Unit-3
DIGITAL ELECTONICS

Introduction

Many number system were introduced with the passage of time like:

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

Number System	Base	Digits/Alphabets used
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F A = 10, B = 11, C = 12, D = 13, E = 14, F = 15

Decimal Number system

In decimal number system we can express any decimal number in units, tens, hundreds, thousands and so on.

Example: 5678.9

$$5*10^3+6*10^2+7*10^1+8*10^0+9*10^{-1}$$

It can be written as 5678.9_{10} where 10 subscript indicates radix or base. The left most digit which has greatest weight is called most significant digit (MSD) and right most digit which has least weight is called least significant digit (LSD).

Binary Number System

The computer system is designed on the **binary number system.** It has a base of 2. There are only two digits, 0 and 1. It can represent any number with these two digits.

The binary number system is also a positional numbering system wherein each binary digit has its own value or weight expressed as a power of 2.

Consider the example, 1011.0101 — a binary number with a fraction.

Positional values (weights)	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}
MSD	1	0	1	1	.	0	1	0	1 LSD

For a binary number, the **binary point** separates the whole number part from the fractional part. The place values of the digits to the right of the binary point are got by raising 2 to successive negative powers.

Octal Number System

The **octal number system** has a base of 8. It has eight digits 0, 1, 2, 3, 4, 5, 6, and 7. Each digit of an octal number can have any value from 0 to 7.

The octal number system is also a positional numbering system. Each octal digit has its own positional value or weight expressed as a power of 8.

As an example, consider 1267.12, an octal number with a fraction.

Hexadecimal Number System

The **hexadecimal number system** has 16 as the base number. It has ten numeric digits 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and six letters A, B, C, D, E, F where A = 10, B = 11, C = 12, D = 13, E = 14 and F = 15. The hexadecimal system is also a positional numbering system. The value of a hexadecimal digit is expressed as the power of 16.

As an example, consider a hexadecimal number with a fraction— A65.C2, and the place values or positional values of each digit as a power of 16.

Positional values (weights)	16^2	16^1	16^0		16^{-1}	16^{-2}
MSD →	A	6	5	.	C	2 ← LSD

Conversion to Decimal Number System

Numbers in the binary, octal, or hexadecimal systems can be converted to decimal numbers.

The steps in general are:

1. Find the positional value of each digit.
2. Multiply the digit with the positional value.
3. Sum up the product calculated in step 2.
4. The total is equivalent to the value in the decimal number system.

Conversion of Binary to Decimal

A binary number can be converted to its equivalent decimal number by adding up the product of each digit value (0 or 1) with its positional value, as shown below:

$$1011.0101_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4}$$

$$= 8 + 0 + 2 + 1 + 0 + 0.25 + 0 + 0.0625$$

$$= 11.3125_{10}$$

CONVERSION OF OCTAL TO DECIMAL

Conversion of Octal to Decimal

To convert an octal number to its equivalent decimal number, add up the product of each digit value (0 to 7) with its positional value, as given below:

$$1267.12_8 = 1 \times 8^3 + 2 \times 8^2 + 6 \times 8^1 + 7 \times 8^0 + 1 \times 8^{-1} + 2 \times 8^{-2}$$
$$= 512 + 128 + 48 + 7 + 0.125 + 0.0312$$
$$= 695.1562_{10}$$

CONVERSION OF HEXADECIMAL TO DECIMAL

Conversion of Hexadecimal to Decimal

Similarly, to convert a hexadecimal number to its equivalent decimal number by add up the product of each digit value (0 to 9, A to F) with its positional value, as shown below:

$$A65.C2_{16} = A \times 16^2 + 6 \times 16^1 + 5 \times 16^0 + C \times 16^{-1} + 2 \times 16^{-2}$$
$$= 10 \times 256 + 6 \times 16 + 5 \times 1 + 12/16 + 2/256$$
$$= 2560 + 96 + 5 + 0.75 + 0.0078$$
$$= 2561.7578_{10}$$

Decimal to Binary

For example, to convert 47.598_{10} to binary number, do the following:

Integer Part

2	47	
2	23	– 1
2	12	– 1
2	6	– 0
2	3	– 0
2	1	– 1
2	0	– 1

$47_{10} = 110011_2$

Fractional Part

0.598
× 2
1.196
× 2
0.392
× 2
0 .784

$0.598_{10} = .100_2$

Decimal to Octal

The conversion method of decimal to octal is the same as that of decimal to binary except that the base taken is 8 instead of 2.

For example, to convert 765.245_{10} to the octal equivalent, do the following.

Integer Part

8	765	
8	95	– 5
8	11	– 7
8	1	– 3
8	0	– 1

$765_{10} = 1375_8$

Fractional Part

0.245
× 8
1.960
× 8
7.680
× 8
5.440

$0.245_{10} = .175_8$

$765.245_{10} = 1375.175_8$

DECIMAL TO HEXADECIMAL

The conversion method of decimal to hexadecimal is the same as that of decimal to binary except that the base taken is 16 instead of 2.

