\bar{\bar{p}}.\bar{q})+(\bar{\bar{q}}.\bar{r})+(\bar{p}+r) && \text{(De Morgan's Law)} \\
&=p.\bar{q}+q.\bar{r}+\bar{\bar{p}}+r \\
&=\bar{p}+p.\bar{q}+r+q.\bar{r} && \text{[Commutative law table 2.6]} \\
&=(\bar{p}+p).(\bar{p}+\bar{q})+(r+q)(r+\bar{r}) && \text{[Distributive law } p+qr=(p+q)(p+r)\text{]} \\
&=1.(\bar{p}+\bar{q})+(r+q).1. && [\because \bar{p}+p=r+\bar{r}=1] \\
&=\bar{p}+\bar{q}+r+q=\bar{p}+r+\bar{q}+q \\
&=p+r+1 && [\because \bar{q}+q=1] \\
&=p+1 && [\because r+1=1] \\
&=1 && [\because p+1=1]
\end{aligned}$$

Hence the result is established.

The inference rules established in above examples are also known as *Modus ponens* and *Chain rule*. That is, **Modus Ponens** is

$$\frac{p \qquad p \Rightarrow q}{q}$$

Chain rule is

$$\frac{p \Rightarrow q \qquad q \Rightarrow r}{p \Rightarrow r.}$$

Let Us Revise

- *Logic is a formal method for reasoning.*
- *A proposition is an elementary atomic sentence that may either be true or false but may take no other value.*
- *A simple proposition is one that does not contain any other proposition as a part.*
- *A compound proposition is one with two or more simple propositions as parts.*
- *A connective joins simple propositions into compounds and joins compounds into larger compounds.*
- *Different types of connectives are disjunctive, conjunctive, conditional, bi-conditional and negation.*
- *The negation is a unary connective.*
- *Well-formed formula (wff) refers to sentences or propositions or statements.*
- *Truth value is defined as truth or falsity of a proposition.*
- *A truth table is a complete list of possible truth values of a proposition.*
- *The logical process of drawing conclusion from propositions is called syllogism.*
- *The propositions used to draw conclusion are called premises.*
- *Two inference rules are Modus Ponens and Chain rule.*
- *Modus ponens is*

$$\frac{p \quad p \Rightarrow q}{q}$$

- *Chain rule is*

$$\frac{p \Rightarrow q \quad q \Rightarrow r}{p \Rightarrow r}$$

2.3 Basic Logic Gates

System of logic was constructed long long ago by Aristotle. Long ago Aristotle constructed a complete system of formal logic and wrote six famous works on the subject, contributing greatly to the organization of man's reasoning. For centuries afterward, mathematicians kept on trying to solve these logic problems using conventional algebra but only *George Boole* could manipulate these symbols successfully to arrive at a solution with his own mathematical system of logic. Boole's revolutionary paper *'An Investigation of the laws of the thought'* was published in 1854 which led to the development of new system, the *algebra of logic,* 'BOOLEAN ALGEBRA'.

Boole's work remained confined to papers only until 1938 when *Claude E. Shannon* wrote a paper titled *'A Symbolic Analysis of Relay Switching Circuits'*. In this paper he applied Boolean Algebra to solve relay logic problems. As logic problems are binary decisions and boolean algebra effectively deals with these binary values. Thus it is also called *'Switching Algebra'*.

After *Shannon* applied boolean algebra in telephone switching circuits, engineers realized that boolean algebra could be applied to computer electronics as well.

In the computers, these boolean operations are performed by logic gates.

What is a Logic Gate ?

Gates are digital (two-state) circuits because the input and output signals are either low voltage (denotes 0) or high voltage (denotes 1). Gates are often called *logic circuits* because they can be analyzed with boolean algebra.

There are *three* basic logic gates and *four* others :

1. Inverter (NOT gate)
2. OR gate
3. AND gate

A gate is simply an electronic circuit which operates on one or more signals to produce an output signal.

2.3.1 Inverter (NOT Gate)

An inverter is also called a NOT gate because the output is not the same as the input. The output is sometimes called the *complement* (opposite) of the input.

Following tables summarize the operation.

An Inverter is a gate with only one input signal and one output signal ; the output state is always the opposite of the input state.

Table 2.7 *Truth Table for NOT gate*

X	$\bar{X}$
Low	High
High	Low

Table 2.8 *Alternative truth table for NOT gate*

X	$\bar{X}$
0	1
1	0

A low input *i.e.*, 0 produces high output *i.e.*, 1, and vice versa. The symbol for inverter is given in adjacent Fig. 2.1.

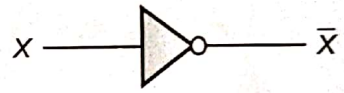


Figure 2.1 NOT gate symbol

2.3.2 OR Gate

If all inputs are 0 then output is also 0. If one or more inputs are 1, the output is 1.

An OR gate can have as many inputs as desired. No matter how many inputs are there, the action of OR gate is the same : one or more 1 (high) inputs produce output as 1. Following tables show OR action :

The OR gate has two or more input signals but only one output signal. If any of the input signals is 1 (high), the output signal is 1 (high).

Table 2.9 *Truth Table for two input OR gate*

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

$F = X + Y$

Table 2.10 *Truth Table for three input OR gate*

X	Y	Z	F
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

$F = X + Y + Z$

The symbol for OR gate is given below :

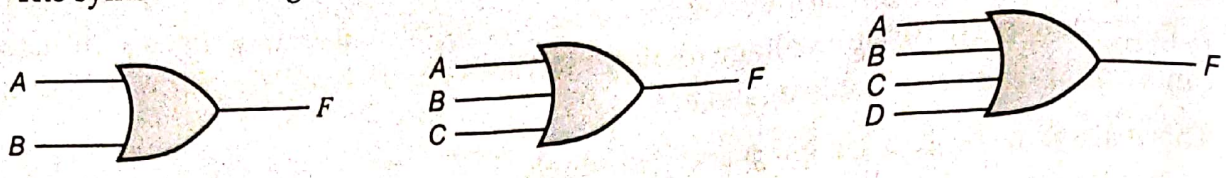


Figure 2.2 (*a*) Two input OR gate (*b*) Three input OR gate (*c*) Four input OR gate.

2.3.3 AND gate

If any of the inputs is 0, the output is 0. To obtain output as 1, all inputs must be 1.

An AND gate can have as many inputs as desired. Following tables illustrate AND action.

The **AND Gate** can have two or more than two input signals and produce one output signal. When all the inputs are 1 *i.e.*, high then the output is 1 otherwise output is 0.

Table 2.11 *Two input AND gate*

X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

Here, F = X . Y

Table 2.12 *Three input AND gate*

X	Y	Z	F
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

Here, F = X . Y . Z

The symbol for AND is

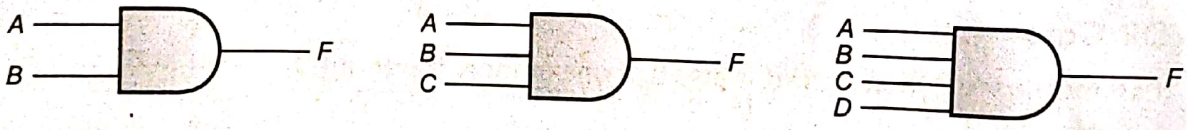


Figure 2.3 (*a*) 2-input AND gate (*b*) 3-input AND gate (*c*) 4-input AND gate

2.4 More About Logic Gates

We have covered *three* basic logic gates NOT, OR, AND so far. But there are some more logic gates also which are derived from three basic gates (*i.e.*, AND, OR and NOT). These gates are more popular than NOT, OR and AND and are widely used in industry. This section introduces NOR, NAND, XOR, XNOR gates.

2.4.1 NOR Gate

If either of the two input is 1 (*high*), the output will be 0 (*low*). NOR *gate is nothing but inverted* OR *gate.*

The NOR gate can have as many inputs as desired. No matter how many inputs are there, the action of NOR gate is the same *i.e.*, all 0 (*low*) inputs produce output as 1.

Following truth Tables (2.13 and 2.14) illustrate NOR action.

