UNIT-3: BOOLEAN ALGEBRA AND LOGIC GATES(4 Hrs)

Topics covered:

- Basic definition, Basic properties and theorem of Boolean algebra, DeMorgan's theorem,
- Logic gates and truth tables, Universality of NAND and NOR gates, Trio-stage logic.

Basics.....

- ✓ Logic gates are the fundamental building blocks of digital systems.
- ✓ The name logic gate is derived from the ability of such devices to make decisions, in the sense that it produces one output level when some combinations of input levels are present

Basics.....

- **Inputs & Outputs for Logic Circuits**

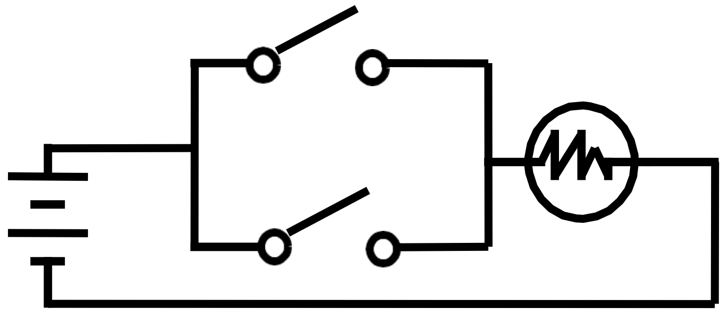
  - ✓ Input & Output of logic gates can occur only in two levels.

HIGH	LOW
True	False
ON	OFF
1	0

Basics.....

- **Truth Table**
  - ✓ A table which lists all the possible combinations of input variables and the corresponding outputs is called a "Truth Table".
  - ✓ It shows how the logic circuits output responds to various combinations of logic levels at the inputs

Switches in parallel => OR

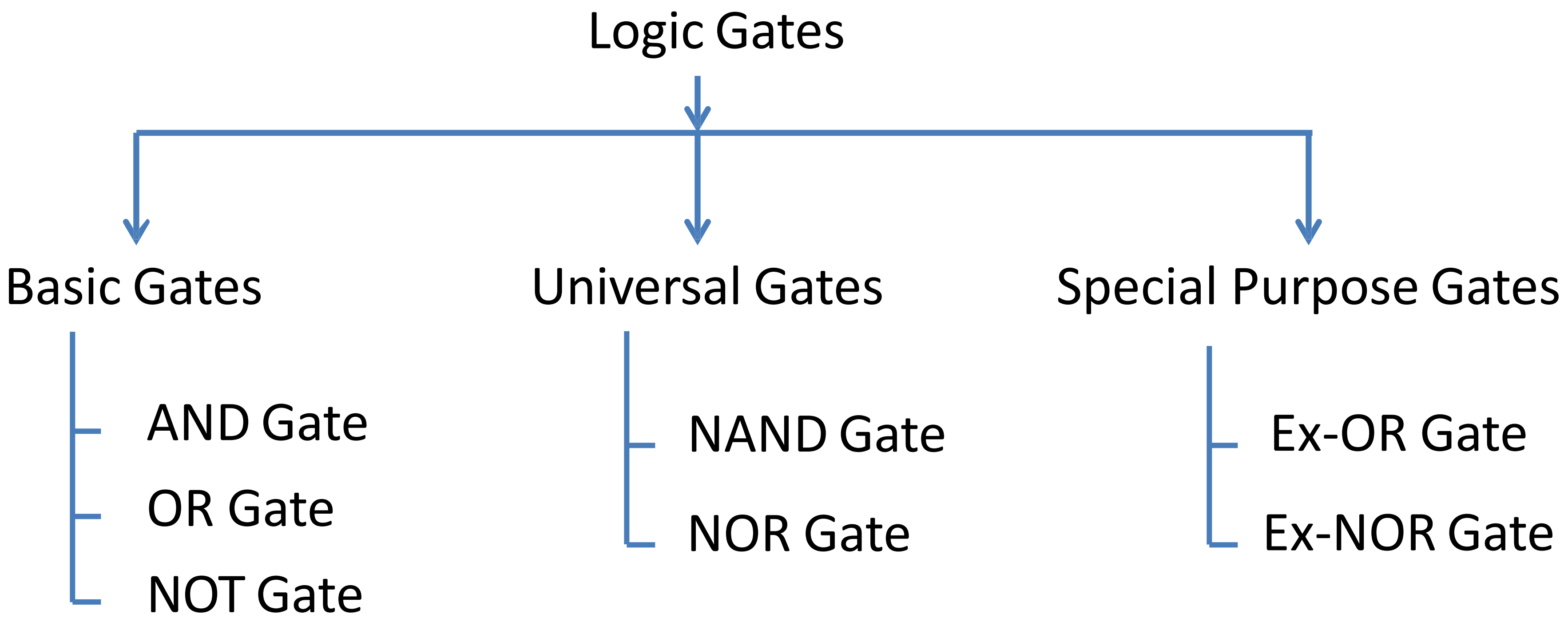


Switch 1	Switch 2	Output
OFF	OFF	OFF
OFF	ON	GLOW
ON	OFF	GLOW
ON	ON	GLOW

Basics.....

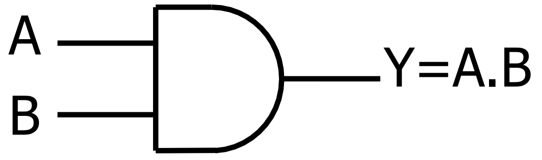
- **Logic**
  - ✓ A logic in which the voltage levels represent logic 1 and logic 0.
  - ✓ Level logic may be Positive or Negative.
  - ✓ A **"Positive Logic"** is the one which the higher of the two voltage levels represents the logic 1 and the lower of the two voltage level represents the logic 0.
  - ✓ A **"Negative Logic"** is the one which the lower of the two voltage levels represents the logic 1 and the higher of the two voltage level represents the logic 0.

Logic Gates


Logic Gates
Basic Gates
Universal Gates
Special Purpose Gates
AND Gate
OR Gate
NOT Gate
NAND Gate
NOR Gate
Ex-OR Gate
Ex-NOR Gate

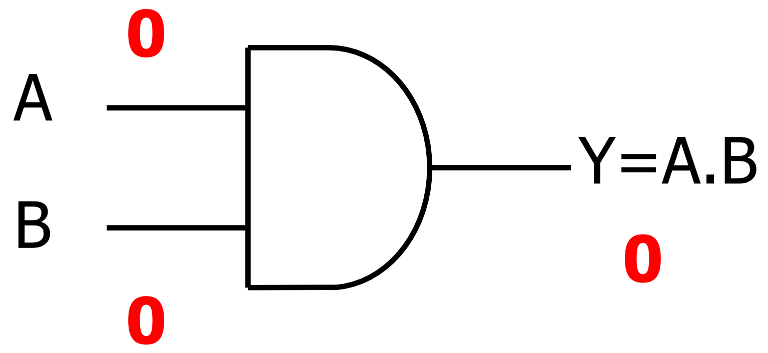
AND Gate

- ✓ An AND gate has two or more inputs but only one output.
- ✓ The output assumes the logic 1 state, when both inputs are at logic 1 state.
- ✓ The output assumes the logic 0 state even if one of its inputs is at logic 0 state.



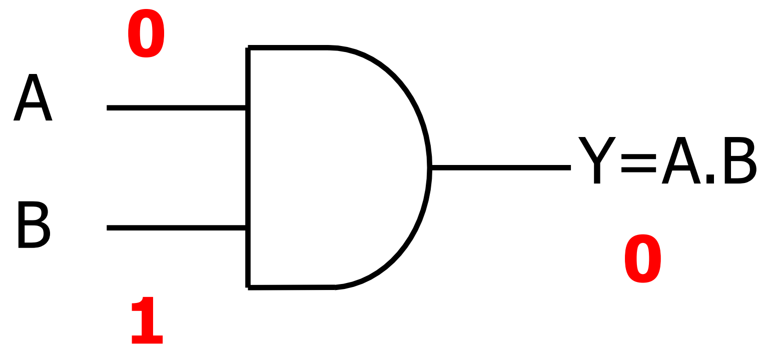
Logic Symbol

AND Gate



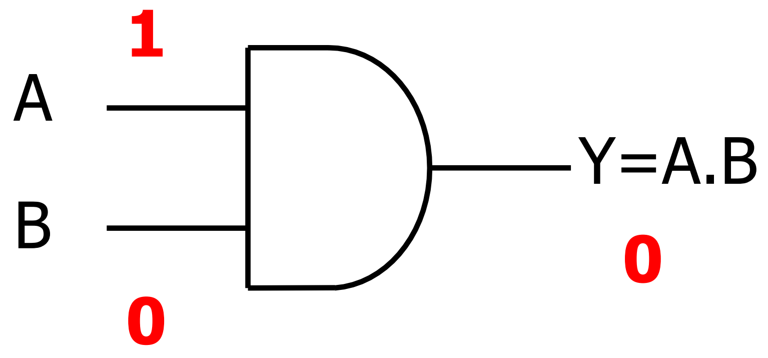
Inputs		Output
A	B	Y=A.B
0	0	0

AND Gate



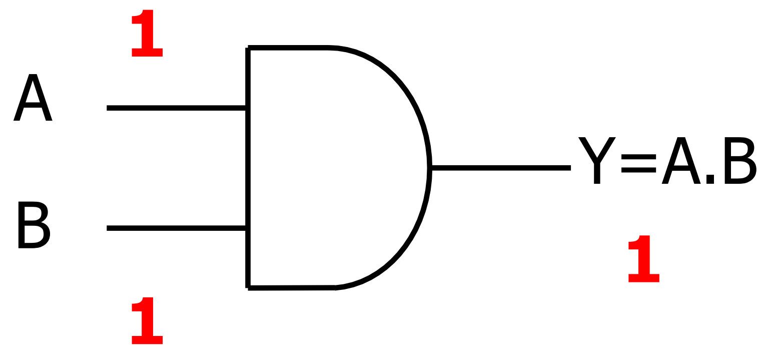
Inputs		Output
A	**B**	**Y=A.B**
0	0	0
0	**1**	**0**

AND Gate



Inputs		Output
A	**B**	**Y=A.B**
0	0	0
0	1	0
1	**0**	**0**

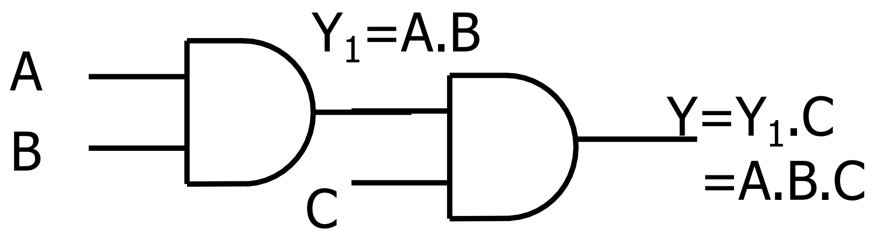
AND Gate



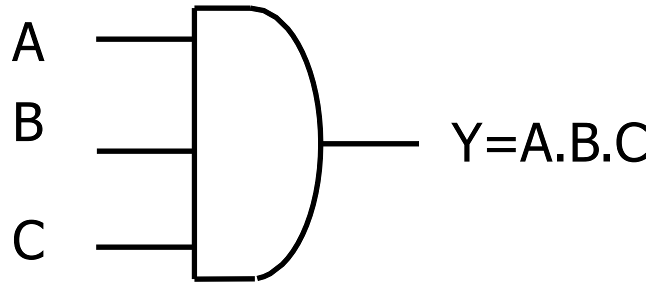
Inputs		Output
A	**B**	**Y=A.B**
0	0	0
0	1	0
1	0	0
1	**1**	**1**

3 - Input AND Gate

3 – Input AND Gate using 2 – Input AND Gate



Symbol : 3 – Input AND Gate

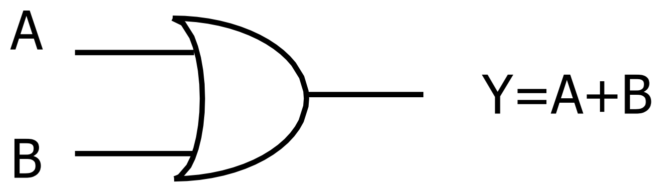


Truth Table : 3 – Input AND Gate

Input			**Output Y=A.B.C**
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

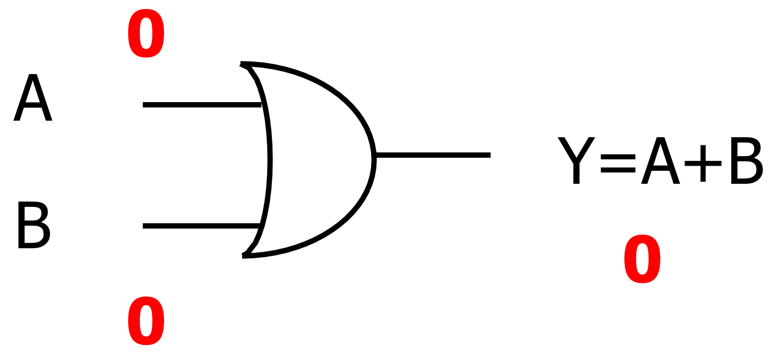
OR Gate

- ✓ An OR gate has two or more inputs but only one output.
- ✓ The output assumes the logic 1 state, when even if one of its inputs is in logic 1 state.
- ✓ The output assumes the logic 0 state only when both the inputs are in logic 0 state.



