Paper Name: Digital Electronic Circuits

Paper Code:EC(EE)302

Number Systems

Radix →It specifies the number of symbols used for corresponding number system.

Radix points →The generalized form of a decimal point. In any positional number system, the radix point makes the dividing line between integer and factional part.

Radix point

N_r = [Integer part . Fractional part]

$N_r = (a_{n-1}, a_{n-2}, \ldots\ldots, a_1, a_0 . a_{-1}, a_{-2}, \ldots\ldots a_{-m})_r$

Positional Weight $r^{n-1}\ r^{n-2}\ \ldots\ r^1\ r^0\ r^{-1}\ r^{-2}\ldots\ldots\ldots r^{-m}$

Where r = Radix (or base of the number system
a = a digit in the set
n = number of digits in integer part
m = number of digits in fractional part
a_{n-1} = most significant digit (MSB)
a_{-m} = Least significant digit (LSB)
} In case of binary number systems

Decimal Number Systems

Radix or Base = 10
Symbols = 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9

Binary Number Systems

Radix or Base = 2
Symbols = 0 and 1
Bit = binary digit 0 or 1
Nibble = a group of four bits.
Byte = a group of **eight bits** or **two nibbles**.

Octal Number Systems

Radix or Base = 8
Symbols = 0, 1, 2, 3, 4, 5, 6 and 7

Hexa-decimal Numbers Systems

Radix or Base = 16
Symbols = 0 to 9 and A to F
A = 10, B = 11, C = 12, D = 13, E = 14 and F = 15

Number System Conversions

The conversion of number systems are given below

Decimal to Binary

Any decimal number consists of integer part and fractional part. The separate conversion procedures are followed for integer and fractional part of decimal number system.

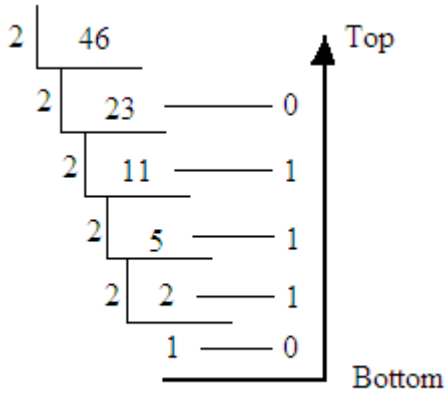
Steps for conversion of decimal integer

The following steps are followed to convert the integer part of decimal number to integer part of binary number systems; the successive division method is used to this conversion of decimal to binary number system.

Step 1 → The given decimal integer is divided by 2, leave a remainder
Step 2 → Divide the quotient obtained from the step 1 will leave a remainder
Step 3 → Repeat *step 2* until the quotient is less than the base 2.
Step 4 → Collect the remainders from bottom to top, to get the equivalent binary number.

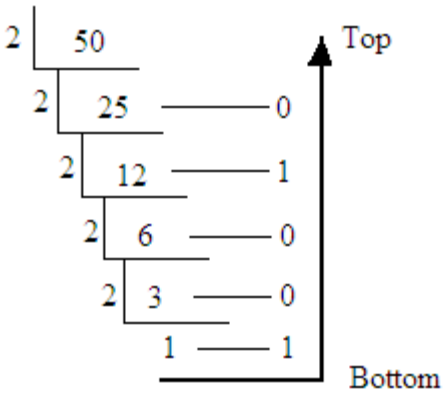
Example:

1. **Convert the decimal number 46 to its equivalent binary number**



The equivalent Binary number is
$(46)_{10} = (101110)_2$

2. **Convert the decimal number 50 to its equivalent binary number**



The equivalent Binary number is
$(50)_{10} = (110010)_2$

Steps for conversion of decimal fraction

This conversion has done by successive multiplication method. First the fractional part is multiplied by 2 to give an integer and a fraction. The new fraction is multiplied by 2 to give a new integer and a new fraction. This process is continued until the fraction becomes 0 or until the number of digit have significant accuracy.

