Switching Theory and Logic Design

Prepared by:
Y.Pavan Kumar Reddy& M.Tejaswi
Dept of ECE
Assisitant Professor
Annamacharya University
Rajampet

SWITCHING THEORY AND LOGIC DESIGN

Unit 1: Number systems, Codes & Boolean Algebra

Philosophy of number systems – r, (r-1)‟s complement, representation of negative numbers, binary arithmetic, binary codes, error detecting & error correcting codes, hamming codes. Boolean algebra: Fundamental postulates of Boolean algebra, Basic theorems and properties, digital logic gates, properties of XOR gate, universal gates.

Unit 2 Switching Functions and their Minimization

Switching Functions-Canonical and Standard forms, algebraic simplification using Boolean theorems, two level & Multilevel Realization of Boolean Functions using Universal Gates. Minimization: K-Map methods, Prime Implicants, don't care combinations, Minimal SOP and POS forms, Tabular Method, Prime-Implicants chart, simplification rules.

Unit 3 Combinational Logic Design & Programmable Logic Devices

Design using conventional logic gates-Binary Adders, Subtractors, Ripple Adder, Magnitude comparator, Encoder, Decoder, Multiplexer, De-Multiplexer, Code converters,PLA, PAL,ROM AND COMPARISION OF PAL,PLA,ROM

Unit 4 Sequential Circuits

Classification of sequential circuits (Synchronous, Asynchronous, Pulse mode, Level mode with examples), Basic flip-flops, Triggering and excitation tables, flip flop conversions, Steps in synchronous sequential circuit design, Design of modulo-N Synchronous counters – up/down counter.

Unit 5 FSM Minimization

Finite state machine- capabilities and limitations, Mealy and Moore models and their conversions, Serial binary adder. Minimization of completely specified sequential machines-Partition techniques. . Salient features of the ASM chart, Simple examples.

Y. pavan Kumar Reddy
9848036672

UNIT-I

NUMBER SYSTEM & BOOLEAN ALGEBRA

Introduction to Analog & Digital Signals & Systems :-

→ The signals may be broadly classified into following two categories:

(i) Analog Signals
(ii) Digital Signals

(i) Analog Signals :-

→ Analog signals are the signals which may have infinite no of different magnitudes or values. They vary continuously with time.

Ex:- Sine wave, triangular wave etc

(ii) Digital Signals :-

→ A Signal is known as a digital signal if it has only a finite no of predetermined distint magnitudes.

Ex:- Binary, Hexadecimal etc

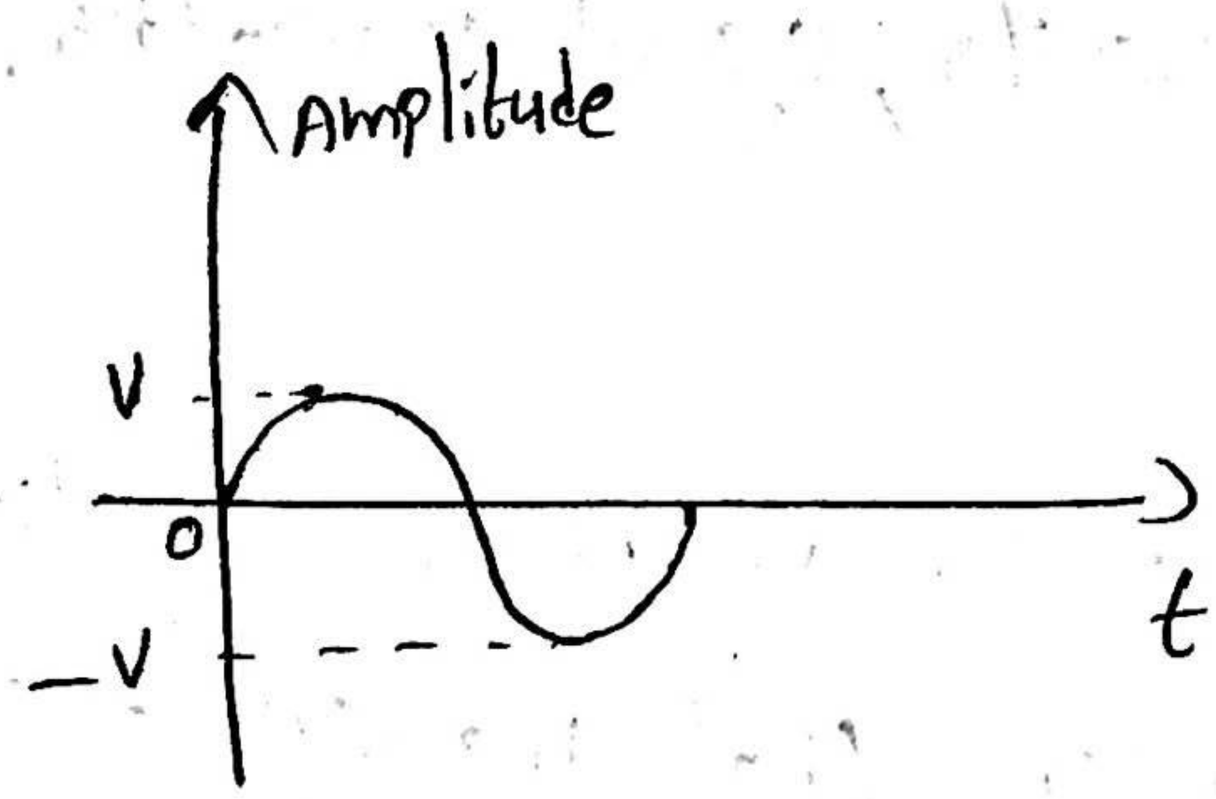


Fig (a): Analog Signal

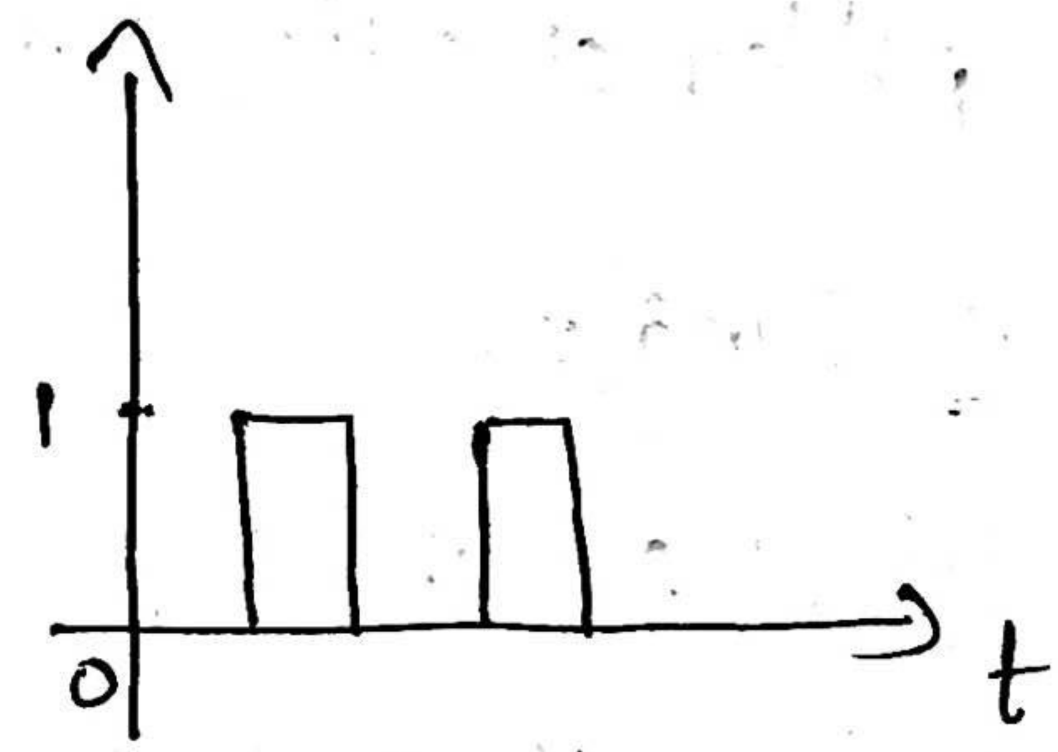


Fig (b): Digital Signal

performance comparison of Digital & Analog signals :-

parameter of comparison	Analog signals	Digital signals
No of values	Infinite	Finite (2, 8, 16 etc)
Nature of signals	continuous-time	Discrete-time
Source of signals	Signal generators, transducers etc	computers, A to D converters etc
Examples	Sine, triangular waves	Binary signal

* Analog signals vary gradually and continuously, while digital Signals produce discrete voltage levels commonly reffered as "High" and "Low".

Analog systems :-

→ The systems which process the analog signals are known as the Analog systems.

Ex:- Filter circuits, Amplifier circuits, signal generators.

Digital systems :-

→ A digital system may be defined as an interconnection of digital modules and it is a system that manipulates discre[te] elements of information that is represented internally in the binary form.

Y. pavan Kumar Reddy

Salient features of Digital systems :-

-> Digital systems manipulate discrete elements of information.

-> Discrete elements are nothing but the digits such as 10 decimal digits, 26 letters of alphabets and so on.

-> Digital systems make use of physical quantities called signals to represent discrete elements.

-> In digital systems, the signals have two discrete value and are therefore said to be binary.

-> A signal in digital system represents one binary digit, called a "bit." The bit has a value either '0' or '1'.

-> Discrete elements of information are represented with group of bits called "binary codes".

Advantages of Digital systems :-

(i) Easy to Design.

-> Digital systems are quite easy to design than analog systems, because digital design involves logic design which does not require special maths skills and its behaviour may be visualized part-by-part.

(ii) Reliability and Reproducibility of Results :-

-> The o/p of analog systems varies with the variation in temperature, power supply voltages, component-aging and several other factors. Therefore, it is difficult to reproduce the same result, each time with same set of i/ps and circuit components. However, this is not in case with digital systems.

→ Digital systems always reproduce exactly same results with same set of i/ps and circuits components.

(iii) Flexibility:-

→ Digital systems are more flexible to design since their design involves a set of logical steps and have various discrete and integrated options avialable to the user.

(iv) Functionality :-

→ It is easy to provide additional functionality in digital systems than analog systems.

(v) programmability :-

→ Nowadays, digital design is carried out by writing programs, in "hardware description language" (HDL). These languages allow us to simulate and test the performance of digital circuits.

→ This feature is quite useful in designing critical digital systems for various applications.

(vi) Device speed :-

→ Digital devices are very fast and they can produce 500 millions results. since their operating speed is smaller than 2 nano seconds.

Y. pavan kumar Reddy

(vii) Economy in manufacturing :-

-> Digital circuits can easily be integrated into a single chip and can be produced in larger quantity to have smaller cost compared to discrete circuits.

viii Upgrading Technology :-

-> As digital technology is becoming more and more popular, more ~~reason~~ research is going on in this field. Therefore, technological upgrade is expected in the digital world and we know that after about every six months, there is a launching of a new processor, new memory technology etc.

Comparison of Analog and Digital systems :-

S.NO	parameter of comparison	Analog systems	Digital systems
1	Type of signals processed	Analog signals	~~DH~~ Digital signals
2	Type of display	Analog meter	Digital displays using LED and LCD
3	Accuracy	Small	High
4	Design complexity	Difficult to design	Easier to design
5	Memory Attached	No memory	They have memory

6	Noise effect	High	Small
7	Distortion	High	Small
8	Effect of temperature & aging on performance	High	Small
9	communication b/w systems	Not easy	Easy
10	Examples	Filters, amplifiers, power supplies	counters, registers, microprocessors.

Number Systems :-

→ Number system is a basis for counting various items.

→ A number system is nothing more than a code that uses symbols to refer to a no of items.

→ In any number system there is an ordered set of symbols known as "digits".

→ Basically two types of number systems are present. They are ① positional number systems
② NON-positional number systems.

→ In positional number system, the position of each digit of a number indicates the significance to be attached to that digit. Ex:- Decimal number system.

→ In non-positional number system, a digit of a number does not indicate any significance of its position & weight.

Ex:- Roman numerical system.

→ The most widely used number system is positional number system.

→ In this system, the digits in a number are placed side by side and each position in the number is assigned a "weight" or "index" of importance by some predesigned rule.

→ The "base or radix" decides the total number of digits (unique) available in that number system.

→ Decimal Number System :-

→ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

→ Radix or Base = 10

→ Ex:- $(578)_{10}$

→ Binary Number System :-

→ 0 & 1

→ Radix or Base = 2

→ Ex $(1011)_2$

→ octal Number system :-

→ 0, 1, 2, 3, 4, 5, 6, 7.

→ Base or Radix = 8

→ Ex $(573)_8$

Hexadecimal :-

→ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

→ Base or Radix = 16

→ Ex $(1AC2)_{16}$

Binary Number system :-

The Bit :- The smallest unit of data is defined as a single "bit".

The Nibble :- A "Nibble" is a combination of "4 bits".

The Byte :- A "Byte" is a combination of "8 bits".

Relation b/w Decimal, Binary, octal & Hexadecimal Numbers :-

Decimal	Binary	octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5

Decimal	Binary	octal	Hexadecimal
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

~~Nt~~

→ **Number base conversions :-**

→ **conversion from any Number system to Decimal Number system :-**

→ Any number can be converted into its decimal equivalent by adding the products of each digit and its weight.

Ex ① convert $(159)_r$ into decimal

~~Sol~~ $(159)_r = (1\times r^2 + 5\times r^1 + 9\times r^0)_{10}$

Ex ② convert $(573\cdot16)_r$ in to decimal

$(573\cdot16)_r = (5\times r^2 + 7\times r^1 + 3\times r^0 + 1\times r^{-1} + 6\times r^{-2})_{10}$

* ~~Binary to decimal conversion~~ :-

→ Conversion of general example

→ convert $(A_{n-1} A_{n-2} A_{n-3} A_{n-4} \cdots A_2 A_1 A_0 \cdot A_{-1} A_{-2} A_{-3} \cdots A_{-m})_r$ in to decimal

Sol
Let $(N)_{10}$ be the decimal equivalent of given number.

$$(N)_{10} = \left(A_{n-1} r^{n-1} + A_{n-2} r^{n-2} + A_{n-3} r^{n-3} + \cdots + A_2 r^2 + A_1 r^1 + A_0 r^0 + A_{-1} r^{-1} + A_{-2} r^{-2} + \cdots + A_{-m} r^{-m}\right)_{10}$$

Binary to decimal conversion :-

Ex:- convert ~~826~~ $(101101)_2$ into decimal.

Sol
$$(101101)_2 = \left(1\cdot 2^5 + 0\cdot 2^4 + 1\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0\right)_{10}$$
$$= (32+0+8+4+0+1)_{10}$$
$$(101101)_2 = (45)_{10}$$

Ex:- convert $(101\cdot 11)_2$ into decimal.

Sol
$$(101\cdot 11)_2 = \left(1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 + 1\cdot 2^{-1} + 1\cdot 2^{-2}\right)_{10}$$
$$= (4+0+1+0\cdot 5+0\cdot 25)_{10}$$
$$(101\cdot 11)_2 = (5\cdot 75)_{10}$$

octal to Decimal conversion:-

Ex:- convert $(475.25)_8$ to decimal

Sol

$$(475.25)_8 = (4\times8^2 + 7\times8^1 + 5\times8^0 + 2\times8^{-1} + 5\times8^{-2})_{10}$$

$$= (256+56+5+0.25+0.078125)_{10}$$

$$(475.25)_8 = (317.32825)_{10}$$

Hexadecimal to decimal:-

Ex:- convert $(3A.2F)_{16}$ to its decimal equivalent.

Sol

$$(3A.2F)_{16} = (3\times16^1 + 10\times16^0 + 2\times16^{-1} + 15\times16^{-2})_{10}$$

$$(3A.2F)_{16} = (58.1836)_{10}$$

other Radix(r) system-to-decimal conversion:-

Ex:- convert $(234)_4$ into decimal system.

Sol

$$(234)_4 = (2\times4^2 + 3\times4^1 + 4\times4^0)_{10}$$

$$= (2\times16 + 3\times4 + 4\times1)_{10}$$

$$= (32+12+4)_{10}$$

$$(234)_4 = (\cancel{48}) \text{ to } 46)_{10}$$

→ other Radix(r) System ~~to~~ from Decimal conversion:-

(or)

Decimal system to-other radix(r) system:-

→ The conversion from decimal to any other radix(r) system is done differently for integer part ~~a~~ and fractional part.

→ The integer part of a decimal number can be converted into its equivalent radix (r) number system by successively dividing by 'r', till the final remainder becomes less than 'r' or equal to zero.

→ The number in r-number system can be obtained by writing all the remainders of each division from bottom to top.

→ For fractional part of the decimal number, the conversion is done successively multiplying by 'r' and keeping the track of the integers (carry) generated.

Ex:- convert $(724.63)_{10}$ into radix 4 & radix 6 number systems.

Sol $(724.63)_{10} = (\quad)_4$

$$\begin{array}{r|ll} 4 & 724 & \\ 4 & 181 & -0 \quad \text{LSD} \\ 4 & 45 & -1 \\ 4 & 11 & -1 \\ 4 & 2 & -3 \\ & 0 & -2 \\ & \text{MSD} & \end{array}$$

$0.63 \times 4 = 2.52 \quad -2$

$0.52 \times 4 = 2.08 \quad -2$

$0.08 \times 4 = 0.32 - 0$

$\therefore (724.63)_{10} = (23110.220)_4$

$(724.63)_{10} = (\ ?\ )_6$

P.T.O

6	724	
6	120	4
6	20	0
6	3	2
	0	3

$0.63 \times 6 = 3.78 \rightarrow 3$
$0.78 \times 6 = 4.68 \rightarrow 4$
$0.68 \times 6 = 4.08 \rightarrow 4$

$$(724.63)_{10} = (3204.344)_6$$

Decimal to binary conversion :-

Ex: → Convert the decimal number $(26.85)_{10}$ into binary.

Sol :-

2	26	
2	13	0 → LSB
2	6	1
2	3	0
2	1	1
	0	1 → MSB

$0.85 \times 2 = 1.7 \rightarrow 1$
$0.70 \times 2 = 1.4 \rightarrow 1$
$0.4 \times 2 = 0.8 \rightarrow 0$
$0.8 \times 2 = 1.6 \rightarrow 1$
$0.6 \times 2 = 1.2 \rightarrow 1$
$0.2 \times 2 = 0.4 - 0$
$0.4 \times 2 = 0.8 - 0$

$$(26.85)_{10} = (11010.1101100)_2$$

→ Successive division is used for integer part conversion.
→ Successive multiplication is used for fractional part conversion.

Decimal to octal conversion :-

→ convert the decimal number 35.45 to octal number.

Sol $(35.45)_{10} = (?)_8$

$$\begin{array}{r|l} 8 & 35 \\ \hline 8 & 4 - 3 \uparrow \\ \hline & 0 - 4 \end{array}$$

$(35)_{10} = (43)_8$

$0.45 \times 8 = 3.6 \rightarrow 3$
$0.6 \times 8 = 4.8 \rightarrow 4$
$0.8 \times 8 = 6.4 \rightarrow 6$
$0.4 \times 8 = 3.2 \rightarrow 3$
$0.2 \times 8 = 1.6 \rightarrow 1$

$(0.45)_{10} = (0.34631)_8$

$(35.45)_{10} = (43.34631)_8$

Decimal to Hexadecimal conversion :-

→ convert $(95.5)_{10}$ in to hexadecimal number

Sol

$$\begin{array}{r|l} 16 & 95 \\ \hline 16 & 5 - 15 \rightarrow F \uparrow \\ \hline & 0 - 5 \end{array}$$

$0.5 \times 16 = 8.0 \rightarrow 8$

$(95.5) = (5F.8)_{16}$ or $(5F.8)H$

Binary to Hexadecimal conversion :-

→ convert $(101101010100111)_2$ in to hexadecimal number.

Sol

$\underbrace{0101}_{5}\,\underbrace{1010}_{A}\,.\,\underbrace{1010}_{A}\,\underbrace{0111}_{7}$

$(0101101010100111)_2 = (5A.A7)_{16}$

→ Ex convert $(101101 \cdot 100110)_2$ into hexadecimal.

Sol

$\overline{0010}\,\overline{1101} \cdot \overline{1001}\,\overline{1000}$

2 D · 9 8

$(101101 \cdot 100110)_2 = (2D \cdot 98)_{16}$

Hexadecimal to Binary conversion :-

→ Ex convert the number $(7AC2 \cdot 1B6)_{16}$ into Binary.

Sol :

$(7AC2 \cdot 1B6)_{16} = (\ \ )_2$

7 A C 2 · 1 B 6

0111 1010 1100 0010 · 0001 1011 0110

$(7AC2 \cdot 1B6)_{16} = (0111101011000010 \cdot 000110110110)_2$

Binary to octal conversion :-

→ Ex - convert $(10110 \cdot 010011)_2$ into octal.

Sol :-

$\overline{010}\,\overline{110} \cdot \overline{010}\,\overline{011}$

2 6 · 2 3

$(10110 \cdot 010011)_2 = (26 \cdot 23)_8$

octal to Binary conversion :-

→ Ex :- convert $(37 \cdot 24)_8$ into Binary

Sol $(37 \cdot 24)_8 = (?)_2$

$$\begin{array}{cccc} 3 & 7 & \cdot\ 2 & 4 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 011 & 111 & \cdot\ 010 & 100 \end{array}$$

$$(37.24)_8 = (011111.010100)_2$$

Octal to Hexadecimal conversion :-

→ Ex: Convert $(32.164)_8$ into hexadecimal.

Sol

$$(32.164)_8 = (\,?\,)_{16}$$

$$\left.\begin{array}{cccccc} 3 & 2 & \cdot & 1 & 6 & 4 \\ \downarrow & \downarrow & \cdot & \downarrow & \downarrow & \downarrow \\ 011 & 010 & \cdot & 011 & 110 & 100 \end{array}\right\} \text{octal to Binary}$$

$$\left.\begin{array}{l} 00011010 \cdot 011110100000 \\ \;\;\;1\;\;\;\;A \;\;\;\;\;\;\; 7\;\;\;\;A\;\;\;\;0 \end{array}\right\} \text{Binary to Hex}$$

$(1A.7A0)_{16}$

$$(32.164)_8 = (1A.7A0)_{16}$$

Hexadecimal to octal conversion :-

→ Ex :- Conver $(2AC.12)_{16}$ into octal

Sol

$$\left.\begin{array}{cccc} 2 & A & C & \cdot\ 12 \\ \downarrow & \downarrow & \downarrow & \\ 0010 & 1010 & 1100 & \cdot\ 00010010 \end{array}\right\} \text{Hex to Binary}$$

$$\left.\begin{array}{l} 001010101100 \cdot 000100100 \\ \;\;1\;\;\;2\;\;\;5\;\;\;4 \;\;\cdot\;\; 0\;\;4\;\;4 \end{array}\right\} \text{Binary to octal.}$$

$$(2AC.12)_{16} = (1254.044)_8$$

→ Binary Arithmetic:

(i) Binary Addition:-

INPUTS		OUTPUTS	
A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

→ $\underset{\text{Sum}}{Y} = \underset{\text{Augend}}{A} + \underset{\text{Addend}}{B}$

→ Ex:- perform binary addition for the number $(10110)_2$ & $(1011011)_2$.

Sol

$$
\begin{array}{r}
0010110 \\
+\ 1011011 \\
\text{carry} \rightarrow 0111100 \\
\hline
(1110001)_2
\end{array}
$$

→ Ex:- perform binary addition b/w 55 & 32.

55 → 110111
32 → 100000

$$\begin{array}{rcr} 55 & \rightarrow & 110111 \\ +32 & \rightarrow & 100000 \\ & & 1000000 \\ \hline 87 & \rightarrow & (1010111)_2 \end{array}$$

Binary subtraction :-

I/PS		O/PS	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$Y = A - B$

Y → Difference, A → Minuend, B → Subtrahend

→ Exe perform binary subtraction b/w the numbers $(101101)_2$ & $(1111)_2$

Sol

$$\begin{array}{r} 101101 \\ -001111 \\ \text{Borrows} \rightarrow 111100 \\ \hline (011110)_2 \end{array}$$

→ Ex:- Perform binary subtraction b/w $(32.5)_{10}$ & $(17.75)_{10}$.

Sol

$(32.5)_{10} \rightarrow (100000.1)_2$

$(17.75)_{10} \rightarrow (10001.11)_2$

$$\begin{array}{rcr}
32.50 & \rightarrow & 100000.10 \\
-\ 17.75 & \rightarrow & 010001.11 \\
\hline
14.75 & \rightarrow & (001110.11)_2
\end{array}$$

Binary Multiplication :-

→ Ex:- perform binary multiplication b/w $(1011)_2$ & $(10.1)_2$

Sol :-

$$\begin{array}{r}
1011 \times 101 \\
\hline
1011 \\
0000 \\
1011 \\
\hline
1110 \\
\hline
100001
\end{array}
\qquad
\begin{array}{r}
1011 \times 10.1 \\
\hline
1011 \\
0000 \\
1011 \\
\hline
1000 \\
\hline
(1101\ 1.1)_2
\end{array}$$

Binary division :-

→ Ex:- perform Binary division for the numbers 10110 % 10

Sol :-

$$\begin{array}{l}
10)10110(1011 \\
\quad\ \ 10 \\
\quad\ \ \overline{0011} \\
\qquad\ \ 10 \\
\qquad\ \overline{010} \\
\qquad\ \ \ 10 \\
\qquad\ \ \overline{00}
\end{array}$$

Complements of Numbers :-

→ complements are used in digital computers for simplifying the subtraction operation and for logical manipulation.

→ There are two type of complements for each 'base-r' system.

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

(1) Radix complement (or) r's complement :-

→ Radix complement of 'N' $= (r^n - N)$

where 'N' is a number with radix 'r'

'n' is no of digits in 'N'

Ex:- 10's complement, 2's complement etc

→ Ex(2) perform radix complement of $(23621)_{10}$.

Sol $10^5 - 23621 = (76379)_{10}$

(2) Diminished radix complement or (r-1)'s complement :-

→ Diminished radix complement of 'N' $= (r^n - 1) - N$

Ex:- 9's complement, 1's complement etc

→ Ex:- Find (r-1)'s complement of $(2576)_{10}$

Sol (r-1)'s (9's) complement of $(2576)_{10} = 10^4 - 1 - 2576$

$= (7423)_{10}$

Note :-

→ Radix complement can be found by adding 1 to diminished radix complement.

Decimal :-

→ Radix complement (or) 10's complement

→ Diminished radix complement or 9's complement

Ex:- ~~perfo~~ Find (radix) 10's complement & 9's complement of the number $(57921)_{10}$

Sol :-

9's complement of 57921 = 99999 − 57921

= 42078

10's complement of 57921 = 9's complement of 57921 + 1

= 42078 + 1

= 42079

~~Decimal~~

Binary

→ Radix complement (or) 2's complement

→ Diminished radix complement (or) 1's complement

→ Ex:- Find 2's & 1's complement of number $(101101)_2$.

Sol

1's complement of $(101101)_2 = 111111 - 101101$

$= (010010)_2$

2's complement of $(101101)_2$ = 1's complement of $(101101)_2 + 1$

$= 010010 + 1$

2's complement of $(101101)_2 = (010011)_2$

Note

→ 1's complement of a binary number can be found simply by replacing 0's with 1's and 1's with 0's in the given number.

octal

→ Radix complement or 8's complement

→ Diminished radix complement or 7's complement

Ex:- Find 8's & 7's complement of s $(6271)_8$.

Sol

7's complement of $(6271)_8 = 7777 - 6271$

$= (1506)_8$

8's complement of $(6271)_8$ = 7's complement of $(6271)_8 + 1$

$= 1506 + 1$

$= 1507$

Hexadecimal :-

→ Radix complement or 16's complement
→ Diminished radix complement or 15's complement

Ex:- Find 16's complement 15's complement of $(5A10)_{16}$

Sol :- 15's complement of $(5A10)_{16}$ = FFFF − 5A10

$= (A5EF)_{16}$

16's complement of $(5A10)_{16}$ = 15's complement of $(5A10)_{16} + 1$

$= A5EF + 1$

$= (A5F0)_{16}$

→ Subtraction using complements :-

* Subtraction using 1's complement :-

Case(i):- "Higher number - smaller number". i.e minuend > subtrahend
(or Larger)

procedure :-

1. Find 1's complement of the smaller number (sub-trahend).
2. Add this complement to Larger number.
3. If carry generated it is called as End Around Carry (EAC).
4. Remove the carry & Add it to the result.

→ Ex:- perform 1's complement subtraction for the following.

(i) $(36)_{10} - (17)_{10}$ (ii) $(45.25)_{10} - (19.5)_{10}$

Sol (i) $(36)_{10} \rightarrow (100100)_2$

$(17)_{10} \rightarrow (010001)_2 \xrightarrow[\text{comp}]{1's} 101110$

$$
\begin{array}{rl}
36 \rightarrow & 100100 \\
-17 \rightarrow & +101110 \\
\hline
19 & 1\;010010 \\
\text{EAC} & +1 \\
\hline
\end{array}
$$

$19 \rightarrow (010011)_2$

(ii) $(45.25)_{10} \rightarrow (101101.01)_2$

$(19.5)_{10} \rightarrow (010011.10)_2 \xrightarrow[\text{comp}]{1's} 101100.01$

$$
\begin{array}{rl}
45.25 \rightarrow & 101101.01 \\
-19.50 & +101100.01 \\
\hline
25.75 & 1\;011001\;10 \\
\text{EAC} & 011001.10 \\
 & +1 \\
\hline
\end{array}
$$

$(25.75)_{10} \rightarrow (011001.11)_2$

Case (ii):- Smaller − Larger, i.e Minuend < Subtrahend

procedure:-

1. Find 1's complement of Larger number (Subtrahend).
2. Add this complement to the smaller number (Minuend).
3. perform 1's complement to the sum to get original result.
4. If carry generated is '0' it is −ve number, if '1' it is +ve number.

→ Ex:- perform 1's complement subtraction for the following.

(i) $(57)_{10} - (93)_{10}$ (ii) $(23)_{10} - (35)_{10}$

Sol

(i) $(57)_{10} \longrightarrow (0111001)_2$

$(93)_{10} \longrightarrow (1011101)_2 \xrightarrow[\text{comp}]{\text{1's}} 0100010$

$$\begin{array}{rcr} 57 & \longrightarrow & 0111001 \\ -93 & \longrightarrow & +0100010 \\ \hline -36 & & 1000000 \\ \hline & & 1011011 \end{array}$$

$\downarrow$ 1's complement

$-(0100100)_2$

→ Here there is no carry. i.e carry is '0'. So it is −ve numbe

(ii) 23 → 010111

35 → 100011 $\xrightarrow{\text{1's comp}}$ 011100

$$\begin{array}{rcr} 23 & \rightarrow & 010111 \\ -35 & \rightarrow & +011100 \\ \hline -12 & & 111000 \\ & & \overline{110011} \end{array}$$

↓ 1's complement

$(001100)_2$

carry = 0 so it is −ve

$-(001100)_2$

Subtraction using 2's complement :-

Case (i): Larger num − Smaller num i.e Minuend > subtrahend.

procedure :-

1. Find the 2's complement of subtrahend (Smaller num).
2. Add this 2's complement to Minuend (Larger number).
3. Discard the carry generated.

Ex:- perform 2's complement subtraction for the following.

(i) $(45.25)_{10} - (19.50)_{10}$

Sol

45.25 → 101101.01

19.50 → 010011.10 $\xrightarrow[\text{comp}]{\text{1's}}$ 101100.01

+1

$\xrightarrow[\text{comp}]{\text{2's}}$ 101100.10

Y. pavan Kumar Reddy (14)

$$\begin{array}{rcr} 45.25 & \longrightarrow & 101101.01 \\ -19.50 & \longrightarrow & +101100.10 \\ & & 1\,011000\;\;00 \\ \hline 25.75 & \longleftarrow \text{Discard} & (011001.11)_2 \end{array}$$

Case ii :- Smaller − Larger i.e Minuend < Subtrahend.

procedure :-

1. Find the 2's complement of the subtrahend.
2. Add this 2's complement to minuend.
3. perform 2's complement to the result to get original result.
4. If carry generated is '0' it is −ve number. If '1' it is +ve number.

Ex :- perform 2's complement subtraction for the following.

1. 37.5 − 58.75

Sol

$$37.5 \longrightarrow 100101.10$$

$$\cancel{69.75}\ 58.75 \longrightarrow 111010.11 \xrightarrow{\text{1's comp}} 000101.00$$

$$\begin{array}{r} 000101.00 \\ +1 \\ \hline 000101.01 \end{array} \quad (\text{2's comp})$$

$$\begin{array}{rcr} 37.5 & \longrightarrow & 100101.10 \\ -58.75 & \longrightarrow & +000101.01 \\ & & 0\,00101000 \\ \hline -21.25 & & 101010.11 \end{array}$$

$$101010.11 \xrightarrow{\text{1's}} 010101.00$$

$$\begin{array}{r} 010101.00 \\ +1 \\ \hline 010101.01 \end{array}$$

$-21.25 \longleftarrow 010101.01$

∵ carry generated is zero, it is −ve number.

→ Representation of signed binary numbers:- 3 types

① Signed Magnitude Representation.
② Signed 1's complement representation.
③ Signed 2's complement representation.

① Signed Magnitude Representation:-

→ In signed magnitude representation an additional bit is added at MSB position (ie Left most position).

→ This bit is called sign bit. Sign bit is '0' for positive numbers and '1' for negative numbers.

Ex:-

+3 → 0011 (sign bit)

−3 → 1011 (sign bit)

+7 → 0111 (sign bit)

−7 → 1111 (sign bit)

+9 → 00001001

−9 → 10001001

② Signed 1's complement Representation:-

→ In signed 1's complement representation, representing of +ve numbers is same as in signed magnitude representation.

→ −ve numbers can be represented by performing 1's complement to +ve number.

Ex +9 → 10001001

-9 → 01110110

③ Signed 2's complement representation :-

→ In this +ve number representation is same as signed magnitude representation.

→ -ve numbers are obtained by performing 2's complement to +ve numbers.

Ex:-

+9 → 00001001

-9 → 01110111

→ convert all decimal numbers from +7 to -7 in-to Signed binary numbers.

Decimal Number	signed magnitude representation	Signed 1's complement representation	Signed 2's complement representation.
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000

−0	1000	1111	—
−1	1001	1110	1111
−2	1010	1101	1110
−3	1011	1100	1101
−4	1100	1011	1100
−5	1101	1010	1011
−6	1110	1001	1010
−7	1111	1000	1001

→ Binary Codes :-

→ The digital data is represented, stored & transmitted as group of binary digits (bits).

→ The group of bits also known as "binary code."

→ Binary codes represent numbers & letters of the Alphabets as well as many special characters & control functions.

Classification of Binary codes :-

Binary codes
- Weighted
  - Binary
  - BCD
    - 8421
    - 2421
    - 5211
    - 4221
    - 5421
    - etc
- Non-weighted
  - XS-3
  - Gray
  - 5-bit BCD
- Reflective
  - 2421
  - 5211
  - XS-3
- Sequential
  - 8421
  - XS-3
- Alphanumeric
  - ASCII
  - EBCDIC
  - Hollerith
- Error detecting & correcting
  - Parity
  - Hamming

① Weighted codes :-

-> In weighted codes, each digit position of the number represents a specific weight.

-> For example, in decimal code, if number is 256, then weight of 2 is 100, weight of 5 is 10 & weight of '6' is 1.

-> In weighted binary codes (8421) each digit has a weight 8, 4, 2 & 1.

-> 8421, 2421, 5211 etc are weighted codes.

② NON-weighted codes :-

-> NON-weighted codes are not assigned with any weight to each digit position.

-> Excess-3 (XS-3) & gray codes are NON-weighted codes.

③ Reflective codes :-

-> A code is said to be reflective when the code for 9 is the complement of code for 0, 8 for 1, 7 for 2, 6 for 3 & 5 for 4.

-> 2421, 5211 & XS-3 codes are reflective codes.

④ Sequential codes :-

-> In "sequential" codes, each succeeding code is one binary number greater than its preceding code.

-> 8421 & Excess-3 codes are sequential codes.

⑤ Alphanumeric codes :-

-> The codes which consists of both numbers & alphabetic characters are called "alphanumeric" codes.

→ ASCII & EBCDIC and Hollerith codes are Alphanumeric codes.

→ ⑥ Error Detecting and correcting codes :-

→ when the digital information in the binary form is transmitted from one circuit or system to another circuit or system, an error may occur.

→ This means that a signal corresponding to '0' may change to '1' or vice-versa due to presence of noise.

→ To maintain the data integrity b/w Tx'er and Rx'er, extra bit or more than one bit are added in the data.

→ These extra bits allows the detection & some times correction of error in the data.

→ The data along with extra bit/bits forms the "code."

→ Codes which allow only error detection are called "error detecting codes" and codes which allows error detection & correction are called "error detecting and correcting codes".

→ parity codes are error detecting codes.

→ Hamming code is error detecting & correcting code.

→ Comparison b/w weighted codes & Non-weighted codes :-

weighted codes	NON-Weighted codes.
→ In this each bit position is assigned a specific weight.	→ In this no specific weights are assigned to bit positions.
→ Ex:- BCD, 8421, 2421, 5211 etc	→ Excess-3, gray codes.

P.T.O

Y. pavan Kumar Reddy

→ **Applications**:- These are used in ⓐ Data manuplation during arithmetic operations. ⓑ For I/o operations in digital ckts. ⓒ To represent decimal digits in calculators, voltmeters etc.	→ **Applications**:- These are used ⓐ To perform certain arithmetic operations. ⓑ In Shift position Encoders. ⓒ For error detecting purpose.

Binary coded Decimal :- (BCD)

→ In BCD code each decimal digit is represented by seperate 4-bits.

→ The most commonly used BCD code is 8421 code.

8421 code :-

Decimal	8 4 2 1 code
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

properties :-

-> It is a weighted code
-> It is not self complementary code.
-> It is a sequential code.

-> 2421 code :-

Decimal	2 4 2 1 code
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	1 0 1 1
6	1 1 0 0
7	1 1 0 1
8	1 1 1 0
9	1 1 1 1

properties :-
-> It is self complementary
-> It is a weighted code

~~Decimal~~

-> 84-2-1 code :-

Decimal	8 4 -2 -1 code
0	0 0 0 0
1	0 1 1 1
2	0 1 1 0

P.T.O | cont....

Decimal	8 4 -2 -1 code
3	0 1 0 1
4	0 1 0 0
5	1 0 1 1
6	1 0 1 0
7	1 0 0 1
8	1 0 0 0
9	1 1 1 1

→ Excess-3 (XS-3) code :-

Decimal	Excess-3 code
0	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
4	0 1 1 1
5	1 0 0 0
6	1 0 0 1
7	1 0 1 0
8	1 0 1 1
9	1 1 0 0

→ Non-weighted code,

→ Reflective, sequential, self complementary

→ **Gray code :-**

Decimal	Gray code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101

Shortcut

	00	01	11	10
00	0	1	2	3
01	7	6	5	4
11	8	9	10	11
10	15	14	13	12

→ Non-weighted code

→ Not-self complementary

→ Reflective code. i.e code of 15 is complement of '0' & 14's code is complement of '1''s code ---.

→ **Binary to Gray code conversion :-**

Steps

1. MSB in gray code is same as MSB in binary.
2. Going from left to right add adjacent pair of bits to get next gray code bits.
3. Neglect all carries generated. (i.e X-OR the bits).

Y. pavan kumar

Ex:- convert the following numbers into Gray code.

(i) $(1011)_2$ (ii) $(527)_8$ (iii) $(5B6)_{16}$ (iv) $(38)_{10}$

Sol

(i) $(1011)_2$

Binary 1 0 1 1

Gray 1 1 1 0

$(1011)_2 \longrightarrow 1110$ (Gray)

(ii) $(527)_8 = (101010111)_2$

1 0 1 0 1 0 1 1 1

1 1 1 1 1 1 1 0 0

~~101~~ $(527)_8 = 111111100$ (Gray)

(iii) $(5B6)_{16} = (010110110110)_2$

0 1 0 1 1 0 1 1 0 1 1 0 (Binary)

0 1 1 1 0 1 1 0 1 1 0 1 (gray)

(iv) $(38)_{10} \longrightarrow (100110)_2$

100110 (Binary)

$(38)_{10} = 110101$ (gray)

Y. pavan Kumar Reddy

Boolean Algebra :-

→ Boolean Algebra is a set of rules, laws & theorems by which logical operations can be mathematically expressed.

→ It is also known as "Switching Algebra".

→ It is a convenient way of expressing the operation in digital circuits & systems.

Boolean operations :-

① AND operation :-

0 AND 0 = 0	$0 \cdot 0 = 0$
0 AND 1 = 0	$0 \cdot 1 = 0$
1 AND 0 = 0	$1 \cdot 0 = 0$
1 AND 1 = 1	$1 \cdot 1 = 1$

→ '.' is sign for AND operation.

② OR operation :-

0 OR 0 = 0	$0 + 0 = 0$
0 OR 1 = 1	$0 + 1 = 1$
1 OR 0 = 1	$1 + 0 = 1$
1 OR 1 = 1	$1 + 1 = 1$

→ '+' is sign for OR operation.

(3) NOT operation :-

NOT 0 = 1 $\quad\quad \overline{0} = 1$

NOT 1 = 0 $\quad\quad \overline{1} = 0$

→ Boolean Laws :-

(1) AND Laws:

$$A \cdot 0 = 0$$
$$A \cdot 1 = A$$
$$A \cdot \overline{A} = 0$$
$$A \cdot A = A$$

(2) OR Laws :-

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

(3) NOT Laws :-

$$\overline{\overline{A}} = A$$

(4) Commutative Law :-

$$A + B = B + A$$
$$A \cdot B = B \cdot A$$

(5) Associative laws:-

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

$$A\cdot(B\cdot C) = (A\cdot B)\cdot C = A\cdot B\cdot C$$

(6) Distributive laws:-

$$A\cdot(B+C) = A\cdot B + A\cdot C$$

$$A+(B\cdot C) = (A+B)\cdot(A+C)$$

(7) Auxillary laws:-(Absorption Law)

→ (i) $A+A\cdot B = A$

→ (ii) $A+\bar{A}B = A+B$

proof (i) $A+A\cdot B = A$

LHS $= A+A\cdot B$

$= A(1+B)$ $\because 1+A=1$

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

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

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

From distributive law

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

$= 1(A+B)$ $\because 1\cdot A = A$

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

Basic Theorems :-

① **Consensus theorem :**

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

proof

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

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

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

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

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

$$AB+\bar{A}C+BC = AB+\bar{A}C = RHS \qquad \because 1\cdot A = A$$

② $(A+B)\cdot(\bar{A}+C)(B+C) = (A+B)(\bar{A}+C)$

proof

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

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

$$= A\bar{A}B+A\bar{A}C+ABC+AC+\bar{A}BB+\bar{A}BC+BBC+BC$$

$$= 0+0+ABC+AC+\bar{A}B+\bar{A}BC+BC+BC$$

$$= ABC+AC+\bar{A}B+\bar{A}BC+BC$$

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

$$LHS = BC+AC+\bar{A}B$$

Y. pavan Kumar Reddy 28

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

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

$RHS = AC + \bar{A}B + BC$

$LHS = RHS$

→ Transportation Theorem :-

→ $AB + \bar{A}C = (A+C)(\bar{A}+B)$

proof

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

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

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

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

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

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

$= AB + \bar{A}C$

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

→ Demorgan's Theorem :-

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

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

Note

positive logic :-

→ More +ve voltage is represented by '1'

→ More -ve voltage is represented by '0'

–ve logic :-

→ More +ve voltage is represented by '0'
→ More –ve voltage is represented by '1'

→ Duals :-

→ Duality of a Boolean function is obtained by replacing "OR" operator by "AND" operator & "AND" operator by 'OR' operator & '1' is replaced by '0' & '0' is replaced by '1'.

→ The implicant of duality concept is that once a theorem or statement is proved, the dual also thus stands proved. This is called the principle of duality.

Given expression	Dual
$\overline{0} = 1$	$\overline{1} = 0$
$0 \cdot 1 = 0$	$1 + 0 = 1$
$0 \cdot 0 = 0$	$1 + 1 = 1$
$1 \cdot 1 = 1$	$0 + 0 = 0$
$A \cdot 0 = 0$	$A + 1 = 1$
$A \cdot A = A$	$A + A = A$
$A \cdot B = B \cdot A$	$A + B = B + A$
$A \cdot (B + C) = AB + AC$	$A + (B \cdot C) = (A + B) \cdot (A + C)$
$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$

Ex:- Find ~~comple~~ the duality of the function $F=(x+y)\cdot(xyz)$

Sol

$$F=(x+y)(xyz)$$

$$F_d=(x\cdot y)+(x+y+z)$$

→ Complement of Boolean function:-

Steps

① Find out the duality of given boolean function.

② Take ~~the~~ complement of each variable (literal) in the dual.

Ex:- Find the complement of boolean function $F(x,y,z)=(x+y+z)(y'+z')$

Sol

$$F(x,y,z)=(x+y+z)(y'+z')$$

$$F_d(x,y,z)=(xyz)+(y'z')$$

$$F'(x,y,z)=(x'y'z')+(yz)$$

→ Simplification of Boolean function using Law, Theorems of Boolean Algebra:-

Ex:- Find out the simplified expression of the function $F(x,y,z)=xy+yz+x'z$

Sol

$$F=xy+yz+x'z$$

$$=xy+1yz+x'z \quad (\because A\cdot 1=A)$$

$= xy + x'z + yz(x+x')$ $\quad \because x+x'=1$

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

$= xy(1+z) + x'z(1+y)$ $\quad \because 1+A=1$

$F = xy + x'z$

→ Reduce the following boolean function into 3 literals.

$F(A,B,C) = A'C' + ABC + AC'$

Sol

$F(A,B,C) = A'C' + ABC + AC'$

$= C'(A+A') + ABC$ $\quad \because A+A'=1$

$= C' \cdot 1 + ABC$ $\quad \because A \cdot 1 = A$

$= \underset{z}{C'} + \underset{x\ y}{\underline{AB}C}$

$= (C+C')(C'+AB)$ $\quad \because A+BC = (A+B)(A+C)$

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

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

→ Reduce the following boolean functions

(i) $F = (BC' + A'D)(AB' + CD')$ into '0' literals.

(ii) $F = A'B(D' + C'D) + B(A + A'CD)$ into '2' literal.