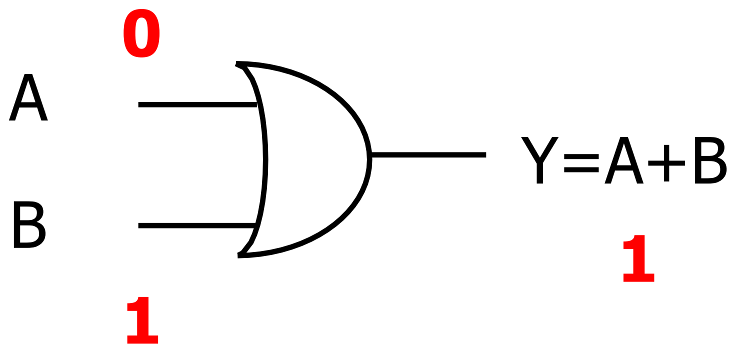
Logic Symbol

OR Gate



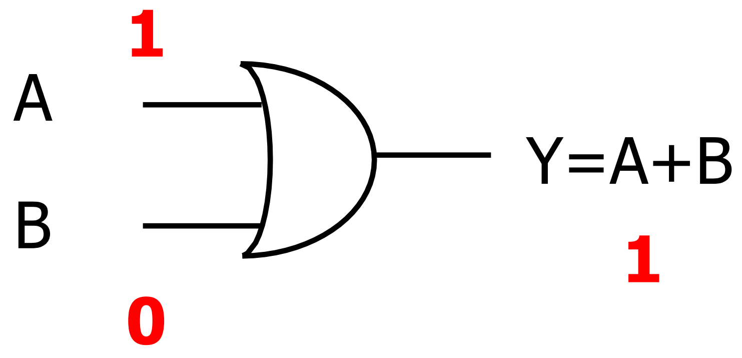
Inputs		Output
A	B	Y=A+B
0	0	0

OR Gate



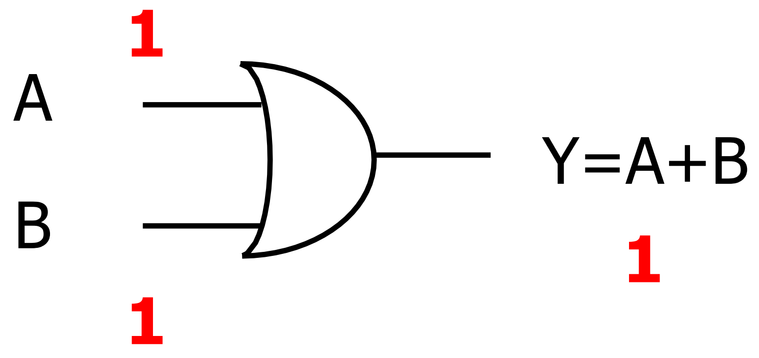
Inputs		Output
A	B	Y=A+B
0	0	0
0	**1**	**1**

OR Gate



Inputs		Output
A	B	Y=A+B
0	0	0
0	1	1
1	0	1

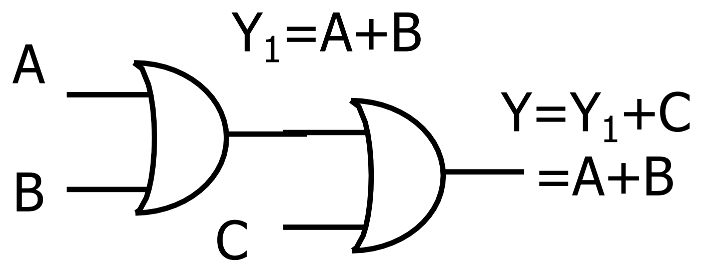
OR Gate



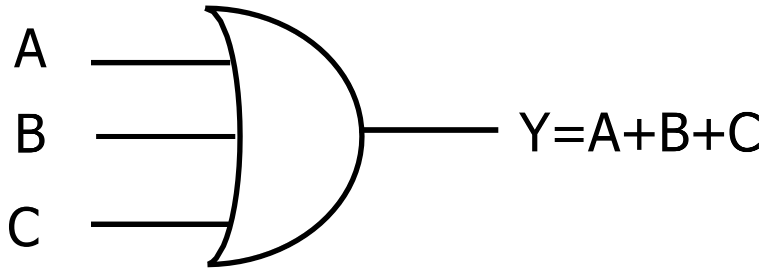
Inputs		Output
A	**B**	**Y=A+B**
0	0	0
0	1	1
1	0	1
1	**1**	**1**

Three Input OR Gate

3 – Input OR Gate using 2 – Input OR Gate



Symbol : 3 – Input OR Gate

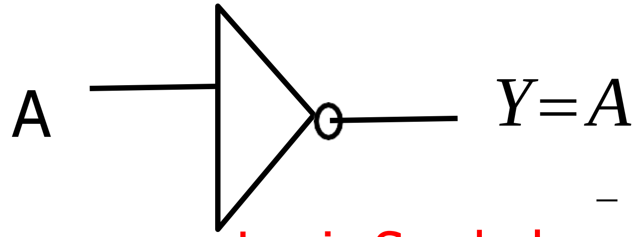


Truth Table : 3 – Input OR Gate

Input			**Output Y=A+B+C**
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

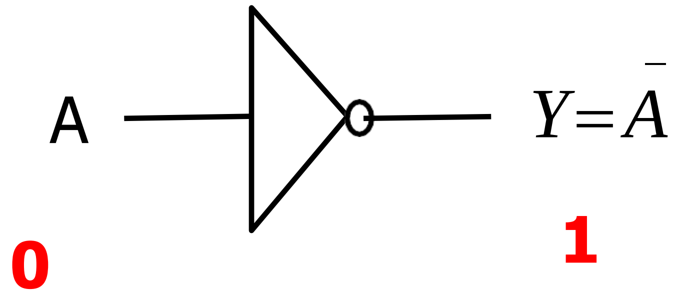
NOT Gate (Inverter)

- ✓ A NOT gate, also called inverter, has only one input and of course only one output.
- ✓ It is a device whose output is always the complement of its input.
- ✓ That is, the output of a NOT gate assumes the logic 1 state when its input is in logic 0 state and vice versa.



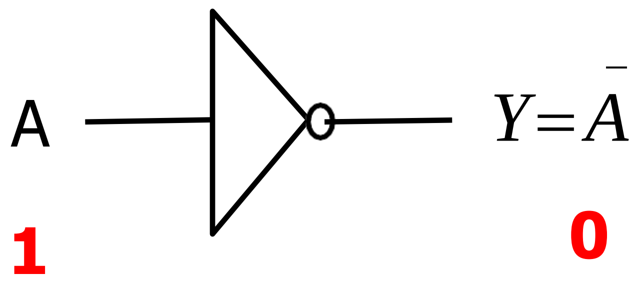
Logic Symbol

NOT Gate



Input	Output
A	$Y=\bar{A}$
0	1

NOT Gate



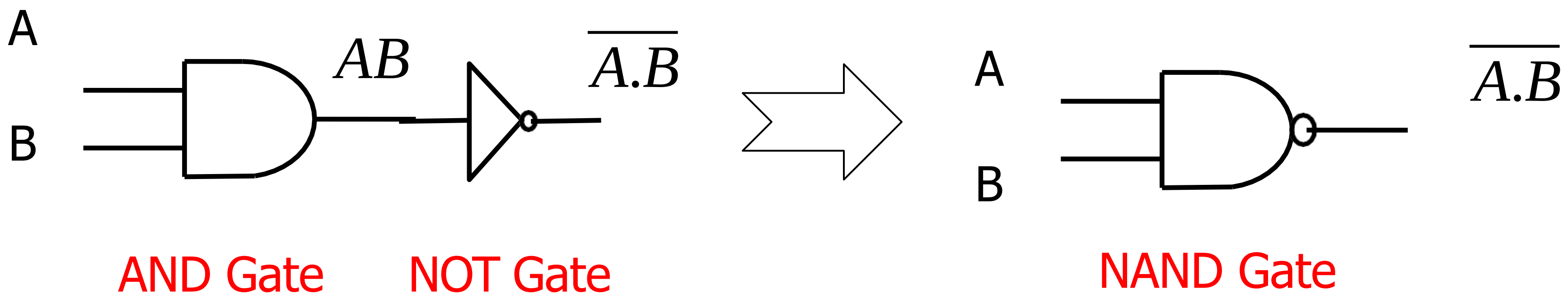
Input	Output
A	$Y=\bar{A}$
0	1
1	**0**

Universal Gates (NAND and NOR Gate)

- ✓ NAND and NOR gates are Universal Gates.
- ✓ Both NAND and NOR gates can perform all the three basic logic functions (AND, OR and NOT).
- ✓ Therefore, AOI logic can be converted to NAND logic or NOR logic

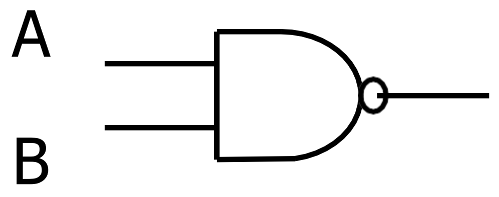
NAND Gate

- ✓ NAND means NOT AND i.e. AND output is inverted.
- ✓ So NAND gate is a combination of an AND gate and a NOT gate.



NAND Gate

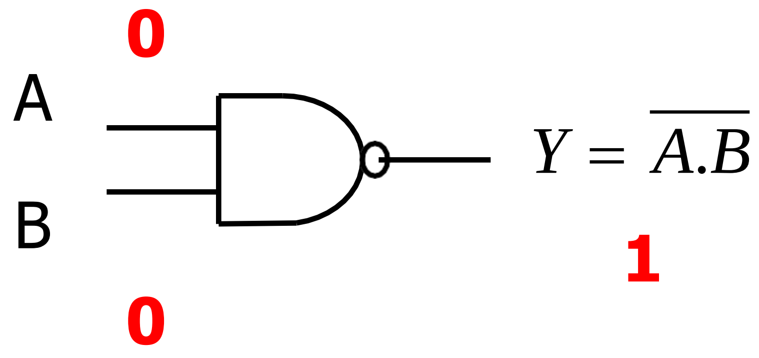
- ✓ The output is logic 0 level, only when all the inputs are logic 1 level.
- ✓ For any other combination of inputs, the output is a logic 1 level.



