

LECTURE NOTES

ON

Digital Electronics

CONTENTS

Sl.No	Chapter Name	Page No
1	Basics of Digital Electronics	3-17
2	Combinational logic circuits	18-24
3	Sequential logic Circuits	25-30
4	Registers, Memories & PLD	31-34
5	A/D and D/A Converters	35-37
6	LOGIC FAMILIES	38-41

Unit-1: Basics of Digital Electronics

A Digital system is an interconnection of digital modules and it is a system that manipulates discrete elements of information that is represented internally in the binary form. Now a day's digital systems are used in wide variety of industrial and consumer products such as automated industrial machinery, pocket calculators, microprocessors, digital computers, digital watches, TV games and signal processing and so on.

Number System-Binary, Octal, Decimal, Hexadecimal - Conversion from one system to another number system.

Number :

The way of quantifying anything , represented through various combination of symbols is called number.

Digit :

The various symbols representing a single number in any number system is called digit. E.g. Decimal number system (Arabic numerals): Digits: 0,1,2,3,4,5,6,7,8,9.

Radix / Base (r) :

The maximum number of different digits of any number system. E.g Decimal NS, $r = 10$

Number system:

The properly structured number formation is called Number system. In number system there are different symbols and each symbol has an absolute value and also has place value.

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

Number various combination digits according to position

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

↑ Radix point

$$= d_n d_{n-1} \dots d_1 d_0 . d_{-1} d_{-2} \dots d_{-m}$$

The value,

$$N_{10} = d_n \times r^n + d_{n-1} \times r^{n-1} + \dots + d_1 \times r^1 + d_0 \times r^0 + d_{-1} \times r^{-1} + d_{-2} \times r^{-2} + \dots + d_{-m} \times r^{-m}$$

* The right most digit of any number is called Least Significant Digit

* The left most digit of any number is called Most Significant Digit

TYPES OF NUMBER SYSTEM:-

There are four types of number systems. They are

1. Decimal number system
2. Binary number system

3. Octal number system

4. Hexadecimal number system

DECIMAL NUMBER SYSTEM:-

The decimal number system contains ten unique symbols 0,1,2,3,4,5,6,7,8 and 9.

- In decimal system 10 symbols are involved, so the base or radix is 10.
- It is a positional weighted system.

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

For example:-

$$\begin{aligned} 9256.26 &= 9 \times 1000 + 2 \times 100 + 5 \times 10 + 6 \times 1 + 2 \times (1/10) + 6 \times (1/100) \\ &= 9 \times 10^3 + 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 + 2 \times 10^{-1} + 6 \times 10^{-2} \end{aligned}$$

BINARY NUMBER SYSTEM:-

- The binary number system is a positional weighted system.
- The base or radix of this number system is 2.
- It has two independent symbols, The symbols used are 0 and 1. □ A binary digit is called a bit

$$(d_n \times 2^n) + (d_{n-1} \times 2^{n-1}) + (d_{n-2} \times 2^{n-2}) + \dots + (d_0 \times 2^0) + (d_{-1} \times 2^{-1}) + (d_{-2} \times 2^{-2}) + \dots + (d_{-k} \times 2^{-k})$$

OCTAL NUMBER SYSTEM:-

- It is also a positional weighted system.
- Its base or radix is 8.
- It has 8 independent symbols 0,1,2,3,4,5,6 and 7.
- Its base $8 = 2^3$, every 3-bit group of binary can be represented by an octal digit.

HEXADECIMAL NUMBER SYSTEM:-

- The hexadecimal number system is a positional weighted system.
- The base or radix of this number system is 16.
- The symbols used are 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E and F
- The base $16 = 2^4$, every 4-bit group of binary can be represented by a hexadecimal digit.

CONVERSION FROM ONE NUMBER SYSTEM TO ANOTHER :-

1. BINARY NUMBER SYSTEM:-

(a) 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.

$$\begin{aligned}(111.101)_2 &= (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3}) \\ &= 4 + 2 + 1 + 0.5 + 0 + 0.125 \\ &= (7.625)_{10}\end{aligned}$$

(b) 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 $(101111010110.110110011)_2$ into octal.

