MODULE-1
NUMBER SYSTEMS AND CODES

1). Convert binary to decimal.

(a). 110110_2

$= 1\times2^0 + 0\times2^1 + 1\times2^2 + 1\times2^3 + 0\times2^4 + 1\times2^5 + 1\times2^6$

$= 1 + 4 + 8 + 32 + 64$

$= 109_{10}$

(b). 0.0011_2

$= 1\times2^{-4} + 1\times2^{-3} + 0\times2^{-2} + 0\times2^{-1} + 0\times2^0$

$= 2^{-4} + 2^{-3}$

$= 0.0625 + 0.125$

$= 0.1875$

(c). 10111101.011_2

$= 1\times2^{-3} + 1\times2^{-2} + 0\times2^{-1} + 1\times2^0 + 0\times2^1 + 1\times2^2$
$+ 1\times2^3 + 1\times2^4 + 1\times2^5 + 0\times2^6 + 1\times2^7$

$= 2^{-3} + 2^{-2} + 1 + 4 + 8 + 16 + 32 + 128$

$= 189.375_{10}$

2). Convert decimal to binary

(i). 29_{10}

2	29
2	14 - 1
2	7 - 0
2	3 - 1
	1 - 1

HSB → Host significant byte.

LSB → Lowest significant byte

$29_{10} = 11101_2$

LSB

HSB

(ii). 57_{10}

2	57
2	28 - 1
2	14 - 0
2	7 - 0
2	3 - 1
	1 - 1

$57_{10} = 111001_2$

(iii). 256_{10}

2	256
2	128 - 0
2	64 - 0
2	32 - 0
2	16 - 0
2	8 - 0
2	4 - 0
2	2 - 0

$256_{10} = 100000000_2$

iv). 0.065_{10}

$2 \times 0.065 = 0.13$
$2 \times 0.13 = 0.26$
$2 \times 0.26 = 0.52$
$2 \times 0.52 = 1.04$
$2 \times 0.04 = 0.08$

$0.065_{10} = 0.00010_2$

(v). 0.625_{10}

$2 \times 0.625 = 1.25$
$2 \times 0.25 = 0.5$
$2 \times 0.5 = 1$

$0.625_{10} = 0.101_2$

(vi). 0.726_{10}

$2 \times 0.726 = 1.452$
$2 \times 0.452 = 0.904$
$2 \times 0.904 = 1.808$
$2 \times 0.808 = 1.616$
$2 \times 0.616 = 1.232$
$2 \times 0.232 = 0.464$

$0.726_{10} = 0.101111_2$

BINARY ADDITION

(1). 1101_2 +
1111_2
11100_2

Verification:
13 +
15
28

$0_{10} \rightarrow 0_2$
$1_{10} \rightarrow 1_2$
$2_{10} \rightarrow 10_2$
$3_{10} \rightarrow 11_2$
$4_{10} \rightarrow 100_2$

(2). 1101101_2 +
1111011_2
11101000_2

BINARY SUBTRACTION

(1). $1^{0}1^{1}0^{10}1_2$ —
10111_2
00011_2

NEGATIVE Nos.

1) → We should take one's compliment · (1's comp)

Eg: 11100101_2

$\Rightarrow 00011010_2$

2). 2's compliment

Take 1's compliment + 1

Eg: 00011011_2

$11100100_2 +$

$\quad\quad\quad\quad\quad\quad 1$

11100101_2

Eg: 1) -25_{10}

2	25
2	12 - 1
2	6 - 0
2	3 - 0
	1 - 1

00011001_2

* we take 8 bits for -ve No.

MSB

-ve ⇒ 1

+ve ⇒ 0

1's compliment

11100110_2

$\Rightarrow 1\ 1\ 1\ 0\ 0\ 1\ 1\ 0_2$

$-2^7 + 2^6 + 2^5 + 2^2 + 2^1$

$-128 + 64 + 32 + 4 + 2$

$= -26 + 1 = -25$

2's compliment

11100111_2

$1\ 1\ 1\ 0\ 0\ 1\ 1\ 1_2$

$= -128 + 64 + 32 + 4 + 2 + 1$

$= -25_{10}$

For 1's complement, when converted to decimal, (weighted sum)
we need to add one.

whereas for 2's complement, we need not add 1.

2). -39

2	39
2	19 – 1
2	9 – 1
2	4 – 1
2	2 – 0
	1 – 0

$\Rightarrow 00100111_2$

1's complement

11011000_2

$= -2^7 2^6 \; 2^4 \; 2^3$

$: -40 + 1 : -39$

2's complement

~~00101000~~

11011001_2

$= -2^7 + 2^6 + 2^4 + 2^3 + 2^0$

$: -39$

Q. Find decimal value of the signed binary no in 1's compliment

Need not add 1 for +ve Nos.

(a) 00010111_2

$= 2^0 + 2^1 + 2^2 + 2^4$

$= \underline{23_{10}}$

(b) 11101000_2

$= 2^3 + 2^5 + 2^6 - 2^7 = -24+1$

$= \underline{-23_{10}}$

Q. Find decimal value of signed binary no in 2's compliment.

(a) 01010110_2

$= 2^1 + 2^2 + 2^4 + 2^6$

$= \underline{86}$

(b) 10101010_2

$= 2^1 + 2^3 + 2^5 - 2^7$

$= \underline{-86}$

Q. Add the signed nos:

(1) (+ve) 0 0 0 0 0 1 1 1$_2$
(+ve) 0 0 0 0 0 1 0 0$_2$
0 0 0 0 1 0 1 1$_2$

$\Rightarrow$ 7 +
4
11

(2) (+ve) 0 0 0 0 1 1 1 1$_2$
(-ve) 1 1 1 1 1 0 1 0$_2$
1 0 0 0 0 1 0 0 1$_2$

Discard this 1

$\Rightarrow$ 15
−6
9

(3) (+ve) 0 0 0 1 0 0 0 0
(-ve) 1 1 1 0 1 0 0 0$_2$
1 1 1 1 1 0 0 0$_2$

discard

16
−24
−8

2. Add the signed numbers:

(1). 11111011_2 + 11110111_2

$$\begin{array}{r} 11111011 \\ +\ 11110111 \\ \hline 1\,11110010 \end{array}$$

-5 + -9

-14

2 Subtract the signed nos:

When you have a -ve sign with a binary no take 2's complement & add.

(1). $00001000_2 - 00000011_2$

↳ take 2's complement & add.

$$\begin{array}{r} 00001000 \\ +\ 11111101 \\ \hline 1\,00000101 \end{array}$$

$\Rightarrow 8 + (-3)$

5

(2). $00001100_2 - 11110111_2$

$12 - (-9) = 21$

$$\begin{array}{r} 00001100 \\ +\ 00001001 \\ \hline 00010101 \end{array}$$

(3) $11100111_2 - 00010011_2$ ⇒ $-25 - 19$

$: -25 + (-19)$

$: -44$

11100111_2

11101101_2 +

$\overline{\overline{1}11010100_2}$

HEXADECIMAL

DECIMAL	BINARY	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Q. Convert binary to hexadecimal

(1) $1100\ 1010\ 0101\ 0111_2$

Group into 4 each.

$1100\ 1010\ 0101\ 0111_2$

$C\ \ A\ \ 5\ \ 7_{16}$ $\Rightarrow$ $CA57_{16}$

(2) $0001\ 1111\ 0001\ 0110\ 1001_2$

$1\ \ F\ \ 1\ \ 6\ \ 9_{16}$ $\Rightarrow$ $1F169_{16}$

Q. Hexa to binary:

(1) $10AA_{16}$

$1\ \ 0\ \ A\ \ A_{16}$

$0001\ 0000\ 1010\ 1010_2$

(2) $4CFA_{16}$

$0100\ 1100\ 1111\ 1010_2$

Q. Hexa to decimal

① Weighted Sum technique

(a). $1C_{16}$

$1 \times 16^1 + 12 \times 16^0 = 28_{10}$

② Hexa → Binary → decimal

(b). $2F_{16}$

$00101111_2 \longrightarrow 1\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^3 + 1\times2^5 = 47_{10}$

Q. Convert Decimal to hexa:

(a). $16 \mid 65_{10}$

$4 - 1 \Rightarrow 41_{16}$

(b). 860_{10}

$16 \mid 860_{10}$

$16 \mid 53 - 12 \Rightarrow 35C_{16}$

$3 \quad -5$

$16 \mid 860$ (53) — 80, 60, 48, 12

$16 \mid 53$ (3) — 48, 5

OCTADECIMAL

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

Q. Convert binary to Octal

$11,101,101_2$

$3\ 5\ 5_8$

Q. Convert octal to binary

203_8

010000011_2

Q. Convert octal to decimal

127_8

$\Rightarrow 7 \times 8^0 = 7$

$2 \times 8^1 = 16$

$1 \times 8^2 = 64$

87_{10}

Q. Convert decimal to Octal

(1). 256_{10}

$8 \mid 256$

$8 \mid 32 - 0$

$4 - 0$

$= 400_8$

ASCII → American Standard Codes for Information Index

① Weighted coding

② Non-weighted coding → Ex-3 code
→ Gray code

(a). Ex-3 code:

Eg: 1 + 4 = 5

1 ⇒ 0001_2 +
(add 3) 0011_2
0100_2

4 ⇒ 0100_2 +
(add 3) 0011_2
0111_2

Now add, 0100_2 +
0111_2
1000 ⇒ 8
(excess 3).

BINARY TO GRAY

① 1 0 (add) $= 11_2$

1st no. write as it is

② 1 0 1 1 $= 1110_2$ → ignore the carry 1.

③ $1101_2 = 1011_2$

GRAY TO BINARY

Eg: 1 0 1 1

↓ ↑ (add)

1101_2

Write 1st no. as it is

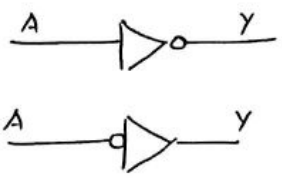
LOGIC GATES

The term gate is used to describe a circuit that performs logic operation

NOT

The inverter or NOT performs the operation called invasion or complementation. The inverter changes one logic level to the opposite level. In terms of bits it changes bit '1' to '0' & '0' to '1'

Symbol.



Truth Table

A	Y
0	1
1	0

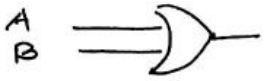
The negative indicater is a bubble (o) that indicates invasion or complementation when it appears on the i/p or o/p of any logic element. Generally inputs are on the left of the logic symbol & the o/p is on the right

0 ⇒ Active low

1 ⇒ Active high.

In boolean algebra, a variable is represented by a letter. The complement of a variable is designated by a bar over the letter. A variable can take on a value of either 1 or 0. Bodean algebra uses variables & operaters to describe a logic circuit.

The logic operation of NOT gate can be expressed as by taking A as input variable & Y as o/p variable $\boxed{Y = \bar{A}}$



TRUTH TABLE

INPUT A	INPUT B	o/p Y
0	0	0
0	1	1
1	0	1
1	1	1

If A & B are two o/p variables and Y is the o/p variable, the logical OR fn. can be expressed by a '+' b/w two variables.

$$Y = A + B$$

NAND GATE

It is a universal gate ie, NAND gates can be used in combination to perform AND, OR & NOT operations. The term NAND is a contraction of NOT-AND & implies an AND function with complimented output.

SYMBOL

The NAND gate produces a low o/p only when all the inputs are high when any of the i/p is low o/p is high.

Truth Table

Input A	Input B	Output
0	0	1
0	1	1
1	0	1
1	1	0

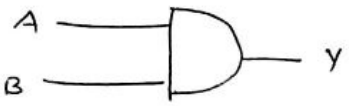
The boolean expression for NAND gate is

$$Y = \overline{AB}$$

AND GATE

An AND gate produces a high o/p only when all the inputs are high. When any one of the i/p is low o/p is low. If A & B are the two inputs then Y is the o/p variable. Its truth table is shown below. It performs logical multiplication

SYMBOL



Truth Table

i/p		o/p
A	B	
0	0	0
0	1	0
1	0	0
1	1	1

The logic expression of AND gate using two variables is represented mathematically either by placing a dot (·) between the two variables as A·B or by simply writing the adjacent letters without the dot as AB.

Thus the Boolean expression for AND gate is

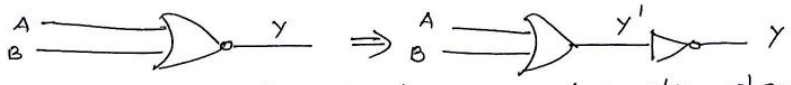
$$Y = AB.$$

OR GATE

An OR gate can have two or more inputs and performs logical addition. An OR gate produces a high on the o/p when any of the i/p is high. The o/p is low only when all the i/ps are low. ∴ an OR gates determine when one or more inputs are high produces a high on the o/p.

NOR GATE

It is also used as universal gate because it can be used in combination to perform AND, OR & NOT operation. The term NOR is a contraction NOT-OR & implies an OR fn. with an inverted o/p.



A NOR gate produces a low o/p when any of the inputs is high & it produces a high on the o/p when all the i/ps are low. Thus the truth table is

Truth table

Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

The boolean expression for NOR gate is

$Y = \overline{A+B}$

EXCLUSIVE OR (EXOR)

The Ex-OR gate has only two inputs. The o/p Ex OR gate is high only when the two inputs are at opposite logic levels.

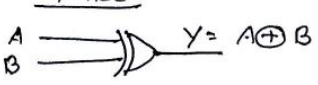
Truth Table

A	B	o/p
0	0	0
0	1	1
1	0	1
1	1	0

Boolean expression for Ex-OR gate is

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

SYMBOL



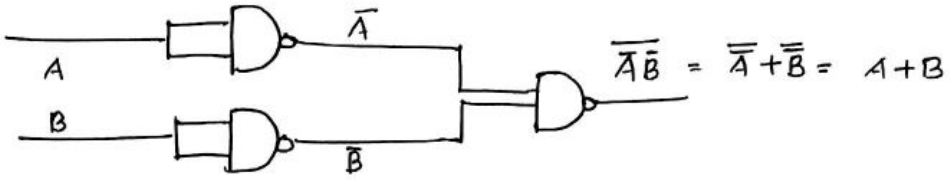
REALISATION OF AND, OR, NOT, AND EX-OR GATES USING NAND GATE

AND

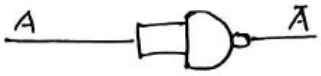
$Y = AB$

OR

$Y = A + B$

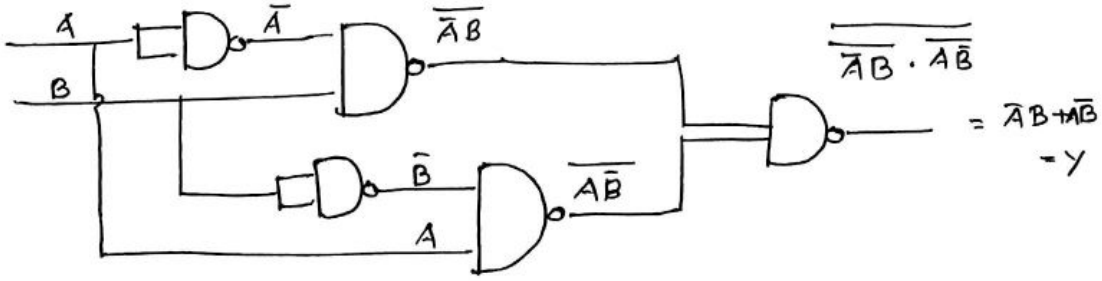


NOT



EX-OR

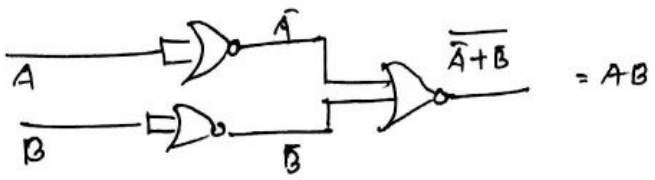
$Y = \bar{A}B + A\bar{B}$



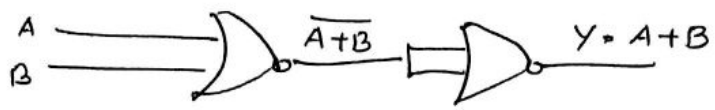
REALIZATION OF AND, OR, NOT & EX-OR GATES USING NOR GATE

AND

$Y = AB$

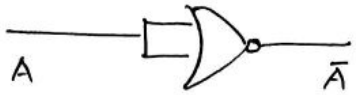


OR

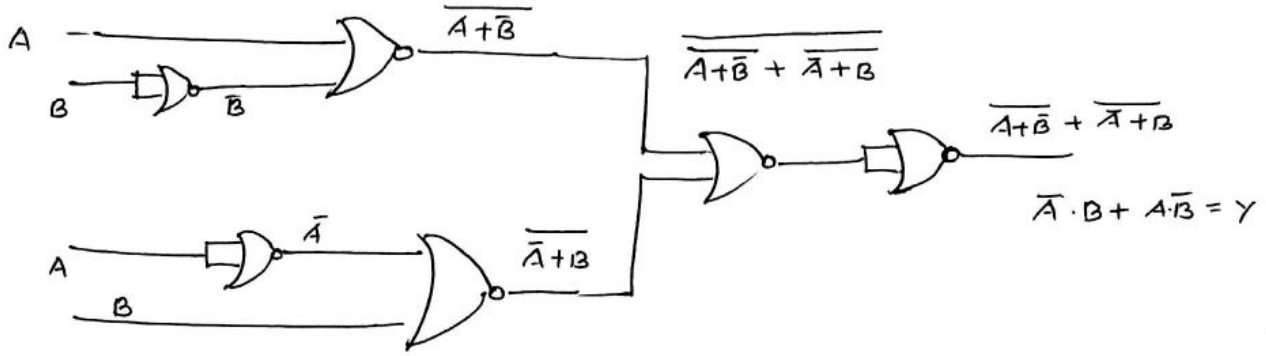


NOT

$Y = \bar{A}$



EX-OR GATE $Y = \bar{A}B + A\bar{B}$



SIMPLIFICATION OF BOOLEAN EXPRESSIONS

RULE 1: Let 'a' be a variable then

$a + \bar{a} = 1$ (sum of a variable & its complement = 1)

