LOGIC GATES:

It is the logic circuit having one or more than one input **but only one output**. Output is either high i.e. 1, or low i.e. 0.

TRUTH TABLES:

It is the representation of Input and output relationships of any logic gate.

Mathematically,

For n number of inputs, we have 2^n number of input row combination.

e. g.

n=1, then total Input row combination=$2^n=2^1=2$

n=2, then total Input row combination=$2^n=2^2=4$

n=3, then total Input row combination=$2^n=2^3=8$

n=4, then total Input row combination=$2^n=2^4=16$

INPUT

n=1, A be the Boolean variable;

n=1, then total Input row combination=$2^n=2^1=2$

Input row combination

A
0
1

INPUT

n=2, A & B be the Boolean variable;

n=2, then total Input row combination=$2^n=2^2=4$

Input row combination

A	B
0	0
0	1
1	0
1	1

n=3, A, B & C be the Boolean variable;

n=3, then total Input row combination=$2^n=2^3=8$

Input row combination

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

n=4 A, B, C & D be the Boolean variable;

n=4, then total Input row combination=$2^n=2^4=16$

Input row combination

A	B	C	D
0	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	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

n=5 A, B, C, D & E be the Boolean variable;

n=5, then total Input row combination=$2^n = 2^5 = 32$

Input row combination

A	B	C	D	E
0	0	0	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	1	1
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	0	1	1	1
0	1	0	0	0
0	1	0	0	1
0	1	0	1	0
0	1	0	1	1
0	1	1	0	0
0	1	1	0	1
0	1	1	1	0
0	1	1	1	1
1	0	0	0	0
1	0	0	0	1
1	0	0	1	0
1	0	0	1	1
1	0	1	0	0
1	0	1	0	1
1	0	1	1	0
1	0	1	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	0
1	1	0	1	1
1	1	1	0	0
1	1	1	0	1
1	1	1	1	0
1	1	1	1	1

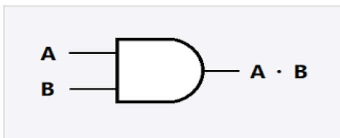
TYPES OF LOGIC GATES:

1. Basic gates
2. Universal gates
3. Additional gates
    1. **Basic gates**
        a. AND gate
        b. OR gate
        c. NOT gate

a. **AND gate:**

It is the logic circuit having two or more than two input **but only one output**. Output is high i.e. 1, when **all its input** is high, otherwise, output is low i.e. 0.

i. **Logic Symbol**:



ii. **Logic Expression**

Y=A.B

iii. **Truth Table**

Mathematically,

For n number of inputs, we have 2^n number of input row combination.

Since, n=2, then total Input row combination=$2^n=2^2=4$

INPUT		OUTPUT
A	**B**	**Y=A.B**
0	0	0
0	1	0
1	0	0
1	1	1

. 2-INPUT AND gate

iv. **Conclusion**

"Output is high i.e. 1, when **all its input** is high, otherwise, output is low **i.e. 0.**"

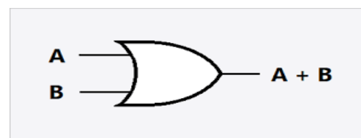
OR

"Output is low i.e. 0, when **atleast one of its input** is low, otherwise, output is high **i.e. 1.**"

b. OR gate:

It is the logic circuit having two or more than two input **but only one output**. Output is high i.e. 1, when **atleast one of its input** is high, otherwise, output is low i.e. 0.

i. **Logic Symbol**:



ii. **Logic Expression**

Y=A+B

iii. **Truth Table**

Mathematically,

For n number of inputs, we have 2^n number of input row combination.

Since, n=2, then total Input row combination=$2^n=2^2=4$

INPUT		OUTPUT
A	**B**	**Y=A+B**
0	0	0
0	1	1
1	0	1
1	1	1

. 2-INPUT OR gate

iv. **Conclusion**

"Output is high i.e. 1, when **atleast one of its input** is high, otherwise, output is low **i.e. 0."**

OR

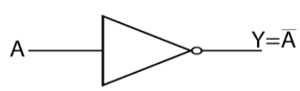
"Output is LOW i.e. 0, when **all its input** is low, otherwise, output is high **i.e. 1."**

c. **NOT gate:**

It is the logic circuit having one input **and only one output**. It is also known as **Inverter**.

Output is high i.e. 1, when **input** is low, output is low i.e. 0, when **input** is high.

i. **Logic Symbol**:



ii. **Logic Expression**

Y=A' or Y=A^c

iii. **Truth Table**

Mathematically,

For n number of inputs, we have 2^n number of input row combination.

Since, n=1, then total Input row combination=2^n=2^1=2

INPUT	OUTPUT
A	**Y=A'**
0	1
1	0

. NOT gate

iv. **Conclusion**

"Output is high i.e. 1, when **input** is low, output is low i.e. 0, when **input** is high."

OR

"Output is LOW i.e. 0, when **input** is high, output is high i.e. 1, when **input** is low."

2. **Universal Gates**

It is the logic gate through which we can design or realize any basic gate (i.e. **AND, OR, & NOT**). There are two types of universal gate.

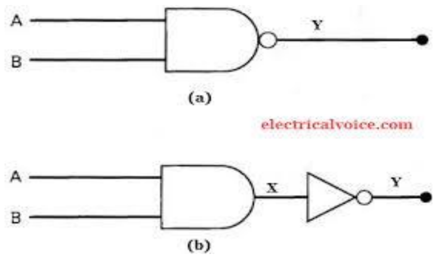
d. NAND gate

e. NOR gate

d. **NAND gate:**

It is the logic circuit having two or more than two input **but only one output**. It is also known as **NOT (AND)** gate. The logic circuit of NAND gate is designed by using AND gate followed by NOT gate. So, the output of NAND gate is just opposite to the output of AND gate. Output is **low i.e. 0**, when **all its input** is high, otherwise, output is high i.e. 1.

i. **Logic Symbol**:



ii. **Logic Expression**

Y=(A.B)’

iii. **Truth Table**

Mathematically,

For n number of inputs, we have 2^n number of input row combination.

Since, n=2, then total Input row combination=2^n =2^2=4

INPUT		OUTPUT
A	**B**	**Y=(A.B)’**
0	0	1
0	1	1
1	0	1
1	1	0

. 2-INPUT NAND gate

INPUT			OUTPUT
A	**B**	**A.B**	**Y=(A.B)’**
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Detailed truth table

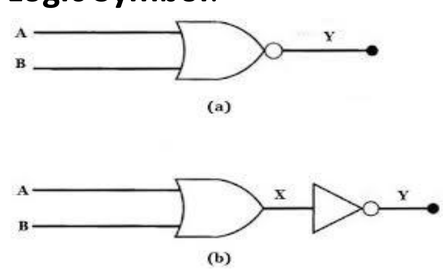
iv. **Conclusion**

“Output is low i.e. 0, when **all its input** is high, otherwise, output is high i.e. **1.”**

e. NOR gate:

It is the logic circuit having two or more than two input **but only one output**. It is also known as **NOT (OR)** gate. The logic circuit of NOR gate is designed by using OR gate followed by NOT gate. So, the output of NOR gate is just opposite to the output of OR gate. Output is **high i.e. 1**, when **all its input** is low, otherwise, output is low i.e. 0.

i. **Logic Symbol**:



ii. **Logic Expression**

Y= (A+B)’

Truth Table

Mathematically,

For n number of inputs, we have 2^n number of input row combination.

Since, n=2, then total Input row combination=2^n =2^2=4

INPUT		OUTPUT
A	B	Y=(A+B)’
0	0	1
0	1	0
1	0	0
1	1	0

. 2-INPUT NOR gate

INPUT			OUTPUT
A	B	A+B	Y=(A+B)’
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Detailed truth table

iii. Conclusion

“Output is **high i.e. 1**, when **all its input** is low, otherwise, output is low i.e. 0.**”**

Question 1:

Draw the logic circuit, and give the truth table of the following expression using basic gates:

1. Y=AB+(AB)’
2. P=AB+(A+B)’
3. W=(A+B)+A’B’
4. Y=A’B+AB’

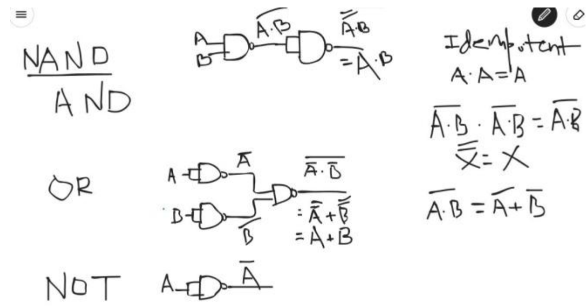
Realization of basic gates from universal gate:

1. From NAND gate
2. From NOR gate

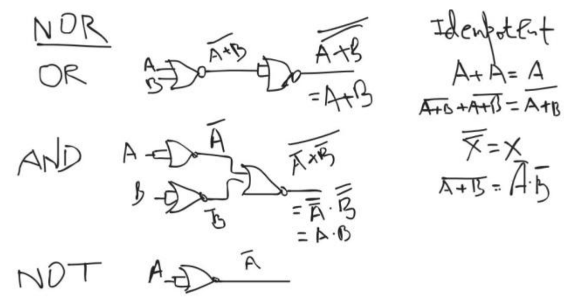
Table for conversion of basic gates

BG → UG ↓	AND	OR	NOT
NAND	2-NAND	3-NAND	1-NAND
NOR	3-NOR	2-NOR	1-NOR

1. Realization of basic gates from NAND gates



2. Realization of basic gates from NOR gates



Question 2:

Draw the logic circuit, and give the truth table of the following expression using:

i) Only NAND gates

ii) Only NOR gates.

