Digital logic circuit:

In an electronic circuit that deal with signals that have only two states 0 and 1

The states 1 and 0 are also commonly referred to as:

On and Off

High and Low

True and False

+5V and 0V

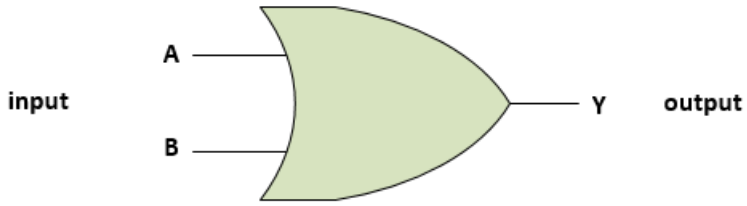
Logic gate:

Is an elementary building block of a digital circuit, it has one output and one or more inputs .

OR Gate:

Is an electronic circuit that gives a high output (1) if one or more of its input are high (1) .

Logic expression



Truth table

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

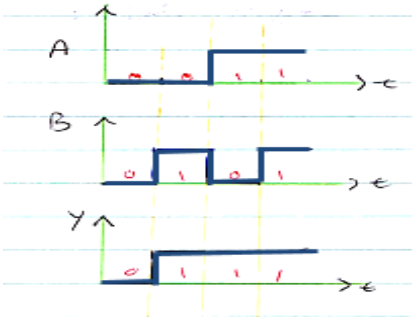
Boolean expression

Y = A + B

A plus (+) is used to show OR operation

Note: the truth table shows all input-output Possibilities for a logic circuits.

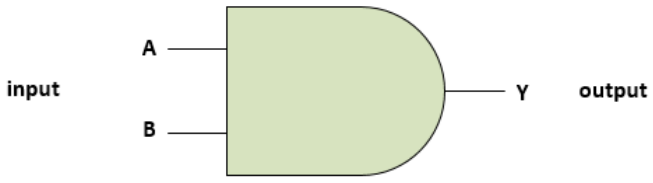
Time diagram



AND Gate:

Is an electronic circuit that gives a high output (1) only if all its inputs are high (1)

Logic expression



Truth table

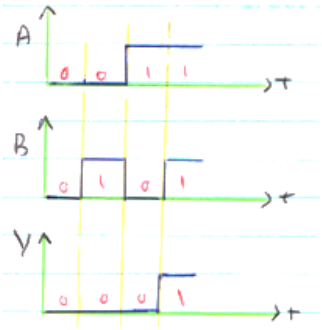
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Boolean expression

$Y = A \cdot B$

A plus (.) is used to show AND operation

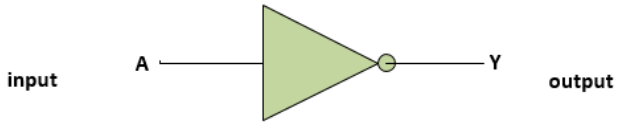
Time diagram



NOT Gate:

Is an electronic circuit that produces an inverted version of the input at its output. Also is known as an inverter

Logic expression



Truth table

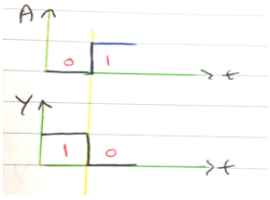
A	Y
0	1
1	0

Boolean expression

$Y = \bar{A}$

If the input is A, the inverted output is A with a bar over the top

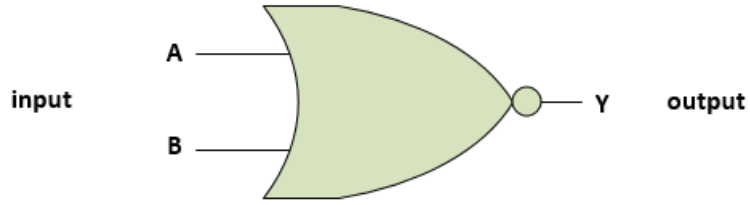
Time diagram



NOR Gate:

NOT – OR gate its equal to an OR gate followed by a NOT gate

Logic expression

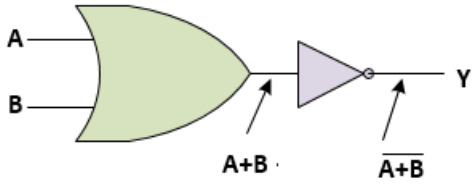


Truth table

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

Boolean expression

Y = $\overline{A+B}$

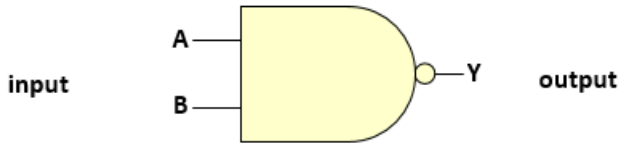


The output of NOR gate are low (0) if any of input are high (1)