Solution :

Group of 3 bits are	101	111	010	110	.	110	110	011
Convert each group into octal =	5	7	2	6	.	6	6	3

The result is $(5726.663)_8$

(c) Binary to Hexadecimal conversion:- For conversion binary to hexadecimal number the binary numbers starting from the binary point, groups are made of 4 bits each, on either side of the binary point.

(ii) Convert $(01011111011.011111)_2$ into hexadecimal.

Solution:

Given Binary number	010	1111	1011	.	0111	11
Group of 4 bits are	= 0010	1111	1011	.	0111	1100
Convert each group into octal =	2	F	B	.	7	C

The result is $(2FB.7C)_{16}$

2. DECIMAL NUMBER SYSTEM:-

(a) Decimal to binary conversion:- In the conversion the integer number are converted to the desired base using successive division by the base or radix.

For example: (i) Convert $(52)_{10}$ into binary.

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

Solution:

Integer part

$$\begin{array}{r} 2 \overline{) 105} \\ 2 \overline{) 52} \quad - 1 \\ 2 \overline{) 26} \quad - 0 \\ 2 \overline{) 13} \quad - 0 \\ 2 \overline{) 6} \quad - 1 \\ 2 \overline{) 3} \quad - 0 \\ 2 \overline{) 1} \quad - 1 \\ 0 \quad - 1 \end{array}$$

Fraction part

$$\begin{array}{l} 0.15 \times 2 = 0.30 \\ 0.30 \times 2 = 0.60 \\ 0.60 \times 2 = 1.20 \\ 0.20 \times 2 = 0.40 \\ 0.40 \times 2 = 0.80 \\ 0.80 \times 2 = 1.60 \end{array}$$

Result of $(105.15)_{10}$ is $(1101001.001001)_2$

(b) Decimal to octal conversion:- To convert the given decimal integer number to octal, successively divide the given number by 8 till the quotient is 0.

(i) Convert $(378.93)_{10}$ into octal.

Solution:

$$\begin{array}{r} 8 \overline{) 378} \\ 8 \overline{) 47} \quad - 2 \\ 8 \overline{) 5} \quad - 7 \\ 0 \quad - 5 \end{array}$$

$$\begin{array}{l} 0.93 \times 8 = 7.44 \\ 0.44 \times 8 = 3.52 \\ 0.52 \times 8 = 4.16 \\ 0.16 \times 8 = 1.28 \end{array}$$

Result of $(378.93)_{10}$ is $(572.7341)_8$

(c) Decimal to hexadecimal conversion:-

(i) Convert $(2598.675)_{10}$ into hexadecimal.

Solution:

	Remainder	
	Decimal	Hex
$16 \overline{) 2598}$		
$16 \overline{) 162} \quad - 6$	6	
$16 \overline{) 10} \quad - 2$	2	
$0 \quad - 10$	A	

	Hex
$0.675 \times 16 = 10.8$	A
$0.800 \times 16 = 12.8$	C
$0.800 \times 16 = 12.8$	C
$0.800 \times 16 = 12.8$	C

Result of $(2598.675)_{10}$ is $(A26.ACCC)_{16}$

3. OCTAL NUMBER SYSTEM:-

(a) Octal to binary conversion:- To convert a given octal number to binary, replace each octal digit by its 3-bit binary equivalent.

For example:

Convert $(367.52)_8$ into binary.

