UNIT-1: NUMBER SYSTEMS, BOOLEAN ALGEBRA AND LOGIC GATES

STRUCTURE

1.0 **OBJECTIVES**

After going through this unit, you will be able to

- understand the decimal, binary, octal and hexadecimal number systems
- convert from one number system into another
- apply arithmetic operations to binary numbers
- understand BCD codes and alpha numeric codes
- learn the operations of logic gates
- apply the basic laws of Boolean algebra
- apply De Morgan's theorems to Boolean expressions

.

1.1 INTRODUCTION

The binary number system and digital codes are fundamental to computers. In this chapter, the binary number system and its relationship to other systems such as decimal, hexadecimal, and octal are introduced. Arithmetic operations with binary numbers are discussed to provide a basis for understanding how computers and many other types of digital systems work. Also binary coded decimal (BCD), and alpha numeric codes are introduced. Binary logic gates are explained with the help of logic diagram, block diagram and truth table. Basic laws of Boolean Algebra are given. De-Morgan's theorems are also stated and proved.

1.2 NUMBER SYSTEMS

A number system relates quantities and symbols. The base or radix of a number system represents the number of digits or basic symbols in that particular number system. In decimal system the base is 10, because of use the numbers 0, 1, 2,3,4,5,6,7,8 and 9.

1.2.1 Binary Number System

A binary number system is a code that uses only two basic symbols. The digits can be any two distinct characters, but it should be 0 or 1. The binary equivalent for some decimal numbers are given below

Decimal	0	1	2	3	4	5	6	7	8	9	10	11
Binary	0	1	10	11	100	101	110	111	1000	1001	1010	1011

Each digit in a binary number has a value or weight. The LSB has a value of 1. The second from the right has a value of 2, the next 4 , etc.,

16	8	4	2	1
2^4	2^3	2^2	2^1	2^0

Binary to decimal conversion:

$(1001)_2 = X_{10}$

$1001 = 1x2^3 + 0x2^2 + 0x2^1 + 1x2^0$
$= 8+0+0+1$
$(1001)_2 = (9)_{10}$

Fractions:
For fractions the weights of the digit positions are written from right of the binary point and weights are given as follows.

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}

E.g.:
$(0.0110)_2 = X_{10}$
$= 0x2^{-1} + 1x2^{-2} + 1x2^{-3} + 0x2^{-4}$
$= 0\ x0.5 + 1x0.25 + 1x0.125 + 0x0.0625$
$= (0.375)_{10}$

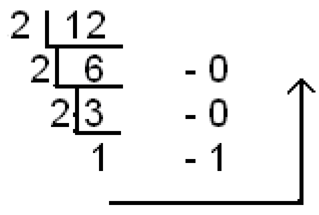
E.g.:
$(1011.101)_2 = X_{10}$
$= 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3}$
$= 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125$
$= (11.625)_{10}$

Decimal to binary conversion:
(Double Dabble method)

In this method the decimal number is divided by 2 progressively and the remainder is written after each division. Then the remainders are taken in the reverse order to form the binary number.

E.g.:

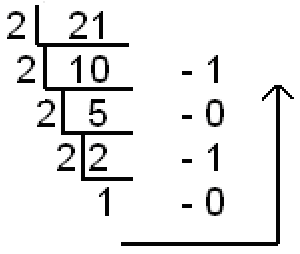
$(12)_{10} = X_2$



$(12)_{10} = (1100)_2$

E.g.:

$(21)_2 = X_2$



$(21)_2 = (10101)_2$

Fractions:

The fraction is multiplied by 2 and the carry in the integer position is written after each multiplication. Then they are written in the forward order to get the corresponding binary equivalent.

E.g.:

$(0.4375)_{10} = X_2$

$2 \times 0.4375 = 0.8750 \Rightarrow 0$
$2 \times 0.8750 = 1.750 \Rightarrow 1$
$2 \times 0.750 = 1.5 \Rightarrow 1$
$2 \times 0.5 = 1.0 \Rightarrow 1$

$(0.4375)_{10} = (0.0111)_2$

1.2.2 Octal Number System

Octal number system has a base of 8 i.e., it has eight basic symbols. First eight decimal digits 0, 1,2,3,4,5,6,7 are used in this system.

