Logic gates :-

Truth tables :-

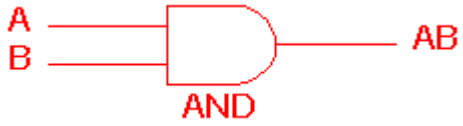
Many logic circuits have more than one input and one or more outputs. A truth table shows how the logic circuit's output responds to the various combinations of logic states at the inputs. The formal for two, three, and four input with one output truth tables are shown below :

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

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

D	C	B	A	X
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	?

1- AND gate



2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

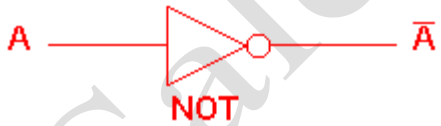
The AND gate is an electronic circuit that gives a **high** output (1) only if **all** its inputs are high. A dot (.) is used to show the AND operation i.e. A.B. Bear in mind that this dot is sometimes omitted i.e. AB

2- OR gate

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

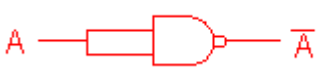
The OR gate is an electronic circuit that gives a high output (1) if **one or more** of its inputs are high. A plus (+) is used to show the OR operation.

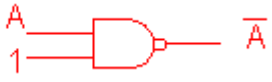
3- NOT gate



NOT gate	
A	$\overline{A}$
0	1
1	0

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an *inverter*. If the input variable is A, the inverted output is known as NOT A. This is also shown as A', or A with a bar over the top, as shown at the outputs. The diagrams below show two ways that the NAND logic gate can be configured to produce a NOT gate. It can also be done using NOR logic gates in the same way.





4- NAND gate

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if **any** of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

5- NOR gate

2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

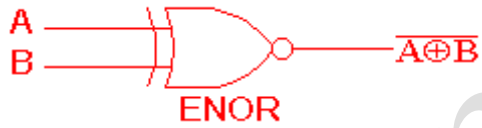
This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if **any** of the inputs are high. The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

6- EX-OR gate

2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

The '**Exclusive-OR**' gate is a circuit which will give a high output if **either, but not both**, of its two inputs are high. An encircled plus sign (⊕) is used to show the EOR operation.

7- EX-NOR gate



2 Input EXNOR gate		
A	B	$\overline{A⊕B}$
0	0	1
0	1	0
1	0	0
1	1	1

The '**Exclusive-NOR**' gate circuit does the opposite to the EOR gate. It will give a low output if **either, but not both**, of its two inputs are high. The symbol is an EXOR gate with a small circle on the output. The small circle represents inversion.

The NAND and NOR gates are called *universal functions* since with either one the AND and OR functions and NOT can be generated.

Note:

A function in *sum of products* form can be implemented using NAND gates by replacing all AND and OR gates by NAND gates.

A function in *product of sums* form can be implemented using NOR gates by replacing all AND and OR gates by NOR gates.

Table 1: Logic gate symbols

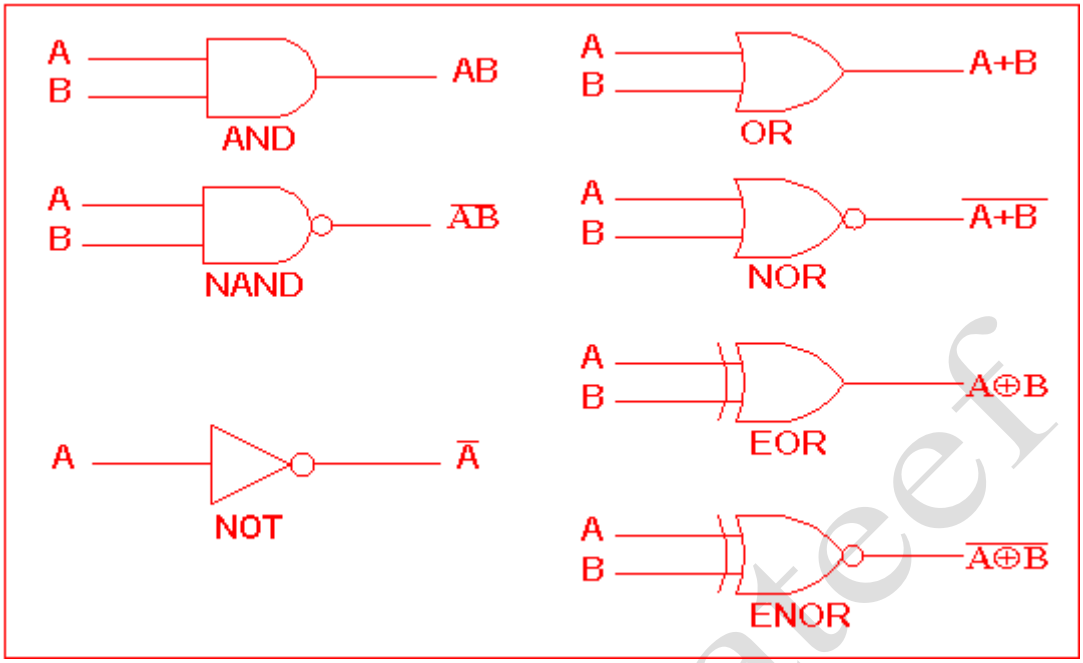


Table 2 is a summary truth table of the input/output combinations for the NOT gate together with all possible input/output combinations for the other gate functions. Also note that a truth table with 'n' inputs has 2^n rows. You can compare the outputs of different gates.