$a \cdot \bar{a} = 0$ (product " " " = 0)

$\bar{\bar{a}} = a$ (double complement)

a	$\bar{a}$	$a + \bar{a}$	$a \cdot \bar{a}$	$\bar{\bar{a}}$
0	1	1	0	0
1	0	1	0	1

RULE 2 Let 'a' be a variable then

$a + 1 = 1$ (any variable + 1 = 1)

$a + 0 = a$ (any variable + 0 = 0)

$a \cdot 1 = a$

$a \cdot 0 = 0$

Proof

a	a+1	a+0	a·1	a·0
0	1	0	0	0
1	1	1	1	0

E3 Absorption law: If a & b are two variables

then $a + ab = a$ or $a(1+b) = a \cdot 1 = a$

Here the variable 'b' is absorbed by variable 'a'

a	b	ab	a+b
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

By $a + \bar{a}b = a + b$

$a(\bar{a} + b) = ab$

$\bar{a} + ab = \bar{a} + b$

$\bar{a}(a + b) = \bar{a}b$

Proof: $a + \bar{a}b = a + ab + \bar{a}b = a + (a+\bar{a})b = a + b$

$a(\bar{a} + b) = a\bar{a} + ab = ab + 1 = ab$

$\bar{a}(a + b) = \bar{a}a + \bar{a}b = 1 + \bar{a}b = \bar{a}b$

RULE 4: Idempotency: Let 'a' be a variable then

$a + a = a$ & $a \cdot a = a$

a	a+a	a·a
0	0	0
1	1	1

RULE 5: COMMUTATIVE LAW: If a & b are two variables then

$$a+b = b+a \quad \& \quad a \cdot b = b \cdot a$$

RULE 6: Associative law: If a, b & c are three variables then

$$a+(b+c) = (a+b)+c \quad \& \quad a \cdot (b \cdot c) = ((a \cdot b) \cdot c)$$

RULE 7: DISTRIBUTIVE LAW If a, b & c are three variables then,

$$a(b+c) = ab + ac \quad \&$$

$$a + bc = (a+b)(a+c)$$

↳ special property of boolean algebra

RULE 8: DEMORGAN'S LAW

If a & b are two variables then

$$\overline{a+b} = \bar{a} \cdot \bar{b} \quad \&$$

$$\overline{ab} = \bar{a} + \bar{b}$$

RULE 9 Consensus theorem: If a, b & c are three variable then

$$ab + bc + ca = ab + c\bar{a}$$

$$ab + bc + c\bar{a} = ab + (a+\bar{a})bc + \bar{a}c$$

$$= ab + abc + \bar{a}bc + \bar{a}c$$

$$= abc(1+c) + \bar{a}c(1+b)$$

$$= ab + \bar{a}c$$

REPRESENTATION OF BOOLEAN EXPRESSION

Boolean expression can be represented in two forms

1. Sum of Products (SOP) form
2. Product of sums (POS) form

SOP FORM

This form is also called Disjunctive Normal form (DNF)

eg: $f(A, B, C) = \bar{A}B + \bar{B}C$

Standard SOP form

This form is also called Disjunctive canonical form (DCF). It is also called the expanded sum of product form or canonical sum of product form. In this form, the fn. is the sum of a number of product terms where each product term contains all the variables of the fn. either in complement or uncomplemented form. This can be derived from the truth table by finding the sum of all the terms that correspond to those combination for which 'f' assumes the value 1, or It can be obtained from the SOP form as follows.

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

A product term which contains all the variables of the fn either in complemented or uncomplemented form is called a minterm. A minterm assumes the value 1 only for one combination of the variables.

An n- variable fn can have in all 2^n minterms. The sum of the minterm whose value = 1 is the std. SOP form of the fn.

The minterms are often denoted as m_0, m_1, m_2 where the suffixes are the decimal codes of the combinations. for a 3- variable fn $m_0 = \bar{A}\bar{B}\bar{C}$ $m_1 = \bar{A}\bar{B}C$ $m_2 = \bar{A}B\bar{C}$ . . . $m_7 = ABC$. Another form of representing fn in std SOP form is by showing the sum of minterms for which the fn equal 1.

$$f(A, B, C) = m_1 + m_2 + m_3 + m_4$$

or by listing the decimal codes of the minterms for which $f = 1$

$$f(A B C) = \Sigma m(1, 2, 3, 5)$$

where Σm represents the sum of all the minterms whose decimal codes are given in the parantheses

CONVERT THE FOLLOWING SOP FORM INTO STD SOP FORM

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

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

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

$= 01 + 10 + 00$

$= m_0 + m_1 + m_2$

$= \Sigma m(0, 1, 2)$

2. $AB + \bar{A}\bar{B} + Ac + \bar{A}\bar{C}$

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

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

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

$= 111 + 110 + 001 + 101 + 010 + 000$

$= \Sigma(7, 6, 1, 5, 2, 0)$

$= \Sigma(0, 1, 2, 5, 6, 7)$.

SIMPLIFICATION OF BOOLEAN EXPRESSION BY USING BOOLEAN ALGEBRA

A simplified Boolean expression uses the fewest gates possible to implement a given expression.

1. Using Boolean algebra technique, simplify the following expression

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

$Y = AB + AB + AC + BB + BC$

$= AB + AC + B + BC$

$= AB + AC + B$

$= AB + B + AC = \underline{B + AC}$

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

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

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

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

3. $Y = [A\bar{B}(C+BD) + \bar{A}\bar{B}]C$

$= [A\bar{B}C + A\bar{B}BD + \bar{A}\bar{B}]C$

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

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

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

$[AB(C + \bar{B}\bar{D}) + \bar{A}\bar{B}]\,CD$

$\Rightarrow [ABC + AB\bar{B}\bar{D} + \bar{A}\bar{B}]\,CD \Rightarrow [ABC + AB(\bar{B}+\bar{D}) + \bar{A}+\bar{B}]\,CD$

$\Rightarrow ABCCD + AB\bar{B}CD + \bar{A}BC\bar{D}D + \bar{A}CD + \bar{B}CD$

$\Rightarrow ABCD + 0 + 0 + \bar{A}CD + \bar{B}CD$

$\Rightarrow ABCD + \bar{A}CD + \bar{B}CD$

$\Rightarrow CD[AB + \bar{A} + B]$

$\Rightarrow CD[A + B + \bar{B}] = CD[\bar{A} + 1] = \underline{CD}$

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

$\Rightarrow (\bar{A}+A)BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + A\bar{B}C$

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

$\Rightarrow BC + A\bar{B} + \bar{A}\bar{B}\bar{C}$

$\Rightarrow BC + \bar{B}(A + \bar{A}\bar{C})$

$\Rightarrow BC + \bar{B}(A + \bar{C})$

$\Rightarrow BC + A\bar{B} + \bar{B}\bar{C}$

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

$\Rightarrow AB\bar{C} + \bar{A}\bar{B}(C+\bar{C}) + \bar{A}BC$

$\Rightarrow AB\bar{C} + \bar{A}\bar{B} + \bar{A}BC$

$\Rightarrow AB\bar{C} + \bar{A}(B + BC)$

$\Rightarrow AB\bar{C} + \bar{A}(\bar{B} + C)$

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

7. $\overline{AB + AC} + \bar{A}\bar{B}C$

$\Rightarrow \quad \bar{A}\bar{B} \cdot \bar{A}\bar{C} + \bar{A}\bar{B}C$

$\Rightarrow \quad A \cdot \bar{A} + \bar{A}\bar{C} + \bar{B}\bar{A} + \bar{B}\bar{C} + \bar{A}\bar{B}C$

$\Rightarrow \quad \bar{A} + \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{B}C$

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

$\Rightarrow \quad \bar{A} + \bar{A}\bar{B} + \bar{B}\bar{C}$

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

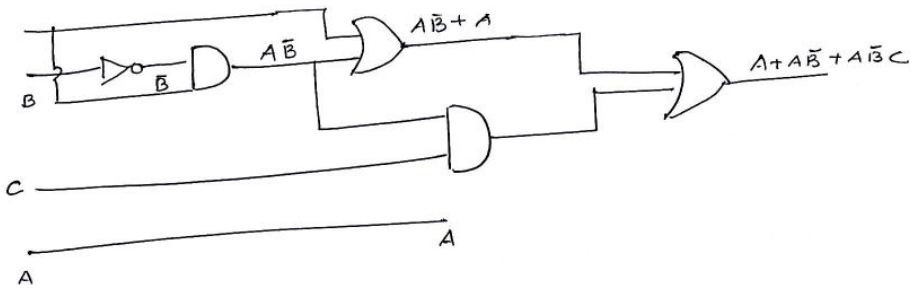
$= \quad \underline{\bar{A} + \bar{B}\bar{C}}$

Q. Simplify the following boolean expression and implement each expression by logic gates. Then implement the simplified expression & compare the no. of gates.

a) $A + A\bar{B} + A\bar{B}C$

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

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



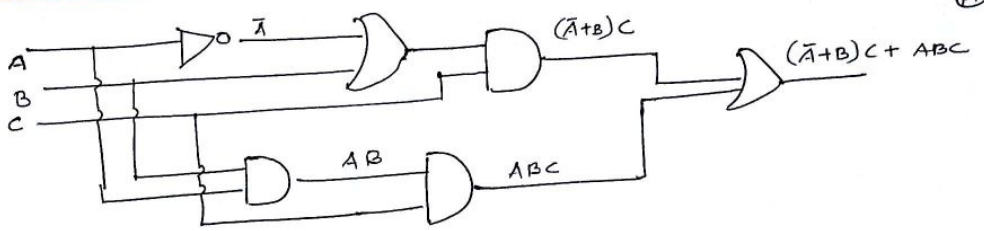
b) $(\bar{A}+B)C + ABC$

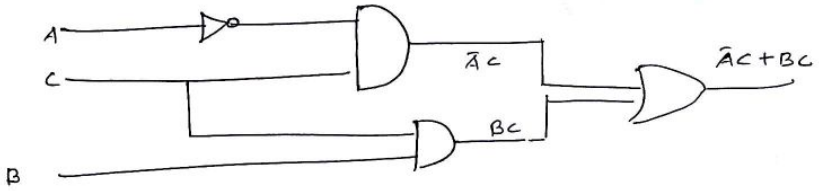
$= \quad \bar{A}C + BC + ABC$

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

$= \quad \bar{A}C + BC$

$= \quad \underline{\bar{A}C + BC}$





PRODUCT OF SUMS FORM

This form is also called conjunctive canonical form It is also called Expanded product of sums form or Canonical product of sums form. This is derived by considering the combinations for which $f = 0$. Each term is a sum of all the variables. A variable appears in uncomplemented form if it has a value of 0 in the combination & appears in complemented form if it has a value of 1 in the combination. The sum term which contains each of the n variables in either complemented or uncomplemented form is called a maxterm. A maxterm assumes the value 0 only for one combination of the variables. For all other combinations it will be 1. There will be all the most 2n maxterms. The product of maxterms corresponding to the rows for which $f=0$ is the standard or canonical product of sums form of the function.

Q. Find the canonical POS form for the following expression

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

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

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

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

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

$= \Sigma m(3, 2, 0)$

$= \Sigma m(0, 2, 3)$

$= \Pi M(1, 4, 5, 6, 7)$

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

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

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

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

$= \Sigma m(3, 1, 6, 4)$

$= \Sigma m(1, 3, 4, 6)$

$= \Pi M(0, 2, 5, 7)$

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

3. $f(A, B, CD) = AC$

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

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

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

$= \Sigma m(15, 14, 11, 10)$

$= \Sigma m(10, 11, 14, 15)$

$= \Pi M(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13)$

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

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

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

4. $F(A,B,C,D) = AB + \bar{A}BC + \bar{C}D$

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

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

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

$\Sigma m(15, 14, 13, 12, 7, 6, 9, 5, 1)$

$\Sigma m(1, 5, 6, 7, 9, 12, 13, 14, 15)$

$\pi M(0, 2, 3, 4, 8, 10, 11)$

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

---

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

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

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

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

$= \Sigma m(0, 1, 2, 5, 6, 7)$

$= \pi M(3, 4)$

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

K- MAP

The Karnaugh map is a systematic method of simplifying the Boolean expressions. The K-map is a chart or a group composed of an arrangement of adjacent cells, each representing a particular combination of variables in sum or product form.

TWO VARIABLE K- MAP

A two variable expression can have $2^2 = 4$ possible combinations. Each of these combinations are $\bar{A}\bar{B}$, $\bar{A}B$, $A\bar{B}$ and AB (in the SOP form) is called a minterm. Instead of representing the minterms in terms of the input variable using short hand notation the minterms may be represented in terms of their decimal designation m_0 — for $\bar{A}\bar{B}$ m_1 for $\bar{A}B$ m_2 for $A\bar{B}$ & m_3 for AB.

MAPPING OF SOP EXPRESSIONS

A two variabe 'k' map has $2^2 = 4$ squares. These squares are called cells. Each square on the K map represents a unique minterm. A '1' placed in any square indicates that the corresponding minterm is included in the output expression & a '0' or no entry in a square indicates that the corresponding minterms does not appear in the expression for input.

MAP FORMAT

a) Two variable

If there are 'n' variables 2^n combinations are there and 2^n cells are there.

A \ B	$\bar{B}$	B
$\bar{A}$	m_0	m_1
A	m_2	m_3

m_0, m_1, m_2 & m_3 are called Min terms

b) Three variable

consider three variables A B & C

AB	$\bar{C}$	C
$\bar{A}\bar{B}$	m_0	m_1
$\bar{A}B$	m_2	m_3
AB	m_6	m_7
$A\bar{B}$	m_4	m_5

3 variable - 3rd row change.

c) Four variables

consider 4 variables A, B, C & D

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	m_0	m_1	m_3	m_2
$\bar{A}B$	m_4	m_5	m_7	m_6
AB	m_{12}	m_{13}	m_{15}	m_{14}
$A\bar{B}$	m_8	m_9	m_{11}	m_{10}

PLOTTING THE BOOLEAN EXPRESSION

Once the Boolean expression is in sum of the product form. Plot each on the K-map by placing a 1 in each cell corresponding to the terms in SOP expression

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

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

(ii) $\bar{A}BC + AB\bar{C} + ABC$

	$\bar{C}$	C
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	0	1
AB	1	1
$A\bar{B}$	0	0

(iii) $\bar{A}\bar{B}CD + \bar{A}BCD + A\bar{B}C\bar{D} + A\bar{B}CD$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	1	0
$\bar{A}B$	0	0	1	0
AB	0	0	0	0
$A\bar{B}$	0	0	1	1

GROUPING OF CELLS.

Group the 1s that are in adjacent cell. Adjacent cells are cells that differ by only a single variable.

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

Adjacent cells of $\bar{A}\bar{B}$ are $\bar{A}B$ & $A\bar{B}$

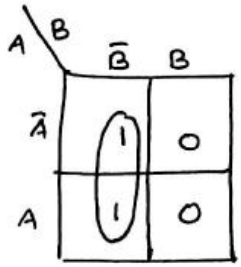
The 1's in adjacent cells must be combined in groups of 1, 2, 4, 8, 16 & so on.

SIMPLIFICATION

Each group of 1 composed of all variable that appear in one form with in group. The variables that appear both complimentated & uncomplimented are eliminate The final simplified expression is formed by summi the part terms of all groups.

2 Variable K-map

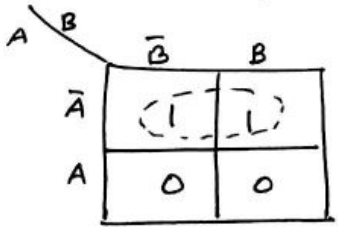
1. $\bar{A}\bar{B} + A\bar{B}$ $\quad S = \Sigma_1(m_0, m_2)$



A & $\bar{A}$ are eliminated

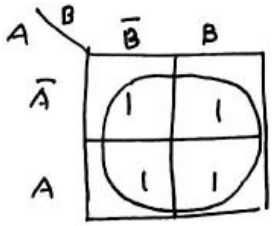
$\bar{A}\bar{B} + A\bar{B} = \bar{B}$

2. $\bar{A}\bar{B} + \bar{A}B$, $S = \Sigma_1(m_0, m_1)$



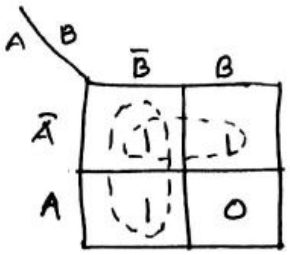
$\bar{A}\bar{B} + \bar{A}B = \bar{A}$

3. $\bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB$



$= 1$

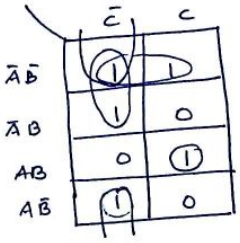
4. $S = \Sigma_1(m_0, m_1, m_2)$



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

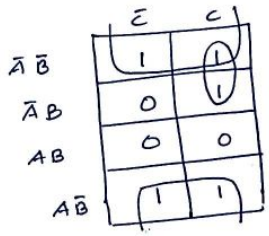
3 VARIABLE

1. $S = \Sigma_1 (m_0, m_1, m_2, m_4, m_7)$



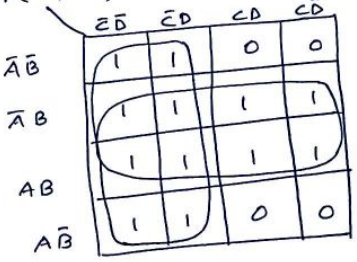
$\Rightarrow \bar{A}\bar{B} + \bar{A}\bar{C} + \bar{B}\bar{C} + ABC$

2. $A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$



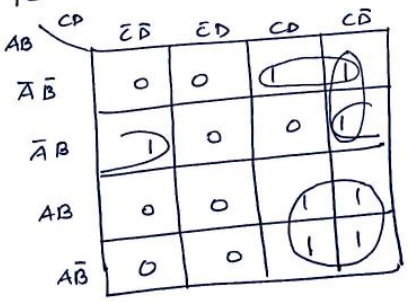
$\Rightarrow \bar{B} + \bar{A}C$

3. $S = \Sigma_1 (m_0, m_1, m_4, m_5, m_6, m_7, m_8, m_9, m_{12}, m_{13}, m_{14}, m_{15})$