1. Y=AB+(AB)’
2. P=AB+(A+B)’
3. W=(A+B)+A’B’
4. Y=A’B+AB’

4. Additional Gates:

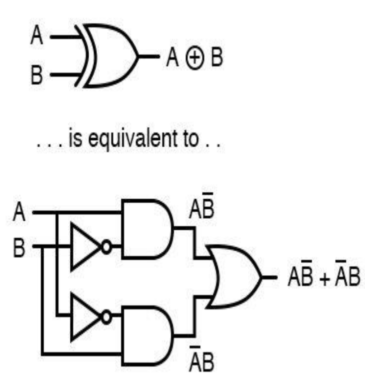
f. XOR gate

g. XNOR gate

f. XOR gate

It is the logic circuit having two or more than two input **but only one output**. It is also known as **Exclusive OR** gate. The logic circuit of XOR gate is designed by using all basic gates i.e. AND, OR, and NOT gate. Output is **high i.e. 1**, for **odd number** of **1** in the input combination, output is **low i.e. 0** for **even number** of **1** in the input combination.

i. **Logic Symbol**:



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

ii. Logic Expression

Y=AB'+ A'B

iii. Truth Table

Mathematically,

For n number of inputs, we have 2^n number of input row combination.

Since, n=2, then total Input row combination=$2^n=2^2=4$

INPUT		OUTPUT
A	**B**	**Y=A'B+AB'**
0	0	0
0	1	1
1	0	1
1	1	0

. 2-INPUT XOR gate

INPUT						OUTPUT
A	**B**	**A'**	**B'**	**A'B**	**AB'**	**Y=A'B+AB'**
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	0	0	0

Detailed truth table

iv. Conclusion

"Output is **high i.e. 1**, for **odd number** of one in the input combination, output is **low i.e. 0** for **even number** of one in the input combination."

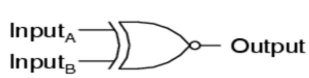
g. XNOR gates

It is the logic circuit having two or more than two input **but only one output**. It is also known as **Exclusive NOR or EX-NOR** gate.

The logic circuit of XNOR gate is designed by using **XOR gate followed by NOT gate**. It is also known as **NOT (XOR)** gate. So, the output of XNOR gate is just opposite to the output of XOR gate. Output is **high i.e. 1**, for **even number** of one in the input combination, output is **low i.e. 0** for **odd number** of one in the input combination.

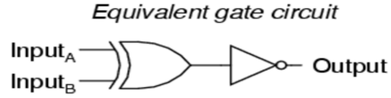
i. **Logic Symbol**:

Exclusive-NOR gate



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Equivalent gate circuit



ii. **Logic Expression**

Y=(AB'+ A'B)'

Or

Y=(AB+A'B')

iii. **Truth Table**

Mathematically,

For n number of inputs, we have 2^n number of input row combination.

Since, n=2, then total Input row combination=$2^n = 2^2 = 4$

INPUT		OUTPUT
A	**B**	**Y=(A'B+AB')'**
0	0	1
0	1	0
1	0	0
1	1	1

. 2-INPUT XNOR gate

INPUT							OUTPUT
A	B	A'	B'	A'B	AB'	(A'B+AB')	Y=(A'B+AB')'
0	0	1	1	0	0	0	1
0	1	1	0	1	0	1	0
1	0	0	1	0	1	1	0
1	1	0	0	0	0	0	1

Detailed truth table

iv. **Conclusion**

"Output is **high i.e. 1**, for **even number** of one in the input combination, output is **low i.e. 0** for **odd number** of one in the input combination."

Question 3:

Give the truth Table for 3-input XOR gate and give the logic expression for the same.

Solution:

INPUT			OUTPUT
A	**B**	**C**	$Y=A \oplus B \oplus C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Y=A’B’C+A’BC’+AB’C’+ABC

OR

Y=A XOR B XOR C

OR

$Y=(A \oplus B \oplus C)'$

Question 4:

The output of a logic gate is ‘1’ when all its input is at logic 0.The gate is either

(a) NAND or an XOR gate

(b) NOR or an XNOR gate

(c) an OR or an XNOR gate

(d) an AND or an XOR gate

Question 5:

Do as directed:

F=AB+A’B’

F=AB+A’B’

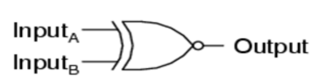
1. Give the truth table.
2. Identify the gate for which the above expression we have.
3. Give the logic symbol.
4. Draw the logic circuit of the above expression using basic gates.
5. Draw the logic circuit of the above expression using only NAND gates.
6. Draw the logic circuit of the above expression using only NOR gates.

Solution:

INPUT						OUTPUT
A	B	A’	B’	AB	A’B’	AB+A’B’
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1

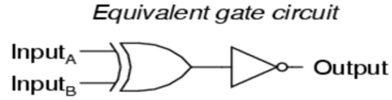
XNOR GATE

Exclusive-NOR gate



A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Equivalent gate circuit



Question 6:

Do as directed

For any Boolean expression

F= A'B'C+A'BC'+AB'C'+ABC

1. Give the truth table.
2. Identify the name of logic gate, for which the equivalent output is generated from the above expression.
3. Give the standard logic symbol for the above expression.
4. Draw the logic circuit for the above expression using BASIC gates.
5. Draw the logic circuit for the above expression using only NAND gates.
6. Draw the logic circuit for the above expression using only NOR gates.

Question 7:

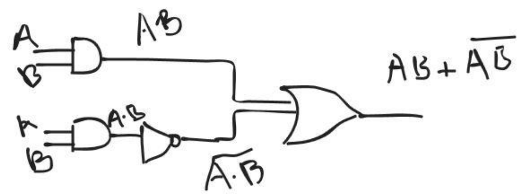
Do as directed

For any Boolean expression

F= A'B'C'+A'BC+AB'C+ABC

1. Give the truth table.
2. Identify the name of logic gate, for which the equivalent output is generated from the above expression.
3. Give the standard logic symbol for the above expression.
4. Draw the logic circuit for the above expression using BASIC gates.
5. Draw the logic circuit for the above expression using only NAND gates.
6. Draw the logic circuit for the above expression using only NOR gates.

$AB + \overline{AB}$



INPUT				OUTPUT
A	**B**	**AB**	**(AB)’**	**Y=AB+(AB)’**
0	0	0	1	1
0	1	0	1	1
1	0	0	1	1
1	1	1	0	1

INPUT				OUTPUT
A	**B**	**AB**	**(A+B)’**	**Y=AB+(A+B)’**
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

Y=AB+A’B’

F=AB+(A+B)’

Demorgan’s theorem

(A+B)’=A’.B’

BOOLEAN ALGEBRA:

Boolean algebra was invented by **George Boole** in 1854. In mathematics and mathematical **logic**, **Boolean algebra** is the branch of **algebra** in which the values of the variables are the truth values **true** and **false**, usually denoted 1 and 0, respectively. It is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1. It is also called as **Binary Algebra** or **logical Algebra**.

Switching circuits are also called logic circuits, gate circuits, and digital circuits. Boolean Algebra is also known as **Switching Algebra**. In other words, it is a way to express logic functions algebraically. Set of rules to simplify any Boolean expression is known as **Boolean Law**. Various **Boolean laws** are as follows.

1. Identity law

This law states that, for any Boolean variable A,

- **i. A+1=1**
- **ii. A.1=A**
- **iii. A+0=A**
- **iv. A.0=0**

Proof:

i. **A+1=1**

By means of truth table,

A	**1**	**A+1**
0	1	1
1	1	1

Since, the output of column iii is equal to 1 for any set of input combination, so, identity law i.e. **A+1=1** is proved.

ii. **A.1=A**

By means of truth table,

A	**1**	**A.1**
0	1	0
1	1	1

Since, the output of column i and column iii is identical, so, identity law i.e. **A.1=A** is proved.

iii.**A+0=A**

By means of truth table,

A	**0**	**A+0**
0	0	0
1	0	1

Since, the output of column i and column iii is identical, so, identity law i.e. **A+0=A** is proved.

iv.**A.0=0**

By means of truth table,

A	0	A.0
0	0	0
1	0	0

Since, the output of column iii is equal to 0 for any set of input combination, so, identity law i.e. **A.0=0** is proved.

2. **Inverse law**

This law states that, for any Boolean variable A,

v. A+A'=1

vi. A.A'=0

v. **A+A'=1**

By means of truth table,

A	A'	A+A'
0	1	1
1	0	1

Since, the output of column iii is equal to 1 for any set of input combination, so, INVERSE law i.e. **A+A'=1** is proved.

vi. **A.A'=0**

By means of truth table,

A	A'	A.A'
0	1	0
1	0	0

Since, the output of column iii is equal to 0 for any set of input combination, so, INVERSE law i.e. **A.A'=0** is proved.

