Logic Gates & Logic Families

- ✓ **Logic Gates:** Symbol, diode/transistor switch circuit and logical expression, truth table of basic gates (AND, OR, NOT), Universal gates (NAND, NOR) and special purpose gates (Ex-OR, Ex-NOR), Tristate Logic.

- ✓ **Boolean Algebra:** Laws of Boolean algebra, Duality Theorem, De-Morgan's Theorem

Logic Gates & Logic Families

Logic Gates & Logic Families

- ✓ **Logic Gates: Symbol, diode/transistor switch circuit and logical expression, truth table of basic gates (AND, OR, NOT),** Universal gates (NAND, NOR) and special purpose gates (Ex-OR, Ex-NOR), Tristate Logic.
- ✓ **Boolean Algebra:** Laws of Boolean algebra, Duality Theorem, De-Morgan's Theorem
- ✓ **Logic Families:** Characteristics of Logic families: Noise Margin, Power Dissipation, Figure of merit, Fan in and Fan out, Speed of operation, Comparison TTL, CMOS, Types of TTL NAND gate.

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.

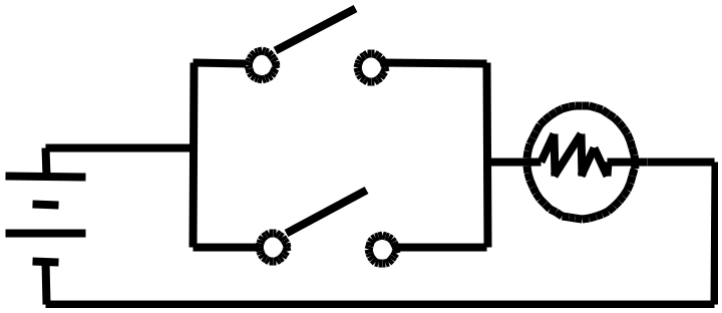
HIGH	LOW
True	False
ON	OFF
1	0

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.

Basics…..

- **Logic**
  - ✓ **Positive Logic**

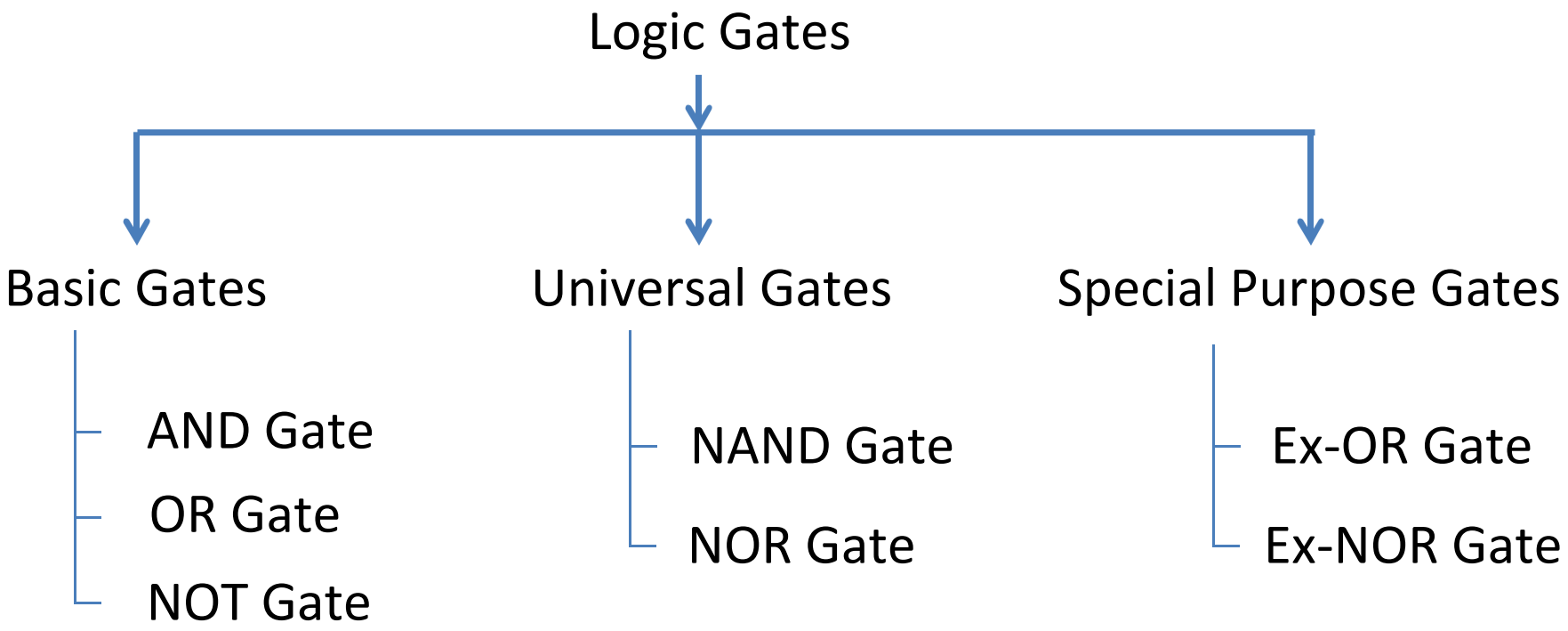
    Logic 0 (LOW)=0V

    Logic 1 (HIGH)=+5V
  - ✓ **Negative Logic**

    Logic 0 (LOW)=+5V

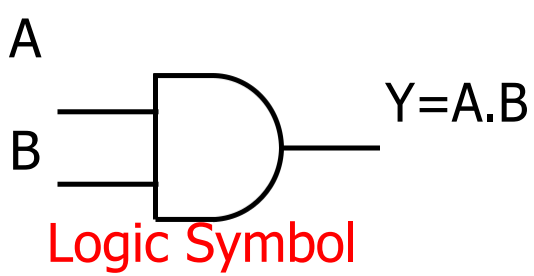
    Logic 1 (HIGH)=0V

Logic Gates


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

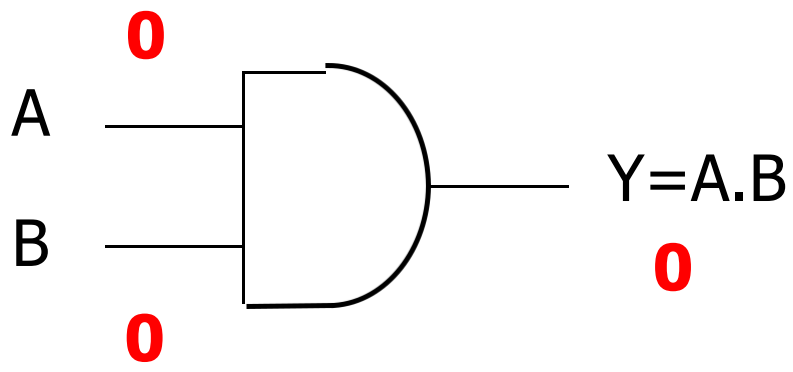
AND Gate

- ✓ 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.



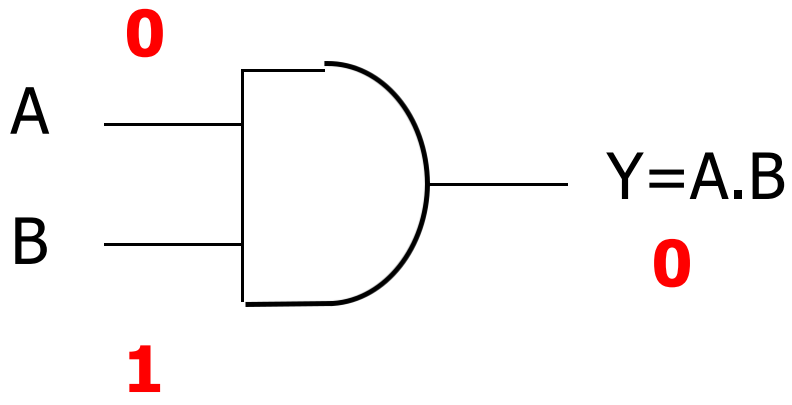
Logic Symbol

AND Gate



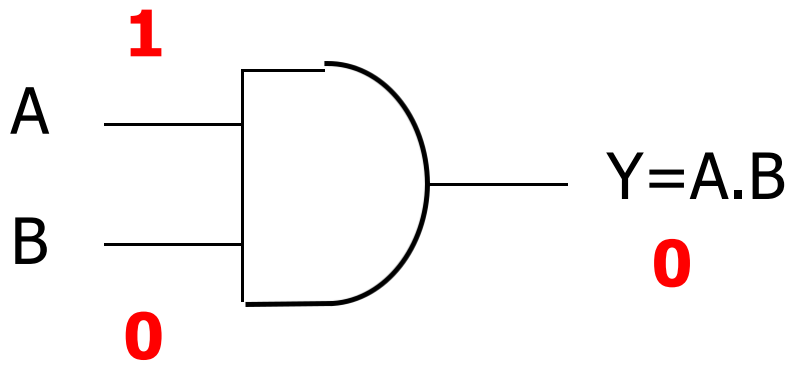
Inputs		Output
A	B	Y=A.B
0	0	0

AND Gate



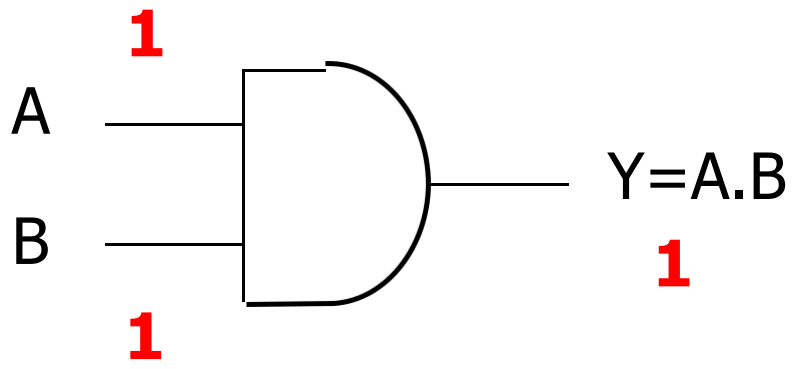
Inputs		Output
A	B	Y=A.B
0	0	0
0	**1**	**0**

AND Gate



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

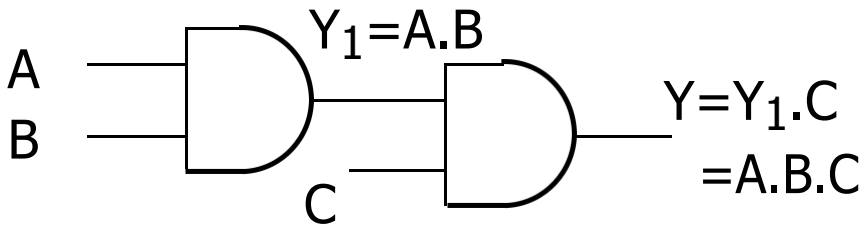
AND Gate



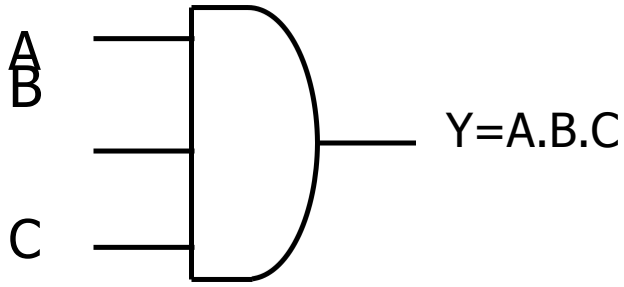
Inputs		Output
A	**B**	**Y=A.B**
0	0	0
0	1	0
1	0	0
1	**1**	**1**

3 - Input AND Gate

3 – Input AND Gate using 2 – Input AND Gate



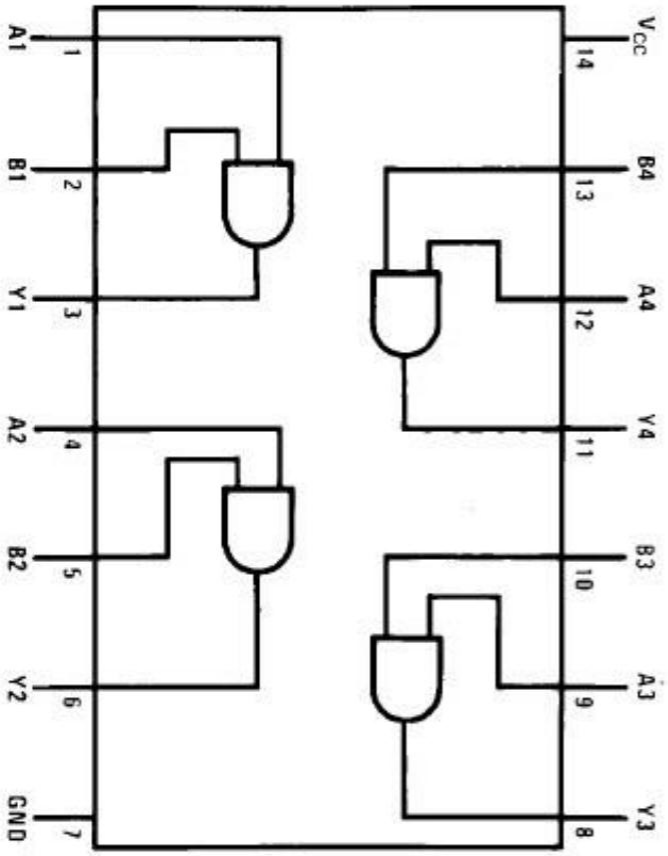
Symbol : 3 – Input AND Gate



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

AND Gate IC – IC 7408 (Quad 2 I/P AND Gate)



Pin Configuration of IC 7408

✓ This device contains four independent gates each of which performs the logic AND function.