$\Rightarrow \bar{C} + B$

4. $S = \Sigma_1 (3, 3, 4, 5, 10, 11, 14, 15)$

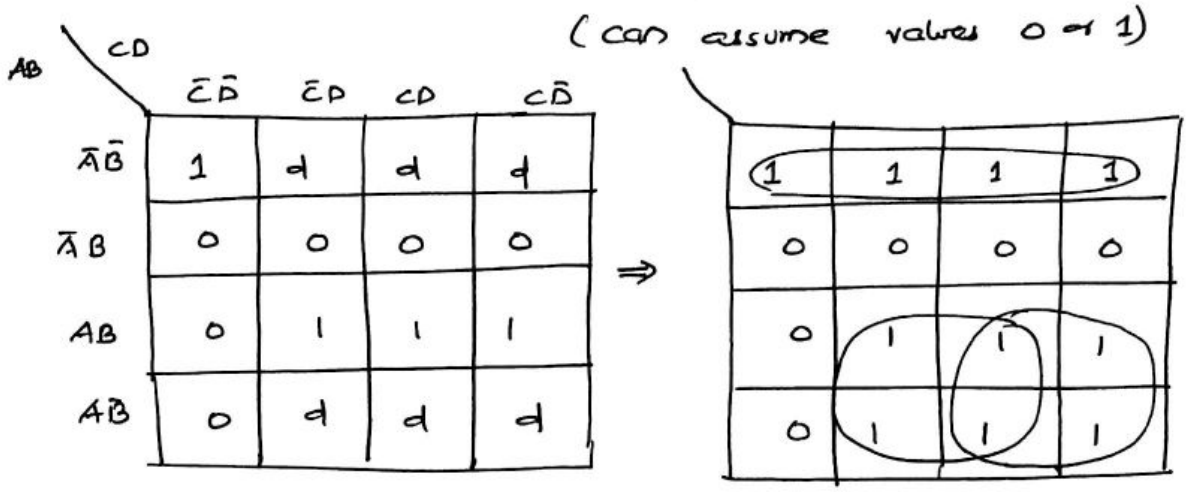


$\Rightarrow Ac + \bar{B}C + \bar{A}B\bar{D} + C\bar{D}$

K- MAP INCLUDING DONT CARE

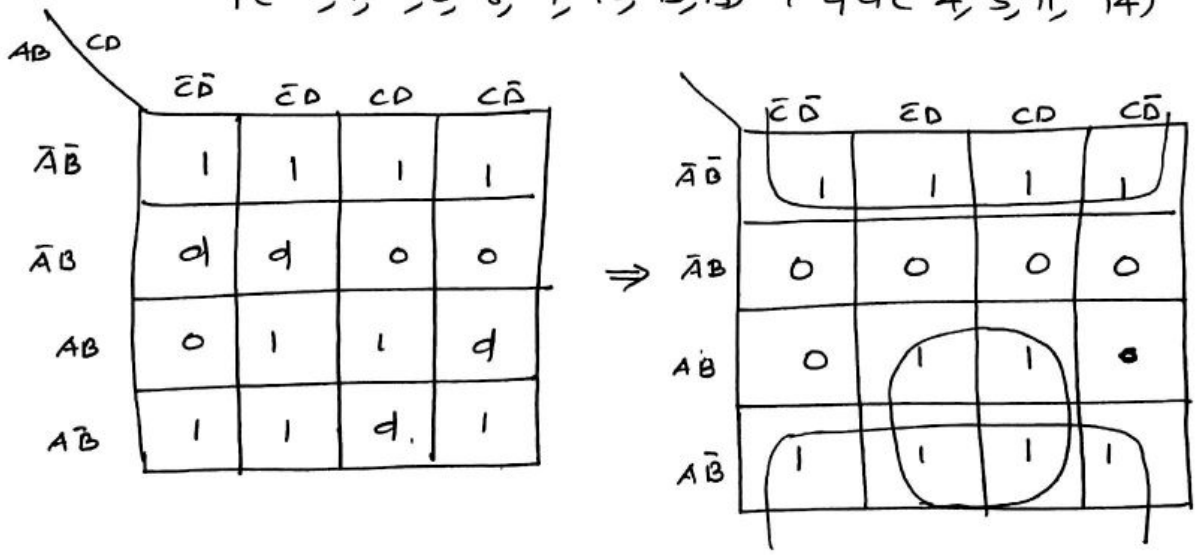
Q. $S = \Sigma(0, 13, 14, 15) + \Sigma d(1, 2, 3, 9, 10, 11)$

↓ min terms ↓ don't care terms (can assume values 0 or 1)



$S = \bar{A}\bar{B} + AD + AC$

Q. $S = \Sigma(0, 1, 2, 3, 8, 9, 10, 13, 15) + \Sigma d(4, 5, 11, 14)$



$S = AD + \bar{B}$

Q. Simplify the following expression using K-map approach.

$S = \Sigma(1, 3, 7, 11, 15) + \Sigma d(0, 2, 5)$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	d	1	1	d
$\bar{A}B$	0	d	1	0
AB	0	0	1	0
$A\bar{B}$	0	0	1	0

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	1
$\bar{A}B$	0	0	1	0
AB	0	0	1	0
$A\bar{B}$	0	0	1	0

II nd module completed.

25/11/17

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

MODULE-3

COMBINATIONAL CIRCUITS

XOR (Exclusive OR)

Same i/p ⇒ o/p = 0
diff i/p ⇒ o/p = 1
} Also known as COMPARATOR

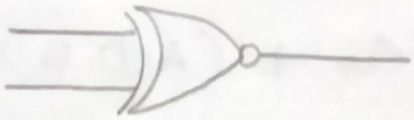
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

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

XNOR (Exclusive NOR)

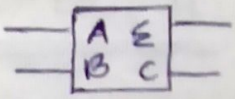
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

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



ADDER CIRCUITS

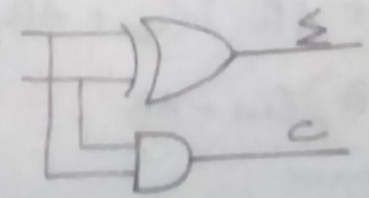
① Hay adder → Only two bits



A	B	Σ	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$\Sigma = A \oplus B$

Carry = AB



② Full adder

A	B	C_{in}	Σ	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$\bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$

$(\bar{A}\bar{B} + AB)C_{in} + (\bar{A}B + A\bar{B})C_{in}$

$= \overline{A \oplus B}\, C_{in} + (A \oplus B)C_{in}$ $\quad A \oplus B = Y$

$= \bar{Y}C_{in} + YC_{in}$

$= Y \oplus C_{in}$

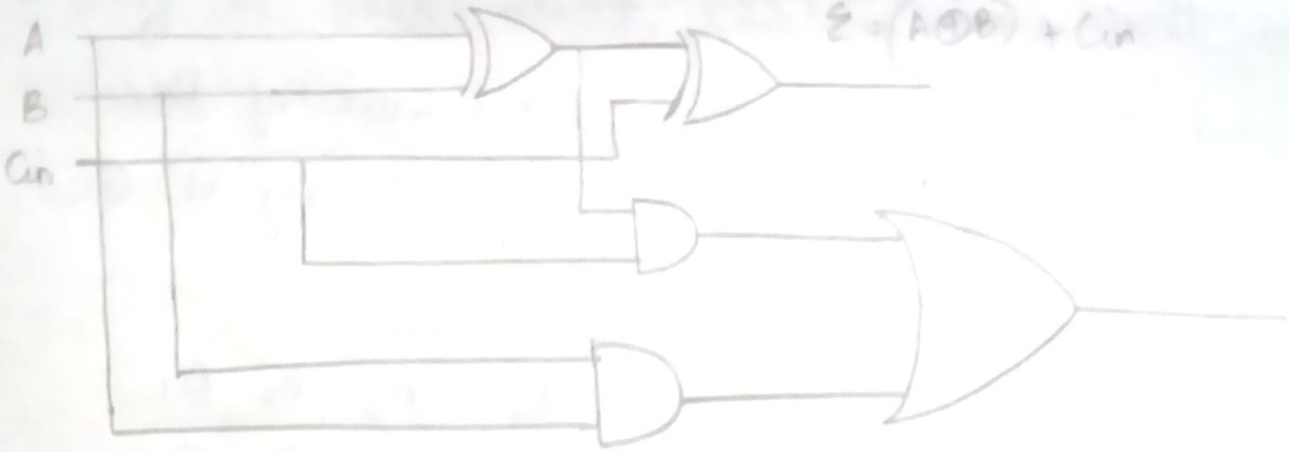
$= [A \oplus B] \oplus C_{in}$

$Cout = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$

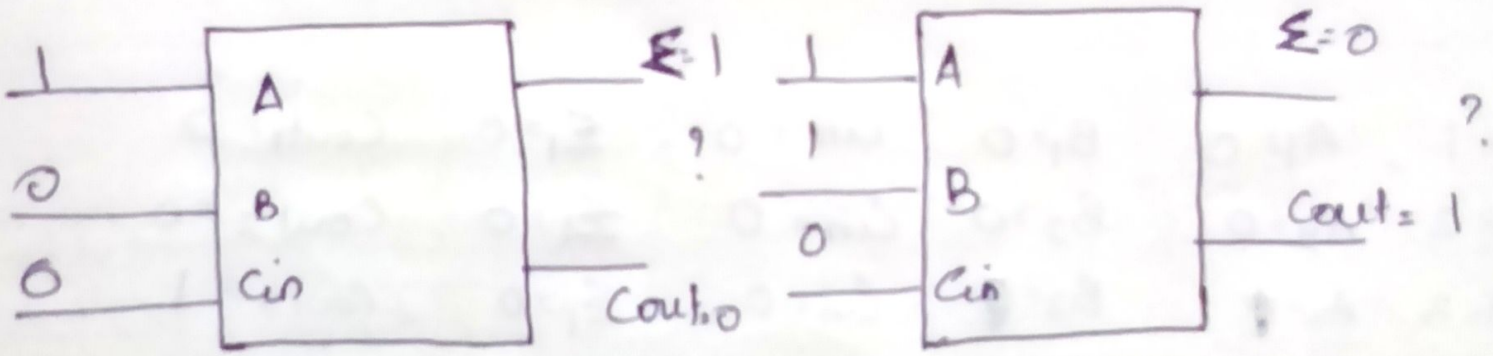
$= \bar{A}BC_{in} + A\bar{B}C_{in} + AB$

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

$= (A \oplus B)C_{in} + AB$



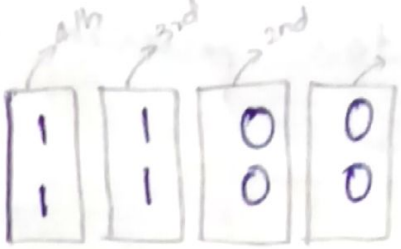
For the full adder shown, determine the outputs:



Q. Use the 4 bit full adder truth table to find the Sum and o/p carry for the addition of the following two 4 bit nos. if i/p carry is 0.

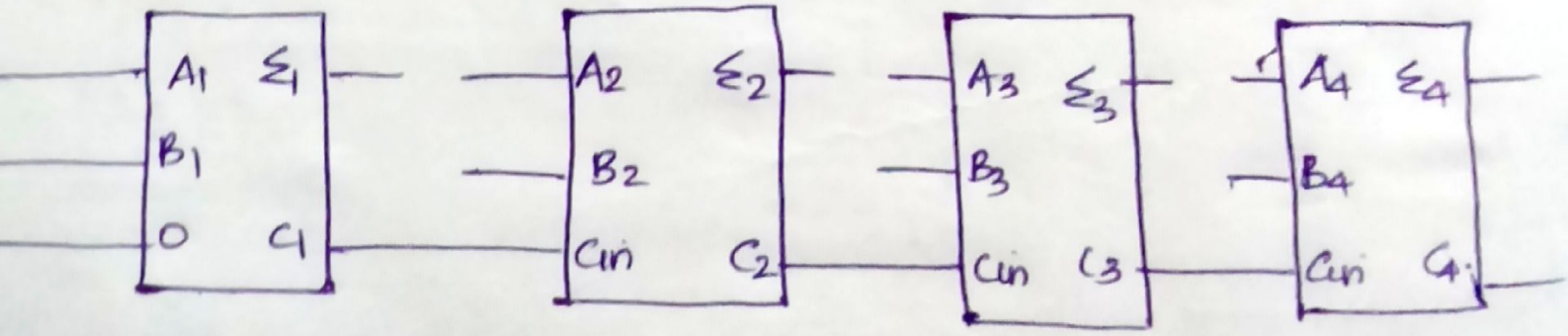
A_4	A_3	A_2	A_1
1	1	0	0

B_4	B_3	B_2	B_1
1	1	0	0



$= 1$,	$A_1 = 0$	$B_1 = 0$	$C_{in} = 0$	$\Sigma_1 = 0$	$Cout_1 = 0$
$= 2$	$A_2 = 0$	$B_2 = 0$	$C_{in} = 0$	$\Sigma_1 = 0$	$Cout_2 = 0$
$= 3$	$A_3 =$ [illegible]	$B_3 =$ [illegible]	$C_{in} = 0$	$\Sigma_1 = 0$	$Cout_3 = 1$
$= 4$	$A_4 = 1$	$B_4 = 1$	$C_{in} = 1$	$\Sigma_1 = 1$	$Cout_4 = 1$

$= 11000_2$



Q. Use the truth table to find the result of adding the binary number 1011 & 1010 using full ladder.

A_4	A_3	A_2	A_1
1	0	1	1
B_4	B_3	B_2	B_1
1	0	1	0

n = 1, $A_1 = 1$, $B_1 = 0$, $C_1 = 0$, $\Sigma_1 = 1$, Cout = 0

n = 2, $A_2 = 1$, $B_2 = 1$, $C_2 = 0$, $\Sigma_2 = 0$, Cout = 1

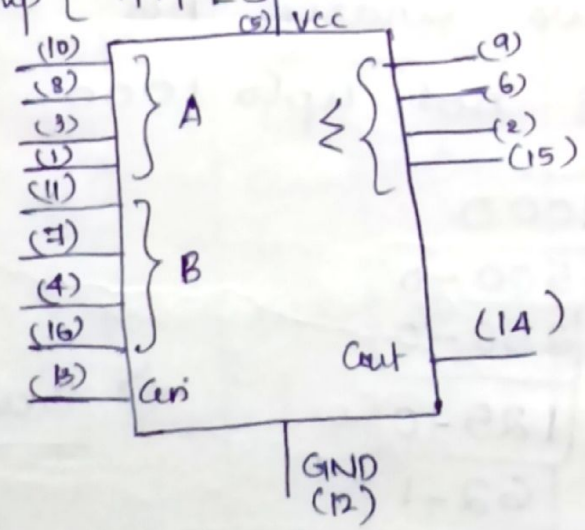
n = 3, $A_3 = 0$, $B_3 = 0$, $C_3 = 1$, $\Sigma_3 = 1$, Cout = 0

n = 4, $A_4 = 1$, $B_4 = 1$, $C_4 = 0$, $\Sigma_4 = 0$, Cout = 1

$\Rightarrow 10101_2$

---

4 bit adder chip — chip name: 74LS83A / 74LS283



Q. Show how two @74LS83A adders can be connected to form a 8-bit ILel adder to add the two eight bits 10111001 & 10011110

P.T.O

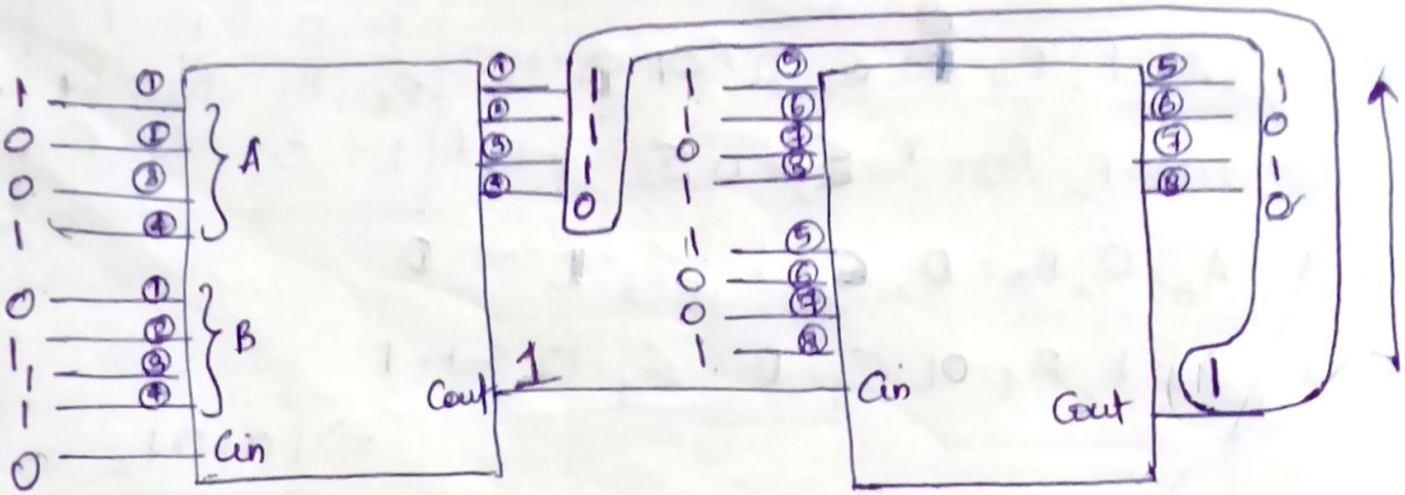
w:

$A_8\ A_7\ A_6\ A_5\ A_4\ A_3\ A_2\ A_1$

1 0 1 1 1 0 0 1 +

1 0 0 1 1 1 1 0

$B_8\ B_7\ B_6\ B_5\ B_4\ B_3\ B_2\ B_1$



$= 101010111_2$

2. How many 74LS283 adders would be required to add two binary nos. each representing decimal nos upto 1000.

2 | 1000
2 | 500 - 0
2 | 250 - 0
2 | 125 - 0
2 | 62 - 1
2 | 31 - 0
2 | 15 - 1
2 | 7 - 1
2 | 3 - 1
2 | 1 - 1