3. **Double complementation law**

This law states that, for any Boolean variable X, the double complementation of that variable is equivalent to variable itself.

vii. (X')'=X

Proof:

By means of truth table,

X	X'	(X')'
0	1	0
1	0	1

Since, the output of column i and column iii is identical, so, double complementation law i.e. **(X')'=X** is proved.

4. **Idempotent law**

This law states that, for any Boolean variable A,

viii. A+A=A

ix. A.A=A

Proof:

viii. A+A=A

By means of truth table,

A	A	A+A
0	0	0
1	1	1

Since, the output of column i and column iii is identical, so, Idempotent law for addition i.e. **A+A=A** is proved.

ix. A.A=A

By means of truth table,

A	A	A.A
0	0	0
1	1	1

Since, the output of column i and column iii is identical, so, Idempotent law for multiplication i.e. **A.A=A** is proved.

5. **Absorption law**

x. A+A.B=A

Proof:

By means of truth table,

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

Since, the output of column i and column iv is identical, so, absorption law i.e. **A+A.B=A** is proved.

xi. A.(A+B)=A

By means of truth table,

A	B	A+B	A.(A+B)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Since, the output of column i and column iv is identical, so, absorption law i.e. **A.(A+B)=A** is proved.

xii. A+A'B=A+B

By means of truth table,

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

Since, the output of column v and column vi is identical, so, absorption law i.e. **A+(A'.B)=A+B** is proved.

xiii. A.(A'+B)=AB

By means of truth table,

A	B	A'	A'+B	A.(A'+B)	A.B
0	0	1	1	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	1	0	1	1	1

Since, the output of column v and column vi is identical, so, absorption law i.e. **A.(A'+B)=AB** is proved.

6. **Commutative law**

xiv. A+B=B+A

Proof:

By means of truth table,

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

Since, the output of column iii and column iv is identical, so, commutative law for addition i.e. **A+B=B+A** is proved.

xv. A.B=B.A

By means of truth table,

A	B	A.B	B.A
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Since, the output of column iii and column iv is identical, so, commutative law for multiplication i.e. **A.B=B.A** is proved.

7. **Associative law**

xvi. A+(B+C)=(A+B)+C

Proof:

By means of truth table,

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

Since, the output of column vi and column vii is identical, so, associative law i.e. **A+(B+C)=(A+B)+C** is proved.

xvii. A.(B.C)=(A.B).C

By means of truth table,

A	B	C	A.B	B.C	A.(B.C)	(A.B).C
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Since, the output of column vi and column vii is identical, so, associative law i.e. **A.(B.C)=(A.B).C** is proved.

8. **Distributive law**

xviii. A.(B+C)=A.B+A.C

Proof:

By means of truth table,

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

Since, the output of column v and column viii is identical, so, distributive law i.e. **A.(B+C)=A.B+A.C** is proved.

xix. A+(B.C)=(A+B).(A+C)

By means of truth table,

A	B	C	B.C	A+(B.C)	A+B	A+C	(A+B).(A+C)
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

Since, the output of column v and column viii is identical, so, distributive law i.e. **A+(B.C)=(A+B).(A+C)** is proved.

BOOLEAN POSTULATES:

Boolean Postulates is a set of rule designed by the help of basic gates i.e. AND, OR, & NOT gate.

1. 0.0=0
2. 0.1=0
3. 1.0=0
4. 1.1=1

derived from AND operation

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

derived from OR operation

9. 0'=1
10. 1'=0

derived from NOT operation

BOOLEAN THEOREMS:

1. **A+1=1**

Proof:

By means of Boolean postulates,

Let A=0,

LHS=A+1

=0+1 (By postulate no. 6)

=1=RHS

Again, let A=1,

LHS=A+1

=1+1 (By postulate no. 8)

=1=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A+1=1 is proved.

2. **A.1=A**

Proof:

By means of Boolean postulates,

Let A=0,

LHS=A.1

=0.1 (By postulate no. 2)

=0=A=RHS

Again, let A=1,

LHS=A.1

=1.1 (By postulate no. 4)

=1=A=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A.1=A is proved.

3. **A+0=A**

Proof:

By means of Boolean postulates,

Let A=0,

LHS=A+0

=0+0 (By postulate no. 5)

=0=A=RHS

Again, let A=1,

LHS=A+0

=1+0 (By postulate no. 7)

=1=A=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A+0=A is proved.

4. **A.0=0**

Proof:

By means of Boolean postulates,

Let A=0,

LHS=A.0

=0.0 (By postulate no. 1)

=0=RHS

Again, let A=1,

LHS=A.0

=1.0(By postulate no. 3)

=0=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A.0=0 is proved.

5. **A+A’=1**

Proof:

By means of Boolean postulates,

Let A=0,

LHS=A+A’

=0+0’ (By p.n. 9)

=0+1 (By p.n. 6)

=1=RHS

Again, Let A=1,

LHS=A+A’

=1+1’ (By p.n. 10)

=1+0 (By p.n. 7)

=1=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A+A'=1 is proved.

6. **A.A'=0**

Proof:

By means of Boolean postulates,

Let A=0,

LHS=A.A'

= 0.0' (By Postulate no.9)

= 0.1 (By postulate no. 2)

=0=RHS

Again, Let A=1,

LHS= A.A'

=1.1' (BY Postulate no. 10)

=1.0 (By Postulate no. 3)

=0=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A.A'=0 is proved.

7. **(X')'=X**

Proof:

By means of Boolean postulates,

Let X=0,

LHS=(X')'

= (0')' (By Postulate no.9)

= 1' (By postulate no. 10)

=0=X=RHS

Again, Let X=1,

LHS=(X')'

= (1')' (By Postulate no.10)

= 0' (By postulate no. 9)

=1=X=RHS

Since, LHS=RHS for X=0 as well as X=1.

Therefore, (X')'=X is proved.

8. **A+A=A**

Proof:

By means of Boolean postulates,

Let A=0,

LHS=A+A

=0+0 (By Postulate no. 5)

=0=A=RHS

Again, Let A=1, then

LHS=A+A

=1+1 (By Postulate no. 8)

=1=A=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A+A=A is proved.

9. **A.A=A**

Proof:

By means of Boolean postulates,

Let A=0,

LHS=A.A

=0.0 (By Postulate no. 1)

=0=A=RHS

Again, Let A=1, then

LHS=A.A

=1.1 (By Postulate no. 4)

=1=A=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A.A=A is proved.

10.**A+A.B=A**

Proof:

By means of Boolean postulates,

Let A=B=0,

LHS=A+A.B

=0+0.0 (By Postulate no. 1)

=0+0 (By Postulate no. 5)

=0=A=RHS

Again, Let A=B=1, then

LHS=A+A.B

=1+1.1 (By Postulate no. 4)

=1+1 (By Postulate no. 8)

=1=A=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A+A.B=A is proved.

11. **A.(A+B)=A**

Proof:

By means of Boolean postulates,

Let A=B=0,

LHS=A.A+B

=0.0+0 (By Postulate no. 5)

=0.0 (By Postulate no. 1)

=0=A=RHS

Again, Let A=B=1, then

LHS=A.A+B

=1.1+1 (By Postulate no. 8)

=1.1 (By Postulate no. 4)

=1=A=RHS

Since, LHS=RHS for A=0 as well as A=1.

Therefore, A.(A+B)=A is proved.

12.**A+A'B=A+B**

Proof:

By means of Boolean postulates,

Let A=B=0,

LHS=A+A'.B

=0+0'.0 (By Postulate no. 9)

=0+1.0 (By Postulate no. 3)

=0+0 (By Postulate no. 5)

=0

RHS=A+B

=0+0 (By Postulate no. 5)

=0

Again, Let A=B=1, then

LHS=A+A'.B

=1+1'.1 (By Postulate no. 10)

=1+0.1 (By Postulate no. 2)

=1+0 (By Postulate no. 7)

=1

RHS=A+B

=1+1 (By Postulate no. 8)

=1

Since, LHS=RHS

Therefore, A+A'.B=A+B is proved.

13. **A.(A'+B)=AB**

Proof:

By means of Boolean postulates,

Let A=B=0,

LHS=A.(A'+B)
=0.(0'+0) (By Postulate no. 9)
=0.(1+0) (By Postulate no. 7)
=0.1 (By Postulate no. 2)
=0
RHS=A.B
=0.0 (By Postulate no. 1)
=0
Again, Let A=B=1, then
LHS=A.(A'+B)
=1.(1'+1) (By Postulate no. 10)
=1.(0+1) (By Postulate no. 6)
=1.1 (By Postulate no. 4)
=1
RHS=A.B
=1.1 (By Postulate no. 4)
=1
Since, LHS=RHS
Therefore, A+.(A'+B)=A.B is proved.

14. **A+B=B+A**

Proof:
By means of Boolean postulates,
Let A=B=0,
LHS=A +B
=0+0 (By Postulate no. 5)
=0
RHS=B+A
=0+0 (By Postulate no. 5)
=0
Again, Let A=B=1, then
LHS=A +B
=1+1 (By Postulate no. 8)
=1
RHS=B+A
=1+1 (By Postulate no. 8)
=1
Since, LHS=RHS
Therefore, A +B=B+A is proved.

