CSE-2323 (Digital logic Design)

Binary systems, Boolean Algebra and logic Gates

Signal: A signal is a function, that represents the variation of physical quantity with respect to any parameter.

Electrical signal: In electrical and electronics, usually signal is variation of electrical quantity (generally I or V) with respect to time. Two types of signal. ① Analog signal ② Digital signal

Analog signal: Analog signal is the signal which can take any value within the given limit.

DTS (Discrete time signal): The signal which is defined for the discrete intervals of time is called DTS.

It is the subset of Analog signal and we need to learn about it digital signals property.

Digital signal: In digital signals we use discrete both time and magnitude.

Need of digital signals:

T- Transmeter
R - Receiver

① Digital data can be easily expressed.

② Digital signals can convey information with less noise, distortion and interferance.

③ Digital signals are more accurate.

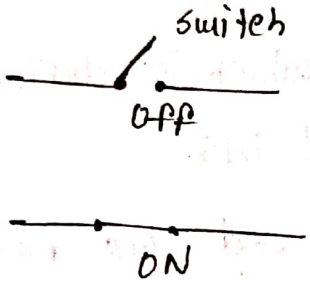
④ Digital signals canbe transmitted over long distances.

Digital electronics: Digital electronics is the branch of electronics that deals with the study of digital signals and the components that use or create them. Exp: computers, digital cameras, digital televisions, mobile phones etc.

ADC – Analog to Digital
DAC – Digital to Analog

Switch and bits intution:

Digital system –
uses less bandwith



off → 0
ON → 1

$n = 2^m$
$m = 3 = 2^3 = 8$

Introduction to Boolean Algebra:

Boolean Algebra is the set of rules used to simplify the given logic expression without changing it functionality.
It is used when number of variables are less.

Rules: ① Complement Rule: $(A')' = A$

② AND Rule: $A.A = A$
$A.0 = 0$
$A.1 = A$
$A.A' = 0$

③ OR Rule: $A+A=A$

$A+0=A$

$A+1=1$

$A+A'=1$

④ Distributive law: $A.(B+C)=A.B+A.C$

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

⑤ Commutative law: $A+B=B+A$

$A.B=B.A$

⑥ Associative ~~Rut~~ law: $(A.B).C=A.(B.C)$

⑦ De Morgan's law: $\overline{(A+B)}=\overline{A}.\overline{B}$

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

Examples: ① $AB+AB'$

Let, $F=AB+AB'$

$=A(B+B')$

$=A.1$

$=A$

$\therefore F=A.$

② $AB+AB'C+AB'C'$

Let, $F=AB+AB'C+AB'C'$

$=A[B+B'C+B'C']$

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

$=A[B+B']$

$=A[1]$

$=A$

③ $F=AB+A'C+BC$

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

$=AB+A'C+ABC+A'BC$

$=AB+ABC+A'C+A'BC$

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

$=AB.1+A'C.1$

$=AB+A'C.$

④ $F = (A+B+C)(A+B'+C)(A+B+C')$

Let, $x = A+B$

$F = (x+C)(A+B'+C)(x+C')$

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

$= (x+C.C')(A+B'+C)$

$= (x+0)(A+B'+C)$

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

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

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

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

$= A + BB' + BC$

$= A + 0 + BC$

$= A + BC$

⑤ $G = (A+B)(A+B')(A'+B)(A'+B')$

$= (A+BB')(A'+BB')$

$= (A+0)(A'+0)$

$= A.A'$

$= 0$

Redundancy Theorem / Consensus theorem:

Conditions:
1. Three Variables
2. Each variable is repeated
3. One variable is complemented
4. Take the complemented variable

1. $F = AB + A'C + BC$
$= AB + A'C$

2. $F = AB + BC' + AC$
$= BC' + AC$

3. $F = AB' + BC + AC$
$= AB' + BC$

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

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

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

Sum of Product:

Sum of product form is a Boolean Algebra expression in which different 'Product' terms from inputs are 'summed' together. This product is not arithmatical multiply but it is Boolean logical AND and the sum is Boolean logical OR.

To understand better about SOP. We need to know about min term.

Min term: Min term means the term that is true for a minimum number of combination of inputs. It is denoted by 'm'.

A	B	C	Min term
0	0	0	$m_0 = \bar{A}\bar{B}\bar{C}$
0	0	1	$m_1 = \bar{A}\bar{B}C$
0	1	0	$m_2 = \bar{A}B\bar{C}$
0	1	1	$m_3 = \bar{A}BC$
1	0	0	$m_4 = A\bar{B}\bar{C}$
1	0	1	$m_5 = A\bar{B}C$
1	1	0	$m_6 = AB\bar{C}$
1	1	1	$m_7 = ABC$

Types of sum of product forms:

1. Canonical SOP form
2. Non-canonical SOP form
3. Minimal SOP form

① Canonical SOP form: This is the standard form of sum of product. This is also known as the sum of min terms. It is represented by summation sign Σ and minterms in the braces for which the output is true.

Exp:

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

For this function the canonical SOP expression is

$F = \Sigma(m_1, m_2, m_3, m_5)$

By expanding,

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

This is the canonical form of the given function.

② Non canonical SOP form: It is the non standardized form of SOP expressions. The product terms are not the min terms but they are simplified.

For exp: $F = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C$

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

$= \bar{A}\bar{B}C + \bar{A}B(1) + A\bar{B}C$

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

Minimal SOP form: This form is the most simplified SOP expression of a function. It is also a form of non-canonical form.