For example, to convert 765.245_{10} to the hexadecimal equivalent, do the following:

Integer Part

16	765
16	47 - 13
16	2 - 15
	0 - 2

Fractional Part

0.245
x 16
3.920
x 16
14.720
x 16
11.520

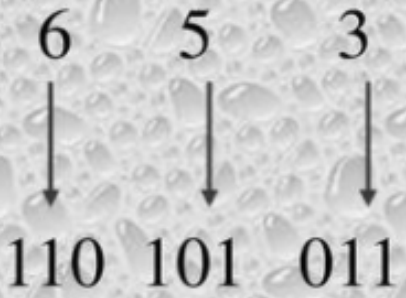
$765.245_{10} = 2FD.3EB_{16}$

OCTAL TO BINARY CONVERSION

Each octal number converts to 3 binary digits

Code
0 - 000
1 - 001
2 - 010
3 - 011
4 - 100
5 - 101
6 - 110
7 - 111

To convert 653_8 to binary, just substitute code:

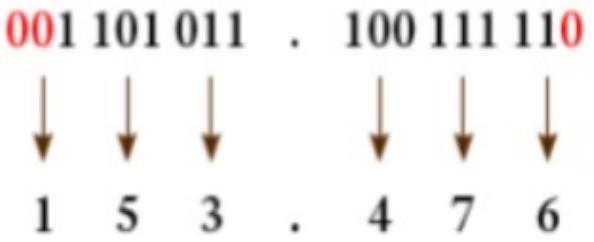


Binary to Octal

Let us now see how to convert a binary number into an octal number.

For example, to convert 1101011.10011111_2 to its octal equivalent, do as follows:

1. For the integer part, divide the binary digits into groups of three from right (LSD) to left (MSD).
2. For the fractional part, make groups of three binary digits each from left (MSD) to right (LSD).
3. In steps 1 and 2, if a group has less than 3 binary digits, insert the required number of 0s (shown in red below).
4. Convert each group to the equivalent octal number as shown in the table above.



The corresponding octal number is 153.476_8

Binary – Hexadecimal Conversion

- $16 = 2^4$
- Each group of 4 bits represents a hexadecimal digit

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Binary to Hexadecimal

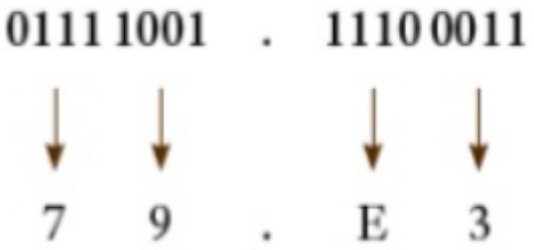
For example, to convert 1111001.11100011_2 to hexadecimal, do as follows:

1. For the integer part, divide the binary digits into groups of four from right (LSD) to left (MSD).

2. For the fractional part, divide the binary digits into groups of four from left (MSD) to right (LSD).

3. In steps 1 and 2, if a group has less than four binary digits, insert the required number of 0s (shown in red below).

4. Convert each group to the equivalent hexadecimal number as given in the table above.



The hexadecimal number is $79.E3_{16}$

BINARY REPRESENTATION OF A NUMBER

BINARY REPRESENTATION OF A NUMBER

Integers may be represented in three different ways in a computer, namely,

1. **Sign and magnitude** method
2. **One's complement** method
3. **Two's complement** method

SIGN AND MAGNITUDE

Sign and Magnitude

In this method, integers are identified by the **sign** (+ or –) and a string of digits which represents the **magnitude.**

To represent the sign of a number, the most significant bit (MSB) is used. It holds the value 0 for a **positive number** and the value 1 for a **negative number.**

For example, if data is stored as a 1 byte word,

+ 23 will be represented as follows:

0	0	0	1	0	1	1	1

–18 will be represented as follows:

1	0	0	1	0	0	1	0

1's Complement

The 1's complement of a binary number is the number that results when we change all 1's to zeros and the zeros to ones.

1	1	0	1	0	0	1	0
NOT OPEARATION							
0	0	1	0	1	1	0	1

ONE'S COMPLEMENT

One's Complement

The **one's complement** method represents positive numbers by their binary equivalents (called **true forms**) and negative numbers by their **one's complement** forms.

Now, let us understand how to find the one's complement of a binary number.

For this, just replace every 0 with 1 and every 1 with 0.

For example, the one's complement of 00011100 is 11100011.

Thus in this method, +28 is represented as follows:

0	0	0	1	1	1	0	0

And –28 is represented as follows:

1	1	1	0	0	0	1	1

2's Complement

The 2's complement the binary number that results when add 1 to the 1's complement. It's given as,

2's complement = 1's complement + 1

Example: Express 35 in 8-bit 2's complement form.

Solution:

35 in 8-bit form is 00100011

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

9's Complement

The nines' complement of a decimal digit is the number that must be added to it to produce 9. The complement of 3 is 6, the complement of 7 is 2.

Example: Obtain 9's complement of 7493

Solution:

```
  9 9 9 9
- 7 4 9 3
---------
  2 5 0 6   <-- 9's complement
```

10's Complement

The 10's complement of the given number is obtained by adding 1 to the 9's complement. It is given as,

10's complement = 9's complement + 1

Example: Obtain 10's complement of 7493

Solution:

```
  9999        2506
- 7493      +    1
-------     ------
  2506        2507   ← 10's complement
```

Binary Addition

The addition consists of four possible elementary operations:

Sr no.	Operations
0.	0+0=0
1.	0+1=1
2.	1+0=1
3.	1+1=10 (0 with carry of 1)

In the last case, sum is of two digits: Higher Significant bit is called Carry and lower significant bit is called Sum.

Binary Addition

e.g.:

	1	1	0	0
+	0	1	1	0
1	0	0	1	0

Carry

Binary Subtraction

The subtraction consists of four possible elementary operations:

Sr no.	Operations
0.	0-0=0
1.	0-1=1(borrow 1)
2.	1-0=1
3.	1-1=0

In case of second operation the minuend bit is smaller than the subtrahend bit, hence 1 is borrowed.