$Y = \overline{A.B}$

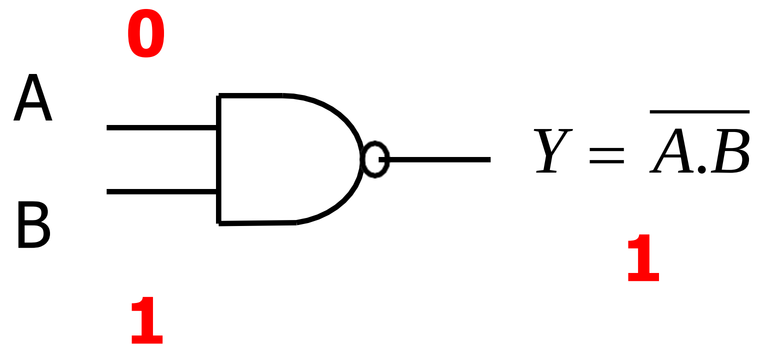
Logic Symbol

NAND Gate



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

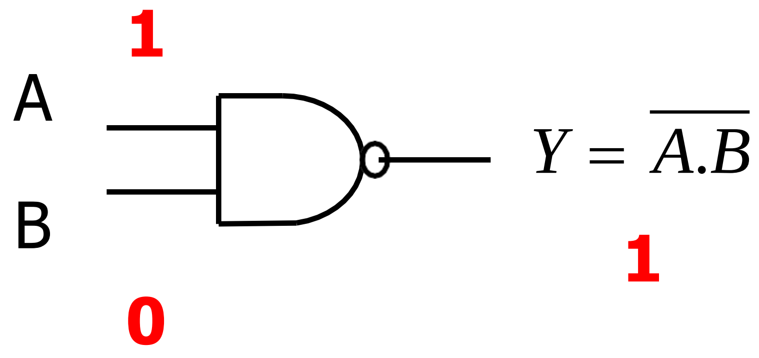
NAND Gate



$Y = \overline{A.B}$

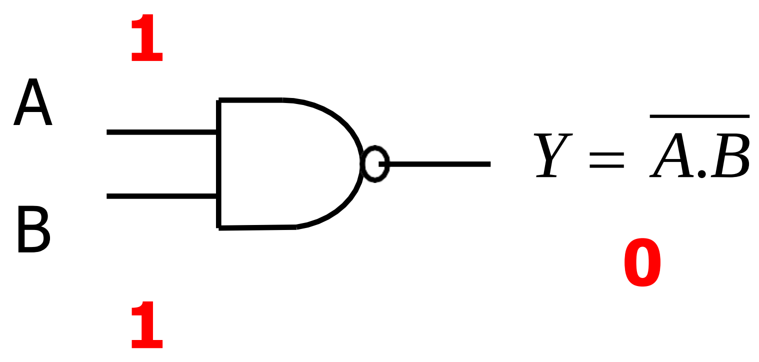
Inputs		Output
A	B	$Y = \overline{A.B}$
0	0	1
0	**1**	**1**

NAND Gate



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

NAND Gate

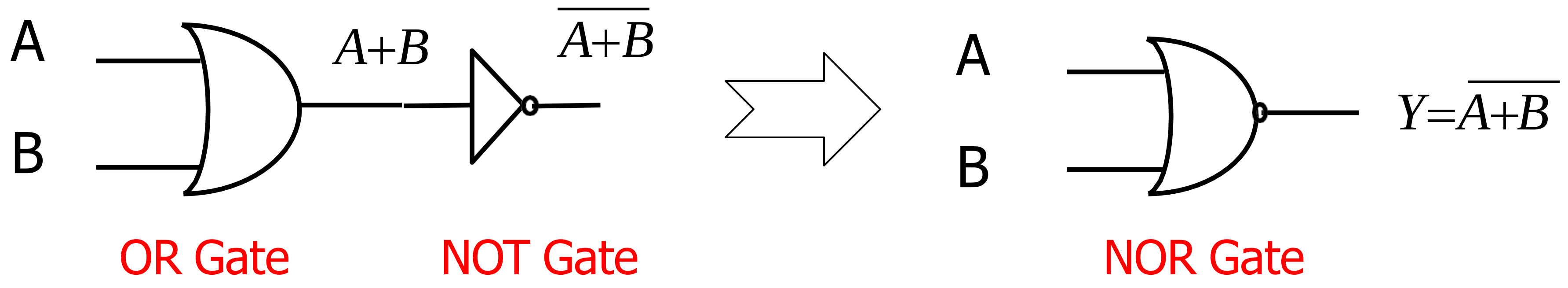


$Y = \overline{A.B}$

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

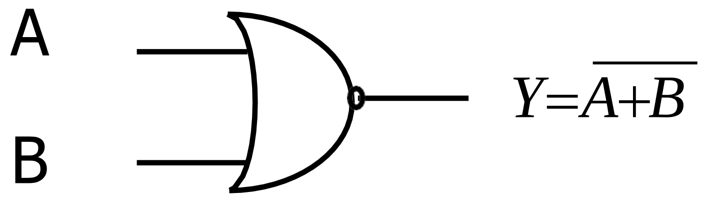
NOR Gate

- ✓ NOR means NOT OR i.e. OR output is inverted.
- ✓ So NOR gate is a combination of an OR gate and a NOT gate.



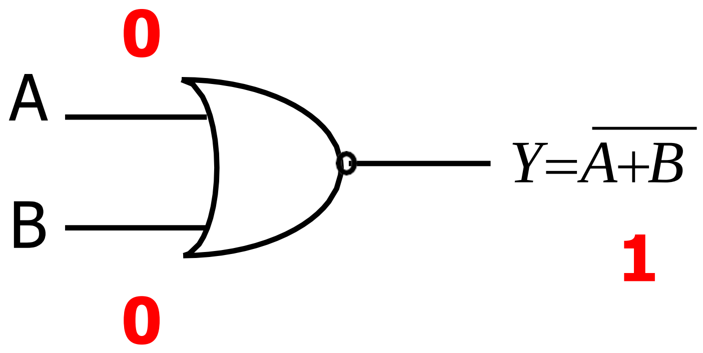
NOR Gate

- ✓ The output is logic 1 level, only when all the inputs are logic 0 level.
- ✓ For any other combination of inputs, the output is a logic 0 level.



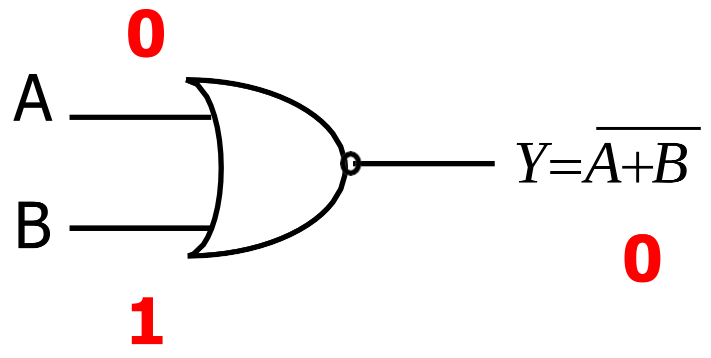
Logic Symbol

NOR Gate



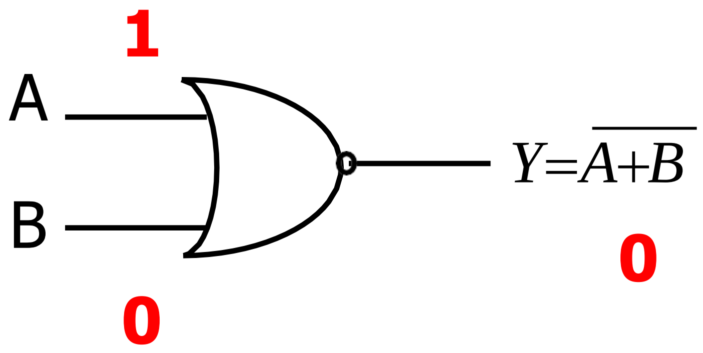
Inputs		Output
A	B	$Y=\overline{A+B}$
0	0	1

NOR Gate



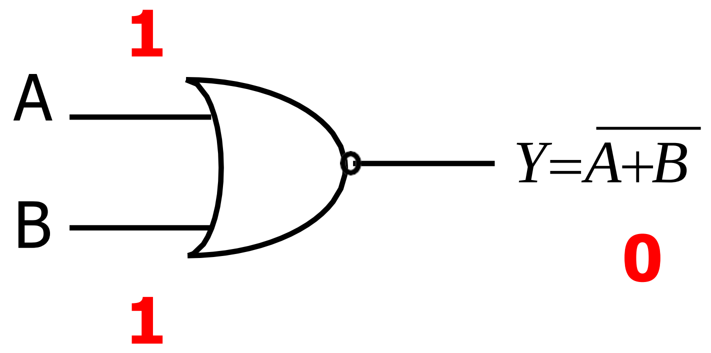
Inputs		Output
A	B	$Y = \overline{A+B}$
0	0	1
0	**1**	**0**

NOR Gate



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

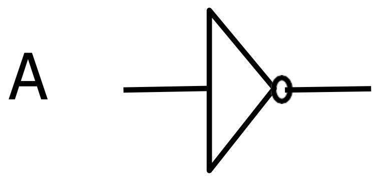
NOR Gate



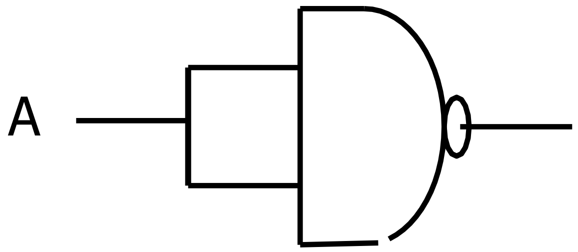
Inputs		Output
A	B	$Y=\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	**1**	**0**

Universal Gate

NOT Gate using NAND Gate



$Y=\bar{A}$

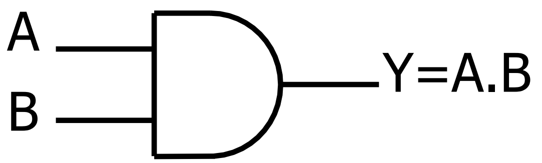


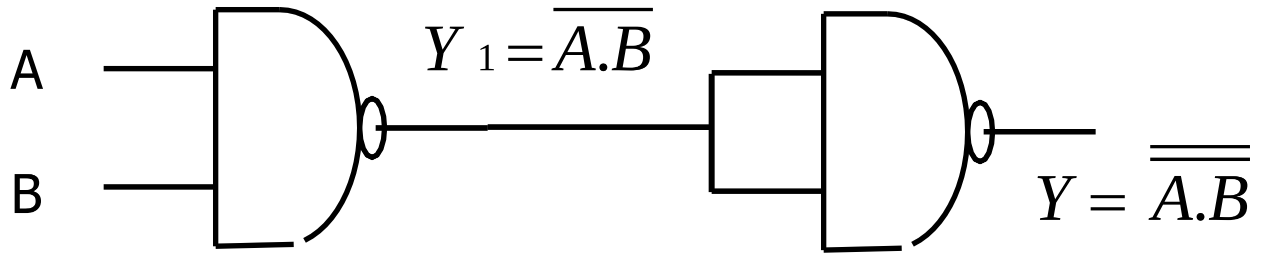
$Y=\overline{A.A}$

$Y=\overline{A}$ ……… (A.A=A)

Universal Gate

AND Gate using NAND Gate





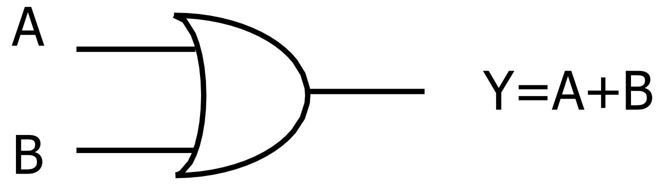
$$Y_1 = \overline{A.B}$$

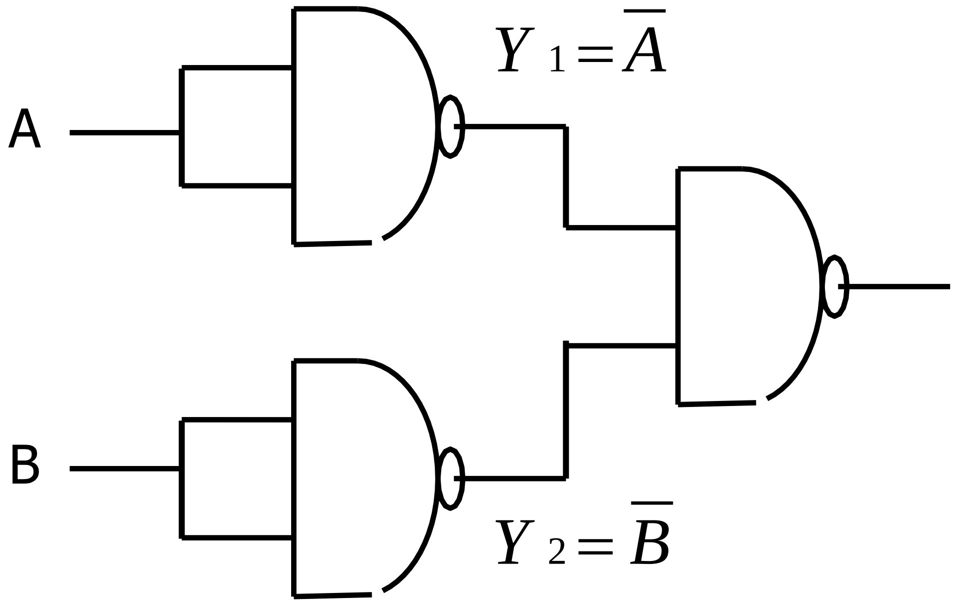
$$Y = \overline{\overline{A.B}}$$

$$Y = A.B$$

Universal Gate

OR Gate using NAND Gate





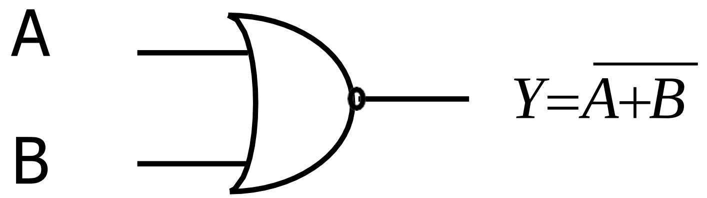
$Y = \overline{\overline{A}.\overline{B}}$