Example:

1. **Convert the decimal number $(0.2)_{10}$ to its equivalent binary number**

		Product					
		Integer		Fraction		Separated Integer	
0.2×2	=	0	.	4	→	0	Top
0.4×2	=	0	.	8	→	0	↓
0.8×2	=	1	.	6	→	1	↓
0.6×2	=	1	.	2	→	1	Bottom

So the equivalent binary number is $(0.2)_{10} = (0.0011)_2$

Exercise → Convert $(25.75)_{10}$ to equivalent binary number [ans: $(11001.110)_2$]

Binary to Decimal Conversion

A binary number can be converted to decimal by forming the sum of the powers of 2 of those coefficients whose value is 1. For example

$(1010.011)_2 = 2^3 + 0^2 + 2^1 + 0^0 + 0^{-1} + 2^{-2} + 2^{-3} = (10.375)_{10}$

Octal and Hexadecimal number to Binary

The conversion from one to binary, octal and hexadecimal plays an important part in digital computers. Since $2^3 = 8$ and $2^4 = 16$, each octal digit corresponds to three binary digits and each hexadecimal digit corresponds four binary digits. The conversion from binary to octal is easily accomplished by partitioning the binary numbers in to groups of three digits each, starting from the binary point and proceeding to the left and to the right. The corresponding octal digits is then assigned to each group.

$$(\underset{6}{1\,1\,0}\ \underset{3}{0\,1\,1}\ \underset{1}{0\,0\,1}\ \underset{4}{1\,0\,0}\ .\ \underset{3}{1\,0\,1}\ \underset{4}{1\,0\,0}\ \underset{7}{1\,1\,1})_2 = (6314.547)_8$$

Binary to Octal

$$(\underset{2}{1\,0}\ \underset{C}{1\,1\,0\,0}\ \underset{6}{0\,1\,1\,0}\ \underset{6}{0\,1\,1\,0}\ .\ \underset{F}{1\,1\,1\,1}\ \underset{2}{0\,0\,1\,0})_2 = (2C66.F2)_{16}$$

Binary to Hexadecimal

Conversion from octal or hexadecimal to binary is done by a procedure reverse to the above mention procedure. Each octal digit is converted to its *three digit binary equivalent*. Similarly, each hexadecimal digit is converted to its *four digit binary equivalent*.

$$(673.124)_8 = (\underset{6}{1\,1\,0}\ \underset{7}{1\,1\,1}\ \underset{3}{0\,1\,1}\ .\ \underset{1}{0\,0\,1}\ \underset{2}{0\,1\,0}\ \underset{4}{1\,0\,0})_2$$

Octal to Binary

$$(306.D)_{16} = (\underset{3}{0\,0\,1\,1}\ \underset{0}{0\,0\,0\,0}\ \underset{6}{0\,1\,1\,0}\ .\ \underset{D}{1\,1\,0\,1})_2$$

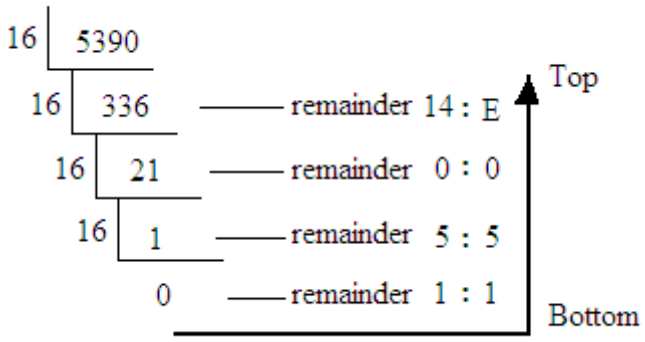
Hexadecimal to Binary

Decimal to Hexadecimal

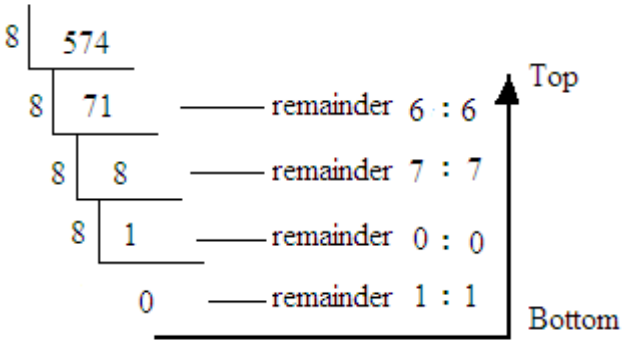
There are two methods for this conversion. One is to convert the decimal number into binary number and then convert the binary number into hexadecimal.

