SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)

B.Tech. - II Year – I Semester

L	T	P	C
3	0	0	3

(R22ECE2112) DIGITAL ELECTRONICS

Course Objectives: This course aims at through understanding of binary number system, logic gates, combination logic and synchronous and asynchronous logic.

UNIT - I:
BOOLEAN ALGEBRA AND LOGIC GATES: Digital Systems, Binary Numbers, Number base conversions, Octal and Hexadecimal Numbers, complements, Signed binary numbers, Binary codes, Binary Storage and Registers, Binary logic.
Basic Definitions, Axiomatic definition of Boolean Algebra, Basic theorems and properties of Boolean algebra, Boolean functions, canonical and standard forms, other logic operations, Digital logic gates.

UNIT - II:
GATE – LEVEL MINIMIZATION: The map method, Four-variable map, Five-Variable map, product of sums simplification Don't-care conditions, NAND and NOR implementation other Two-level implementations, Exclusive – Or function.

UNIT - III:
COMBINATIONAL LOGIC: Combinational Circuits, Analysis procedure Design procedure, Binary Adder-Subtractor Decimal Adder, Binary multiplier, magnitude comparator, Decoders, Encoders, Multiplexers, HDL for combinational circuits.

UNIT - IV:
SEQUENTIAL LOGIC: Sequential circuits, latches, Flip-Flops Analysis of clocked sequential circuits, state Reduction and Assignment, Design Procedure. Registers, shift Registers, Ripple counters, synchronous counters, other counters.

UNIT - V
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC: Introduction, Random-Access Memory, Memory Decoding, Error Detection and correction Read-only memory, Programmable logic Array programmable Array logic, Sequential Programmable Devices.
Introduction, Analysis Procedure, Circuits with Latches, Design Procedure, Reduction of state and Flow Tables, Race-Free state Assignment Hazards, Design Example.

TEXT BOOKS:
1. Digital Design – Third Edition, M. Morris Mano, Pearson Education/PHI.
2. Digital Principles and Applications Albert Paul Malvino Donald P. Leach TATA McGraw Hill Edition.
3. Fundamentals of Logic Design, Roth, 5th Edition, Thomson.

REFERENCE BOOKS:
1. Switching and Finite Automata Theory by Zvi. Kohavi, Tata McGraw Hill.
2. Switching and Logic Design, C.V.S. Rao, Pearson Education
3. Digital Principles and Design – Donald D.Givone, Tata McGraw Hill, Edition.
4. Fundamentals of Digital Logic and Microcomputer Design, 5TH Edition, M. Rafiquzzaman JohnWiley.

DIGITAL ELECTRONICS [R22ECE2112]

UNIT-1 : BOOLEAN ALGEBRA & LOGIC GATES

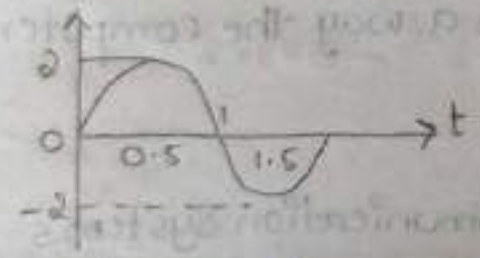
Introduction:

Signal: which carriers information from one place to another place there are two types of signals

1. Analog Signals
2. Digital Signals

1. Analog signals: the signal is present at each interval of time "t"

Ex: Sine signal, triangle signal, cosine signal



at t=0 signal present

Analog systems: The system which process the analog signals are known as analog systems

Ex: Filter circuits, Amplifiers, Signal generators, etc

DrawBacks of Analog Systems:

1. Less Accurate
2. Less Reliabity

3. Temp effect

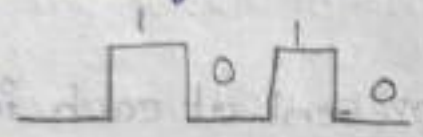
4. Noice

5. No Memory

6. No processor

2. Digital Signals: they obtain only two values 0 and 1 a Signal is known as digital signal it has only a finite number of pre determined discrete magnitude.

→ these are classified into binary, decimal, octal, Hexa decim



Digital Systems: In modern technology the term digital is used associated with a computer.

→ The term digital is derived from a way the computer performed by counting digits.

Ex: LED TV's, Digital watches, communication Systems

Advantages of Digital Systems:

1. Easy to design

2. Reliability

3. Flexibility

4. Functionality

5 programmability

6. Speed

7. Upgrading Technology

NUMBER SYSTEM:

Generally numbers are 0 to 9 these are decimal number system which we use in general terms but our basic system i.e., computers take the language of binary number system that is 0's and 1's.

→ Generally decimal numbers can be represented in binary format but the decimal number increases the representation in binary format becomes very complicated so we go for other number systems like octal $(\ )_8$, Hexa Decimals $(\ )_{16}$, Binary coded decimal (BCD). These number system represent binary number in compressed form.

Decimal Number Representation:

It is represented by 10 digits that is 0 to 9 and its base (or) radix (or) weight is $(\ )_{10}$.

Any digital number can be represented in units 10's, 100's

Ex: $(5,6,78.9)_{10}$ where 10 is radix (or) base find decimal number equivalent for given system.

$$(5\ 6\ 7\ 8\ .\ 9)_{10}$$
$$3\ 2\ 1\ 0\ \ -1$$

$= 5\times10^3 + 6\times10^2 + 7\times10^1 + 8\times10^0 + 9\times10^{-1}$

$= (5678.9)_{10}$

Ex: $(1010.11)_2$
$8\ 2\ 1\ 0\ -1\ -2$

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

$= 8 + 0 + 2 + \frac{1}{2} + \frac{1}{4}$

$= \frac{40+2+1}{4} = \frac{43}{4} = 10.75$

Converting any radix to decimal (power of base)

Ex: 1. convert $(11011.1101)_2$ — $(\ )_{10}$?

$(\ )_2 \rightarrow (\ )_8 \rightarrow (\ )_{10}$
$(\ )_{16}$

$(1\ 1\ 0\ 1\ 1\ .\ 1\ 1\ 0\ 1)_2$
$4\ 3\ 2\ 1\ 0\ \ -1\ -2\ -3\ -4$

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

$= (27.8125)_{10}$

$= (11011.1101)_2 = (27.8125)_{10}$

2. convert $(126.75)_8$ — $(\ )_{10}$?

$(1\ 2\ 6\ .\ 7\ 5)_8$
$2\ 1\ 0\ \ -1\ -2$

$= 1\times8^2 + 2\times8^1 + 6\times8^0 + 7\times8^{-1} + 5\times8^{-2}$

$= 64 + 16 + 0 + 7\times\frac{1}{8} + 5\times\frac{1}{64}$

$= 64 + 16 + 0 + \frac{7}{8} + \frac{5}{64}$

$= (86.953125)_{10}$

3. convert $(AF.39)_{16} - (\ )_{10}$?

A → 10
B → 11
C → 12
D → 13
E → 14
F → 15

$$\underset{1\ 0}{(AF}\cdot\underset{-1\ -2}{B9)_{16}}$$

$= A\times16^1 + F\times16^0 + B\times16^{-1} + 9\times16^{-2}$

$= 10\times16^1 + 15\times1 + 11\times16^{-1} + 9\times16^{-2}$

$= (175.723)_{10}$

4. convert $(3102.12)_4 - (\cdot)_{10}$?

$$\underset{3\ 2\ 1\ 0\ -1\ -2}{(3\ 1\ 0\ 2\cdot 1\ 2)_4}$$

$= 3\times4^3 + 1\times4^2 + 0\times4^1 + 2\times4^0 + 1\times4^{-1} + 2\times4^{-2}$

$= 3\times64 + 16 + 0 + 2 + 1\times\frac{1}{4} + \frac{2}{16_8}$

$= 192 + 16 + 2 + \frac{1}{4} + \frac{1}{8}$

$= (210.375)_{10}$

Ex:

1. $(1010\,10.110)_2 - (\ )_{10}$? $(42.75)_{10}$
2. $(100\,110\,110.1101)_2 - (\ )_{10}$? $(310.81)_{10}$
3. $(156.473)_8 - (\ )_{10}$? $(110.615)_{10}$
4. $(745.312)_8 - (\ )_{10}$? $(485.39)_{10}$
5. $(9C7.8F)_{16} - (\ )_{10}$? $(2503.55)_{10}$
6. $(1BE.A7)_{16} - (\ )_{10}$? $(446.65)_{10}$

1. $(101010.110)_2 - (\ )_{10}$?

$$(1\ 0\ 1\ 0\ 1\ 0 \cdot 1\ 1\ 0)_2$$
$$5\ 4\ 3\ 2\ 1\ 0\ -1\ -2\ -3$$

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

$$= 32 + 0 + 8 + 0 + 2 + 0 + 0.5 + 0.25 + 0$$

$$= (42.75)_{10}$$

2. $(100\ 110110.1101)_2 - (\ )_{10}$?

$$(1\ 0\ 0\ 1\ 1\ 0\ 1\ 1\ 0 \cdot 1\ 1\ 0\ 1)_2$$
$$8\ 7\ 6\ 5\ 4\ 3\ 2\ 1\ 0\ -1\ -2\ -3\ -4$$

$$= 1\times2^8 + 0\times2^7 + 0\times2^6 + 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 1\times2^{-4}$$

$$= 256 + 0 + 0 + 32 + 16 + 0 + 4 + 2 + 0 + 0.5 + 0.25 + 0 + 0.0625$$

$$= (310)\cdot(310.81)_{10}$$

3. $(156.473)_8 - (\ )_{10}$?

$$(1\ 5\ 6 \cdot 4\ 7\ 3)_8$$
$$2\ 1\ 0\ -1\ -2\ -3$$

$$= 1\times8^2 + 5\times8^1 + 6\times8^0 + 4\times8^{-1} + 7\times8^{-2} + 3\times8^{-3}$$

$$= 64 + 40 + 6 + \frac{4}{8} + \frac{7}{64} + \frac{3}{192}$$

$$= (110.625)_{10}$$

4. $(745.312)_8 - (\ )_{10}$?

$$(745.312)_8$$
$$2\ 1\ 0\ -1\ -2\ -3$$

$= 7\times8^2 + 4\times8^1 + 5\times8^0 + 3\times8^{-1} + 1\times8^{-2} + 2\times8^{-3}$

$= 448 + 32 + 5 + \frac{3}{8} + \frac{1}{64} + \frac{2}{192}$

$= (485.39)_{10}$

5. $(9C7.8F)_{16} - (\ )_{10}$?

$$(9\ C\ 7.8\ F)_{16}$$
$$2\ 1\ 0\ -1\ -2$$

$= 9\times16^2 + C\times16^1 + 7\times16^0 + 8\times16^{-1} + F\times16^{-2}$

$= 9\times16^2 + 12\times16^1 + 7\times16^0 + 8\times16^{-1} + 15\times16^{-2}$

$= 2304 + 192 + 7 + \frac{8}{16} + \frac{15}{256}$

$= (2503.55)_{10}$

6. $(1BE.A7)_{16} - (\ )_{10}$?

$$(1\ B\ E.A\ 7)_{16}$$
$$2\ 1\ 0\ -1\ -2$$

$= 1\times16^2 + B\times16^1 + E\times16^0 + A\times16^{-1} + 7\times16^{-2}$

$= 1\times16^2 + 11\times16^1 + 14\times16^0 + 10\times16^{-1} + 7\times16^{-2}$

$= 256 + 176 + 14 + \frac{10}{16} + \frac{7}{256}$

$= (446.65)_{10}$

Relation between decimal, binary, Hexadecimal, octal:

Decimal	Binary	octal	Hexadecimal
	8421		
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
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

Conversion of Decimal Number to any radix number:

These conversion of decimal number to any radix number involves 2 steps:

(i) we have to convert integer part by using Successive division method

(ii) we have to convert fractional part by using Successive multiplication method.

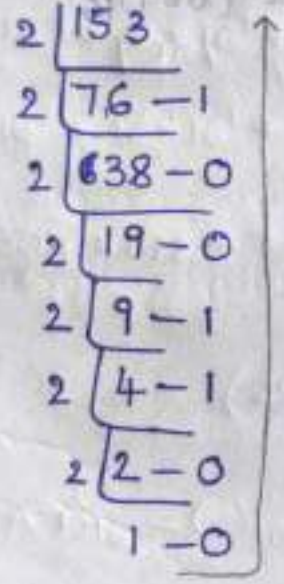
$( )_{10}$ → $( )_2$, $( )_8$, $( )_{16}$

ii) Successive Division Method For Integer part: (~~Integ~~

Ex: 1. convert $(41)_{10} - (\ )_2$?

$(41)_{10} = (101001)_2$

2. convert $(153)_{10} - (\ )_2$?



$(153)_{10} = (10011001)_2$

3. convert $(8532)_{10} - (\ )_8$?

$(8532)_{10} = (20524)_8$

4. $(1236)_{10} - (\ )_{16}$?

16 | 1236
16 | 77 — 4
4 — 13

$(1236)_{10} = (4D4)_{16}$

5. convert $(1625)_{10} - (\ )_{16}$?

16 | 1625
16 | 101 — 9
6 — 5

$(1625)_{10} = (659)_{16}$

Ex:

1. $(75321)_{10} - (\ )_8$? $(223071)_8$
2. $(214)_{10} - (\ )_8$? $(326)_8$
3. $(198)_{10} - (\ )_2$?
4. $(895)_{10} - (\ )_2$?

① $(895)_{10} - (\ )_2$?

2 | 895
2 | 447 — 1
2 | 223 — 1
2 | 111 — 1
2 | 55 — 1
2 | 27 — 1
2 | 13 — 1
2 | 6 — 1
2 | 3 — 0
1 — 1

$(895)_{10} = (1101111111)_2$

② $(198)_{10} - (\ )_2$?

2 | 198
2 | 99 - 0
2 | 49 - 1
2 | 24 - 1
2 | 12 - 0
2 | 6 - 0
2 | 3 - 0
2 | 1 - 1

$(198)_{10} = (11000110)_2$

③ $(214)_{10} - (\ )_8$?

8 | 214
8 | 26 - 6
3 - 2

$(214)_{10} = (326)_8$

④ $(75321)_{10} - (\ )_8$?

8 | 75321
8 | 9415 - 1
8 | 1176 - 7
8 | 147 - 0
8 | 18 - 3
8 | 2 - 2

$(75321)_{10} = (223071)_8$

(ii) Successive multiplication method (fractional part):

i) multiply the fractional part of decimal number by desired base number.

ii) Record the integer part of product as carry and fractional part as new fractional part.

iii) Repeat steps 1 & 2 until fractional part of product becomes to zero (or) until you have many digits as necessary for your application

Ex: convert $(12.125)_{10} - (\ )_2$?

$(12.125)_{10}$

12 → Integer → Successive division method

0.125 → fractional part → multiplication method

$$\begin{array}{r|l} 2 & 12 \\ 2 & 6 - 0 \\ 2 & 3 - 0 \\ & 1 - 1 \end{array} \uparrow$$

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

$(0.125)_{10} = (001)_2$

$0.125 \times 2 = 0.25 \rightarrow 0.25$ with carry 0

$0.25 \times 2 = 0.5 \rightarrow 0.5$ with carry 0

$0.5 \times 2 = 1.0 \rightarrow 0.0$ with carry 1 ↓

fractional 0 stop

$(12.125)_{10} = (1100.001)_2$

Ex: convert $(24.6)_{10} - (\ )_2$?

24 → Integer

0.6 → fractional part

2	24	
2	12	-0
2	6	-0
2	3	-0
	1	-1

$(24)_{10} = (11000)_2$

$0.6 \times 2 = 1.2 \rightarrow 0.2$ with carry 1

$0.2 \times 2 = 0.4 \rightarrow 0.4$ with carry 0

$0.4 \times 2 = 0.8 \rightarrow 0.8$ with carry 0

$0.8 \times 2 = 1.6 \rightarrow 0.6$ with carry 1

$0.6 \times 2 = 1.2 \rightarrow 0.2$ with carry 1 → repeated stop

$(0.6)_{10} = (10011)_2$

$(24.6)_{10} = (11000.10011)_2$

Ex: convert $(0.8125)_{10} - (\ )_2$?

0 → Integer

8125 → fractional part

$0.8125 \times 2 = 1.625 \rightarrow 0.625$ with carry 1

$0.625 \times 2 = 1.25 \rightarrow 0.25$ with carry 1

$0.25 \times 2 = 0.5 \rightarrow 0.5$ with carry 0

$0.5 \times 2 = 1.0 \rightarrow 0.0$ with carry 1

$(0.8125)_{10} = (1101)_2$

Ex: convert $(35.45)_{10} - (\ )_8$?

35 → Integer

45 → fractional part

8 | 35
4 – 3 ↑

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

$0.45 \times 8 = 3.6 \rightarrow 0.6$ with carry 3

$0.6 \times 8 = 4.8 \rightarrow 0.8$ with carry 4

$0.8 \times 8 = 6.4 \rightarrow 0.4$ with carry 6

$0.4 \times 8 = 3.2 \rightarrow 0.2$ with carry 3

$0.2 \times 8 = 1.6 \rightarrow 0.6$ with carry 1

$(0.45)_{10} = (34631)_8$

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

Ex: convert $(52.65)_{10} - (\ )_8$?

52 → Integer

0.65 → fractional part

8 | 52
6 – 4 ↑

$(52)_{10} = (64)_8$

$0.65 \times 8 = 5.2 \rightarrow 0.2$ with carry 5

$0.2 \times 8 = 1.6 \rightarrow 0.6$ with carry 1

$0.6 \times 8 = 4.8 \rightarrow 0.8$ with carry 4

$0.8 \times 8 = 6.4 \rightarrow 0.4$ with carry 6

$0.4 \times 8 = 3.2 \rightarrow 0.2$ with carry 3 (repeated)

$(0.65)_{10} = (51463)_8$

$(52.65)_{10} = (64.51463)_8$

Ex: convert $(22.64)_{10} - (\ )_{16}$?

22 → Integer

0.64 → Fractional part

$16 \mid 22$
$1 - 6$

$(22)_{10} = (16)_{16}$

$0.64 \times 16 = 10.24 \rightarrow 0.24$ with carry A

$0.24 \times 16 = 3.84 \rightarrow 0.84$ with carry 3

$0.84 \times 16 = 13.44 \rightarrow 0.44$ with carry D

$0.44 \times 16 = 7.04 \rightarrow 0.04$ with carry 7

$0.04 \times 16 = 0.64 \rightarrow 0.64$ with carry 0

$0.64 \times 16 = 10.24 \rightarrow 0.24$ with carry A (repeated)

$(0.64)_{10} = (A3D70A)_{16}$

$(22.64)_{10} = (16.A3D70A)_{16}$

Number Base Conversion:

1. Binary to octal conversion $[(\ )_2 - (\ )_8]$:

→ Binary for octal number is 8 and base for binary is 2 the base for octal is the third power of binary that is 2^3.

→ So by grouping three digits of binary numbers and then converting each group to digit to octal equivalent.

Ex: convert $(10101101.0111)_2$ to octal $(\ )_8$ equivalent?

$$\overset{2}{1}\overset{1}{0}\,|\,\overset{4}{1}\overset{2}{0}\overset{1}{1}\,|\,\overset{4}{1}\overset{2}{0}\overset{1}{1}\,.\,\overset{4}{0}\overset{2}{1}\overset{1}{1}\,|\,\overset{4}{1}$$

$$(2\ 5\ \ 5\cdot 3\ 4)_8$$

convert $(01001000.1101)_2$ to $(\ )_8$?

$$\overset{1}{0}\,|\,\overset{4}{1}\overset{2}{0}\overset{1}{0}\,|\,\overset{4}{1}\overset{2}{0}\overset{1}{0}\,.\,\overset{4}{1}\overset{2}{1}\overset{1}{0}\,|\,\overset{4}{1}$$

$$(0\ 4\ 4\cdot 6\ 4)_8$$

convert $(000111101.0101)_2 - (\ )_8$?

$$\overset{4}{0}\overset{2}{0}\overset{1}{0}\,|\,\overset{4}{1}\overset{2}{1}\overset{1}{1}\,|\,\overset{4}{1}\overset{2}{0}\overset{1}{1}\,.\,\overset{4}{0}\overset{2}{1}\overset{1}{0}\,|\,\overset{4}{1}$$

$$(0\ 75\cdot 24)_8$$

convert $(10110011.110)_2 - (\ )_8$?

$$\overset{2}{1}\overset{1}{0}\,|\,\overset{4}{1}\overset{2}{1}\overset{1}{0}\,|\,\overset{4}{0}\overset{2}{1}\overset{1}{1}\,.\,\overset{4}{1}\overset{2}{1}\overset{1}{0}$$

$$(263\cdot 6)_8$$

2. Octal to Binary conversion:

→ Octal to conversion is a reverse operation of binary to octal

→ each digit of octal number is individually converted to its binary equavalent to get octal to binary.

Ex: convert $(125.62)_8$ to $(\ )_2$?

1	2	5	.	6	2
↓	↓	↓		↓	↓
001	010	101		110	010

4 2 1

$(001010101.110010)_2$

convert $(623.76)_8$ to $(\ )_2$?

6	2	3	.	7	6
↓	↓	↓		↓	↓
110	010	011		111	110

$(110010011.111110)_2$

convert $(534.67)_8 - (\ )_2$?

5	3	4	.	6	7
↓	↓	↓		↓	↓
101	011	100		110	111

$(101011100.110111)_2$

3. Binary to Hexadecimal conversion:

→ Base for Hexadecimal is 16 and base for binary is 2 base of Hexadecimal number is fourth power of binary that is 24.

→ Grouping four digit of binary number and then converting each group digit to hexa decimal equivalent

Ex: convert $(1101101110.1001101)_2 - (\ )_{16}$?

2 1 | 8 4 2 1 | 8 4 2 1 . 8 4 2 1 | 8 4 2
1 1 | 0 1 1 0 | 1 1 1 0 . 1 0 0 1 | 1 0 1

$(3\,6E.9A)_{16}$

convert $(1101100010011011)_2 - (\ )_{16}$?

8 4 2 1 | 8 4 2 1 | 8 4 2 1 | 8 4 2 1
1 1 0 1 | 1 0 0 0 | 1 0 0 1 | 1 0 1 1

$(D89B)_{16}$

convert $(11001101.110110) - (\ )_{16}$?

8 4 2 1 | 8 4 2 1 . 8 4 2 1 | 8 4
1 1 0 0 | 1 1 0 1 . 1 1 0 1 | 1 0

$(CD.D8)_{16}$

4. Hexadecimal to Binary conversion:

→ Hexadecimal to binary conversion is a reversal of binary to Hexadecimal conversion

→ write equivalent of four bit binary number of each hexadecimal digit

Ex: convert $(8A9.B4)_{16} - (\ )_2$?

8 A 9 . B 4
↓ ↓ ↓ ↓ ↓
1000 1001 1001 1011 0100

$(10001010\,1001 \cdot 10110100)_2$

convert $(5DE \cdot AB)_{16} - (\ )_2$?

5 D E · A B
↓ ↓ ↓ ↓ ↓
0101 1101 1110 1010 1011

$(0101\,1101\,1110 \cdot 1010\,1011)_2$

convert $(FC\,7 \cdot 9A\,D)_{16} - (\ )_2$?

F C 7 · 9 A D
↓ ↓ ↓ ↓ ↓ ↓
1111 1100 0111 1001 1010 1101

$(1111\,1100\,0111 \cdot 1001\,1010\,1101)_2$

5. Octal to Hexadecimal conversion:

→ convert octal number to its binary equivalent that is 3 bits.

→ convert binary to its hexadecimal equivalent that is

Ex: convert $(615 \cdot 25)_8 - (\ )_{16}$?

6 1 5 · 2 5
↓ ↓ ↓ ↓ ↓
110 001 101 · 010 101

$(110\,0011\,01 \cdot 010\,101)_2$

8 4 2 1 | 8 4 2 1 | 8 4 2 1 | 8 4
1 | 1 0 0 0 | 1 1 0 1 · 0 1 0 1 | 0 1

$(1\ 8\ D \cdot 5\ 4)_{16}$

convert $(645.32)_8 - (\ )_{16}$?

6 4 5 . 3 2

↓ ↓ ↓ ↓ ↓

110 100 101 011 010

$(110100101011010)_2$

1 8 4 2 1 8 4 2 1 8 4 2 1 8 4

1 1 0 1 0 0 1 0 1 . 0 1 1 0 1 0

$(1A5.68)_{16}$

convert $(532.461)_8 - (\ )_{16}$?

5 3 2 . 4 6 1

↓ ↓ ↓ ↓ ↓ ↓

101 011 010 100 110 001

$(101011010.100110001)_2$

1 8 4 2 1 8 4 2 1 8 4 2 1 8 4 2 1

1 0 1 0 1 1 0 1 0 . 1 0 0 1 1 0 0 0 1

$(15A.988)_{16}$

6. Hexadecimal to octal conversion:

→ write equivalent 4 bit binary number for each hexadecimal

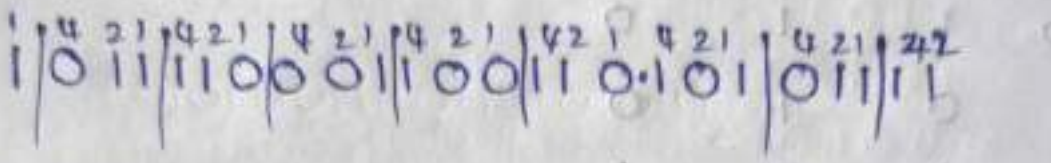
→ Group of 3 bits from LSB for integer MSB for fractional part by adding 0's at end if required.

Ex: convert $(BC66.AF)_{16} - (\ )_8$?

B C 6 6 · A F

1011 1100 0110 0110 1010 1111

$(10111100011001 10 \cdot 10101111)_2$



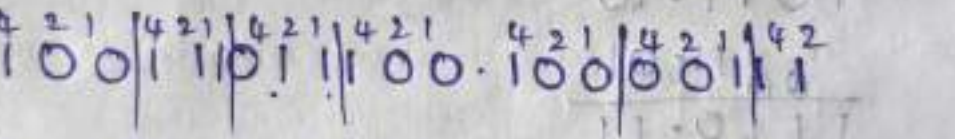
$(136146.536)_8$

convert $(9EC \cdot 87)_{16} - (\ )_8$?

9 E C · 8 7

1001 1110 1100 1000 0111

$(100111011100 \cdot 10000111)_2$



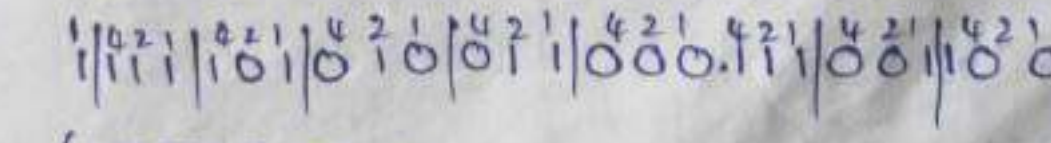
$(4754 \cdot 416)_8$.

convert $(FA98 \cdot E65)_{16} - (\ )_8$?

F A 9 8 · E 6 5

1111 1010 1001 1000 1110 0110 0101

$(1111101010011000 1110011001 0101)_2$



$(175230 \cdot 7145)_8$

Binary Arithemetic:

Rules for Binary Addition:

A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

1. Add the binary numbers $(1010)_2$ & $(0011)_2$

$$\begin{array}{r} 1010 \rightarrow 10 \\ 0011 \rightarrow 3 \\ \hline 1101 \quad 13 \end{array}$$

2. Add $1011.10 + 11.01$

$$\begin{array}{r} 1011.10 \\ 11.01 \\ \hline 1110.11 \end{array}$$

3. Add 28 & 15 in binary.

$$\begin{array}{r} \begin{matrix} 16 & 8 & 4 & 2 & 1 \end{matrix} \\ 28 \rightarrow 1\ 1\ 1\ 0\ 0 \\ 15 \rightarrow 0\ 1\ 1\ 1\ 1 \\ \hline 43 \quad 1\ 0\ 1\ 0\ 1\ 1 \end{array}$$

4. Add 368 & 22

5. 512 & 963

6. 47 & 52

Subtraction:

Rules for Subtraction:

A	B	Subtract A-B	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

1. Subtract $(0101)_2$ from $(1011)_2$

$1011 \rightarrow 11$

$0101 \rightarrow 5$

$0110 \rightarrow 6$

2. Subtract $(111.111)_2$ from $(1010.01)_2$

```
  10 10.010
  0111.111
  ---------
  0010.011
```

3. Sub (10110) from (11011)

```
  11011
  10110
  -----
  00101
```

Rules for Multiplication:

A	B	product
0	0	0
0	1	0
1	0	0
1	1	1

1. Multiply $(1101)_2$ by $(110)_2$

```
  1101 → 13
  x110 → 6
  ----   --
  0000   78
 1101x
1101xx
-------
1001110
-------
```

2. Multiply $(1110)_2$ by $(1010)_2$

```
  1110 → 14
  1010 → 10
  ----
  0000
 1110x
0000xx
1110xxx
-------
10001100
-------
```

3. Multiply

$$
\begin{array}{r}
1011.101 \\
\times\ 101.01 \\
\hline
1011101 \\
0000000\times \\
101110 1\times\times \\
000000 0\times\times \\
10111 01\times\times\times \\
\hline
111101.00001
\end{array}
$$

Binary Division:

$0 \div 1 = 0$

$1 \div 1 = 1$

1. divide $(1101101 1)_2 \div 110$

$$
\begin{array}{r|l}
110 & 11011011 \\
\hline
 & 10010.1 \\
\hline
110\downarrow\downarrow\downarrow & \\
\hline
000110 & \\
110\downarrow\downarrow & \\
\hline
000110 & \\
110 & \\
\hline
000 &
\end{array}
$$

2. divide $1001110 \div 100$

$$
\begin{array}{r|l}
100 & 1001110 \\
\hline
 & 1001.1 \\
\hline
100\downarrow\downarrow\downarrow & \\
\hline
000111 & \\
100\downarrow & \\
\hline
0110 & \\
100 & \\
\hline
0100 & \\
100 & \\
\hline
00 &
\end{array}
$$

r's compliment and (r-1)'s compliment:

2's, 8's, 10's, 16's r's compliment (M-N) operation	1's, 7's, 9's, 15's (r-1)'s compliment (M-N) operation
1. r's complement of N	1. (r-1)'s complement of N
2. M+r's complement of N	2. M+(r-1)'s complement of N
3. If carry is generated then the result is +ve, ignore the carry.	3. If carry is generated then the result is +ve & add carry to the result
4. If carry is not generated then the result is -ve and r's complement is result	4. If carry is not generated then the result is -ve & (r-1)'s complement is result

1's compliment method Subtraction:

1. perform 5-3 using 1's complement method?

M-N
5-3

Find the (r-1)'s complement of N i.e 1's complement of N

3 → 011 → N complement

1) 1's complement of N → 100

2) M+(r-1)'s complement of N

5 → M → 101

1's comple of N → 100

```
 101
 100
-----
 001
```

If carry is generated then the result is +ve & Add carry to the result

```
  0 0 1
    +1
  ------
1) 0 1 0   = +2
```

5-3=2

perform 12-8 using 1's complement method?

12 - 8

M N

Find the 1's complement of N

N → 8 → 1000

1's complement of N → 0111

M + 1's complement of N

M → 12 → 1100

1's comple of N → 0111

```
  1 1 0 0
  0 1 1 1
  -------
1) 0 0 1 1
```

If carry is generated then the result is +ve & Add carry to the result

```
  0 0 1 1
  ① ①1
  -------
  0 1 0 0  = +4
```

12-8 = 4

perform 47-43 using 1's complement method?

47 - 43
M N

Find the 1's complement of N

N → 43 → 101 011

1's complement of N → 010100

M+1's complement

M → 47 → 101 111

1's complement of N → 010100

If carry is generated then the result is +ve and add carry to the result

```
  101111
  010100
  (1)(1)(1)
 1)000011
```

```
  000011
    (1)(1)1
 ------------
  000100 = +4
```

47 - 43 = 4

perform 15 - 28 using 1's complement method?

15 - 28
M N

Find the 1's complement of N

N → 28 → 11100

M → 15 → 01111

1's complement of N → 00011

M+1's complement

```
  01111
  00011
 ------
  10010
```

If carry is not generated then the result is -ve and (r-1)'s complement is result

10010 → 01101

01101 → -13

15 - 28 = -13

36-42 using 1's complement method?

M - N

36 - 42

M → 36 → 100100

N → 42 → 101010

Find the 1's complement of N

N → 42 → 010101

M + 1's complement 100100
010101
111001

If carry is not generated the result is -ve and (r-1)'s complement is result

111001 → 000110 → -6

36 - 42 = -6

2's complement method subtraction:

Find the 2's complement of 12.

12 → 1100

1's complement → 0011

2's complement → 0011 + 1
0100

Find the 2's complement of 15 - 28?

M → 15 → 1111

N → 28 → 11100

1's complement → 00011

2's complement → +1 → 00100

M + 2's → 01111 + 00100 = 10011

If don't have carry then the result is -ve and (r-1)'s complement is result

1's complement of result 01100

2's comple of result + 1

01101

15 - 28 = -13

Find ~~12-8~~ 2's ~~complement~~ method?

M - N

12 - 8

Subtracte $(101011)_2$ from $(111001)_2$ by using 1's complement and 2's complement

12 → N

43 - 57

M - N

Find 1's complement of N

N → 111001

1's complement → 000110

M+1's complement

M → 101011
000110
000
110001

If carry is not generated then the result -ve & find 1's complement of result

1's complement of result → 001110 = -14

43-57=-14

Using 2's complement:

Find 2's complement of N

N → 111001

1's complement → 000110

2's complement → 1
000111

M+2's complement of N

M → 101011
2's → 000111
000
011010

If carry is not generated then the result is -ve find 2's complement of result

11 00 10 → result
00 11 01 → 1's complement of result
01 → 2's complement of result
00 11 10
=-14

36-42 using 2's complement method?

M → 36 → 100100

N → 42 → 101010

Find 2's complement of N

N → 42 → 101010

1's complement → 010101

2's complement → $\overset{\textcircled{1}\,1}{}$
010110

M + 2's complement of N

M → 100100
010110
111010

If carry is not generated then the result is -ve Find 2's complement of result

1's complement of result → 000101

2's complement of result → $\overset{\textcircled{1}\,1}{}$
000110 = -6

36-42 = -6

9's and 10's complement method of subtraction:

9's complement:

It is obtained by subtracting each decimal number from 9

Ex: Find the 9's complement of 346

999
346
653 → 9's complement

10's complement:

It is obtained by adding '1' to its 9's complement

Ex: 10's complement of 346

```
  999
  346
  653  → 9's complement
    1
  ---
  654  → 10's complement
```

9's complement of subtraction:

Subtract 72532 − 3250 by using 9's complement?

M N

Find 9's complement of N

```
  99999
N→ 03250
  -----
  96749  → 9's complement
```

M+9's complement of N

```
M→ 72532
   96749
  -------
  1)69281
```

If carry is generated then the result is +ve and Add carry to result

```
  69281
     +1
  -----
+ 69282
```

Subtract 3250 − 72532 by using 10's complement

Find 10's complement

N → 99999
72532
27467 → 9's complement
+1
27468 → 10's complement

M+10's complement

M → 03250
27468
30718

If carry is not generated then the result is -ve find 10's complement of result

99999
30718
69281 → 9's
+1
69282 → 10's

7's and 8's complement method of subtraction:

7's complement:

Find 7's complement by ~~each~~ subtract each octal number from 7.

8's complement:

By adding '1' to its 7's complement.

Octal Arithematic Rules:

→ Sum of two octal digits is 8 or > 8, then Subtraction 8 to obtain the octal digit.

→ A carry '1' is provided

Ex: Add 472 and 577

```
  4 7 2
  5 7 7
① ① ① ①
-------
(10-8)(15-8)(9-8)
  2   7   1
```

Subtract $(413)_7 - (516)_7$ by using 7's complement?