Solution:

Given Octal number is

$$\begin{array}{ccccccc} & 3 & 6 & 7 & . & 5 & 2 \\ = & 011 & 110 & 111 & . & 101 & 010 \end{array}$$

Convert each group octal to binary

Result of $(367.52)_8$ is $(011110111.101010)_2$

(b) Octal to decimal conversion:- For conversion octal to decimal number, multiply each digit in the octal number by the weight of its position and add all the product terms

For example: -

Convert $(4057.06)_8$ to decimal

Solution:

$$\begin{aligned} (4057.06)_8 &= 4 \times 8^3 + 0 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 0 \times 8^{-1} + 6 \times 8^{-2} \\ &= 2048 + 0 + 40 + 7 + 0 + 0.0937 \\ &= (2095.0937)_{10} \end{aligned}$$

Result is $(2095.0937)_{10}$

(c) Octal to hexadecimal conversion:- For conversion of octal to Hexadecimal, first convert the given octal number to binary and then binary number to hexadecimal

(4) HEXADECEMAL NUMBER SYSTEM :- (a) Hexadecimal to binary conversion:- For conversion of hexadecimal to binary, replace hexadecimal digit by its 4 bit binary group

Convert $(3A9E.B0D)_{16}$ into binary.

Solution:

Given Hexadecimal number is

$$\begin{array}{ccccccc} & 3 & A & 9 & E & . & B & 0 & D \\ = & 0011 & 1010 & 1001 & 1110 & . & 1011 & 0000 & 1101 \end{array}$$

Convert each hexadecimal digit to 4 bit binary

Result of $(3A9E.B0D)_{16}$ is $(0011101010011110.101100001101)_2$

(b) Hexadecimal to decimal conversion:- For conversion of hexadecimal to decimal, multiply each digit in the hexadecimal number by its position weight and add all those product terms.

Convert $(A0F9.0EB)_{16}$ to decimal

Solution:

$$\begin{aligned} (A0F9.0EB)_{16} &= (10 \times 16^3) + (0 \times 16^2) + (15 \times 16^1) + (9 \times 16^0) + (0 \times 16^{-1}) + (14 \times 16^{-2}) + (11 \times 16^{-3}) \\ &= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026 \\ &= (41209.0572)_{10} \end{aligned}$$

Result is $(41209.0572)_{10}$

((c) Hexadecimal to Octal conversion:- For conversion of hexadecimal to octal, first convert the given hexadecimal number to binary and then binary number to octal

Arithmetic Operation-Addition, Subtraction, Multiplication, Division, 1's & 2's complement of Binary numbers & Subtraction using complements method

1. BINARY ADDITION:-

The binary addition rules are as follows

	carry	1	carry
$0+0=0$;	$0+1=1$;	$1+0=1$;	$1+1=10$, SUM, $1+1=11$

Add $(100101)_2$ and $(1101111)_2$.

Solution :-

$$\begin{array}{r}
 100101 \\
 + 1101111 \\
 \hline
 10010100
 \end{array}$$

Result is $(10010100)_2$

SUM

2. BINARY SUBTRACTION:-

The binary subtraction rules are as follows

$0-0=0$; $1-1=0$; $1-0=1$; $0-1=11$, with a borrow of 1

Subtract $(111.111)_2$ from $(1010.01)_2$.

Solution :-

$$\begin{array}{r}
 1010.010 \\
 - 111.111 \\
 \hline
 0010.011
 \end{array}$$

Result is $(0010.011)_2$

3. BINARY MULTIPLICATION:-

The binary multiplication rules are as follows

$0 \times 0 = 0$; $1 \times 1 = 1$; $1 \times 0 = 0$; $0 \times 1 = 0$

Multiply $(1101)_2$ by $(110)_2$.

Solution :-

$$\begin{array}{r}
 1101 \\
 \times 110 \\
 \hline
 0000 \\
 1101 \\
 + 1101 \\
 \hline
 1001110
 \end{array}$$

Result is $(1001110)_2$

4. BINARY DIVISION:-

The binary division is very simple and similar to decimal number system.

So we have only 2 rules $0 \div 1 = 0$

$$1 \div 1 = 1$$

$$110 \overline{) 101101} \text{ (} 111.1$$

$$\begin{array}{r}
 - 110 \\
 \hline
 1010 \\
 \hline
 110 \\
 \hline
 1001 \\
 \hline
 110 \\
 \hline
 110 \\
 \hline
 110 \\
 \hline
 000
 \end{array}$$

Result is $(111.1)_2$

1's COMPLEMENT REPRESENTATION :-

The 1's complement of a binary number is obtained by changing each 0 to 1 and each 1 to 0.