Sol (i)

$F = (BC' + A'D)(AB' + CD')$

$= BC' \cdot AB' + BC'CD' + A'DAB + A'DCD'$

$= 0+0+0+0$

$= 0$

$$F = A'B(D' + C'D) + B(A + A'CD)$$

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

$$= B[A'D(C + C') + A'D' + \cancel{A'C'D}\, A]$$

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

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

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

$$\boxed{F = B}$$

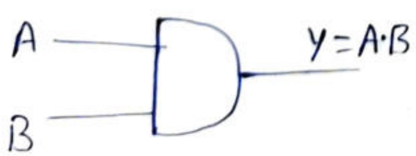
→ Logic gates :-

AND gate :-

→ The AND gate has two or more i/ps but only one o/p.

→ The o/p assumes the logic 1 state, when all i/ps are at logic '1'.

→ AND gate also known as "all or nothing gate".



Logic symbol

A	B	Y=AB
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

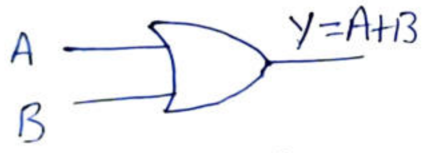
OR gate :-

-> The "OR" gate has two or more i/ps but only one o/P.

-> The o/p assumes the logic 1 state, even if one of its i/p is in logic '1' state.

-> The o/p assumes the logic '0', when all i/ps are at logic '0' state.

-> OR gate is also called as "any gate"



2 i/p OR gate Logic Symbol

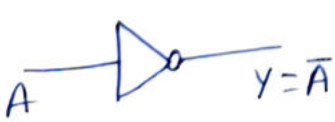
A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Truth table.

NOT gate

-> NOT gate also called as an "inverter"

-> It has one i/p. & one o/p only.



Logic symbol

A	Y
0	1
1	0

Truth table

Note

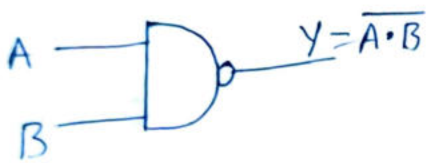
→ The AND, OR, NOT gates are "basic gates"

Universal gates.

→ NAND gate, NOR gate are universal gates.

NAND gate :-

→ The NAND gate has two or more i/ps and only one o/p.

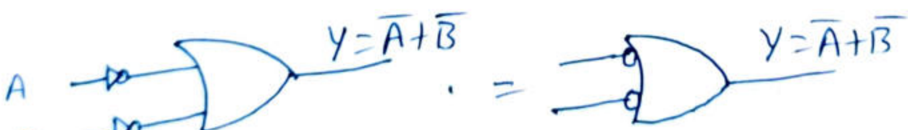


Logic symbol

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

Truth table.

Bubbled OR gate :-



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

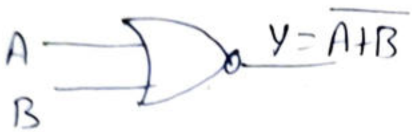
Truth table

→ The truth table of bubbled OR gate & NAND gate are same

$$\therefore \boxed{\bar{A}+\bar{B} = \overline{A \cdot B}}$$

NOR gate :-

→ The NOR gate has two or more i/ps and only one o/p.

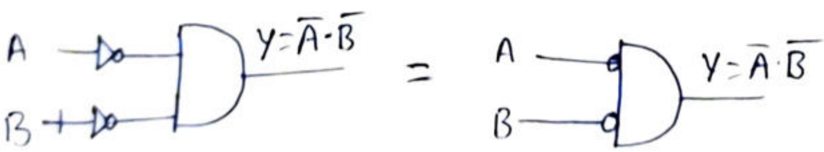


Logic symbol

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

Truth table.

Bubbled AND gate :-



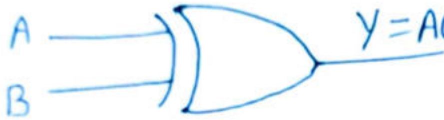
A	B	$\bar{A}$	$\bar{B}$	$Y = \bar{A} \cdot \bar{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

→ The truth table of bubbled AND gate & NOR gate are same

So $\boxed{\bar{A}\cdot\bar{B} = \overline{A+B}}$

→ Exclusive – OR gate or X-OR gate :–

→ The X-OR gate has only two i/p & one o/p.



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

Logic symbol

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

Truth table

XOR gate as an inverter :–

→ An X-OR gate can be used as an inverter by connecting one of the two i/p terminals to logic '1' and feeding the i/p sequence to be inverted to the other terminal as shown in fig.

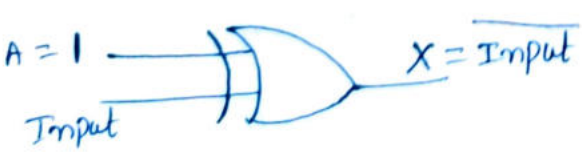


fig: X-OR as an inverter

A	Input	X = $\overline{\text{Input}}$
1	0	1
1	1	0

Truth table

→ properties of Exclusive-OR gate :-

① EX-OR as modulo 2 Adder

→ The X-OR gate can be used as a modulo 2 adder because its truth table is same as the truth table of modulo-2 adder.

$0+0 = 0$		$0 \oplus 0 = 0$
$0+1 = 1$	$=$	$0 \oplus 1 = 1$
$1+0 = 1$		$1 \oplus 0 = 1$
$1+1 = 0$		$1 \oplus 1 = 0$

② $A \oplus 0 = A$

③ $A \oplus 1 = \bar{A}$

④ $A \oplus A = 0$

⑤ $A \oplus \bar{A} = 1$

⑥ $(AB) \oplus (AC) = A(B \oplus C)$

⑦ If $A \oplus B = C$ then $A \oplus C = B$, $B \oplus C = A$ & $A \oplus B \oplus C = 0$

Exclusive-NOR (X-NOR) gate :-

→ The X-NOR gate has two i/p & one o/p only.

$Y = \overline{A \oplus B} = A \odot B = AB + \bar{A}\bar{B}$

Logic symbol

A	B	$Y = A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

Truth table of X-NOR gate.

→ Representation of the boolean functions :-

The boolean functions are represented by following 4 ways.

1. Canonical form
2. Standard form
3. Gate Implementation form (or) Logic diagram representation.
4. Truth table form.

① Canonical form :-

→ If the boolean function is in sum of minterms (or) product of maxterms form then the function is said to be in canonical form.

Minterms :-

→ The following table gives the three variable minterms & its designations.

Minterms	Designations
$x'y'z'$	m_0
$x'y'z$	m_1
$x'yz'$	m_2
$x'yz$	m_3
$xy'z'$	m_4
$xy'z$	m_5
xyz'	m_6
xyz	m_7

$$\left[\begin{array}{l} x' = 0 \\ x = 1 \end{array}\right]$$

Maxterms :-

→ The following table gives the three variable maxterms & its designations.

Max term	Designation
$x+y+z$	M_0
$x+y+z'$	M_1
$x+y'+z$	M_2
$x+y'+z'$	M_3
$x'+y+z$	M_4
$x'+y+z'$	M_5
$x'+y'+z$	M_6
$x'+y'+z'$	M_7

$x = 0$

$x' = 1$

Ex:-

$$F = x'y'z' + xyz + xyz'$$

or

$$F = (x+y+z)(x'+y'+z')(x'+y+z')$$

} canonical form

Sum of minterms form:-

$$F = x'y'z' + xyz + x'yz'$$

$$= m_0 + m_7 + m_2$$

$$F = \Sigma(0,2,7)$$

product of Maxterms:-

$$F = (x+y+z)(x'+y'+z')(x+y+z')(x'+y'+z)$$

$$F = M_0 \cdot M_7 \cdot M_1 M_6$$

$$F = \Pi(0,1,6,7)$$

→ Conversion of general boolean function into canonical form:-

→ Ex: Find out the canonical form of the function

$$F(x,y,z) = x + y'z$$

Sol Given function $F(x,y,z) = x + y'z$

Sum of minterms form

$$F = x \cdot 1 + y'z \cdot 1 \qquad \because A \cdot 1 = A$$

$$F = x(y+y') + y'z(x+x') \quad \because A+\bar{A}=1$$

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

$$= xy \cdot 1 + xy' \cdot 1 + xy'z + x'y'z$$

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

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

$$F(x,y,z) = xyz + xyz' + xy'z + xy'z' + x'y'z \quad \because A \cdot A = A$$

$$= m_7 + m_6 + m_5 + m_4 + m_1$$

$$F(x,y,z) = \Sigma(1,4,5,6,7)$$

<u>product of Maxterms</u>

$$F(x,y,z) = x + y'z$$

$$= (x+y')(x+z) \quad \because A+BC = (A+B)(A+C)$$

$$= (x+y'+0)(x+z+0) \quad \because A+0=A$$

$$= (x+y'+(z \cdot z'))(x+z+(y \cdot y')) \quad \because A \cdot \bar{A} = 0$$

$$= \underset{M_2}{(x+y'+z)}\,\underset{M_3}{(x+y'+z')}\,\underset{M_0}{(x+z+y)}\,\underset{M_2}{(x+y'+z)}$$

$$= M_2 \cdot M_3\, M_0$$

$$F(x,y,z) = \Pi(0,2,3)$$

Conversion within the canonical form :-

Statement ①:- The complement of the epre expression of sum of minterms is equal to sum of missing minterms from the original function.

Statement -2 :-

→ The complement of the product of maxterms is equal to product of missing max terms from the original functions.

→ Find out the product of max terms form of the function

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

Sol Given function $F(A,B,C) = A'B'C' + ABC + AB'C + AB'C'$

$$F(A,B,C) = m_0 + m_7 + m_5 + m_4$$

According to statement 1

$$F'(A,B,C) = m_1 + m_2 + m_3 + m_6$$

Again take the complement on both sides.

$$F''(A,B,C) = \{A'B'C + A'BC' + A'BC + ABC')'$$

From demorgan's theorem

$$F(A,B,C) = (A'B'C)'(A'BC')'(A'BC)'(ABC')'$$

Again Applying demorgan's theorem

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

$$= M_1 \cdot M_2 \cdot M_3 M_6$$

$$F(A,B,C) = \Pi(1,2,3,6)$$

→ Find out Sum of minterms form of the function

$$F(A,B,C) = (A'+B'+C')(A+B+C)(A'+B+C') \cdot (A'+B'+C)$$

Sol Given function

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

From Statement ②

$$F(A,B,C) = M_7 M_0 M_5 M_6$$

$$F'(A,B,C) = M_1 M_2 M_3 M_4$$

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

Again talking complement on both sides

$$F''(A,B,C) = \left((A+B+C')(A+B+C)(A+B'+C')(A'+B+C)\right)'$$

I Applying demorgans theorem

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

I Again Applying demorgans theorem

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

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

Standard form :-

① Sum of products form (SOP)

② product of sums form (POS)

① Sum of products :- (SOP)

→ The SOP expression contains two or more AND functions ORed together.

→ Each AND term (product term) consists of one or more variables appearing in either complemented or uncomplemented form.

→ All these product terms are connected with OR operation. This kind of logic expression is called as "SOP" - Sum of products.

Ex:- $f = AB + BC' + CA$

② products of sums (POS) :-

→ The POS expression contains two or more OR functions ANDed together.

→ Each OR term (Sum term) contains one or more Variable in complemented or uncomplemented form.

→ These sum terms are connected by AND operation.

→ This kind of logic expression is called as "POS" - product of sums.

Ex:- $f(A,B,C) = (A'+B)(B+C')(A+B+C)$

→ Find out SOP form of the function, $F = x(x+y+yz)$

Sol $F = x(x+y+yz)$

$= x \cdot x + x \cdot y + xyz$ $\quad \because A \cdot A = A$

$F = x + xy + xyz$

→ Find out the POS form of the function $F = x + yz + xy$

Sol $F = x+yz+xy$

$= (x+y)(x+z) + xy$ from distributive law

i.e ~~A+BC+~~

$\therefore A+BC = (A+B)(A+C)$

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

$\therefore A+A=A$

$= (x+y+x)(x+y+y)(x+z+x)(x+z+y)$

$= (x+y)(x+y)(x+z)(x+y+z)$ $\quad \therefore A \cdot A = A$

$F(x,y,z) = (x+y)(x+z)(x+y+z)$

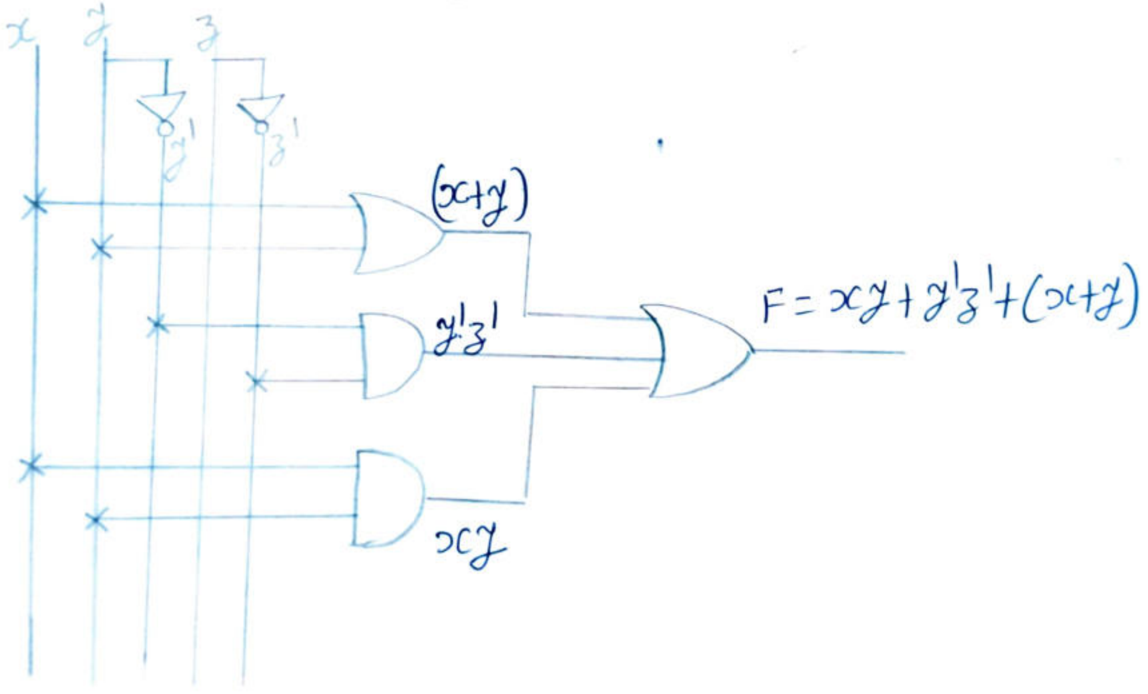
Logic diagram representation (or) Gate Implementation form :-

→ The conversion of boolean functions into gate implementation process having the following operator precedance.

1. paranthesis.
2. NOT - operation.
3. AND - operation.
4. OR - operation.

→ Find out gate implementation form of the function.

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



Truth table form :

→ The truth table consists of all possible combinations of i/ps & corresponding o/ps.

→ Find the truth table form of the function, $F = (A+B)(\overline{A}+C)$

Sol Given function

$$F = (A+B)(\overline{A}+C)$$

A	B	C	$\overline{A}$	$A+B$	$\overline{A}+C$	$F = (A+B)(\overline{A}+C)$
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	1	0	1	1	1
1	1	0	0	1	0	0
1	1	1	0	1	1	1

GATE LEVEL MINIMIZATION

→ Karnaugh map (or) K-map :-

→ Karnaugh map method is a straight forward method for the simplification of the boolean functions.

→ K-map is made up of group of squares.

→ Each square represent one minterm/maxterm.

→ The no of squares in a K-map dependent on the number of variables in a given boolean function.

→ Depending on number of variable K-maps are divided as

1. 2-Variable K-map
2. 3-Variable K-map
3. 4-Variable K-map
4. 5-Variable K-map.

→ Minterms in K-maps :-

→ In this K-maps each and every square represents the minterms.

1. 2-Variable K-map :-

→ This K-map consists of $2^2 = 4$ squares.

→ Each square represents one 2-variable minterm.

→ The following diagram gives the minterm representation & designation representation of 2-variable K-map.

$x \backslash y$	0	1
0	$x'y'$ (0)	$x'y$ (1)
1	xy' (2)	xy (3)

$x \backslash y$	0	1
0	m_0 (0)	m_1 (1)
1	m_2 (2)	m_3 (3)

3-Variable K-map :-

→ This K-map consists of $2^3 = 8$ squares.

$x \backslash yz$	00	01	11	10
0	$x'y'z'$ (0)	$x'y'z$ (1)	$x'yz$ (3)	$x'yz'$ (2)
1	$xy'z'$ (4)	$xy'z$ (5)	xyz (7)	xyz' (6)

$x \backslash yz$	00	01	11	10
0	m_0 (0)	m_1 (1)	m_3 (3)	m_2 (2)
1	m_4 (4)	m_5 (5)	m_7 (7)	m_6 (6)

4-Variable K-map :-

→ This K-map consists of $2^4 = 16$ squares.

$wx \backslash yz$	00	01	11	10
00	$w'x'y'z'$ (0)	$w'x'y'z$ (1)	$w'x'yz$ (3)	$w'x'yz'$ (2)
01	$w'xy'z'$ (4)	$w'xy'z$ (5)	$w'xyz$ (7)	$w'xyz'$ (6)
11	$wxy'z'$ (12)	$wxy'z$ (13)	$wxyz$ (15)	$wxyz'$ (14)
10	$wx'y'z'$ (8)	$wx'y'z$ (9)	$wx'yz$ (11)	$wx'yz'$ (10)

$wx \backslash yz$	00	01	11	10
00	m_0 (0)	m_1 (1)	m_3 (3)	m_2 (2)
01	m_4 (4)	m_5 (5)	m_7 (7)	m_6 (6)
11	m_{12} (12)	m_{13} (13)	m_{15} (15)	m_{14} (14)
10	m_8 (8)	m_9 (9)	m_{11} (11)	m_{10} (10)

→ Maxterms in K-map

→ In this K-maps each square represents one maxterm.

2-Variable K-map :-

→ ~~In~~ This K-map consists of $2^2 = 4$ squares.

→ Each square represents one 2-variable minterms

→ The following diagram gives the max terms representation & its designation representation of 2-variable K-map.

x \ y	0	1
0	$x+y$ 0	$x+y'$ 1
1	$x'+y$ 2	$x'+y'$ 3

x \ y	0	1
0	M_0 0	M_1 1
1	M_2 2	M_3 3

3-Variable K-map :-

→ This K-map consists of $2^3 = 8$ squares.

x \ yz	00	01	11	10
0	$x+y+z$ 0	$x+y+z'$ 1	$x+y'+z'$ 3	$x+y'+z$ 2
1	$x'+y+z$ 4	$x'+y+z'$ 5	$x'+y'+z'$ 7	$x'+y'+z$ 6

x \ yz	00	01	11	10
0	M_0 0	M_1 1	M_3 3	M_2 2
1	M_4 4	M_5 5	M_7 7	M_6 6

Four variable K-map :-

→ This K-map consists of $2^4 = 16$ squares.

wx \ yz	00	01	11	10
00	$w+x+y+z$ 0	$w+x+y+z'$ 1	$w+x+y'+z'$ 3	$w+x+y'+z$ 2
01	$w+x'+y+z$ 4	$w+x'+y+z'$ 5	$w+x'+y'+z'$ 7	$w+x'+y'+z$ 6
11	$w'+x'+y+z$ 12	$w'+x'+y+z'$ 13	$w'+x'+y'+z'$ 15	$w'+x'+y'+z$ 14
10	$w'+x+y+z$ 8	$w'+x+y+z'$ 9	$w'+x+y'+z'$ 11	$w'+x+y'+z$ 10

wx \ yz	00	01	11	10
00	M_0 0	M_1 1	M_3 3	M_2 2
01	M_4 4	M_5 5	M_7 7	M_6 6
11	M_{12} 12	M_{13} 13	M_{15} 15	M_{14} 14
10	M_8 8	M_9 9	M_{11} 11	M_{10} 10

→ Simplification of Boolean functions by using K-map method :-

→ The following steps are needed to find out the simplified expression of given boolean function using K-map.

① ⓐ If the given boolean function is in sum of minterms form or sum of products form, then take the minterms K-map.

(b) If the given boolean function is in product of maxterms or product of sums form, then take the maxterms K-map.

(2) Mark '1' on the corresponding squares by using the minterms or maxterms ~~form~~ from the given boolean function.

(3) After marking combine the maximum of adjacent squares. But the max number of adjacent squares is always 2^n. when $n = 0, 1, 2, 3 \ldots$.

(4) Find expression for each group & combine them to get simplified expression.

→ Find out the simplified expression of the function using K-map. $F(x,y) = x'y + xy' + xy$

Sol $F(x,y) = x'y + xy' + xy$

$x \backslash y$	0	1
0	0	1
1	1	1 3

→ G_1

G_2

$G_1 - 1, 3$

		$x\ y$
$G_1 \rightarrow$	$1 \rightarrow$	$\cancel{0}\ 1$
	$3 \rightarrow$	$\cancel{1}\ 1$
		$\times\ 1$

$G_1 \rightarrow y$

$G_2 - 2, 3$

$G_2 \rightarrow 2 \rightarrow 1\ \cancel{0}$

$3 \rightarrow 1\ \cancel{1}$

$1 \times$

$G_2 \rightarrow x$

$F(x,y) = x + y$

→ Find simplified expression of the functions using K-map.

(7) (or)

→ Simplify the following boolean functions in to minimal SOP form using K-map

(i) $F(x,y,z) = \Sigma(0,1,5,7)$

(ii) $F(A,B,C) = \Sigma(0,2,5,7)$

(iii) $F(A,B,C) = \Sigma(0,2,3,4,6,7)$

Sol (i) $F(x,y,z) = \Sigma(0,1,5,7)$

x \ yz	00	01	11	10
0	1 (0)	1 (1)	(3)	(2)
1	(4)	1 (5)	1 (7)	(6)

G1, G2

$G_1 \to 0,1 \to x'y'$

$G_2 \to 5,7 \to xz$

$G_1 \to 0,1$

$0 \to 0\,0\,0$

$1 \to 0\,0\,1$

$0\,0\,\times$

$x'y'$

$F_{min}(x,y,z) = x'y' + xz$

$G_2 \to 5,7$

$5 \to 1\,0\,1$

$7 \to 1\,1\,1$

$1\,\times\,1$

xz

(ii) Given function

$F(A,B,C) = \Sigma(0,2,5,7)$

A \ BC	00	01	11	10
0	1 (0)	(1)	(3)	1 (2)
1	(4)	1 (5)	1 (7)	(6)

G2, G1

$G_1 \to 5,7 \to AC$

$G_2 \to 0,2 \to A'C'$

$G_1 \rightarrow 5, 7$

$5 \rightarrow 1\,\cancel{0}\,1$

$7 \rightarrow 1\,\cancel{1}\,1$

$1 * 1$

$A\cancel{B}C$

$G_2 \rightarrow 0, 2$

$0 \rightarrow 0\,\cancel{0}\,0$

$2 \rightarrow 0\,\cancel{1}\,0$

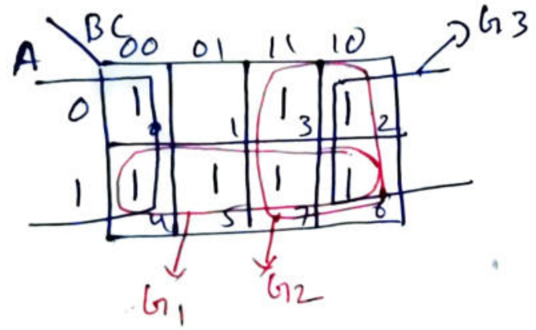
$0 * 0$

$A'C'$

$$F_{min_{SOP}}(A,B,C) = AC + A'C'$$

(iii) Given function

$F = (0, 2, 3, 4, 6, 7)$



$G_1 \rightarrow 4, 5, 6, 7 \rightarrow A$

$4 \rightarrow 1\,\cancel{0}\,\cancel{0}$

$5 \rightarrow 1\,\cancel{0}\,\cancel{1}$

$6 - 1\,\cancel{1}\,\cancel{0}$

$7 - 1\,\cancel{1}\,\cancel{1}$

$1 * *$

A

$G_1 \rightarrow 4, 5, 6, 7 \rightarrow A$

$G_2 \rightarrow 2, 3, 6, 7 \rightarrow B$

$G_3 \rightarrow 0, 2, 4, 6 \rightarrow C'$

$G_2 \rightarrow 2, 3, 6, 7 \rightarrow B$

$2 \rightarrow \cancel{0}\,\cancel{0}\,1\,\cancel{0}$

$3 \rightarrow \cancel{0}\,\cancel{0}\,1\,\cancel{1}$

$6 \rightarrow \cancel{0}\,\cancel{1}\,1\,\cancel{0}$

$7 \rightarrow \cancel{0}\,\cancel{1}\,1\,\cancel{1}$

$* 1 *$

B

$G_3 \rightarrow 0, 2, 4, 6 - C'$

$0 \rightarrow \cancel{0}\,\cancel{0}\,0$

$2 \rightarrow \cancel{0}\,\cancel{1}\,0$

$4 \rightarrow \cancel{1}\,\cancel{0}\,0$

$6 \rightarrow \cancel{1}\,\cancel{1}\,0$

$* * \; 0$

C'

$$F_{min_{SOP}}(A,B,C) = A + B + C'$$

→ Simplify the following boolean functions using K-map

(i) $F(A,B,C) = ABC' + ABC + A'B'C + A'B'C'$

Sol Given

$F(A,B,C) = ABC' + ABC + A'B'C + A'B'C'$

$= m_6 + m_7 + m_1 + m_0$

$F(A,B,C) = \Sigma(0,1,6,7)$

A \ BC	00	01	11	10
0	1 (0)	1 (1)	(3)	(2)
1	(4)	(5)	1 (7)	1 (6)

G_1 (0, 1); G_2 (7, 6)

$G_1 \to 0,1$

$0 \to 00\cancel{0}$

$1 \to 00\cancel{1}$

00 *

$A'B'$

$G_1 \to 0,1 \to A'B'$

$G_2 \to 6,7 \to AB$

$G_2 \to 6,7$

$6 \to 11\cancel{0}$

$7 \to 11\cancel{1}$

$A'B'$ *

$F_{min}(A,B,C) = A'B' + AB$

→ Simplify the following boolean functions into minimal SOP form using K-map

(i) $F(A,B,C,D) = \Sigma(0,1,2,3,4,6,8,9,10,11)$

(ii) $F(A,B,C,D) = \Sigma(0,1,3,7,8,9,11,15)$

Sol Given $F(A,B,C,D) = \Sigma(0,1,2,3,4,6,8,9,10,11)$

AB \ CD	00	01	11	10
00	1 (0)	1 (1)	1 (3)	1 (2)
01	1 (4)	(5)	(7)	1 (6)
11	(12)	(13)	(15)	(14)
10	1 (8)	1 (9)	1 (11)	1 (10)

$G_1 - 0,1,2,3,\cancel{4}8,9,10,11 \to B'$

$G_2 - 0,2,4,6 \to A'D'$

$G_1 \rightarrow 0,1,2,3,8,9,10,11$

0 → ø0øø
1 → ø0øx
2 → ø0xø
3 → ø0xx
8 → x0øø
9 → x0øx
10 → x0xø
11 → x0xx

---

* 0 * *

B'

$G_2 - 0,2,4,6$

0 → 0øø0
2 → 0øx0
4 → 0xø0
6 → 0xx0

---

0 * * 0

$A'D'$

$$F_{\text{min sop}}(A,B,C,D) = A'D' + B'$$

(ii) Given $F(A,B,C,D) = \Sigma(0,1,3,7,8,9,11,15)$

AB \ CD	00	01	11	10
00	1 (0)	1 (1)	1 (3)	(2)
01	(4)	(5)	1 (7)	(6)
11	(12)	(13)	1 (15)	(14)
10	1 (8)	1 (9)	1 (11)	(10)

$G_1 - 3,7,11,15 \rightarrow CD$
$G_2 \rightarrow 0,1,8,9 \rightarrow B'C'$

$G_1 - 3,7,11,15$

3 → øø11
7 → øx11
11 → xø11
15 → xx11

---

* * 1 1

CD

$G_2 \rightarrow 0,1,8,9$

0 → ø00ø
1 → ø00x
8 → x00ø
9 → x00x

---

* 0 0 *

$B'C'$

$$F_{\text{min sop}}(A,B,C,D) = CD + B'C'$$

→ Simplify the following switching functions into minimal SOP using K-map.

(i) $F(A,B,C,D) = \Sigma(1,3,6,8,9,10)$

(ii) $F(W,X,Y,Z) = W'X'Y'Z' + WXY'Z' + W'X'YZ + WXYZ$

(iii) $F(A,B,C,D) = AB'C' + AC + A'CD'$

Sol (i) Given $F(A,B,C,D) = \Sigma(1,3,6,8,9,10)$

AB \ CD	00	01	11	10
00	0	1 (1)	1 (3)	2
01	4	5	7	1 (6)
11	12	13	15	14
10	1 (8)	1 (9)	11	1 (10)

$G_1 \to 1,3 \to A'B'D$

$G_2 \to 8,9 \to AB'C'$

$G_3 \to 8,10 \to AB'D'$

$G_4 \to 6 \to A'BCD'$

$G_1 \to 1,3$

$1 \to 0001$
$3 \to 0011$

$00*1$
$A'B'D$

$G_2 \to 8,9$

$8 \to 1000$
$9 \to 1001$

$100*$
$AB'C'$

$G_3 \to 8,10$

$8 \to 1000$
$10 \to 1010$

$10*0$
$AB'D'$

$G_4 \to 6$

$6 \to 0110$
$A'BCD'$

$$F_{\text{min SOP}}(A,B,C,D) = A'B'D + AB'C' + AB'D' + A'BCD'$$

(ii) Given

$$F(W,X,Y,Z) = \underset{0000}{W'X'Y'Z'} + \underset{1100}{WXY'Z'} + \underset{0011}{W'X'YZ} + \underset{1111}{WXYZ}$$

$$F(W,X,Y,Z) = m_0 + m_{12} + m_3 + m_{15}$$

$F(w,x,y,z) = \Sigma(0,3,12,15)$

wx \ yz	00	01	11	10
00	1_{0}	$_{1}$	1_{3}	$_{2}$
01	$_{4}$	$_{5}$	$_{7}$	$_{6}$
11	1_{12}	$_{13}$	1_{15}	$_{14}$
10	$_{8}$	$_{9}$	$_{11}$	$_{10}$

$G_1 - 0 \rightarrow 0000 \rightarrow \cancel{A'B'C'D'}\ w'x'y'z'$

$G_2 \rightarrow 3 \rightarrow 0011 \rightarrow \cancel{A'B'CD}\ w'x'yz$

$G_3 \rightarrow 12 \rightarrow 1100 \rightarrow \cancel{ABC}\ wxy'z'$

$G_4 \rightarrow 15 \rightarrow 1111 \rightarrow wxyz$

$F_{minsop}(w,x,y,z) = w'x'y'z' + w'x'yz + wxy'z' + wxyz$

(iii) Given $F(A,B,C,D) = AB'C' + AC + A'CD'$

$AB'C' \Rightarrow 100\underline{0} \rightarrow m_8$

$100\underline{1} \rightarrow m_9$

$AC \rightarrow 1\underline{0}1\underline{0} \rightarrow m_{10}$

$1\underline{0}1\underline{1} \rightarrow m_{11}$

$1\underline{1}1\underline{0} \rightarrow m_{14}$

$1\underline{1}1\underline{1} \rightarrow m_{15}$

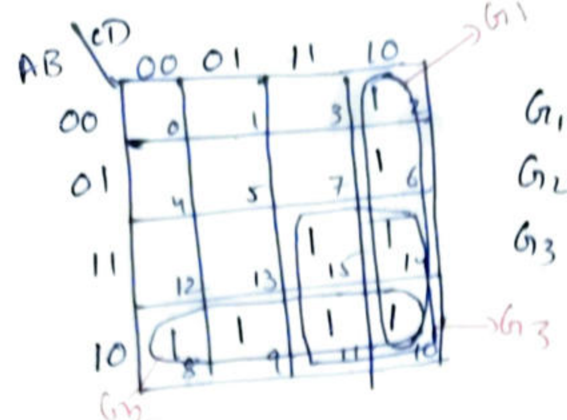
$A'CD' \rightarrow 0\underline{0}10 \rightarrow m_2$

$0\underline{1}10 \rightarrow m_6$

$F(A,B,C,D) = m_2 + m_6 + m_8 + m_9 + m_{10} + m_{11} + m_{14} + m_{15}$

$= \Sigma(2,6,8,9,10,11,14,15)$

Y. pavan kumar Reddy



$G_1 \to 2,6,10,14 \to CD'$

$G_2 \to 8,9,10,11 \to AB'$

$G_3 \to 10,11,14,15 \to AC$

$G_1 \to 2,6,10,14$

2 → ØØ10
6 → ØX10
10 → XØ10
14 → XX10

$**10$

CD'

$G_2 \to 8,9,10,11$

8 → 10ØØ
9 → 10ØX
10 → 10XØ
11 → 10XX

$10**$

AB'

$G_3 \to 10,11,14,15$

10 → 1Ø1Ø
11 → 1Ø1X
14 → 1X1Ø
15 → 1X1X

$1*1*$

AC

$$F(A,B,C,D)_{\text{min SOP}} = CD' + AB' + AC$$

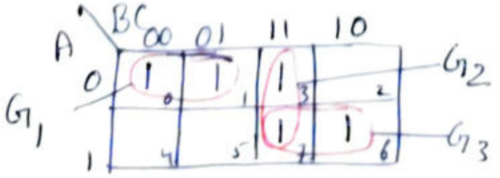
→ Simplify the following Boolean function into minimal POS form.

(i) $F(A,B,C) = \Pi(0,1,3,6,7)$

(ii) $F(A,B,C,D) = \Pi(0,1,3,5,6,7,12,13,15)$

(iii) $F(A,B,C,D) = (A+B+C)(A'+B+C+D)(B'+C'+D')(A'+B')$

Sol (i) $F(A,B,C) = \Pi(0,1,3,6,7)$



$G_1 \rightarrow 0, 1 \rightarrow (A+B)$

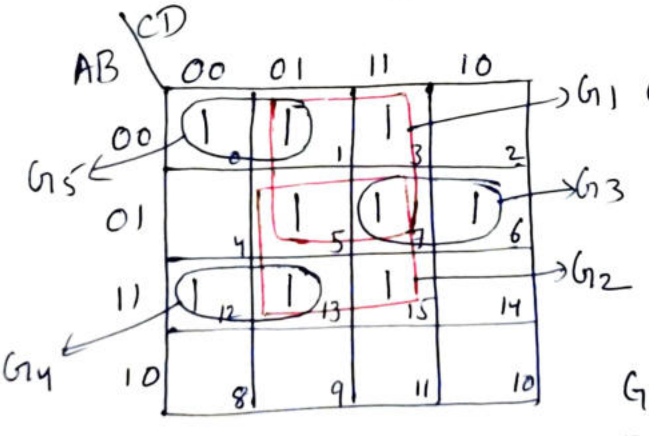
$G_2 \rightarrow 3, 7 \rightarrow (B'+C')$

$G_3 \rightarrow 6, 7 \rightarrow (A'+B')$

$G_1 \rightarrow 0, 1$	$G_2 \rightarrow 3, 7$	$G_3 \rightarrow 6, 7$
$0 \rightarrow 0\,0\,\cancel{0}$	$3 \rightarrow \cancel{0}\,1\,1$	$6 \rightarrow 1\,1\,\cancel{0}$
$1 \rightarrow 0\,0\,\cancel{1}$	$7 \rightarrow \cancel{1}\,1\,1$	$7 \rightarrow 1\,1\,\cancel{1}$
$0\,0\,*$	$*\,1\,1$	$1\,1\,*$
$(A+B)$	$(B'+C')$	$(A'+B')$

$$F_{sop}(A,B,C) = (A+B)(B'+C')(A'+B')$$

(ii) $F(A,B,C,D) = \Pi(0,1,3,5,6,7,12,13,15)$



ABCD

$G_1 \rightarrow 1 - 0\,0\,\cancel{0}\,1$

$3 - 0\,0\,\cancel{1}\,1$

$5 - 0\,1\,0\,1$

$7 - 0\,\cancel{1}\,\cancel{1}\,1$

$0\,*\,*\,1$

$A + B'$

$G_1 \rightarrow A+D'$

ABCD

$G_2 - 5 - \cancel{0}\,1\,0\,1$

$7 - \cancel{0}\,1\,1\,1$

$13 - \cancel{1}\,1\,0\,1$

$15 - \cancel{1}\,1\,\cancel{1}\,1$

$*\,1\,*\,1$

$B'+D$

$G_2 \rightarrow B'+D$

$G_1 \rightarrow 1, 3, 5, 7$

$G_2 \rightarrow 5, 7, 13, 15$

$G_3 \rightarrow 6, 7$

$G_4 \rightarrow 12, 13$

$G_5 \rightarrow 0, 1$

ABCD

$G_3 - 6 - 0\,1\,1\,\cancel{0}$

$7 - 0\,1\,1\,\cancel{1}$

$0\,1\,1\,*$

$A+B'+C'$

$G_3 \rightarrow A+B'+C'$

ABCD

$G_4 - 12 - 1\,1\,0\,\cancel{0}$

$13 - 1\,1\,0\,\cancel{1}$

$1\,1\,0\,*$

$A'+B'+C$

$G_4 \rightarrow A+B+C$

ABCD

$G_5 - 0 - 0\,0\,0\,0$

$1 - 0\,0\,0\,1$

$0\,0\,0\,*$

$A+B+C$

$F(A,B,C,D) = G_1 G_2 G_3 G_4 G_5$

$F(A,B,C,D) = (A+D')(B'+D')(A+B'+C')(A'+B'+C)(A+B+C)$

A'=1 A=0

(iii) $F(A,B,C,D) = (A+B+C)(A'+B+C+D)(B'+C'+D')(A'+B')$ (44)

$A+B+C = 000\underline{0} - M_0$ $A'+B+C+D = 1000 - M_8$

$000\underline{1} - M_1$

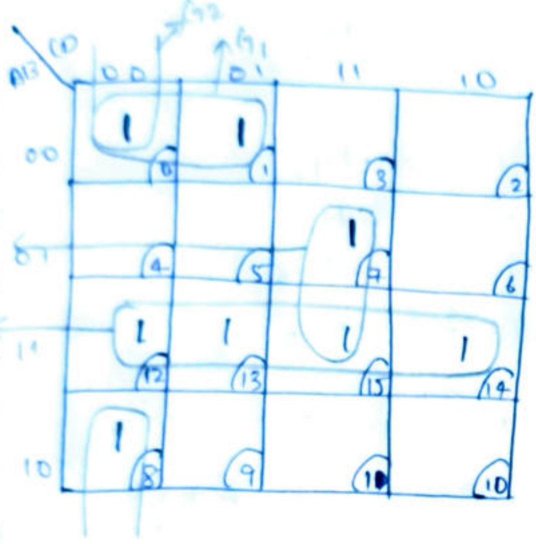
$B'+C'+D' = \underline{0}111 - M_7$ $A'+B' = 11\underline{0}\underline{0} - M_{12}$

$\underline{1}111 - M_{15}$ $11\underline{0}\underline{1} - M_{13}$

$11\underline{1}0 - M_{14}$

$11\underline{1}1 - M_{15}$

$F(A,B,C,D) = \pi(0,1,7,8,12,13,14,15)$



ABCD

$G_1 - 0 - 000\cancel{0}$

$1 - \underline{000*}$

$000*$

$A+B+C$

$G_3 - 7 - \cancel{0}111$

$15 - \underline{\cancel{1}111}$

$*111$

$B'+C'+D'$

ABCD

$G_2 - 0 - \cancel{0}000$

$8 - \underline{\cancel{1}000}$

$*000$

$B+C+D$

$G_4 \to 12 - 11\cancel{0}\cancel{0}$

$13 - 11\cancel{0}\cancel{1}$

$14 - 11\cancel{1}0$

$15 - \underline{11\cancel{1}\cancel{1}}$

$11**$

$A'+B'$

$F(A,B,C,D) = G_1 G_2 G_3 G_4$.

$G_1 \to A+B+C$

$G_2 \to B+C+D$

$G_3 \to B'+C'+D'$

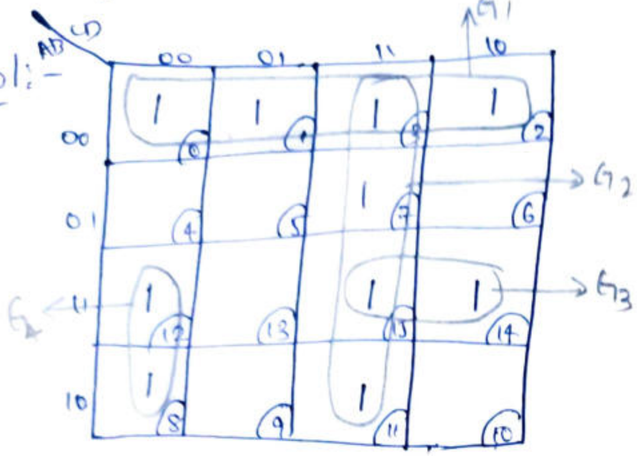
$G_4 \to A'+B'$

$F(A,B,C,D) = (A+B+C)(B+C+D)(B'+C'+D')(A'+B')$.

Impliment the following function using NOR gates.

(1) $F(A,B,C,D) = \Sigma(0,1,2,3,7,8,11,12,14,15)$.