15. **A.B=B.A**

Proof:
By means of Boolean postulates,
Let A=B=0,
LHS=A .B
=0.0 (By Postulate no. 1)
=0
RHS=B.A
=0.0 (By Postulate no. 1)
=0
Again, Let A=B=1, then
LHS=A .B
=1.1 (By Postulate no. 4)
=1
RHS=B.A
=1.1 (By Postulate no. 4)
=1
Since, LHS=RHS
Therefore, A.B=B.A is proved.

16. **A+(B+C)=(A+B)+C**

Proof:
By means of Boolean postulates,

Let A=B=C=0, then,

LHS=A+(B+C)

=0+(0+0) (By Postulate no. 5)

=0+0 (By Postulate no. 5)

=0

RHS= (A+B)+C

=(0+0)+0 (By Postulate no. 5)

=0+0 (By Postulate no. 5)

=0

Similarly, Let A=B=C=1, then,

LHS=A+(B+C)

=1+(1+1) (By Postulate no. 8)

=1+1 (By Postulate no. 8)

=1

RHS=(A+B)+C

=(1+1)+1 (By Postulate no. 8)

=1+1 (By Postulate no. 8)

= 1

Since, LHS=RHS

Therefore, A+(B+C)=(A+B)+C is proved.

17. **A.(B.C)=(A.B).C**

Proof:

By means of Boolean postulates,

Let A=B=C=0, then,

LHS=A.(B.C)

=0.(0.0) (By Postulate no. 1)

=0.0 (By Postulate no. 1)

=0

RHS= (A.B).C

=(0.0).0 (By Postulate no. 1)

=0.0 (By Postulate no. 1)

=0

Similarly, Let A=B=C=1, then,

LHS=A.(B.C)

=1.(1.1) (By Postulate no. 4)

=1.1 (By Postulate no. 4)

=1

RHS=(A.B).C

=(1.1).1 (By Postulate no. 4)

=1.1 (By Postulate no. 4)

= 1

Since, LHS=RHS

Therefore, A.(B.C)=(A.B).C is proved.

18. **A.(B+C)=A.B+A.C**

Proof:

By means of Boolean postulates,

Let A=B=C=0, then,

LHS=A.(B+C)

=0.(0+0) (By Postulate no. 5)

=0.0 (By Postulate no. 1)

=0

RHS=A.B+A.C

=0.0+0.0 (By Postulate no. 1)

=0+0 (By Postulate no. 5)

= 0

Similarly, Let A=B=C=1, then,

LHS=A.(B+C)

=1.(1+1) (By Postulate no. 8)

=1.1 (By Postulate no. 4)

=1

RHS=A.B+A.C

=1.1+1.1 (By Postulate no. 4)

=1+1 (By Postulate no. 8)

= 1

Since, LHS=RHS

Therefore, A.(B+C)=A.B+A.C is proved.

19. **A+(B.C)=(A+B).(A+C)**

Proof:

By means of Boolean postulates,

Let A=B=C=0, then,

LHS=A+(B.C)

=0+(0.0) (By Postulate no. 1)

=0+0 (By Postulate no. 5)

=0

RHS=(A+B)(A+C)

=(0+0)(0+0) (By Postulate no. 5)

=0.0 (By Postulate no. 1)

= 0

Similarly, Let A=B=C=1, then,

LHS=A+(B.C)

=1+(1.1) (By Postulate no. 4)

=1+1 (By Postulate no. 8)

=1

RHS=(A+B)(A+C)

=(1+1)(1+1) (By Postulate no. 8)

=1.1 (By Postulate no. 4)

= 1

Since, LHS=RHS

Therefore, A+(B.C)=(A+B)(A+C) is proved.

DEMORGAN'S THEOREM:

Demorgan's theorem states that,

1. Complement of the union (i.e. OR operation) is equivalent to intersection (i.e. AND operation) of complements.

For any two Boolean variable A and B,

(A+B)'=A'.B'

Proof:

By means of Boolean postulates,

Let A=B=0, then

LHS=(A+B)'

=(0+0)' (By Postulate no. 5)

= 0' (By Postulate no. 9)

=1

RHS=A'.B'

=0'.0' (By Postulate no. 9)

= 1.1 (By Postulate no. 4)

=1

Similarly, Let A=B=1, then

LHS=(A+B)'

=(1+1)' (By Postulate no. 8)

= 1' (By Postulate no. 10)

=0

RHS=A'.B'

=1'.1' (By Postulate no. 10)

= 0.0 (By Postulate no. 1)

=0

Since, LHS=RHS

Therefore,

(A+B)’=A’.B’ is proved.

OR

Proof:

By means of Truth Table,

A	B	A’	B’	A’.B’	A+B	(A+B)’
0	0	1	1	1	0	1
0	1	1	0	0	1	0
1	0	0	1	0	1	0
1	1	0	0	0	1	0

Since, the output of column v and column vii is identical.

Therefore, **(A+B)’=A’.B’** is proved.

2. Complement of the intersection (i.e. AND operation) is equivalent to union (i.e. OR operation) of complements.

For any two Boolean variable A and B,

(A.B)’=A’+B’

Proof:

By means of Boolean postulates,

Let A=B=0, then

LHS=(A.B)’

=(0.0)’ (By Postulate no. 1)

= 0’ (By Postulate no. 9)

=1

RHS=A’+B’

=0’+0’ (By Postulate no. 9)

= 1+1 (By Postulate no. 8)

=1

Similarly, Let A=B=1, then

LHS=(A.B)’

=(1.1)’ (By Postulate no. 4)

= 1’ (By Postulate no. 10)

=0

RHS=A’+B’

=1’+1’ (By Postulate no. 10)

= 0+0 (By Postulate no. 5)

=0

Since, LHS=RHS

Therefore,

(A.B)’=A’+B’ is proved.

OR

Proof:

By means of Truth Table,

A	B	A’	B’	A’+B’	A.B	(A.B)’
0	0	1	1	1	0	1
0	1	1	0	1	0	1
1	0	0	1	1	0	1
1	1	0	0	0	1	0

Since, the output of column v and column vii is identical.

Therefore, **(A.B)’=A’+B’** is proved.

Principle of Duality or Dual of Boolean Expression:

Principle of duality is used to derive a new Boolean function from the existing Boolean function by applying following change:

1. All ‘0’ is converted into ‘1’
2. All ‘1’ is converted into ‘0’
3. All ‘+’ is converted into ‘.’
4. All ‘.’ is converted into ‘+’

e.g.

A.B’+A+B+1

Dual:

A+B’.A.B.0

Complement of Boolean Expression:

To find out complement of any Boolean expression following steps are to be followed.

1. All ‘0’ is converted into ‘1’
2. All ‘1’ is converted into ‘0’
3. All ‘+’ is converted into ‘.’
4. All ‘.’ is converted into ‘+’
5. Literals are also complemented.

   e.g.

   A.B’+A+B+1

   Dual:

   A+B’.A.B.0

   Complement:

   A’+B.A’.B’.0

Q. Prove Demorgan’s theorem for three variables.

1. using truth table
2. using Boolean postulates

OR

Q. Prove Demorgan’s theorem for three variable USING Truth table.

OR

Q. Prove Demorgan’s theorem for three variable USING Boolean postulates.

OR

Q. Prove Demorgan’s theorem for three variables.

OR

Q. (a) Prove that: (A+B+C)’=A’.B’.C’

(b) Prove that: (A.B.C)’=A’+B’+C’

Solution: (a) (A+B+C)’=A’.B’.C’

Proof:

By means of Boolean postulates,

Let A=B=C=0, then

LHS=(A+B+C)’

=(0+0+0)’ (By Postulate no. 5)

= (0+0)’ (By Postulate no. 5)

=0’ (By Postulate no. 9)

=1

RHS=A’.B’.C’

=0’.0’.0’ (By Postulate no. 9)

= 1.1.1 (By Postulate no. 4)

=1.1 (By Postulate no. 4)

=1

Similarly, Let A=B=C=1, then

LHS=(A+B+C)’

=(1+1+1)’ (By Postulate no. 8)

= (1+1)’ (By Postulate no. 8)

=1’ (By Postulate no. 10)

=0

RHS=A'.B'.C'

=1'.1'.1' (By Postulate no. 10)

= 0.0.0 (By Postulate no. 1)

=0.0 (By Postulate no. 1)

=0

Since, LHS=RHS

Therefore,

(A+B+C)'=A'.B'.C' is proved.

OR

Proof:

By means of Truth Table,

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

Since, the output of column vii and column ix is identical.

Therefore, **(A+B+C)'=A'.B'.C'** is proved.