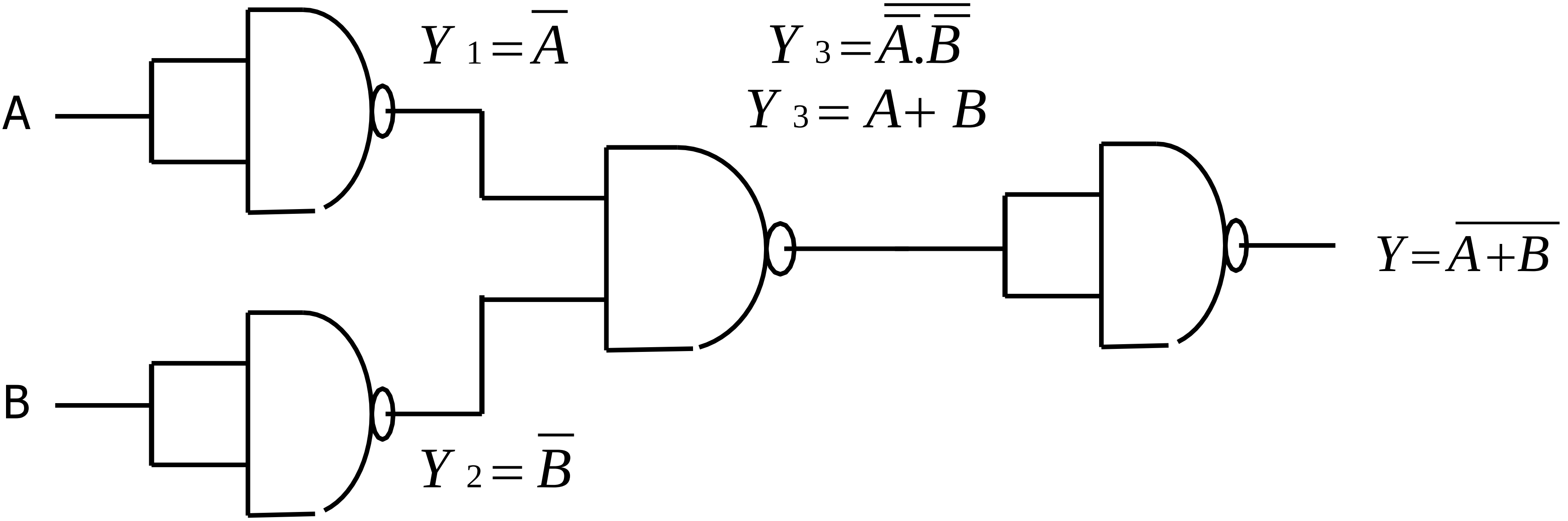
$Y = \overline{\overline{A}} + \overline{\overline{B}}$ (Demorgan's Theorem)

$Y = A + B$

Universal Gate

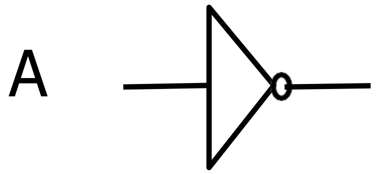
NOR Gate using NAND Gate



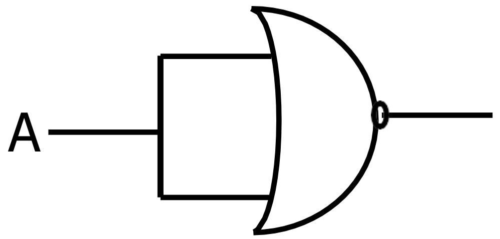


Universal Gate

NOT Gate using NOR Gate



$Y = \overline{A}$

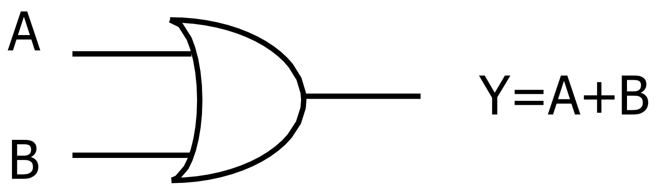


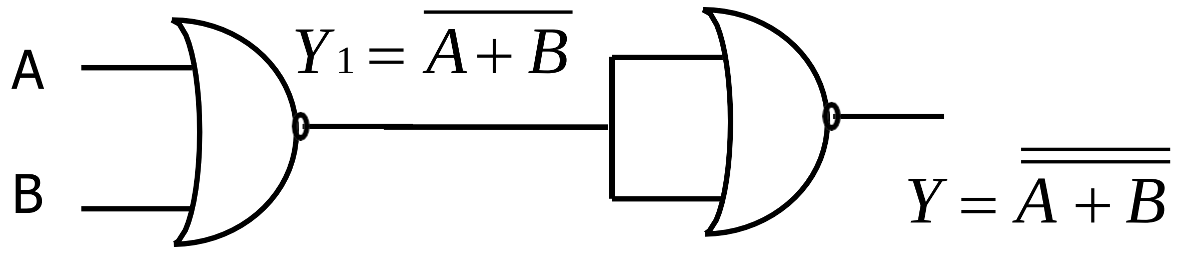
$$Y = \overline{A + A}$$

$$Y = \overline{A} \quad \text{.........} \ (A+A=A)$$

Universal Gate

OR Gate using NOR Gate





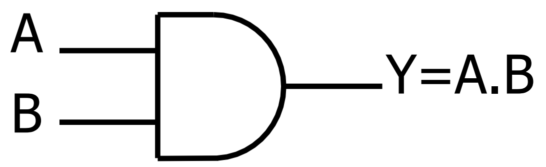
$Y_1 = \overline{A+B}$

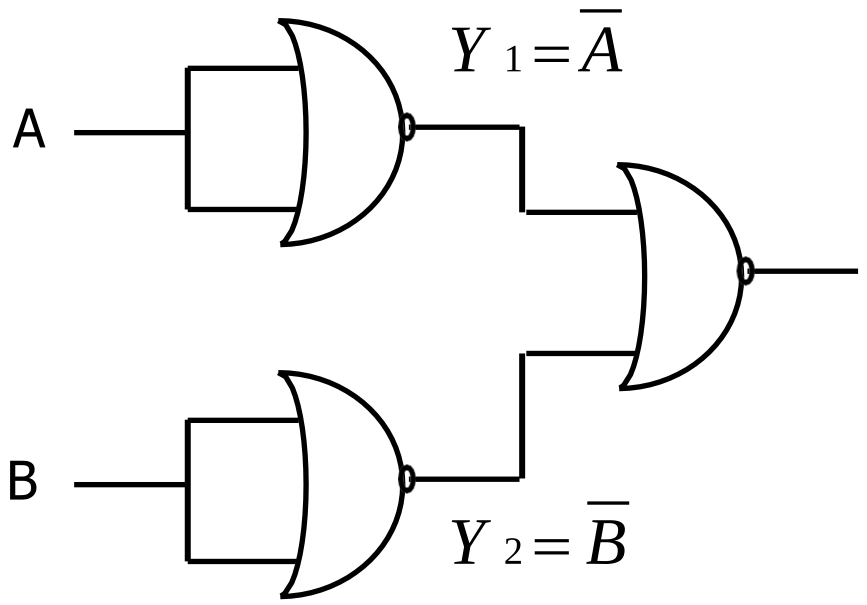
$Y = \overline{\overline{A+B}}$

$Y = A + B$

Universal Gate

AND Gate using NOR Gate





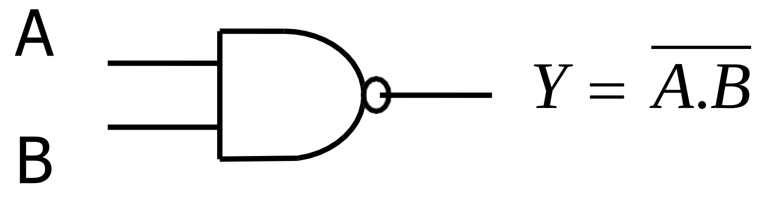
$$Y = \overline{\overline{A} + \overline{B}}$$

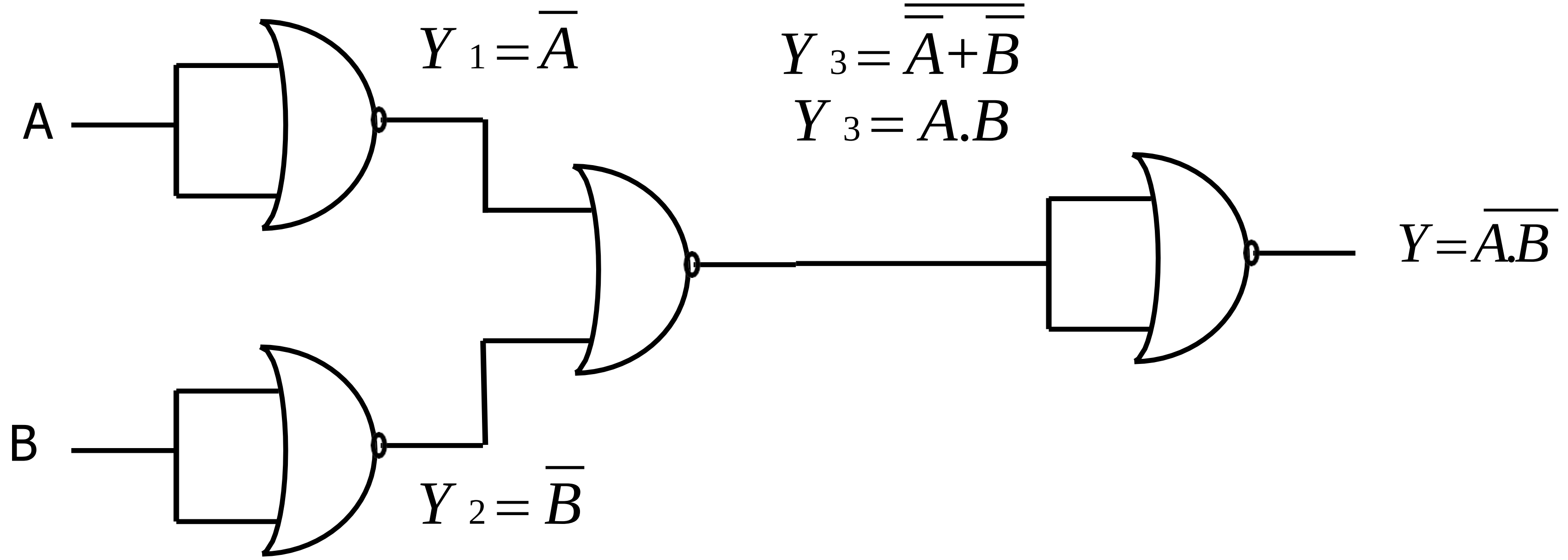
$$Y = \overline{\overline{A}}.\overline{\overline{B}} \quad \text{(Demorgan's Theorem)}$$

$$Y = A.B$$

Universal Gate

NAND Gate using NOR Gate



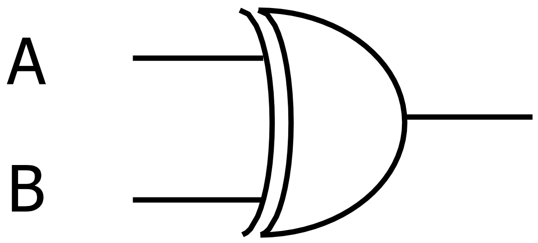


Special Purpose Gate – Ex-OR Gate

- ✓ An Ex-OR gate is two input, one output logic circuit.
- ✓ The output assumes the logic 1 state, when one and only one of its two inputs assumes a logic 1 state.
- ✓ Under the conditions when both the inputs assume the logic 0 state or logic 1 state, the output assumes logic 0.