The **NOR gate** has two or more input signals but only one output signal. If all the inputs are 0 (*i.e.*, low), then the output signal is 1 (high)

Table 2.13 *2-input NOR gate*

X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

$F = \overline{X + Y}$

Table 2.14 *3-input NOR gate*

X	Y	Z	F
0	0	0	1
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	0

$F = \overline{X + Y + Z}$

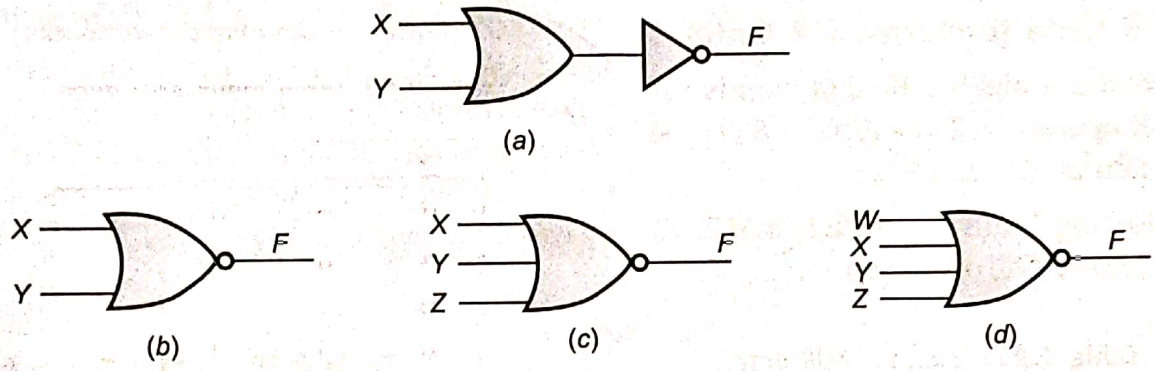


Figure 2.4 (*a*) Logical meaning of NOR gate (*b*) 2 input NOR gate (*c*) 3 input NOR gate (*d*) 4 input NOR gate

2.4.2 NAND Gate

NAND *gate is inverted* AND *gate.* Thus, for all 1 (*high*) inputs, it produces 0 (*low*) output, otherwise for any other input combination, it produces a 1 (*high*) output. NAND gate can also have as many inputs as desired.

The NAND gate has two or more input signals but only one output signal. If all of the inputs are 1 (high), then the output produced is 0 (low).

NAND action is illustrated in following Truth Tables (2.15 and 2.16).

Table 2.15 *2-input NAND gate's table*

X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

$F = \overline{XY}$

Table 2.16 *3-input NAND gate's table*

X	Y	Z	F
0	0	0	1
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	0

$F = \overline{XYZ}$

The logical meaning of NAND gate can be shown as follows :

The symbols of 2, 3, 4 input NAND gates are given below :

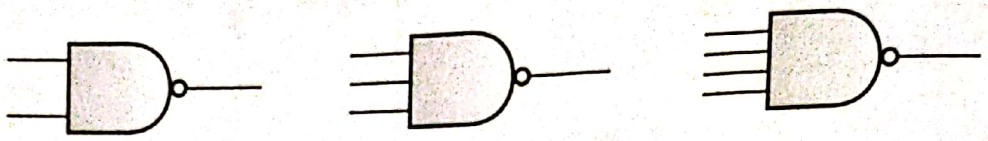


Figure 2.5 (*a*) 2-input NAND gate (*b*) 3-input NAND gate (*c*) 4-input NAND gate.

2.4.3 XOR Gate (Exclusive OR Gate)

In boolean algebra ⊕ sign stands for XOR operation. Thus *A* XOR *B* can be written as $A \oplus B$.

Following Truth Tables (2.17 and 2.18) illustrate XOR operation.

The XOR Gate can also have two or more inputs but produces one output signal. Exclusive-OR gate is different from OR gate. OR gate produces output 1 for any input combination having one or more 1's, but XOR gate produces output 1 for only those input combinations that have odd number of 1's.

Table 2.17 *2-intput XOR gate.*

No. of 1's even/odd	*X*	*Y*	*F*
Even	0	0	0
Odd	0	1	1
Odd	1	0	1
Even	1	1	0

Table 2.18 *3-input XOR gate.*

No. of 1's	*X*	*Y*	*Z*	*F*
Even	0	0	0	0
Odd	0	0	1	1
Odd	0	1	0	1
Even	0	1	1	0
Odd	1	0	0	1
Even	1	0	1	0
Even	1	1	0	0
Odd	1	1	1	1

Note Remember odd number of 1's produce output 1 in XOR operation.

The symbols of XOR gates are given below :

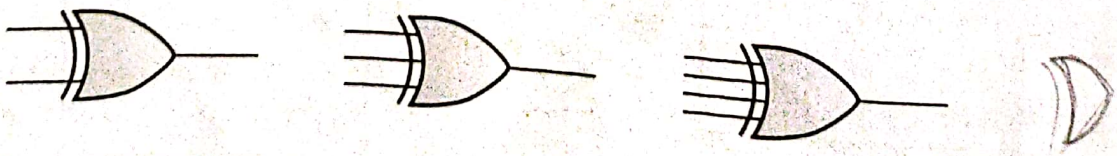


Figure 2.6 (*a*) 2-input XOR gate (*b*) 3-input XOR gate (*c*) 4-input XOR gate.

XOR addition can be summarized as follows :

$0 \oplus 0 = 0$; $0 \oplus 1 = 1$; $1 \oplus 0 = 1$; $1 \oplus 1 = 0$

2.4.4 XNOR Gate (Exclusive NOR gate)

The XNOR Gate is logically equivalent to an inverted XOR i.e., XOR gate followed by a NOT gate (inventor). Thus XNOR produces 1 (high) output when the input combination has even number of 1's.

Following Tables (2.19 and 2.20) illustrate XNOR action.

Table 2.19 *2 input XNOR gate*

No. of 1's even/odd	X	Y	F
Even	0	0	1
Odd	0	1	0
Odd	1	0	0
Even	1	1	1

Table 2.20 *3-input XNOR gate*

No. of 1's	X	Y	Z	F
Even	0	0	0	1
Odd	0	0	1	0
Odd	0	1	0	0
Even	0	1	1	1
Odd	1	0	0	0
Even	1	0	1	1
Even	1	1	0	1
Odd	1	1	1	0

Following are the XNOR gate symbols :

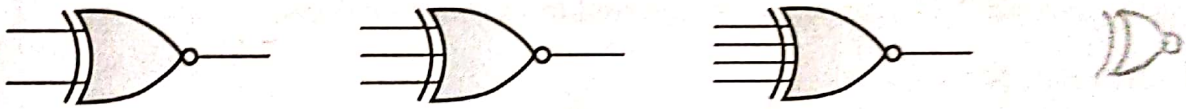


Figure 2.7 (*a*) 2-input XNOR gate (*b*) 3-input XNOR gate (*c*) 4-input XNOR gate.

The bubble (small circle), on the outputs of NAND, NOR, XNOR gates represents complementation.

Now that you are familiar with logic gates, you can use them in designing logic circuits.

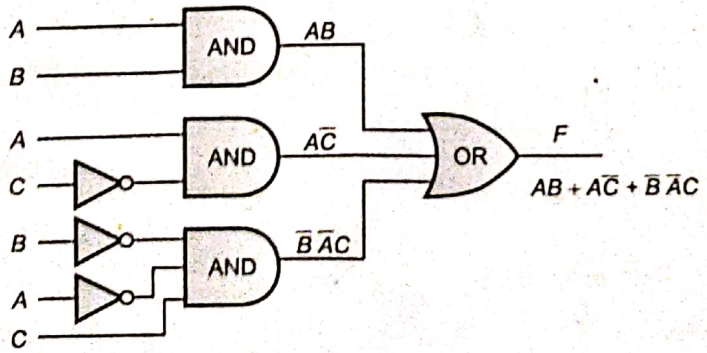
Example 2.13. *Design a circuit to realize the following :* $F(a, b, c) = AB + A\overline{C} + \overline{B}\,\overline{A}C$

Solution. The given boolean expression can also be written as follows :

$$F(a, b, c) = A.B + A.\overline{C} + \overline{B}.\overline{A}.C$$

or $F(a, b, c) =$ (A AND B) OR (A AND (NOT C)) OR ((NOT B) AND (NOT B) AND C)

Now these logical operators can easily be implemented in form of logic gates. Thus circuit diagram for above expression will be as follows.