Exp: 1. For the given truth table, minimize the SOP expression.

A	B	y	
0	0	0	m_0
0	1	1	m_1
1	0	0	m_2
1	1	1	m_3

Solution: $y = m_1 + m_3$

$= \bar{A}B + AB$

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

$= B \cdot 1$

$= B$ (Ans)

② Simplify the expression for $y(A,B) = \Sigma m(0,2,3)$

Solution: $y = \Sigma m(0,2,3)$

$= m_0 + m_2 + m_3$ —①

A	B	y	
0	0	1	$\rightarrow m_0 = \bar{A}\bar{B}$
0	1	0	$\rightarrow m_1 = \bar{A}B$
1	0	1	$\rightarrow m_2 = A\bar{B}$
1	1	1	$\rightarrow m_3 = AB$

From ①,

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

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

$= \bar{B} + AB$

$= \bar{B} + A$ $[\because \bar{A} + AB = \bar{A} + B]$

$= A + \bar{B}$

Product of Sum (POS): The product of sum is a form in which products of different sum terms of inputs are taken.

To better understand about product of sum, we need to know about Max term (M):

Max term: Max term means the term that is true for a maximum number of input combinations.

Max term of two input variable are given below:

A	B	Max term
0	0	$M_0 = A+B$
0	1	$M_1 = A+\bar{B}$
1	0	$M_2 = \bar{A}+B$
1	1	$M_3 = \bar{A}+\bar{B}$

Types of product of sum forms: ① Canonical
② Non-canonical
③ Minimal

① Canonical POS form: The canonical form contains all inputs either complemented or non-complemented in its each sum term.

② Non-Canonical form: The product of sum expression that is not in standard form is called non-canonical form.

③ Minimal form: This is the most simplified and optimized form.

Exp: ① For the given truth table minimize the POS expression

A	B	y
0	0	1
0	1	0
1	0	1
1	1	0

or simplify the expression for $y = \pi(M_1, M_3)$

Solution: $(A+\bar{B}) \cdot (\bar{A}+\bar{B})$

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

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

$= \bar{B}$ (Ans)

② For the given truth table minimize the SOP and POS form.

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

Solution: The output is high for $m_0, m_2, m_3, m_6, m_7,$

∴ Sum of product form,

$$Y = m_0 + m_2 + m_3 + m_6 + m_7$$

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

or, $Y = \bar{A}\bar{B}\bar{C} + \bar{A}B(C+\bar{C}) + AB(C+\bar{C})$

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

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

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

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

Now, the output is low for M_1, M_4, M_5:

Hence, the POS form,

$$Y = \pi M(1,4,5)$$

$$= M1 . M4 . M5$$

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

$$= (A+B+\bar{C}) . (\bar{A}+B+C.\bar{C}) \quad [(A+B)(A+C) = A+B.C]$$

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

$$= B + (A+\bar{C}) . \bar{A} \quad [\text{using distributive law}]$$

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

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

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

Minimal to canonical form (SOP):

① $y = A + B'C$

Solution: Given that, $y = A + B'C$

Step-1: 3 Variables A, B and C

Step-2: $m_1 \rightarrow A \checkmark \quad B \times \quad C \times$ $\qquad$ $m_2 \rightarrow A \times \quad B \checkmark \quad C \checkmark$

Step-3:-

$$y = A.1.1 + 1.B'.C$$
$$= A(B+B')(C+C') + (A+A')B'C$$
$$= (AB + AB)(C+C') + AB'C + A'B'C$$
$$= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C$$
$$\therefore y = ABC + ABC' + AB'C + AB'C' + A'B'C.$$

Minimal to canonical form (POS)

① $(A+B+C')(A'+C)$

Solution: Step-1: 3 Variables A, B, C

Step-2: $M_1 \rightarrow A \checkmark \quad B \checkmark \quad C \checkmark$ $\qquad$ $M_2 \rightarrow A \checkmark \quad B \times \quad C \checkmark$

Step-3: $F = (A+B+C')(A'+C+0)$

$$= (A+B+C')(A'+C+B.B')$$
$$= (A+B+C')(x+B.B')$$
$$= (A+B+C')(x+B)(x+B')$$
$$= (A+B+C')(A'+C+B)(A'+C+B')$$
$$= (A+B+C')(A'+B+C)(A'+B'+C) \quad \text{(Ans)}$$

Positive and Negative logic:

In positive logic the higher voltage correspond to logic '1' and lower voltage correspond to logic '0'

In negative logic the higher voltage correspond to logic '0' and lower voltage correspond to logic '1'

Positive and negative logic AND gate:

Input		output
A	B	C
L	L	L
L	H	L
H	L	L
H	H	H

⊕ logic

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

⊖ logic

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

Positive and negative logic OR gate:

Input		output
A	B	C
L	L	L
L	H	H
H	L	H
H	H	H

⊕ logic

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

⊖ logic

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

Proofs: 1. Prove that positive logic AND gate and negative OR gate are equivalent.

2. Prove that, Positive logic OR gate and negative logic AND gate are equivalent.

Dual form: The dual of a boolean expression is the expression one obtains by interchanging addition and multiplication and interchanging 0's and 1's.

Rules to write dual form:-

$A.B \xrightarrow{\text{dual}} A+B$

$A+B \xrightarrow{\text{dual}} A.B$

Self Dual: A function is said to be self dual if and only if its dual is equivalent to the given function.

