

Subject - Basic Electronics Engg. (104010)

FE - 2019 course

UNIT-III: Number System and Logic Gates

Syllabus

- **Number System:**
 - Binary, Octal, Decimal, Hexadecimal, BCD, their conversion and arithmetic
 - De-Morgan's theorems.
- **Logic Gates: Classification**
 - Basic gates: AND, OR, NOT
 - Universal gate: NAND, NOR
 - Other gates: XOR, XNOR
- **Combinational Design**
 - Half adder
 - Full adder
- **Flip Flop's:**
 - SR Latch, Flip Flops - SR, JK, T and D
- **Introduction to,**
 - Microprocessor (μ P) - Block diagram (Architecture) and explanation
 - Microcontroller (μ C) - Block diagram (Architecture) and explanation

Subject - Basic Electronics Engg. (104010)**FE - 2019 course****UNIT-III: Number System and Logic Gates**

#Number systems:	3
Explain the different number systems.....	3
# Code Conversions: Conversion of a Number from one number system to another	5
Convert the following numbers form one number system to another number systems.	5
# COMPLEMENT NUMBER	13
Explain the different complement number representations. Find the 1's complement & 2's.....	13
complement of a given number.....	13
# BINARY ARITHMETIC OPERATIONS:	14
Perform the different arithmetic operations on given binary numbers.....	14
#DeMorgan's Theorems:	18
State and Prove the DeMorgan's theorems.	18
#LOGIC GATES:	19
Explain the Logic gates. Give the classification of logic gates. Explain the NOR & NAND gates as	19
universal gates.....	19
#Adder Circuit	21
What are the different types of adders? Explain the Half adder and full adder	21
# Flip Flop	23
Explain the Flip Flop. What are the different types of Flip Flops? Explain the SR Latch. Explain the.....	23
different types of Flip Flops. Give the applications of Flip Flops.....	23
#Microprocessor & Microcontroller	27
Differentiate between Microprocessor and Microcontroller.....	27
Explain the architecture of Microprocessor	28
Explain the architecture of Microcontroller	28

#Number systems:**Explain the different number systems****Answer:****Introduction:**

In number system there are different symbols and each symbol has an absolute value and also has place value.

Radix or Base (R):

The radix or base of a number system is defined as the number of different digits which can occur in each position in the number system.

Radix Point:

The generalized form of a decimal point is known as radix point. In any positional number system the radix point divides the integer and fractional part.

$$N_r = [\text{Integer part} . \text{Fractional part}]$$

↑
Radix point / (Binary pt./Decimal pt./ Octal pt./Hex. pt.)

Number System:

In general a number in a system having base or radix 'r' can be written as

$$\underbrace{a_n a_{n-1} \dots a_0}_{\text{Integer Part}} \cdot \underbrace{a_{-1} a_{-2} \dots a_{-m}}_{\text{Fractional Part}}$$

↑
Dec. point

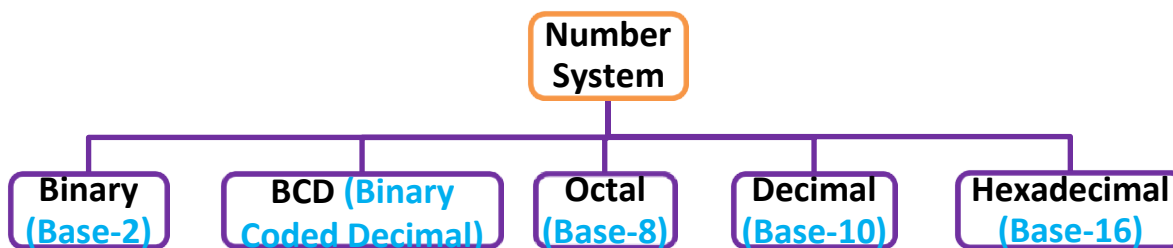
This will be interpreted as,

$$(N)_r = (a_n \times r^n + a_{n-1} \times r^{n-1} + \dots + a_0 \times r^0) \cdot (a_{-1} \times r^{-1} + a_{-2} \times r^{-2} + \dots + a_{-m} \times r^{-m})$$

Where, **N** = entire number (with integer & fractional part)

r = radix

a_n = the value of the nth position

#Types of Number Systems: (classification)

There are five types of number systems. They are

1. Decimal number system
2. Binary number system
3. Octal number system
4. Hexadecimal number system
5. Binary coded decimal (BCD)

Note: Except BCD, all the above number systems are **positional weighted systems**.

Number system	Base or Radix of the system	No. of digits/elements used	Digits/Elements/Symbols used in the system	Least element of the system	Largest element of the system
Binary (B)	2	2	0, 1	0	1
Octal (O)	8	8	0, 1, 2, 3, 4, 5, 6, 7	0	7
Decimal (D)	10	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0	9
Hexadecimal (H)	16	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	0	F

1. Decimal Number System: (Base = 10)

- The decimal number system contains ten unique symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- The value attached to the symbol depends on its location with respect to the decimal point.
- The decimal point separates the integer and fraction parts.

In general,

$$\underbrace{d_n d_{n-1} \dots d_0}_{\text{Integer Part}} \cdot \overset{\substack{\uparrow \\ \text{Dec.} \\ \text{point}}}{d_{-1} d_{-2} \dots d_{-m}}_{\text{Fractional Part}}$$

is given by,

$$(N)_{10} = (d_n \times 10^n + d_{n-1} \times 10^{n-1} + \dots d_0 \times 10^0) \cdot (d_{-1} \times 10^{-1} + d_{-2} \times 10^{-2} \dots d_{-m} \times 10^{-m})$$

For example:

$$(N)_{10} = (214.25)_{10} = (2 \times 10^2 + 1 \times 10^1 + 4 \times 10^0) \cdot (2 \times 10^{-1} + 5 \times 10^{-2})$$

2. Binary Number System: (Base = 2)

- The binary number system contains two unique symbols 0, 1 (bits).
- The value attached to the symbol depends on its location with respect to the binary point.
- The binary point separates the integer and fraction parts.

In general,

$$b_n b_{n-1} \dots b_0 \cdot b_{-1} b_{-2} \dots b_{-m}$$

3. Octal Number System: (Base = 8)

- The octal number system contains eight unique symbols 0, 1, 2, 3, 4, 5, 6, 7.
- The value attached to the symbol depends on its location with respect to the octal point.
- The octal point separates the integer and fraction parts.
- The octal number is 3-bit binary equivalent. ($8 = 2^3$)

In general,

$$O_n O_{n-1} \dots O_0 \cdot O_{-1} O_{-2} \dots O_{-m}$$

4. Hexadecimal Number System: (Base = 16)

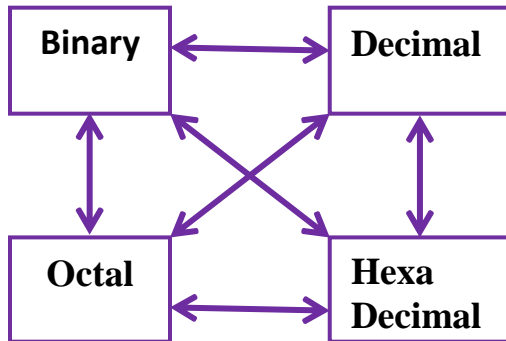
- The hexadecimal number system contains 16 unique symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- The value attached to the symbol depends on its location with respect to the octal point.

- The hexadecimal point separates the integer and fraction parts.
- The hexadecimal number is 4-bit binary equivalent. ($16 = 2^4$)

In general,

$$H_n H_{n-1} \dots H_0 \cdot H_{-1} H_{-2} \dots H_{-m}$$

Code Conversions: Conversion of a Number from one number system to another



1. Binary \leftrightarrow Decimal
2. Octal \leftrightarrow Decimal
3. Hexadecimal \leftrightarrow Decimal
4. Binary \leftrightarrow Octal
5. Binary \leftrightarrow Hexadecimal
6. Octal \leftrightarrow Hexadecimal

Convert the following numbers from one number system to another number systems.