NAND Gate:

NOT – AND gate its equal to an AND gate followed by a NOT gate

Logic expression

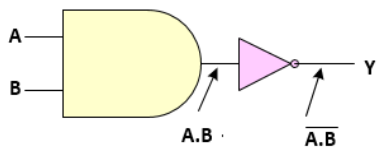


Truth table

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

Boolean expression

Y = $\overline{A.B}$



Note: the total number of possible combinations of binary inputs to a gate is determined by **N= 2^n**

N is number of input combination, **n** is number of input variable

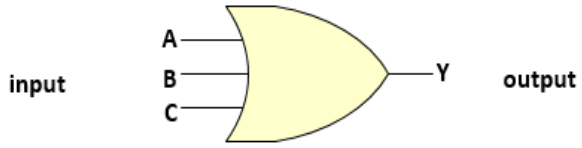
Two variable n=2 , N=2^2=4 combination

Three variable n=3 , N=2^3=8 combination

Four variable n=4 , N=2^4=16 combination

Three – input OR gate:

Logic expression



Boolean expression

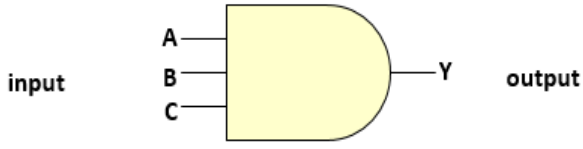
Y= A + B + C

Truth table

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

Three – input AND gate:

Logic expression



Boolean expression

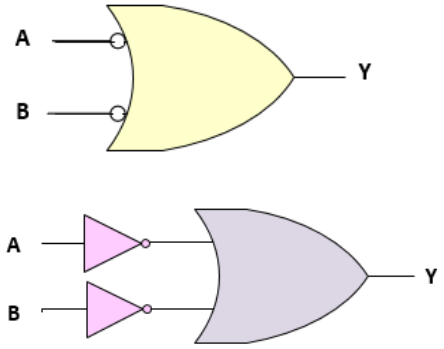
Y= A . B . C

Truth table

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

Negative-OR:

Logic expression



Truth table

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

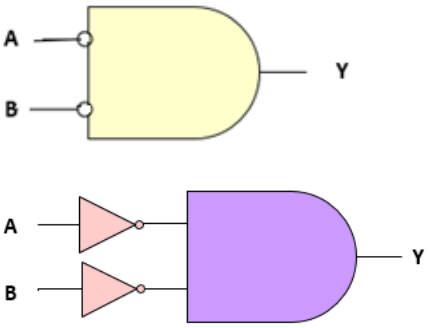
Boolean expression

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

Note: the negative – OR is equivalent operation of NAND Gate

Negative-AND :

Logic expression



Truth table

A	B	$\bar{A}$	$\bar{B}$	$Y = \bar{A}.\bar{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

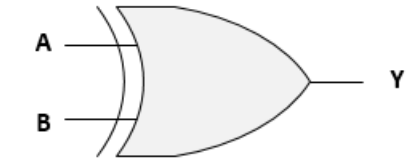
Boolean expression

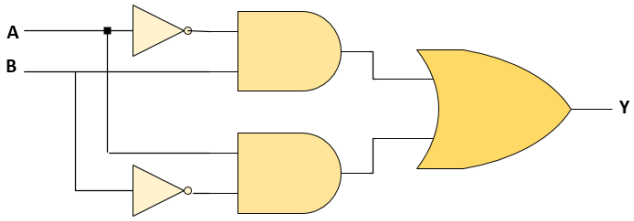
$Y = \bar{A}.\bar{B}$

Note: the negative – AND is equivalent operation of NOR Gate

Exclusive OR Gate (XOR):

Logic expression





Truth table

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

Boolean expression

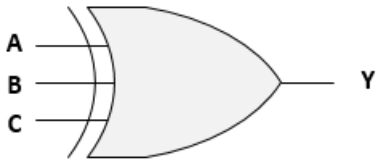
Y = $A \oplus B$

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

Note: like binary addition and we disregard carries the output is obtain.