Solution: (b)(A.B.C)'=A'+B'+C'

Proof:

By means of Boolean postulates,

Let A=B=C=0, then

LHS=(A.B.C)'

=(0.0.0)' (By Postulate no. 1)

= (0.0)' (By Postulate no. 1)

=0' (By Postulate no. 9)

=1

RHS=A'+B+C'

=0'+0'+0' (By Postulate no. 9)

= 1+1+1 (By Postulate no. 8)

=1+1 (By Postulate no. 8)

=1

Similarly, Let A=B=C=1, then

LHS=(A.B.C)'

=(1.1.1)' (By Postulate no. 4)

= (1.1)' (By Postulate no. 4)

=1' (By Postulate no. 10)

=0

RHS=A'+B'+C'

=1'+1'+1' (By Postulate no. 10)

= 0+0+0 (By Postulate no. 5)

=0+0 (By Postulate no. 5)

=0

Since, LHS=RHS

Therefore,

(A.B.C)'=A'+B'+C' is proved.

OR

Proof:

By means of Truth Table,

A	B	C	A'	B'	C'	A'+B'+C'	A.B.C	(ABC)'
0	0	0	1	1	1	1	0	1
0	0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0	1
0	1	1	1	0	0	1	0	1
1	0	0	0	1	1	1	0	1
1	0	1	0	1	0	1	0	1
1	1	0	0	0	1	1	0	1
1	1	1	0	0	0	0	1	0

Since, the output of column vii and column ix is identical.

Therefore, **(A.B.C)'=A'+B'+C'** is proved.

e.g.

i) Prove that: **A'B+A'B'=A'**

(use Boolean theorem/use truth table/use Boolean postulate)

Proof:

By using Boolean theorems,

LHS= A'B+A'B'

=A'(B+B') (**By inverse law i.e. X+X'=1**)

= A'.1 (**By Identity law i.e. A.1=A**)

=A'=RHS

Since, LHS=RHS.

Therefore, A'B+A'B'=A' is proved.

OR

Proof:

By means of truth table,

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

Since the output of column iii and column vii is identical.

Therefore, A'B+A'B'=A' is proved.

OR

Proof:

By means of Boolean postulates,

Let A=B=0, then,

LHS= A'B+A'B'

=0'.0+0'.0' (By postulate no. 9)

=1.0+1.1 (By postulate no. 4)

=1.0+1 (By postulate no. 3)

=0+1 (By postulate no. 6)

=1=A'=RHS

Let A=B=1, then,

LHS= A'B+A'B'

=1'.1+1'.1' (By postulate no. 10)

=0.1+0.0 (By postulate no. 1)

=0.1+0 (By postulate no. 2)

=0+0 (By postulate no. 5)

=0=A'=RHS

Since, LHS=RHS.

Therefore, A'B+A'B'=A' is proved.

Q1. Prove the identity **A(A+A')+B=A+B** using Boolean theorem.

Solution:

LHS= A(A+A')+B (**By inverse law i.e. X+X'=1**)

=A.1+B (**By Identity law i.e. A.1=A**)

=A+B=RHS

Since, LHS=RHS.

Therefore, A(A+A')+B =A+B is proved.

Simplification of Boolean expression by using Boolean theorems:

Every Boolean expression must be reduced to as simple a form as possible before realization, because every logic operation in the expression represents a corresponding element of hardware. Realization of a digital circuit with minimal expression, therefore, results in reduction of cost and complexity and the corresponding increase in reliability. To reduce Boolean expressions, all the laws of Boolean algebra may be used. The techniques used for these reductions are similar to those used in ordinary algebra. The procedure is:

(a) Multiply all variables necessary to remove parentheses.

(b) Look for identical terms. Only one of those terms be retained and all others dropped.
e.g. AB+AB+AB+AB=AB

(c) Look for a variable and its negation in same term. This term can be dropped.
e.g. A.B.B'=A.0=0 ABCC'=AB.0=0

(d) Look for pairs of terms that are identical except for one variable which may be missing in one of the terms. The larger term can be dropped.
e.g. ABC'D'+ABC'
=ABC' (D'+1)
= ABC'.1
=ABC'

(e) Look for pairs of terms which have the same variables, with one or more variables complemented. If a variable in one term of such pair is complemented while in the second term it is not, then such terms can be combined into a single term with that variable dropped.
e.g. ABC'D'+ABC'D
=ABC' (D'+D)
= ABC'.1
=ABC'

AB(C+D) +AB(C+D)'
=AB[(C+D)+(C+D)']
=AB.1
=AB

Q1. Simplify the Boolean expression:

A'B+A'B'+AB+AB'

Solution:

A'B+A'B'+AB+AB'

=A'(B+B')+A(B+B')

(By inverse law i.e. X+X'=1)

=A'.1+A.1

(By Identity law i.e. A.1=A)

=A'+A

(By inverse law i.e. X+X'=1)

=1

Q2. Simplify the Boolean expression:

A'B+A'B'+AB

Solution:

A'B+A'B'+AB

=A'(B+B')+AB

(By inverse law i.e. X+X'=1)

=A'.1+AB

(By Identity law i.e. A.1=A)

=A'+AB

A+AB

=A(1+B) (By identity law)

=A.1 (By identity law)

=A

A+AB (By absorption law)

=A

Do as directed:

Q1. Find the **dual** and **complement** of the following Boolean expression.

a) X+Y'.0
Solution:
Dual: X.Y'+1
Complement: X'.Y+1

b) XY+XY'+X'Y
Solution:
Dual: X+Y.X+Y'.X'+Y
Complement: X'+Y'.X'+Y.X+Y'

c) A+A'B+B'+1
Solution:
Dual: A.A'+B.0
Complement: A'.A+B'.0

d) (X+Y'+Z)(X+Y)
Solution:
Dual: (X.Y'.Z)+(X.Y)
Complement: (X'.Y.Z')+(X'+Y')

e) X'YZ'+X'Y'Z+1
Solution:
Dual: (X'+Y+Z').(X'+Y'+Z).0
Complement: (X+Y'+Z).(X+Y+Z').0

f) A'B+A'B'+1+0
Solution:
Dual: A'+B.A'+B'.0.1
Complement: A+B'.A+B.0.1

Q2. Prove the following identity by applying Boolean theorems.

a) A[B+C'(AB+AC')']=AB
Solution:
Applying Boolean theorems,
LHS= A [B+C'(AB+AC')']
= A[B+C'((AB)'.(AC')')]
= A[B+C'((A'+B').(A'+(C')')]
= A[B+C'((A'+B').(A'+C)]
= A[B+C'(A'.A'+A'C+A'.B'+B'C)]
= A[B+C'(A'+A'C+A'B'+B'C)]
= A[B+A'C'+A'CC'+A'B'C'+B'CC']
= A[B+A'C'+0+A'B'C'+0]
= AB+A.A'C'+A.A'B'C'
= AB+0+0
=AB=RHS
Since, LHS=RHS
Therefore, A[B+C'(AB+AC')']=AB is proved.

b) A+B[AC+(B+C')D]=A+BD
Solution:
Applying Boolean theorems,
LHS= A+B[AC+(B+C')D]
= A+B[AC+ BD+C'D]
= A+ABC+BBD+BC'D
= A+ABC+BD+BC'D
= A+ABC+BD(1+C')
= A+ABC+BD
=A(1+BC)+BD
= A.1+BD
= A+BD=RHS
Since, LHS=RHS
Therefore,
A+B[AC+(B+C')D]=A+BD is proved.

c) (A+(BC)')'(AB'+ABC)=0
Solution:
Applying Boolean theorems,
LHS=(A+(BC)')'(AB'+ABC)
= (A+(B'+C'))'(A(B'+BC))
= A'.(B'+C')'(AB')
=A'(B.C)AB'
=0=RHS

d) (B+BC)(B+B'C)(B+D)=B

e) AB+AB'C'+BC'=AC+BC'

Q3. Given that: **(AB)'+A'B=C**, then, find **(AC)'+A'C**.
Solution:
According to the question,
C=(AB)'+A'B
= A'+B'+A'B
= A'+A'B+B'
= A'+B'
Therefore,
(AC)'+A'C
=(A(A'+B'))'+A'(A'+B')
= (A.A'+A.B')'+A'.A'+A'.B'

= (0+A.B′)′+ A′+A′.B′
=(A.B′)′+ A′(1+B′)
=(A′+B)+A′.B′
=A′+A′B′+B
=A′(1+B′)+B
=A′B′+B

Q4. Simplify the following Boolean expression: **A′B′C′+A′BC′+AB′C′+ABC′**
Solution:
A′B′C′+A′BC′+AB′C′+ABC′
=A′C′ (B′+B) + AC′ (B′+B)
= A′C′.1+AC′.1
=A′C′+AC′
=C′(A′+A)
=C′.1
=C′

Q5. Given AB′+A′B=C, show that AC′+A′C=B
Solution:
According to the question,
C=AB′+A′B
C′=(AB′+A′B)′
=(AB′)′.(A′B)′
=(A′+B).(A+B′)
=A′.A+A′B′+AB+B.B′
=0+A′B′+AB+0
=A′B′+AB
LHS=AC′+A′C
= A(A′B′+AB)+A′(AB′+A′B)
=A.A′B′+A.AB+A′.AB′+A′.A′B
=0+AB+0+A′.B
=AB+A′B
=B(A+A′)
=B.1
=B=RHS Proved.