Answer:

1. Binary to Decimal conversion:

In this method, each binary digit of the number is multiplied by its positional weight and the product terms are added to obtain decimal number.

i) Convert $(10101.1101)_2$ to decimal.

Solution: $(10101.1101)_2 = (?)_{10}$

Given Binary	1	0	1	0	1	.	1	1	0	1
	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
Position of bits	4	3	2	1	0	Binary point	-1	-2	-3	-4

$$\begin{aligned}
 (10101.1101)_2 &= (1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) \cdot (1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}) \\
 &= (1 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1) + (1 \times 0.5 + 1 \times 0.25 + 0 \times 0.125 + 1 \times 0.0625) \\
 &= (16 + 0 + 4 + 0 + 1) + (0.5 + 0.25 + 0 + 0.0625) \\
 &= (21.8125)_{10}
 \end{aligned}$$

$$(10101.1101)_2 = (21.8125)_{10}$$

ii) Convert $(1011.101)_2$ to decimal.

Solution: $(1011.101)_2 = (?)_{10}$

Given Binary	1	0	1	1	.	1	0	1
	↑	↑	↑	↑	↑	↑	↑	↑
Position of bits	3	2	1	0	Binary point	-1	-2	-3

$$\begin{aligned}
 (1011.101)_2 &= (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0) \cdot (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}) \\
 &= (1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1) + (1 \times 0.5 + 0 \times 0.25 + 1 \times 0.125) \\
 &= (8 + 0 + 2 + 1) + (0.5 + 0 + 0.125) \\
 &= (11.625)_{10}
 \end{aligned}$$

$$(1011.101)_2 = (11.625)_{10}$$

2. Octal to Decimal conversion:

In this method, each octal digit of the number is multiplied by its positional weight and the product terms are added to obtain decimal number.

i) Convert $(127.25)_8$ to decimal.

Solution: $(127.25)_8 = (?)_{10}$

Given Octal	1	2	7	.	2	5
	↑	↑	↑	↑	↑	↑
Position of digits	2	1	0	Octal point	-1	-2

$$\begin{aligned}
 (127.25)_8 &= (1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0) \cdot (2 \times 8^{-1} + 5 \times 8^{-2}) \\
 &= (1 \times 64 + 2 \times 8 + 7 \times 1) + (2 \times 0.125 + 5 \times 0.015625) \\
 &= (64 + 16 + 7) + (0.25 + 0.078) \\
 &= (87.328)_{10}
 \end{aligned}$$

$$(127.25)_8 = (87.328)_{10}$$

3. Hexadecimal to Decimal conversion:

In this method, each hexadecimal digit of the number is multiplied by its positional weight and the product terms are added to obtain decimal number.

i) Convert $(B89A.03)_{16}$ to decimal.

Solution: $(B89A.03)_{16} = (?)_{10}$

Given Hexadecimal	B	8	9	A	.	0	3
Decimal Equivalent	11	8	9	10		0	3
	↑	↑	↑	↑	↑	↑	↑
Position of digits	3	2	1	0	Hex. point	-1	-2

$$\begin{aligned}
 (1C5.0A)_{16} &= (11 \times 16^3 + 8 \times 16^2 + 9 \times 16^1 + 10 \times 16^0) \cdot (0 \times 16^{-1} + 3 \times 16^{-2}) \\
 &= (11 \times 4096 + 8 \times 256 + 9 \times 16 + 10 \times 1) + (0 + 3 \times 0.625) \\
 &= (45056 + 2048 + 144 + 10) + (1.875) \\
 &= (47129.875)_{10}
 \end{aligned}$$

$$(B89A.03)_{16} = (47129.875)_{10}$$

ii) Convert $(1C5.0A)_{16}$ to decimal.

Solution: $(1C5.0A)_{16} = (?)_{10}$

Given Hexadecimal	1	C	5	.	0	A
Decimal Equivalent	1	12	5		0	10
	↑	↑	↑	↑	↑	↑
Position of digits	2	1	0	Hex. point	-1	-2

$$\begin{aligned}
 (1C5.0A)_{16} &= (1 \times 16^2 + 12 \times 16^1 + 5 \times 16^0) \cdot (0 \times 16^{-1} + 10 \times 16^{-2}) \\
 &= (1 \times 256 + 12 \times 16 + 5 \times 1) + (0 + 10 \times 0.625) \\
 &= (256 + 192 + 5) + (0 + 6.25) \\
 &= (459.25)_{10}
 \end{aligned}$$

$$(1C5.0A)_{16} = (459.25)_{10}$$

4. Binary to Octal conversion

For conversion binary to octal the binary numbers are divided into groups of 3 bits each, starting at the binary point and proceeding towards left and right.

i) Convert $(10\ 101\ 111\ 001 . 011\ 1)_2$ to Octal

Solution: $(10\ 101\ 111\ 001 . 011\ 1)_2 = (?)_8$

Binary Number given	10	101	111	001	.	011	1
Group of 3 bits (*)	<u>010</u>	<u>101</u>	<u>111</u>	<u>001</u>	.	<u>011</u>	<u>100</u>
Convert each group into octal	2	5	7	1	.	3	4

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

(*Note: if number of bits in leftmost or rightmost group will be less than of 3, then add required number of 0's to form a group of 3-bits.)

$$(10\ 101\ 111\ 001 . 011\ 1)_2 = (2571.34)_8$$

5. Binary to Hexadecimal conversion

For conversion binary to Hexadecimal the binary numbers are divided into groups of 4 bits each, starting at the binary point and proceeding towards left and right.

i) Convert $(10\ 1111\ 1011 . 0111\ 11)_2$ to Hexadecimal

Solution: $(10\ 1111\ 1011 . 0111\ 11)_2 = (?)_{16}$

Binary Number given	10	1111	1011	.	0111	11
Group of 4 bits (*)	<u>0010</u>	<u>1111</u>	<u>1011</u>	.	<u>0111</u>	<u>1100</u>
Convert each group into Hexadecimal	2	F	B	.	7	C

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

(*Note: if number of bits in leftmost or rightmost group will be less than of 4, then add required number of 0's to form a group of 4-bits.)

$$(10\ 1111\ 1011 . 0111\ 11)_2 = (2FB.7C)_{16}$$

6. Decimal to Binary conversion:

In the conversion the integer number are converted to the desired base using successive division by the base or radix i.e. 2.

For Integer part: (successive division by 2)

Divide the given decimal number successively by 2 read the integer part remainder upwards to get equivalent binary number.

For Fraction part: (successive multiplication by 2)

Multiply the fraction part by 2. Keep the integer in the product as it is and multiply the new fraction in the product by 2. The process is continued and the integers are read in the products from top to bottom.

i) Convert $(105.15)_{10}$ into binary.

Solution: $(105.15)_{10} = (?)_2$

Integer Part			Fraction Part		
2	105	Remainder		Multiplication	Remainder
2	52	1	0.15 x 2	0.30	0
2	26	0	0.30 x 2	0.60	0
2	13	0	0.60 x 2	1.20	1
2	6	1	0.20 x 2	0.40	0
2	3	0	0.40 x 2	0.80	0
2	1	1	0.80 x 2	1.60	1
	0	1	0.60 x 2	1.20	1
$(105)_{10} = (1101001)_2$			$(0.15)_{10} = (0.0010011 \dots)_2$		

$$(105.15)_{10} = (1101001.0010011)_2$$

7. Decimal to Octal conversion:

In the conversion the integer number are converted to the desired base using successive division by the base or radix i.e. 8.

For Integer part: (successive division by 8)

Divide the given decimal number successively by 8 read the integer part remainder upwards to get equivalent binary number.