The direct method is done by successive division by 16 and the write the hexadecimal equivalents of remainders.

Example → Convert 5390_{10} into Hexadecimal



So the hexadecimal equivalent of 5390_{10} is $150E_{16}$



Decimal to octal

It is done through successive division by 8 and by writing the remainders. The remainders when read upwards give the octal number.

So the octal equivalent of $(574)_{10}$ is $(1076)_8$

BCD (Binary Coded Decimal)

A weighted binary code is one in which each number carries a certain weight. A string of four bits is known as *nibble*. BCD means that each decimal digit is represented by a nibble. Many BCD codes have been proposed. Among them 8421 is the most predominant BCD code. The designation 8421 indicates the weights of the four bits starting from the left most bit.

So in BCD 0101 means 5_{10} because $0101 = 8\times0+1\times4+0\times2+1\times1 = 5_{10}$

Gray Code

It is an unweighted code. The most important characteristics of this code is that only one single bit change occurs when going from one code number to next.

Decimal	Binary	Gray Code
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

Binary to Gray

Step 1 → The MSB is the same for both the codes

1010

↓

Gray 1

Step 2 → Add the left most bit to the adjacent bit. Discard carry.

(1+0)10 – Binary

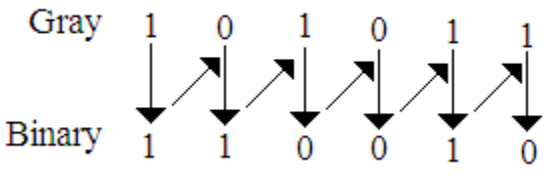
Gray 11

Continue the above process till completion.

1010 – Binary

1111 – Gray

Gray to Binary



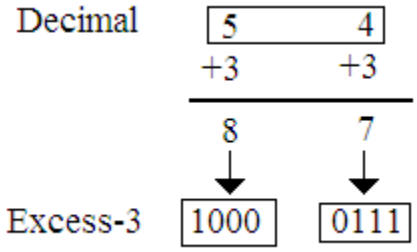
in all the addition always discard the carry.

So $(101011)_{Gray} = (110010)_{Binary}$

Excess-3 Code

Excess -3 is an unweighted code obtained by adding three to each decimal digit and then converting the result to four bit Binary (BCD).

Convert 54_{10} to excess-3 code



Binary Arithmetic

Binary Addition

0 + 0 = 0
0 + 1 = 1

1 + 0 = 1
1 + 1 = 0 with a carry 1 to the next higher bit
1 + 1 + 1 = 1 with a carry 1 to the next higher bit

Add $101110_2 + 1101011_2$

$$\begin{array}{rcl} 1\,0\,1\,1\,1\,0 & \longrightarrow & 46_{10} \\ +\ 1\,1\,0\,1\,0\,1\,1 & \longrightarrow & 107_{10} \\ \hline 1\,0\,0\,1\,1\,0\,0\,1 & \longrightarrow & 153_{10} \end{array}$$

Binary Subtraction

0 - 0 = 0
1 - 0 = 1
1 - 1 = 0
0 - 1 = 0 with a borrow 1 to the next bit

Subtract 1000_2 from 10101_2

$$\begin{array}{rcl} 1\,0\,1\,0\,1 & \longrightarrow & 21_{10} \\ -\ 1\,0\,0\,0 & \longrightarrow & 8_{10} \\ \hline 0\,1\,1\,0\,1 & \longrightarrow & 13_{10} \end{array} \qquad \begin{array}{rcl} 1\,0\,0\,0 & \longrightarrow & 8_{10} \\ -\ 0\,1\,1\,1 & \longrightarrow & 7_{10} \\ \hline 0\,0\,0\,1 & \longrightarrow & 1_{10} \end{array}$$

Binary Multiplication

0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1

Signed Binary numbers

1. Sign Magnitude form

   To represent negative numbers in the binary system digit '0' is used for the '+' sign and digit '1' for the '-' sign. The MSB is the sign bit followed by the magnitude bits.