Binary Subtraction

e.g.:

$$\begin{array}{rcccc} & 0 & 1 & 0 & 1 \\ - & 0 & 1 & 1 & 0 \\ \hline & 1 & 1 & 1 & 1 \end{array}$$

Binary Multiplication

Rules for Binary Multiplication are:

Sr no.	Operations
0.	0*0=0
1.	0*1=0
2.	1*0=0
3.	1*1=1

e.g.: Multiply 110 by 10

```
        1   1   0
    *       1   0
  -------------------
        0   0   0
+   1   1   0   0
  -------------------
    1   1   0   0
```

Binary Division

Rules for Binary Division are:

Sr no.	Operations
0.	0/0=0
1.	1/0=0
2.	0/1=0
3.	1/1=1

e.g.: Divide 110 by 10

```
          1   1
      ┌──────────
1   0 │ 1   1   0
        1   0
      ──────────
        0   1   0
            1   0
      ──────────
            0   0
```

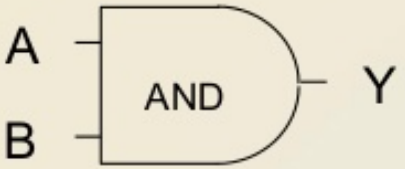
Logic Gate

- A gate is an digital circuit which operates on one or more signals and produce single output.
- Gates are digital circuits because the input and output signals are denoted by either 1(high voltage) or 0(low voltage).
- Three type of gates are as under:
    1. AND gate
    2. OR gate
    3. NOT gate

AND Function

Text Description ⇒ Output Y is TRUE if inputs A *AND* B are TRUE, else it is FALSE.

Logic Symbol ⇒



Truth Table ⇒

INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

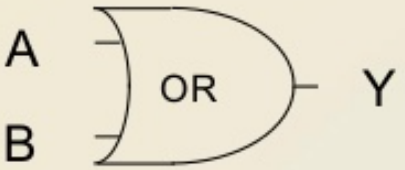
AND Gate Truth Table

Boolean Expression ⇒ $Y = A \times B = A \cdot B =$ (AND Symbol)

OR Function

Text Description ⇒ Output Y is TRUE if input A *OR* B is TRUE, else it is FALSE.

Logic Symbol ⇒



Truth Table ⇒

INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

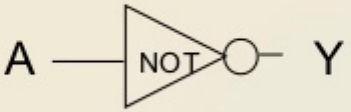
OR Gate Truth Table

OR Symbol

Boolean Expression ⇒ $Y = A + B$

NOT Function (inverter)

Text Description ⇒ Output Y is TRUE if input A is FALSE, else it is FALSE. Y is the inverse of A.

Logic Symbol ⇒ 

Truth Table ⇒

INPUT A	OUTPUT Y
0	1
1	0

NOT Gate Truth Table

Boolean Expression ⇒ $Y = \overline{A}$ (NOT Bar)

Alternative Notation

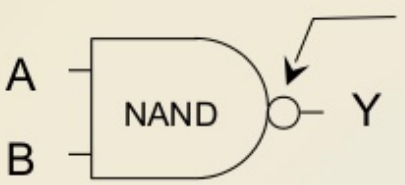
Y = A'

Y = !A

NAND Function

Text Description ⇒ Output Y is FALSE if inputs A *AND* B are TRUE, else it is TRUE.

Logic Symbol ⇒



Truth Table ⇒

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

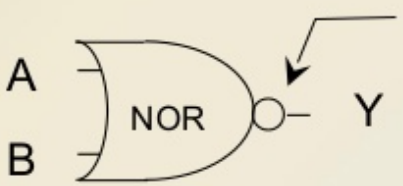
NAND Gate Truth Table

Boolean Expression ⇒ $Y = \overline{A \times B} = \overline{AB}$

NOR Function

Text Description ⇒ Output Y is FALSE if input A *OR* B is TRUE, else it is TRUE.

Logic Symbol ⇒



Truth Table ⇒

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate Truth Table

Boolean Expression ⇒ $Y = \overline{A + B}$

Ex- OR Gate

(a) Circuit symbol

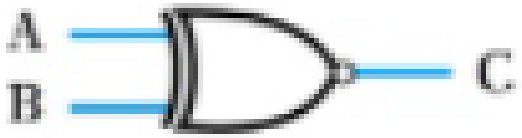
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

(b) Truth table

$$C = A \oplus B$$

(c) Boolean expression

The Exclusive NOR gate



(a) Circuit symbol

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

(b) Truth table

$$C = \overline{A \oplus B}$$

(c) Boolean expression

Basic Theorem of Boolean Algebra

T1 : Properties of 0

(a) $0 + A = A$

(b) $0\,A = 0$

T2 : Properties of 1

(a) $1 + A = 1$

(b) $1\,A = A$

Basic Theorem of Boolean Algebra

T3 : Commutative Law

(a) $A + B = B + A$

(b) $A\,B = B\,A$

T4 : Associate Law

(a) $(A + B) + C = A + (B + C)$

(b) $(A\,B)\,C = A\,(B\,C)$

T5 : Distributive Law

(a) $A\,(B + C) = A\,B + A\,C$

(b) $A + (B\,C) = (A + B)\,(A + C)$

(c) $A+A'B = A+B$

Basic Theorem of Boolean Algebra

T6 : Indempotence (Identity) Law

(a) $A + A = A$

(b) $A\ A = A$

T7 : Absorption (Redundance) Law

(a) $A + A\ B = A$

(b) $A\ (A + B) = A$

Basic Theorem of Boolean Algebra

T8 : Complementary Law

(a) $X+X'=1$

(b) $X.X'=0$

T9 : Involution

(a) $x'' = x$

T10 : De Morgan's Theorem

(a) $(X+Y)'=X'.Y'$

(b) $(X.Y)'=X'+Y'$

Laws in Boolean Algebra

- **Commutative Law**

A.B = B.A

A+B = B+A

- **Associative Law**

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

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

- **Distributive Law**

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

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

- **Absorption**

A+ (A.B)=A

A.(A+B)=A

A+AB = A
A+A'B =A+B
(A+B)(A+C) = A+BC

- **AND Law**

A.0 = 0

A.1 =A

A.A = A

A.A' =0

- **OR law**

A+0 = A

A+1=1

A+A=A

A+A' = 1

- **Inversion Law(Involution)**

A'' = A

- **DeMorgan's Theorm**

(x.y)' = x' + y'