For Fraction part: (successive multiplication by 8)

Multiply the fraction part by 8. Keep the integer in the product as it is and multiply the new fraction in the product by 8. The process is continued and the integers are read in the products from top to bottom.

i) Convert $(105.15)_{10}$ into octal.

Solution: $(105.15)_{10} = (?)_8$

Integer Part			Fraction Part		
8	105	Remainder		Multiplication	Remainder
8	13	1	0.15 x 8	1.20	1
8	1	5	0.20 x 8	1.60	1
	0	1	0.60 x 8	4.80	4
$(105)_{10} = (151)_8$			0.80 x 8	6.40	6
			0.40 x 8	3.20	3
			0.20 x 8	1.60	1
			$(0.15)_{10} = (0.114631 \dots)_8$		

$$(105.15)_{10} = (151.114631 \dots)_8$$

8. Decimal to Hexadecimal conversion:

In the conversion the integer number are converted to the desired base using successive division by the base or radix i.e. 16.

For Integer part: (successive division by 16)

Divide the given decimal number successively by 16 read the integer part remainder upwards to get equivalent binary number.

For Fraction part: (successive multiplication by 16)

Multiply the fraction part by 16. Keep the integer in the product as it is and multiply the new fraction in the product by 16. The process is continued and the integers are read in the products from top to bottom.

i) Convert $(432.14)_{10}$ into hexadecimal.

Solution: $(432.14)_{10} = (?)_{16}$

Integer Part				Fraction Part		
16	432	Remainder			Multiplication	Remainder
		Decimal Hex.				
16	27	0	0	0.14 x 16	2.24	2
16	1	11=>	B	0.24 x 16	3.84	3
	0	1	1	0.84 x 16	13.44	13=>D
				0.44 x 16	7.04	7
				0.04 x 16	0.64	0
$(432)_{10} = (1B0)_{16}$				$(0.14)_{10} = (0.23D70 \dots)_{16}$		

$$(432.14)_{10} = (1B0.23D70 \dots)_{16}$$

ii) Convert $(105.15)_{10}$ into hexadecimal.

Solution: $(105.15)_{10} = (?)_{16}$

Integer Part				Fraction Part		
16	108	Remainder			Multiplication	Re ainder
		Decimal Hex				
16	6	12	C	0.15 x 16	2.40	2
	0	6	6	0.40 x 16	6.4	6
				0.40 x 16	6.40	6
$(108)_{10} = (C6)_{16}$				$(0.15)_{10} = (0.266 \dots)_{16}$		

$$(108.15)_{10} = (C6.266 \dots)_{16}$$

9. Octal to Binary conversion:

To convert a given octal number to binary, replace each octal digit by its 3- bit binary equivalent

i) Convert $(2572.34)_8$ to Binary

Solution: $(2572.34)_8 = (?)_2$

Given Octal 2 5 7 2 . 3 4
 3-bit binary equivalent 010 101 111 010 . 011 100
 of each digit
 By skipping leftmost & 10 101 111 010 . 011 1
 rightmost 0's
 $(2572.34)_8 = (10 101 111 010 . 011 1)_2$

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

10. Hexadecimal to Binary conversion:

To convert a given hexadecimal number to binary, replace each octal digit by its 4- bit binary equivalent.

i) Convert $(2FB.7C)_{16}$ to Binary

Solution: $(2FB.7C)_{16} = (?)_2$

Given Hexadecimal 2 F B . 7 C
 4-bit binary equivalent of 0010 1111 1011 . 0111 1100
 each digits
 By skipping leftmost & 10 1111 1011 . 0111 11
 rightmost 0's
 $(2FB.7C)_{16} = (10 1111 1011 . 0111 11)_2$

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

11. Octal to Hexadecimal conversion (“Octal to Binary” then “Binary to Hexadecimal”)

To convert a given octal number to binary, replace each octal digit by its 3- bit binary equivalent

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

i) Convert $(462.27)_8$ to hexadecimal

Solution: $(462.27)_8 = (?)_{16}$

Given octal digits	4	6	2	.	2	7
3-bit binary equivalent of each digit	<u>100</u>	<u>110</u>	<u>010</u>	.	<u>010</u>	<u>111</u>
Rearranging in group of 4 bits	1	0011	0010	.	0101	11
Add 0's to left or right (if required)	0001	0011	0010	.	0101	1100
4-bit hexadecimal equivalent	1	3	2	.	5	C

$(462.27)_8 = (132.5C)_{16}$

ii) Convert $(2572.34)_8$ to hexadecimal

Solution: $(2572.34)_8 = (?)_{16}$

Given octal digits	2	5	7	2	.	3	4
3-bit binary equivalent of each digit	<u>010</u>	<u>101</u>	<u>111</u>	<u>010</u>	.	<u>011</u>	<u>100</u>
Rearranging in group of 4 bits	0101	0111	1010	.	0111	0000	
4-bit hexadecimal equivalent	5	7	A	.	7	0	

$(2572.34)_8 = (57A.7)_{16}$

12. Hexadecimal to Octal conversion (“Hexadecimal to Binary” then “Binary to Octal”)

To convert a given hexadecimal number to binary, replace each octal digit by its 4- bit binary equivalent

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

i) Convert $(2BA.0C)_{16}$ to Octal

Solution: $(2BA.0C)_{16} = (?)_8$

Given hexadecimal digits	2	B	A	.	0	C
4-bit binary equivalent of each digit	0010	1011	1010	.	0000	1100
Rearranging in group of 3 bits	001	010	111	010	.	000 011
3-bit octal equivalent	1	2	7	2	.	0 3

$(2BA.0C)_{16} = (1272.03)_8$

ii) Convert $(57A.7)_{16}$ to Octal

Solution: $(57A.7)_{16} = (?)_8$

Given hexadecimal digits	5	7	A	.	7	
4-bit binary equivalent of each digit	0101	0111	1010	.	0111	
Rearranging in group of 3 bits	010	101	111	010	.	011 100
3-bit octal equivalent	2	5	7	2	.	3 4

$(57A.7)_{16} = (2572.34)_8$

#Binary Coded Decimal (BCD) or 8421 code:

BCD is a weighted code. In weighted codes, each successive digit from right to left represents weights equal to some specified value and to get the equivalent decimal number add the products of the weights by the corresponding binary digit. **8421** is the most common because 8421 BCD is the most natural amongst the other possible codes.

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Example 1: Find the BCD of $(417.25)_{10}$

Solution: $(417.25)_{10} = (?)_{BCD}$

Decimal Number	→	4	1	7	.	2	5
BCD or 8421 code	→	0100	0001	0111	Dec. point	0010	0101

$(417.25)_{10} = (0100\ 0001\ 0111 \cdot 0010\ 0101)_{BCD}$

Example 2: Find the BCD of $(567)_{10}$

Solution: $(567)_{10} = (?)_{BCD}$

Decimal Number → 5 6 7
 BCD or 8421 code → 0101 0110 0111
 $(567)_{10} = (0101\ 0110\ 0111)_{\text{BCD}}$

COMPLEMENT NUMBER:

Explain the different complement number representations. Find the 1's complement & 2's complement of a given number.

Answer:

- Complement number system is used for representation of negative numbers only.
- Negative numbers in radix-r system can be represented using,
 - r's (Radix's) complement &
 - (r-1)'s [(Radix minus 1)'s] complement.
- Negative binary numbers can be represented using following complements systems,
 - **1's complement representation**
 - **2's complement representation**

1's Complement Representation:

The 1's complement of a binary number is obtained by replacing 0's by 1's & 1's by 0's.

Example: Find 1's complement of $(11001)_2$

Solution:

Given binary number => 1 1 0 0 1
 1's complement of given binary number => 0 0 1 1 0

1's complement of $(11001)_2$ is $(00110)_2$

2's Complement Representation:

The 2's complement of a binary number is a binary number which is obtained by adding 1 to the 1's complement of a number i.e.