Exp: Consider the function : $F(A,B,C) = AB+BC+CA$

The dual of this function is,

$F_d(A,B,C) = (A+B).(B+C).(C+A)$

$= AB+BC+CA$

Clearly, $F(A,B,C) = F_d(A,B,C)$

$\therefore F(A,B,C)$ is a self dual function.

Conditions for self dual function:

① The function must be a neutral function.
② The function must not contain any mutually exclusive terms.

Binary Numbers: A digital system is an interconnection of digital modules. To understand the operation of each digital module, it is necessary to have a basic knowledge of digital circuits and their logical function.

Number system: Number systems defines a set values used to represent quantity.

Types: 1. Binary, 2. Octal, 3. Decimal 4. Hexadecimal

Binary number system: A binary number is a number expressed in the base-2 numeral system.

Bits: Binary digit (0 and 1) are called bits.

MSB and LSB: In a binary number the left most bit is called MSB and the right most bit is called LSB.

Bit is the smallest unit of data.

- 1 nibble = 4 bits
- 1 byte = 8 bits
- 1 word = 16 bits = 2 bytes
- 1 double word = 32 bits = 4 bytes

Decimal to Binary Conversion: To convert decimal to any other base 'n', divide integer part by n and multiply fractional part by n.

OR,

2 | 13	
2 | 6	— 1
2 | 3	— 0
2 | 1	— 1 ↑ MSB
0	— 1

$(1101)_2$

Exp: ① $(13)_{10} \rightarrow$ ~~(2)~~ $(?)_2$

Divisor	Dividend	Remainder
2	13	1
2	6	1
2	3	0
2	1	1
	0	

↑ MSB (Most significant Bit)

$\therefore (13)_{10} = (1101)_2$ (Ans)

② $(25.625)_{10} \rightarrow (?)_2$

For integer part,

Divisor	Divided	Remainder
2	25	1
2	12	0
2	6	0
2	3	1
2	1	1
	0	

↑ MSB

$\therefore (25)_{10} = (11001)_2$

For the fractional part,

	integer	fraction
$.625 \times 2 = 1.25$	1	.25
$.25 \times 2 = 0.5$	0	.5
$.5 \times 2 = 1.0$	1	.0

↓ LSB

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

$\therefore (25.625) = (11001.101)_2$ (Ans)

Decimal to Octal :

① $(112)_{10} \rightarrow (?)_8$

Solution:

Divider	Dividend	Remainder
8	112	0
8	14	6
8	1	1
	0	

↑ MSD (Most significant digit)

$\therefore (112)_{10} = (160)_8$ (Ans)

② $(25.625)_{10} = (?)_8$

Solution: For integer part,

Divisor	Dividend	Remainder
8	25	1
8	3	3
	0	

↑ MSD

$\therefore (25)_{10} = (31)_8$

For fractional part,

$.625 \times 8 = 5.000$ ↓ LSD (Least significant part)

$\therefore (.625)_{10} = (.5)_8$

$\therefore (25.625)_{10} = (31.5)_8$ (Ans)

Decimal to Hexadecimal:

① $(25.625)_{10} = (?)_{16}$

Solution: For integer part,

Divisor	Dividend	Remainder	
16	25	9	↑ (MSB)
16	1	1	
	0		

$\therefore (25)_{10} = (19)_{16}$

For fractional part,

$.625 \times 16 = 10\ (A)$ ↓ (LSB)

$\therefore (25.625)_{10} = (19.A)_{16}$

(Ans)

Binary to Decimal:

① $(10101.11)_2 = (?)_{10}$

Solution: $(10101.11)_2 \rightarrow (1\times2^4) + (0\times2^3) + (1\times2^2) + (0\times2^1) + (1\times2^0) + (1\times2^{-1}) + (1\times2^{-2})$

$= 16 + 0 + 4 + 0 + 1 + \frac{1}{2} + \frac{1}{4}$

$= (21.75)_{10}$

(Ans)

Octal to Decimal:

① $(57.4)_8 \rightarrow (?)_{10}$

Solution: $(57.4)_8 \rightarrow (5\times8^1) + (7\times8^0) + (4\times8^{-1})$

$= 40 + 7 + \frac{4}{8}$

$= 47 + 0.5$

$= (47.5)_{10}$ (Ans)

Hexadecimal to Decimal:

① $(BAD)_{16} \rightarrow (?)_{10}$

Solution: $(BAD)_{16} = (B \times 16^2) + (A \times 16^1) + (D \times 16^0)$

$= (11 \times 16^2) + (10 \times 16^1) + (13 \times 16^0)$

$= 2816 + 160 + 13$

$= (2989)_{10}$

$\therefore (BAD)_{16} = (2989)_{10}$ Ans

Octal - Binary Conversion:

① $(37.45)_8 = (?)_2$

Solution: $(3)_8 = (011)_2$

$(7)_8 = (111)_2$

$(4)_8 = (100)_2$

$(5)_8 = (101)_2$

$\therefore (37.45)_8 = (011111.100101)_2$

BCD code

4	2	1	
0	1	1	3
1	1	1	7
1	0	0	4
1	0	1	5

② $(10110.11)_2 \rightarrow (?)_8$

Solution:

Hexadecimal binary conversion:

① $(259A)_{16} \rightarrow (?)_2$

Solution: $(2)_{16} = (0010)_2$

$(5)_{16} = (0101)_2$

$(9)_{16} = (1001)_2$

$(A)_{16} = (1010)_2$

$\therefore (259A)_{16} = (0010\,0101\,1001\,1010)_2$

(Ans)

② $(10001001.11)_2 = (?)_2$

Solution: $(10001001.11)_2 = (10001001.1100)_2$

$= (89.C)_{16}$

Hexadecimal-Octal conversion:

Hexadecimal — ~~Octal~~ Binary — Octal

① $(CAD)_{16} \rightarrow (?)_8$

Solution: $(C)_{16} \rightarrow (1100)_2$

$(A)_{16} \rightarrow (1010)_2$

$(D)_{16} \rightarrow (1101)_2$

$\therefore (CAD)_{16} = (110010101101)_2$

Now,

$(110010101101)_2 = (6255)_8$

$(CAD)_{16} = (6255)_8$

② $(652)_8 - (?)_{16}$

Solution: $(6)_8 = (110)_2$

$(5)_8 = (101)_2$

$(2)_8 = (010)_2$

$\therefore (652)_8 = (110101010)_2$

Now, $(110101010)_2 = (000110101010)_2 = (1AA)_{16}$

$\therefore (652)_8 = (1AA)_{16}$

Binary Addition:

	Sum	Carry
0 + 0	0	0
0 + 1	1	0
1 + 0	1	0
1 + 1	0	1

Example:
$$\begin{array}{r} 110 \\ +101 \\ \hline 1011 \end{array}$$

Binary Subtraction:

$$\begin{array}{r} 1\,1\,0\,1\,1 \\ -1\,0\,1\,1\,0 \\ \hline 0\,0\,1\,0\,1 \end{array} \qquad \begin{array}{r} 1\,1\,1\,0 \\ -\,1\,1\,1 \\ \hline 0\,1\,1\,1 \end{array}$$

Binary Multiplication:

$$\begin{array}{r} 1010 \\ \times\ 101 \\ \hline 1010 \\ 0000\,\times \\ 1010\,\times\times \\ \hline 110010 \end{array}$$

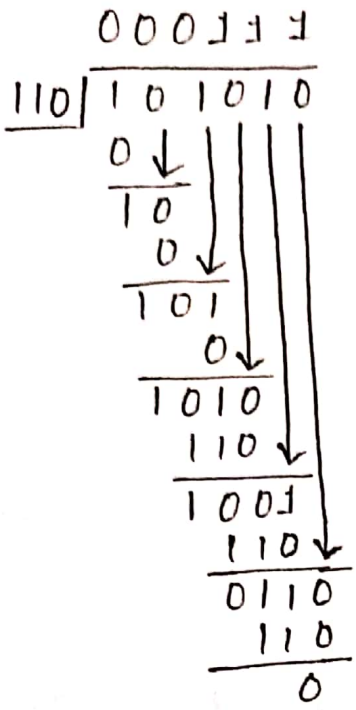
$0 \times 0 = 0$

$1 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 1 = 1$

Binary division:



$\therefore 101010 \div 110 = 111$ (Ans)

Complement: Complement refers to all the objects in one set that are not in another set.

Two types of complement:

① r's complement (Radix comp.)

② (r-1)'s complement (Diminished Radix comp.)

r's complement: Formula to find r's complement $r^n - N$

Exp: ① If $N = 5690$ then determine 10's complement

Solution: Here, $r = 10$

$N = 5690$

$n = 4$

$\therefore$ r's complement $= r^n - N$

$= 10^4 - 5690$

$= 4310$ (Ans)

② $N = 1101$. Find 2's complement.

Solution:

Here, $r = 2$

$N = 1101$

$n = 4$

$\therefore$ r's complement $= r^n - N$

$= 2^4 - 1101$

$= (16)_{10} - 1101$

$= 10000 - 1101$

$= 11$ (Ans)

(r-1)'s complement: Formula to find (r-1)'s complement: $r^n - N - 1$. In case of (r-1)'s complement, no borrow operation is involved.

r's complement $= r^n - N$

(r-1)'s complement $= r^n - N - 1$

or, (r-1)'s complement = r's complement -1

$\therefore$ (r-1)'s complement $+ 1$ = r's complement

Again,

	r's comp	(r-1)'s comp
when $r = 10$	10's	9's
when $r = 2$	2's	1's
when $r = 8$	8's	7's
when $r = 16$	16's	15's

Exp: ① Find 7's complement of $(5674)_8$

Solution: Here, $n-1=7$

$\therefore n=8$

$N=5674,\ n=4$

$\therefore$ 7's complement $= 8^4 - 5674 - 1$

$= (4096)_{10} - 5674 - 1$

$= 10000 - 5674 - 1$

$= 7777 - 5674$

$= 2103$ (Ans)

② Find 1's complement of $(1101)_2$

Solution: Here, $n-1=1$

$\therefore n=2$

$N=1101,\ n=4$

$\therefore$ 1's complement $= 2^4 - 1101 - 1$

$= (16)_{10} - 1101 - 1$

$= 10000 - 1101 - 1$

$= 1111 - 1101$

$= 10$ (Ans)

③ Find 1's complement of $(100111)_2$

Solution: Here, $n-1=1$

$n=2$

$N=100111,\ n=6$

$\therefore$ 1's complement $= n^n - N - 1$

$= 2^6 - 100111 - 1$

$= (64)_{10} - 100111 - 1$

$= 1000000 - 100111 - 1$

$= 111111 - 100111$

$= 11000$ (Ans)