Sol:-



$G_1 \rightarrow 0,1,2,3$

0 - 0 0 0 0
1 - 0 0 0 1
2 - 0 0 1 0
3 - 0 0 1 1

0 0 * *

$A'B'$

$G_2 \rightarrow 3,7,11,15$

3 - 0 0 1 1
7 - 0 1 1 1
11 - 1 0 1 1
15 - 1 1 1 1

* * 1 1

C D

$G_3 \rightarrow 14, 15$

14 - 1 1 1 0
15 - 1 1 1 1

1 1 1 *

A B C

$G_4 \rightarrow 8, 12$

8 - 1 0 0 0
12 - 1 1 0 0

1 * 0 0

$A C' D'$

$F(A,B,C,D) = G_1 + G_2 + G_3 + G_4$

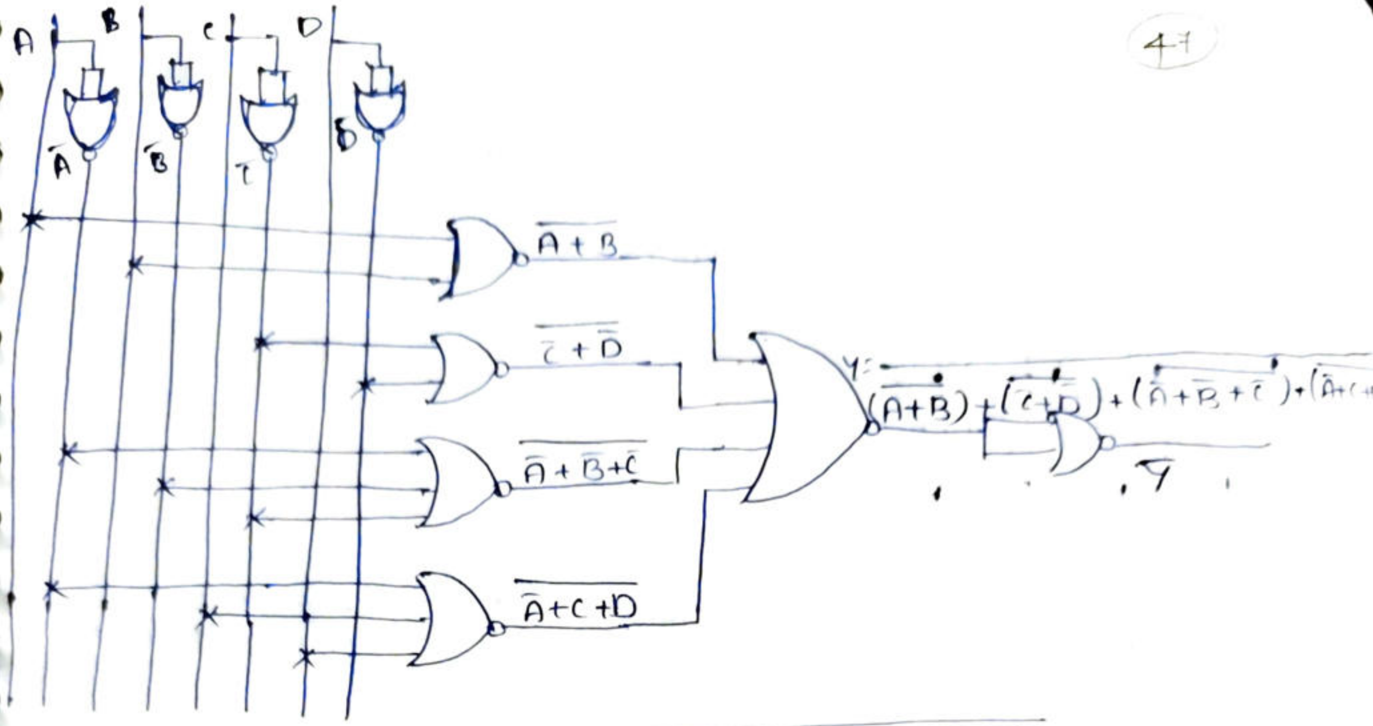
$F(A,B,C,D) = (A'B') + (CD) + (ABC) + (AC'D')$

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

$= \overline{\overline{(\bar{A}\bar{B})}\,\overline{(CD)}\,\overline{(ABC)}\,\overline{(A\bar{C}\bar{D})}}$

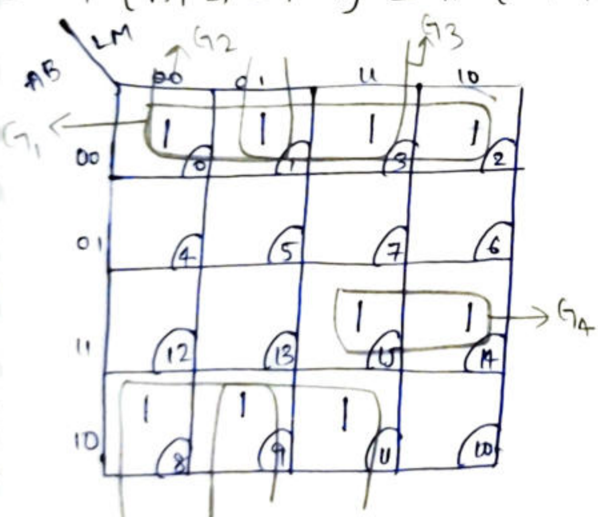
$= \overline{(A+B)(\bar{C}+\bar{D})(\bar{A}+\bar{B}+\bar{C})(\bar{A}+C+D)}$

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



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

ii) $F(A,B,L,M) = \Pi(0,1,2,3,8,9,11,14,15)$



$G_1 \rightarrow 0,1,2,3$

0 - 0000
1 - 0001
2 - 0010
3 - 0011

00**

$A+B$

$G_2 \rightarrow 0,1,8,9$

0 - 0000
1 - 0001
8 - 1000
9 - 1001

00

$B+L$

$G_3 \rightarrow 1,3,9,11$

1 - 0001
3 - 0011
9 - 1001
11 - 1011

*0*1

$B+M'$

$G_4 \rightarrow 15,14$

14 - 1110
15 - 1111

111*

$A'+B'+L'$

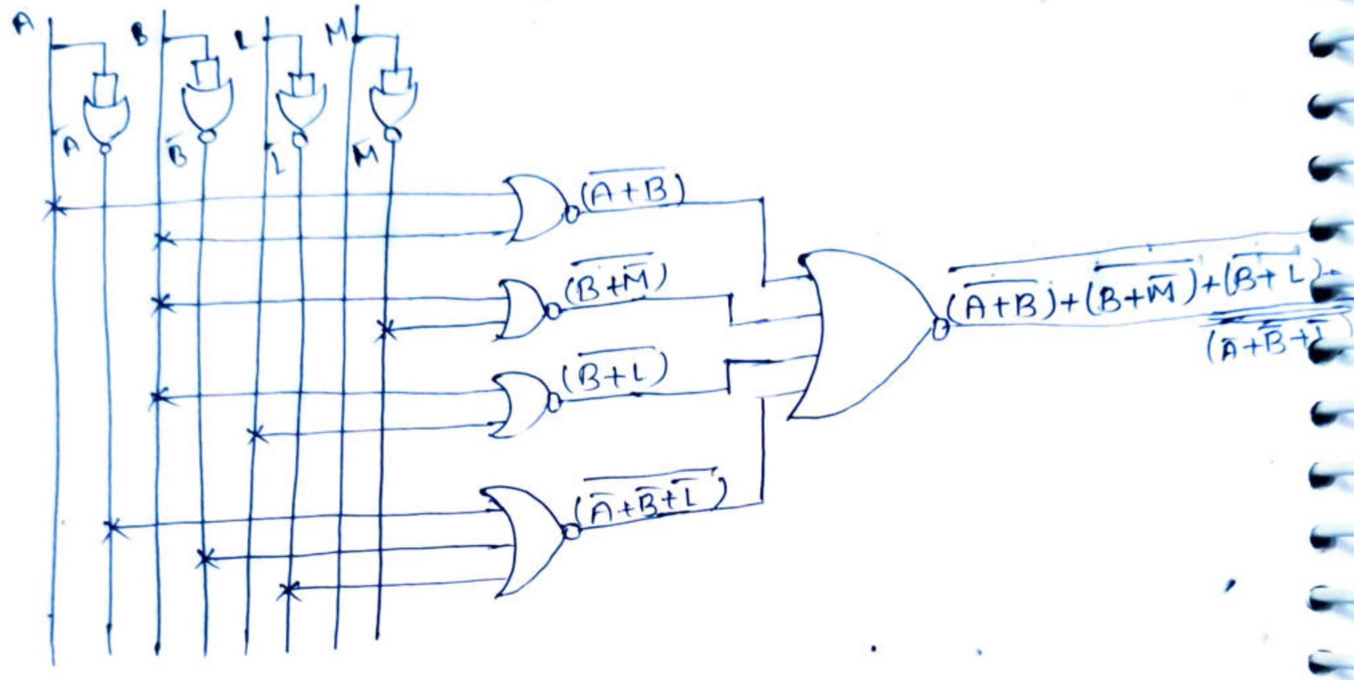
$F(A,B,L,M) = G_1 G_2 G_3 G_4$

$= (A+B)(B+L)(B+M')(A'+B'+L')$

$$F(A,B,C,D) = (A+B)(B+\bar{M})(B+L)(\bar{A}+\bar{B}+\bar{L})'$$

$$= \overline{\overline{(A+B)(B+\bar{M})(B+L)(\bar{A}+\bar{B}+\bar{L})}}$$

$$F(A,B,C,D) = \overline{\overline{(A+B)} + \overline{(B+\bar{M})} + \overline{(B+L)} + \overline{(\bar{A}+\bar{B}+\bar{L})}}$$



$F(A,B,C,D) = G_1 + G_2 + G_3 + G_4$

$= A'B' + CD + ABC + AC'D'$

SOP ~~to~~ from POS $\Rightarrow F(A,B,C,D) = \Sigma(0,1,2,3,7,8,11,12,14,15)$

AB \ CD	00	01	11	10
00	1 (0)	1 (1)	1 (3)	1 (2)
01	(4)	(5)	1 (7)	(6)
11	1 (12)	(13)	1 (15)	1 (14)
10	1 (8)	(9)	1 (11)	(10)

AB \ CD	00	01	11	10
00	(0)	(1)	(3)	(2)
01	0 (4)	0 (5)	(7)	0 (6)
11	(12)	0 (13)	(15)	(14)
10	(8)	0 (9)	(11)	0 (10)

G_1, G_2, G_3, G_4

$G_1 \rightarrow 4, 5$

4 - 0 1 0 0

5 - 0 1 0 1

0 1 0 *

$A'BC'$

$G_2 \rightarrow 4, 6$

4 - 0 1 0 0

6 - 0 1 1 0

0 1 * 0

$A'BD'$

$G_3 \rightarrow 13, 9$

9 - 1 0 0 1

13 - 1 1 0 1

1 * 0 1

$AC'D$

G_4 - 10 - 1 0 1 0

$AB'CD'$

$F(A,B,C,D) = G_1 + G_2 + G_3 + G_4$

$= A'BC' + A'BD' + AC'D + AB'CD'$

Don't Care Condition :-

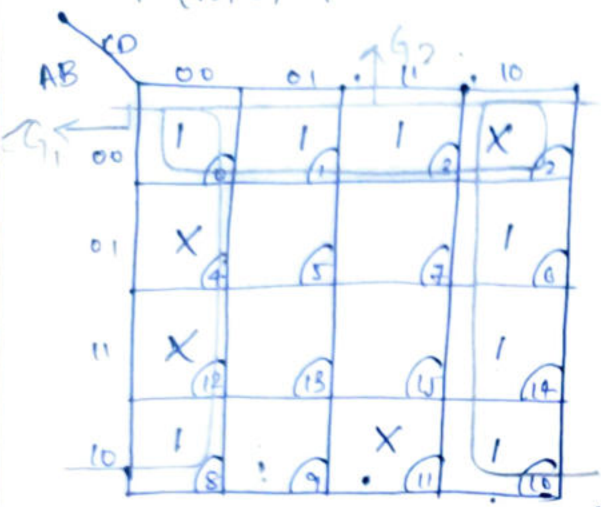
* The undefined combinations are called as Don't care conditions. It is indicated by 'x'.
* By using don't care condition we can reduce boolean functions to a greater extent.

Ex:- In BCD 1010 to 1111 are undefined

* Don't Care Conditions are grouped when the size of the group is increasing.

Simplify the following boolean function in minimal SOP using K-Map.

$F(A,B,C,D) = \Sigma(0,1,3,6,8,10,14) + d(2,4,11,12)$.



$G_1 \rightarrow 0, 4, 12, 8, 2, 6, 14, 10$

0 - 0 0 0 0
2 - 0 0 1 0
4 - 0 1 0 0
6 - 0 1 1 0
8 - 1 0 0 0
10 - 1 0 1 0
12 - 1 1 0 0
14 - 1 1 1 0

* * * 0

D'

$G_2 \rightarrow 0, 1, 2, 3$

0 - 0 0 0 0
1 - 0 0 0 1
2 - 0 0 1 0
3 - 0 0 1 1

0 0 * *

$A'B'$.

$F(A,B,C,D) = G_1 + G_2$

$F(A,B,C,D) = D' + A'B'$

Implimentation using universal gates :-

Implimentation using NAND gate :-

NOT gate using NAND :-

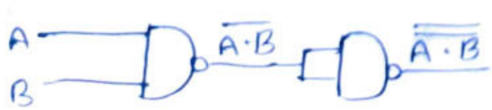
$Y = \bar{A}$

$= \bar{A}$

AND gate using NAND :-

$Y = A \cdot B$

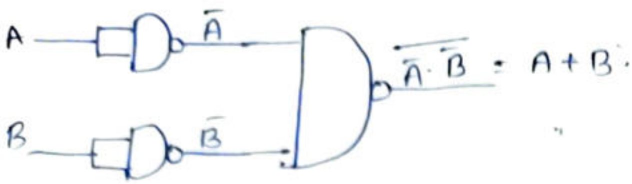
$Y = \overline{\overline{(A \cdot B)}}$



$\Rightarrow Y = A \cdot B$.

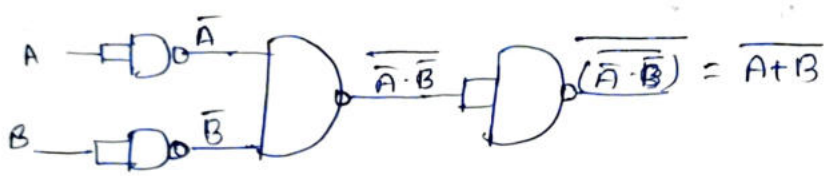
OR gate using NAND:-

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



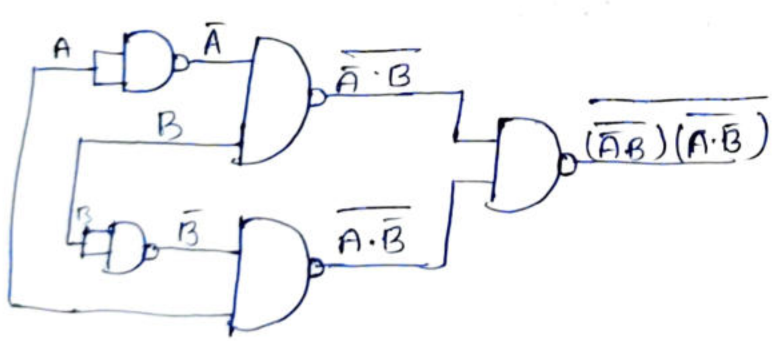
NOR gate using NAND:-

$$Y = \overline{A+B} = \overline{A}\cdot\overline{B}$$
$$= \overline{(\overline{\overline{A}\cdot\overline{B}})}$$



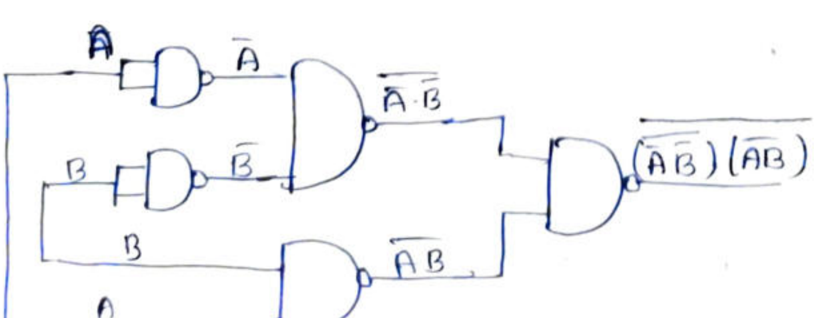
X-OR using NAND:-

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



X-NOR using NAND:-

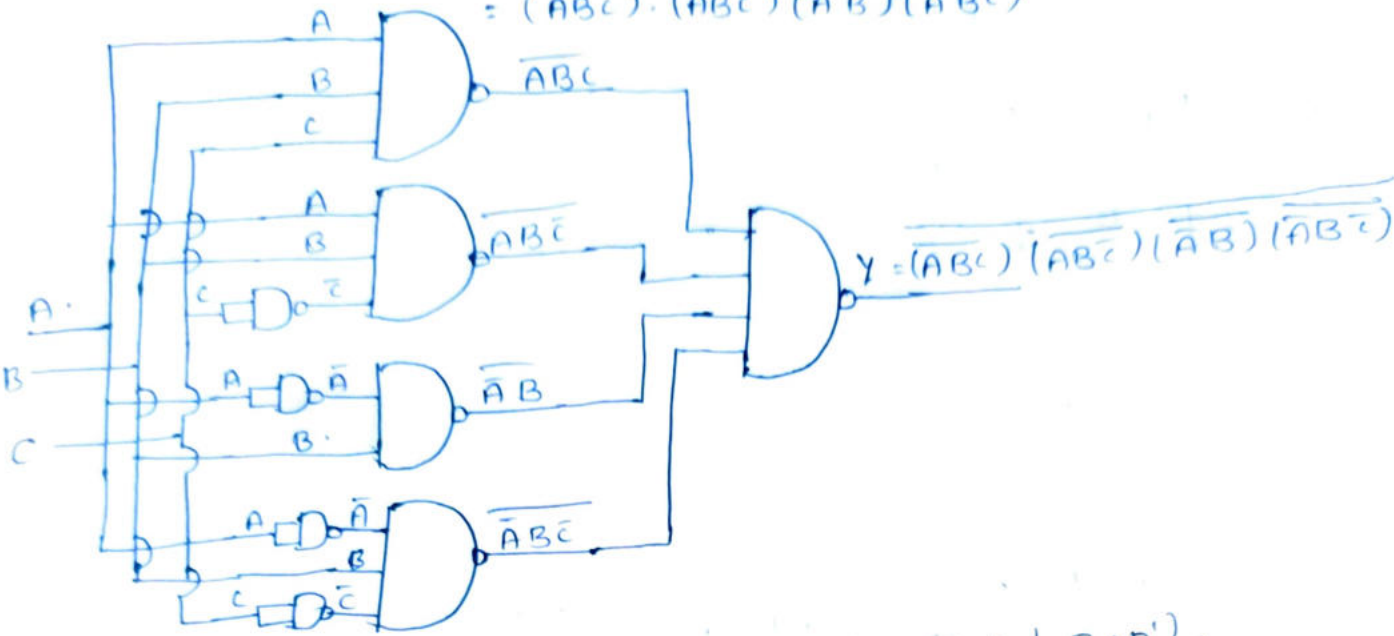
$$Y = \overline{A}\,\overline{B} + AB = \overline{\overline{\overline{A}\,\overline{B}+AB}} = \overline{((\overline{\overline{A}\,\overline{B}})\cdot(\overline{AB}))}$$



Impliment the following function using NAND gates.

(i) SOP $\Rightarrow$ $Y = ABC + AB\bar{C} + \bar{A}B + \bar{A}B\bar{C} = \overline{\overline{ABC + AB\bar{C} + \bar{A}B + \bar{A}B\bar{C}}}$

$= \overline{(\overline{ABC})\cdot(\overline{AB\bar{C}})(\overline{\bar{A}B})(\overline{\bar{A}B\bar{C}})}$



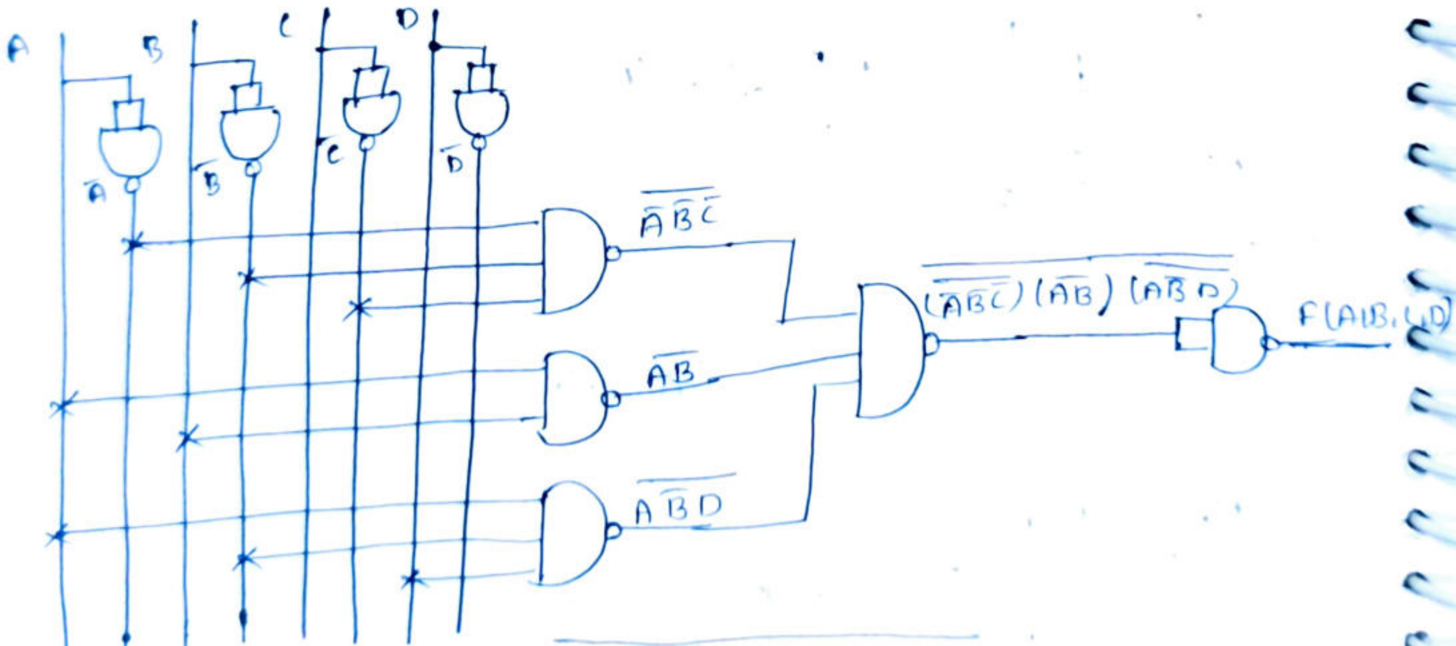
(ii) POS $\Rightarrow$ $F(A,B,C,D) = (A+B+C)(A'+B')(A'+B+D')$.

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

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

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

$F(A,B,C,D) = \overline{(\overline{\bar{A}\bar{B}\bar{C}})(\overline{AB})(\overline{A\bar{B}D})}$



$F(A,B,C,D) = \overline{\overline{(\overline{\bar{A}\bar{B}\bar{C}})(\overline{AB})(\overline{A\bar{B}D})}}$

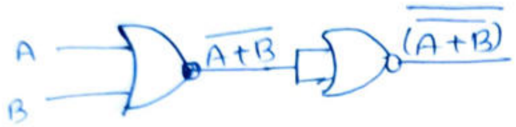
Implimentation using NOR gate :-

NOT using NOR :-

$$Y = \bar{A}$$

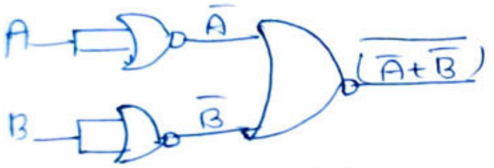
OR using NOR :-

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



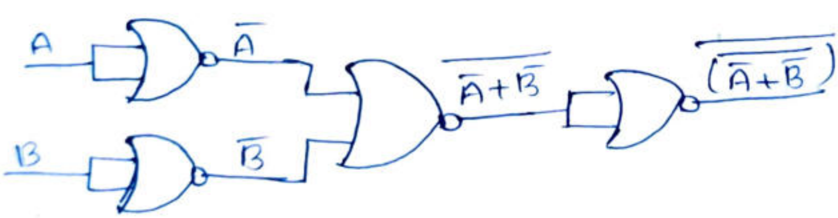
AND using NOR :-

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



NAND using NOR :-

$$Y = \overline{A\cdot B} = \overline{\overline{(\bar{A}+\bar{B})}}$$

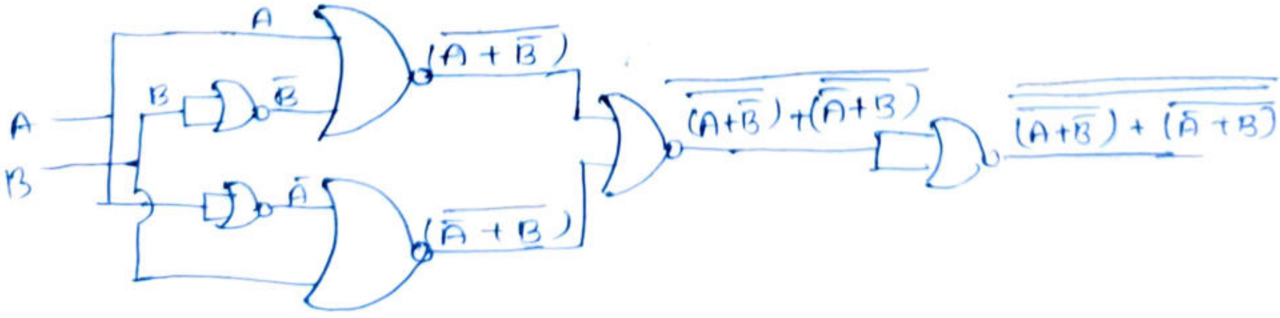


X-OR using NOR :-

$$Y = \bar{A}B + A\bar{B} = \overline{\overline{(\bar{A}B+A\bar{B})}} = \overline{(\overline{\bar{A}B})\cdot(\overline{A\bar{B}})} = \overline{(A+\bar{B})(\bar{A}+B)}$$

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

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



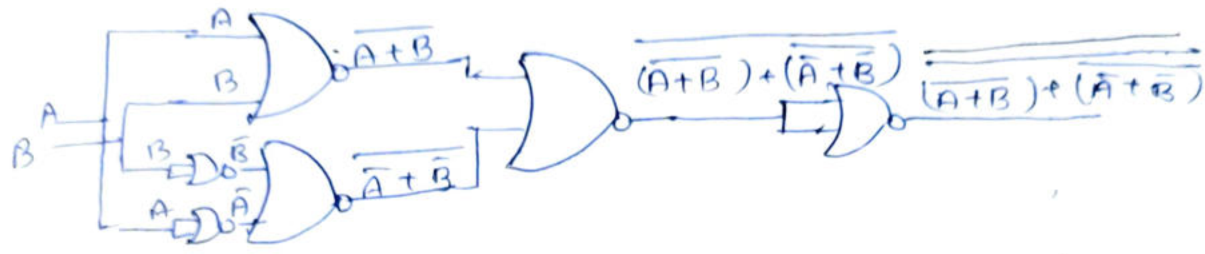
X-NOR using NOR :-

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

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

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

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



Impliment the following function using NAND Gates.

(i) $F(A,B,C,D) = \Sigma(0,1,2,8,9,10,14) = B'C' + A'B'D' + ACD'$

$F(A,B,C,D) = \Sigma(0,1,2,8,9,10,14)$.

AB \ CD	00	01	11	10
00	1 (0)	1 (1)	(3)	1 (2)
01	(4)	(5)	(7)	(6)
11	(12)	(13)	(15)	1 (14)
10	1 (8)	1 (9)	(11)	1 (10)

$G_1 \rightarrow 0, 1, 8, 9$

0 - 0000
1 - 0001
8 - 1000
9 - 1001

$* 0 0 *$

$B'C'$.

$G_2 \rightarrow 0, 2$

0 - 0000
2 - 0010

$0 0 * 0$

$A'B'D'$

$G_3 \rightarrow 14, 10$

14 - 0110
10 - 1010

$* * * D'$

$G_3 \rightarrow 14, 10$

14 - 1110
10 - 1010

$1 * 1 0$

$A C D'$

$F(A,B,C,D) = G_1 + G_2 + G_3$

$= B'C' + A'B'D' + ACD'$

$= \overline{\overline{\bar{B}\bar{C} + \bar{A}\bar{B}\bar{D} + AC\bar{D}}}$

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

5-Variable K-Map

Simplify the following boolean function into minimal sop using K-Map

$F(A,B,C,D,E) = (0,1,5,7,11,12,13,16,20,21,23,25,26,28,30,31)$.

A = 0

BC \ DE	00	01	11	10
00	1 (0)	1 (1)	(3)	(2)
01	(4)	1 (5)	1 (7)	(6)
11	1 (12)	1 (13)	(15)	(14)
10	(8)	(9)	1 (11)	(10)

A = 1

BC \ DE	00	01	11	10
00	1 (16)	(17)	(19)	(18)
01	1 (20)	1 (21)	1 (23)	(22)
11	1 (28)	(29)	1 (31)	1 (30)
10	(24)	1 (25)	(27)	1 (26)

$G_1 - 0, 1 = A'B'C'D'$

$G_2 - 5, 7, 21, 23 = B'CE$

$G_3 - 12, 13 = A'BCD'$

$G_4 - 11 = A'BC'DE$

$G_5 - 16, 20 = AB'D'E'$

$G_6 - 28, 30 = ABCE'$

$G_7 - 23, 31 = ACDE$

$G_8 - 25 = ABC'D'E$

$G_9 - 26, 30 = ABDE'$

$$\begin{array}{l} \quad\ \ A\,B\,C\,D\,E \\ G_1 \to 0 - 0\,0\,0\,0\,0 \\ \quad\ \ 1 - 0\,0\,0\,0\,1 \\ \hline \quad\ \ \ \ 0\,0\,0\,0\,* \\ \quad\ \ \ \ A'B'C'D' \end{array}$$

$$\begin{array}{l} G_2 \to 5 - 0\,0\,1\,0\,1 \\ \quad\ \ 7 - 0\,0\,1\,1\,1 \\ \quad 21 - 1\,0\,1\,0\,1 \\ \quad 23 - 1\,0\,1\,1\,1 \\ \hline \quad\quad\ \ *\,0\,1\,*\,1 \\ \quad\quad\quad B'\,C\,E \end{array}$$

$$\begin{array}{l} G_3 \to 12 - 0\,1\,1\,0\,0 \\ \quad\ \ 13 - 0\,1\,1\,0\,1 \\ \hline \quad\quad\ \ 0\,1\,1\,0\,* \\ \quad\quad\ \ A'BCD' \end{array}$$

$$\begin{array}{l} G_4 - 11 - 0\,1\,0\,1\,1 \\ \quad\quad\quad A'BC'DE \end{array}$$

$G_5 \to$ ~~16 - 10000~~ ~~20 - 01100~~

$$\begin{array}{l} 16 - 1\,0\,0\,0\,0 \\ 20 - 1\,0\,1\,0\,0 \\ \hline \quad\ \ 1\,0\,*\,0\,0 \\ \quad\ \ A\,B'\,D'\,E' \end{array}$$

$$\begin{array}{l} G_6 \to 28 - 1\,1\,1\,0\,0 \\ \quad\ \ 30 - 1\,1\,1\,1\,0 \\ \hline \quad\quad\ \ 1\,1\,1\,*\,0 \\ \quad\quad\ \ A\,B\,C\,E' \end{array}$$

$$\begin{array}{l} G_7 \to 23 - 1\,0\,1\,1\,1 \\ \quad\ \ 31 - 1\,1\,1\,1\,1 \\ \hline \quad\quad\ \ 1\,*\,1\,1\,1 \\ \quad\quad\ \ A\,C\,D\,E \end{array}$$

$$\begin{array}{l} G_8 \to 25 - 1\,1\,0\,0\,1 \\ \quad\quad\ \ A\,B\,C'D'E \end{array}$$

$$\begin{array}{l} G_9 \to 26 - 1\,1\,0\,1\,0 \\ \quad\ \ 30 - 1\,1\,1\,1\,0 \\ \hline \quad\quad\ \ 1\,1\,*\,1\,0 \\ \quad\quad\ \ A\,B\,D\,E' \end{array}$$

$F(A,B,C,D,E) = G_1 + G_2 + G_3 + G_4 + G_5 + G_6 + G_7 + G_8 + G_9$

$= (A'B'C'D') + (B'CE) + (A'BCD') + (A'BC'DE) + (AB'D'E') +$

$(ABCE') + (ACDE) + (ABC'D'E) + (ABDE')$

Tabular Method :-

$F(A,B,C,D,E) = \Sigma(0,1,4,5,7,12,13,15,16,17,20,21,22,28,30)$

Col 1		Col 2		Col 3		Col 4		Col 5		Col 6	
Min terms	Binary Representation	Min terms	Binary Representation	Min terms	Binary Representation	Min terms	Binary Representation	Min terms	Binary Representation	Min terms	Binary Representation
m_0	00000	m_0	00000	(0,1)	0000-	(0,1,4,5)	00-0-	(0,1,4,5,16,17,20,21)	-0-0-	~~0,1,4,5~~	~~4,5,12,13~~
m_1	00001	m_1	00001	(0,4)	00-00	(0,1,16,17)	-000-			4,5,12,13	0-10-
m_4	00100	m_4	00100	(0,16)	-0000	(0,4,16,20)	-0-00			4,12,20,28	--100
m_5	00101	m_{16}	10000	(1,5)	00-01	(1,5,17,21)	-0-01			5,17,13,15	0-1-1
m_7	00111	m_5	00101	(1,17)	-0001	(4,5,12,13)	0-10-			20,22,28,30	1-1-0
m_{12}	01100	m_{12}	01100	(4,5)	0010-	(4,5,20,21)	-010-			0,1,4,5,16,17,20,21	-0-0-
m_{13}	01101	m_{17}	10001	(4,20)	-0100	(4,20,12,28)	--100				
m_{15}	01111	m_{20}	10100	(16,17)	1000-	(16,17,20,21)	10-0				
m_{16}	10000	m_7	00111	(16,20)	10-00						
m_{17}	10001	m_{13}	01101	(5,7)	001-1						
m_{20}	10100	m_{21}	10101	(5,13)	0-101						
m_{21}	10101	m_{22}	10110	(5,21)	-0101						
m_{22}	10110	m_{28}	11100	(12,13)	0110-						
m_{28}	11100	m_{15}	01111	(12,28)	-1100						
m_{30}	11110	m_{30}	11110	(17,21)	10-01						
				(20,21)	1010-						
				(20,22)	101-0						
				(20,28)	1-100						
				(7,15)	0-111						
				(13,15)	011-1						
				(22,30)	1-110						
				(28,30)	111-0						

Prime Implicant Chart :-

Min terms	m_0	m_1	m_4	m_5	m_7	m_{12}	m_{13}	m_{15}	m_{16}	m_{17}	m_{20}	m_{21}	m_{22}	m_{28}	m_{30}
EPI (4,5,12,13)			x	x		x	x								
PPI (4,12,20,28)			x			x					x			x	
EPI (5,7,13,15)				x	x		x	x							
EPI (20,22,28,30)											x		x	x	x
(0,1,4,5,16,17,20,21) EPI	x	x	x	x					x	x	x	x			

$$F(A,B,C,D,E) = A'CD' + A'CE + ACE' + B'D'$$

Tabular Method (Quine-Mc Cluskey) :-

→ $\bar{A}BC$, $\bar{A}B\bar{C}$ } 1's place differs.

→ We can say that the minterms whose binary equivalent differ only in one place can be combined to reduce the minterms. This is the fundamental Principle of the Q.C Method.

Algorithm for generating Prime Implicants :-

1) Mark '.' according to minterms.

2) Compare each binary number with every term in the adjacent next higher category & if they differ only by one position, put a check mark and copy the term in the next column with '-' in the position that they differed.

3) Apply the same process described in step 2 for the resultant column & continue these cycles until a single pass through cycle yield no further elimination of literals

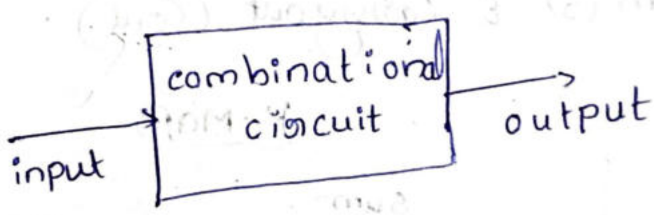
4) List all Prime Implicants

5) Select the minimum num of prime implicants which must cover all the minterms.

3. COMBINATIONAL LOGIC DESIGN

combinational logic circuit contains combination of logic gates. In combinational circuit output depends on present inputs only.

* we cannot store any information
* Timing required for this combinational circuit is less
* Hardware complexity is more.



Binary adders :-

In binary adders generally we are using half adder, full adder, ripple carry adder etc

(i) Half adder :-

Half adder contains two inputs A & B and two outputs sum (S) and carry (Cout).

The truth table of half adder is as follows

A	B	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K map :-

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

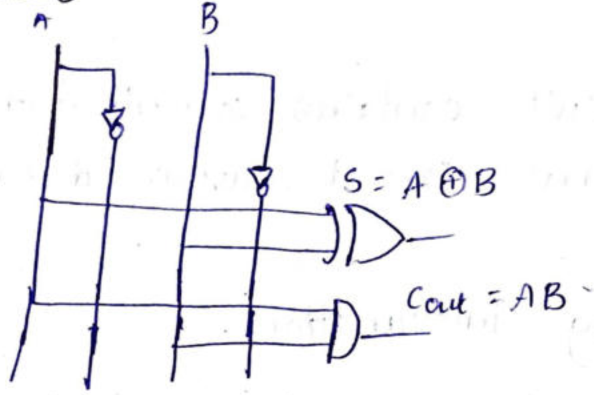
$S = \bar{A}B + A\bar{B}$

$S = A \oplus B$

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

$C_{out} = AB$

Logic diagram.



(2) Full adder :-

Full adder contains three inputs A, B, C_{in} and two outputs sum (S) & carryout (C_{out}).

Truthtable :-

Input			output	
A	B	Cin	S	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

K-Maps

sum :-

	$\bar{B}\bar{C}_{in}$ 00	$\bar{B}C_{in}$ 01	BC_{in} 11	$B\bar{C}_{in}$ 10
$\bar{A}$ 0	2 (0)	1 (1)	(3)	1 (2)
A 1	1 (4)	(5)	1 (7)	(6)

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

$$= \bar{A}(\bar{B}C_{in} + B\bar{C}_{in}) + A(\bar{B}\bar{C}_{in} + BC_{in})$$

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

$$sum = A \oplus B \oplus C_{in}$$

carry :-

	$\bar{B}\bar{C}_{in}$	$\bar{B}C_{in}$	BC_{in}	$B\bar{C}_{in}$
$\bar{A}$	(0)	(1)	1 (3)	(2)
A	(4)	1 (5)	1 (7)	1 (6)

$$C_{out} = AC_{in} + AB + BC_{in}$$

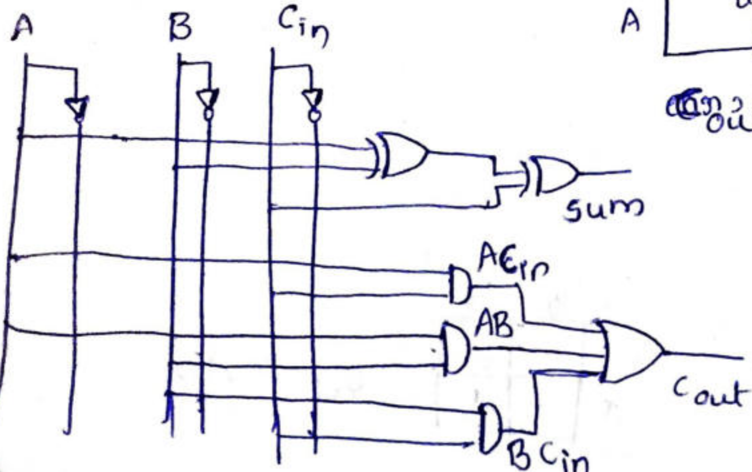
$$= AB + BC_{in}(A+\bar{A}) + AC_{in}(B+\bar{B})$$

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

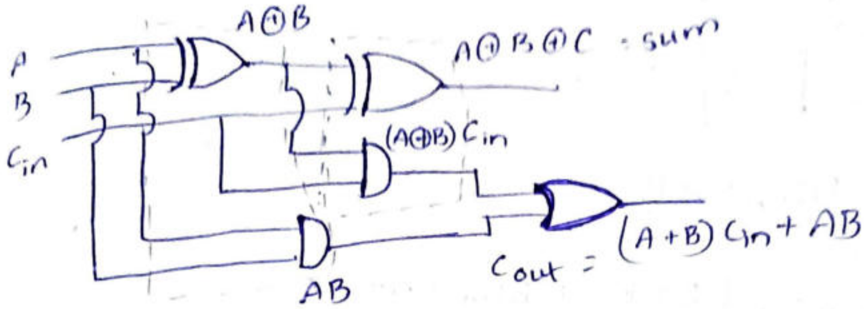
$$= AB(1 + C_{in} + C_{in}) + C_{in}(\bar{A}B + A\bar{B})$$

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

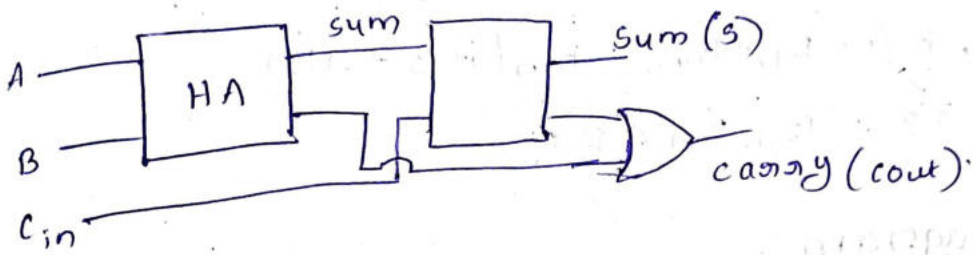
Logic diagram :-



(3) Full adder using two half adder:



Block diagram:-



(4) Half subractor:-

Truth table:-

A	B	D	B_{out}
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-maps:-

difference:-

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

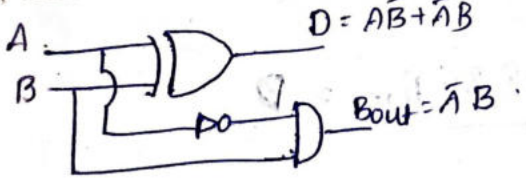
$D = A\bar{B} + \bar{A}B$

B_{out}:-

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

$B_{out} = \bar{A}B$

logic diagram:-



(5) Full subractor:-

Truth table:-

A	B	B_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0

K-maps:-

difference:

	$\bar{B}\bar{B}_{in}$ 00	$\bar{B}B_{in}$ 01	BB_{in} 11	$B\bar{B}_{in}$ 10
$\bar{A}$ 0		1		1
A 1	1		1	

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

$= A'(\bar{B}B_{in} + B\bar{B}_{in}) + A(\bar{B}\bar{B}_{in} + BB_{in})$

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

$D = A \oplus B \oplus B_{in}$

B_{out} :-

	$\bar{B}\bar{B}_{in}$ 00	$\bar{B}B_{in}$ 01	BB_{in} 11	$B\bar{B}_{in}$ 10
$\bar{A}$ 0	0	1 1	3 1	2 1
A 1	4	5	7 1	6

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

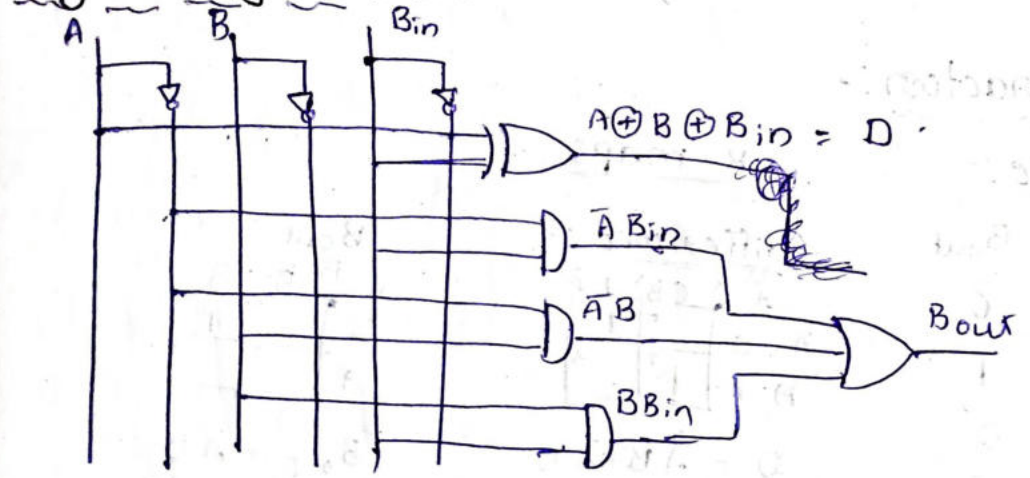
$$= \bar{A}B + \bar{A}B_{in}(B+\bar{B}) + BB_{in}(A+\bar{A})$$

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

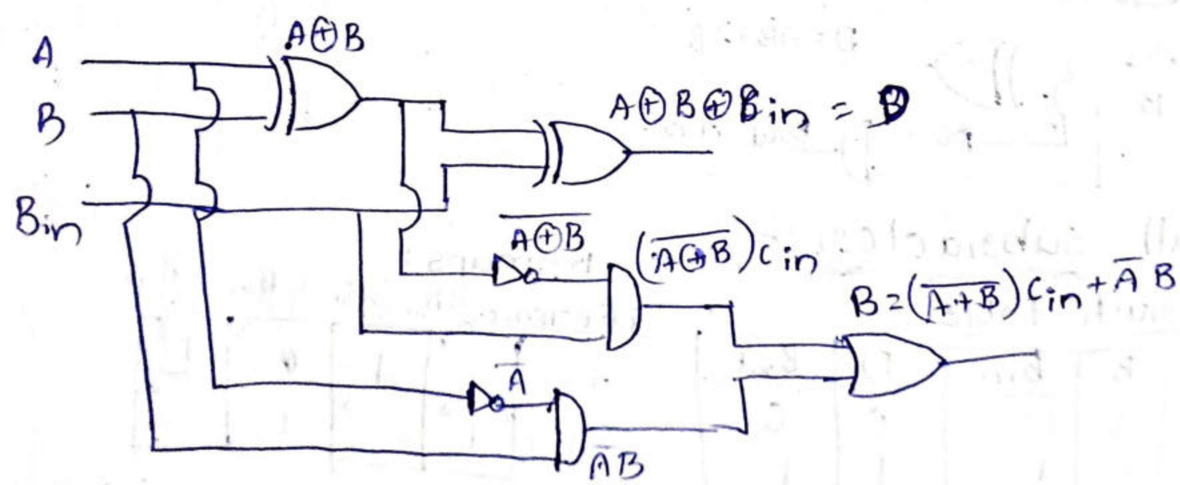
$$= \bar{A}B(1 + B_{in} + B_{in}) + B_{in}(\bar{A}\bar{B} + AB)$$

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

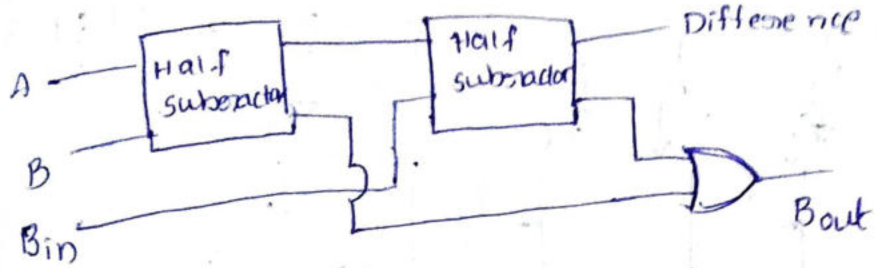
Logic diagram :-



⑥ Full subractor using two Half subractors.



Block diagram :-



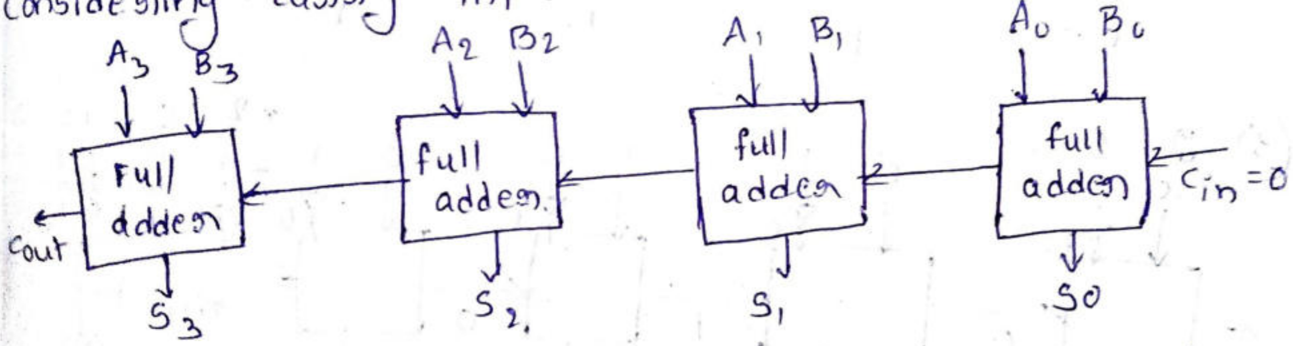
10/10.

1 Ripple carry adder :-

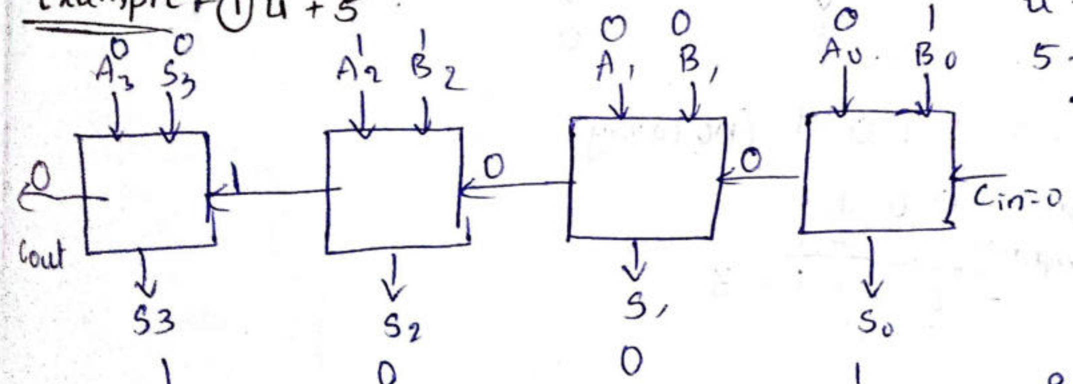
Ripple carry adder is also known as binary adder or parallel adder.