Find $(1100)_2$ 1's complement.

Solution :-

Given
1's complement is

$$\begin{array}{cccc}
 1 & 1 & 0 & 0 \\
 0 & 0 & 1 & 1
 \end{array}$$

Result is $(0011)_2$

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.

$$2's \text{ complement} = 1's \text{ complement} + 1$$

Find $(1010)_2$ 2's complement.

Solution :-

Given	1	0	1	0
1's complement is	0	1	0	1
	+	1		
2's complement	0	1	1	0

Result is $(0110)_2$

SIGNED NUMBER :-

In sign - magnitude form, additional bit called the sign bit is placed in front of the number. If the sign bit is 0, the number is positive. If it is a 1, the number is negative.

$$\begin{array}{ccccccc} 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ \uparrow & & & & & & \\ \text{Sign bit} & & & & & & \end{array} = +41$$

$$\begin{array}{ccccccc} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ \uparrow & & & & & & \\ \text{Sign bit} & & & & & & \end{array} = -41$$

SUBTRACTION USING COMPLEMENT METHOD :-

1's COMPLEMENT :-

In 1's complement subtraction, add the 1's complement of subtrahend to the minuend. If there is a carry out, then the carry is added to the LSB. This is called end around carry. If the MSB is 0, the result is positive. If the MSB is 1, the result is negative and is in its 1's complement form. Then take its 1's complement to get the magnitude in binary.

Subtract $(10000)_2$ from $(11010)_2$ using 1's complement.

Solution:-	11010	11010	= 26
	- 10000	+ 01111 (1's complement)	= -16
	Carry → 101001	+ 10	
	+ 1		
	01010	= +10	

Result is +10

2's COMPLEMENT :-

In 2's complement subtraction, add the 2's complement of subtrahend to the minuend. If there is a carry out, ignore it. If the MSB is 0, the result is positive. If the MSB is 1, the result is negative and is in its 2's complement form. Then take its 2's complement to get the magnitude in binary.

Subtract $(1010100)_2$ from $(1010100)_2$ using 2's complement.

Solution:-

$$\begin{array}{r}
 1010100 \\
 - 1010100 \\
 \hline
 \end{array}
 \Rightarrow
 \begin{array}{r}
 1010100 \\
 + 0101100 \text{ (2's complement)} \\
 \hline
 10000000 \text{ (Ignore the carry)} \\
 0 \text{ (result = 0)} \\
 \hline
 \end{array}
 \begin{array}{r}
 = 84 \\
 = - 84 \\
 \underline{0}
 \end{array}$$

Hence MSB is 0. The answer is positive. So it is $+0000000 = 0$

Digital Code & its application & distinguish between weighted & non-weight Code, Binary codes, excess-3 and Gray codes.

DIGITAL CODES:-

In practice the digital electronics requires to handle data which may be numeric, alphabets and special characters. This requires the conversion of the incoming data into binary format before it can be processed. There is various possible ways of doing this and this process is called encoding. To achieve the reverse of it, we use decoders.

WEIGHTED AND NON-WEIGHTED CODES

There are two types of binary codes

- 1) Weighted binary codes : In weighted codes, for each position (or bit) ,there is specific weight attached. For example, in binary number, each bit is assigned particular weight 2^n where 'n' is the bit number for $n = 0,1,2,3,4$ the weights are 1,2,4,8,16 respectively.
Example :- BCD

- 2) Non- weighted binary codes:

Non-weighted codes are codes which are not assigned with any weight to each digit position, i.e., each digit position within the number is not assigned fixed value. Example:- Excess - 3 (XS -3) code and Gray codes

BINARY CODED DECIMAL (BCD):- 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.

BCD ADDITION:-

Addition of BCD (8421) is performed by adding two digits of binary, starting from least significant digit. In case if the result is an illegal code (greater than 9) or if there is a carry out of one then add 0110(6) and add the resulting carry to the next most significant.