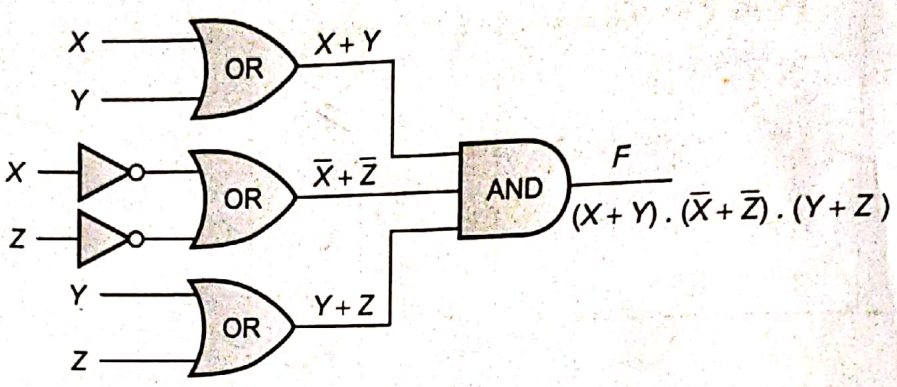
Example 2.14. *Draw the diagram of digital circuit for the function*

$$F(X, Y, Z) = (X + Y).(\overline{X} + \overline{Z}).(Y + Z)$$

Solution. Above expression can also be written as

$F(X, Y, Z)$ = (X OR Y) AND ((NOT X) OR (NOT Z)) AND (Y OR Z)

Thus circuit diagram will be



2.4.5 NAND to NAND and NOR to NOR Design

We can design circuits using AND, OR, NOT gates as we have done so far, but NAND and NOR gates are more popular as these are less expensive and easier to design. And also other switching functions (AND, OR) can easily by implemented using NAND/NOR gates. Thus NAND, NOR gates are also referred to as *Universal Gates.*

NAND-to-NAND Logic

AND and OR operations from NAND gates are shown below :

AND Operation

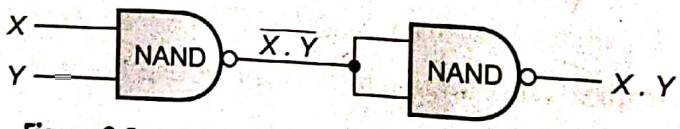


Figure 2.8 AND operation using NAND-NAND logic.

AND operation using NAND is

$$X.Y = (X \text{ NAND } Y) \text{ NAND } (X \text{ NAND } Y)$$

Proof. X NAND $Y = \overline{X.Y}$

$= \overline{X} + \overline{Y}$

(X NAND Y) NAND (X NAND Y) (*De Morgan's Second Theorem*)

$= (\overline{X} + \overline{Y})$ NAND $(\overline{X} + \overline{Y})$

$= \overline{(\overline{X} + \overline{Y}).(\overline{X} + \overline{Y})}$

$= (\overline{\overline{X} + \overline{Y}}) + (\overline{\overline{X} + \overline{Y}})$

$= \overline{\overline{X}}.\overline{\overline{Y}} + \overline{\overline{X}}.\overline{\overline{Y}}$ (*De Morgan's Second Theorem*)

$= X.Y + X.Y$ (*De Morgan's First Theorem*)

$= XY$ ($\overline{\overline{X}} = X$)

($X + X = X$)

OR Operation

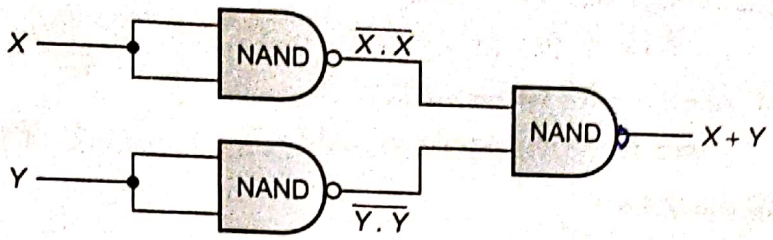


Figure 2.9 OR operation usign NAND to NAND logic.

OR operation using NAND is

$$X + Y = (X \text{ NAND } X) \text{ NAND } (Y \text{ NAND } Y)$$

Proof. $X \text{ NAND } X = \overline{X.X}$

$= \overline{X} + \overline{X}$ (*De Morgan's Second Theorem*)

$= \overline{X}$ $(X + X = X)$

Similarly, $Y \text{ NAND } Y = \overline{Y}$

Therefore, $(X \text{ NAND } X) \text{ NAND } (Y \text{ NAND } Y)$

$= \overline{X} \text{ NAND } \overline{Y}$

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

$= \overline{\overline{X}} + \overline{\overline{Y}}$ (*De Morgan's Second Theorem*)

$= X + Y$ $(\overline{\overline{X}} = X, \overline{\overline{Y}} = Y)$

Not Operation

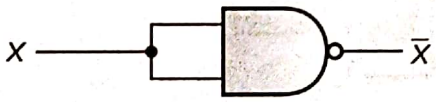


Figure 2.10 NOT operation using NAND logic.

NOT operation using NAND gates is $\text{NOT } X = X \text{ NAND } X$

Proof. $X \text{ NAND } X = \overline{X.X} = \overline{X}$ $(\because X.X = X)$

NAND-to-NAND logic is best suited for boolean expression in *Sum-of-Products* form.

Design rule for NAND-TO-NAND logic Network (only for two-level-circuits)

1. Derive simplified sum-of-products expression.
2. Draw a circuit diagram using AND, OR gates
3. Just replace AND and OR gates with NAND gates

For example, $XY\overline{Z} + Z\overline{X}$ can be drawn as follows :

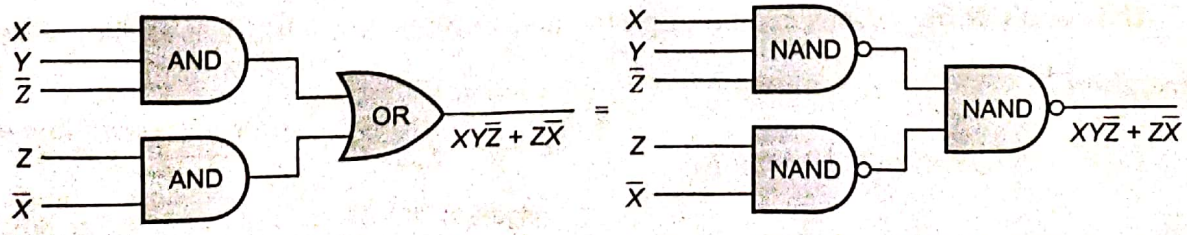


Figure 2.11 (*a*) AND-to-OR implementation. (*b*) NAND-to-NAND implementation.

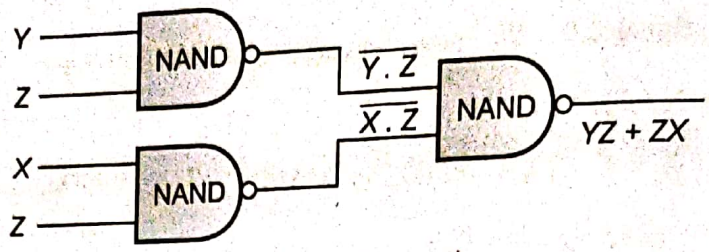
Example 2.15. *Draw the diagram of a digital circuit for the function*

$$F(X, Y, Z) = YZ + XZ \text{ using NAND gates only.}$$

Solution. $F(X, Y, Z) = YZ + XZ$ can be written as

$$= (Y \text{ NAND } Z) \text{ NAND } (X \text{ NAND } Z)$$

Thus logic circuit diagram is



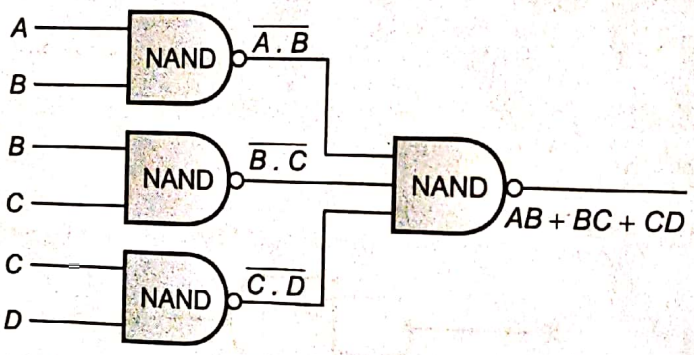
Example 2.16. *Draw the diagram of digital circuit for*

$$F(a, b, c) = AB + BC + CD \text{ using NAND-to-NAND logic.}$$

Solution. $F(a, b, c) = AB + BC + CD$

$$= (A \text{ NAND } B) \text{ NAND } (B \text{ NAND } C) \text{ NAND } (C \text{ NAND } D)$$

Thus logic circuit is



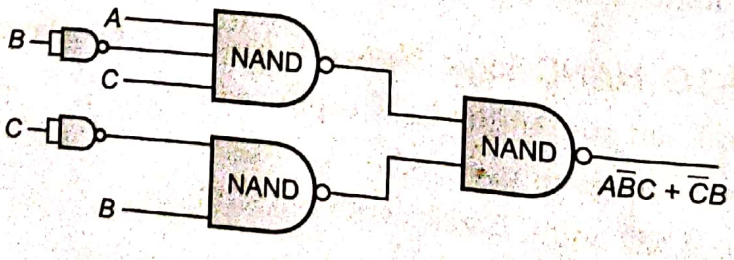
Example 2.17. *Draw the circuit diagram for*

$$F = A\overline{B}C + \overline{C}B \text{ using NAND-to-NAND logic only.}$$

Solution. $F = A\,\overline{B}\,C + \overline{C}\,B$

$$= ((A) \text{ NAND (NOT } B) \text{ NAND (C)) NAND ((NOT C) NAND } B)$$

Thus logic circuit is



NOR-to-NOR Logic

AND and OR operations can be implemented in NOR-to-NOR form as shown below :

OR operation

$$A + B = (A \text{ NOR } B) \text{ NOR } (A \text{ NOR } B)$$

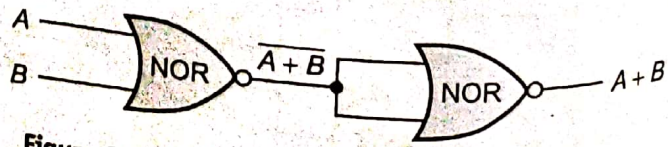


Figure 2.12 OR operation using NOR to NOR logic.

