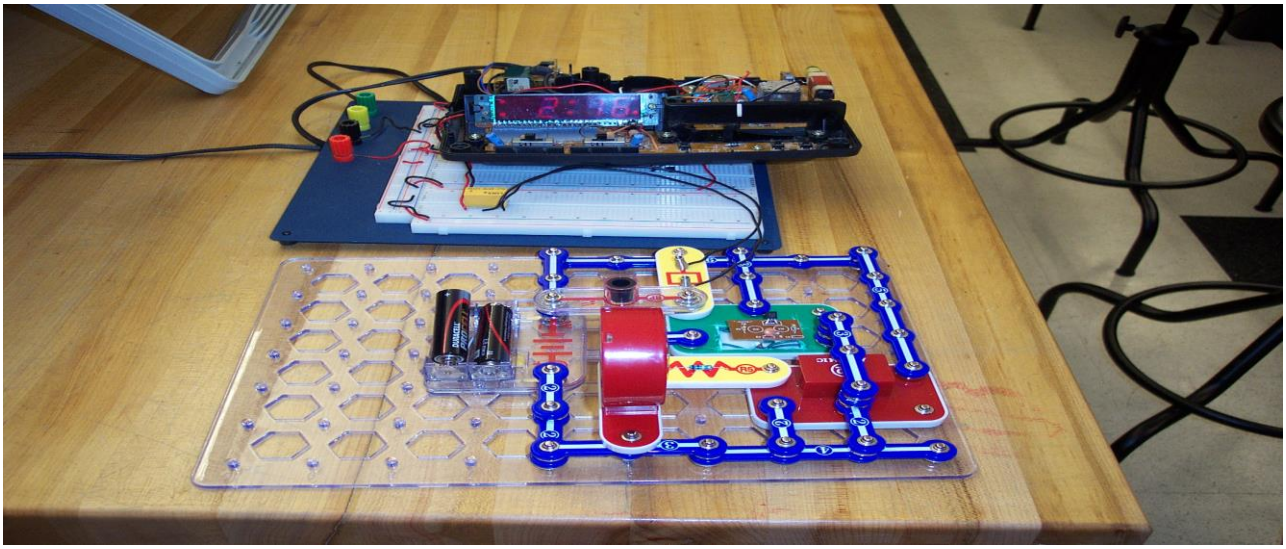
DEPARTMENT OF PHYSICS
INDO – AMERICAN COLLEGE CHEYYAR

III B.SC PHYSICS – ODD SEMESTER
ELECTIVE I
DIGITAL ELECTRONICS – BEPH54A



SYLLABUS:

UNIT – I: DIGITAL FUNDAMENTALS AND LOGIC GATES

Number systems – decimal, binary, octal and hexadecimal system –Conversion from one number system to another Codes – BCD code –Excess 3 code, Gray code – ASCII code - Binary arithmetic –Binary addition – subtraction – unsigned binary numbers – sign magnitude numbers – 1's and 2's complement – Binary multiplications and division - AND, OR circuits using diodes and transistors – NOT using transistors – NAND, NOR and EXOR – functions and truth tables. NAND & NOR as universal gates.

CHAPTER I - NUMBER SYSTEMS

Introduction:

A digital computer is a programmable machine that process binary data, represented in binary number system. We have to understand computers and digital electronics only with the help of binary number system.

1.1 Decimal number system:

The decimal number system makes use of ten digits namely 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. Since ten basic symbols or digits are used, the decimal number system is said to have a **base of ten**. All the digits in the decimal number system are expressed in powers of 10 like $\mathbf{10^0, 10^1, 10^2}$ etc. for integer part and $\mathbf{10^{-1}, 10^{-2}, 10^{-3}}$ etc. for the fractional part.

Example:

543.421

$3 \times 10^0 = 3$

$4 \times 10^1 = 40$

$5 \times 10^2 = 500$

543

.421

$1 \times 10^{-3} = 1/1000 = 0.001$

$2 \times 10^{-2} = 2/100 = 0.02$

$4 \times 10^{-1} = 4/10 = 0.4$

0.421

1.2 Binary number system:

A binary number system uses the symbols or digits namely 0 and 1. The binary system has a **base 2**. A binary digit 0 or 1 is called a **bit**. All the bits will have powers of 2 like $\mathbf{2^0, 2^1, 2^2}$ etc. for the integer portion $\mathbf{2^{-1}, 2^{-2}}$ etc.

1.2.1 Binary to decimal conversion:

$(110)_2 = (6)_{10}$

110

$0 \times 2^0 = 0$

$1 \times 2^1 = 2$

$1 \times 2^2 = 4$

6

$(.101)_2 = (.625)_{10}$

.101

$1 \times 2^{-3} = 1/2^3 = 0.125$

$0 \times 2^{-2} = 0/2^0 = 0$

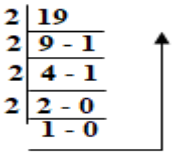
$1 \times 2^{-1} = 1/2 = 0.5$

0.625

1.2.2 Decimal to binary conversion:

A decimal no. can be converted into binary by repeatedly dividing the number by 2 and collecting the remainders.

$(19)_{10} = (10011)_2$



For decimal fraction:

For decimal fractions, the fractional part has to be multiplied by 2 successively and collecting the carriers from top to bottom. The multiplication can be repeated till the fractional part becomes zero. If the fractional part is not zero after 4 or 5 steps the process can be stopped and we have to be satisfied with the nearest value.