$2's\ complement = 1's\ complement + 1$

Example: Find 2's complement of $(11001)_2$

Solution:

Given binary number => 1 1 0 0 1
 1's complement of given binary number => 0 0 1 1 0
 Add 1 in 1's complement + 1
 2's complement of given binary number 0 0 1 1 1

2's complement of $(11001)_2$ is $(00111)_2$

BINARY ARITHMETIC OPERATIONS:

Perform the different arithmetic operations on given binary numbers.

Answer:

1. Binary Addition:

The binary addition rules are as follows

Inputs (Bits)		Outputs of addition		Operation
A	B	Sum	Carry	
0	0	0	0	$0 + 0 = 0\ 0$
0	1	1	0	$0 + 1 = 1\ 0$
1	0	1	0	$1 + 0 = 1\ 0$
1	1	0	1	$1 + 1 = 0\ 1$

Carry is generated if both inputs are 1.

(**Note:** if carry is generated after addition of bits forward that carry for next addition)

Example 1: $(111011.11)_2 + (100100.01)_2 = (?)_2$

Solution:

$$\begin{array}{r}
 \text{Carry (generated by adding bits)} \quad 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\
 \text{1st Binary Number} \quad 1\ 1\ 1\ 0\ 1\ 1\ .\ 1\ 1 \\
 \text{2nd Binary Number} \quad + \quad \underline{1\ 0\ 0\ 1\ 0\ 0\ .\ 0\ 1} \\
 \text{Result} \quad 1\ 1\ 0\ 0\ 0\ 0\ 0\ .\ 0\ 0
 \end{array}$$

$$(111011.11)_2 + (100100.01)_2 = (1100000.00)_2$$

Example 2: $(1101.101)_2 + (0101.100)_2 = (?)_2$

Solution:

$$\begin{array}{r}
 \text{Carry (generated by adding bits)} \quad 1\ 1\ 1\ 1 \\
 \text{1st Binary Number} \quad 1\ 1\ 0\ 1\ .\ 1\ 0\ 1 \\
 \text{2nd Binary Number} \quad + \quad \underline{0\ 1\ 0\ 1\ .\ 1\ 0\ 0} \\
 \text{Result} \quad 1\ 0\ 0\ 1\ 1\ .\ 0\ 0\ 1
 \end{array}$$

$$(1101.101)_2 + (0101.100)_2 = (10011.001)_2$$

Example 3: $(1000011)_2 + (1110001)_2 = (?)_2$

Solution:

$$\begin{array}{r}
 \text{Carry (generated by adding bits)} \quad 1\ 1\ 1 \\
 \text{1st Binary Number} \quad 1\ 0\ 0\ 0\ 0\ 1\ 1 \\
 \text{2nd Binary Number} \quad + \quad \underline{1\ 1\ 1\ 0\ 0\ 0\ 1} \\
 \text{Result} \quad 1\ 0\ 1\ 1\ 0\ 1\ 0
 \end{array}$$

$$(1000011)_2 + (1110001)_2 = (10110100)_2$$

2. Binary Subtraction:

a. Binary Subtraction: (Regular Method)

The binary subtraction rules are as follows

Inputs (Bits)		Outputs of addition		Operation
A	B	Difference	Borrow	
0	0	0	0	$0 - 0 = 0\ 0$
0	1	1	1	$0 - 1 = 1\ 1$
1	0	1	0	$1 - 0 = 1\ 0$
1	1	0	0	$1 - 1 = 0\ 0$

Borrow is generated if first bit is 0 & second is 1.

(Note: if borrow is generated after subtraction of bits forward that borrow for next subtraction)

Example 1: $(11101)_2 - (10011)_2 = (?)_2$

Solution:

$$\begin{array}{r}
 \text{Borrow (generated by 0-1)} \quad \quad \quad 1 \\
 1^{\text{st}} \text{ Binary Number} \quad \quad 1\ 1\ 1\ 0\ 1 \\
 2^{\text{nd}} \text{ Binary Number} \quad - \quad 1\ 0\ 0\ 1\ 1 \\
 \text{Subtract Borrow} \quad - \quad \underline{1} \\
 \text{Result} \quad \quad \quad 0\ 1\ 0\ 1\ 0
 \end{array}$$

$$(11101)_2 - (10011)_2 = (01010)_2$$

Example 2: $(1011.010)_2 - (0110.101)_2 = (?)_2$

Solution:

$$\begin{array}{r}
 \text{Borrow (generated by 0-1)} \quad \quad \quad 1 \quad \quad 1 \quad 1 \\
 1^{\text{st}} \text{ Binary Number} \quad \quad 1\ 0\ 1\ 1 \cdot 0\ 1\ 0 \\
 2^{\text{nd}} \text{ Binary Number} \quad - \quad 0\ 1\ 1\ 0 \cdot 1\ 0\ 1 \\
 \text{Subtract Borrow} \quad - \quad \underline{1 \quad 1 \quad 1} \\
 \text{Result} \quad \quad \quad 0\ 1\ 0\ 0 \cdot 1\ 0\ 1
 \end{array}$$

$$(1011.010)_2 - (0110.101)_2 = (0100.101)_2$$

b. Binary Subtraction: (using 1's Complement Method)

Example 1: $(11101)_2 - (10011)_2 = (?)_2$

Solution:

Step-I: Find 1's complement of -ve number by replacing 1's by 0's & 0's by 1's.

\therefore 1's complement of (10011) is (01100)

Step-II: Add 1's complement result in 1st number

$$\begin{array}{r}
 \text{Carry (generated by adding bits)} \quad \quad \quad 1\ 1\ 1 \\
 1^{\text{st}} \text{ Binary Number} \quad \quad \quad 1\ 1\ 1\ 0\ 1 \\
 1^{\text{'s}} \text{ complement of } 2^{\text{nd}} \text{ Binary Number} \quad + \quad \underline{0\ 1\ 1\ 0\ 0} \\
 \text{Result} \quad \quad \quad 1\ 0\ 1\ 0\ 0\ 1
 \end{array}$$

Step-III: If finally carry is generated add that carry in sum

$$\begin{array}{r}
 \text{Carry (generated by adding bits)} \quad \quad \quad 1 \\
 \text{Result of previous addition} \quad \quad \quad 0\ 1\ 0\ 0\ 1 \\
 \text{Add final carry} \quad + \quad \underline{1} \\
 \quad \quad \quad 0\ 1\ 0\ 1\ 0
 \end{array}$$

$$(11101)_2 - (10011)_2 = (01010)_2$$

c. Binary Subtraction: (using 2's Complement Method)

Step-I: Find 2's complement of $-ve$ number by (1's complement + 1)

Step-II: Add 2's complement (i.e. result of step-I) in 1st number

Step-III: If finally carry is generated discard or ignore that carry & remaining will be result.

Step-IV: If MSB of result is 0, then result is positive.

If MSB of result is 1, then result is negative. Again find the 2's complement of result.

Example 1: $(11101)_2 - (10011)_2 = (?)_2$

Solution:

Step-I: Find 2's complement of $-ve$ number by (1's complement + 1)

\therefore 1's complement of (10011) is (01100)

Add 1 in 1's complement: $01100 + 1 = 01101$

Step-II: Add 2's complement result in 1st number

Carry (generated by adding bits)		1	1	1		1
1 st Binary Number		1	1	1	0	1
1's complement of 2 nd Binary Number	+	0	1	1	0	1
Result		1	0	1	0	1

Step-III: If finally carry is generated discard or ignore that carry & remaining will be result.

$$(11101)_2 - (10011)_2 = (01010)_2$$

Step-IV: MSB of result is 0, hence result is positive.

$$(11101)_2 - (10011)_2 = (01010)_2$$

Example 2: $(11011)_2 - (100000)_2 = (?)_2$

Solution:

Step-I: Find 2's complement of $-ve$ number by (1's complement + 1)

\therefore 1's complement of (100000) is (011111)

Add 1 in 1's complement: $011111 + 1 = 100000$

Step-II: Add 2's complement result in 1st number

Carry (generated by adding bits)		1	1	0	1	1
1 st Binary Number		1	1	0	1	1
1's complement of 2 nd Binary Number	+	1	0	0	0	0
Result		1	1	1	0	1

Step-III: If finally carry is generated discard or ignore that carry & remaining will be result.

$$(11011)_2 - (100000)_2 = (111011)_2$$

Step-IV: MSB of result is 1, hence result is negative.