(x+y)' = x' . y'

Idempotent Law

Complement Law

Boolean Algebric Function

- Consider the following Boolean function:

Canonical Form

$F1 = x'y'z + xy'z' + xy'z + xyz' + xyz$

After Simplification

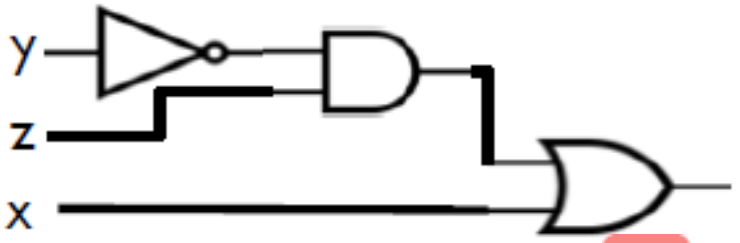
$F1 = x + y'z$

Non Canonical Form

- A Boolean function can be represented in a truth table.

Truth Table

x	y	z	F1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1



Realization of Boolean Function using Gates

Representation of Boolean expression

Boolean expression can be represented by either

(i)Sum of Product(SOP) form or

(ii)Product of Sum (POS form)

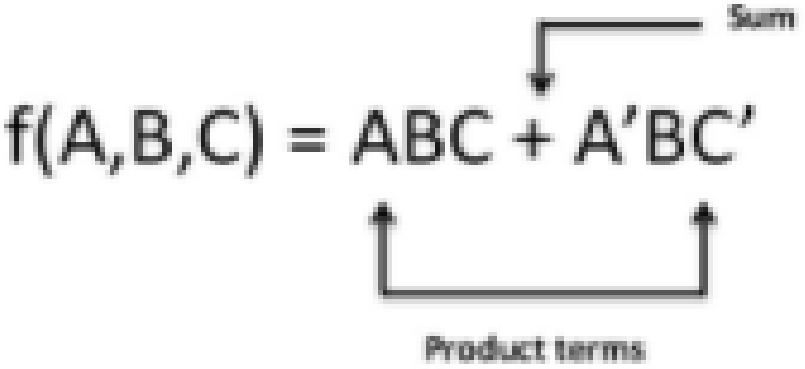
e.g.

AB+AC → SOP
(A+B)(A+C) → POS

In above examples both are in SOP and POS respectively but they are not in Standard SOP and POS.

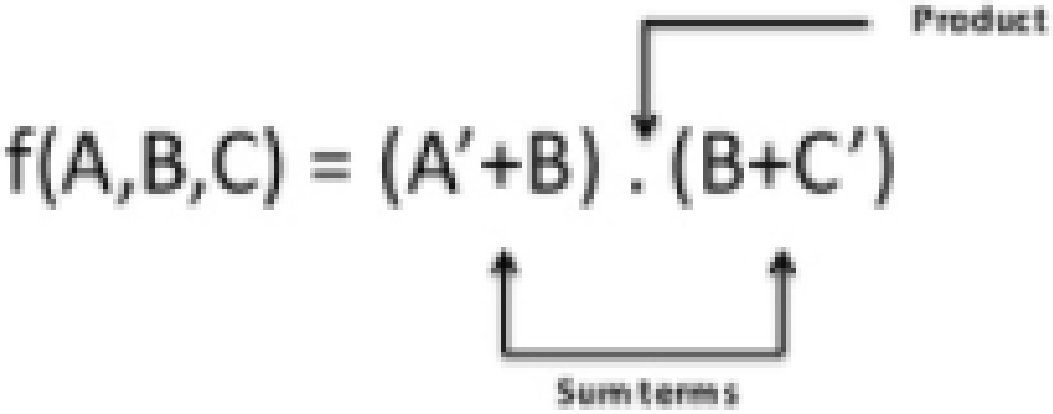
Sum of Product

- The sum-of-products (**SOP**) form is a method (or form) of simplifying the Boolean expressions of logic gates.
- Sum and product derived from the symbolic representations of the OR and AND functions.
- OR (+) , AND (.) , addition and multiplication.



Product of Sum

- When two or more sum terms are multiplied by a Boolean OR operation.
- Sum terms are defined by using OR operation and the product term is defined by using AND operation.



Canonical Forms For Boolean Function

- SoP form – Example

F1= x'yz' + xy'z + xyz' + xyz

F1 = (m2+m5+m6+m7)

F1 = ∑(m2,m5,m6,m7)

F1 = ∑ (2, 5,6,7)

Decimal numbers in the above expression indicate the subscript of the minterm notation

x	y	z	F1	Minterms	
0	0	0	0	x'y'z'	m0
0	0	1	0	x'y'z	m1
0	1	0	1	x'yz'	m2
0	1	1	0	x'yz	m3
1	0	0	0	xy'z'	m4
1	0	1	1	xy'z	m5
1	1	0	1	xyz'	m6
1	1	1	1	xyz	m7

PoS form – Example

F2=(x+y+z).(x+y+z').(x+y'+z').(x'+y+z)

F2 = (M1.M2.M4.M5)

F2 = $\prod$(M1,M2,M4,M5)

F2 = $\prod$(1, 2,4,5)