Octal to decimal conversion:

In the octal number system each digit corresponds to the powers of 8. The weight of digital position in octal number is as follows

8^4	8^3	8^2	8^1	8^0	8^{-1}	8^{-2}	8^{-3}

To convert from octal to decimal multiply each octal digit by its weight and add the resulting products.

E.g.:

$(48)_8 = X_{10}$

$48 = 4 \text{ x } 8^1 + 7 \text{ x } 8^0$
$= 32 + 7$
$= 39$

$(48)_8 = (39)_{10}$

E.g.:

$(22.34)_8 = X_{10}$

$22.34 = 2 \text{ x } 8^1 + 2 \text{ x } 8^0 + 3 \text{ x } 8^{-1} + 4 \text{ x } 8^{-2}$
$=16 + 2 + 3 \text{ x } 1/8 + 4 \text{ x } 1/64$
$= (18.4375)$

$(22.34)_8 = (18.4375)_{10}$

Decimal to octal conversion:

Here the number is divided by 8 progressively and each time the remainder is written and finally the remainders are written in the reverse order to form the octal number. If the number has a fraction part, that part is multiplied by 8 and carry in the integer part is taken. Finally the carries are taken in the forward order.

E.g.:

$(19.11)_{10} = X_8$

$$\begin{array}{r|l} 8 & \underline{19} \\ & 2 - 3 \end{array}$$

0.11 x 8 = 0.88 => 0
0.88 x 8 = 7.04 => 7
0.04 x 8 = 0.32 => 0
0.32 x 8 = 2.56 => 2
0.56 x 8 = 4.48 => 4

$(19.11)_{10} = (23.07024)_8$

Octal to binary conversion:

Since the base of octal number is 8, i.e., the third power of 2, each octal number is converted into its equivalent binary digit of length three.

E.g.:

$(57.127)_8 = X_2$

```
   5      7  .  1      2      7
  101    111 . 001   010    111
```

$(57.127)_8 = (101111001010111)_2$

Binary to octal:

The given binary number is grouped into a group of 3 bits, starting at the octal point and each group is converted into its octal equivalent.

E.g.:

$(1101101.11101)_2 = X_8$

```
001   101   101.111  010
 1      5     5 . 7    2
```

$(1101101.11101)_2 = (155.72)_8$

1.2.3 Hexadecimal Number System:

The hexadecimal number system has a base of 16. It has 16 symbols from 0 through 9 and A through F.

Decimal	Hexadecimal	Binary
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

Binary to hexadecimal:

The binary number is grouped into bits of 4 from the binary point then the corresponding hexadecimal equivalent is written.

E.g.:

$(100101110\ .\ 11011)_2 = X_{16}$

```
0001 0010 1110 . 1101 1000
 1    2    E   .  D    8
```

$(100101110\ .\ 11011)_2 = (12E\ .\ D8)_{16}$

Hexadecimal to binary:

Since the base of hexadecimal number is 16, i.e., the fourth power of 2, each hexadecimal number is converted into its equivalent binary digit of length four.

E.g.:

$(5D.\ 2A)_{16} = X_2$

```
  5     D  .  2     A
0101  1101 . 0010  1010
```

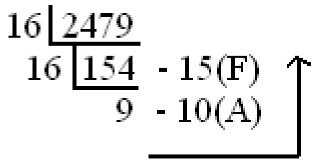
$(5D.\ 2A)_{16} = (01011101.00101010)_2$

Decimal to hexadecimal:

The decimal number is divided by 16 and carries are taken after each division and then written in the reverse order. The fractional part is multiplied by 16 and carry is taken in the forward order.