$(.625)_{10} = (.101)_2$

.625 x 2 = 1.250 carry is 1

.250 x 2 = 0.500 carry is 0

.500 x 2 = 1.000 carry is 1

$(.101)_2$

1.2.3 Octal number system:

The octal number system has a **base 8.** The basic digits used are 0, 1, 2, 3, 4, 5, 6, and 7. There is no 8 or 9 in the octal number system. The base of the octal number system is **8** and the base of the binary number system is 2. Since $2^3 = 8$, it follows that any octal digit can be represented by a group can be represented by a group of 8 bit binary sequence.

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

1.2.4 Octal to decimal conversion:

The octal to decimal conversion is similar to binary to decimal conversion, only the weights are different. In this case, the weights are $8^0, 8^1, 8^2$ etc. for integer part $8^{-1}, 8^{-2}$ etc. for fractional part. In octal number system we have an octal point to separate the integer and fractional parts.

Example:

$(75)_8 = (61)_{10}$

7 5

$5 \times 8^0 = 5$

$7 \times 8^1 = 56$

61

$(.45)_8 = (.575)_{10}$

.4 5

$5 \times 8^{-2} = 5/8^2 = 0.075$

$4 \times 8^{-1} = 4/8 = 0.5$

0.0575

Decimal to octal conversion:

To convert a decimal number to octal, we have to divide the decimal number by 8 repeatedly and collect the remainder 5 from top to the bottom, the remainders also taken in octal.

Example:

$(68.5)_{10} = (104.4)_8$

8	68
8	8 - 4
8	1 - 0
8	0 - 1

0.5 x 8 = 4.0 carry 4

1.2.5 Octal to binary conversion:

To convert a decimal number to binary we have adopted a procedure of repeatedly dividing the given decimal number by 2. Since the base of octal number system is 8 which is equal to 2^3, to convert an octal number to binary all we have to do is replace each octal digit with its equivalent of any octal number.

Octal	Binary
27	010 111
45.5	100101.101

1.2.6 Binary to octal conversion:

To convert a binary number to octal, we have to arrange the bits into groups of 3 bits starting from least significant bit. If the final group has less than 3 bits, just include a zero for the first group at the front.

For example to convert (10101) into octal arrange the bits as (10 101). Now include a zero for the first group at the front. The binary combination now becomes (010 101). Replace each 3 bit binary group by its equivalent octal digit.

Ie) 010 =2 and 101 = 5 therefore $(10101)_2 = (010\ 101)_2 = (25)_8$.

For integer part groups are formed starting from the octal point and moving towards left. For the fractional part 3 bit groups are formed starting from the octal point and moving towards right.

Example:

$(110.101)_2 = (6.5)_8$

$(10111.1)_2 = (010\ 111.100)_2 = (27.4)_8$

1.2.7 Hexadecimal number system:

Though binary digit are used in computers, it is not a compact code for it requires a large number of bits to represent bigger numbers for example,

$(255)_{10} = (1111\ 1111)_2$ - 8 bits are required

$(1023)_{10} = (11\ 1111\ 1111)_2$ - 10 bits required

$(65,534)_{10} = (1111\ 1111\ 1111\ 1110)_2$ - 16 bit required

It is very difficult to remember and work with such a long binary sequence. Hexadecimal numbers represent a group of bits in hex digits.

The hexadecimal number system has a **base 16**. The basic digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. since $2^4 = 16$.

It follows that any hex digit can be represented by a group of four bit binary sequence.

The table given below has decimal, hexadecimal and the four bit binary equivalence for the decimal numbers 0 to 15.

Decimal	Hexadecimal	Binary $2^3 + 2^2 + 2^1 + 2^0$ 8 + 4 + 2 + 1
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

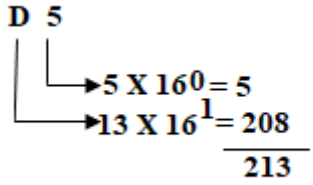
1.2.8 Hexadecimal to decimal conversion:

The hex to decimal conversion is similar to binary to decimal conversion, only the weights are different.

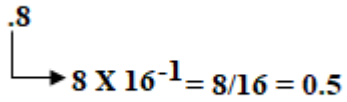
In this case, the weights used are $\mathbf{16^0, 16^1, 16^2}$ etc. for the integer part and $\mathbf{16^{-1}, 16^{-2}}$ etc. for the fractional part. Just as we have decimal point and binary point. In hexadecimal number system we have a hexadecimal point.

Example:

$(D5)_{16} = (213)_{10}$



$(.8)_{16} = (.5)_{10}$

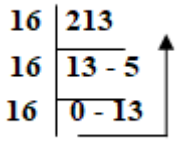


1.2.9 Decimal to hexadecimal conversion:

To convert a decimal number to hex, we have to divide the decimal number by 16 repeatedly and collect the remainders from top to bottom. The remainders also must be taken in hex.

Example:

$(213)_{10} = (D5)_{16}$



$(.625)_{10} = (A)_{16}$

0.625 x 16 = 10.000

10 = A

The fraction .625 is converted into hex by multiplying .625 by 16 and collecting the carry.

1.2.10 Hexadecimal to binary conversion:

To convert a decimal number to binary, since the base of hexadecimal number system is 16 which is equal to 2^4 to convert a hexadecimal number to binary. We have to do is replace each hex digit with its equivalent 4 bit binary.

Example:

$(3A.7)_{16} = (0011\ 1010.0111)_2$

The integer part of $(3A.7)_{16}$, ie) 3A is taken as two digits as two digits 3 and A. we get the binary equivalent as,

$(3A)_H = (0011\ 1010)_2$

The fractional part $(.7)_H$ is converted to binary by taking the four bit equivalent for 7.
$(.7)_H = (.0111)_2$

1.2.11 Binary to hexadecimal conversion:

To convert a binary number to hex, we have to arrange the bits into group of 4 bits starting from LSB. If the final group is less than 4 bits, just include zeros to make it a group of 4 bits. For example, to convert $(100101)_2$ into hex, arrange the bits as $(10\ 0101)_2$. Now include two zeros for the first group at the front. The binary combination now becomes $(0010\ 0101)_2$. In the last step replace each 4 bit binary group by its equivalent hex digit. Therefore, $(100101)_2 = (0010\ 0101)_2 = (25)_H$.

