UNIT- 2
BOOLEAN ALGEBRA & LOGIC GATES

TOPIC 2.1
THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

By -Prof. Sonali B. Jadhav

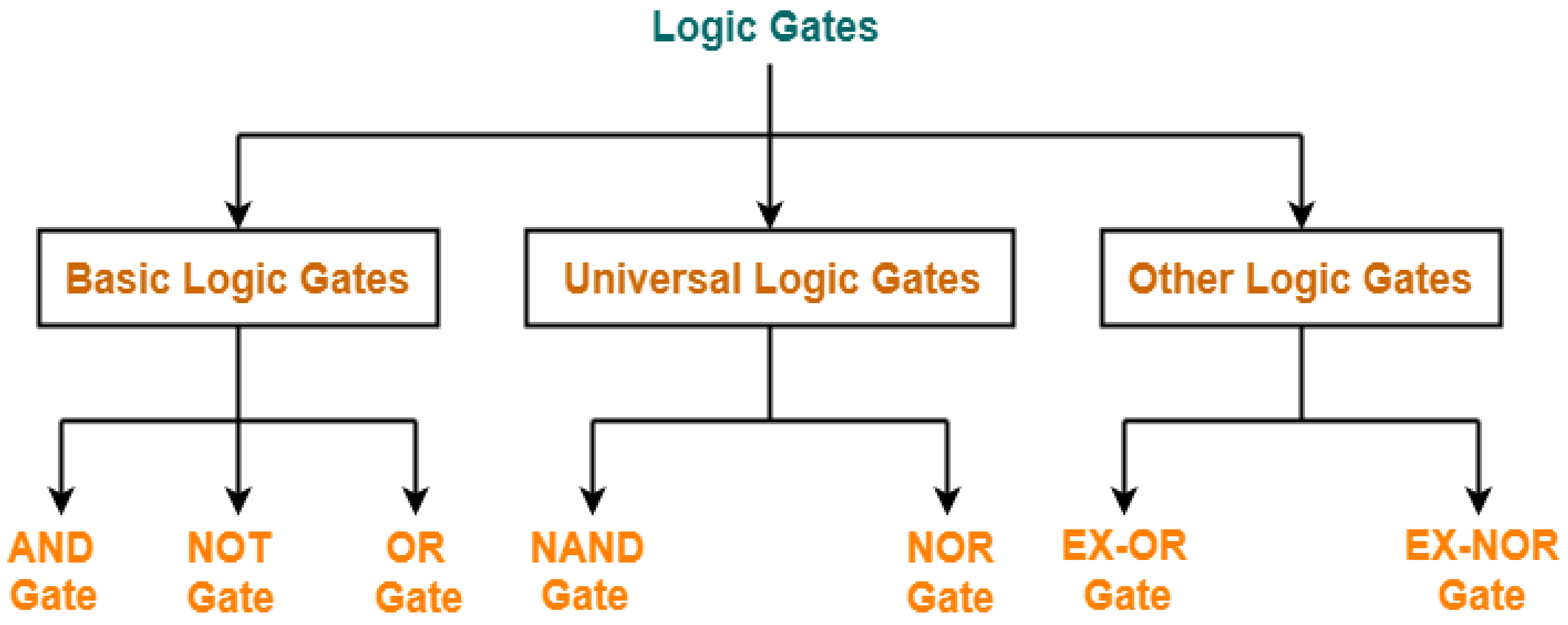
LOGIC GATES

- Logic gates perform basic logical functions.
- They are fundamental building blocks of digital integrated circuits.
- Most logic gates take an input of two binary values, and output a single value of a 1 or 0.
- Some circuits may have only a few logic gates, while others, such as microprocessors, may have millions of them.
- There are seven different types of logic gates, which are outlined.

LOGIC GATES

- Logic gates are electronic circuits that implement the basic functions of Boolean Algebra. There is a symbol for each gate.
- Logic levels (0 or 1) are represented by means of a voltage level.
  - High voltage (5V, 3.3V, 2.5V, etc.) is 1
  - Low voltage (0V) is 0
- The table used to represent the boolean expression of a logic gate function is commonly called a **Truth Table**. A logic gate truth table shows each possible input combination to the gate or circuit with the resultant output depending upon the combination of these input(s).
- There are "four" possible input combinations or 2^2 of "OFF" and "ON" for the two inputs.
- Then the four possible combinations of A and B for a 2-input logic gate is given as:
  - Input Combination 1. – "OFF" – "OFF" or (0, 0)
  - Input Combination 2. – "OFF" – "ON" or (0, 1)
  - Input Combination 3. – "ON" – "OFF" or (1, 0)
  - Input Combination 4. – "ON" – "ON" or (1, 1)
- Therefore, a 3-input logic circuit would have 8 possible input combinations or 2^3 and a 4-input logic circuit would have 16 or 2^4, and so on as the number of inputs increases. Then a logic circuit with "n" number of inputs would have 2^n possible input combinations of both "OFF" and "ON".

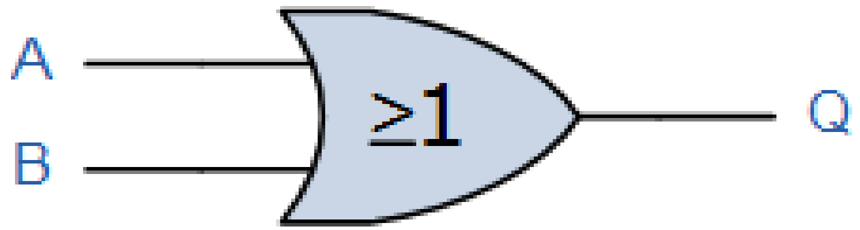
TYPES OF LOGIC GATES



Types of Logic Gates

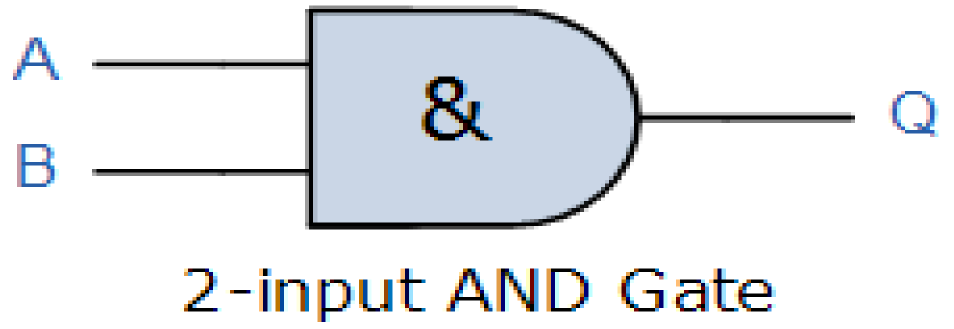
OR GATE

For a 2-input OR gate, the output Q is true if EITHER input A “OR” input B is true, giving the Boolean Expression
of: (Q = A or B).

Symbol	Truth Table		
 2-input OR Gate	A	B	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	1
Boolean Expression Q = A+B	Read as A OR B gives Q		

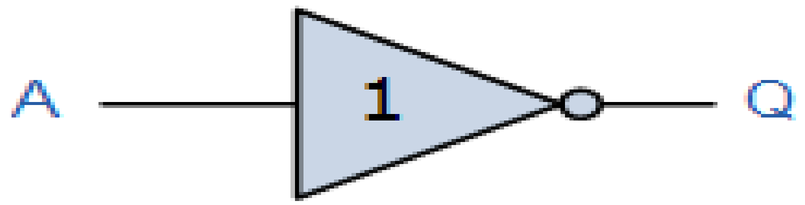
AND GATE

- For a 2-input AND gate, the output Q is true if BOTH input A "AND" input B are both true, giving the Boolean Expression of: (Q = A and B).

Symbol	Truth Table		
	A	B	Q
	0	0	0
	0	1	0
	1	0	0
	1	1	1
Boolean Expression Q = A.B	Read as A AND B gives Q		

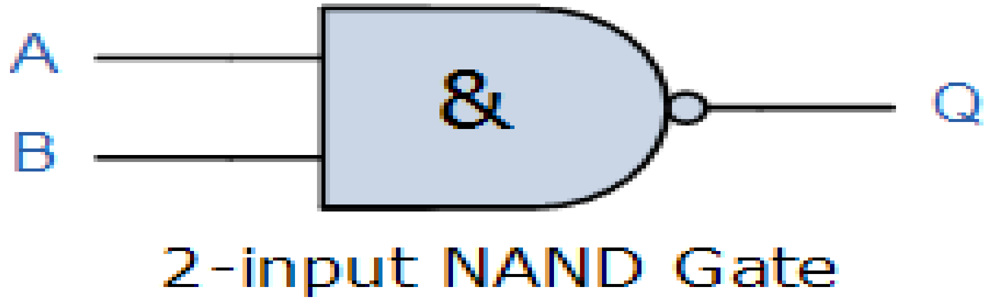
NOT GATE

- For a single input NOT gate, the output Q is ONLY true when the input is "NOT" true, the output is the inverse
or complement of the input giving the Boolean Expression of: (Q = NOTA).