AND operation

$A \cdot B = (A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B)$

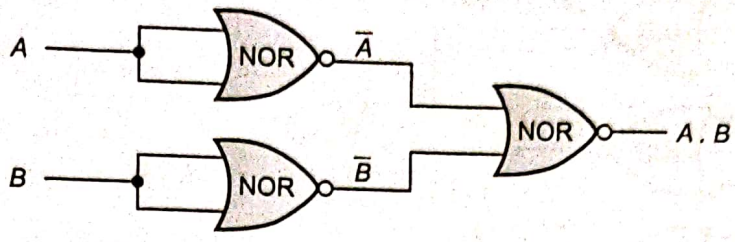


Figure 2.13 AND operation using NOR to NOR logic.

NOT Operation

NOT $A = A$ NOR A

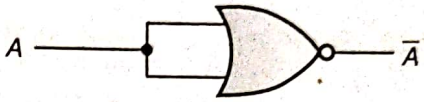


Figure 2.14 NOT operation using NOR Logic

NOR-to-NOR *logic is best suited for boolean expression in Product-of-Sums form.*

Design rule for NOR-to-NOR logic network (only for 2-level-circuits)

1. Derive a simplified Product-of-Sums form of the expression.
2. Draw a circuit diagram using OR, AND gates
3. Finally substitute NOR gates for OR, NOT and AND gates

For example, $(X + Y)(Y + Z)(Z + X)$ can be implemented as follows :

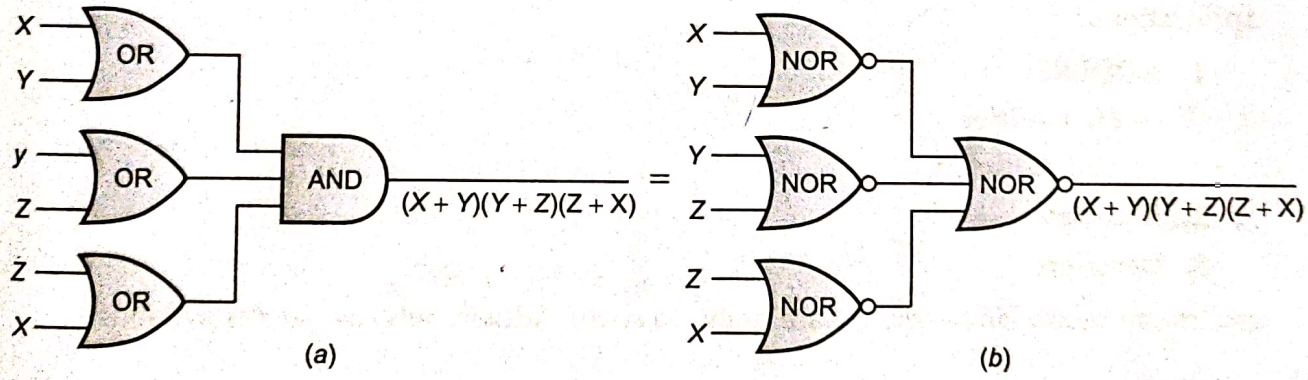
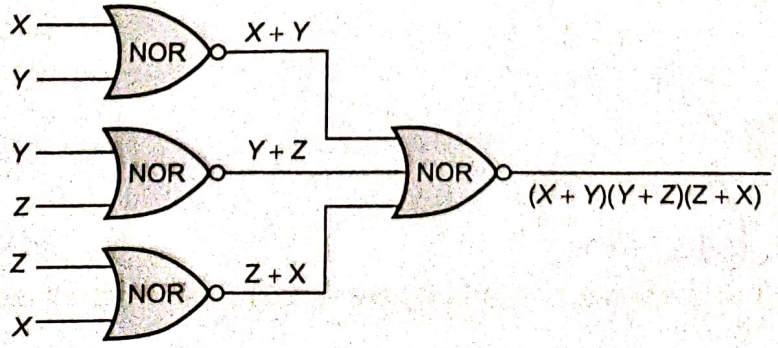


Figure 2.15 (a) OR-to-AND implementation (*b*) NOR-to-NOR circuit.

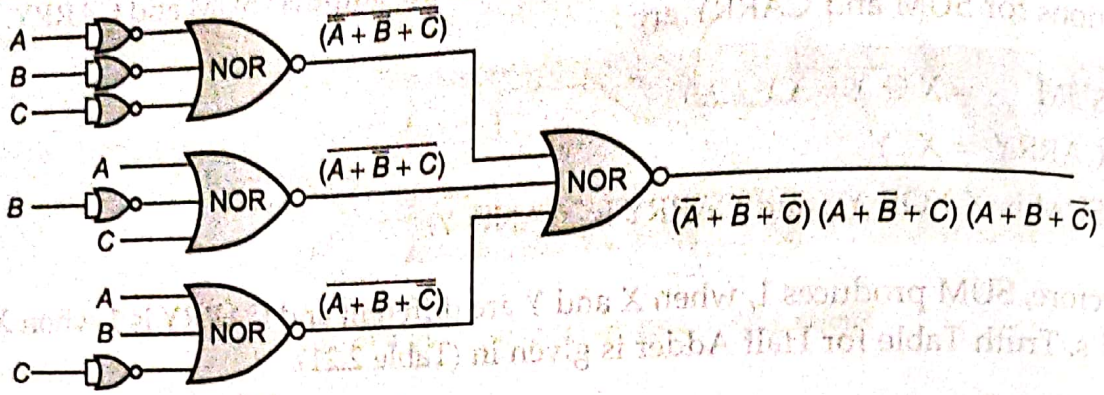
Example 2.18. *Represent* $(X + Y)(Y + Z)(Z + X)$ *in NOR-to-NOR form.*

Solution. $(X + Y)(Y + Z)(Z + X) = (X \text{ NOR } Y) \text{ NOR } (Y \text{ NOR } Z) \text{ NOR } (Z \text{ NOR } X)$



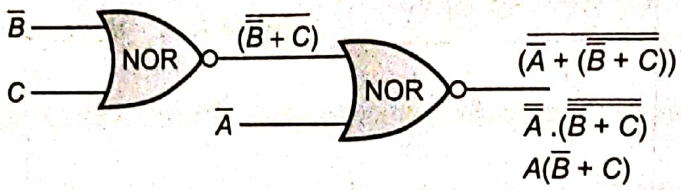
Example 2.19. *Represent* $(\overline{A}+\overline{B}+\overline{C})(A+\overline{B}+C)(A+B+\overline{C})$ *in NOR-to-NOR logic network.*

Solution.



Example 2.20. *Show* $A(\overline{B}+C)$ *using NOR gates only.*

Solution.



2.5 Applications of Logic Gates

The design and maintenance of digital computers are greatly facilitated by the use of boolean algebra and logic circuits. Logic networks are designed making use of logic gates.

Logic gates have several applications to the computers. These are used in following useful applications :

1. ADDERS
   - Half Adder
   - Full Adder
2. Encoders
3. Decoders

and many more. However, we are going to cover Adders only, as per the syllabus.

2.5.1 Adders

We are familiar with ALU (*Arithmetic Logic Unit*) which performs all arithmetic and logic operations. But ALUs do not process decimal numbers ; they process binary numbers. Thus adders in ALU also work on binary numbers.

Before we get into details, all you need to know is *rules for Binary addition, which are given below :*

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=10$
$1+1+1=11$

See, the result is having two bits, right bit gives you the *Sum* and left bit gives you the *Carry*.

