Module 6

DIGITAL ELECTRONICS

Department of Electronics and Communication Engineering,
Manipal Institute of Technology, Manipal, INDIA

DIGITAL ELECTRONICS

- In analog system, the output can be continuously controlled by the input & the output is linearly proportional to the input. In digital system, the digital logic used only two values, either HIGH or LOW. i.e. they have only two discrete values and are called BINARY. The binary may be either logic 0 or logic ‘1’. A logic value of ‘0’ or ‘1’ is often called as BINARY DIGIT or BIT.

Number System

- **Number System:** Many number systems are used in digital technology. Most common are binary, decimal, octal & hexadecimal system.
- **Binary Number system:** A number system that uses only two digits ‘0’&’1’ is called binary number system. The binary number system is also called as Base 2 system or Radix 2 system.
- Examples: $(100010)_2$
- $(0.1011)_2$

Convert the given binary number into decimal equivalent number

1. $(100010)_2 = 0\times2^0+1\times2^1+0\times2^2+0\times2^3+0\times2^4+1\times2^5$

$= 0+2+0+0+0+32$

$= (34)_{10}$

2. $(0.1011)_2 = 1\times2^{-1}+0\times2^{-2}+1\times2^{-3}+1\times2^{-4}$

$= 0.5+0+0.125+0.0625$

$= (0.6875)_{10}$

3. $(10101.011)_2$

Integer part:

$$(10101)_2 = 1\times2^0+0\times2^1+1\times2^2+0\times2^3+1\times2^4$$
$$=1+0+4+0+16 = (21)_{10}$$

Decimal part: $(0.011)_2 = 0\times2^{-1}+1\times2^{-2}+1\times2^{-3}$

$$= (0.375)_{10}$$
$$= (21.375)_{10}$$

Octal Number System: A number system that uses 8 digit (0-7) is called an octal number system. It has base 8.

Example: $(723)_8$, $(676)_8$

Octal Numbers	Binary Equivalent number
	4 2 1 (weights)
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

Hexadecimal Number System:

The hexadecimal number system has base 16. It has 16 distinct digit symbols. It uses the digits 0-9 & letters A,B,C,D,E,F as 16 digit symbols.

Hexadecimal number system	Equivalent binary number 8 4 2 1 (weights)
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0 = A
11	1 0 1 1 = B
12	1 1 0 0 = C
13	1 1 0 1 = D
14	1 1 1 0 = E
15	1 1 1 1 = F

Convert the following octal number into decimal number system

1. $(2376)_8 = (?)_{10}$

$= 2\times8^3+3\times8^2+7\times8^1+6\times8^0$

$= 1024+192+56+6$

$= (1278)_{10}$

2. $(1234.567)_8=1\times8^3+2\times8^2+3\times8^1+4\times8^0$

$+5\times8^{-1}+6\times8^{-2}+7\times8^{-3}$

$=512+128+24+4+0.625+0.09375+0.01367$

$= (668.7324219)_{10}$

Convert the following hexadecimal number into decimal number system

1. $(269)_{16} = 2\times16^2+6\times16^1+9\times16^0$

$= 2\times256+96+9$

$= (617)_{10}$

2. $(2B8D.E2)16 = 2\times16^3+11\times16^2+8\times16^1+13\times16^0 +14\times16^{-1}+2\times16^{-2}$

$= 8192+2816+128+13+0.875+0.0078125$

$= (11149.88281)_{10}$

Conversion form Decimal number system to Binary number system:

*The given decimal number (integer) is repeatedly divided by 2, which is the base number of binary system till quotient becomes ‘0’ and collect the remainder from bottom to top.

*To convert the fractional part into binary, multiply fraction by 2 repeatedly, record any carry in integer place. The string of integer obtained from top to bottom gives the equivalent fraction in binary number system.

1. $(734)_{10}=(?)_2$

$2\angle 734$

$2\angle 367$ - - 0

$2\angle 183$ - - 1

$2\angle 91$ - - 1

$2\angle 45$ - - 1

$2\angle 22$ - - 1

$2\angle 11$ - - 0

$2\angle 5$ - - - 1

$2\angle 2$ - - - 1

1 - - - - - 0 ↑

Take the numbers from bottom to top,

$(734)_{10} = (1011011110)_2$

2. $(0.705)_{10}$

0.705×2=1.410 ---1

0.410×2=0.820 ---0

0.82×2= 1.64------1

0.64×2= 1.28------1

0.28×2= 0.56------0

0.56×2=1.12-------1

0.12×2=0.24-------0

0.24×2=0.48-------0

0.48×2=0.96-------0

0.96×2=1.92-------1

Take the number from top to bottom,

$(0.705)_{10} = (0.1011010001)_2$

3. $(41.915)_{10}$

$2 \angle 41$

$2 \angle 20 - - 1$

$2 \angle 10 - - 0$

$2 \angle 5 - - 0$

$2 \angle 2 - - 1$

$1 - - - - 0 \uparrow$

$(41)_{10} = (101001)_2$

$0.915 \times 2 = 1.830 --- 1$

$0.830 \times 2 = 1.660 --- 1$

$0.660 \times 2 = 1.320 --- 1$

$0.32 \times 2 = 0.64 ------ 0$

$0.64 \times 2 = 1.28 ----- 1$

$(0.915)_{10} = (11101)_2$

$(41.915)_{10} = (101001.11101)_2$

Conversion form Decimal number system to Octal number system:

- To convert a decimal number (integer) into octal equivalent, repeatedly divide by 8 and take the remainder string from bottom to top.
- For fraction part repeatedly multiply by 8, record carry in integer place & take the string of integer from top to bottom.

$(632.97)_{10} = (?)_8$

8 ∠ 632

8 ∠ 79 — — — 0

8 ∠ 9 — — — —7

1 — — — — — — 1 ↑

$(632)_{10} = (1170)_8$

$0.97 \times 8 = 7.76$ --- 7

$0.76 \times 8 = 6.08$ --- 6

$0.08 \times 8 = 0.64$ --- 0

$0.64 \times 8 = 5.12$ --- 5

$0.12 \times 8 = 0.96$ --- 0

$(0.76050)_8$

$(632.97)_{10} = (1170.76050)_8$

Decimal number system to Hexadecimal number system:

To convert a decimal number (integer) into a hexadecimal equivalent, repeatedly divide by 16 and take the remainder string from bottom to top.

For traction part repeatedly multiplied by 16, record carry in integer place & take the string of integer from top to bottom.

