25.13. LOGIC GATES

A *logic gate* is an electronic circuit which has the ability to make logical decisions in the sense that it produces output level when some combinations of input levels are present and a different output when other combinations are present. A logic gate is in fact a switching circuit (that is a digital circuit) which makes logical decisions regarding the existence of output depending on the nature of the input. Therefore, a digital circuit with one or more input signals but only one output signal is called a logic gate. A digital circuit performs logical functions using the two binary symbols 1 and 0. The logic functions can be described in terms of algebraic statements. Obviously, the terms in the algebraic expressions all have the designations of either 1 or 0. We assume here that logical 1 corresponds to most positive voltage and logical 0 represents the most negative (or 0) voltage. There are just three basic types of logic gates, namely (*i*) OR gate (ii) AND gate and (iii) NOT gate. The operation of a logic gate is described by a truth table or Boolean algebra. A ***truth table*** is a table that shows all the input-output possibilities of a logic circuit. We study here the ***functional*** behaviour of gates, i.e., about the nature of logical decisions they perform, without any reference to the electronic circuitry.

25.13.1. The AND Gate

An AND gate is a device whose output is 1 if and only if all its inputs are 1.

AND gates can have two or more inputs but only one output. The output of an AND gate is only 1 when all the inputs are 1. The output is 0 if any one or more of inputs to the AND gate are 0.

[illegible] is described conveniently assigning symbols A,B,C,...... to the inputs.

A table is made by listing all possible combinations of values that A and B can have and the output of the gate corresponding to each input combination. The table is called a truth table. The truth table for a 2-input AND gate is shown below.

INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

It may be noticed that the list of input combinations in the truth table is the same as counting in binary from 00 to 11. There are $2^2 = 4$ possible combinations of values that the input variables can have.

The logical AND function is expressed symbolically as

$$A \cdot B = Y$$

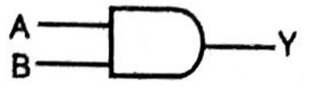
$\uparrow$

AND symbol

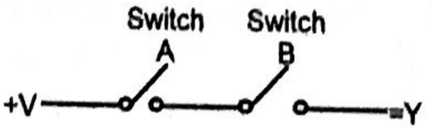
We should read the above expression as A and B (not as A into B). It is seen that we use a dot for the AND operation. We now say that 0 ANDed with any variable equals 0 and 1 ANDed with 1 equals 1.

The logical symbol for AND gate is shown in Figure 25.1 along with schematic representation. The AND gate is very much similar to switches connected in series as in Fig. 25.1(a). When either of switches output is not obtained. Output will be obtained only when both A and B are closed.

The logic symbol and truth table for a 3-input AND gate is shown in Fig. 25.2

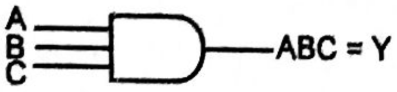


(a)



(b) Simplified schematic representation

Figure 25.1. (*a*) Logic symbol for a 2-input AND gate.
(*b*) schematic representation



(*a*) Logic symbol

A	B	C	Y
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

(*b*) Truth Table

Fig 25.2. Logic Symbol and Truth table for a 3-input AND gate

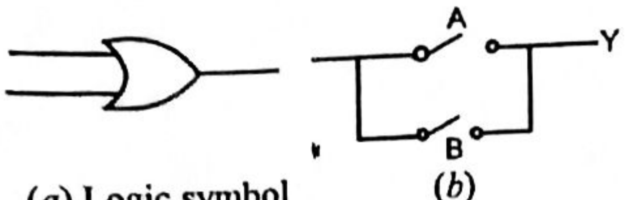
25.13.2. The OR Gate

An OR gate is a device whose output is 1 if at least one of its inputs is 1. The output of an OR gate is zero when all its inputs are 0. The output of an OR gate is 1 if any or all the inputs are 1. The name OR is given to the gate due to the fact that its output is 1 if either A or B (or both) is 1. For the same reason the OR gate is called "any or all" gate.

The OR function is designated by the symbol " + ". The logical OR function is expressed as

$$\underset{\substack{\uparrow \\ \text{OR symbol}}}{A + B} = Y$$

The logic symbol and truth table for a 2-input OR gate is shown in Fig. 25.3. The schematic representation of OR gate is shown in Fig.25.3(*b*). Output is obtained even when either or both of the switches A,B are closed.