Solution:-

$$\begin{array}{r}
 679.6 \\
 + 536.8 \\
 \hline
 1216.4
 \end{array}$$

Result is 1216.4

BCD SUBTRACTION:-

The BCD subtraction is performed by subtracting the digits of each 4-bit group of the subtrahend from corresponding 4-bit group of the minuend in the binary starting from the LSD. If there is no borrow from the next higher group then no correction is required. If there is a borrow from the next group, then 6 (0110) is subtracted from the difference term of this group.

Add 679.6 from 536.8 using BCD

$$\begin{array}{r}
 0110 \ 0111 \ 1001 \ . \ 0110 \\
 \Rightarrow + 0101 \ 0011 \ 0110 \ . \ 1000 \\
 \hline
 1011 \ 1010 \ 1111 \ . \ 1110 \\
 + 0110 \ 0110 \ 0110 \ . \ 0110 \\
 \hline
 0001 \ 0010 \ 0001 \ 0110 \ . \ 0100 \\
 1 \quad 2 \quad 1 \quad 6 \quad 4
 \end{array}$$

(536.8)
(All are illegal)
(Add 0110 to each)
(corrected sum = 1216.4)

Solution:-

$$\begin{array}{r}
 206.7 \\
 - 147.8 \\
 \hline
 58.9
 \end{array}$$

Result is (58.9)₁₀

Subtract 147.8 from 206.7 using 8421 BCD code.

$$\begin{array}{r}
 0010 \ 0000 \ 0110 \ . \ 0111 \\
 \Rightarrow - 0001 \ 0100 \ 0111 \ . \ 1000 \\
 \hline
 0000 \ 1011 \ 1110 \ . \ 1111 \\
 - 0110 \ 0110 \ . \ 0110 \\
 \hline
 0101 \ 1000 \ . \ 1001 \\
 5 \quad 8 \quad 9
 \end{array}$$

(206.7 in BCD)
(147.8 in BCD)
(Borrows are present)
(corrected difference = 58.9)

EXCESS THREE(XS-3) CODE:-

The Excess-3 code, also called XS-3, is a non-weighted BCD code. This derives its name from the fact that each binary code word is the corresponding 8421 code word plus 0011(3). It is a sequential code. It is a self-complementary code. Excess-3 code is non-weighted and self-complementary code. A self-complementary binary code is always complement itself. The complement of a binary number can be obtained from that number by replacing 0's with 1's and 1's with 0's. The sum of binary number and its complement is always equal to decimal 9. In other words, the 1's complement of an excess-3 code is the excess-3 code for the 9's complement of the corresponding decimal number.

For example, the excess-3 code for decimal number 5 is 1000 and 1's complement of 1000 is 0111, which is excess-3 code for decimal number 4, and it is 9's complement of number 5.

ASCII CODE:-

The American Standard Code for Information Interchange (ASCII) pronounced as 'ASKEE' is widely used alphanumeric code. This is basically a 7 bit code. The number of

can be used to encode both the uppercase and lowercase characters of the alphabet (52 symbols) and some special symbols in addition to the 10 decimal digits. It is used extensively for printers and terminals that interface with small computer systems. The table shown below shows the ASCII groups.

GRAY CODE:-

The gray code is a non-weighted code. It is not a BCD code. It is cyclic code because successive words in this differ in one bit position only i.e it is a unit distance code. Gray code is used in instrumentation and data acquisition systems where linear or angular displacement is measured.

BINARY- TO - GRAY CONVERSION:-

If an n-bit binary number is represented by $B_n B_{n-1} \dots B_1$ and its gray code equivalent by $G_n G_{n-1} \dots G_1$, where B_n and G_n are the MSBs, then gray code bits are obtained from the binary code as follows

$$G_n = B_n$$

$$G_{n-1} = B_n \oplus B_{n-1}$$

.