$(22.64)_{10} = (?)_8$

$16 \angle 22$

1------6 ↑

$(22)_{10} = (16)_{16}$

0.64×16=10.24---10-A

0.24×16=3.84-----3

0.84×16=13.44—13-D

0.44×16=7.04---7

0.04×16=0.64---0

$(22.64)_{10} = (16.A3D70)_{16}$

Conversion from Octal number system into Binary number system:

When an octal number is to be converted to its equivalent binary number, each of its digits is replaced by equivalent group of three binary digits.

$(7423.245)_8$

7	4	2	3	.	2	4	5
111	100	010	011	.	010	100	101

$(7423.245)_8 = (111100010011.010100101)_2$

- **Conversion from Binary number system to Octal number system:**

To convert, Starting from the binary point, the binary digits are arranged in groups of three on both sides. Each in group of binary digit is replaced by its octal equivalent.

Note: 0's can be added on either side, if needed to complete a group of three.

$(11101101110.11111)_2$

011	101	101	110	.	111	110
3	5	5	6	.	7	6

$(11101101110.11111)_2 = (3556.72)_8$

Conversion from Hexadecimal number system to Binary number system

When a hexadecimal number is to be converted its equivalent binary number, each of its digits is replaced by equivalent group of 4 binary digits.

$(347.28)_{16}$

3	4	7	.	2	8
0011	0100	0111	.	0010	1000

$(347.28)_{16} = (001101000111.00101000)_2$

Conversion from Binary number system to Hexadecimal number system:

To convert, Starting from the binary point, the binary digits are arranged in groups of four on both sides. Each in group of binary digit is replaced by its hexadecimal equivalent. Note: 0's can be added on either side, if needed to complete a group of four.

$(110111101.01)_2=(?)_{16}$

0001	1011	1101	.	0100
1	B	D	.	4

$(110111101.01)_2=(1BD.4)_{16}$

Conversion from Octal number system to Hexadecimal number system:

Write down the three bit binary equivalent of octal digit and then rearranging into group of four bits with '0's added on either side of decimal point, if needed to complete the group of four.

$(46)_8 = (?)_{16}$

Octal equivalent	4 6 100 110
Hexadecimal equivalent	0010 0110 2 6

$(46)_8 = (26)_{16}$

Conversion from Hexadecimal number system to Octal number system:

First write down the 4 bit binary equivalent of hexadecimal digit and then rearranging into group of three bit each.

$(2AB.9)_{16} = (?)_8$

Hexadecimal equivalent	2 A B . 9 0010 1010 1011 . 1001
Octal equivalent	<u>001</u> <u>010</u> <u>101</u> <u>011</u> . <u>100</u> <u>100</u> 1 2 5 3 . 4 4

$(2AB.9)_{16} = (1253.44)_8$

Represent the $(743.6)_{10}$ in BCD

7	4	3	.	6
0111	0100	0011	.	0110

$(743.6)_{10}=(011101000011.0110)_{BCD}$

Convert the following BCD into Decimal:
11011100001001

0011	0111	0000	1001
3	7	0	9

$(11011100001001)_{BCD}=(3709)_{10}$

Binary Addition: The rules adopted for binary additions are

0		0		1		1	
+0		+1		+0		+1	
C=0	S=0	C=0	S=1	C=0	S=1	C=1	S=0

The sum of two 1's gives binary '10' i.e.2, there is a carry.

The carry is taken to the next higher column.

Example : $(1010)_2+(0111)_2$

1010 =10
+0111 =7
10001 =17

$(1010)_2+(0111)_2=(10001)_2$

1's complement subtraction:

Step1: Add minuend to the 1's complement of the subtrahend.

Step2: Inspect the result obtained in step1 for an end carry. (a) If an end carry occurs, add 1 to the least significant bit. (end round carry) (b) If an end carry doesn't occur, take 1's complement of the number obtained in step1 and place a negative sign in front of it.

Example; $(1000)_2$ from $(1101)_2$

1101—minuend

1000— subtrahend

1's complement of subtrahend = 0111

Add minuend and 1's complement of subtrahend,

	1	1	0	1
+	0	1	1	1
1	**0**	**1**	**0**	**0**
End carry	Add to LSB			+1
	0	**1**	**0**	**1**

$(1101)_2 - (1000)_2 = (0101)_2$

Example $(6)_{10}-(14)_{10}$

6=0110—minuend

14=1110— subtrahend

1’s complement of subtrahend = 0001

Add minuend and 1’s complement of subtrahend,

	0	1	1	0
+	0	0	0	1
No End carry	**0**	**1**	**1**	**1**

Take 1’s complement of the 0111 & place negative sign in front of it =-1000

$(0110)_2$ - $(1110)_2$= $-(1000)_2$

2’s complement:

To find the 2’s form of any binary number, obtain the 1’s complement of the given number and then add ‘1’ to the LSB.

(100100)2

Take 1’s complement of the number = 011011

Add ‘1’ to LSB to get 2’s complement = 011011

$$\begin{array}{r} 011011 \\ \underline{+1} \\ 011100 \end{array}$$

2’s complement = $(011100)_2$

2’s complement subtraction:

Step1: find 2’s complement of subtrahend

Step2: Add minuend and 2’s complement subtrahend

Step3: (a) If an end carry occurs, discard it. (b) If an end carry doesn’t occur, take 2’s complement of the number obtained in step2 and place a negative sign in front of it.

Example 1. $(1111)_2-(1100)_2$

1111—minuend

1100— subtrahend

2’s complement of subtrahend = 0100

Add minuend and 2’s complement of subtrahend,

	1	1	1	1
+	0	1	0	0
1 Neglect end carry	**0**	**0**	**1**	**1**

$(1111)_2-(1100)_2=(0011)_2$

BOOLEAN ALGEBRA

George Boole in 1854 invented a new kind of algebra known as Boolean algebra. It is sometimes called switching algebra. Boolean algebra is the mathematical frame work on which logic design based. It is used in synthesis & analysis of binary logical function.