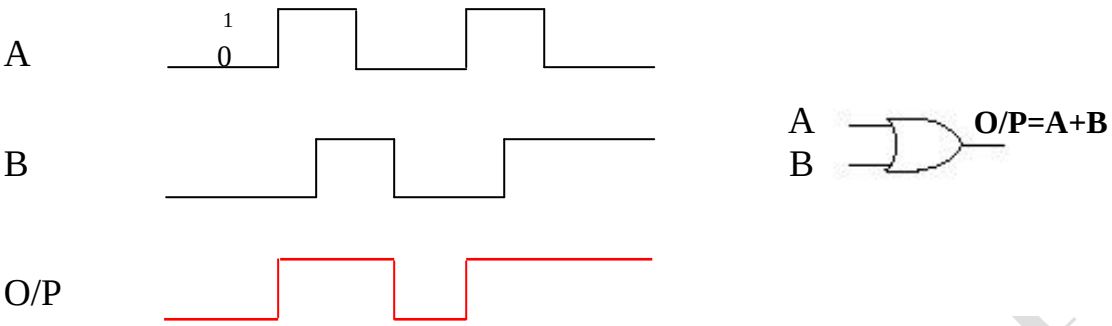
Table 2: Logic gates representation using the Truth table

NOT gate	
A	$\overline{A}$
0	1
1	0

INPUTS		OUTPUTS					
A	B	AND	NAND	OR	NOR	EXOR	EXNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

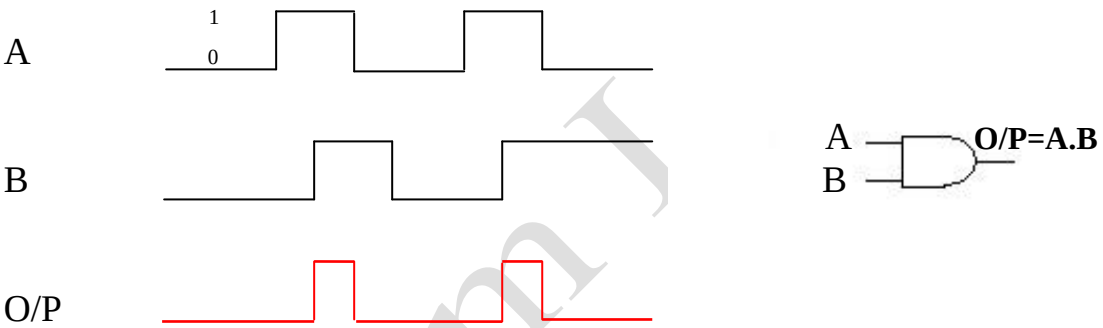
Example:

Determine the OR gate output in the fig. shown below. The OR gate inputs A and B are varying according to the timing diagrams shown.



Example:

Repeat the previous example for AND gate

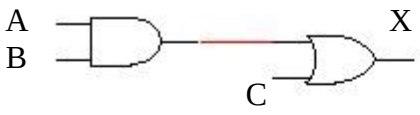


Describing logic circuits algebraically :-

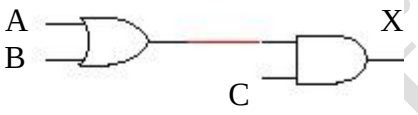
Any logic circuit, no matter how complex, may be completely described using the Boolean operations previously defined.

Example :-

Determine the output expression for the logic shown below :-



X=A.B+C



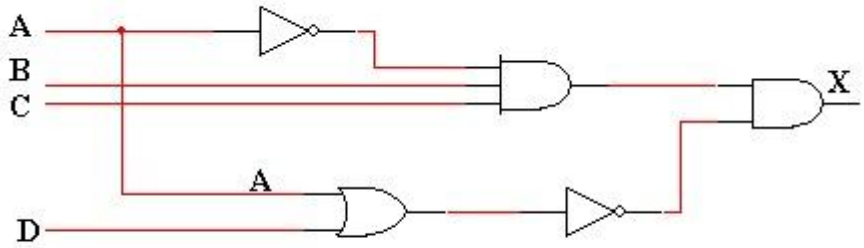
X=(A+B).C

$X = \bar{A} + B$

$X = \overline{A + B}$

H.W :-

a) determine the output expression for the following circuit



b) determine the output logic level if A=1, B=0 and C=1, D=1

Implementing circuits from Boolean expressions :-

If the operation of a circuit is defined by a Boolean expression, a logic-circuit can be implemented directly from that expression.

Example :

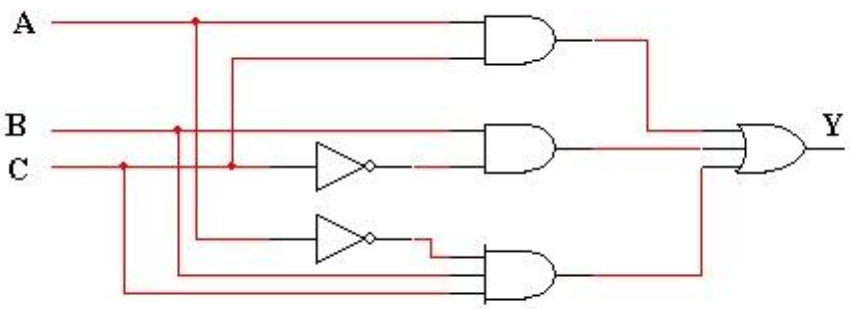
Implement the logic circuits defined by the following Boolean expressions :

a) $Y = AC + B\bar{C} + \bar{A}BC$

b) $X = AB + \bar{B}C$

Solution :-

a)



b)