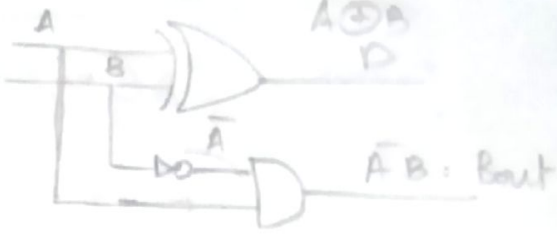
⇒ 3 adders

HALF SUBTRACTOR

A	B	D	Bout
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

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

$Bout = \bar{A}B$



FULL SUBTRACTOR

A	B	Bin	Bout	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$D = \bar{A}\,\bar{B}\,Bin + \bar{A}\,B\,\overline{Bin} + A\,\bar{B}\,\overline{Bin} + A\,B\,Bin$

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

$= \overline{(A \oplus B)}\,Bin + (A \oplus B)\,\overline{Bin}$

$= (A \oplus B) \oplus Bin$

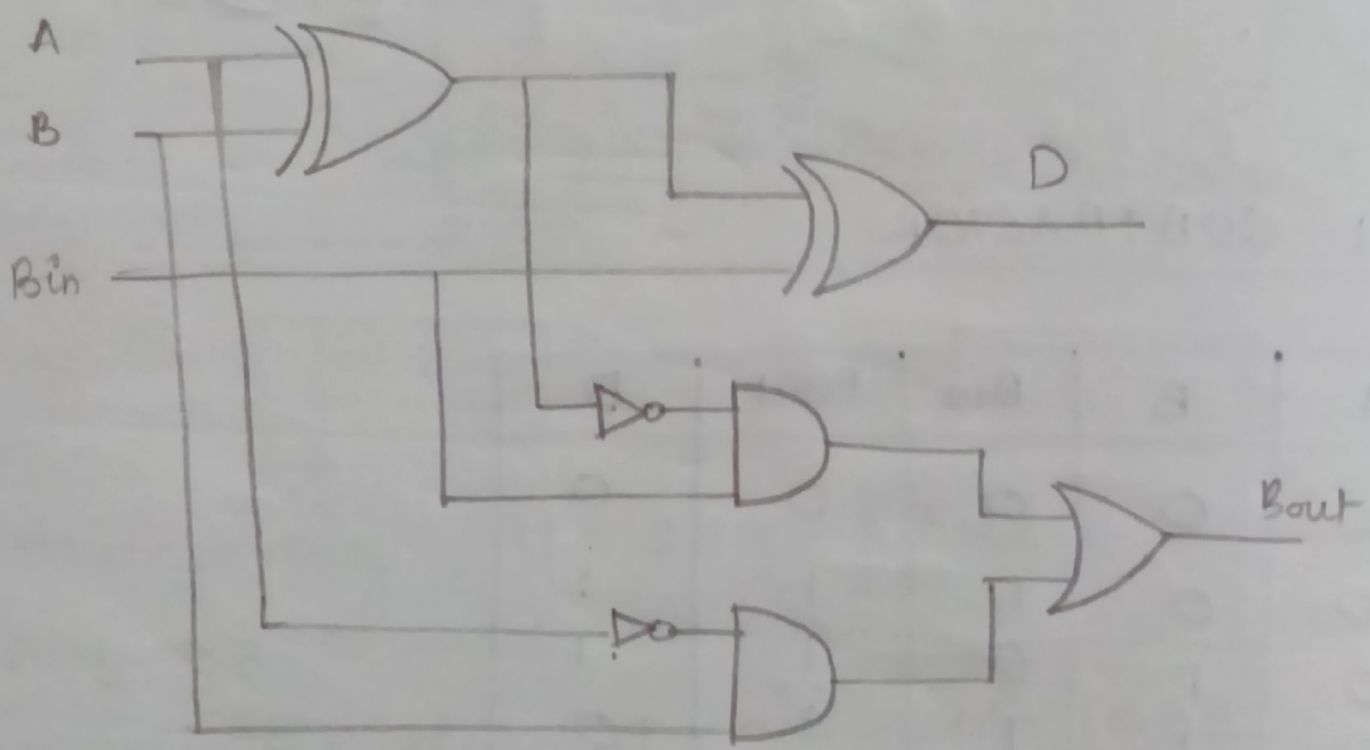
$B_{out} = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + \bar{A}BB_{in} + ABB_{in}$

$= \bar{A}\bar{B}B_{in} + \bar{A}B + ABB_{in}$

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

$= (\overline{A \oplus B})B_{in} + \bar{A}B$

$\left[\overline{A \oplus B} = \bar{A}\bar{B} + AB\right.$

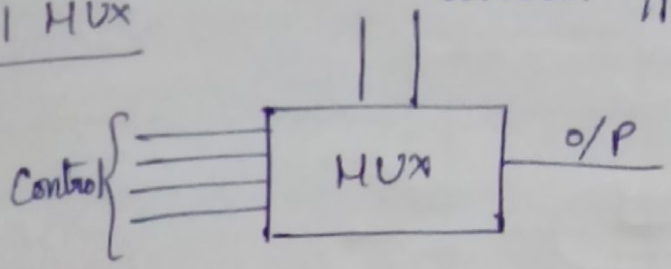


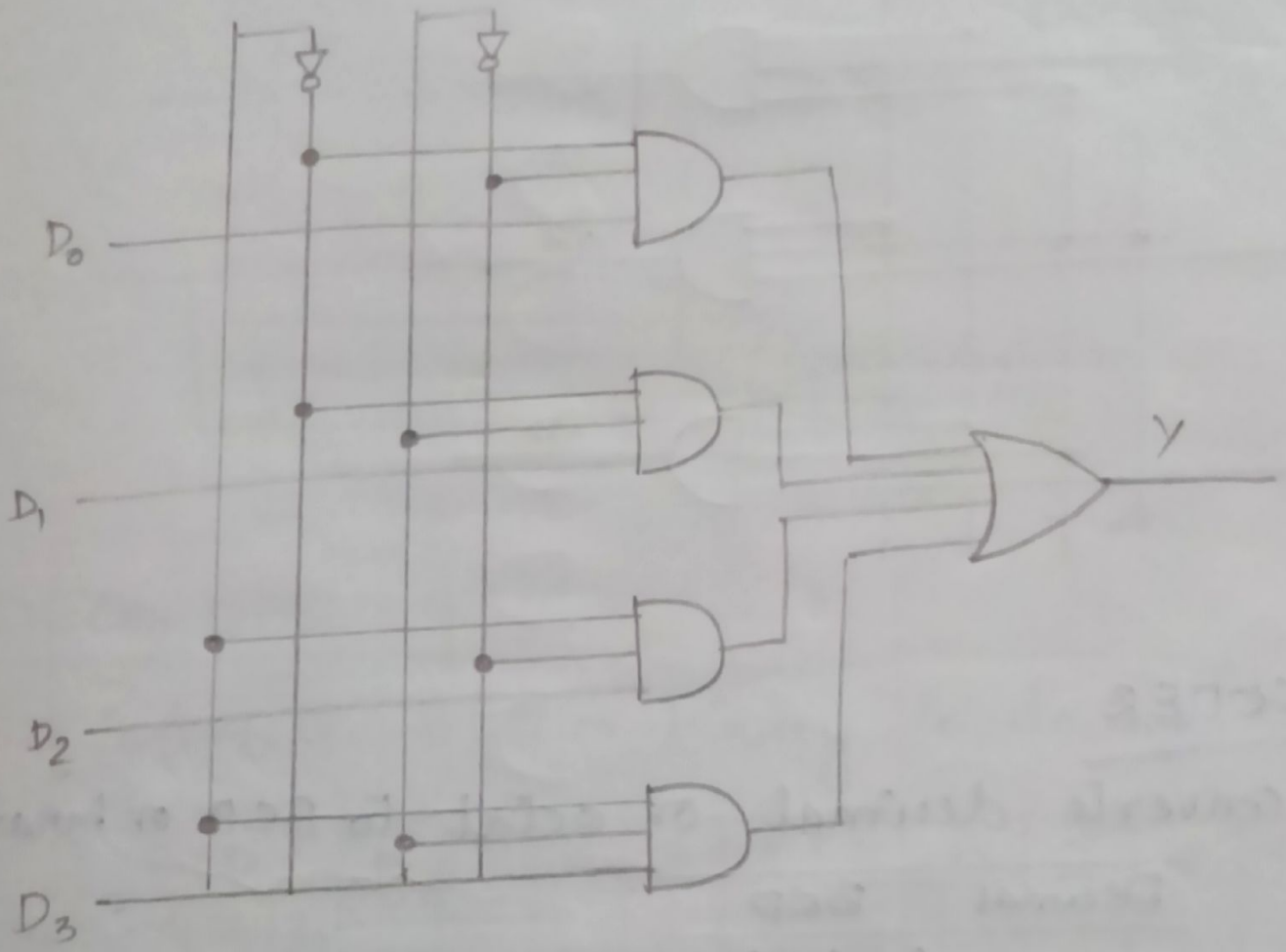
2. Explain ALU using block diagram.

2. Implement a full subtractor using a full adder.

MULTIPLEXER (MUX)

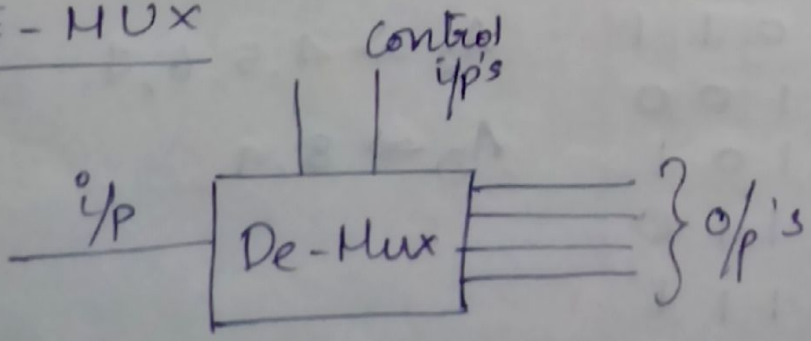
⇒ 4 to 1 MUX

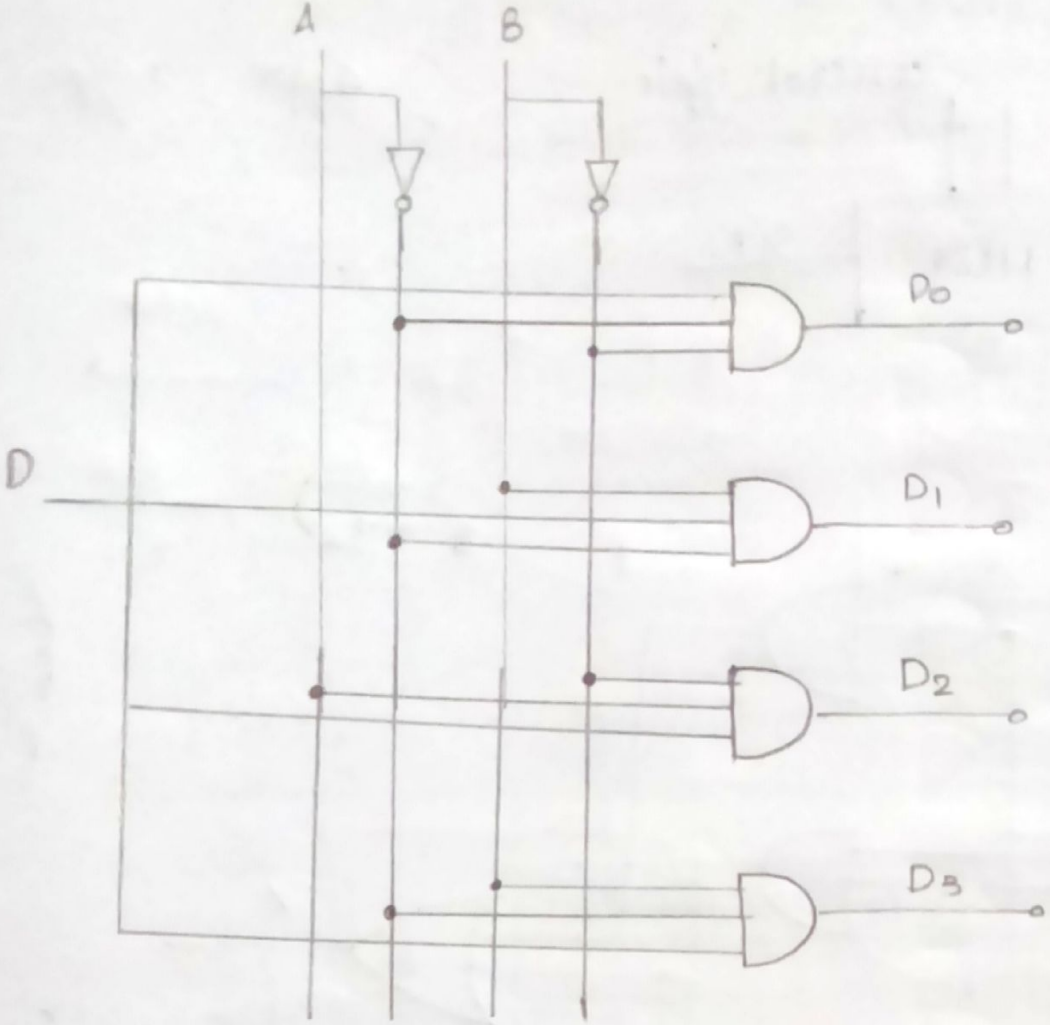




* when A = 0, B = 0, Y = D0

DE-MUX





ENCODER

converts decimal or octal to BCD or binary

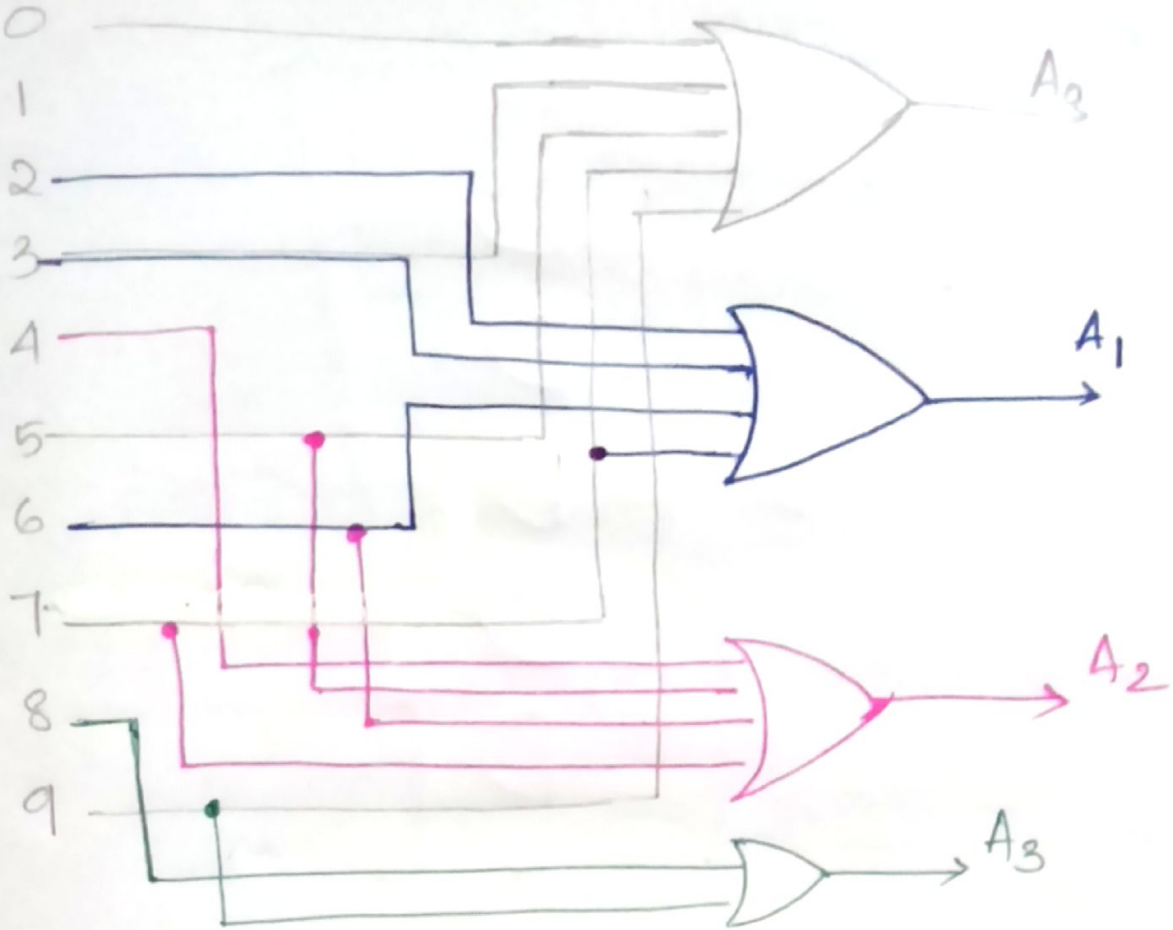
Decimal	BCD			
	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