Basic Laws of Boolean algebra:

1. **Laws of complementation**: The term compliment means invert. i.e. to change 0's to 1's and 1' to 0's. The following are the laws of compliment $\bar{0}=1;\ \bar{1}=0;\ \bar{\bar{A}}=A.$

2. " **OR" laws**

$$0+0=0;\ 0+1=1;\ 1+0=1;1+1=1;1+A=1;\ A+\bar{A}=1;\ A+A=A;\ 1+\bar{A}=1$$

3. "**AND' laws:** $0.0=0;\ 0.1=0;1.0=0;\ 1.1=1;\ A.\bar{A}=0;\ A.A=A$

Commutative Law:

Property 1 : A+B=B+A

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

Property 2: This property of multiplication states that the order in which the variables are AND'ed makes no difference in the output. i.e. A.B=B.A

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

Associative property:

Property1: This property states that in the OR'ing of the several variables, the result is same regardless of grouping of variables. For three variables i.e.(A OR'ed with B)or'ed with C is same as A OR'ed with (B OR'ed with C)

i.e. (A+B)+C = A+(B+C)

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

Property2: The associative property of multiplication states that, it makes no difference in what order the variables are grouped when AND ’ing several variables. (A.B)C = A(B.C)

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

Distributive property:

Property 1: A(B+C) = A.B + A.C

1	2	3	4	5	6	7	8
A	B	C	B+C	A(B+C)	A.B	A.C	A.B+A.C
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Column number 5= Column number 8, hence the proof.

Property 2: $A + \bar{A}B = A + B$

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

Duality: The important property to Boolean algebra is called Duality principle. The Dual of any expression can be obtained easily by the following rules.

1. Change all 0’s to 1’s

2. Change all 1’s to 0’s

3. **.**’s (dot’s) are replaced by +’s (plus)

4. +’s (plus) are replaced by **.**’s (dot’s)

Examples:

$$\bar{0}=1 \equiv \bar{1}=0$$

$$X+0=X \equiv X.1=X$$

$$X+Y=Y+X \equiv X.Y=Y.X$$

$$X+1=0 \equiv X.0=1$$

De Morgon's Theorems: It is one of the important properties of Boolean algebra. It is extensively useful in simplifying complex Boolean expression.

De Morgon's First Theorem: It states that "the compliments of product of two variables equal to sum of the compliments of individual variable".

i.e. $\overline{AB} = \overline{A} + \overline{B}$

A	B	$\overline{A}$	$\overline{B}$	$A.B$	$\overline{AB}$		$\overline{A}+\overline{B}$
0	0	1	1	0	1		1
0	1	1	0	0	1	≡	1
1	0	0	1	0	1		1
1	1	0	0	1	0		0

De Morgan's Second Theorem: It states that compliment of sum of two variables is equal to product of compliment of two individual variables.

i.e. $\overline{A+B} = \overline{A} \cdot \overline{B}$

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

Simplify the Boolean expression

1. $\bar{X}\bar{Y}Z + \bar{X}YZ$

$= \bar{X}Z[\bar{Y} + Y]$

$= \bar{X}Z[1] = \bar{X}Z$

2. $f = X(\bar{X} + Y)$

$= X\bar{X} + XY = 0 + XY = XY$

3. $f = B(A+C)+C$

$= BA+BC+C$

$= BA+C(1+B)$

$= BA+C$

4. $XY+XYZ+XY\bar{Z}+\bar{X}YZ$

$= XY(Z+\bar{Z})+XYZ+XY\bar{Z}+\bar{X}YZ$

$XYZ+XY\bar{Z}+XY\bar{Z}+\bar{X}YZ+XYZ$

$XYZ(1+1)+XY\bar{Z}(1+1)+\bar{X}YZ$

$XYZ+XY\bar{Z}+\bar{X}YZ$

$XY(Z+\bar{Z})+\bar{X}YZ$

$XY+\bar{X}YZ$

$Y(X+\bar{X}Z)$

$Y(X+Z)$

5. $XYZ+\bar{X}Y+XY\bar{Z}$

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

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

$= Y\bar{X}+Y\bar{Z}+XY\bar{Z}$

$= Y\bar{X}+Y(Z+X\bar{Z})$

$= Y\bar{X}+Y(Z+X)$

$= Y(Z+\bar{X}+X)$

$= Y(Z+1)$

$= Y$

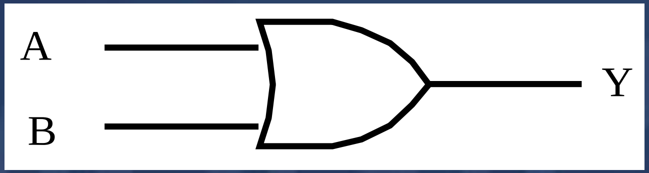
Logic Gates

- It is an electronic circuit which makes logic decisions. A logic gate is a digital circuit with one or more input signal and only one output signal. All input or output signals either low voltage or high voltage. A digital circuit is referred to as logic gate for simple reason i.e. it can be analyzed on the basis of Boolean algebra.
- To make logical decisions, three gates are used. They are OR, AND and NOT gate. These logic gates are building blocks which are available in the form of IC.
- The input and output of the binary variables for each gate can be represented in a tabular column or truth table.

1. OR Gate: The OR gate performs logical additions commonly known as OR function. The OR gate has two or more inputs and only one output. The operation of OR gate is such that a HIGH(1) on the output is produced when any of the input is HIGH. The output is LOW(0) only when all the inputs are LOW.

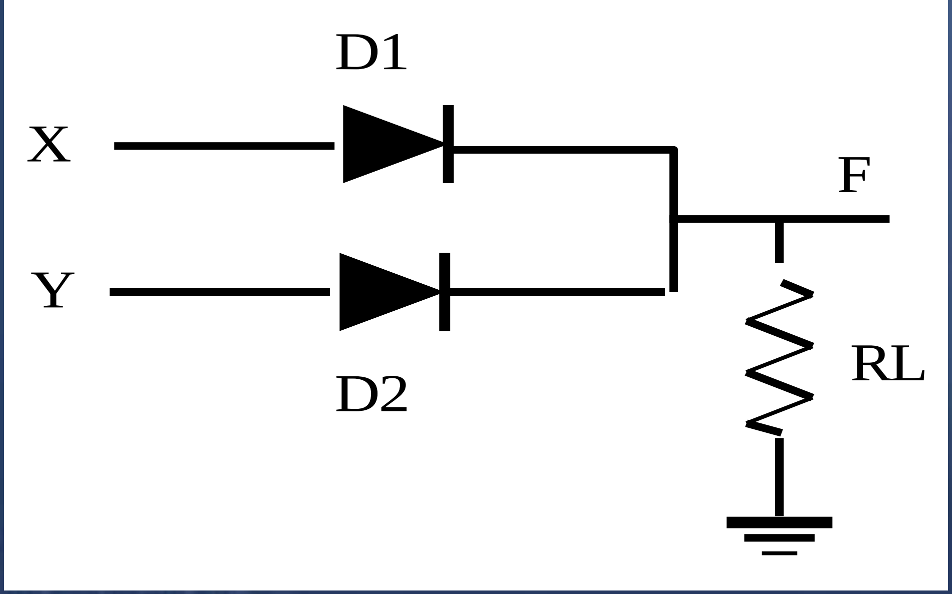
If A & B are the input variables of an OR gate and c is its output, then A+B. similarly for more than two variables, the OR function can be expressed as Y=A+B+C.