Decimal numbers in the above expression indicate the subscript of the Maxterm notation

x	y	z	F2	Maxterms	
0	0	0	0	x + y+z	M1
0	0	1	0	x+y+z'	M2
0	1	0	1	x+y' + z	M3
0	1	1	0	x+y'+z'	M4
1	0	0	0	x'+y+z	M5
1	0	1	1	x' +y+z'	M6
1	1	0	1	x'+y'+z	M7
1	1	1	1	x'+y'+z'	M8

- **Example: Express the following in SoP form**

F1 = x + y'z

- **Solution:**

=(y+y')x + y'z(x+x') [because x+x'=1]

=xy + xy' + xy'z + x'y'z

=xy(z+z') + xy'(z+z') + xy'z + x'y'z

=xyz + xyz' + xy'z + xy'z' + xy'z + x'y'z

=xyz + xyz' + (xy'z + xy'z) + xy'z' + x'y'z

= xyz + xyz' + xy'z + xy'z' + x'y'z [because x+x =x]

= m7 + m6 + m5 + m4 + m1

= ∑(m7, m6, m5, m4, m1)

= ∑(1,4,5,6,7)

- Using Boolean algebra techniques, simplify this expression: AB + A(B + C) + B(B + C)
- Solution

=AB + AB + AC + BB + BC (Distributive law)

=AB + AB + AC + B + BC (B.B=B)

= AB + AC + B + BC (AB+AB=AB)

= AB + AC + B (B+BC =B)

=B+AC (AB+B =B)

- Minimize the following Boolean expression using Algebric Simplification

$F(A,B,C)=A'B+BC'+BC+AB'C'$

- Solution

$=A'B+(BC'+BC')+BC+AB'C'$ [indeponent law]

$= A'B+(BC'+BC)+(BC'+AB'C')$

$= A'B+B(C'+C)+C'(B+AB')$

$=A'B + B.1+ c' (B+A)$

$= B(A'+1)+C'(B+A)$

$=B + C'(B+A)$ [A'+1=1]

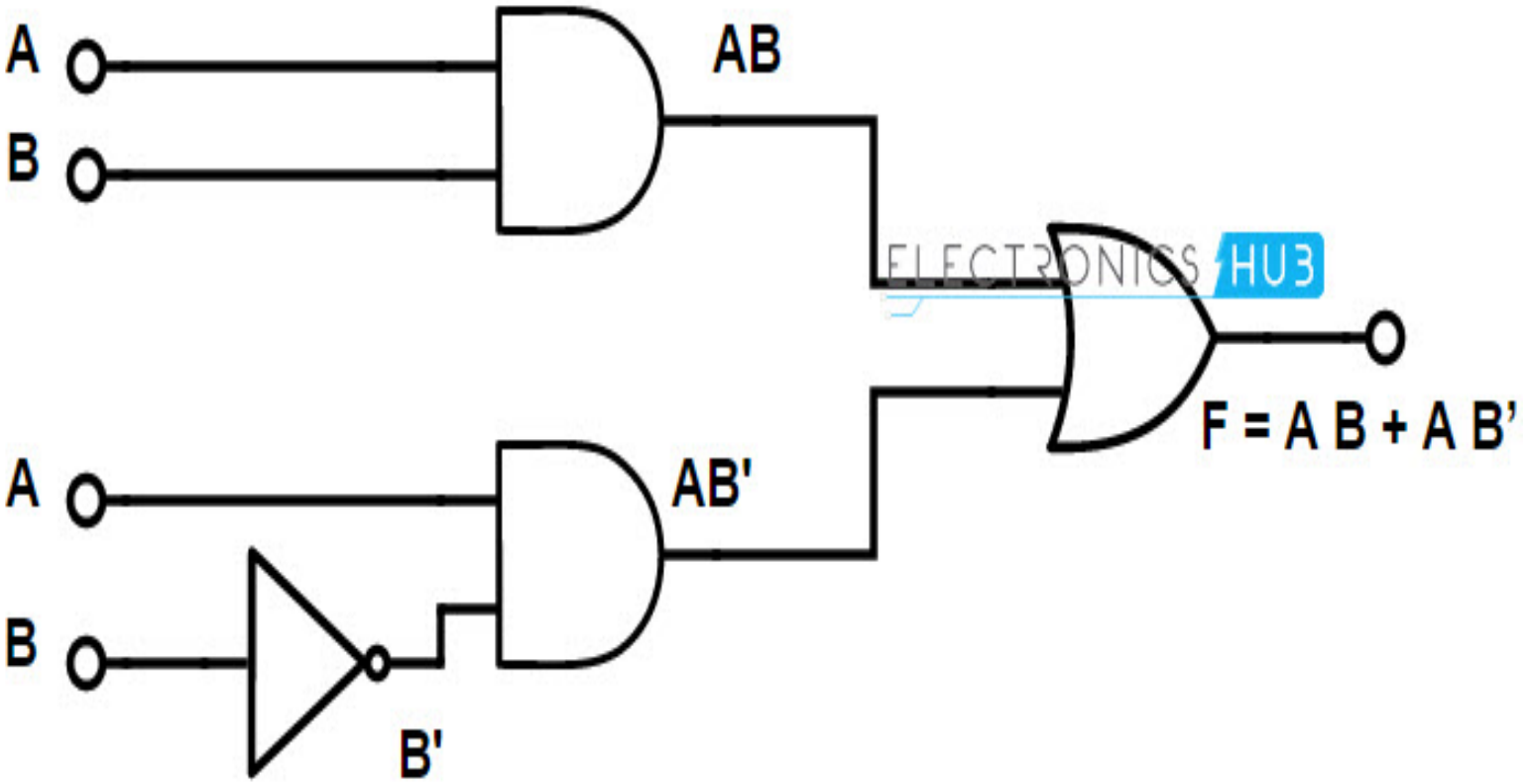
$= B+BC'+AC'$

$= B(1+C')+AC'$

$= B+AC'$ [1+C' = 1]

