Digital Logic (BIT 103)

Full Marks: 60 + 20 + 20

Pass Marks: 24 + 8 + 8

Text Book: M. Morris Mano: Digital |Logic and Computer Design

Number system is concerned with different numbers and representation of that number in different system. It is the set of symbols used to express quantities as the basis for counting, determining order, comparing amounts, performing calculations and representing value. The given number system is determined by the base value of that number.

Base/Radix is defined as the total number of digits available in the number system. On the basis of base the number system is categorized by:

- Decimal Number System
- Binary Number system
- Octal Number System
- Hexadecimal Number System

Decimal Number System: The number system that uses 10 digits from 0 to 9 is known as decimal number system and its base value is 10.

Binary Number System: The number system that consists of only two digits i.e. binary digits 0 and 1 and whose base value is 2 is called binary number system.

Octal number system: Number system that consists of numbers from 0 to 7 is known as octal number system. Its base value is 8.

Hexadecimal Number System: Number system that consists of 16 digits, ten numbers: 0 to 9 and six letters: A, B, C, D, E and F. The letters A, B, C, D, E and F represents the decimal numbers 10, 11, 12, 13, 14 and 15. Its base value is 16.

Equivalence of numbers in different Number Systems:

Decimal	Binary	Hexadecimal	Octal
0	000	0	0
1	001	1	1
2	010	2	2
3	011	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Number Base Conversion

➢ **Decimal to Binary:** Steps to convert a Decimal number to its binary equivalent.

  Conversion of Integer part:

I. Divide the given integer part of decimal number by 2 successively and write down the remainders until the quotient is zero.

II. Write all the remainders starting with the MSB (Most Significant Bit) i.e. from bottom to LSB (Least Significant Bit) i.e. top.

  Conversion of Fractional part:

I. Multiply the given fractional part of decimal number by 2 successively until the fractional part becomes zero.

II. Note down the integer part starting from first.

        Note: If fractional part does not become zero then, result has been taken up to 6 places.

Example: Convert $(25.125)_{10}$ into binary

For Integer part:

2	25	1	(Least Significant Bit)
2	12	0	
2	6	0	
2	3	1	
2	1	1	(Most Significant Bit)
	0		

$$(11001)_2$$

For Fractional part:

0.125	0.25	0.5
×2	×2	×2
0.250	0.50	1.0
↓	↓	↓
0	0	1

∴ $(25.125)_{10} = (11001.001)_2$

➢ **Binary to decimal:** Steps to convert a Binary number to its Decimal equivalent.

Conversion of Integer part:

I. Multiply each digit of binary numbers with its place value. The place value of binary number is positive power of 2.

II. Add all the products of multiplication. The total number is the decimal equivalent number of this binary number.

Conversion of Fractional part:

I. Multiply first digit of positional value by 2^{-1} and so on up to last fractional digit.

Example: Convert $(1101.011)_2$ into decimal.

Here;

    Given $(1101.011)_2$

    $= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$

    $= 8 + 4 + 0 + 1 + 0 + 1/4 + 1/8$

    $= 13 + 0.25 + 0.125$

    $= (13.375)_{10}$

➢ **Decimal to Octal:** Steps to convert a Decimal number to its octal equivalent.

Conversion of Integer part:

I) Divide the given integer part of decimal number by 8 successively and write down the remainders until the quotient is zero.

II) Write all the remainders starting with the MSB (Most Significant Bit) i.e. from bottom to LSB (Least Significant Bit) i.e. top.

Conversion of Fractional part:

I) Multiply the given fractional part of decimal number by 8 successively until the fractional part becomes zero.

II) Note down the integer part starting from first.

Note: If fractional part does not become zero then, result has been taken up to 6 places.

Example: Convert $(63.0625)_{10}$ into Octal

For Integer part:

$$
\begin{array}{c|cc}
8 & 63 & 7 \quad \text{(Least Significant Bit)} \\
\hline
8 & 7 & 7 \quad \text{(Most Significant Bit)} \\
\hline
& 0 &
\end{array}
$$

For Fractional part:

$$
\begin{array}{cc}
0.0625 & 0.5 \\
\times 8 & \times 8 \\
\hline
0.5000 & 4.0 \\
\downarrow & \downarrow \\
0 & 4
\end{array}
$$

$\therefore (63.0625)_{10} = (77.04)_8$

➤ **Octal to Decimal:** Steps to convert Octal number to its Decimal equivalent.

Conversion of Integer part:

I) Multiply each digit of Octal number with its place value. The place value of octal number is positive power of 8.

II) Add all the products of multiplication. The total number is the decimal equivalent number of this binary number.

Conversion of Fractional part:

I) Multiply first digit of positional value by 8^{-1} and so on up to last fractional digit.

Example: Convert $(61.25)_8$ into decimal.

Here;

Given,

$(61.25)_8$

$= 6 \times 8^1 + 1 \times 8^0 + 2 \times 8^{-1} + 5 \times 8^{-2}$

$= 48 + 1 + 2/8 + 5/64$

$= 49 + 0.25 + 0.078125$

$= (49.328125)_{10}$

➢ **Binary to Octal:** Steps to convert Binary number to its octal equivalent.

   I)   As $8 = 2^3$, for binary to octal conversion groups of 3 binary bits each are formed in the binary number.

  II) After forming groups, each group of three binary bits is converted to its octal equivalent.

 III) For integer part of the binary number, the group of three bits is formed from right to left. In the binary fraction the group of three bits is formed from left to right.