Ex-OR Gate

- ✓ If input variables are represented by A and B and the output variable by Y the representation for the output of this gate is as



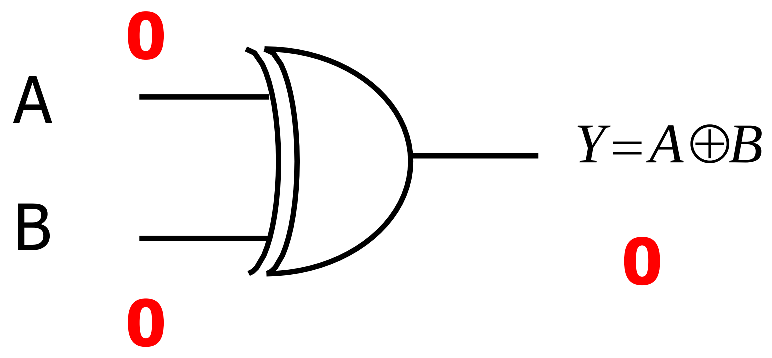
$$Y = A \oplus B$$

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

Logic Symbol

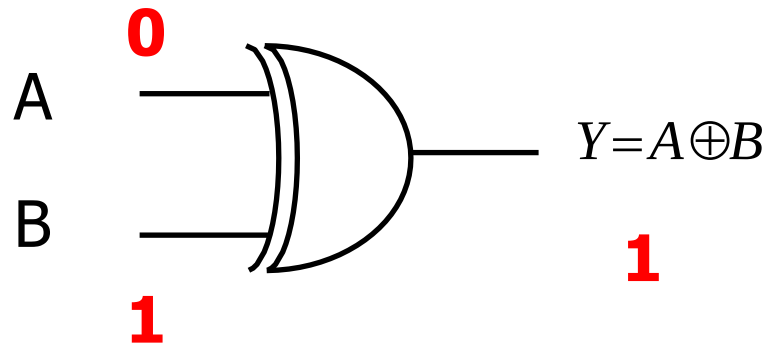
Logic Expression

Ex-OR Gate



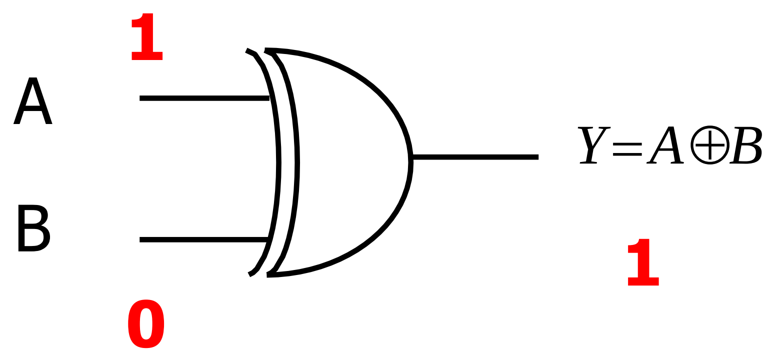
Inputs		Output
A	B	$Y = A \oplus B$
0	0	0

Ex-OR Gate



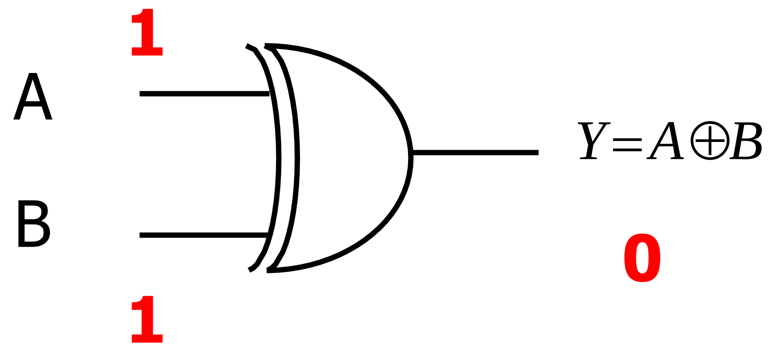
Inputs		Output
A	B	$Y = A \oplus B$
0	0	0
0	**1**	**1**

Ex-OR Gate



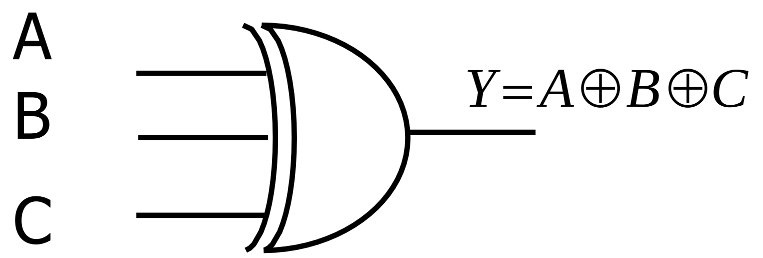
Inputs		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1

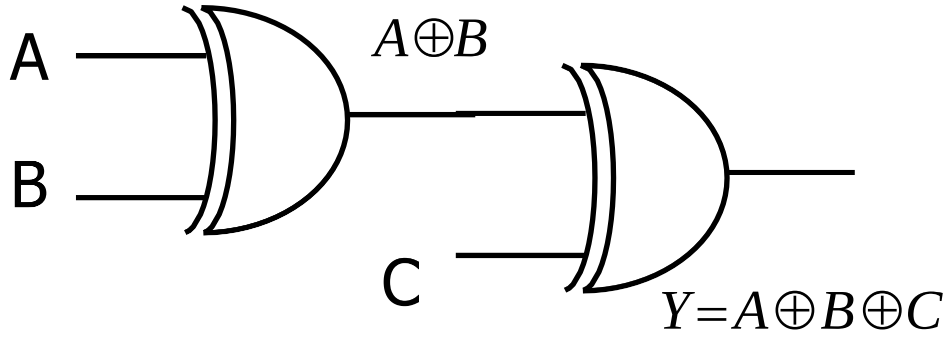
Ex-OR Gate



Inputs		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	**1**	**0**

3 – Input Ex-OR Gate





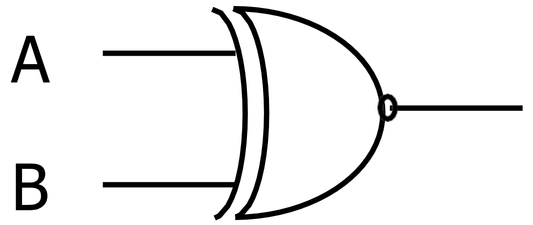
INPUT			OUTPUT
A	B	C	$Y=A\oplus B\oplus C$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Special Purpose Gate – Ex-NOR Gate

- ✓ An Ex-NOR gate is two input, one output logic circuit.
- ✓ The output assumes a logic 0 state, when one of the input assumes a logic 0 state and other a logic 1 state.
- ✓ The output assumes a logic 1 state only when both the inputs assume a logic 0 state or when both the inputs assume a logic state.

Ex-NOR Gate

✓ If input variables are represented by A and B and the output variable by Y the representation for the output of this gate is as



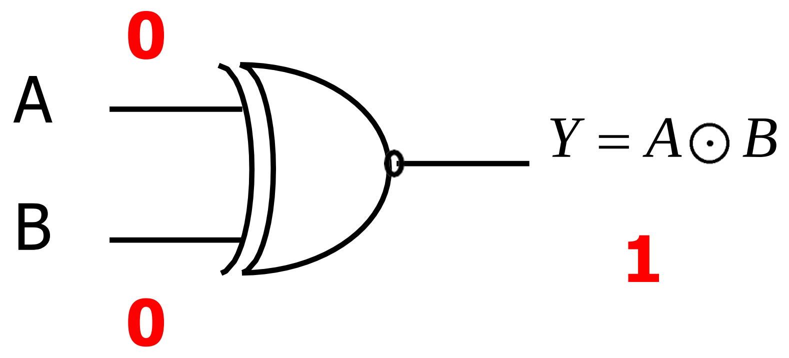
$$Y = A \odot B$$

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

Logic Symbol

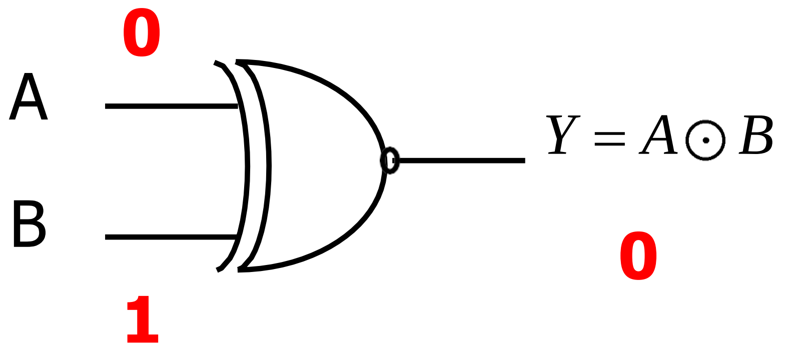
Logic Expression

Ex-NOR Gate



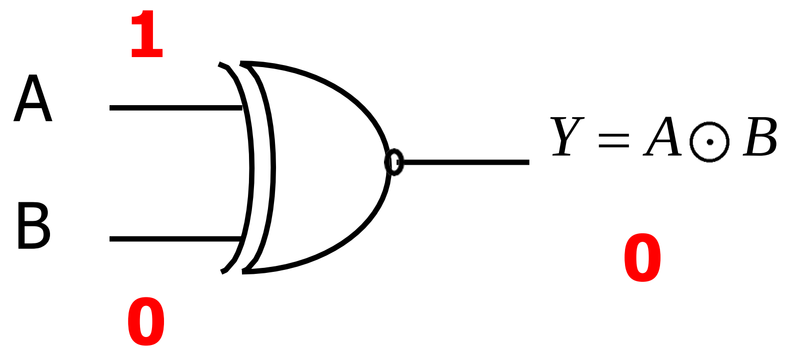
Inputs		Output
A	B	$Y = A \odot B$
0	0	1

Ex-NOR Gate



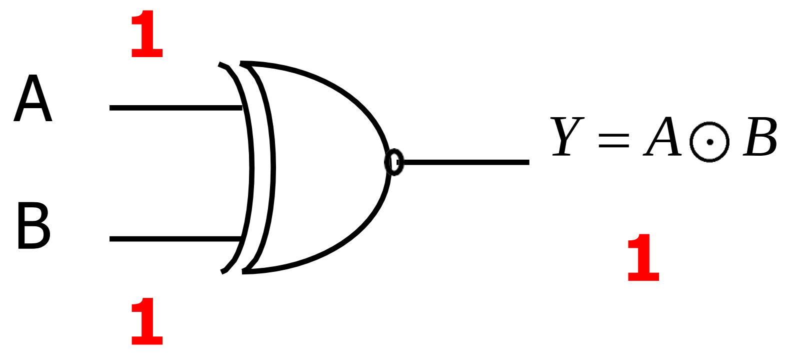
Inputs		Output
A	B	$Y = A \odot B$
0	0	1
0	1	0

Ex-NOR Gate



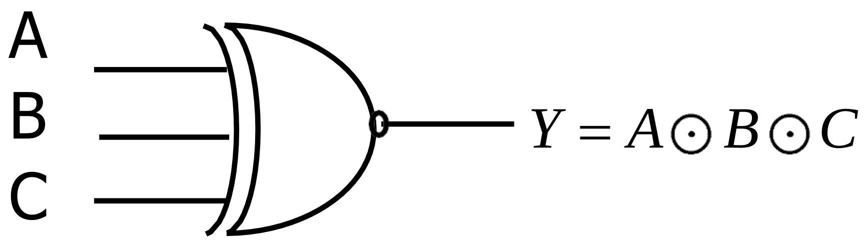
Inputs		Output
A	**B**	$Y = A \odot B$
0	0	1
0	1	0
1	**0**	**0**

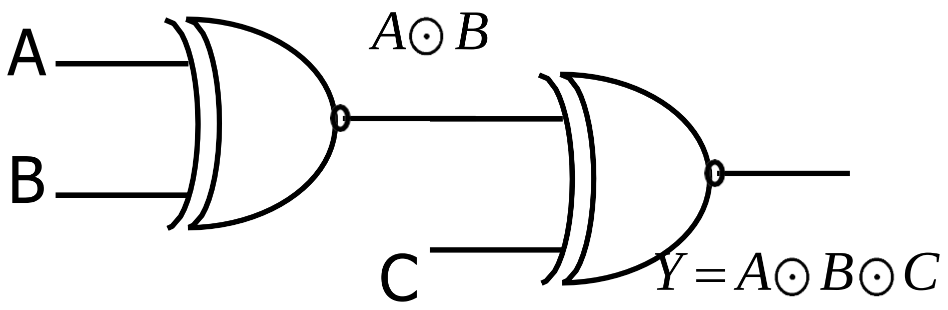
Ex-NOR Gate



Inputs		Output
A	B	$Y = A \odot B$
0	0	1
0	1	0
1	0	0
1	**1**	**1**

3 – Input Ex-NOR Gate





INPUT			OUTPUT
A	B	C	$Y = A \odot B \odot C$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Boolean Algebra

- ✓ Boolean Algebra is used to analyze and simplify the digital (Logic) circuit.