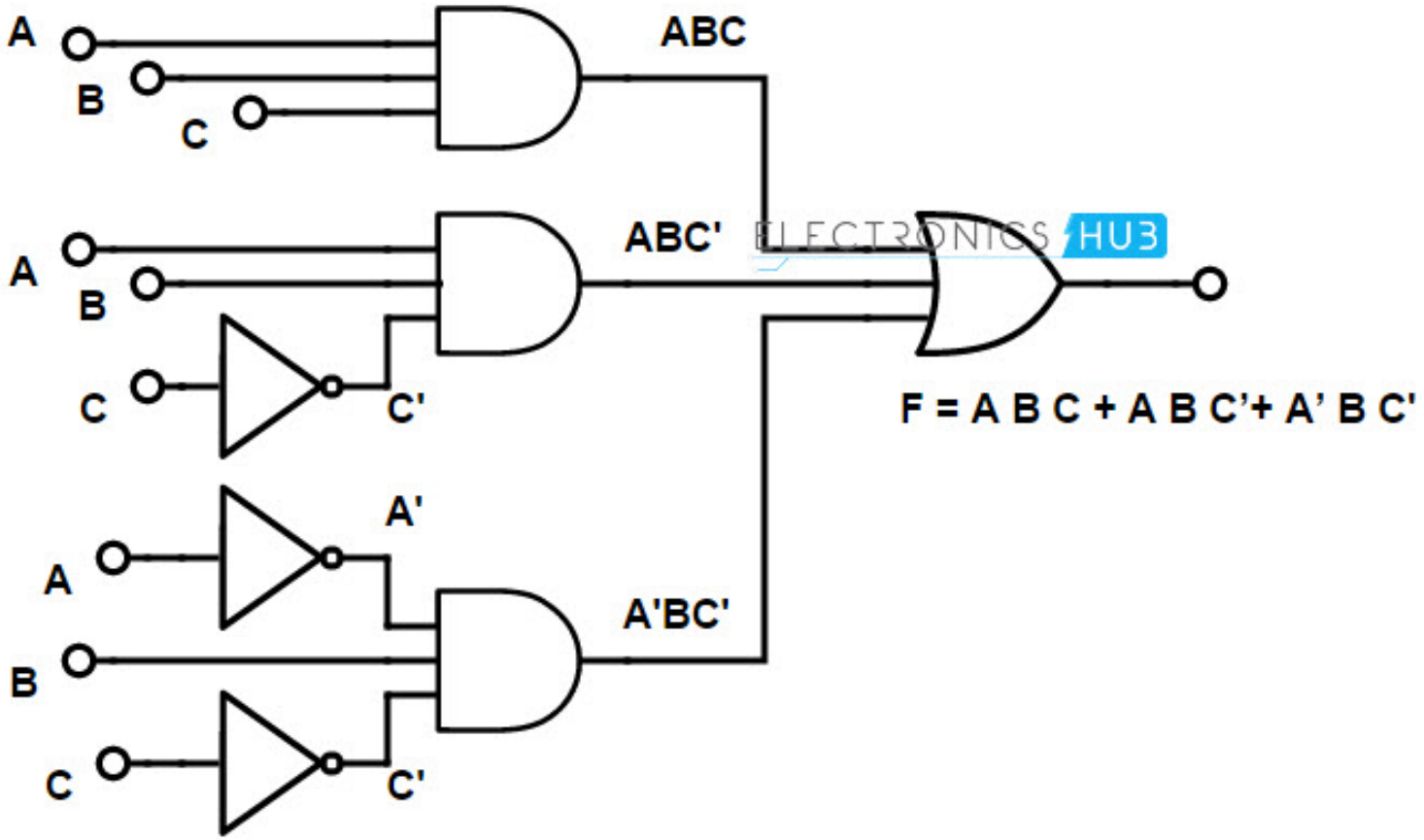
Implementation of Boolean Function using Logic Gates

Ex: F = A B + A B’