.

$$G_1 = B_2 \oplus B_1$$

Where the symbol $\oplus$ stands for Exclusive OR (X-OR)

GRAY- TO - BINARY CONVERSION:-

If an n-bit gray number is represented by $G_n G_{n-1} \dots G_1$ and its binary equivalent by $B_n B_{n-1} \dots B_1$, then binary bits are obtained from Gray bits as follows :

$$B_n = G_n$$

$$B_{n-1} = B_n \oplus G_{n-1}$$

.

.

$$B_1 = B_2 \oplus G_1$$

Logic gates: AND, OR, NOT, NAND, NOR, Exclusive-OR, Exclusive-NOR--
Symbol, Function, expression, truth table & timing diagram

LOGIC GATES:-

- Logic gates are the fundamental building blocks of digital systems.
- There are 3 basic types of gates AND, OR and NOT.
- Logic gates are electronic circuits because they are made up of a number of electronic devices and components.
- Inputs and outputs of logic gates can occur only in 2 levels(logic 1, logic 0). These two levels are termed HIGH and LOW, or TRUE and FALSE, or ON and OFF □ The table which lists all the possible combinations of input variables and the corresponding output of any logic circuit/device, called a truth table.

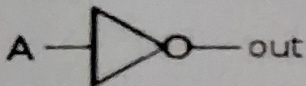
DIFFERENT TYPES OF LOGIC GATES

NOT GATE (INVERTER):-

- A NOT gate, also called an inverter, has only one input and one output.
- It is a device whose output is always the complement of its input.
- The output of a NOT gate is the logic 1 state when its input is in logic 0 state and the logic 0 state when its input is in logic 1 state.

IC No. :- 7404

Logic Symbol



Truth table

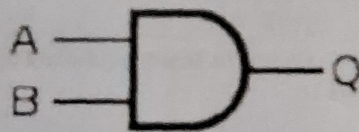
INPUT A	OUTPUT A
0	1
1	0

AND GATE:-

- An AND gate has two or more inputs but only one output.
- The output is logic 1 state only when each one of its inputs is at logic 1 state. □ The output is logic 0 state even if one of its inputs is at logic 0 state.

IC No.:- 7408

Logic Symbol



Truth Table

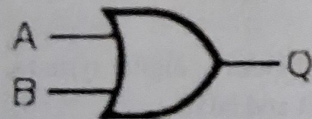
INPUT		OUTPUT
A	B	$Q = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

OR GATE:-

- An OR gate may have two or more inputs but only one output.
- The output is logic 1 state, even if one of its inputs is 1
- The output is logic 0 state, only when each one of its inputs is in logic state.

IC No.:- 7432

Logic Symbol



Truth Table

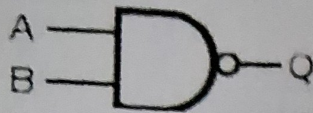
INPUT		OUTPUT
A	B	$Q = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

NAND GATE:-

- NAND gate is a combination of an AND gate and a NOT gate.
- The output is logic 0 when each of the input is logic 1 and for any other combination of inputs, the output is logic 1.

IC No.:- 7400 two input NAND gate

Logic Symbol



Truth Table

INPUT		OUTPUT
A	B	$Q = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

NOR GATE:-

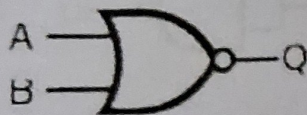
- NOR gate is a combination of an OR gate and a NOT gate.
- The output is logic 1, only when each one of its input is logic 0 and for any other combination of inputs the output is a logic 0 level.

IC No.:- 7402 two input NOR gate

Truth Table

INPUT		OUTPUT
A	B	$Q = A + B$
0	0	1
0	1	0
1	0	0
1	1	0

Logic Symbol

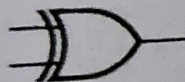


EXCLUSIVE - OR (X-OR) GATE

- An X-OR gate is a two input, one output logic circuit.
- The output is logic 1 when one and only one of its two inputs is logic 1. When both the inputs is logic 0 or when both the inputs is logic 1, the output is logic 0.