- ✓ Since it uses only the binary numbers i.e. 0 and 1 it is also called as “Binary Algebra” or “Logical Algebra”.

Boolean Algebra

- ✓ The rules of Boolean Algebra are different from those of the conventional algebra.
- ✓ It is invented by George Boole in the year 1854.

Boolean Algebra

- **Axioms**
  - ✓ Axioms or postulates of Boolean algebra are set of logical expressions that we accept without proof and upon which we can build a set of useful theorems.
  - ✓ Actually, axioms are nothing more than the definitions of the three basic logic operations that we have already discussed AND, OR and INVERT.

Boolean Algebra

- **Axioms**

AND Operation

Axiom 1:	$0 . 0 = 0$
Axiom 2:	$0 . 1 = 0$
Axiom 3:	$1 . 0 = 0$
Axiom 4:	$1 . 1 = 1$

Boolean Algebra

- **Axioms**

OR Operation

Axiom 5:	$0 + 0 = 0$
Axiom 6:	$0 + 1 = 1$
Axiom 7:	$1 + 0 = 1$
Axiom 8:	$1 + 1 = 1$

Boolean Algebra

- **Axioms**

NOT Operation

Axiom 9: $\overline{1} = 0$

Axiom 10: $\overline{0} = 1$

Boolean Algebra

- **Inversion Law (or Complementation Law)**
  - ✓ The term complement means to invert i.e. to change 0's to 1's and 1's to 0's.

Law 1: $\overline{1} = 0$

Law 2: $\overline{0} = 1$

Law 3: If A=0, then $\overline{A} = 1$

Law 4: If A=1, then $\overline{A} = 0$

Law 5: $\overline{\overline{A}} = A$ (Double Inversion Law)

Boolean Algebra

- **AND Laws**

Law 1:	$A \cdot 0 = 0$	Null Law
Law 2:	$A \cdot 1 = A$	Identity Law
Law 3:	$A \cdot A = A$	
Law 4:	$A.\bar{A} = 0$	

➢ OR Laws

Law 1:	$A + 0 = A$	Null Law
Law 2:	$A + 1 = 1$	Identity Law
Law 3:	$A + A = A$	
Law 4:	$A + \bar{A} = 1$	

Boolean Algebra

- **Commutative Laws**

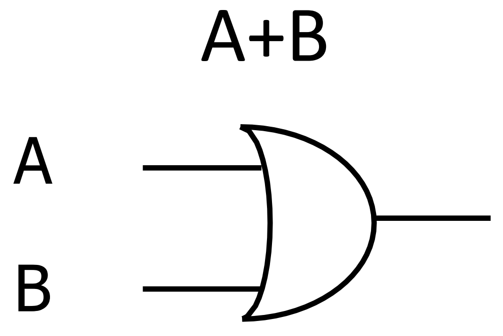
Law 1: $A+B = B+A$

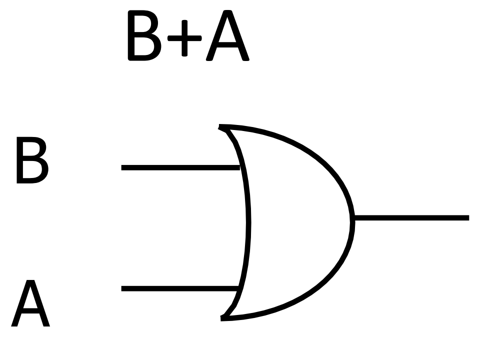
- ✓ This Law states that, A OR B is the same as B OR A i.e. the order in which the variables are ORed is immaterial.

- ✓ This means that it makes no difference which input of an OR gate is connected to A and which to B.

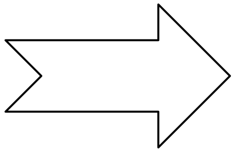
Boolean Algebra

Proof:





Inputs		Output
A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1



Inputs		Output
B	A	Y=B+A
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Algebra

- **Commutative Laws**

  ✓ This law can be extended to any number of variables. For example,

$$A+B+C = B+C+A = C+A+B = B+A+C$$

➢ Commutative Laws

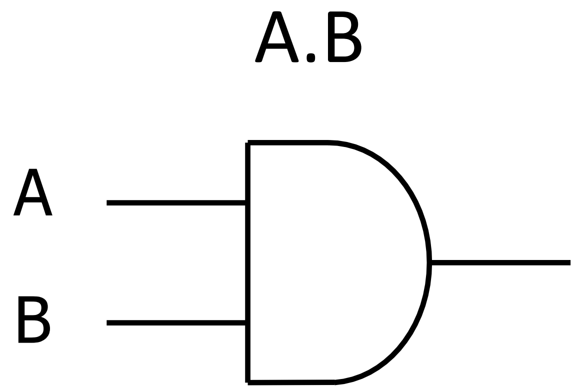
Law 2: $A.B = B.A$

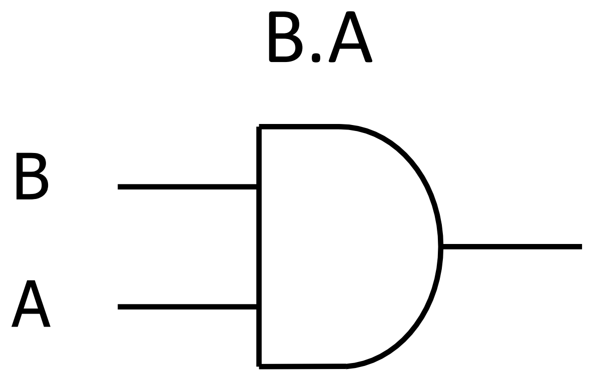
✓ This Law states that, A AND B is the same as B AND A i.e. the order in which the variables are ANDed is immaterial.

✓ This means that it makes no difference which input of an AND gate is connected to A and which to B.

Boolean Algebra

Proof:





Inputs		Output
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

$\Rightarrow$

Inputs		Output
B	A	Y=B.A
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Algebra

➢ Commutative Laws

✓ This law can be extended to any number of variables. For example,

A.B.C = B.C.A = C.A.B = B.A.C

Boolean Algebra

- **Associative Laws**

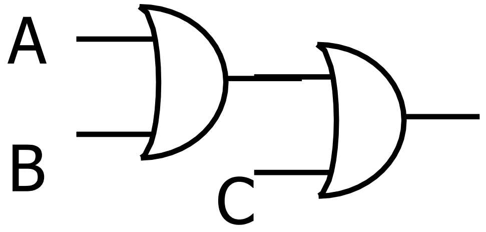
Law 1: $(A+B)+C = A+(B+C)$

- ✓ A OR B ORed with C is the same as A ORed with B OR C.
- ✓ This law states that the way the variables are grouped and ORed is immaterial.

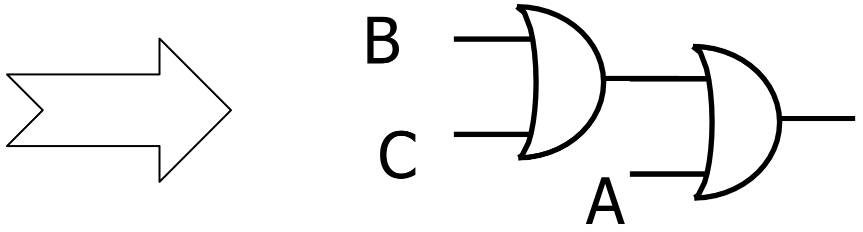
Boolean Algebra

Proof:

(A+B)+C



A+(B+C)



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

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

Boolean Algebra

- **Associative Laws**

  - ✓ This law can be extended to any number of variables. For example,

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

Boolean Algebra

- **Associative Laws**

Law 2: $(A.B).C = A.(B.C)$

✓ A AND B ANDed with C is the same as A ANDed with B AND C.

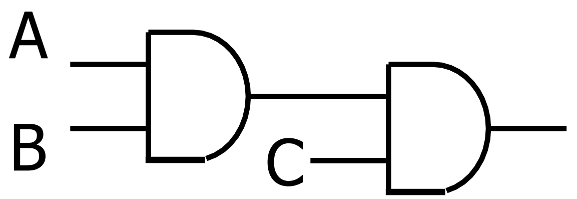
✓ This law states that the way the variables are grouped and ANDed is immaterial.

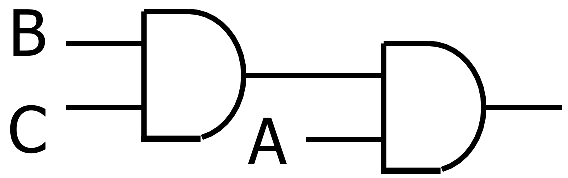
Boolean Algebra

Proof:

(A.B).C

A.(B.C)





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

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

Boolean Algebra

- **Associative Laws**

  - ✓ This law can be extended to any number of variables. For example,

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

Boolean Algebra

- **Distributive Laws**

Law 1: $A(B+C) = AB+AC$

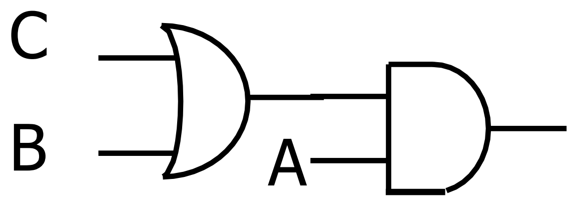
- ✓ This law states that ORing of several variables and ANDing the result with a single variable is equivalent to ANDing that single variable with each of the several variables and then ORing the products.

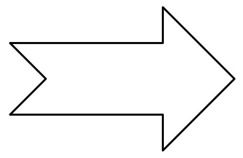
Boolean Algebra

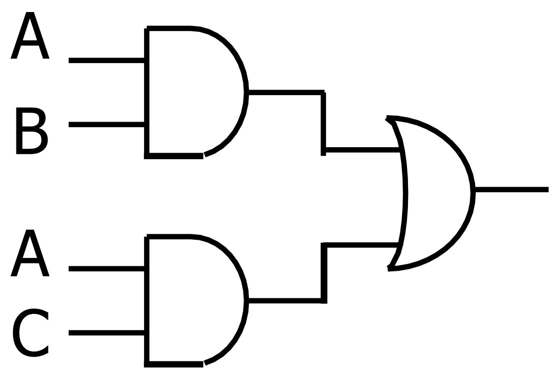
Proof:

A.(B+C)

AB+AC







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

A	B	C	AB	AC	AB+AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

➢ Distributive Laws

- ✓ This law can be extended to any number of variables. For example,

$$ABC(D+E) = ABCD + ABCE$$

$$AB(CD+EF) = ABCD + ABEF$$

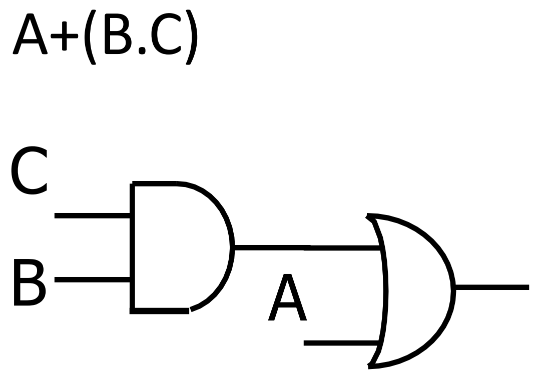
➢ Distributive Laws

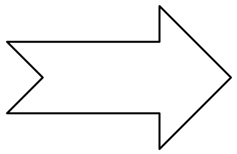
Law 2: $A+BC = (A+B).(A+C)$

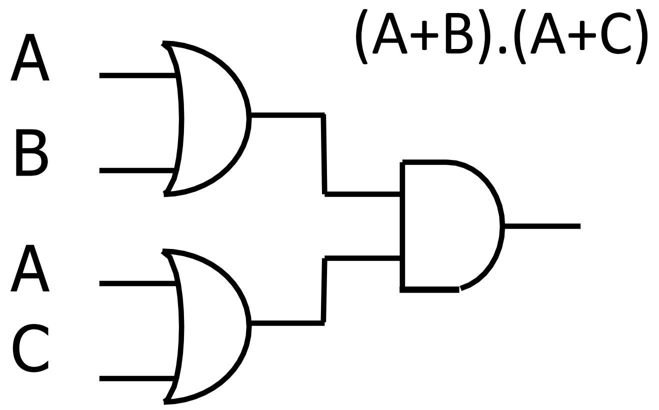
✓ This law states that ANDing of several variables and ORing the result with a single variable is equivalent to ORing that single variable with each of the several variables and then ANDing the products.