4-bit parallel adder :-

In 4-bit parallel adder we have 4-full adders by considering carry input as '0'.



Example :- ① 4 + 5



$A_3 A_2 A_1 A_0$

4 - 0100 A

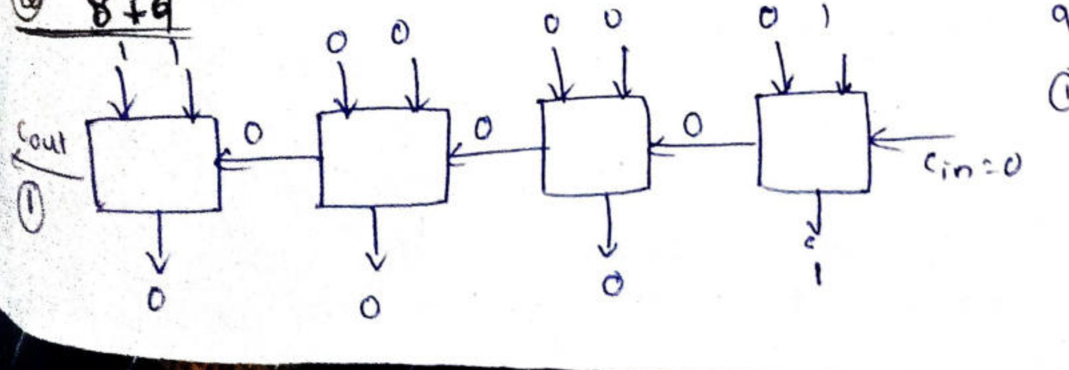
5 - 0101 B

1001

$S_3 S_2 S_1 S_0$

9

② **8 + 9**



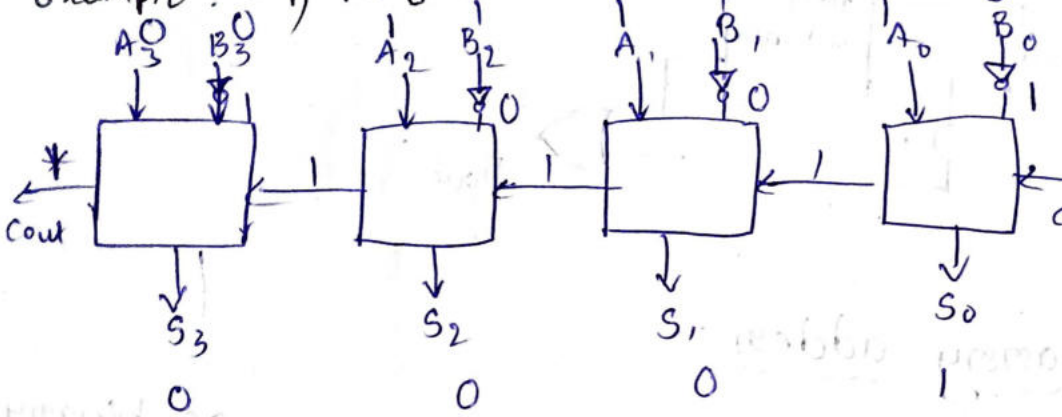
8 - 1000 - A

9 - 1001 - B

① - 0001 = 1

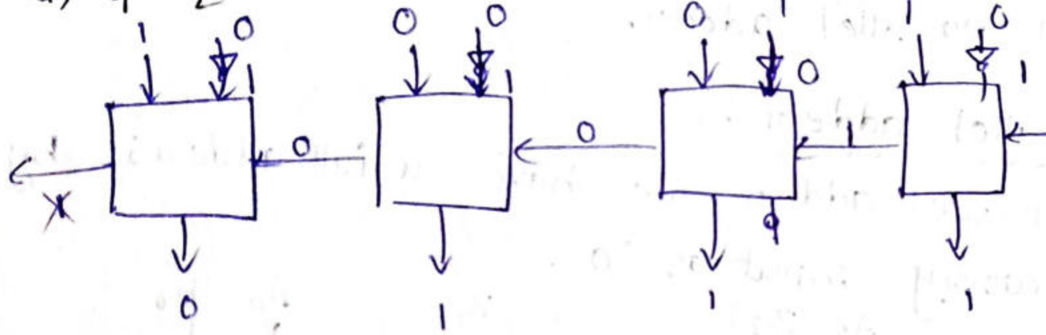
4-bit parallel subtractor :-

Example : - 1) 7 - 6



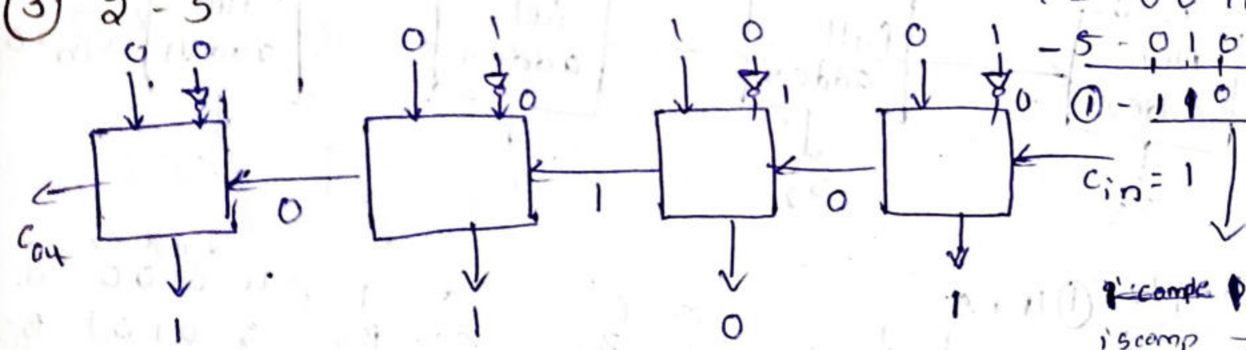
7 - 0111 - A
6 - 0110 - B
0001

2) 9 - 2



9 - 1001
-2 - 0010
0111
7

③ 2 - 5



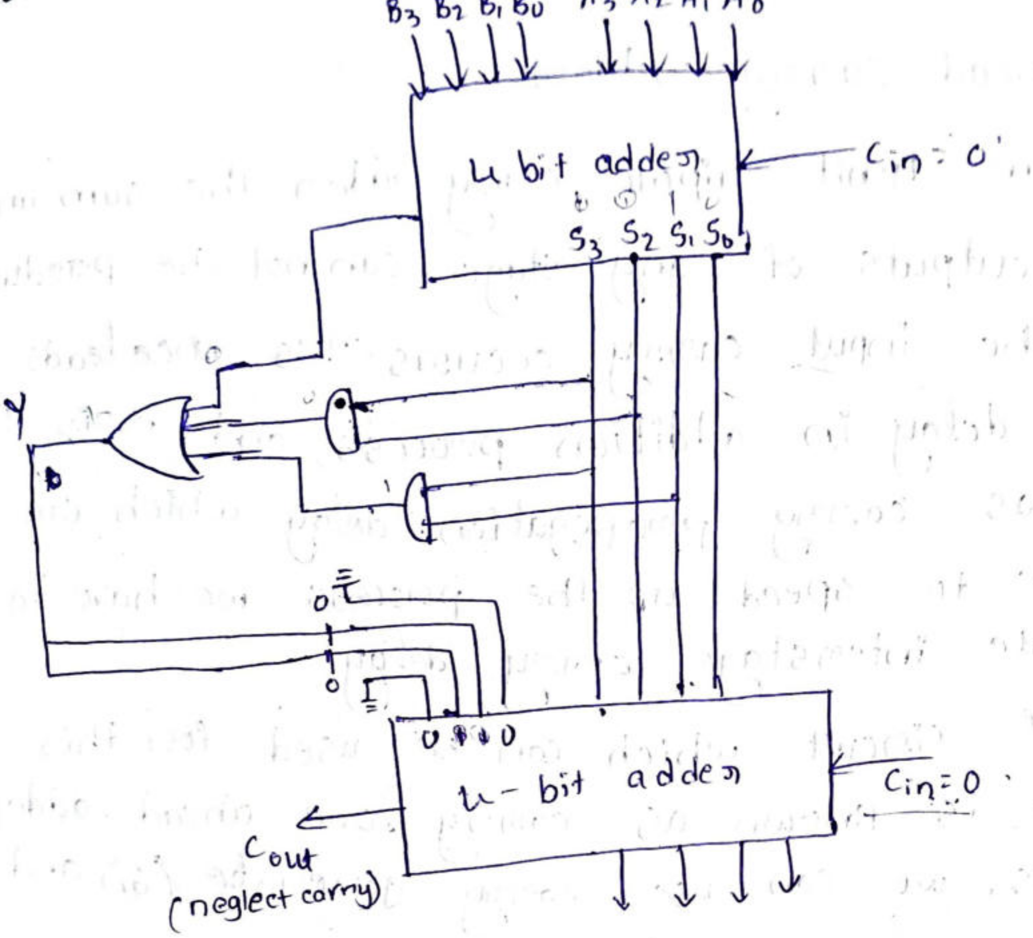
2 - 0010
-5 - 0101
① - 1101

1's comple 0010
2's comp +1
0011

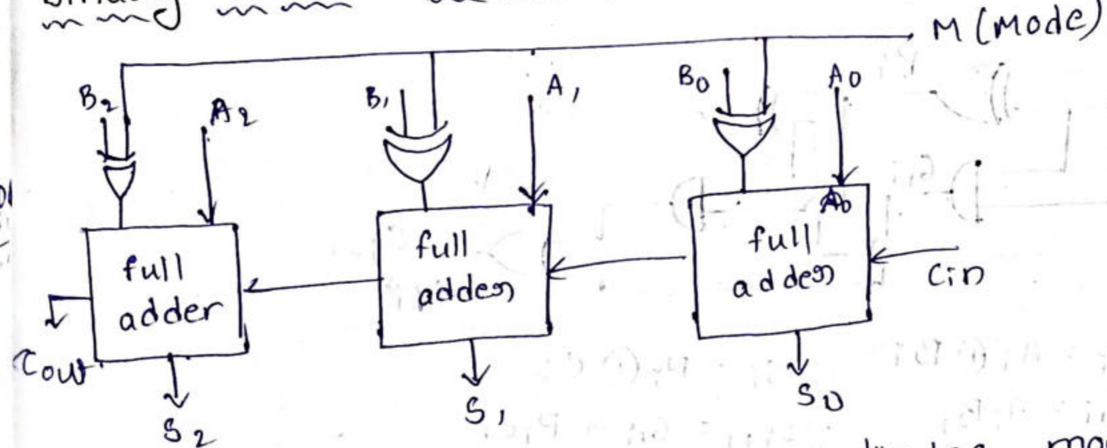
=> 1 1 0 1 (no carry)

so, 1's comple => 0 0 1 0
2's comple => + 1
- 0 0 1 1 = 3

⑧ Decimal adder   BCD adder :-



Binary adder - subtractor :-



* In adder-subtractor circuit M indicates mode.
* For adder, consider $m = 0$, $cin = 0$.
* For subtractor, $m = 1$, $cin = 1$ and C_{out} has to be neglected.

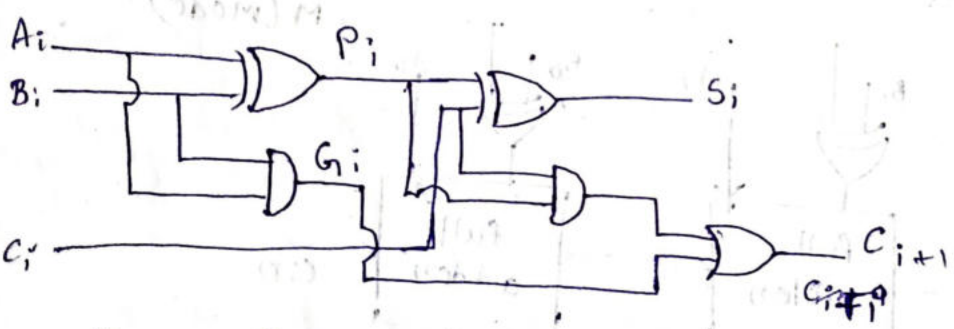
NOTE:-
For n-bit adder-subtractor we have to take n full adder. For example, if it is a add 3-bit

adder-subtractor circuit, we ~~have~~ add 3 full adder

Look-ahead carry Adder :-

From 4-bit ripple carry adder the sum and carry outputs of any stage can not be produced until the input carry occurs. This ~~the~~ leads to a time delay in addition process, and it is also known as carry propogation delay. which can be overcome to speed up the process we have to eleminate interstage carry delay.

The circuit which can be used for this purpose is known as carry look ahead adder. In this, we can use carry generate (G) and carry propogate.



$P_i = A_i \oplus B_i$ $\quad S_i = P_i \oplus C_i$

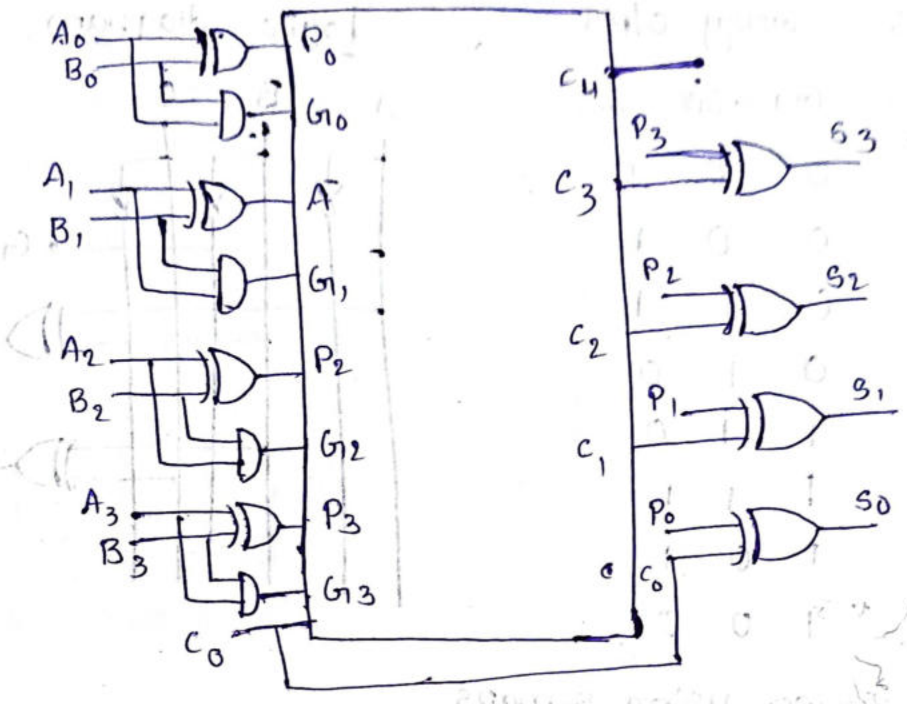
$G_i = A_i B_i$ $\quad C_{i+1} = G_i + P_i C_i$

$P_0 = A_0 \oplus B_0$ $\quad G_0 = A_0 B_0$ $\quad S_0 = P_0 \oplus C_0$ $\quad C_{i+1} \Rightarrow C_1 = G_0 + P_0 C_0$

$P_1 = A_1 \oplus B_1$ $\quad G_1 = A_1 B_1$ $\quad S_1 = P_1 \oplus C_1$ $\quad C_{i+2} \Rightarrow C_2 = G_1 + P_1 C_1$

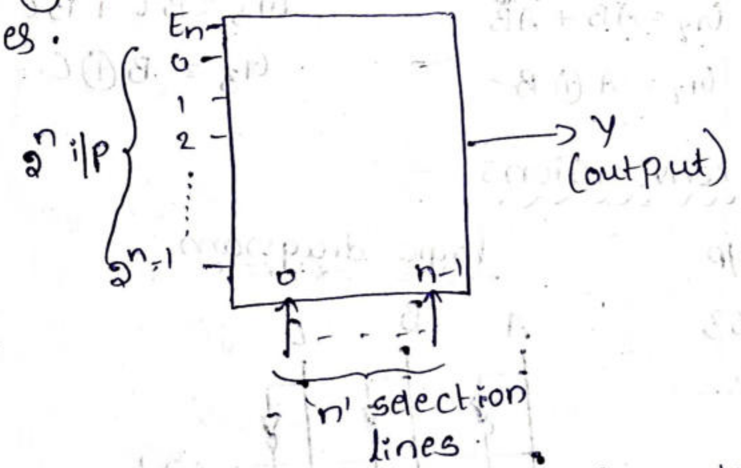
$P_2 = A_2 \oplus B_2$ $\quad G_2 = A_2 B_2$ $\quad S_2 = P_2 \oplus C_2$ $\quad C_3 = G_2 + P_2 C_2$

$P_3 = A_3 \oplus B_3$ $\quad G_3 = A_3 B_3$ $\quad S_3 = P_3 \oplus C_3$ $\quad C_4 = G_3 + P_3 C_3$



Multiplexers :-

Multiplexer is a combinational circuit which is having 2^n inputs and one output with n selection lines.



It is a digital switch, the selection of i/p lines are controlled by selection lines. That's why, MUX is also known as data selector and it is also known as "many-to-one digital circuit".

NOTE:-

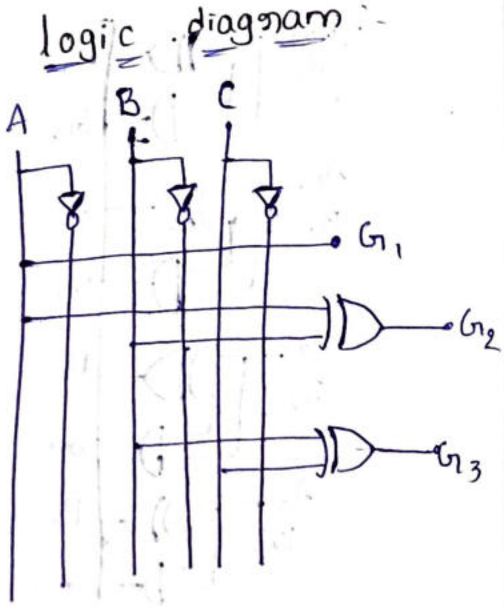
In multiplexer we have 2^n i/p's and each input size will be n bits. For example, let us consider 2 input, 4-bit MUX which is having 2 inputs and 1 output where each and every input size is 4-bits.

3 bit binary to Grey conversions :-

Binary i/p's — Grey o/p's

A	B	C	G1	G2	G3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

logic diagram :-



Finding expression using K-maps

G1 :-

A \ BC	00	01	11	10
0				
1	1	1	1	1

$G_1 = A$

G2 :-

A \ BC	00	01	11	10
0			1	1
1	1	1		

$G_2 = A\bar{B} + \bar{A}B$

$G_2 = A \oplus B$

G3 :-

A \ BC	00	01	11	10
0		1		1
1		1		1

$G_3 = \bar{B}C + B\bar{C}$

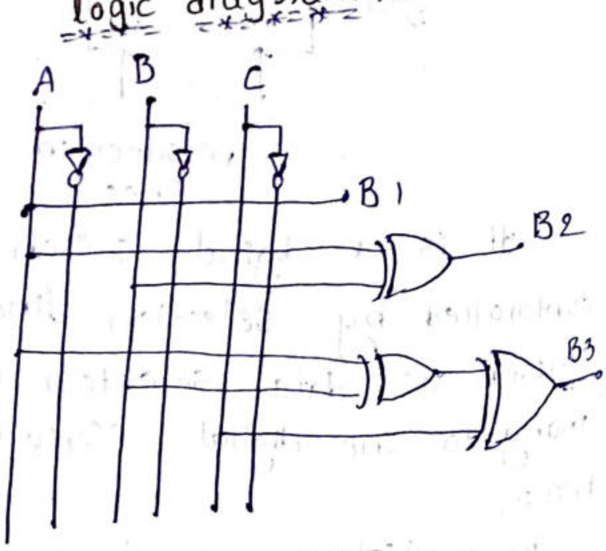
$G_3 = B \oplus C$

Grey to binary conversions :-

Grey i/p's — Binary o/p

A	B	C	B1	B2	B3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	1	1	0
1	1	0	1	0	0
1	1	1	1	0	1

logic diagram :-



K maps :-

B1 :-

A \ BC	00	01	11	10
0				
1	1	1	1	1

$B_1 = A$

B2 :-

A \ BC	00	01	11	10
0			1	1
1	1	1		

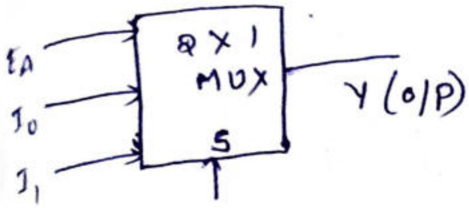
$B_2 = A\bar{B} + \bar{A}B = A \oplus B$

B3 :-

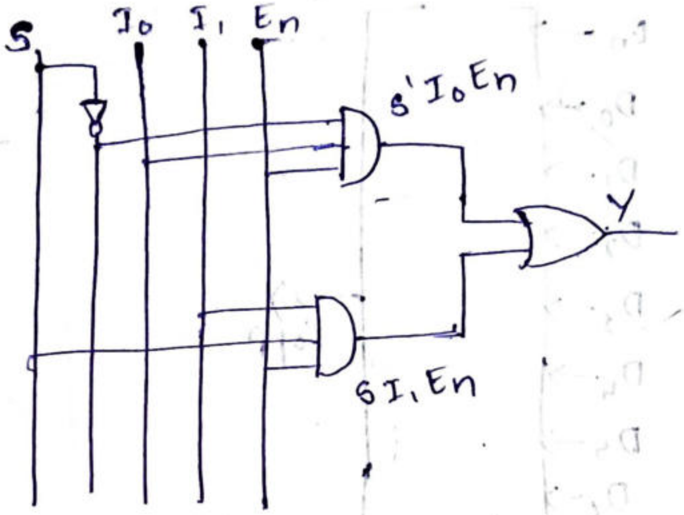
A \ BC	00	01	11	10
0		1		1
1	1		1	

$B_3 = A \oplus B \oplus C$

(1) 2×1 MUX :- (2 inputs, 1-bit MUX)



logic diagram :-

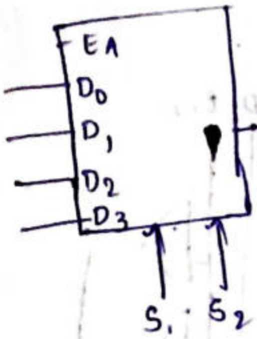


Truth table :-

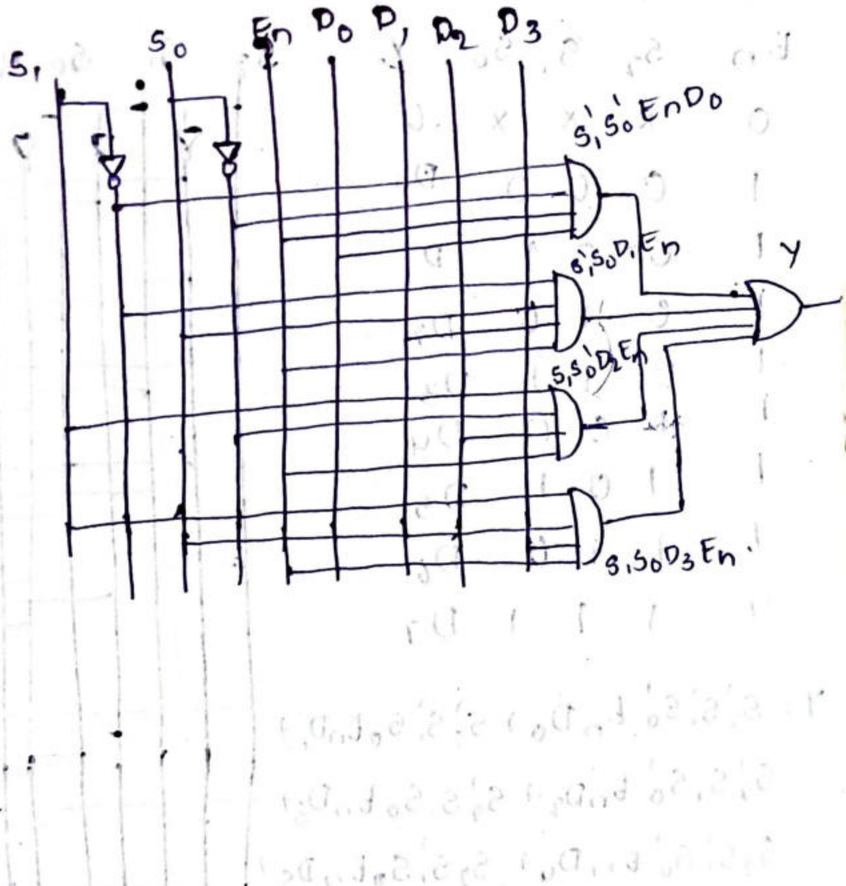
E_n	S	Y (O/P)
0	X	0
1	0	I_0
1	1	I_1

$Y = S'I_0E_n + SI_1E_n$

(2) 4×1 MUX :- (4 inputs, 1 bit mux) :-



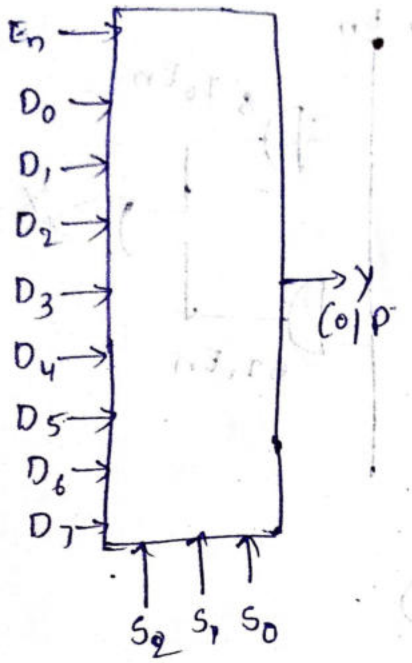
logic diagram :-



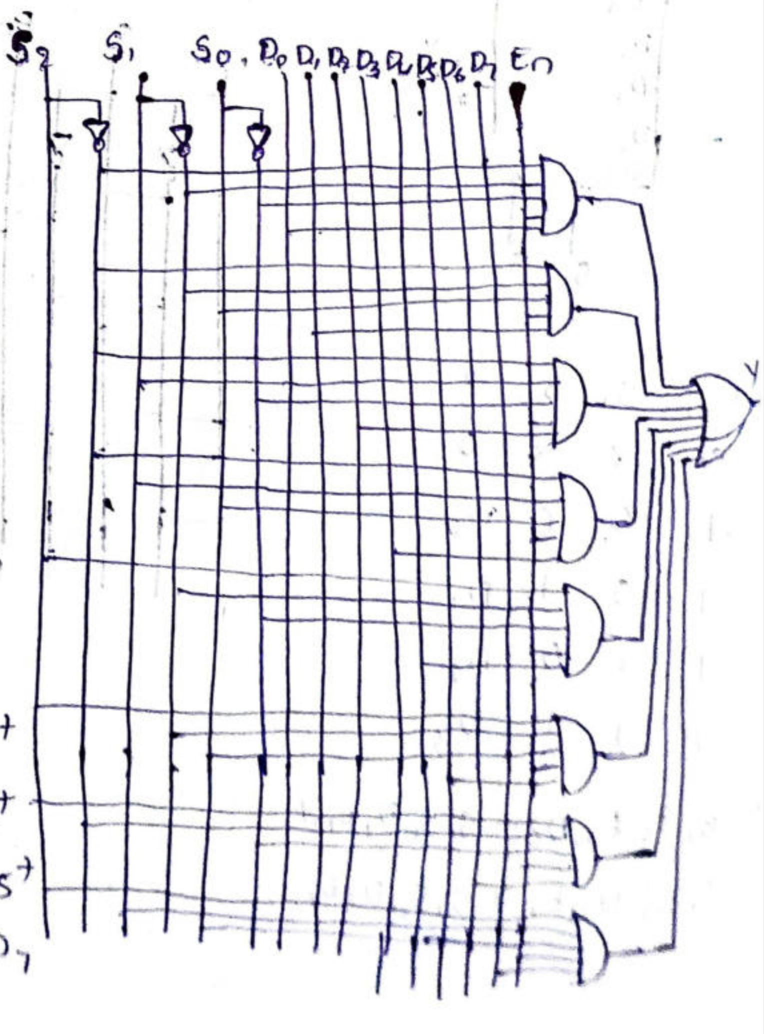
E_n	S_1	S_0	Y
0	X	X	0
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3

$Y = S_1'S_0'E_nD_0 + S_1'S_0D_1E_n + S_1S_0'D_2E_n + S_1S_0D_3E_n$

(3) 8 x 1 Mux (8 inputs, 1-bit Mux)

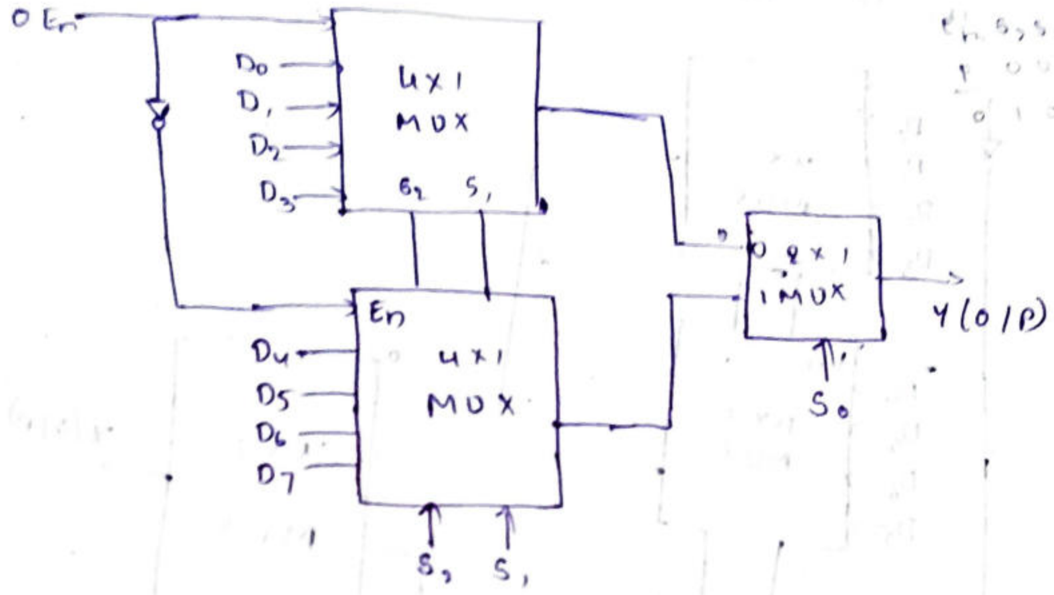


En	S_2	S_1	S_0	Y
0	x	x	x	0
1	0	0	0	D_0
1	0	0	1	D_1
1	0	1	0	D_2
1	0	1	1	D_3
1	1	0	0	D_4
1	1	0	1	D_5
1	1	1	0	D_6
1	1	1	1	D_7

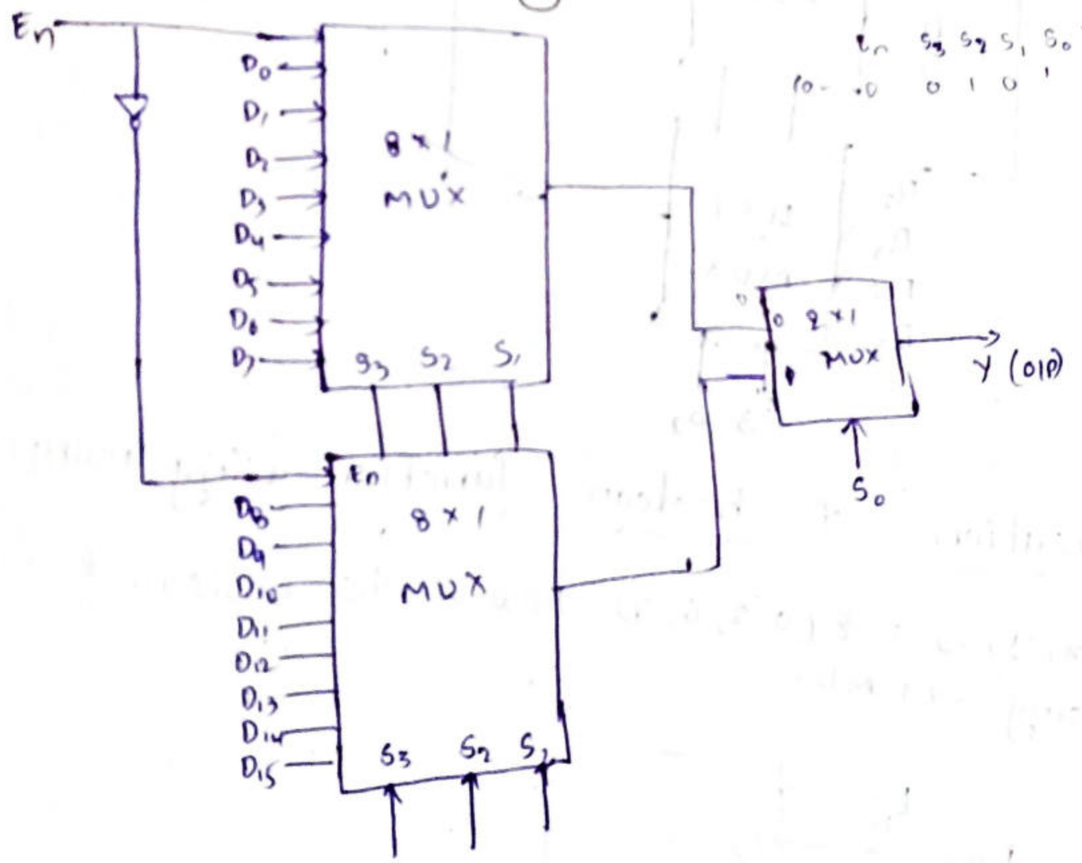


$$Y = S_2'S_1'S_0'E_nD_0 + S_2'S_1'S_0E_nD_1 + S_2'S_1S_0'E_nD_2 + S_2'S_1S_0E_nD_3 + S_2S_1'S_0'E_nD_4 + S_2S_1'S_0E_nD_5 + S_2S_1S_0'E_nD_6 + S_2S_1S_0E_nD_7$$

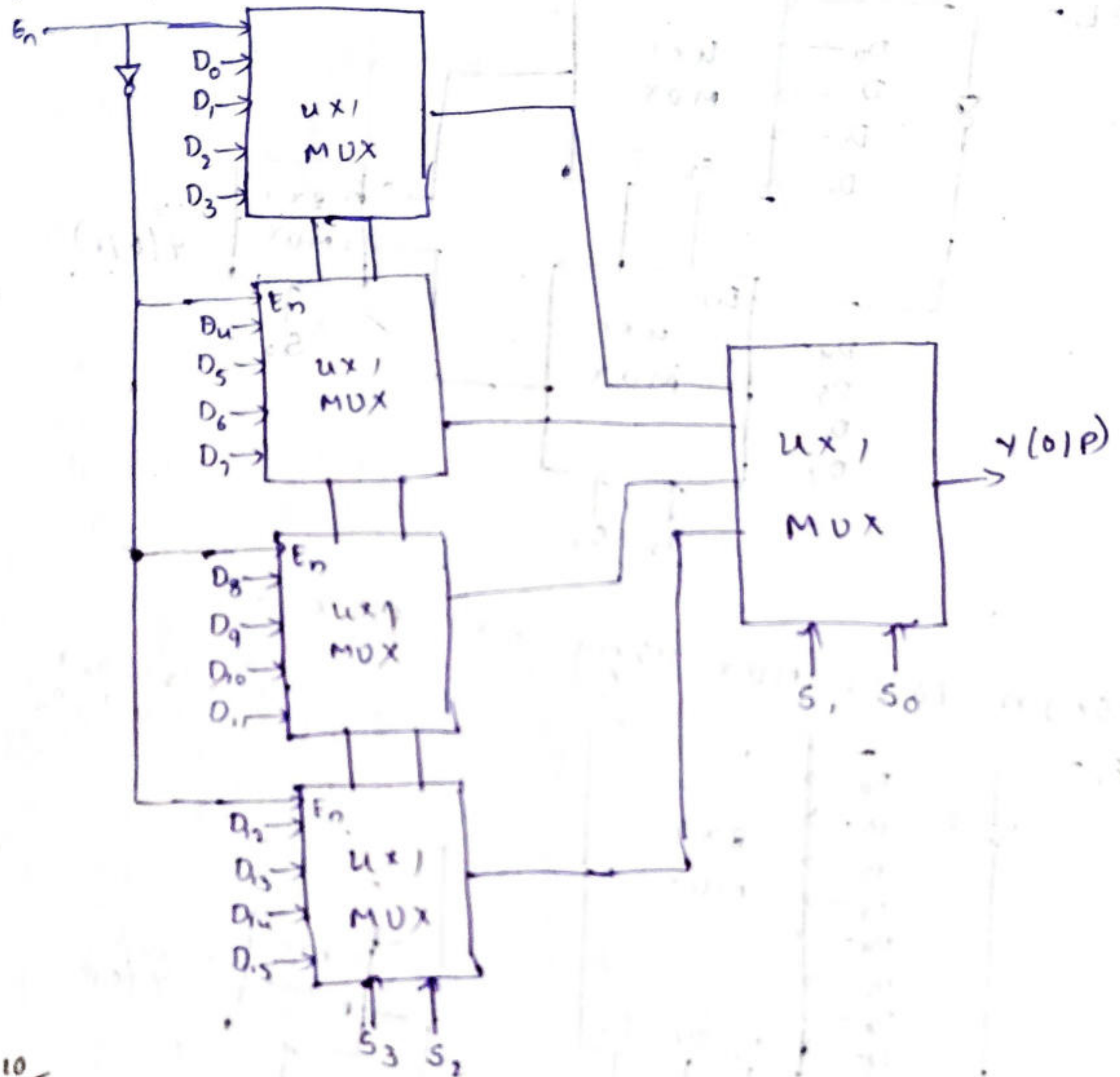
a) Design 8 X 1 MUX using two 4X1 MUX :-



b) Design 16 X 1 MUX using 8X1 MUX :



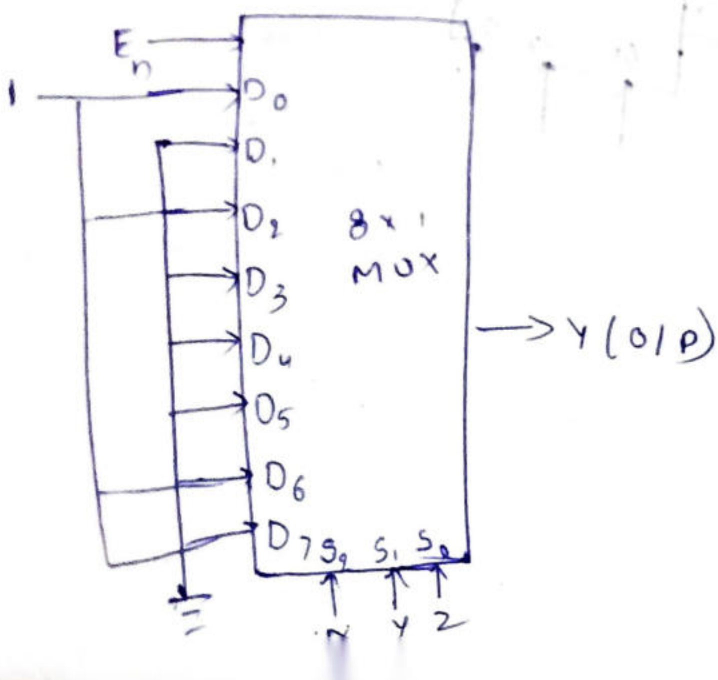
c) Design 16 x 1 MUX using 4 x 1 MUX.



16/10

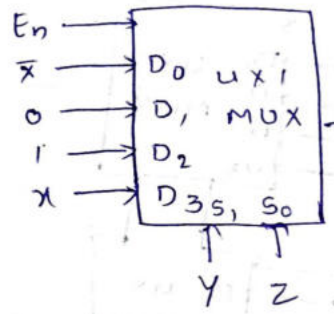
Realization of boolean function using multiplexers

1) $F(x, y, z) = \Sigma(0, 2, 6, 7)$ Realise the boolean function using 8 x 1 MUX



2) $F(x, y, z) = \Sigma(0, 2, 6, 7)$ realize the boolean function using 4×1 MUX.

circled



	D_0	D_1	D_2	D_3
$\bar{x}$	0	1	2	3
x	4	5	6	7
	$\bar{x}$	0	1	x

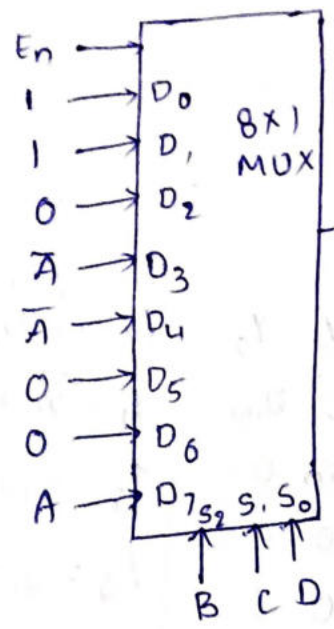
p)

* If the two minterms in a coloumn are not circled '0' is applied to the MUX input.
* If the two minterms in a coloumn are circled '1' is applied to the MUX input.
* If the minterms in the second row is circled and the minterm in the first row is not circled, 'x' is applied to the corresponding MUX input
* If the minterm in the first row is circled, and the minterm in the second row is not circled, '$\bar{x}$' is applied to the corresponding MUX input.

H.W)

3) $F(A, B, C) = \Sigma(1, 3, 5, 6)$ realise the boolean function using 8×1 MUX & 4×1 MUX.

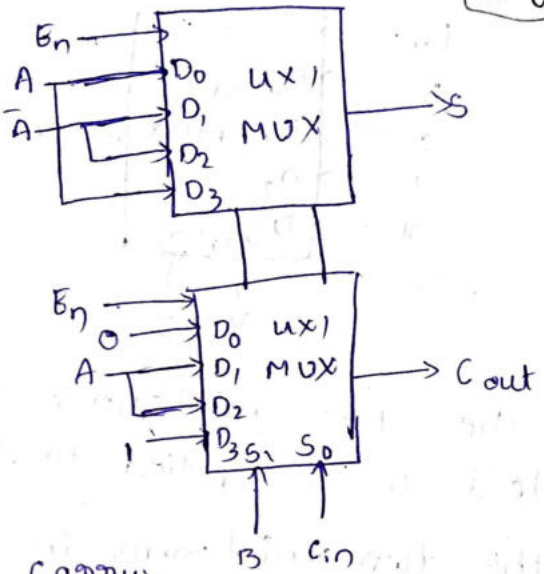
4) $F(A, B, C, D) = \Sigma(0, 1, 3, 4, 8, 9, 15)$ realise the boolean function using 8×1 MUX



	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A

5) Implement full adder using 4x1 MUX / H.W full subractor using 4x1 MUX 1x8

A	B	C_{in}	S	C_{out}
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



Sum :-

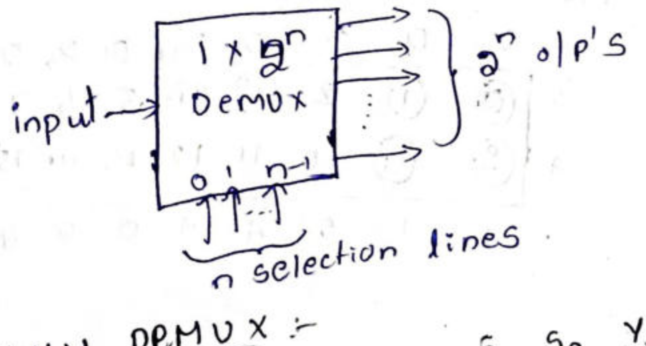
	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	A	$\bar{A}$	$\bar{A}$	A

Carry :-

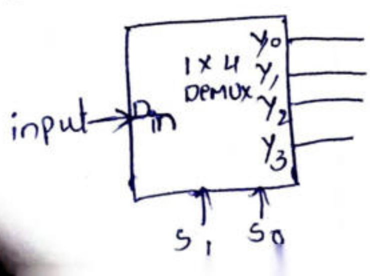
	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	0	A	A	1

DeMultiplexer :-

DeMultiplexer is a combinational circuit, which is having one input and 2^n output with n selection lines. Depending on the selection lines any one of the output contains input, which is the reverse operation of Multiplexer.



① 1x4 DEMUX :-



S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	Din
0	1	0	0	Din	0
1	0	0	Din	0	0
1	1	Din	0	0	0

$Y_0 = S_1' S_0' \, Din$

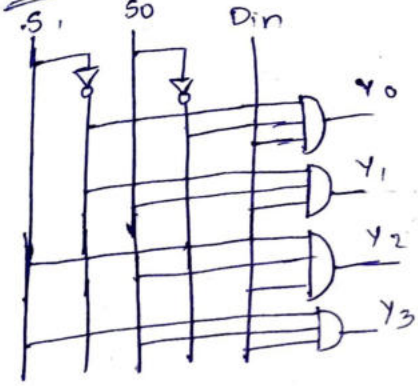
$Y_1 = S_1' S_0 \, Din$

$Y_2 = S_1 S_0' \, Din$

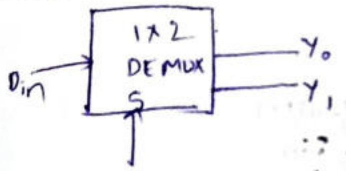
$Y_3 = S_1 S_0 \, Din$

H.W
full subtractor
using 4x1 MUX
1x8

logic diagram :-



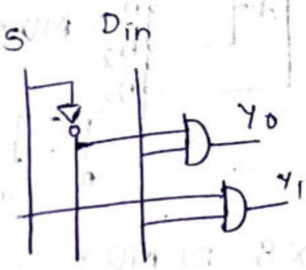
2) 1x2 DEMUX :-



S	Y_0	Y_1
0	Din	0
1	0	Din

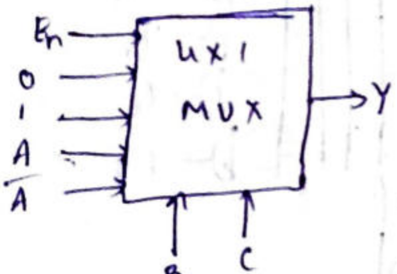
$Y_0 = S'Din$

$Y_1 = S\,Din$



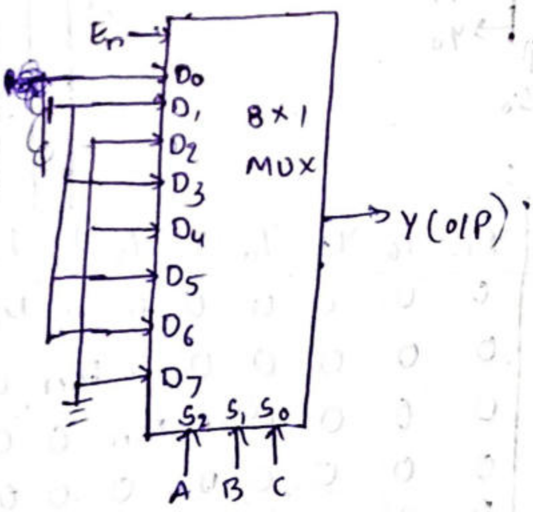
3. Ans) Given that $F(A,B,C) = \Sigma(1,3,5,6)$

4x1 MUX :-



	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	0	1	A	$\bar{A}$

8x1 MUX :-



h is
on
of
se

6.A) Implement full subtractor using 4x1 MUX.