The digit whether it is an integer part or fractional part must be represented by a group of 4 bits. For integer part, 4 bit groups are formed starting from the hexadecimal point and moving towards left. For the fractional part, 4 bit groups are formed starting from the hexadecimal point and moving towards right. Example $(110.101)_2 = (0110.1010)_2 = (6.A)_H$.

1.3 Binary codes:

All digital circuits operate with only two states namely HIGH and LOW or ON and OFF or 1 and 0. In binary number system, the number of bits required goes on increasing as the number become longer and larger. When the number is large the conversion to get the binary equivalent is tedious. Also, some special binary codes are required to represent alphabets and special characters like digital calculators, multimeters, frequency meters etc.

1.3.1 BCD CODES – 8421 code:

A group of bits which are used to represent decimal numbers 0 to 9 are called **Binary coded decimal codes or BCD codes.** The most popular BCD code is the 8421 code. The 8421 indicates the binary weights of the four bits ($\mathbf{2^3 + 2^2 + 2^1 + 2^0}$)**.** Using the four bits with weights we can easily represent the decimal number 0 to 9 as given in the table

From the table, the four bit binary combination given is only the first ten combinations namely 1010, 1011, 1100, 1101, 1110 and 1111 are invalid 8421 BCD codes. Any decimal number greater than 9 can be easily represented in 8421 BDC, by repeated using the four bit code for each digit. Few examples are given below.

Decimal	BCD 8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Decimal	BCD 8421
29	0010 1001
468	0100 0110 1000
97.5	1001 0111 0101

1.3.2 Excess 3 codes:

The excess – 3 codes is another BCD code used in earlier computers. The excess – 3 codes for a decimal digit is obtained by adding 0011(3) to the 8421 BCD code. The excess – 3code also has ten valid codes and six invalid codes.

The six invalid codes are 0000, 0001, 0010, 1100, 1101, 1110 and 1111. The valid excess 3codes are given in the table. Like in 8421 BCD code, in excess 3 code also, if the number is greater than 9. The code is given separately for each digit.

Few examples are given below. We also note that the excess 3 code is not a weighted code and it is also a self-complementing code.

Decimal	BCD 8421	Excess 3 code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Decimal number	Excess 3 codes
26	0101 1001
97	1100 1010
853	1011 1000 0110

1.3.3 Gray codes:

Gray code is another important code that can be used in sequence counting. When the count advances by one, to reduce error the number of changes in the bits has to be kept minimum..

Gray code representation:

When the count advances by one, there is only one change. For example, when the count changes from 3 to 4, the corresponding gray code change is from 0010 to 0110. Let us imagine a mirror placed between thee gray code of 7 to 8. Between 7 and 8 we have drawn a line represents a mirror. Except for the MSB we can see that the lower 8 combinations are mirror images of the upper eight combinations. The gray code for 7 is the mirror image of the gray code for 9. The gray code for 0 is the mirror image of the gray code for 15. For this reason the gray code is called as **reflected binary code.**

Decimal	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

1.3.4 Binary to gray conversion:

To convert a given binary number to its equivalent gray code the following rules are applied.

1. The MSB of the gray code is the same as the MSB of the binary.
2. Coding from left to right, add each adjacent pair of bits to get the next bit of the gray code. Omit the carries.

Example: $(1011)_2 = (1110)_G$

Step 1:

The left MSB in gray code is the same as the MSB of the binary.

1011 Binary
↓
1 Gray

Step 2:

Add the left MSB to the adjacent one.

1 + 0 + 1 1 Binary
↓
1 1 Gray

Step 3:

Add the next adjacent pair.

1 0 + 1 1 Binary
↓
1 1 1 Gray

Step 4:

Add the next adjacent pair and omit the carry.

1 0 1 + 1 Binary
↓
1 1 1 0 Gray

The conversion is now complete $(1011)_2 = (1110)_G$. the suffix G used is to represent Gray code.

1.3.5 Gray to binary conversion:

1. The MSB of the binary is the same as the MSB of the Gray.
2. Coding from left to right, add the binary digit generated to the adjacent Gray bit to get the next bit of the binary omit the carries if occurs.

Example: $(1110)_G = (1011)_2$

Step 1:

The LSB in the binary is the same as the MSB of the Gray.

1 1 1 0 Gray
↓
1 Binary

Step 2:

Add the binary digit generated to the adjacent bit of the Gray code.

1 1 1 0 Gray
↗↓
1 0 Binary

Step 3:

Add the binary digit generated to the adjacent bit of the Gray code.

1 1 1 0 Gray
↗ ↓
1 0 1 1 Binary

The conversion is now complete $(1110)_G = (1011)_2$.

1.3.6 ASCII code:

ASCII stands for American standard code for information interchange. This is 7 bit code used to represent decimal digits 0 to 9, alphabets A to Z both upper & lower case and some special characters that is an alpha numeric code. Since ASCII is a 7 bit code, there are 128(2^7) possible binary combinations. The ASCII codes are listed below. A few examples are given below. The ASCII code includes some characters like DEL (delete), ESC (escape), STX (start of text), ETX (end of text) etc.