E.g.:

$(2479.859))_{10} = X_{16}$



$16 \times 0.859 = 13.744 \quad => 13\ (D)$

```
16 x 0.744 = 11.904   => 11 (B)
16 x 0.904 = 14.464   => 14 (E)
16 x 0.464 =  7.424   =>  7
16 x 0.424 =  6.784   =>  6
```

$(2479.859)_{10} = (9AF.DBE76)_{16}$

Hexadecimal to decimal:

Each digit of the hexadecimal number is multiplied by its weight and then added.

E.g.:

$(81.21)_{16} = X_{10}$

$= 8 \times 16^1 + 1 \times 16^0 + 2 \times 16^{-1} + 1 \times 16^{-2}$

$= 8 \times 16 + 1 \times 1 + 2/16 + 1/16^2$

$= (129.1289)_{10}$

$(81.21)_{16} = (129.1289)_{10}$

1.2.4 Binary Arithmetic

Binary Addition:

To perform the binary addition we have to follow the binary table given below.

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 => plus a carry-over of 1

Carry-overs are performed in the same manner as in decimal arithmetic. Since 1 is the largest digit in the binary system, any sum greater than 1 requires that a digit be considered over.

E.g.:

```
  111           1010            11.01
 _110          _1101           101.11
 1001          10111          1001.00
```

Binary Subtraction:

To perform the binary subtraction the following binary subtraction table should be followed.

0 – 0 = 0

1 – 0 = 1
1 – 1 = 0
0 – 1 = 1   with a borrow of 1 is equivalent to 10 – 1 = 1

E.g.:

$$\begin{array}{l} 111 \\ \underline{010\ \ } \\ \underline{101} \end{array}$$

E.g.:

$$\begin{array}{l} 110.01 \\ \underline{100.10} \\ \underline{001.11} \end{array}$$

1's complement:

To obtain 1's complement of a binary number each bit of the number is subtracted from 1.

E.g.:

Binary number	1's Complement
0101	1010
1001	0110
1101	0010
0001	1110

Thus 1's complement of a binary number is the number that results when we change each 0 to a 1 and each 1 to a 0.

1's complement subtraction:

Instead of subtracting the second number from the first, the 1's complement of the second number is added to the first number. The last carry which is said to be a END AROUND CARRY, is added to get the final result.

E.g.:

$$\begin{array}{llr} 7\ - & 111\ \text{-----------------}> & 111\ + \\ \underline{\ 3} & 011\ \text{1's complement} & \underline{100} \\ \underline{\ 4} & & 1011\ + \\ & & \llcorner\underline{\to 1} \\ & & \underline{100\ \ } \rightarrow \text{result} \end{array}$$

If there is no carry in the 1’s complement subtraction, it indicates that the result is a negative and number will be in its 1’s complement form. So complement it to get the final result.

E.g.:

```
   8 -      1000   ----------------> 1000   +
  10        1010  1’s complement  0101
   4                              1101   1’s complement    - 0010 → result
```

The following points should be noted down when we do 1’s complement subtraction.

1. Write the first number (minuend) as such.
2. Write the 1’s complement of second number(subtrahend)
3. Add the two numbers.
4. The carry that arises from the addition is said to be “end around carry”.
5. End-around carry should be added with the sum to get the result.
6. If there is no end around carry find out the 1’s complement of the sum and put a negative sign before the result as the result is negative.

2’s Complement:

2’s complement results when we add ‘1’ to 1’s complement of the given number i.e., 2’s complement =1’s complement + 1

Binary Number	1’s complement	2’s complement
1010	0101	0110
0101	1010	1011
1001	0110	0111
0001	1110	1111

2’s Complement Subtraction:

Steps:

1. Write the first number as such
2. Write down the 2’s complement of the second number.
3. Add the two numbers.
4. If there is a carry, discard it and the remaining part (sum) will be the result (positive).
5. If there is no carry, find out the 2’s complement of the sum and put negative sign before the result as the result is negative.