Logical Symbol:

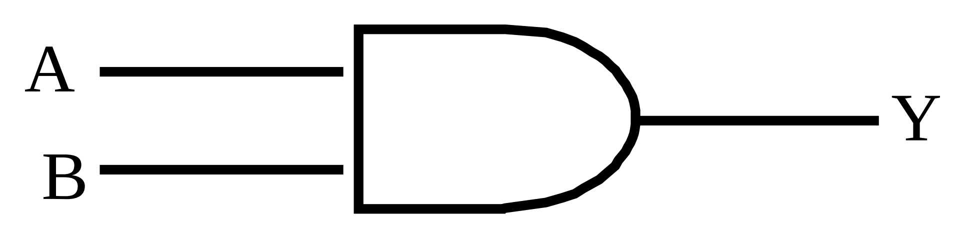


Input		Output
A	B	Y= A+B
0	0	0
0	1	1
1	0	1
1	1	1

- **Realization of OR gate using diodes:**
- Two input OR gate using "diode-resistor" logic is shown in figure below. Where X, Y are the inputs and F is the output.

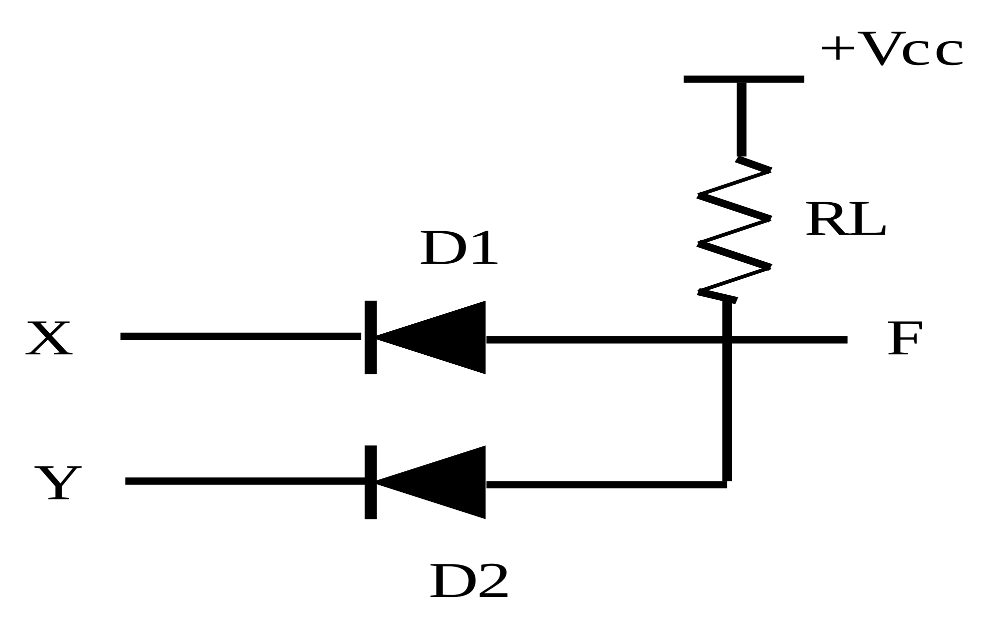


- **2. AND Gate:** The AND gate performs logical multiplication, commonly known as AND function. The AND gate has two or more inputs and a single output. The output of an AND gate is HIGH only when all the inputs are HIGH. Even if any one of the input is LOW, the output will be LOW. If a & b are input variables of an AND gate and c is its output, then Y=A.B

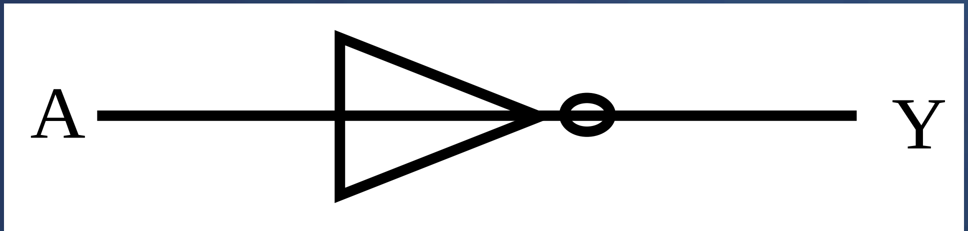


Input		Output
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

- **Realization of AND gate using diodes:**
- Two input AND gate using "diode-resistor" logic is shown in figure below. Where X, Y are inputs and F is the output.



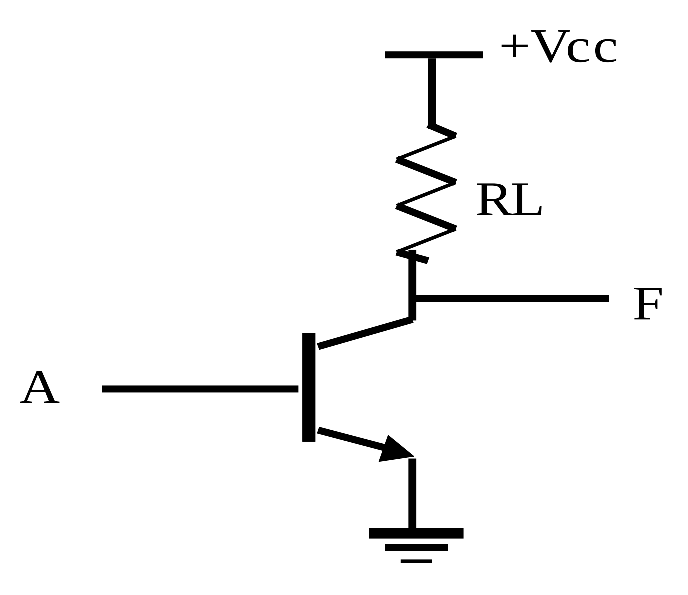
3. Not Gate (Inverter): The NOT gate performs the basic logical function called inversion or complementation. The purpose of his gate is to convert one logic level into the opposite logic level. It has one input and one output. When a HIGH level is applied to an inverter, a LOW level appears at the output and vice-versa.