ASCII symbol	Decimal	Hex	7 binary
0	48	30	011 0000
1	49	31	011 0001
2	50	32	011 0010
9	57	39	011 1001
:	58	3A	011 1010
?	63	3F	011 1111
A	65	41	100 0001
B	66	42	100 0010
Z	90	5A	101 1010
a	97	61	110 0001
b	98	62	110 0010
z	122	7A	111 1010
DEL	127	7F	111 1111

1.4 Binary arithmetic:

To perform arithmetic operation like addition, subtraction, multiplication and division in binary number system.

1.4.1 Binary addition:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10 (read as 0 with carry 1)

(1 + 1 + 1 is 3 which is $(11)_2$ the third bit is usually the carry. The addition of two binary numbers followed by the rules,

Example:

1. **Add 5 and 6**

   5 ⟶ 0101

   6 ⟶ 0110

   1011

2. **Add 5.125 and 7.75**

   5 ⟶ 0101

   7 ⟶ 0111

   1100

   0.125 x 2 = 0.250 carry 0

   0.250 x 2 = 0.500 carry 0

   0.500 x 2 = 1.000 carry 1

   0.001

   And

   0.75 x 2 = 1.50 carry 1

   0.50 x 2 = 1.00 carry 1

   0.110

   0.001

   0.110

   0.111

   Therefore **5.125 + 7.75 = 1100.111**

1.4.2 Binary subtraction:

To subtract two numbers the following rules are used,

0 – 0 = 0

0 – 1 = 1 read as difference 1 with borrow 1

1 – 0 = 1

1 – 1 = 0

Example:

Subtract 5 from 7

7 ⟶ 0111 –

5 ⟶ 0101

0010

Subtract 3 from 13

Applying the rules of binary subtraction,

13 ⟶ 1101 -

3 ⟶ 0011

1010

The binary results is given above, since a borrow is involved the procedure is done step by step.

1101

0011

0

In the first column LSB, 1 – 1 = 0, in the second column we have to subtract 0 – 1. A 1 has to be borrowing from the third column as shown below.

1101 –

0011

0

After the borrowing, the minuend of second column is taken as 10 and10 – 1 gives 1. Since a 1 has been borrowed from the third column of the minuend, the bit in the third column of borrow.

1001 –

0011

10

The third column 0 – 0 gives 0. The fourth column 1 – 0 gives 1. The final step is,

1001 –

0011

1010

1.4.3 Binary multiplication:

The following rules are used to multiply two binary numbers.

0 x 0 = 0

1 x 0 = 0

0 x 1 = 0

1 x 1 = 1

Multiplication is done in binary. The process involves forming particle products and shifting successive particle products to the left one. Finally all the particle products are added to give the result. Also we find that whenever the multiplier bit is 1, the multiplicand is taken as the particle product and whenever the multiplier bit is 0. Particle product is taken as 0 that is the multiplication is done by shift and add method.

Example:

1101 x 1011

1101

1 1 0 1

0 0 0 0

1 1 0 1

10001111

1.4.4 Binary division:

Binary division has only two rules,

0 ÷ 1 = 0

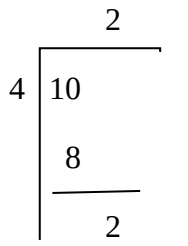
1 ÷ 1 = 1

Division by o is not allowed. The division in binary is done exactly as in decimal.

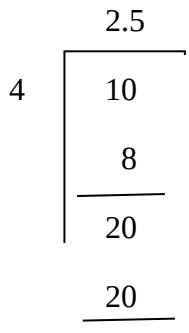
Example:

Divide 10 by 4 in decimal number method

Solution:

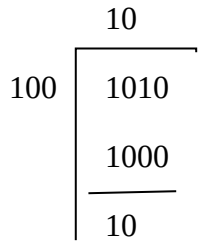


Here the quotient is 2 and the remainder is also 2.

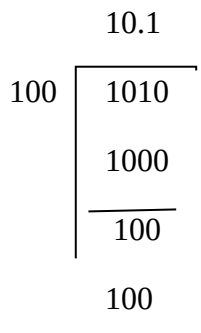


Including the fractional part the quotient is 2.5

Solution in binary method:



Here the quotient is 10(2) and the remainder is also 10(2).



Including the fractional part, the quotient is 10.1(2.5)

1.5 Unsigned binary numbers:

There are two types of complements for binary number system, namely 1's complement and 2's complement can be used to perform subtraction using adder circuits. Also they are used to represent negative numbers.

1's complement:

The 1's complement of the binary digit is defined as 1 minus that bit. Ie) 1's complement of 1 is 1 – 1 = 0 and 1's complement of 0 is 1 – 0 = 1.

However, it is easy to remember that is complement of 0 is 1 and 1's complement of 1 is 0.

In other words 1's complement of any binary number is formed by simply changing all the 1's to0's and all 0's to 1's. Few examples are given below.

Binary number	1's complement
1011	0100
110001	001110
100100	011011
11001110	00110001
10101101	01010010

2's complement:

2's complement of a binary number is formed by adding 1 to the 1's complement of that number.

For example let us find the 2's complement of 1010.

1's complement of 1010 = 0101 +

Add 1 = 1

2's complement of 10101 = 0110

Examples:

Binary number	1's complement	2's complement
1011	0100	0101
110001	001110	001111
100100	011011	011100
11001110	00110001	00110010
10101101	01010010	01010011

1.5.1 Signed binary numbers:

In decimal number system, we use plus sign (+) for positive numbers and a minus sign (-) for negative numbers. But in digital circuits only two binary symbols 0 and 1 are used. To represent the sign of the number, the usual method is to allot a separate bit to indicate the sign of the number.