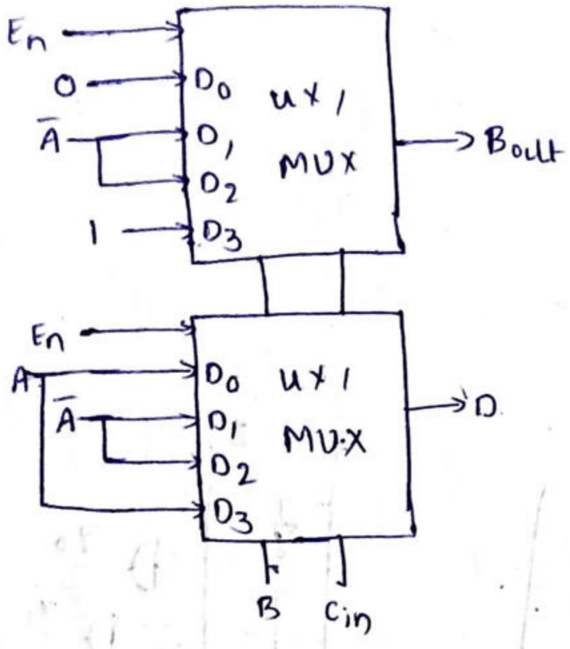
A	B	Cin	Bout	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Difference :-

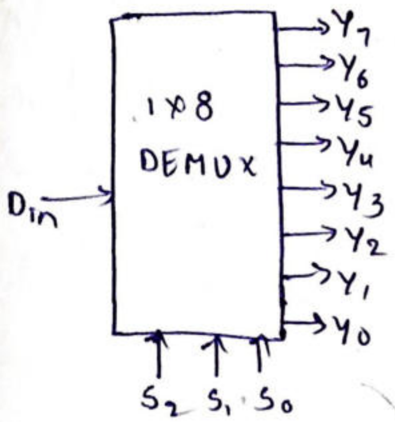
	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	A	$\bar{A}$	$\bar{A}$	A

Bout :-

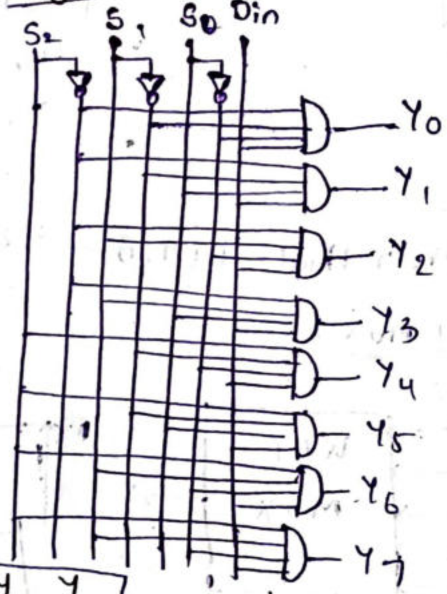
	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	3
A	4	5	6	7
	0	$\bar{A}$	$\bar{A}$	1



③ 1 × 8 DEMUX :-



logic diagram :-



Truthtable :-

S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	Din
0	0	1	0	0	0	0	0	0	Din	0
0	1	0	0	0	0	0	0	Din	0	0
0	1	1	0	0	0	0	Din	0	0	0
1	0	0	0	0	0	Din	0	0	0	0
1	0	1	0	0	Din	0	0	0	0	0
1	1	0	0	Din	0	0	0	0	0	0
1	1	1	Din	0	0	0	0	0	0	0

$Y_0 = S_2' S_1' S_0' D_{in}$, $Y_1 = S_2' S_1' S_0 D_{in}$, $Y_2 = S_2' S_1 S_0' D_{in}$,

$Y_3 = S_2' S_1 S_0 D_{in}$, $Y_4 = S_2 S_1' S_0' D_{in}$, $Y_5 = S_2 S_1' S_0 D_{in}$

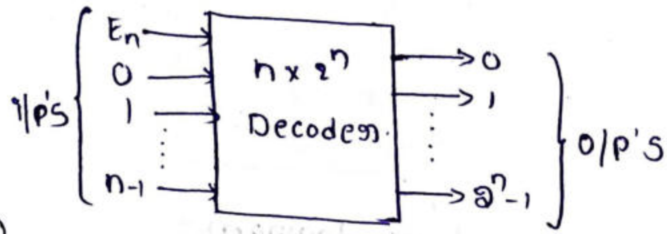
$Y_6 = S_2 S_1 S_0' D_{in}$, $Y_7 = S_2 S_1 S_0 D_{in}$

16/10

Decoder :-

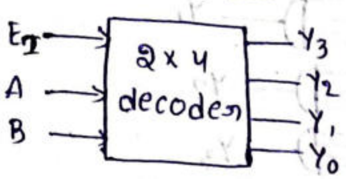
Decoder is a combinational circuit which is having n inputs and 2^n outputs. It is used in receiver side of the communication system.

logic diagram :-

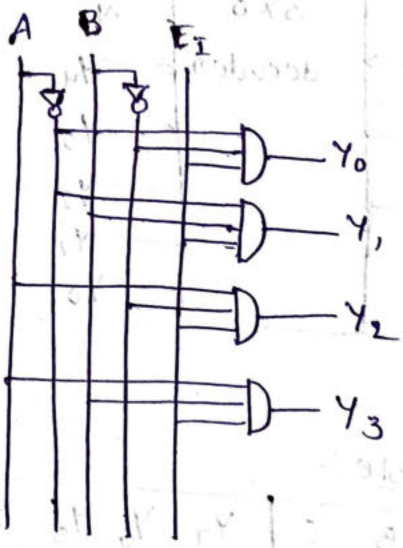


① ⓐ 2x4 decoder :-

Block diagram :-



logic diagram :-



Truth table :-

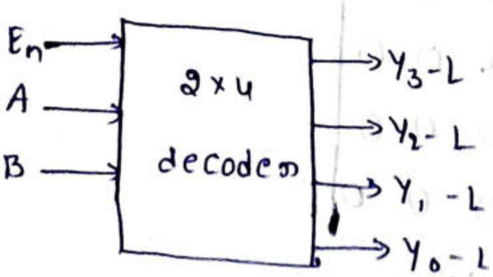
E_I	A	B	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

$Y_0 = A'B'E_I$, $Y_1 = A'BE_I$

$Y_2 = AB'E_I$, $Y_3 = ABE_I$

ⓑ 2x4 decoder with low outputs

Block diagram :-

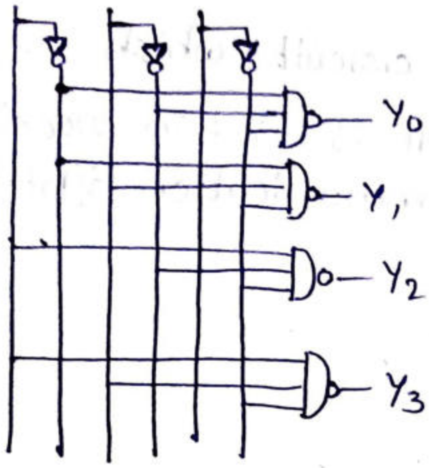


Truthtable :-

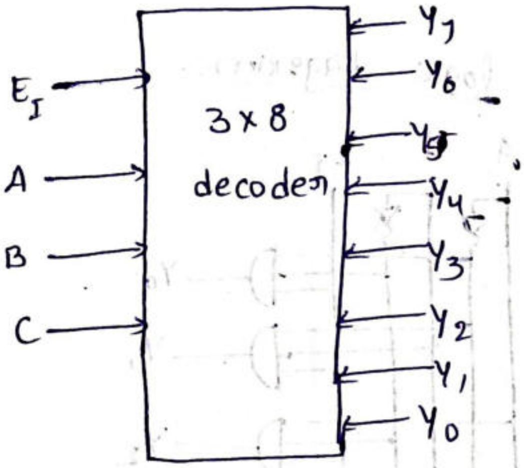
E_I	A	B	Y_3	Y_2	Y_1	Y_0
1	X	X	1	1	1	1
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1

$Y_0 = A'B'E_I'$, $Y_1 = A'BE_I'$, $Y_2 = AB'E_I'$, $Y_3 = ABE_I'$.

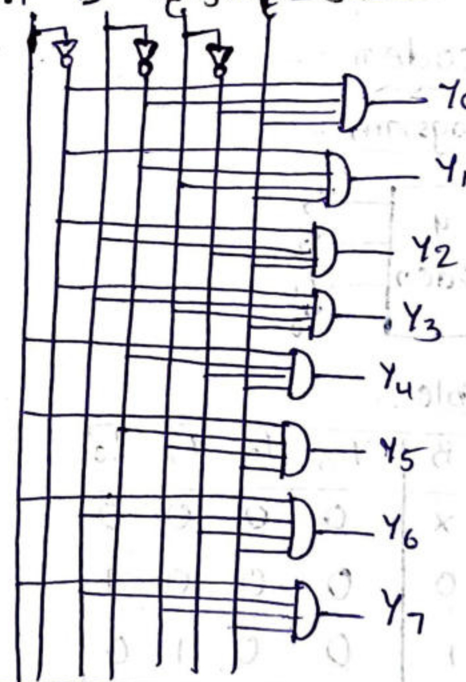
logic diagram :-



② a) 3 x 8 decoder :-



logic diagram:-



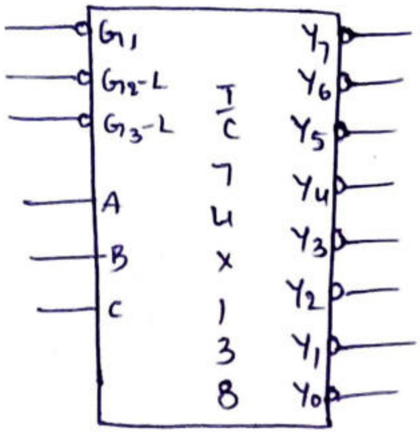
Truth table :-

E_I	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

$Y_0 = A'B'C'E_I$, $Y_1 = A'B'CE_I$, $Y_2 = A'BC'E_I$, $Y_3 = A'BCE_I$.

$Y_4 = AB'C'E_I$, $Y_5 = AB'CE_I$, $Y_6 = ABC'E_I$, $Y_7 = ABCE_I$

b) 3 x 8 decoder (74 x 138 IC)

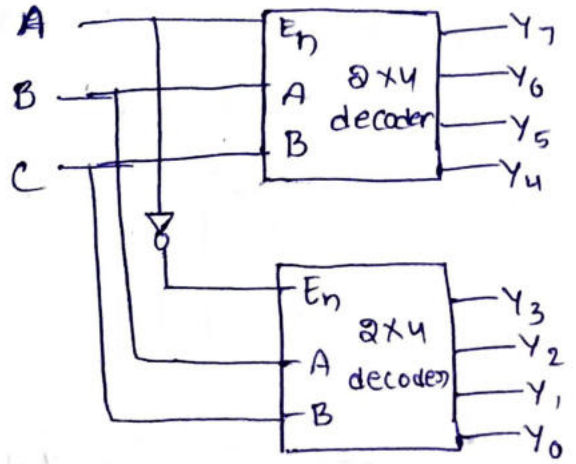


Truthtable :-

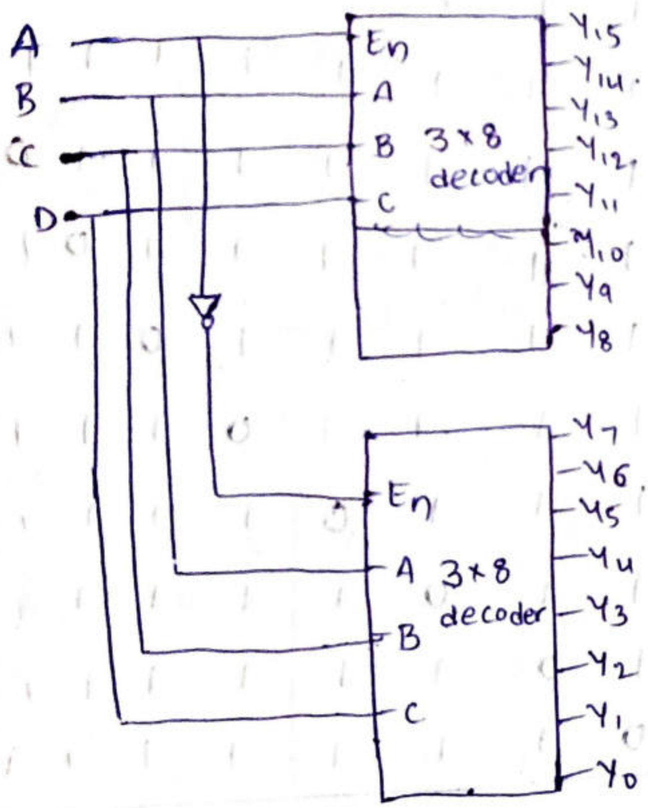
G1	G2-L	G3-L	A	B	C	Y7-L	Y6-L	Y5-L	Y4-L	Y3-L	Y2-L	Y1-L	Y0-L
0	X	X	X	X	X	1	1	1	1	1	1	1	1
0	1	X	X	X	X	1	1	1	1	1	1	1	1
0	0	1	X	X	X	1	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1

11/10

Design 3x8 Decoder using 2x4 Decoder :-



Design 4x16 using 3x8 decoder.

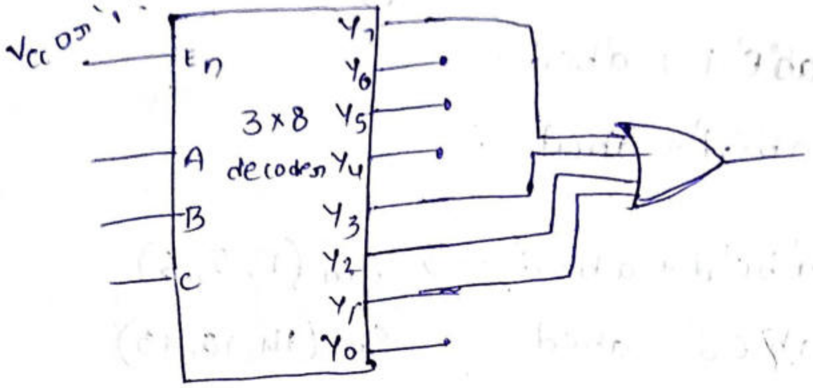


Realisation of boolean function using decoder

① Implement the following boolean function using 3x8 decoder. $F = \Sigma m(1, 2, 3, 7)$.

Y_4 = ..

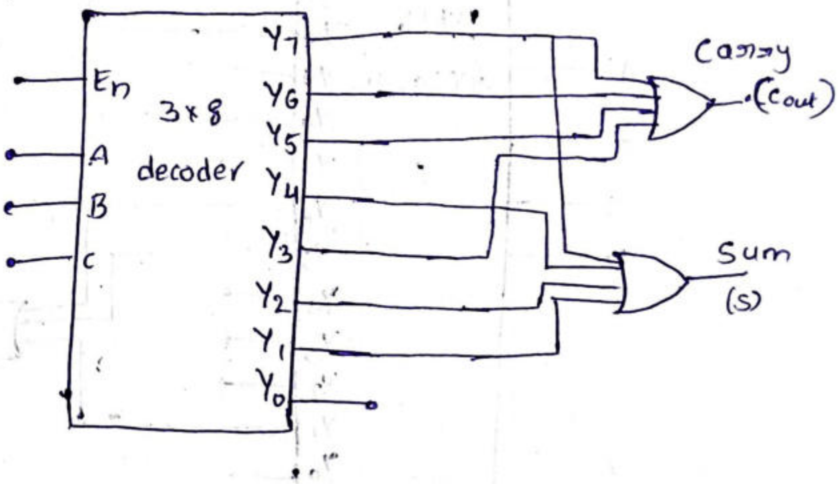
Given $F = \Sigma m(1, 2, 3, 7)$



2) Implement full adder using 3×8 decoder.
3) Implement full subtractor using 3×8 decoder.

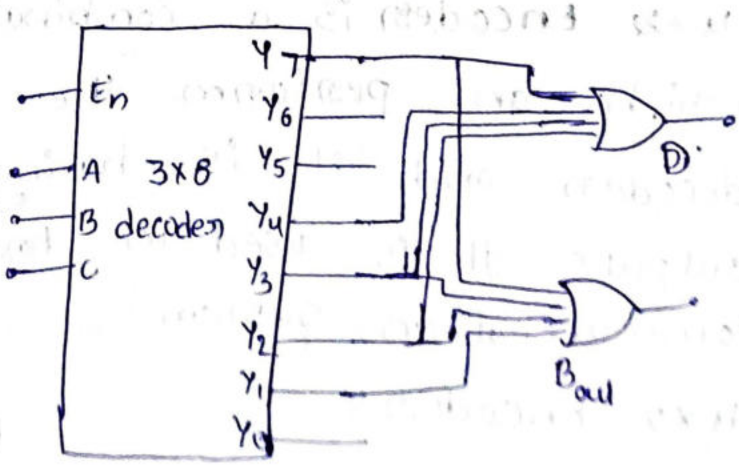
2. Ans) Full adder

A	B	C	S	C_{out}
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



3.
3. A) Full Subtractor :-

A	B	C	B_{out}	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	0	1	0	0
1	1	0	1	1



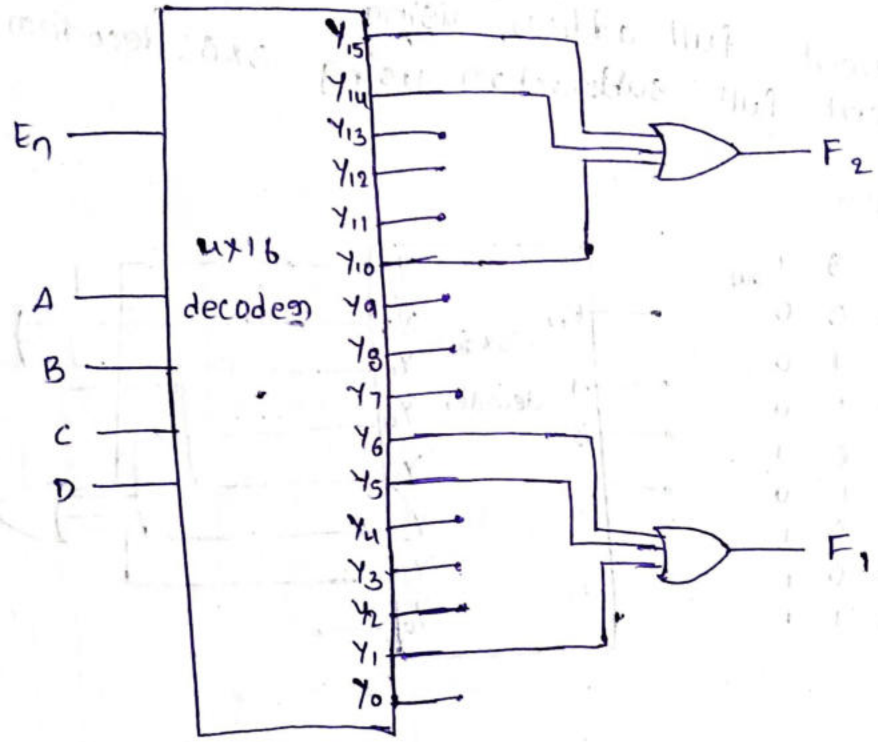
4) Implement a combinational logic circuit and defined by a function using decoder.

$F_1 = a'b'c'd + a'bc'd + a'bcd'$

$F_2 = abcd' + ab'cd' + abcd$

Ans) $F_1 = a'b'c'd + a'bc'd + a'bcd' \Rightarrow \Sigma m(1, 5, 6)$

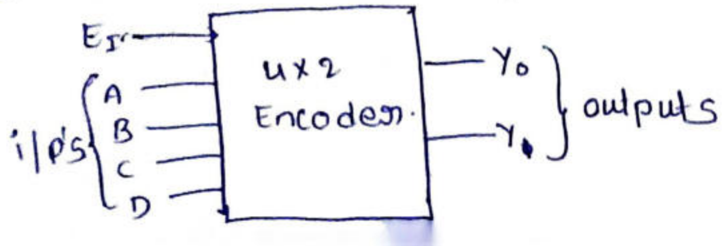
$F_2 = abcd' + ab'cd' + abcd \Rightarrow \Sigma m(14, 10, 15)$



ENCODER :-

1) Encoder is a combination logic circuit, which can perform the reverse operation of decoder and it is having 2^n inputs and n outputs. It is used in transmission side of the communication system.

1) 4x2 Encoder :-



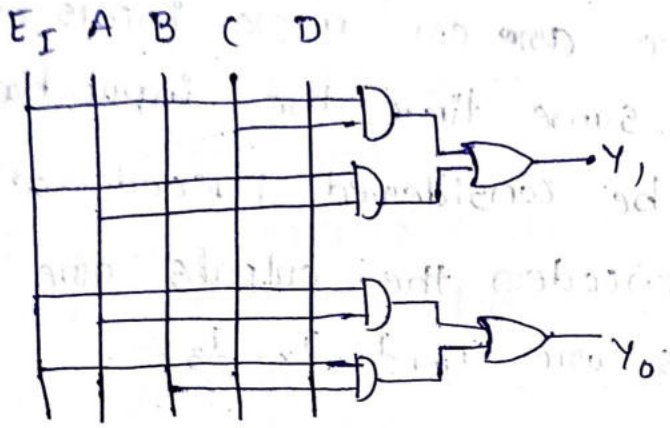
Truth table :-

E_I	A	B	C	D	Y_0	Y_1
0	X	X	X	X	0	0
1	0	0	0	1	0	0
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	1	0	0	0	1	1

$Y_1 = E_I C + E_I A$

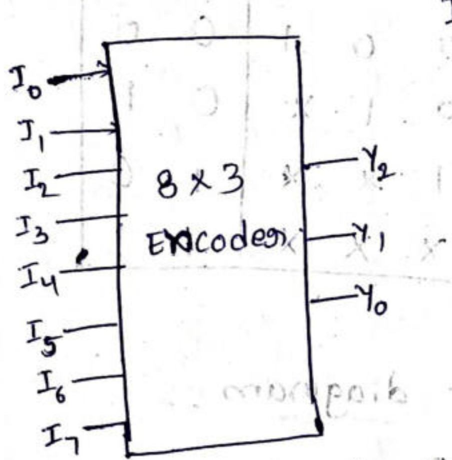
$Y_0 = E_I A + E_I B$

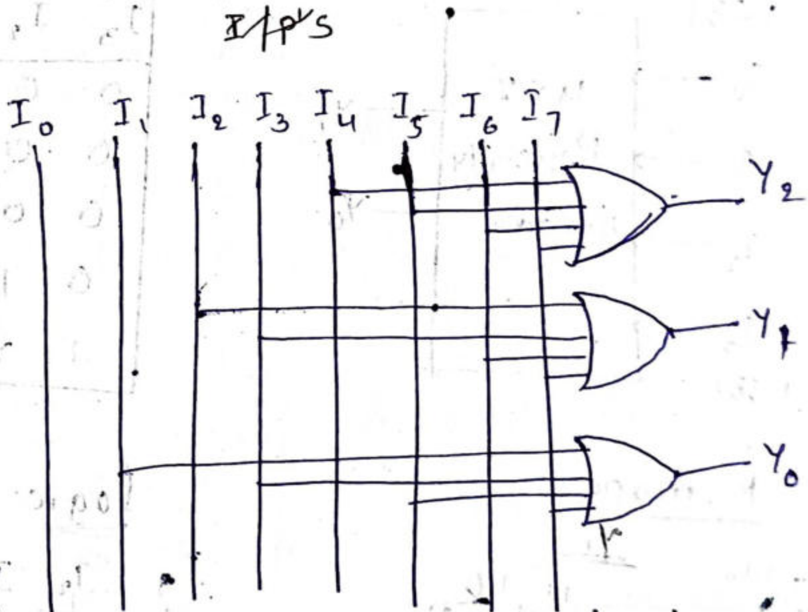
logic diagram :-



19/10

2) 8 × 3 Encoder :-





Truth table :-

I/p's								O/p's		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Y_2	Y_1	Y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$Y_2 = I_4 + I_5 + I_6 + I_7$

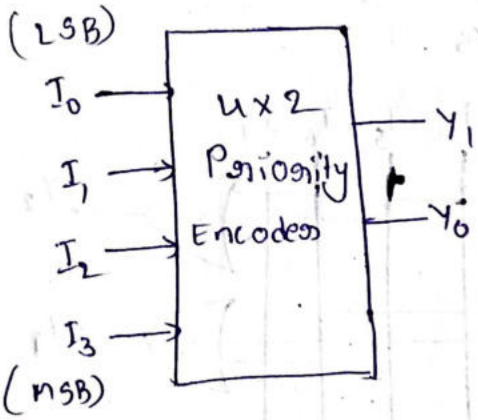
$Y_1 = I_2 + I_3 + I_6 + I_7$

$Y_0 = I_1 + I_3 + I_5 + I_7$

Priority encoder :-

A priority encoder is a combinational logic circuit, which includes a priority function. In priority encoder if two or more inputs are equal to '1' at the same time, the input having highest priority will be considered precedence.

In 4x2 priority encoder, the outputs are Y_1, Y_0 and the inputs are I_0, I_1, I_2, I_3.



Truth table

I_3	I_2	I_1	I_0	Y_1	Y_0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

K-maps :-

Y_1

$I_3I_2 \backslash I_1I_0$	00	01	11	10
00	X			
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

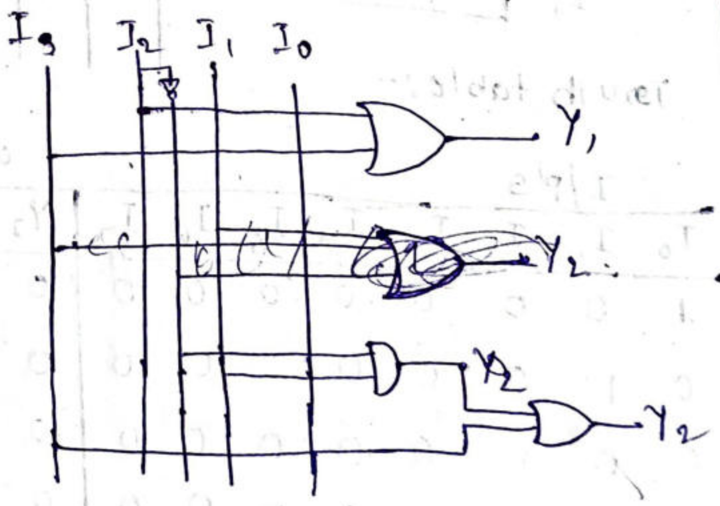
$Y_1 = I_2 + I_3$

Y_2

$I_3I_2 \backslash I_1I_0$	00	01	11	10
00	X		1	1
01				
11	1	1	1	1
10	1	1	1	1

$Y_2 = I_3 + I_2' I_1$

logic diagram :-



Magnitude comparator :-

1) 2-bit magnitude comparator :-

In 2-bit magnitude comparator we have two inputs A & B. where each and every input size is 2 bits and A contains a_1, a_0 & B contains b_1, b_0. Therefore, possible input combinations are 16 (0-15). For this, we have three outputs. A=B, A>B, A<B

Truth table

a_1	a_0	b_1	b_0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

K-maps

A>B

a_1a_0 \ b_1b_0	00	01	11	10
00	0	1	3	2
01	1 (4)	5	7	6
11	1 (12)	1 (13)	15	1 (14)
10	1 (8)	1 (9)	11	10

$$A>B = a_1\bar{b}_1 + a_0\bar{b}_1\bar{b}_0 + \bar{b}_0 a_1 a_0$$

$$= a_0\bar{b}_0(\bar{b}_1 + a_1) + a_1\bar{b}_1$$

A<B

a_1a_0 \ b_1b_0	00	01	11	10
00		1	1	1
01			1	1
11				
10			1	

$$A<B = b_1\bar{a}_1 + \bar{a}_0 b_1 b_0 + b_0\bar{a}_1\bar{a}_0$$

$$A<B = \bar{a}_0 b_0(\bar{a}_1 + b_1) + b_1\bar{a}_1$$

A=B

a_0a_1 \ b_0b_1	00	01	11	10
00	1			
01		1		
11			1	
10				1

$$A=B = \bar{a}_0\bar{a}_1\bar{b}_0\bar{b}_1 + \bar{a}_0 a_1\bar{b}_0 b_1 + a_0 a_1 b_0 b_1 + a_0\bar{a}_1 b_0\bar{b}_1$$

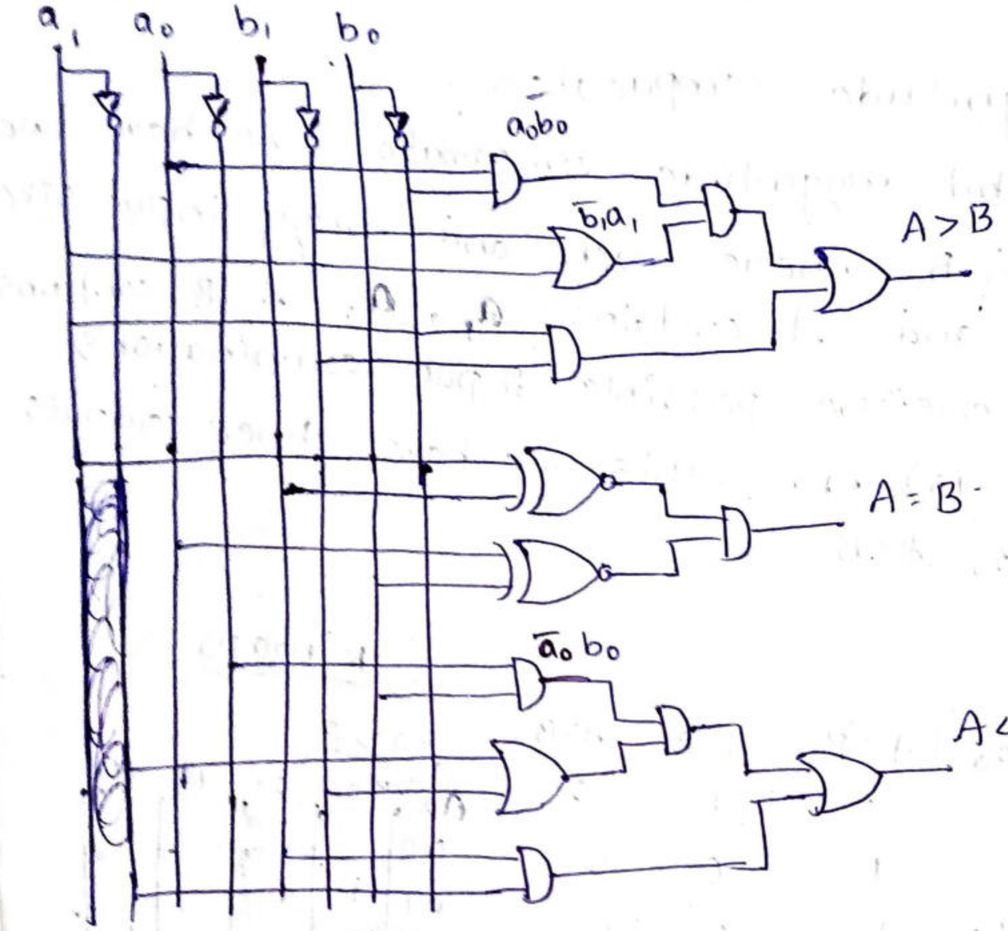
$$= \bar{a}_0\bar{b}_0(\bar{a}_1\bar{b}_1 + a_1 b_1) + a_0 b_0(a_1 b_1 + \bar{a}_1\bar{b}_1)$$

$$= \bar{a}_0\bar{b}_0(\overline{a_1 \oplus b_1}) + a_0 b_0(\overline{a_1 \oplus b_1})$$

$$= (\overline{a_1 \oplus b_1})(\bar{a}_0\bar{b}_0 + a_0 b_0)$$

$$\therefore A=B = (\overline{a_1 \oplus b_1}) \cdot (\overline{a_0 \oplus b_0})$$

Logic diagram :-



CODE CONVERTERS :-

1) Design a combination circuit with 4 i/P'S and 1 o/p. The output goes high if and only if A & C inputs are high. Draw the truth table and also draw the logic diagram.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1

K-map :-

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15: 1	14: 1
10	8	9	11: 1	10: 1

$Y = AC$

1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

logic diagram :-

2) Design a combinational circuit BCD to excess 3 code.

A	B	C	D	X_0	X_1	X_2	X_3
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

K-maps :-

X_0 :- BD

AB \ CD	00	01	11	10
00				
01		1	1	1
11	X	X	X	X
10	1	1	X	X

BC

A

$X_0 = A + BC + BD$

X_1 :- $B'D$

AB \ CD	00	01	11	10
00		1	1	1
01	1			
11	X	X	X	X
10		1	X	X

$B'C$

$BC'D'$

$X_1 = BC'D' + B'D + B'C$

X_2 :-

AB \ CD	00	01	11	10
00	1		1	
01	1		1	ϕ
11	X	X	X	X
10	1		X	X

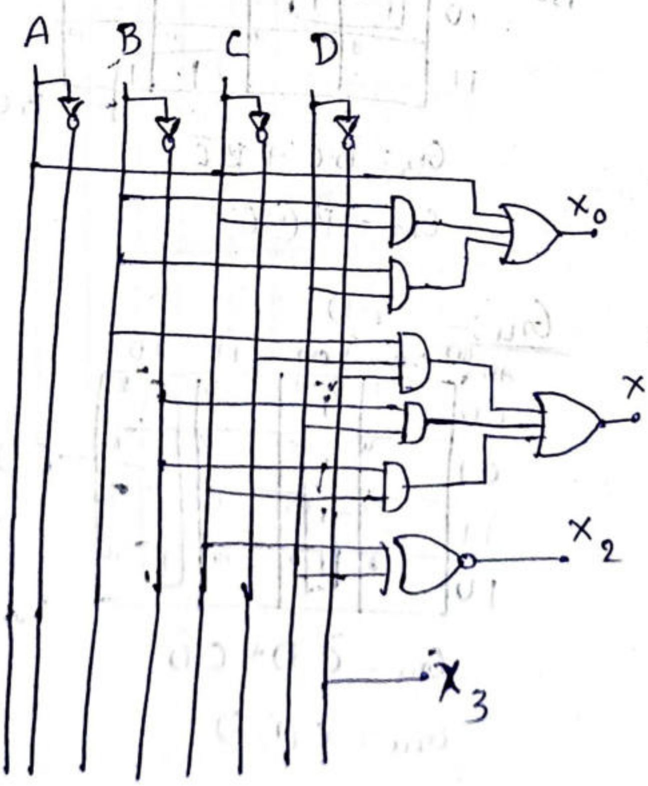
$X_2 = C'D' + CD = \overline{C \oplus D}$

X_3 :-

AB \ CD	00	01	11	10
00	1			1
01	1			1
11	X	X	X	X
10	1		X	X

$X_3 = D'$

Logic diagram :-

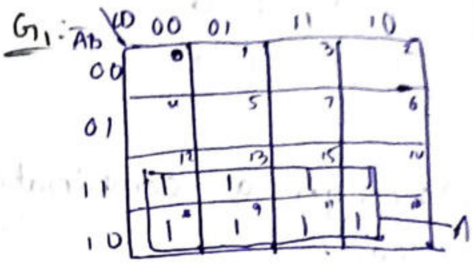


4-bit binary to Grey conversions:-

Binary i/p				Grey outputs			
A	B	C	D	G1	G2	G3	G4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

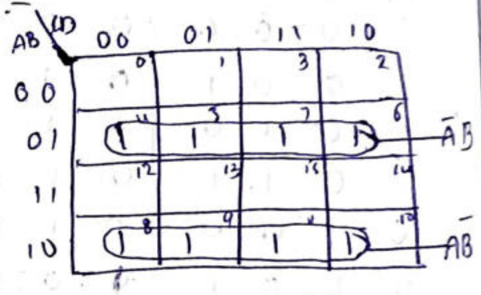
K-maps:-

G_1:-



$G_1 = A$

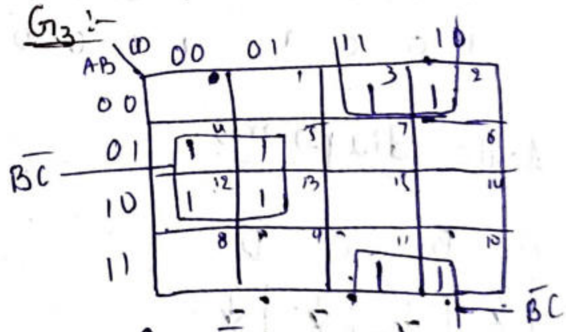
G_2:-



$G_2 = \bar{A}B + A\bar{B}$

$G_2 = A \oplus B$

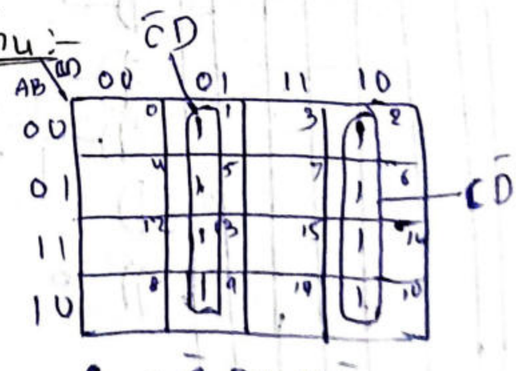
G_3:-



$G_3 = \bar{B}C + B\bar{C}$

$G_3 = B \oplus C$

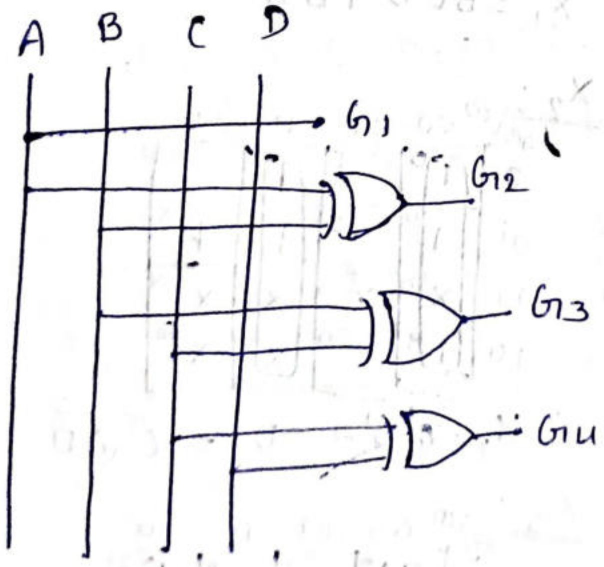
G_4:-



$G_4 = \bar{C}D + C\bar{D}$

$G_4 = C \oplus D$

logic diagram:-



20/10

1) Design a combinational logic circuit whose input is 3-bit binary and whose output is 2's complement of the input

A	B	C	Y_2	Y_1	Y_0 (2's complement)
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	0	1	1
1	1	0	0	1	0
1	1	1	0	0	1

K-maps

Y_2:

A \ BC	00	01	11	10
0	–	1	1	1
1	1			

$\bar{A}C$, $\bar{A}B$, $A\bar{B}\bar{C}$

$Y_2 = \bar{A}C + \bar{A}B + A\bar{B}\bar{C}$

$Y_2 = A'C + A'B + AB'C'$

Y_1:

A \ BC	00	01	11	10
0		1		1
1		1		1

$Y_1 = B'C + BC'$

$Y_1 = B \oplus C$

Y_0:

A \ BC	00	01	11	10
0		1	1	
1		1	1	

$B'C$, BC

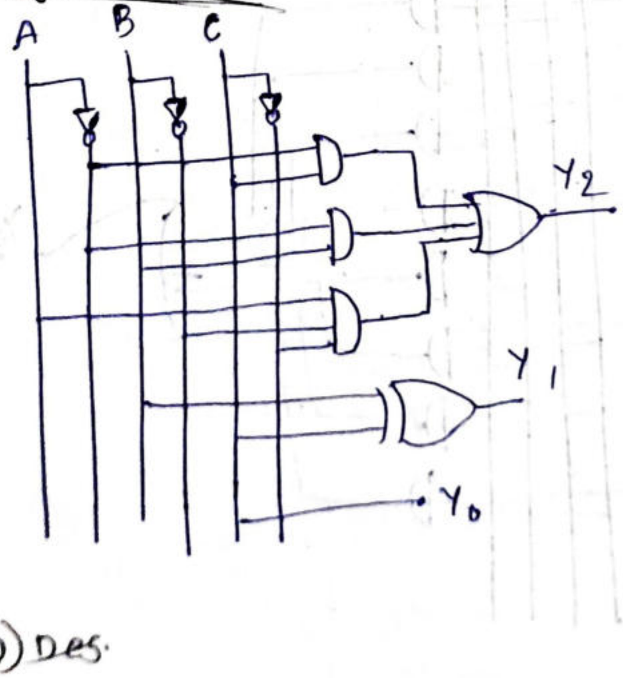
~~$Y_0 = B'C + BC$~~ $Y_0 = C$

Y_0:

A \ BC	00	01	11	10
0		1	1	
1		1	1	

$Y_0 = C$

logic circuit:



2) Des.

1) Design a logic circuit which has 4 input's and output is 'one'.

i) when all inputs are 1

ii) None of the inputs equal to 1

iii) An odd number of inputs are equal to 1

An) Truth table.

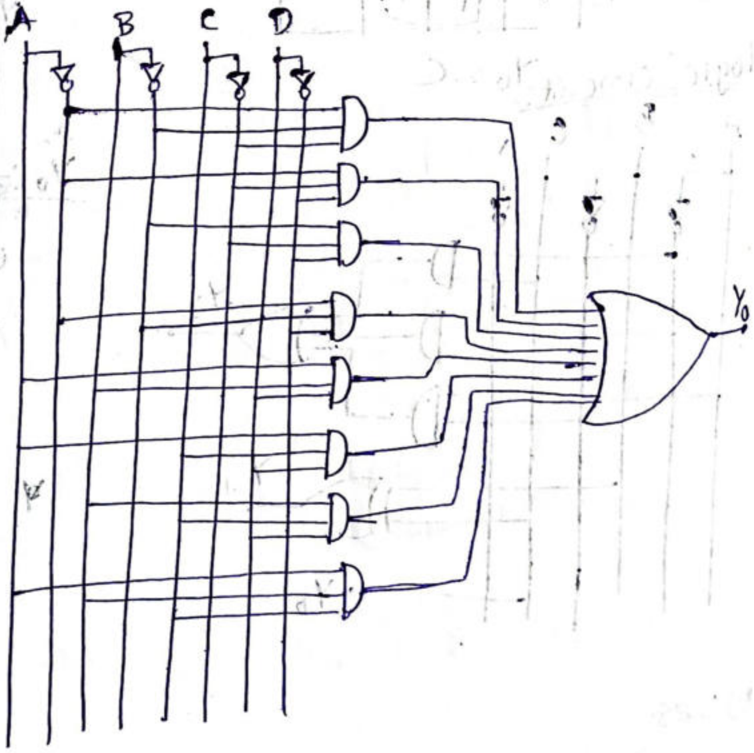
A	B	C	D	Y_0
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

K-maps :-

AB \ CD	00	01	11	10
00	1	1		1
01	1		1	
11		1	1	1
10	1		1	

$A'B'C'$, $A'B'D'$, BCD, ABC, ACD, $A'C'D'$, $C'D'B'$, ABD

$$Y_0 = A'B'C' + A'C'D' + B'C'D' + A'B'D' + ABD + ACD + BCD + ABC$$



NOTE:-

*) During the conversion of codes from one form to other form, is there any insufficient data (not full data (or) not total combination) is available then we have to consider 'don't care' combinations

Ex:- BCD to Grey, BCD to excess-3

*) For full data on total combinations examples are binary to Grey, Grey to excess 3 code.

1) Design a combinational circuit which can convert 8, 4, 2, 1 code to Grey code

2) Design a combinational circuit which can convert 3-bit binary to excess-3 code.