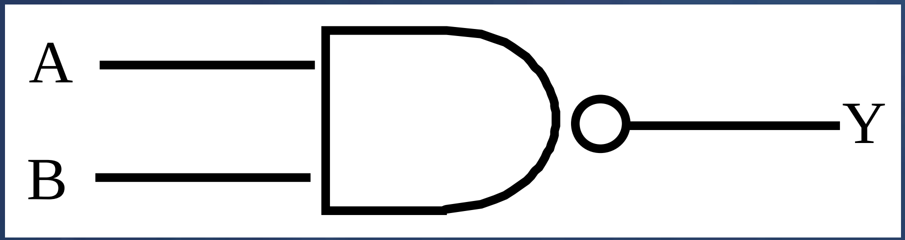
Truth Table:

Input	output
A	$Y = \overline{A}$
0	1
1	0

- **Realization of NOT gate using Transistors:**
- A NOT gate using a transistor is shown in below figure. ‘A’ represents the input and ‘F’ represents the output. When the input is HIGH, the transistor is in the ON state and the output is LOW. If the input is LOW, the transistor is in the OFF state and the output F is HIGH.



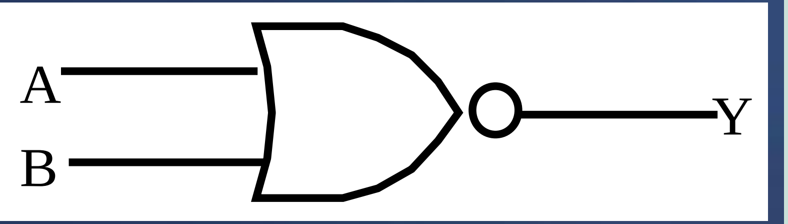
4.NAND Gate: The output of a NAND gate is LOW only when all inputs are HIGH and output of the NAND is HIGH if one or more inputs are LOW.



Truth Table:

Input		Output
A	B	$Y = \overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

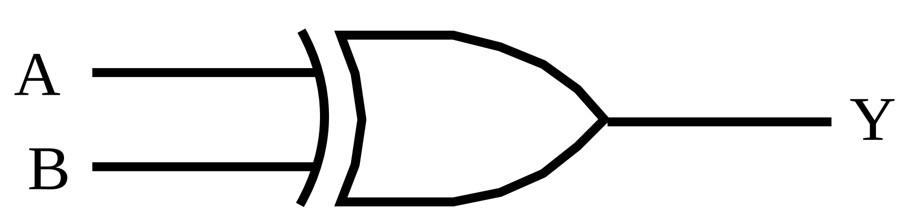
5. NOR Gate: The output of the NOR gate is HIGH only when all the inputs are LOW.



Truth Table:

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

6.XOR Gate or Exclusive OR gate: In this gate output is HIGH only when any one of the input is HIGH. The circuit is also called as inequality comparator, because it produces output when two inputs are different.

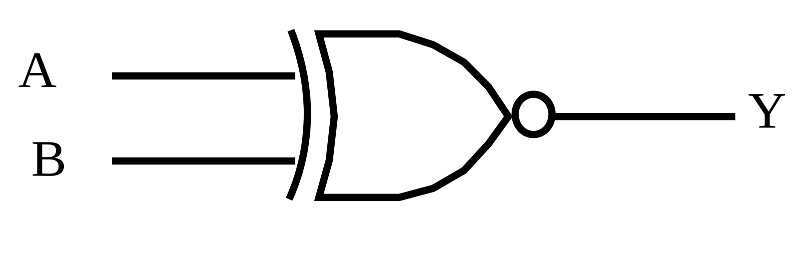


Truth Table:

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

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

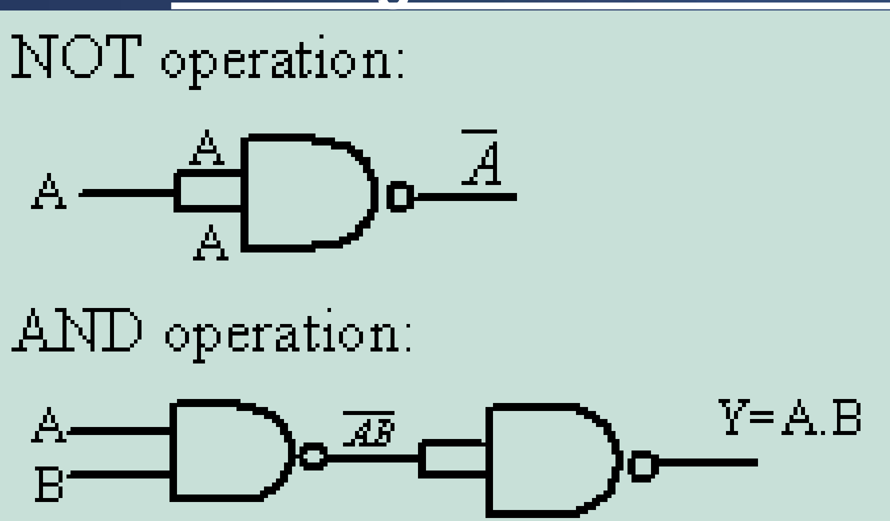
7. XNOR Gate or Exclusive NOR Gate: An XNOR gate is a gate with two or more inputs and one output. XNOR operation is complimentary of XOR operation. i.e. The output of XNOR gate is High, when all the inputs are identical; otherwise it is low.