Three – input XOR gate :

Logic expression



Truth table

A	B	C	Y
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

Boolean expression

Y= A ⊕ B ⊕ C

Note: the output of XOR can be obtained by :

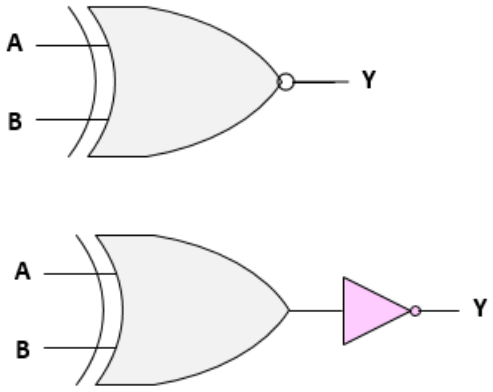
1- If we have an even number of ones the output =0, if we have an odd number of ones the output = 1.

2- The result can be obtained from a binary addition with disregard carries.

3- Taking XOR with the first two inputs and the result will be XOR with third one.

Exclusive NOR Gate (XNOR):

Logic expression



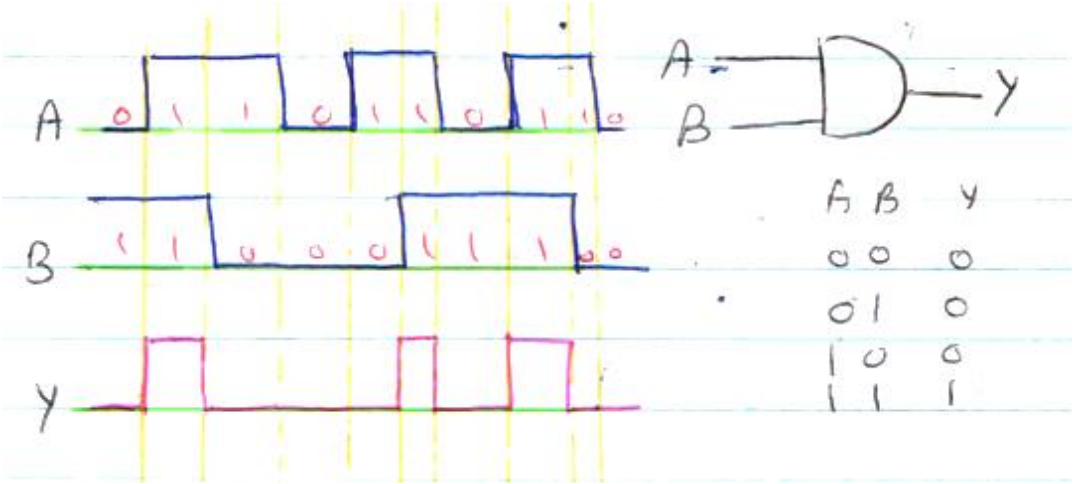
Truth table

A	B	A⊕B	Y=$\overline{A \oplus B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