Half Adder

It is a logic circuit that adds two bits. It produces the outputs : SUM and CARRY. The boolean equations for SUM and CARRY are :

$$\text{SUM} = X \oplus Y = \overline{X}Y + X\overline{Y}$$
$$\text{CARRY} = X \cdot Y$$

SUM is X XOR Y; and CARRY is X AND Y

Therefore, SUM produces 1, when X and Y are different and CARRY is 1 when X and Y both are 1's. Truth Table for Half Adder is given in (Table 2.21).

Table 2.21 *Truth Table for half adder*

X	Y	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Logic circuit for Half adder has been given in Fig. 2.16.

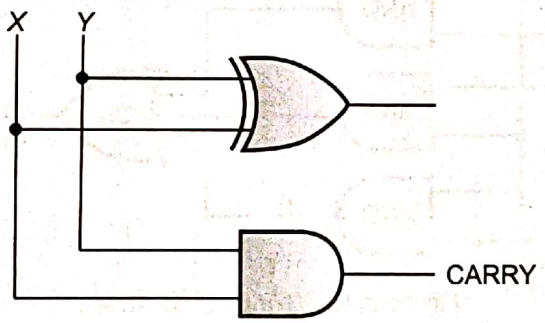


Figure 2.16 2-input Half adder.

Applications of Half adder are limited as only two bits can be added. Instead we need a circuit that can add three bits at a time.

Full Adder

It is a logic circuit that can add three bits. It produces two outputs : SUM and CARRY. The boolean equations for SUM and CARRY are : (Refer to Truth table 2.21)

$$\text{SUM} = X \oplus Y \oplus Z = (\overline{X}\,\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XYZ)$$
$$\text{CARRY} = XY + YZ + ZX \quad \text{(Simplified from expression } \overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + XYZ)$$

i.e., SUM equals X XOR Y XOR Z

CARRY equals XY OR YZ OR ZX

Therefore, SUM produces 1 when input is containing odd number of 1's and CARRY is when there are two or more 1's in input.

Truth Table for Full Adder is given below (Table 2.22).

Table 2.22 *Truth Table for full adder*

X	Y	Z	Carry		Sum	
0	0	0	0		0	
0	0	1	0		1	$\overline{X}\,\overline{Y}Z$
0	1	0	0		1	$\overline{X}Y\overline{Z}$
0	1	1	1	$\overline{X}YZ$	0	
1	0	0	0		1	$X\,\overline{Y}\,\overline{Z}$
1	0	1	1	$X\,\overline{Y}\,Z$	0	
1	1	0	1	$XY\,\overline{Z}$	0	
1	1	1	1	XYZ	1	XYZ

Logic circuit for Full Adder has been given in Fig. 2.17.

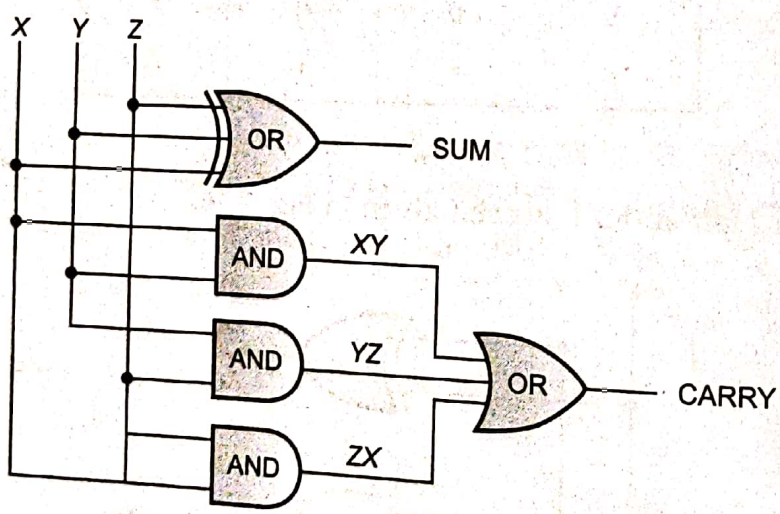


Figure 2.17 3-input Full Adder.

Let Us Revise

- *The NOT gate or inverter is a gate with only one input signal and one output signal; the output state is always the opposite of the input state.*
- *The OR gate has two or more input signals but only one output signal. If any of the input signals is 1 (high), the output signal is 1 (high).*
- *The AND gate can have two or more input signals and produce an output signal. When all the inputs are 1 i.e., high then the output is 1 otherwise output is 0.*
- *The NOR gate has two or more input signals but only one output signal. If all the inputs are 0 (i.e., low), then the output is 1 (high).*
- *The NAND gate has two or more input signals but only one output signal. If all of the inputs are 1 (high), then the output produced is 0 (low).*
- *XOR gate has two or more inputs but one output. It produces output 1 for only those input combinations that have odd number of 1's.*

- *XNOR gate is inverted XOR gate. It produces 1 (high) output when the input combination has even number of 1's.*
- *Design rules for NAND-to-NAND logic Network : (i) derive simplified sum-of-products expression. (ii) draw a circuit diagram using AND, OR gates. (iii) just replace AND and OR gates with NAND gates.*
- *Design rule for NOR-to-NOR logic Network (i) derive simplified P-O-S expression (ii) draw a circuit using OR, AND gates. (iii) replace OR and AND gates with NOR gates.*
- *Logic network have several applications, some of them are : Adders (half adder, full adder), Encoders, Decoders etc.*
- *Half adder is logic circuit that adds two bits.*
- *Full adder is logic circuit that adds three bits.*

Solved Problems

1. *Construct a truth table for the expression $\overline{\overline{a}}$. What single term is the expression equivalent to ?*

Solution.

a	$\overline{a}$	$\overline{\overline{a}}$
0	1	0
1	0	1

Comparing the columns a and $\overline{\overline{a}}$, we find that $\overline{\overline{a}} = a$. Hence the expression $\overline{\overline{a}}$ is equivalent to a.

2. *Prove that $x + yz = (x + y)(x + y)$*

Solution.

x	y	z	yz	$x + yz$	$(x + y)$	$(x + z)$	$(x + y)(x + z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Comparing the columns, we find that $x + yz = (x + y)(x + z)$. Hence proved.

3. *Verify using truth table that $(x + y)' = x' \cdot y'$.*

Solution. As it is a 2-variable expression, truth table will be as follows :

x	y	$x + y$	$(x + y)'$	x'	Y'	$x'y'$
0	0	0	1	1	1	
0	1	1	0	1	0	
1	0	1	0	0	1	
1	1	1	0	0		

Comparing the columns $(x + y)'$ and $x'y'$, both of the col·

4. *From the premises $p \Rightarrow q$ and $q \Rightarrow p$, conclude $\overline{q} + pq$.*

Solution. Given premises are :

$P1$: $p \Rightarrow q$

$P2$: $q \Rightarrow p$

Conclusion (C) to be drawn : $\bar{q} + pq$, we have to find out $(P1 . P2) \Rightarrow C$.

p	q	$p \Rightarrow q$	$q \Rightarrow p$	$P1 . P2$ $(p \Rightarrow q) . (q \Rightarrow p)$	$\bar{q}$	pq	C $\bar{q} + pq$	$(P1 . P2) \Rightarrow C$
0	0	1	1	1	1	0	1	1
0	1	1	0	0	0	0	0	1
1	0	0	1	0	1	0	1	1
1	1	1	1	1	0	1	1	1

Since $(P1 . P2) \Rightarrow C$ results into tautology. Hence concluded.

5. *From premises $p \Rightarrow q$ and $q \Rightarrow p$, conclude $\bar{q} + pq$ algebraically.*

Solution. Given premises are :

$$P1 : p \Rightarrow q \text{ and } P2 : q \Rightarrow p$$

Conclusion (C) to be drawn : $\bar{q} + pq$.

We have to find out $(P1 . P2) \Rightarrow C$

$$= [(p \Rightarrow q) . (q \Rightarrow p)] \Rightarrow (\bar{q} + pq)$$

Carrying out conditional elimination, we get

$$= [(\bar{p} + q) . (\bar{q} + p)] \Rightarrow (\bar{q} + pq)$$

Carrying out conditional elimination once again, we get

$$= \overline{[(\bar{p} + q) . (\bar{q} + p)]} + (\bar{q} + pq)$$

$$= (\overline{\bar{p} + q}) + (\overline{\bar{q} + p}) + \bar{q} + pq$$

[De Morgan's Law]

$$= (\bar{\bar{p}} . \bar{q}) + (\bar{\bar{q}} . \bar{p}) + \bar{q} + pq$$

[De Morgan's Law]

$$= p\bar{q} + q\bar{p} + \bar{q} + pq$$

$$= p\bar{q} + \bar{q} + q\bar{p} + pq$$

$$= \bar{q}(p + 1) + q(\bar{p} + p)$$

$$= \bar{q} . 1 + q . 1$$

$$= \bar{q} + q$$

$[\because p + 1 = 1 ; \bar{p} + p = 1]$

$$= 1$$

$[\because \bar{q} + q = 1]$

Hence concluded.