The bit 0 is used to represent (+) or positive numbers and the bit 1 is used to represent (-) or negative numbers. The bit that is used to represent the sign of the number is called the **sign bit.** The sign bit is the MSB of a binary number. For a +ve number, the MSB is equal to 0 and the remaining bit represent the magnitude. For a negative number, the MSB is equal to 1 and remaining bits represent the magnitude. This type of representation is called **sign magnitude form.**

Decimal	Sign magnitude
+14	0 000 1110
-14	1 000 1110
+95	0 101 1111
-95	1 101 1111
+127	0 111 1111
-127	1 111 1111

Sign 1's complement:

To represent a negative number in 1's complement form, the following two steps are used.

1. Write the +ve binary form of the number, including the sign bit.
2. Invert each bit, including the sign bit ie) take 1's complement.

For example the sign -1's complement representation for -12 is obtained as follows

+ 12 = 0 000 1100

- 12 = 1 111 0011

Sign 2's complement:

To represent a negative number in 2's complement form, the following three steps are used.

1. Write the + ve binary form of the number, including the sign bit.
2. Invert each bit, including the sign bit ie) take 1's complement.
3. Add 1 to the result.

For example the sign -2's complement representation for -12 is obtained as follows.

+12 = 0 000 1100

-12 = 1 111 0011 **1's complement**

Add 1 = 1

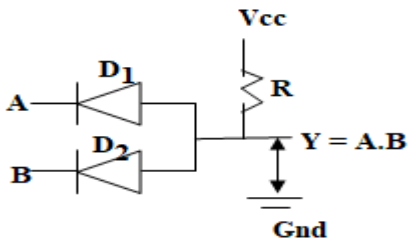
111 0100 **2's complement**

1.6 Logic gates and logic families:

1.6.1 AND circuit using diode:

The circuit for a two input AND gate using diodes is shown in figure. The truth table is given below. Let the inputs be taken as A and B and the output as Y. When A = 0 and B = 0, both the diodes conducts and the output voltage is equal to the voltage drop across the diode. Therefore Y = 0. When A = 0 and B = 1, the first diode conducts and again and the output is equal to the voltage drop across the diode. Therefore Y = 0. The same result is obtained when A = 1 and B = 0. That is Y = 0

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

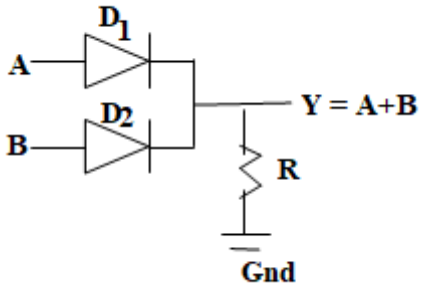


When A = 1 and B = 1, both the diodes do not conduct. Since there is no voltage drop across the resistance the output reaches the supply voltage, that is Y = 1.

1.6.2 OR circuit using diode:

Let us construct a 2 input OR gate using diode logic. The circuit and truth table is given below.

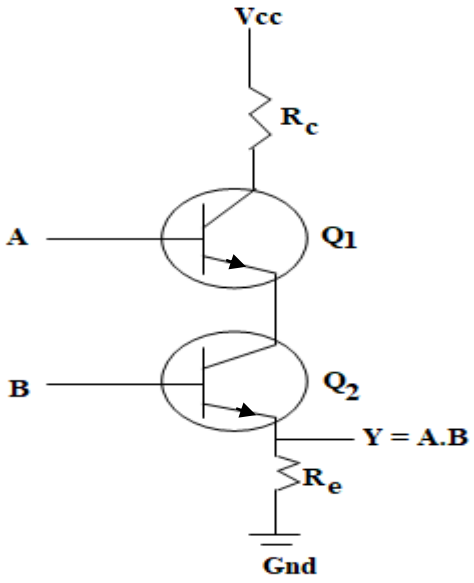
A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1



When A = 0 and B = 0, no current flows through the diodes or resistor. Therefore the output voltage is equal to 0 volts. Therefore Y = 0. When one of the inputs is in 1 state, the diode conducts and produces a voltage drop across the resistor. Which is taken as 1 therefore, for the remaining three combinations Y = 1. The circuit obeys the OR truth table.

1.6.3 AND gate using transistor:

AND gate can also be constructed using transistors shows two inputs. Q_1 and Q_2 are two NPN transistors connected in series, A and B are the inputs applied to the base of the two transistors through the resistors R_1 and R_2. Y is the output taken across the emitter resistance R_e



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

Working:

For two inputs there are four possible combinations,

1. When A = 0, B = 0, no base current flows through Q_1 and Q_2. Hence both the transistors are not conducting. So no current flows through R_e. Hence output voltage is zero ie) Y = 0.

2. When A = 1, B = 0, no base current flows through Q_2. Hence Q_2 is not conducting, even though base current in Q_1 flows, since Q_2 is cut off the circuit is not closed. Hence no current flows through R_e. Hence output is zero ie) Y =0.

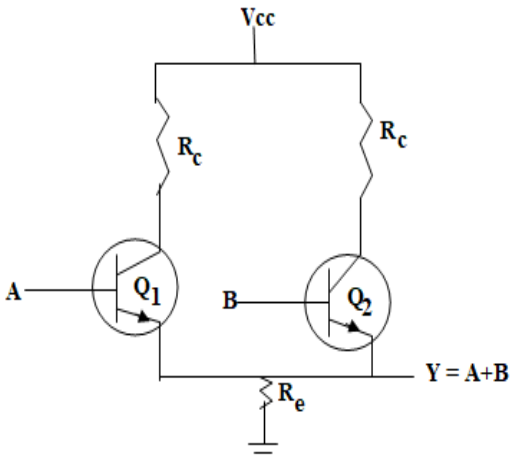
3. When A = 0, B = 1, no base current flows through Q_1. Hence Q_1 is not conducting, even though base current in Q_2 flows, since Q_1 is cut off the circuit is not closed. Hence no current flows through R_e. Hence output is zero ie) Y =0.