④ Find 2's complement of $(1100111)_2$

Solution:

$R = 2,$

$N = 1100111$

$n = 7$

1's complement $= 2^7 - 1100111 - 1$

$= (128)_{10} - 1100111 - 1$

$= 10000000 - 1100111 - 1$

$= 1111111 - 1100111$

$= 11000$

$\therefore$ 2's complement $= 11000 + 1 = 11001.$

(Ans)

Data Representation: Data representation refers to the form in which data is stored, processed and transmitted. It can be done in various ways:-

1. Using unsigned magnitude
2. Using signed magnitude
3. Using 1's complement
4. Using 2's complement

Binary subtraction using 1's complement:-

Step 1: Convert number to be subtracted to its 1's complement form.

Step 2: Perform the addition.

Step 3: If the final carry is 1, then add it to the result obtained in step 2. If final carry is 0. result obtained in step 2 is negative and in the 1's complement form.

Examples:

① Perform $(1100)_2 - (0101)_2$

Solution: Let, A = 1100

B = 0101

1's complement of B = 1010 = -B

$\therefore A + (-B) = 1100 + 1010$

$= 10110$

Final carry bit is 1

$\therefore$ Final answer = 0110 + 1 = 0111

(Ans)

② Perform $(0101)_2 - (1100)_2$

Solution: Let, A = 0101

B = 1100

$\therefore -B = 0011$

$\therefore A + (-B) = 0101 + 0011 = 1000$

Final carry bit is 0

$\therefore$ Final answer = 0111.

(Ans)

Binary Subtraction using 2's complement :-

Step 1:- Find 2's complement of number to be Subtracted.

Step- 2:- Perform the addition.

Step 3:- If final carry is generated then the result is positive and in its true form. If final carry is not produced, then the result is negative and in its 2's complement form.

Examples:

① Perform $(1001)_2 - (0100)_2$ using 2's complement.

Solution: Let, A = 1001

B = 0100

1's comp. of B = 1011

2's comp. of B = 1011 + 1 = 1100

= −B

∴ A + (−B) = 1001 + 1100 = 10101

Final carry bit is 1.

∴ Final answer 0101 0

② Perform $(0110)_2 - (1011)_2$

Solution: Let, A = 0110

B = 1011

∴ 1's complement of B = 0100

∴ 2's complement of B = 0100 + 1 = 0101 = −B

∴ A + (−B) = 0110 + 0101

= 1011

Final carry is 0

That means, our result is in negative.

Binary Codes

Codes: It is a group of symbols which is used to represent data.

Classification of codes:

1. **Weighted codes:** Each position has fixed value. such as: Binary 8421, 2421.
2. **Non-weighted codes:** positions does not have any fixed value. Such as: XS-3, Gray.
3. **Reflective Codes:** It is self complementing code. Such as: 2421, XS-3.
4. **Sequential Codes:** Next data will increment by 1. Such as: binary, XS-3, 8421.
5. **Alphanumeric codes:** Alphabets + Numbers. Such as: ASCII
6. **Error Detecting and Correcting Codes:-** Such as: Hamming Code, Cyclic code

BCD code: BCD- Binary coded decimal code. In this code, each decimal digits are represented by 4 bits binary code.

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

BCD code for decimal digits

Conversion of BCD from Decimal / Decimal to BCD:

① $(144)_{10} \rightarrow (?)_{BCD}$

Solution: Here $(1)_{10} \equiv (0001)_{BCD}$

$(4)_{10} \equiv (0100)_{BCD}$

$\therefore (144)_{10} = (0001\ 0100\ 0100)_{BCD}$

Conversion of BCD to Decimal:-

① $(10100)_{BCD} \rightarrow (?)_{10}$

Solution: Here, $(10100)_{BCD} \equiv (0001\ 0100)_{BCD}$

$\equiv (14)_{10}$

Comparison of binary and BCD code:

$(15)_{10} \rightarrow (1111)_2 \rightarrow (0001\ 0101)_{BCD}$

$(28)_{10} \rightarrow (11100)_2 \rightarrow (0010\ 1000)_{BCD}$

$(1)_{10} \rightarrow (0001)_2$ $\quad$ $(2)_{10} \rightarrow (0010)_2$

$(5)_{10} \rightarrow (0101)_2$ $\quad$ $(8)_{10} \rightarrow (1000)_2$

$\therefore (15)_{10} = (0001\ 0101)_{BCD}$ $\quad$ $\therefore (28)_{10} = (0010\ 1000)_{BCD}$

Here, in BCD code, we need more bits for same information compared to Binary. So, BCD code is less efficient code compared to Binary code.

BCD Addition: If Nibble to Nibble carry is 1, then add 0110 to that Nibble. If Addition of Nibble with Nibble is greater than 9. then add 0110 to that Nibble.