$M \rightarrow (413)_7$ $N \rightarrow (516)_7$

7's complement of N

```
  7 7 7
N→5 1 6
-------
  2 6 1
```

M+7's complement of N

```
M→ 4 1 3
   2 6 1
---------
   6 7 4
```

If the carry is not generated then the result is -ve and 7's complement is result

```
  7 7 7
  6 7 4
- 1 0 3
-------
```

8's complement of subtraction:

Ex: Subtract $(516)_8 - (413)_8$ by using 8's complement.

8's complement of N

```
   7 7 7
N→ 4 1 3
   -----
   3 6 4 → 7's
     + 1
   -----
   3 6 5 → 8's
```

M+8's complement of N

```
M → 5 1 6
    3 6 5
   ---------------------
1) (9-8)(8-8)(11-8)
 ↓   1    0    3
```

If carry is generated then the result is +ve, and ignore the carry

=+103

Hexadecimal Arithematic rules for 15's and 16's complement:

15's complement:

To find 15's complement by sub each hexadecimal no. form 15

16's complement:

By adding 1 to 15's complement.

→ If sum of two Hexadecimal digits is 16 (or) greater than 16, subtract 16 to obtain the hexadecimal digit

→ If sum ≥ 16 subtract 16 with carry 1

→ If sum ≥ 32 subtract 32 with carry 2

(B02)16 − (98F)16 by using 15's comp

```
  15 15 15
   9  8  F
  --------
   6  7  0
   B  0  2
  --------
1) (17-16) 7 2
    1  7  2
     +  1
```

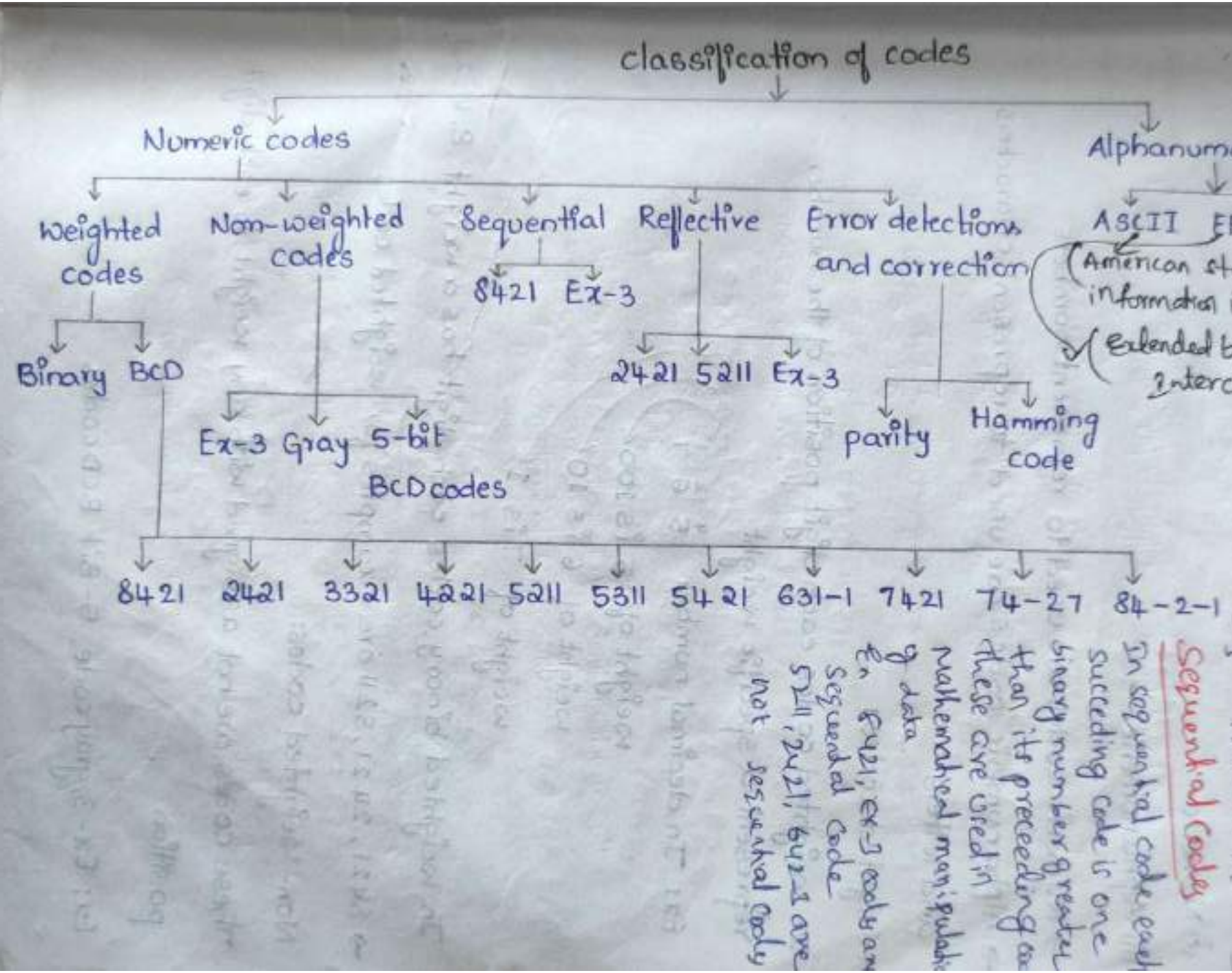
```
 (98F)
  9 8 F
B (1) 7 3
 (16-16)(18-16)
B  0  2
```

CB2 − 972 16's

```
 15 15 15
  9  7  2
 --------
  6  8 13
  6  8  E
  6  8  E (+1)
  C  B  2
 ------------
 (19-16)(20-16)(16-16)
1)  3     4     0
```

② 698 − C14 → 15's [−579]

③ 387 − 854 → 16's. [−490]



Binary Codes:

(American standard code for information interchange)

(Extended binary coded decimal interchange)

Reflective code

A code is said to be reflective when the code for 9 is the complement for 0, 8 for 1, 7 for 2, 6 for

Ex 2421, 5211, Ex-3

9 → 1100

Ex-3 → 0 → 0011 complement

1100 → 9

Sequential Codes

In sequential code each succeeding code is one binary number greater than its preceding code these are used in mathematical manipulation of data

Ex, 8421, Ex-3 codes are Sequential Code

5211, 2421, 6421 are not sequential codes

→ Data is transmitted and represented in a group of bits called binary code in digital transmission.

→ the binary code may be represent numbers, alphabets & special characters and control functions

→ codes are divided into 2 types:

(i) Numeric

(ii) Alphanumeric

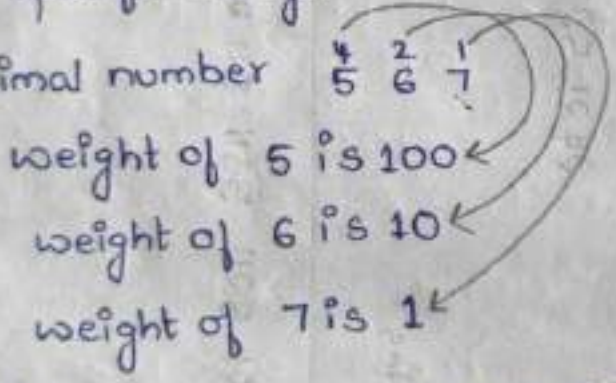
→ Numeric codes are used to represent numbers

→ Alphanumeric codes are used to represent characters and letters

Weighted codes:

In weighted codes each digit position of the number represent a specific weight.

Ex: In decimal number $\overset{4}{5}\ \overset{2}{6}\ \overset{1}{7}$



In weighted binary codes each digit has a weight 8, 4, 2, 1

→ 8421, 2421, 5211 are examples for weighted codes

Non-weighted codes:

→ these codes are not assigned with any weight to each digit position

Ex: Ex-3, gray code, 5-Bit BCD codes.

Binary coded decimal (BCD):

BCD is a numeric code in which each digit of a decimal number is represented by a seperate group of bits

→ most common BCD is 8-4-2-1 in which weights are 8421 from left to right

Advantage of BCD:

→ It is easy to convert between decimal and BCD

Disadvantage:

→ BCD is less efficient than binary

→ BCD addition is little more complicated

Ex: convert the decimal number 175 to BCD

1	7	5
↓	↓	↓
0001	0111	0101

BCD arithemetic Rules:

We add two BCD numbers A & B.

1. Sum ≤ 9, carry = 0 → Answer is correct, No correction is required

2. Sum ≤ 9, carry = 1 → we add '6' to the Sum to get the correct answer

3. Sum > 9, carry = 0 → we add '6' to the Sum to get correct answer

Ex: add 7 & 6 in BCD?

7 6

0111 0110

7 → 0111

6 → 0110

1101 → Sum > 9, C = 0

we add '6'

1101

0110

1) 0011

0001 0011

1 3

Add 57 & 26 in BCD?

57 → 0101 0111

26 → 0010 0110

83

S ≤ 9, C = 0 ← 0111 1101 → S > 9, C = 0

answer is correct

0110 add '6'

1000 0011

8 3

Add 83 & 34 in BCD?

83 → 1000 0011

34 → 0011 0100

11 7

S > 9, C = 1 ← 1011 0111 → S ≤ 9, C = 0

we add '6'

0110

1) 0001 0111

0001 0001 0111

1 1 7

Add 7484 + 3668 in BCD

7484 → 0111 0100 1000 0100

3668 → 0011 0110 0110 1000

1010 1010 1110 1100

S>9, C=0

we add '6'

0110 0110 0110 0110

1) 0001 0001 0101 0010

1 1 5 2

0001

1

Add 8254 + 8277

8254 → 1000 0010 0101 0100

8277 → 1000 0010 0111 0111

16531

1) 0000 0100 1100 1011

0110 0110 0110

0001 0110 0101 0011 0001

1 6 5 3 1

BCD subtraction using 9's compliment:

→ find the 9's compliment of negative number

→ add 2 numbers using BCD addition

→ if carry is generated then the result is positive add carry to result

→ if carry is not generated result is negative and find 9's compliment of the result

Ex: Sub 3 from 7 in BCD by using 9's compliment

M - N

7 - 3

9's compliment of -ve no $\begin{array}{r} 9 \\ 3 \\ \hline 6 \end{array}$

7 → 0111

6 → 0110

$\begin{array}{r} 0110 \\ \hline 1101 \end{array}$ S>9, C=0

$\begin{array}{r} 0110 \\ \hline 0011 \end{array}$ add '6'

If carry is generated result is +ve add carry to result

$\begin{array}{r} 0011 \\ +\ 1 \\ \hline 0100 \end{array}$ → +4

perform $(83)_{10} - (21)_{10}$ by using BCD 9's complement?

M - N

83 21

9's complement of -ve no $\begin{array}{r} 99 \\ 21 \\ \hline 78 \end{array}$

M+N's compliment of N

83 → 1000 0011

78 → 0111 1000

$\begin{array}{r} 1111 \\ 0110 \\ \hline 0110 \end{array}$ S>9, C=0 we add '6'

$\begin{array}{r} 1011 \\ 0110 \\ \hline 0001 \end{array}$ S>9, C=0 we add '6'

add → carry to result

$\begin{array}{r} 0110 \\ +\ 1 \\ \hline 0110 \end{array}$ $\begin{array}{r} 0001 \\ +\ 1 \\ \hline 0010 \end{array}$

6 2 → +62

perform 52 - 89

M − N

52 89

9's compliment of −ve no $\begin{array}{r} 99 \\ 89 \\ \hline 10 \end{array}$

M+9's compliment of N

52 → 0101 0010

10 → 0001 0000

(1)

$S \leq 9, C=0$ 0110 0010 $S \leq 9, C=0$

6 2

If carry is not generated result is −ve

find its 9's complement

$\begin{array}{r} 99 \\ 62 \\ \hline -37 \end{array}$

BCD Subtraction by using 10's compliment:

→ find 10's compliment of N

→ m+10's compliment of N

→ add 2 BCD numbers

→ If carry is generated result is positive and ignore the carry

→ If carry is not generated result is negative and find 10's compliment of the result

Ex: 9 from 4 by using 10's compliment

M N

$\begin{array}{r} 9 \\ 4 \\ \hline 5 \\ +1 \\ \hline 6 \end{array}$

M + 10's compliment of N

9 → 1001

6 → 0110

1111 → S>9, C=0

we add '6'

0110

If carry generate → 1) 0101 → +5

result is +ve and

ignore the carry

Subtract 54-22 in BCD using 10's compliment:

M - N

99

22

77 → 9's

1

78 → 10's

M + 10's compliments of N

54 → 0101 . 0100

78 → 1111 1000

1100 1100

S>9, C=0 add '6' 0110 0110

S>9, C=0 add 6

1) 0011 0010

3 2

if carry generated result is +ve ignore the carry

Subtract 22-54

M - N

99

54

45

1

46

22 → 0010 0010

46 → 0100 0110

(1)(1)

0110 1000 $S \leq 9, C=0$

$S \leq 9, C=0$ 6 8

carry not generated result is negative

find 10's complement

99

68

31

+1

-32

Excess-3 code:

→ Excess-3 code is a modified form of a BCD number

→ Excess-3 code can be derived from the natural BCD code by adding 3 to each coded number

Excess-3 addition rule: perform Ex-3 addition

Add two Ex-3 numbers

→ if carry = 1 add 3 to the sum of two digits

→ if carry = 0 Subtract 3

Ex: add 8+6 by using Ex-3 addition

8+3=11 → 1011

6+3=9 → 1001

(1)(1)

1) 0100

↓

0001 0100

0011 0011

0100 0111

4 7

Add 16+29 using Ex-3

16	29
33	33
49	5 12

49 → 0100 1001

5 12 → 0101 1100

	1010	0101	$8 \nleq 9, c=01$
$8 \leq 9, c=0$			add '3'
sub 3	0011	0011	
	0111	1000	
	7	8	

Add 103+287 in Ex-3

103	287
333	383
436	51110

436 → 0100 0011 0110

51110 → 0101 1011 1010

	1001	1111	0000	$8 > 9, c=1$
$8 \leq 9, c=0$				add '3'
sub 3	0011	0011	0011	
	0110	1100	0011	
	6	12	3	

Excess-3 Subtraction:

→ complement of N

→ M+N is complement of N

→ If carry=1 result is +ve add 3 & Add end arround carry to the result

→ If carry=0 result is -ve sub 3 and compliment of the result.

Ex: Sub 5-8 by using Ex-3 Subtraction.

5+3=8

8+3=11

complement of N

11 → 1011

0100 → complement of N

M+N's complement of N

8 → 1000

compl of N → 0100

1100

0011 c=0, sub 3

-1001

complementing the result

-0110 → -6

Sub 21-12

21+33=54

12+33=45

complement of N

45 → 0100 0101

1011 1010 → comple of N

M+N's complement of N

54 → 0101 0100

compof N → 1011 1010

c=0, sub 3

1) 0000 1110

c=1, add '3', Add end around carry to the result.

0011 0011

0011 1011

0011 1100

21-12=

21
-12
09
11
9 12

0011 1100.

Sub 645 - 319

645 + 333 = 978

319 + 333 = 6 4 12

complement of N

64 12 → 0110 0100 1100
1001 1011 0011

M+N's complement of N

978 → 1001 0111 1000
1001 1011 0011

1) 0011 0010 1011 C=0 sub 3
0011 0011 0011

0110 0101 1000
0001

0110 0101 1001
6 5 9

C=1
add '3'

Gray code:

In gray code each code difference in one bit position so it is called as unique distance code

→ these codes are also called as cyclic codes and reflective codes.

Decimal	1bit	2bit	3bit	4bit	5bit binary
0	0	00	000	0000	0000
1	1	01	001	0001	0001
2		11	011	0011	0010
3		10	010	0010	0011
4			110	0110	0100
5			111	0111	0101
6			101	0101	0110
7			100	0100	0111
8				1100	1000
9				1101	1001
10					1010
11					1011
12					1100
13					1101
14					1110
15					1111

Binary to gray conversion:

consider Binary = MSB LSB

$B_4\ B_3\ B_2\ B_1$

Gray = $G_4\ G_3\ G_2\ G_1$

$G_4 = B_4$; $G_3 = B_4 \oplus B_3$; $G_2 = B_3 \oplus B_2$; $G_1 = B_2 \oplus B_1$

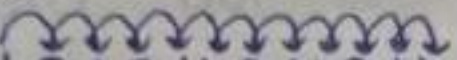
Ex-OR

0 0 0
0 1 1
1 0 1
1 1 0

convert Binary 1010110111 to gray code

1 0 1 0 1 1 0 1 1
1 1 1 1 1 0 1 1 0

convert Binary 10101101101 1 to gray code



1 0 1 0 1 1 0 1 1 0 1 1
1 1 1 1 1 0 1 1 0 1 1 0

Gray to Binary conversion:

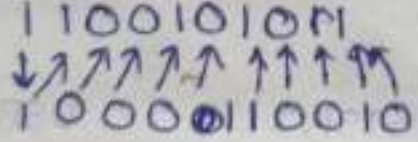
Gray = MSB LSB

$G_4\ G_3\ G_2\ G_1$

Binary = $B_4\ B_3\ B_2\ B_1$

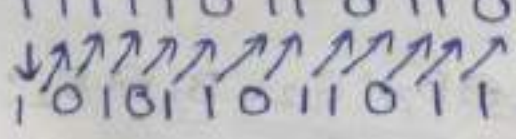
$B_4 = G_4$; $B_3 = B_4 \oplus G_3$; $B_2 = B_3 \oplus G_2$; $B_1 = B_2 \oplus G_1$

convert given 1100101011 gray to Binary.



1 1 0 0 1 0 1 0 1 1
1 0 0 0 0 1 1 0 0 1 0

convert given 111110110110 gray to Binary



1 1 1 1 1 0 1 1 0 1 1 0
1 0 1 0 1 1 0 1 1 0 1 1

Axiomatic definition of Boolean Algebra and switching functions:

Laws of boolean Algebra [properties of Boolean Algebra]

postulates	(a)	(b)
1. postulate 2	$A+0=A$	$A'=1=A$
2. postulate 5	$A+\bar{A}=1$	$A\cdot\bar{A}=0$
3. postulate 3 (Commutative law)	$A+B=B+A$	$AB=BA$
4. postulate 4 (Distributive law)	$A(B+C)=AB+AC$	$A+BC=(A+B)(A+C)$
5. postulate 1 (Associative law)	$A+(B+C)=(A+B)+C$	$(AB)_C=A(BC)$

→ In 1854 George boole introduced a systematic treatment of logic and developed for the purpose an algebraic System now known as is called "Boolean Algebra".

→ Boolean Algebra is like mathematical System defined with Set of elements. Operators and numbers of postulates.

→ Boolean Algebra is a system of mathematical logic it is different from both ordinary algebra and the binary number system.

→ For Boolean $1+1=1$

→ For Binary $1+1=0$, carry$=1$

I commutative law proof:

1) Law $A+B=B+A$ ii) $AB=BA$

Proof

A	B	A+B	B+A	AB	BA
0	0	0	0	0	0
0	1	1	1	0	0
1	0	1	1	0	0
1	1	1	1	1	1

II Associative law proof:

1) Law $A+(B+C)=(A+B)+C$ ii) $(AB)C=A(BC)$

A B C	(B+C)	A+(B+C)	(A+B)	(A+B)+C	AB	(AB)C	BC	A(BC)
0 0 0	0	0	0	0	0	0	0	0
0 0 1	1	1	0	1	0	0	0	0
0 1 0	1	1	1	1	0	0	0	0
0 1 1	1	1	1	1	0	0	1	0
1 0 0	0	1	1	1	0	0	0	0
1 0 1	1	1	1	1	0	0	0	0
1 1 0	1	1	1	1	1	0	0	0
1 1 1	1	1	1	1	1	1	1	1

III Distributive law

1) $A(B+C)=AB+AC$ ii) $A+BC=(A+B)(A+C)$

A B C	(B+C)	A(B+C)	AB	AC	AB+AC	BC	A+BC	(A+B)(A+C)
0 0 0	0	0	0	0	0	0	0	0
0 0 1	1	0	0	0	0	0	0	0
0 1 0	1	0	0	0	0	0	1	0
0 1 1	0	0	0	0	0	1	1	1
1 0 0	1	1	0	0	0	0	1	1
1 0 1	1	1	0	1	1	0	1	1
1 1 0	1	1	1	0	1	0	1	1
1 1 1	1	1	1	1	1	1	1	1

Basic theorems and properties:

Duality Theorem:

principal of duality theorem says that starting with boolean relation, you can derive another boolean relation by

→ changing each OR GATE sign to AND sign

→ changing each AND sign to OR sign

→ complementing any 0(or)1 appearing in the expression

Example: Dual of relation $A+\bar{A}=1$ is $A\cdot\bar{A}=0$

Basic theorems:

theorem 1(a) $A+A=A$

$0+0=0$

$1+1=1$

proof: $A+A=(A+A)\cdot 1$

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

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

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

$=A+A=A(1+1)$

$A+A=A$

theorem 1(b) $A\cdot A=A$

$0\cdot 0=0$

$1\cdot 1=1$

proof: $A\cdot A=A\cdot A+0$

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

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

$A \cdot A = A(1)$

$A \cdot A = A$

theorem (2a) $A+1=1$

$1+0=1$

$1+1=1$

proof: $A+1 = 1 \cdot (A+1)$

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

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

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

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

$= A + \bar{A}$

$A+1 = 1$

Theorem (2b) $A \cdot 0 = 0$

$0 \cdot 0 = 0$

$0 \cdot 1 = 0$

proof: $A \cdot 0 = 0$ by duality of theorem 2(a)

theorem (3): $\bar{\bar{A}} = A$

$\bar{\bar{0}} = 0$

$\bar{\bar{1}} = 1$

proof: complement of $\bar{A}$ is A and also $\bar{\bar{A}}$

theorem (4a) $A + AB = A$

proof: $A + AB = A \cdot 1 + AB$

$= A(1+B)$

$= A \cdot 1$

$A + AB = A$

theorem(4 b) $A(A+B)=A$

proof: $A(A+B) = A\cdot A + AB$

$= A\cdot A + AB$

$= A + AB$

$= A(1+B)$

$A(A+B) = A$

theorem 5(a) $A(\bar{A}+B) = AB$

proof: $A(\bar{A}+B) = (A+AB)(\bar{A}+B)$ → Substitute 4(a) $A+AB=A$

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

$= A\bar{A} + AB + ABB$

$= AB + AB$

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

theorem 5(b) $A + \bar{A}B = A+B$

proof: $A+\bar{A}B = A + AB + \bar{A}B$

$= A + B(A+\bar{A})$ [∵ from 4(a) $A = A+AB$

$= A + B(1)$

$= A + B$

Demorgan's Theorem: (or) Demorgan's laws:

Demorgan's suggested two theorems that form an important th part of a boolean algebra in the equation form there are two types:

(i) $\overline{A+B} = \bar{A}\cdot\bar{B}$ Demorgan's first law

(ii) $\overline{AB} = \bar{A}+\bar{B}$ second law

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

A	B	A+B	$\overline{A+B}$	$\bar{A} \cdot \bar{B}$
0	0	0	1	1
0	1	1	0	0
1	0	1	0	0
1	1	1	0	0

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

A	B	AB	$\overline{AB}$	$\bar{A} + \bar{B}$
0	0	0	1	1
0	1	0	1	1
1	0	0	1	1
1	1	1	0	0

consensus theorem:

In simplification of boolean expression an expression of the form $AB+\bar{A}C+BC$ the term BC is redundant and can be eliminated to form the equivalent expression $AB+\bar{A}C$ the theorem used for the simplification is known as consensus theorem and it is stated as $AB+\bar{A}C+BC = AB+\bar{A}C$

proof:

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

Proof

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

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

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

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

$= AB+\bar{A}C$

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

$A+\bar{A} = 1$

$1+C = 1$

$1+B = 1$

Solving by using consensus theorem

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LHS = RHS.

Find the complement of $A + B + AB\bar{C}$

demorgan's law

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

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

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

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

$\bar{A}\,\bar{B}\,[\overline{AB} + C]$

$\bar{A}\bar{B}\,[\bar{A} + \bar{B} + C]$

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

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

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

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

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

Switching Functions or boolean functions:

→ Boolean expressions are ~~connected~~ constructed by boolean connecting the boolean constants and variables with boolean operation.

→ These boolean expressions are also known as Boolean formula.

→ Use boolean expression to describe boolean function

$f(A, B, C, D) = A + BC + ACD$

product term

literals

boolean function

The literals and terms are arranged in one of the two form

(i) SOP (Sum of product).

Ex: $f(A, B, C) = AB + AC$

$f(A, B, C)$ → Boolean function

$AB + AC$ → Boolean expression

(ii) POS (product of Sum)

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

M- Notations — Minterms & Maxterms

variables A B C	Min terms m_i	Max terms M_i
0 0 0	$\bar{A}\bar{B}\bar{C} = m_0$	$A+B+C = M_0$
0 0 1	$\bar{A}\bar{B}C = m_1$	$A+B+\bar{C} = M_1$
0 1 0	$\bar{A}B\bar{C} = m_2$	$A+\bar{B}+C = M_2$
0 1 1	$\bar{A}BC = m_3$	$A+\bar{B}+\bar{C} = M_3$
1 0 0	$A\bar{B}\bar{C} = m_4$	$\bar{A}+B+C = M_4$
1 0 1	$A\bar{B}C = m_5$	$\bar{A}+B+\bar{C} = M_5$
1 1 0	$AB\bar{C} = m_6$	$\bar{A}+\bar{B}+C = M_6$
1 1 1	$ABC = m_7$	$\bar{A}+\bar{B}+\bar{C} = M_7$

Canonical form Boolean expression contain all i/p variables either in true form (or) its complement form

Canonical SOP → $A\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC$

Canonical POS → $(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)$

Standard form It doesn't contain all i/p variables in each term.

Ex: Standard SOP → $AB + BC + CA$

Standard POS → $(A+B)(B+C)(C+A)$

Converting SOP to SSOP: (min terms) (Canonical SOP)

→ Find missing literal in each product term.

→ AND each product term having missing literals with terms formed by ORING the literal and its compliment.

→ Expand the terms by applying distributive law and re-order the product terms.

→ Reduce the expression by using $A+A=A$

Steps to convert POS to SPOS form (max term) (Canonical POS)

→ Find missing literal in each sum term

→ OR each sum term having missing literal form by AND ing the literal and its compliment.

→ Expand the terms by applying distributive law and re-order the sum terms.

→ Reduce the expression by using $A \cdot A = A$

Ex: convert given expression to ~~standard~~ Canonical SOP form

$$f(A,B,C) = AC + AB + BC$$

SOP → SSOP → (min term (or) canonical min) Σm

$$= AC + AB + BC$$

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

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

$$= \underset{111}{ABC} + \underset{101}{A\bar{B}C} + \underset{110}{AB\bar{C}} + \underset{011}{\bar{A}BC}$$

$$m_7 + m_5 + m_6 + m_3$$

$\Sigma m\,(7,5,6,3)$

$\Sigma m\,(3,5,6,7)$

Ex: convert given expression to &pos form (maxterm) (Canonical POS)

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

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

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

Write the minterms & maxterms for the following functions

1. $f = \Sigma\,(1,4,6,7,9)$

$$\bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D}$$

2. $f = \Pi M\,(3,5,7,11,14)$

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

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

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

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

$$= \Sigma M\,(3,9,12)$$

conversion from one canonical form to another.

1. $f(A,B,C) = \Sigma m\,(0,1,2,3,5,7)$

$$= \Pi M\,(4,6)$$

2. $f(A,B,C,D) = \Pi M\,(3,7,10,13,14)$

$$= \Sigma m\,(0,1,2,4,5,6,8,9,11,12)$$

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

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

$= AC(B+\bar{B}) + AB\bar{C}$ $\quad [A+\bar{A}=1]$

$= AC + AB\bar{C}$ $\quad [A+\bar{A}B = A+B]$

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

$= A(C+B)$

$= A(B+C)$

Simplify the expression

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

Demorgan's law $\overline{A+B} = \bar{A}\cdot\bar{B}$

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

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

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

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

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

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

$= AB + A\bar{B}$

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

$= A$

Q. * $\bar{B}\,\bar{C}D + (\overline{B+C+D}) + B\bar{C}\bar{D}E$ simplify the equation (or) minimize the given eqn

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

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

$= \bar{B}\bar{C}D + B\bar{C}\bar{D} + B\bar{C}\bar{D}E$

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

$= \bar{B}C\bar{D} + \bar{B}\bar{C}\bar{D}$ $\quad [1+A=1,\ A+\bar{A}=1]$

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

Q8.

Simplify the expression $y = AB + \bar{A}C + A\bar{B}C(AB+C)$

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

$= AB + \bar{A}C + A\bar{B}C$ $\quad [A\cdot\bar{A}=0,\ A+A\bar{B}=A+B]$

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

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

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

Q9. compliment of the boolean function.

$f = (b\bar{c} + \bar{a}d)(a\bar{b} + c\bar{d}) = \overline{(b\bar{c} + \bar{a}d)(a\bar{b} + c\bar{d})}$

$f = \overline{(b\bar{c} + \bar{a}d)} + \overline{(a\bar{b} + c\bar{d})}$

$= (\overline{b\bar{c}}\,\overline{\bar{a}d}) + (\overline{a\bar{b}}\,\overline{c\bar{d}})$

$= (\overline{b\bar{c}})(\overline{\bar{a}d}) + (\overline{a\bar{b}})(\overline{c\bar{d}})$

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

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

Q10.

Simplify the logic function $f(A,B,C,D) = AB + AC + C + AD + ABC + ABC$

$= AB + C(1+A) + AD + ABC$

$= AB + C + AD + ABC$

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

$= AB + C + AD$

$= A(B+D) + C$

Q4 $A+B+\bar{A}\bar{B}C$ reduce (or) minimize (or) Simplify

$$(A+B)+\bar{A}\bar{B}C \qquad \left[\begin{matrix} A+\bar{A}B = A+B \\ B+\bar{B}C = B+C \end{matrix}\right]$$

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

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

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

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

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

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

$$= A+B+C$$

Q5

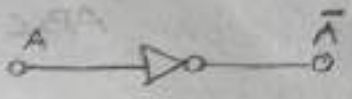
Digital Logic Gates:

Logic gate is the fundamental building block of digital System the name logic gate is derived because the output and input patterns assigned logically.

Logic gates are the basic elements that make a digital system these are logic gates (NOT, AND, OR, NAND, NOR, EXOR, EXNOR)

A gate is a circuit with one or more input voltages but only one output voltage

1. NOT(or) Inverter Gate:

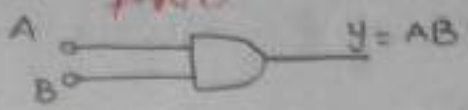


Truth Table

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

2. AND Gate:

AND



Truth Table

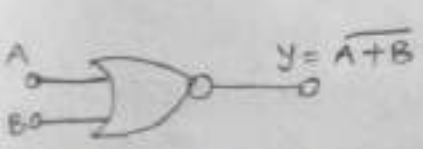
A	B	y = AB
0	0	0
0	1	0
1	0	0
1	1	1

3. OR Gate:

Truth Table

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

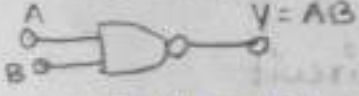
4. NOR Gate:



Truth Table

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

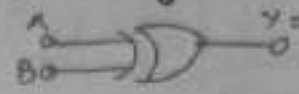
5. NAND Gate:



Truth Table

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

6. Ex-OR gate: [Exclusive-OR]



(or)

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

Truth Table

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

7. Ex-NOR Gate:

(or)

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

Truth Table

A	B	$y = AB + \bar{A}\bar{B}$
0	0	1
0	1	0
1	0	0
1	1	1

properties of Ex-OR gate:

1. $A \oplus A = 0$
2. $A \oplus \bar{A} = 1$
3. $A \oplus 1 = \bar{A}$
4. $A \oplus 0 = A$
5. Ex-OR as a modulo 2 adder
6. $(AB) \oplus (AC) = A(B \oplus C)$
7. $A \oplus B = C$ then

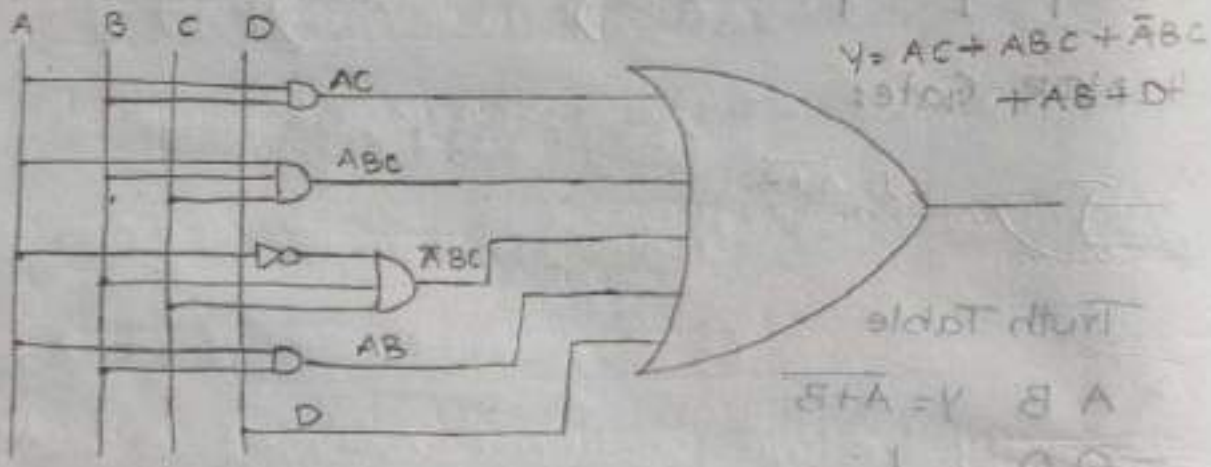
   $A \oplus C = B$

   $B \oplus C = A$

   $A \oplus B \oplus C = 0$

construct the logic diagram for the given expression

Q8. $Y = AC + ABC + \bar{A}BC + AB + D$

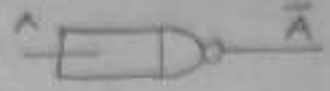


Q9 Universal gates - multilevel NAND/NOR Realization:

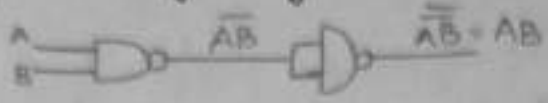
→ NAND and NOR gates are known as universal gate since any logic function can be implemented using NAND or NOR gates and also we can implement any circuit.