$$Y = AB$$

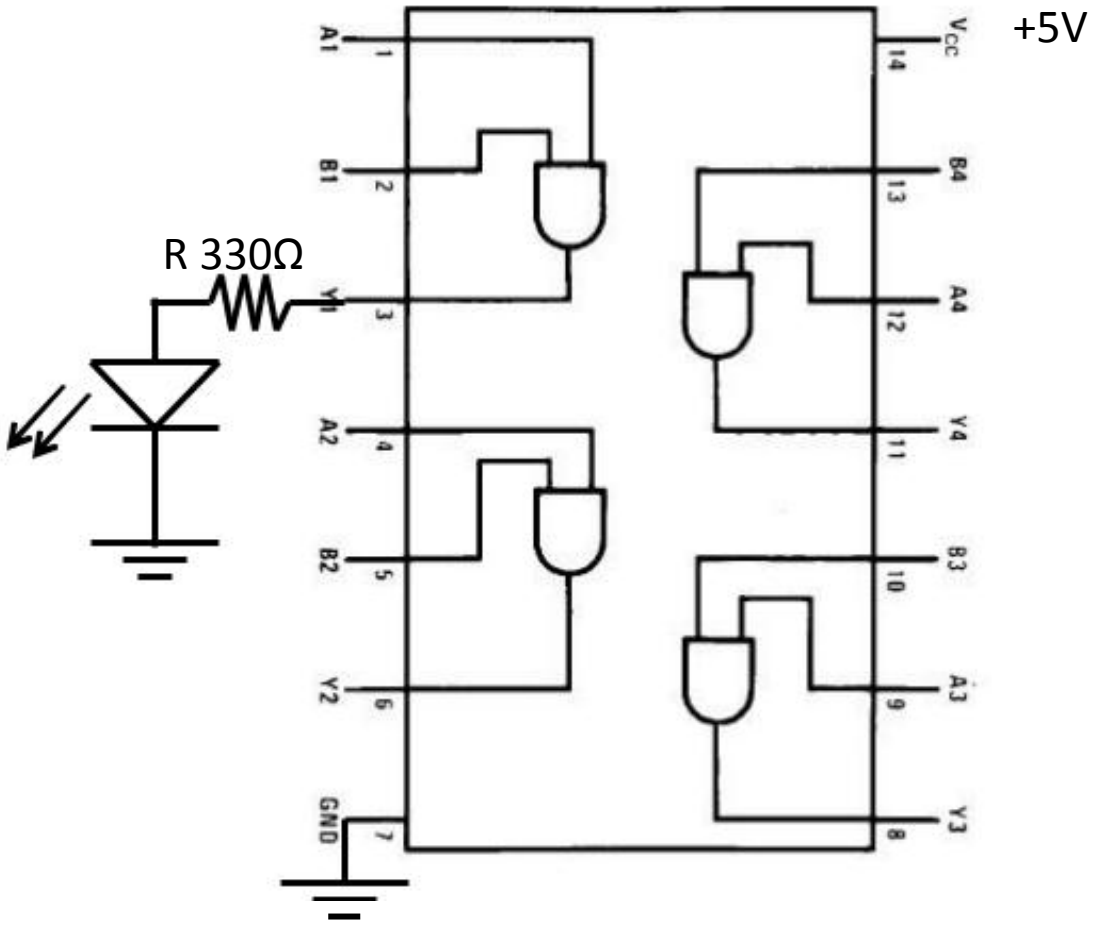
Inputs		Output
A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

H = High Logic Level

L = Low Logic Level

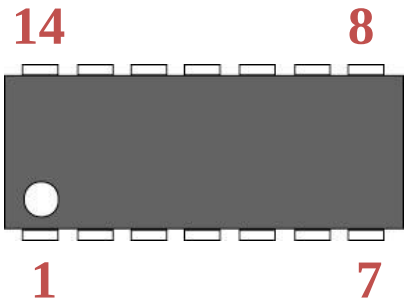
Function Table of IC 7408

Verification of AND Gate IC 7408

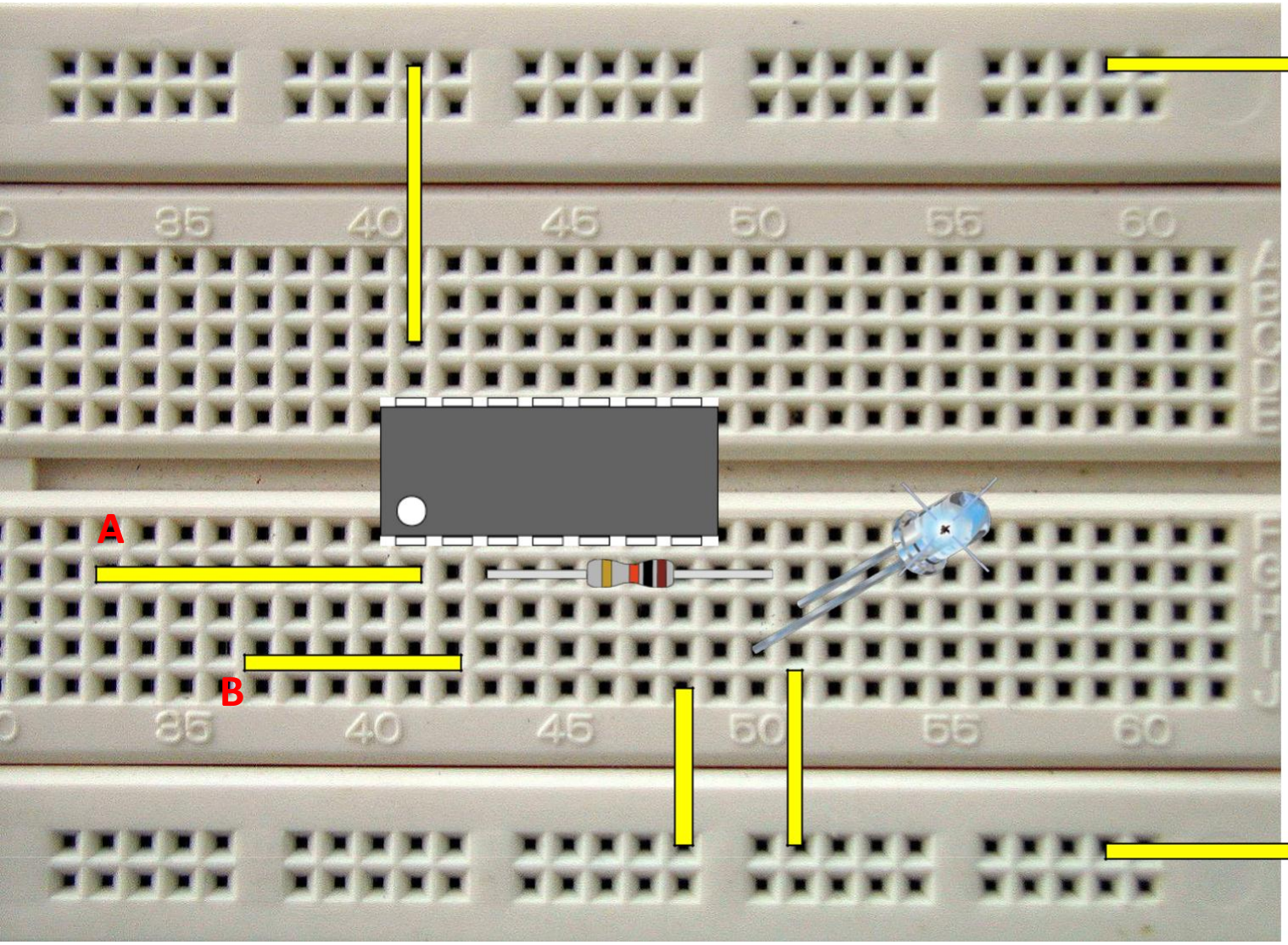


Verification of AND Gate IC 7408

Double check the pin numbers when working with ICs.



Verification of AND Gate IC 7408

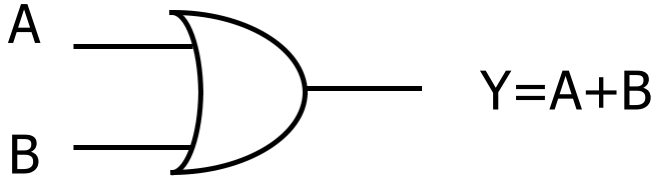


V_{CC}

GND

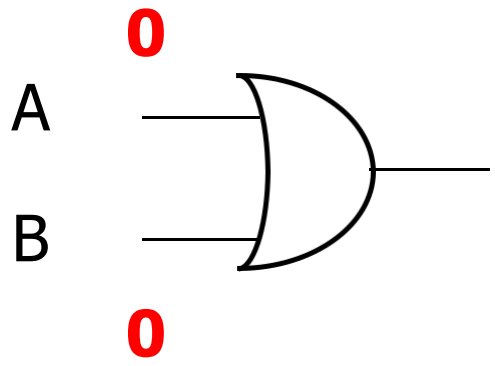
OR Gate

- ✓ 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

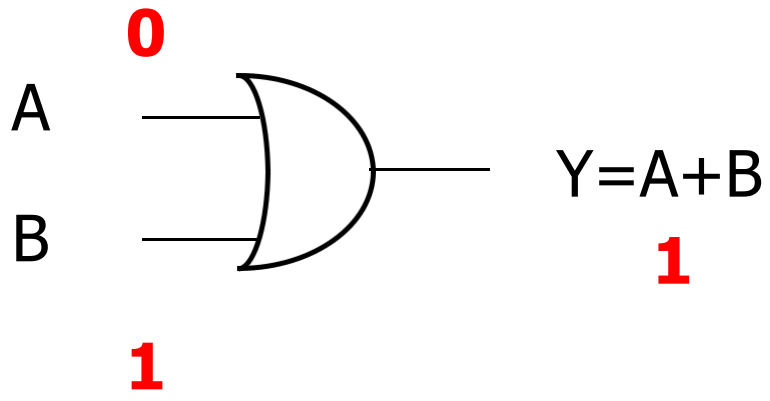


Y=A+B

0

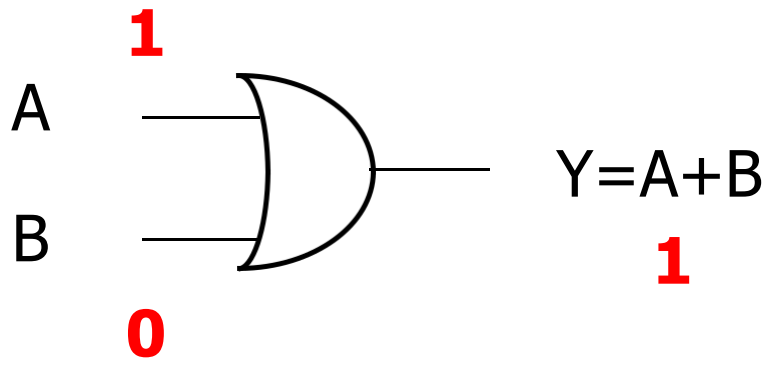
Inputs		Output
A	B	Y=A+B
0	0	0

OR Gate



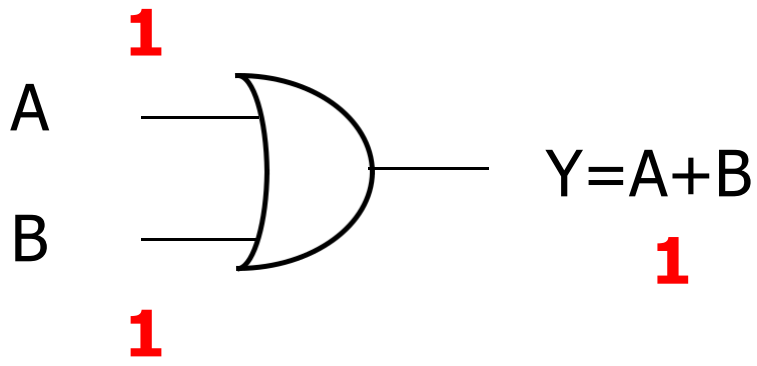
Inputs		Output
A	B	Y=A+B
0	0	0
0	**1**	**1**

OR Gate



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

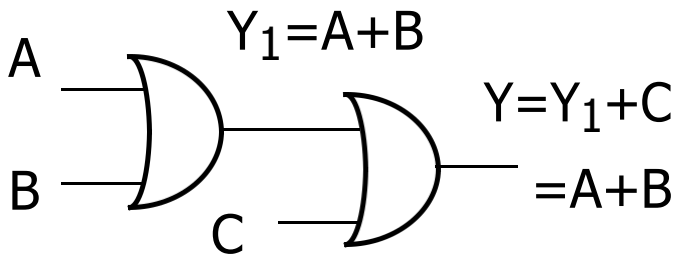
OR Gate



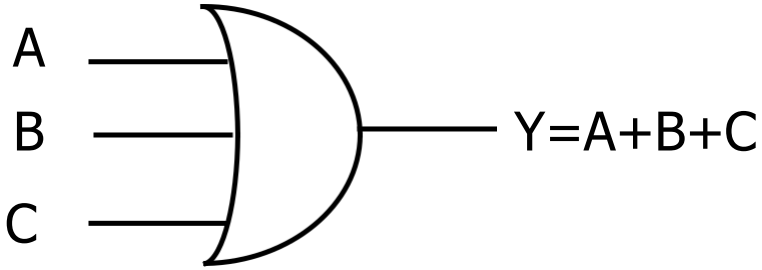
Inputs		Output
A	**B**	**Y=A+B**
0	0	0
0	1	1
1	0	1
1	**1**	**1**