4. When A = 1, B = 1, base current flow through Q_1 and Q_2 and so both are switched on. Hence current flows through R_e. Hence there is an output voltage ie) Y =1.

1.6.4 OR gate using transistor:

OR gate can also be constructed using transistors Q_1 and Q_2 connected in parallel, A and B are the inputs applied to the base of Q_1 and

Q_2 through the resistors R_1 and R_2, Y is the output taken across the emitter resistance.



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

Working:

For 2 inputs, there are 4 four possible combinations,

1. When A = 0, B = 0 no base current flows through Q_1 and Q_2. Hence both transistors are not conducting. So no current flows through R_e. Hence the output voltage Y = 0.

2. When A = 1 and B = 0, Q_1 is conducting and Q_2 is cut off. Since Q_1 is conducting emitter current flows through R_e. Hence Y = 1.

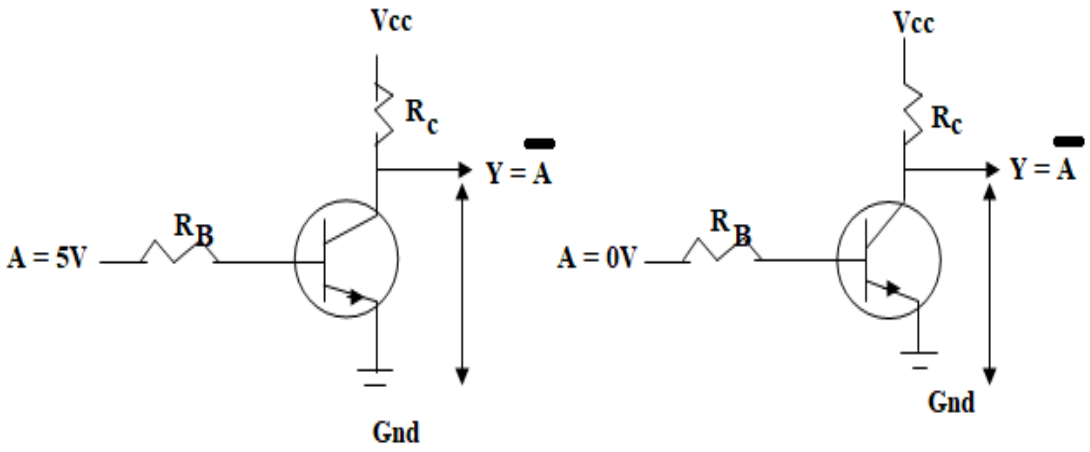
3. When A = 0 and B = 1, then Q_2 is conducting. While Q_1 is cut off ie) since Q_2 is conducting emitter current flows through R_e and hence there is an output ie) Y = 1.

4. When A = 1, B = 1, both the transistors are switched on, and so current flows through the emitter resistance R_e and hence there is an output ie) Y = 1.

1.6.5 NOT using transistor:

The transistor will be only in one of two logic states, namely ON or OFF. The emitter of the transistor is grounded and the collector of the transistor is connected to the positive end of the power supply through a resistor R_c.

The input voltage is applied to the base of the transistor through a resistor R_B. Let the input applied to the base A and the output taken at the collector is Y.



A	Y = A
0	1
1	0

Working:

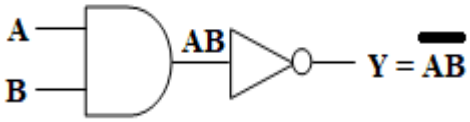
When A = 0 volts (0 state) the base emitter junction is not forward biased and the transistor does not conduct. The collector current is zero and hence there is no voltage drop across the collector resistance R_c. Therefore the output voltage Y at the collector w.r.t ground reaches 5 volts.

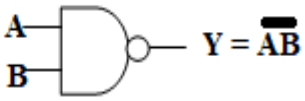
When A = 5 volts (1 state), the base emitter junction is not forward biased and the transistor conducts to saturation. When a transistor conducts to saturation, the base emitter

voltage is 0.7 volt. The collector emitter voltage is 0.2 volt, therefore the output voltage Y at the collector w.r.t ground is 0.2 volts.

When the transistor does not conduct, the output, Y = 5volts. When the transistor conducts to saturation, the output Y = 0.2 volt. The circuit acts as NOT gate or inverter, obeying the truth table. A NOT has one input A and one output Y. the output is the complement of the input. Ie) Y = A.

1.7 NAND gate:

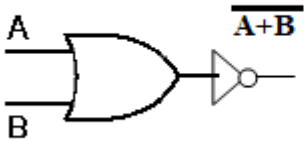


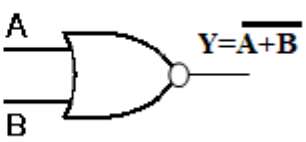


A	B	Y = A.B
0	0	1
0	1	1
1	0	1
1	1	0

NAND is nothing but an AND gate followed by a NOT gate as shown below. The symbol for NAND and its truth table are shown. The output is 1 when any input goes to 0. The output is 0 only when all the inputs are in 1 state.

1.7.1 NOR gate:





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

NOR can be imagined to be an OR gate followed by a NOT gate. The symbol and its truth table are shown above. The output is 0 when any input goes to 1. The output is 1 only when all the inputs are in 0 states.