(1) $(8)_{10} + (7)_{10}$

$(8)_{10} \rightarrow (1000)_2$

$(7)_{10} \rightarrow (0111)_2$

```
  1000
+ 0111
------
  1111
+ 0110
------
0001 0101
  1    5
```

(2) $(9)_{10} + (8)_{10}$

$(9)_{10} \rightarrow (1001)_2$

$(8)_{10} \rightarrow (1000)_2$

```
  1001
+ 1000
------
 10001
+ 0110
------
0001 0111
  1    7
```

BCD to Binary Conversion:

BCD → Decimal → Binary

1. $(0010\ \ 1001\ \ 0010)_{BCD} \equiv (292)_2$

$(292)_{10} \rightarrow (100100100)_2$

Binary to BCD Conversion:

1. $(11001100 1)_2 \rightarrow (?)_{BCD}$

Here, $(11001100 1)_2 \equiv (1\times2^8) + (1\times2^7) + (0\times2^6) + (0\times2^5) + (1\times2^4) + (1\times2^3) + (0\times2^2) + (0\times2^1) + (1\times2^0)$

$\equiv (409)_{10}$

$\equiv (0100\ \ 0000\ \ 1001)_{BCD}$

$\therefore (1100 11001)_2 \rightarrow (0100\ \ 0000\ \ 1001)_{BCD}$

2421 BCD Code:

⇒ It is a BCD code.

⇒ Sometimes it is referred as 2*-4-2-1 code.

Decimal → 2*-4-2-1:-

0 → 0000	5 → 0101
1 → 0001	6 → 0110
2 → 0010	7 → 0111
3 → 0011	8 → 1110
4 → 0100	9 → 1111

Decimal -to 2-4-2-1 :-

0 → 0000	5 → 1011
1 → 0001	6 → 1100
2 → 0010	7 → 1101
3 → 0011	8 → 1110
4 → 0100	9 → 1111

It is the actual 2-4-2-1 code that follows self complementary property.

Self complementary property means,

9 is complement of 0

8 is complement of 1

7 is complement of 2

6 is complement of 3

5 is complement of 4

BCD code examples :- 7421, 5421, 3321, $84\bar{2}\bar{1}$ and $74\bar{2}\bar{1}$

Excess-3 code:-

The excess-3 code is a non-weighted and self complementary BCD code used to represent the decimal numbers.

Decimal numbers → 8421 BCD code $\xrightarrow[0011]{\text{Add}}$ Excess-3

⊛ $(4)_{10} \rightarrow 0100 \rightarrow 0100 + 0011 \Rightarrow (0111)_{\text{Excess-3}}$

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

Based on bits, there is no well defined weightage. So, it is a unweighted code.

→ From the data table we can see,

9 is a complement of 0
8 is a complement of 1
7 is a complement of 2
6 is a complement of 3
5 is a complement of 4

That's why, we can say that Excess-3 code is a self complementary code.

Decimal to Excess-3 code conversion:-

1. $(36)_{10}$

Here, $(36)_{10} = (0011\ \ 0110)_{8421\ BCD}$

$$\begin{array}{r} 0011 \quad 0110 \\ +\ 0011 \quad 0011 \\ \hline (0110 \quad 1001)_{XS-3} \end{array}$$

(Ans)

Excess-3 Addition:

Step 1:- Convert given numbers into Excess 3

Step 2:- Add given Excess-3 numbers

Step 3:- which Nibble to Nibble carry

carry 1 → Add 0011

carry 0 → Subtract 0011

① $(3)_{10} + (6)_{10}$

$(3)_{10} \rightarrow 0011 \xrightarrow{\text{Add } 0011} 0110$

$(6)_{10} \rightarrow 0110 \xrightarrow{\text{Add } 0011} 1001$

Step-2

$$\begin{array}{r} 0110 \\ +1001 \\ \hline 1111 \\ -0011 \\ \hline 1100 \end{array}$$ (Ex-3).

Identification of self complementing code:

If code satisfy given condition

9 is complement of 0
8 is complement of 1
7 is complement of 2
6 is complement of 3
5 is complement of 4

The given code is complementary code.

⇒ If given code is $(w_3 - w_2 - w_1 - w_0)$ and, $w_3 + w_2 + w_1 + w_0 = 9$ then given code is self complementing code.

Examples: $\bar{7}421 \rightarrow 7+4+2+1 = 14 \neq 9$ ✗

$5921 \rightarrow 5+9+2+1 = 17 \neq 9$ ✗

$3321 \rightarrow 3+3+2+1 = 9$ ✓

$842\bar{1} \rightarrow 8+4-2-1 = 9$ ✓

$74\bar{2}\bar{1} \rightarrow 7+4-2-1 = 8 \neq 9$ ✗

Gray code:

- Code is named after Frank Gray.
- It is unweighted code.
- It is also referred as reflected binary code.
- It is unit distance code.
- It is cyclic code.

1 bit gray code	2 bit gray code	3 bit gray code	4 bit gray code
0	0 0	0 0 0	0 0 0 0
1	0 1	0 0 1	0 0 0 1
	1 1	0 1 1	0 0 1 1
	1 0	0 1 0	0 0 1 0
		1 1 0	0 1 1 0
		1 1 1	0 1 1 1
		1 0 1	0 1 0 1
		1 0 0	0 1 0 0
			1 1 0 0
			1 1 0 1
			1 1 1 1
			1 1 1 0
			1 0 1 0
			1 0 1 1
			1 0 0 1
			1 0 0 0

- As it is a unit distance code, switching time for gray code is less than any other code.

Application :- ① k-Map function reduction
② Error detection
③ Digital communication (cable TV)

Binary to Gray conversion:

Step 1: Take MSB as it is.