Three Input OR Gate

3 – Input OR Gate using 2 – Input OR Gate



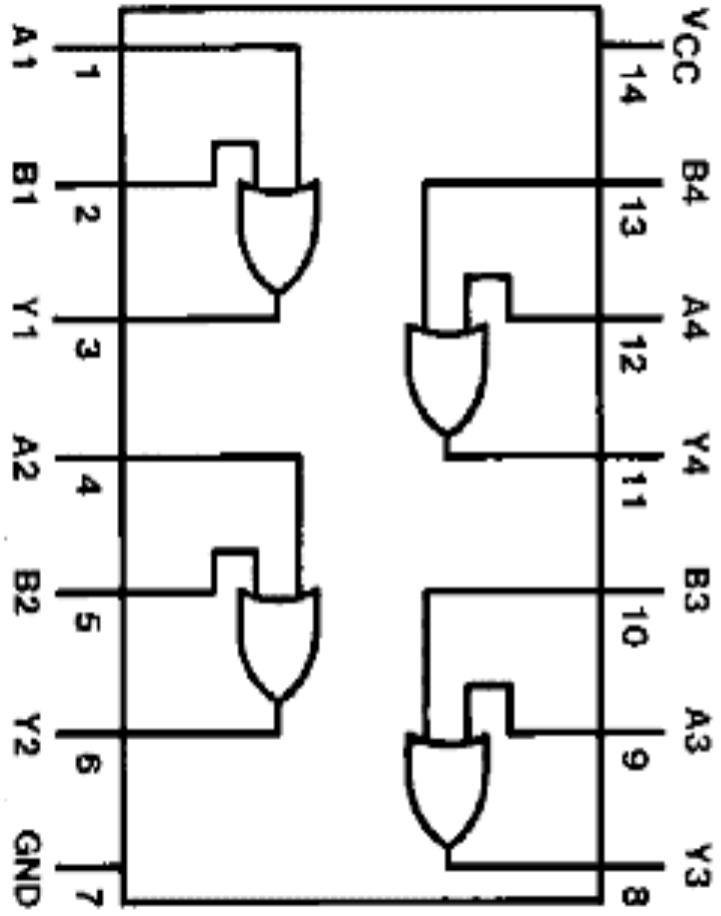
Symbol : 3 – Input OR Gate



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

OR Gate IC – IC 7432 (Quad 2 I/P OR Gate)



Pin Configuration of IC 7432

- ✓ This device contains four independent gates each of which performs the logic OR function

$$Y = A + B$$

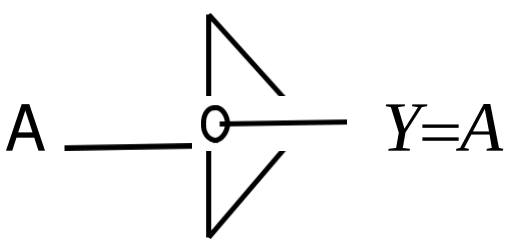
Inputs		Output
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	H

H = High Logic Level
L = Low Logic Level

Function Table of IC 7432

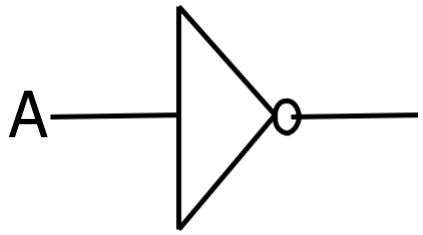
NOT Gate (Inverter)

- ✓ 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



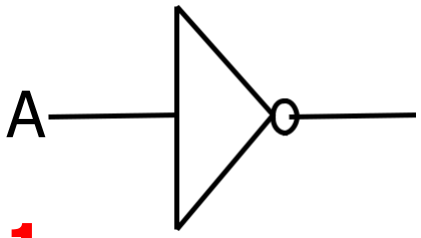
$Y = \bar{A}$

0

1

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

NOT Gate



$Y = \bar{A}$

1

0

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

Unit II – Logic Gates & Logic Families

- ✓ **Logic Gates: Symbol, diode/transistor switch circuit and logical expression, truth table of** basic gates (AND, OR, NOT), **Universal gates (NAND, NOR)** and special purpose gates (Ex-OR, Ex-NOR), Tristate Logic.

- ✓ **Boolean Algebra:** Laws of Boolean algebra, Duality Theorem, De-Morgan's Theorem

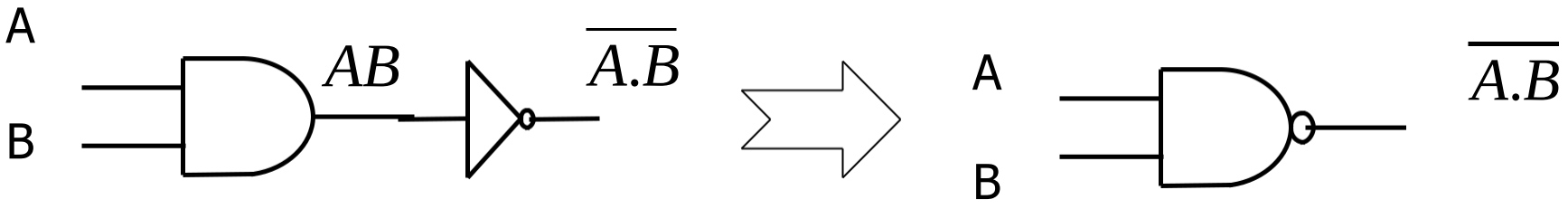
- ✓ **Logic Families:** Characteristics of Logic families: Noise Margin, Power Dissipation, Figure of merit, Fan in and Fan out, Speed of operation, Comparison TTL, CMOS, Types of TTL NAND gate.

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.

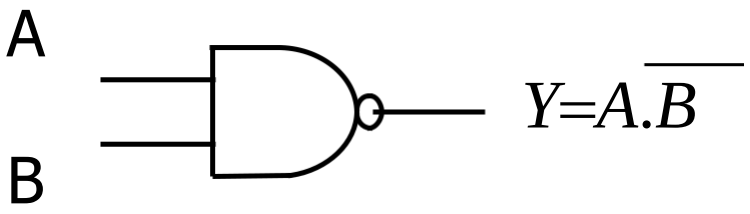


AND GateNOT Gate NAND Gate

NAND Gate

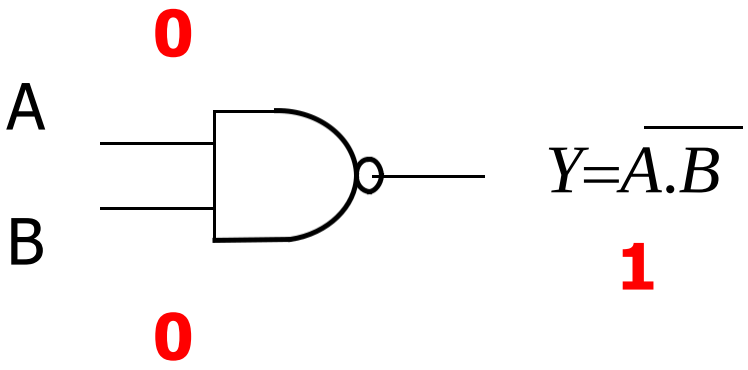
- ✓ The output is logic 0 level, only when all the inputs are logic 1 level.

- ✓ For any other combination of inputs, the output is a logic 1 level.



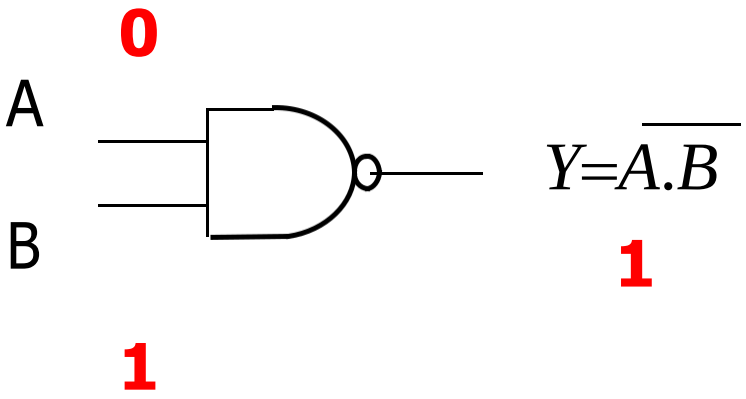
Logic Symbol

NAND Gate



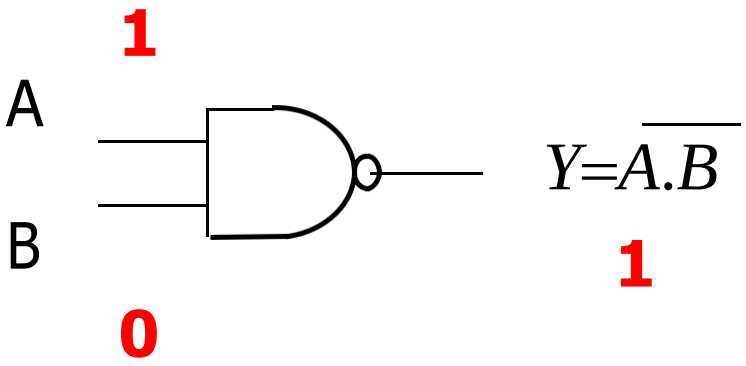
Inputs		Output
A	B	$Y = \overline{A.B}$
0	0	1

NAND Gate



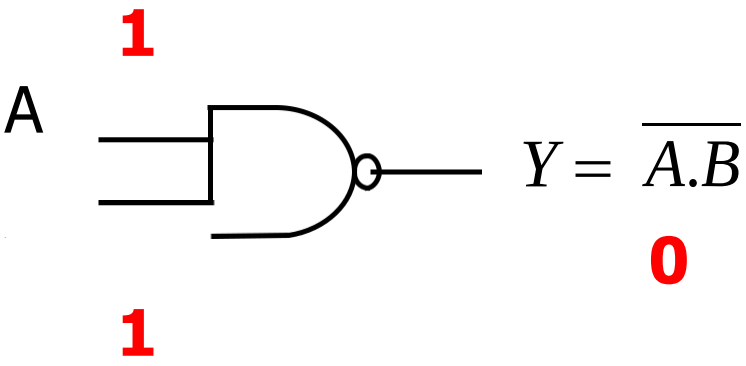
Inputs		Output
A	B	$Y=\overline{A.B}$
0	0	1
0	**1**	**1**

NAND Gate



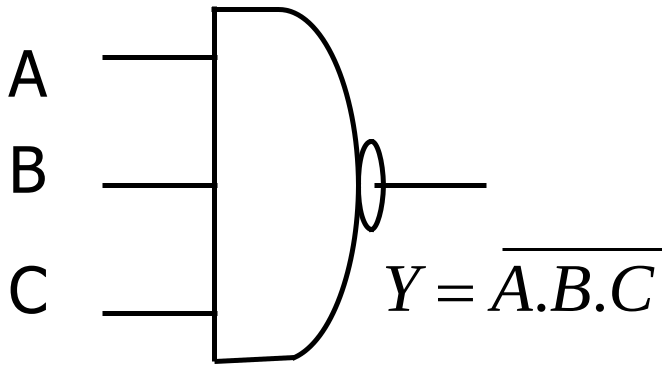
Inputs		Output
A	B	$Y=\overline{A.B}$
0	0	1
0	1	1
1	**0**	**1**

NAND Gate



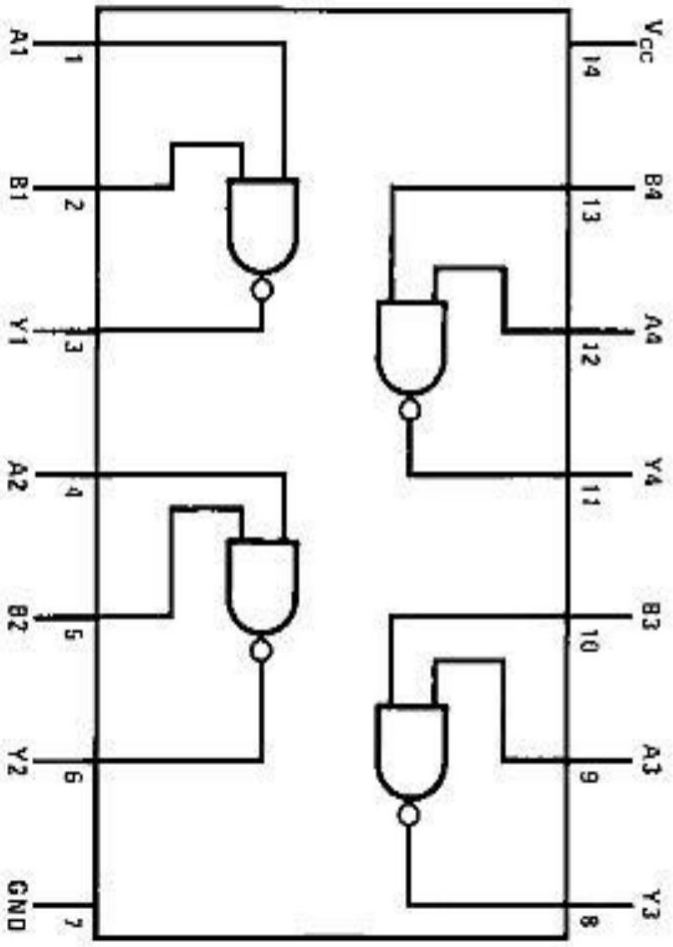
Inputs		Output
A	B	$Y=\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	**1**	**0**

3 - Input NAND Gate



INPUT			OUTPUT
A	**B**	**C**	**Y**
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	0

NAND Gate IC – IC 7400 (Quad 2 I/P NAND Gate)



Pin Configuration of IC 7400

✓ This device contains four independent gates each of which performs the logic NAND function.

$$Y = \overline{AB}$$

Inputs		Output
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

H = HIGH Logic Level

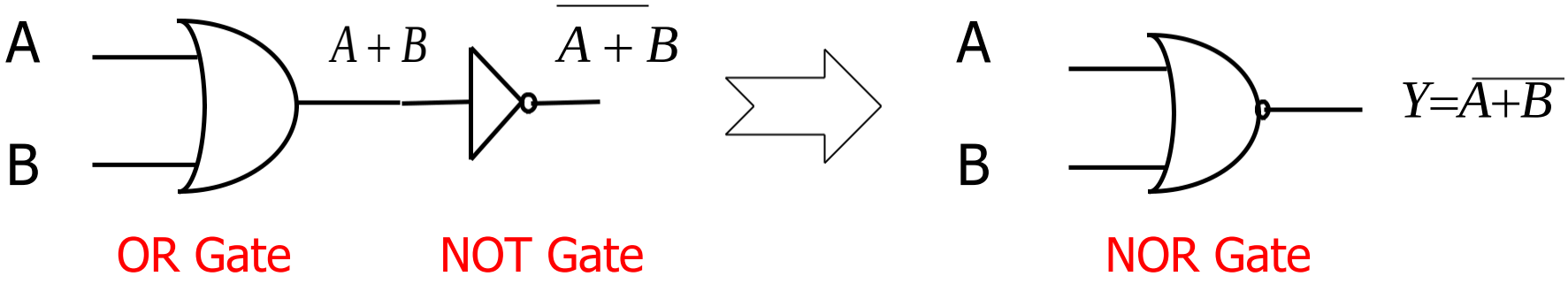
L = LOW Logic Level

Function Table of IC 7400

NOR Gate

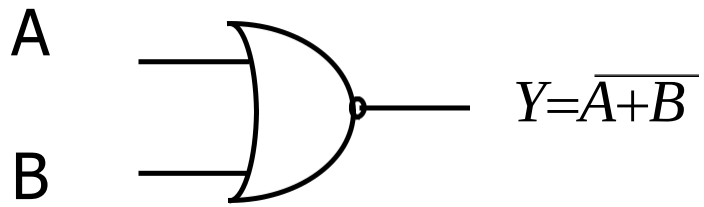
- NOR means NOT OR i.e. OR output is inverted.