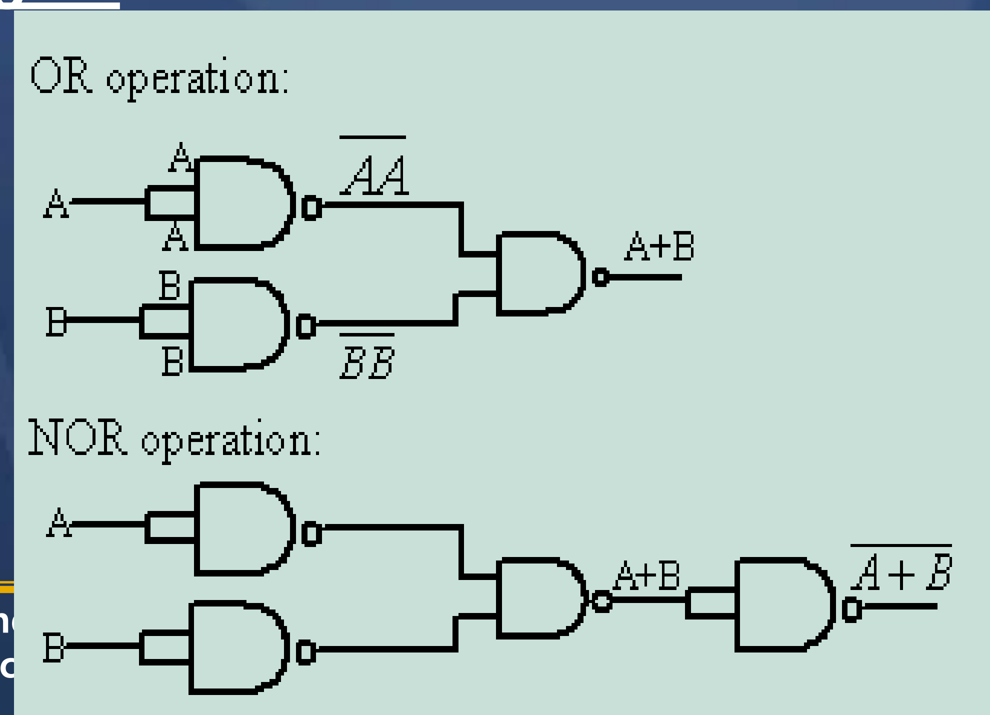


Truth Table:

Input		Output
A	B	$Y = \overline{A}\,\overline{B} + AB$
0	0	1
0	1	0
1	0	0
1	1	1

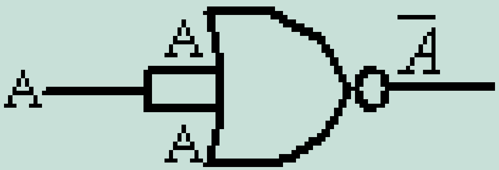
- **Universal Logic Gate:** NAND and NOR gates are called Universal gates or Universal building blocks, because both can be used to implement any gate like AND,OR an NOT gates or any combination of these basic gates.
- **<u>NAND gate as Universal gate</u>**



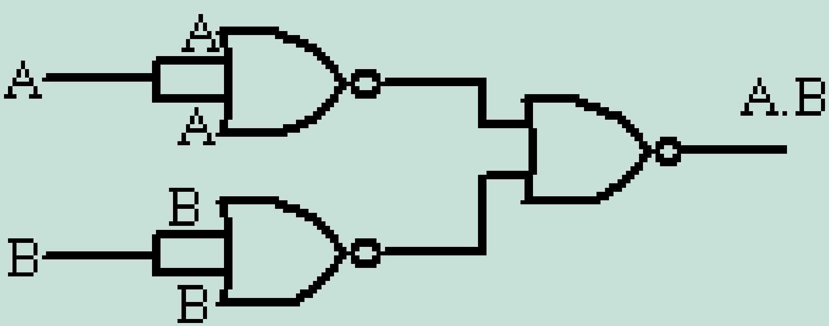


NOR gate as Universal gate:

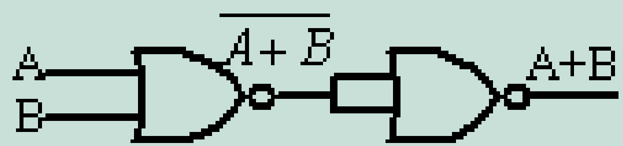
NOT operation:



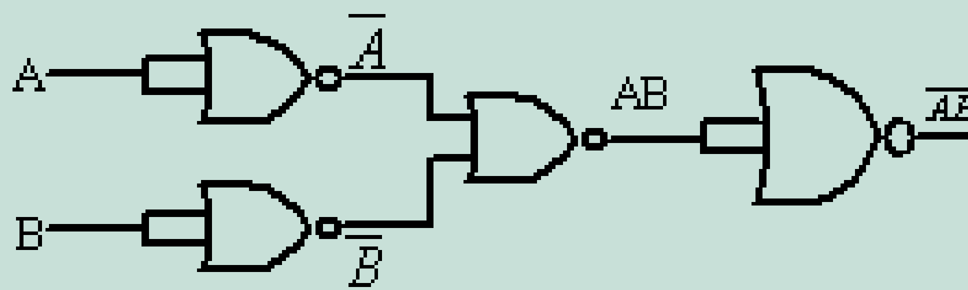
AND operation:



OR operation:

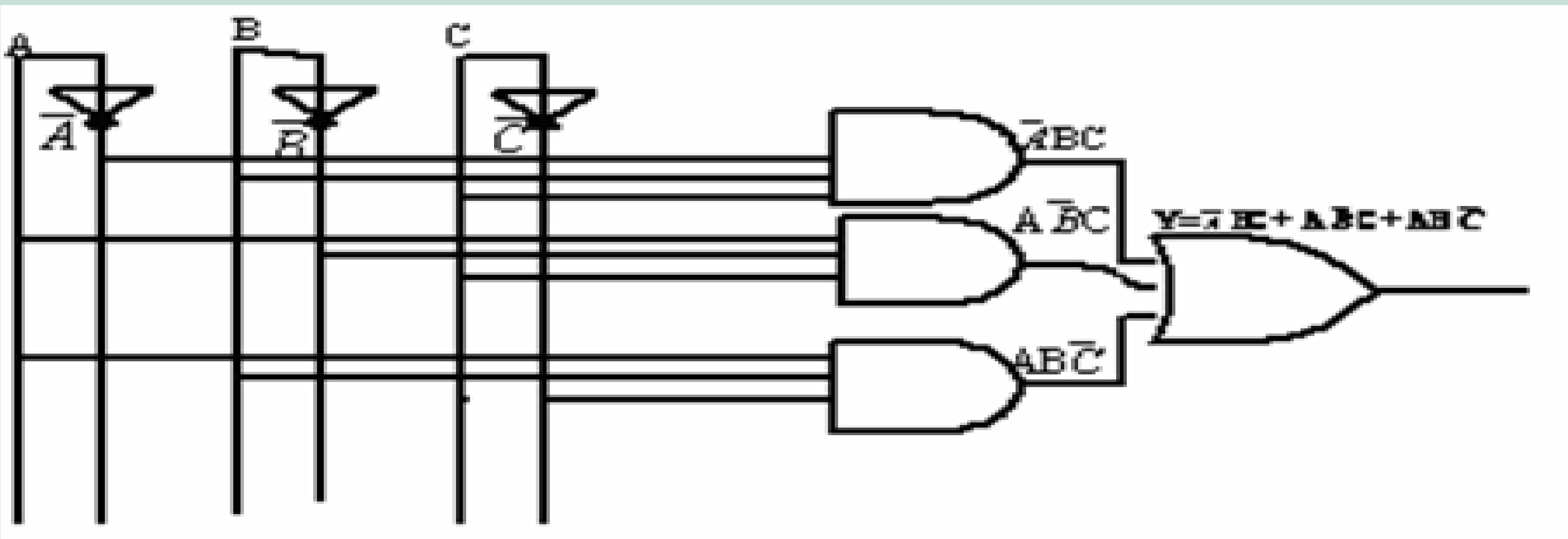


NAND operation:



Draw the logic circuit for the Boolean expression.