  Example: Convert $(10011010.10001101)_2$ into Octal

Here;

    Given,

      $(10011010.10001101)_2$

      $= 010 \quad 011 \quad 010 \,.\, 100 \quad 011 \quad 010$

      $= (232.432)_8$

➢ **Octal to Binary:** To convert Octal number to its Binary equivalent, each digit of given octal number is directly converted to its 3 – bit binary equivalent.

Example: Convert $(57.342)_8$ into Binary

Here;

Given,

$(57.342)_8$

$= 101 \quad 111 . 011 \quad 100 \quad 010$

$= (101111.011100010)_2$

➢ **Decimal to hexadecimal:** Steps to convert a decimal number to its hexadecimal equivalent.

Conversion of Integer part:

I. Divide the given integer part of decimal number by 16 successively and write down the remainders until the quotient is zero.

II. Write all the remainders starting with the MSB (Most Significant Bit) i.e. from bottom to LSB (Least Significant Bit) i.e. top.

Conversion of Fractional part:

I. Multiply the given fractional part of decimal number by 16 successively until the fractional part becomes zero.

II. Note down the integer part starting from first.

Note: If fractional part does not become zero then, result has been taken up to 6 places.

Example: Convert $(952.62)_{10}$ into hexadecimal
For Integer part:

| 16 | 952 | 8 | (Least Significant Bit) |

| 16 | 59 | 11 |

| 16 | 3 | 3 | (Most Significant Bit) |

| 0 |

0.62	0.92	0.72	0.52	0.32	0.12
×16	×16	×16	×16	×16	×16
9.92	14.72	11.52	8.32	5.12	1.92
↓	↓	↓	↓	↓	↓
9	14	11	8	5	1

$\therefore \quad (952.62)_{10} = (3B8.9EB851)_{16}$

➢ **Hexadecimal to decimal:** Steps to convert a hexadecimal number to its decimal equivalent.

   Conversion of Integer part:

I. Multiply each digit of hexadecimal numbers with its place value. The place value of binary number is positive power of 16.

II. Add all the products of multiplication. The total number is the decimal equivalent number of the hexadecimal number.

   Conversion of Fractional part:

I. Multiply first digit of positional value by 16^{-1} and so on up to last fractional digit.

Example: Convert $(6E.2B)_{16}$ into decimal.

Here;

   Given $(6E.2B)_{16}$

   $= 6 \times 16^1 + 14 \times 16^0 + 2 \times 16^{-1} + 11 \times 16^{-2}$

   $= 96 + 14 + 2/16 + 11/256$

   $= 110 + 0.125 + 0.4296875$

   $= (110.554688)_{10}$

➢ **Binary to Hexadecimal:** Steps to convert Binary number to its Hexadecimal equivalent.

    I)   As $16 = 2^4$, for binary to hexadecimal conversion groups of 4 binary bits each are formed in the binary number.

    II) After forming groups, each group of four binary bits is converted to its hexadecimal equivalent.

    III) For integer part of the binary number, the group of four bits is formed from right to left. In the binary fraction the group of four bits is formed from left to right.

  Example: Convert $(10011010.10001101)_2$ into Hexadecimal

Here;

    Given,

      $(1001101011.10001101)_2$

      $= 0010\ \ 0110\ \ 1011.\ 1000\ \ 1101$

      $= (26B.8D)_{16}$

➢ **Hexadecimal to Binary:** To convert from hexadecimal to its binary equivalent, each digit of the given hexadecimal number is converted to its 4-bit binary equivalent.

Example: Convert $(6D.3A)_{16}$ into binary

Here;

    Given,

       $(6D.3A)_{16}$

      $= 0110 \quad 1101.0011 \quad 1010$

      $= (1101101.00111010)_2$

➤ **Octal to Hexadecimal:** Steps to convert from octal to its hexadecimal equivalent.

   i) Each digit of given octal number is converted into its 3-bit binary equivalent.

   ii) Now, form the groups of 4 binary bits to obtain its hexadecimal equivalent.

Example: Convert $(46.57)_8$ into binary

Here;

   Given,

$$(46.57)_{16}$$

   Converting 46.57 first to its 3-bit binary equivalent we get

     46.57 = 100   110.101   111

   Now forming the groups of 4 binary bits to obtain its hexadecimal equivalent we get,

     100110.101111 = 0010   0110.1011   1100 = 2 6 11 12

$$= (26.BC)_{16}$$

➤ **Hexadecimal to octal:** Steps to convert from hexadecimal to its octal equivalent.

   i) Each digit of given hexadecimal number is converted into its 4-bit binary equivalent.

   ii) Now, form the groups of 3 binary bits to obtain its octal equivalent.