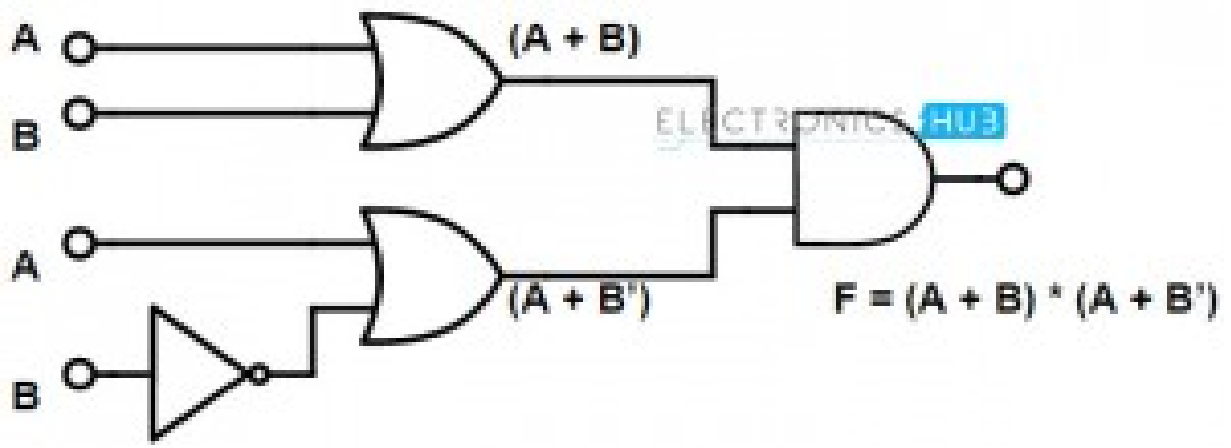


Implementation for 3 Input Variables

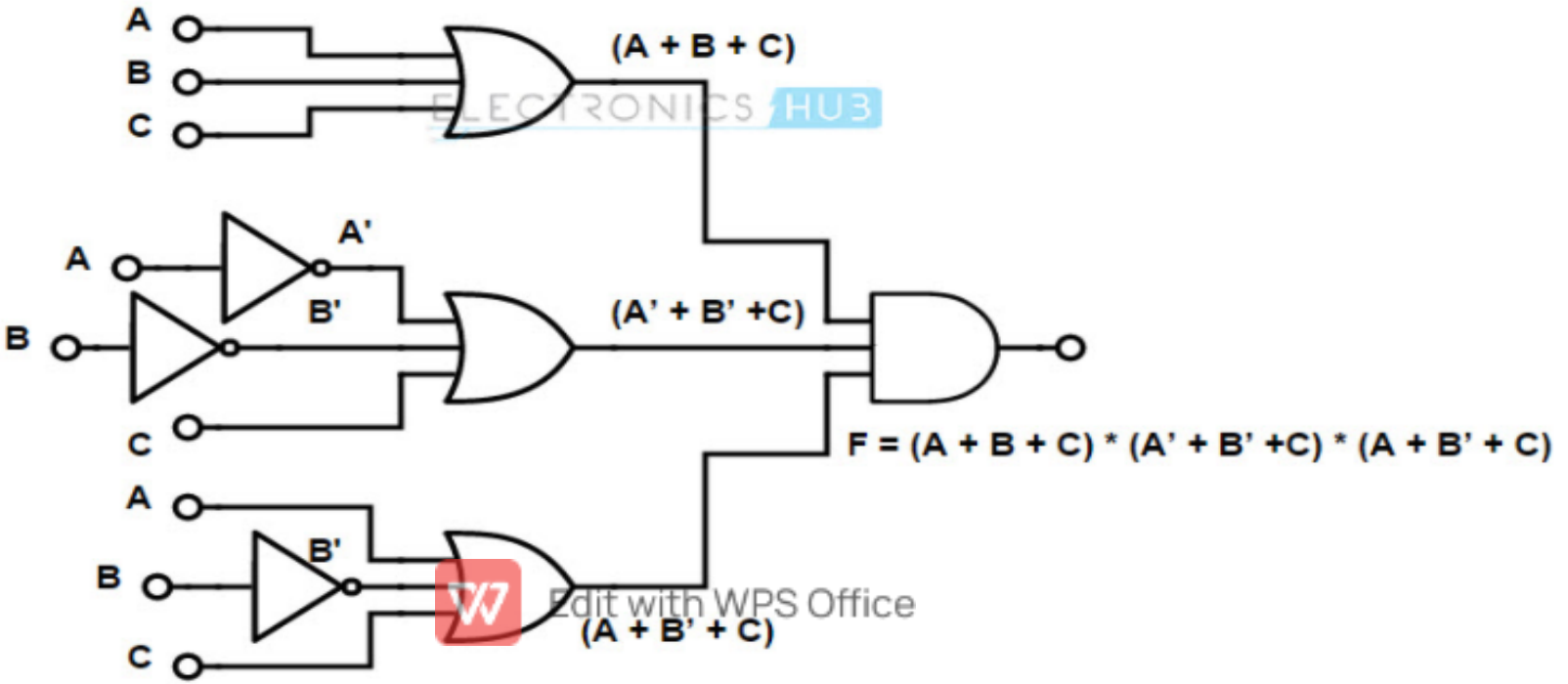
Ex: F = A B C + A B C’ + A’ B C’



F = (A + B) . (A + B)'