3) Design a combinational circuit which can convert 4-bit binary to Grey code

1. An) Truth table :-

A	B	C	D	G_1	G_2	G_3	G_4
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1

K-maps :-

G_1 :-

AB \ CD	00	01	11	10
00				
01				
11	X	X	X	X
10	1	1	X	X

$G_1 = A$

G_2 :-

AB \ CD	00	01	11	10
00				
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

$G_2 = B + A$

G_4 :-

AB \ CD	00	01	11	10
00		1		1
01		1		1
11	X	X	X	X
10		1	X	X

CD'

$C'D$

$G_4 = C'D + CD'$

$G_4 = C \oplus D$

G_3 :-

AB \ CD	00	01	11	10
00			1	1
01	1	1		
11	X	X	X	X
10			X	X

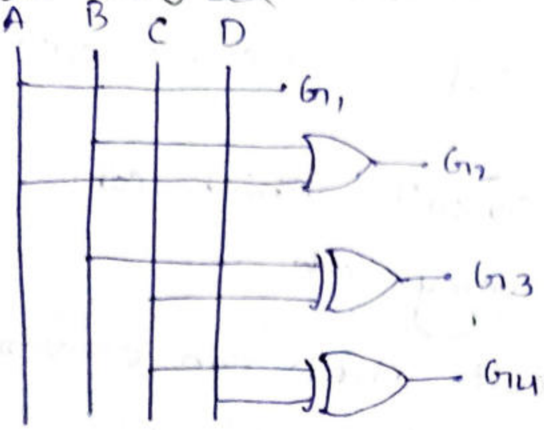
BC'

$B'C$

$G_3 = BC' + B'C$

$G_3 = B \oplus C$

logic diagram :-

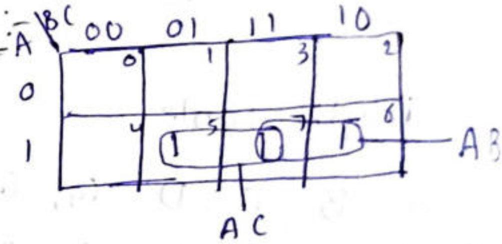


2. Ans) 3-bit binary to excess-3 code :-

A	B	C	Y_0	Y_1	Y_2	Y_3
0	0	0	0	0	1	1
0	0	1	0	1	0	0
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	0	0
1	1	0	1	0	0	1
1	1	1	1	0	1	0

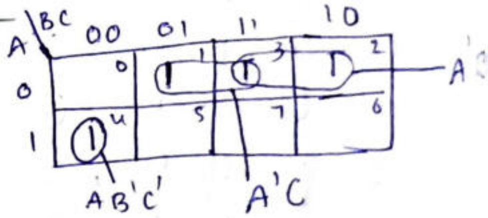
K-maps :-

Y_0 :-



$Y_0 = AB + AC = A(B+C)$

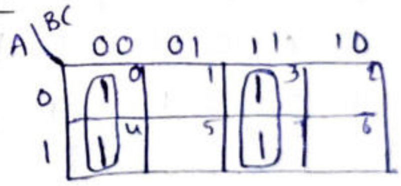
Y_1 :-



$Y_1 = AB'C' + A'C + A'B$

$Y_1 = AB'C' + A'(B+C)$

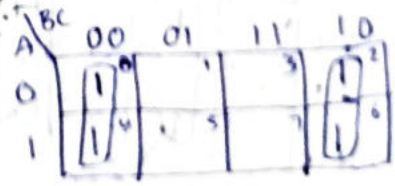
Y_2 :-



$Y_2 = B'C' + BC$

$Y_2 = \overline{B \oplus C}$

Y_3 :-

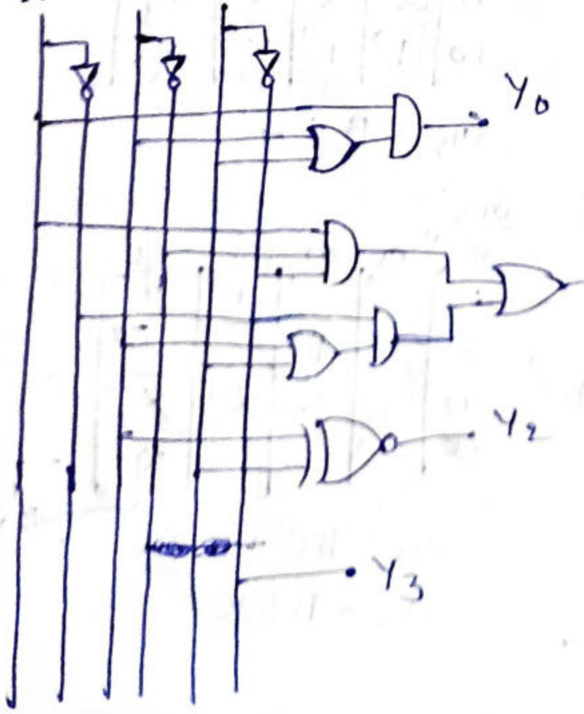


$Y_3 = B'C' + BC'$

$Y_3 = C'(B'+B)$

$Y_3 = C'$

logic diagram :-



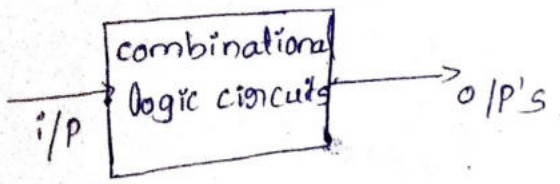
26/10

UNIT - 4

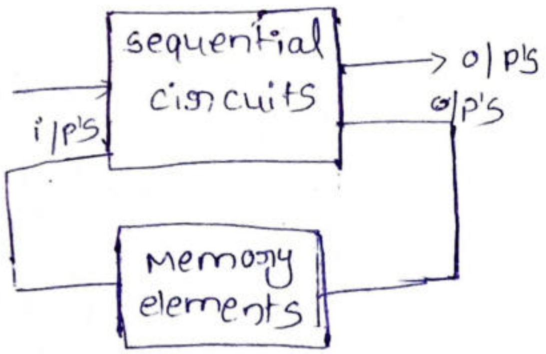
SEQUENTIAL CIRCUITS

combinational circuits	sequential circuits
1) Combinational logic circuit contains, combination of logic gates.	1) sequential circuits contains, combination of logic gates and memory elements.
2) In Combinational circuit, output depends on present i/p's only.	2) In sequential circuits o/p depends on present i/p's and past o/p's.
3)	3)
4) we can not store the information.	4) we can store the information by using memory elements.
5) The operation is speed	5) The operation is slow.
6) In combinational, the hardware complexity is more.	6) less hard ware complexity.

3) Combinational circuits:



3) Sequential circuits:



classification of sequential circuit :-

sequential circuits are classified into two types :-

1) synchronous sequential circuit

2) Asynchronous sequential circuit

⇒ In synchronous sequential circuit, signals can effect the memory elements only at discrete time intervals.

⇒ In Asynchronous sequential circuits, signals can effect the memory elements at any instant of time intervals

⇒ The memory elements used in both circuits are flip-flops. Flip-flops are capable of storing 1-bit of information.

Synchronous	Asynchronous
1) Memory elements are clocked flip-flops.	1) Memory elements are unclocked flip-flops.
2) The change in i/p signal can effect the memory element upon the activation of clock signal.	2) change in i/p signal can effect the memory element at any instant of time.
3) Operating speed of clock depends on time delay	3) Absence of clock, these are faster than synchronous circuit.
4) easy to design	4) Difficult to design.

27/10

Latches and flipflops :-

Latches and flipflops are the basic building blocks of sequential circuits. The difference b/w latches and flipflops are

Latches	flip-flops
1) Latches are transparent	1) flip-flops are not transparent
2) Latches are level triggered	2) flip-flops are edge triggered

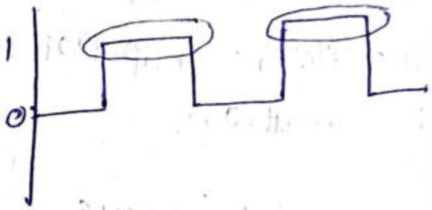
3) Latch has no memory element.	3) Flip flops are memory elements
4) The o/p of latches changed when the i/p's are level triggered	4) The o/p of flip-flops changes if the i/p's are edge triggered

level triggering :-

The o/p state is allowed to changed according to the i/p's when the active level is maintained at the clock input

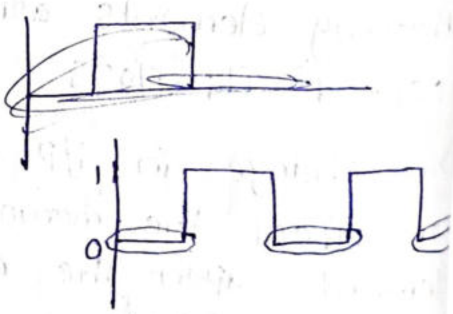
a) Positive level triggered :-

The o/p of latch response to the input changes only when i/p is high.



b) Negative level triggered :-

The o/p of latch response to the input changes only when the input is low.

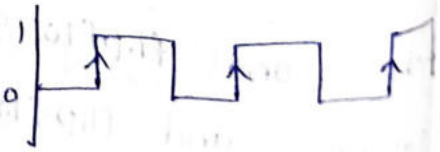


Edge triggering :-

The o/p response to the changes in i/p at the positive and negative edges of the clock pulse.

a) Positive edge triggering :-

The o/p response to the changes in the i/p only at the positive edge of the clock pulse.



b) Negative edge triggering :-

The o/p response to the changes in the i/p only at the negative edge of the clock pulse

Synchronous circuits:-

synchronous or clocked sequential circuits are represented by using two models

a) Mealy machine

b) Moore machine.

a) Mealy machine :-

In mealy machine the o/p depends on present stage and i/p's

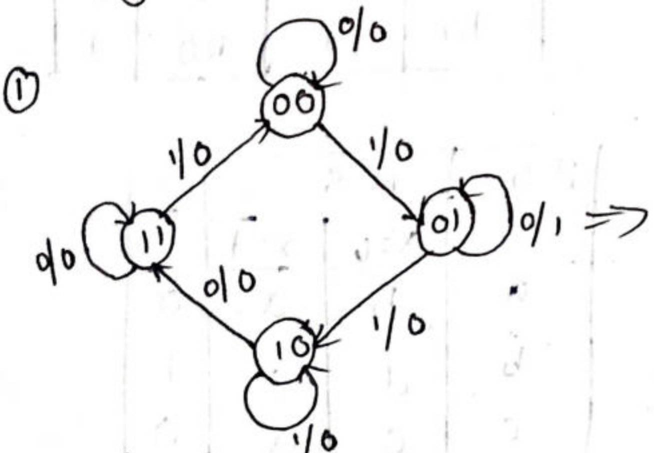
b) Moore machine :-

In moore machine the o/p depends only on the present i/p's.

These sequential circuits are also known as finite state machine (FSM). The name derives, the fact that functional behaviour of these circuits can be represented using finite number of states.

Mealy machine problems

①

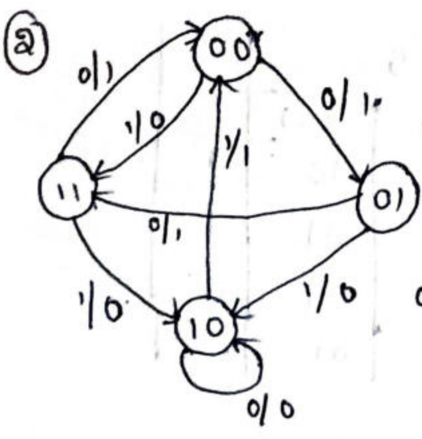


state diagram.

P.S	N.S, Z (o/p)	
	X=0	X=1
00	00, 0	01, 0
01	01, 1	10, 0
10	11, 0	10, 0
11	11, 0	00, 0

State table

②



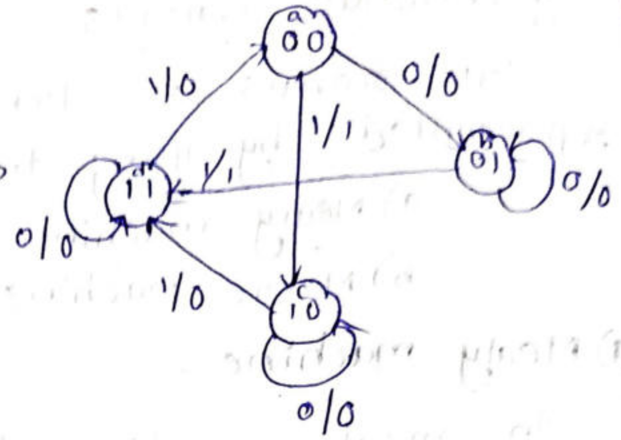
assign State 00 = a
01 = b
10 = c

P.S	N.S, Z(o/p)	
	X=0	X=1
00 - a	b, 1	d, 0
01 - b	d, 1	c, 0
10 - c	c, 0	a, 1
11 - d	a, 1	c, 0

3)

P.S	N.S, Z (O/P)	
	X=0	X=1
00 - a	b, 0	c, 1
01 - b	b, 0	d, 1
10 - c	c, 0	d, 0
11 - d	d, 0	a, 0

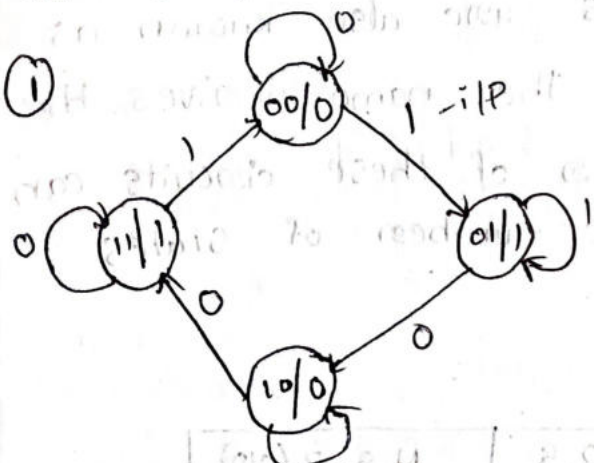
⟹



Moore state machine :-

In moore state machine, we have both present and next states, when control i/p either '0' or '1'. where each and every state having only 1 output but not as compare to mealy machine.

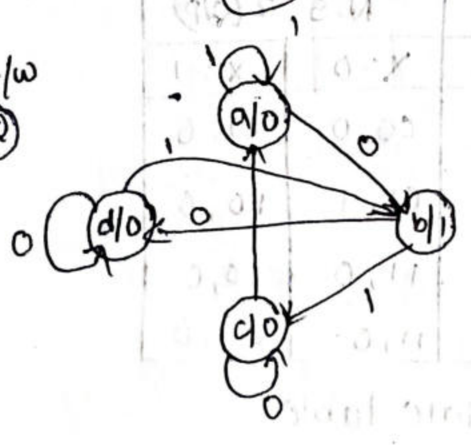
1



⟹

P.S	N.S		Z
	X=0	X=1	
00	00	01	0
01	10	01	1
10	11	10	0
11	11	00	1

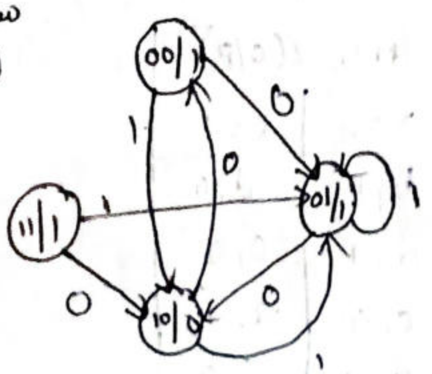
H/W
2



⟹

P.S	N.S		Z
	X=0	X=1	
a	b	a	0
b	d	c	1
c	c	a	0
d	d	b	0

H/W
3



⟹

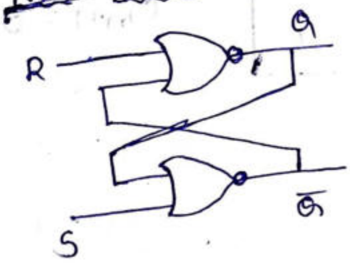
P.S	N.S		Z
	X=0	X=1	
00	01	10	1
01	10	01	1
10	00	01	0
11	10	01	

30/10

Latch :-

RS latch :-

NOR Latch :-



NOR Latch truth table

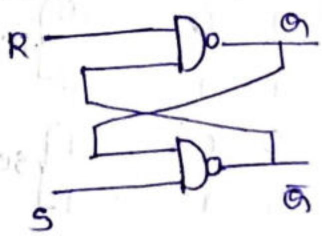
R	S	Q	$\bar{Q}$
0	0	No change	
0	1	1	0 (set state)
1	0	0	1 (Reset state)
1	1	Invalid state	

Nor gate truth table

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

if any one of the i/p is '0' then o/p is '1'.

NAND Latch :-



NOR Latch truth table

R	S	Q	$\bar{Q}$
0	0	Invalid state	
0	1	1	0 (set)
1	0	0	1 (Reset)
1	1	No change	

NAND gate truth table

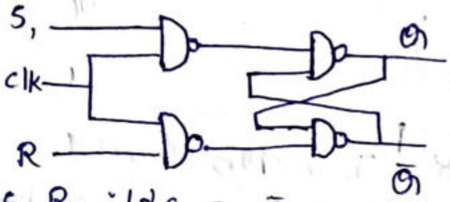
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

if any one of i/p is '0' then o/p is '1'.

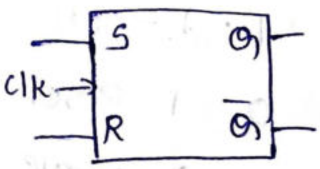
31/10

SR flipflop :-

SR NAND flipflop :-



S, R i/p's Q, $\bar{Q}$ are o/p's



SRFF symbol

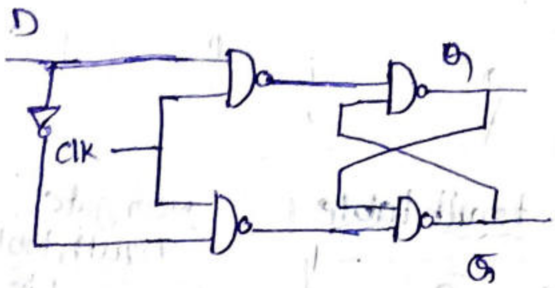
Characteristic table.

Clk	S	R	Q	Q_{n+1}
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	X
1	1	1	1	X
0	X	X	0	
	X	X	1	

S \ RQ_n	00	01	11	10
0		1		
1	1	1	X	X

$Q_{n+1} = S + \bar{R} Q_n$

D-flipflop :-

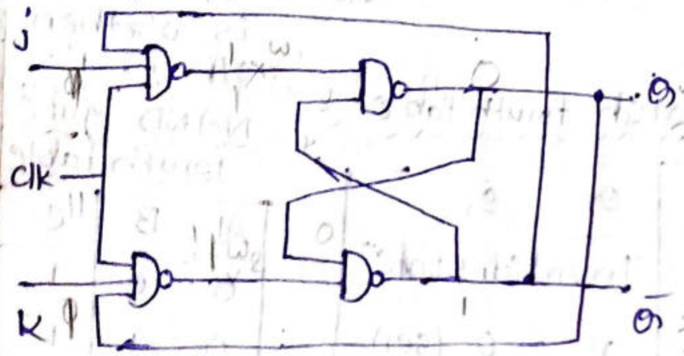


Clk	D	Q_n	Q_{n+1}
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

D \ Q_n	0	1
0	0	0
1	1	1

$Q_{n+1} = D$

JK-flipflop :-



Clk	J	K	Q_n	Q_{n+1}	
1	0	0	0	0	No change
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	set
1	1	0	1	1	
1	1	1	0	1	Toggle state
1	1	1	1	0	

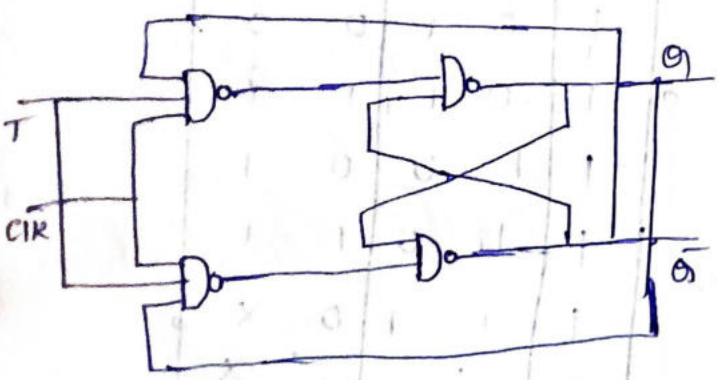
K-map :-

J \ KQ_n	00	01	11	10
0	0	1	0	0
1	1	1	0	1

$Q_{n+1} = \bar{K} Q_n + J \bar{Q}_n$

2/11

T(Toggle)-flip flop :-



Clk	T	Q_n	Q_{n+1}	
1	0	0	0	No change
1	0	1	1	set
1	1	0	1	Reset
1	1	1	0	Toggle

k-map :-

T \ Q_n	0	1
0		1
1	1	

$Q_{n+1} = T' Q_n + T Q_n'$

$= T \oplus Q_n$

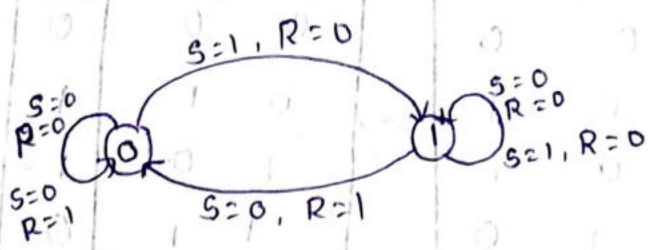
Excitation tables :-

SR flipflop :-

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

state diagram :-

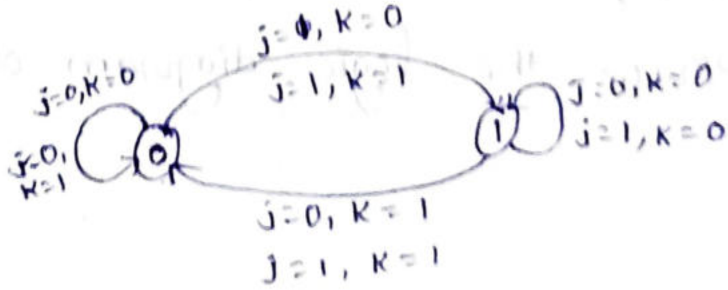


JK-Flipflop :-

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

state diagram :-



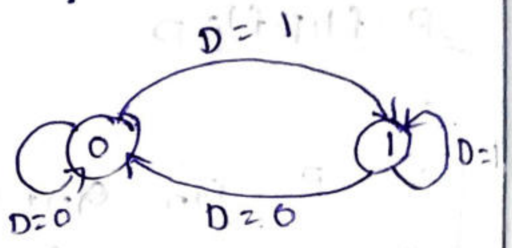
D-flipflop :-

D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

→

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

State diagram:-



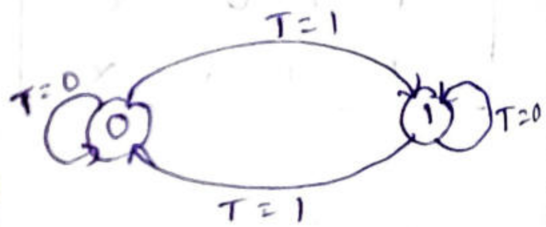
T- flip flop :-

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

→

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

State diagram :-



Flip - Flop conversions :-

1) First write required flip flop characteristic table.
2) Fill this table by considering present and next state with the available flipflop values (excitation table of available flip flop).
3) Draw the K-maps for the available flip-flop values
4) Draw the logic diagram of required flip-flop.

① SR to JK flipflop conversion :-

J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

K-maps :-

S :-

J \ KQ_n	00	01	11	10
0	0	X	0	0
1	1	X	0	1

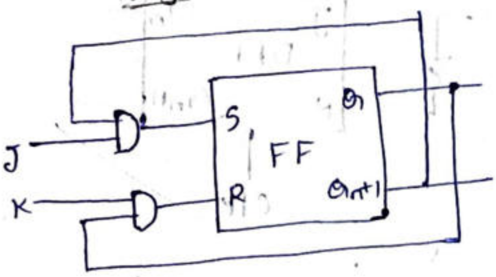
$Q = J Q_n'$

R :-

J \ KQ_n	00	01	11	10
0	⊗	⊗	1	X
1	⊗	⊗	1	⊗

$R = K Q_n$

logic diagram :-



② JK to T flip flop :-

T	Q_n	Q_{n+1}	J	K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

K-maps :-

J :-

T \ Q_n	0	1
0		X
1	1	X

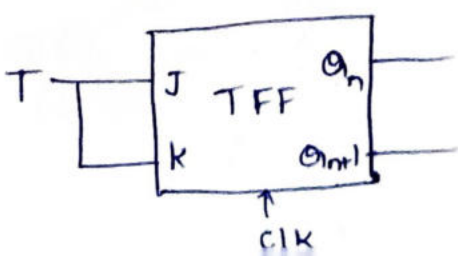
$J = T$

K :-

T \ Q_n	0	1
0	X	
1	X	1

$K = T$

logic diagram :-



③ JK to D flip-flop :-

D	Q_n	Q_{n+1}	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0

K-maps :-

J :-

D \ Q_n	0	1
0		X
1	1	X

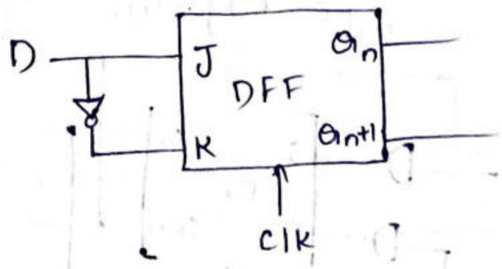
$J = D$

K :-

D \ Q_n	0	1
0	X	1
1	X	

$K = D'$

logic diagram :-



④ JK to SR flip-flop :-

S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	0	X	X	X
1	1	1	X	X	X

K-maps :-

J :-

S \ RQ_n	00	01	11	10
0		X	X	
1	1	X	X	X

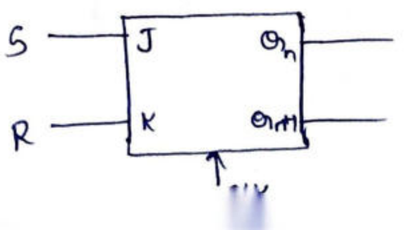
$J = S$

K :-

S \ RQ_n	00	01	11	10
0	X		1	X
1	X		X	X

$K = R$

logic diagram :-

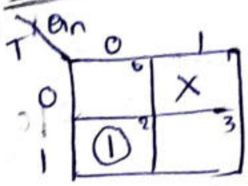


(5) SR to T flip flop :-

T	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	X	0
1	0	1	1	0
1	1	0	0	1

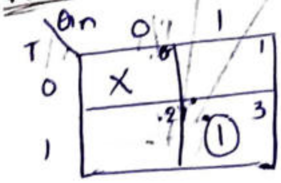
K-maps:-

S :-



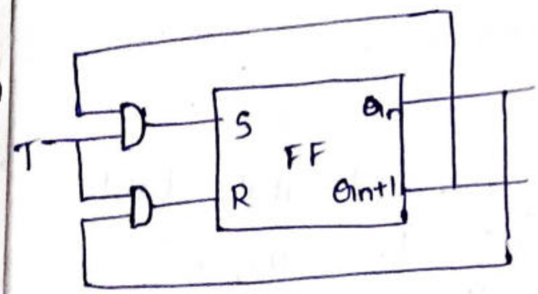
$S = T.Q_n'$

R :-



$R = T.Q_n$

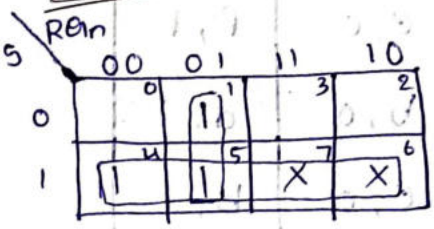
logic diagram :-



(6) D to SR flipflop :-

S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	X	X
1	1	1	X	X

K-maps:-

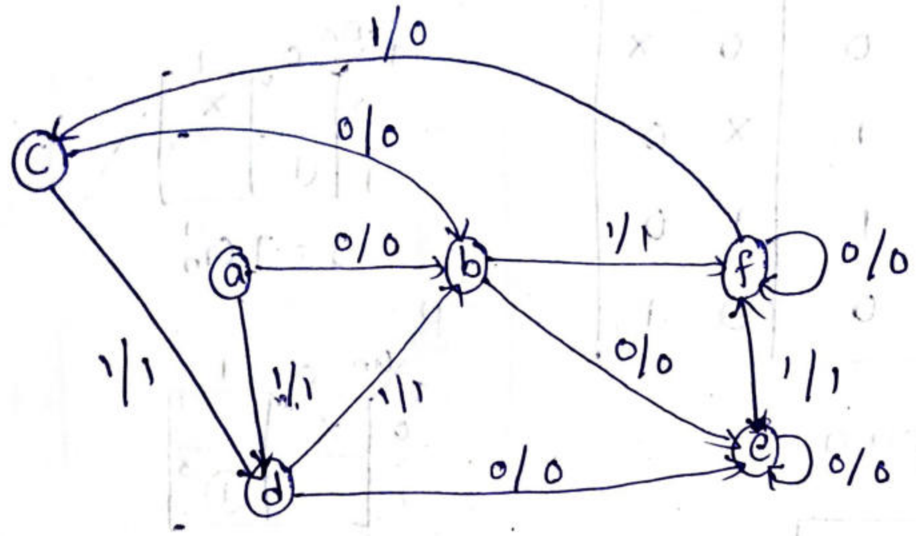


$D = S + R'Q_n$

logic diagram :-

16/11

State reduction Technique :-



a) State diagram

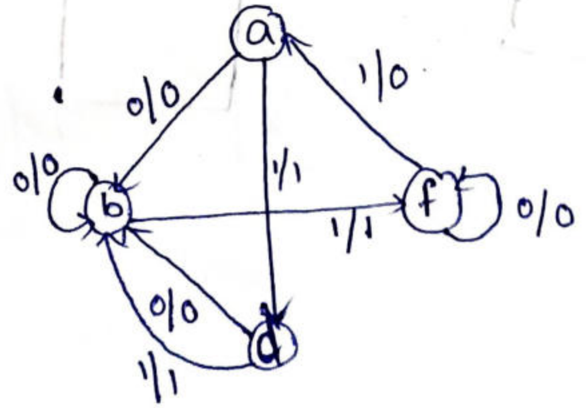
State table

P.S	NS, Z X = 0	NS, Z X = 1
a	b, 0	d, 1
b	e, 0	f, 1
c	b, 0	d, 1
d	e, 0	b, 1
e	e, 0	f, 1
f	f, 0	c, 0

Reduced State table

P.S	NS, Z X = 0	NS, Z X = 1
a	b, 0	d, 1
b	b, 0	f, 1
d	b, 0	b, 1
f	f, 0	a, 0

Reduced state diagram.



steps in synchronus sequencial circuit design:-

step 1 :- obtain state table from the given circuit which is represented as state diagram.

step 2 :- The number of states may be reduced by state reduction technique, of the sequencial circuit can be characterised by input & o/p relationships independent of number of steps.

step 3 :- Assign binary values to each state in the table

step 4 :- Determine the number of flipflops (FF's) needed a letter symbol) for each flipflop

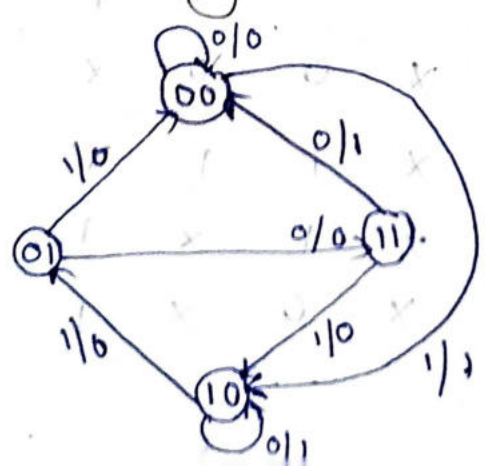
step 5 :- choose the type of flipflop to be used

step 6 :- From the state table derive the circuit excitation and o/p tables. (flow table).

step 7 :- Using the K-map method derive the circuit o/p equations and flipflop input functions.

step 8 :- Derive the sequencial circuit logic diagram

① For the given state diagram, design a sequential circuit using JK flipflops



① State table

PS	NS, Z	
	x = 0	x = 1
0 0	00, 0	10, 1
0 1	11, 0	00, 0
1 0	10, 1	01, 0
1 1	00, 1	10, 0

④ Here, we are using two bit's to represents four states that is why we are using two flip flops.

⑤ JK - flip-flop

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

⑥ Flow table :-

P.S		input	N.S		flipflop i/p's				output
Q_A	Q_B	X	Q_A^+	Q_B^+	J_A	K_A	J_B	K_B	Z
0	0	0	0	0	0	X	0	X	0
0	0	1	1	0	1	X	0	X	1
0	1	0	1	1	1	X	X	0	0
0	1	1	0	0	0	X	X	1	0
1	0	0	1	0	X	0	0	X	1
1	0	1	0	1	X	1	1	X	0
1	1	0	0	0	X	1	X	1	1
1	1	1	1	0	X	0	X	1	0

K-m

J_A

J_B

J_B

Z :-

$Z = Q_A'$

K-maps :-

J_A :-

$Q_A \backslash Q_B X$	00	01	11	10
0	0	1	0	1
1	X	X	X	X

$$J_A = Q_B' X + Q_B X'$$

$$J_A = Q_B \oplus X$$

K_A :-

$Q_A \backslash Q_B X$	00	01	11	10
0	X	X	X	X
1	0	1	0	1

$$K_A = Q_B' X + Q_B X'$$

$$K_A = Q_B \oplus X$$

J_B :-

$Q_A \backslash Q_B X$	00	01	11	10
0	0	0	X	X
1	0	1	X	X

$Q_A X$

$$J_B = Q_A X$$

K_B :-

$Q_A \backslash Q_B X$	00	01	11	10
0	X	X	1	0
1	X	X	1	1

X

Q_A

$$K_B = \cancel{Q_B} X + Q_A$$

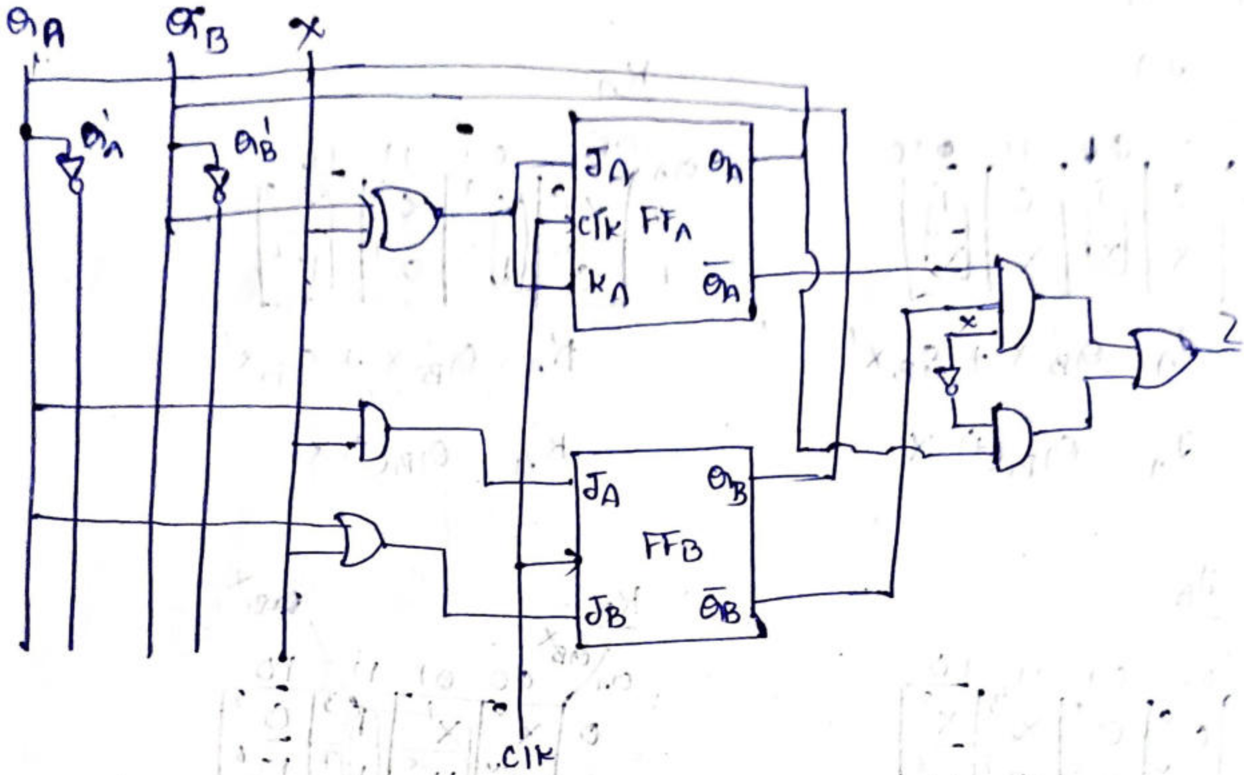
Z :-

$Q_A' Q_B' X$

$Q_A \backslash Q_B X$	00	01	11	10
0	0	1	0	0
1	1	0	0	1

$Q_A X'$

$$Z = Q_A' Q_B' X + Q_A X'$$



Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

⑤ D - flip flop :-

⑥ Flow table :-

P.S		i/p	N.S		flipflop - i/p's		output
Q_A	Q_B	X	Q_A^+	Q_B^+	D_A	D_B	Z
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1
0	1	0	1	1	1	1	0
0	1	1	0	0	0	0	0
1	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	1
1	1	1	1	0	1	0	0

K-maps:-

$Q_A \backslash Q_B x$	00	01	11	10
0	0^{0}	1^{1}	0^{3}	1^{2}
1	1^{4}	0^{5}	1^{7}	0^{6}

$$D_A = Q_A Q_B' x' + Q_A' Q_B x + Q_A Q_B x + Q_A Q_B x'$$

$$D_A = Q_A \oplus Q_B \oplus x$$

D_B:-

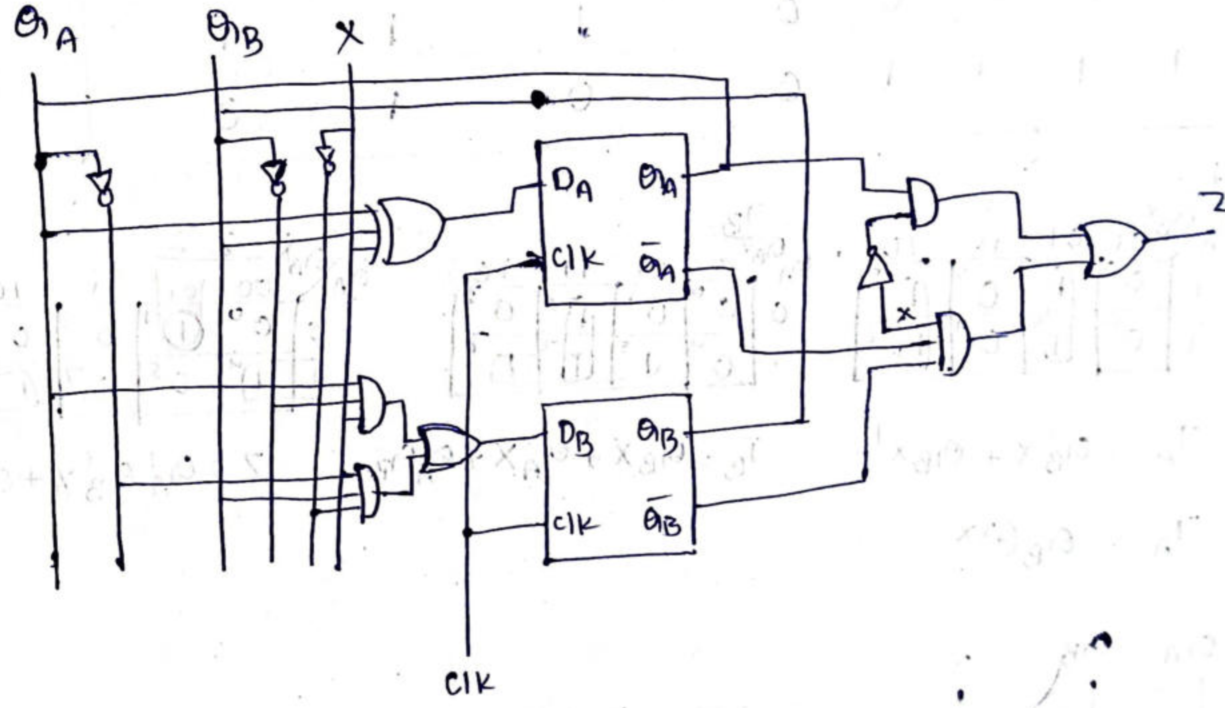
$Q_A \backslash Q_B x$	00	01	11	10
0	0^{0}	0^{1}	0^{3}	1^{2}
1	0^{4}	1^{5}	0^{7}	0^{6}

$$D_B = Q_A Q_B' x + Q_A' Q_B x'$$

Z :-

$Q_A \backslash Q_B x$	00	01	11	10
0	0^{0}	1^{1}	0^{3}	0^{2}
1	1^{4}	0^{5}	0^{7}	1^{6}

$$Z = Q_A x' + Q_A' Q_B' x$$



T- flip flop

0	0	0
0	1	1
1	0	1
1	1	0

flow table

P.S		i/p	N.S		flipflop i/p's		output
Q_A	Q_B	x	Q_A^+	Q_B^+	T_A	T_B	Z
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1
0	1	0	1	1	1	0	0
0	1	1	0	0	0	1	0
1	0	0	1	0	0	0	1
1	0	1	0	1	1	1	0
1	1	0	0	0	1	1	1
1	1	1	1	0	0	1	0

K-maps

$Q_A \backslash Q_B x$	00	01	11	10
0	0	1	0	1
1	0	1	0	1

T_B

$Q_A \backslash Q_B x$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Z

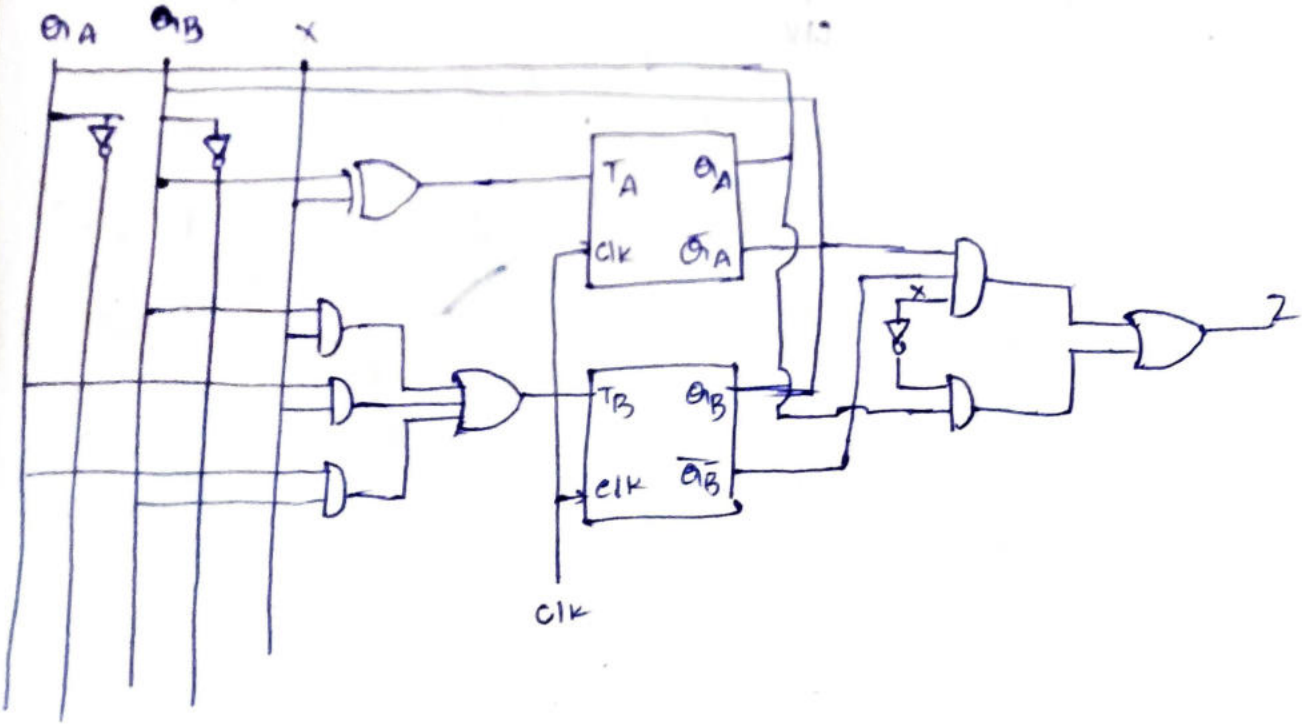
$Q_A \backslash Q_B x$	00	01	11	10
0	0	1	0	0
1	1	0	0	1

$T_A = Q_B' x + Q_B x'$.

$T_A = Q_B \odot x$

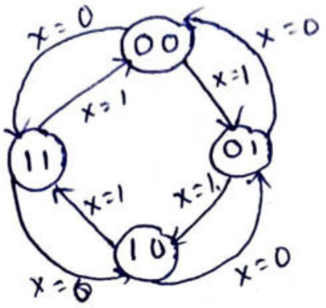
$T_B = Q_B x + Q_A x + Q_A Q_B$.