Example: Convert $(5B.3A)_{16}$ into binary

Here;

   Given,

      $(5B.3A)_{16}$

   Converting 5B.3A first to its 4-bit binary equivalent we get,

    5B.3A = 0101   1011.0011   1010

   Now forming the groups of 3 binary bits to obtain its octal equivalent we get,

    01011011.00111010 = 001   011   011.001   110   100

        $= (133.164)_8$

Binary Addition:

Rules:

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	0 (Carry over 1)

Example:

```
     1  1  0  1
   + 1  1  1  1
   _____
   1  1  1  0  0
```

Binary Subtraction:

Rules:

A	B	A-B
0	0	0
0	1	1 (Borrow 1)
1	0	1
1	1	0

Example:

```
     1  1  0  0  1
   - 1  0  1  1  0
   _____
     0  0  0  1  1
```

Binary Multiplication:
Rules:

A	B	A*B
0	0	0
0	1	0
1	0	0
1	1	1

Example:

```
        1 1 0 1
      X 1 0 1
    -----------
        1 1 0 1
      0 0 0 0
  + 1 1 0 1
  -----------------
  1 0 0 0 0 0 1
```

Division:

Example: 11101 / 101

```
101 ) 1 1 1 0 1 ( 1 0 1
      - 1 0 1
      ---------
        0 1 0 0
        - 0 0 0
        ---------
          1 0 0 1
          - 1 0 1
          ---------
            1 0 0
```

R = 100
Q = 101

Octal addition:

Example: Add 307 +206

$$\begin{array}{r} 307 \\ +\ 206 \\ \hline 51\ 5 \end{array}$$

Hexadecimal addition

$$\begin{array}{r} 1\ A\ 5 \\ +\ 2\ 3\ B \\ \hline 3\ E\ 0 \end{array}$$

Complements: Complements are used in digital computer to perform subtraction.

There are two types of complements for each base – r system.

   1. r's complement

   2. (r – 1)'s complement

1. r's complement: For the given positive number N in base – r with an integer part of n – digits the r's

   complements of N is defined as r^n – N.

<u>For decimal number</u>: r's complement in decimal number is 10's complement.

 Examples:

 The 10's complement of $(52520)_{10}$ is

   $= 10^5 - 52520$

   $= 100000 - 52520 = 47480$

 The 10's complement of $(0.3267)_{10}$ is

   $= 10^0 - 0.3267$

   $= 1 - 0.3267 = 0.6733$

For binary number: r's complement in binary number is 2's complement.

Example:

The 2's complement of $(101100.0110)_2$ is

$= 2^6 - (101100.0110)_2$

$= 64 - (101100.0110)_2$ 1000000.0000

$= 1000000 - 101100.0110$ 101100.0110

$= 10011.1010$

2. (r − 1)'s complement: For the given positive number N in base − r with an integer part of n decimal digits

And the fractional part of m digits. The (r − 1)'s complement of N is defined as $r^n - r^{-m} - N$.

For decimal number: (r − 1)'s complement in decimal number is 9's complement.

Example:

The 9's complement of $(52520.3267)_{10}$ is

$= 10^5 - 10^{-4} - 52520.3267$

$= 100000 - 1/10^4 - 52520.3267$

$= 100000 - 0.0001 - 52520.3267$

$= 99999.9999 - 52520.3267$

$= 47479.6732$

For binary number: (r – 1)'s complement in binary number is 1's complement.

Example:

The 1's complement of $(101100.0110)_2$ is

$= 2^6 - 2^{-4} - (101100.0110)_2$

$= 64 - \frac{1}{2}^4 - (101100.0110)_2$

$= 64 - 0.0625 - (101100.0110)_2$

$= 63.9375 - (101100.0110)_2$

$= (111111.1111)_2 - (101100.0110)_2$

$= 10011.1001$

Alternate method to find out the r's and (r-1)'s complement:

For decimal number:

9's complement is (r-1)'s complement in decimal.

Subtract each digit of decimal number from largest number of decimal that is 9 to get 9's complement.

Example: Find out 9's complement of $(4598.743)_{10}$

Here, total number of digits used in integer part is four and total number of digits used in fractional part is three, hence subtract 9999.999 – 4598.743 to get 9's complement.

$$9\ 9\ 9\ 9.9\ 9\ 9$$
$$\underline{4\ 5\ 9\ 8.7\ 4\ 3}$$
$$5\ 4\ 0\ 1.2\ 5\ 6$$

Alternate method to find out the r's and (r-1)'s complement:

10's complement is r's complement in decimal.

Firstly, find out 9's complement of given decimal number and add 1 to the LSB position of 9's complement number to get 10's complement.

Example: Find out 10's complement of $(57932.497)_{10}$

Here, 9's complement of 57932.497 = 99999.999 – 57932.497 = 42067.502

Now, adding 1 to the LSB position of 42067.502

$$
\begin{array}{r}
42067.502 \\
+\ 1 \\
\hline
42067.503
\end{array}
$$

Alternate method to find out the r's and (r-1)'s complement:

For binary numbers:

 1's complement is (r-1)'s complement in binary.

1's complement can be obtained by subtracting each digit of binary number from largest binary number i.e. 1 or can be obtained replacing 0 by 1 and 1 by 0.

Example: Find out 1's complement of $(11101001.10110)_2$

 Here, subtracting each digit of given binary number form largest binary number 1

$$1\ 1\ 1\ 1\ 1\ 1\ 1\ 1.1\ 1\ 1\ 1\ 1$$
$$\underline{-1\ 1\ 1\ 0\ 1\ 0\ 0\ 1.\ 1\ 0\ 1\ 10}$$
$$0\ 0\ 0\ 1\ 0\ 1\ 1\ 0.\ 0\ 1\ 0\ 0\ 1$$

or to get 1's complement, 1 is place by 0 and vice-versa.

Alternate method to find out the r's and (r-1)'s complement:

2's complement is r's complement in binary.

2's complement can be obtained by adding 1 to the LSB bit of 1's complement.