Find 2's complement of $-ve$ number by (1's complement + 1)

\therefore 1's complement of (111011) is (000100)

Add 1 in 1's complement: $000100 + 1 = 000101$.

$$(11011)_2 - (100000)_2 = (000101)_2$$

3. Binary Multiplication:

Example 1: $(1101)_2 \times (110)_2 = (?)_2$

Solution:

1st Binary Number	1 1 0 1
2nd Binary Number X	1 1 1
	1 1 0 1
1 st Number x 1	1 1 0 1
1 st Number x 0	1 1 0 1 x
1 st Number x 1	1 1 0 1 x x
Carry	1 1 1 1
Result	1 0 1 1 0 1 1

$$(1101)_2 \times (110)_2 = (1011011)_2$$

Example 2: $(1011)_2 \times (0101)_2 = (?)_2$

Solution:

1st Binary Number	1 0 1 1
2nd Binary Number X	0 1 0 1
	1 0 1 1
1 st Number x 1	1 0 1 1
1 st Number x 0	0 0 0 0 x
1 st Number x 1	1 0 1 1 x x
1 st Number x 0	0 0 0 0 x x x
Result	0 1 1 0 1 1 1

$$(1011)_2 \times (0101)_2 = (0110111)_2$$

4. Binary Division:

Example 1: $(110011)_2 \div (101)_2 = (?)_2$

Solution:

	1 0 1 0 . 0 0 0 1
101	1 1 0 0 1 1
	- 1 0 1
	0 0 1 0 1
	- 1 0 1
	0 0 0 1 0 0 0
	- 1 0 1
	0 0 1 1

$$(110011)_2 \div (101)_2 = (1010.0001)_2$$

Example 2: $(1001)_2 \div (10)_2 = (?)_2$

Solution:

	1 0 0 . 1
10	1 0 0 1
	- 1 0
	0 0 0 1 0
	- 1 0
	0 0

$$(1001)_2 \div (10)_2 = (100.1)_2$$

#DeMorgan's Theorems:

State and Prove the DeMorgan's theorems.

Answer:

#DeMorgan's Theorem:

DeMorgan's states two theorems:

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

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

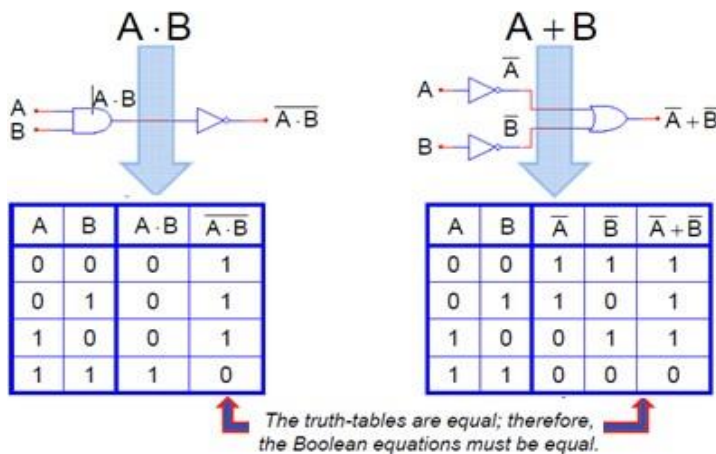
1. DeMorgan's First Theorems:

Statement: Complement of AND is logically equivalent to OR of individual complement.

Logical Expression: $\overline{A \cdot B} = \overline{A} + \overline{B}$

Proof:

Diagram:



Truth Table:

1	2	3	4	5	6	7
A	B	$A \cdot B$	$\overline{A \cdot B}$	\overline{A}	\overline{B}	$\overline{A} + \overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

Entries in 4th & 7th columns are identical, hence DeMorgan's first theorem is proved

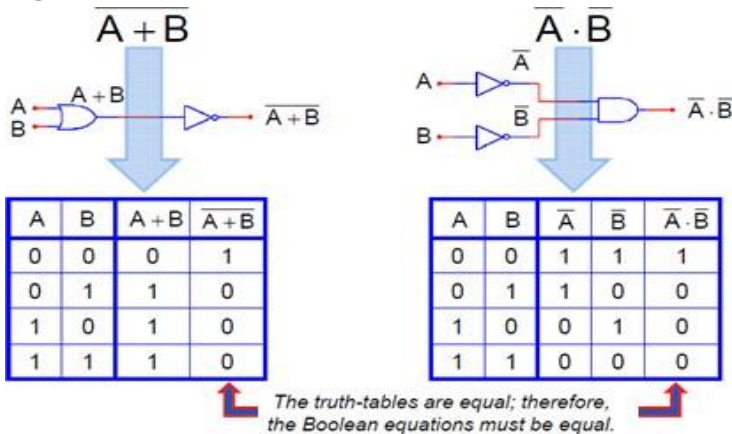
2. DeMorgan's Second Theorems:

Statement: Complement of OR is logically equivalent to AND of individual complement.

Logical Expression: $\overline{A + B} = \overline{A} \cdot \overline{B}$

Proof:

Diagram:



Truth Table:

1	2	3	4	5	6	7
A	B	$A + B$	$\overline{A + B}$	\overline{A}	\overline{B}	$\overline{A} \cdot \overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Entries in 4th & 7th columns are identical, hence DeMorgan's second theorem is proved

Note: DeMorgan's theorems are also applicable for n-number of Inputs. (It is not limited for 2-inputs)

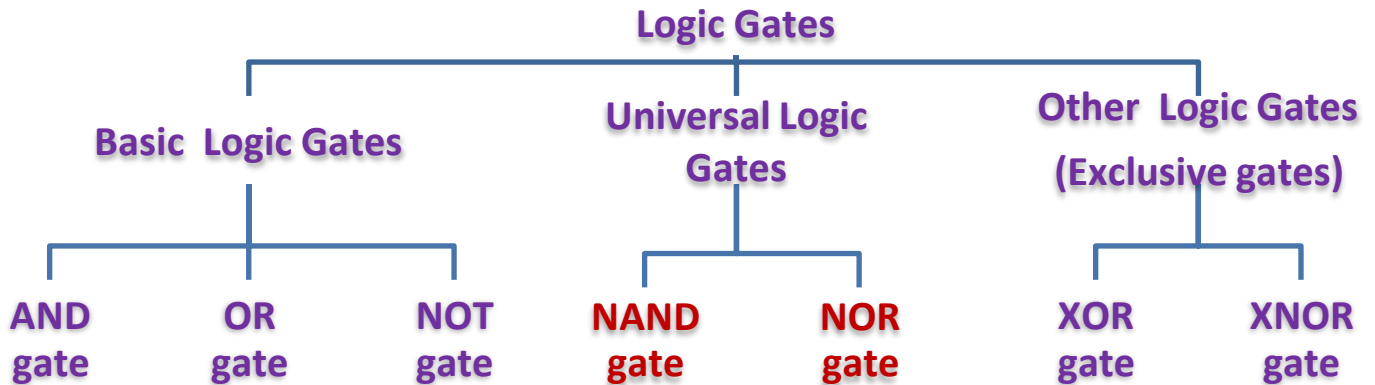
#LOGIC GATES:

Explain the Logic gates. Give the classification of logic gates. Explain the NOR & NAND gates as universal gates.