Boolean Algebra

Proof:







A	B	C	BC	A+BC
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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

Boolean Algebra

Redundant Literal Rule

Law 1: $A + \bar{A}B = A + B$

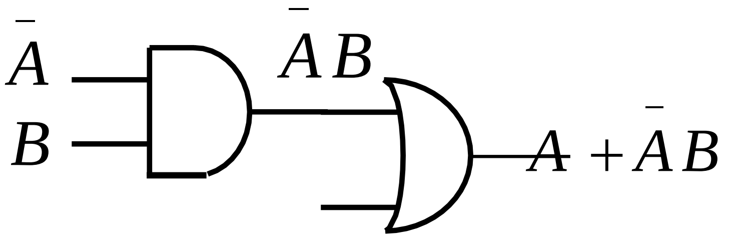
- ✓ This law states that ORing of variable with the AND of the complement of that variable with another variable, is equal to the ORing of the two variables

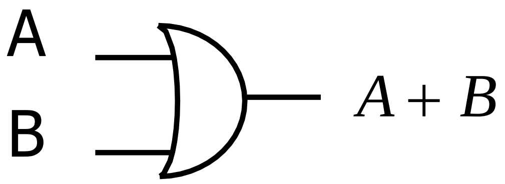
Boolean Algebra

Proof:

$A + \bar{A}B$

$A + B$





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

$\Rightarrow$

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

➢ **Redundant Literal Rule**

Law 2: $A(\bar{A} + B) = A.B$

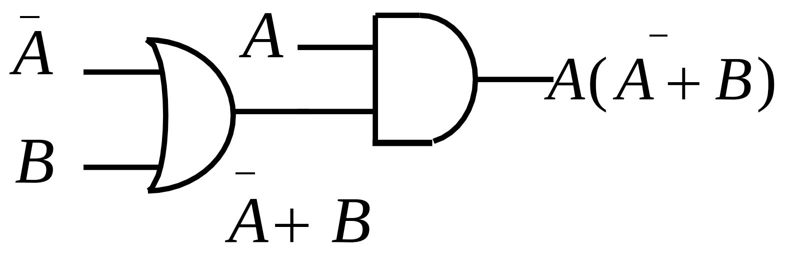
✓ This law states that ANDing of variable with the OR of the complement of that variable with another variable, is equal to the ANDing of the two variables

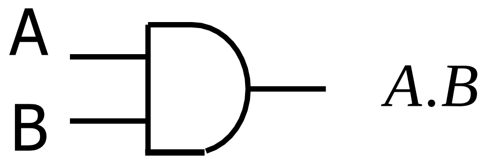
Boolean Algebra

Proof:

$A(\bar{A}+B)$

$A.B$





A	B	$\bar{A}+B$	$A(\bar{A}+B)$
0	0	1	0
0	1	1	0
1	0	0	0
1	1	1	1

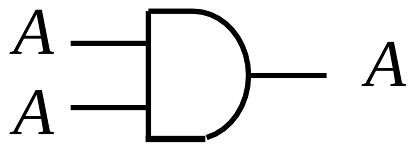
$\Rightarrow$

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

Boolean Algebra

- **Idempotence Laws**

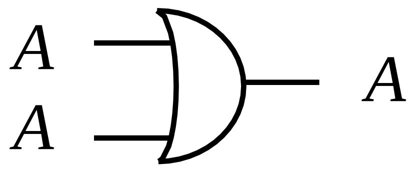
Law 1: $A.A = A$



- ✓ Idempotence means the same value
- ✓ If A=0, then A.A = 0.0 = 0 = A
- ✓ If A=1, then A.A = 1.1 = 1 = A
- ✓ This law states that ANDing of a variable with itself is equal to that variable only.

Boolean Algebra

- **Idempotence Laws**



Law 2: $A + A = A$

- ✓ Idempotence means the same value
- ✓ If A=0, then A+A = 0+0 = 0 = A
- ✓ If A=1, then A+A = 1+1 = 1 = A
- ✓ This law states that ORing of a variable with itself is equal to that variable only.

Boolean Algebra

- **Absorption Laws**

Law 1: $A + A.B = A$

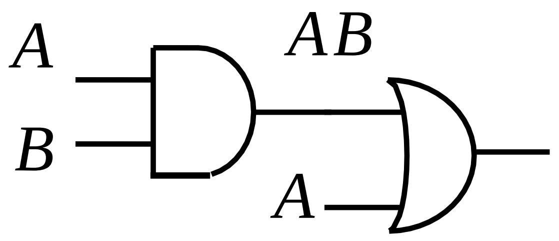
- ✓ This law states that ORing of a variable with AND of that variable and another variable is equal to that variable itself.
- ✓ Therefore,

A + A. Any Term = A

Boolean Algebra

Proof:

$A + A.B$ A



$A + AB = A$

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

➢ Absorption Laws

Law 2: $A(A+B) = A$

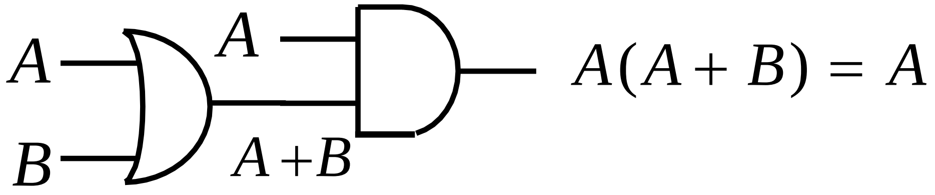
- ✓ This law states that ANDing of a variable with OR of that variable and another variable is equal to that variable itself.
- ✓ Therefore,

  A . (A + Any Term) = A

Boolean Algebra

Proof:

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



A	B	$A+B$	$A(A+B)$
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Boolean Algebra

➢ De-Morgan's Theorem

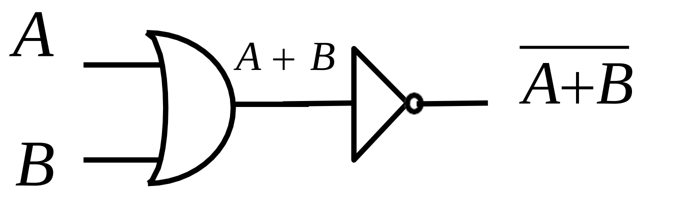
First Theorem: $\overline{A+B} = \bar{A}.\bar{B}$

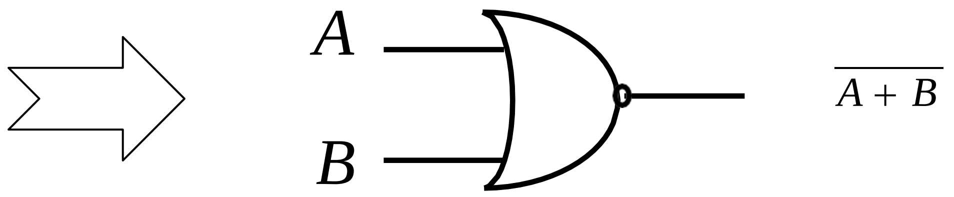
- ✓ This theorem states that the complement of a sum of variables is equal to the product of their individual complements.
- ✓ What it means is that the complement of two or more variables ORed together, is the same as the AND of the complements of each of the individual variables

Boolean Algebra

Proof: Logic Diagram

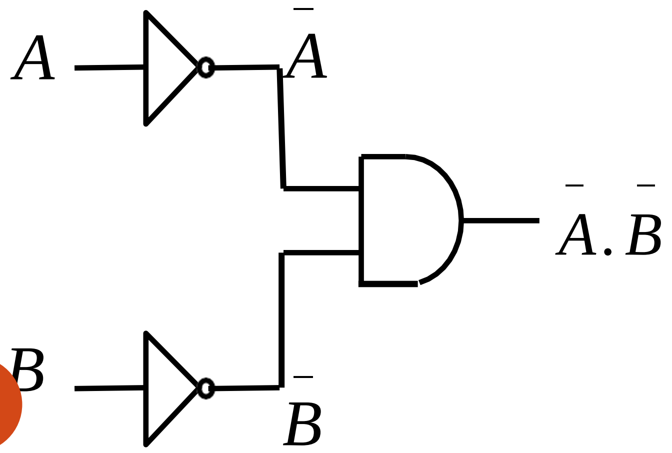
L.H.S.

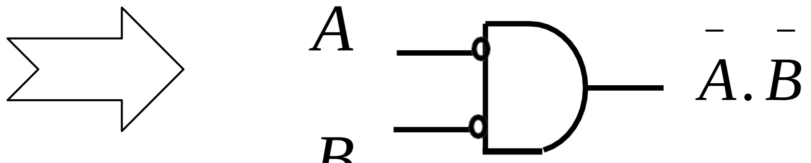




R.H.S.

NOR Gate





Bubbled AND Gate

Boolean Algebra

Proof: Logic Table

$\overline{A+B}$

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

$\Rightarrow$

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

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

➢ De-Morgan's Theorem

✓ This law can be extended to any number of variables. For example,

$$\overline{A+B+C+D+\ldots\ldots\ldots} = \bar{A}.\bar{B}.\bar{C}.\bar{D}\ldots\ldots\ldots$$

$$\overline{AB+CD+EFG+\ldots\ldots\ldots} = (\overline{AB}).(\overline{CD}).(\overline{EFG})\ldots\ldots\ldots$$

➢ De-Morgan's Theorem

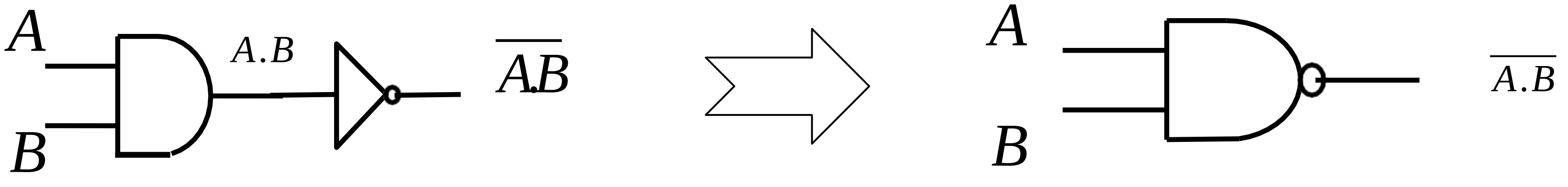
Second Theorem: $\overline{A.B} = \bar{A} + \bar{B}$

- ✓ This theorem states that the complement of a product of variables is equal to the sum of their individual complements.
- ✓ What it means is that the complement of two or more variables ANDed together, is the same as the OR of the complements of each of the individual variables

Boolean Algebra

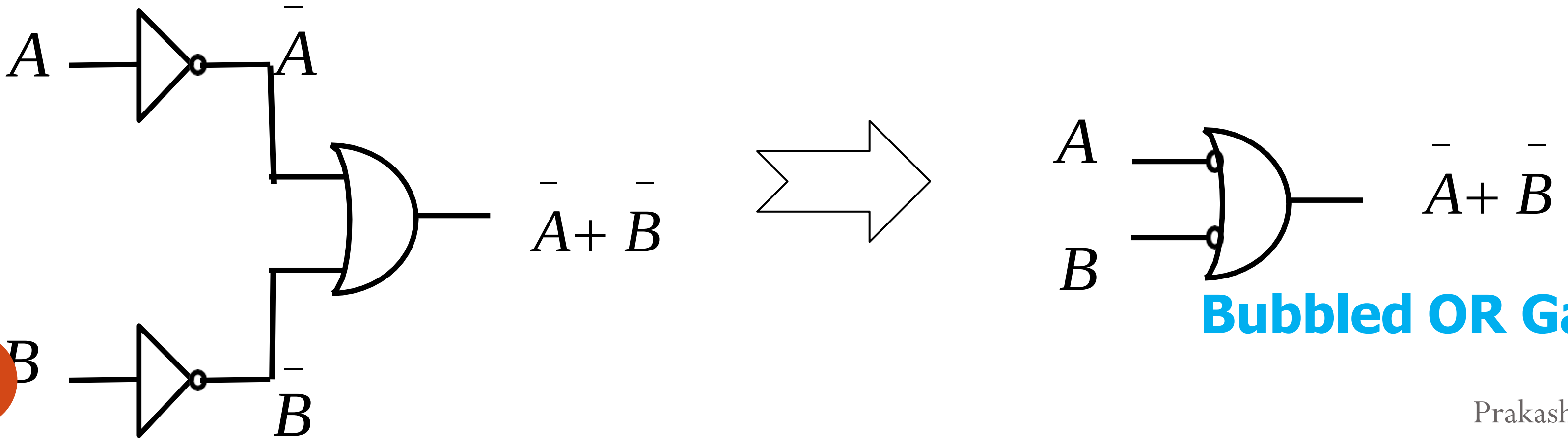
Proof: Logic Diagram

L.H.S.