Example: Find 2's complement of $(110101.110)_2$

Here, 1's complement of $110101.110 = 001010.001$

Now adding 1 to the LSB bit of 1's complement to get 2's complement

$$
\begin{array}{r}
1010.001 \\
+\ 1 \\
\hline
1010.010
\end{array}
$$

2's complement of $110101.110 = 1010.010$

<u>Subtraction using r's complement</u>: The subtraction of two positive numbers(M – N)
both of base – r can be done as follows:

1. Add the minuend M to the r's complement of the subtrahend N.
2. Inspect the result obtained in step 1 for end carry.
   a) If an end carry occurs, discard it.
   b) If an end carry does not occur, take the r's complement of the number obtained
      in step 1 and place a negative sign in front.

For decimal numbers:
Example1: Using 10's complement, subtract (72532 – 3250).
Here,

$$M = 72532$$
$$N = \underline{03250}$$

10's complement of $N = (10^5 - 03250)$
$$= 100000 - 03250 = 96750$$

Now,

$$
\begin{array}{r}
72532 \\
+\ 96750 \\
\hline
169282
\end{array}
$$

End carry,
Discard it.

So, answer = 69282

Example2: Using 10's complement, subtract (3250 – 72532).

Here,

$$M = 03250$$
$$N = 72532$$

10's complement of N $= (10^5 - 72532)$
$$= (100000 - 72532)$$
$$= 27468$$

Now,

$$
\begin{array}{r}
03250 \\
+\ 27468 \\
\hline
30718
\end{array}
$$

No end carry

∴ – (10's complement of 30718) $= -(10^5 - 30718)$
$$= -(100000 - 30718)$$
$$= -69282$$

For binary numbers:

Example1: Subtract (1000100 – 1010100) using 2's complement.

Solution:

Here,

$$M = 1000100$$
$$N = 1010100$$

$$\text{2's complement of } N = (2^7)_{10} - (1010100)_2$$
$$= (128)_{10} - (1010100)_2$$
$$= 10000000 - 1010100$$
$$= 0101100$$

Now,

$$\begin{array}{r} 1000100 \\ + \, 0101100 \\ \hline 1110000 \end{array}$$

No end carry ⬋

$$\therefore - (\text{2's complement of } 1110000) = - \{(2^7)_{10} - (1110000)_2\}$$
$$= - \{(128)_{10} - (1110000)_2\}$$
$$= - (1000000 - 1110000)$$
$$= - 10000$$

Example2: Subtract (1110.111 – 1010.101) using 2's complement.

Solution:

Here,

$$M = 1110.111$$
$$N = 1010.101$$

2's complement of $N = (2^4)_{10} - (1010.101)_2$

$$= (16)_{10} - (1010.101)_2$$
$$= 10000 - 1010.101$$
$$= 0101.011$$

Now,

$$\begin{array}{r} 1110.111 \\ + 0101.011 \\ \hline 1\ 0100.010 \end{array}$$

End carry,
Discard it.

So, answer $= 100.010$

Subtraction using (r – 1) 's complement: The subtraction of two positive numbers (M – N)both of base – r can be done as follows:

1. Add the minuend M to the (r – 1)'s complement of the subtrahend N.
2. Inspect the result obtained in step 1 for end carry.
   a) If an end carry occurs, add 1 to the least significant bit i.e. end around carry.
   b) If an end carry does not occur, take the (r – 1)'s complement of the number obtained in step 1 and place a negative sign in front.

For decimal numbers:
Example1: Using 9's complement, subtract (453.35 – 321.17).

Here,

$$M = 453.35$$
$$N = 321.17$$

9's complement of $N = 10^3 - 10^{-2} - 321.17$
$$= 1000 - 0.01 - 321.17$$
$$= 999.99 - 321.17$$
$$= 678.82$$

Now,

```
       453.35
  +    678.82
     1 132.17
```

End carry occurred

$+ 1$

End Around carry

```
       132.18
```

.
.   . Answer = 132.18

Example2: Using 9's complement, subtract (3250 – 72532).

Here,

$$M = 03250$$
$$N = \ 72532$$

9's complement of N = 99999 – 72532
$$= 27467$$

Now,

$$
\begin{array}{r}
03250 \\
+ \ \ 27467 \\
\hline
30717
\end{array}
$$

No end carry

∴ – (9's complement of 30717) = – (99999 – 30717)
$$= -\ 69282$$

For binary numbers:

Example1: Subtract (1010100 – 1000100) using 1's complement.

Solution:

    Here,

$$M = 1010100$$
$$N = 1000100$$

1's complement of N = 0111011

    Now,

$$
\begin{array}{r}
1010100 \\
+\ 0111011 \\
\hline
1\ 0001111 \\
+\ 1 \\
\hline
0010000
\end{array}
$$

End carry occurred

End around carry

      ∴ Answer = 10000

Example2: Subtract (1000100 – 1010100) using 1's complement.

Solution:

Here,

$$M = 1000100$$
$$N = 1010100$$

1's complement of N = 0101011

Now,

$$
\begin{array}{r}
1000100 \\
+\ 0101011 \\
\hline
1101111
\end{array}
$$

No end carry ↙

∴ − (1's complement of 1101111) = − 10000

Binary Codes: Electronic digital system uses two distinct values and circuit elements that have two stable states 0 and 1 i.e. OFF and ON. In the coding, when numbers, letters or words are represented by specific group of symbols, it is said that the number, letter or word is being encoded. The group of symbols is called a code. The digital data is represented, stored and transmitted as group of binary bits. This group is also called binary code. The binary code is represented by the number as well as alphanumeric letter.