   -8 = 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 (16 bit representation)

   Sign bit Magnitude bits

2. Two's Compliment representation

-8 = 1 1 1 1 1 0 0 0 (8 bit representation)

1's and 2's Compliments of Binary Numbers

1's compliment subtraction

(i) M = 1010100; N = 1000100. Perform (M-N)

M = 1 0 1 0 1 0 0
1's compliment of N = 0 1 1 1 0 1 1 +
1 0 0 0 1 1 1 1
End around carry → 1 +
0 0 1 0 0 0 0 answer $[1\ 0\ 0\ 0\ 0]_2$

(ii) M = 1000100; N = 1010100. Perform (M-N)

M = 1 0 0 0 1 0 0
1's compliment of N = 0 1 0 1 0 1 1 +
1 1 0 1 1 1 1
No carry
Take the 1's compliment and put a minus sign '-' before answer – $(10000)_2$

2's compliment subtraction

(i) M = 1010100; N = 1000100. Perform (M-N)

M = 1 0 1 0 1 0 0
2's compliment of N = 0 1 1 1 1 0 0 +
1 0 0 1 0 0 0 0
End carry
Discard it, so answer $(10000)_2$

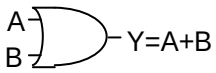
A	B	Y
0	0	0
0	1	1

(ii) M = 1000100; N = 1010100. Perform (M-N)

M = 1 0 0 0 1 0 0
2's compliment of N = 0 1 0 1 1 0 0 +
1 1 1 0 0 0 0
No carry
Take the 2's compliment and put a minus sign '-' before answer – $(10000)_2$

Logic GATES

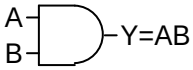
OR GATE



Truth Table

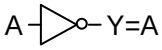
1	0	1
1	1	1

AND GATE



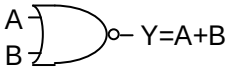
Truth Table

NOT GATE



Truth Table

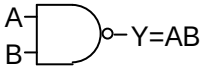
NOR GATE



Truth Table

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

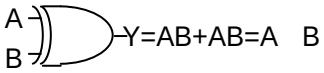
NAND GATE



Truth Table

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Exclusive OR / Ex-OR GATE



Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Boolean Algebra

- Only CAPITAL letters are used
- $\bar{A}$ is the compliment of A

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Laws of Boolean algebra

1. **Cumulative Law**

   A + B = B + A

   A.B = B.A

A	Y
0	1
1	0

2. **Associative Law**

A+(B+C) = (A+B)+C

A.(B.C) = (A.B).C

3. **Distributive Law**

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

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

4. **OR Law**

A+A = A A+ 1 = 1

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

5. **AND Law**

A.A = A A.1 = A

A.0 = 0 A. $\bar{A}$ = 0

6. **Others**

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

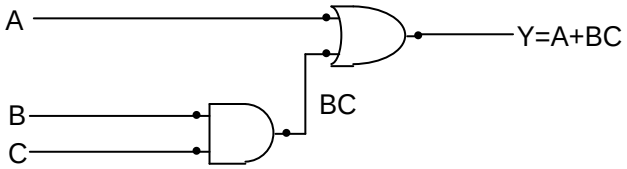
A + $\bar{A}$ B = A + B

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

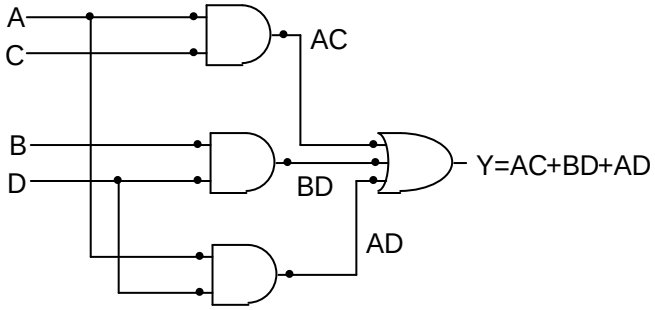
Boolean Variable → A variable having only two possible values, such as high or low.

Boolean Algebra → A system of algebra that operates on *Boolean variables*.

Boolean expression from logic diagram →



Logic diagram from Boolean expression →Y=AC+BD+AD



Truth table from logic diagram (or) Boolean expression → Y=C+AB

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0

1	0	1	1
1	1	0	1
1	1	1	1

Demorgan's Theorem →

Theorem 1 → It states that the compliment of a product of variables is equal to the sum of the compliment of the variables.