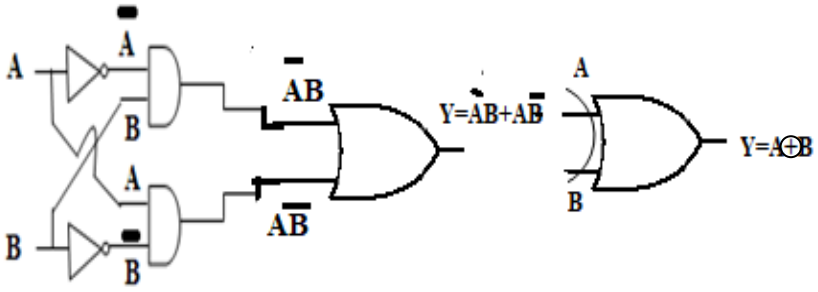
1.7.2 EX – OR gate:

In a two input exclusive or (EX – OR (or) OR), the outputs is high only when one of the inputs is high. The truth table for a two input EX – OR is given below. This is different from the regular OR gate,

For OR gate, when A = 1, B = 1 the output Y = 1

For the EX – OR gate, when A = 1, B = 1 the output Y = 0. The Boolean expression from the given truth table

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



From the table,

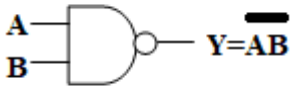
Y = 1 for A = 0, B = 1 the standard product is AB also $\bar{Y}$ = 1 for A = 1, B = 0 the standard product is AB. Therefore the expression is,

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

This expression can be implemented using two AND gates, two NOT and an OR gate. The output is given by the expression,

$$\mathbf{Y = \bar{A}B + A\bar{B}} \text{ also } \mathbf{Y = A \oplus B}$$

1.8 NAND as universal gate:



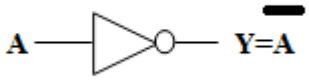
A	B	$Y = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

The symbol for NAND gate and the truth table is shown above. By connecting NAND gates in different ways, it is possible to get the function of any other gate ie) the function of

NOT, AND, OR etc. Therefore, the **NAND gate** is called as **universal gates** or **universal building block.**

1.8.1 NAND as NOT:

A NOT gate has only one input and one output. If the input is A the output is given as A. the symbol and truth table is given below.

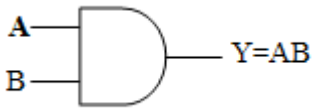


A	Y = $\overline{A}$
0	1
1	0

In a 2 input NAND gate, if both the inputs are marked as $\overline{A}$, then the output is given by $\overline{A}\,\overline{A}$, which is equal to $\overline{A}$. since the two inputs are the same, the 2 inputs of the NAND gate can be connected together and used as a gate with a single input. This arrangement makes the NAND to function as NOT. Since the 2 inputs are connected together and used as a single input, a single line can be used to represent the input. NAND as NOT is shown in figure.

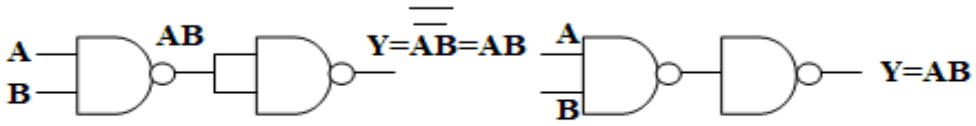
1.8.2 NAND as AND:

The symbol and the truth table for the AND gate is given below.



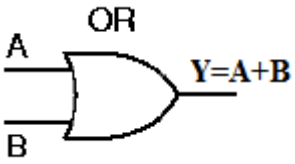
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

The output of the NAND gate is AB. To make it as AB, the output of the NAND gate is inverted once. But the inverter is also implemented using a second NAND gate. The arrangement is shown below.



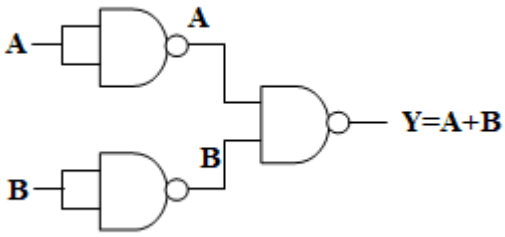
1.8.3 NAND as OR:

The symbol and the truth table for OR gate is given below.



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

To get the OR expression A + B, the inputs are first inverted and then passed through a NAND gate. Three NAND gates are needed to get OR function.



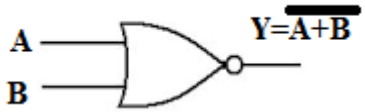
The output of the first NOT gate = A. The output of the second NOT gate = B these two outputs are fed as inputs to a two input NAND gate, the output of the NAND gate AB.

By De Morgan's theorem,

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

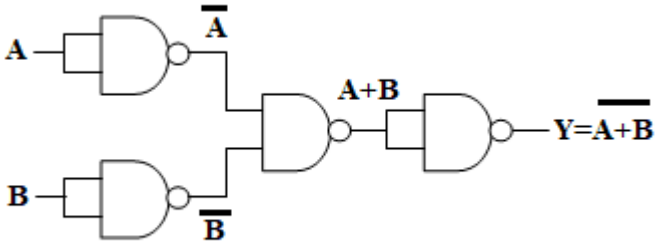
1.8.4 NAND as NOR:

The symbol and the truth table for the NOR gate is given below.



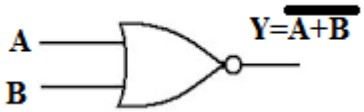
A	B	Y = $\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

The NOR gate is obtained by simply inverting the output of the OR gate.



1.9 NOR as universal gate:

The symbol and the truth table for the NOR gate is given below.