$Z = Q_A' Q_B' x + Q_A x'$



17/11

2-bit up/down counter



P.S	N.S X=0	N.S X=1
00	11	01
01	00	10
10	01	11
11	10	00

JK - flipflop :-

Q	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

flow table :-

P.S		i/p	N.S		flipflop i/p's			
Q_A	Q_B	X	Q_A^+	Q_B^+	J_A	K_A	J_B	K_B
0	0	0	1	1	1	X	1	X
0	0	1	0	1	0	X	1	X
0	1	0	0	0	0	X	X	1
0	1	1	1	0	1	X	X	1
1	0	0	0	1	X	1	1	X
1	0	1	1	1	X	0	1	X
1	1	0	1	0	X	0	X	1
1	1	1	0	0	X	1	X	1

K-maps :-

J_A :-

$Q_A \backslash Q_B X$	00	01	11	10
0	1		1	
1	X	X	X	X

$J_A = Q_B' X' + Q_B X$

$J_A = Q_B \odot X$

K_A :-

$Q_A \backslash Q_B X$	00	01	11	10
0	X	X	X	X
1	1		1	

$K_A = Q_B \odot X$

J_B :-

$Q_A \backslash Q_B X$	00	01	11	10
0	1	1	X	X
1	1	1	X	X

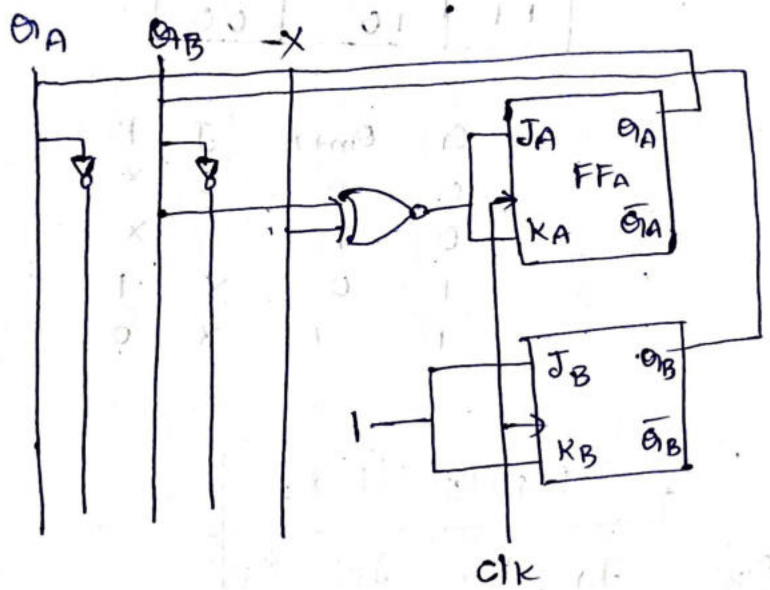
$J_B = \cancel{Q_B}\ 1$

K_B:-

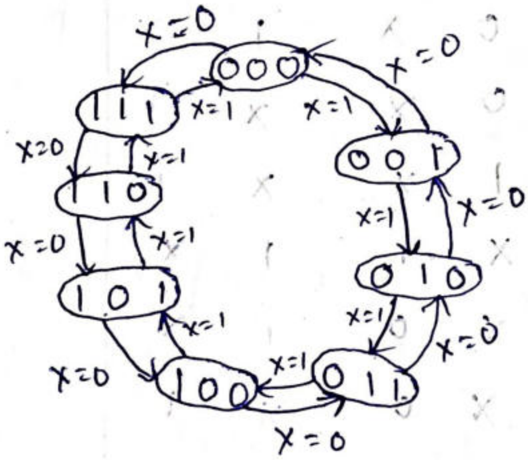
$Q_A \backslash Q_B X$	00	01	11	10
0	X	X	1	1
1	X	X	1	1

$K_B = 1$

logic diagram:-



3-bit up/down counter:-



P.S	N.S x=0	N.S x=1
000	111	001
001	000	010
010	001	011
011	010	100
100	011	101
101	100	110
110	101	111
111	110	000

T- flipflop:-

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Flow table :-

P.S			i/P	N.S			flipflop i/p's		
Q_A	Q_B	Q_C	X	Q_A^+	Q_B^+	Q_C^+	T_A	T_B	T_C
0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	0	0	0	0	1
0	0	1	1	0	1	0	0	1	1
0	1	0	0	0	0	1	0	1	1
0	1	0	1	0	1	1	0	0	1
0	1	1	0	0	1	0	0	0	1
0	1	1	1	1	0	0	1	1	1
1	0	0	0	0	1	1	1	1	1
1	0	0	1	1	0	1	0	0	1
1	0	1	0	1	0	0	0	0	1
1	0	1	1	1	1	0	0	1	1
1	1	0	0	1	0	1	0	1	1
1	1	0	1	1	1	1	0	0	1
1	1	1	0	1	1	0	0	0	1
1	1	1	1	0	0	0	1	1	1

K-maps :-

T_A :-

$Q_AQ_B \backslash Q_CX$	00	01	11	10
00	1			
01			1	
11			1	
10	1			

$T_A = Q_B' Q_C' X' + Q_B Q_C X$

$\cancel{T_A = Q_B \odot Q_C \odot X}$

T_B :-

$Q_AQ_B \backslash Q_CX$	00	01	11	10
00	1		1	
01	1		1	
11	1		1	
10	1		1	

$T_B = Q_C' X' + Q_C X$

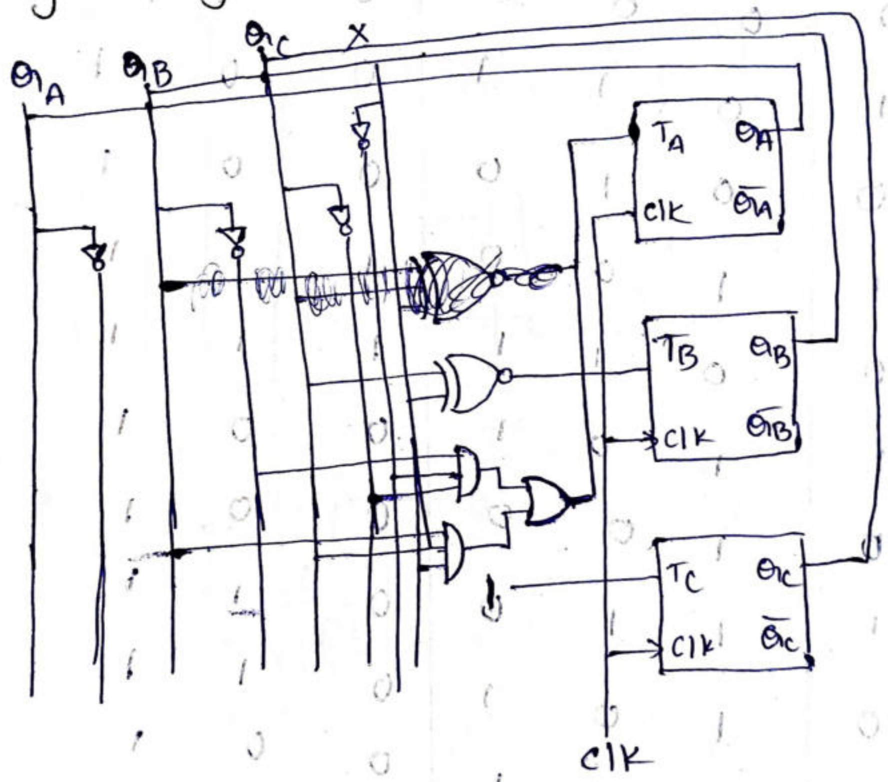
$T_B = Q_C \odot X$

T_C :-

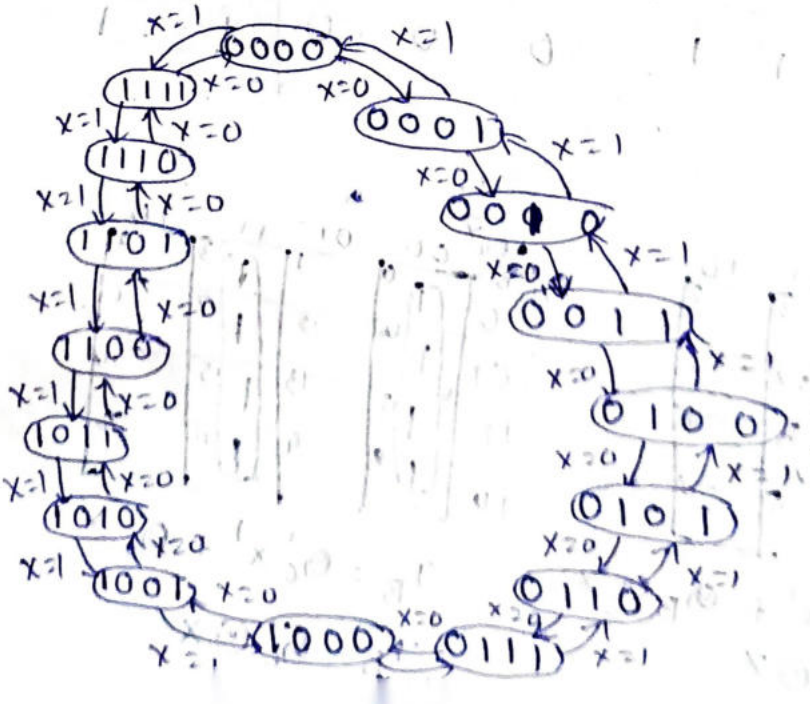
Q_AQ_B \ Q_CX	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$T_C = 1$

logic diagram :-



4-bit up/down counter :-



P.S	N.S X = 0	N.S X = 1
0000	0001	1111
0001	0010	0000
0010	0011	0001
0011	0100	0010
0100	0101	0011
0101	0110	0100
0110	0111	0101
0111	1000	0110
1000	1001	0111
1001	1010	1000
1010	1011	1001
1011	1100	1010
1100	1101	1011
1101	1110	1100
1110	1111	1101
1111	0000	1110

D - flip flop :-

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Flow table :-

P.S				i/p	N.S				flipflop i/p's			
Q_A	Q_B	Q_C	Q_D	X	Q_A^+	Q_B^+	Q_C^+	Q_D^+	D_A	D_B	D_C	D_D
0	0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	1	0	0	1	1
0	0	1	0	1	0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0	0	0	1	0	0

Q_A	Q_B	Q_C	Q_D	X	Q_A^+	Q_B^+	Q_C^+	Q_D^+	D_A	D_B	D_C	D_D	
0	0	1	1	1	0	0	1	0	0	0	1	0	
0	1	0	0	0	0	1	0	1	0	1	0	1	
0	1	0	0	1	0	0	1	1	0	0	1	1	
0	1	0	1	0	0	1	1	0	0	1	1	0	
0	1	0	1	1	0	1	0	0	0	1	0	0	
0	1	1	0	0	0	1	1	1	0	1	1	1	
0	1	1	0	1	0	1	0	1	0	1	0	1	13
0	1	1	1	0	1	0	0	0	1	0	0	0	14
0	1	1	1	1	0	1	1	0	0	1	1	0	15
1	0	0	0	0	1	0	0	1	1	0	0	1	16
1	0	0	0	1	0	1	1	1	0	1	1	1	17
1	0	0	1	0	1	0	1	0	1	0	1	0	18
1	0	0	1	1	1	0	0	0	1	0	0	0	19
1	0	1	0	0	1	0	1	1	1	0	1	1	20
1	0	1	0	1	1	0	0	1	1	0	0	1	21
1	0	1	1	0	1	1	0	0	1	1	0	0	22
1	0	1	1	1	1	0	1	0	1	0	1	0	23
1	1	0	0	0	1	1	0	1	1	1	0	1	24
1	1	0	0	1	1	0	1	1	1	0	1	1	25
1	1	0	1	0	1	1	1	0	1	1	1	0	26
1	1	0	1	1	1	1	0	0	1	1	0	0	27
1	1	1	0	0	1	1	1	1	1	1	1	1	
1	1	1	0	1	1	1	0	1	1	1	0	1	
1	1	1	1	0	0	0	0	0	0	0	0	0	

Q_A Q_B

1 1

K-maps

D_A :-

$Q_BQ_C \backslash Q_DX$	00
00	
01	
11	
10	

D_A

D_B :-

$Q_BQ_C \backslash Q_DX$	00
00	
01	
11	1
10	1

D_B =

D_C :-

$Q_BQ_C \backslash Q_DX$	00
00	
01	1
11	1
10	

D_C =

D_C

Q_A	Q_B	Q_C	Q_D	X	Q_A	Q_B	Q_C	Q_D	D_A	D_B	D_C	D_D
1	1	1	1	1	1	1	1	0	0	1	1	0

K-maps :-

D_A :-

$Q_A = 0$

Q_BQ_C \ Q_DX	00	01	11	10
00	(0)	(1) 1	(3)	(2)
01	(4)	(5)	(7)	(6)
11	(12)	(13)	(15)	(14) 1
10	(8)	(9)	(11)	(10)

$Q_A = 1$

Q_BQ_C \ Q_DX	00	01	11	10
00	(16) 1	(17)	(19) 1	(18) 1
01	(20) 1	(21) 1	(23) 1	(22) 1
11	(28) 1	(29) 1	(31) 1	(30)
10	(24) 1	(25) 1	(27) 1	(26) 1

$$D_A = Q_A' Q_B' Q_C' Q_D' X + Q_A' Q_B Q_C Q_D X' + Q_A Q_D' X' + Q_A Q_C Q_D +$$
$$Q_A Q_C X + Q_A Q_B' Q_D$$

D_B :-

Q_A

Q_BQ_C \ Q_DX	00	01	11	10
00	(0)	(1) 1	(3)	(2)
01	(4)	(5)	(7)	(6) 1
11	(12) 1	(13) 1	(15) 1	(14)
10	(8) 1	(9)	(11) 1	(10) 1

Q_A'

Q_BQ_C \ Q_DX	00	01	11	10
00	(16)	(17) 1	(19)	(18)
01	(20)	(21)	(23)	(22) 1
11	(28) 1	(29) 1	(31) 1	(30)
10	(24) 1	(25)	(27) 1	(26) 1

$Q_B' Q_C' Q_D' X$

$Q_B' Q_C Q_D X'$

$$D_B = Q_B Q_D' X' + Q_B Q_C' Q_D + Q_B Q_C X + Q_B' Q_C' Q_D' X + Q_B' Q_C Q_D X'$$

D_C :-

Q_A

Q_BQ_C \ Q_DX	00	01	11	10
00	(0)	(1) 1	(3)	(2) 1
01	(4) 1	(5)	(7) 1	(6)
11	(12) 1	(13)	(15) 1	(14)
10	(8)	(9) 1	(11)	(10) 1

Q_A'

Q_BQ_C \ Q_DX	00	01	11	10
00	(16)	(17) 1	(19)	(18) 1
01	(20) 1	(21)	(23) 1	(22)
11	(28) 1	(29)	(31) 1	(30)
10	(24)	(25) 1	(27)	(26) 1

$$D_C = Q_C Q_D' X' + Q_C Q_D X + Q_C' Q_D' X + Q_C' Q_D X'$$
$$D_C = Q_C (\overline{Q_D \oplus X}) + Q_C' (Q_D \oplus X) = Q_C \oplus Q_D \oplus X$$

D_D :-

Q_A

$Q_BQ_C \backslash Q_DX$	00	01	11	10
00	1	1		
01	1	1		
11	1	1		
10	1	1		

Q_A'

	00	01	11	10
	1	1		
	1	1		
	1	1		
	1	1		

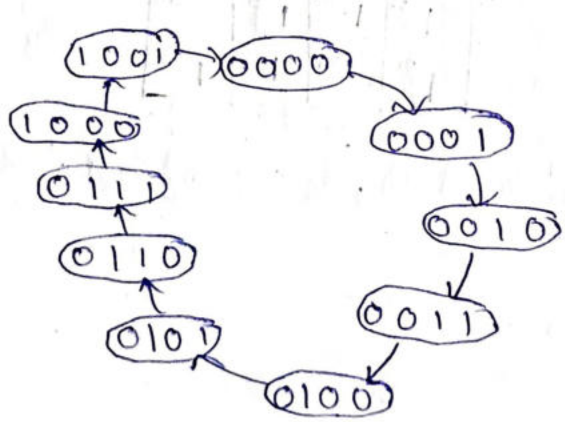
$D_D = Q_D'$

18/11

DESIGN of MOD-N COUNTERS :-

In the design of MOD-N counters, we have to consider 0 to N-1 number of states depends on the n-value. The number of bits required for each and every state can be defined

MOD-10 counter (Decade counter) (BCD counter) :-



P.S	N.S
0000	0001
0001	0010
0010	0011
0011	0100
0100	0101
0101	0110
0110	0111
0111	1000
1000	1001
1001	0000

SR - flip flop :-

0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

terms, we

of States

of bits

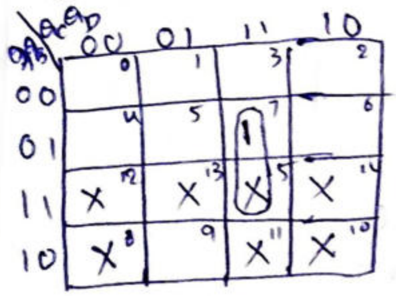
be defined

Q_A	Q_B	Q_C	Q_D	Q_A'	Q_B'	Q_C'	Q_D'	S_A	R_A	S_B	R_B	S_C	R_C	S_D	R_D
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	0
0	0	0	1	0	0	1	0	0	X	0	X	1	0	0	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	0
0	0	1	1	0	1	0	0	0	X	1	0	0	1	0	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	0
0	1	0	1	0	1	1	0	0	X	X	0	1	0	0	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	0
0	1	1	1	1	0	0	0	1	0	0	1	0	1	0	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	0
1	0	0	1	0	0	0	0	0	1	0	X	0	X	0	1

N.S
0001
0010
0011
0100
0101
0110
0111
1000
1001
0000

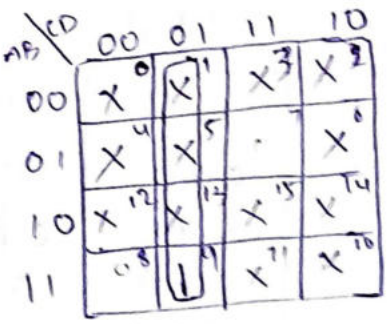
K-maps :-

S_A :-



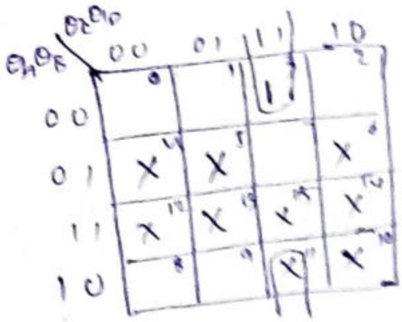
$S_A = Q_B Q_C Q_D$

R_A :-



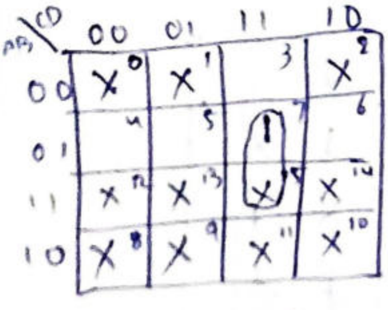
$R_A = Q_C' Q_D$

S_B :-



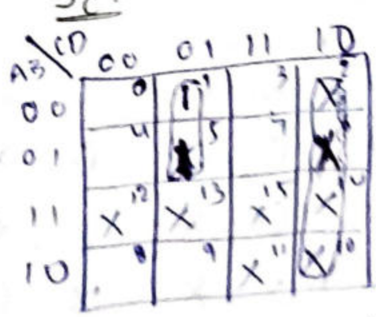
$S_B = Q_B' Q_C Q_D$

R_B :-



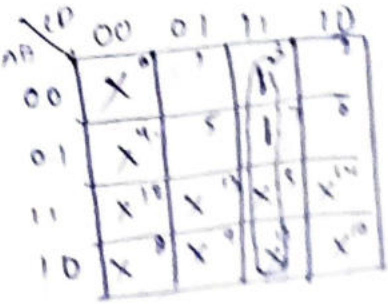
$R_B = Q_B Q_C Q_D$

S_C :-



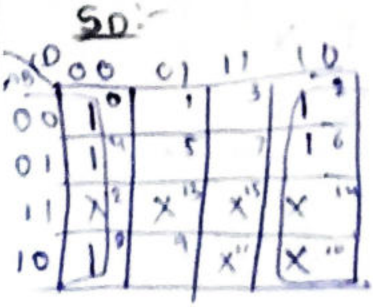
$S_C = \cancel{Q_C Q_D'} + Q_A' Q_C' Q_D$

R_C :-



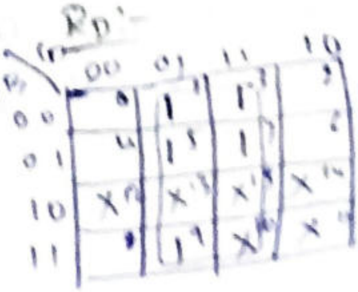
$R_C = Q_C Q_D$

S_D :-

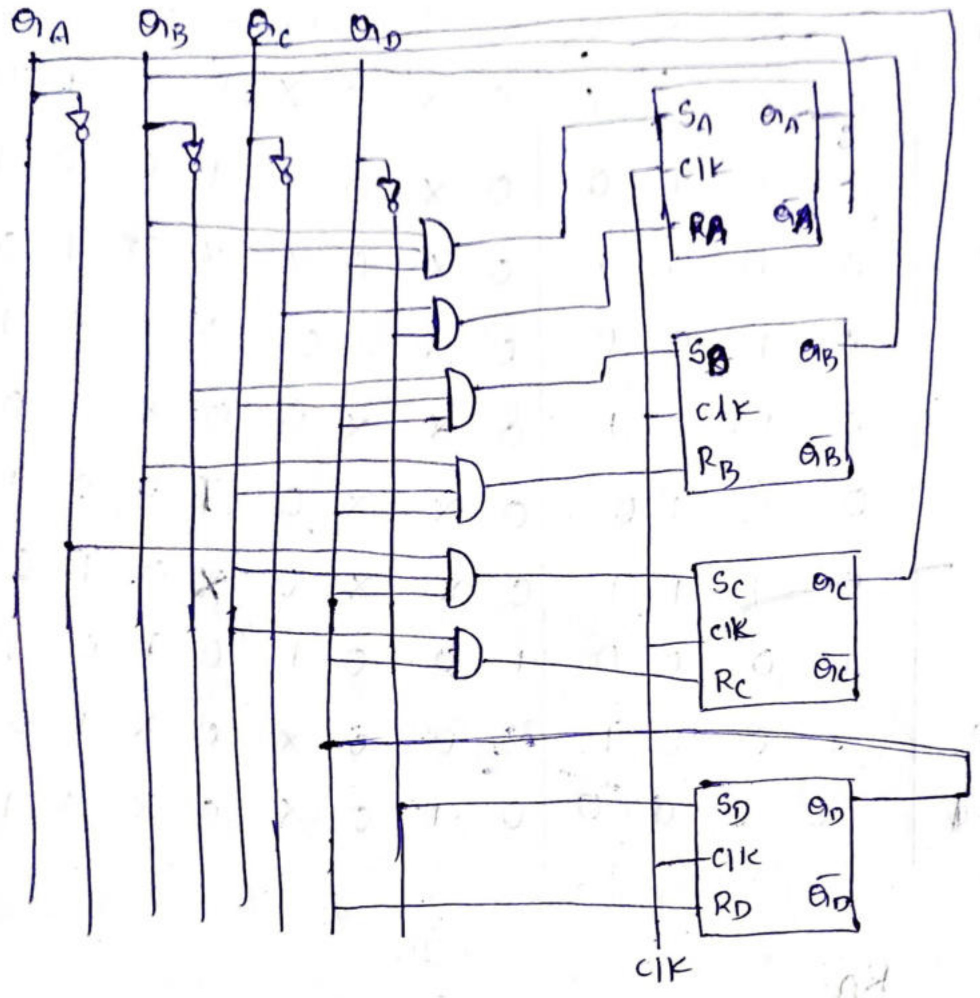


$S_D = Q_D'$

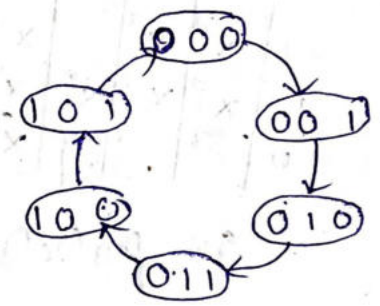
R_D :-



$R_D = Q_D$



* MOD-6 counter using JK FF



P.S	N.S
000	001
001	010
010	011
011	100
100	101
101	000

0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK FF :-

Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	0	X	1	0	X	X	1

K-Maps:

J_A :-

	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

J_A :-

$Q_A \backslash Q_BQ_C$	00	01	11	10
0			1	
1	X	X	X	X

$J_A = Q_B Q_C$

K_A :-

$Q_A \backslash Q_BQ_C$	00	01	11	10
0	X	X	X	X
1		1	X	X

$K_A = Q_C$

J_B :-

$Q_A \backslash Q_BQ_C$	00	01	11	10
0		1	X	X
1			X	X

$J_B = Q_A Q_C$

K_B :-

$Q_A \backslash Q_BQ_C$	00	01	11	10
0	X	X	1	
1	X	X	X	X

$K_B = Q_C$

J_C :-

$Q_A \backslash Q_BQ_C$	00	01	11	10
0	1	X	X	1
1	1	X	X	X

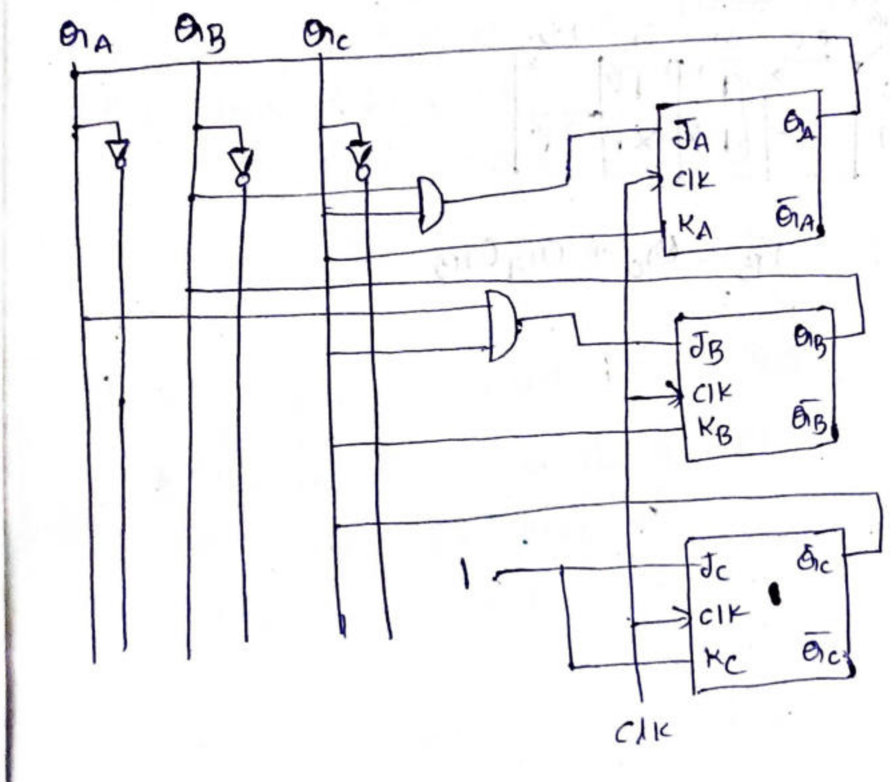
$J_C = 1$

K_C :-

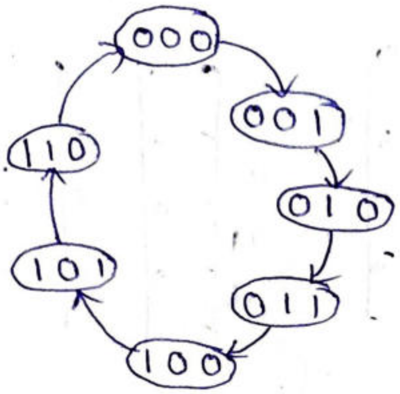
$Q_A \backslash Q_BQ_C$	00	01	11	10
0	X	1	1	X
1	X	1	X	X

$K_C = 1$

logic diagram :-



MOD-7 using T-flipflop:-



P.S	N.S
0 0 0	0 0 1
0 0 1	0 1 0
0 1 0	0 1 1
0 1 1	1 0 0
1 0 0	1 0 1
1 0 1	1 1 0
1 1 0	0 0 0

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

T-Flipflop:-

Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	0	0	0	1	1	0

K-maps:-

T_A:-

$Q_A \backslash Q_B Q_C$	00	01	11	10
0			1	
1			X	1

$$T_A = Q_B Q_C + Q_A Q_B$$

T_B:-

$Q_A \backslash Q_B Q_C$	00	01	11	10
0		1	1	
1		1	X	1

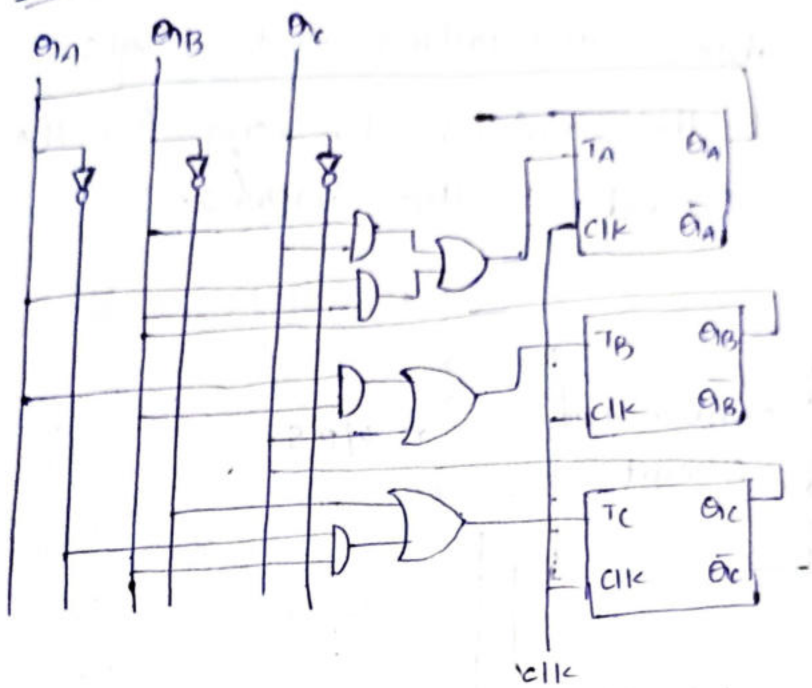
$$T_B = Q_C + Q_A Q_B$$

T_C:-

$Q_A \backslash Q_B Q_C$	00	01	11	10
0	1	1	1	1
1	1	1	X	

$$T_C = Q_B' + Q_A' Q_B$$

logic diagram



20/11

level mode and pulse mode asynchronus sequential circuit

* It consists of a combinational logic circuits and delay elements.

* According to how input variables are to be considered, there are two types of asychronus sequential circuits
   1) level mode ~~asy~~ asynchronus sequential circuit
   2) Pulse mode asynchronus sequencitial circuit

* level mode circuit assumes that
   * The input variables change only when the circuit is stable
   * only one input variable can change at given time.
   * input variables are levels, not pulses.

Pulse mode circuit assumes that

* Input variables are pulses not levels.
* The width of the pulses is long for the circuit to respond to the inputs.

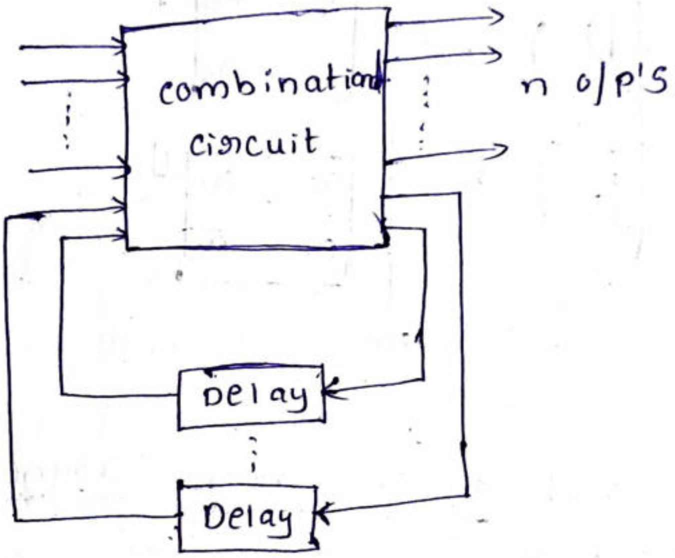


fig: asynchronous sequential circuit

FSM MINIMIZATION

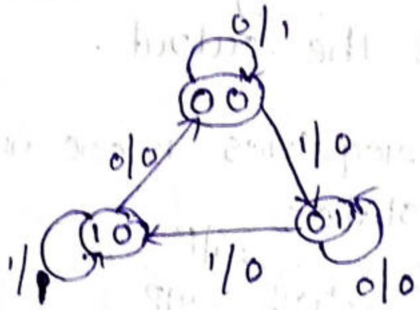
FSM :

Finite state machine is represented as divided into two types. 1) Mealy finite state machine
2) Moore finite state machine.

Both mealy & moore machine's are synchronous sequential circuits are or clocked sequential circuits

1) Mealy FSM :-

state diagram :-

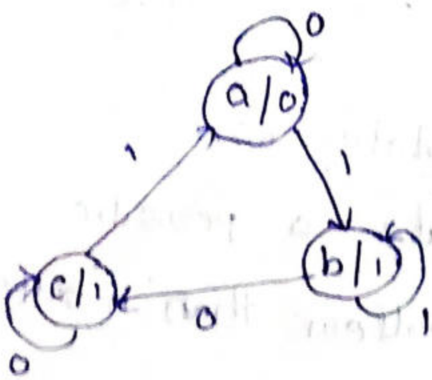


state table :-

P.S	N.S, Z	
	X=0	X=1
00	00,1	01,0
01	01,0	10,0
10	00,0	10,1

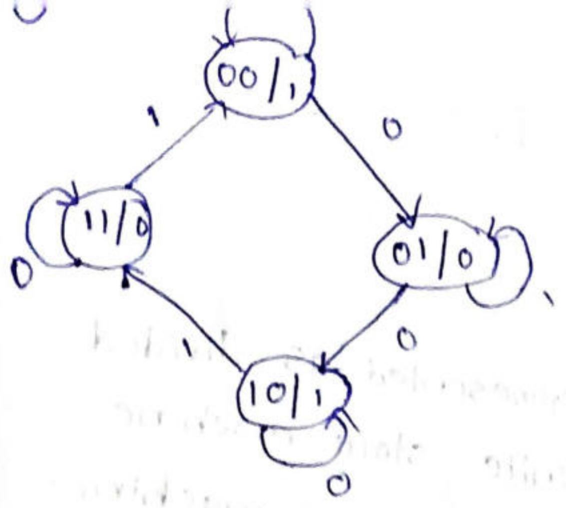
2) Moore FSM :-

a) state diagram :-



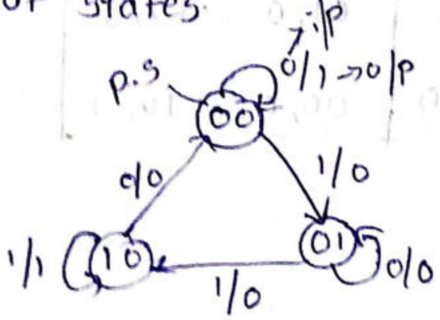
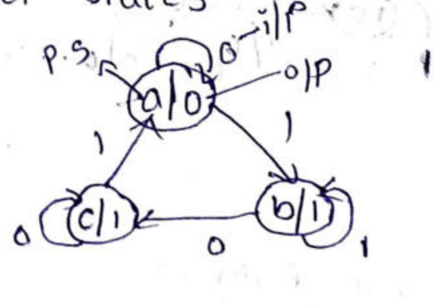
state table :-

P.S	N.S		Z
	X=0	X=1	
a	a	b	0
b	c	b	1
c	c	a	1



P S	N·S		Z
	X = 0	X = 1	
00	01	00	1
01	10	01	0
10	10	11	1
11	11	00	0

Difference b/w Mealy & Moore

Mealy Machine	Moore Machine
1) output is a function of Present state & Present state inputs	1) output is a function of present state only.
2) Input changes may effect the output.	2) Input changes does not effect the output.
3) It requires less number of states	3) It requires more number of states
4) 	4) 

Capabilities & limitations of finite state Machine

1) Periodic sequence of finite state: ~~w~~ we can generate a periodic with 'n' state sequence of n state or smaller than 'n' state

- For example, If it is a 6-state machine we can have 0, 1, 2, 3, 4, 5 states (or) less than 6 states

2) NO infi…
- witho… fi… is pos…

3) limited …

The f… memory … outputs.

Conversion…

1) Mealy to…

(1)

P S	NS
	X = 0…
A	C, 0
B	A, 1…
C	B, 1…
D	D, …

State diagra…

2) No infinite sequence :- with fini

with finit state machine No infinite sequence is possible

3) limited memory :-

The finite state machine having limited memory. that is why we can not find certain outputs.

Conversions of finite state machine :-

I) Mealy to Moore FSM conversions :-

(i)

P.S	NS, Z	
	X = 0	X = 1
A	C, 0	B, 0
B	A, 1	D, 0
C	B, 1	A, 1
D	D, 1	C, 0

$A \rightarrow 1$

$B_0 \rightarrow 0$

$B_1 \rightarrow 1$

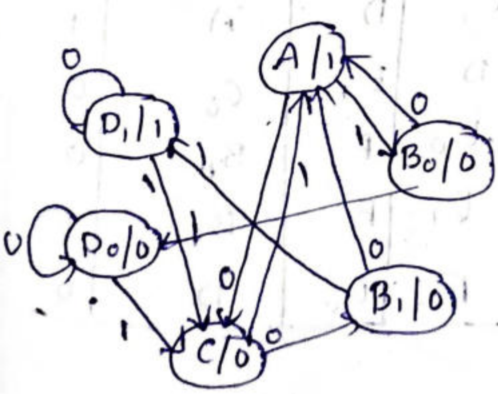
$C \rightarrow 0$

$D_0 \rightarrow 0$

$D_1 \rightarrow 1$

P.S	N.S		Z
	X = 0	X = 1	
A	C	B_0	1
B_0	A	D_0	0
B_1	A	D_1	1
C	B_1	A	0
D_0	D_0	C	0
D_1	D_1	C	1

State diagram :-

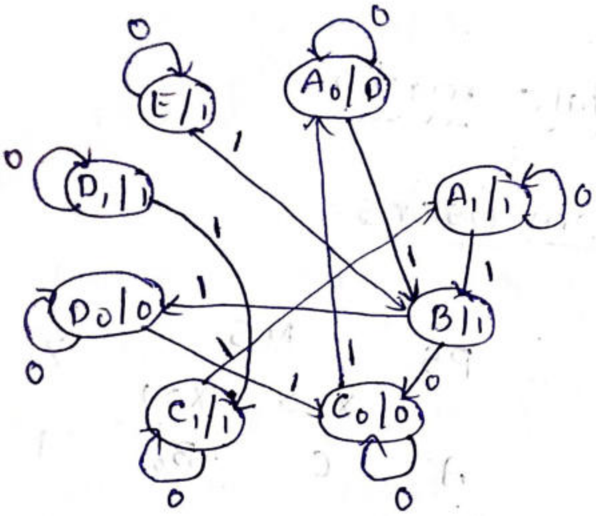


②

PS	NS, Z x=0	NS, Z x=1
A	A,0	B,1
B	C,0	D,0
C	C,1	A,1
D	D,1	C,0
E	E,1	B,1

$A_0 \to 0$
$A_1 \to 1$
$B \to 1$
$C_0 \to 0$
$C_1 \to 1$
$D_0 \to 0$
$D_1 \to 1$
$E \to 1$

PS	NS x=0	NS x=1	Z
A_0	A_0	B	0
A_1	A_1	B	1
B	C_0	D_0	1
C_0	C_0	A_0	0
C_1	C_1	A_1	1
D_0	D_0	C_0	0
D_1	D_1	C_1	1
E	E	B	1

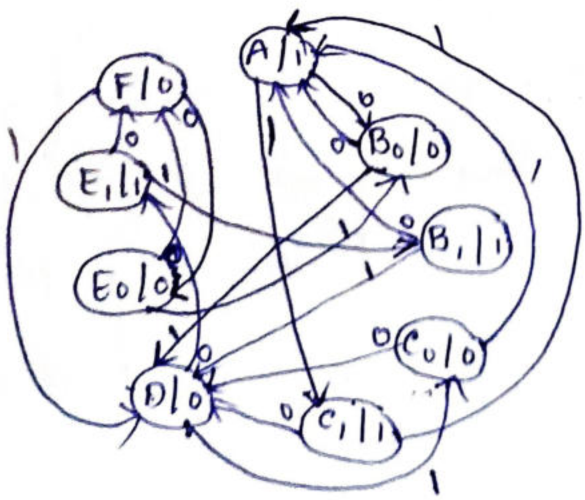


③

PS	NS, Z x=0	NS, Z x=1
A	B,0	C,1
B	A,1	D,0
C	D,0	A,1
D	E,1	C,0
E	F,0	B,1
F	E,0	D,0

$A \to 1$
$B_0 \to 0$
$B_1 \to 1$
$C_0 \to 0$
$C_1 \to 1$
$D \to 0$
$E_0 \to 0$
$E_1 \to 1$
$F \to 0$

P.S	N.S x=0	N.S x=1	Z
A	B_0	C_1	1
B_0	A	D	0
B_1	A	D	1
C_0	D	A	0
C_1	D	A	1
D	E_1	C_0	0
E_0	F	B_0	0
E_1	F	B_1	1
F	E_0	D	0



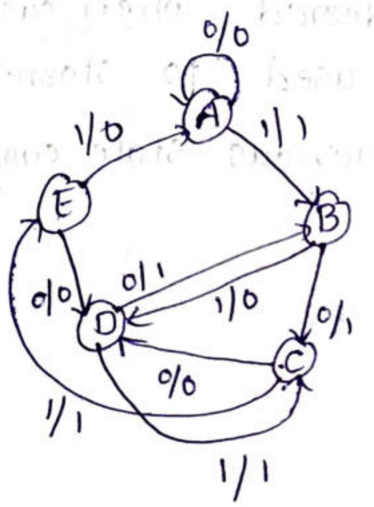
Moore to Mealy conversion :-

(1)

P.S	N.S X=0	N.S X=1	Z
A	A	B	0
B	C	D	1
C	D	E	1
D	B	C	0
E	D	A	1

A → 0
B → 1
C → 1
D → 0
E → 1

P.S	N.S, Z X=0	N.S, Z X=1
A	A, 0	B, 1
B	C, 1	D, 0
C	D, 0	E, 1
D	B, 1	C, 1
E	D, 0	A, 0

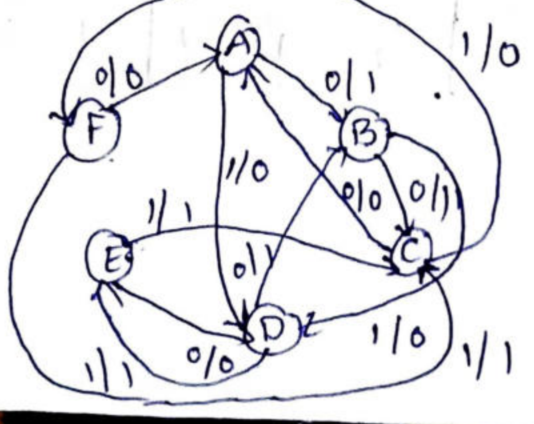


(2)

P.S	N.S X=0	N.S X=1	Z
A	B	D	0
B	C	D	1
C	A	F	1
D	B	E	0
E	D	C	1
F	A	C	0

A → 0
B → 1
C → 1
D → 0
E → 1
F → 0

P.S	N.S, Z X=0	N.S, Z X=1
A	B, 1	D, 0
B	C, 1	D, 0
C	A, 0	F, 0
D	B, 1	E, 1
E	D, 0	C, 1
F	A, 0	C, 1

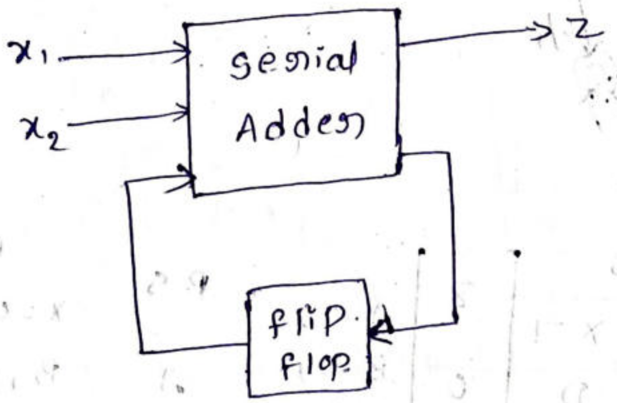


21/4

Serial binary adder:-

In serial binary adder, we are having two inputs x_1 & x_2 and one output z. where x_1 and x_2 are the control inputs when present state is moving to the next state, we have to consider control inputs x_1 & x_2.

In serial binary adder, we have two states '0' & '1' (or) 'A' & 'B', A is considered when carry = 0, B is considered when carry = 1. From diagram, flip flop is used to store the one-bit information i.e., previous state carry value.



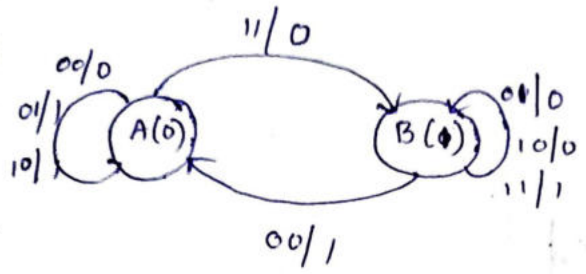
A = 0 (when carry is 0)

i/p's			o/p's	
x_1	x_2	C_{in}	S	C_{out} (next state)
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1

B = 1 (when carry is 1)

i/p's			o/p's	
x_1	x_2	C_{in}	S	C_{out}
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

state diagram



P.S	NS, Z			
	00	01	10	11
A	A,0	A,1	A,1	B,0
B	A,1	B,0	B,0	B,1

0	0	0
0	1	1
1	0	0
1	1	1

Flow table :-

P S (Q_A)	x_1	x_2	N.S (Q_n^+)	D_A	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1

K-maps :-

Q_A :-

Q_A \ x_1x_2	00	01	11	10
0			1	
1		1	1	1

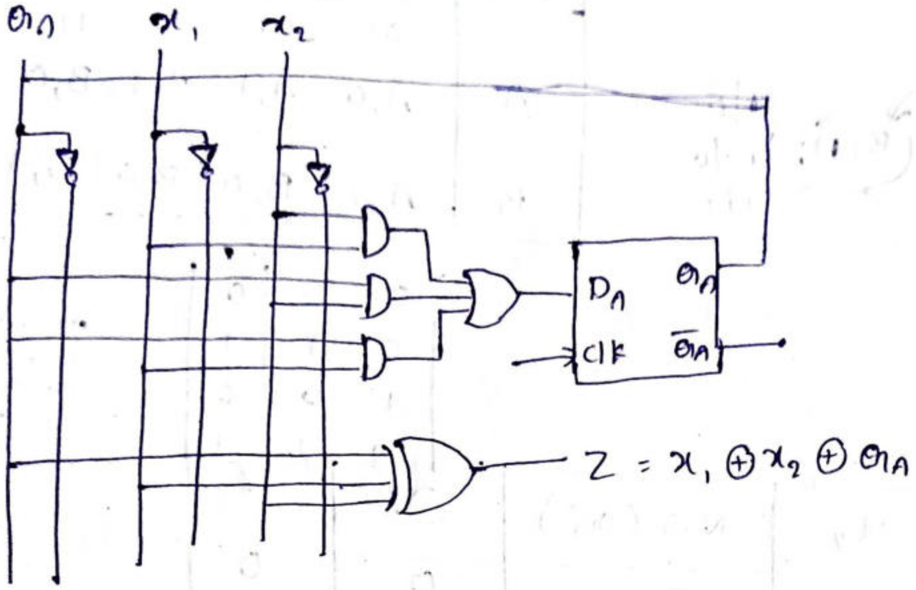
$Z = x_1x_2 + Q_Ax_2 + Q_Ax_1$

Z :-

Q_A \ x_1x_2	00	01	11	10
0		1		1
1	1		1	

$Z = Q_A'x_1'x_2 + Q_Ax_1'x_2' + Q_Ax_1x_2 + Q_A'x_1x_2'$

$\therefore Z = Q_A \oplus x_1 \oplus x_2$



22/11

ASM Charts (Algorithmic state Machine) :-

1) The ASM means algorithmic state machine. It describes the sequence of events as well as timing relationship between the states of a sequencial controller and the events that occur while going from one state to the next state.