Symbol	Truth Table	
 Inverter or NOT Gate	A	Q
	0	I
	I	0
Boolean Expression Q = NOT A or $\overline{A}$	Read as inversion of A gives Q	

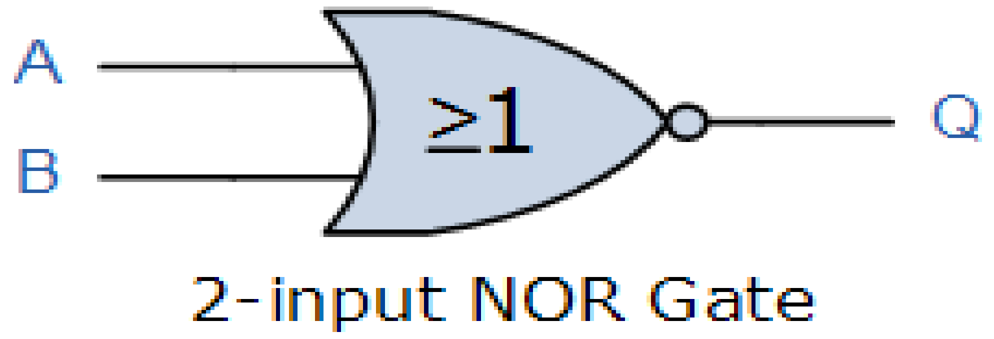
NAND GATE

- For a 2-input NAND gate, the output Q is NOT true if BOTH input A and input B are true, giving the Boolean
  Expression of: (Q = not(A AND B)).

<table>
<tr><th>Symbol</th><th colspan="3">Truth Table</th></tr>
<tr><td rowspan="5"> </td><td>A</td><td>B</td><td>Q</td></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
<tr><td>Boolean Expression $Q = \overline{A.B}$</td><td colspan="3">Read as A AND B gives NOT-Q</td></tr>
</table>

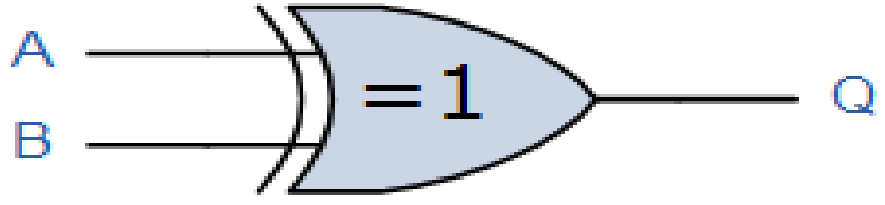
NOR GATE

- For a 2-input NOR gate, the output Q is true if BOTH input A and input B are NOT true, giving the Boolean Expression of: (Q = not(A OR B)).

Symbol	Truth Table		
	A	B	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	0
Boolean Expression $Q = \overline{A+B}$	Read as A OR B gives NOT-Q		

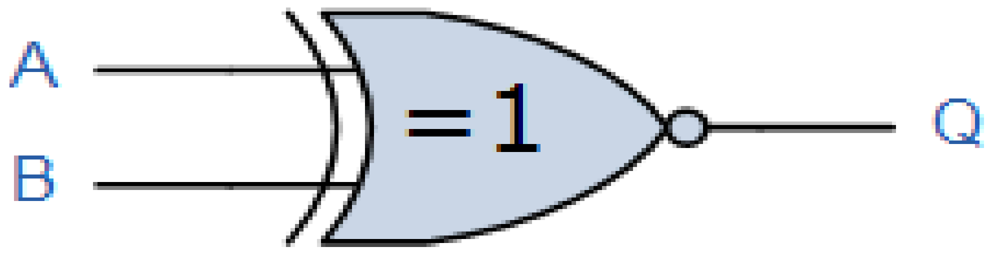
EX-OR GATE(Exclusive-or)

- Along with standard logic gates there are also two special types of logic gate function called an Exclusive-OR Gate and an Exclusive- NOR Gate.
- The Boolean expression to indicate an Exclusive-OR or Exclusive-NOR function is to a symbol with a plus sign inside a circle, ($\oplus$).
- For a 2-input Ex-OR gate, the output Q is true if EITHER input A or if input B is true, but NOT both giving the Boolean Expression of:
  (Q = (A and NOT B) or (NOT A and B)).

Symbol	Truth Table		
	A	B	Q
 2-input Ex-OR Gate	0	0	0
	0	1	1
	1	0	1
	1	1	0
Boolean Expression Q = A $\oplus$ B			

EX-NOR GATE

- For a 2-input Ex-NOR gate, the output Q is true if BOTH input A and input B are the same, either true or false,
  giving the Boolean Expression of: (Q = (A and B) or (NOT A and NOT B)).

Symbol	Truth Table		
 2-input Ex-NOR Gate	A	B	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	1
Boolean Expression $Q = \overline{A \oplus B}$			

SUMMARY OF 2-INPUT LOGIC GATES

The following Truth Table compares the logical functions of the 2-input logic gates discussed.

Inputs		Truth Table Outputs For Each Gate					
A	B	AND	NAND	OR	NOR	EX-OR	EX-NOR
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

The following table gives a list of the common logic functions and their equivalent Boolean notation.

Logic Function	Boolean Notation
AND	$A.B$
OR	$A+B$
NOT	A
NAND	$\overline{A\,.B}$
NOR	$\overline{A+B}$
EX-OR	$A \oplus B$
EX-NOR	$\overline{A \oplus B}$

- **Boolean theorems** and **laws** are used to simplify the various logical expressions.
- In a digital designing problem, a unique logical expression is evolved from the truth table.
- If this logical expression is simplified the designing becomes easier.
- The Boolean algebra is mainly used in digital electronics, set theory and digital electronics.
- It is also used in all modern programming languages.
- There are few basic laws and theorems of Boolean algebra, some of which are familiar to everyone such as
  - Commutative Law
  - Associative Law
  - Distributive law
  - DeMorgan's Theorems
  - Double Inversion law and
  - Duality Theorems.

BOOLEAN THEOREMS

- **The Commutative Law**
- The below two equations are based on the fact that the output of an OR or AND gate remains unaffected while the inputs are exchanged themselves.
- The equation of the commutative law is given below.

$$A + B = B + A \dots\dots\dots (1) \quad \text{and}$$

$$AB = BA \quad \dots\dots\dots (2)$$

- **The Associative Law**
- The equation is given as
- These laws illustrate that the order of combining input variables has no effect on the final answer.

$$A + (B + C) = (A + B) + C \quad \dots\dots\dots (3) \quad \text{and}$$

$$A(BC) = (AB)C \quad \dots\dots\dots (4)$$

BOOLEAN THEOREMS

- **The Distributive Law**
- The equation is given below.

A *(B + C) = AB + AC……….(5)

- **Double Inversion Law**
- The equation is given as
- The law states that the double complement (complement of the complement) of a variable equals the variable itself.

$$\overline{\overline{A}} = A$$

BOOLEAN THEOREMS

- **The Identity Law (OR)**
- basic identities of OR operations:

$$A + 0 = A$$

$$A + 1 = 1$$

$$A + \overline{A} = 1$$

$$A + A = A$$

- The authentication of the above all equations can be checked by substituting the value of A = 0 or A = 1.

- **The Identity Law (AND)**
- basic identities of AND operations:

$$A . 1 = A$$

$$A . A = A$$

$$A . 0 = 0$$

- One can check the validity of the above identities by substituting the value of A=0 or A = 1.

BOOLEAN THEOREMS

- **The DeMorgan's theorem**
- The equations are given below.
- The equation (6) says that a NOR gate is equivalent to a bubbledAND gate,and the equation (7) says that NAND gate is equivalent to a bubbled OR gate.

$$\overline{A + B} = \overline{A} \,.\, \overline{B} \dots\dots\dots (6) \text{ and}$$

$$\overline{A \,.\, B} = \overline{A} + \overline{B} \dots\dots\dots (7)$$

- **DualityTheorems**
- The new Boolean relation can be derived with the help of Duality theorem.According to this theorem for the given Boolean relation, the new Boolean relation can be derived by the following steps.
  - Changing each OR sign to an AND sign.
  - Changing eachAND sign to an OR sign.
  - Complementing each 0 or 1 appearing in the given Boolean identity.