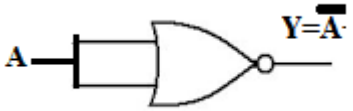


A	B	Y= $\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

1.9.1 NOR as NOT:

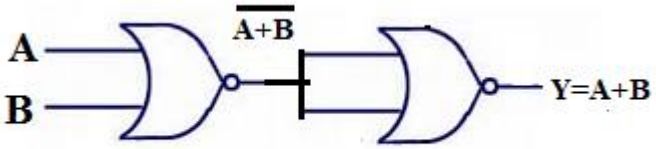
In a two input NOR gate, if both the inputs are marked as A, then the output is given by $\overline{A+A}$ which is equal to A.

Since the two inputs are the same, the two inputs of the NOR gate can be connected together and used as a gate with a single input this arrangement makes the NOR to function as NOT. The arrangement is shown in figure.



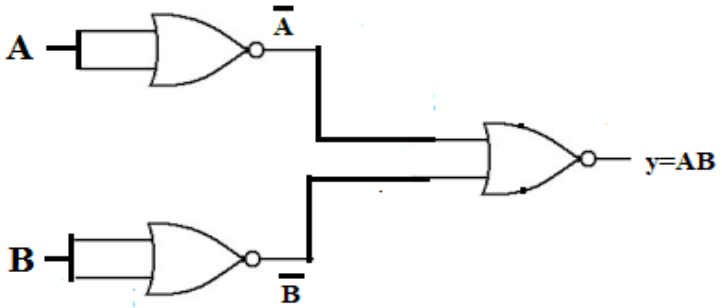
1.9.2 NOR as OR:

The outputs of the NOR gate is $\overline{A+B}$ to make it as A+B, the output of the NOR gate is inverted once. But the inverter is also implemented using a second NOR gate. The arrangement is shown in fig.



1.9.3 NOR as AND:

To get the AND expression A.B, the inputs are first inverted and then passed through a NOR gate. This arrangement is shown in figure. Three NOR gates are needed to get the AND function.



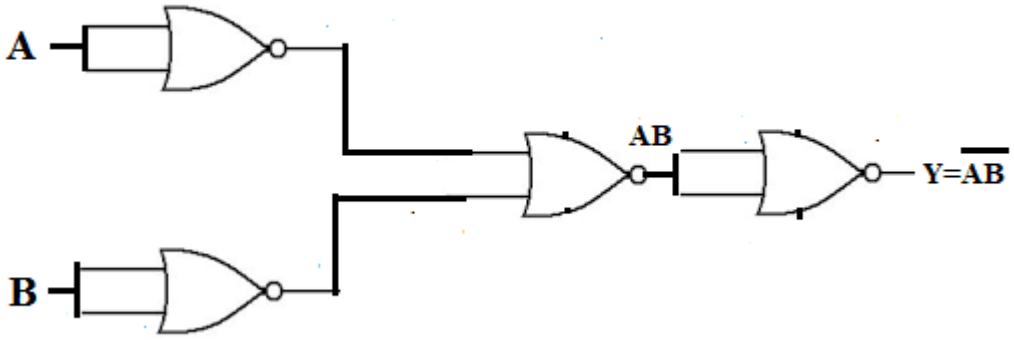
The output of the first NOT gate = $\overline{A}$, the output of the second NOT gate = $\overline{B}$. These two outputs are fed as inputs to a two input NOR gate. The output of NOR gate = $\overline{\overline{A}+\overline{B}}$.

By DE Morgan's theorem,

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

1.9.4 NOR as NAND:

The NAND gate is obtained by simply inverting the output of the AND gate.



References:

1. **Digital electronics by Vijayendran**
2. **Malvino and Leech, (2000), Digital Principles and Application, 4th Edition, Tata McGraw Hill, New Delhi. 2.**
3. **Millman and Halkias, (1972), Integrated Electronics, International Edition, McGraw Hill, New Delhi.**
4. **Arul Thalapapathi, Fundamentals of digital computers, Comptek publishers, Chennai, 1995.**
5. **Digital fundamentals – Floyd – Pearson Education 8th Edition 2004 S Chand Publications**

Referred links:

1. **https://www.geeksforgeeks.org/digital-electronics-logic-design-tutorials/**
2. **https://www.javatpoint.com/conversion-of-number-system-in-digital-electronics**
3. **https://learnabout-electronics.org/Digital/dig12.php**
4. **https://pages.uoregon.edu/rayfrey/DigitalNotes.pdf**
5. **https://www.cl.cam.ac.uk/teaching/0708/DigElec/Digital_Electronics_pdf.pdf**
6. **http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/Lectures/Lecture%2011%20-%20Logic%20gates%20and%20Boolean%20(x1).pdf**

Important questions:

2 mark:

1. Convert $(83.525)_{10}$ to Hexadecimal.
2. Convert $(11100011)_2$ into octal & hexadecimal equivalent.
3. Convert $(78.45)_8$ octal into decimal
4. Find the BCD codes for 29, 45.90 and 789.
5. Convert binary to gray (1110 0100)
6. Add 10 and 50
7. Draw the symbol for AND gate with its truth table.
8. Draw the circuit for NOT gate using transistor.
9. The excess 3 code for 785 and 67.
10. Convert 30 decimal to binary.

5 mark:

1. Explain the working of AND, OR gate using transistor.
2. Subtract $(1110)_2$ from $(1010)_2$ using 1's and 2's complement method.
3. Multiply 1100 X 1010
4. Divide 781 by 20
5. With neat circuit diagram explain OR and AND gates by using diodes.
6. Explain ASCII codes

10 mark:

1. Show that both NAND and NOR as universal gates.
2. Explain the following
    1. Excess 3 codes
    2. Gray codes
    3. BCD codes