$A_0 \Rightarrow 1, 3, 5, 7, 9$

$A_1 \Rightarrow 2, 3, 6, 7$

$A_2 \Rightarrow 4, 5, 6, 7$

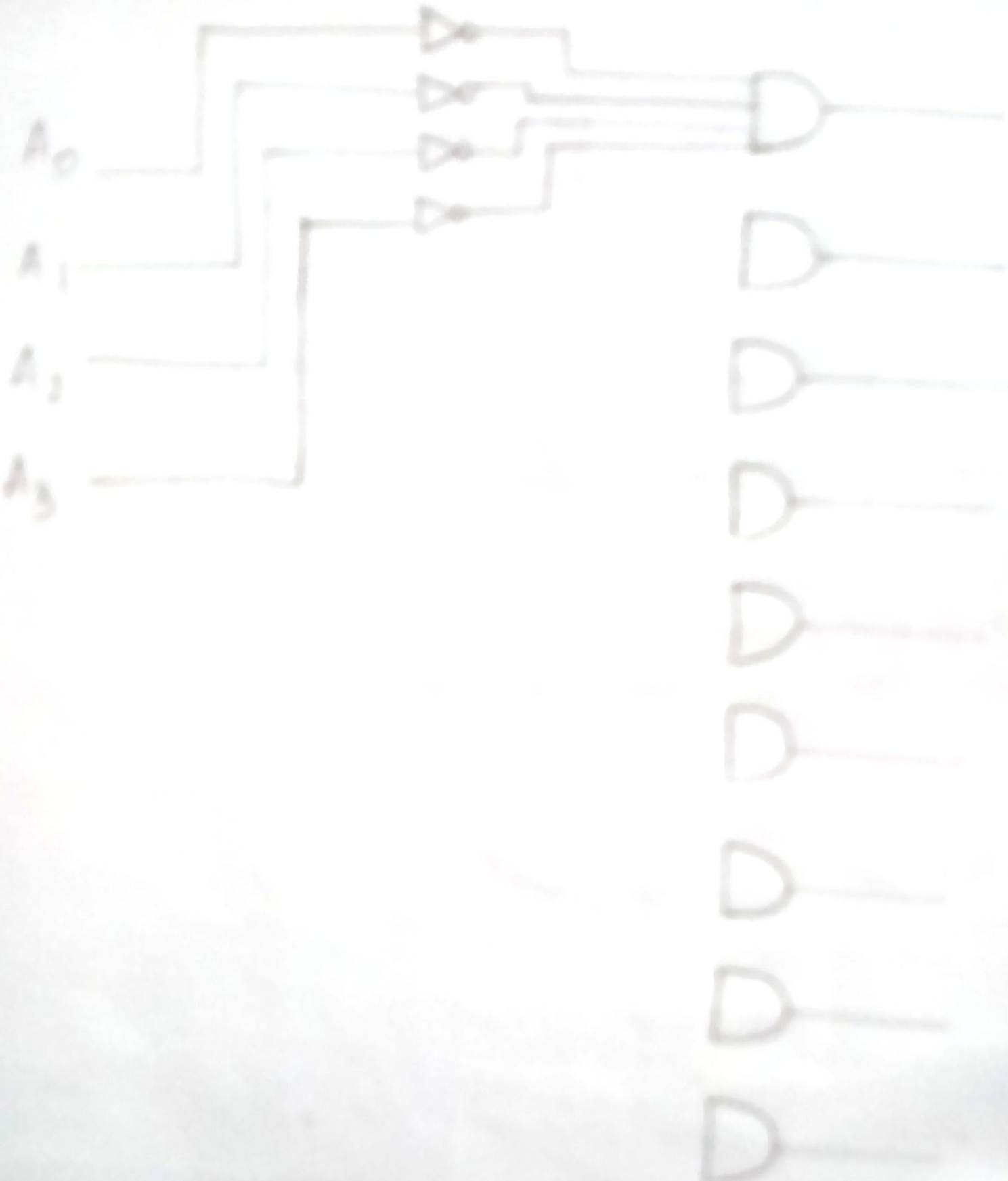
$A_3 \Rightarrow 8, 9$



DECORDER

converts BCD or binary to decimal or octal.

BCD				0	1	2	3	4	5	6	7	8	9
A_3	A_2	A_1	A_0										
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1



MODULE - 4

FLIP FLOP

→ Logic gates – Combinational circuits

– because it doesnot care about the previous i/p & it doesnot have a memory.

Sequential circuit → It has an i/p & o/p.

→ It has a memory sequence and is given as input through a feedback.

FLIP - FLOPS

* Printed notes.

CONVERSION OF FLIP-FLOP

Excitation table

Q_t (present)	Q_{t+1} (next)	S	R	J	K	D	T
0	0	0	x	0	x	0	0
0	1	1	0	1	0	1	1
1	0	0	1	0	1	0	1
1	1	x	0	x	0	1	0

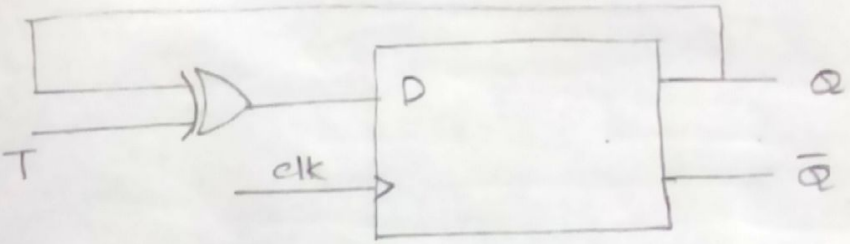
Q. Convert DFF to TFF

From the excitation table:

T	Q	D
0	0	0
1	0	1
1	1	0
0	1	1

$D = T\bar{Q} + Q\bar{T}$

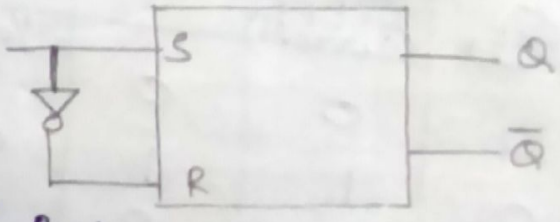
$= T \oplus Q$

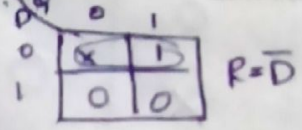


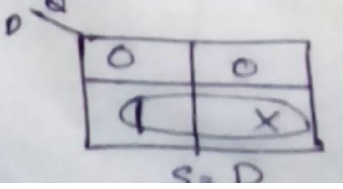
Q. Convert SRFF to DFF

From the excitation table:

S	R	D	Q
0	X	0	0
1	0	1	0
0	1	0	1
X	0	1	1







$R = D'$

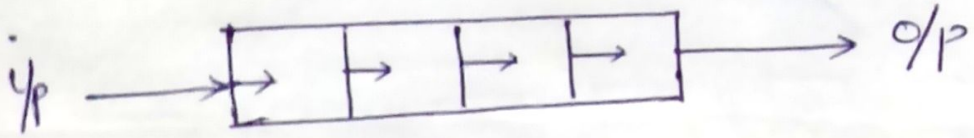
$S = D$

REGISTERS

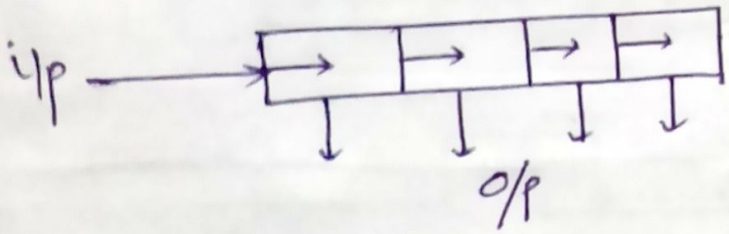
↳ store values

↳ shifts values.

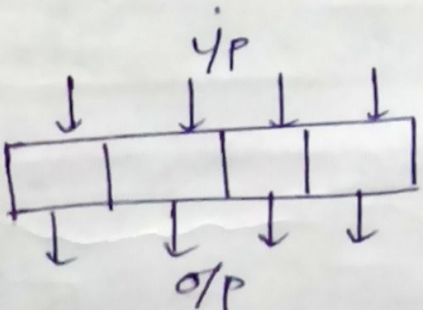
1). Serial i/p Serial o/p (SISO)



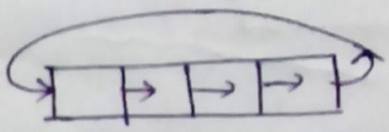
2). Serial input Parallel o/p (SIPO)



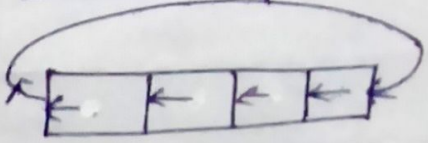
3) Parallel i/p Parallel o/p (PIPO)



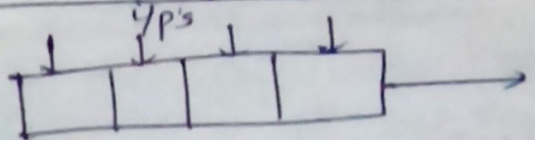
Rotate right:



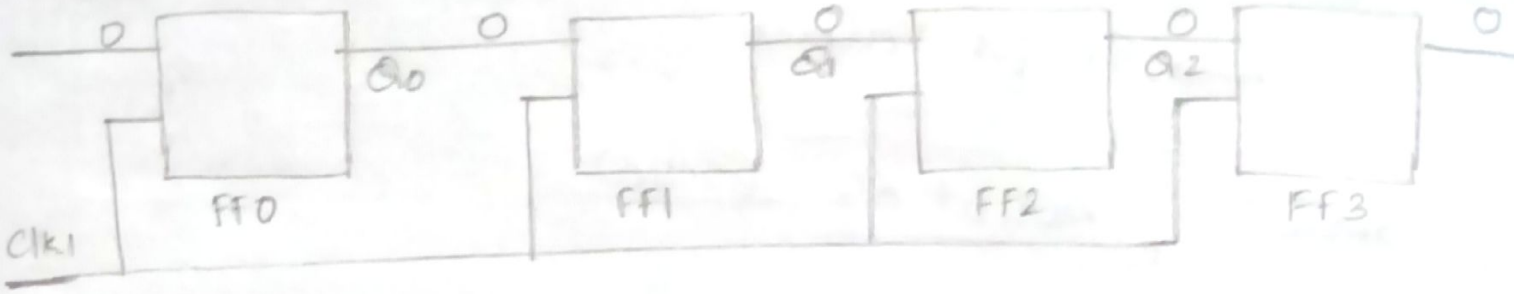
Rotate left:

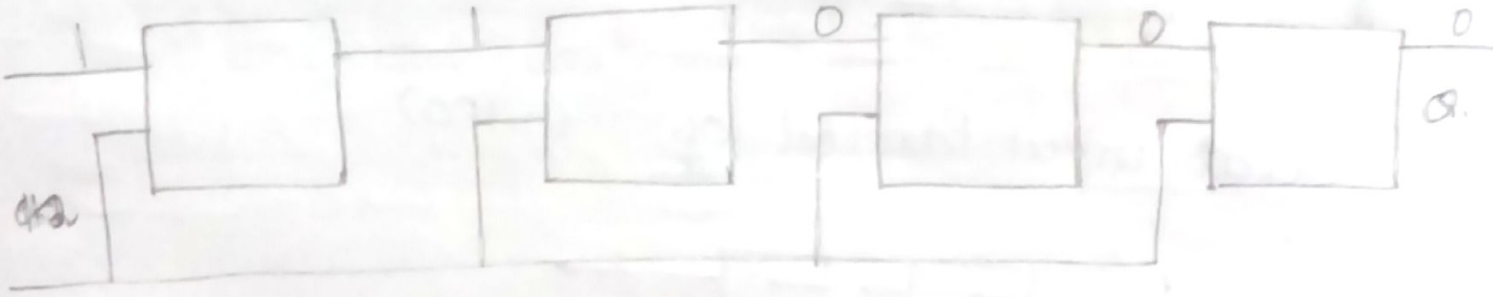


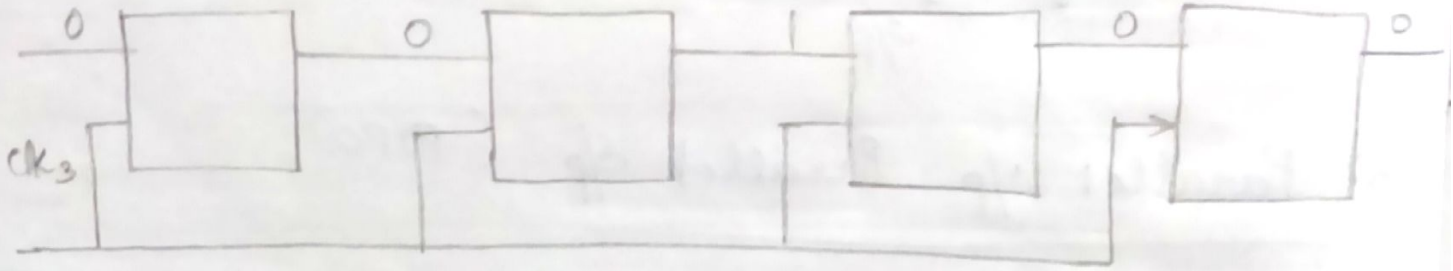
4). Parallel i/p Serial o/p (PISO)

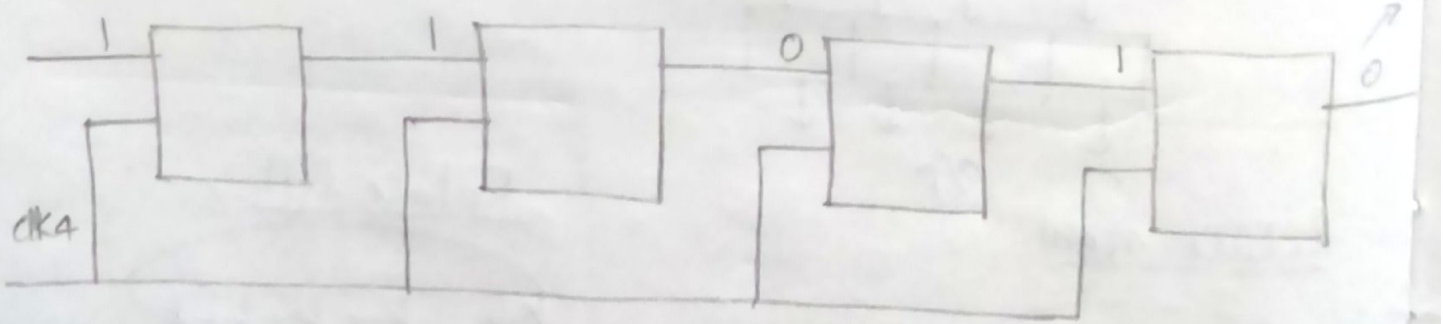


SERIAL i/p SERIAL o/p (S190)









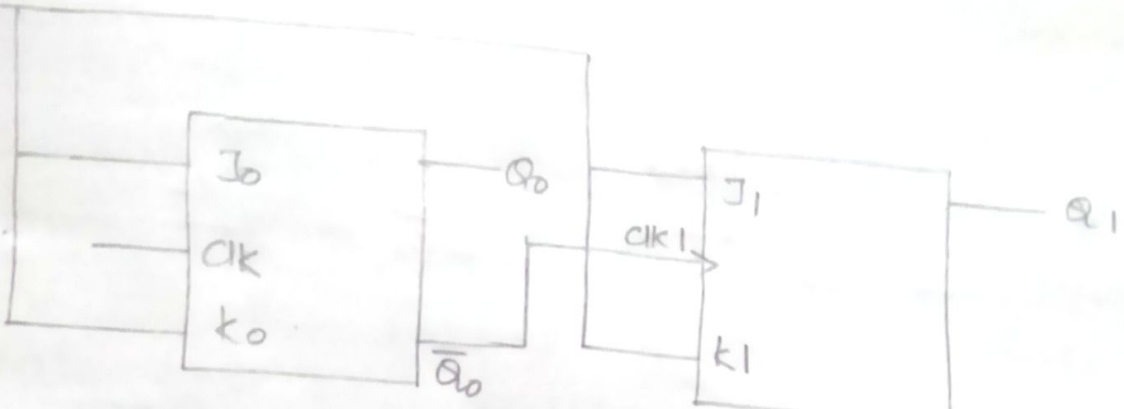
clk 5 → Q_2 as o/p
clk 6 → Q_1 as o/p
clk 7 → Q_0 as o/p
clk 8 → 0

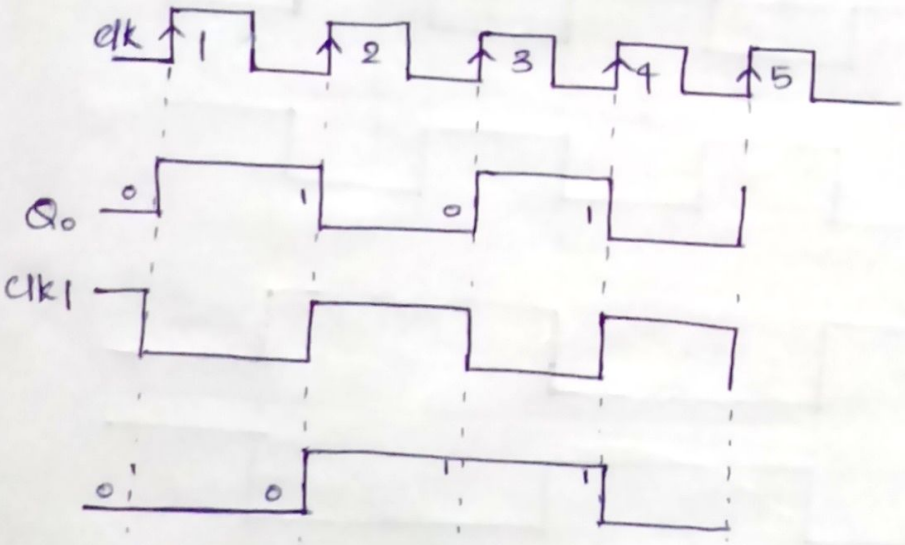
Flip Flop clear

ASYNCHRONOUS COUNTER (RIPPLE COUNTER)