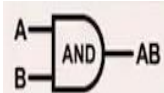
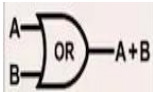
Answer:

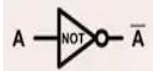
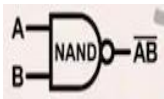
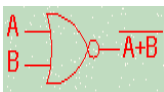
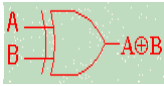
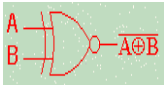
LOGIC GATES:

- Logic gates are the fundamental building blocks of digital systems which has one or more inputs and single output.
- Classification of Logic gates:
Logic gates can be classified as,
 - Basic gates – AND ,OR, NOT
 - Universal gates – NAND, NOR (we can design any digital circuit from these gates)
 - Other gate (Exclusive gates) – XOR, XNOR



- Inputs and outputs of logic gates can occur only in 2 levels. These two levels are termed High and Low, or True and False, or ON and OFF or simply 1 and 0.
- The table which lists all the possible combinations of input variables and the corresponding outputs is called a truth table.

S. N.	Logic Gate	Symbol	Logical Expression	Truth Table	Definition																		
1	AND		$Y = A \cdot B$	<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>$Y = A \cdot B$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Input		Output	A	B	$Y = A \cdot B$	0	0	0	0	1	0	1	0	0	1	1	1	Output of AND gate is high (i.e. 1) if and only if all the inputs should be high otherwise output will be low. (Logical multiplication)
Input		Output																					
A	B	$Y = A \cdot B$																					
0	0	0																					
0	1	0																					
1	0	0																					
1	1	1																					
2	OR		$Y = A + B$	<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>$Y = A + B$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Input		Output	A	B	$Y = A + B$	0	0	0	0	1	1	1	0	1	1	1	1	Output of OR gate is high (i.e. 1) if at least one of the inputs should be high otherwise output will be low. (Logical addition)
Input		Output																					
A	B	$Y = A + B$																					
0	0	0																					
0	1	1																					
1	0	1																					
1	1	1																					

S. N.	Logic Gate	Symbol	Logical Expression	Truth Table	Definition																		
3	NOT		$Y = \bar{A}$	<table><tr><th>Input</th><th>Output</th></tr><tr><td>A</td><td>$Y = \bar{A}$</td></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	Input	Output	A	$Y = \bar{A}$	0	1	1	0	Output of NOT gate is complement of input. (Inverter)										
Input	Output																						
A	$Y = \bar{A}$																						
0	1																						
1	0																						
4	NAND		$Y = \overline{AB}$	<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>$Y = \overline{AB}$</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	Input		Output	A	B	$Y = \overline{AB}$	0	0	1	0	1	1	1	0	1	1	1	0	Output of NAND gate is high (i.e. 1) if at least one of the inputs should be low otherwise output will be low. (Cascade combination of AND-NOT)
Input		Output																					
A	B	$Y = \overline{AB}$																					
0	0	1																					
0	1	1																					
1	0	1																					
1	1	0																					
5	NOR		$Y = \overline{A+B}$	<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>$Y = \overline{A+B}$</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	Input		Output	A	B	$Y = \overline{A+B}$	0	0	1	0	1	0	1	0	0	1	1	0	Output of NOR gate is high (i.e. 1) if all the inputs should be low otherwise output will be low. (Cascade combination of OR-NOT)
Input		Output																					
A	B	$Y = \overline{A+B}$																					
0	0	1																					
0	1	0																					
1	0	0																					
1	1	0																					
6	XOR		$Y = A \oplus B$	<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>$Y = A \oplus B$</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	Input		Output	A	B	$Y = A \oplus B$	0	0	0	0	1	1	1	0	1	1	1	0	Output of XOR gate is high (i.e. 1) for odd number of high inputs.
Input		Output																					
A	B	$Y = A \oplus B$																					
0	0	0																					
0	1	1																					
1	0	1																					
1	1	0																					
7	XNOR		$Y = \overline{A \oplus B}$	<table><tr><th colspan="2">Input</th><th>Output</th></tr><tr><th>A</th><th>B</th><th>$Y = \overline{A \oplus B}$</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Input		Output	A	B	$Y = \overline{A \oplus B}$	0	0	1	0	1	0	1	0	0	1	1	1	Output of XNOR gate is low (i.e. 0) for odd number of high inputs.
Input		Output																					
A	B	$Y = \overline{A \oplus B}$																					
0	0	1																					
0	1	0																					
1	0	0																					
1	1	1																					

Summary of Truth Table of Logic Gates:

Inputs		AND	OR	NOT-A	NOT-B	NAND	NOR	XOR	XNOR
A	B	$Y = A \cdot B$	$Y = A + B$	$Y = \bar{A}$	$Y = \bar{B}$	$Y = \overline{AB}$	$Y = \overline{A+B}$	$Y = A \oplus B$	$Y = \overline{A \oplus B}$
0	0	0	0	1	1	1	1	0	1
0	1	0	1	1	0	1	0	1	0
1	0	0	1	0	1	1	0	1	0
1	1	1	1	0	0	0	0	0	1

#Adder Circuit:

What are the different types of adders? Explain the Half adder and full adder.

Answer:

Adder: (Binary Adder)

Types of Adders,

1. Half Adder
2. Full Adder

1. Half Adder:

- **Function:** Half adder will perform addition of two 1-bit numbers without previous carry input.
- It has 2-Inputs & 2-outputs.
- **Inputs:** A, B
- **Outputs:** Sum (S), Carry (C)

Block Diagram:

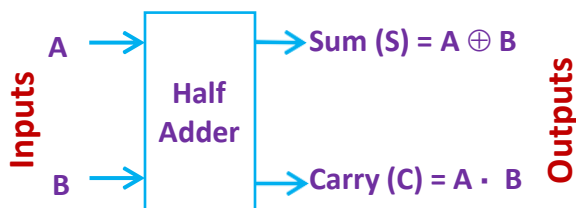


Fig. A : Block diagram of Half Adder

Circuit Diagram:

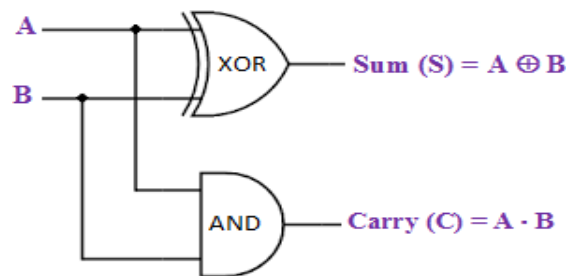


Fig. B : Circuit diagram of Half Adder

Truth Table:

Inputs		Outputs	
A	B	Sum ($S = A \oplus B$)	Carry ($C = A \cdot B$)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

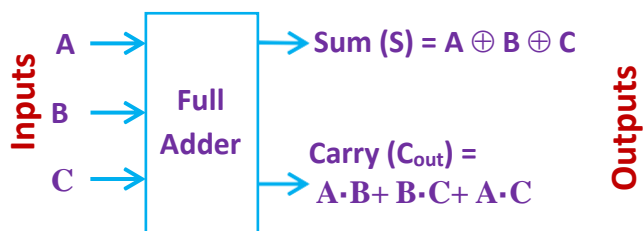
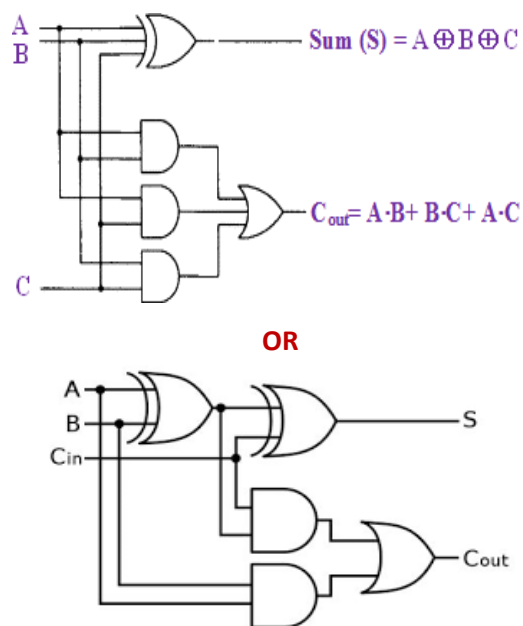
Fig. C : Truth table of Half Adder