E.g.:

```
1)  10 -     1010   -----------------> 1010  +
     8       1000  2's complement 1000
     2                           10010

                                     0010 → result

2)   5 -     0101   -----------------> 0101  +
    12       1100  2's complement 0100
     4                            1001   2's complement  – 0111→ result
```

Binary multiplication:

The table for binary multiplication is given below

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

E.g.:

```
     1011 x 110

          1011 x
           110
          0000
         1011
        1011__
       1000010_
```

E.g.:

```
    101.01 x 11.01

          101.01 x
             11.01
          101 01
         00000
        10101
       _10101__
      10001.0001
```

Binary division:

The table for binary division is as follows.

$0 \div 1 = 0$
$1 \div 1 = 1$

As in the decimal system division by zero is meaning less.

E.g.:

1) $1100 \div 11$

```
      100
 11 | 1100
    | 11
      0
```

2) $1001 \div 10$

```
       100.1
 10 | 1001
    | 10
      0010
        10
         0
```

1.2.5 BCD Addition

Binary Coded Decimal(BCD) is a way to express each of the decimal digits with a binary code. There are only ten code groups in the BCD system. The 8421 code is a type of BCD code. In BCD each decimal digit , 0 through 9 is represented by a binary code of four bits. The designation of 8421 indicates the binary weights of the four bits (2^3,2^2,2^1,2^0). The largest 4-bit code is 1001. The numbers 1010, 1011, 1100, 1101, 1110, and 1111 are called forbidden numbers. The following table represents the decimal and 8421 equivalent numbers.

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

8421 Addition:

In 8421 addition, if there is a carry or if it results in a forbidden group, then 0110(6) should be added in order to bring the result to the 8421 mode again.

E.g.:

$$\begin{array}{rlr}
8 & + & 1000\ + \\
\underline{\ 7} & & \underline{0111} \\
\underline{15} & & 1111 \\
 & & \underline{+\ 0110} \\
 & & \underline{0001\ 0101}
\end{array}$$

E.g.:

$$\begin{array}{rlrl}
18 & + & & 0001\ 1000\ + \\
\underline{\ 2} & & & \underline{0000\ 0010} \\
\underline{20} & & & 0001\ 1010 \\
 & & + & \underline{0000\ 0110} \\
 & & & \underline{0010\ 0000}
\end{array}$$

1.2.6 Alphanumeric code

Computers, printers and the other devices must process both alphabetic and numeric information. Serial coding systems have been developed to represent alphanumeric information as a series of 1's and 0's. The characters to be coded are alphabets(26), numerals (10) and special characters such as +,-, /,*, $ etc,

In order to code a character, string of binary digits is used. In order to ensure uniformity in coding, two standard codes have been used.

1. ASCII: American Standard Code for Information Interchange.
2. EBCDIC: Extended Binary Coded Decimal Interchange Code. It is an 8 bit code.

ASCII is 7-bit code of the form X_6, X_5, X_4, X_3, X_2, X_1, X_0 and is used to code two types of information. One type is the printable character such as alphabets, digits and special characters. The other type is known as control characters which represent the coded information to control the operation of the digital computer and are not printed.

CHECK YOUR PROGRESS 1

1. $2 \times 10^1 + 8 \times 10^0$ is equal to
(a) 10 (b) 280 (c) 2.8 (d) 28

2. The binary number 1101 is equal to the decimal number
(a) 13 (b) 49 (c) 11 (d) 3

3. The decimal 17 is equal to the binary number
(a) 10010 (b) 11000 (c) 10001 (d) 01001

4. The sum of 11010 + 01111 equals
(a) 101001 (b) 101010 (c) 110101 (d) 101000

5. The difference of 110 – 010 equals
(a) 001 (b) 010 (c) 101 (d) 100

6. The 1's complement of 10111001 is
(a) 01000111 (b) 01000110 (c) 11000110 (d) 10101010

7. The 2's complement of 11001000 is
(a) 00110111 (b) 00110001 (c) 01001000 (d) 00111000

8. The binary number 10110011100101010000l can be written in octal as
(a) 5471230_8 (b) 5471241_8 (c) 2634521_8 (d) 23162501_8

9. The binary number 10001101010001101111 can be written in hexadecimal as
(a) $AD467_{16}$ (b) $8C46F_{16}$ (c) $8D46F_{16}$ (d) $AE46F_{16}$

10. The BCD number for decimal 473 is
(a) 111011010 (b) 1110111110101001 (c) 010001110011 (d) 010011110011

1.3 BOOLEAN ALGEBRA AND LOGIC GATES

1.3.1 AND gate

A gate is simply an electronic circuit which operates a one or more signals to produce an output signal. The output is high only for certain combination of input signals.