→ NAND gate is most preferable then NOR gate it consumes less power

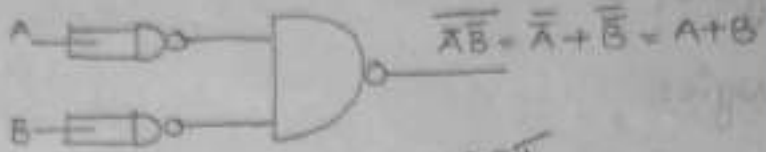
1. Using NAND gate generate (or) implement NOT gate:



2. Using NAND gate generate AND gate:



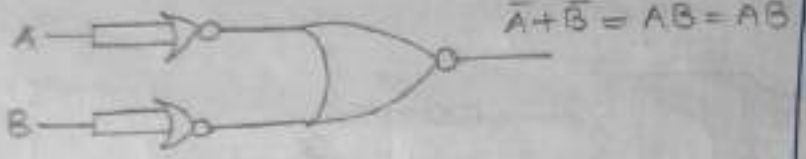
3. Using NAND gate generate OR gate:



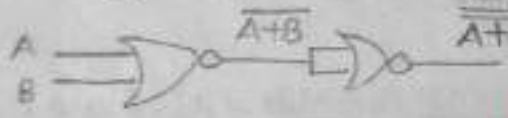
4. Using NOR gate generate NOR gate:

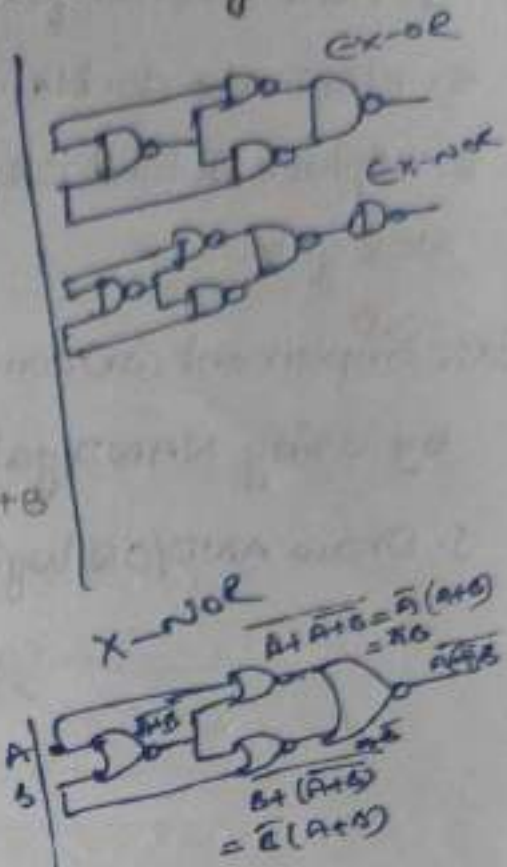
NOT

5. Using NOR gate generate AND gate:



6. Using NOR gate generate OR gate:





NAND & NOR Implementation

Q8 Conversion of AND/OR/NOT logic to NAND/NOR logic gate:

Steps: To reduce Ic's required we can use NAND or NOR gates to implement Boolean expression

1. AND (or) OR logic
2. If NAND hardware has been choosen add bubles output of each AND gate and input Said of OR gates
3. If NOR hardware has been choosen add bubbles output of each OR gate and input Said of each AND gate

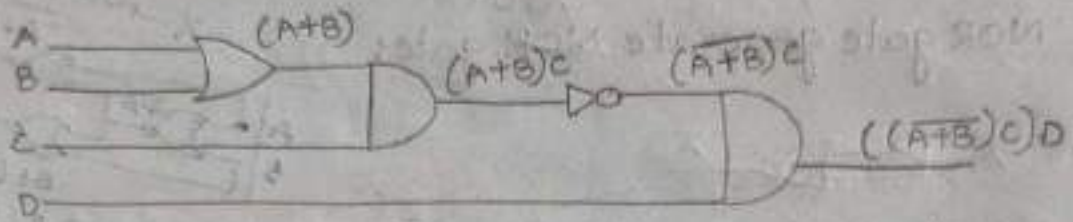
4. Add inverter for each buble

5. Eliminate double inverters

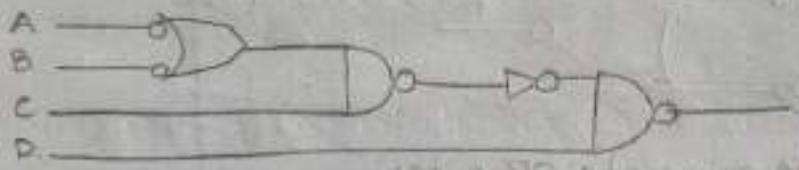
6. replace bubled OR gate by NAND and bubled AND gate by NOR gate

Ex: Complement (or) construct given Boolean expression $[\overline{(A+B)C}]D$ by using NAND gate?

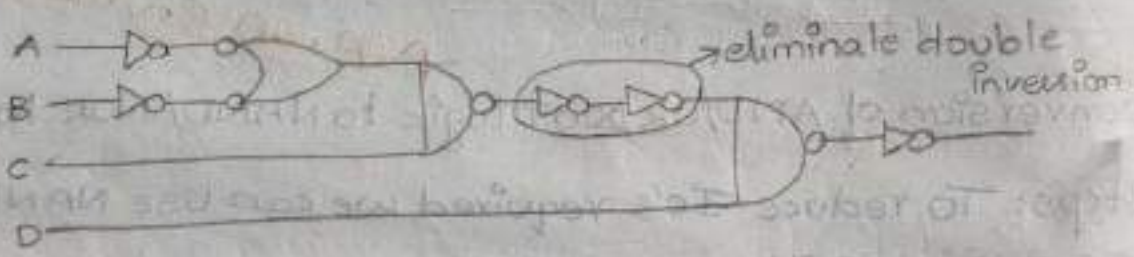
1. Draw AND/OR logic:



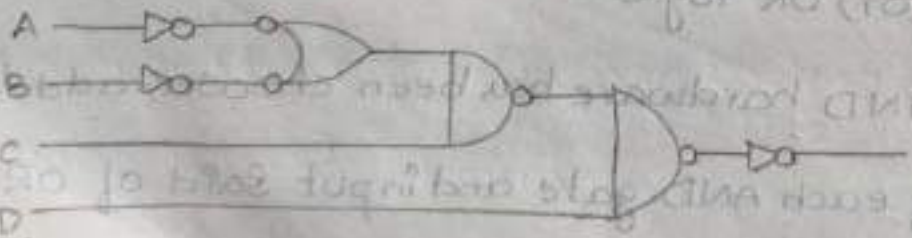
2. Add bubble at o/p side of AND of i/p side of OR gate:



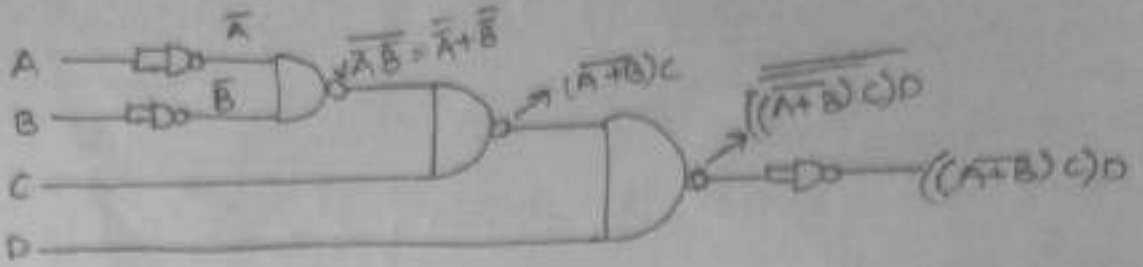
3. Add inverter for each Bubble:



4. Eliminate double inversion:



5. Replace bubbled OR by NAND and replace invert by single i/p NAND

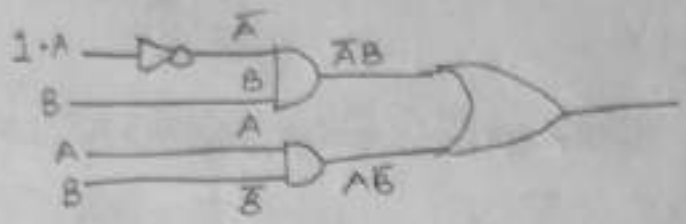


Q8

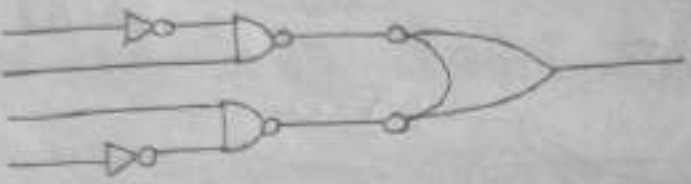
Implement given expression for EX-OR by using NAND gate

$$Y = \bar{A}B + A\bar{B} \text{ (OR) } A \oplus B$$

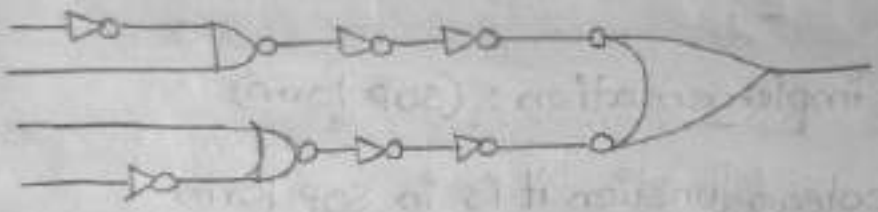
1.



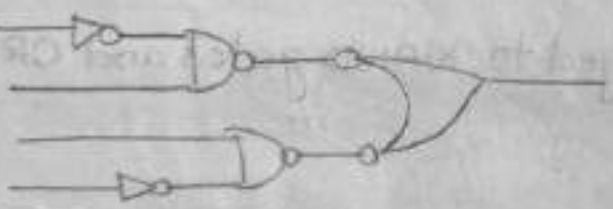
2.



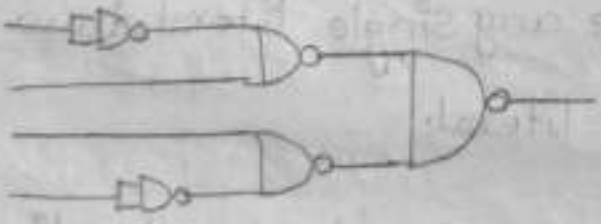
3.



4.



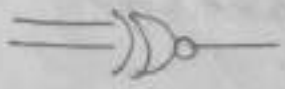
5.



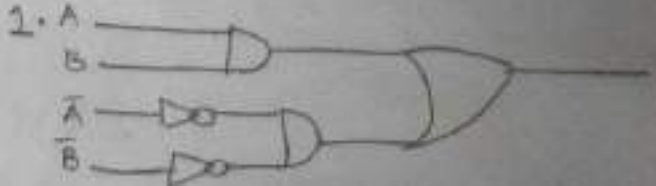
NOR gate implementation by using NOR

EX-NOR by using NOR gate:

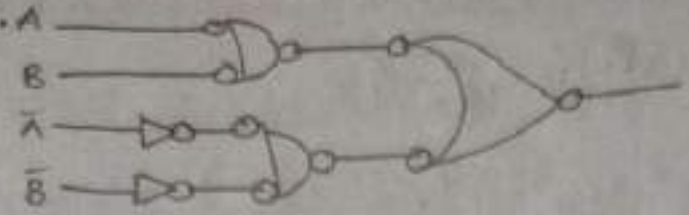
$$Y = AB + \bar{A}\bar{B}$$



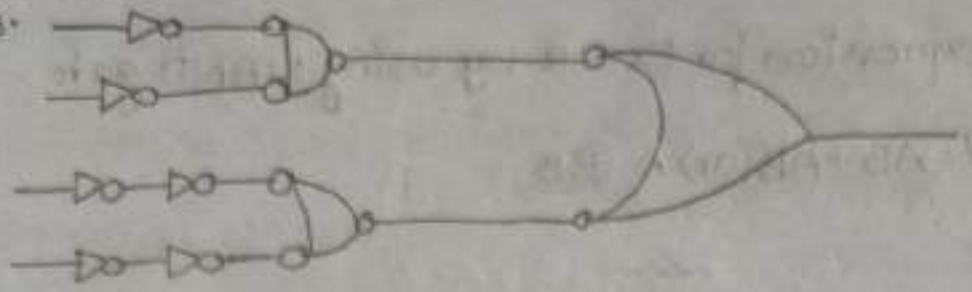
1.



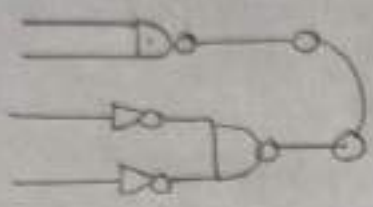
2.



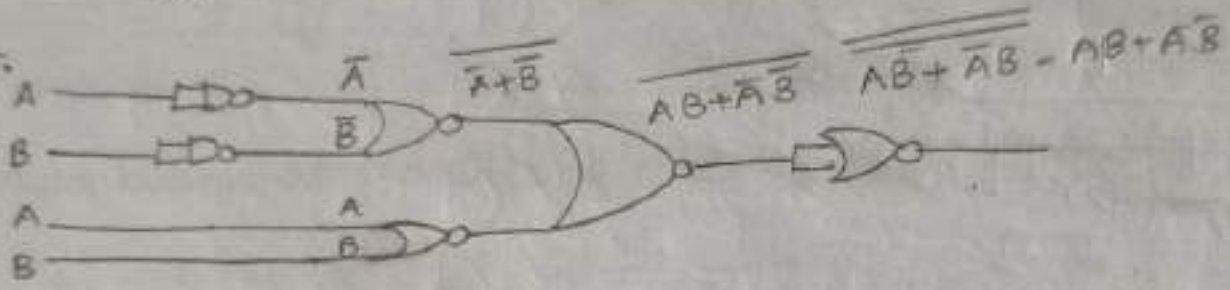
3.



4.



5.

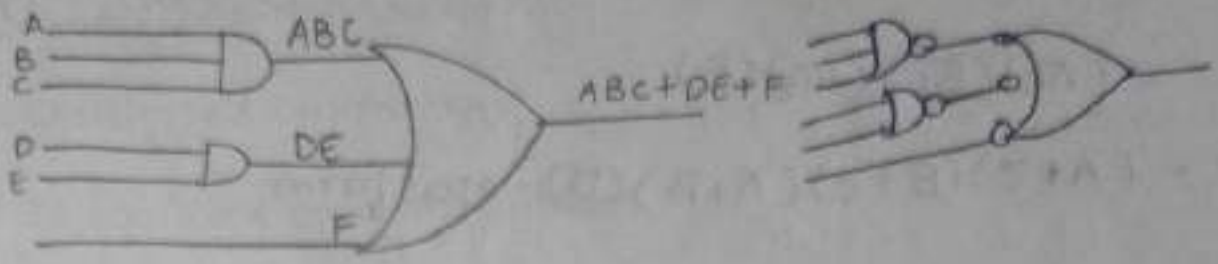


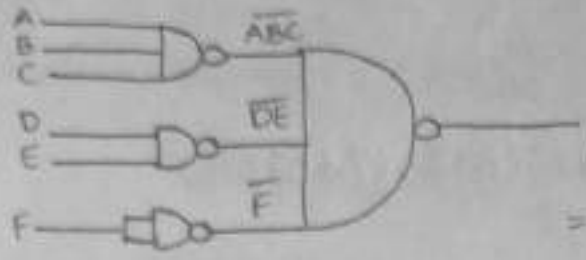
NAND – NAND implementation: (SOP form).

1. Simplify the boolean function it is in SOP form
2. All AND gates are changed to NAND gates and OR gate to bubbled OR gates.
3. if boolean function include any single literal draw NAND gate for each single literal.

Ex: $F = ABC + DE + F$ implement given boolean function by using NAND – NAND implementation.

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





$(\overline{ABC})(\overline{DE})(F) =$

$((\overline{ABC})(\overline{DE}))+\overline{F}$

$= \overline{\overline{ABC}} + \overline{\overline{DE}} + \overline{F}$

$= ABC + DE + F$

Ex: $AC + ABC + \bar{A}BC + AB + D$

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

NOR - NOR Implementation (POS form):

1. Simplify the given boolean function in the form POS form
2. Draw NOR gate for each Sum term that has two or more literals.
3. If boolean function includes any single literal Draw a NOR gate for each single literal
4. Draw a single NOR gate in the second level

Ex: $f = AC + BC + AB + D$ implement the given boolean expression by using NOR - NOR implementation

Apply Demorgan's Theorem

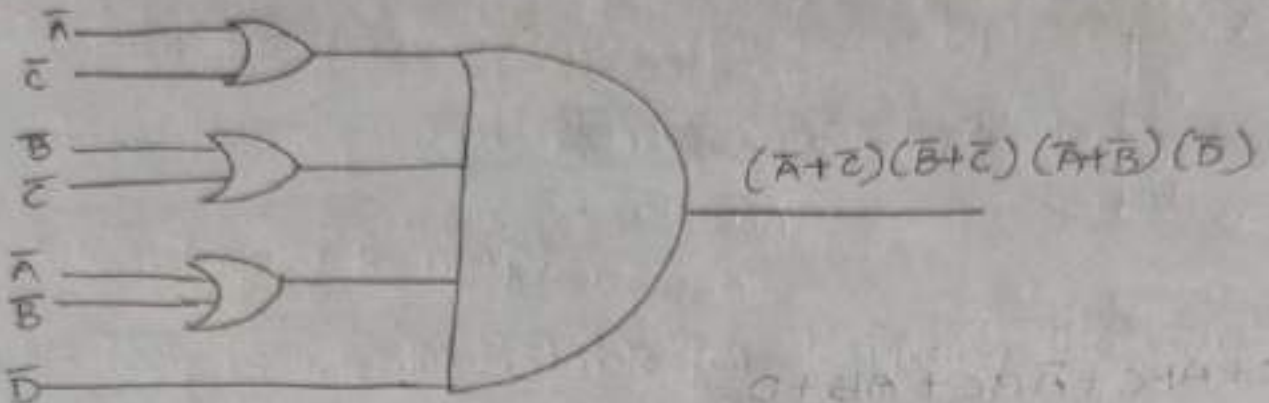
$f = AC + BC + AB + D$

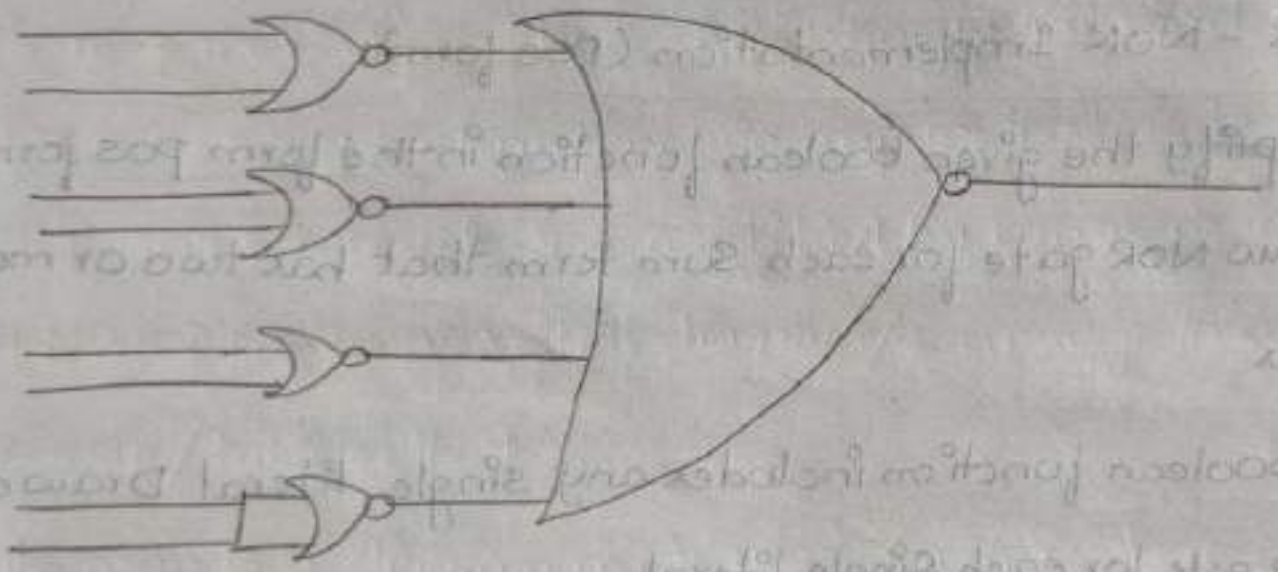
$\bar{f} = \overline{\underbrace{AC + BC}_{a} + \underbrace{AB + D}_{b}}$

$= (\overline{AC + BC})(\overline{AB + D})$

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

$= (\overline{A}+\overline{C})(\overline{B}+\overline{C})(\overline{A}+\overline{B})(\overline{D}) \rightarrow$ pos form





Ref:- $AC+BC+AB+D$ implement by using NOR-NOR implementation

$f = AC+BC+AB+D \rightarrow SOP$

$=$ We need to convert into pos by applying demorgan's theorem it is complement

$$\overline{AC+BC+AB+D}$$

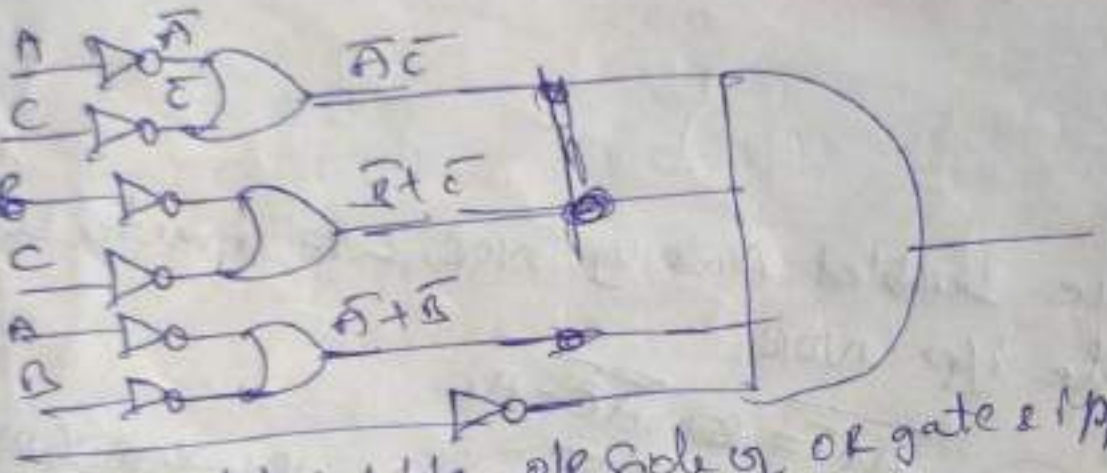
$$\overline{(AC+BC)}\,\overline{(AB+D)}$$

$$(\overline{AC})(\overline{BC})(\overline{AB})(\overline{D})$$

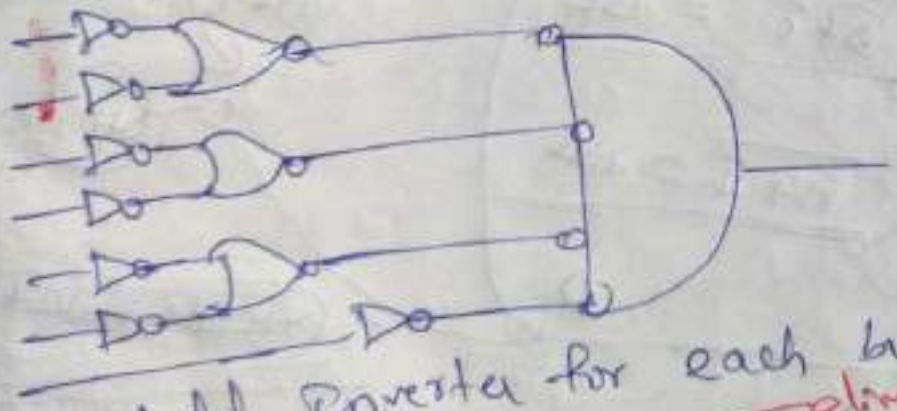
$$(\overline{A}+\overline{C})(\overline{B}+\overline{C})(\overline{A}+\overline{B})(\overline{D}) \rightarrow \text{POS}.$$

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

$\overline{AB} = \overline{A}+\overline{B}$

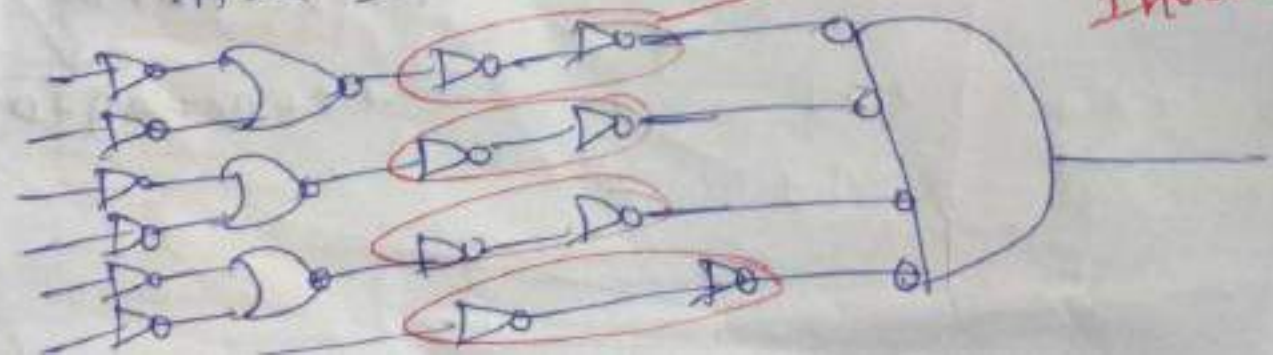


Add bubble o/p side of OR gate & i/p of AND



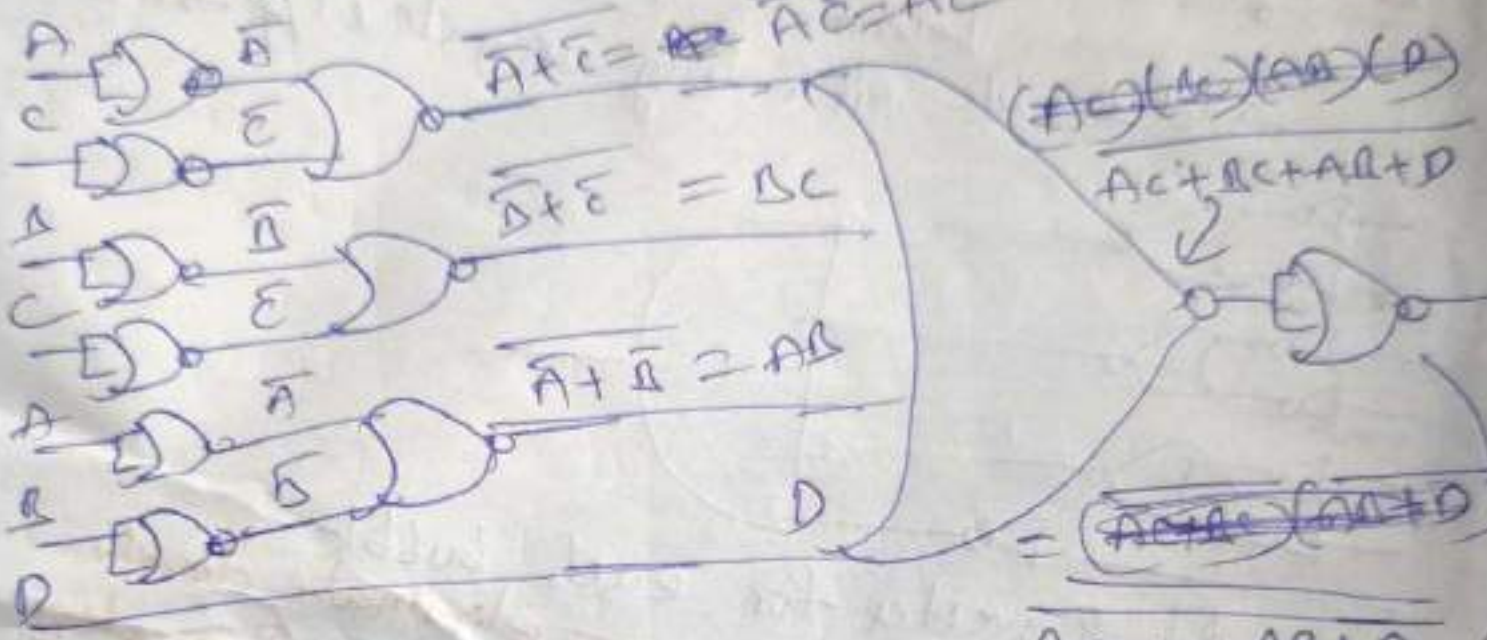
Add Inverts for each bubble

eliminate double Inverters



(3)

Replace bubbled AND by NOR, & Inverter by single I/P NOR.



$= AC + BC + AB + D$

Signed Binary Numbers: Signed Binary numbers

1. Unsigned Numbers:

→ It contains only magnitude

→ +ve number

decimal number = $(108)_{10}$

$= (1101100)_2$

2. Signed Numbers:

System can't understand symbol so we use signed representation Signed number represent in 3 ways.

a) Sign Magnitude form:

0 1101100 → +108 → Total 8 bits
sign magnitude

1 1101100 → −108
sign magnitude

b) 1's comp form:

+108 → 01101100 ↘

−108 → 10010011 → 1's comp

c) 2's comp form:

+108 → 01101100 ↘

−108 → 10010011 → 1's comp

10010100 → 2's comp

1) Ex: +96 → 0 1100000
−96 → 1 1100000

Ex: +64 → 01000000
−64 → 11000000

2)

Find Signed Binary addition for the following numbers using 2's comp. form.

1) $-6, +13$.

$-6 \rightarrow 11111010$

$+13 \rightarrow 00001101$

$+7 \quad 1\,00000111$

Ignore carry

$+7$

$-6+13 = +7$ Ans.

$+6 \rightarrow 0000\,0110$

1's comp $-6 \rightarrow 1111\,1001$

$+1$

2's comp $\rightarrow 1111\,1010$

$+13 \rightarrow 0000\,1101$

$-13 \rightarrow 1111\,0010$

$+1$

2's $\quad 1111\,0011$

2) $+6, -13$

$+6 \rightarrow 0000\,0110$

$-13 \rightarrow 1111\,0011$

$-7 \quad 1111\,1001$

-7.

$+7 \rightarrow 0000\,0111$

$-7 \rightarrow 1111\,1000$

$+1$

2's comp $\quad 1111\,1001$

Answer correct.

ASCII

35 - #, 36 - $, 37 → %, 38 - & and dollar

48 - 57 → 0 to 9

65 - 90 → (A-Z)

(97 - 122) → (a-z)

126 → ~

Ex 1001101 0001111 Convert to ASCII

1001101 1000111 1100111

M G g

Ans is MGg.

Ex ASCII is BC4 convert to Binary

B(66) C(99) 4(52)

1000010 1100011 0110100

Aw ① 100111110011111101111 into ASCII ?

② ASCII BB48 to Binary ?.

1) 1001111 1001111 1101111

79 79 111

O O o

Ans OOo

2) B(66) b(98) 4(52) 8(56)

1000010 1100010 0110100 0111000

Other logic operations

16 different logic operations with 2 binary variables

$2^{2^n} = 2^{2\times 2} \Rightarrow 2^4 = 16$. [n=2 binary variables x, y]

Truth tables for the 16 functions of two binary variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean expressions for 16 functions of two variables

Boolean function	Operator symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	EX-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x+y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(\overline{x \oplus y})$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (\overline{xy})'$	$x \uparrow y$	NAND	NOT-AND
$F_{15} = 1$		Identity	Binary constant 1

Binary storage (saving data as 0's & 1's)
computers store everything as 0's & 1's.
one such 1 is called a bit, it can store one bit (FF → flip-flop)

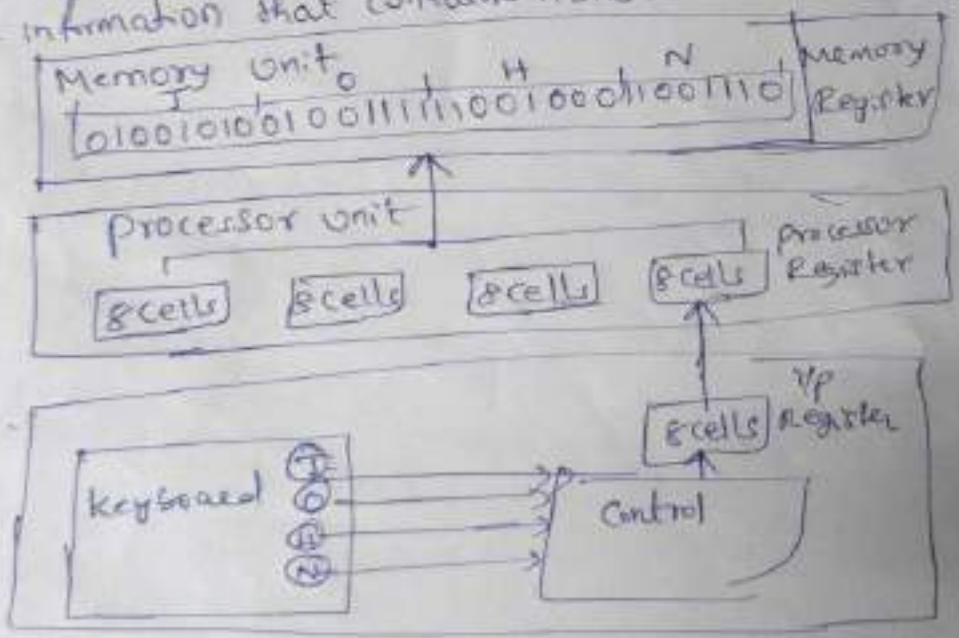
Registers (tiny, super fast storage places in the CPU for quick use)
Registers are small storage boxes inside the CPU. → They hold numbers, instructions, or addresses while the CPU is working.
Ex An 8-bit register can store 8 values of 0/1 like 10101011

Binary logic Binary logic means using only two values: 0 & 1
0 = false/off, 1 = true/on.
Computers use this logic to do all calculations and decisions. Ex" AND, OR, NOT gate.

Binary storage and registers

Binary information in a digital computer must have a physical existence in some medium for storing individual bits. A binary cell is a device that possesses two stable states and is capable of storing one bit (0 or 1) of information.

Register: Register is a contiguous group of binary cells. A register with n cells can store any discrete quantity of information that contains n bits.



Transfer of information among registers

EBCDIC

A-Z a-z 0-9

A-I F0-F9

UNIT-II

GATE LEVEL MINIMIZATION

Drawback of boolean Algebraic method:

Simplify the given boolean expression by using theorem it is time taken process it requires 5 to 6 steps

Advantage of k-map:

→ k-map is ideally suitable for designing the combinational logic circuits using either sop (or) pos

→ k-map method provides a state straight forward procedure for minimising boolean function map method is first proposed by veitch and modified by karnaugh in 1953.

It is also known as veitch diagram / karnaugh map / k-map / v-map / table method.

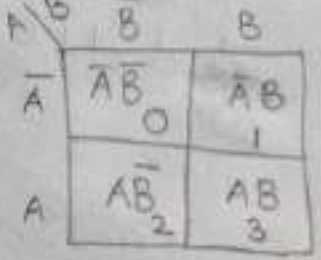
→ the map is a diagram made up of squares where each square represent one min term or max term.

→ For sop a boolean expression with the sum term can be ploted by placing 1 in each cell

→ For pos placing 0 in each cell.

2- variable k-map:

$2^2 = 4$ cells are required



4-variable k

$2^4 = 16$ cell

→ once the

have to

function

→ Groupi

Pair:

A pair

cancel

Quad

Quad

varia

Octa

Octa

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	$\bar{A}\bar{B}\bar{C}$ 0	$\bar{A}\bar{B}C$ 1	$\bar{A}BC$ 3	$\bar{A}B\bar{C}$ 2
A	$A\bar{B}\bar{C}$ 4	$A\bar{B}C$ 5	ABC 7	$AB\bar{C}$ 6

3-variable k-map:

$2^3 = 8$ cells.

4-variable k-map:

$2^4 = 16$ cells.

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}\bar{D}$ 0	$\bar{A}\bar{B}\bar{C}D$ 1	$\bar{A}\bar{B}CD$ 3	$\bar{A}\bar{B}C\bar{D}$ 2
$\bar{A}B$	$\bar{A}B\bar{C}\bar{D}$ 4	$\bar{A}B\bar{C}D$ 5	$\bar{A}BCD$ 7	$\bar{A}BC\bar{D}$ 6
AB	$AB\bar{C}\bar{D}$ 12	$AB\bar{C}D$ 13	$ABCD$ 15	$ABC\bar{D}$ 14
$A\bar{B}$	$A\bar{B}\bar{C}\bar{D}$ 8	$A\bar{B}\bar{C}D$ 9	$A\bar{B}CD$ 11	$A\bar{B}C\bar{D}$ 10

[Here follow Gray code Rule]
at a time one bit position is different

→ once the boolean function is plotted on the k-map we have to use grouping technique to simplify the boolean function

→ Grouping is combining terms in adjacent cells.