Types of binary codes:

- Weighted codes
- Non-Weighted codes
- BCD code(Binary Coded Decimal)
- Alphanumeric Codes
- Error detecting codes
- Error correcting codes

Weighted Codes: Weighted binary codes obey the positional weight principle. Each position of the number represent a specific weight. Several systems of the codes are used to express the decimal digits 0 through 9. In these code each decimal digit is represented by a group of four bits. Examples are 8421, hex, octal etc.

Types

1. Positive Weighted code: 8421, 2421 etc.

2. Negative weighted code: 84-2-1, 74-2-1 etc.

Positive Weighted code: Example: 8421

Decimal	8 4 2 1 (Binary)
0	0 0 0 0
1	0 0 0 1
7	0 1 1 1 (4 + 2 + 1 = 7)
9	1001 (8 +1 =9)

Negative Weighted code: Example: 8 4 -2 -1

Decimal	8 4 -2 -1 (Binary)
1	$0\ 1\ \ 1\ \ 1\ (4-2-1=4-3=1)$
7	$1\ 0\ 0\ \ 1\ (8-1=7)$
9	$1\ 1\ 1\ 1\ \ (8+4-2-1=9)$
10	$1\ 1\ 1\ 0$

Non-Weighted Codes: In this type of binary codes, the positional weights are not assigned. Examples are   Excess-3 code, gray code, ASCII Code etc.

BCD Code (Binary Coded Decimal Code): It is a four bits binary equivalent of given decimal number. BCD is a way to express each of the decimal digits with a four bits binary codes. It is also called 8421 code. Because the weights in the BCD codes are 8, 4, 2, 1. In BCD, with four bits we can represent sixteen numbers (0000 to 1111). i.e. $2^4 = 16$. But in BCD code only first 10 digits (0000 to 1001). The remaining six combinations (1010 to 1111) are invalid in BCD.

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

Ex. Decimal: 42

$$\begin{matrix} & 4 & 2 \\ \text{BCD Code:} & 0100 & 0010 = 01000010 \end{matrix}$$

Excess-3 Code: It is non-weighted code, which is used to express decimal numbers. The Excess-3 codes are derived from the 8421 BCD code. Excess-3 code is obtained by adding $(0011)_2$ Or $(3)_{10}$ to 8421 BCD code. It is used for encryption in LDST (Logic Design and Switching Theory).

Decimal Number $\longrightarrow$ 8421 BCD + 0011 = Excess-3

Decimal	BCD Code 8 4 2 1	Excess-3 Code BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Gray Code/Reflected Code: Digital systems can be designed to process data in discrete form only. Many physical systems supply continuous output data. These data must be converted into digital form before they are applied to a digital system. Analog information can be converted into digital form using analog to digital converter. It is sometimes convenient to use the reflected code to represent the digital data converted from the analog. The number in the reflected code changes by only one bit as it proceeds from one number. Application of reflected code occurs when the analog data are represented by a continuous change of a shaft position. The shaft is partitioned into segments and each segment is assigned a number. AS only one bit changes at a time, the gray code is called as a unit distance code. It is a non-weighted code and cannot be used for arithmetic operation.

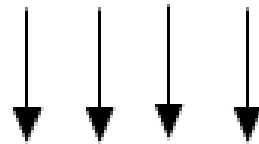| Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

Conversion from binary to gray code:
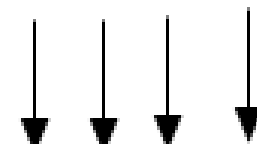
Binary X_3 X_2 X_1 X_0

X_3 $(X_3 \oplus X_2)$ $(X_2 \oplus X_1)$ $(X_1 \oplus X_0)$

E.g. Binary 0 0 0 0

Gray code 0 0 0 0

E.g. Binary 0 0 1 0

Gray code 0 0 1 1

ASCII (American Standard Code for Information Interchange Code): It is a seven bit alphanumeric code. It is a code for representing 128 English characters as numbers, with each letter assigned a number from 0 to 127. Most of the computers use ASCII codes to represent text, which makes it possible to transfer data from one computer to another.

Character	7-bit ASCII code	Decimal Code
A	100 0001	65
B	100 0010	66
C	100 0011	67
D	100 0100	68
E	100 0101	69
F	100 0110	70
G	100 0111	71
H	100 1000	72
I	100 1001	73
J	100 1010	74
K	100 1011	75
L	100 1100	76
M	100 1101	77
N	100 1110	78
O	100 1111	79

P	101 0000	80
Q	101 0001	81
R	101 0010	82
S	101 0011	83
T	101 0100	84
U	101 0101	85
V	101 0110	86
W	101 0111	87
X	101 1000	88
Y	101 1001	89
Z	101 1010	90

0	011 0000	48
1	011 0001	49
2	011 0010	50
3	011 0011	51
4	011 0100	52
5	011 0101	53
6	011 0110	54
7	011 0111	55
8	011 1000	56
9	011 1001	57

Error detecting and correcting Codes: Error detection code can be used to detect errors during transmissions. The detected error cannot be corrected but its presence is indicated. During data transmission, sometime 001 can be changed into 011. So, there is error in receiving end. The simple method to check the errors during transmission is to add an extra bit to the message is called parity bit.

Parity bit: It is an extra bit included with a binary message to make the number of 1's either odd or even. The message including the parity bit is transmitted and then checked at the receiving end for errors. An error is detected if the checked parity does not correspond to the one transmitted.