An AND gate (Figure 1.1) has a high output only when all inputs are high. The output is low when any one input is low.

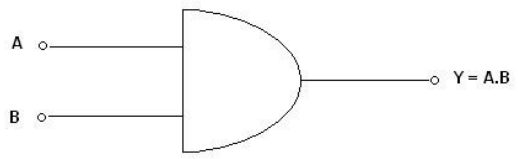


Figure 1.1 AND gate

Boolean expression for AND gate operation is

$$Y=A\,.\,B$$

Truth table

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

1	1	1

1.3.2 OR gate

An OR gate (Figure 1.2) produces a high output when any or the entire inputs are high. The output is low only when all the inputs are low.

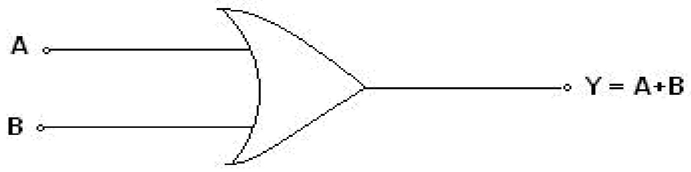


Figure 1.2 OR gate

The Boolean expression for an OR gate is

$$Y=A+B$$

Truth table:

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

1.3.3 NOT gate:

A NOT gate (Figure 1.3) is also called an inverter. The circuit has one input and one output. The output is the complement of the input. If the input signal is high, the output is low and vice versa.

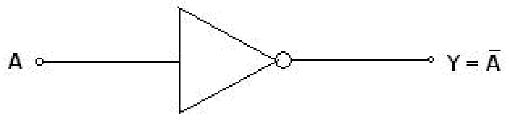


Figure 1.3 NOT gate

The Boolean expression for NOT gate is

$$Y = \bar{A}$$

Truth table:

A	Y = Ā
0	1
1	0

If two NOT gates are cascaded then the output will be same as the input and the circuit is called buffer circuit.

1.3.4 NAND gate

A NAND (Figure 1.4) gate has two or more input signals but only one output signal. All input signals must be high to get a low output. When one AND gate is combined with a NOT gate, a NAND gate is obtained.

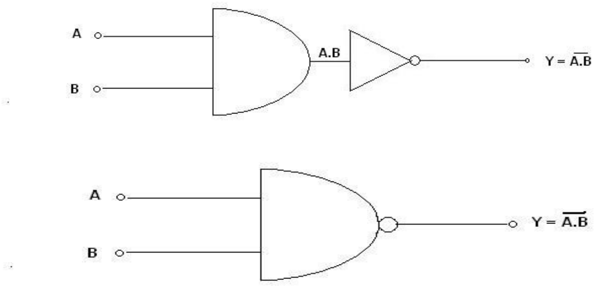


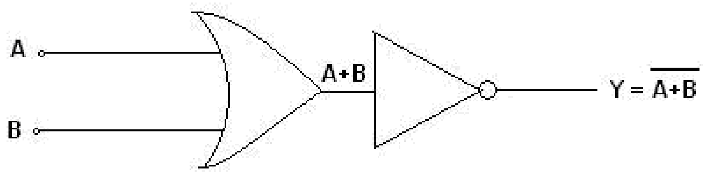
Figure 1.4 NAND gate

Truth table:

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

1.3.5 NOR gate:

NOR gate (Fig. 1.5) has two or more input signals and one output signal. It consists of one OR gate followed by an inverter. A NOR gate produces a high output only when all the inputs are low.