- So NOR gate is a combination of an OR gate and a NOT gate.



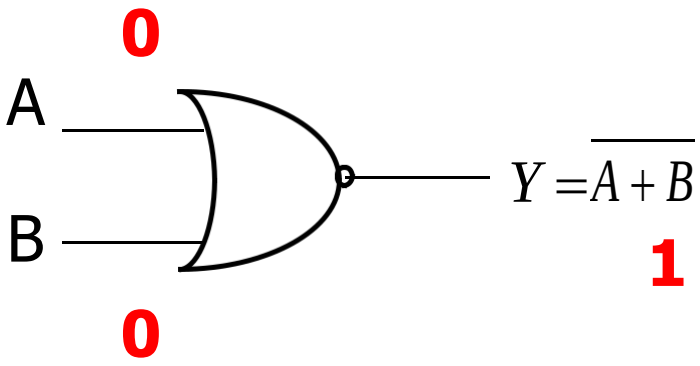
NOR Gate

- ✓ 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.



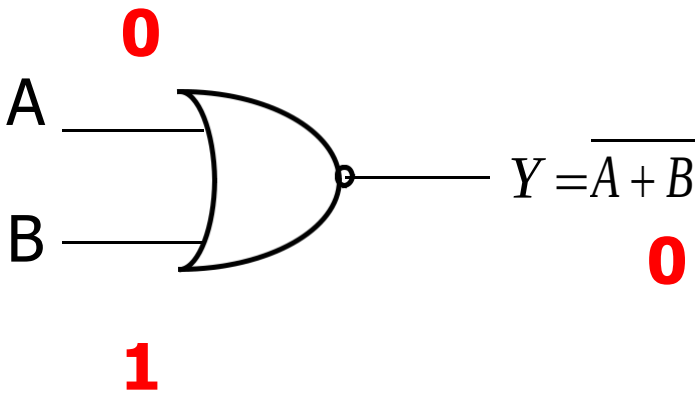
Logic Symbol

NOR Gate



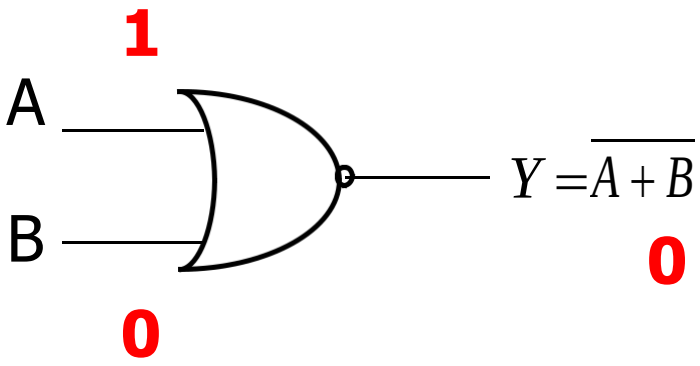
Inputs		Output
A	B	$Y = \overline{A + B}$
0	0	1

NOR Gate



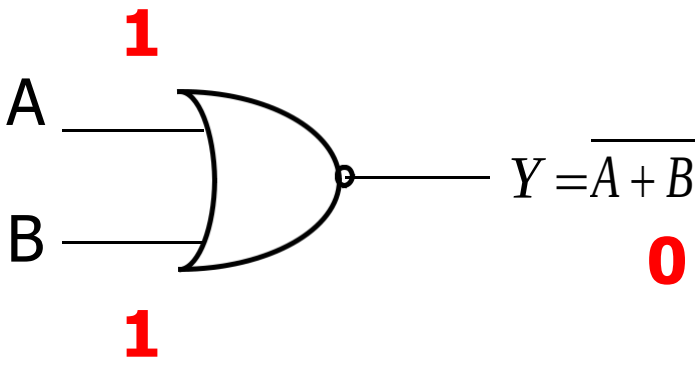
Inputs		Output
A	B	$Y = \overline{A + B}$
0	0	1
0	**1**	**0**

NOR Gate



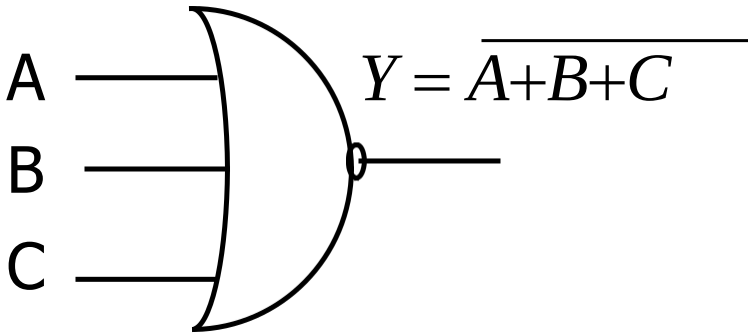
Inputs		Output
A	B	$Y = \overline{A + B}$
0	0	1
0	1	0
1	**0**	**0**

NOR Gate



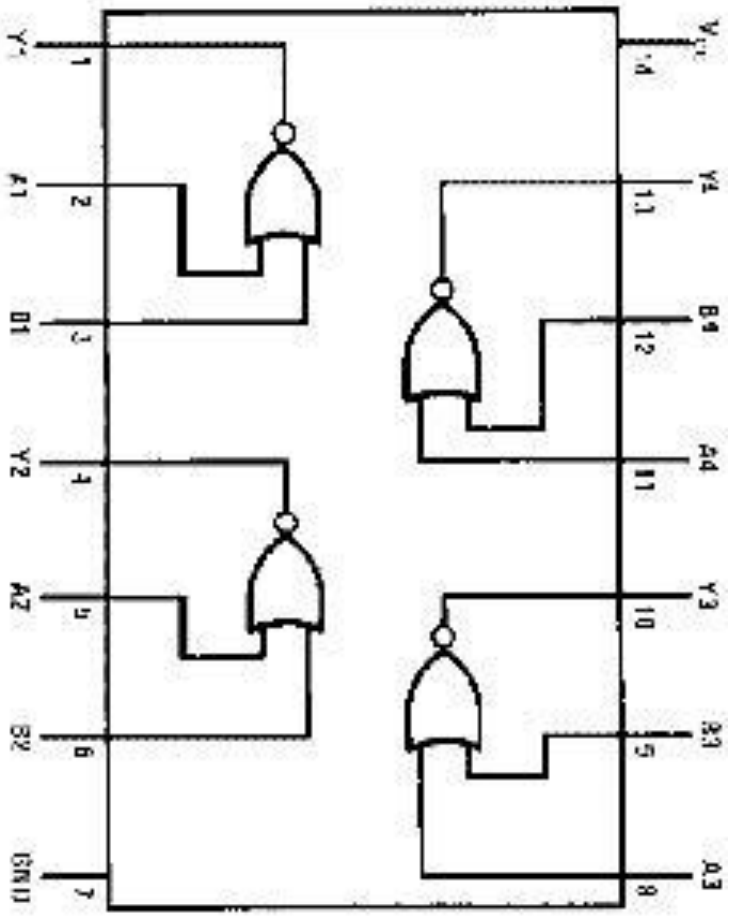
Inputs		Output
A	**B**	$Y = \overline{A + B}$
0	0	1
0	1	0
1	0	0
1	**1**	**0**

3 – Input NOR Gate



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

NOR Gate IC – IC 7402 (Quad 2 I/P NOR Gate)



Pin Configuration of IC 7402

- This device contains four independent gates each of which performs the logic NOR function.

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

Inputs		Output
A	B	Y
L	L	H
L	H	L
H	L	L
H	H	L

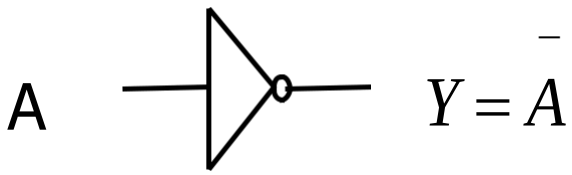
H = HIGH Logic Level

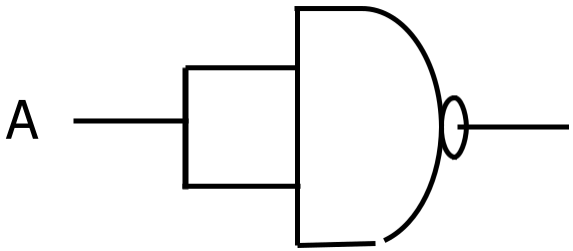
L = LOW Logic Level

Function Table of IC 7402

Universal Gate

NOT Gate using NAND Gate



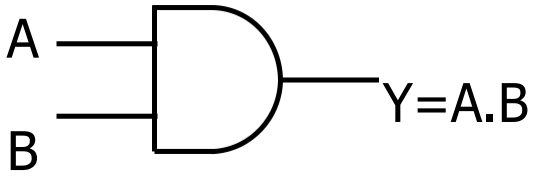


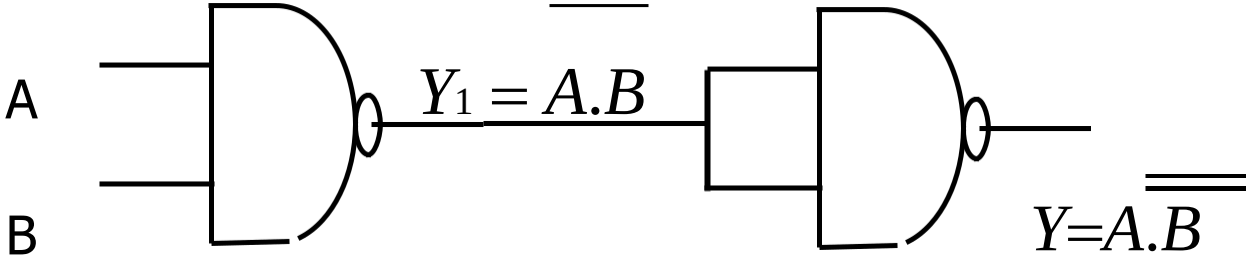
$Y = A.\overline{A}$

$Y = \bar{A}$ ..... (A.A=A)

Universal Gate

AND Gate using NAND Gate

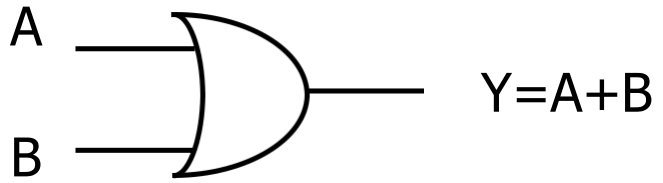


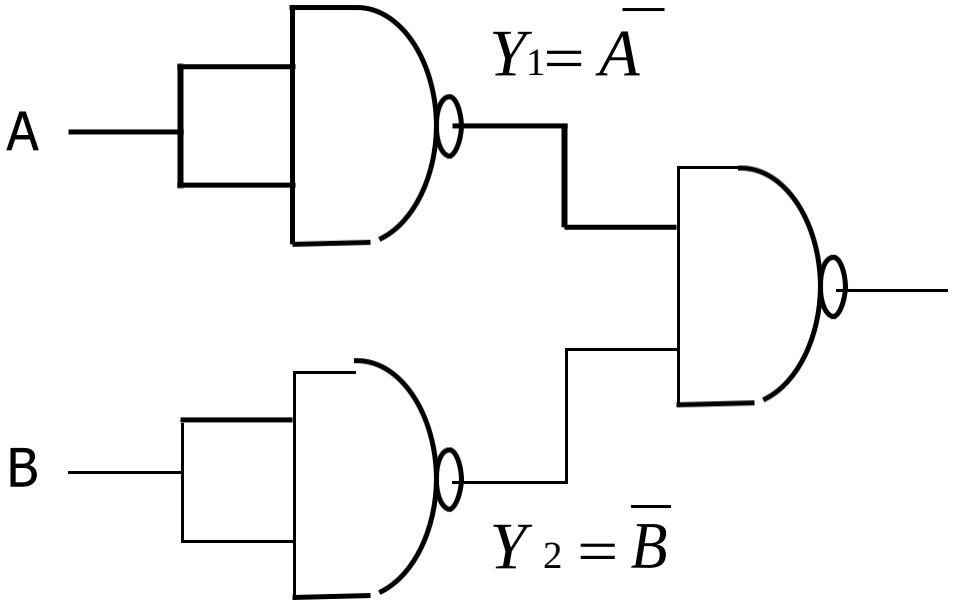


$$Y = \overline{\overline{A.B}} \because (\overline{\overline{A}} = A)$$

Universal Gate

OR Gate using NAND Gate





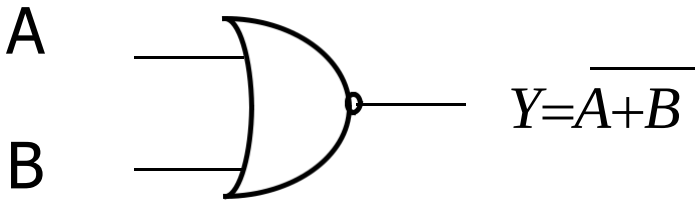
$$Y = \overline{\overline{A}.\overline{B}}$$