Parity bit generation:

Message	P(odd)	Message	P (even)
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1
0011	1	0011	0
0100	0	0100	1
0101	1	0101	0
0110	1	0110	0
0111	0	0111	1
1000	0	1000	1
1001	1	1001	0
1010	1	1010	0
1011	0	1011	1
1100	1	1100	0
1101	0	1101	1
1110	0	1110	1
1111	1	1111	0

Parity generator: The circuit that generates the parity bit in the transmitter is called parity generator. Consider a three bit message to be transmitted with an odd parity bit.
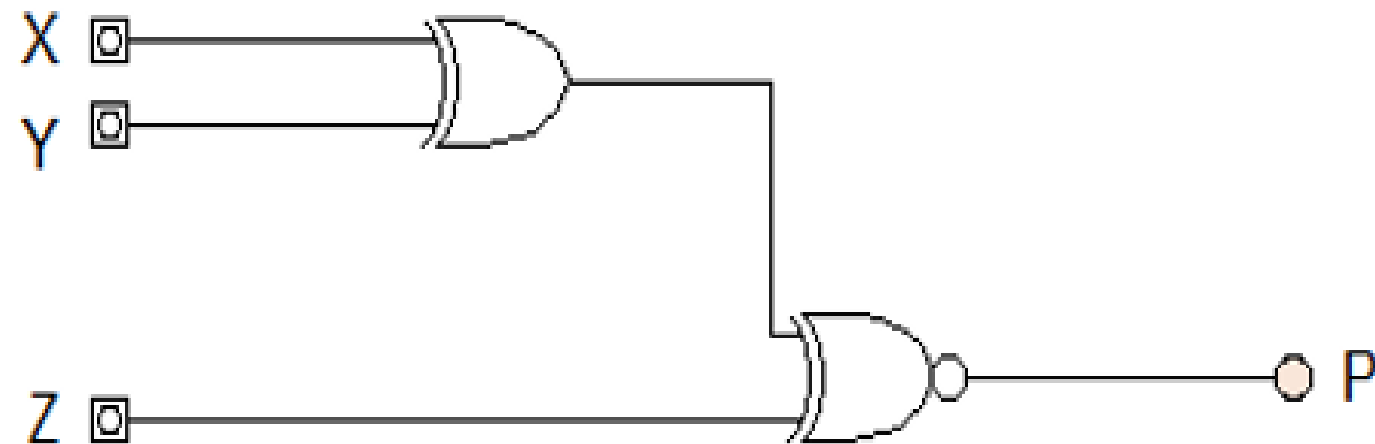
Odd Parity generator:

Truth table of odd parity generation:

Three bit message			Parity bit generated
X	Y	Z	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Logic function of odd parity can be expressed as:

$$P = \overline{X \oplus Y \oplus Z}$$

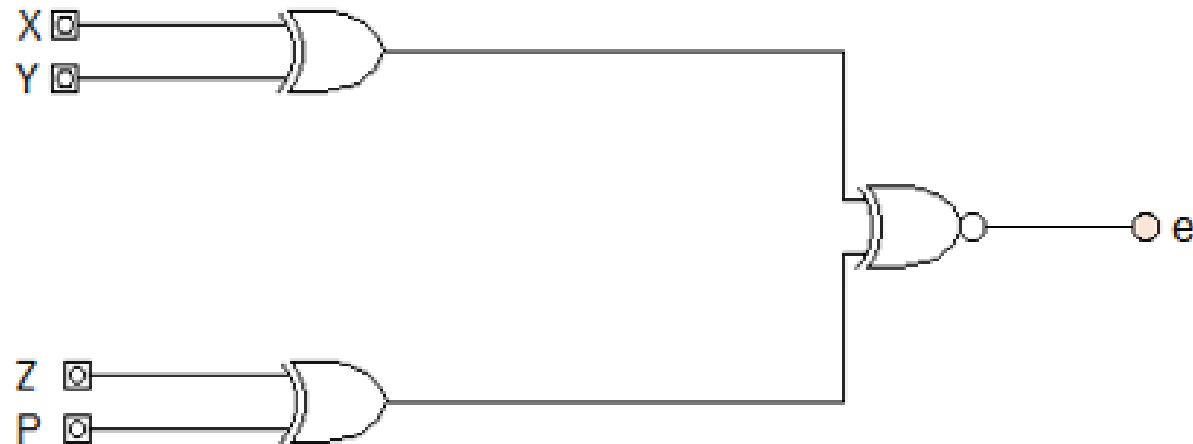**Logic diagram of odd parity generation:**

Odd Parity checker:

Truth table of odd parity checker:

Received bit				Parity error check
X	Y	Z	P	C
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Logic function:

$$c = \overline{X \oplus Y \oplus Z \oplus P}$$

Logic diagram of parity checking:

Note:

If c = 0, then there is no error in the received signal. The received data becomes odd parity.