0-3 COUNTER

High



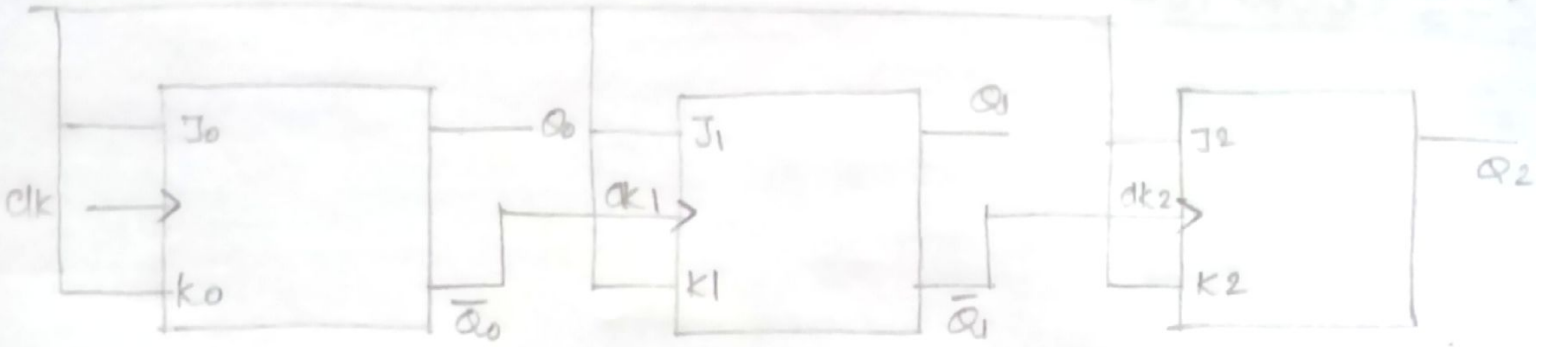


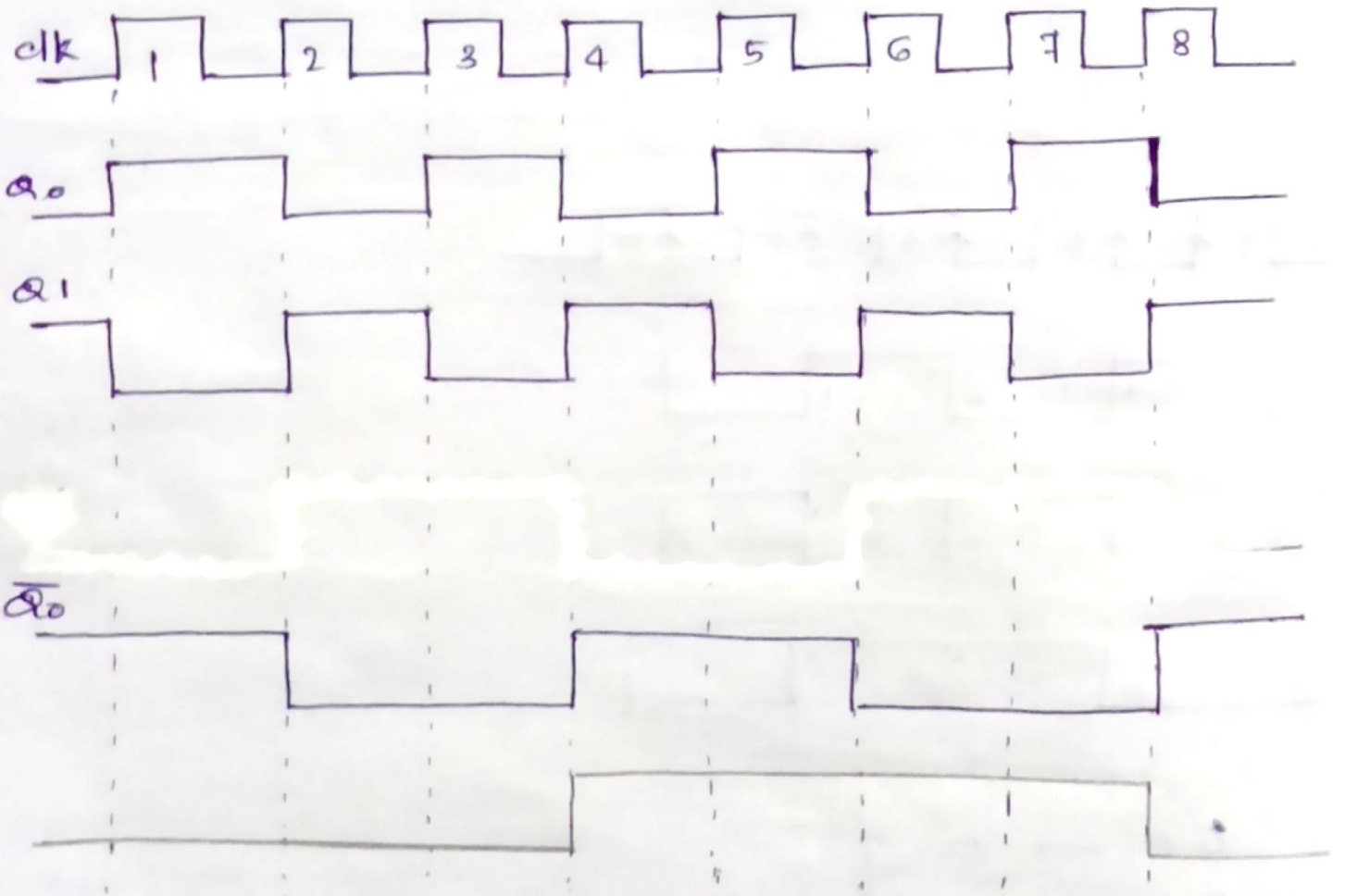
0 0 → 0
0 1 → 1
1 0 → 2
1 1 → 3

} 0-3 counter.

→ 1st flip flop is clocked by external clock pulse & then each successive flip-flop is clocked by the o/p of the preceding flip flop An Asynchronous counter is one in which the flip-flops within the counter donot change the states exactly at the same time because they donot have a common clock pulse.

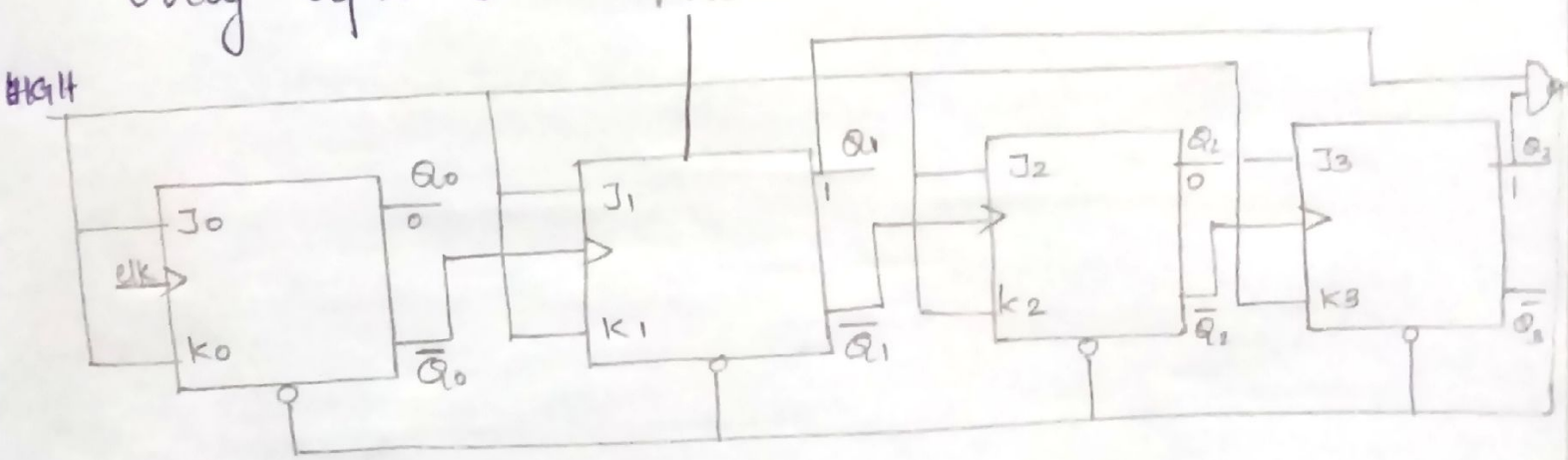
0-7 COUNTER



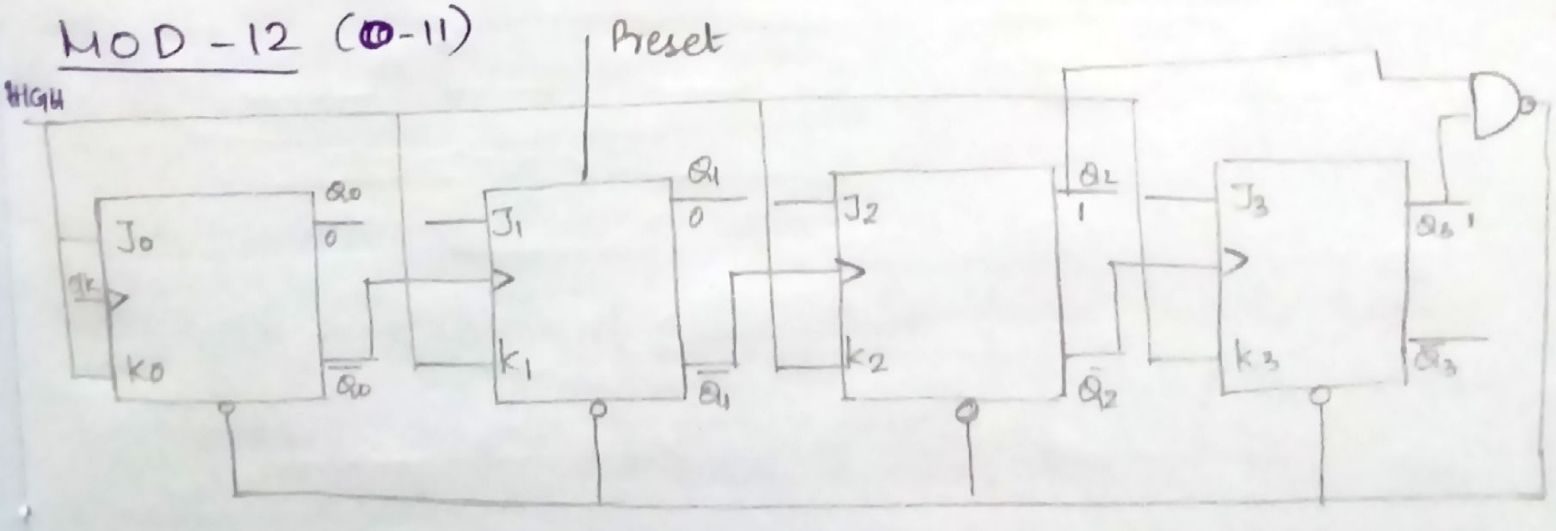


Mod-10 (0-9)

To count 10 Nos (0-9) we need 4 flip-flops But 4 flip-flops will count upto 0-15. So use 'clear'. Here, when the 10th clock comes a high (1) will come to 'clear' and sets all the flipflops to 0. Thus containing only upto 1.



MOD-12 (00-11)

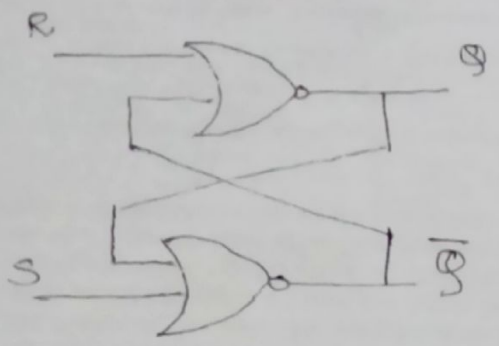


* with bubbles – Active Low
* without bubbles – Active high.

S-R FLIPFLOP.

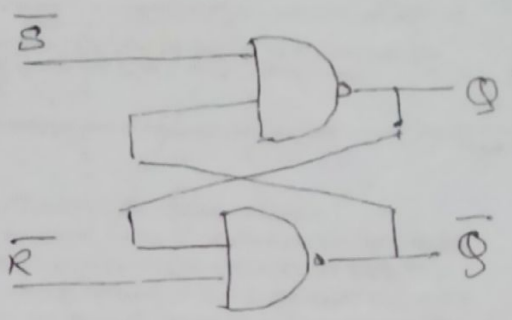
Two versions:

(1)



Active High i/P SR Latch.

(2)



Active LOW i/P SR Latch.

TRUTH TABLE:—

(1)

INPUTS		OUTPUTS		COMMENTS
S	R	Q	$\bar{Q}$	
0	0	NC	NC	Remains same
0	1	0	1	Latch RESET
1	0	1	0	Latch SET
1	1	0	0	Invalid

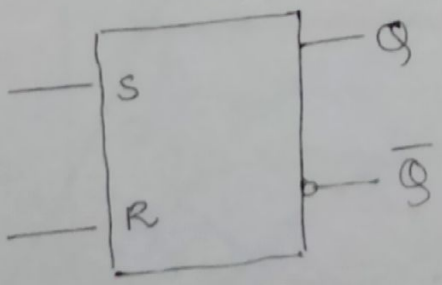
(2)

INPUTS		OUTPUTS		COMMENTS
S	R	Q	$\bar{Q}$	
0	0	NC	NC	Remain same
0	1	0	1	Latch RESET
1	0	1	0	Latch SET
1	1	1	1	Invalid

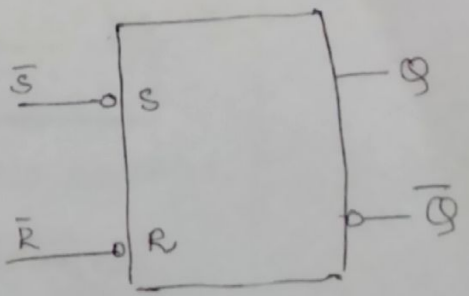
* NC – No change (o/p as previous)

LOGIC SYMBOL:—

(1)



(2)

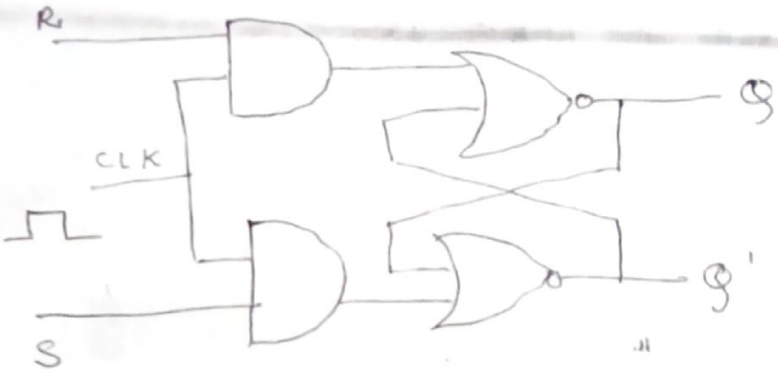


Active high i/p → indicates the circuit is activated when a HIGH signal is given.

Active low i/p → indicates the circuit is activated when a LOW signal is given.

GATED / CLOCKED / ENABLED SR FLIP FLOP

(a)



(b) LOGIC SYMBOL

- O/P of the AND gates remain 0 as long as clock pulse is 0.
- With both S=1 ; R=1 and clock pulse ↑ the O/P goes to 0, both Q & Q'. Then when pulse goes to LOW the O/P of AND gates becomes 0, which causes it to choose a state Q=0 or Q=1 and thus it is called indeterminate.

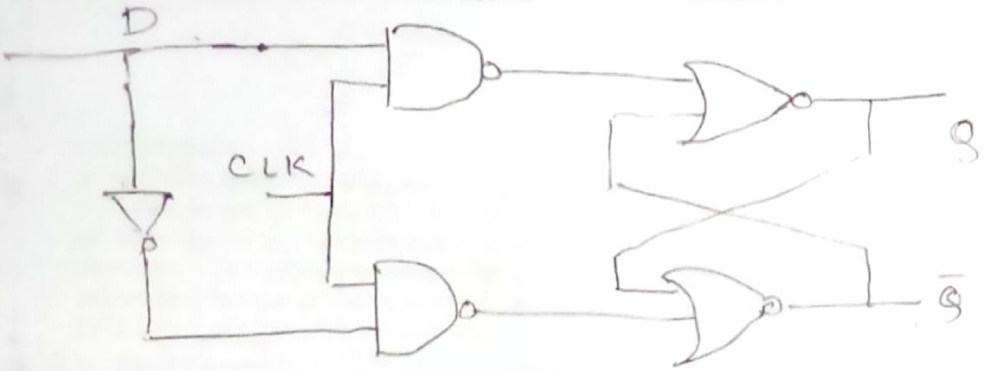
TRUTH TABLE OF POSITIVE EDGE TRIGGERED S-R FF :-

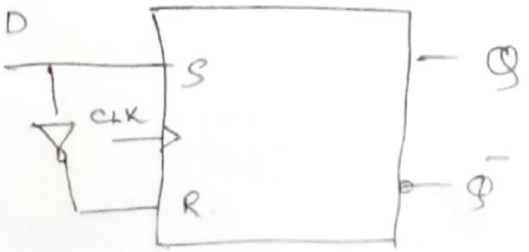
INPUTS			OUTPUTS		COMMENTS
S	R	CLK	Q	$\bar{Q}$	
0	0	X	Q_0	$\bar{Q}_0$	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	Invalid		

* ↑ → clock transition LOW to HIGH
  X → irrelevant ("don't care")
  Q_0 → previous state ; prior to clock transition.

D FLIP FLOP :

'D' — Data

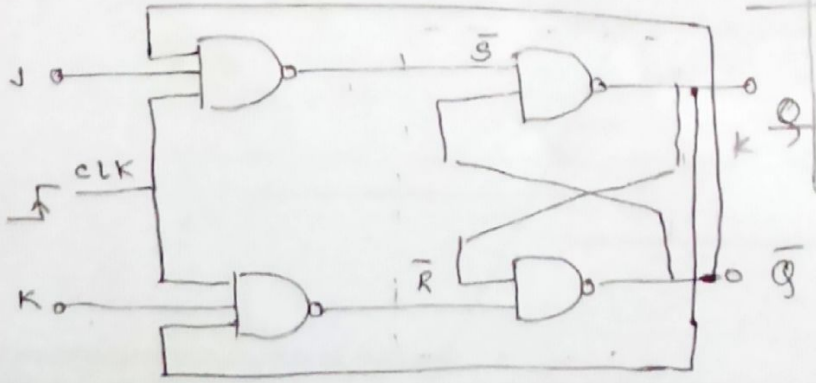


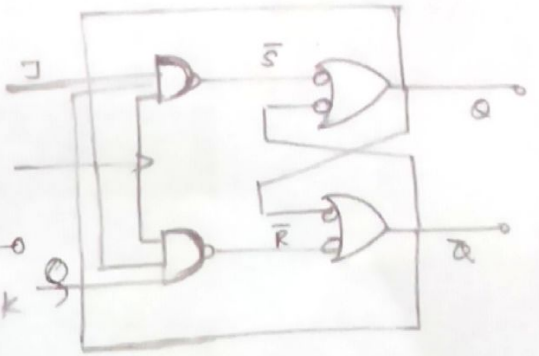


operation of positive edge triggered.