Pair:

A pair is a group of two adjacent cells in a k-map it cancels one variable in simplification

Quard:

Quard is a group of four adjacent cells it cancels two variable in simplification

Octate:

Octate is a group of 8-adjacent cells in a k-map it

Cancels three variables in simplification

Minimal SOP Forms:

→ plot the k-map and place 1's in those cells

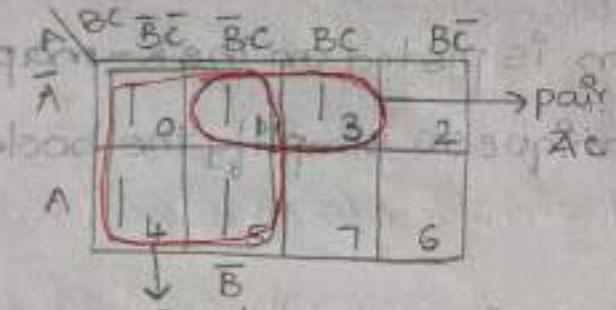
→ check the k-map for adjacent 1's and encircle the isolated 1's.

→ check for quards and octates of adjacent 1's

→ Make sure there are minimum no of groups

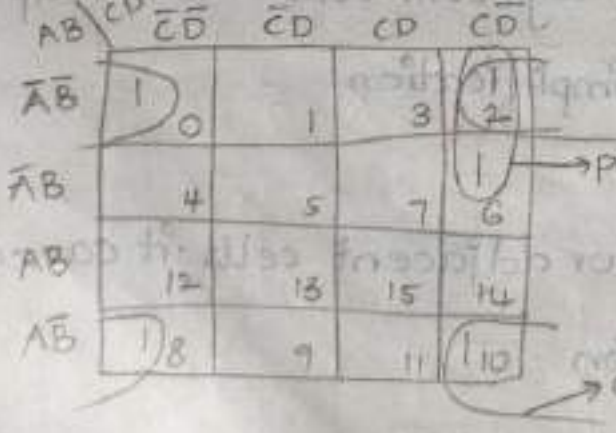
Ex: Minimize the expression by using k-map & realize with NAND gate

$$Y = A\bar{B}C + \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$$
$$101 \quad 001 \quad 011 \quad 100 \quad 000$$



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

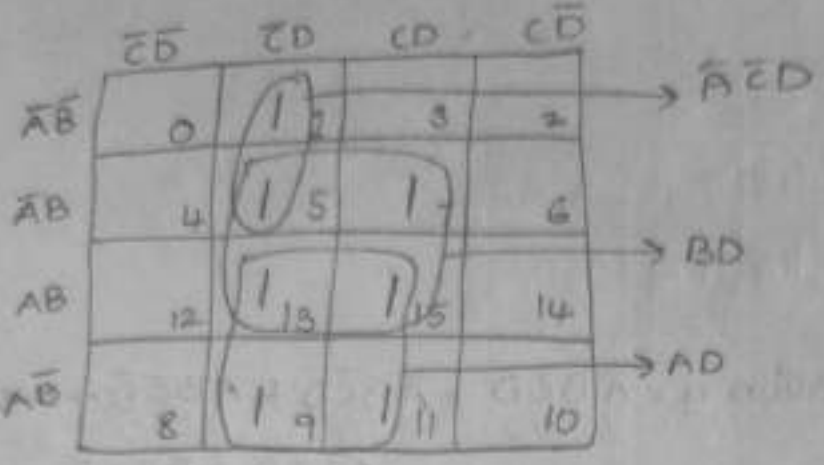
$$f = (A, B, C, D) = \Sigma (0, 2, 6, 8, 10)$$



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

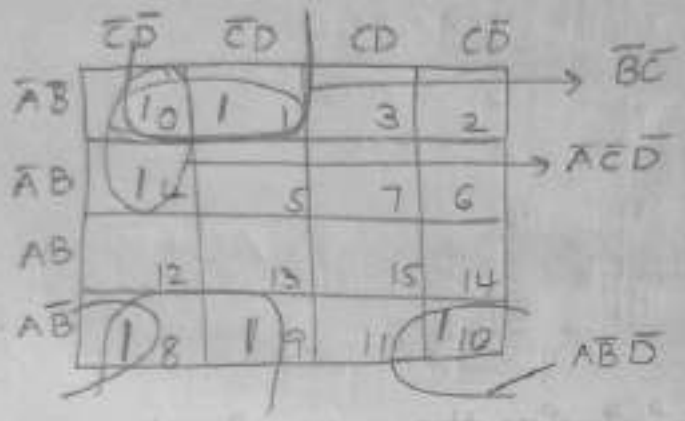
Q8 Minimize the exp using k-map

$y = \Sigma m(1,5,7,9,11,13,15)$



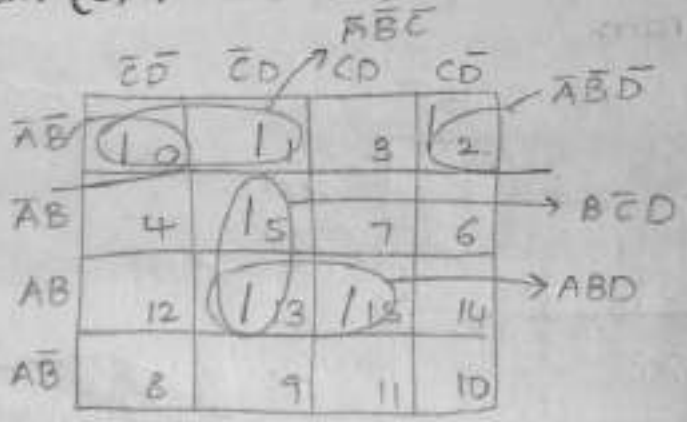
$y = AD + BD + \bar{A}\bar{C}D$

$f(A,B,C,D) = \Sigma m(0,1,4,8,9,10)$



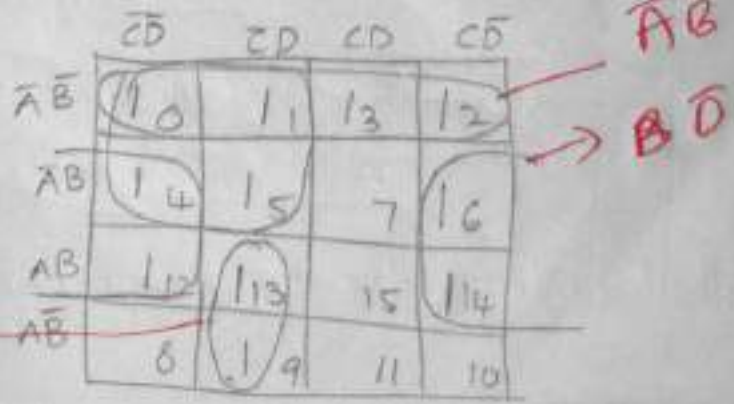
$f = \bar{B}\bar{C} + \bar{A}\bar{C}\bar{D} + A\bar{B}\bar{D}$

$y = \Sigma m(0,1,2,5,13,15)$

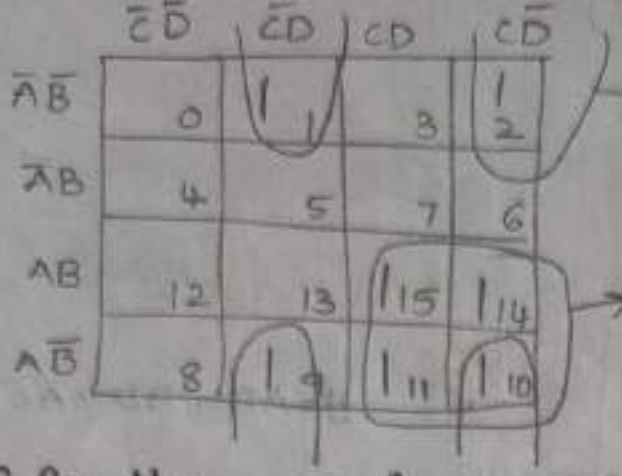


$y = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{D} + B\bar{C}D + ABD$

$y = \Sigma m(0,1,2,3,4,5,6,9,12,13,14)$

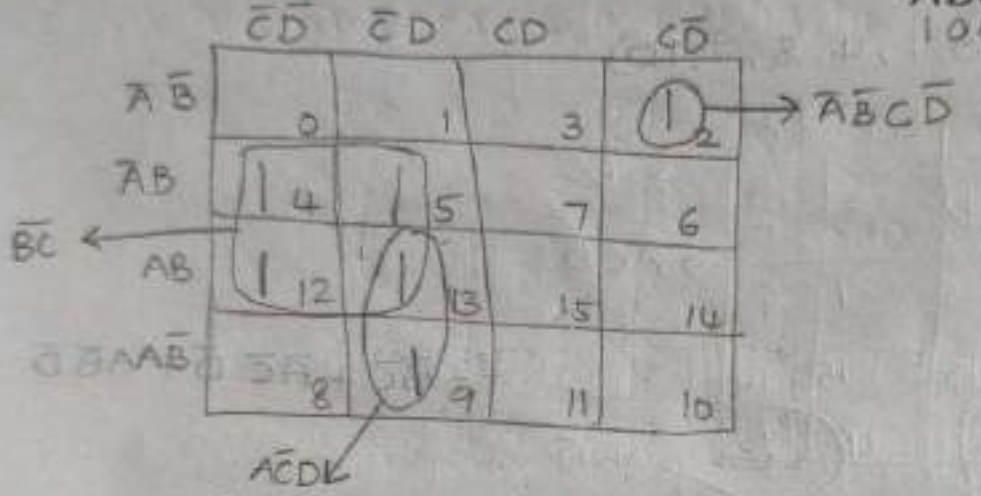


$y = \Sigma m(1,2,9,10,11,14,15)$

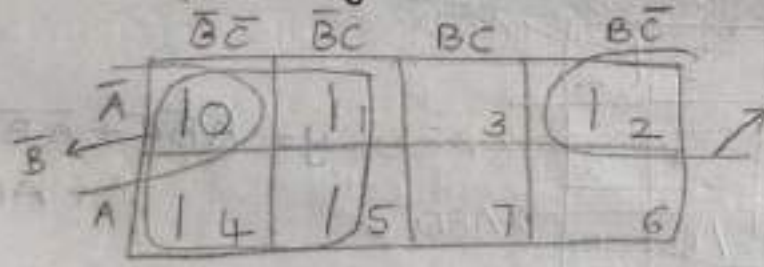


Minimize the expression $y = \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D + A\bar{B}\bar{C}D + \bar{A}\bar{B}CD$

0100 0101 1100 1101 1001 0010

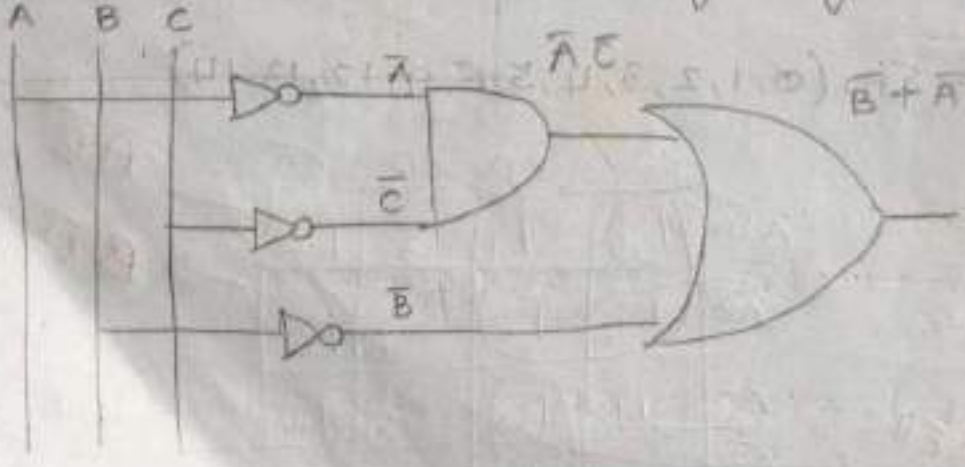


$y = \Sigma m\,(0,1,2,4,5)$ minimize the exp using k-map & draw the logic diagram.



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

Logic diagram



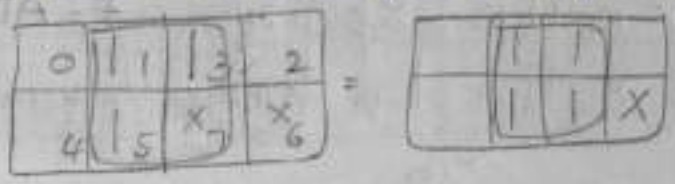
Drawbacks of Don't care conditions:

- An incompletely specified function is a don't care condition

A don't care condition is either '0'(or)'1' in order to produce the simplist o/p expression

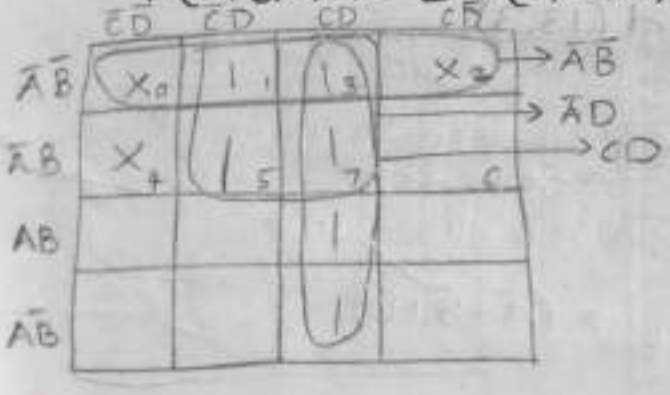
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	x
1	1	1	x

* In some logic circuits, certain i/p conditions never occur therefore the corresponding o/p never appears
* In such cases the o/p level is not defined if can be either High (or) low
* Here to make a quad by placing '1'



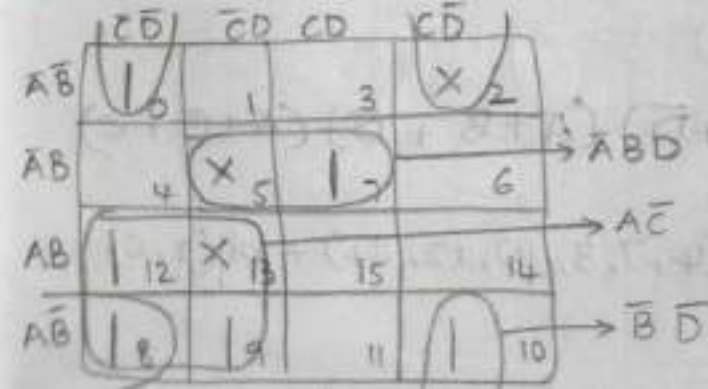
Find the reduced Sop form of the following function.

$f(A,B,C,D) = \Sigma m(1,3,5,7,11,15) + \Sigma d(0,2,4)$ (or) ϕ



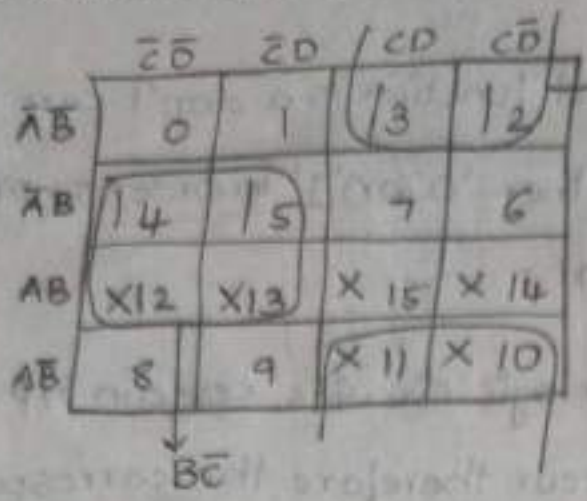
$f = \bar{A}\bar{B} + \bar{A}D + CD$

$f(A,B,C) = \Sigma m(0,7,8,9,10,12) + \Sigma d(2,5,13)$ d (or)



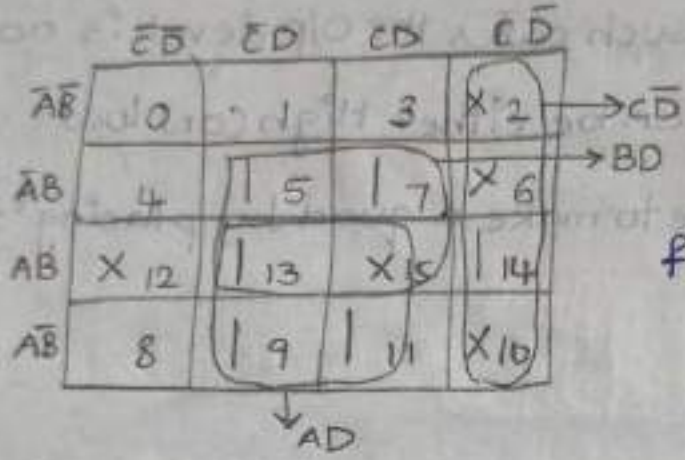
$f = \bar{A}BD + A\bar{C} + \bar{B}\bar{D}$

$f(A,B,C,D) = \Sigma m(2,3,4,5) + \Sigma d(10,11,12,13,14,15)$



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

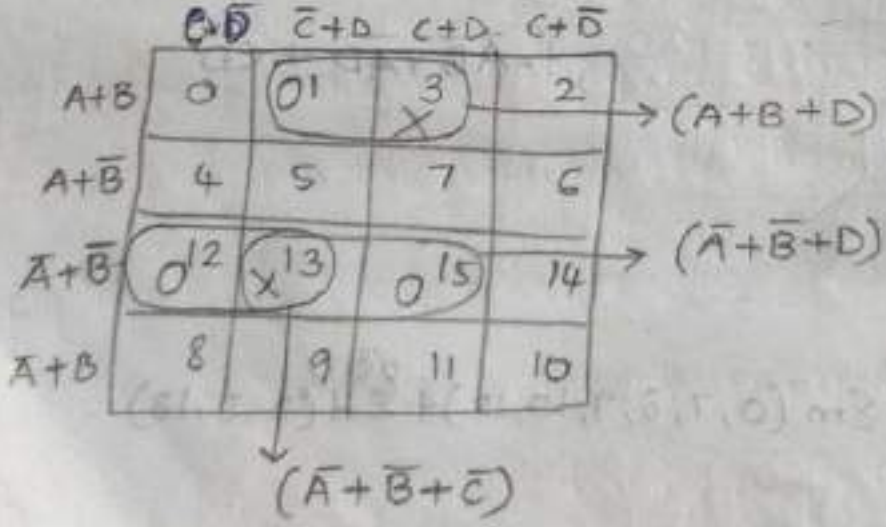
$f = (A,B,C,D) = \Sigma M(5,7,9,11,13,14) + \Sigma d(2,6,10,12,15)$



$f = AD + BD + C\bar{D}$

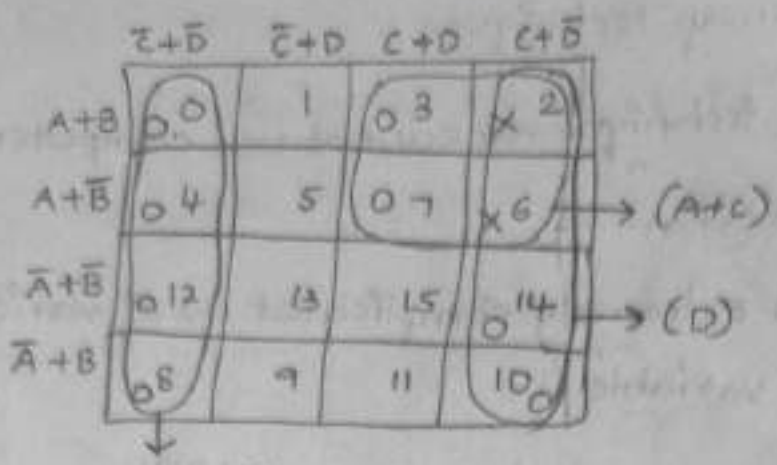
Pos form:

$f(A,B,C,D) = \Sigma\Pi M(1,12,15) + d(13,3)$



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

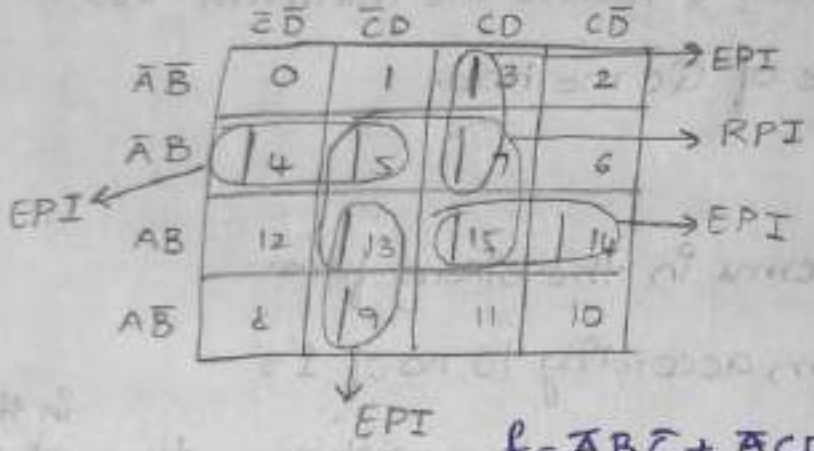
$f(A,B,C,D) = \Pi M(0,3,4,7,8,10,12,14) + d(2,6)$



$f = (D)(A+C)$

prime Implements and essential prime Implements:

Q) $f(A, B, C, D) = \Sigma\, m(3, 4, 5, 7, 9, 13, 14, 15)$

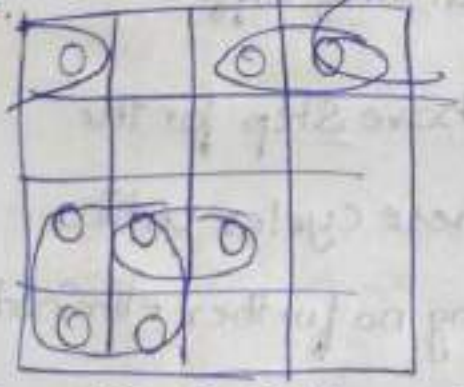


$f = \bar{A}B\bar{C} + \bar{A}CD + A\bar{C}D + A\bar{B}C$

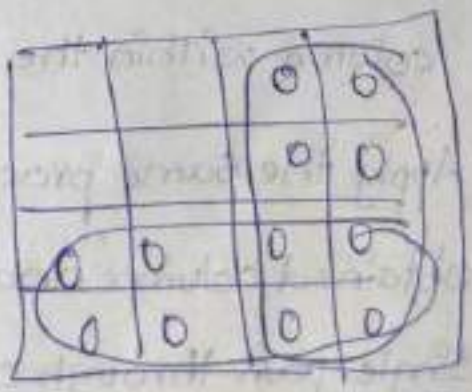
Drawbacks of k-map technique:

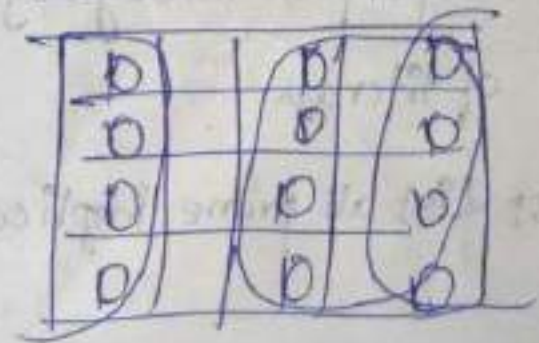
→ It is a manual technique we cannot use computer

Q) $f = \Pi M(0, 2, 3, 8, 9, 12, 13, 15)$

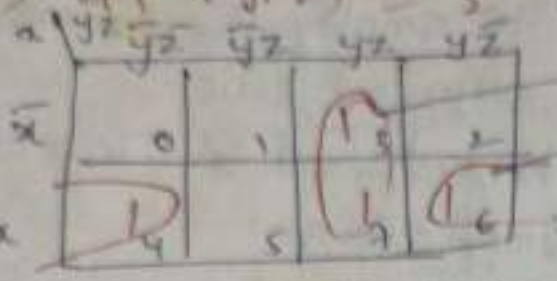


$(\bar{A}+C)(\bar{C}+\bar{B}+\bar{D})(A+B+\bar{C})(A+B+D)$





=> $f(x, y, z) = \sum (3, 4, 5, 7)$

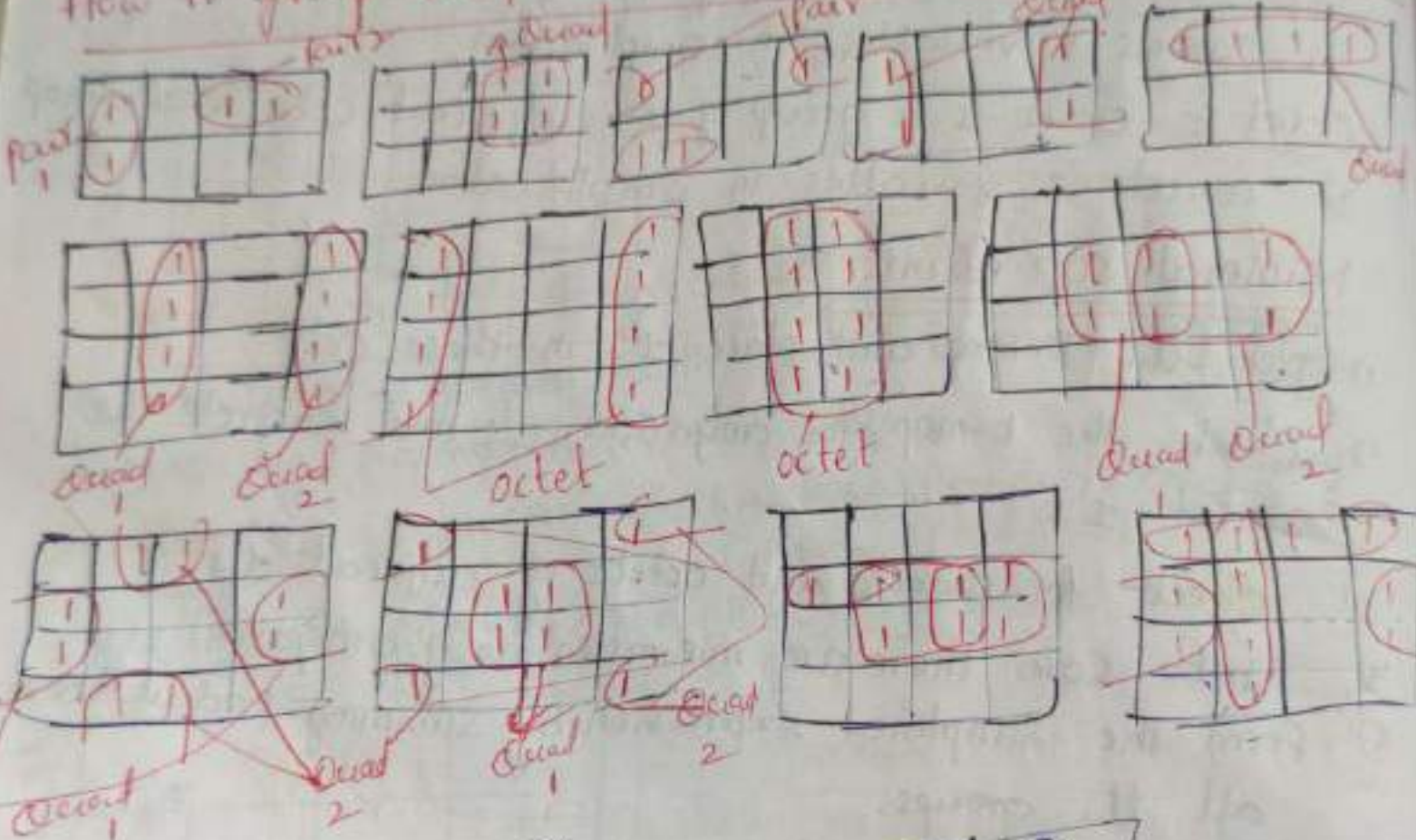


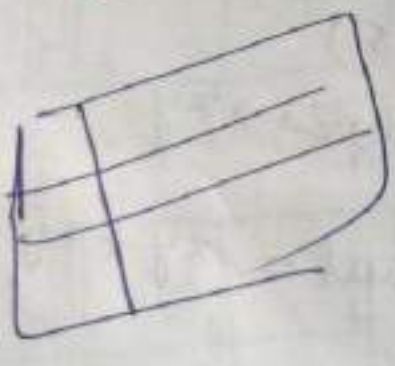
$\rightarrow \bar{x}yz + xyz = yz$

$\rightarrow x\bar{y}\bar{z} + xy\bar{z} = x\bar{z}$

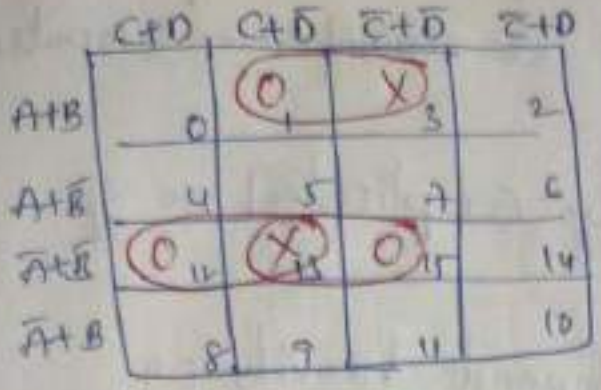
Result $= yz + x\bar{z}$ Ans.

How to group two pair, Quad octet on K map Examples

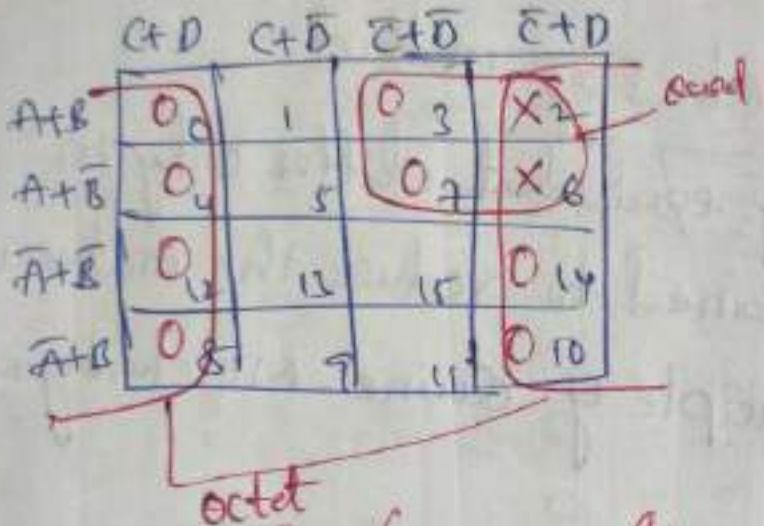




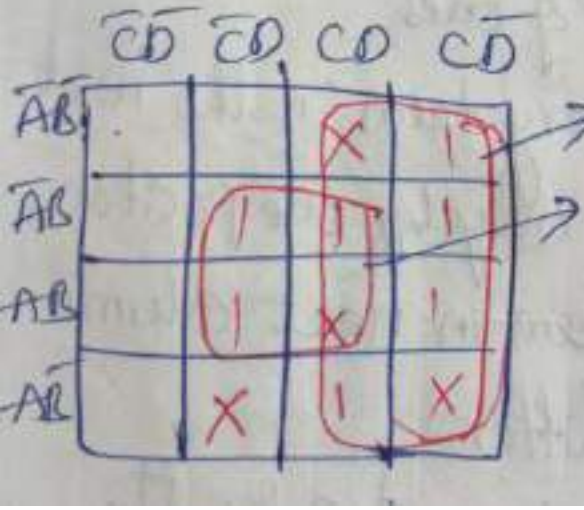
$f(A,B,C,D) = \Pi M(1,12,15) + Sd(3,13)$

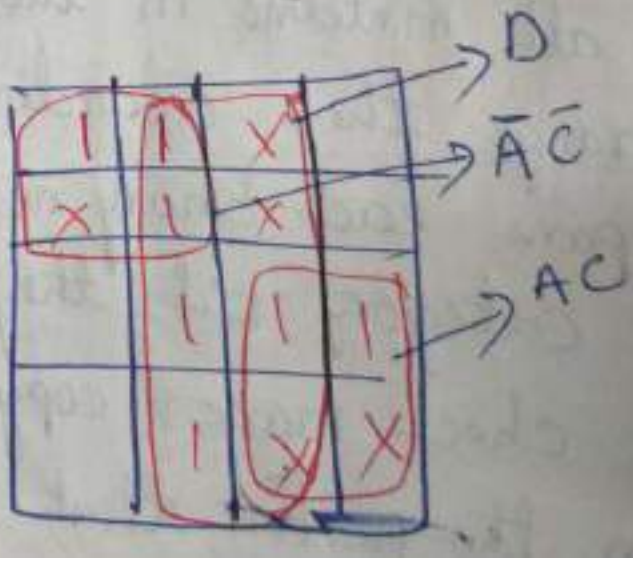


$f(A,B,C,D) = \Pi M(0,3,4,7,8,10,12,14) + d(2,6)$



$f(A,B,C,D) = (2,3,5,6,7,9,10,11,13,14,15)$

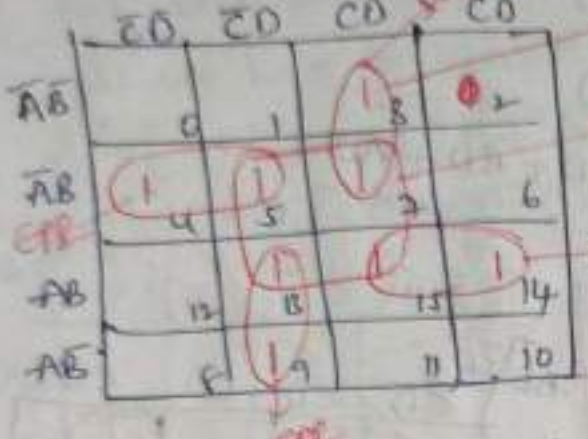




DNF

Prime Implicants & Essential Prime Implicants

$f(A,B,C,D) = \sum(2,4,5,7,9,13,14,15)$ = $\bar{A}CD + A\bar{B}C + ACD + AB\bar{C}$



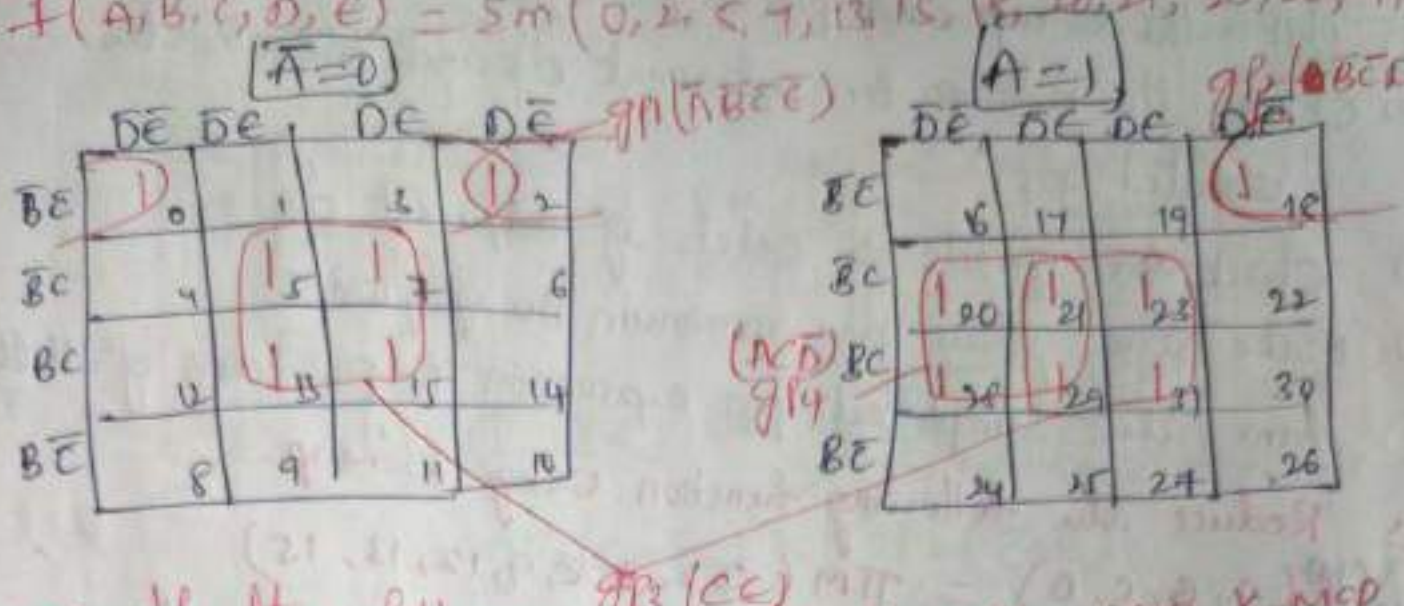
each of these subcubes is called a PI (Prime Implicant). the prime implicant which contains at least one 1 which cannot be covered by any other prime Implicant is called an (EPI)