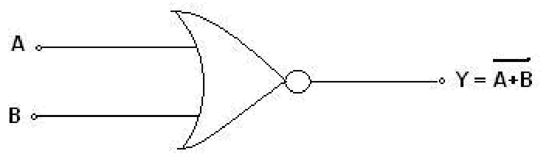


Figure 1.5 NOR gate

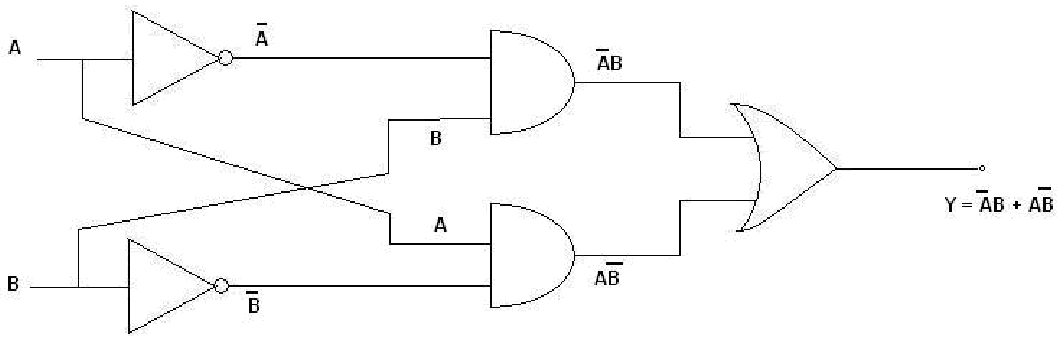
Truth table:

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

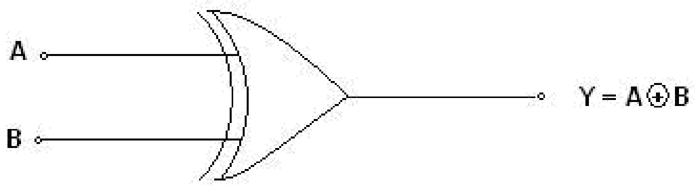
1.3.6 XOR gate

XOR (Figure 1.6) gate is an abbreviation of exclusive OR gate. It has two inputs and one output. For a two input XOR gate, the output is high when the inputs are different and the output is low when the inputs are same. In general, the output of an XOR gate is high when the number of its high inputs is odd. The Boolean expression of the XOR gate is

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



a) Logic diagram



b) Logic symbol:

Figure 1.6 XOR gate

Truth table:

A	B	Y = A⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Uses of XOR gate:

1. Binary to Gray Converter

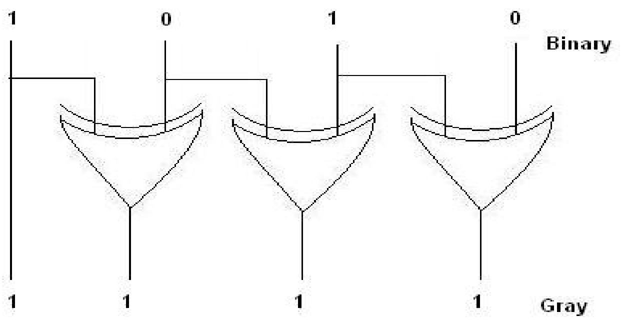


Figure 1.7 Binary to Gray Converter

The Figure 1.7 shows the way to convert binary number to gray number using XOR gates. Since mod-2 addition is involved in the conversion, XOR gate is used for this purpose.

2. Gray to Binary Converter:

XOR gate is also used to convert gray code to a binary number. The circuit diagram for this operation is shown in the Figure 1.8.

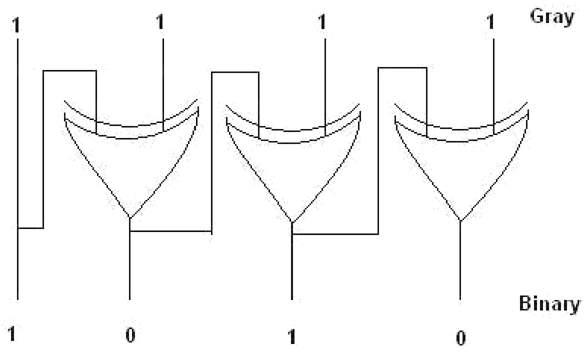


Figure 1.8 Gray to Binary Converter

3. Parity checker:

Parity checker can be designed using XOR gates as given in the Figure 1.9. Here the parity of the word ABCD is checked. The circuit adds the bits of ABCD. A final sum of 0 implies even parity and a sum of 1 means odd parity.