For example:

The distributive law states that $A * (B+C) = AB + AC \dots\dots (8)$

Now,by using the duality theorem,we can get the new relation by interchanging each OR andAND sign.The equation (8) becomes.

$$A + BC = (A + B).(A + C) \dots\dots\dots\dots (9)$$

The equation (9) is a new Boolean relation.Similarly,for any other Boolean relation,its dual relation can also be derived.

DeMorgan’s Theorem

- **DeMorgan’s Theorem** is mainly used to solve the various Boolean algebra expressions.
- The Demorgan’s theorem defines the uniformity between the gate with same inverted input and output.
- It is used for implementing the basic gate operation likes NAND gate and NOR gate.
- The Demorgan’s theorem mostly used in digital programming and for making digital circuit diagrams.
- **There are two DeMorgan’s Theorems.**
  - **DeMorgan’s First Theorem**
  - **DeMorgan’s Second Theorem**
- DeMorgan’s first theorem states that two (or more) variables NOR´ed together is the same as the two variables inverted (Complement) and AND´ed, while the second theorem states that two (or more) variables NAND´ed together is the same as the two terms inverted (Complement) and OR´ed.
- That is replace all the OR operators with AND operators, or all the AND operators with an OR operators.

DeMorgan's First Theorem

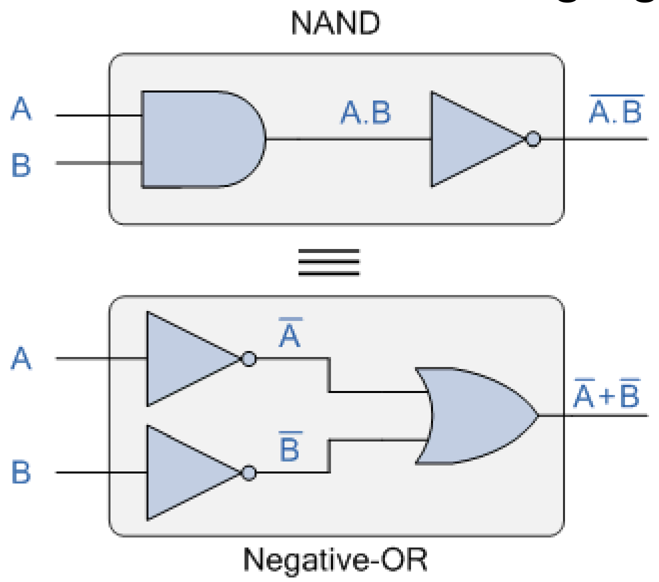
- DeMorgan's First theorem proves that when two (or more) input variables are AND'ed and negated, they are equivalent to the OR of the complements of the individual variables. Thus the equivalent of the NAND function and is a negative-OR function proving that $\overline{A.B} = \overline{A}+\overline{B}$ and we can show this using the following table.

Verifying DeMorgan's First Theorem using Truth Table

Inputs		Truth Table Outputs For Each Term					
B	A	A.B	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$	
0	0	0	**1**	1	1	**1**	
0	1	0	**1**	0	1	**1**	
1	0	0	**1**	1	0	**1**	
1	1	1	**0**	0	0	**0**	

DeMorgan's First Law Implementation using Logic Gates

We can also show that $\overline{A.B} = \overline{A}+\overline{B}$ using logic gates as shown.



DeMorgan's Second Theorem

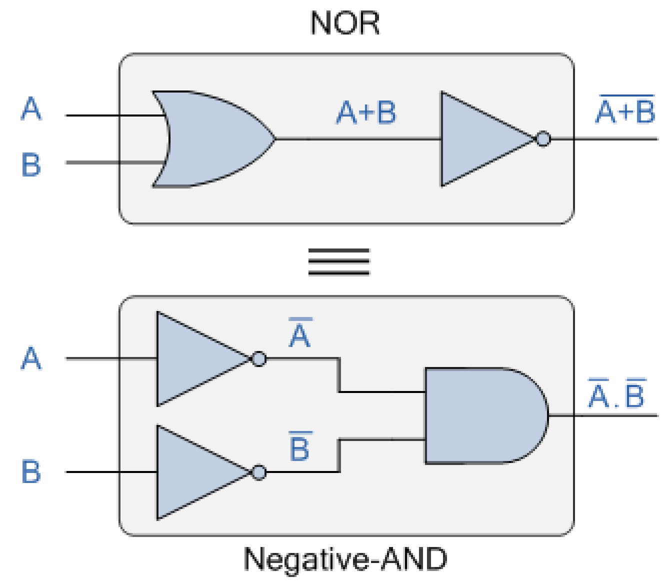
- DeMorgan's Second theorem proves that when two (or more) input variables are OR'ed and negated, they are equivalent to the AND of the complements of the individual variables. Thus the equivalent of the NOR function and is a negative-AND function proving that $\overline{A+B} = \overline{A}.\overline{B}$ and again we can show this using the following truth table.

Verifying DeMorgan's Second Theorem using Truth Table

Inputs		Truth Table Outputs For Each Term				
B	A	A+B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}\,.\,\overline{B}$
0	0	0	**1**	1	1	**1**
0	1	1	**0**	0	1	**0**
1	0	1	**0**	1	0	**0**
1	1	1	**0**	0	0	**0**

DeMorgan's Second Law Implementation using Logic Gates

We can also show that $\overline{A+B} = \overline{A}\,.\,\overline{B}$ using logic gates as shown.



DeMorgan’s Theorem

- Although we have used DeMorgan’s theorems with only two input variables A and B, they are equally valid for use with three, four or more input variable expressions,
- for example:

- For a 3-variable input
  - $\overline{A.B.C} = \overline{A} + \overline{B} + \overline{C}$
- and also
  - $\overline{A+B+C} = \overline{A} \, . \, \overline{B} \, . \, \overline{C}$

- For a 4-variable input
  - $\overline{A.B.C.D} = \overline{A} + \overline{B} + \overline{C} + \overline{D}$
- and also
  - $\overline{A+B+C+D} = \overline{A} \, . \, \overline{B} \, . \, \overline{C} \, . \, \overline{D}$

And so on.

PERFECT INDUCTION

- **Perfect induction, Proof by exhaustion,** also known as **proof by cases, or the brute force method**.
- It is a method of mathematical proof in which the statement to be proved is split into a **finite number of cases** and **each case is checked** to see if the proposition in question holds.
- Perfect induction says that if you check the veracity of a theorem for all possible input combinations, then the theorem is true in its entirety.
- **This is, if it is fulfilled in each case, it is fulfilled in general.**
- This path can be used in Boolean Algebra since the variables have only two possible values: 0 and 1, whilst in our algebra each variable can have infinite values.

For example, to demonstrate the distributed property of the sum against the product (which is not fulfilled in common algebra). $X+(Y \cdot Z) = (X+Y) \cdot (X+Z)$

X	Y	Z	X+(Y·Z)	(X+Y)·(X+Z)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

In the table, all possible values for X, Y and Z, the expressions $X+(Y \cdot Z)$ and $(X+Y) \cdot (X+Z)$ are identical, and thus, by perfect induction, both expressions are equivalent.

TEN BASIC RULES OF BOOLEAN ALGEBRA

Sr. No.	Rule	Equation
1	Anything ANDed with a 0 is equal to 0.	$A * 0 = 0$
2	Anything ANDed with a 1 is equal to itself.	$A * 1 = A$
3	Anything ORed with a 0 is equal to itself.	$A + 0 = A$
4	Anything ORed with a 1 is equal to 1.	$A + 1 = 1$
5	Anything ANDed with itself is equal to itself.	$A * A = A$
6	Anything ORed with itself is equal to itself.	$A + A = A$
7	Anything ANDed with its own complement equals 0.	$A \cdot \overline{A} = 0$
8	Anything ORed with its own complement equals 1.	$A + \overline{A} = 1$
9	Anything complemented twice is equal to the original.	$\overline{\overline{A}} = A$
10	The two variable rule.	$A + \overline{A}B = A + B$

Some Important law-

1. $A + BC = (A+B)(A+C)$

2. $\overline{A} + AB = \overline{A} + B$

3. $\overline{A} + A\overline{B} = \overline{A} + \overline{B}$

4. $A + AB = A$

TEN BASIC RULES OF BOOLEAN ALGEBRA

Boolean Expression	Boolean Algebra Law or Rule
A + 1 = 1	Annulment
A + 0 = A	Identity
A . 1 =A	Identity
A . 0 = 0	Annulment
A + A =A	Idempotent
A .A = A	Idempotent
NOT A' =A	Double Negation

Boolean Expression	Boolean Algebra Law or Rule
A + A' = 1	Complement
A .A' = 0	Complement
A+B = B+A	Commutative
A.B = B.A	Commutative
(A+B)' = A'.B'	de Morgan's Theorem
(A.B)' = A'+B'	de Morgan's Theorem

REDUCTION OF LOGIC EXPRESSION USING BOOLEAN ALGEBRA

- Complex combinational logic circuits must be reduced without changing the function of the circuit.
- Reduction of a logic circuit means the same logic function with fewer gates and/or inputs.
- The first step to reducing a logic circuit is to write the Boolean Equation for the logic function.
- The next step is to apply as many rules and laws as possible in order to decrease the number of terms and
  variables in the expression.
- To apply the rules of Boolean Algebra it is often helpful to first remove any parentheses or brackets.
- After removal of the parentheses, common terms or factors may be removed leaving terms that can be reduced
  by the rules of Boolean Algebra.
- The final step is to draw the logic diagram for the reduced Boolean Expression.