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



Digital Circuits

Types of Digital Circuits:

Basically digital circuits can be classified into two types.

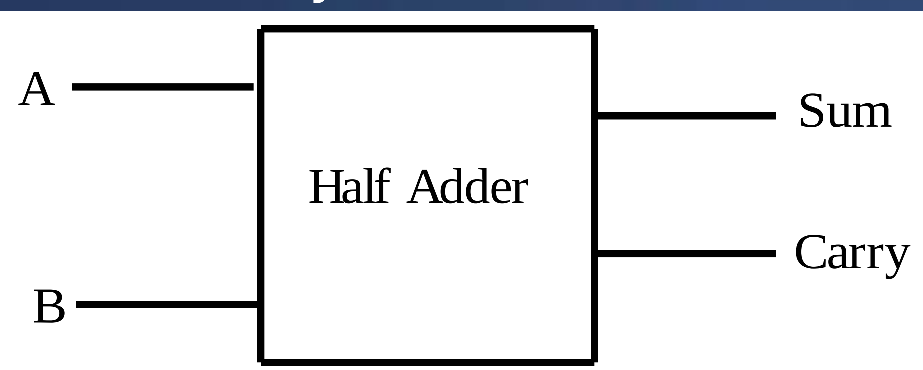
- **Sequential Digital Circuits:** The logic circuits whose output at any instant of time depend not only on the present input but also on the past outputs are called Sequential Circuits.

  In sequential circuits, the output signals are fedback to the input. Thus, an output signal is a function of present input signals and a sequences of the past input signal. i.e. the output signals.

- **Combinational Digital Circuits:** The logic circuits whose output at any instant of time are entirely dependent upon the input signals present at that time are known as combinational digital circuits.

  In particular, the output of the combinational circuit doesn't depend upon any past input or output So that the circuit doesn't possess any memory. The output signals of combinational circuits are not fedback to any other part of the circuit.

Half Adder: A combinational circuit which performs the arithmetic addition of two binary digits is called Half Adder. In the half adder circuit, there are two inputs, one is addend and augend and two outputs are Sum and Carry.