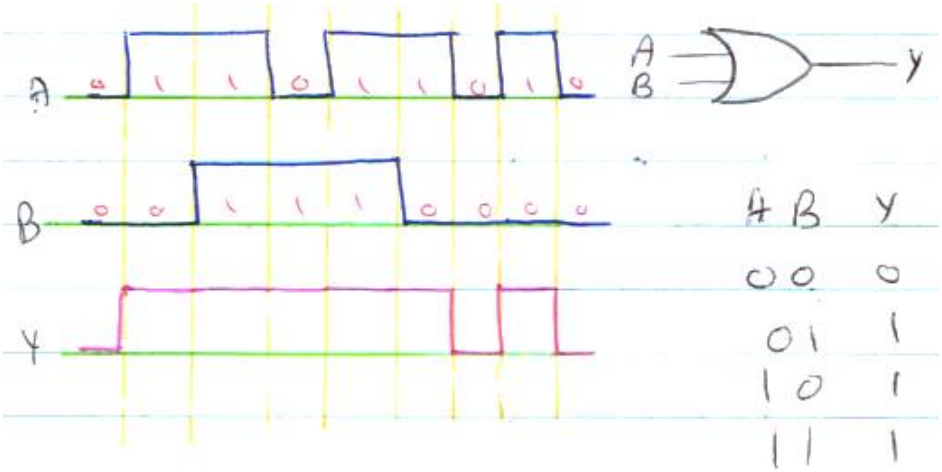
Boolean expression

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

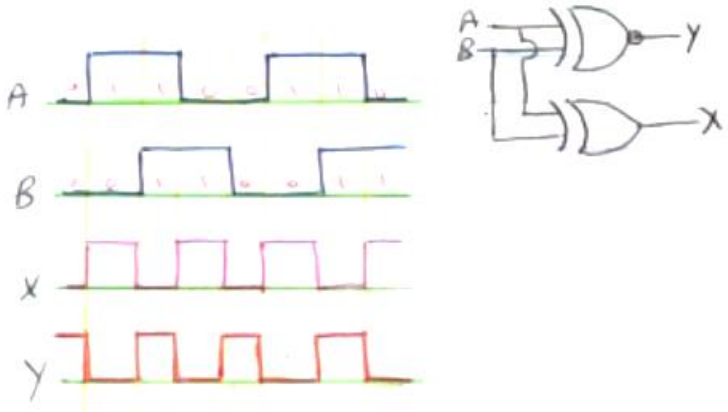
ex: show the output waveform for the 2-input AND gate bellow:



ex: show the output waveform for the 2-input OR gate bellow:



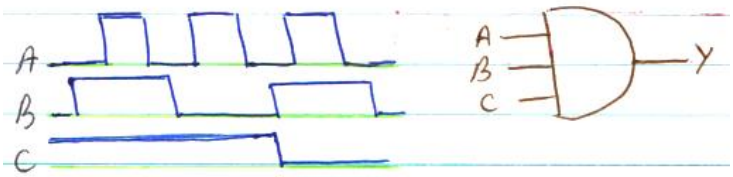
ex: show the output waveform for the XOR gate and XNOR gate



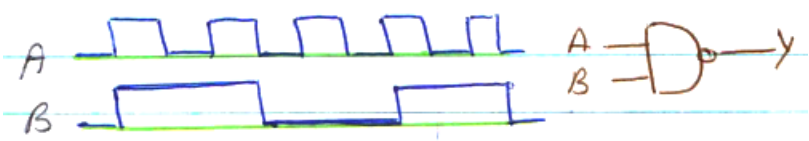
H.W:

Show the output waveform:

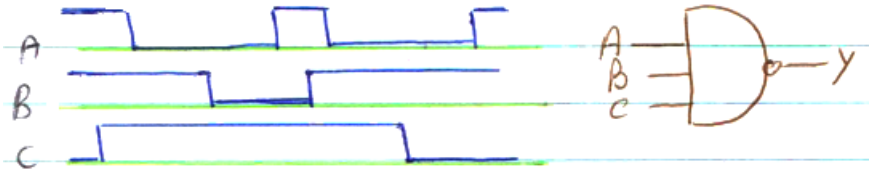
1-



2-



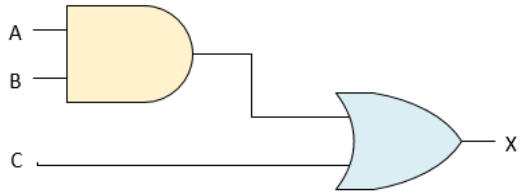
3-



Describing logic circuit output algebraically:

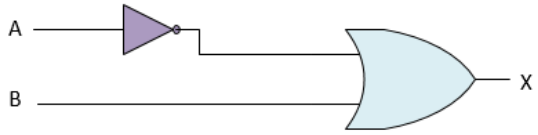
Any logic cct. no matter how complex may be completely described using Boolean expression previously defined because the OR , AND , NOT are the basic building block of a digital system.

ex: write the output Boolean expression of the cct. bellow:



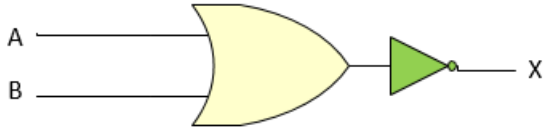
Sol: X= (A . B) + C

ex: write the output expression



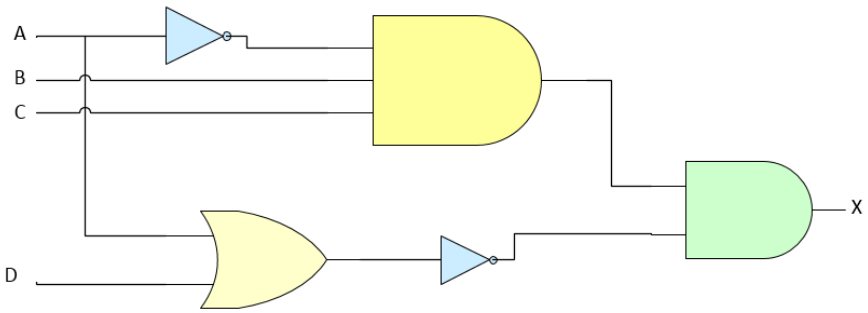
Sol: X=$\bar{A} + B$

ex: write the output expression



Sol: X= $\overline{A + B}$

ex: write the output expression and the truth table of the cct. Bellow:



Sol: X= $(\bar{A} \cdot B \cdot C) \cdot (\overline{A + D})$

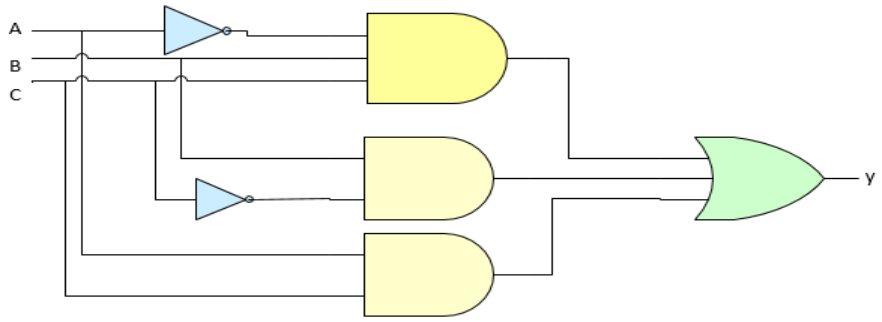
ABCD	$\bar{A}$	$\bar{A} \cdot B \cdot C$	$A + D$	$\overline{A + D}$	X= $(\bar{A} \cdot B \cdot C) \cdot (\overline{A + D})$
0000	1	0	0	1	0
0001	1	0	1	0	0
0010	1	0	0	1	0
0011	1	0	1	0	0
0100	1	0	0	1	0
0101	1	0	1	0	0
0110	1	1	0	1	1
0111	1	1	1	0	0
1000	0	0	1	0	0
1001	0	0	1	0	0
1010	0	0	1	0	0
1011	0	0	1	0	0
1100	0	0	1	0	0
1101	0	0	1	0	0
1110	0	0	1	0	0
1111	0	0	1	0	0

Implementing circuits from Boolean expression:

ex: construct a logic cct. Whose output :

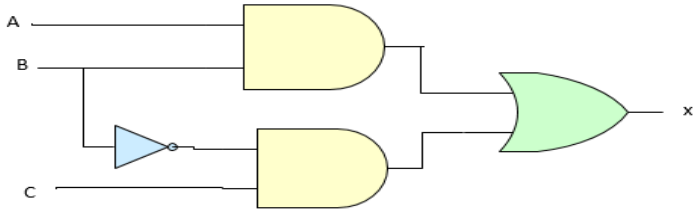
$$Y = A \cdot C + B \cdot \bar{C} + \bar{A} \cdot B \cdot C$$

Sol:



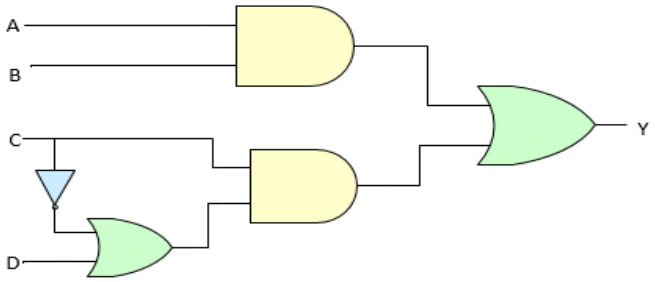
ex: $X = AB + \bar{B}C$ draw the cct. Diagram that implements this expression and write truth table.

Sol:



ABC	$\bar{B}$	$A \cdot B$	$\bar{B} \cdot C$	$X = A \cdot B + \bar{B} \cdot C$
000	1	0	0	0
001	1	0	1	1
010	0	0	0	0
011	0	0	0	0
100	1	0	0	0
101	1	0	1	1
110	0	1	0	1
111	0	1	0	1

ex: write the Boolean expression for the output of the logic cct. Shown, determine the output level A = B = D = 1 and C = 0



Sol: $Y = A \cdot B + C \cdot (\bar{C} + D)$

Output level y= 1 .1 + 0 . (1+1) = 1+ 0 = 1

Note: the following rules must always be followed when evaluating a Boolean expression

1- Perform all operation with parentheses
2- Perform an AND operation before an OR operation unless parentheses indicate otherwise
3- If an expression has a bar over it performs the operation of the expression first and then inverts the result.