Logical equations: (Output equations)

1. $\text{Sum (S)} = A \oplus B$
2. $\text{Carry (C)} = A \cdot B$

2. Full Adder:

- **Function:** Full adder will perform addition of two 1-bit numbers with previous carry input.
- It has 3-Inputs & 2-outputs.
- **Inputs:** A, B, C or (C_{in})
- **Outputs:** Sum (S), Carry (C_{out})

Block Diagram:**Fig. A : Block diagram of Full Adder****Circuit Diagram:****Fig. B : Circuit diagram of Full Adder****Truth Table:**

Inputs			Outputs	
A	B	C	Sum ($S = A \oplus B \oplus C$)	Carry ($C_{out} = A \cdot B + B \cdot C + A \cdot C$)
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

Fig. C : Truth table of Half Adder**Logical equations: (Output equations)**

1. Sum (S) = $A \oplus B \oplus C$
2. Carry (C_{out}) = $A \cdot B + B \cdot C + A \cdot C$

Flip Flop:

Explain the Flip Flop. What are the different types of Flip Flops? Explain the SR Latch. Explain the different types of Flip Flops. Give the applications of Flip Flops.

Answer:

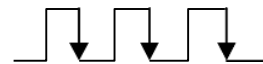
Flip flop (FF) and Latch:

- Flip flop or latch is a one bit memory cell. It can be used to store one bit of information at a time. (0's or 1's)
- A flip-flop or latch is a bistable multivibrator. It has two stable states i.e. 0 and 1.
- It has two outputs. Both outputs are complement of each other. (i.e. Q and \bar{Q}).
- Flip-flops using the clock signal are called clocked flip-flops.
- Clock-signals may be positive-edge triggered or negative-edge triggered.

Positive-edge triggered flip-flops are those in which state transitions take place only at positive edge of the clock pulse.



Negative-edge triggered flip-flops are those in which state transition take place only at negative edge of the clock pulse.

**Types of Flip flops,**

1. S-R Flip flop
2. J-K Flip flop
3. T Flip flop
4. D Flip flop

S-R (Set-Reset) latch:

- The SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates.
- It has two inputs i.e. $S \Rightarrow$ Set and $R \Rightarrow$ Reset.
- It has two outputs i.e. Q and \bar{Q} which are complement of each other.
- The latch has two useful states.
when $Q=0$ and $\bar{Q}=1$ the condition is called Reset state and
when $Q=1$ and $\bar{Q}=0$ the condition is called Set state.

SR Latch –with two NOR gates		SR Latch –with two NAND gates	
Circuit Diagram: 	Symbol: 	Circuit Diagram: 	Symbol:
For NOR gate:		For NAND gate:	

If any one of the inputs is high output will be low.
If all the inputs are low output will be high.

Truth Table:

Input		Output		Remark
S	R	Present State (O_n)	Next State (O_{n+1})	
0	0	0	0	No change (Hold)
0	0	1	1	
0	1	0	0	Reset (Clear)
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	X	Intermediate (invalid)
1	1	1	X	

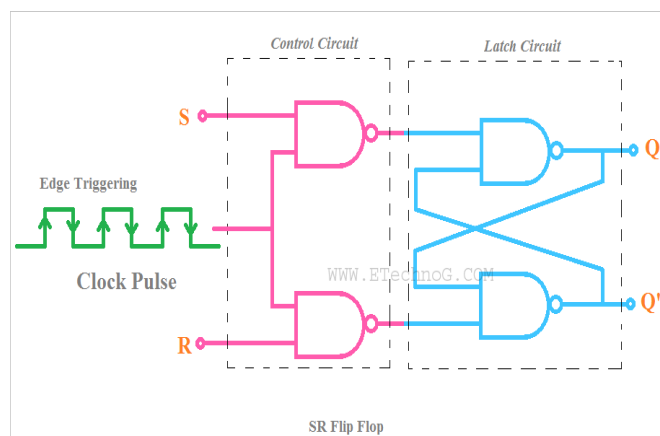
If any one of the inputs is low output will be high.
If all the inputs are high output will be low.

Truth Table:

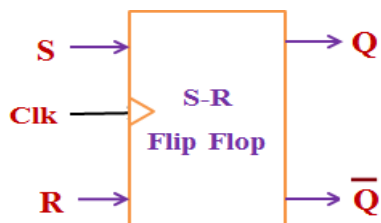
Input				Output		Remark
\bar{S}	\bar{R}	S	R	Present State (O_n)	Next State (O_{n+1})	
1	1	0	0	0	0	No change (Hold)
1	1	0	0	1	1	
1	0	0	1	0	0	Reset (Clear)
1	0	0	1	1	0	
0	1	1	0	0	1	Set
0	1	1	0	1	1	
0	0	1	1	0	X	Intermediate (invalid)
0	0	1	1	1	X	

S-R (Set-Reset) Flip Flop:

Circuit Diagram:



Symbol:



Truth Table:

Input			Output		Remark
Clk	S	R	Present State (O_n)	Next State (O_{n+1})	
1	0	0	0	0	No change (Hold)
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	X	Intermediate (invalid)
1	1	1	1	X	

P.S. – Present State (Q_n)

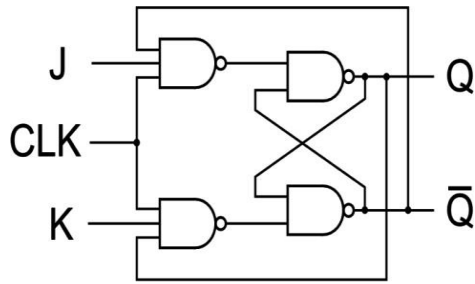
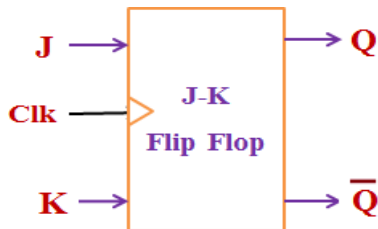
N.S. – Next State (Q_{n+1})

Operation:

If Clk=1, then

1. If S=0, R=0 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 0$ or 1 respectively, hence No change in Next state.
2. If S=0, R=1 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 0$, hence Next State will be Reset or Clear.
3. If S=1, R=0 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 1$, hence Next State will be Set.
4. If S=1, R=1 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 0$, $\bar{Q}_{n+1} = 0$ (both un-complemented and complemented will try to become 0's), hence Next State will be intermediate (invalid).

This condition is called as Race condition.

J-K Flip Flop:**Circuit Diagram:****Syllabus:****Truth Table:**

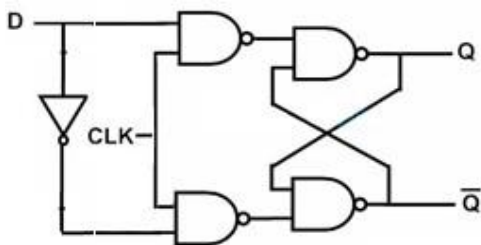
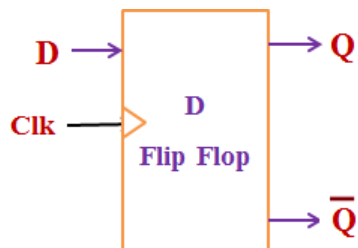
Input			Output		Remark
Clk	J	K	Present State (Q_n)	Next State (Q_{n+1})	
1	0	0	0	0	} Q_n No change (Hold)
1	0	0	1	1	
1	0	1	0	0	} 0 Reset (Clear)
1	0	1	1	0	
1	1	0	0	1	} 1 Set
1	1	0	1	1	
1	1	1	0	1	} \bar{Q} Complement (Toggle)
1	1	1	1	0	

P.S. – Present State (Q_n)N.S. – Next State (Q_{n+1})**Operation:**

If Clk=1, then

1. If J=0, K=0 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 0$ or 1 respectively, hence No change in Next state.
2. If J=0, K=1 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 0$, hence Next State will be Reset or Clear.
3. If J=1, K=0 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 1$, hence Next State will be Set.
4. If J=1, K=1 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 1$ or 0 respectively, hence Next State will be complement of Present state.