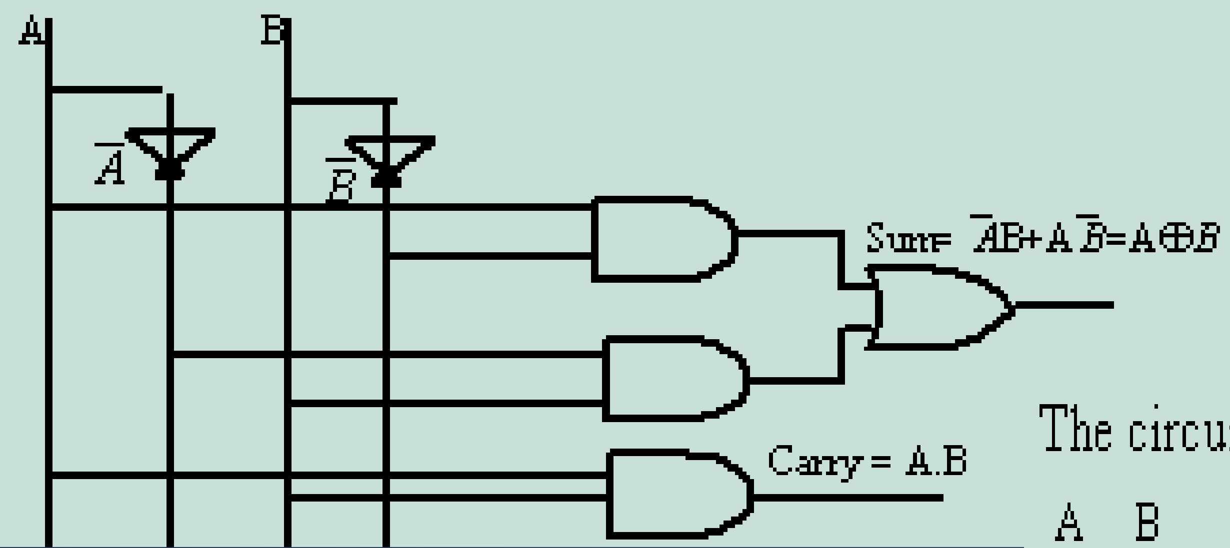
Truth Table for Half Adder

Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

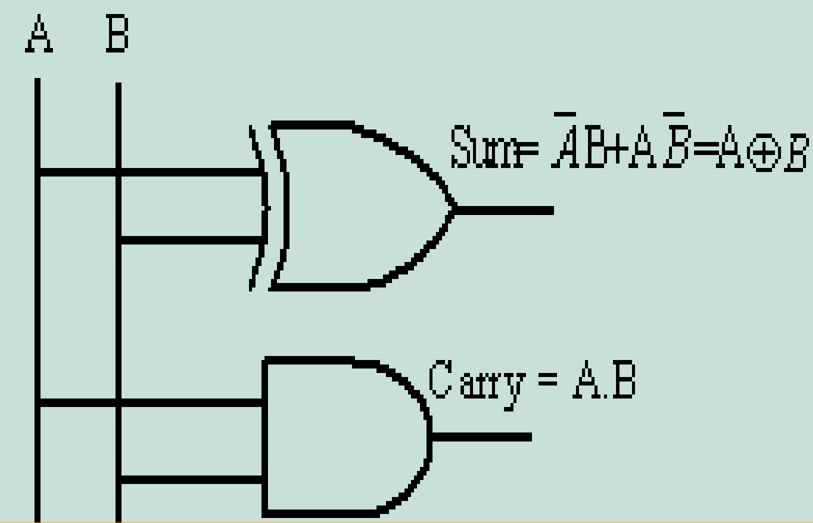
Sum= $\bar{A}B+A\bar{B}=A\oplus B$

Carry= $A.B$

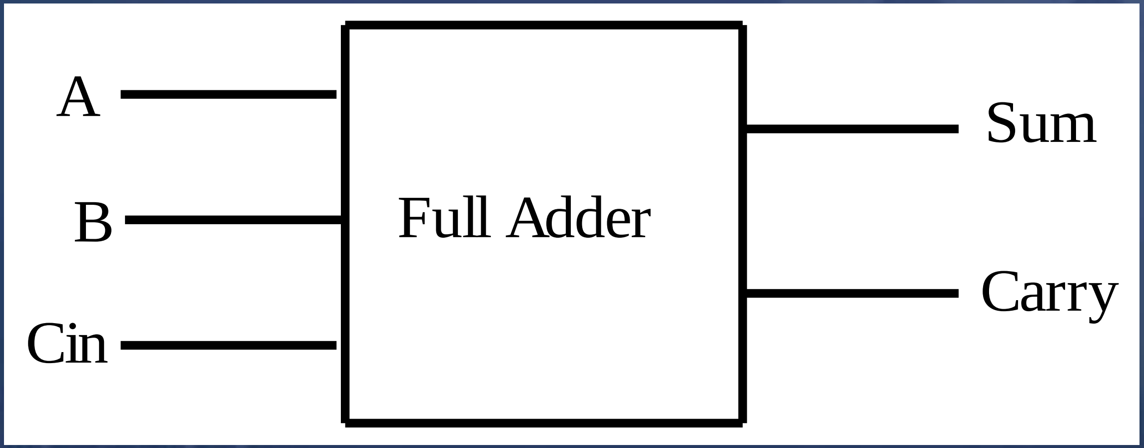
The circuit for Half Adder using Basic Gates is as follows:



The circuit for Half Adder using XOR gate:



- **Full Adder:** The full adder is a combinational circuit that performs the arithmetic sum of three input bits.
- It consists of three inputs and two outputs. Two of the inputs are variables, denoted by A and B, represent the two significant bit to be added The third input Cin represents carry form the previous lower significant position.



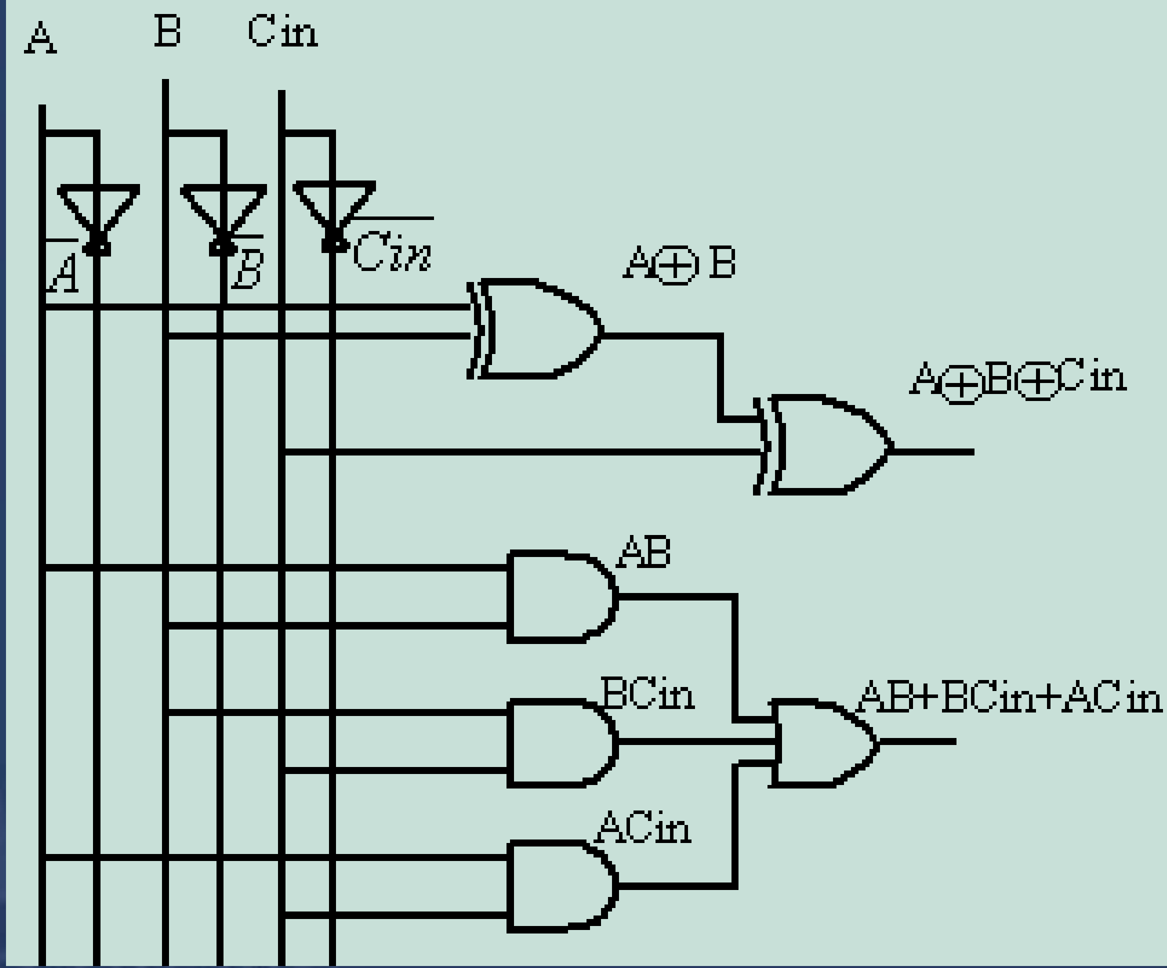
Truth Table for Full Adder

Input			Output	
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$
\begin{aligned}
\text{Sum} &= \bar{A}\,\bar{B}\,Cin + \bar{A}\,B\,\overline{Cin} + A\,\bar{B}\,\overline{Cin} + ABCin \\
&= \bar{A}\,[\,\bar{B}\,Cin + B\,\overline{Cin}\,] + A[\,\bar{B}\,\overline{Cin} + BCin] \\
&= \bar{A}\,[B \oplus Cin] + A[\,\overline{B \oplus Cin}\,] \\
&= A \oplus B \oplus Cin
\end{aligned}
$$

$$
\begin{aligned}
\text{Carry} &= \bar{A}BCin + A\bar{B}\,Cin + AB\,\overline{Cin} + ABCin \\
&= \bar{A}BCin + A\bar{B}\,Cin + AB(\,\overline{Cin} + Cin) \\
&= \bar{A}BCin + A\bar{B}\,Cin + AB \\
&= \bar{A}BCin + A(\,\bar{B}\,Cin + B) \\
&= \bar{A}BCin + AB + ACin \\
&= B(\,\bar{A}\,Cin + A) + ACin \\
&= B(A + Cin) + ACin \\
&= AB + BCin + ACin
\end{aligned}
$$

Implementation of Full Adder:



END