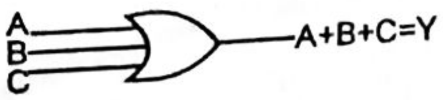
(*a*) Logic symbol (*b*)

A	B	y
0	0	0
0	1	1
1	0	1
1	1	1

(*c*) Truth Table

Fig 25.3. Logic Symbol, (*b*) schematic representation and Truth table for a 2-input OR gate

The logical symbol and truth table for a 3-input OR gate are shown in Fig 25.4.



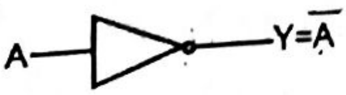
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Fig 25.4. Logic Symbol and truth table for a 3-input OR gate.

25.13.3. The NOT gate or Inverter

The NOT gate is the simplest among all the logic gates. ***The NOT gate or inverter is a device whose output is 1 when its input is 0 and whose output is 0 when its input is 1.*** Thus, the gate ***complements*** a bit. It is called a NOT gate because its output is not the same as its input. It is more often called an inverter because it inverts the input. The complement of a number is designated by drawing a bar over the number.

Thus, $\overline{A}$ (read as A-bar) is the complement of A.



(*a*) Logical Symbol

A	Y
0	1
1	0

(*b*) Truth Table

Fig 25.5. shows the logic symbol and truth table for an inverter.

25.13.4 The NAND gate

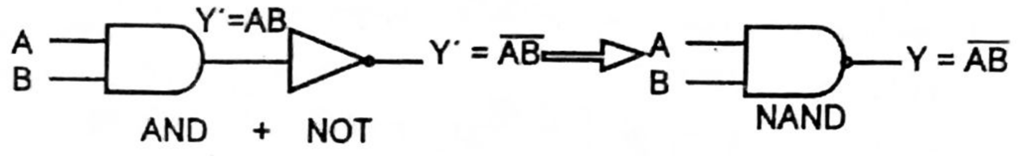
combination of NOT gate AND gate. The gate performs the same logic as an AND gate followed by an inverter. It means that the output of a NAND gate is opposite to the AND gate. The output of a NAND gate is 1 if at least one of its inputs is 0.

The NAND function is expressed as

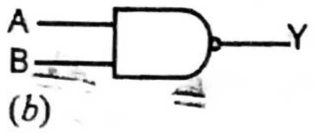
$$Y = \overline{A \circ B}$$

We read it as Y = not A ANDed B. First the inputs are ANDed and then inverted.

The logic symbol of NAND is like that of AND gate with a circle (or bubble symbol) at the output.



(*a*)



(*b*)

A	B	$\overline{A \cdot B} = Y$
0	0	1
0	1	1
1	0	1
1	1	0

(*c*)

Fig. 25.6. NAND gate.

The logical symbol and truth table for NAND gate is given above for a 2-input NAND gate.

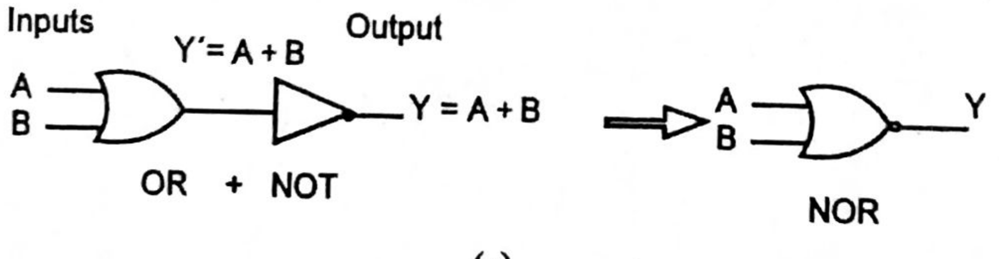
25.13.5. NOR Gate

A NOR gate is a combination of NOT gate and OR gate. It performs the same logic function as an OR gate followed by an inverter. Thus the output of a NOR gate is opposite to that of OR gate. The output of a NOR gate is 1, if all the inputs are 0 and it is 0 if atleast one of its inputs is 1.

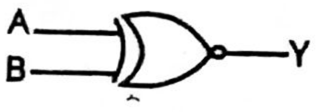
The NOR function is expressed as

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

We read it as Y = not A OR B. The inputs are first ORed and then inversion is performed.