Q_0	D	Q
0	0	0
0	1	1
1	0	0
1	1	1

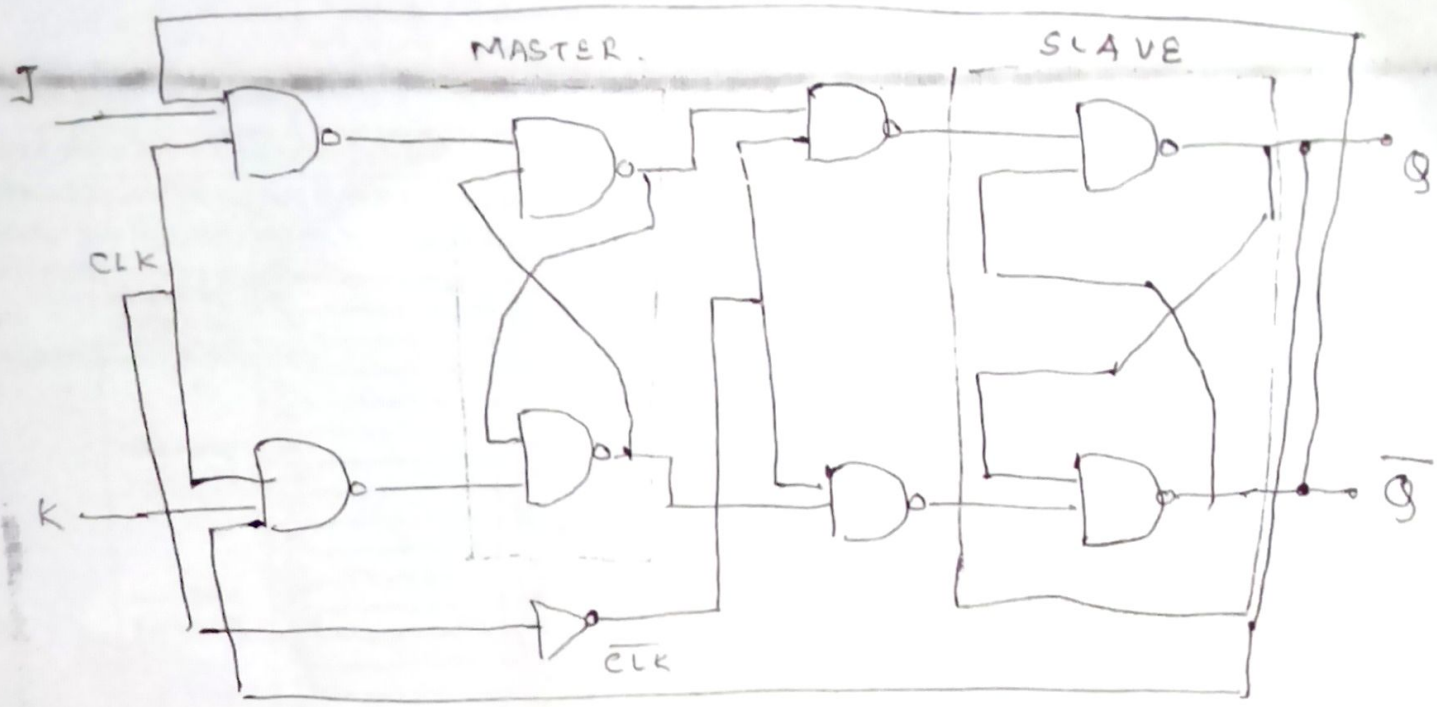
J K FLIP FLOP : -





INPUTS			OUTPUTS		COMMENTS
J	K	CLK	Q	$\bar{Q}$	
0	0	↑	Q_0	$\bar{Q}_0$	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	$\bar{Q}_0$	Q_0	Toggle

ASTER SLAVE JK FLIP FLOP



The input J K is connected to master SR. As clock to 'slave' FF is complement of master clock input, the slave SR flip flop does not toggle.

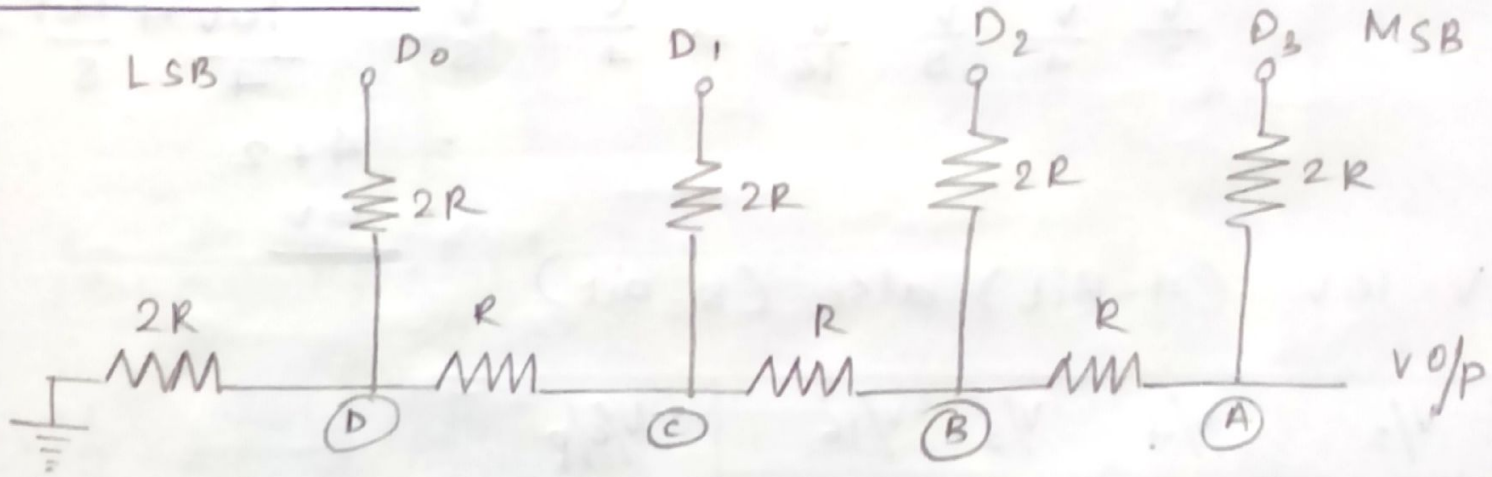
MODULE-6

DIGITAL TO ANALOG CONVERTOR (DAC)

DIGITAL TO ANALOG CONVERTER (DAC)

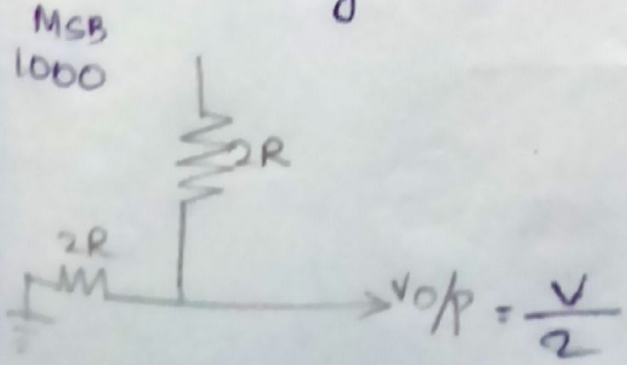
- R-2R Ladder (Binary Ladder)
- Weighted Resistor

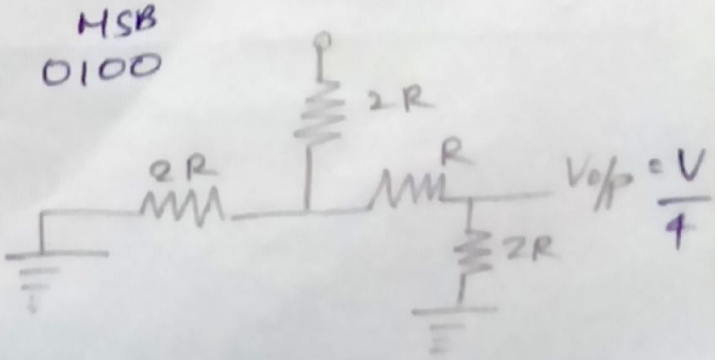
1) R-2R LADDER

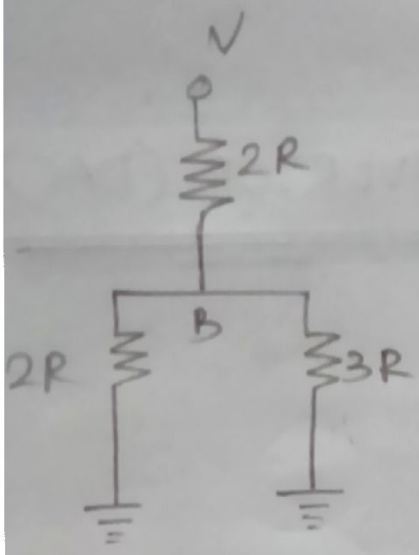


* R and 2R
* A, B, C, D ⟶ R
* Voltage is weighted.

→ From any node to ground, resistance is 2R.







$$V_B = \frac{V\left(\frac{6}{5}\right)R}{16 \times \frac{R}{5}} = \frac{3}{8}V$$

$$V_B = \frac{V \times \frac{6}{5}R}{\frac{6}{5}R + 2R}$$

16 V ⟶ 0110

$\frac{V}{2}$ $\frac{V}{4}$ $\frac{V}{8}$ $\frac{V}{16}$ ⇒ $\frac{V}{4} + \frac{V}{8} = \frac{16V}{4} + \frac{16V}{8}$

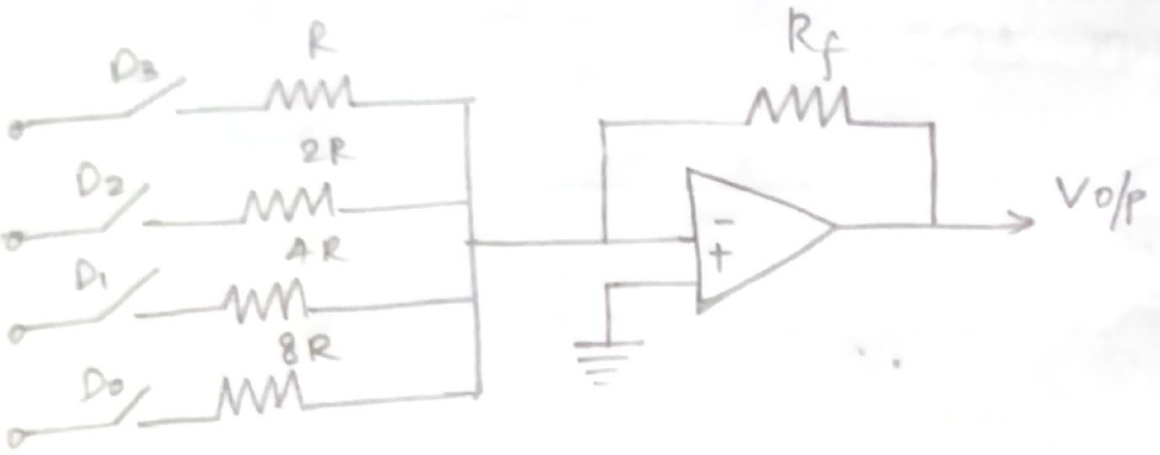
$= 4 + 2$

$= 6V$

V · 16V (4-Bit) also (8-Bit)

V/2	V/4	V/8	V/16	V o/p
D_3	D_2	D_1	D_0	
0	0	0	0	0
0	0	0	1	1V
0	0	1	0	2V
0	0	1	1	3V
0				

4-BIT WEIGHTED RESISTOR (Check 8-bit weighted Resistor)



$$V_{O/P} = -R_f\left[\frac{V}{R} + \frac{V}{2R} + \frac{V}{4R} + \frac{V}{8R}\right]$$

$$= -R_f\left[\frac{D_3V}{R} + \frac{D_2V}{2R} + \frac{D_1V}{4R} + \frac{D_0V}{8R}\right]$$

Assume $R_f = R = 1$

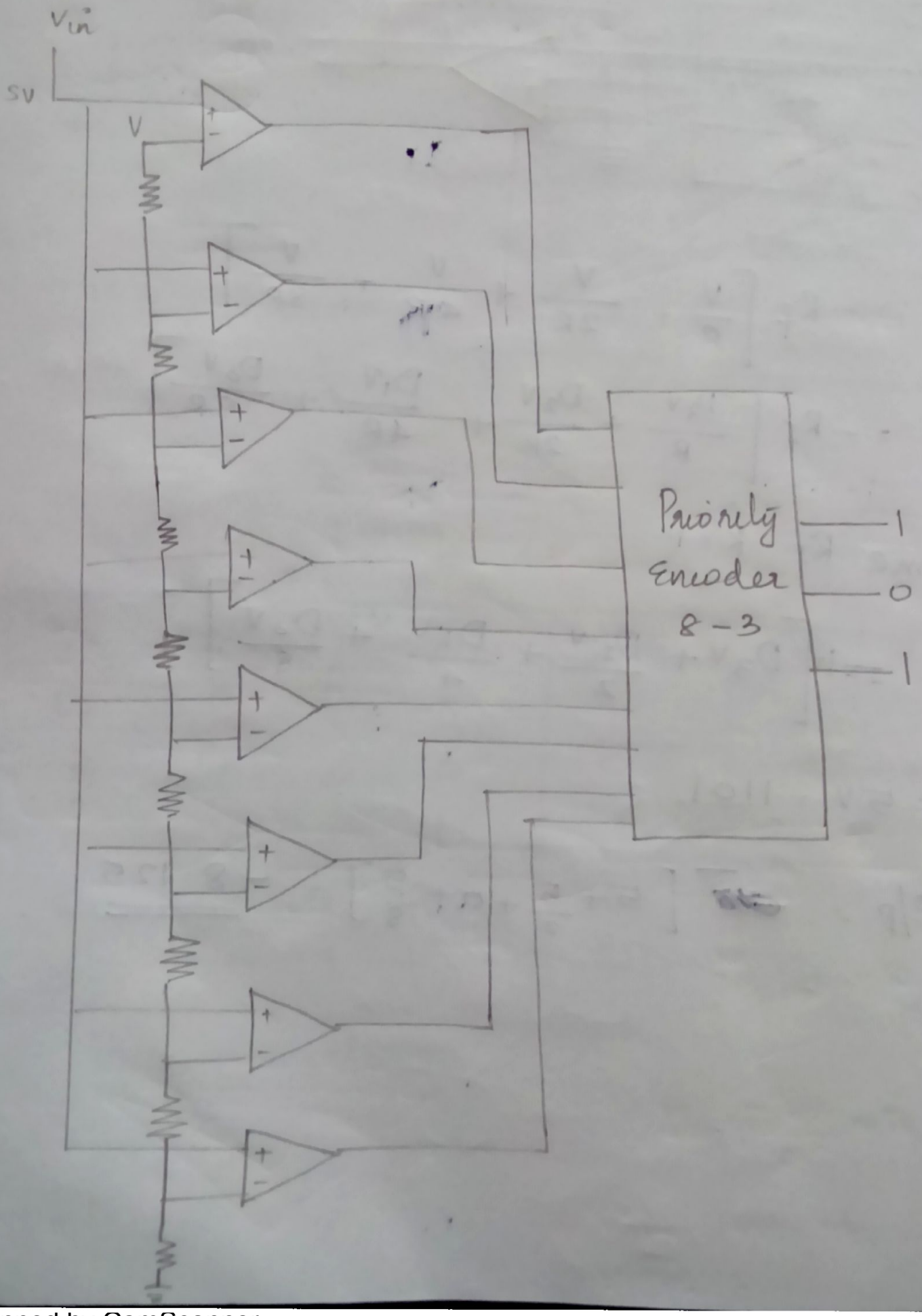
$$V_{O/P} = -1\left[D_3V + \frac{D_2V}{2} + \frac{D_1V}{4} + \frac{D_0V}{8}\right]$$

$V = 5V = 1101$

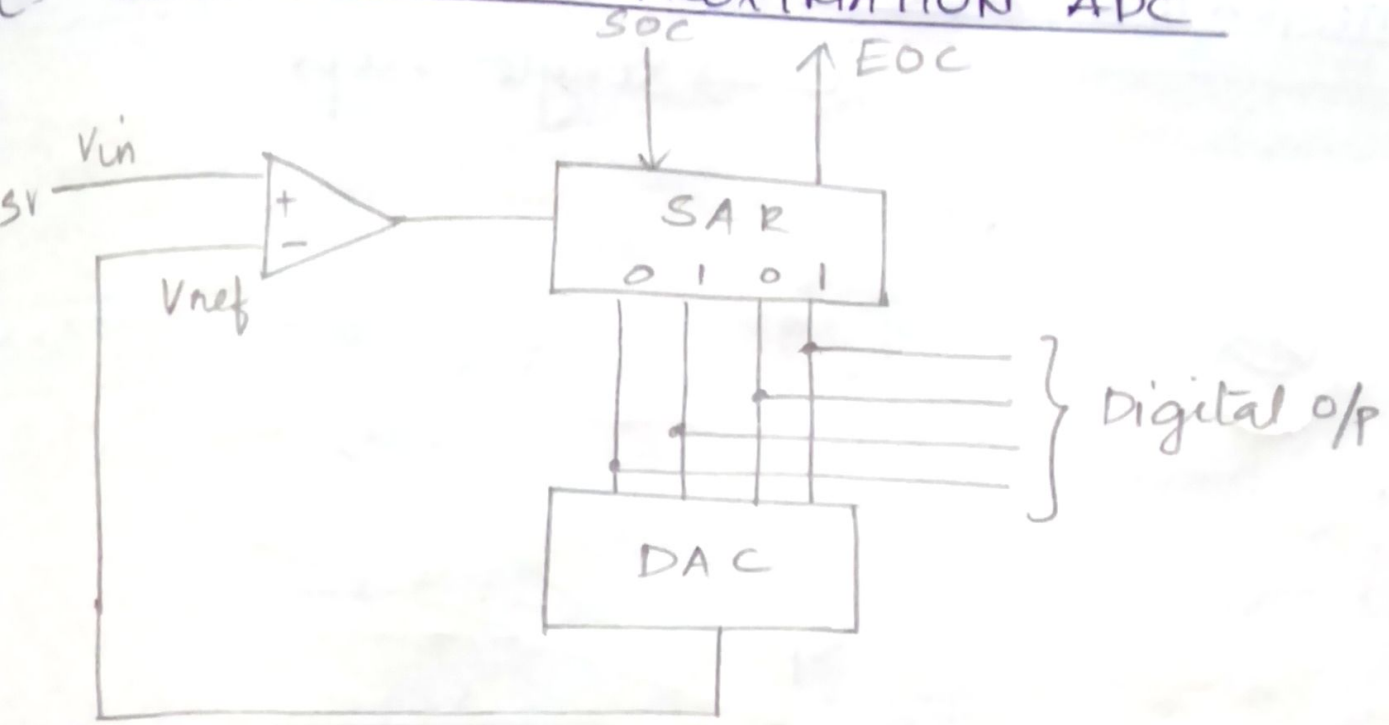
$$V_{O/P} = -\left[5 + \frac{5}{2} + 0 + \frac{5}{8}\right] = -8.125$$