Step 2: X-OR MSB with next bit

Step 3: Repeat Step 2

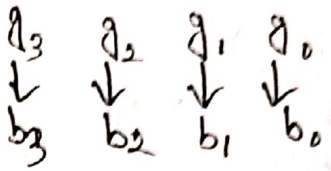
$b_3 - b_2 - b_1 - b_0 \longrightarrow g_3 - g_2 - g_1 - g_0$

$g_3 = b_3$

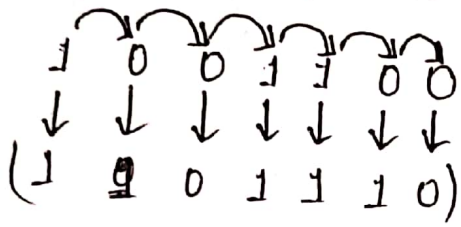
$g_2 = b_3 \oplus b_2$

$g_1 = b_2 \oplus b_1$

$g_0 = b_1 \oplus b_0$



① Convert $(1001100)_2$ to gray.



gray code

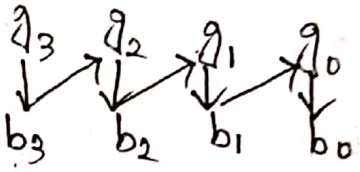
Gray to Binary Conversion:

Step 1:- Take MSB as it is

Step 2:- X-OR MSB to next bit of gray code

Step 3:- Repeat step 2

$g_3 - g_2 - g_1 - g_0 \longrightarrow b_3 - b_2 - b_1 - b_0$



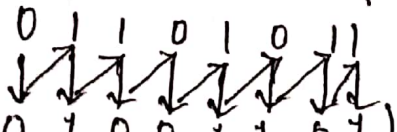
$b_3 = g_3$

$b_2 = b_3 \oplus g_2 = g_3 \oplus g_2$

$b_1 = b_2 \oplus g_1 = g_3 \oplus g_2 \oplus g_1$

$b_0 = b_1 \oplus g_0 = g_3 \oplus g_2 \oplus g_1 \oplus g_0$

① Convert 01101011 gray code into binary code.



$(0\ 1\ 0\ 0\ 1\ 1\ 0\ 1)_2$ Binary code.

Hamming Code:

→ It is given by RW hamming
→ It is used to detect and correct error
→ In Hamming code, we send data along with parity bits or Redundent bits.
→ It is represented by (n,k) code.

(n, k)
↓ → Message bits
Total bits

Parity bits, $p = n-k$

→ To identify the parity bits, it should satisfy given Condition.

$$2^p \geq p+k+1$$

So for $k=4$ message bits

$$2^p \geq p+4+1$$

$$\Rightarrow 2^p \geq p+5$$

For $p=1$: $2^1 \geq 6$ ✗

For $p=2$: $2^2 \geq 7$ ✗

For $p=3$: $2^3 = 8$ ✓

∴ $p=3$ for $k=4$ bits

$\Rightarrow n = p+k = 3+4 = 7$ bits

∴ This is $(n,k) = (7,4)$ code.

Positions of parity bits: positions of parity bits can be calculated by the formula 2^n.

Example: $P_1 = 2^0 = 1$

$P_2 = 2^1 = 2$

$P_4 = 2^2 = 4$

$P_8 = 2^3 = 8$

Value of parity bits:

For example, in a (7,4) code

D_7	D_6	D_5	P_4	D_3	P_2	P_1

For P_1, check 1 bit and skip 1 bit

For P_2, check 2 bits and skip 2 bits

For P_4, check 4 bits and skip 4 bits

$P_1 = D_3 \oplus D_5 \oplus D_7$

$P_2 = D_3 \oplus D_6 \oplus D_7$

$P_4 = D_5 \oplus D_6 \oplus D_7$

Generation of Hamming Code:

① 5 bit data 01101 is given, Represent given data in Hamming Code.

Here, $k = 5$ bits

Identify parity bits,

$\Rightarrow 2^p \geq p + k + 1$

$\Rightarrow 2^p \geq p + 6$

For, $p=1$ $\quad 2^1 \geq 7$ ✗

For, $p=2$ $\quad 2^2 \geq 8$ ✗

For, $p=3$ $\quad 2^3 \geq 9$ ✗

For, $p=4$ $\quad 2^4 \geq 10$ ✓

Positions of parity bits: positions of parity bits can be calculated by the formula 2^n.

Example: $P_1 = 2^0 = 1$

$P_2 = 2^1 = 2$

$P_4 = 2^2 = 4$

$P_8 = 2^3 = 8$

Value of parity bits:

For example, in a (7,4) code

D_7	D_6	D_5	P_4	D_3	P_2	P_1

For P_1, check 1 bit and skip 1 bit

For P_2, check 2 bits and skip 2 bits

For P_4, check 4 bits and skip 4 bits

$P_1 = D_3 \oplus D_5 \oplus D_7$

$P_2 = D_3 \oplus D_6 \oplus D_7$

$P_4 = D_5 \oplus D_6 \oplus D_7$

Generation of Hamming Code:

① 5 bit data 01101 is given, Represent given data in Hamming Code.

Here, $k = 5$ bits

Identify parity bits,

$\Rightarrow 2^p \geq p + k + 1$

$\Rightarrow 2^p \geq p + 6$

For, $p=1$ $\quad 2^1 \geq 7$ ✗

For, $p=2$ $\quad 2^2 \geq 8$ ✗

For, $p=3$ $\quad 2^3 \geq 9$ ✗

For, $p=4$ $\quad 2^4 \geq 10$ ✓

So, for $p = 4$, $k = 5$,

$n = p + k = 9$

∴ This is, $(n, k) = (9, 5)$ code.