→ The prime implicant which is neither an essential prime implicant nor a redundant prime implicant is called a (Selective PI)

V.IM

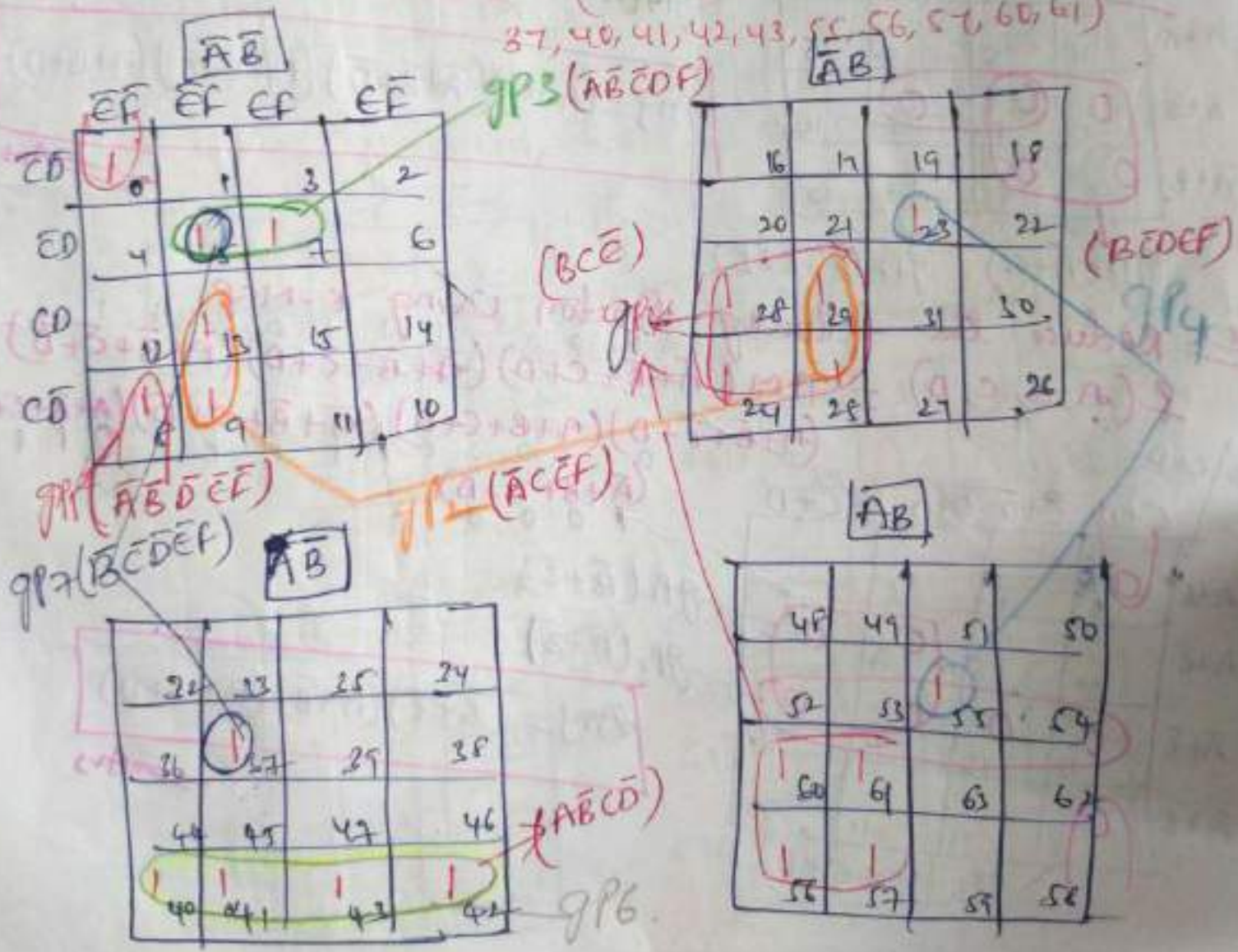
5 variable k-map

$f(A,B,C,D,E) = \Sigma m(0,2,5,7,13,15,16,20,21,23,28,29,31)$



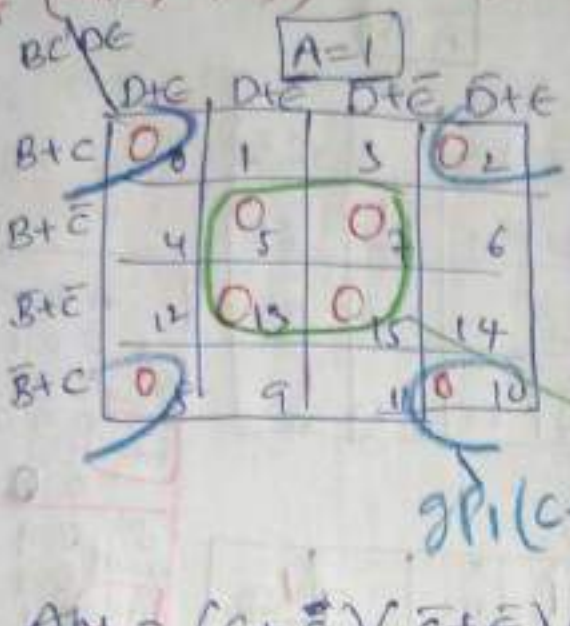
simplify the following 6variable Expression using K-Map.

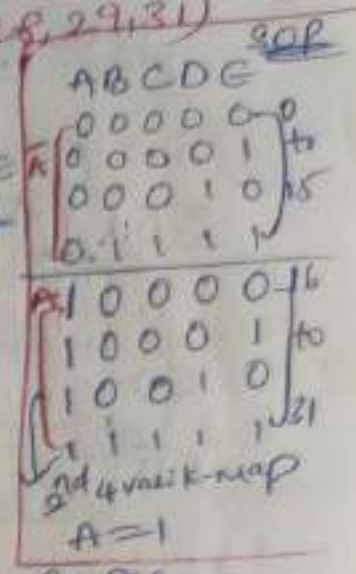
$f(A,B,C,D,E,F) = \Sigma m(0,5,7,8,9,13,23,24,25,28,29,37,40,41,42,43,55,56,57,60,61)$



5 variable K map for POS

$f(A,B,C,D,E) = \Pi M(0,2,5,7,8,10,13,15,16,18,20,21,23,24,26,29,31)$





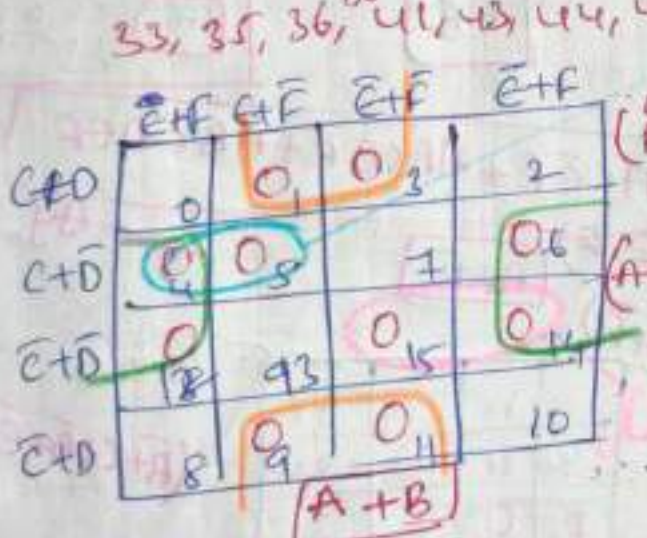
$gp_1(C+\bar{E})$ $gp_2(\bar{C}+\bar{E})$ $gp_3(\bar{A}+D+E)$

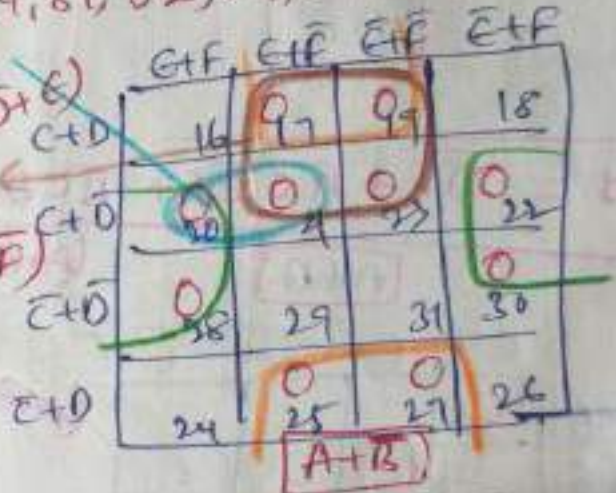
In POS
First K-map A=1
2nd K-map $\bar{A}=0$

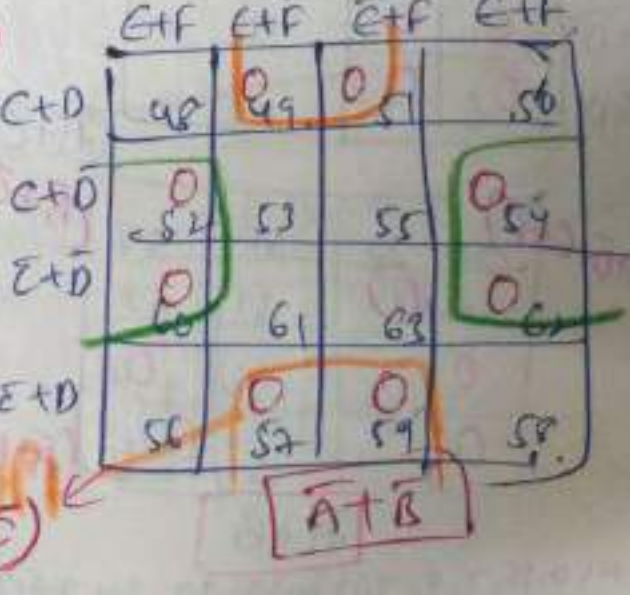
$f = (C+\bar{E})(\bar{C}+\bar{E})(\bar{A}+D+E)$

6 variable K-map for POS

$f = \Pi M(1,3,4,5,6,9,11,12,14,15,17,19,20,21,22,23,25,27,28,30,33,35,36,38,41,43,44,46,49,51,52,54,57,59,60,62)$



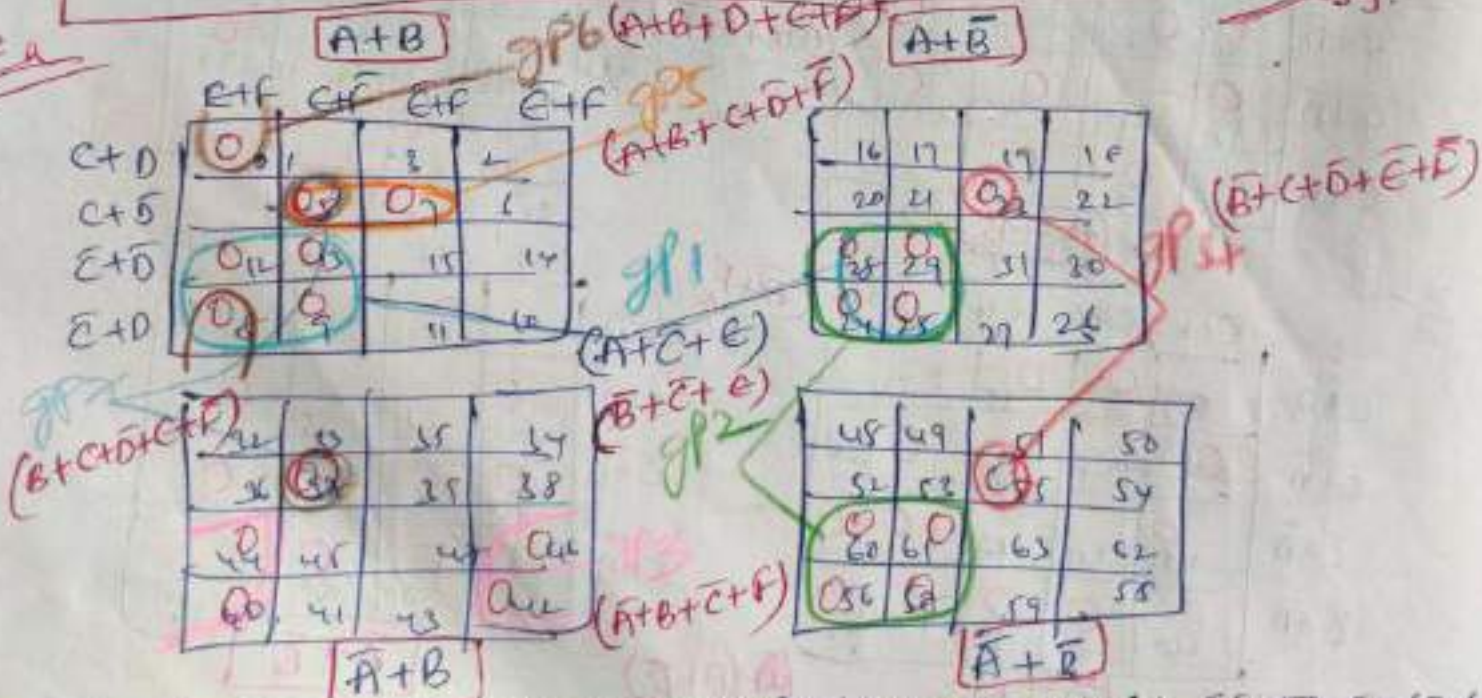




Ex 1 $f = \Sigma m(0,2,4,8,10,13,15,16,18,20,23,24,26,32,34,40,41,42,45,47,48,50,56,57,58,60,61)$

$Ay = \bar{D}F + \bar{A}\bar{C}\bar{E}\bar{F} + \bar{B}CDF + AC\bar{E}F + AB\bar{C}D\bar{E} + \bar{A}B\bar{C}DEF$ — Ay.

Ex 2



$f = \Pi M(0,5,7,8,9,12,13,23,24,25,26,29,37,40,[illegible],44,46,55,56,57,60,61)$

Draw backs of k-map Technique

1) It is a manual technique. we cannot use computers for k-map solution.
2) Minimization is extremely complicated no. of variable exceeds 6.
3) The simplification process depends heavily on human user's abilities.

Tabular Quine - Mc Cluskey method.

The minterms whose binary equivalent differ only in one place can be combined to reduce the minterms. This is fundamental principle of Quine - Mc Cluskey meth

Rules (or) steps

1) List all minterms in the binary form
2) Arrange them according to no. of ones
3) Compare each binary no with every term in the next higher category & if they differ only by one position put a check mark & copy the term in next column with in the position that they differ.
4) Apply the same process in the above step for the obtained column and continue these cycles until a single pass through cycle is having no further elimination of literals.
5) List all prime implicants
6) select the minimum no of prime implicants which must cover all the minterms

minterm	Binary	minterm	binary	minterm	binary
m_0	0000	0 0000 ✓	(0,2) 00-0 ✓	(0,2,8,10)	-0-0
m_2	0010	2 0010 ✓	(0,8) -000 ✓	(0,8,2,10)	-0-0
m_3	0011	8 1000 ✓	(2,3) 001- ✓	(2,3,6,7)	0-1-
m_6	0110	3 0011 ✓	(2,6) 0-10 ✓	(2,6,3,7)	0-1-
m_7	0111	6 0110 ✓	(2,10) -010 ✓		
m_8	1000	10 1010 ✓	(8,10) 10-0 ✓		
m_{10}	1010	12 1100	(8,12) 1-00		
m_{12}	1100	7 0111	(3,7) 0-11 ✓		
m_{13}	1101	13 1101	(6,7) 011- ✓		
			(12,13) 110-		

prime Implicants

(8,12)	1-00	$A\bar{C}\bar{D}$
(12,13)	110-	$AB\bar{C}$
(0,2,8,10)	-0-0	$\bar{B}\bar{D}$
(2,3,6,7)	0-1-	$\bar{A}C$

Selection Chart

	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{13}
$A\bar{C}\bar{D}$ (8,12)									⊙				⊙	
$AB\bar{C}$ (12,13)													⊙	⊙
$\bar{B}\bar{D}$ (0,2,8,10)	⊙		⊙						⊙		⊙			
$\bar{A}C$ (2,3,6,7)			⊙	⊙			⊙	⊙						

$Ay = \bar{A}C + \bar{B}\bar{D} + AB\bar{C}$ Ans.

Q. Simplify the following function by using Quine-McCluskey method.

$f(A,B,C,D) = \sum m(0,2,3,6,8,10,12)$

Minterm	Binary
m_2	0010
m_4	0100
m_5	0101
m_9	1001
m_{12}	1100
m_{13}	1101

minterm	Binary
(2)	0010
(4)	0100 ✓
(5)	0101 ✓
(9)	1001 ✓
(12)	1100 ✓
(13)	1101 ✓

minterm	Binary
(4,5)	010 — ✓
(4,12)	—100 ✓
(5,13)	—101 ✓
(9,13)	1—01
(12,13)	110 — ✓

minterm	Binary
(4,5,12,13)	—10—
(4,12,5,13)	—10—

Prime Implicants

ABCD

(2) 0010 → $\bar{A}\bar{B}C\bar{D}$

(9,13) 1—01 → $A\bar{C}D$

(4,5,12,13) —10 → $B\bar{C}$

Selection chart

Prime Implicants	m_2	m_4	m_5	m_9	m_{12}	m_{13}
(2) $\bar{A}\bar{B}C\bar{D}$	⊙ ✓					
(9,13) $A\bar{C}D$				⊙		⊙
(4,5,12,13) $B\bar{C}$		⊙	⊙		⊙	⊙

write the boolean Expression for Single dot

$Ay = \bar{A}\bar{B}C\bar{D} + B\bar{C} + A\bar{C}D$ Ans.

Minimize the Expression using Quine Mcclukey method

$Y = \sum m(2,4,5,9,12,13)$

(1)

m_2	0010
m_4	0100
m_5	0101
m_9	1001
m_{12}	1100
m_{13}	1101

$Y = \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + A\bar{B}C\bar{D} + \bar{A}\bar{B}CD$

$m_4 + m_5 + m_{12} + m_{13} + m_9 + m_2$

minterm	Binary
m0	00000
m1	00001
m8	01000
m9	01001
m11	01011
m15	01111
m24	11000
m29	11101
m30	11110
m31	11111

minterm	binary
(0)	00000 ✓
(1)	00001 ✓
(8)	01000 ✓
(9)	01001 ✓
(24)	11000 ✓
(11)	01011 ✓
(15)	01111 ✓
(29)	11101 ✓
(30)	11110 ✓
(31)	11111 ✓

minterm	binary
(0,1)	0000- ✓
(0,8)	0-000 ✓
(1,9)	0-001 ✓
(8,9)	01 00- ✓
(8,24)	-1000
(9,11)	010-1
(11,15)	01-11
(15,31)	-1111
(29,31)	111-1
(30,31)	1111-

minterm	Binary
(0,1,8,9)	0-00-
(0,8,1,9)	0-00-

Prime Implicants

(0,1,8,9) 0-00- ($\bar{A}\bar{C}\bar{D}$)
(8,24) -1000 ($B\bar{C}\bar{D}\bar{E}$)
(9,11) 010-1 ($\bar{A}B\bar{C}E$)
(11,15) 01-11 ($\bar{A}BDE$)
(15,31) -1111 ($BCDE$)
(29,31) 111-1 ($ABCE$)
(30,31) 1111- ($ABCD$)

Selection Chart

	m0	m1	m8	m9	m11	m15	m24	m29	m30	m31
(0,1,8,9) $\bar{A}\bar{C}\bar{D}$	⊙	⊙	⊙	⊙						
(8,24) $B\bar{C}\bar{D}\bar{E}$			⊙				⊙			
(9,11) $\bar{A}B\bar{C}E$				⊙	⊙					
(11,15) $\bar{A}BDE$					⊙	⊙				⊙
(15,31) $BCDE$						⊙				⊙
(29,31) $ABCE$								⊙		⊙
(30,31) $ABCD$									⊙	⊙

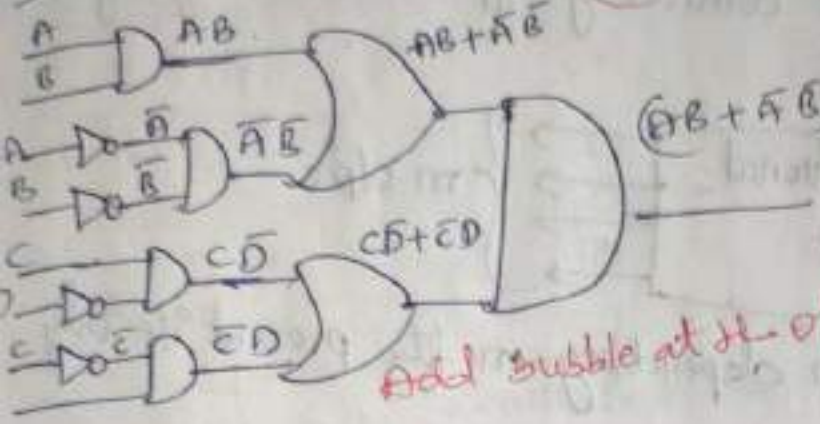
$A_M = \bar{A}\bar{C}\bar{D} + B\bar{C}\bar{D}\bar{E} + ABCE + ABCD$ A_M

(8) Simplify the following boolean expression using QM method & realize by using NAND gates.
$f = \Sigma m(0,1,8,9,24,11,15,29,30) + d(8,11,31)$

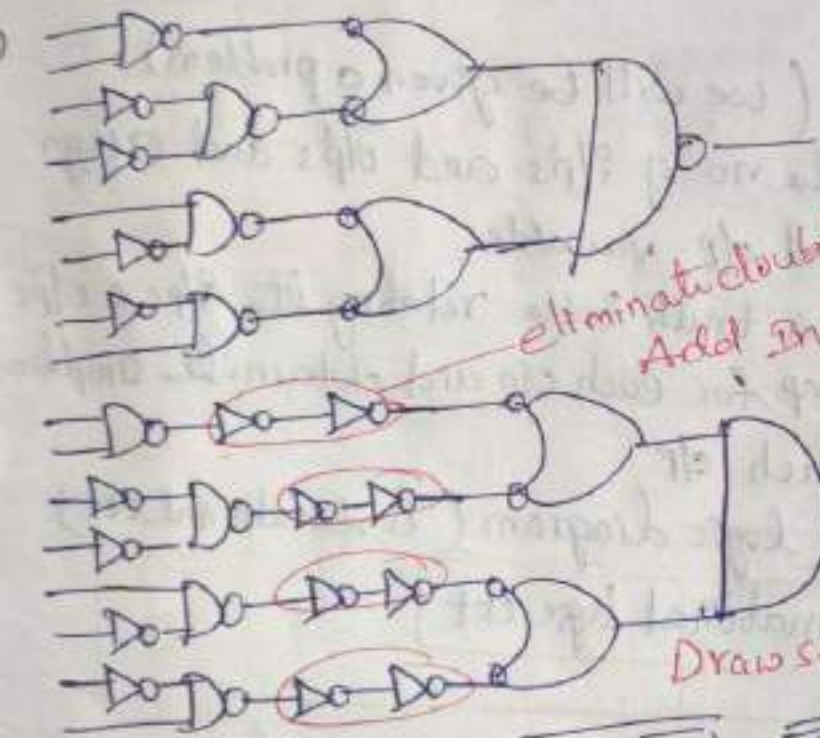
Q8 Construct logic diagram using only one two i/p NAND gate to implement the following exp
$(AB+\bar{A}\bar{B})(C\bar{D}+\bar{C}D)$.

Q9 Discuss about OR-OR implementation

$(AB+\bar{A}\bar{B})(C\bar{D}+\bar{C}D)$ → by using NAND & AL



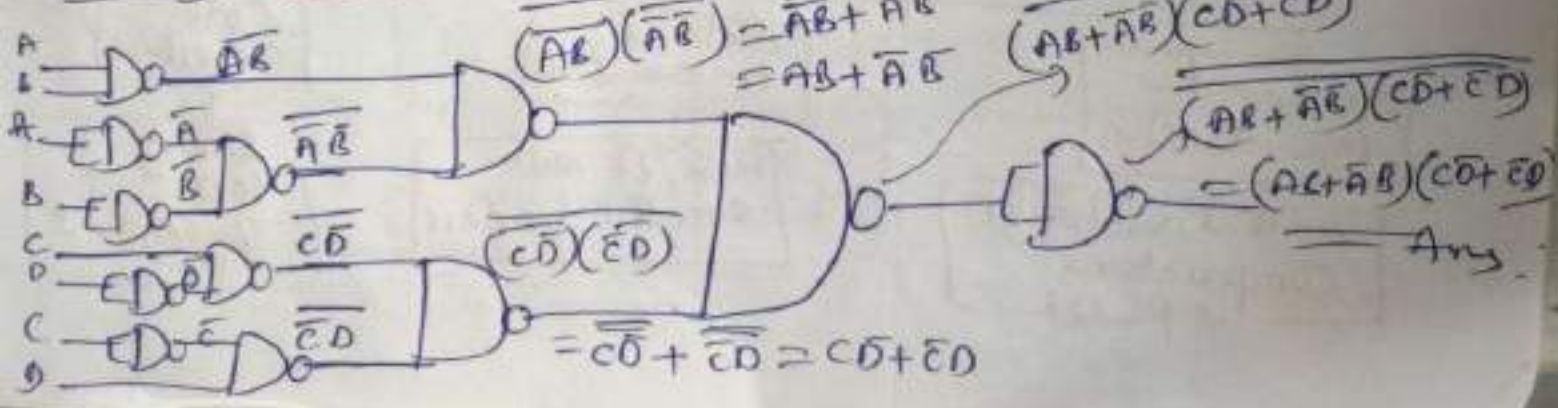
Add bubble at AL o/p side & AND & i/p side of OR.

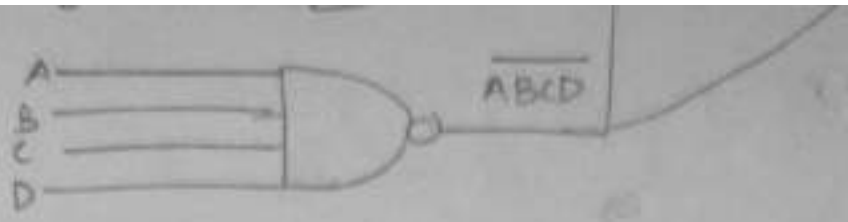


eliminate double Inversion

Add Inverter for each bubble

Draw single i/p NAND in place of NOT & replace bubbled OR by NAND





$= (\overline{AC\bar{D}})(\overline{BC\bar{D}\bar{E}}) + (\overline{ABCE})(\overline{ABCD})$

$= \overline{\overline{A}CD} + \overline{B\bar{C}\bar{D}\bar{E}} + \overline{AB\bar{C}E} + \overline{ABC}$

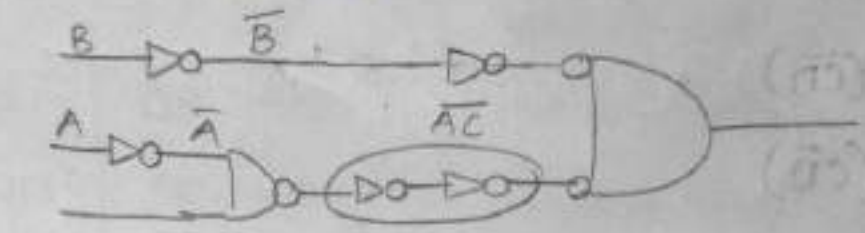
$= A\bar{C}\bar{D} + B\bar{C}\bar{D}\bar{E} + AB\bar{C}E + ABC$

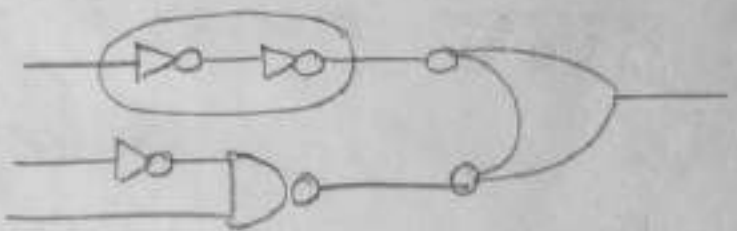
Simplify the fun by using k-map & Implement (or) realize the expusing NAND gates.

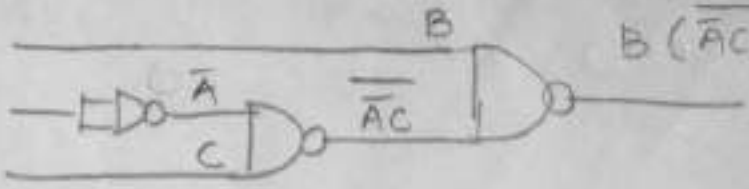
$y = \Sigma m(0,1,3,4,5)$

0	1	3	2
4	5	7	6

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







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

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

Two level Implementation of Boolean fun

If the Boolean fun is in sum of products form a product of sums then the fun can be implemented in two-levels.

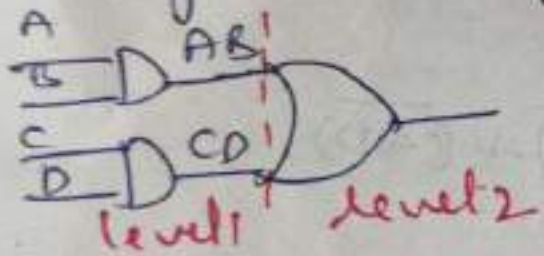
Level 1	Level 2
AND	OR
OR	AND
NAND	NAND
NOR	NOR.

Sum of product forms ⓑ only NAND

ⓐ Ex) AND, OR

$F = AB + CD$

Using AND, OR gates

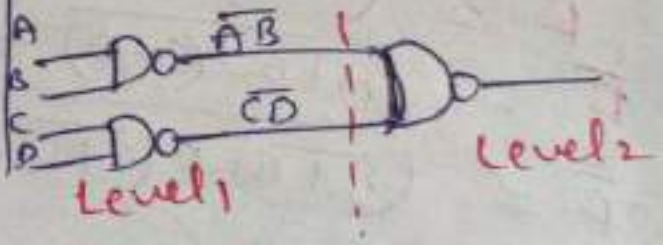


Ex $F = AB + CD$

Using only NAND gate

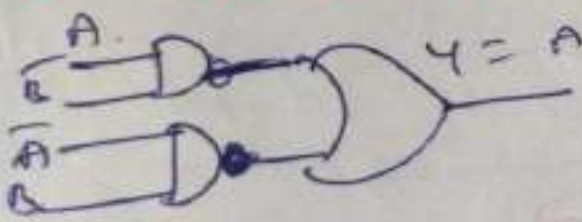
$\bar{F} = \overline{AB + CD} = \bar{A}\bar{B} \cdot \bar{C}\bar{D}$

$\bar{\bar{F}} = \overline{\bar{A}\bar{B} \cdot \bar{C}\bar{D}} = A$



Using AND, OR gates

ⓒ $F = A\bar{B} + \bar{A}B$



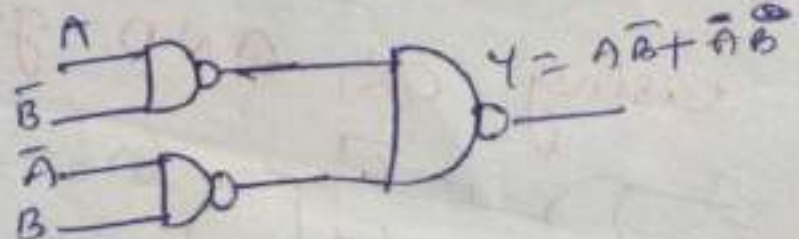
d) $Y = A\bar{B} + \bar{A}B$

Using NAND

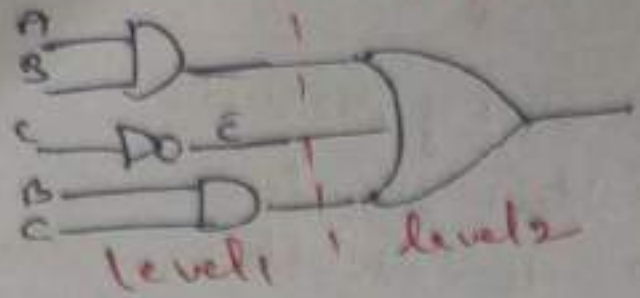
$\bar{Y} = A\bar{B} + \bar{A}B$

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

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



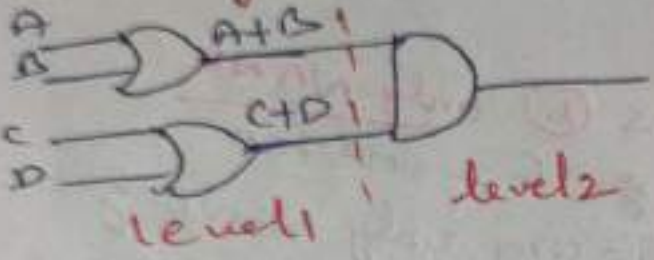
$F = AB + \bar{C} + BC$



product of sums form

Ex $f = (A+B)(C+D)$

a) Using OR, AND gates

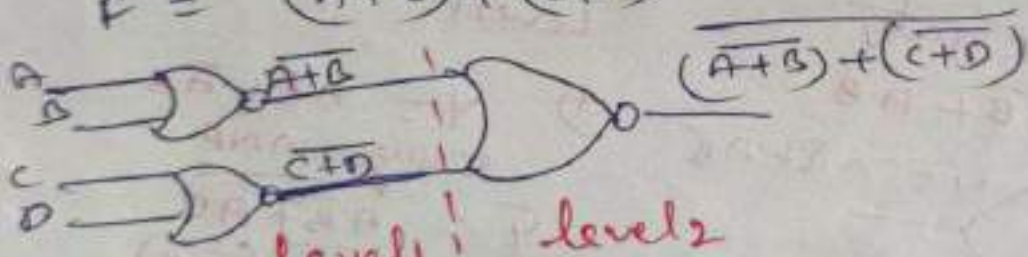


b) Using only NOR gates

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

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

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



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

Using OR, AND gates

(5) Using only NOR gates.