If c = 1, then there is error in the received signal. That is data becomes even parity.
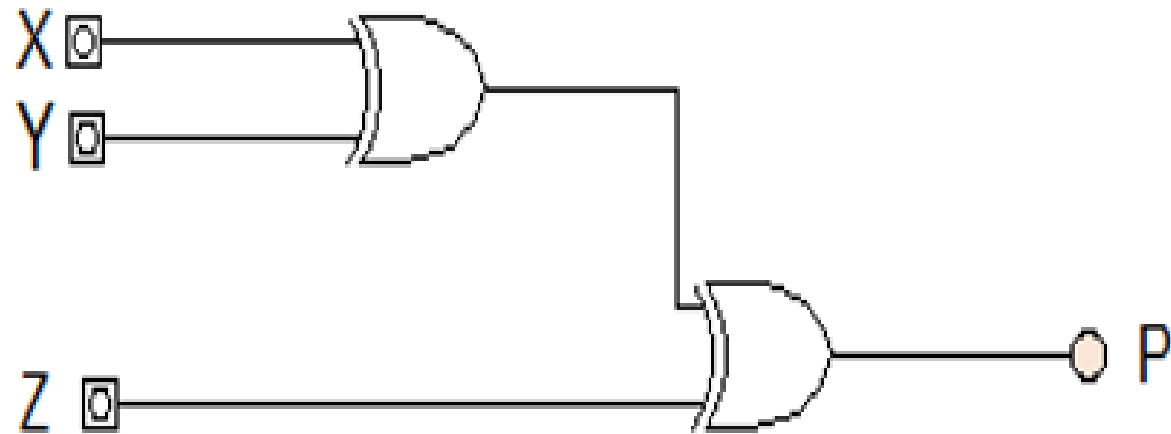
Even Parity generator:

Truth table of even parity generation:

Three bit message			Parity bit generated
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Logic function can be expressed as:

$$P = X \oplus Y \oplus Z$$

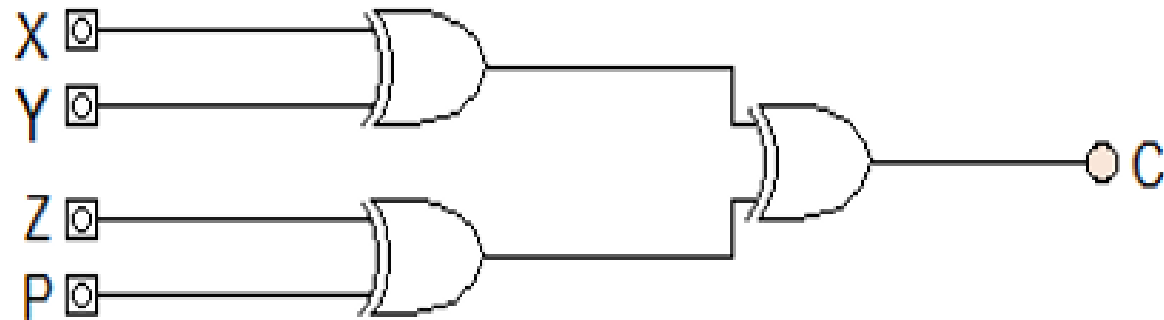**Logic diagram of even parity generation:**

Even Parity checker:

Truth table of odd parity checker:

Received bits				Parity error check
X	Y	Z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Logic function:

$$c = X \oplus Y \oplus Z \oplus P$$

Logic diagram of even parity checking:

Note:

If c = 0, then there is no error in the received signal. The received data becomes odd parity.
If c = 1, then there is error in the received signal. That is data becomes even parity.

Representation of Unsigned and Signed numbers:

Unsigned Numbers

Unsigned numbers contain only magnitude of the number. They don't have any sign. That means all unsigned binary numbers are positive.

Signed Numbers

Signed numbers contain both sign and magnitude of the number. Generally, the sign is placed in front of number. So, we have to consider the positive sign for positive numbers and negative sign for negative numbers. Therefore, all numbers can be treated as signed numbers if the corresponding sign is assigned in front of the number.

If sign bit is zero, which indicates the binary number is positive. Similarly, if sign bit is one, which indicates the binary number is negative.

Representation of Un-Signed Binary Numbers

The bits present in the un-signed binary number holds the **magnitude** of a number. That means, if the un-signed binary number contains **'N'** bits, then all **N** bits represent the magnitude of the number, since it doesn't have any sign bit.

Example

Consider the **decimal number 108**. The binary equivalent of this number is **1101100**. This is the representation of unsigned binary number.

$$108_{10} = 1101100_2$$

It is having 7 bits. These 7 bits represent the magnitude of the number 108.

Representation of Signed Binary Numbers

The Most Significant Bit MSB of signed binary numbers is used to indicate the sign of the numbers. Hence, it is also called as **sign bit**. The positive sign is represented by placing '0' in the sign bit. Similarly, the negative sign is represented by placing '1' in the sign bit.

If the signed binary number contains 'N' bits, then N−1 bits only represent the magnitude of the number since one bit MSB is reserved for representing sign of the number.

There are three **types of representations** for signed binary numbers

- Sign-Magnitude form
- 1's complement form
- 2's complement form

Representation of a positive number in all these 3 forms is same. But, only the representation of negative number will differ in each form.

Example

Consider the **positive decimal number +108**. The binary equivalent of magnitude of this number is 1101100. These 7 bits represent the magnitude of the number 108. Since it is positive number, consider the sign bit as zero, which is placed on left most side of magnitude.

$$+108_{10} = 01101100_2$$

Therefore, the **signed binary representation** of positive decimal number +108 is **01101100**. So, the same representation is valid in sign-magnitude form, 1's complement form and 2's complement form for positive decimal number +108.

Sign-Magnitude form:

In sign-magnitude form, the MSB is used for representing **sign** of the number and the remaining bits represent the **magnitude** of the number. So, just include sign bit at the left most side of unsigned binary number. This representation is similar to the signed decimal numbers representation.