R.H.S.

NAND Gate



Bubbled OR Gate

Boolean Algebra

Proof:

$$\overline{A.B} \Rightarrow \bar{A}+\bar{B}$$

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

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

De-Morgan’s Theorem

- ✓ This law can be extended to any number of variables. For example,

$$\overline{A.B.C.D.........} = \bar{A} + \bar{B} + \bar{C} + \bar{D}...........$$

$$\overline{(AB)(CD)(EFG).........} = \overline{AB} + \overline{CD} + \overline{EFG} + ...$$

Duality

- ✓ Duality represents relation between expressions in positive logic system and expression in negative logic system.

Duality

- ✓ The distinction between positive and negative logic system is important.
- ✓ An OR gate in positive logic system becomes an AND gate in negative logic system and vice versa.
- ✓ Positive & negative logics thus give rise to a basic duality in all Boolean identities.

Duality

- ✓ When changing from one logic system to another 0 becomes 1 and 1 becomes 0.
- ✓ Furthermore, an AND gate becomes an OR gate and an OR gate becomes AND gate.

Duality

- ✓ Given Boolean identity, we can produce a dual identity by changing all '+' signs to '.' signs, all '.' signs to '+' signs and complementing all 0's and 1's.
- ✓ The variables are not complemented in this process.

Examples of Dual Identities

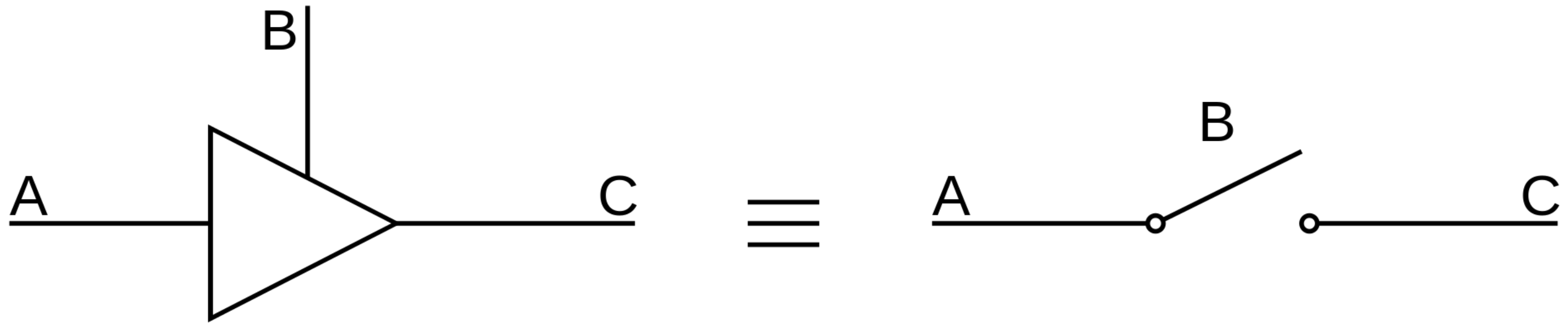
Sr. No.	Given Expression	Dual
1	$\overline{0} = 1$	$\overline{1} = 0$
2	$0.1 = 0$	$1 + 0 = 1$
3	$0.0 = 0$	$1 + 1 = 1$
4	$1.1 = 1$	$0 + 0 = 0$
5	$A.0 = 0$	$A + 1 = 1$
6	$A.1 = A$	$A + 0 = A$

Examples of Dual Identities

Sr. No.	Given Expression	Dual
7	$A.B = B.A$	$A + B = B + A$
8	$A.(B.C) = (A.B).C$	$A + (B + C) = (A + B) + C$
9	$A.(B + C) = A.B + A.C$	$A + BC = (A + B)(A + C)$
10	$A.(A + B) = A$	$A + AB = A$
11	$A.(A.B) = A.B$	$A + A + B = A + B$
12	$\overline{A.B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}.\bar{B}$
13	$(A + B)(\bar{A} + C) = (A + B)(\bar{A} + C)$	$AB + \bar{A}C = AB + \bar{A}C$

TRI-STATE LOGIC

- In digital electronics **three-state**, **tri-state**, or **3-state** logic allows an output port to assume a high impedance state, effectively removing the output from the circuit, in addition to the 0 and 1 logic levels.
- Three-state outputs are implemented in many registers, bus drivers and flip-flops in the 7400 and 4000 series as well as in other types, but also internally in many integrated circuits.



A tristate buffer can be thought of as a switch. If *B* is on, the switch is closed. If B is off, the switch is open

Example 1

Reduce the following Boolean Expression using Boolean Laws:

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

Example 1

Reduce the following Boolean Expression using Boolean Laws:

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

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

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

$$=A.(\overline{B}+B)+\overline{A}(B+B) \qquad (\quad B+B=1)$$

$$=A+A \qquad (\quad A+\overline{A}=1)$$

$$=1$$

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

Example 2

Reduce the following Boolean Expression using Boolean Laws:

$$A\overline{B}C + \overline{A}BC + ABC$$

Example 2

Reduce the following Boolean Expression using Boolean Laws:

$$A\overline{B}C + \overline{A}BC + ABC$$

$$= A\overline{B}C + \overline{A}BC + ABC$$

$$= A\overline{B}C + BC(\overline{A} + A)$$

$$= A\overline{B}C + BC \qquad (\quad A + \overline{A} = 1)$$

$$= C(A\overline{B} + B)$$

$$= C(B + A)(\overline{B} + B) \qquad (\quad \text{Distributive Law})$$

$$= C(B + A) \qquad (\quad \overline{B} + B = 1)$$

$$= AC + BC$$

Example 3

Realize $Y=AB+AC$ using one OR gate and one AND gate

Example 3

Realize Y=AB+AC using one OR gate and one AND gate

$$Y = AB + AC$$

A.B is one product term
Hence requires 1 AND gate

A.C is one product term
Hence requires 1 AND gate

A.C & A.B is one sum term
Hence requires 1 OR gate

Hence to implement Y=AB+AC equation we require 2 AND gates and 1 OR gate

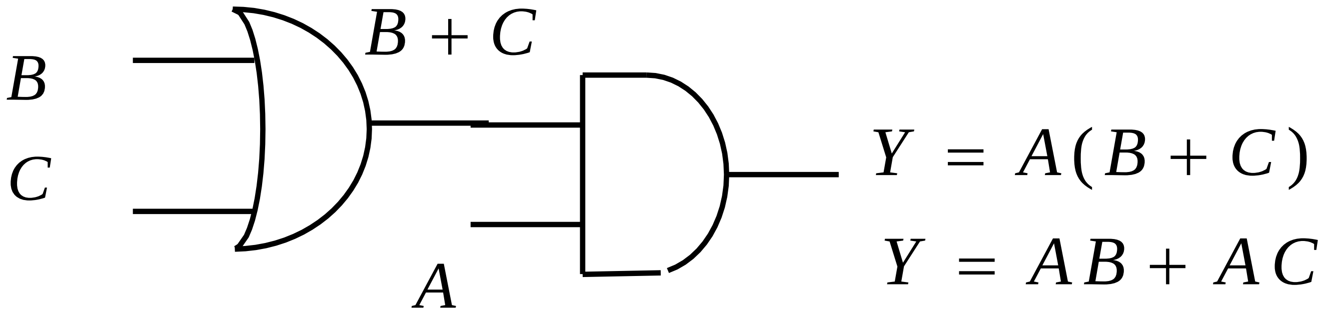
But we have to use only 1 AND gate and 1 OR gate

Hence simplification is necessary

Example 3 Continue........

$$Y = AB + AC$$

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



Example 4

Prove that:

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

Example 4

Prove that:

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

$$\begin{aligned}
\text{L.H.S} &= A + \overline{A}B \\
&= A(1) + \overline{A}B \\
&= A(1 + B) + \overline{A}B \\
&= A + AB + \overline{A}B \\
&= A + B(A + \overline{A}) \\
&= A + B \qquad\qquad (\quad A + \overline{A} = 1)
\end{aligned}$$

$$\text{L.H.S} = \text{R.H.S}$$

Example 5

Prove that:

$$(A+B)(A+\overline{B})=A$$

Example 5

Prove that:

$$(A+B)(A+\overline{B}) = A$$

$$\text{L.H.S} = (A+B)(A+\overline{B})$$

$$= AA + A\overline{B} + AB + B\overline{B}$$

$$= A + A\overline{B} + AB + 0 \qquad (\quad \text{AA=A, B}\overline{\text{B}}\text{=0})$$

$$= A + A(\overline{B} + B)$$

$$= A + A \qquad (\quad \overline{B} + B = 1)$$

$$= A \qquad (\quad A + A = A)$$

$$\text{L.H.S} = \text{R.H.S}$$

Example 6

With the help of Boolean Laws, Prove that:

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

Example 6

With the help of Boolean Laws, Prove that:

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

$$\text{L.H.S.}=(A+\overline{B}+AB)(A+B).\overline{A}\overline{B}$$

$$(A+\overline{B}+AB)(\mathrm{A}\overline{\mathrm{A}}\overline{\mathrm{B}}+\mathrm{A}\mathrm{B}\overline{\mathrm{B}})$$

$$=(A+\overline{B}+AB).(0) \qquad (\quad \mathrm{A}.\overline{\mathrm{A}}=0,\ \mathrm{B}.\overline{\mathrm{B}}=0)$$

$$=0$$

$$\text{L.H.S}=\text{R.H.S}$$

Example 7

With the help of Boolean Laws, Prove that:

$$AB + \overline{A}B + \overline{A}\,\overline{B} = \overline{A} + B$$

Example 7

With the help of Boolean Laws, Prove that:

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

$$\text{L.H.S.}=AB+\overline{A}B+\overline{A}\overline{B}$$

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

$$=\overline{A}(B+\overline{B})+AB$$

$$=\overline{A}+AB \qquad (\quad B+\overline{B}=0)$$

$$=(A+\overline{A})(\overline{A}+B) \qquad (\quad \overline{A}+AB=(A+\overline{A})(\overline{A}+B))$$

$$=1.(\overline{A}+B) \qquad (\quad A+\overline{A}=1)$$

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

$$\text{L.H.S}=\text{R.H.S}$$

Example 8

Simplify; $F = XY + XYZ + XYZ + X\overline{Z}Y$

Example 8

Simplify; $F = XY + XYZ + XYZ + X\overline{Z}Y$

$$F = XY + XYZ + XYZ + X\overline{Z}Y$$

$$= XY + XYZ + X\overline{Z}Y \qquad (\quad XYZ+XYZ=XYZ)$$

$$= XY(1 + Z + \overline{Z})$$

$$= XY \qquad (\quad 1 + Z + \overline{Z} = 1)$$

$$= XY$$

Example 9

Prove that;

$$AB + ABC + A\overline{B} = A$$

Example 9

Prove that; $AB + ABC + A\overline{B} = A$

$$L.H.S. = AB + ABC + A\overline{B}$$

$$= AB(1 + C) + A\overline{B}$$

$$= AB + A\overline{B} \qquad (\quad 1 + C = 1)$$

$$= A(B + \overline{B})$$

$$= A \qquad (\quad B + \overline{B} = 1)$$

$$\text{L.H.S} = \text{R.H.S}$$

Logic Families

IC Chip Manufacturing Process

Logic Families

- ✓ Small Scale Integration (SSI)
- ✓ Medium Scale Integration (MSI)
- ✓ Large Scale Integration (LSI)
- ✓ Very Large Scale Integration (VLSI)
- ✓ Ultra Large Scale Integration (ULSI)
- ✓ Giant Scale Integration (GSI)

Logic Families

- Gate/transistor ratio is roughly

– SSI	< 12 gates/chip
– MSI	< 100 gates/chip
– LSI	…1K gates/chip
– VLSI	…10K gates/chip
– ULSI	…100K gates/chip
– GSI	…1Meg gates/chip

Moore’s Law

- ✓ A prediction made by Moore (a co-founder of Intel) in 1965: “… a number of transistors to double every 2 years.”