ANALOG TO DIGITAL CONVERTER (ADC)

D. FLASH TYPE ADC



(2). SUCCESSIVE APPROXIMATION ADC



Vin = 5 V

Vin	Vref	
5	0	0 000
5V	8V	1000
5V	4V	0100
5V	6V	0110
5V	5V	0101

Vin = 10V

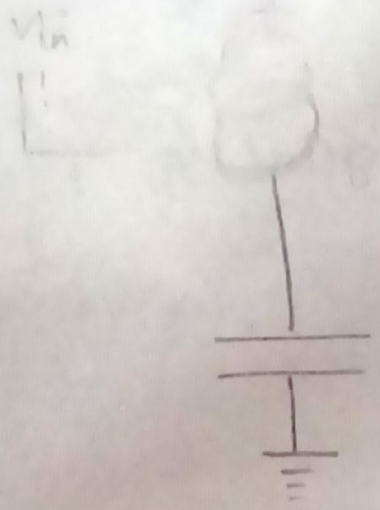
Vin	Vref	
10V	0	0000
10V	8	1 000
10V	12	1 100
10V	10	1010

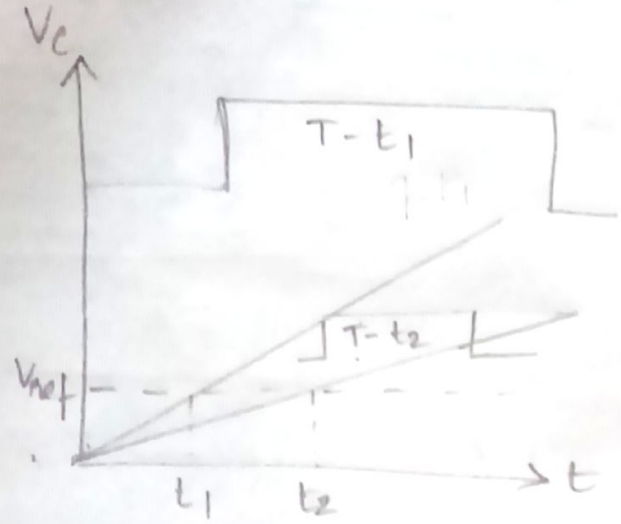
→ Dual Slope

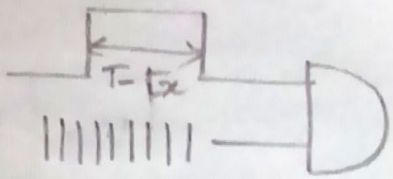
ANALOG TO DIGITA

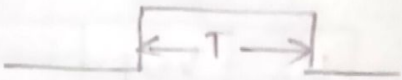
→ Single Slope

D. FLASH TYPE A

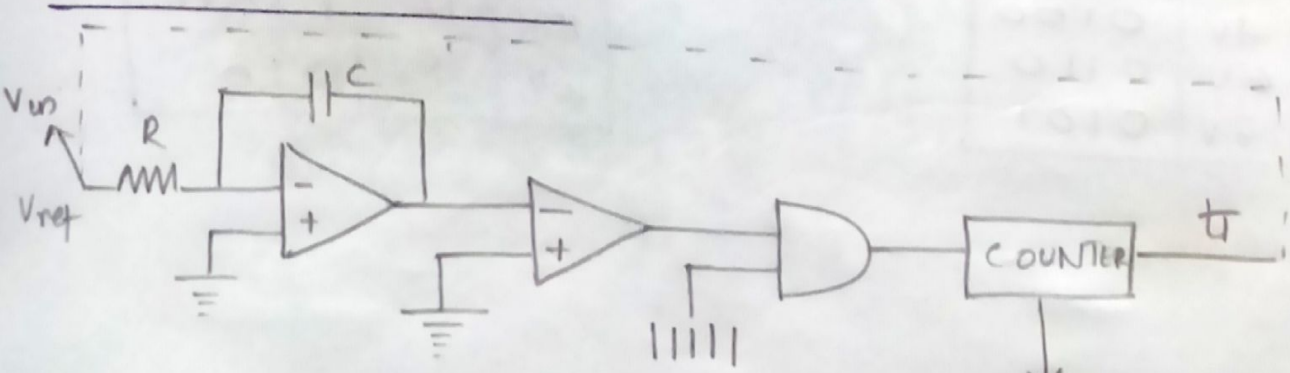


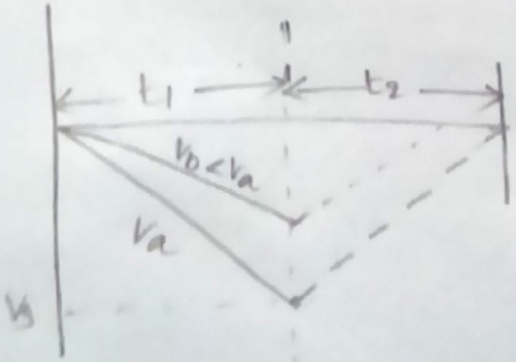






DUAL SLOPE





$$V_o = \frac{-V_{in}}{RC} \times t_1 = \frac{V_{ref}}{RC} \times t_2$$

constants

$V_{in} \times t_1 = V_{ref} \times t_2$ → variable

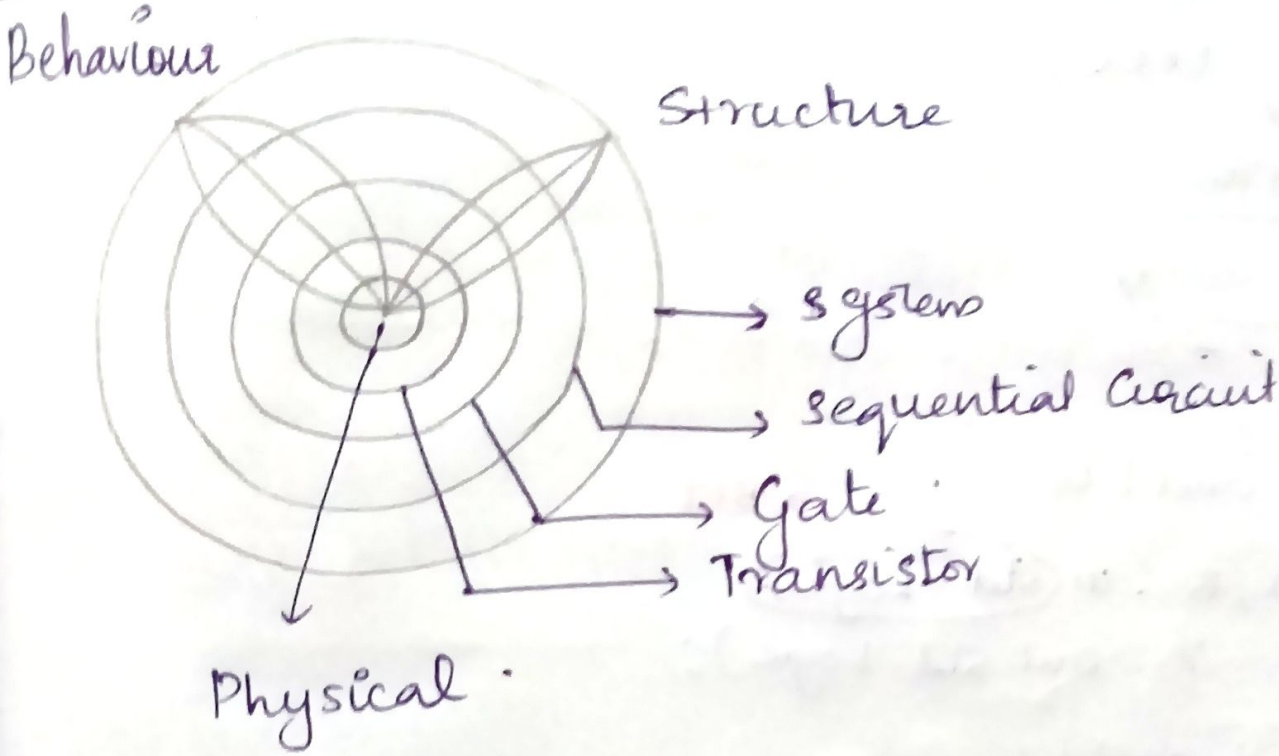
$V_{in} \propto t_2$

VHDL

VHSIC Hardware Design Language.

Very high Scale Integrated Circuit.

GAJSKI KUHN Y-CHART



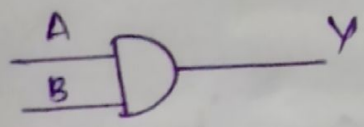
1) Entity level – Name plate — i/p, o/p
2) Architecture level
3) Configuration
4) Package declaration
5) Package body.

ARCHITECTURE LEVEL → Behaviour } Large ckt
→ Data flow
→ Structural } Smaller size ckt

CONFIGURATION

```
library ieee;
```

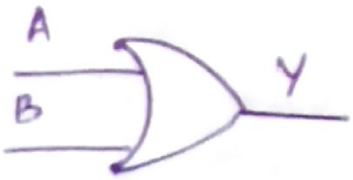
AND GATE




```
entity and1 is
Port (A, B : in std_Logic;
      Y : out std_Logic);
end and1;
architecture behaviour1 is
begin
Y <= A and B
end behaviour1;
```

BIT

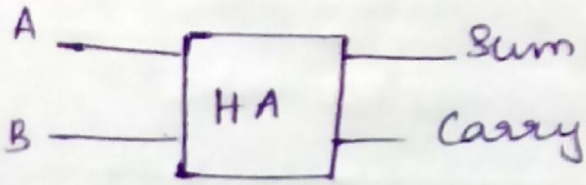
OR GATE

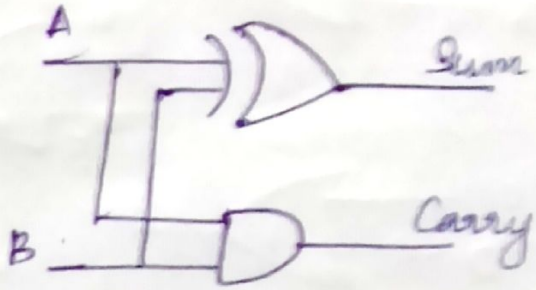



```
entity or1 is
Port (A, B: in Std-Logic;
      Y : out std-Logic;

end or1;
architecture behaviour1 is
begin
Y <= A or B
end behaviour1;
```

Half adder





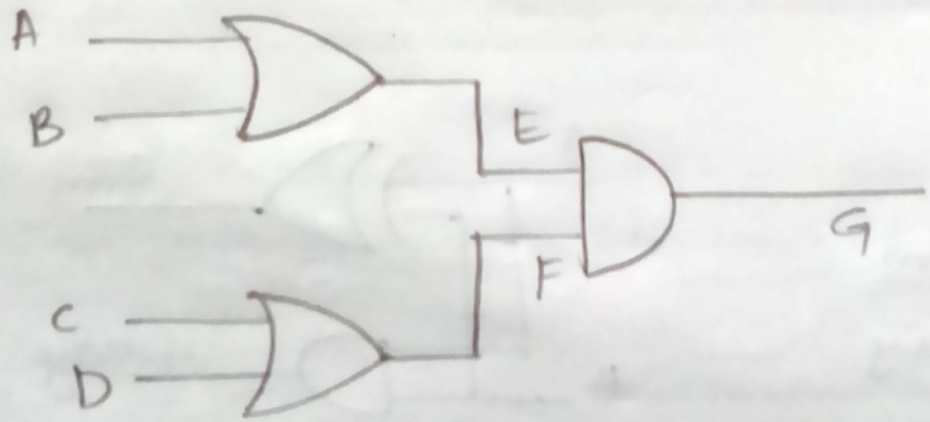

```
entity hay adder1 is
Port (A, B: in BIT;
      SUM, CARRY : OUT BIT);
end half-adder1;
architecture structural-HA of half adder1 is
Component XOR1 is
port (A, B : in BIT;
```

architecture structural_HA of halfadder is

Component and1 is
port (A, B : in BIT;
Carry : out BIT);
end component;

begin
x1 : XOR1 (A, B, Sum);
x2 : AND1 (A, B, Carry);
end structural HA;

Q. Write the VHDL code for:



Ans:

```
entity ckt1 is
Port (A, B, C, D : In BIT;
      G : Out BIT);
end ckt1;
architecture structural 1 of ckt1 is
component OR1 is
Port (X, Y : In BIT;
      Z : Out BIT);
end component;
Component NAD1
Signal E, F : BIT;
begin (Instances).
X1 : OR1  port map (A, B, E)
X2 : OR1  port map (C, D, F)
X3 : AND1 port map (E, F, G)
end structural 1
```

PLD (PROGRAMMABLE LOGIC DEVICE)

→ PAL (Programmable array Logic)

- Program AND gate
- program OR gate

→ PLA (Programmable Logic Array)

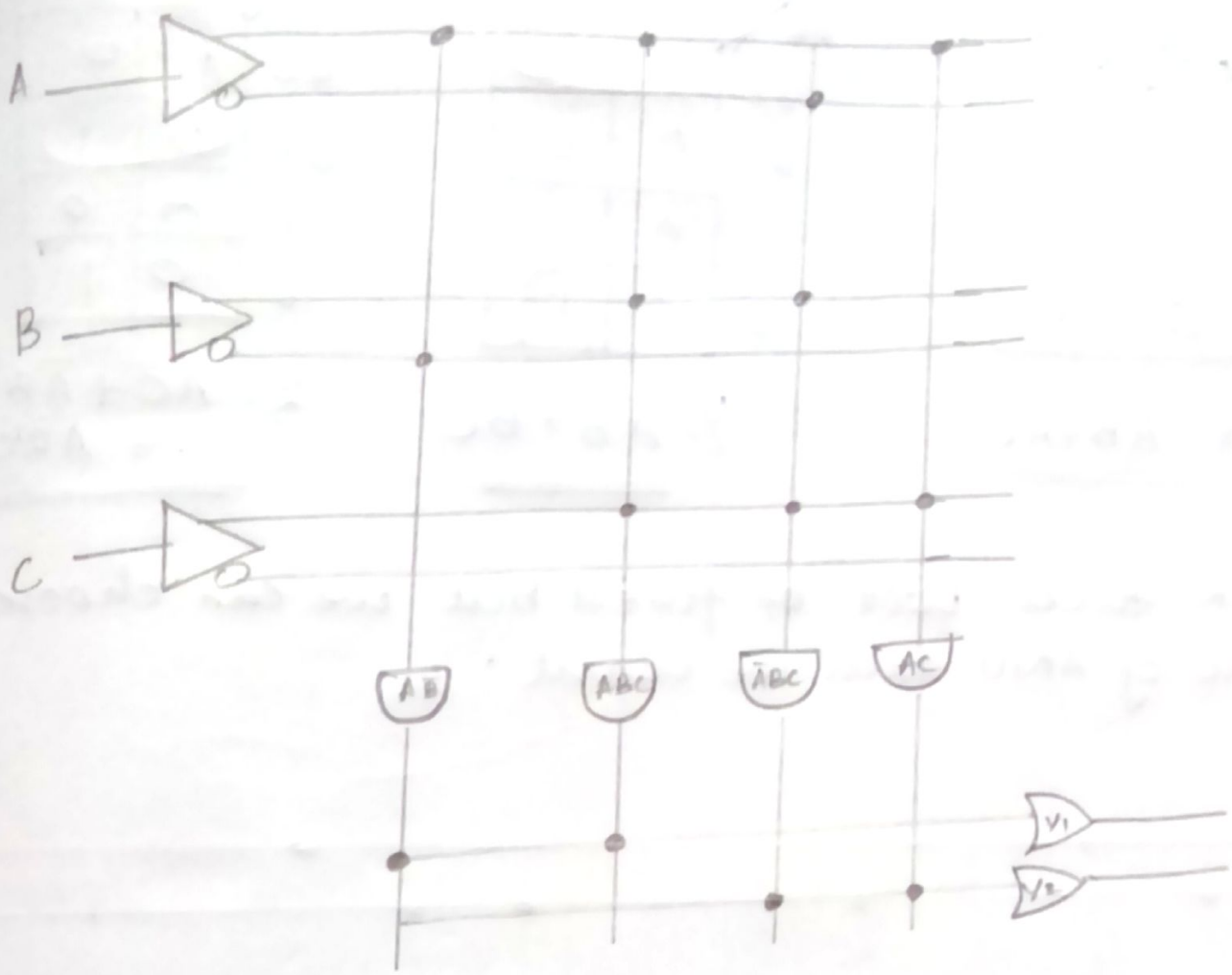
- Program AND gate
- Fixed OR gate

PAL

A	B	C	Y_1	Y_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

$Y_1 = A\bar{B} + A\dot{B}C$

$Y_2 = \bar{A}BC + AC$



PLA

$X = \sum m(2, 3, 5, 7)$

$Y = \sum m(0, 1, 5)$

$Z = \sum m(0, 2, 3, 5)$

AB \ C	0	1
00	0	0
01	1	1
11	0	1
10	0	1

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

AB \ C	0	1
00	1	1
01	0	0
11	0	0
10	0	1

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

AB \ C	0	1
00	1	0
01	1	1
11	0	0
10	0	1

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

⇒ The OR gates will be fixed but we can choose the no of AND gates we want.