6. *Find out the equivalent expression for $(p \Leftrightarrow q) + (p \Rightarrow q)$, without having any conditional or bi-conditional.*

Solution. Carrying out bi-conditional elimination, we get

$$p \Leftrightarrow q = (p . q) + (\bar{p} . \bar{q})$$

Carrying conditional elimination, we get

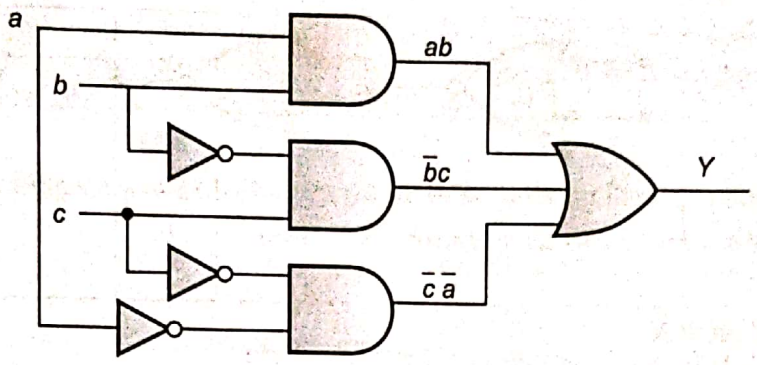
$$p \Rightarrow q = \bar{p} + q.$$

Thus, the equivalent expression is :

$$pq + \bar{p} . \bar{q} + \bar{p} + q.$$

7. *Draw logic circuit diagram for the following expression :* $Y = ab + \bar{b}\,c + \bar{c}\,\bar{a}$

Solution.



8. *Draw the simplified logic diagram using only NAND gates to implement :*

$$\bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,\bar{y}\,z + \bar{x}\,y\,z + x\,\bar{y}\,z$$

Solution. $\bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,\bar{y}\,z + \bar{x}\,y\,z + x\,\bar{y}z = \bar{x}\,\bar{y}\,(\bar{z} + z) + \bar{x}\,y\,z + \bar{y}\,z$

$= \bar{x}\,\bar{y}\,.1 + \bar{x}\,yz + x\,\bar{y}\,z$ $(\bar{z} + z = 1)$

$= \bar{x}\,\bar{y} + \bar{x}yz + x\bar{y}z$

$= \bar{x}\,(\bar{y} + yz) + x\bar{y}z$

$= \bar{x}\,(\bar{y} + z) + x\bar{y}z$ $(\bar{y} + yz = \bar{y} + z$ Table 2.6)

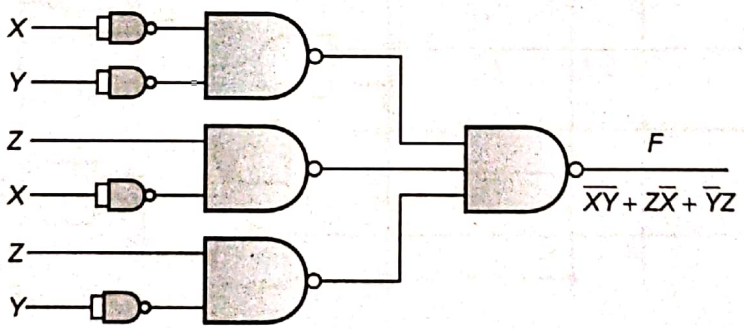
$= \bar{x}\,\bar{y} + \bar{x}z + x\bar{y}z$

$= \bar{x}\,\bar{y} + z\,(\bar{x} + x\bar{y})$

$= \bar{x}\,\bar{y} + z\,(\bar{x} + \bar{y})$ $(\bar{x} + \bar{y} = \bar{x} + \bar{y}$ *Table* 2.6)

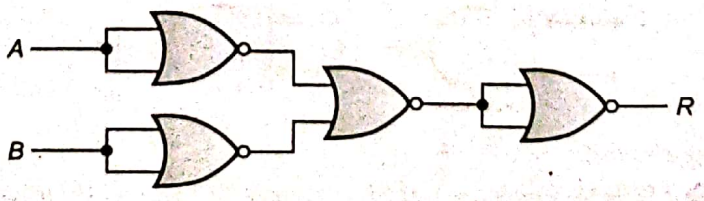
$= \bar{x}\,\bar{y} + z\bar{x} + \bar{y}z$

NAND-to-NAND circuit is



9. *Draw the logic of NAND gate using NOR gates only.*

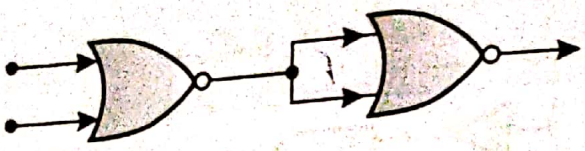
Solution.



R will give *A* NOR *B*

10. (a) *Represent NOT using only NOR gate(s).*

(b) *Given the following circuit :*



What is the output if (i) both inputs are FALSE (ii) one is FALSE and the other is TRUE ?

(c) *Draw the circuit of a Half Adder using only NAND gates.*

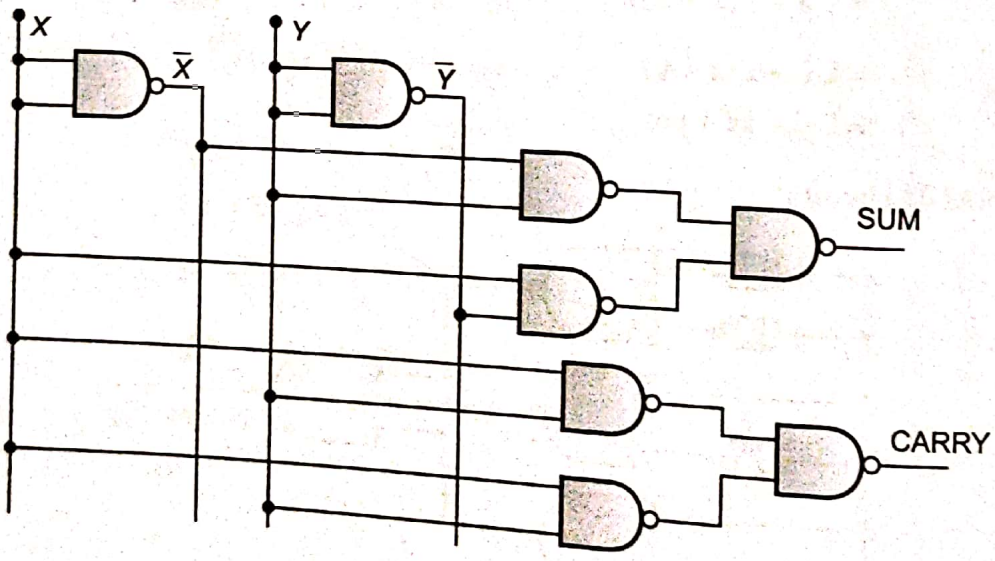
Solution.

(a) $\overline{X}$ = NOT X = X NOR X

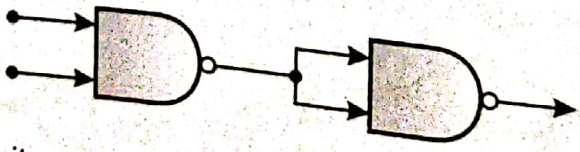
(b) (i) False (ii) True

(c) In half adder Sum = $\overline{x}\,y + x\,\overline{y}$, Carry = $X \cdot Y$

x	y	Carry	Sum
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0



11. (a) *Seven inverters are cascaded one after another. What is the output if the input is 1 ?*



(b) *Given the following circuit :*

What is the output if (i) both inputs are FALSE (ii) one is FALSE and the other is TRUE ?

(c) *Derive the expression for a Full Adder.*

Solution.