Q6. (a) If A+B=A+C and A′+B=A′+C, then B=C.
Solution:
Applying Boolean postulates,
Let A=B=C=0, then
A+B
=0+0 (By postulate no.5)
=0
A+C
=0+0 (By postulate no.5)
=0
Again,
A′+B
=0′+0 (By postulate no.9)
=1+0 (By postulate no.7)
=1
A′+C
=0′+0 (By postulate no.9)
=1+0 (By postulate no.7)
=1
Now, Let A=1,B=1 and C=0
A+B
=1+1 (By postulate no.8)
=1
A+C
=1+0 (By postulate no.7)
=1
Again,
A′+B
=1′+1 (By postulate no.10)
=0+1 (By postulate no.6)
=1
A′+C

=1’+0 (By postulate no.10)
=0+0 (By postulate no.5)
=0
Therefore, A+B=A+C and A’+B=A’+C, only when B=C.
(b) If A+B=A+C and A.B=A.C, then B=C.
Solution:
Applying Boolean postulates,
Let A=B=C=0, then
A+B
=0+0 (By postulate no.5)
=0
A+C
=0+0 (By postulate no.5)
=0
Again,
A.B
=0.0 (By postulate no.1)
=0
A.C
=0.0 (By postulate no.1)
=0
Now, Let A=1,B=1 and C=0
A+B
=1+1 (By postulate no.8)
=1
A+C
=1+0 (By postulate no.7)
=1
Again,
A.B
=1.1 (By postulate no.4)
=1
A.C
=1.0 (By postulate no.3)
=0
Therefore, A+B=A+C and A.B=A.C, only when B=C.

Q7. Prove that: **(use Boolean theorems)**

((AB)’+A’+AB)’=0

Number System

1. **Decimal(0,…,9)**
2. **Binary(0,1)**
3. **Octal(0,..,7)**
4. **Hexadecimal(0,.,9,A,.,F)**

Digit: Individual one

Number: Collective one

BINARY ARITHMETIC:

1. BINARY ADDITION
2. BINARY SUBTRACTION
3. BINARY MULTIPLICATION
4. BINARY DIVISION

1. BINARY ADDITION

If A & B be the two Boolean variable then,

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

1	1	10 i.e. 0 as SUM CARRY 1

1 CARRY

1	1	1	0
+0	1	0	1
1 0	0	1	1

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

IN ALU,

Adder circuit is designed on the basis of SUM (XOR gate) & CARRY (AND gate) output.

Problem:

$(1111011)_2 + (1010101)_2$

1111011	123
+1010101	85
(11010000)	208

Equivalent decimal:

$1x2^7 + 1x2^6 + 1x2^4$

$=128+64+16$

$= (208)_{10}$

2. **BINARY SUBTRACTION**

If A & B be the two Boolean variable then,

A	B	A-B
0	0	0
0	1	1 BORROW 1
1	0	1
1	1	0

$(1111101)_2 - (1110111)_2 = ?$

```
 1111101
-1110111
 0000110
```

SUBTRACTOR CIRCUIT is designed by the help of XOR gate (A'B+AB') and for borrows expression is A'B.

3. **BINARY MULTIPLICATION**

If A & B be the two Boolean variable then,

A	B	AXB
0	0	0

0 1 0
1 0 0
1 1 1

1111X1110

```
   1111
 X 1110
--------
    0000
   1111
  1111
 1111
--------
10010010
```

1
1
10
1
11
1
100
1
101
1
110
1
111

4. BINARY DIVISION

If A & B be the two Boolean variable then,

A	**B**	**A/B**
0	0	0
0	1	0
1	0	NOT DEFINED
1	1	1

1111/11=101
101011/11=1110

9'scomplement / 10's complement:

Digit: individual form
Number: collective form

Digit:
0,1,2,3,4,5,6,7,8,9

9's Complement of any digit =**9-digit**

e.g. 9's complement of 6=9-6=3

10's complement of any digit=9's complement +1

e.g. 10's complement of 6 =9's complement of 6+1=9-6+1=3+1=4

Number:
Digit:
0,1,2,3,4,5,6,7,8,9

9’s complement of any number=take 9 as per digit existence in the number-number

e.g. 9’s complement of 123 =999-123=876

10’s complement of any number=9’s complement of that number+1

e.g. 10’s complement of 123=9’s complement of 123+1=876+1=877

1’s complement / 2’s complement:
Digit: individual form
Number: collective form

Digit:
0, 1
1’s Complement of any digit =**1-digit**

e.g. 1’s complement of 0=1-0=1
1’s complement of 1=1-1=0

2’s complement of any digit=1’s complement of that digit +1

e.g. 2’s complement of 1 =1’s complement of 1+1=0+1=1

Number:
Digit:
0, 1

1’s complement of any number=take 1 as per digit existence in the number-number

e.g. 1’s complement of 101 =111-101=010

2’s complement of any number=1’s complement of that number+1

e.g. 2’s complement of 101=1’s complement of 101+1=010+1=011

Binary Addition using 1's complement method:

A+B

Augend: A

Addend: B

There are three different cases possible when we add two binary numbers which are as follows:

Case 1: Addition of the positive number with a negative number when the positive number has a greater magnitude.

Initially,

Step 1: Calculate the 1's complement of the given negative number.

Step 2: Sum up with the given positive number.

Step 3: If we get the end-around carry 1, it gets added to the LSB.

Example: 1101 and -1001

1. First, find the 1's complement of the negative number 1001. So, for finding 1's complement, change all 0 to 1 and all 1 to 0. The 1's complement of the number 1001 is **0110.**
2. Now, add both the numbers, i.e., 1101 and 0110; 1101+0110=1 0011

   1101

   \+ 0110

   ---

   1 0011

3. By adding both numbers, we get the end-around carry 1. We add this end around carry to the LSB of 0011. 0011+1=0100

   1 0011

   1

   0100

Problem: Perform addition between 101101 and -100111 using 1's complement method.

Solution:

1. First, find the 1's complement of the negative number 100111. So, for finding 1's complement, change all 0 to 1 and all 1 to 0. The 1's complement of the number 100111 is **011000.**
2. Now, add both the numbers, i.e., 101101 and 011000; 101101+011000=1 000101

   101101

   \+ 011000

   ---

   1 000101

3. By adding both numbers, we get the end-around carry 1. We add this end around carry to the LSB of 000101. 000101+1=000110

   1 000101

   1

   000110

Case 2: Adding a positive value with a negative value in case the negative number has a higher magnitude.

Initially, calculate the 1's complement of the negative value. Sum it with a positive number. In this case, we did not get the end-around carry. So, take the 1's complement of the result to get the final result.

Note: The resultant is a negative value.

Example: 1101 and -1110

1. First find the 1's complement of the negative number 1110. So, for finding 1's complement, we change all 0 to 1, and all 1 to 0. 1's complement of the number 1110 is 0001.
2. Now, add both the numbers, i.e., 1101 and 0001; 1101+0001= 1110
3. Now, find the 1's complement of the result 1110 that is the final result. So, the 1's complement of the result 1110 is 0001, and we add a negative sign before the number so that we can identify that it is a negative number.

Case 3: Addition of two negative numbers

In this case, first find the 1's complement of both the negative numbers, and then we add both these complement numbers. In this case, we always get the end-around carry, which get added to the LSB, and for getting the final result, we take the 1's complement of the result.

Note: The resultant is a negative value.

Example: -1101 and -1110 in five-bit register

1. Firstly find the 1's complement of the negative numbers 01101 and 01110. So, for finding 1's complement, we change all 0 to 1, and all 1 to 0. 1's complement of the number 01110 is 10001, and 01101 is 10010.
2. Now, we add both the complement numbers, i.e., 10001 and 10010; 10001+10010= 1 00011
3. By adding both numbers, we get the end-around carry 1. We add this end-around carry to the LSB of 00011.
   00011+1=00100
4. Now, find the 1's complement of the result 00100 that is the final answer. So, the 1's complement of the result 00100 is 110111, and add a negative sign before the number so that we can identify that it is a negative number.

Subtraction using 1's complement method:

A-B

Minuend: A

Subtrahend: B

Step 1: In the first step, find the 1's complement of the subtrahend.

Step 2: Next, add the complement number with the minuend.

Step 3: If got a carry, add the carry to its LSB. **Else** take 1's complement of the result which will be negative.

Note: The subtrahend value always get subtracted from minuend.

Example 1: 10101 - 00111

We take 1's complement of subtrahend 00111, which comes out 11000. Now, sum them. So,

10101+11000 =1 01101.

In the above result, we get the carry bit 1, so add this to the LSB of a given result, i.e., 01101+1=01110, which is the answer.

10101

- 00111

01110

Example 2: 10101 - 10111

We take 1's complement of subtrahend 10111, which comes out 01000. Now, add both of the numbers. So,

10101+01000 =11101.

In the above result, we didn't get the carry bit. So calculate the 1's complement of the result, i.e., 00010, which is the negative number and the final answer.

10111

- 10101

00010

Addition and Subtraction using 2's complement

Addition using 2's complement

There are three different cases possible when we add two binary numbers using 2's complement, which is as follows:

Case 1: Addition of the positive number with a negative number when the positive number has a greater magnitude.

Initially

Step 1: Find the 2's complement of the given negative number.

Step 2: Sum up with the given positive number.