SIMPLIFICATION: EXAMPLE 1

- Using the above laws, simplify the following expression: (A + B)(A + C)

Q =	(A + B).(A + C)	
	A.A + A.C + A.B + B.C	– Distributive law
	A + A.C + A.B + B.C	– Idempotent AND law (A.A = A)
	A(1 + C) + A.B + B.C	– Distributive law
	A.1 + A.B + B.C	– Identity OR law (1 + C = 1)
	A(1 + B) + B.C	– Distributive law
	A.1 + B.C	– Identity OR law (1 + B = 1)
Q =	A + (B.C)	– Identity AND law (A.1 = A)

Then the expression: (A + B)(A + C) can be simplified to A + (B.C) as in the Distributive law.

SIMPLIFICATION: EXAMPLE 2

- Previous Example can also be solved in following manner:

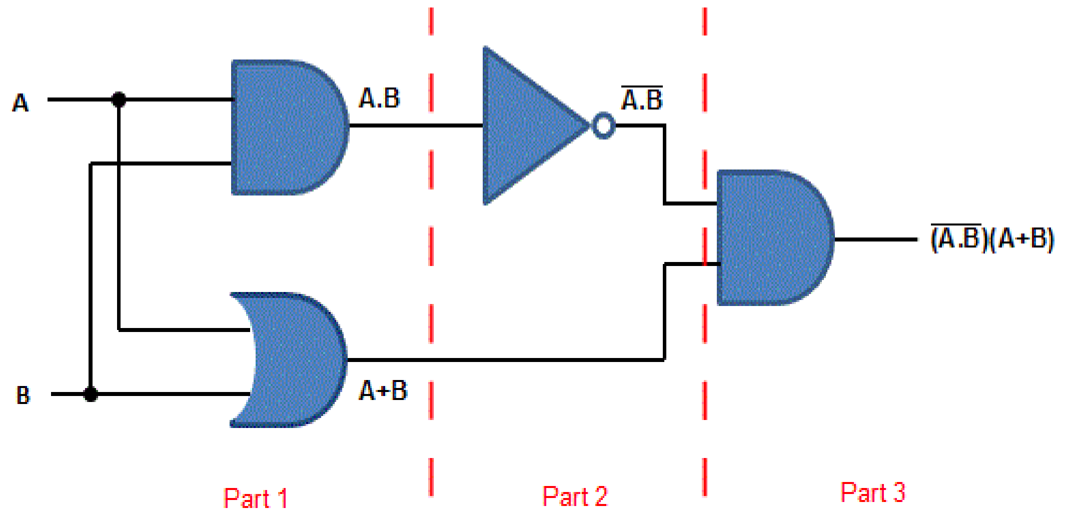
$$
\begin{aligned}
(A+B)(A+C) &= AA + AC + AB + BC \\
&= A + AC + AB + BC \\
&= A(1 + C + B) + BC \\
&= A \cdot 1 + BC \\
&= A + BC
\end{aligned}
$$

SOME EXAMPLES OF SIMPLIFICATION

$(\bar{A} + B)(A + B) =$ $B(\bar{A} + A)$ $B(1)$ B	$A\bar{C} + AB\bar{C} =$ $A\bar{C}(1 + B)$ $A\bar{C}(1)$ $A\bar{C}$	$A\bar{B}D + A\bar{B}\bar{D} =$ $A\bar{B}(D + \bar{D})$ $A\bar{B}(1)$ $A\bar{B}$
$ABC + \bar{A}B + AB\bar{C}$ $B(AC + \bar{A} + A\bar{C})$ $B(A[C + \bar{C}] + \bar{A})$ $B(A \cdot 1 + \bar{A})$ $B(A + \bar{A})$ $B \cdot 1$ B	$\bar{A}BC + AC$ $C(\bar{A}B + A)$ $C(A + B)$	$\overline{(A + B)}\,(\bar{A} + \bar{B})$ $\bar{A}\bar{B}(\bar{A} + \bar{B})$ $\bar{A}\bar{B} + \bar{A}\bar{B}$ $\bar{A}\bar{B}$

DERIVING BOOLEAN EXPRESSION FROM GIVEN CIRCUIT

- Sometimes, we need to be able to derive the Boolean expression of a logic gate diagram. We might want to do this, for example, so that we can investigate simplifying the design. We will look at some examples of this now.
- **Example 1**
  Consider the following circuit gate diagram:



Next, you need to start on the left and work your way through each part, working out what the output is after each gate.

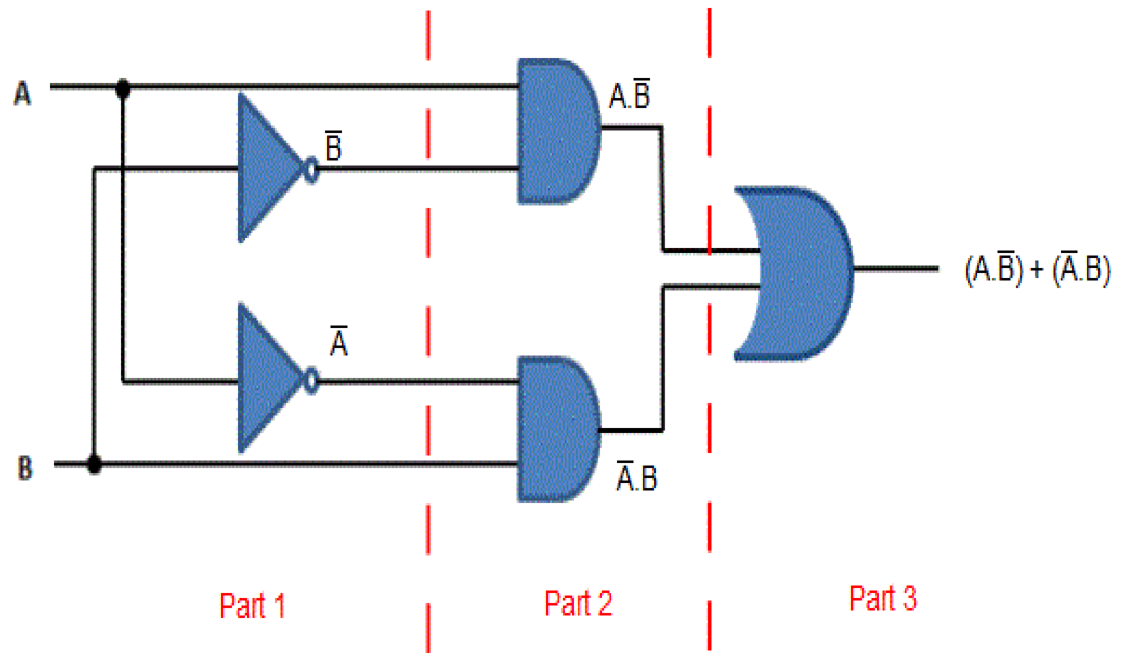
After part 1, we have A.B and A + B
After part 2, we have inverted A.B so we now have A.B After part 3, we have put the parts together in an AND gate from part 2 and have

(A.B)(A + B)
Let's draw out the truth table for this diagram:

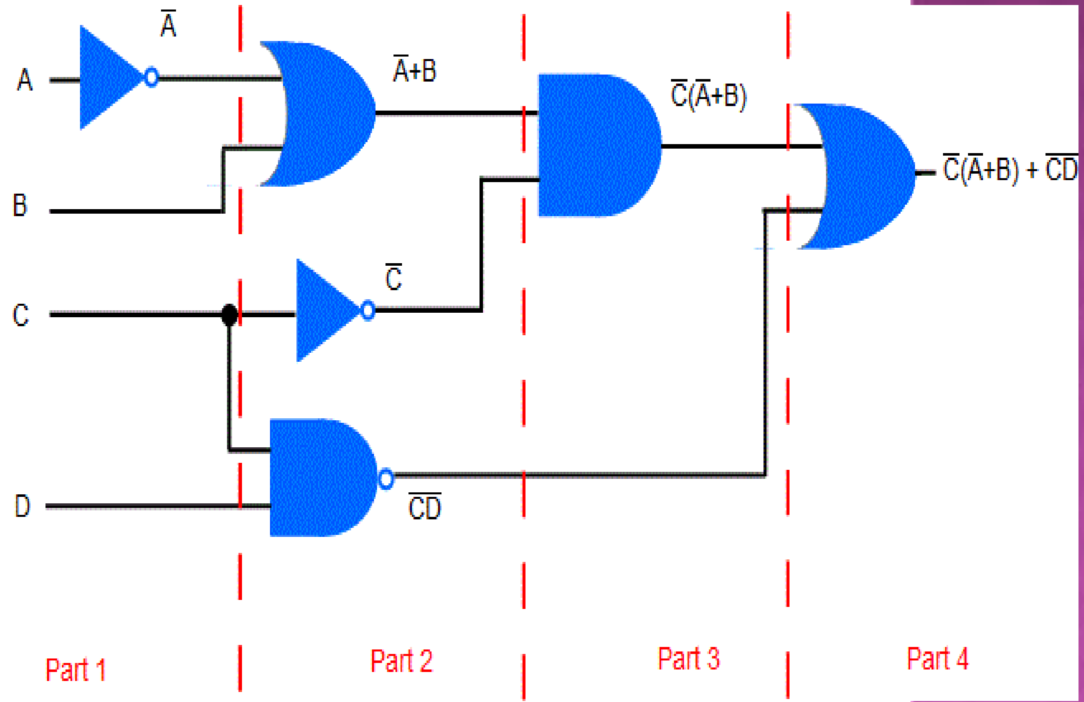
A	B	$\overline{A}$	$\overline{B}$	A.B	$\overline{A.B}$	A+B	$(\overline{A.B})(A+B)$
0	0	1	1	0	1	0	0
0	1	1	0	0	1	1	1
1	0	0	1	0	1	1	1
1	1	0	0	1	0	1	0

DERIVING BOOLEAN EXPRESSION FROM GIVEN CIRCUIT: EXAMPLE

- **Example 2**
  Consider the following logic gate diagram.



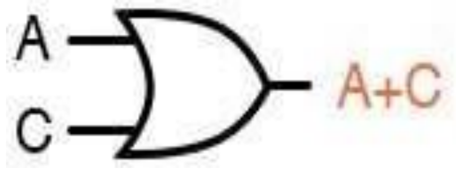
- **Example 3**
  It doesn't matter how many inputs you have,or even if
  your circuit diagram includes NAND or NOR gates.
  Consider the following logic gate diagram.



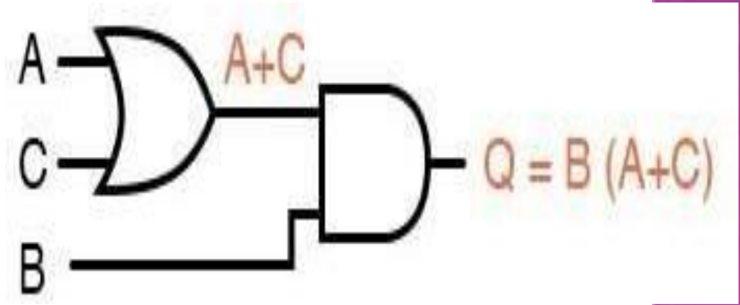
GENERATING SCHEMATIC DIAGRAMS FROM BOOLEAN EXPRESSIONS

- **Example: B(A + C)**
- To do this, evaluate the expression, following proper mathematical order of operations (multiplication before addition, operations inside parentheses before anything else), and draw gates for each step.
- Remember again that OR gates are equivalent to Boolean addition, while AND gates are equivalent to Boolean multiplication.
- In this case, we would begin with the sub-expression "A + C", which is an OR gate.
- The next step in evaluating the expression "B(A + C)" is to multiply (AND
gate) the signal B by the output of the previous gate (A + C).

Step 1:

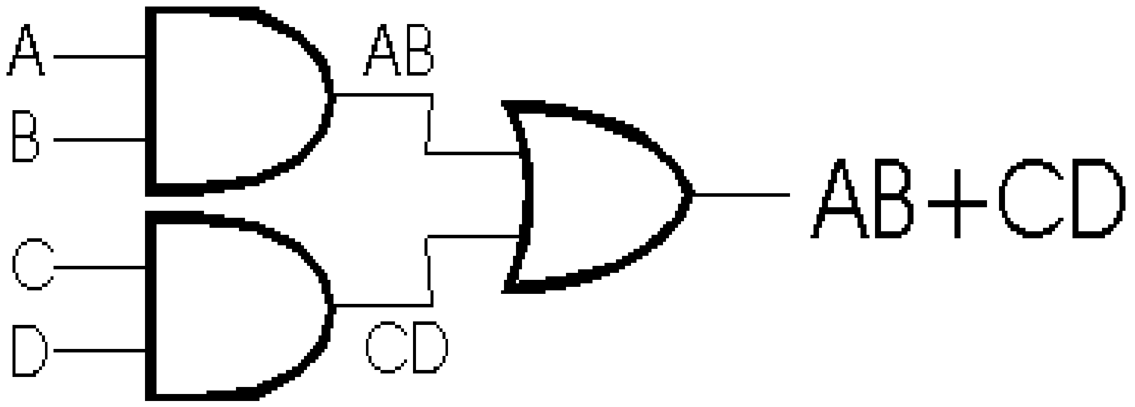


Step 2:

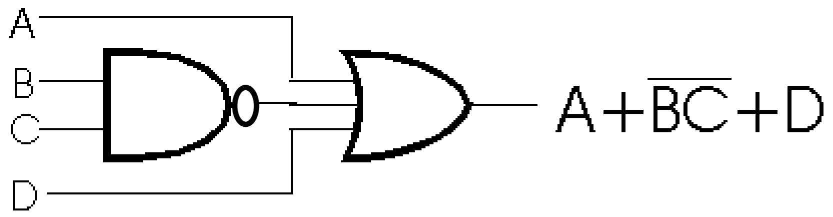


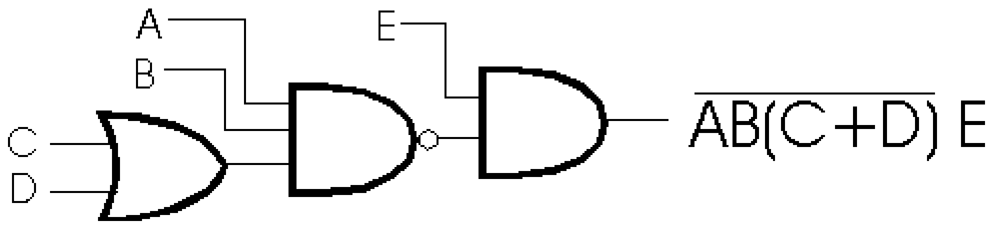
GENERATING SCHEMATIC DIAGRAMS FROM BOOLEAN EXPRESSIONS: EXAMPLES

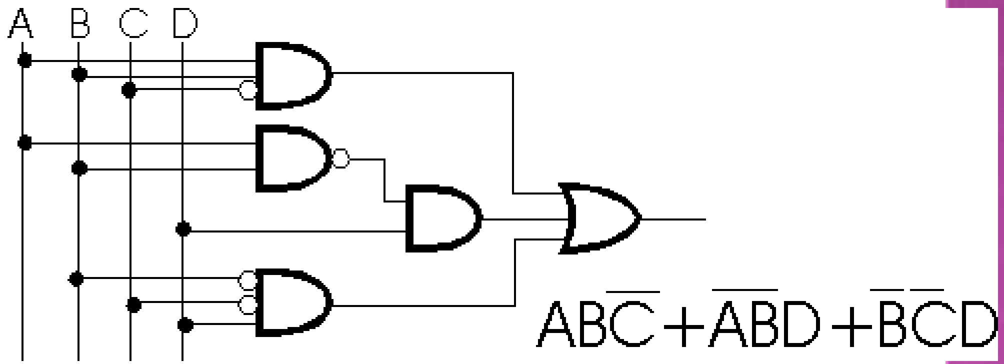
- Consider the Boolean expression **AB+CD**.
  The corresponding digital logic circuit is:



Other examples:







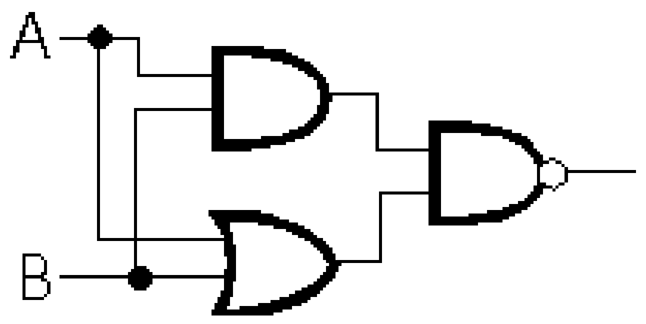
GENERATING SCHEMATIC DIAGRAMS FROM BOOLEAN EXPRESSIONS: HOMEWORK

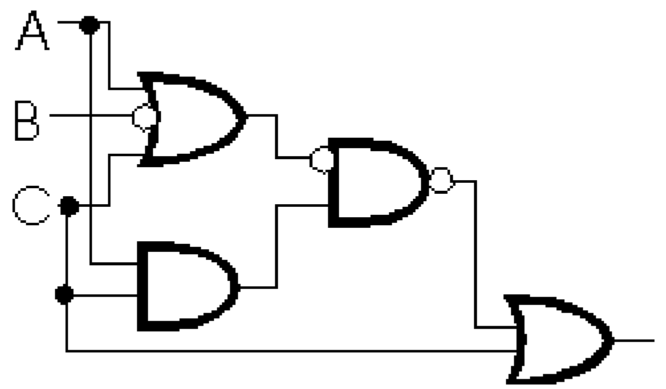
1. Draw digital logic circuits for these Boolean expressions:

$$(AB+CD)\overline{AD}$$

$$(AC+\overline{B+CD})E$$

1. Give the equivalent Boolean expressions for these digital logic circuits

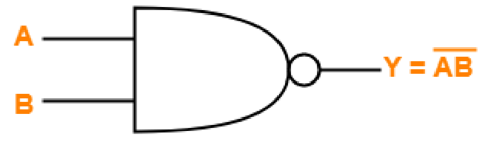




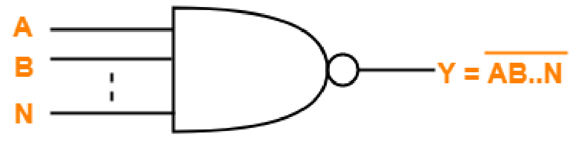
UNIVERSAL LOGIC GATES & IMPLEMENTATION OF OTHER GATES USING UNIVERSAL GATES

- They are called as"**Universal Gates**" because-
  - They can realize all the binary operations.
  - All the basic logic gates can be derived from them.
- They have the following properties-
  - Universal gates are not associative in nature.
  - Universal gates are commutative in nature.
- There are following two universal logic gates-
  - NAND Gate
  - NOR Gate

1. NAND Gate-
A NAND Gate is constructed by connecting a NOT Gate at the output terminal of the AND Gate.
The output of NAND gate is high ('1') if at least one of its inputs is low ('0').
The output of NAND gate is low ('0') if all of its inputs are high ('1').

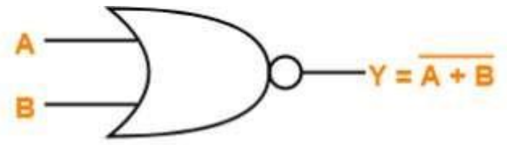


2-Input NAND Gate

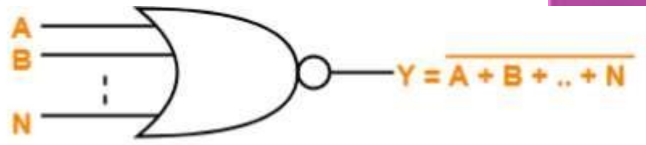


N-Input NAND Gate

2. NOR Gate-
A NOR Gate is constructed by connecting a NOT Gate at the output terminal of the OR Gate.
The output of OR gate is high ('1') if all of its inputs are low ('0').
The output of OR gate is low ('0') if any of its inputs is high ('1').



2-Input NOR Gate



N-Input NOR Gate

NAND GATE IS A UNIVERSAL GATE

- NAND gates can be connected to form any other logic gates.
- Figures 1,2,3 show how NAND gates can be connected to form INVERTER, AND, and OR gates.
- These gates can be combined to form the other logic gates according to the symbolic logic definitions.

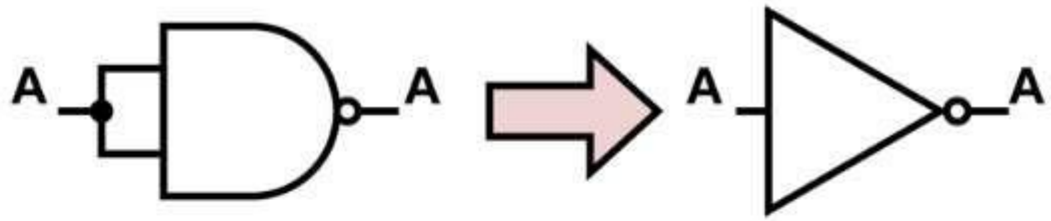


Figure 1: Connecting a NAND gate to make an INVERTER gate

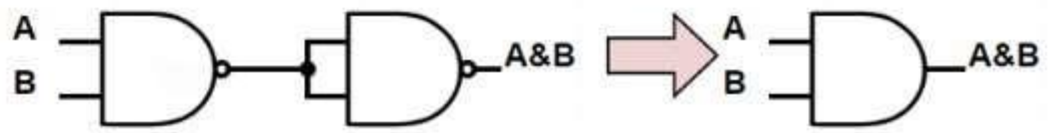


Figure 2: Connecting NAND gates to make an AND gate

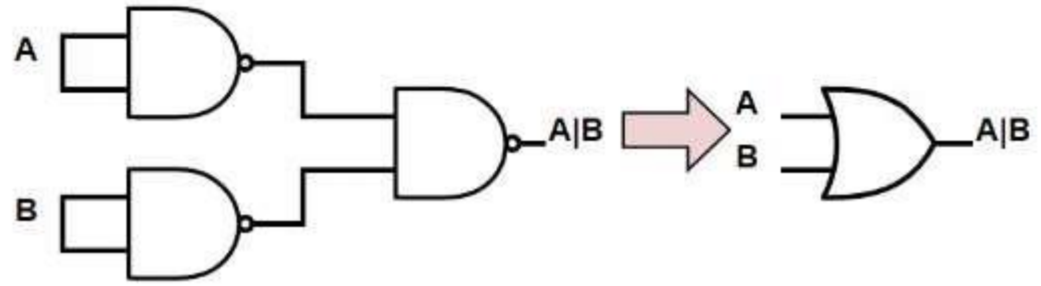
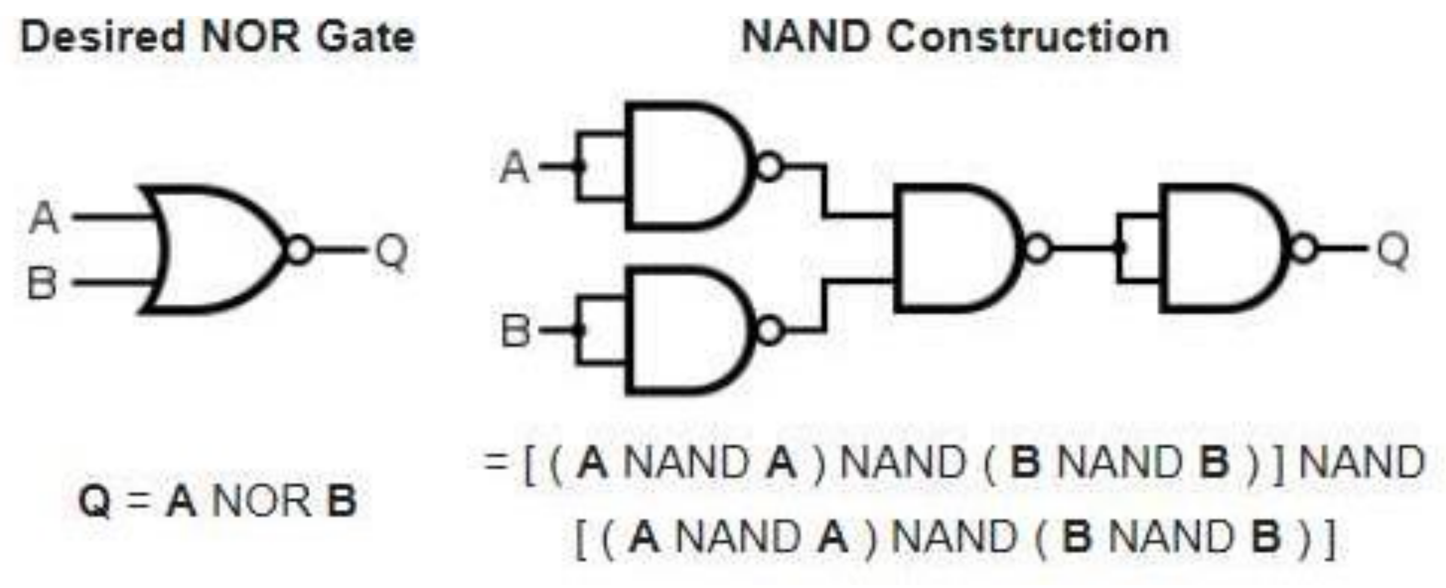