(*a*) 0

(*b*) (*i*) False (*ii*) False

(*c*) **Full Adder** It is a logic circuit that can add three bits. It produces two outputs : SUM and CARRY. The boolean equations for SUM and CARRY are

$$SUM = X \oplus Y \oplus C, \qquad CARRY = XY + YZ + ZX$$

i.e., SUM equals X XOR Y XOR Z CARRY equals XY OR YZ OR ZX

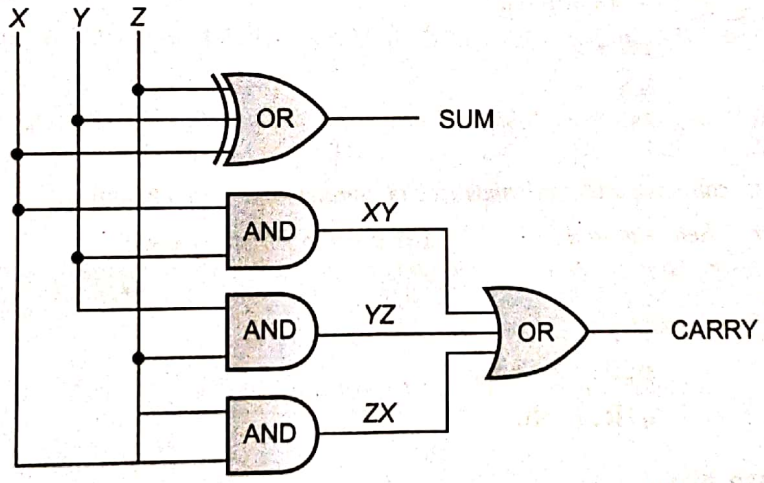
Therefore, SUM produces 1 when input is containing odd number of 1's and CARRY is 1 when there are two or more 1's in input.

Truth Table for Full Adder is given below :

Truth Table for full adder

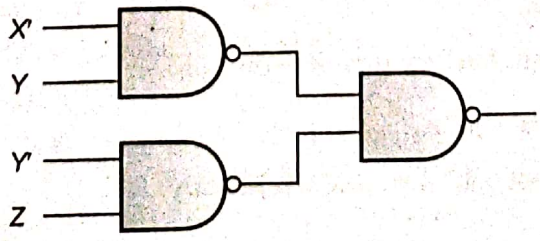
X	Y	Z	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Logic circuit for Full Adder has been given in following figure :



12. *Represent the Boolean expression $X'Y + Y'Z$ with the help of NAND gates only.*

Solution.



13. *Given wffs are : $p \Rightarrow q$ and $\sim q$, show that $\sim p$ is a logical consequence of two preceding wffs.*

Solution. We have to determine $[(p \Rightarrow q) . \sim q] \Rightarrow \sim p$

p	q	$p \Rightarrow q$	$\sim q$	$(p \Rightarrow q) . \sim q$	$\sim p$	$[(p \Rightarrow q) . \sim q] \Rightarrow \sim p$
0	0	1	1	1	1	1
0	1	1	0	0	1	1
1	0	0	1	0	0	1
1	1	1	0	0	0	1

$[(p \Rightarrow q) . \sim q] \Rightarrow \sim p$ is a tautology. Hence,

$$\begin{array}{l} p \Rightarrow q \\ \sim q \\ \hline \sim p \end{array}$$

14. *Solve the previous question algebraically.*

Solution. We have to determine $[(p \Rightarrow q) . \sim q] \Rightarrow \sim p$.

Carrying out conditional elimination, we get

$$= [(\sim p + q) . \sim q] \Rightarrow \sim p$$
$$= (\sim p . \sim q + q . \sim q) \Rightarrow \sim p$$
$$= (\sim p . \sim q + 0) \Rightarrow \sim p \qquad [\because q . \sim q = 0]$$

Carrying out conditional elimination once again, we get

$$\sim(\sim p . \sim q) + \sim p$$
$$= \sim\sim p + \sim\sim q + \sim p \qquad [\because \sim(\sim p . \sim q) = \sim\sim p + \sim\sim q, \text{ De Morgan's Law}]$$
$$= p + q + \sim p$$
$$= p + \sim p + q$$
$$= 1 + q \qquad [\because p + \sim p = 1]$$
$$= 1 \qquad [\because 1 + q = 1]$$

Hence established.

15. *Construct the inverse, converse and contrapositive of conditionals given below :*

(i) If it is January, then it is cold. *(ii) If $y + 5 \neq 7$, then $y < 0$*

Solution.

(i) Let $p \Rightarrow q$ = If it is January, then it is cold.

i.e.,

p : It is January

q : It is cold.

Converse will be $q \Rightarrow p$, *i.e.,*

If it is cold then it is January.

Inverse will be $\sim p \Rightarrow \sim q$, *i.e.,*

If it is not January then it is not cold.

Contra-positive will be $\sim q \Rightarrow \sim p$, *i.e.,*

If it is not cold then it is not January.

(*ii*) Let $a \Rightarrow b$ = If $y + 5 \neq 7$, then $y < 0$, *i.e.*,

a : $y + 5 \neq 7$

b : $y < 0$

Converse will be $b \Rightarrow a$, *i.e.*,

If $y < 0$ then $y + 5 \neq 7$

Inverse will be $\sim a \Rightarrow \sim b$, *i.e.*,

If $y + 5 = 7$ then $y \nless 0$

Contra-positive will be $\sim b \Rightarrow \sim a$, *i.e.*,

If $y \nless 0$ then $y + 5 = 7$.

Glossary

AND *gate* A logic circuit whose output is 1 (*high*) only when all inputs are 1 (*high*).

Connective Operator joining simple statements into compounds.

Gate A logic circuit with one or more input signals but only one output signal.

NAND *gate* Logically means an AND gate followed by an inverter. All inputs must be 1 (*high*) to get a 0 (*low*) output.

NOR *gate* Logically means an OR gate followed by an inverter. All inputs must be 0 (*low*) to get a 1 (*high*) output.

NOT *gate* A gate with one input and one output signal, output being complement of the input.

OR *gate* A logic circuit whose output is 1 (*high*) only when one or more inputs are 1 (*high*)

Proposition Elementary atomic sentence that may either be true or false.

Syllogism Logical process of drawing conclusion from given premises.

Truth table A table of combinations showing all input and output possibilities for a boolean expression.

Truth value Truth or falsity of a proposition.

XNOR *gate* Logically means an XOR gate followed by an inverter. Inputs must have even number of 1's to get a 1 (*high*) output.

XOR *gate* A gate which produceses 1 (*high*) output only when inputs have even number of 1's.

Assignments

TYPE A : VERY SHORT ANSWER QUESTIONS

1. What is proposition ?
2. What do you mean by contingency, tautology and contradiction ?
3. What is a logic gate ? Name the three basic logic gates.
4. Which gates implement logical addition, logical multiplication and complementation ?
5. Write the precedence order of evaluation of logical operators.
6. What is the other name of NOT gate ?
7. What is a truth table ? What is the other name of truth table ?

   (*i*) $A + 0 = ?$ (*ii*) $A + 1 = ?$ (*iii*) $A.0 = ?$ (*iv*) $A.1 = ?$

8. According to which law following expressions are true :
$X+(Y+Z)=(X+Y)+Z$ and $X(YZ)=(XY)Z$
9. Which law states that $X(Y+Z)=XY+XZ$?
10. What is the following property of called ? $X.(Y+Z)=X.Y+X.Z$
11. What is the following property of called ? $X+XY=X$
12. Rule $\overline{X+Y}=\overline{X}.\overline{Y}$ and Rule $\overline{X.Y}=\overline{X}+\overline{Y}$ are known as ?
13. Which of the following is/are incorrect ? Write the correct forms of the incorrect ones :
(a) $A+A'=1$ (b) $A+0=A$ (c) $A.1=A$ (d) $AA'=1$
(e) $A+AB=A$ (f) $A(A+B)'=A$ (g) $(A+B)'=A'+B$ (h) $(AB)'=A'B'$
(i) $A+1=1$ (j) $A+A=A$ (k) $A+A'B=A+B$ (l) $X+YZ=(X+Y)(X+A)$
14. What is a connective ?
15. Name the process for the compounds given below :
(i) $p+q$ (ii) $p.q$ (iii) $p\Rightarrow q$ (iv) $\sim p$ (v) $p\Leftrightarrow q$
16. What are contingencies ?
17. Form the converse, inverse and contrapositive of condition $p\Rightarrow q$.
18. What is syllogism ?
19. Why are NAND and NOR gates called Universal gates ?
20. Which gates are called Universal gates and why ?
21. Draw a logic circuit diagram using NAND or NOR gates only to implement the Boolean function $F(a,b)=a'b'+ab$
22. How does half adder differ from full adder ?
23. What is inverted AND gate called ? What is inverted OR gate called ?
24. When does an XOR gate produce a high output ? When does an XNOR gate produce a high output ?
25. Write some applications of logic networks.