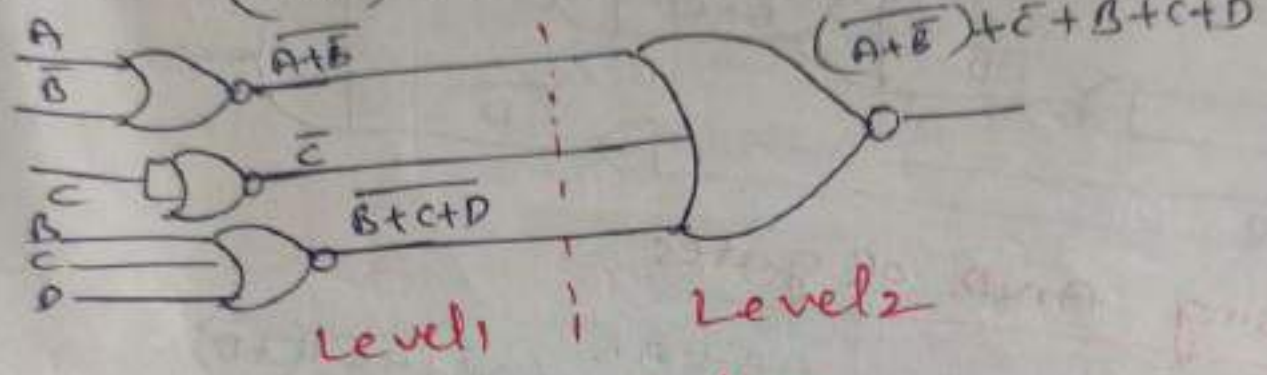
$F = (A+\bar{B}).C\,(B+C+D)$

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

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

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

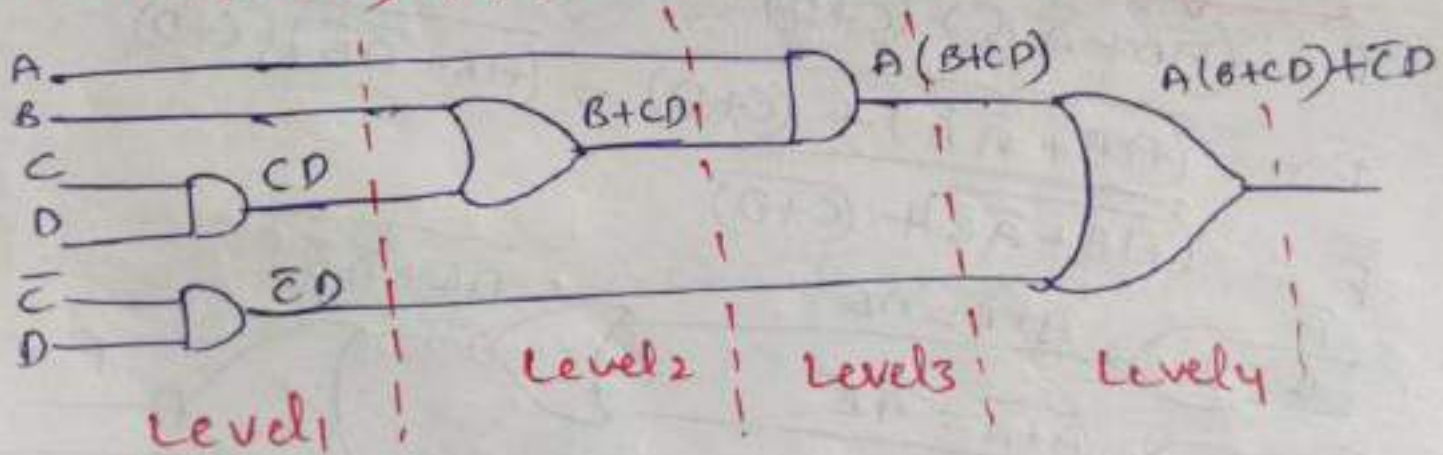
$F = \overline{\overline{(A+\bar{B})} + \bar{C} + \overline{B+C+D}}$



Multilevel Implementation

The Levels are three more than three is called Multilevel Implementation.

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



Using NAND.

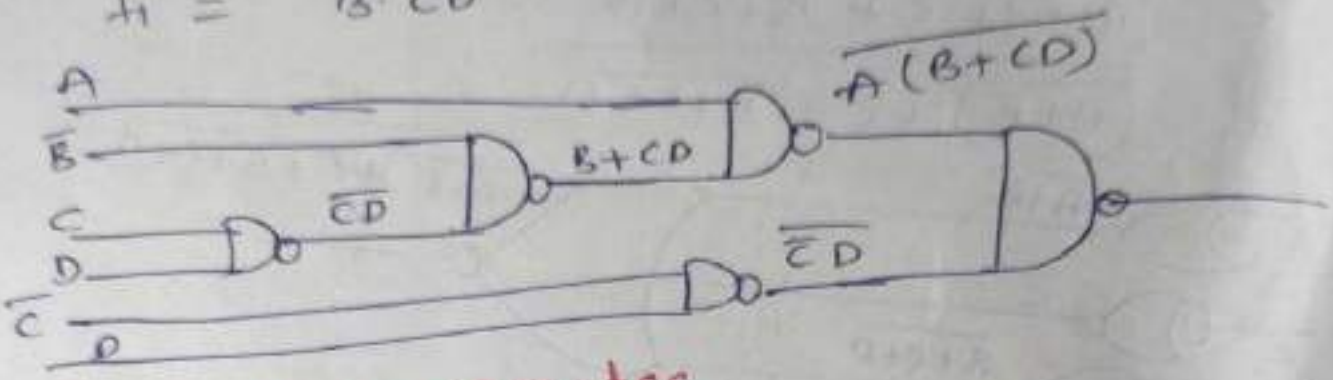
$$F = A(B+CD) + \bar{C}D \Rightarrow \bar{F} = \overline{A(B+CD) + \bar{C}D}$$

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

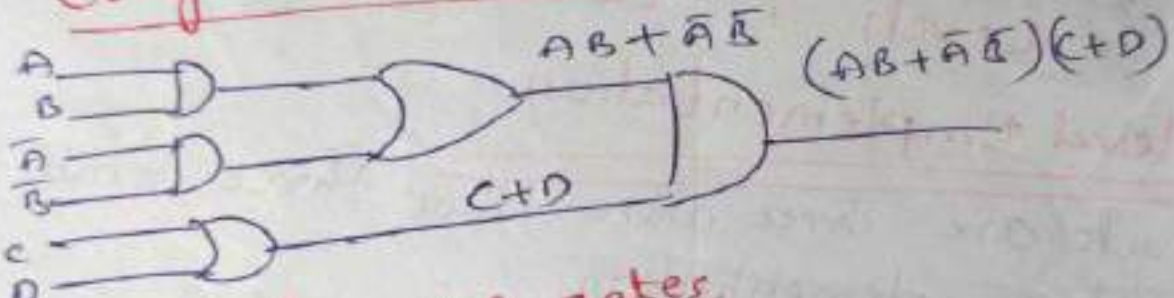
$$\bar{\bar{F}} = \overline{\overline{A(B+CD)} \cdot \overline{\bar{C}D}}$$

$$f_1 = B+CD$$

$$\bar{f}_1 = \overline{B \cdot CD}$$



Using AND, OR gates

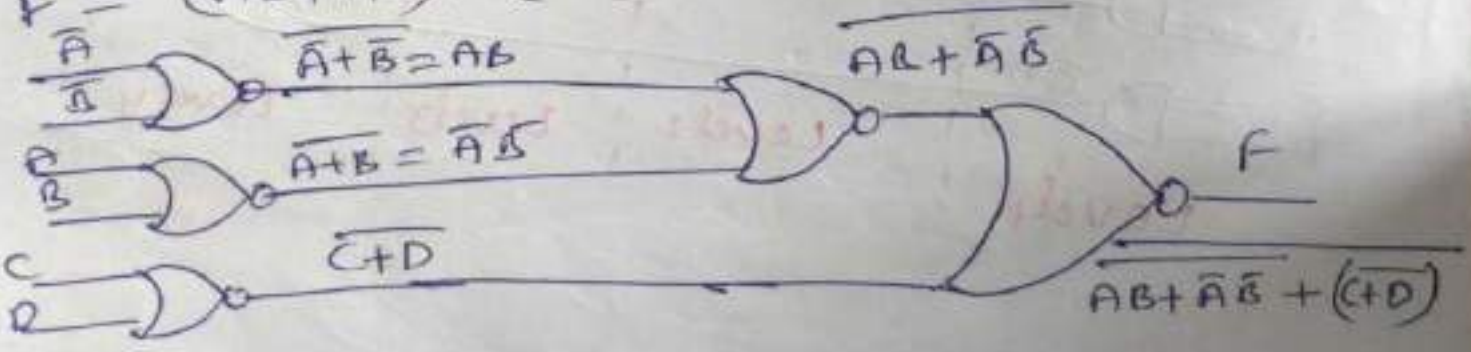


Using only NOR gates.

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

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

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



Degenerate and Non degenerate forms of Two-level implementations.

Two level Combinations of gates:

AND, OR, NAND, NOR

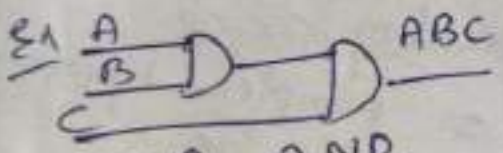
AND-OR	OR-OR	NAND-OR	NOR-OR
AND-AND	OR-AND	NAND-AND	NOR-NAND
AND-NAND	OR-NAND	NAND-NAND	NOR-NOR.
AND-NOR	OR-NOR	NAND-NOR	

16- Combinations.

Degenerate forms. If Two level implementation is degenerate to single operation.

Eight combinations are degenerate forms.

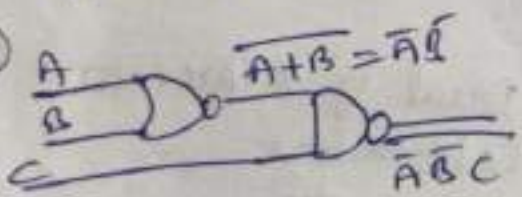
AND-AND	OR-OR	NAND-OR	NOR-AND
AND-NAND	OR-NOR	NAND-NOR	NOR-NAND.

Ex: 

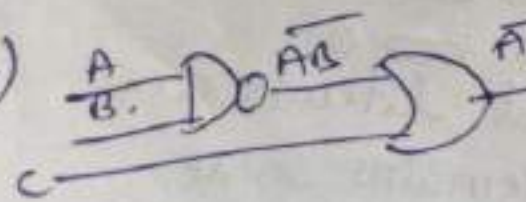
AND-AND.

② 

OR-NOR. [and fun AND.

③ 

= A+B+C̄ [Single op-OR]

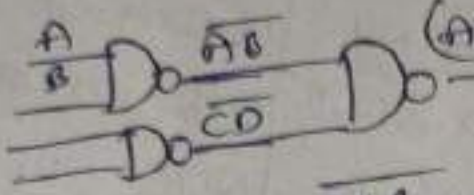
④ 

⇒ Ā + B̄ + C (OR) function.

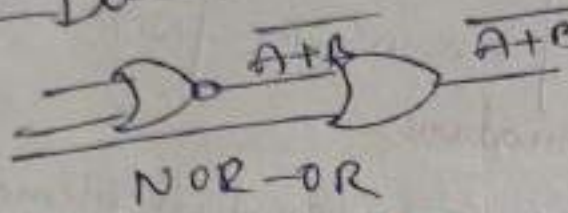
Non-degenerate forms

These forms produce an implementation in sum of products (or) product of Sums forms

AND-OR	OR-AND	NAND-NAND	NOR-OR
AND-NOR	OR-NAND	NAND-AND	NOR-NOR

Ex: 

$\overline{AB}+\overline{CD}$ (SOP) [NAND-NAND]

Ex: 

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

NOR-OR

Duality b/w gates

AND $\xrightarrow{\text{Duality}}$ OR
OR $\xrightarrow{\text{D}}$ AND
NOR $\longrightarrow$ NAND
NAND $\longrightarrow$ NOR
Ex-OR $\longrightarrow$ Ex-NOR.

- AND → OR
- OR → AND
- 1 → 0
- 0 → 1

Ex: Ex-OR $Y = A\overline{B} + \overline{A}B$

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

$Y_D = (A+\overline{B}).(\overline{A}+B)$ Dual of Ex-OR
$= AB + \overline{A}\overline{B}$ = Complement of Ex-OR
= Ex-NOR

Ex: Ex-NOR $Y = AB + \overline{A}\overline{B}$
$Y_D = (A+B).(\overline{A}+\overline{B})$
$= A\overline{B} + \overline{A}B$

Dual of Expression
$(\cdot) \Leftrightarrow (+)$
AND $\Leftrightarrow$ OR
1 $\Leftrightarrow$ 0
But literals are remains same.

Applications of EX-OR & EX-NOR functions (4)

EX-OR

Two variable

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

- Arithmetic gate
- → staircase gate
- → inequality detector
- → Anticoincedence

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

$= A \oplus B$

EX-NOR

another name equality detector

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

Inclusive OR

Coincidence gate

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

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

Properties of EX-OR function

$x \oplus 0 = x$

$x \oplus 1 = \bar{x}$

$x \oplus x = 0$

$x \oplus \bar{x} = 1$

$x \oplus \bar{y} = \bar{x} \oplus y = \overline{x \oplus y} = x \odot y$.

Properties of EX-NOR.

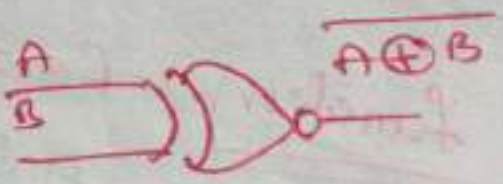
$x \odot 0 = \bar{x}$

$x \odot 1 = x$

$x \odot x = 1$

$x \odot \bar{x} = 0$

$x \odot \bar{y} = \bar{x} \odot y = \overline{x \odot y} = x \oplus y$



Odd function — 3 variable.

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

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

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

$\Rightarrow A\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC + \bar{A}\bar{B}C$
(100, 010, 111, 001)

$\Rightarrow \Sigma(1, 2, 4, 7)$

	A	B	Y	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Total no's 1's is 1 — Total no's 1's is 2 — Total no of 1's is 4

Whenever odd no of 1's function value is 1 (o/p)

Even function — $F_e = \Sigma(0, 3, 5, 6)$

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

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

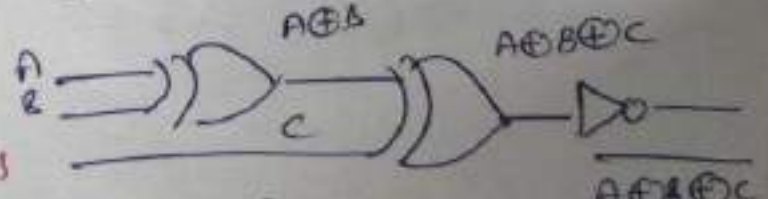
$= \overline{A \oplus B}\,\bar{C} + (A \oplus B)C$

$\bar{x}\bar{y} + xy \Rightarrow x \odot y$

$\Rightarrow [(A \oplus B) \odot C]$

	A	B	C	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

Total no's of 1's is even.

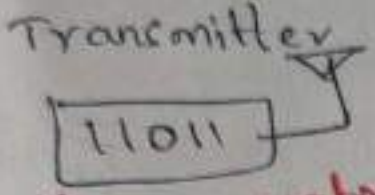


(OR)

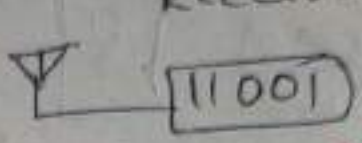
Applications

1) Ex-OR gate is used in Arithmetic ckts in digital systems.
2) Ex-OR & Ex-NOR used in error detection and correction ckts.

Parity Generator & parity checker.



Parity generator



Parity checker

Even parity generator at the Tx side

A	B	C	P
	Message		Parity bit
0	0	0	0
1 0	0	1 ←	1
2 0	1	0 ←	1
0	1	1	0
4 1	0	0 ←	1
1	0	1	0
1	1	0	0
7 1	1	1 ←	1

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

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

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

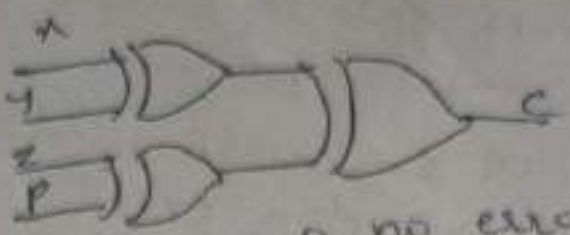
$\bar{A}(B \oplus C) + A(\overline{B \oplus C})$

$A \oplus B \oplus C$

Even parity checker

X	Y	Z	P	Checker
	Received bits			
0	0	0	0	0
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
	0	0	0	1

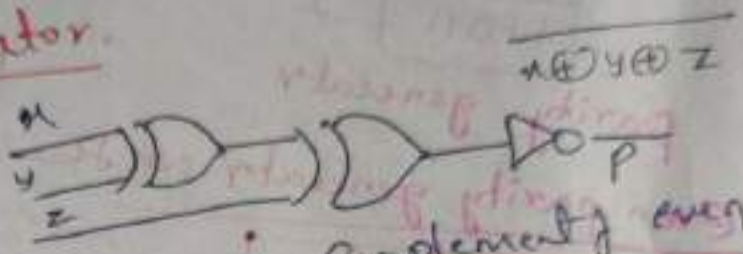
1 0 0 1	0
1 0 1 0	0
1 0 1 1	1
1 1 0 0	0
1 1 0 1	1
1 1 1 0	1
1 1 1 1	0



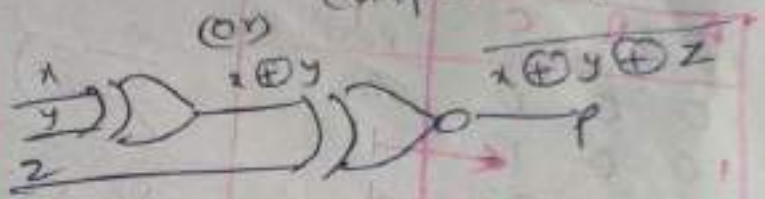
when c = 0 no error
c = 1 error is their.

Odd parity generator

x	y	z	P
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0



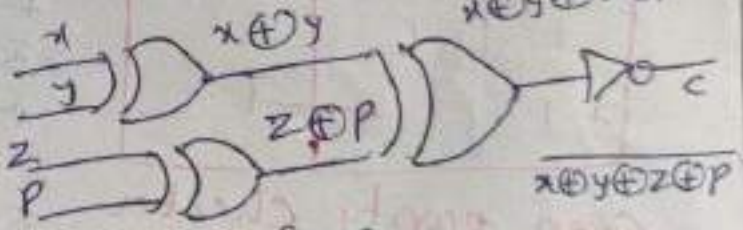
Complement of even



Odd parity checker

x	y	z	P	c
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

even fun (or) odd parity checker



(or)

It can detect only '1' bit of error.

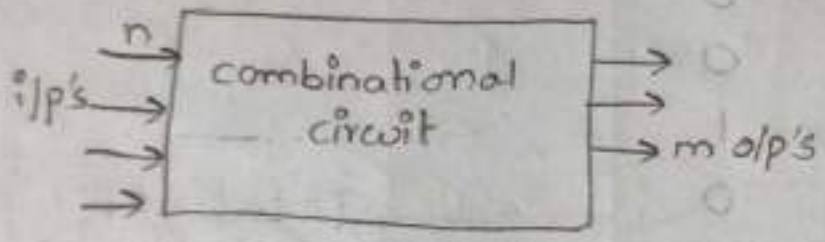
UNIT-3

COMBINATIONAL LOGIC DESIGN

→ A circuit which consists of logic gate to produce a specify output for specified i/p variable with no storage involved is called a combinational logic

→ A combinations circuit consists of i/p variable and logic gates and o/p variable.

→ In combinational circuit o/p depending on the present i/p only.



Design procedure:

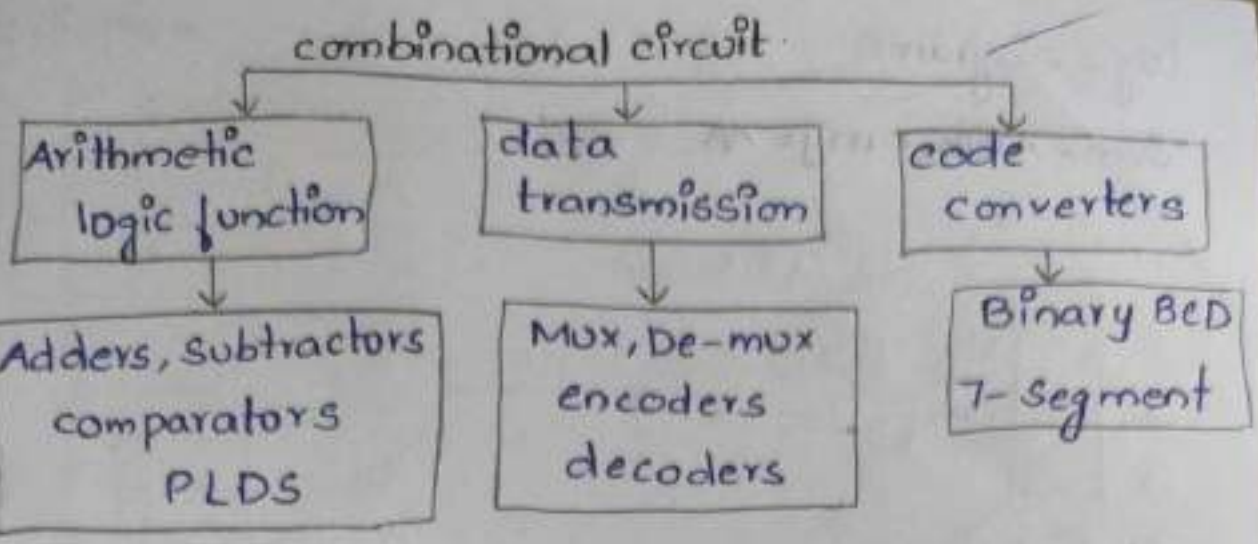
→ The problem is stated (question will be given)

→ Then we determine the no of inputs and outputs and assign letter symbols to input and output variables.

→ After that we write a truth table relating the inputs and outputs.

→ then draw the k-map and write for each output and obtain the simplified boolean expression for each output

Draw the logic diagram or combinational circuit

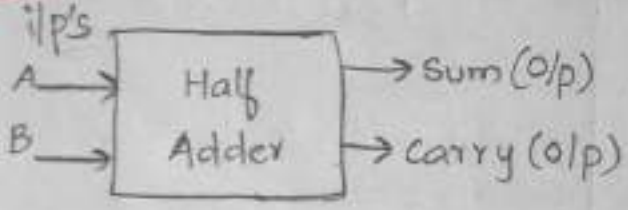


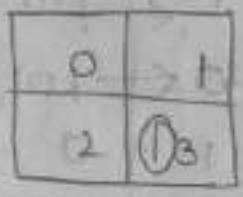
Adders:

The basic circuit includes the digital circuits which perform arithmetic operations.

→ The basic operation is the addition of two binary digits.

Half Adders:





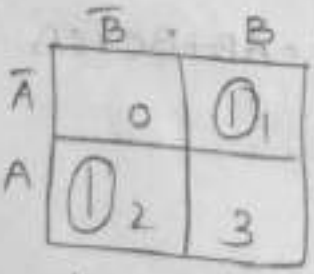
k-map for carry

Carry = AB

Truth Table:

A	B	Sum	Carry
i/p's		o/p	o/p
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2-variable k-map



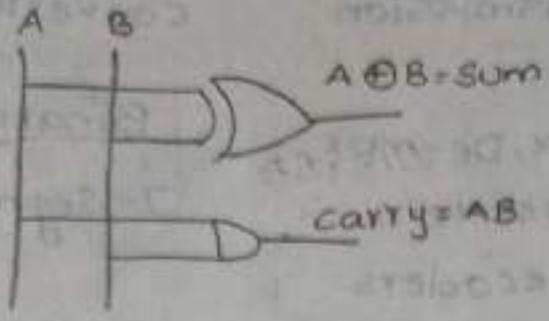
K-map for Sum

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

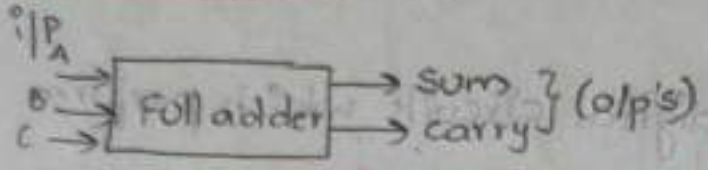
$Sum = A \oplus B$

logic diagram:

Sum = $A \oplus B$, carry = AB



Full Adder:



truth table

A	B	C	Sum	carry
i/p's			o/p	o/p
0	0	0	0	0
0	0	1	1 (1)	0
0	1	0	1 (2)	0
0	1	1	0	1
1	0	0	1 (4)	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1 (7)	1

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	3	2
A	4	5	7	6

k-map for Sum

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

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

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

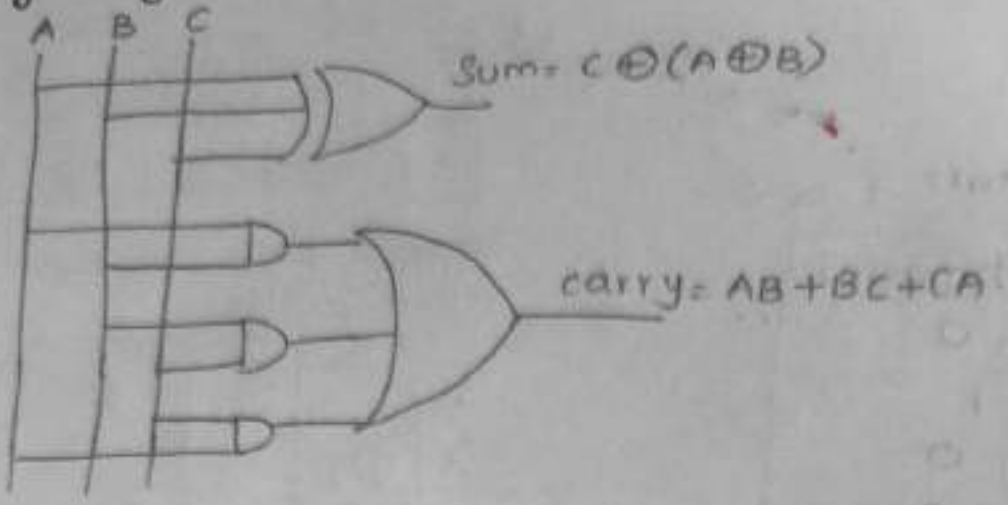
$Sum = C \oplus (A \oplus B)$

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	3	2
A	4	5	7	6

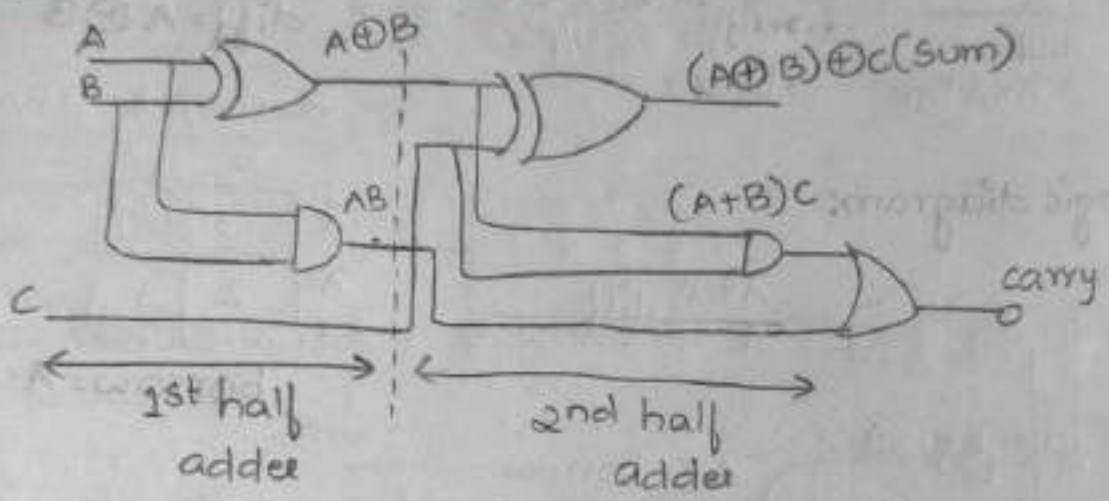
$carry = AC + BC + AB$

$carry = AB + BC + CA$

logic diagram:



Implementation of a Full Adder with 2 two Half Adders:



$carry = AB + AC + BC$

$AB + AC + BC = AB + C(A \oplus B)$

proof:

$AB + AC + BC(A + \bar{A})$

$AB + AC + ABC + \bar{A}BC$

$AB(1 + C) + AC + \bar{A}BC$

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

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

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

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

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

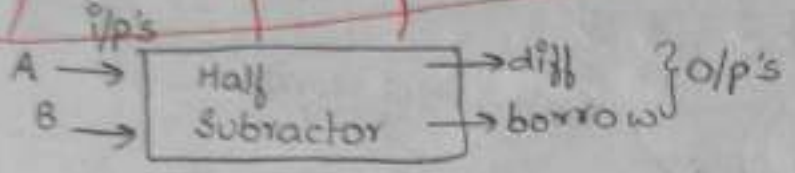
$$AB + AC + BC = AB + C(A \oplus B)$$

Subtractors:

truth table

Half Subtractor:

A	B	diff	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



K-map diff:

	$\bar{B}$	B
$\bar{A}$	0	1
A	2	3

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

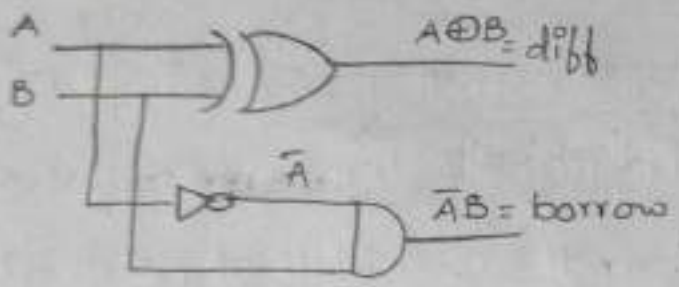
$diff = A \oplus B$

K-map borrow:

	$\bar{B}$	B
$\bar{A}$	0	1
A	2	3

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

Logic diagram:



Full subtractor:

A	B	C	diff	borrow
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	1
1	1	1	1	1

K-map for diff:

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	1	3	2
A	4	5	7	6

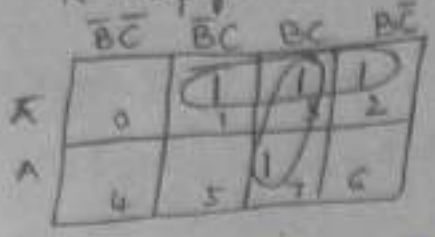
$diff = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$

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

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

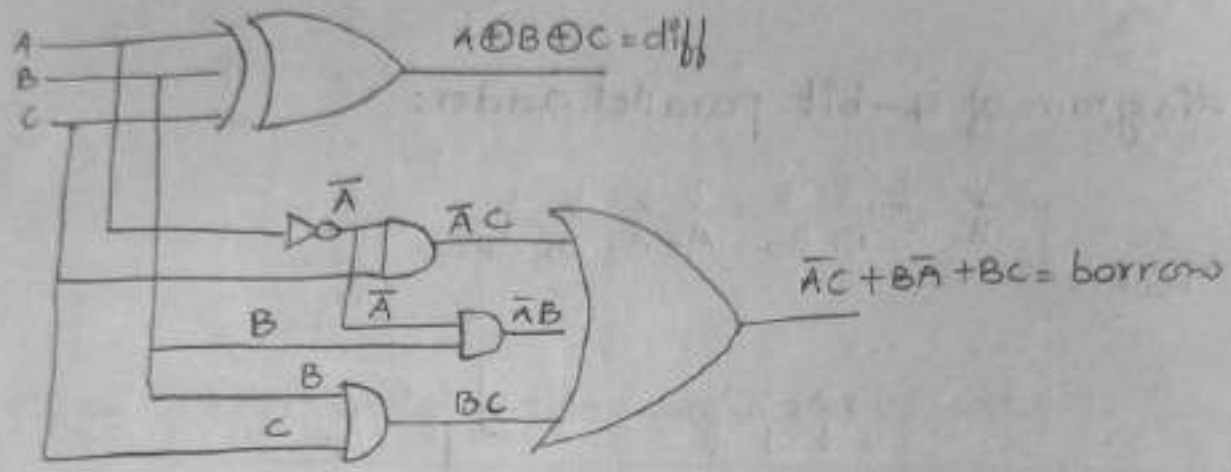
$diff = C \oplus (A \oplus B)$

K-map for borrow

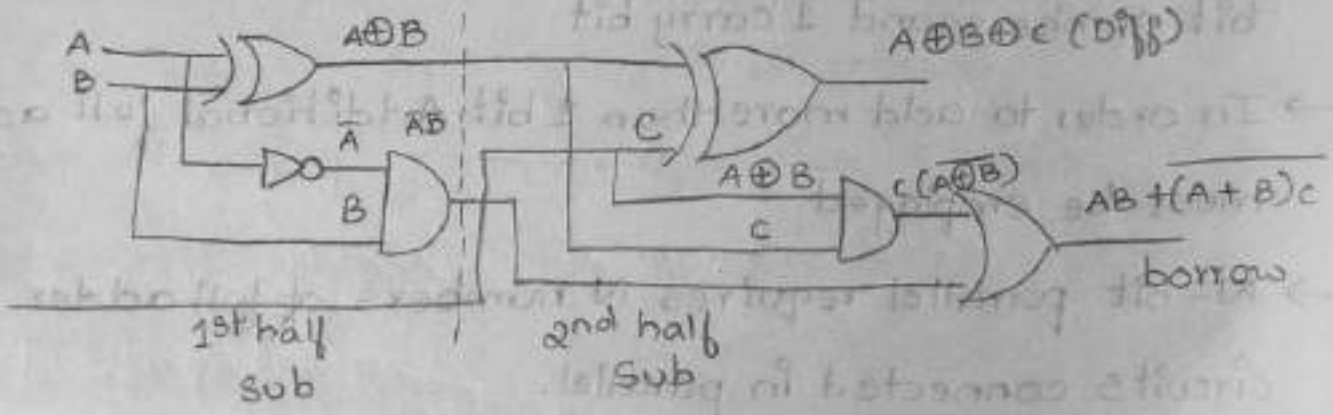


borrow = $\bar{A}C + \bar{A}B + BC$

Logic diagram:



Implementation of a full subtractor with a half subtractor



Find the comp of the following expression

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

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

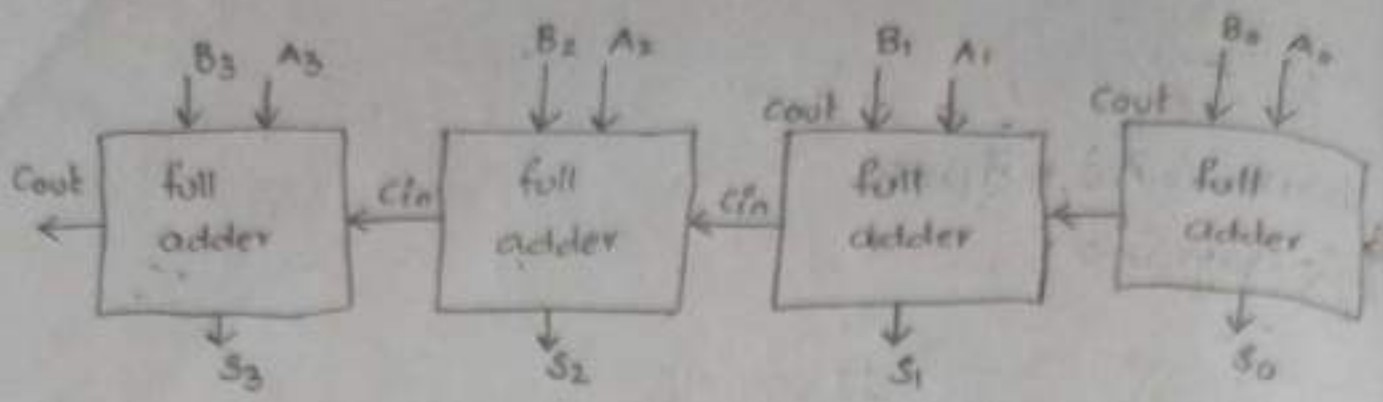
$$\bar{B}\bar{C}D + [\bar{B}\bar{C}\bar{D}] + \bar{B}C\bar{D}E$$

$$\bar{B}\bar{C}[D+\bar{D}] + \bar{B}C\bar{D}E$$

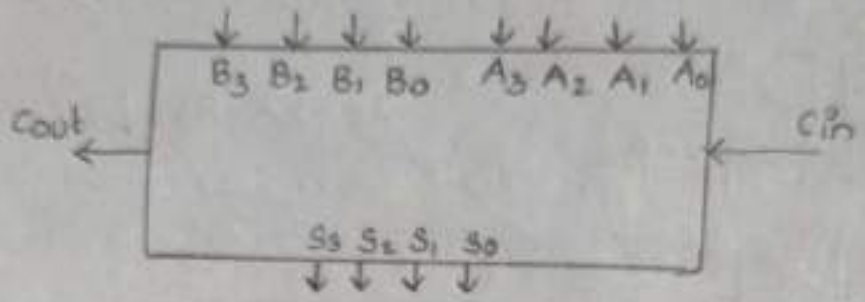
$$\bar{B}\bar{C} + \bar{B}C\bar{D}E$$

$$\bar{B}\bar{C}[1+\bar{D}E]$$

Binary adder/parallel adder:



logic diagram of 4-bit parallel adder:

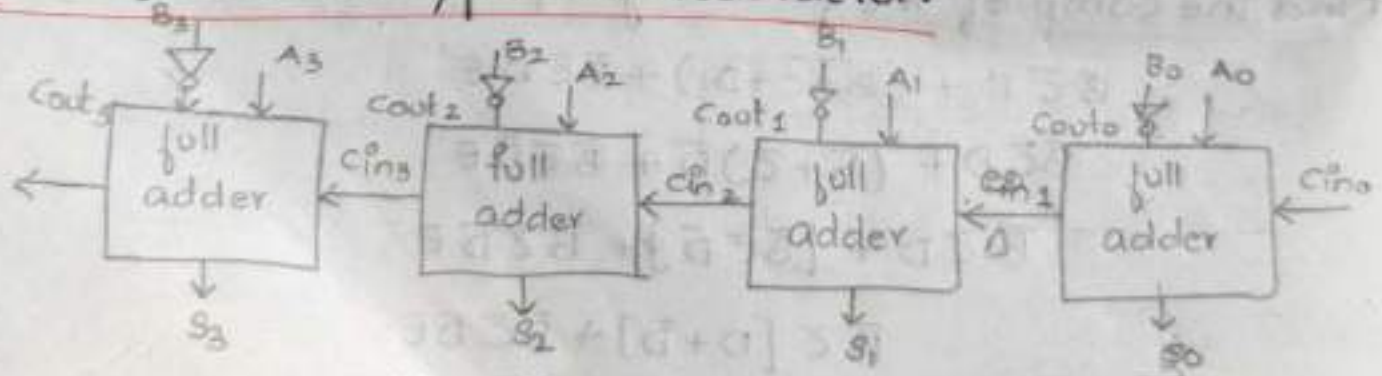


→ It is observed that a single subtractor will add 2 one bit numbers and 1 carry bit.