Positions of parity bits –

$P_1 = 2^0 = 1$

$P_2 = 2^1 = 2$

$P_4 = 2^2 = 4$

$P_8 = 2^3 = 8$

0 D_9	P_8	1 D_7	1 D_6	0 D_5	P_4	1 D_3	P_2	P_1

Value of parity bits,

$P_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$

$P_2 = D_3 \oplus D_6 + D_7 = 1 \oplus 1 \oplus 1 = 1$

$P_4 = D_5 \oplus D_6 \oplus D_7 = 0 \oplus 1 \oplus 1 = 0$

$P_8 = D_9 = 0$

∴ The hamming code is

001100110

(Ans)

Hamming Code Error detection and Correction:

① If received hamming code is 1110101 with even parity then detect and correct error.

Solution:

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	1	1	0	1	0	1

$P_1 = D_3 \oplus D_5 \oplus D_7 \Rightarrow 1 = 1 \oplus 1 \oplus 1 \rightarrow P_1 = 0$ (True that's why 0)

$P_2 = D_3 \oplus D_6 \oplus D_7 \Rightarrow 0 = 1 \oplus 1 \oplus 1 \Rightarrow P_2 = 1$ (False)

$P_4 = D_7 \oplus D_6 \oplus D_5 \rightarrow 0 = 1 \oplus 1 \oplus 1 \Rightarrow P_4 = 1$ (False)

$P_4P_2P_1 = 110 = 6$th bit error

$E =$

0	1	0	0	0	0	0

$R =$

1	1	1	0	1	0	1

Connected data = $R \oplus E$

= 1010101

(Ans)

Biquinary code: It's a seven bit weighted code.

The weights of the bits are 5, 0, 4, 3, 2, 1, 0.

Decimal		Biquinary (5043210)
0	→	0100001
1	→	0100010
2	→	0100100
3	→	0101000
4	→	0110000
5	→	1000001
6	→	1000010
7	→	1000100
8	→	1001000
9	→	1010000

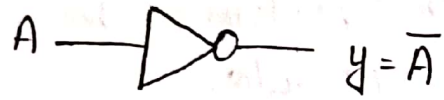
Logic gates

① Basic Gates: AND, OR, NOT

② Universal Gates: A universal gate is a gate which can implement any boolean function without need to use any other gate type. NAND and NOR gates are universal gates.

③ Arithmetic gates – X-OR, X-NOR

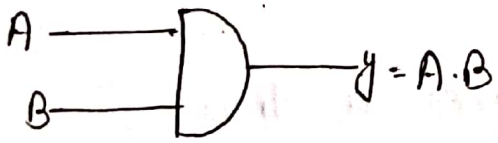
NOT Gate:



Truth table

A	Ā
0	1
1	0

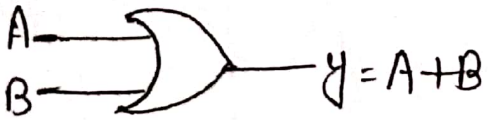
AND Gate: An And gate is an electrical circuit that combines two signals so that the output is ON if both signals are present.



Truth table -

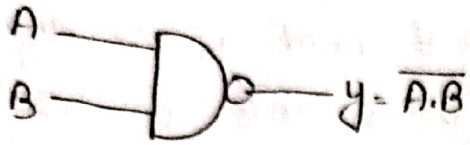
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate: An OR gate is an electrical ~~gate~~ circuit that performs logical addition.



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

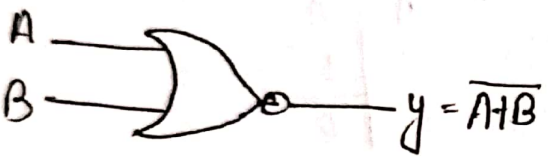
NAND Gate: The NAND gate is an universal gate that performs the operation of AND gate followed by the operation of NOT gate.



A	B	y	$\bar{y}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NOR gate:

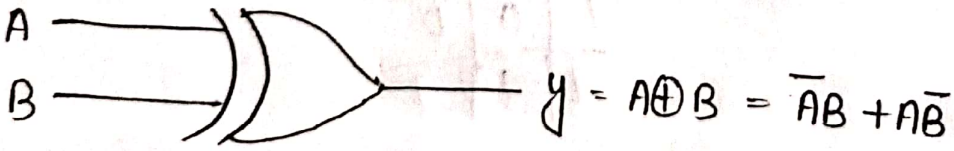
A NOR gate is an universal gate that performs the operation of OR gate followed by NOT gate.



A	B	y	$\bar{y}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

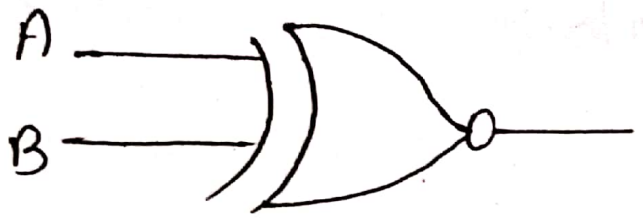
X-OR

XOR Gate is a digital gate that gives a true output when the number of true input is odd.



A	B	$y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

X-NOR: The X-NOR is a combination of XOR gate followed by an inverter.



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

$= \overline{AB}$

A	B	$A \oplus B$	$\overline{A \oplus B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1