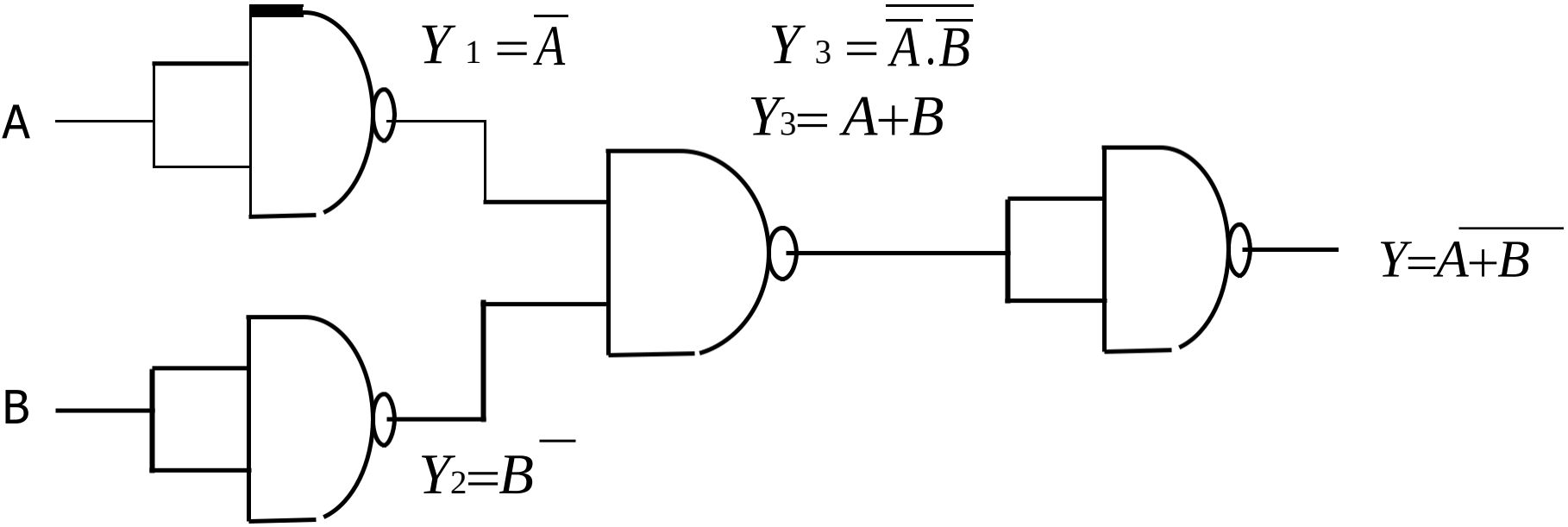
$$Y = \overline{\overline{A}} + \overline{\overline{B}} \quad (\because \text{Demorgan's Theorem})$$

$$Y = A + B$$

Universal Gate

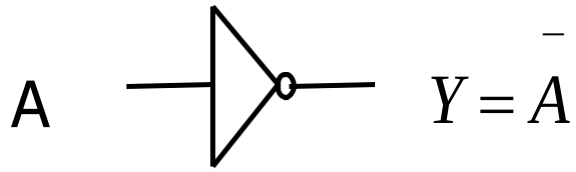
NOR Gate using NAND Gate

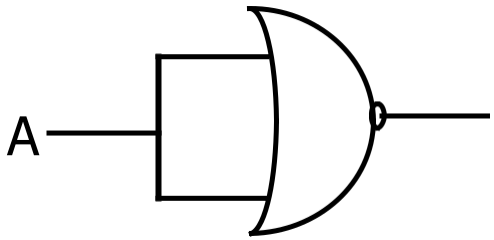




Universal Gate

NOT Gate using NOR Gate



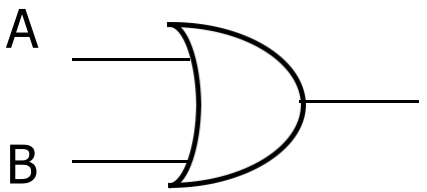


$$Y = \overline{A+A}$$

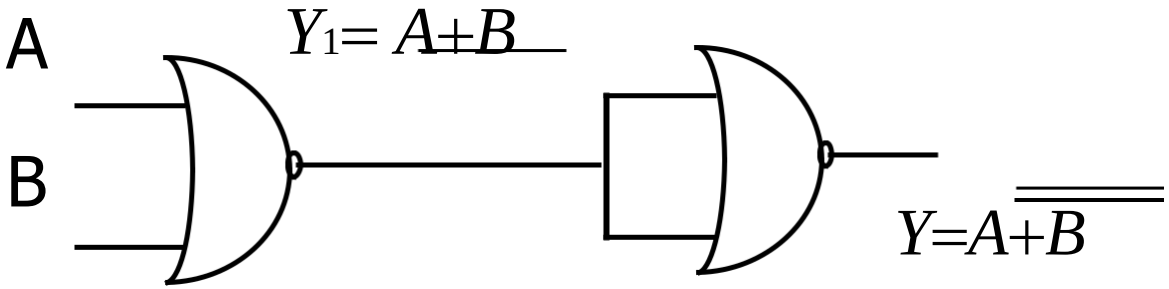
$$Y = \bar{A} \quad \text{..... (A+A=A)}$$

Universal Gate

OR Gate using NOR Gate



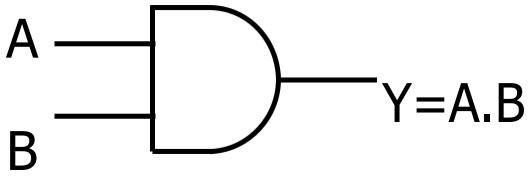
$Y=A+B$

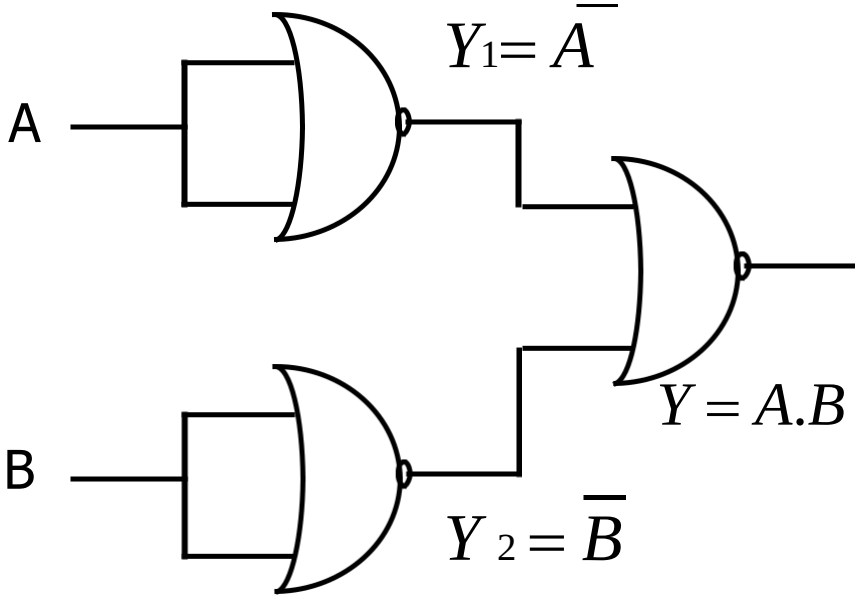


$Y = A+B \qquad (\because \overline{\overline{A}}=A)$

Universal Gate

AND Gate using NOR Gate





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

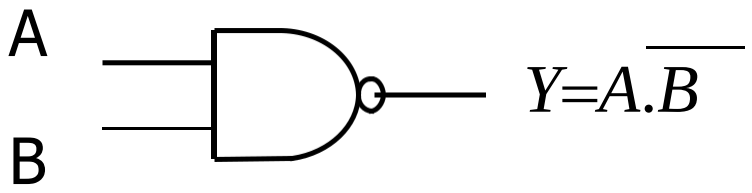
$= =$

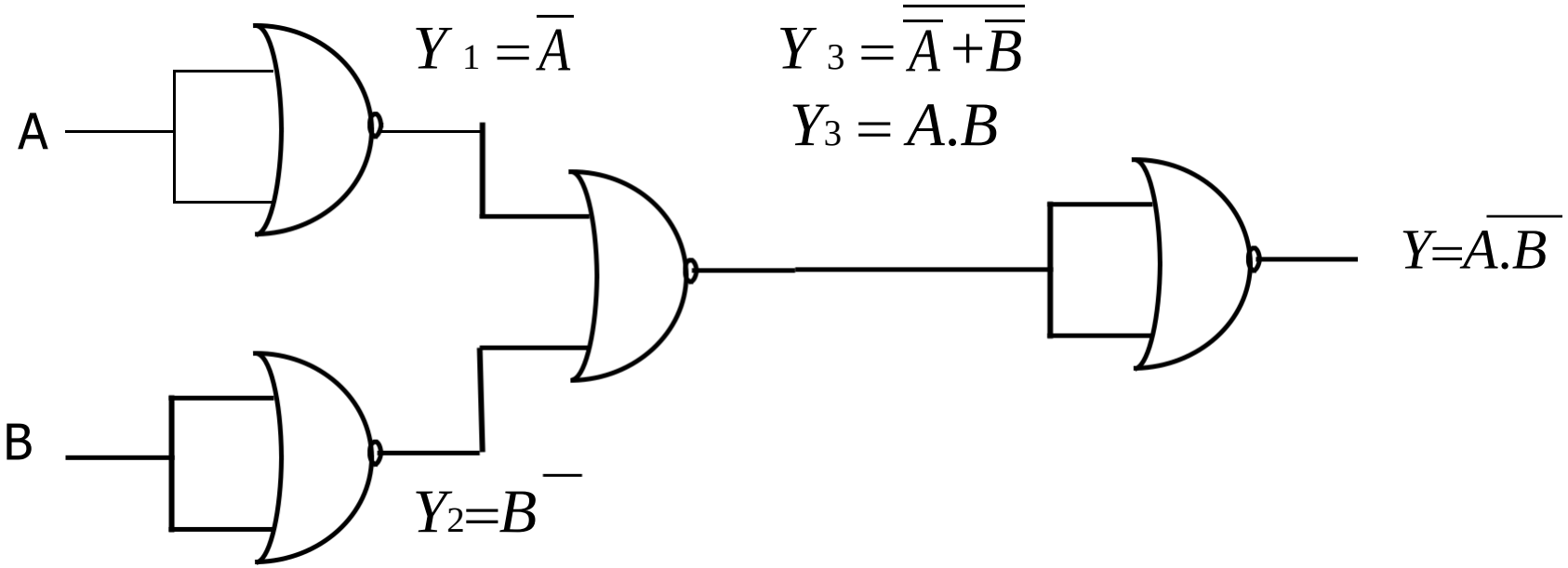
($\because$ Demorgan's Theorem)

$Y = A.B$

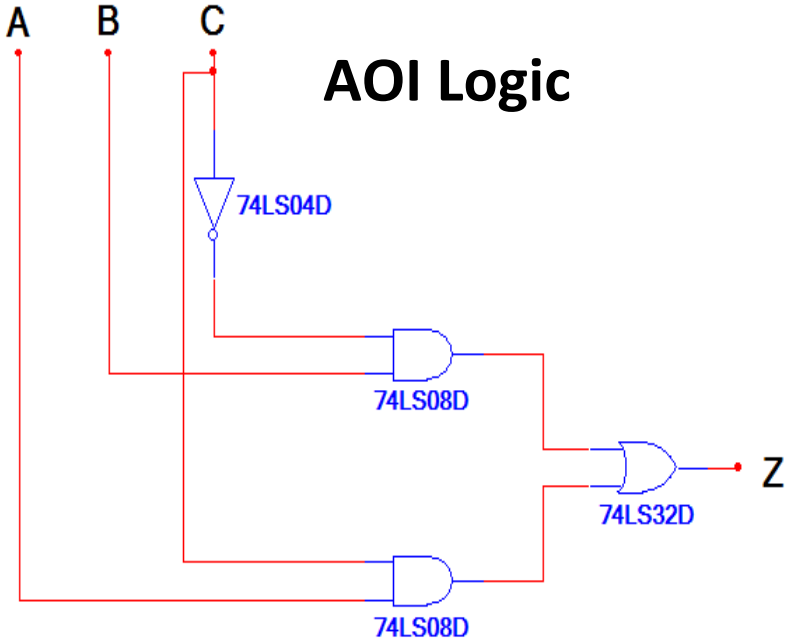
Universal Gate

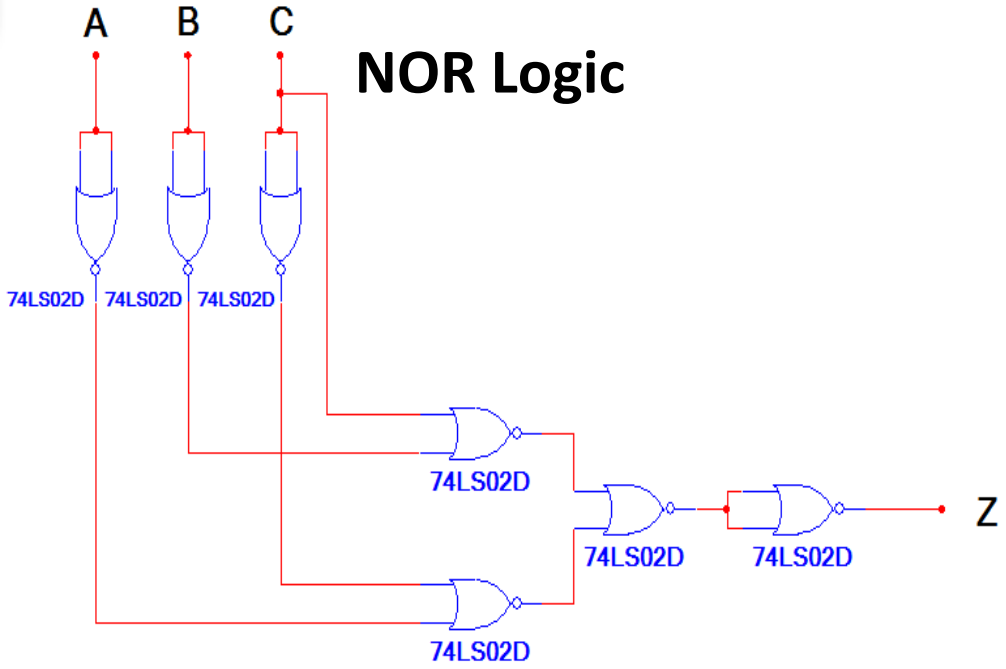
NAND Gate using NOR Gate





Why NAND Logic or NOR Logic?