2) An ASM chart contains 1 or more inter connected ASM blocks. Each ASM block contains exactly one state box with any number of decisional and conditional of output boxes associated with the states.

3) Every block in an ASM chart specifies the operations that are to be performed during one common clock pulse.

4) As ASM block has exactly one entrance path and one or more exit paths represented by the structure of the decision boxes.

5) A path trough an ASM block from etrance to exit is known as link path. The internal feedback within the ASM block is not permitted.

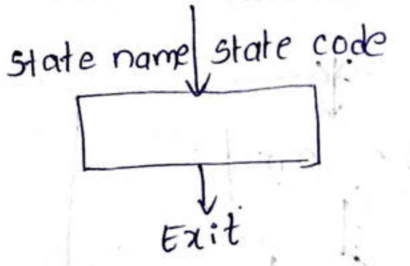
6) The operation specified within the state & conditional output boxes in the block are performed in the data path sub system.

*Components of ASM charts :-

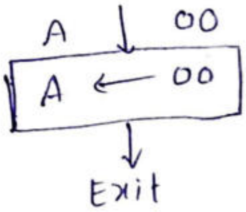
It includes
1) State box
2) Decision box
3) Conditional output box

State box :-

State box of an ASM chart is in rectangular shape.

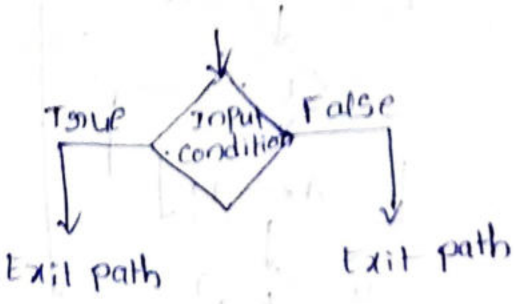


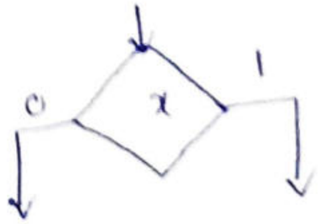
Ex :-



Decision box :-

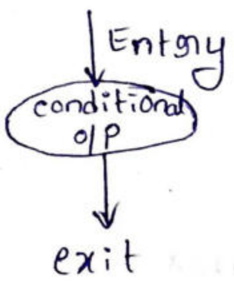
The Decision box is in rhombous shape and the decision boxes, the inputs are taken from the decision boxes, there for more number of exits path are the Decision box

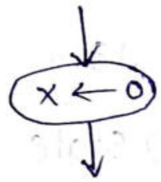




Conditional output box:

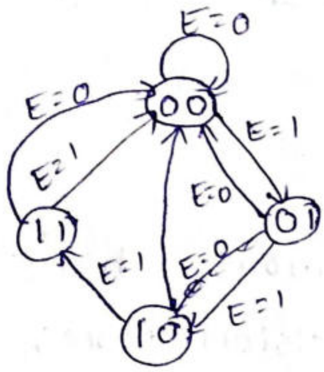
The shape of the conditional output box is ovel. It is used always after the decision box. Before decision box we can not use the conditional output boxes.

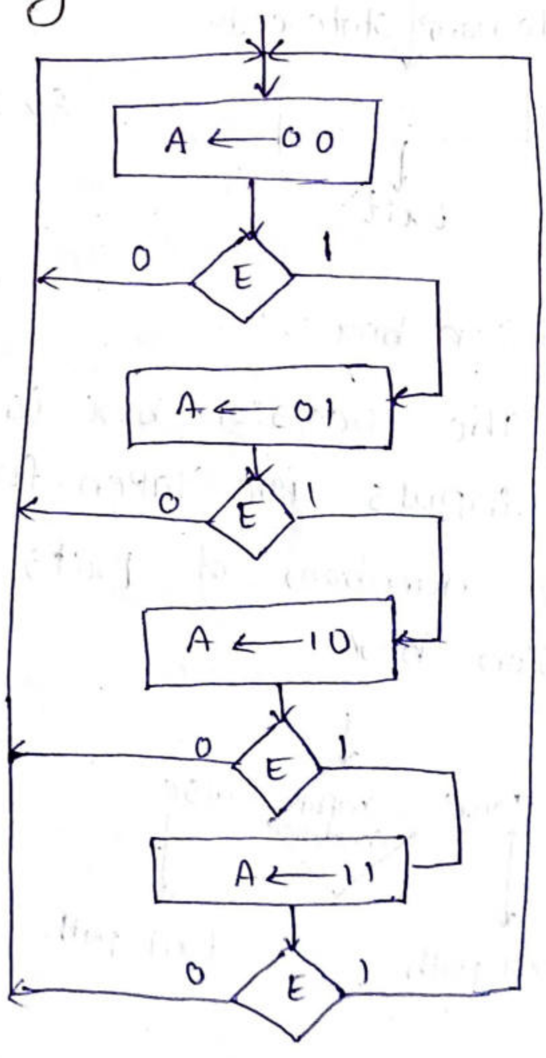




Examples:-

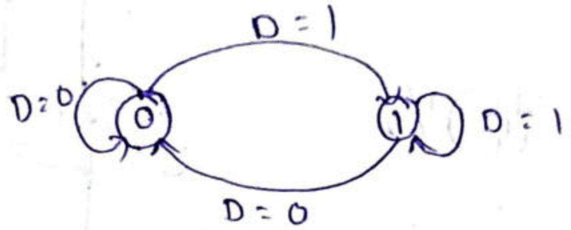
1) Draw the ASM chart for a two bit binary counter having one enable input 'E' such that E=1, then counting is enabled otherwise It is disable.

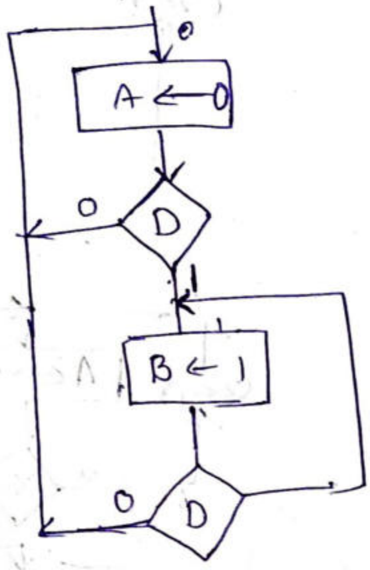




2) Draw the ASM chart for D flip flop :

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

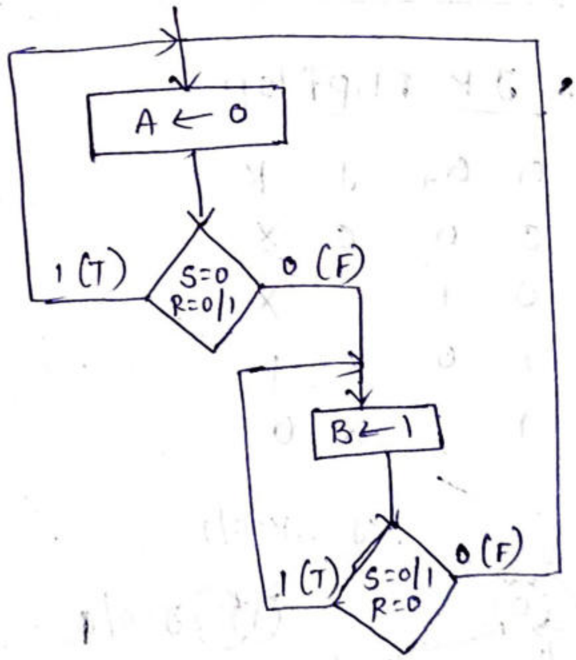


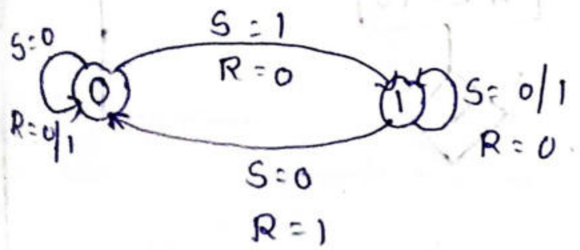


H.W JK flip flop

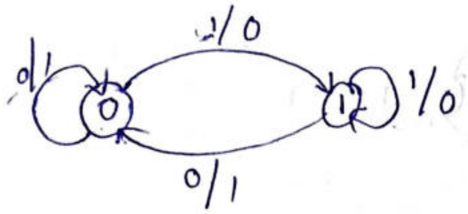
3) Draw the ASM chart for SR flip flop :-

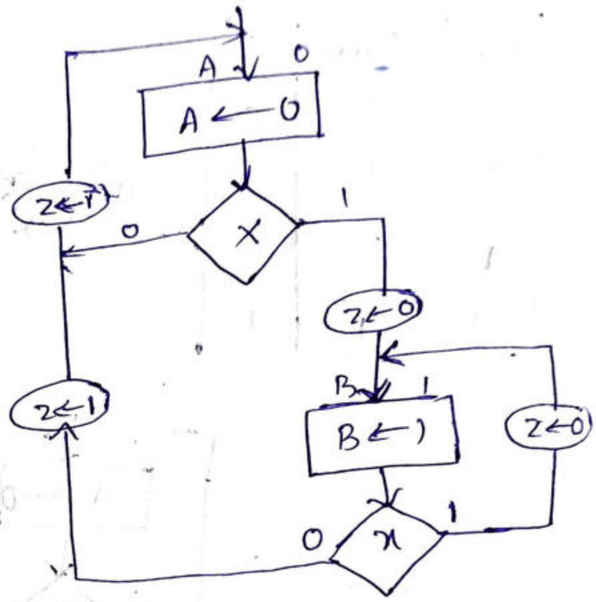
Q	Q_+	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0





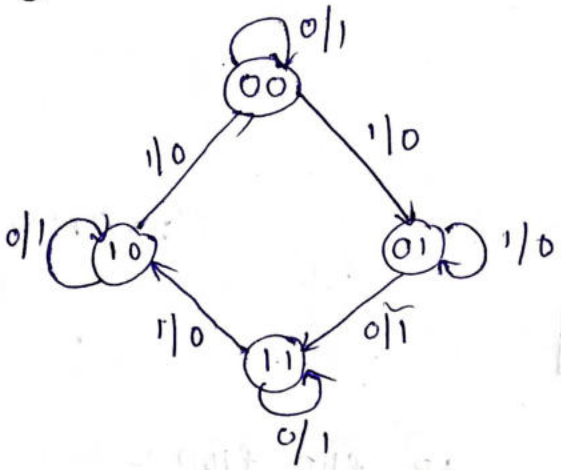
4) Draw the ASM chart for the given state diagram

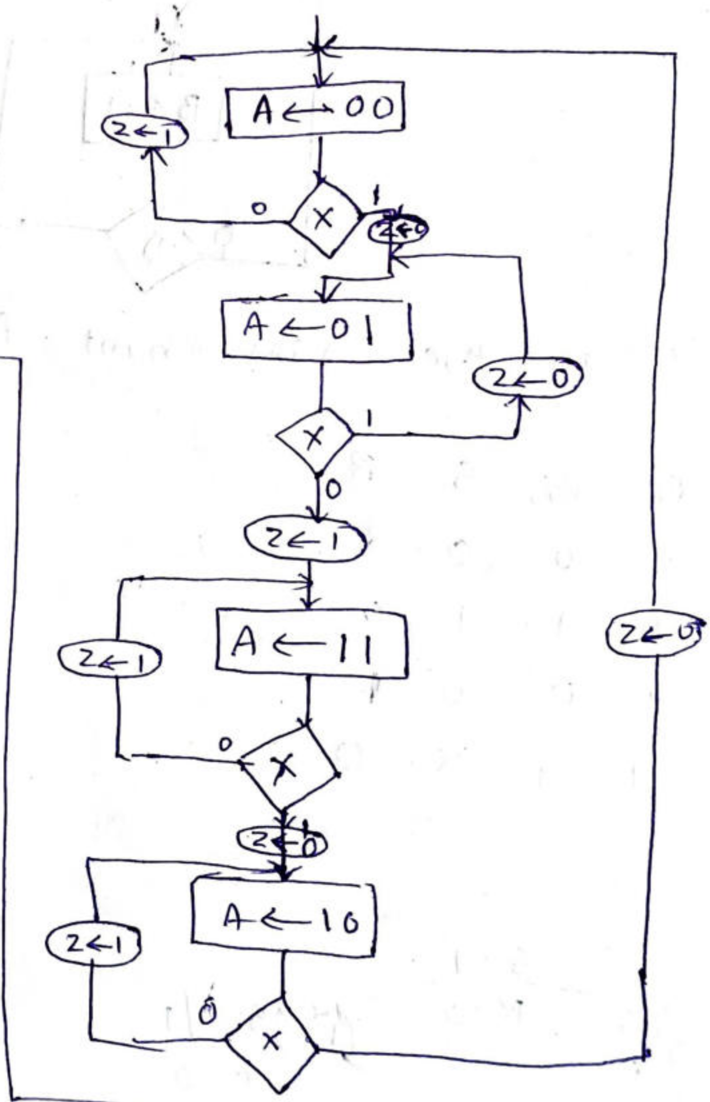




H.W

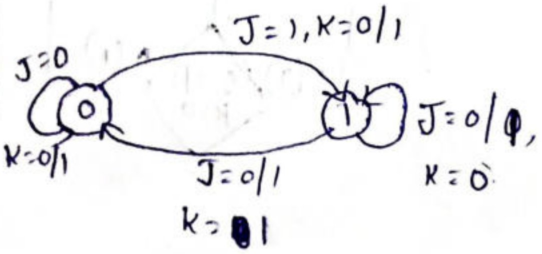
5

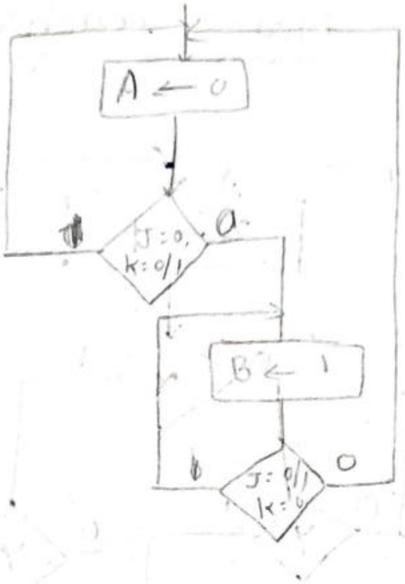




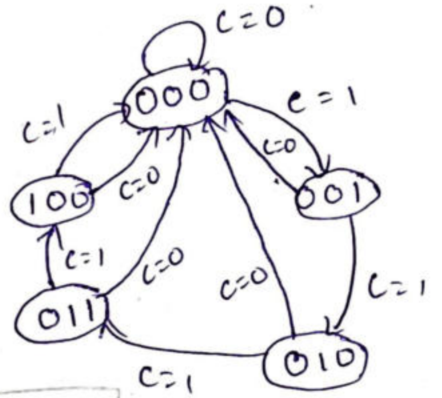
* JK Flipflop

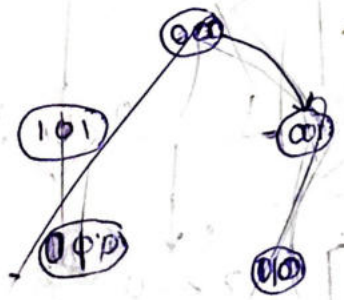
Q_n	Q_+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

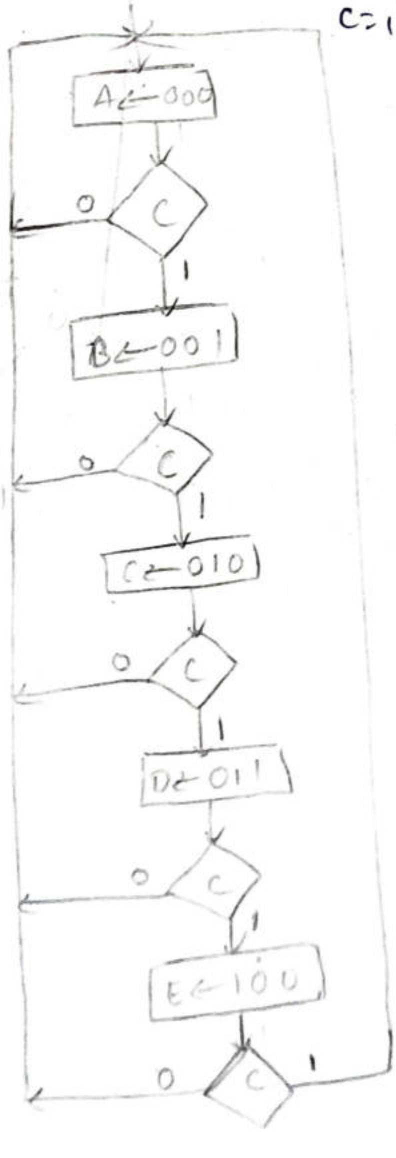




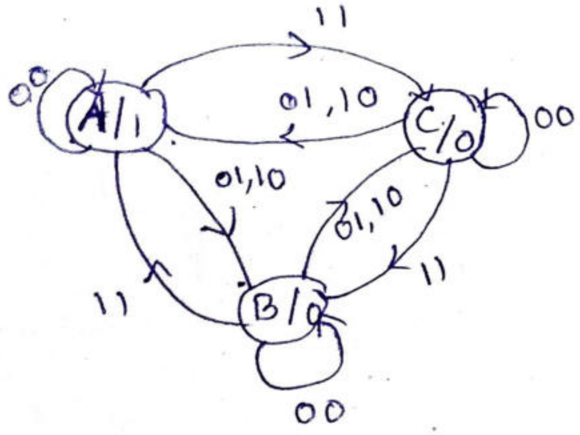
Draw the ASM charts for MOD-5 counter. If C=1, enable operation is performed and if C=0 it is disable.

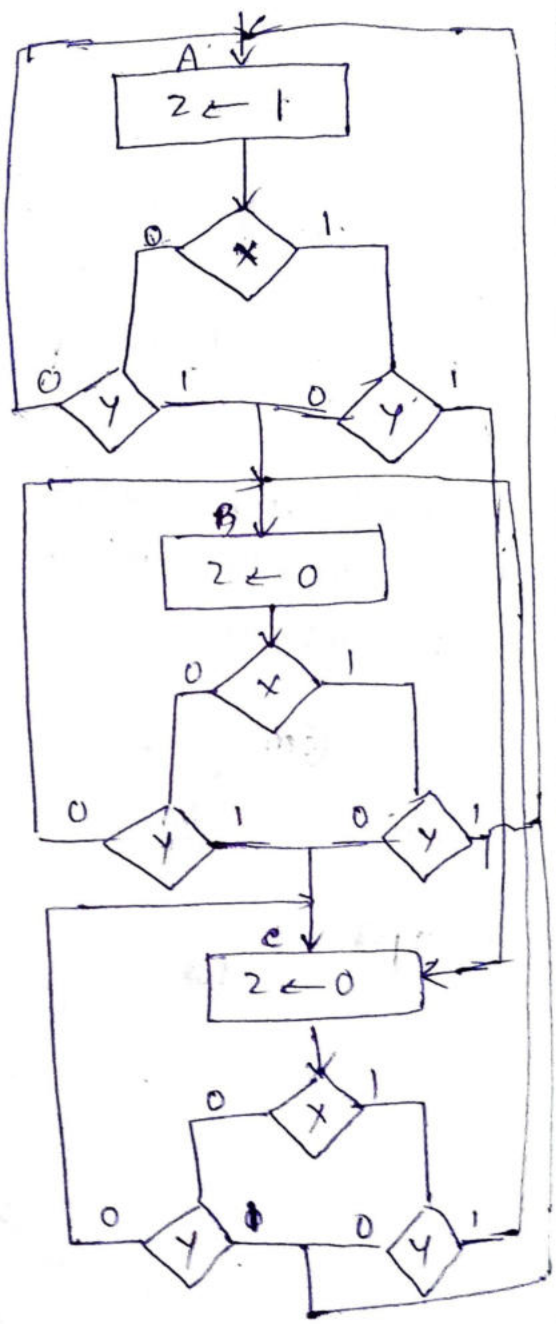




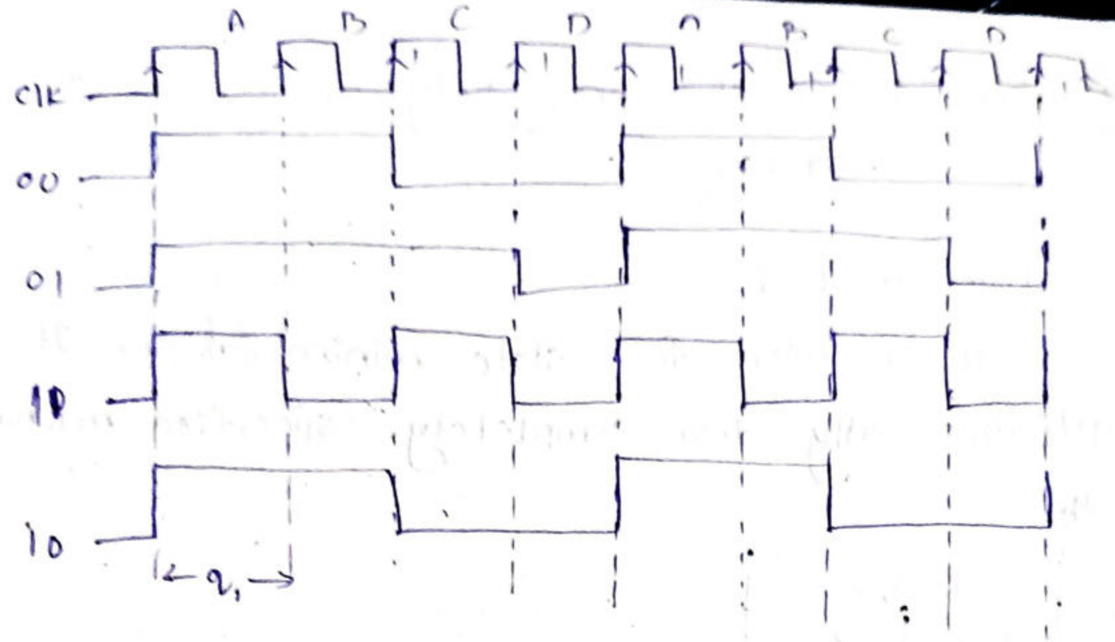


Draw an ASM chart for Moore machine:-

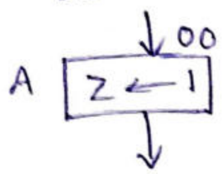


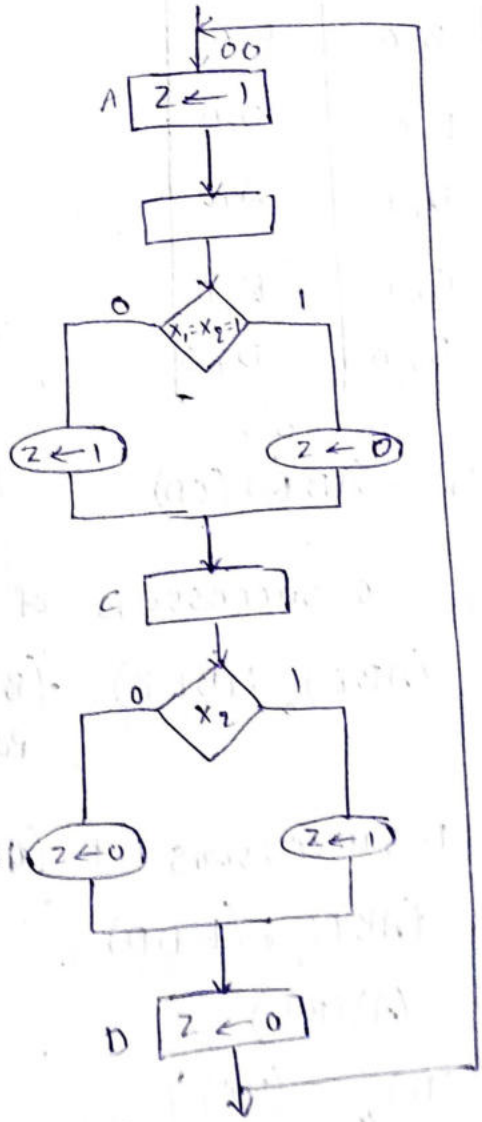


②

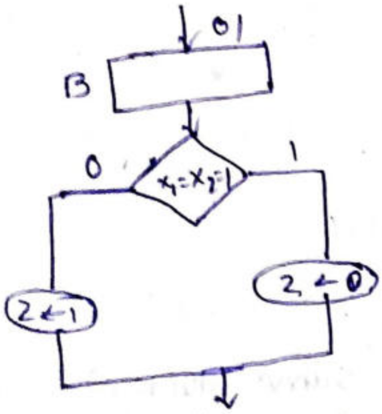


ASM Block for 1st clock pulse

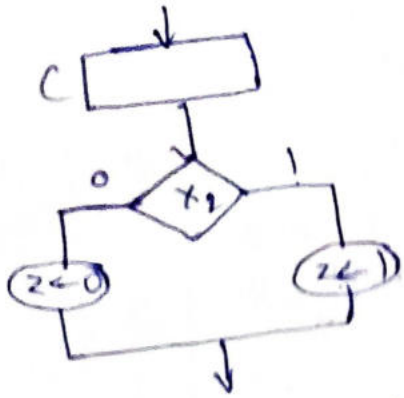




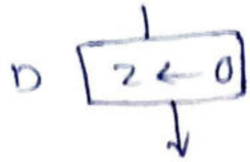
second clock pulse



third clock pulse



fourth clock pulse



Minimization of completely specified sequential machine's

Partition method :-

It is used for state minimization. It is applicable only for completely specified machine.

so the.

P.S	N.S, Z	
	$x=0$	$x=1$
A	B, 0	E, 0
B	E, 0	D, 0
C	D, 1	A, 0
D	C, 1	E, 0
E	B, 0	D, 0

Ans) P_1 :- (ABE) (CD)
output same

P_2 :- 0-successors of (ABE)

$(ABE)_0$: (BEB) -(B & E are in same block, so Partition is not possible)

1-successors of (ABE)

$(ABE)_1$: (EDD)

(A)(BE)

→ $(CD)_0$: (DC)
$(CD)_1$: (AE)
No partition requ

P_2 = (A) (BE) (CD)

P_3 :- $(BE)_0 \rightarrow (EB)$
$(BE)_1 \rightarrow (DD)$ } no partition

$(CD)_0 \rightarrow (DC)$
$(CD)_1 \rightarrow (AE)$

$(C)(D)$

$P_3 = (A)(C)(D)(BE)$.

→ B & E are equivalent states,
so $B = E$.

~~states~~,

Reduced state table

PS	NS, Z	
	x=0	x=1
A	B,0	B,0
B	B,0	D,0
C	D,1	A,0
D	B,1	B,0

(2)

PS	NS, Z	
	x=0	x=1
A	C,0	F,0
B	D,1	F,0
C	E,0	B,0
D	B,1	E,0
E	D,0	B,0
F	D,1	B,0

~~P_1 :- (ACE)(BDF)~~

~~P_2 :- 0-successors of (ACE)~~

$(ACE)_0 = (CED)$

$(AC)(E)$

$(ACE)_1 = (FBB)$

$P_2 = (AC)(E)(BDF)$

P_3 :- $(AC)_0 - (CE)$, $(A)(C)$

$(AC)_1 - (FB)$

~~$(BDF)_0 - (DBD)$, $(BDF)_1 : (FEB)$~~

~~$(BF)(D)$~~

~~$P_3 = (AC)(E)(BF)(D)$~~

~~$(BF)_0 - (DD)$, $(BF)_1 - (FB)$~~

~~$P_3 = (AC)(E)(BF)$~~ D

$P_1 \Rightarrow (ACE)(BDF)$

$P_2 \Rightarrow (ACE)_0 - (CED) \Rightarrow (AC)(E)$

$(ACE)_1 - (FBB)$

$(BDF)_0 - (DBD)$

$(BDF)_1 - (FEB) \Rightarrow (BF)(D)$

$P_2 = (AC)(E)(BF)(D)$

$P_3 \Rightarrow (AC)_0 - (CE) \Rightarrow (A)(C)$

$(AC)_1 - (FB)$

$(BF)_0 - (DD)$

$(BF)_1 - (FB)$

$P_2 = (A)(C)(E)(BF)(D)$

$\Rightarrow$ B & F are equivalent states

$\therefore B = F$

P.S	NS, Z	
	X=0	X=1
A	C,0	B,0
B	D,1	B,0
C	E,0	B,0
D	B,1	E,0
E	D,0	B,0

(3)

PS	NS, Z x = 0	NS, Z x = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	E, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

$P_1 \Rightarrow (BDF)(ACE)$

$P_2 \Rightarrow (BDF)_0 - (FEB) \Rightarrow (BF)(D)$

$(BDF)_1 - (DBC) \Rightarrow (BD)(F)$

Take only 1 partition

$(ACE)_0 - (EEC)$

$(ACE)_1 - (DBF)$

$P_2 = (BD)(F)(ACE)$

$P_3 \Rightarrow (BD)(F)(ACE)$

$(BD)_0 - (FE) \Rightarrow (B)(D)$

$(BD)_1 - (DB)$

*[~~$P_3 = (B)(D)(F)(ACE)$~~

~~ACEEC are equivalent states~~

~~A=C=E~~]*

P.S	NS, Z x = 0	NS, Z x = 1
A	E, 0	
B	F, 0	
C		
D		
E		
F		

$(ACE)_0 = (EEC)$

$(ACE)_1 = (DBF) \Rightarrow (AC)(E)$

$P_3 = (B)(D)(F)(AC)(E)$

$P_4 \Rightarrow (AC)_0 - (EE)$

$(AC)_1 - (DB) \rightarrow (A)(C)$

$P_4 = (A)(B)(C)(D)(E)(F)$

Here we can not minimize the states because there is no equivalent states present in the given state table.

4

P.S	NS, Z	
	X=0	X=1
A	F,0	B,1
B	G,0	A,1
C	B,0	C,1
D	C,0	B,1
E	D,0	A,1
F	E,1	F,1
G	E,1	G,1

P_1 :- (ABCDE) (FG)

P_2 :- $(ABCDE)_0$ - (FGBCD)

⇒(AB) (CDE)

$(ABCDE)_1$ - (BACBA)

$(FG)_0$ - (EE)

$(FG)_1$ - (FG)

P_2 = (AB) (CDE) (FG)

⇒ P_3 ⇒ $(AB)_0$ - (FG)

$(AB)_1$ - (BA)

⇒$(CDE)_0$ - (BCD) ⇒ (C) (DE)

$(CDE)_1$ - (CBA) ⇒ (C) (DE)

⇒$(FG)_0$ - (EE)

$(FG)_1$ - (FG)

P_3 = (AB) (C) (DE) (FG)

⇒ P_4 ⇒ $(AB)_0$ - (FG)

$(AB)_1$ - (AB)

$(DE)_0$ - (CD) - (D) (E)

$(DE)_1$ - (BA)

$(FG)_0$ - (EE)

$(FG)_1$ - (FG)

P_4 = (AB) (C) (D) (E) (FG)

P S	NS, Z	
	X=0	X=1
A	F, 0	A, 1
C	A, 0	C, 1
D	C, 0	A, 1
E	D, 0	A, 1
F	E, 1	F, 1

⇒ P_5 ⇒ $(AB)_0$ - (FG)

$(AB)_1$ - (AB)

$(FG)_0$ - (EE)

$(FG)_1$ - (FG) ⇒ P_5 = (AB) (C) (D) (E) (FG)

∴ E

SI

∴ E

A & B and F & G are equivalent states

Then A = B, F = G

(5)

P.S	N.S	
	X = 0	X = 1
A	B, 1	H, 1
B	F, 1	D, 1
C	D, 0	E, 1
D	C, 0	F, 1
E	D, 1	C, 1
F	C, 1	C, 1
G	C, 1	D, 1
H	C, 0	A, 1

P_1 :- (ABEFG)(CDH)

P_2 :- $(ABEFG)_0$ - (BFDCC) ⇒ (AB)(EFG)

$(ABEFG)_1$ - (HDCCD)

$(CDH)_0$ - (DCC)

$(CDH)_1$ - (EFA) ⇒

P_2 = (AB)(EFG)(CDH).

P_3 :- $(AB)_0$ - (BF) ⇒ (A)(B)

$(AB)_1$ - (HD)

$(EFG)_0$ - (DCC)

$(EFG)_1$ - (CCD)

$(CDH)_0$ - (DCC)

$(CDH)_1$ - (EFA) ⇒ (CD)(H)

P_3 = (A)(B)(EFG)(CD)(H)

P_4 :- $(EFG)_0$ - (DCC)

$(EFG)_1$ - (CCD)

$(CD)_0$ - (DC)

$(CD)_1$ - (EF)

P_4 = (A)(B)(EFG)(CD)(H).

P.S	N.S	
	X = 0	X = 1
A	B, 1	H, 1
B	E, 1	C, 1
C	C, 0	E, 1
E	C, 1	C, 1
H	C, 0	A, 1

∴ EFG & CD are equivalent states.

∴ E = F = G & C = D

COUNTERS:

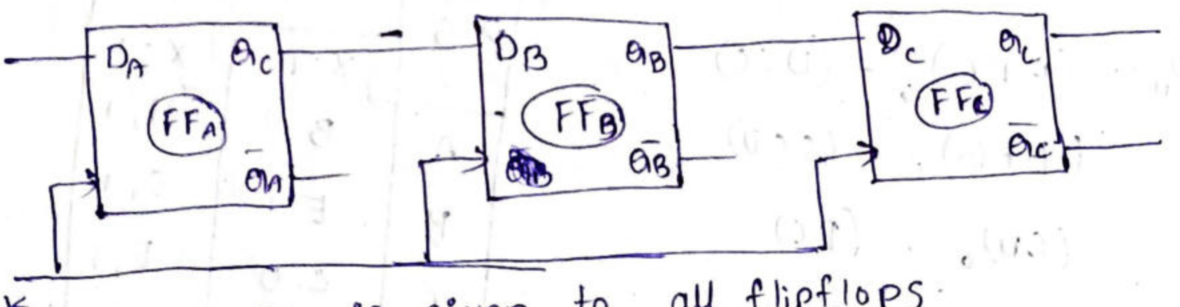
A digital counter is a set of flipflops whose states change in response pulses applied at input to the counter. The flip flops are interconnected such that, their combined state at any time is the binary equivalent of the total no. of Pulses.

* Counter is used to count the pulses.
* These are also used to perform the timing function as in digital watches, frequency counters etc.

Counters are divided into two types

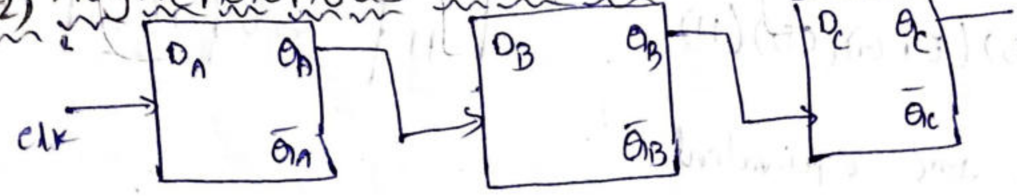
1) synchronous counters
2) Asynchronous counters / ripple counters

1) Synchronous counters:



Same clk is given to all flipflops.

2) Asynchronous counters:-



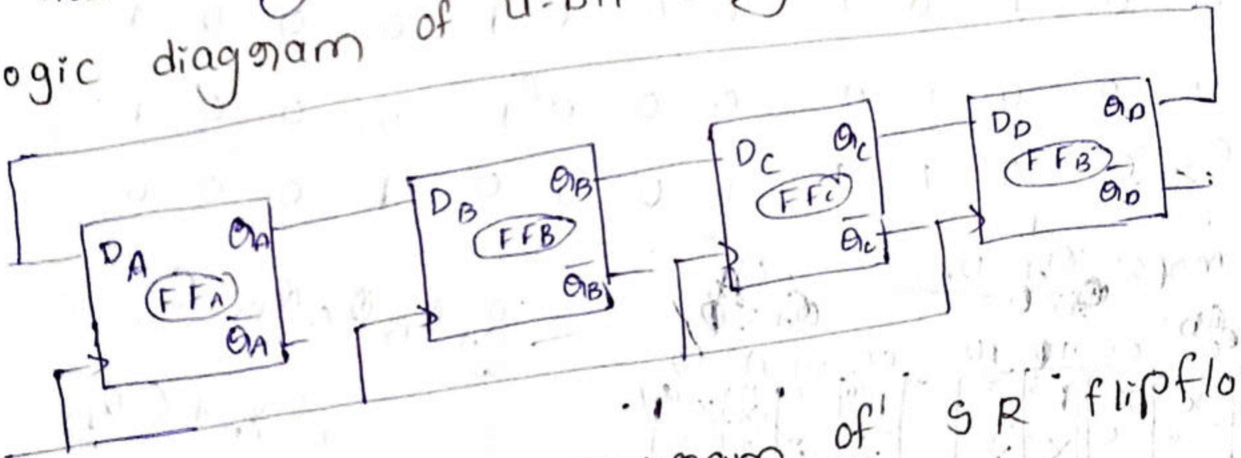
Shift register counters :-

1) ring counter
2) Twisted ring counter / Johnson counter.

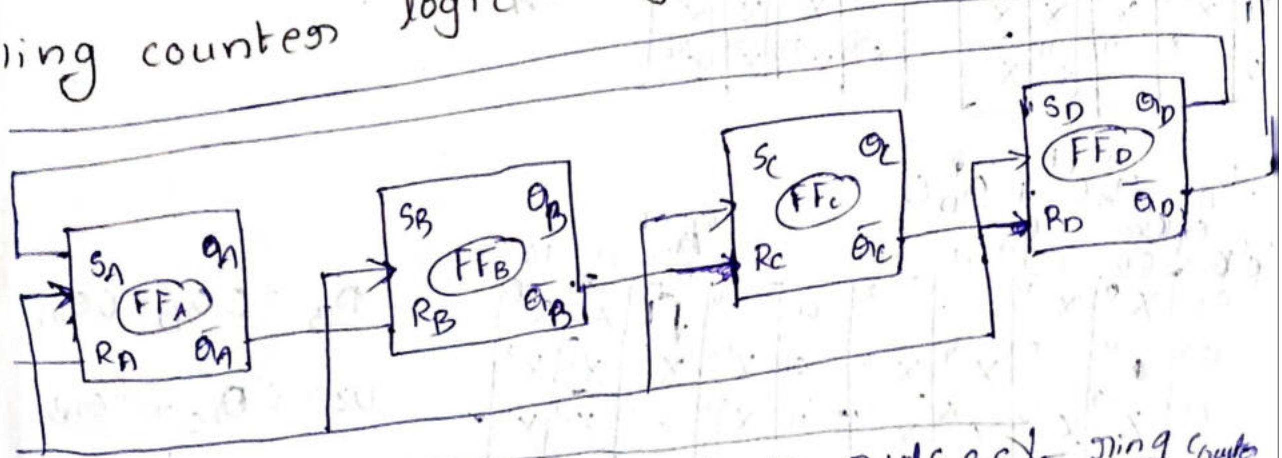
1) Ring counter :-

It is the simplest shift register counter. In this flipflops are arranged in a normal shift register i.e., Q output of each stage is connected to the D input of the next stage. But the Q output of last flipflop is connected back to the D input of first flip flop, such that, the array of flipflops is arranged in a ring and therefore the name ring counter.

Logic diagram of 4-bit ring counter :-

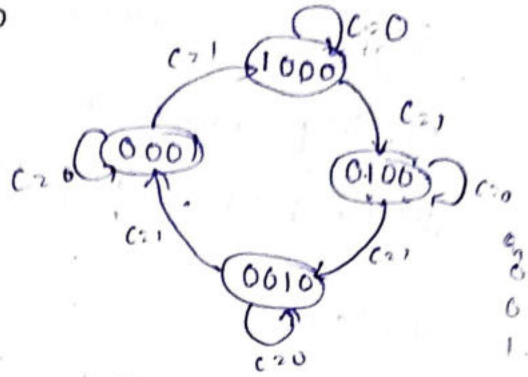


Ring counter logic diagram of SR flipflop:



$$\text{No. of pulses} = \begin{cases} \text{No. of clock pulses} & \text{- ring counter} \\ 2 \times \text{No. of clock pulses} & \text{- Johnson} \end{cases}$$

Clk	Q_A	Q_B	Q_C	Q_D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

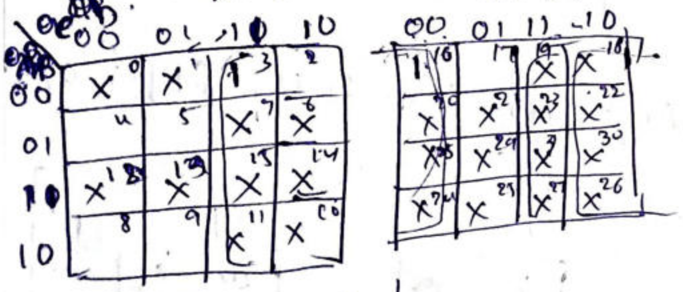


P.S

Q_A	Q_B	Q_C	Q_D	i/p C	Q_A^+	Q_B^+	Q_C^+	Q_D^+	D_A	D_B	D_C	D_D
φ 0	0	0	0	0	1	0	0	0	1	0	0	0
φ 0	0	0	0	1	0	1	0	0	0	1	0	0
8 0	1	0	0	0	0	1	0	0	0	1	0	0
9 0	1	0	0	1	0	0	1	0	0	0	1	0
4 0	0	1	0	0	0	0	1	0	0	0	1	0
5 0	0	1	0	1	0	0	0	1	0	0	0	1
2 0	0	0	1	0	0	0	0	1	0	0	0	1
3 0	0	0	1	1	1	0	0	0	1	0	0	0

K-maps: Q_A D_A =

$Q_A = 0$ $Q_A = \phi$



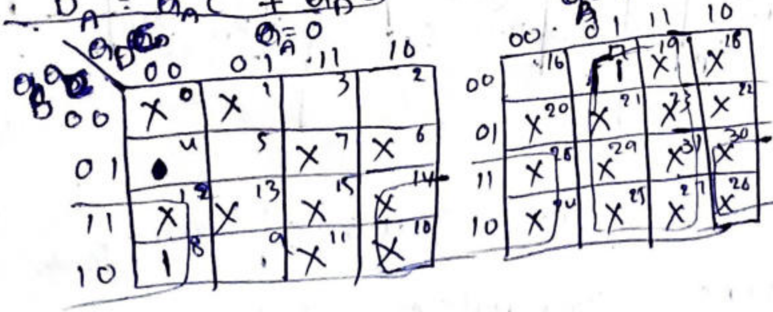
$Q_A Q_B Q_C Q_D C$

$\bar{C} Q_A + C Q_D$

$Q_A' C'$

$D_A = Q_A' C' + Q_D C$

$Q_A = 0$ $Q_A = 1$



$D_B = C' Q_B + C Q_A$

$D_C = C' Q_C + C Q_B$

$D_D = C' Q_D + C Q_C$

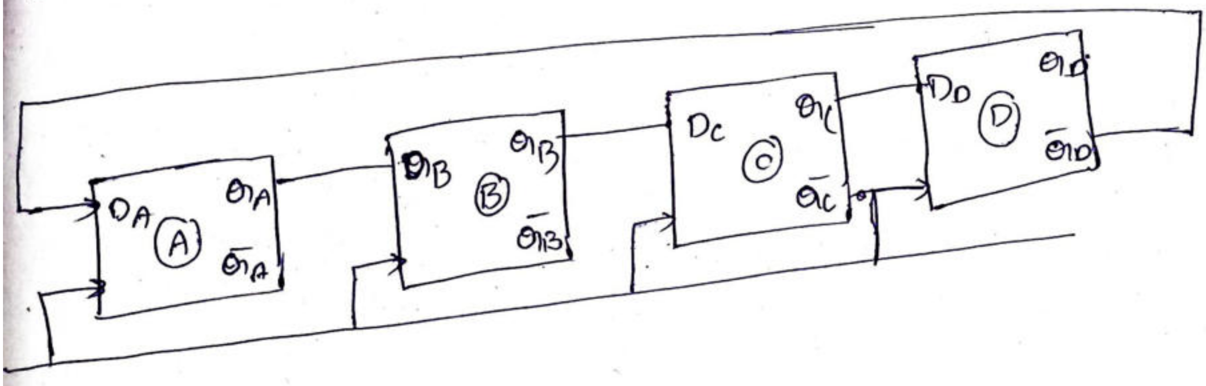
to
th
th
th
as
ca
ak
Q_A
* An
State

Twisted ring counter:-

This counter is obtained from a serial in, serial out shift register by providing feedback from the inverted o/p of the last flip from to the d-input of the first flip flop.

The Q output of each state is connected to the D-input of the next state, but $\bar{Q}$ i/p of the D-input of the last stage is connected to the D-input of the first stage. Therefore the name arrival as "twisted ring counter".

The logic diagram of a 4-bit Johnson counter is shown in the below figure.



Q_A	Q_B	Q_C	Q_D	After clk pulse
1	0	0	0	1
1	1	0	0	2
1	1	1	0	3
1	1	1	1	4
0	1	1	1	5
0	0	1	1	6
0	0	0	1	7
0	0	0	0	8

* An n-flipflop Johnson counter can have 2n unique state and can count upto 2n pulses.