Figure 3: Connecting NAND gates to make an OR gate

NAND AS NOR



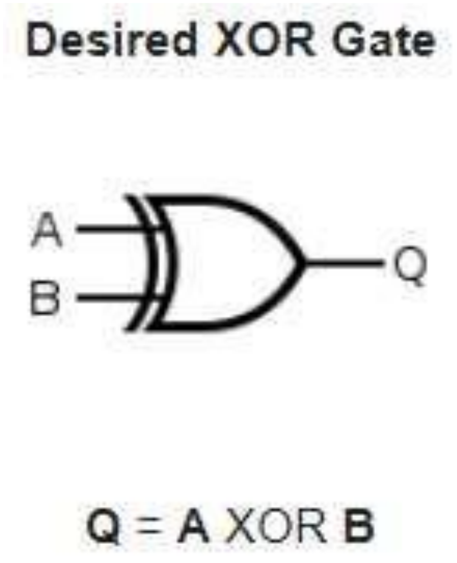
Q = A NOR B

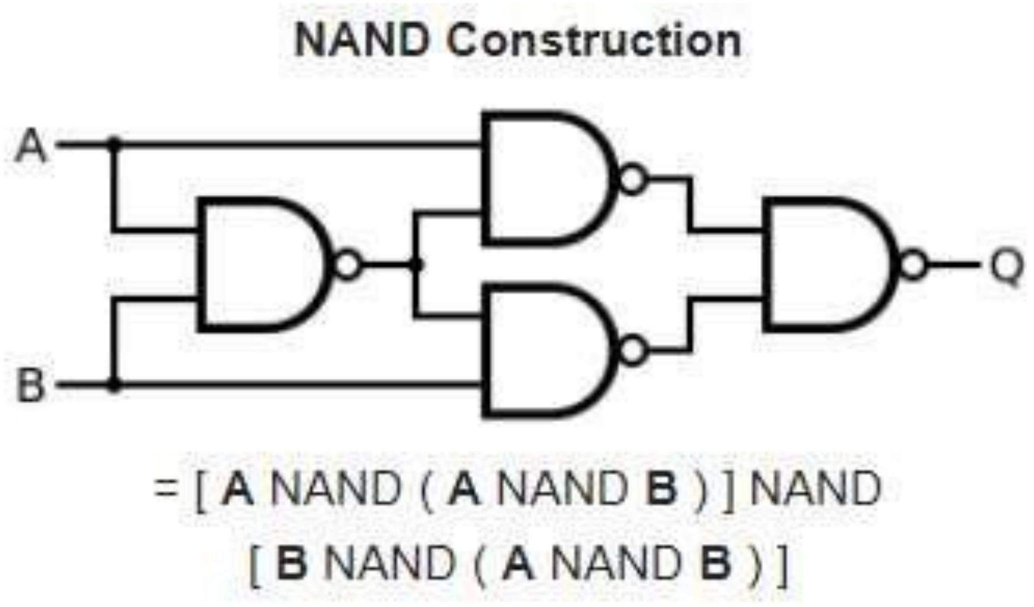
= [(A NAND A) NAND (B NAND B)] NAND [(A NAND A) NAND (B NAND B)]

Truth Table

Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	0

NAND AS XOR

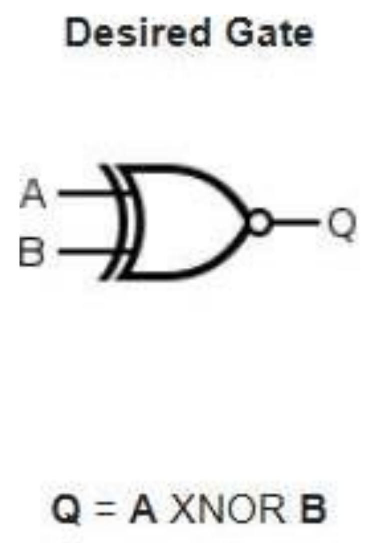


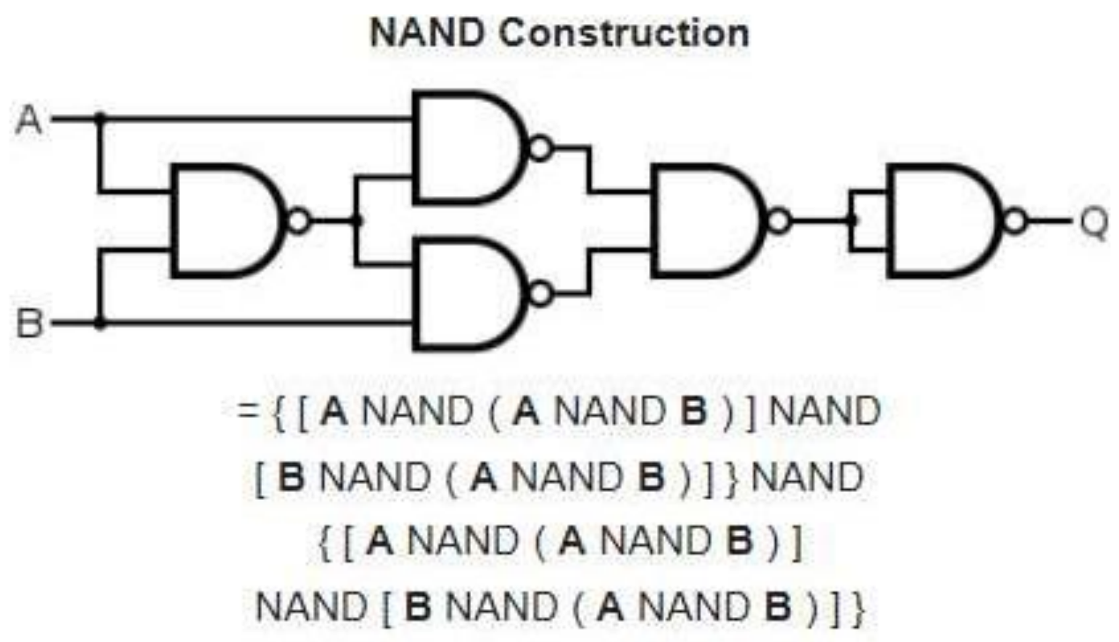


Truth Table

Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	0

NAND AS XNOR





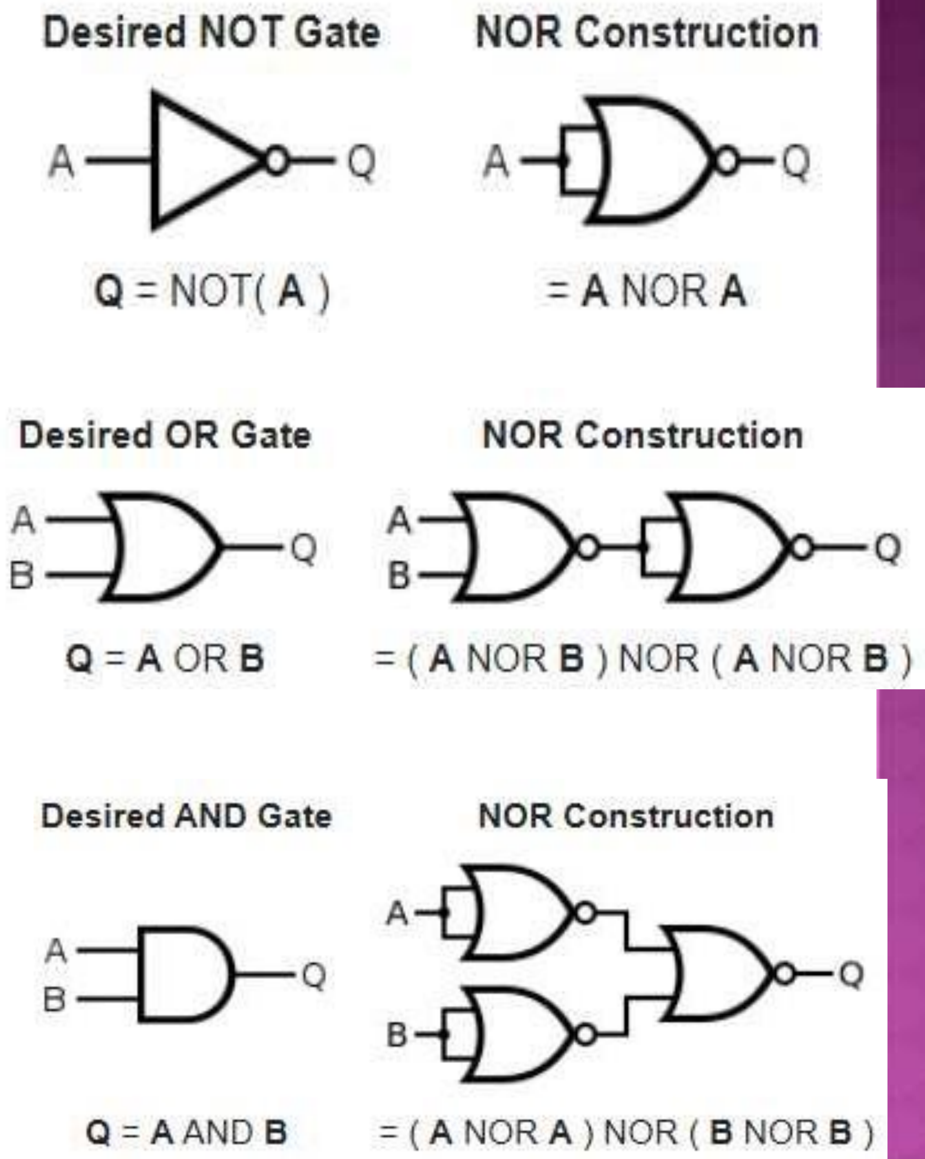
= { [A NAND (A NAND B)] NAND [B NAND (A NAND B)] } NAND { [A NAND (A NAND B)] NAND [B NAND (A NAND B)] }