(*a*)



(b)

A	B	$Y=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

(c)

Fig 25.7. Logic symbol and the truth table for a NOR gate.

25.13.6. Exclusive –OR gate

The exclusive-or gate is obtained by using OR, AND and NOT gates as shown in Fig 25.8.

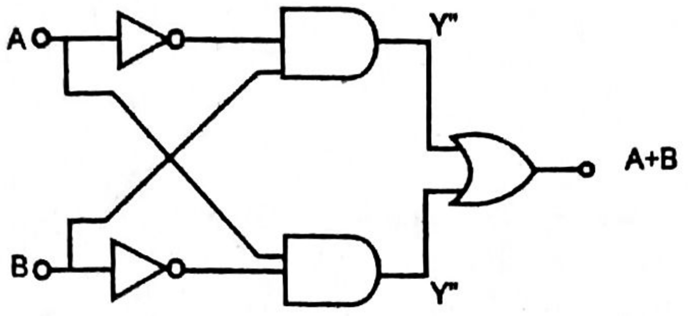
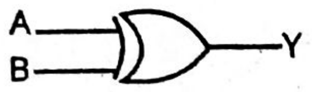


Fig 25.8. Exclusive-OR gate using OR, NAND & NOT gate.

The logic operations in the circuit are as under :

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \circ B$	$A \circ \overline{B}$	$Y = \overline{A} \circ \overline{B} + A \circ \overline{B}$
0	0	1	1	$1.0 = 0$	$0.1 = 0$	$0 + 0 = 0$
0	1	1	0	$1.1 = 1$	$0.0 = 0$	$1 + 0 = 1$
1	0	0	1	$0.0 = 0$	$1.1 = 1$	$0 + 1 = 1$
1	1	0	0	$0.1 = 0$	$1.0 = 0$	$0 + 0 = 0$

The logic symbol and truth table for an exclusive-OR gate are shown below :



(a) Logical Symbol

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

(*b*) Truth Table

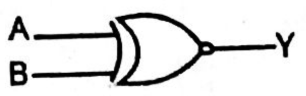
Fig 25.9. EX-OR gate

The output of Ex-OR gate is 1 if any input is 1 but not if all inputs are 1.

25.13.7. Exclusive – NOR gate

Exclusive – NOR gate also is obtained by using OR, AND and NOT gates. It produces an output opposite to that of EX-OR gate.

The logic symbol and truth table for an exclusive-NOR gate is shown below:



(*a*) Logic symbol of a 2-input EX-NOR GATE.

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

(*b*) Truth Table

Fig 25.10.

It may be stated that the output of EX-NOR gate is 1 if inputs are the same either 0 or 1.

25.14. BOOLEAN ALGEBRA

Boolean algebra is a set of rules, laws and the theorems by which logical operations can be expressed symbolically in equation form and be manipulated mathematically. George Boole invented a new kind of algebra in 1854 which contained the basic ideas of mathematical theory of logic. The algebra used to symbolically describe logic functions is therefore named as *Boolean algebra*. In Boolean algebra, variables can have only two values, either 0 or 1. There are basically three Boolean operators namely plus (+), times (×) and overbar (–) which may be thought of as codes for the basic gates representing the three basic logic operations AND, OR and INVERT. There are a set of axioms which we accept without proof and use to build a number of useful theorems.

Axiom 1	$0.0=0$	Axiom 6	$0+1=1$
Axiom 2	$0.1=0$	Axiom 7	$1+0=1$
Axiom 3	$1.0=0$	Axiom 8	$1+1=1$
Axiom 4	$1.1=1$	Axiom 9	$1=0$
Axiom 5	$0+0=0$	Axiom 10	$0=1$

Axiom 8 is known as ***Idempotent*** relation.

25.15. LAWS OF BOOLEAN ALGEBRA

In order to apply Boolean algebra properly, we have to follow certain well-developed laws and rules. The three most important laws are: the commutative law, the associative law and the distributive law. Each of the laws may be established using axioms, truth tables, logic diagrams or previously proved laws or theorems.

25.15.1. The Commutative Laws

The ***commutative law of addition*** for two variables is written algebraically as:

$$A+B=B+A$$

This law states that the order in which the variables are ORed makes no difference. Fig 25.11 illustrates the commutative law as applied to the OR gate.