This condition is called ad Race around condition.

D-Flip Flop:**Circuit Diagram:****Syllabus:****Truth Table:**

Input		Output		Remark
Clk	D	Present State (Q_n)	Next State (Q_{n+1})	
1	0	0 or 1	0	} D Same as D-Input
1	1	0 or 1	1	} D Same as D-Input

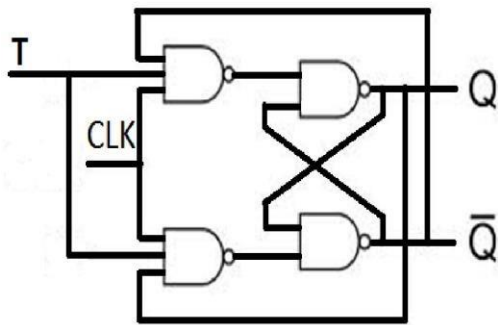
P.S. – Present State (Q_n)N.S. – Next State (Q_{n+1})**Operation:**

If Clk=1, then

1. If D=0 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 0$, hence Next State will be same as D-Input.
2. If D=1 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 1$, hence Next State will be same as D-Input.

T-(Toggle) Flip Flop:

Circuit Diagram:



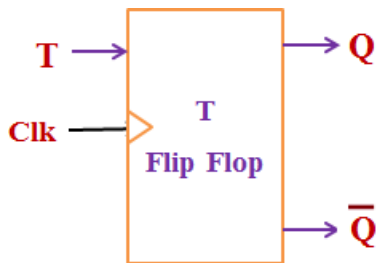
Truth Table:

Input		Output		Remark
Clk	T	Present State (Q_n)	Next State (Q_{n+1})	
1	0	0	0	} Q_n
		1	1	
1	1	0	1	} \bar{Q}
		1	0	

P.S. – Present State (Q_n)

N.S. – Next State (Q_{n+1})

Syllabus:



Operation:

If Clk=1, then

1. If T=0 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 0$ or 1 respectively, hence No change in Next State.
2. If T=1 when P.S. $Q_n = 0$ or 1 then N.S. $Q_{n+1} = 1$ or 0 respectively, hence Next State will be toggle.

Applications of Flip Flops:

Flip flops can be used,

1. as a one bit memory cell.
2. in counters.
3. in shift registers.
4. in sequence generators.
5. in sequence detectors.

#Microprocessor & Microcontroller:

Differentiate between Microprocessor and Microcontroller.

Answer:

Sr. No.	Parameters	Microprocessor	Microcontroller
1.	Contains	ALU, General purpose register, stack pointer, program counter, timing & control circuit, interrupt circuit.	Circuitry of Microprocessor and has in built RAM, ROM, I/O devices, Timers, Counters and other peripheral
2.	RAM, ROM, I/O devices, Timers, Counters and other peripheral	Not present on the chips.	Embedded on the single chip. (inbuilt)
3.	External bus to interface Memory, I/O devices, Timers and other peripherals.	Required	Not required
4.	Instructions to process.	Complex and expensive with large number	Simple and inexpensive with less number
5.	Memory address space & data	Large & more respectively	Small & less respectively
6.	Multifunction pins	Less	More
7.	Clock speed	High (1 GHz to 4 GHz)	Less (1 MHz to 300 MHz)
8.	Design	Very flexible	Less flexible
9.	Reliability	Less	High
10.	Power consumption	High	Low
11.	Cost	Expensive	Cheap
12.	Available in size	32-bits, 64 bits	8-bits, 16 bits
13.	Used for	General purpose applications	Specific applications
14.	Example	8085, 8086	8051, PIC, ARM

Explain the architecture of Microprocessor.

Answer:

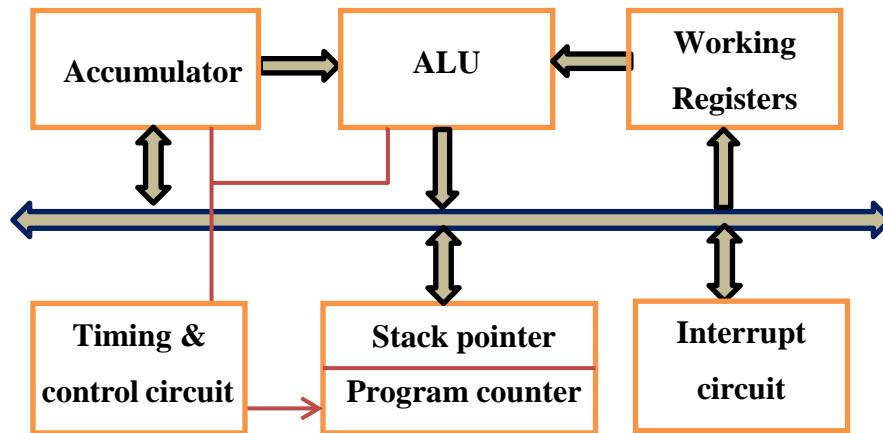


Fig. Architecture (Block diagram) of Microprocessor

Microprocessor consists of,

- ALU (Arithmetic Logic Unit)
- Accumulator
- Stack pointer & Program counter
- Working registers
- Timing & control circuit
- Interrupt circuit

Explain the architecture of Microcontroller.

Answer:

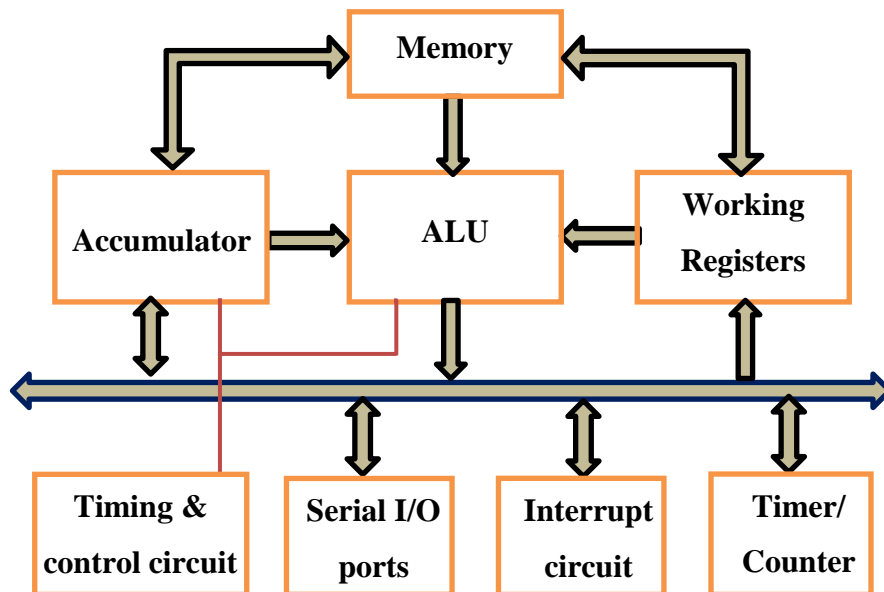


Fig. Architecture (Block diagram) of Microcontroller

Microcontroller consists of circuitry of Microprocessor and has in built RAM, ROM, I/O devices, Timers, Counters and other peripheral