Step 3: If we get the end-around carry 1 then the number will be a positive number and the carry bit will be discarded and remaining bits are the final result.

Example: 1101 and -1001

1. First, find the 2's complement of the negative number 1001. So, for finding 2's complement, change all 0 to 1 and all 1 to 0 or find the 1's complement of the number 1001.

The 1's complement of the number 1001 is 0110, and adds 1 to the LSB of the result 0110. So the 2's complement of number 1001 is 0110+1=0111

2. Add both the numbers, i.e., 1101 and 0111; 1101+0111=1 0100
3. By adding both numbers, we get the end-around carry 1. We discard the end-around carry. So, the addition of both numbers is 0100.

Problem: Perform addition between 101101 and -100111 using 2's complement method.

Solution:

1. First, find the 2's complement of the negative number 100111. So, for finding 2's complement, change all 0 to 1 and all 1 to 0 or find the 1's complement of the number 100111. The 1's complement of the number 100111 is 011000, and adds 1 to the LSB of the result 011000. So the 2's complement of number 100111 is 011000+1=011001
2. Add both the numbers, i.e., 101101 and 011001; 101101+011001=1 000110

By adding both numbers, we get the end-around carry 1. We discard the end-around carry. So, the addition of both numbers is 000110.

Case 2: Adding of the positive value with a negative value when the negative number has a higher magnitude.

Initially,

Step 1: Add a positive value with the 2's complement value of the negative number.

Step 2: Here, no end-around carry is found. So, we take the 2's complement of the result to get the final result.

Note: The resultant is a negative value.

Example: 1101 and -1110

1. First, find the 2's complement of the negative number 1110. So, for finding 2's complement, add 1 to the LSB of its 1's complement value 0001.
0001+1=0010
2. Add both the numbers, i.e., 1101 and 0010; 1101+0010= 1111
3. Find the 2's complement of the result 1110 that is the final result. So, the 2's complement of the result 1110 is 0001, and add a negative sign before the number so that we can identify that it is a negative number.

Problem: Perform addition between 1011 and -1111 using 2's complement method.

Solution:

1. First, find the 2's complement of the negative number 1111. So, for finding 2's complement, add 1 to the LSB of its 1's complement value 0001.
0000+1=0001

2. Add both the numbers, i.e., 1011 and 0001;
   1011+0001= 1100
3. Find the 2's complement of the result 1100 that is the final result. So, the 2's complements of the result 1100 is 0011+1=0100, and add a negative sign before the number so that we can identify that it is a negative number.

Case 3: Addition of two negative numbers

Step 1: First, find the 2's complement of both the negative numbers,

Step 2: And then we will add both these complement numbers.

Step 3: we will always get the end-around carry, which will be discarded, and to obtain the final result we will take the2's complement of the result.

Note: The resultant is a negative value.

Example: -1101 and -1110 in five-bit register

1. Firstly find the 2's complement of the negative numbers 01101 and 01110. So, for finding 2's complement, we add 1 to the LSB of the 1's complement of these numbers. 2's complement of the number 01110 is **10010**, and 01101 is **10011**.

   Description:

   2’s complement of 01110 is

   = 1’s complement of 01110+1

   =10001+1

   =10010

   2’s complement of 01101 is

   = 1’s complement of 01101+1

   =10010+1

   =10011

2. We add both the complement numbers, i.e., 10001 and 10010;
   10010+10011= **1 00101**
3. By adding both numbers, we get the end-around carry 1. This carry is discarded and the final result is the 2’s complement of the result **00101**. So, the 2's complement of the result 00101 is 11011, and we add a negative sign before the number so that we can identify that it is a negative number.

Problem: Perform addition between -1011 and -1111 using 2’s complement method.

Solution:

2’s complement of 01011=10101

2’s complement of 01111=10001

10101

+10001

1 00110

2’s complement of 00110=11001+1=11010

Subtraction using 2's complement

These are the following steps to subtract two binary numbers using 2's complement

- In the first step, find the 2's complement of the subtrahend.
- Add the complement number with the minuend.
- If we get the carry by adding both the numbers, then we discard this carry and the result is positive else take 2's complement of the result which will be negative.

Example 1: 10101 - 00111

We take 2's complement of subtrahend 00111, which is 11001. Now, sum them. So,

10101+11001 =1 01110.

In the above result, we get the carry bit 1. So we discard this carry bit and remaining is the final result and a positive number.

Example 2: 10101 - 10111

We take 2's complement of subtrahend 10111, which comes out 01001. Now, we add both of the numbers. So,

10101+01001 =11110.

In the above result, we didn't get the carry bit. So calculate the 2's complement of the result, i.e., 00010. It is the negative number and the final answer.

Simplification Tool:

Sum of Products and Products of Sums:

Logical functions are generally expressed in terms of logical variables. Values taken on by the logical functions and logical variables are in binary form. An arbitrary logic function can be expressed in the following forms:

i. Sum of Products (SOP)
ii. Product of Sum (POS)

Product term

The AND function (. sign) is referred as a *product*. The logical product of several variables on which function depends is considered to be a product term. *The variables in a product term can appear either in* ***complemented or uncomplemented form.***

e. g.

AB, ABC, A’B, A’B’, A’B’C’

Sum term

The OR function (+ sign) is referred as a *sum*. The logical sum of several variables on which function depends is considered to be a sum term. *The variables in a sum term can appear either in **complemented or uncomplemented form.***

e. g.

A+B, A+B+C, A’+B’+C’, A’+B

1. **SOP (Sum of Products)**
   The logical sum of two or more logical product terms is called a *Sum of Products* expression. It is basically an OR operation of AND operated variables such as:
   e. g.
   F=AB+AB’+A’B’
   F=ABC+A’B’C’+A’BC’+ABC

2. **POS (Product of Sums)**
   A product of Sums expression is a logical product of two or more logical sum terms. It is basically an AND operation of OR operated variables such as:
   e. g.

F=(A+B).(A+B’).(A’+B’)
F=(A+B+C).(A’+B’+C’).(A’+B+C’)

Minterm

A product term containing all the K variables of the function in either complemented or uncomplemented form is called a ***minterm***.

2-variable has four possible combinations:

Let A & B be the two literals then,

A	B	minterm
0	0	A’B’
0	1	A’B
1	0	AB’
1	1	AB

3-variable has eight possible combinations:

Let A, B & C be the three literals then,

A	B	C	minterm
0	0	0	A’B’C’
0	0	1	A’B’C
0	1	0	A’BC’
0	1	1	A’BC
1	0	0	AB’C’
1	0	1	AB’C
1	1	0	ABC’
1	1	1	ABC

4-variable has sixteen possible combinations:

Let A, B, C & D be the three literals then,

A	B	C	D	minterm
0	0	0	0	A’B’C’D’
0	0	0	1	A’B’C’D
0	0	1	0	A’B’CD’
0	0	1	1	A’B’CD
0	1	0	0	A’BC’D’
0	1	0	1	A’BC’D
0	1	1	0	A’BCD’
0	1	1	1	A’BCD
1	0	0	0	AB’C’D’
1	0	0	1	AB’C’D
1	0	1	0	AB’CD’
1	0	1	1	AB’CD
1	1	0	0	ABC’D’
1	1	0	1	ABC’D
1	1	1	0	ABCD’
1	1	1	1	ABCD

Maxterm

A sum term containing all the K variables of the function in either complemented or uncomplemented form is called a ***maxterm***.

2-variable has four possible combinations:

Let A & B be the two literals then,

A	B	minterm	maxterm
0	0	A’B’	A+B
0	1	A’B	A+B’
1	0	AB’	A’+B
1	1	AB	A’+B’

3-variable has eight possible combinations:

Let A, B & C be the three literals then,

A	B	C	minterm	maxterm
0	0	0	A’B’C’	A+B+C
0	0	1	A’B’C	A+B+C’
0	1	0	A’BC’	A+B’+C
0	1	1	A’BC	A+B’+C’
1	0	0	AB’C’	A’+B+C
1	0	1	AB’C	A’+B+C’
1	1	0	ABC’	A’+B’+C
1	1	1	ABC	A’+B’+C’

4-variable has sixteen possible combinations:

Let A, B, C & D be the three literals then,

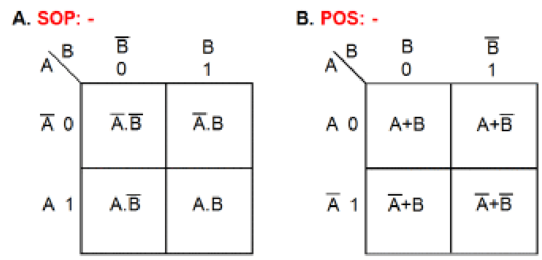
A	B	C	D	minterm	maxterm
0	0	0	0	A’B’C’D’	A+B+C+D
0	0	0	1	A’B’C’D	A+B+C+D’
0	0	1	0	A’B’CD’	A+B+C’+D
0	0	1	1	A’B’CD	A+B+C’+D’
0	1	0	0	A’BC’D’	A’+B+C’+D’
0	1	0	1	A’BC’D	A+B’+C+D’
0	1	1	0	A’BCD’	A+B’+C’+D