[illegible]

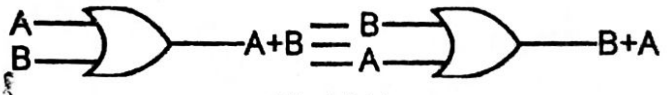


Fig 25.11

The truth tables are identical.

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

=

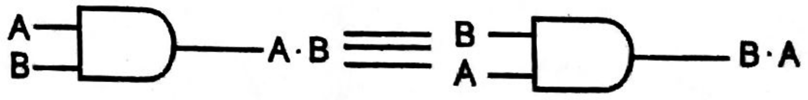
A	B	B+A
0	0	0
0	1	1
1	0	1
1	1	1

The ***commutative law of multiplication*** of two variable is:

$$A \circ B = B \circ A$$

Or $$AB = BA$$

This law states that the order in which the variable are ANDed makes no difference. Figure 25.12 illustrates the law as applied to the AND gate.



The commutative law as applied to AND gate.

Fig. 25.12

The truth tables are identical.

A	B	A ° B
0	0	0
0	1	0
1	0	0
1	1	1

=

A	B	B ° A
0	0	0
0	1	0
1	0	0
1	1	1

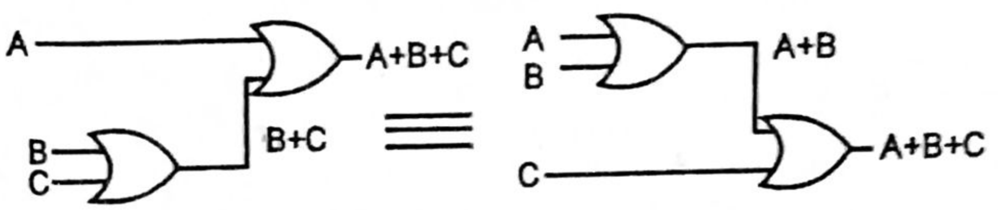
The commutative properties can be extended to any number of variables. Thus, $A + B + C = B + A + C$ and $ABC = BAC = CAB$ and so on.

25.15.2. The Associative Laws

The ***associative law of addition*** for three variables is:

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

This law states that in the ORing of several variables the result is the same irrespective of th way the variables are grouped. Fig 25.13 illustrates the law as applied to OR gates.



The associative law as applied to OR gates.

Fig 25.13

A ORed with B OR C is the same as A OR B ORed with C. The truth tables are identical.

A	B	C	B + C	A + (B + C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	1	0	1	1
1	1	1	1	1

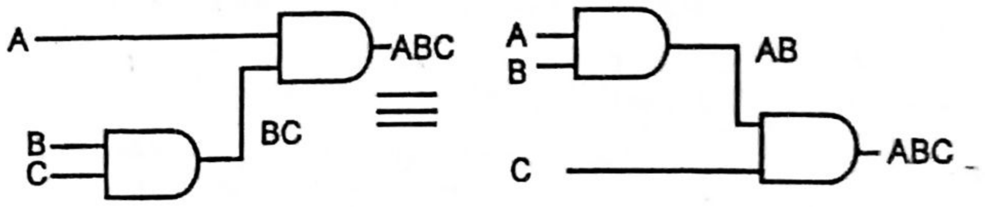
=

A	B	C	B + C	A + (B + C) ✗
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	1	0	1	1
1	1	1	1	1

The associative law of multiplication for three variables is:

$$A(BC) = (AB)C$$

This law states that it makes no difference in what order the variables are grouped in ANDing several variables. Fig 25.14 illustrates the law applied to AND gates.



The associative law as applied to AND gates.

Fig 25.14

A ANDed with B AND C is the same as A AND B ANDed with C. The truth tables are identical

A	B	C	BC	A(BC)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

=

A	B	C	BC	A(BC)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The associative properties can be extended to any number of variables.