Example

Consider the **negative decimal number** -**108**. The magnitude of this number is 108. We know the unsigned binary representation of 108 is 1101100. It is having 7 bits. All these bits represent the magnitude.

Since the given number is negative, consider the sign bit as one, which is placed on left most side of magnitude.

$$-108_{10} = 11101100_2$$

Therefore, the sign-magnitude representation of -108 is **11101100**.

1's complement form:

The 1's complement of a number is obtained by **complementing all the bits** of signed binary number. So, 1's complement of positive number gives a negative number. Similarly, 1's complement of negative number gives a positive number.

That means, if you perform two times 1's complement of a binary number including sign bit, then you will get the original signed binary number.

Example

Consider the **negative decimal number -108**. The magnitude of this number is 108. We know the signed binary representation of 108 is 01101100.

It is having 8 bits. The MSB of this number is zero, which indicates positive number. Complement of zero is one and vice-versa. So, replace zeros by ones and ones by zeros in order to get the negative number.

$$-108_{10} = 10010011_2$$

Therefore, the **1's complement of** 108_{10} is 10010011_2.

2's complement form:

The 2's complement of a binary number is obtained by **adding one to the 1's complement** of signed binary number. So, 2's complement of positive number gives a negative number. Similarly, 2's complement of negative number gives a positive number.

That means, if you perform two times 2's complement of a binary number including sign bit, then you will get the original signed binary number.

Example

Consider the **negative decimal number -108**.

We know the 1's complement of $(108)_{10}$ is $(10010011)_2$

2's compliment of 108_{10} = 1's compliment of $108_{10} + 1$.

= 10010011 + 1

= 10010100

Therefore, the **2's complement of** 108_{10} is 10010100_2.

Activate

iii) $A = 14$ and $B = -25$

iv) $A = 8$ and $B = +12$

Solution:

i) $A = +15$ and $B = -25$

* $A - B = (+15) - (-25)$
$= (+15) + (+25)$

$+15 = 0 0 0 0 1 1 1 1$
$+25 = 0 0 0 1 1 0 0 1$
$+40$ _____
$0 0 1 0 1 0 0 0$

eighth bit $\leftarrow$
is 0
so it is true. $+40$

* $-A + B = -(+15) + (-25)$
$= (-15) + (-25)$
and -25

To represent -15 and -25 we have to take 2's complement of $+15$ and $+25$.

$-15 = 1 1 1 1 0 0 0 0$
$-25 = 1 1 1 0 0 1 1 1$ _____
$-40 \leftarrow 0 1 1 0 1 1 0 0 0$
discard

Note:
Numbers should be represented in 8 bits.
Min 8-bit registers in computer.

2's complement of $+25$.
Firstly 1's comp.
$1 1 1 0 0 1 1 0$
$+ 1$ _____
$1 1 1 0 0 1 1 1$

2's complement of $+15$
copy 1's complement
$1 1 1 1 0 0 0 0$
$+ 1$ _____
$1 1 1 1 0 0 0 1$

Rslt = − (2's complement of 11011000)

= 00100111
   + 1
———————————
  00101000
= − (00101000)
= − 40 in decimal.

⇒ 32 16 8 4 2 1
   32 + 8 = 40

* A + B = (+15) + (−25)

+ 15   00001111
− 25   11100111
————————————————
−10   11110110

Rslt = − (2's complement of 11110110)
= − (00001010)
= − 10 in decimal.

1's complement
11110110
= 00001001
   + 1
————————
  00001001
   + 1
————————
  00001010

*   B - A

$= -25 - (+15)$

$\therefore$ -25   1 1 1 0 0 1 1 1

$\therefore$ - 15   1 1 1 1 0 0 0 1

$\overline{\quad -40 \quad}$ $\overline{\quad 1\, 1\, 1\, 0\, 1\, 1\, 0\, 0\, 0 \quad}$

discard $\leftarrow$ 1 $\underbrace{1\,1\,0\,1\,1\,0\,0\,0}$

Result — (2's complement of 1 1 0 1 1 0 0 0)

$= -(0\,0\,1\,0\,1\,0\,0\,0)$

$= -40$ in decimal.

similarly   ii, iii, iv   can be done

Q.2. perform the arithmetic operation $(+42)$
+ (-13) and $(-42) - (-1)$ in binary using
signed 2's complement representation for negative
number.

solution: $(+42) + (-13)$ using 2's complement

$+42 = 0\,0\,1\,0\,1\,0\,1\,0$ 1 1 1 1 0 0 1 0

$+13 = 0\,0\,0\,0\,1\,1\,0\,1$ + 1

$\downarrow$ 2's complement $\overline{1\,1\,1\,1\,0\,0\,1\,1}$

$-13 = 1\,1\,1\,1\,0\,0\,1\,1$

$$+42 = 00101010$$
$$-13 = 11110011$$
$$\overline{+29} = 1\,0001\,1\,1\,01$$

Discard & ⌣
end carry

∴ Result = 00011101

= +29 in decimal.

* $(-42) - (-1)$ using 2's complement

$= (-42) + (+1)$

$+ 42 = 00101010$
  ↓ 2's complement

$-42 = 11010110$
$+ 1 = 00000001$
$\overline{}$
$-41 \quad 11010111$

	1's complement
	11010101
	+1
	$\overline{11010110}$

Result = −(2's complement of 11010111)

= − 41 in decimal.