IC Type	Gates	Gate / IC	# ICs
74LS04	1	6	1
74LS08	2	4	1
74LS32	1	4	1
		Total Number of ICs →	3

IC Type	Gates	Gate / IC	# ICs
74LS02	7	4	2
		Total Number of ICs →	2

Unit II – Logic Gates & Logic Families

- ✓ **Logic Gates: Symbol, diode/transistor switch circuit and logical expression, truth table of** basic gates (AND, OR, NOT), Universal gates (NAND, NOR) and **special purpose gates (Ex-OR, Ex-NOR),** Tristate Logic.

- ✓ **Boolean Algebra:** Laws of Boolean algebra, Duality Theorem, De-Morgan's Theorem

- ✓ **Logic Families:** Characteristics of Logic families: Noise Margin, Power Dissipation, Figure of merit, Fan in and Fan out, Speed of operation, Comparison TTL, CMOS, Types of TTL NAND 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.

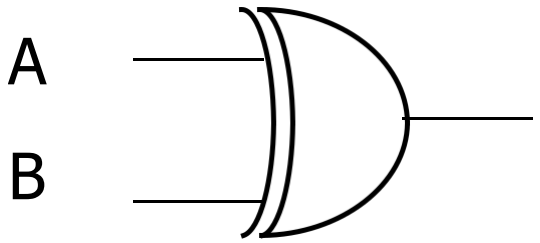
- ✓ Under the conditions when both the inputs assume the logic 0 state or logic 1 state, the output

  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



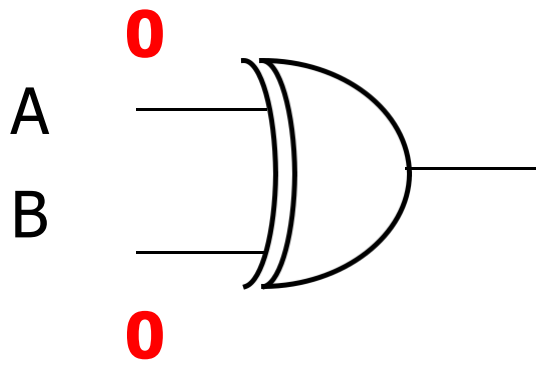
$$Y = A \oplus B$$

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

Logic Symbol

Logic Expression

Ex-OR Gate

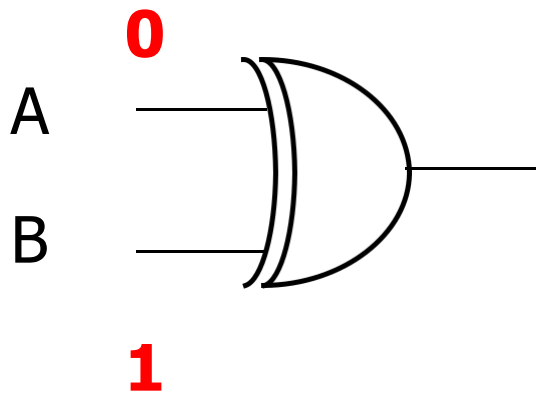


$Y=A\oplus B$

0

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

Ex-OR Gate

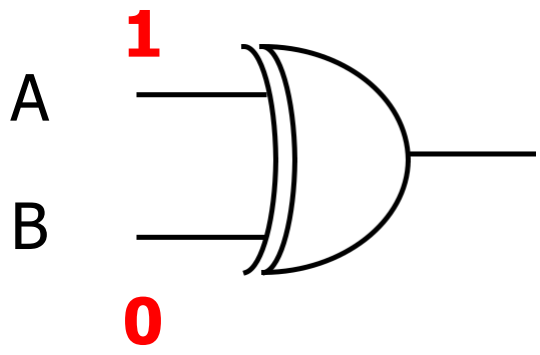


$Y=A\oplus B$

1

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

Ex-OR Gate

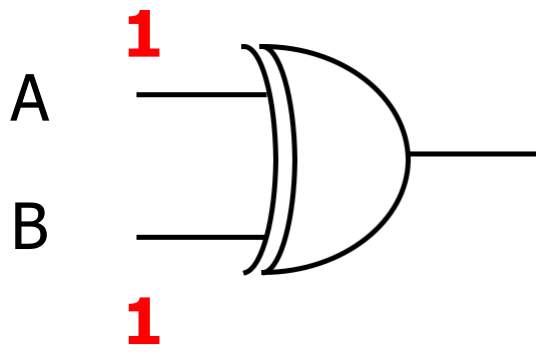


$Y=A\oplus B$

1

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

Ex-OR Gate

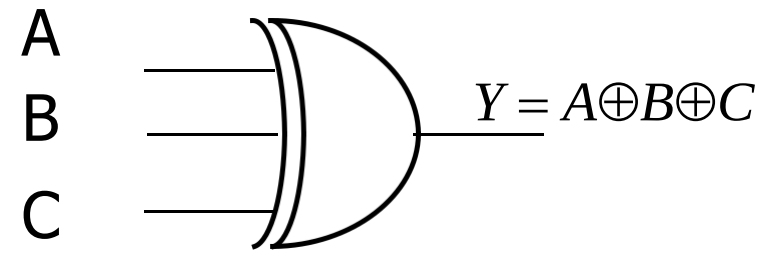


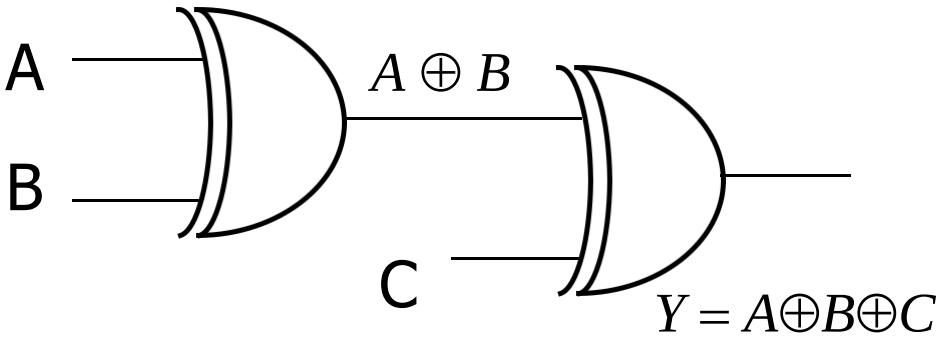
$Y=A\oplus B$

0

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

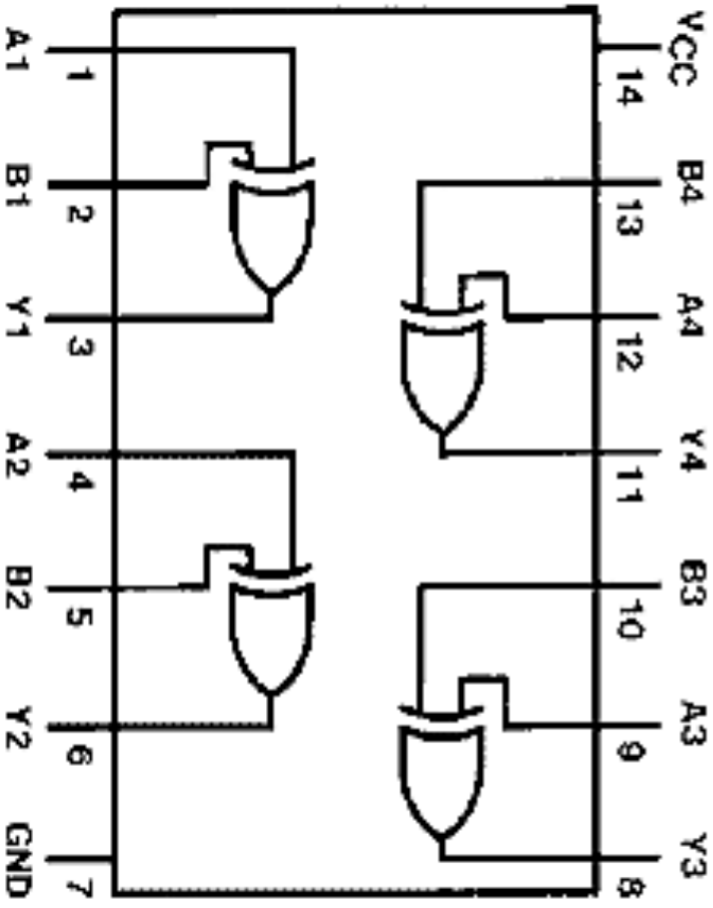
3 – Input Ex-OR Gate





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

Ex-OR Gate IC – IC 7486 (Quad 2 I/P Ex-OR Gate)



Pin Configuration of IC 7486

- This device contains four independent gates each of which performs the logic XOR function.