IC No.:- 7486

Logic Symbol



INPUTS are A and B

OUTPUT is $Q = A \oplus B$

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

Truth Table

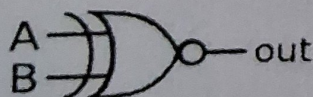
INPUT		OUTPUT
A	B	$Q = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

EXCLUSIVE - NOR (X-NOR) GATE

- An X-NOR gate is the combination of an X-OR gate and NOT gate
- An X-NOR gate is a two input, one output logic circuit.
- The output is logic 1 only when both the inputs are logic 0 or when both the inputs is 1.
- The output is logic 0 when one of the inputs is logic 0 and other is 1

IC No.:- 74266

Logic Symbol



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

$$= A \text{ XNOR } B$$

INPUT		OUTPUT
A	B	$OUT = A \text{ XNOR } B$
0	0	1
0	1	0
1	0	0
1	1	1

Universal Gates & its Realisation

UNIVERSAL GATES

There are 3 basic gates AND, OR and NOT, there are two universal gates NAND and NOR. Both NAND and NOR gates can perform all logic functions i.e. AND, OR, NOT, EXOR and EXNOR.

NAND GATE

(i) NOT gate from NAND gate

Input = A
Output Q = $\bar{A}$



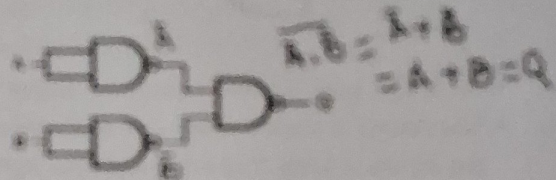
(ii) AND gate from NAND gate

Input s are A and B
Output Q = $A \cdot B$



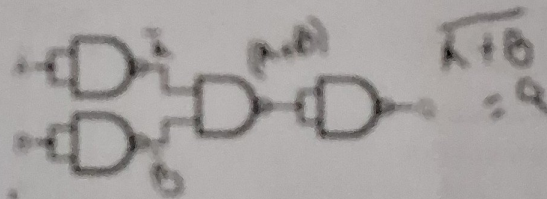
(iii) OR gate from NAND gate

Inputs are A and B
Output Q = $A + B$



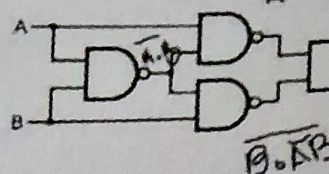
(iv) NOR gate from NAND gate

Inputs are A and B
Output Q = $\overline{A + B}$



e) EX-OR gate from NAND gate

Inputs are A and B
Output $Q = A\bar{B} + \bar{A}B$



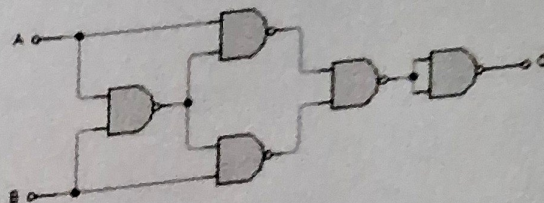
$$\begin{aligned} & \overline{(A \cdot \bar{B}) (B \cdot \bar{A})} \\ &= \overline{A \cdot \bar{B}} + \overline{B \cdot \bar{A}} \\ &= A(\bar{A} + B) + B(\bar{B} + \bar{A}) \\ &= A\bar{A} + AB + \bar{B}B + \bar{B}\bar{A} \\ &= 0 + AB + 0 + \bar{A}\bar{B} \\ &= AB + \bar{A}\bar{B} = Q \end{aligned}$$

e) EX-OR

Inputs are A and B
Output C

f) EX-NOR gate From NAND gate

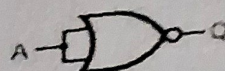
Inputs are A and B
Output $Q = AB + \bar{A}\bar{B}$



NOR GATE:-

a) Inverter from NOR gate

Input = A
Output $Q = \bar{A}$



b) AND gate from NOR gate

Inputs are A and B
Output $Q = A \cdot B$