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



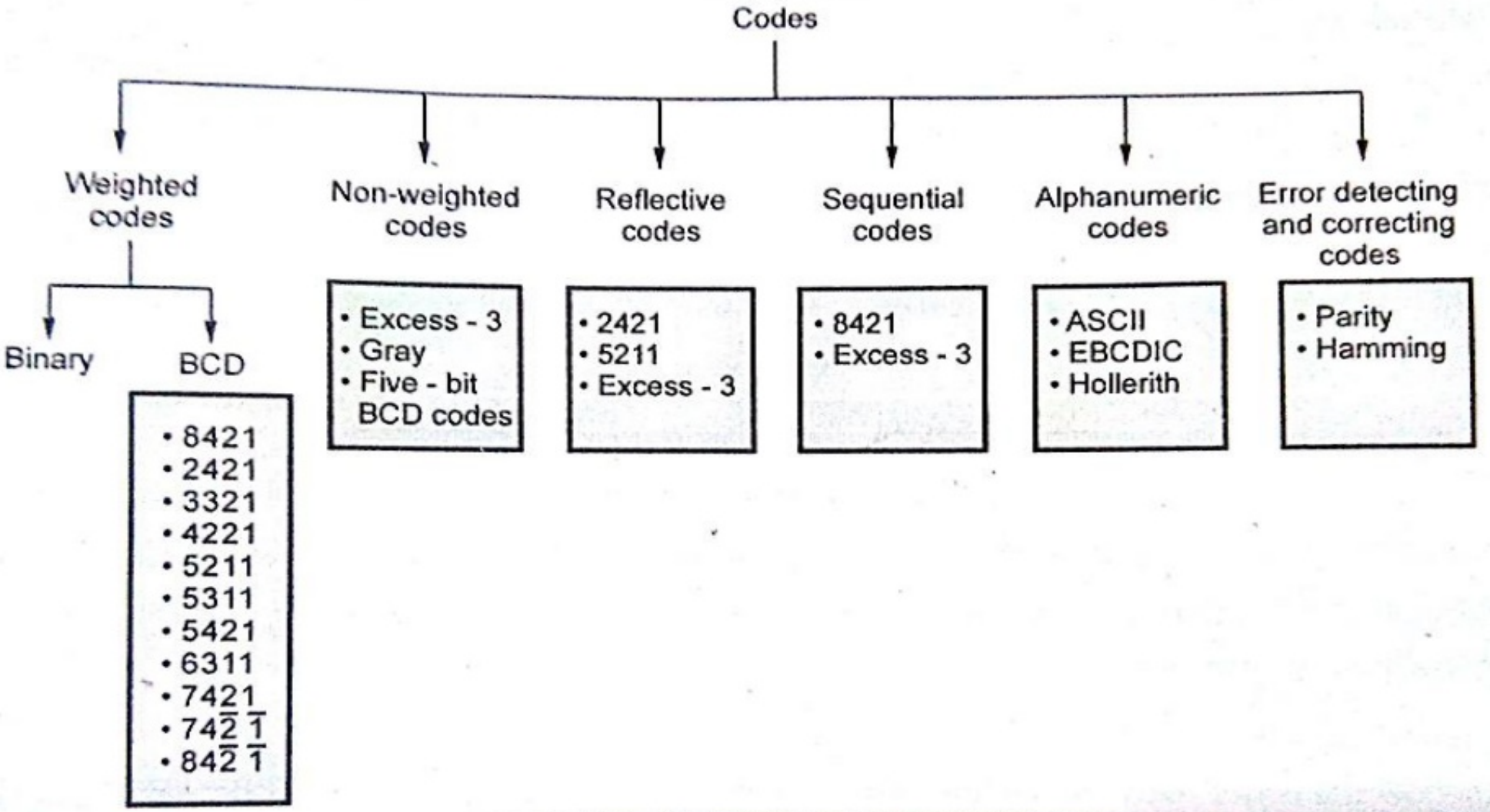
BINARY CODES

When numbers, alphabets or words are represented by a specific group of symbols i.e., they are encoded

The group of symbols used to encode them are called codes. The digital data is represented, stored and transmitted as groups of binary digits (bits)

Group of bits--- binary code---- numeric and alphanumeric code

CLASSIFICATION OF BINARY CODES



- ***Weighted codes:***
  - ✓ In weighted codes, each digit position of the number represents a specific weight
  - ✓ Examples: 93→(1001)(0011) → 8421 code

    93→(1100)(0011) →5421 code
- ***Non-weighted codes***
  - ✓ Non-weighted codes are not assigned with any weight to each digit position, i.e., each digit position within the number is not assigned fixed value
  - ✓ Excess-3 and gray codes are the non-weighted codes

Binary Codes

- BCD Code

It is a binary code in which each decimal digit is represented by a group of four bits. it is also called an 8421 code. The 8 4 2 1 indicates the binary weights of the four bits (2^3, 2^2, 2^1, 2^0).

- A number with N decimal digits will require 4N bits in BCD.
- Decimal 396 is represented in BCD with 12bits as 0011 1001 0110, with each group of 4 bits representing one decimal digit.
- A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9.

Binary-Coded Decimal (BCD)

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

Binary Codes)

- Gray Code
  - The advantage is that only bit in the code group changes in going from one number to the next.
    - Error detection.
    - Representation of analog data.
    - Low power design.

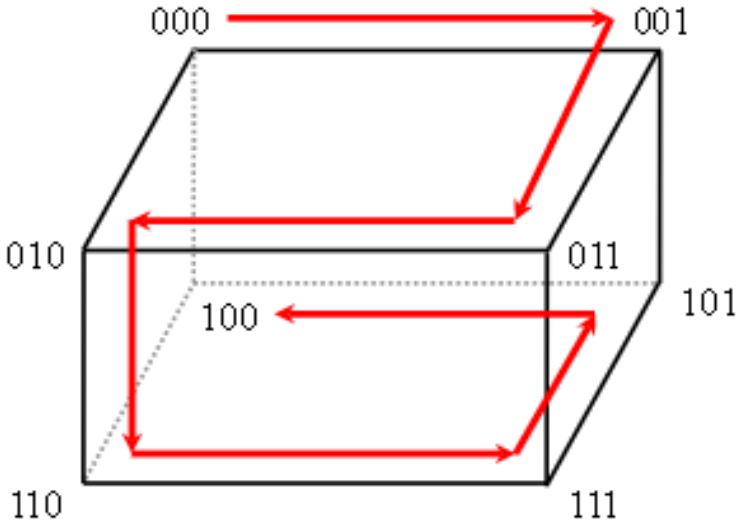
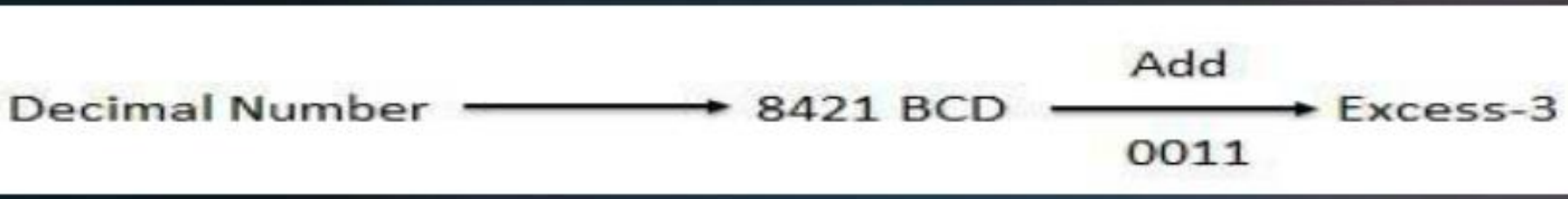


Table 1.6
Gray Code

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

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



Decimal	BCD 8 4 2 1	Excess-3 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