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

Theorem 2 → It states that the compliment of a sum of variables is equal to the product of the compliment of the variables.

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

Standard forms of Boolean expression →

- **Product term →**A term in a Boolean expression where where one or more compliment or un-compliment variables are ANDed.
- **Sum term →** A term in a Boolean expression where where one or more compliment or un-compliment variables are ORed.
- **Sum of Products (SOP) →** A type of Boolean expression where several product terms are summed (ORed) together.

$$Z=AB+BC+\overline{A}CD$$

- **Product of Sum (POS) →** A type of Boolean expression where several sum terms are multiplied (ANDed) together.

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

- **Minterm →** Product of Boolean expression where all possible variables appear once in compliment or un-compliment form.
  0 – represent compliment variable
  1 – represent un-compliment variable
  Designated by m_0, m_1, m_2 ……..

- **Maxterm →** A sum term in a Boolean expression where all possible variables appear once in compliment or un-compliment form.
  1 – represent compliment variable
  0 – represent un-compliment variable
  Designated by M_0, M_1, M_2 ……..

- **Canonical form for Sum of Product Expression →**It is define as the logical sum of all the minterms derived from the rows of a truth table, for which the value of the function is1.

$Z=\sum m(0, 10, 15)$

$= m_0 + m_{10} + m_{15}$

$= \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + ABCD$

- **Converting Product terms to standard SOP →** If some product terms are not contain all the variables it is ANDed with an expression such as $(X+\bar{X})$ where X is the missing variable.

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

  1st term $A=A(B+\bar{B})(C+\bar{C}) = (AB + A\bar{B})(C+\bar{C}) = ABC+AB\bar{C}+A\bar{B}C+A\bar{B}\bar{C}$

  2nd term $\bar{B}C(A+\bar{A}) = A\bar{B}C + \bar{A}\bar{B}C$

  $\therefore$ F = 1st term + 2nd term $= ABC+AB\bar{C}+A\bar{B}C+A\bar{B}\bar{C} + A\bar{B}C + \bar{A}\bar{B}C$

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

  $= m_7 + m_6 + m_5 + m_4 + m_1$

  Canonical form $F=\sum m(1, 4, 5, 6, 7)$

- **Canonical form for POS expression →**

  $Z = P(1, 2, 4, 7)$

  $= M_1 M_2 M_4 M_7$

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

- **Truth table for Minterm & Maxterm representation (for 3 variables)**

A	B	C	Minterm	Designation	Maxterm	Designation
0	0	0	$\bar{A}\bar{B}\bar{C}$	m_0	$A+B+C$	M_0
0	0	1	$\bar{A}\bar{B}C$	m_1	$A+B+\bar{C}$	M_1
0	1	0	$\bar{A}B\bar{C}$	m_2	$A+\bar{B}+C$	M_2
0	1	1	$\bar{A}BC$	m_3	$A+\bar{B}+\bar{C}$	M_3
1	0	0	$A\bar{B}\bar{C}$	m_4	$\bar{A}+B+C$	M_4
1	0	1	$A\bar{B}C$	m_5	$\bar{A}+B+\bar{C}$	M_5
1	1	0	$AB\bar{C}$	m_6	$\bar{A}+\bar{B}+C$	M_6
1	1	1	ABC	m_7	$\bar{A}+\bar{B}+\bar{C}$	M_7

- **Converting SOP to POS →**
  - **Step 1 →** Evaluate each product term in the SOP expression. Write the binary equivalent to the given variable of each product term (assign '0' is compliment and '1' is un-compliment).
  - **Step 2→** Determine all the binary numbers not included in the evaluation in step 1.

- **Step 3** → Write the equivalent sum terms for each binary number from step 2 (assign '1' is compliment and '0' is un-compliment).

Example → Convert the following SOP expression to its equivalent POS expression.

SOP expression = $\bar{A}\bar{B}\bar{C}+\bar{A}B\bar{C}+\bar{A}BC+A\bar{B}C+ABC$

$= 000 + 010 + 011 + 101 + 111$

Remaining term = 001, 100 and 110

So the equivalent POS expression = $(A+B+\bar{C})(\bar{A}+B+C)(\bar{A}+\bar{B}+C)$