Truth Table

Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	1

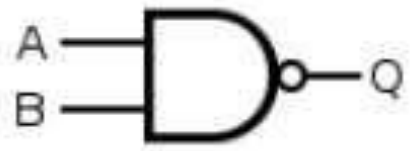
NOR GATE IS A UNIVERSAL GATE

- NOT gate is made by joining the inputs of a NOR gate.As a NOR gate is equivalent to an OR gate leading to NOT gate, this automatically sees to the "OR" part of the NOR gate, eliminating it from consideration and leaving only the NOT part.
- An OR gate is made by inverting the output of a NOR gate. Note that we already know that a NOT gate is equivalent to a NOR gate with its inputs joined.
- An AND gate gives a 1 output when both inputs are 1. Therefore,an AND gate is made by inverting the inputs of a NOR gate. Again, note that a NOT gate is equivalent to a NOR with its inputs joined.



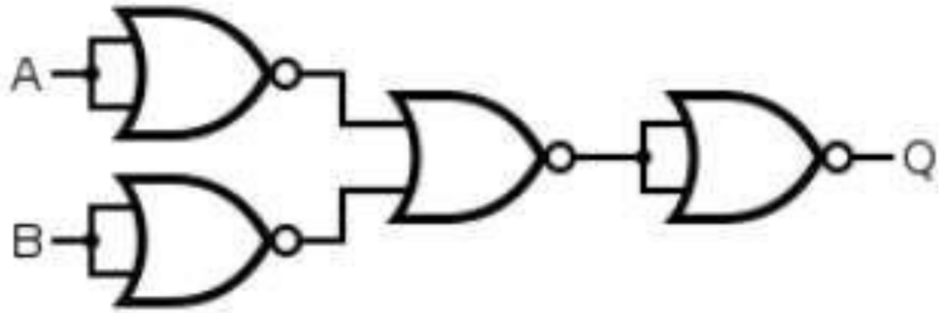
NOR AS NAND

Desired NAND Gate



Q = **A** NAND **B**

NOR Construction



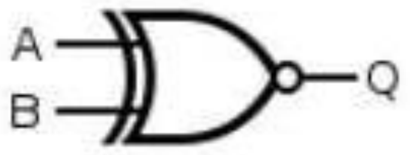
= [(**A** NOR **A**) NOR (**B** NOR **B**)] NOR [(**A** NOR **A**) NOR (**B** NOR **B**)]

Truth Table

Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

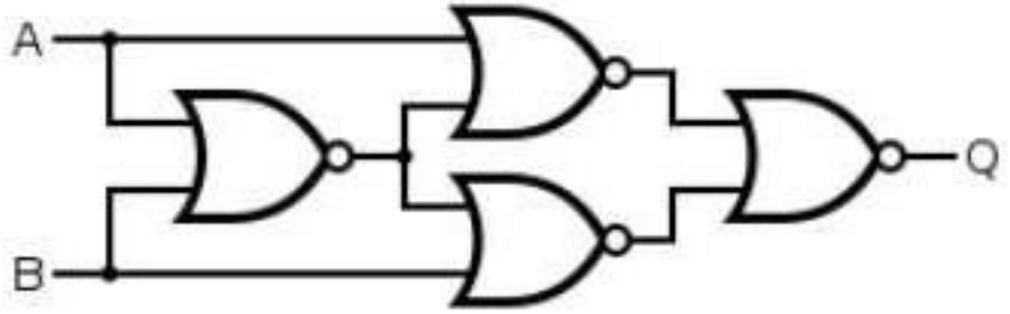
NOR AS XNOR

Desired XNOR Gate



Q = **A** XNOR **B**

NOR Construction



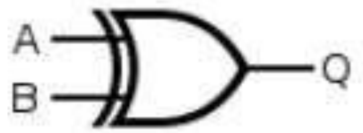
= [**A** NOR (**A** NOR **B**)] NOR [**B** NOR (**A** NOR **B**)]

Truth Table

Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	1

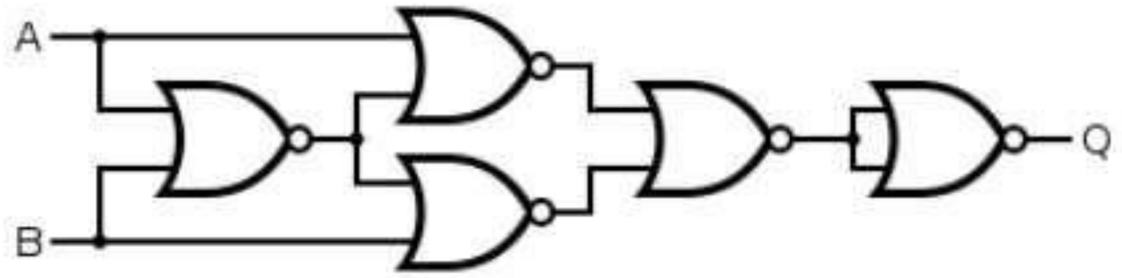
NOR AS XOR

Desired Gate



Q = **A** XOR **B**

NOR Construction



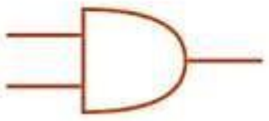
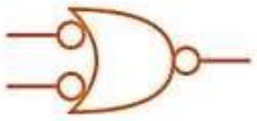
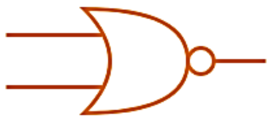
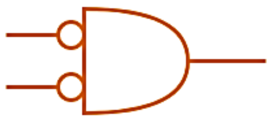
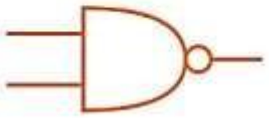
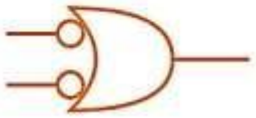
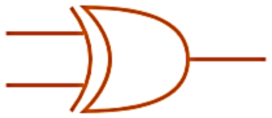
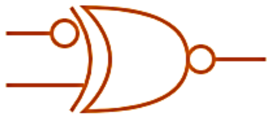
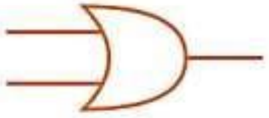
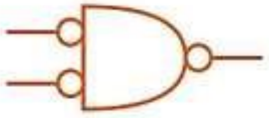
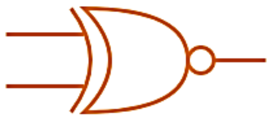
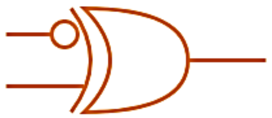
= { [**A** NOR (**A** NOR **B**)] NOR
[**B** NOR (**A** NOR **B**)] } NOR
{ [**A** NOR (**A** NOR **B**)]
NOR [**B** NOR (**A** NOR **B**)] }

Truth Table

Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	0

INPUT BUBBLED LOGIC:ALTERNATIVE LOGIC GATES

- Alternative logic gate is an alternate logic gate that produces the same output as the original logic gate.
- and can be used during the unavailability of the original logic gate to serve the same purpose.
- Alternative logic gates are also called as **Alternate Gates**.
- Alternative logic gates are also called as **Bubbled Gates** since they contain bubbles in them.
- The following table shows the original logic gate and its corresponding alternate gate

Original Gates	Alternate Gates	Original Gates	Alternate Gates
			
			
			

Bubble pushing is a technique to apply De Morgan's theorem directly to the logic diagram.

1. Change the logic gate (AND to OR and OR to AND).
2. Add bubbles to the inputs and outputs where there were none, and remove the original bubbles.