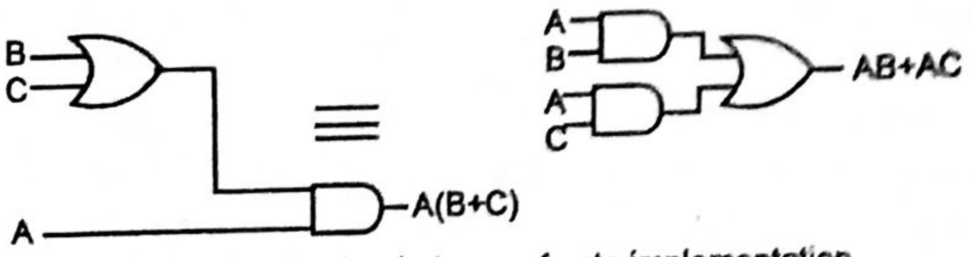
$$(A+B+C)+D=(A+B)+(C+D)=(A+B+C)+D=A+B+C+D$$

$$A(BCD)=(AB)\ (CD)=(ABC)D=ABCD$$

25.15. 3. The Distributive Law

The ***distributive law*** for three variables is written as follows:

$$A\,(B+C)=AB+AC$$



The distributive law in terms of gate implementation.

Fig. 25.15

The law states that ORing several variables and ANDing the result with a single variable is equivalent to ANDing the single variable with each of the several variables and ORing the products. Fig 25.15 illustrates the distributive law.

The truth tables are identical in both the cases.The truth tables are identical in both the cases.

A	B	C	B + C	A (B + C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

=

A	B	C	B + C	(B + C)
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

The distributive property applies to combinations of variables and also to single variables. Thus,

$$AB\,(C+DE)=ABC+ABDE$$

$$(A+B)(C+D)=(A+B)C+(A+B)D=AC+BC+AD+BD$$

25.16. RULES FOR BOOLEAN ALGEBRA

There are some basic rules which are useful in manipulating and simplifying Boolean expressions

1. A.A = A
2. A+A = A

The idempotent properties.

Rule 1 states that if a variable A is ANDed with itself, the result is equal to the variable.

If A = 0, then A.A = 0.0 = 0 = A

If A = 1, then A.A = 1.1 = 1 = A

In either case the output of an AND gate is equal to the value of the input variable A. It is illustrated in Fig 25.16.

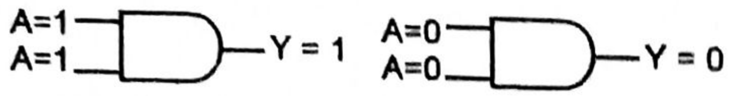


Fig 25.16. Rule 1: A•A = A

Rule 2 states that if a variable is ORed with itself, the output is equal to the variable.

If A = 0, then A+A = 0+0 = 0 = A

If A = 1, then A+A = 1+1 = 1 = A

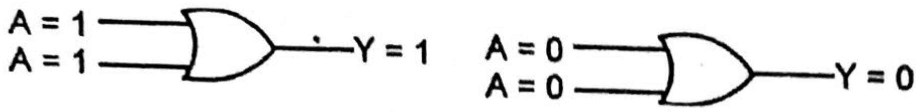


Fig 25.17. Rule 2: A+A = A

3. A . 1 = A   Identity Properties
4. *A + 1 = 1*

Rule 3 states that if a variable A is ANDed with 1, the result is equal to the variable.

If A = 0, then A .1 = 0.1 = 0 = A

If A = 1, then A .1 = 1.1 = 1 = A

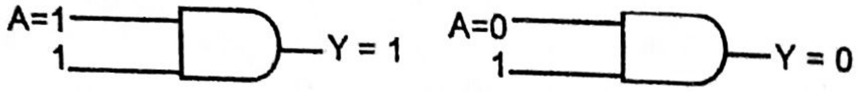


Fig 25.18. Rule 3: A•1 = A

Rule 4 states that a variable A Ored with 1 is always equal to 1.

If A = 0, then A+ 1 = 0+1 = 1

If A = 1, then A +1 = 1+1 = 1

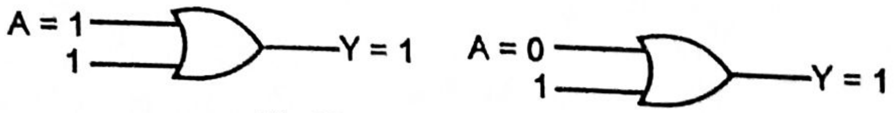


Fig 25.19. Rule 4: A + 1 = 1

5. A.0 = 0
6. A+0 = A

The Null properties

Rule 5 states that a variable ANDed with a 0 always produces a 0.