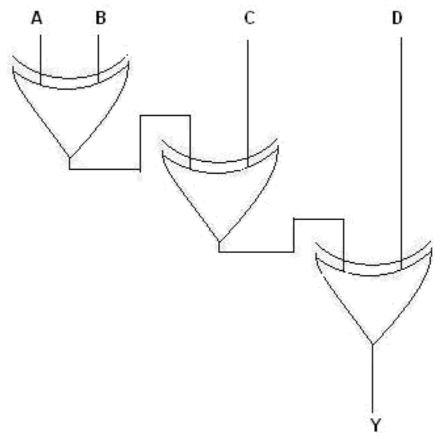


Figure 1.9 Parity checker

1.3.7 Basic Laws of Boolean Algebra

Commutative law:

A + B = B + A
B + A = A + B

Associative law:

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

Distributive law

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

Other laws of Boolean algebra:

1. A + 0 = A

2. A + 1 = 1

3. A + A = A

4. A + $\bar{A}$ = 1

5. A .0 = 0

6. $A \,.1 = A$

7. $A \,.A = A$

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

9. $\overline{\overline{A}} = A$

10. $A + A.B = A$

11. $A.(A + B) = A$

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

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

14. $A.(\bar{A} + B) = A.B$

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

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

1.3.8 De Morgan's Theorems:

I Theorem statement:

The complement of a sum is equal to the product of the complements.

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

II Theorem Statement:

The complement of a product is equal to the sum of the complements.

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

Proof of first theorem:

To prove $\overline{A + B} = \overline{A} \,.\, \overline{B}$

Case 1:_ A=0, B=0

L.H.S => $\overline{A + B} = \overline{0 + 0} = \overline{0} = 1$

R.H.S => $\overline{A} \,.\, \overline{B} = \overline{0} \,.\, \overline{0} = 1 \,.1 = 1$

Case 2: A=0, B=1

L.H.S => $\overline{A + B} = \overline{0 + 1} = \overline{1} = 0$

R.H.S => $\overline{A} . \overline{B} = \overline{0} . \overline{1} = 1 . 0 = 0$

Case 3: A=1, B=0

L.H.S => $\overline{A + B} = \overline{1 + 0} = \overline{1} = 0$

R.H.S => $\overline{A} . \overline{B} = \overline{1} . \overline{0} = 0 . 1 = 0$

Case 4: A=1, B=1

L.H.S => $\overline{A + B} = \overline{1 + 1} = \overline{1} = 0$

R.H.S => $\overline{A} . \overline{B} = \overline{1} . \overline{1} = 0 . 0 = 0$

Truth table

A	B	$\overline{\overline{A} + B}$	$\overline{A} . \overline{B}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Proof of second theorem:

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

Case 1: A=0, B=0

L.H.S => $\overline{A . B} = \overline{0 . 0} = \overline{0} = 1$

R.H.S => $A + B = \overline{0} + \overline{0} = \overline{1} + \overline{1} = 1$

Case 2: A=0, B=1

L.H.S => $\overline{A . B} = \overline{0 . 1} = \overline{0} = 1$

R.H.S => A + B = 0 + 1 = 1 + 0 = 1

Case 3: A=1, B=0

L.H.S => $\overline{A . B} = \overline{1 . 0} = \overline{0} = 1$

R.H.S => $\overline{A} + \overline{B} = \overline{1} + \overline{0} = 0 + 1 = 1$

Case 4: A=1, B=1

L.H.S => $\overline{A . B} = \overline{1 . 1} = \overline{1} = 0$

R.H.S => $\overline{A} + \overline{B} = \overline{1} + \overline{1} = 0 + 0 = 0$

Truth table

A	B	$\overline{A . B}$	$\overline{A} + \overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