TYPE B : SHORT ANSWER QUESTIONS

1. Given the following simple propositions :
p = It is raining
q = It is not a sunny day.
Construct the compound sentences for the following expressions :
(i) q' (ii) $p.q$ (iii) $p+q$ (iv) $p\Rightarrow q$ (v) $p\Leftrightarrow q$
(vi) $p'+q'$ (vii) $p'.q'$ (viii) $\sim p\Rightarrow q$ (ix) $\sim p\Leftrightarrow\sim q$ (x) $(p+q)\Rightarrow(p.q)$
2. (a) If x represents "I like coffee" and y represents "I like tea" then write in symbolic form :
(i) I like coffee and tea ; (ii) I like coffee but not tea ;
(iii) It is false that I don't like coffee or tea ; (iv) Neither I like coffee nor tea ;
(v) Either I like coffee or I donot like coffee but like tea.
(b) For what statement does $\sim x\wedge y$ stand for ?
3. If s stands for the statement "I will not go to school", and t for the statement, "I will watch a movie", then what does $\sim s+t$ stand for ?
4. Establish the following using truth tables :
(i) $\sim(a\Rightarrow b)=p.\sim q$ (ii) $a\Rightarrow b=\sim a\Rightarrow\sim b$
(iii) $(a\Rightarrow b).(b\Rightarrow a)=a\Leftrightarrow b$
(iv) $(\sim a+b)\wedge(\sim b+a)=a\Leftrightarrow b$
(v) $(p\Rightarrow r).(q\Rightarrow r)=(p+q)\Rightarrow r$

5. What will be the result of following compounds if given inputs are

(*i*) $x=0, y=1$; (*ii*) $x=1, y=1$; (*iii*) $x=1, y=0$

(*a*) $y \wedge x$ (*b*) $y \vee x$ (*c*) $\sim x \vee y$ (*d*) $x \wedge \sim y$ (*e*) $(\sim x \wedge \sim y) \vee \sim y$

6. Determine the converse, inverse and contrapositive for conditional (*i*) $x \Rightarrow y$, (*ii*) $p \Rightarrow q$, where

x : It is raining, y : I am enjoying it.

$p: 2+3 \neq 6$, $q = a > 0$

[**Hint.** Refer to solved problem 14]

7. Find out which of the following are tautologies and which of them are contradictions ?

(*a*) $p.q(p+q)'$ (*b*) $(p \Rightarrow q).p$

(*c*) $(p.q) \Rightarrow (\sim p.\sim q)$ (*d*) $[(p \Rightarrow q) \Leftrightarrow \sim q \Rightarrow (\sim p \wedge \sim q)]$

(*e*) $(p+q).(p'.q')$ (*f*) $[(p \Rightarrow q) \wedge (q \Rightarrow r)]$

8. Given that : $p: 2+3=5$

$q: 2 \times 3 = 6$

Now construct the truth table for following compounds :

(*i*) if $2+3=5$, then $2 \times 3 = 6$ (*ii*) if $2+3 \neq 5$, then $2 \times 3 = 6$

(*iii*) if $2+3=5$, then $2 \times 3 \neq 6$ (*iv*) if $2+3 \neq 5$, then $2 \times 3 \neq 6$

9. Establish the validity for the following :

$$\frac{\begin{array}{l} p \Rightarrow q \\ p \Rightarrow r \end{array}}{\sim r \Rightarrow q}$$

10. Establish that

$$\frac{\begin{array}{l} a+b \\ \sim a \Rightarrow b \end{array}}{\sim b}$$

11. Establish that

$$\frac{\begin{array}{l} a \Rightarrow b \\ b \Rightarrow c \\ a \end{array}}{c}$$

12. Establish that

$$\frac{\begin{array}{l} x \Leftrightarrow \sim y \\ y \Rightarrow z \end{array}}{z}$$

13. Draw the conclusions from the following premises :

*P*1 : John is a father.

*P*2 : If John is a father then John has a child.

[**Hint.** Modus ponens]

14. Determine whether each of the following sentences is

(*a*) satisfiable (*b*) contradictory (*c*) valid

$S_1 : (p \,\&\, q) \vee \sim (p \,\&\, q)$ $S_2 : (p \vee q) \rightarrow (p \,\&\, q)$

$S_3 : (p \,\&\, q) \rightarrow r \vee \sim q$ $S_4 : (p \vee q) \,\&\, (p \vee \sim q) \vee p$

$S_5 : p \rightarrow q \rightarrow \sim p$ $S_6 : p \vee q \,\&\, \sim p \vee \sim q \,\&\, p$

15. Find the meaning of the statement :

$$(\sim p \vee q) \,\&\, r \rightarrow s \vee (\sim r \,\&\, q)$$

for each of the interpretations given below :

(*a*) I_1 : p is true, q is true, r is false, s is true.

(*b*) I_2 : p is true, q is false, r is true, s is true.

16. What do you understand by 'truth value' and 'proposition' ? How are these related ?

17. What do you understand by 'logical function' ? What is its alternative name ? Give examples for logical functions.

18. What is meant by tautology and contradiction ? Prove that $1 + Y$ is a tautology and $0.Y$ is a contradiction.

19. What is a truth table ? What is its significance ?
20. Verify using truth table that $X + XY = X$ for each X, Y in {0, 1}.
21. Verify using truth table that $(X + Y)' = X'Y'$ for each X, Y in {0, 1}.
22. Give truth table for the Boolean Expression $(X + Y')'$.
23. Draw the truth table for the following equations : (a) $M = N(P + R)$ (b) $M = N + P + N\overline{P}$
24. Using truth table, prove that $AB + BC + C\overline{A} = AB + C\overline{A}$
25. State the distributive laws. How do they differ from the distributive laws of ordinary algebra ?
26. Prove the idempotence law with the help of a truth table.
27. Prove the complementarity law with the help of a truth table.
28. Give the truth table proof for distributive law.
29. Prove the following :
    (i) $A(B + C + D) = A$ (ii) $\overline{A}\,\overline{AB} = A + \overline{B}$
    (iii) $(x + y + z).(\overline{x} + y + z) = y + z$ (iv) $AB + BC + BCD = A(\overline{AD} + C)$
30. Draw logic circuit diagrams for the following :
    (i) $xy + x\overline{y} + \overline{x}z$ (ii) $(A + B)(B + C)(\overline{C} + \overline{A})$ (iii) $\overline{A}B + BC$ (iv) $xyz + \overline{x}y\overline{z}$
31. Design a circuit (3 input) which gives a high input, when there is even number of low inputs.
32. Design a circuit (3 input) which given a high input only when there is even number of low or high inputs.
33. Design a logic circuit to realize the Boolean function $f(x, y) = x \cdot y + x' \cdot y'$
34. Draw the logic circuit for this boolean equation :
    $$y = \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}\,\overline{C}D + AB\overline{C}D + ABC\overline{D}$$
35. Draw the AND-OR circuit for $y = A\overline{B}\,\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + ABCD$
36. Convert the above circuit into NAND-to-NAND logic circuit.
37. Why are NAND and NOR gates more popular ?
38. Draw the logical circuits for the following using NAND gates only :
    (i) $xy + x\overline{y}z + xyz$ (ii) $ABC + A\overline{B}\,\overline{C} + \overline{A}BC$
39. Draw the logical circuits for the following using NOR gates only :
    (i) $(X + Y).(\overline{X} + Y).(\overline{X} + \overline{Y})$ (ii) $(X + Y + Z).(X + \overline{Y} + \overline{Z})$
40. (a) State De Morgan's Laws. Verify them using truth tables.
    (b) Prove $(A + B).(A' + C) = (A + B + C).(A + B + C').(A' + B + C).(A' + B' + C)$.
    (c) Give the truth table for a Full-adder.
    (d) Draw the circuit diagram for the Boolean function $F(X, Y, Z) = (X' + Y)(Y' + Z)$ using NOR gates only.
41. (a) State the distributive law. Verify the law using truth table.
    (b) Prove $x + x'y = x + y$.
    (c) Draw the logic circuit for a half-adder.
    (d) Represent the Boolean expression $(x + y)(y + z)(z + x)$ with the help of NOR gates only.
42. (a) State the Idempotence law. Verify the law using truth table.
    (b) Prove $(x + y)(x + z) = x + xz$ algebraically.
    (c) Draw the logic circuit for a Full Adder.
    (d) Represent the Boolean expression $yz + xz$ with the help of NAND gates only.
43. (a) State Absorption Laws. Verify one of the Absorption laws using a truth table.
    (b) Represent the Boolean expression $X + Y \cdot Z'$ with the help of NAND gates only.