If A = 0, then A .0 = 0.0 = 0

If A = 1, then A .0 = 1.0 = 0

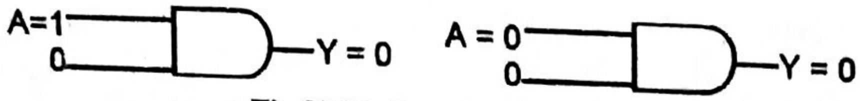


Fig 25.20. Rule 5: A • 0 = 0

Rule 6 states that a variable ORed with a 0 is equal to the value of the variable.

If A = 0, then A + 0 = 0+0 = 0 = A

If A = 1, then A + 0 = 1 + 0 = 1 = A

If A = 1, then A + 0 = 1 + 0 = 1 = A

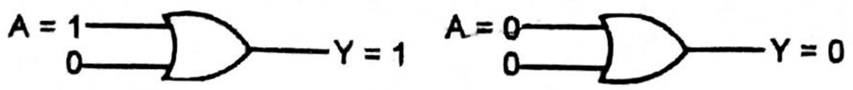


Fig 25.21. Rule 6A + 0 = A.

7. $A \cdot \overline{A} = 0$] **Negation Properties**
8. $A + A = 1$]

Rule 7 states that if a variable is ANDed with its complement, the result is 0.

If A = 0, then $\overline{A} = 1$, and $A \cdot \overline{A} = 0.1 = 0$
If A = 1, then $\overline{A} = 0$, and $A \cdot \overline{A} = 1.0 = 0$

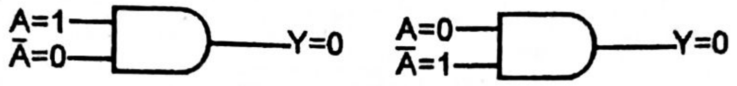


Fig 25.22. $A \cdot \overline{A} = 0$.

Rule 8 states that if a variable and its complement are ORed, the result is always 1.

If A = 0, then $\overline{A} = 1$ and $A + \overline{A} = 0 + 1 = 1$
If A = 1, then $\overline{A} = 0$ and $A + \overline{A} = 1 + 0 = 1$

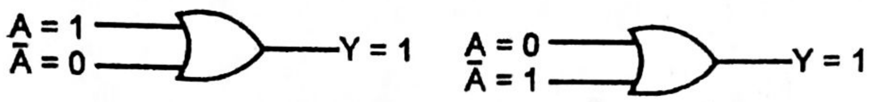


Fig 25.23. $A + \overline{A} = 1$.

9. $\overline{\overline{A}} = A$ *The Double Negation property.*

Rule 9 states that if a variable is complemented twice, the result is the variable itself.

If A = 0, then $\overline{A} = 1$ and $\overline{\overline{A}} = \overline{1} = 0 = A$
If A = 1, then $\overline{A} = 0$ and $\overline{\overline{A}} = 0 = 1 = A$

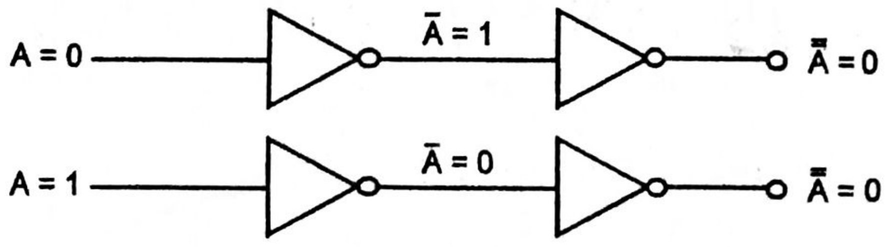


Fig 25.24. Rule 9: $A = \overline{\overline{A}}$.

10. $A + AB = A$]
11. $A + AB = A + B$] The Absorption properties.
12. $A(A + B) = A$]

Rule 10: $A + AB = A$

$A + AB = A(1+B)$ Distributive Law
$= A \cdot 1$ Rule 4
$= A$ Rule 3

Rule 11: $A + AB = A + B$

$A + AB = \underbrace{A + AB}_{A} + AB$ Rule 10

$= A + (A + A)B$
$= A + 1.B$ Rule 8
$= A + B$ Rule 3

Rule 12: $A(A+B) = A$

$$A(A+B) = A.A + A.B$$
$$= A + A.B \quad \text{Rule 1}$$
$$= A \qquad \text{Rule 10}$$

25.17. DEMORGAN'S THEOREMS

DeMorgan proposed two theorems which are an important part of Boolean algebra.