INPUTS		OUTPUT
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

H = high level, L = low level

Function Table of IC 7486

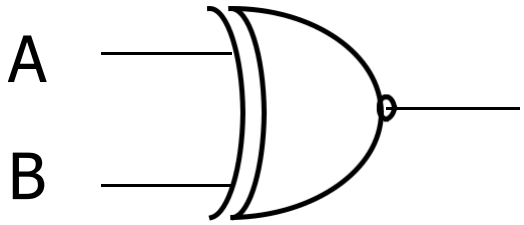
Special Purpose Gate – Ex-NOR Gate

- ✓ 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



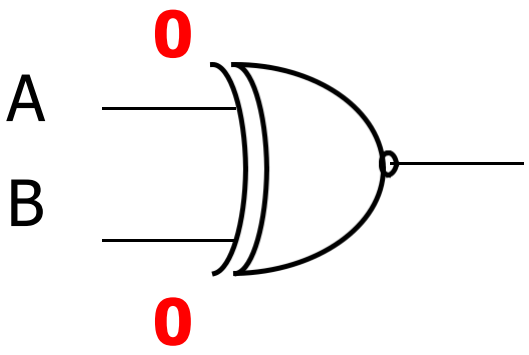
$$Y = A \odot B$$

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

Logic Symbol

Logic Expression

Ex-NOR Gate

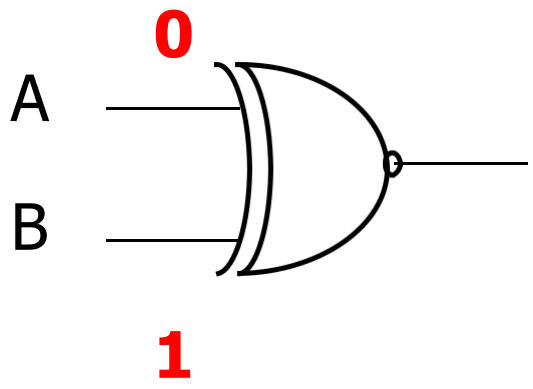


$Y = A \odot B$

1

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

Ex-NOR Gate

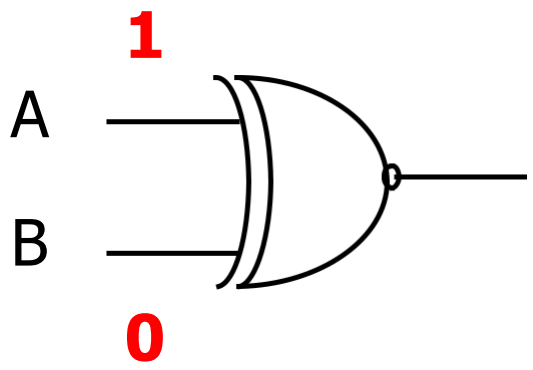


$Y = A \odot B$

0

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

Ex-NOR Gate

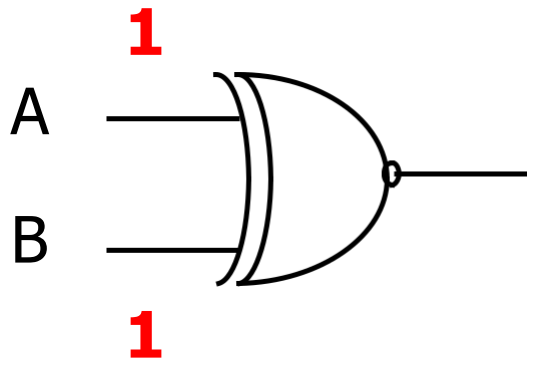


$Y = A \odot B$

0

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

Ex-NOR Gate

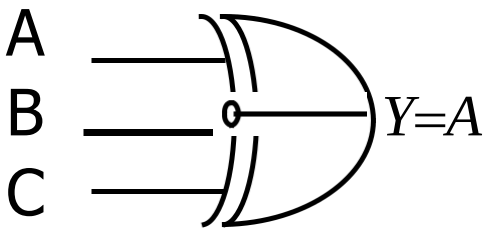


$Y = A \odot B$

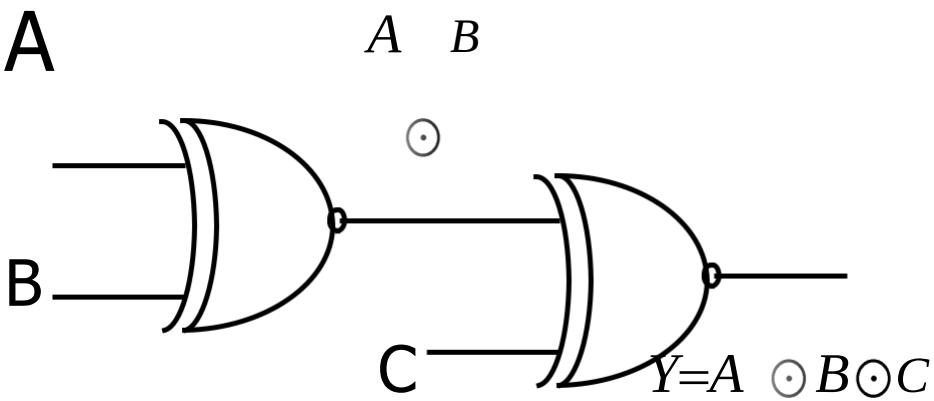
1

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

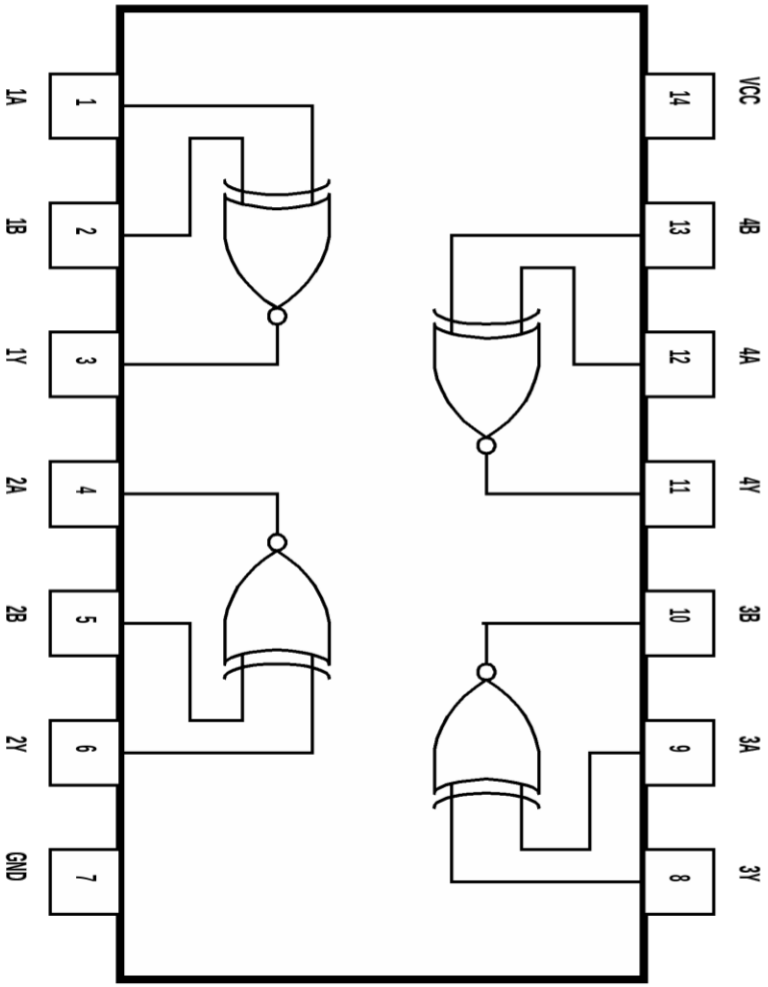


$\odot B \odot C$



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

Ex-NOR Gate IC – IC 74266



Pin Configuration of IC 74266

- ✓ This device contains four independent gates each of which performs the logic XNOR function.

Unit II – Logic Gates & Logic Families

- ✓ **Logic Gates:** Symbol, diode/transistor switch circuit and logical expression, truth table of basic gates (AND, OR, NOT), Universal gates (NAND, NOR) and special purpose gates (Ex-OR, Ex-NOR), Tristate Logic.
- ✓ **Boolean Algebra:** Laws of Boolean algebra, Duality Theorem, De-Morgan's Theorem
- ✓ **Logic Families:** Characteristics of Logic families: Noise Margin, Power Dissipation, Figure of merit, Fan in and Fan out, Speed of operation, Comparison TTL, CMOS, Types of TTL NAND gate.

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.

Unit II – Logic Gates & Logic Families

- ✓ **Logic Gates:** Symbol, diode/transistor switch circuit and logical expression, truth table of basic gates (AND, OR, NOT), Universal gates (NAND, NOR) and special purpose gates (Ex-OR, Ex-NOR), Tristate Logic.
- ✓ **Boolean Algebra: Laws of Boolean algebra**, Duality Theorem, De-Morgan's Theorem
- ✓ **Logic Families:** Characteristics of Logic families: Noise Margin, Power Dissipation, Figure of merit, Fan in and Fan out, Speed of operation, Comparison TTL, CMOS, Types of TTL NAND gate.

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: $\bar{1} = 0$

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

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

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

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

Boolean Algebra

AND Laws

Law 1:	A . 0 = 0	Null Law
Law 2:	A . 1 = A	Identity Law
Law 3:	A._A=A	
Law 4:	$A.A$ =0	

Boolean Algebra

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

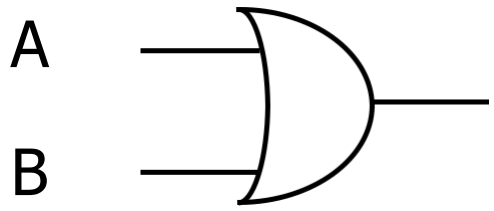
- This Law states that, A OR B is the same as B OR A i.e. the order in which the variables are ORed is immaterial.

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

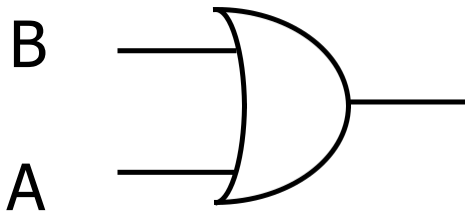
Boolean Algebra

Proof:

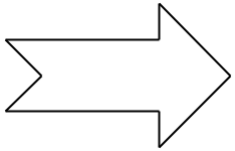
A+B



B+A



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,

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

Boolean Algebra

Commutative Laws

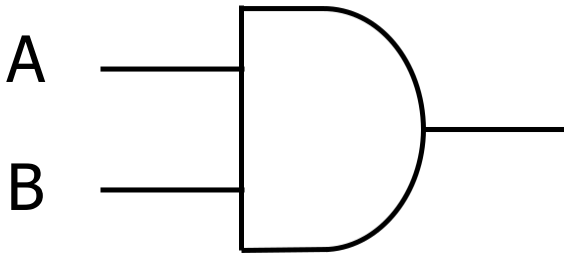
Law 2: A.B = B.A

- ✓ 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.

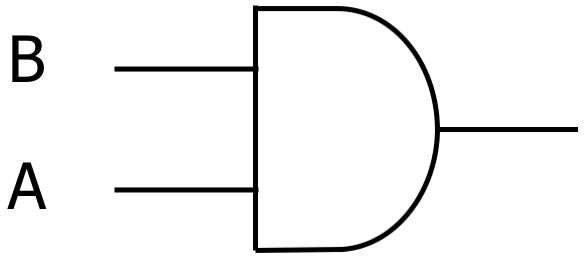
Boolean Algebra

Proof:

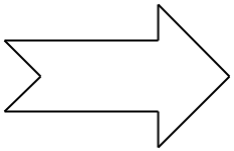
A.B



B.A



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



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)

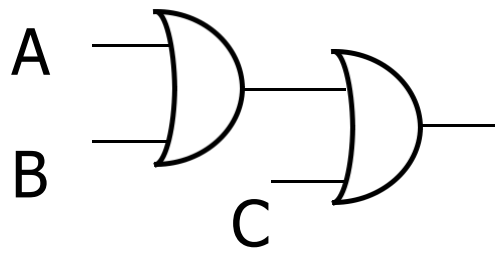
- ✓ A OR B ORed with C is the same as A ORed with B OR C.

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

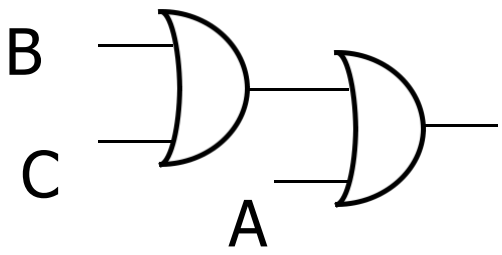
Boolean Algebra

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

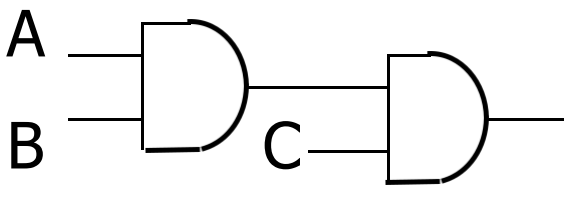
Law 2: (A.B).C = A.(B.C)

- ✓ 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.

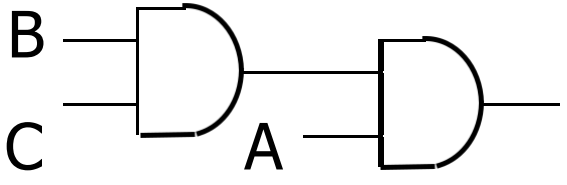
Boolean Algebra

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**

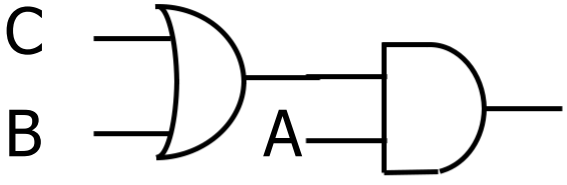
Law 1: A(B+C) = AB+AC

- ✓ 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.

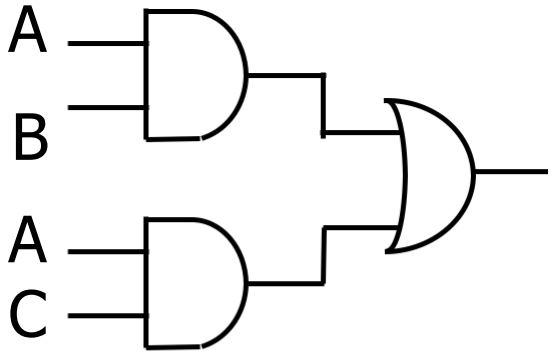
Boolean Algebra

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

Boolean Algebra

Distributive Laws

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

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

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

Boolean Algebra

- **Distributive Laws**

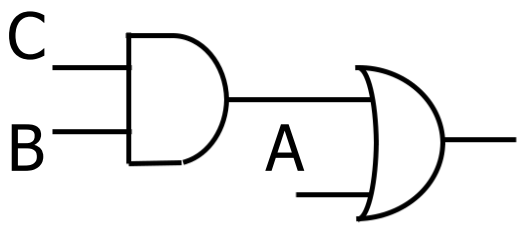
Law 2: A+BC = (A+B).(A+C)

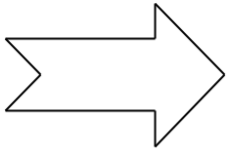
- ✓ This law states that ANDing of several variables and ORing the result with a single variable is equivalent to ORing that single variable with each of the several variables and then ANDing the products.

Boolean Algebra

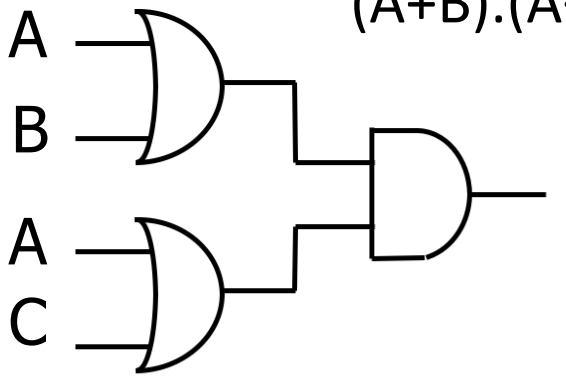
Proof:

A+(B.C)





(A+B).(A+C)



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

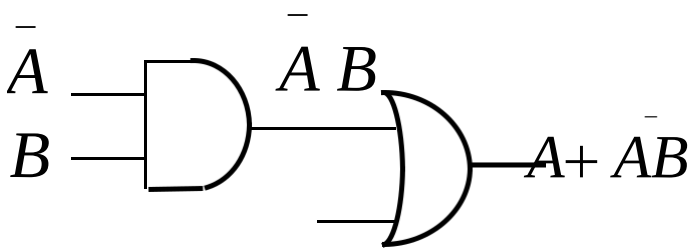
Law 1: $A + \bar{A}B = A + B$

✓ 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

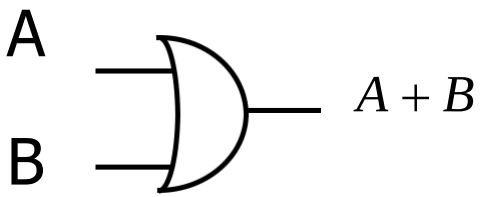
Boolean Algebra

Proof:

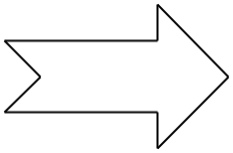
$A + \bar{A}B$



$A + B$



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



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

Boolean Algebra

Redundant Literal Rule

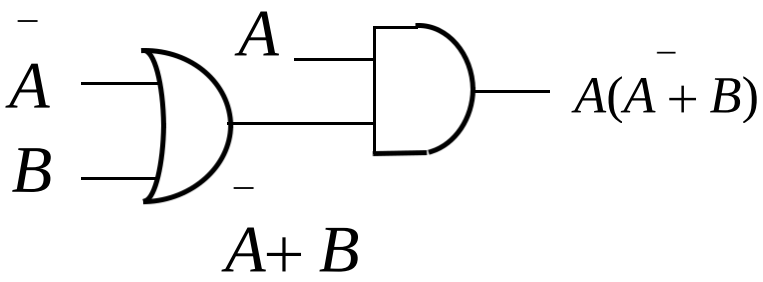
Law 2: $A(A + B) = A.B$

- 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

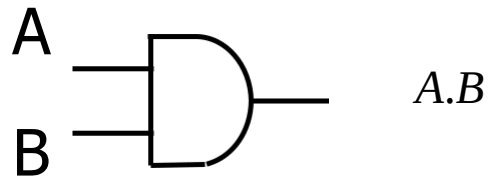
Proof:

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



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

$$A.B$$

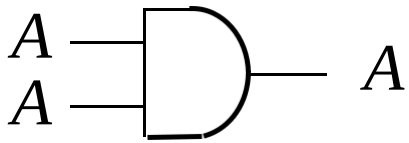


A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Algebra

Idempotence Laws

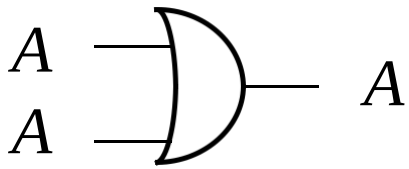
Law 1: $A.A = A$



- ✓ 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: $A+A.B=A$

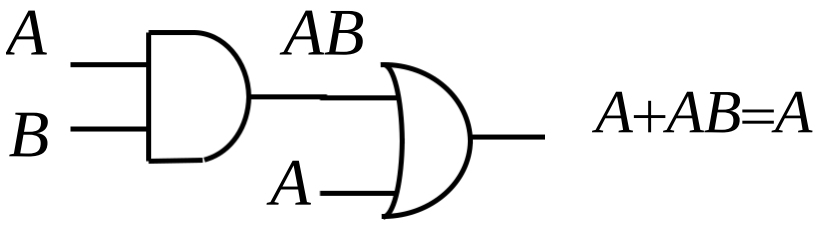
- ✓ This law states that ORing of a variable with AND of that variable and another variable is equal to that variable itself.
- ✓ Therefore,

A + A. Any Term = A

Boolean Algebra

Proof:

$A + A.B$ A



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

Boolean Algebra

Absorption Laws

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

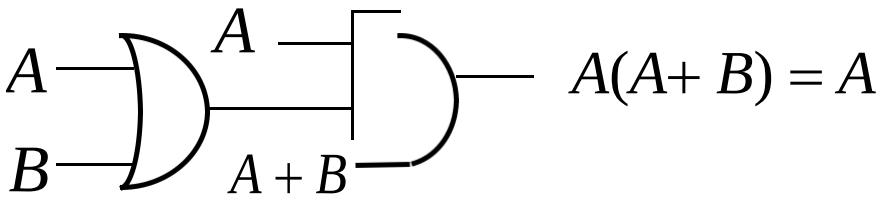
- This law states that ANDing of a variable with OR of that variable and another variable is equal to that variable itself.
- Therefore,

  A . (A + Any Term) = A

Boolean Algebra

Proof:

$A(A+B)$ $\qquad\qquad$ A



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

Unit II – Logic Gates & Logic Families

- **Logic Gates:** Symbol, diode/transistor switch circuit and logical expression, truth table of basic gates (AND, OR, NOT), Universal gates (NAND, NOR) and special purpose gates (Ex-OR, Ex-NOR), Tristate Logic.
- **Boolean Algebra:** Laws of Boolean algebra, Duality Theorem, **De-Morgan's Theorem**
- **Logic Families:** Characteristics of Logic families: Noise Margin, Power Dissipation, Figure of merit, Fan in and Fan out, Speed of operation, Comparison TTL, CMOS, Types of TTL NAND gate.

Boolean Algebra

De-Morgan's Theorem

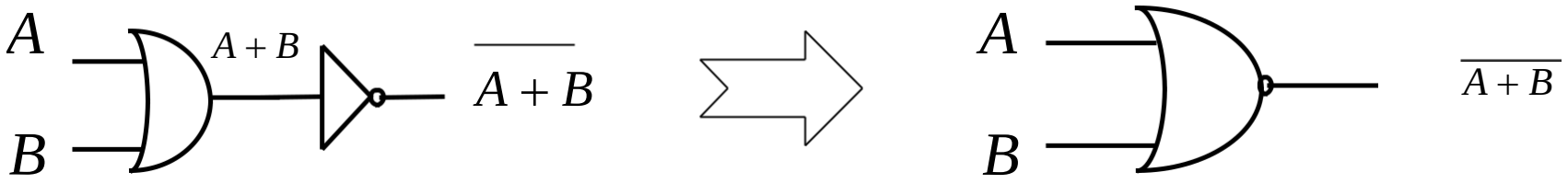
First Theorem: $\overline{A+B} = \bar{A}.\bar{B}$

- ✓ 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

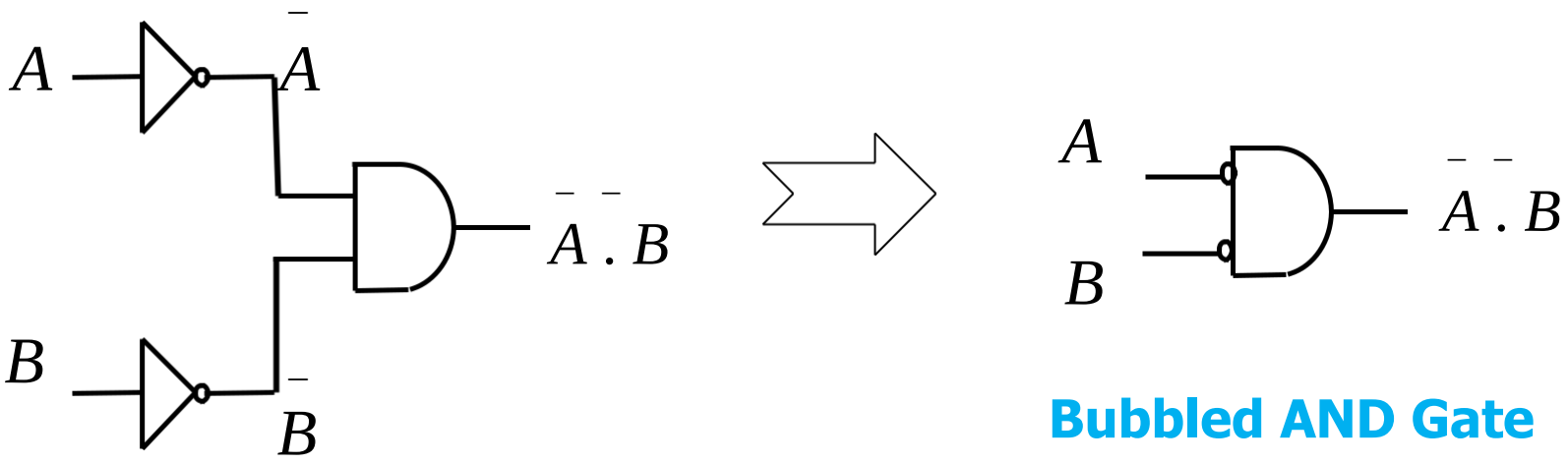
Proof: Logic Diagram

L.H.S.



NOR Gate

R.H.S.



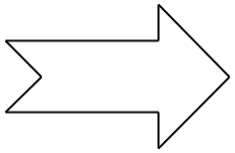
Bubbled AND Gate

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



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

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,

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

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

De-Morgan's Theorem

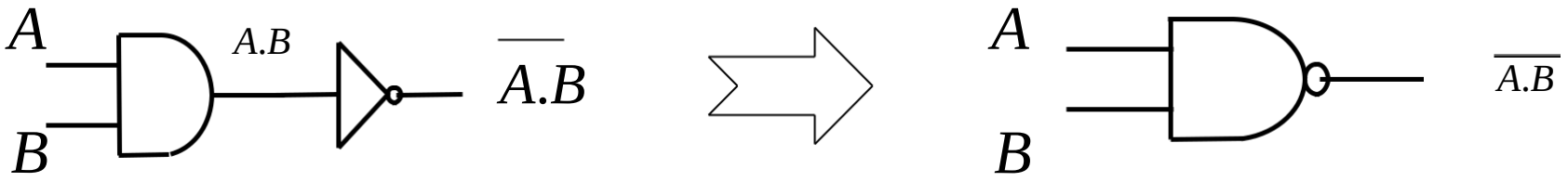
Second Theorem: $\overline{A.B} = \bar{A} + \bar{B}$

- ✓ 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

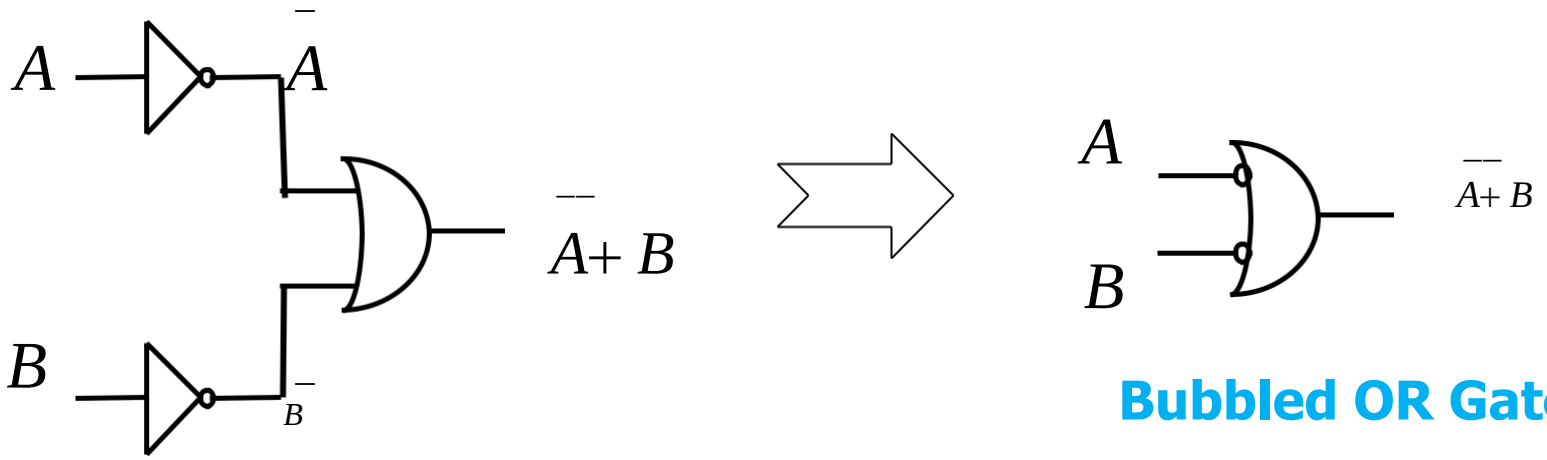
Proof: Logic Diagram

L.H.S.



NAND Gate

R.H.S.



Bubbled OR Gate

Boolean Algebra

Proof:

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

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

A	B	$\bar{A}$	$\bar{B}$	$\bar{\bar{A}} + \bar{\bar{B}}$
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,

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

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

Unit II – Logic Gates & Logic Families

- ✓ **Logic Gates:** Symbol, diode/transistor switch circuit and logical expression, truth table of basic gates (AND, OR, NOT), Universal gates (NAND, NOR) and special purpose gates (Ex-OR, Ex-NOR), Tristate Logic.
- ✓ **Boolean Algebra:** Laws of Boolean algebra, **Duality Theorem,** De-Morgan's Theorem
- ✓ **Logic Families:** Characteristics of Logic families: Noise Margin, Power Dissipation, Figure of merit, Fan in and Fan out, Speed of operation, Comparison TTL, CMOS, Types of TTL NAND gate.

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	$\bar{0} = 1$	$\bar{1} = 0$
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	$A.(\mathrm{A}+ B) = A$	$A+AB=A$
11	$A.(\mathrm{A}.B) = A.B$	$A+\mathrm{A}+B= A+B$
12	$\overline{A.B}= \bar{A}+\bar{B}$	$\overline{A+B}= \bar{A}.\bar{B}$
13	$(\mathrm{A}+ B)(\mathrm{A}+ \mathrm{C}) = (A + B)(\mathrm{A}+ \mathrm{C})$	$\mathrm{A}\bar{B}+\mathrm{AC}= \bar{A}B+\mathrm{AC}$

Example 1

Reduce the following Boolean Expression using Boolean Laws:

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

Example 1

Reduce the following Boolean Expression using Boolean Laws:

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

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

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

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

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

$$= 1$$

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

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:

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

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

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

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

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

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

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

$$= AC + BC$$

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

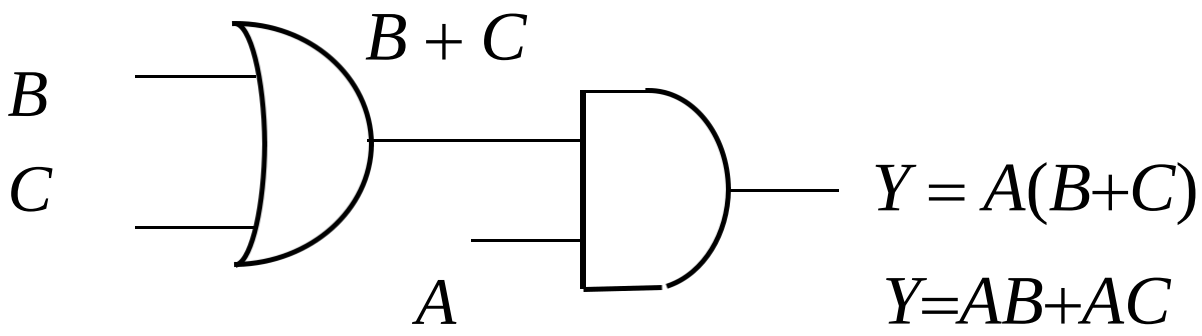
But we have to use only 1 AND gate and 1 OR gate

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:

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

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

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

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

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

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

$$= A+B \qquad \because (\ A+\bar{A}=1)$$

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

Example 5

Prove that:

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

Example 5

Prove that:

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

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

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

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

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

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

$= A$ $(\because A+A=A)$

L.H.S = 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:

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

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

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

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

$=0$

L.H.S = 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:

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

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

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

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

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

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

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

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

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

Example 8

Simplify;

$$F=XY+XYZ+XY\overline{Z}+XZY$$

Example 8

Simplify; $F=XY+XYZ+XY\bar{Z}+XZY$

$F=XY+XYZ+XY\bar{Z}+XZY$

$= XY + XYZ + X\bar{Z}Y$ $\quad (\because XYZ+XYZ=XYZ)$

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

$= XY$ $\quad (\because 1+Z+\bar{Z}=1)$

$= XY$

Example 9

Prove that;

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

Example 9

Prove that;

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

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

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

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

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

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

L.H.S = R.H.S