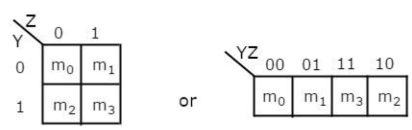
0	1	1	1	A’BCD	A+B’+C’+D’
1	0	0	0	AB’C’D’	A’+B+C+D
1	0	0	1	AB’C’D	A’+B+C+D’
1	0	1	0	AB’CD’	A’+B+C’+D
1	0	1	1	AB’CD	A’+B+C’+D’
1	1	0	0	ABC’D’	A’+B’+C+D
1	1	0	1	ABC’D	A’+B’+C+D’
1	1	1	0	ABCD’	A’+B’+C’+D
1	1	1	1	ABCD	A’+B’+C’+D’

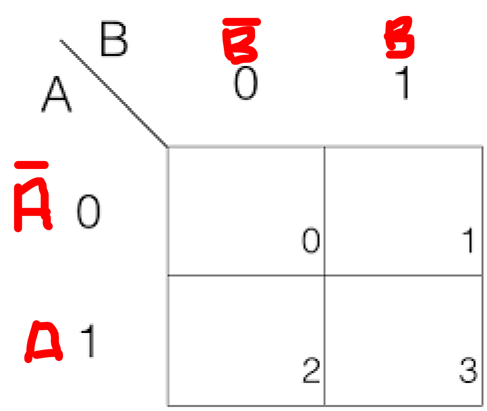
K-Map (Karnaugh Map):

As we know that, the simplification of the switching functions using Boolean Laws and theorems becomes complex with the increase in the number of variables and terms. The Karnaugh Map technique provides a systematic method for simplifying and manipulating switching expressions. In this technique, the information contained in a truth table or available in the POS or SOP form is represented on the Karnaugh Map (K-Map). The K-Map is actually a modified form of truth table. It is a simplification tool used to simplify the Boolean function. It is useful for simplifying the Boolean expression upto 4 variables. As like truth table, where number of input rows depends on the number of input variable, In K-map n variable generates 2^n number of cells.

2-variable map:

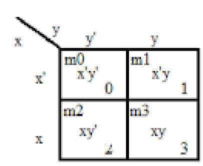






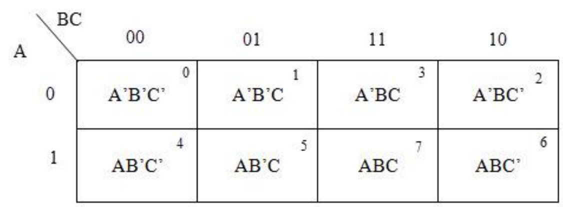
x	y	F
0	0	m0
0	1	m1
1	0	m2
1	1	m3

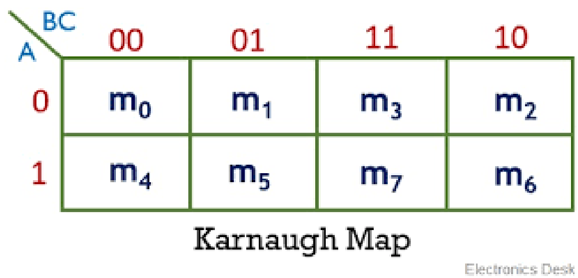
Truth Table of 2-variable



K-Map 2-Variable

3-variable map:

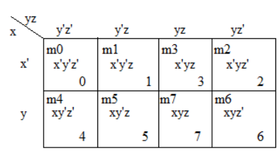




Karnaugh Map

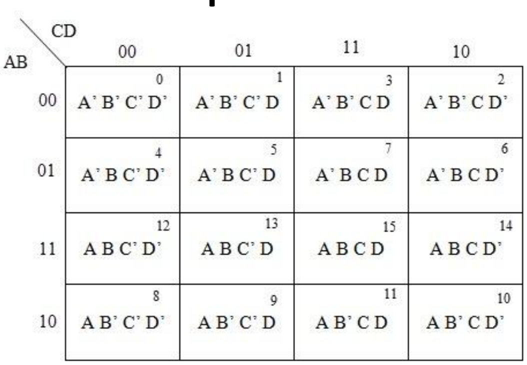
x	y	z	F
0	0	0	m0
0	0	1	m1
0	1	0	m2
0	1	1	m3
1	0	0	m4
1	0	1	m5
1	1	0	m6
1	1	1	m7

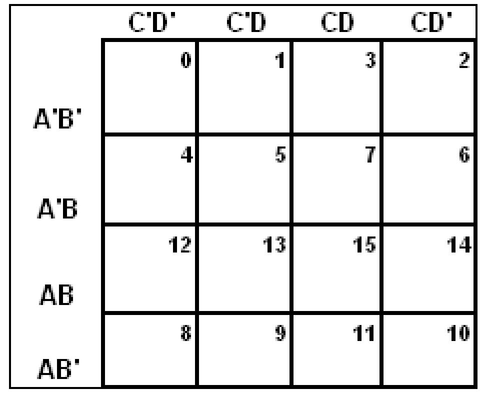
3-variable Truth table of F



3-Variable K-Map, minterm and cell position

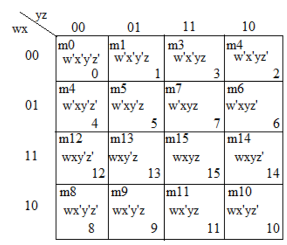
4-variable map:





w	x	y	z	F
0	0	0	0	m0
0	0	0	1	m1
0	0	1	0	m2
0	0	1	1	m3
0	1	0	0	m4
0	1	0	1	m5
0	1	1	0	m6
0	1	1	1	m7
1	0	0	0	m8
1	0	0	1	m9
1	0	1	0	m10
1	0	1	1	m11
1	1	0	0	m12
1	1	0	1	m13
1	1	1	0	m14
1	1	1	1	m15

Truth Table 4-variable function



K-Map 4-Variable

Arithmetic Circuit

1. **Combinational circuit**
2. **Sequential Circuit**

Combinational circuits are designed by using Logic Gates. The combinational circuits generate the **output** on the basis of **current input**. Whereas, Sequential circuits are designed by using Flip Flops. The Sequential Circuits generate the **output** on the basis of not only **current input** but also with **past output values**.

Combinational circuits

- **Adder**
- **Subtractor**
- **Decoder**
- **Encoder**
- **Multiplexer**
- **Demultiplexer**

Adder:

It is the combinational circuit used to perform arithmetic addition of **2** or **3** bit of binary number. It is available with ALU section of the CPU. There are two types of Adder circuits i.e. as follows.

1. Half Adder
2. Full Adder

A+B

A: Augend bit

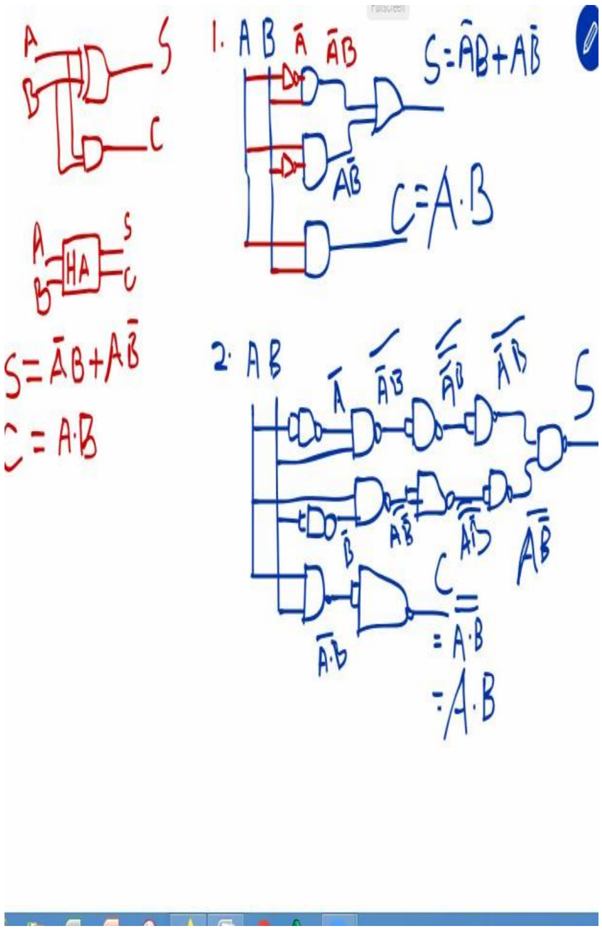
B: Addend bit

Half Adder:

It is the combinational circuit used to perform arithmetic addition of **2** bit of binary number. Here, two inputs are in the form of **Augend bit** (i.e. **A**) and **Addend bit** (i.e. **B**) and output is in the form of **Sum** (i.e. S) and **Carry** (i.e. C). The truth table for half adder circuit is as follows:

INPUT		OUTPUT	
A	**B**	**S**	**C**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

From the above truth table, the output of Sum is equivalent to the output of XOR gate and the output of Carry is equivalent to the output of AND gate. So, the logic circuit for Half Adder is designed by the help of **XOR gate** as well as **AND gate**.



1. A B
S = $\bar{A}B + A\bar{B}$
C = A·B
HA
S = $\bar{A}B + A\bar{B}$
C = A·B
2. A B
S