Theorem 1. $\overline{AB} \equiv \overline{A} + \overline{B}$

It states that the complement of two or more variables ANDed is same as the OR of the complements of each individual variable.

The theorem implies that a NAND gate performs the same operation as an OR gate whose inputs are inverted:

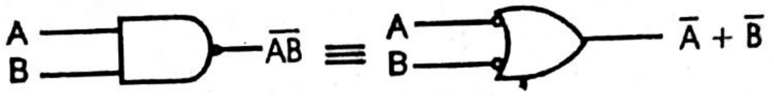


Fig 25.25

A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

=

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

Theorem 2: $\overline{A+B} = \overline{A}\,\overline{B}$.

It states that the complement of two or more variable ORed is the same as the AND of the complements of each individual variable.

This theorem implies that a NOR gate performs the same operation as an AND gate whose inputs are inverted.

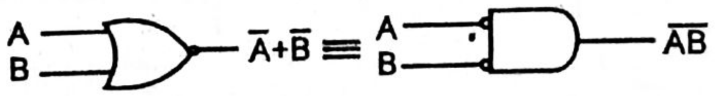


Fig 25.26

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

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \circ \overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

It is clear from DeMorgan's first theorem that all logic operations can be performed using only AND and NOT gates. Since a NAND gate is a combination of a NOT and an AND gate, it may be concluded that any logical function can be carried out by using repeatedly the NAND gate alone. We see below how the three basic logic gates can be produced from NAND gate.

(i) NOT gate from NAND gate:

If the two inputs of a 2-input NAND gate are connected together, it will have one input and acts as a NOT gate.

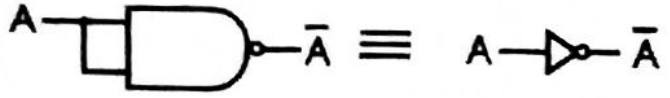


Fig 25.27

(ii) AND gate from NAND gate:

It requires two NAND gates. If the output of first NAND is given to the second NAND gate acting as an inverter, the resulting circuit is the AND gate.

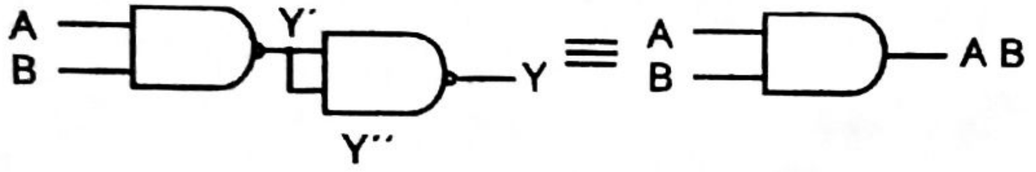


Fig 25.28

The output of first NAND gate is inverted output of AND gate. The second NAND gate with its inputs joined together acts as an inverter and further inverts its input so that the final output is that of an AND gate.

(iii) OR gate from NAND gate:

This requires three NAND gates. The first two NAND gates are operated as NOT gates by connecting together the inputs of each NAND gate. The outputs of these NOT gates are given to the third NAND gate. The resulting circuits is an OR gate.

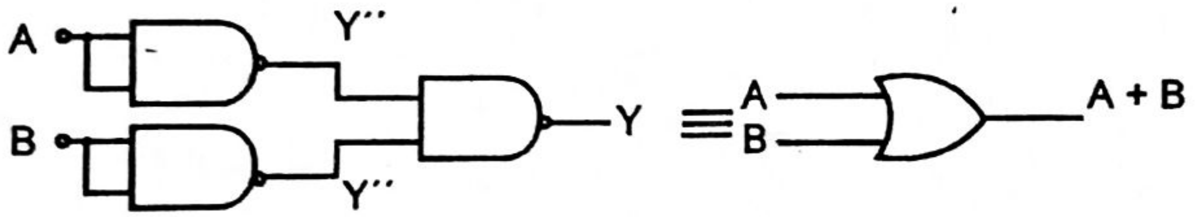


Fig 25.29

A similar argument using DeMorgan's second theorem it can be shown that all logic operations can be performed by using NOR circuit repeatedly.

The NAND and NOR gates are *universal gates* because their repeated use can produce all other logic gates. Hence they are building blocks of all digital circuits.