CHECK YOUR PROGRESS 2

1. An inverter performs an operation known as
(a) Complementation (b) assertion
(c) Inversion (d) both answers (a) and (c)

2.The output of gate is LOW when at least one of its inputs is HIGH. It is true for
(a) AND (b) NAND (c) OR (d) NOR

3.The output of gate is HIGH when at least one of its inputs is LOW. It is true for
(a) AND (b) OR (c) NAND (d) NOR

4.The output of a gate is HIGH if and only if all its inputs are HIGH. It is true for
(a)XOR (b) AND (c) OR (d) NAND

5. The output of a gate is LOW if and only if all its inputs are HIGH. It is true for
(a)AND (b) XNOR (c) NOR (d) NAND

6.Which of the following gates cannot be used as an inverter?

(a)NAND (b) AND (c) NOR (d) None of the above

7.The complement of a variable is always
(a) 0 (b) 1 (c) equal to the variable (d) the inverse of the variable

8.Which one of the following is not a valid rule of Boolean algebra?
(a) $A + 1 = 1$ (b) $A = \bar{A}$ (c) $A.A = A$ (d) $A + 0 = A$

9. Which of the following rules states that if one input of an AND gate is always 1 , the output is equal to the other input ?
(a) $A + 1 = 1$ (b) $A + A = A$ (c) $A.A = A$ (d) $A . 1 = A$

1.4 SUMMARY

- A binary number is a weighted number in which the weight of each whole number digit is a positive power of 2 and the weight of each fractional digit is a negative power of 2.
- The 1's complement of a binary number is derived by changing 1s to 0s and 0s to 1s
- The 2's complement of a binary number can be derived by adding 1 to the 1's complement.
- The octal number system consists of eight digits, 0 through 7.
- The hexadecimal number system consists of 16 digits and characters, 0 through 9 followed by A through F.
- The ASCII is a 7-bit alphanumeric code that is widely used in computer systems for input/output of information.
- The output of an inverter is the complement of its input
- The output of an AND gate is high only if all the inputs are high
- The output of an OR gate is high if any of the inputs is high
- The output of an NOR gate is low if any of the inputs is high
- The output of an NAND gate is low only if all the inputs are high
- The output of an exclusive-OR gate is high when the inputs are not the same

1.5 GLOSSARY

Alphanumeric : Consisting of numerals, letters, and other characters.

ASCII : American Standard Code for Information Interchange.

BCD : Binary Coded Decimal ; a digit code in which each of the decimal digits , 0 through 9, is represented by a group of four bits.

Binary : Describes a number system that has a base of two and utilizes 1 and 0 as its digits.

Boolean algebra : The mathematics of logic circuits.

Gate : A logic circuit that performs a specified logic operation , such as AND, OR or NOT.

Hexadecimal : Describes a number system with a base of 16

Octal : Describes a number system with a base of 8.

Parity : Parity is based on the number of 1's in a binary word. If the number of 1's in a word is odd, then it is a odd parity word and if the number of 1's is even, then it is an even parity word.

Truth table : A table showing the inputs and corresponding output levels of a logic circuit.

Universal gate : Either a NAND gate or a NOR gate.

1.6 REFERENCES

1. Moris Mano, "Digital Computer Fundamentals" TMH 3rd Edition
2. Thomas C Bartee "Computer Architecture and Logic and logic Design" TMH
3. Malvino and Leech "Digital Principles and Applications", TMH
4. Thomas L.Floyd "Digital fundamentals" Pearson Education 8th Edition

1.7 ANSWERS TO CHECK YOUR PROGRESS QUESTIONS

Check Your Progress 1

1.(d) 2.(a) 3.(c) 4.(a) 5.(d) 6.(b) 7.(d) 8.(b) 9.(c) 10.(c)

Check Your Progress 2

1.(d) 2.(d) 3.(c) 4.(b) 5.(d) 6.(b) 7.(d) 8.(b) 9.(d)