→ In order to add more than 1 bit Additional full adders must be employed.

→ N-bit parallel requires N numbers of full adder circuits connected in parallel

Binary Subtractor/parallel subtractor:



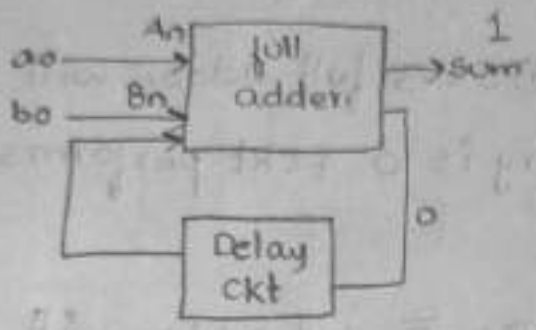
→ In general subtraction of binary numbers done by complements i.e

m-N = M+2's complement of N

2's complement = 1 + 1's comp

→ Here is complement can be implemented with inventors (NOT gate) and 1 can be added to the sum with input carry

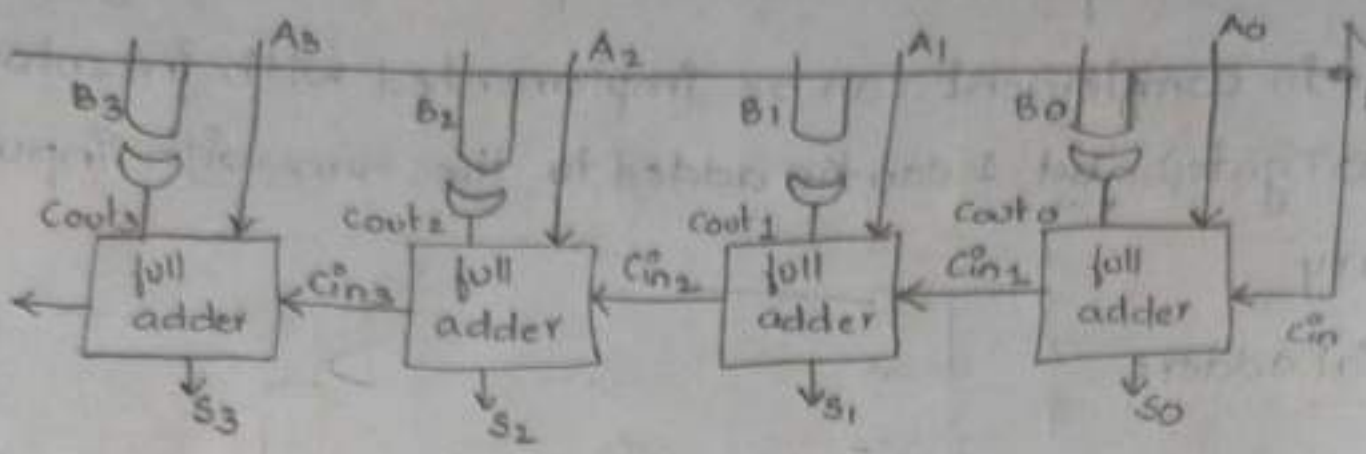
Serial adder:



Difference b/w serial adder and parallel adder:

Serial adder	parallel adder
• It adds only one bit at a time i.e one by one entering	• All bits are added simultaneously
• operation is slow	• operation is faster
• It uses shift register	• It uses registers with parallel load capacity
• It requires only one full adder circuit	• N no. of bits requires • N no. of full adders
• This is sequential circuit	• This is combinal circuit

parallel Adder / Subtractor:



4-Bit subtractor

1. When M = 0 we have $B \oplus 0 = B$ full adder will receive the value of B, the i/p carry is 0, & ckt performs A plus B A+B

2. When M = 1, we have $B \oplus 1 = \overline{B}$, full adder will receive complemented B & 1 is added through i/p carry. The ckt performs A + 2's complement of B i.e A − B

Binary Multiplier:

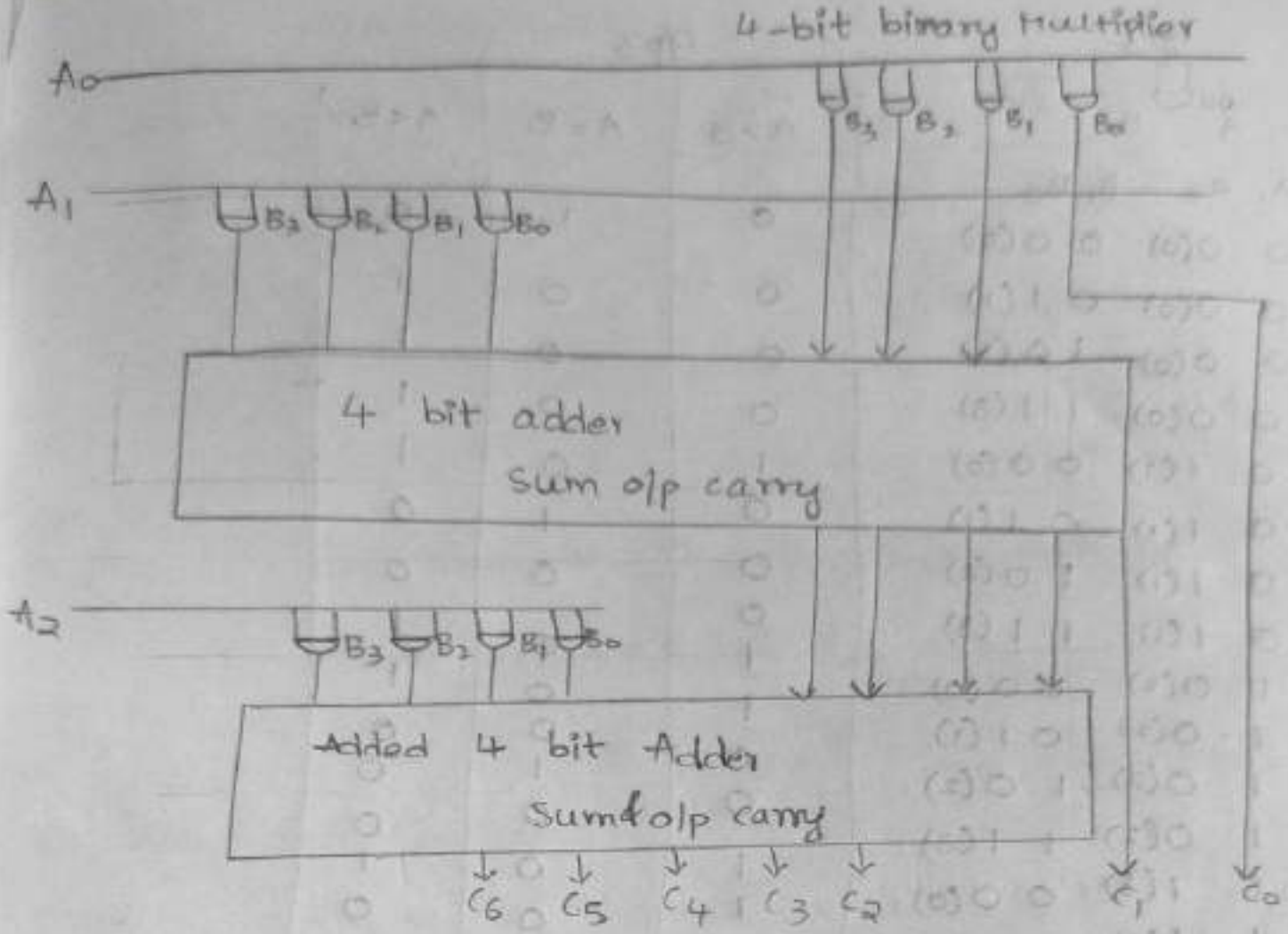
$$\begin{array}{cccccc} & & A_3 & A_2 & A_1 & A_0 \\ \times & & B_3 & B_2 & B_1 & B_0 \\ \hline & A_3B_0 & A_2B_0 & A_1B_0 & A_0B_0 & \end{array}$$

$A_3B_1 \quad A_2B_1 \quad A_2B_1 \quad A_2B_0$

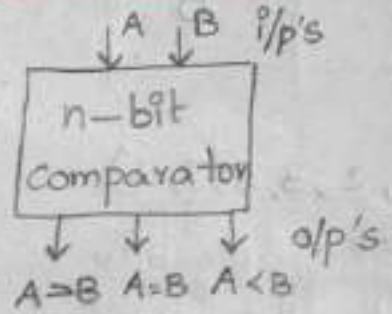
$A_3B_2 \quad A_2B_2 \quad A_1B_2 \quad A_0B_2$

$+ \quad A_3B_3 \quad A_2B_3 \quad A_1B_3 \quad A_0B_3 \quad \times \quad \times \quad \times$

P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

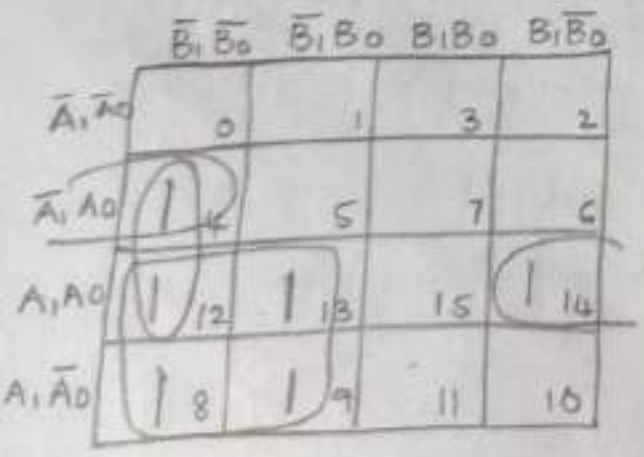


Magnitude comparator:



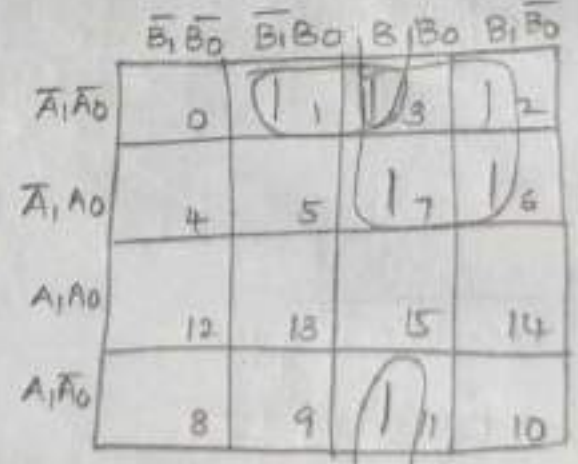
i/p's				o/p's		
A		B		A>B	A=B	A<B
A_1	A_0	B_1	B_0			
0	0 (0)	0	0 (0)	0	1	0
0	0 (0)	0	1 (1)	0	0	1
0	0 (0)	1	0 (2)	0	0	1
0	0 (0)	1	1 (3)	0	0	1
0	1 (1)	0	0 (0)	1	0	0
0	1 (1)	0	1 (1)	0	1	0
0	1 (1)	1	0 (2)	0	0	0
0	1 (1)	1	1 (3)	0	0	1
1	0 (2)	0	0 (0)	1	0	1
1	0 (2)	0	1 (1)	1	0	0
1	0 (2)	1	0 (2)	0	1	0
1	0 (2)	1	1 (3)	0	0	0
1	1 (3)	0	0 (0)	1	0	1
1	1 (3)	0	1 (1)	1	0	0
1	1 (3)	1	0 (2)	1	0	0
1	1 (3)	1	1 (3)	0	1	0

$A>B\ (4, 8, 9, 12, 13, 14)$



$A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0$

$A<B\ (1, 2, 3, 6, 7, 11)$



$\bar{A}_1B_1 + \bar{A}_1B_0 + \bar{A}_0B_1B_0$

$A=B\ (0,5,10,15)$

	$\bar{B}_1\bar{B}_0$	$\bar{B}_1B_0$	B_1B_0	$B_1\bar{B}_0$
$\bar{A}_1\bar{A}_0$	(1) 0	1	3	2
$\bar{A}_1A_0$	4	(1) 5	7	6
A_1A_0	12	13	(1) 15	14
$A_1\bar{A}_0$	8	9	11	(1) 10

$$\bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0 + A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0$$

$$= \bar{A}_0\bar{B}_0(\bar{A}_1\bar{B}_1 + A_1B_1) + A_0B_0(\bar{A}_1\bar{B}_1 + A_1B_1)$$

$$= [\bar{A}_1\bar{B}_1 + A_1B_1][\bar{A}_0\bar{B}_0 + A_0B_0]$$

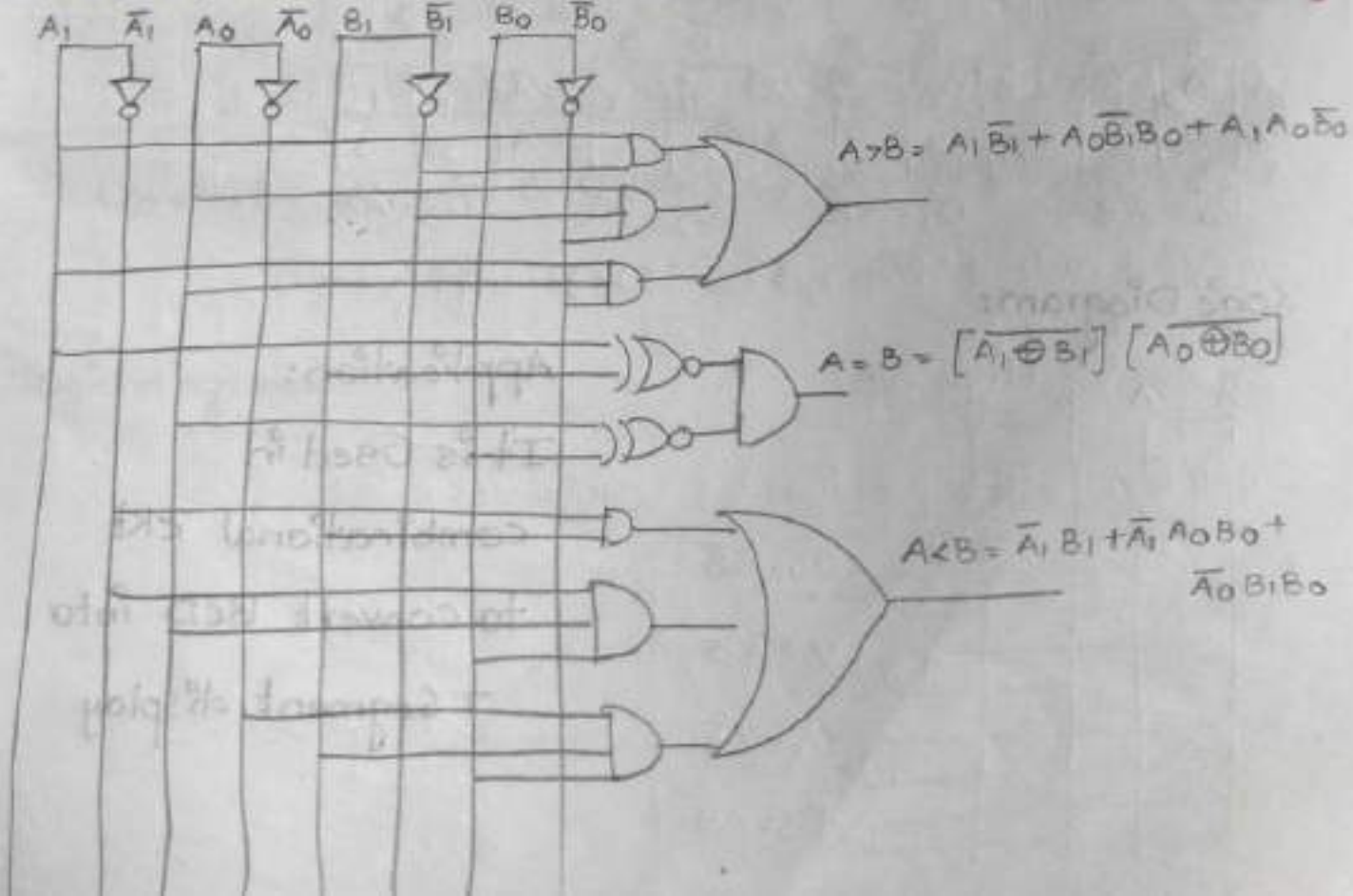
$$= [\overline{A \oplus B_1}][\overline{A_0 \oplus B_0}]$$

Draw the logic diagram:

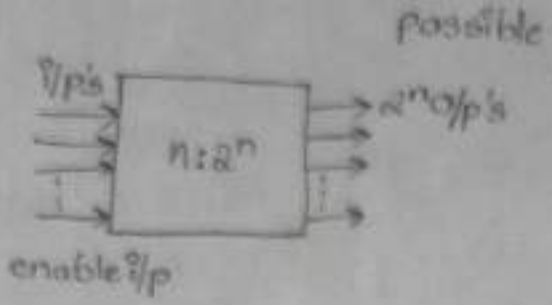
1. $A>B = A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0$

2. $A=B = [\overline{A_1 \oplus B_1}][\overline{A_0 \oplus B_0}]$

3. $A<=B = \bar{A}_1B_1 + \bar{A}_1A_0B_0 + \bar{A}_0B_1B_0$



Decoders:



2 to 4 decoder:

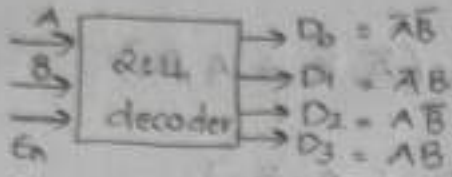
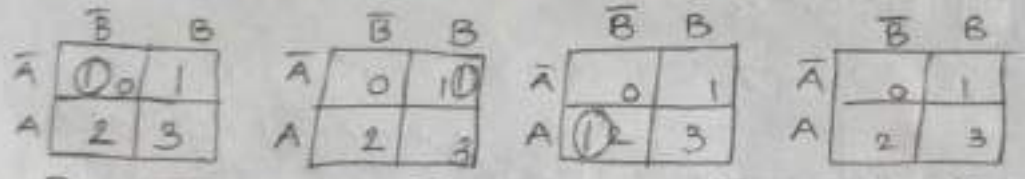
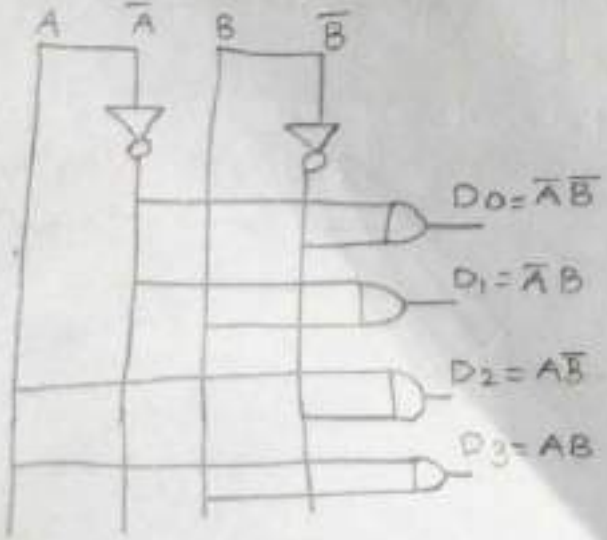


Table:

En	i/p's		o/p's			
	A	B	D_0	D_1	D_2	D_3
1	0	0	1(0)	0	0	0
1	0	1	0	1(1)	0	0
1	1	0	0	0	1(2)	0
1	1	1	0	0	0	1(3)



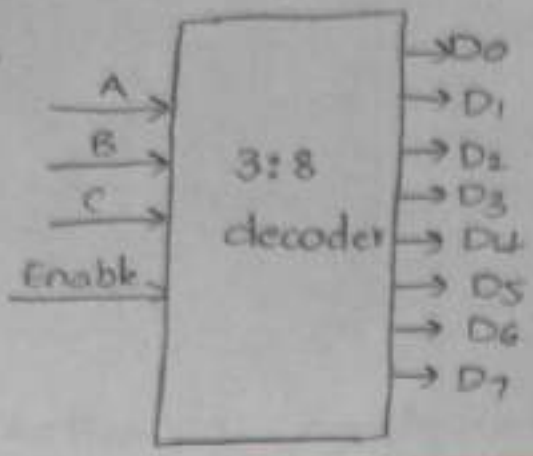
Logic Diagram:



Application:

It is used in combinational ckt to convert BCD into 7 Segment display

Draw the ckt for 3:8 decoder:



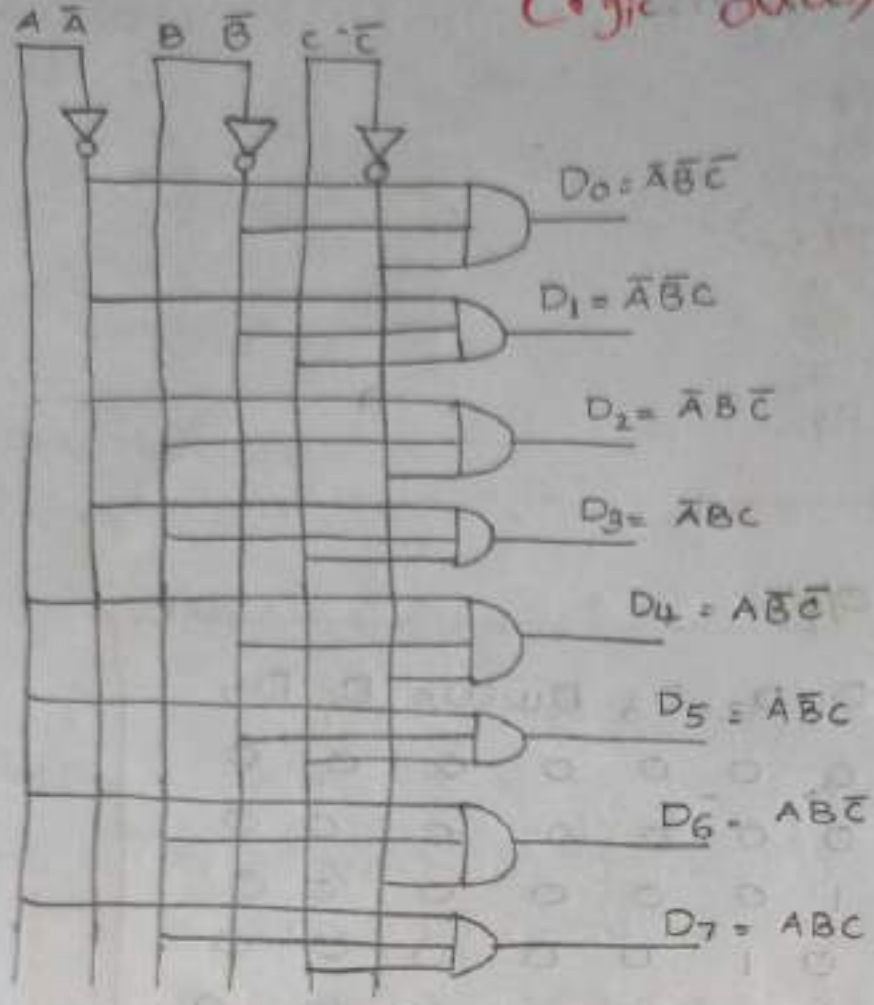
i/p's				o/p's							
En	A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
x	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

$D_0 = \bar{A}\bar{B}\bar{C}$, $D_1 = \bar{A}\bar{B}C$, $D_2 = \bar{A}B\bar{C}$, $D_3 = \bar{A}BC$, $D_4 = A\bar{B}\bar{C}$,

$D_5 = A\bar{B}C$, $D_6 = AB\bar{C}$, $D_7 = ABC$

logic diagram:

Logic diagram



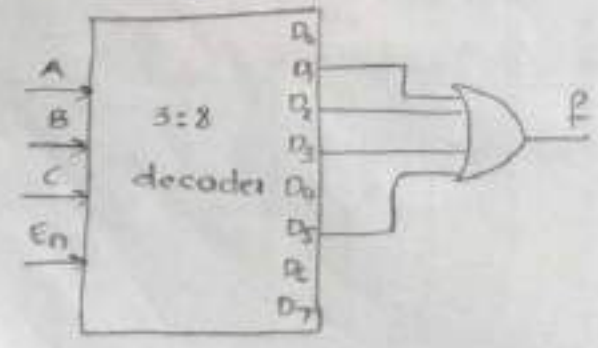
→ A decoder is a multiple input, multiple output circuit which converts coded inputs into coded outputs when both are different there is 1 to 1 mapping from input code words to output code words

→ A decoder is a connected with enable inputs to activate decoded output based on data inputs.

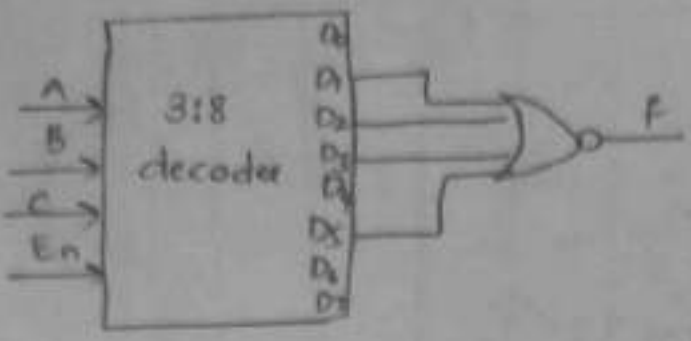
For Active High o/p:

Implements the function by using decoder.

1. $f = \Sigma m(1,2,3,5)$

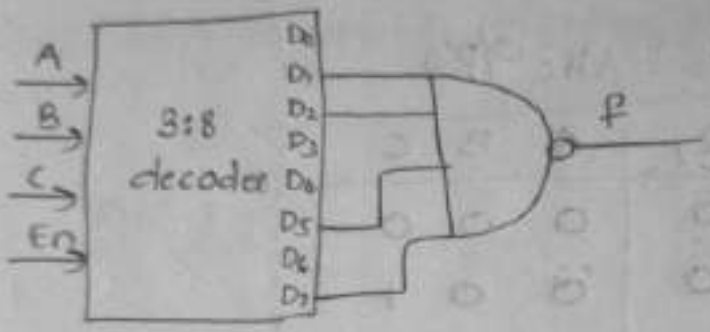


2. $f = \pi M(1,2,3,5)$

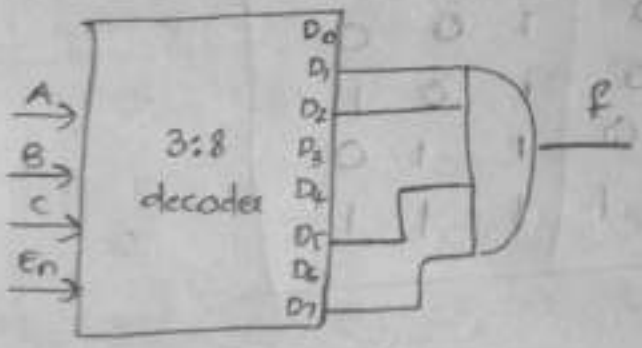


In Active Low configuration:

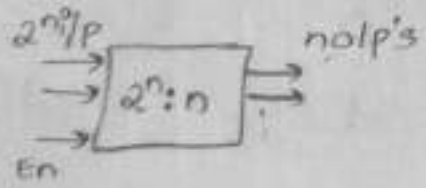
1. $f = \Sigma m(1,2,5,7)$



2. $f = \pi m(1,2,5,7)$



Encoder:



If $n = 2$

$2^2 : 2 \Rightarrow 4:2$ encoder

If $n = 3$

$2^3 : 3 \Rightarrow 8:3$ encoder

If $n = 4$

$2^4 : 4 \Rightarrow 16:4$ encoder

8:3 encoder (or) octal to Binary encoder:

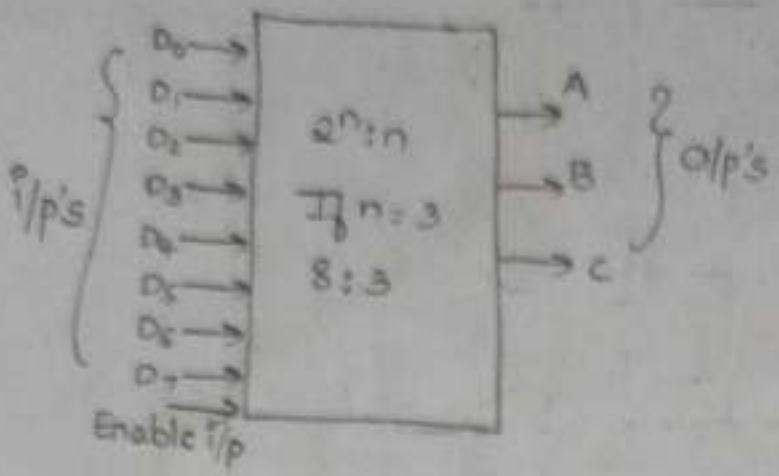


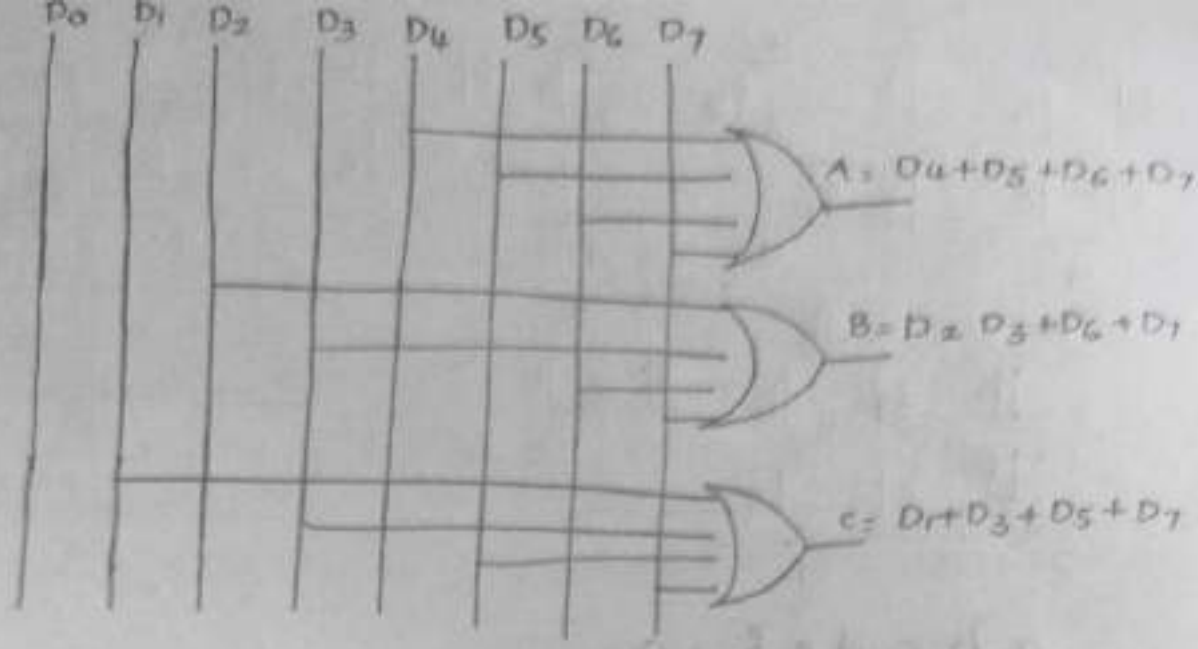
	Table i/p's								o/p's		
En	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	0	0	0	1	1
1	0	0	0	0	1	0	0	0	1	0	0
1	0	0	0	0	0	1	0	0	1	0	1
1	0	0	0	0	0	0	1	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1	1

$A = D_4 + D_5 + D_6 + D_7$

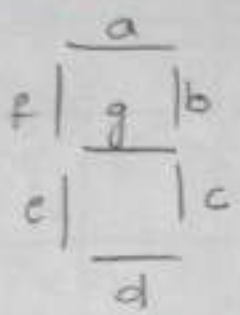
$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$

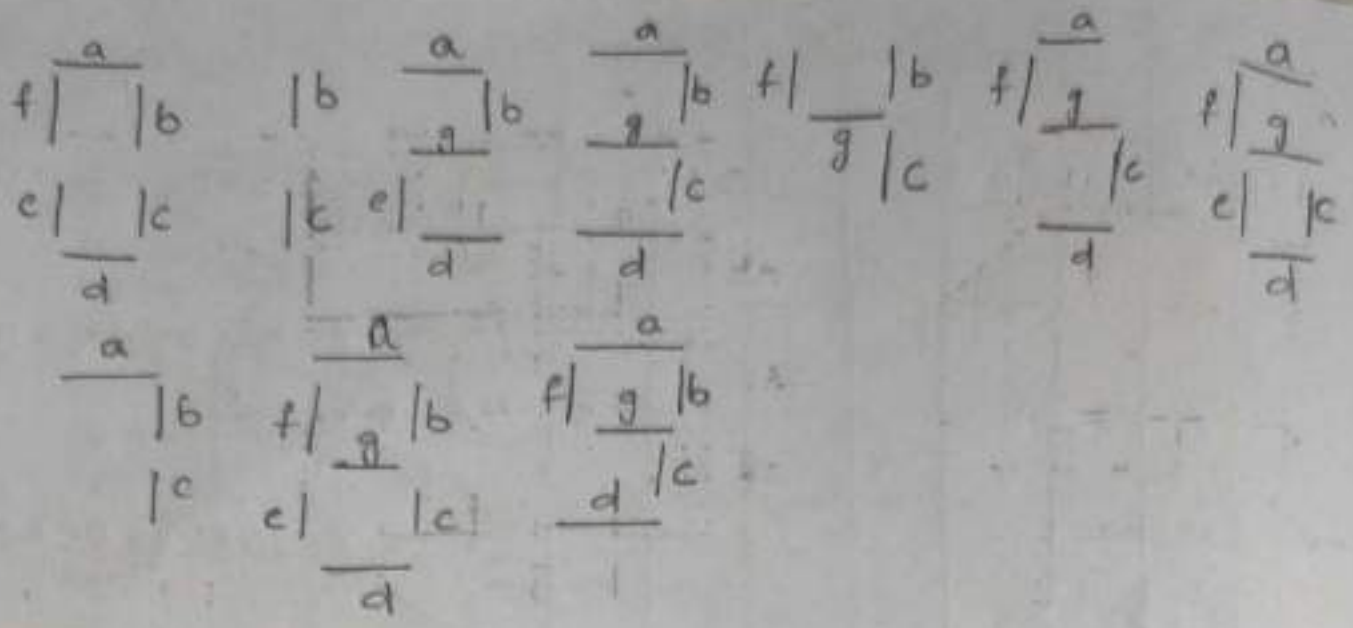
Logic Diagram:



BCD to 7 Segment Display decoder:



Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1(0)	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1(2)	1	0	1	1	0	1
3	0	0	1	1	1(3)	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1(5)	0	1	1	0	1	1
6	0	1	1	0	1(6)	0	1	1	1	1	1
7	0	1	1	1	1(7)	1	1	0	0	0	0
8	1	0	0	0	1(8)	1	1	1	1	1	1
9	1	0	0	1	1(9)	1	1	1	0	1	1



a b c d e f = 0

b, c = 1

a, b, g, e, d = 2

a, b, g, c, d = 3

f, g, b, c = 4

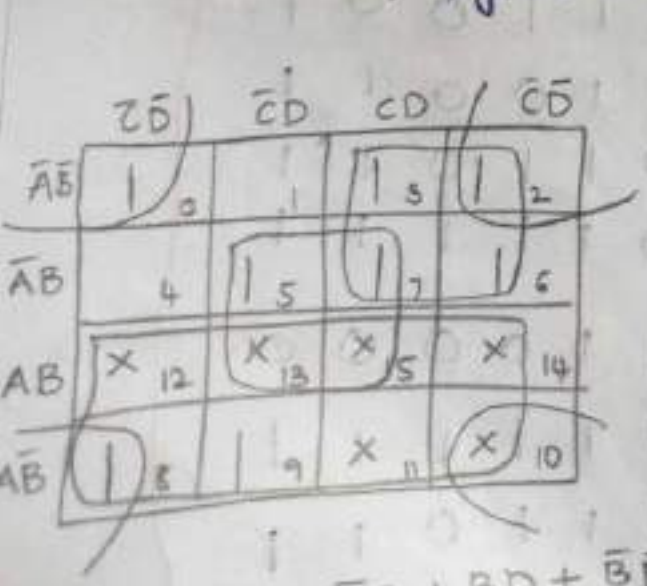
a, f, g, c, d = 5

a, f, e, d, c, g = 6

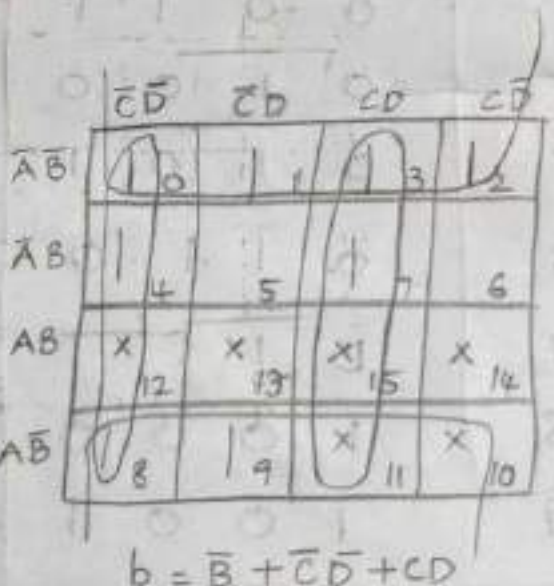
a, b, c = 7

a, b, c, d, e, f, g = 8

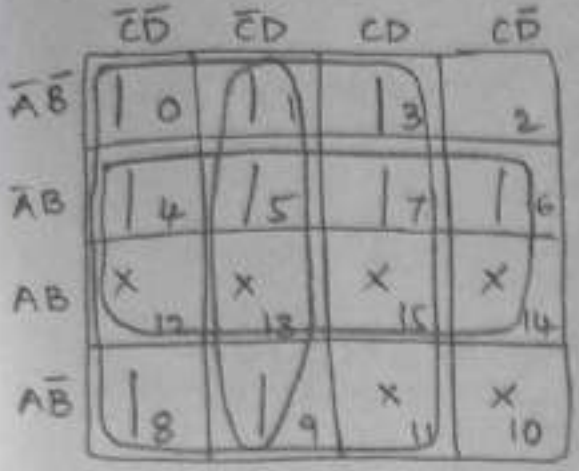
a, f, g, b, c, d = 9



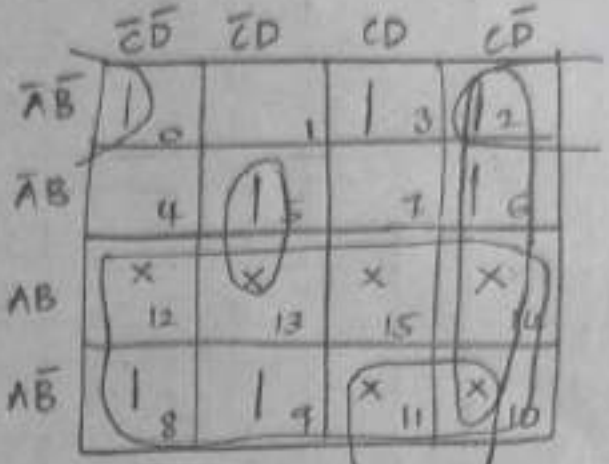
$a = A + \bar{A}C + BD + \bar{B}\bar{D}$



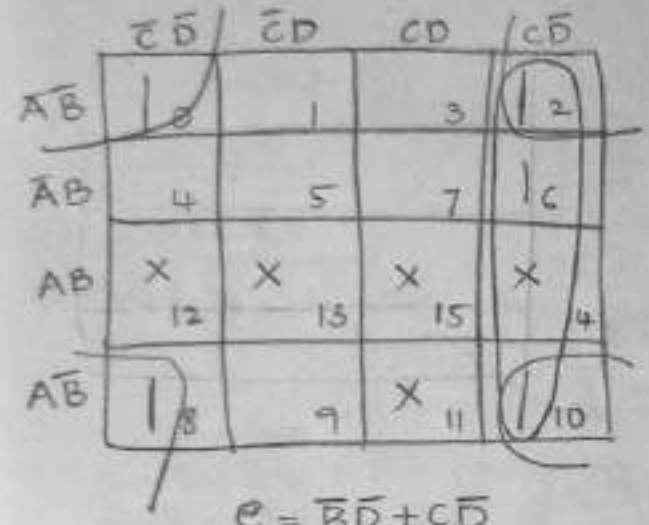
$b = \bar{B} + \bar{C}\bar{D} + CD$



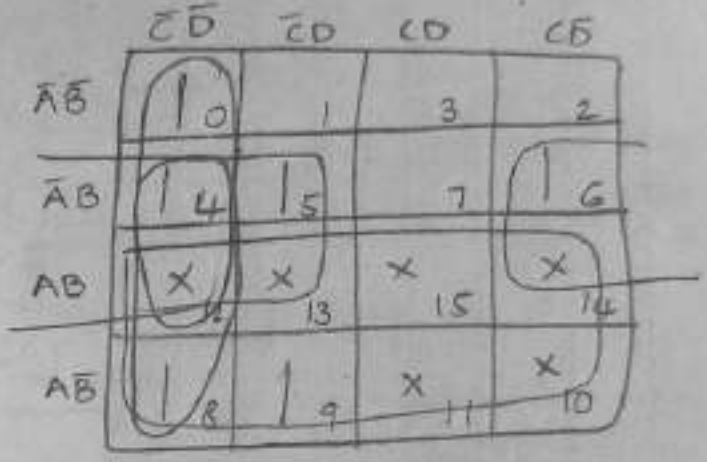
$C = B + \bar{C} + D$



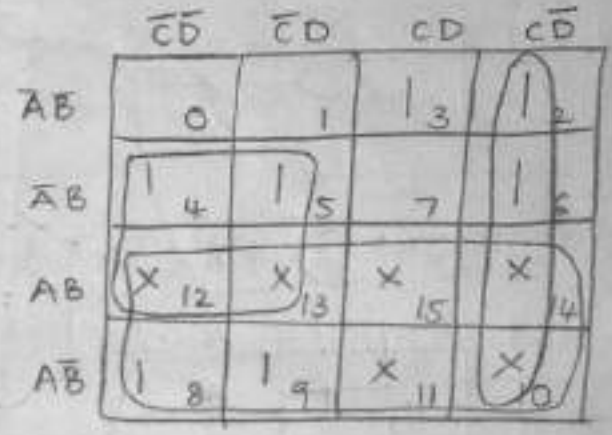
$d = \bar{B}\bar{D} + C\bar{D} + B\bar{C}D + \bar{B}C + A$



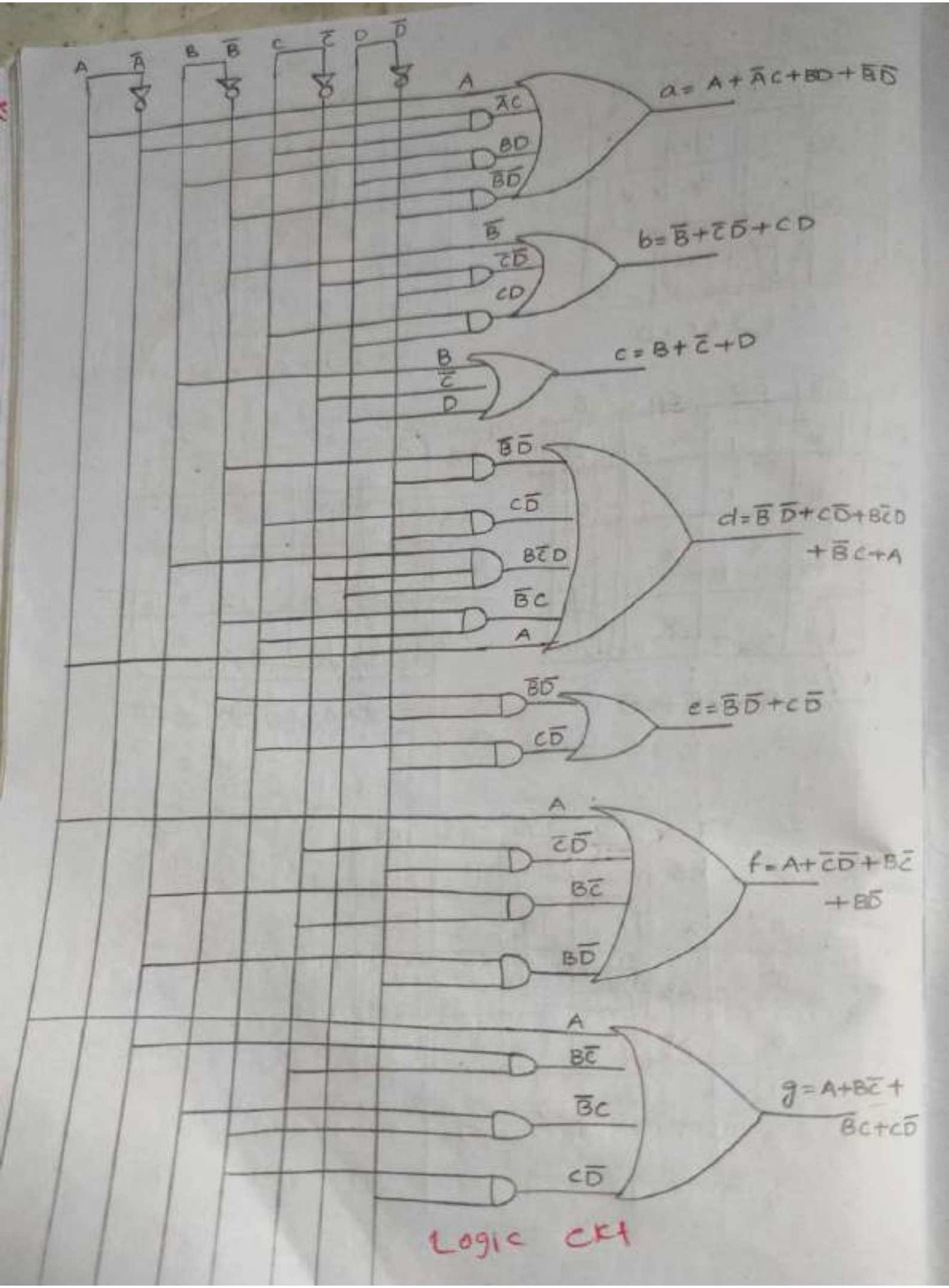
$e = \bar{B}\bar{D} + C\bar{D}$



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

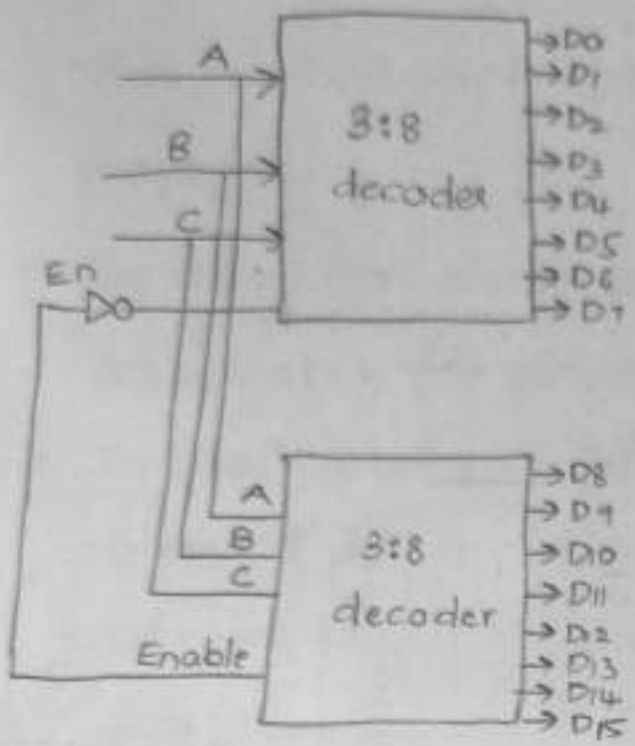


$g = A + B\bar{C} + \bar{B}C + C\bar{D}$


A
$\bar{A}$
B
$\bar{B}$
C
$\bar{C}$
D
$\bar{D}$
A
$\bar{A}C$
$B\bar{D}$
$\bar{B}\bar{D}$
$a = A + \bar{A}C + B\bar{D} + \bar{B}\bar{D}$
$\bar{B}$
$\bar{C}\bar{D}$
$C\bar{D}$
$b = \bar{B} + \bar{C}\bar{D} + C\bar{D}$
B
$\bar{C}$
D
$c = B + \bar{C} + D$
$\bar{B}\bar{D}$
$C\bar{D}$
$B\bar{C}D$
$\bar{B}C$
A
$d = \bar{B}\bar{D} + C\bar{D} + B\bar{C}D + \bar{B}C + A$
$\bar{B}\bar{D}$
$C\bar{D}$
$e = \bar{B}\bar{D} + C\bar{D}$
A
$\bar{C}\bar{D}$
$B\bar{C}$
$B\bar{D}$
$f = A + \bar{C}\bar{D} + B\bar{C} + B\bar{D}$
A
$B\bar{C}$
$\bar{B}C$
$C\bar{D}$
$g = A + B\bar{C} + \bar{B}C + C\bar{D}$

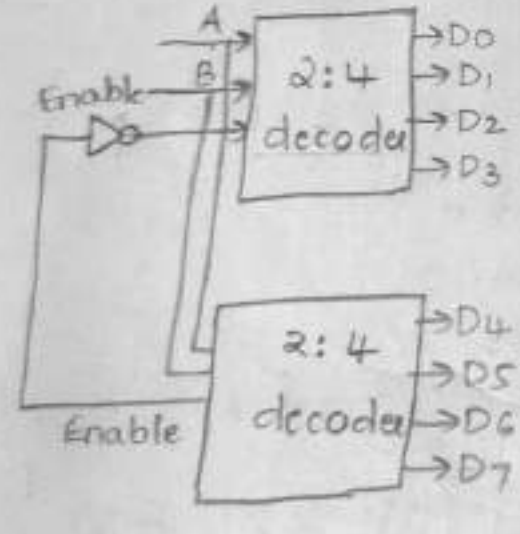
Logic ckt

construct of 4:16 decoder by using 3:8 decoder:



E	A	B	C	O/P
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7
1	0	0	0	D8
1	0	0	1	D9
1	0	1	0	D10
1	0	1	1	D11
1	1	0	0	D12
1	1	0	1	D13
1	1	1	0	D14
1	1	1	1	D15

construct 3:8 decoder by using 2:4 decoder:



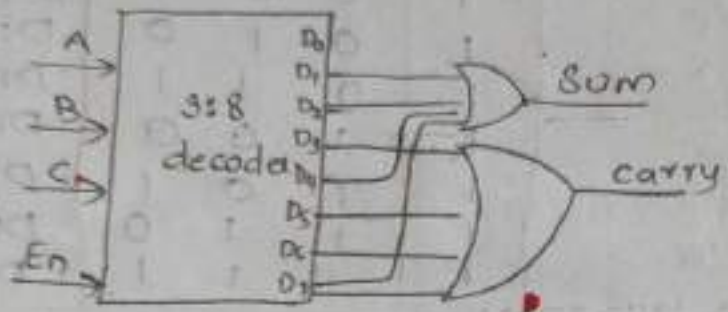
E	A	B	O/P
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

Implement full adder by using decoder:

Sum (A, B, C) = Σm (1, 2, 4, 7)

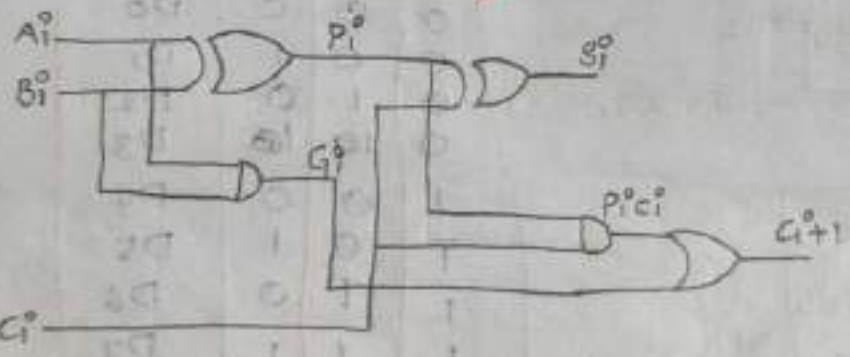
carry (A, B, C) = Σm (3, 5, 6, 7)

i/p's A B C	o/p Sum	carry
0 0 0	0	0
0 0 1	1 (1)	0
0 1 0	1 (2)	0
0 1 1	0	1 (3)
1 0 0	1 (4)	0
1 0 1	0	01 (5)
1 1 0	0	1 (6)
1 1 1	1 (7)	1 (7)



carry look ahead adder (continuation of parallel Adder)

consider full adder ckt



→ The parallel adder is ripple carry type in which the carry output of each full adder stage is connected to carry input of next stage.

→ The sum and carry of any stage cannot be produced until input carry occurs, this leads delay is known as carry propogation delay.

→ the method of speeding of these process by eliminating inter stage carry delay is called "look ahead carry addition" this method uses 2 functions carry generate and carry propogate.

→ By using look ahead carry generator we can easily construct 4-bit parallel adder.

→ In circuit each sum output requires 2 EXOR gates the output of 1st Ex-OR gate generates Pi and AND gate generates Gi.

Expressions are from fulladder ckt

$P_i = A_i \oplus B_i$

$G_i = A_i B_i$

O/p Sum & carry can be expressed as

$S_i = P_i \oplus c_i$

$c_{i+1} = P_i c_i + G_i$

where G_i = carry generate, P_i = carry propogate

now the boolean function for each stage is

if $i=1$

$c_2 = G_1 + P_1 c_1$

$c_3 = G_2 + P_2 c_2$

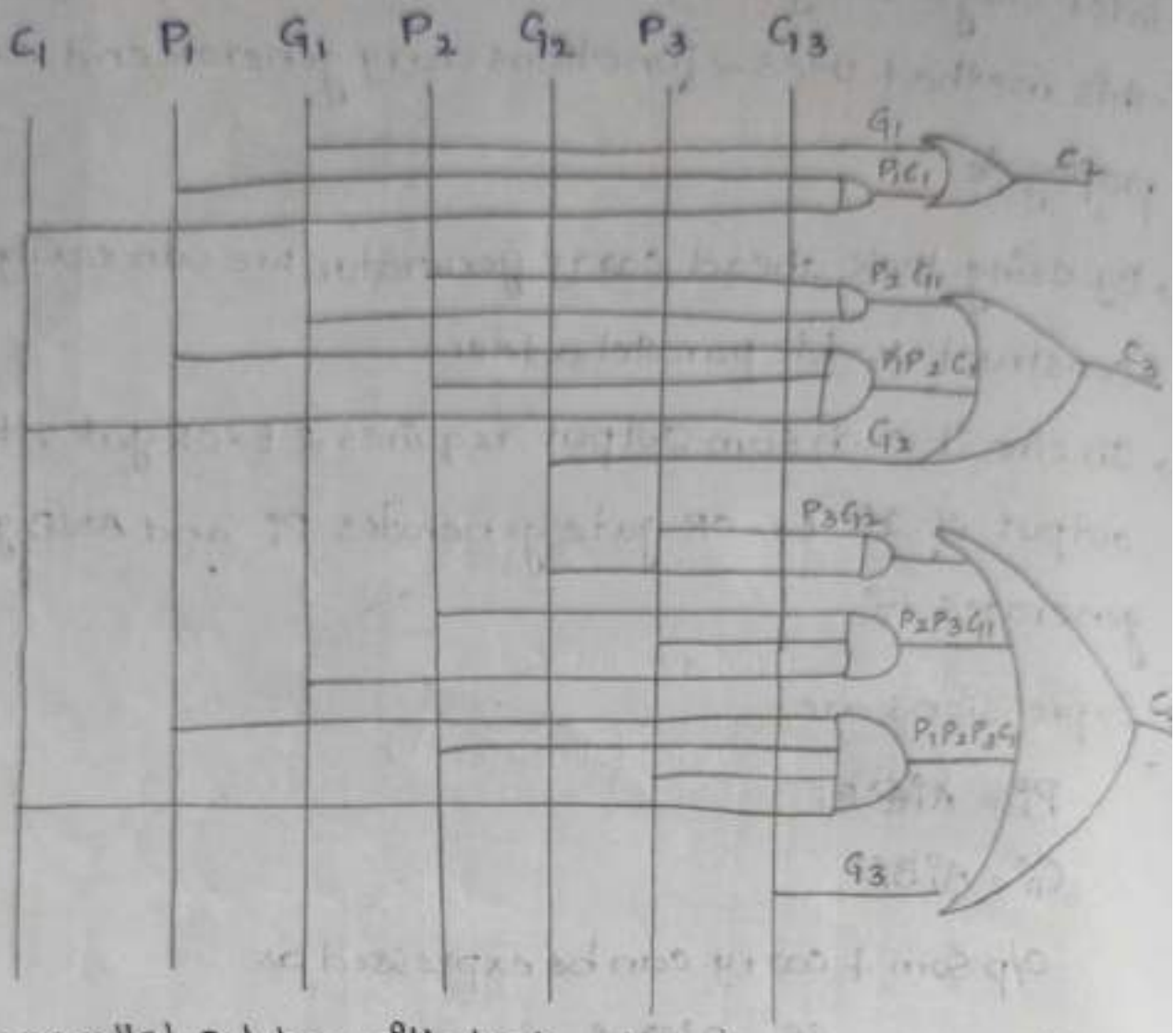
$= G_2 + P_2 (G_1 + P_1 c_1)$

$c_3 = G_2 + P_2 G_1 + P_1 P_2 c_1$

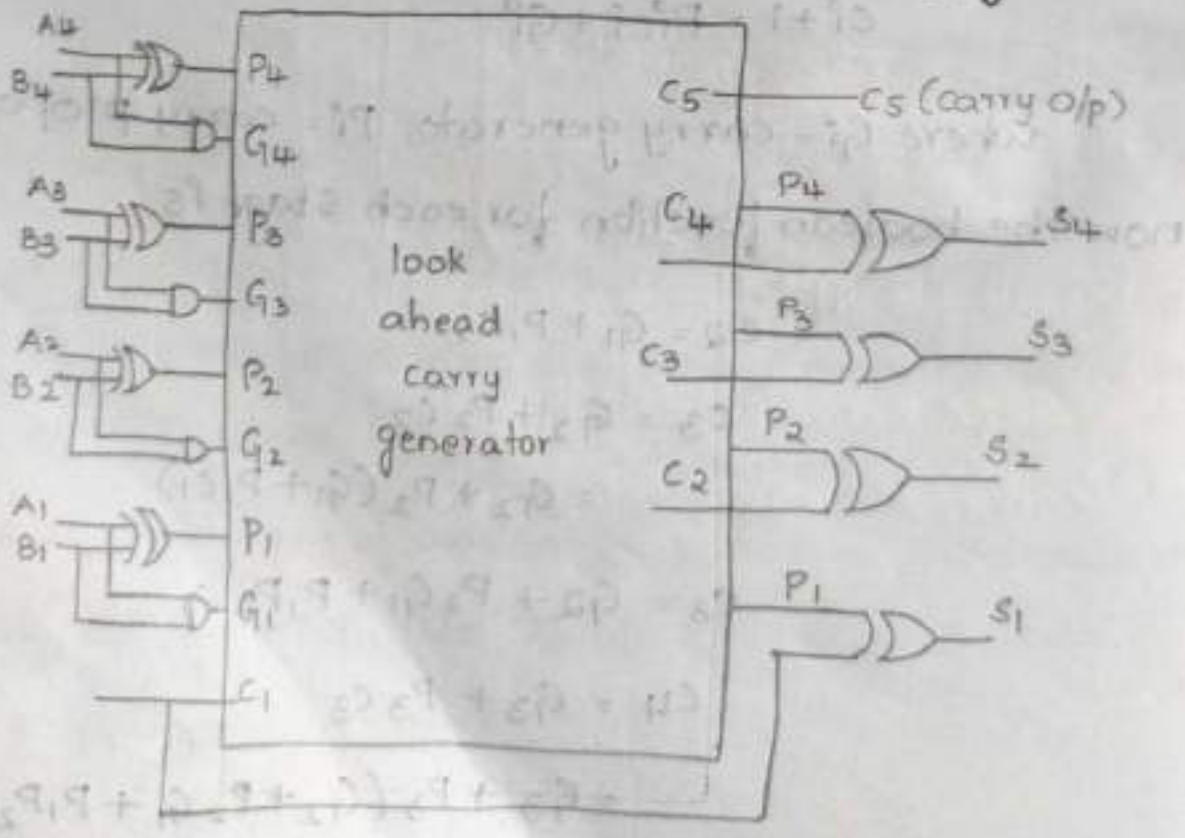
$c_4 = G_3 + P_3 c_3$

$= G_3 + P_3 (G_2 + P_2 G_1 + P_1 P_2 c_1)$

$$C_4 = G_3 + P_3 G_2 + P_2 P_3 G_1 + P_1 P_2 P_3 C_1$$



4-bit parallel adder with look ahead carry generator:

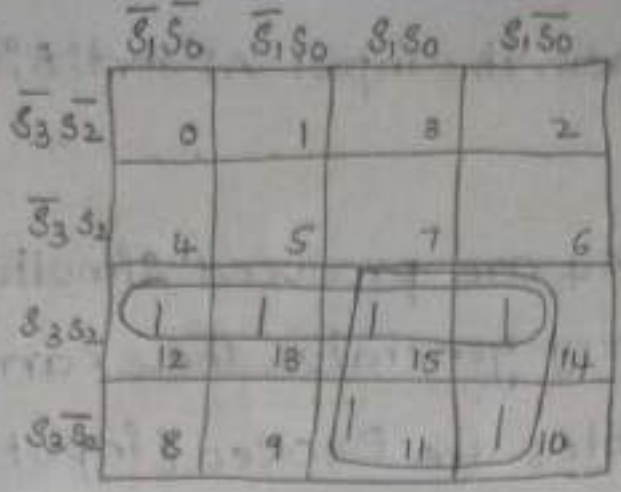


→ P_i & G_i are obtained from the input A_i and B_i by using Ex-OR and AND gates

→ The carry output c_1 to c_4 are produced simultaneously by the look ahead carry generator, these carry outputs and P_i variables are ExORed (inputs of Ex-OR) to produce the sum outputs S_1 to S_4

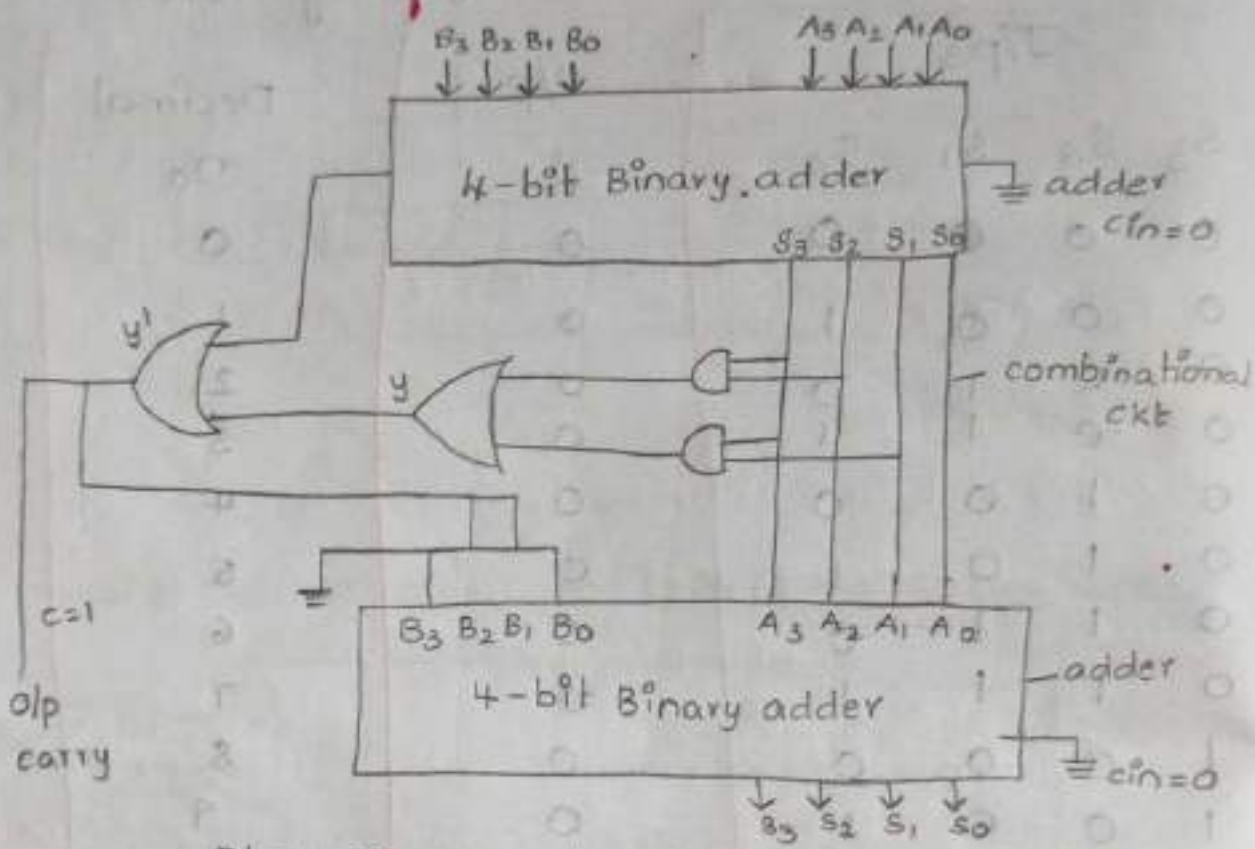
Decimal adder (or) BCD Adder by using 4 bit:

I/p's				o/p	Decimal
S_3	S_2	S_1	S_0	y	n_o
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	2
0	0	1	1	0	3
0	1	0	0	0	4
0	1	0	1	0	5
0	1	1	0	0	6
0	1	1	1	0	7
1	0	0	0	0	8
1	0	0	1	0	9
1	0	1	0	1	10
1	0	1	1	1	11
1	1	0	0	1	12
1	1	0	1	1	13
1	1	1	0	1	14
1	1	1	1	1	15



$Y = S_3\bar{S}_2 + S_3 S_1$

k-map for o/p 'y'



Block Diagram of BCD adder

priority Encoder:

i/p's				o/p's		
D_0	D_1	D_2	D_3	A	B	C
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

In this if 2 or more i/p's are equal to 1 then at the same time the i/p having the highest priority will precede first.

xx10 → A xxx1

0010 → 2

0110 → 6

1010 → 10

1110 → 14

xxx1 → B

0001 → 1

0011 → 3

0101 → 5

0111 → 7

1001 → 9

1011 → 11

1101 → 13

1111 → 15

x100

0100 → 4

1100 → 12

A = xx1, xxx1

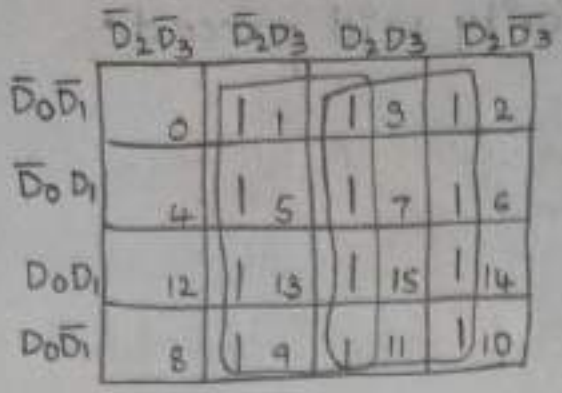
$= \Sigma m(1,2,3,5,6,7,9,10,11,13,14,15)$

B = x100, xxx1

$= \Sigma m(1,3,4,5,7,9,11,12,13,15)$

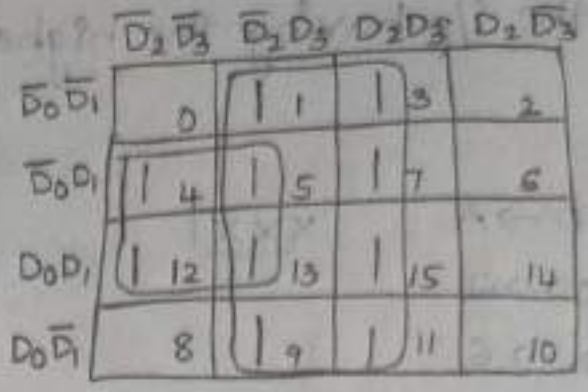
$y = D_3 + D_2 + D_1 + D_0$

k-map for A:

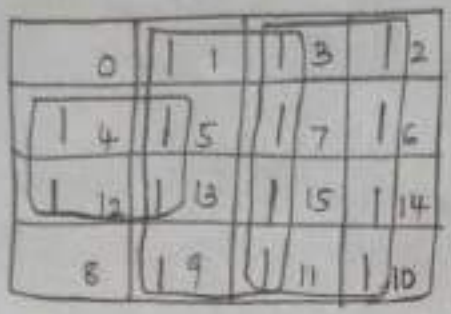


$A = D_3 + D_2$

K-map for B:

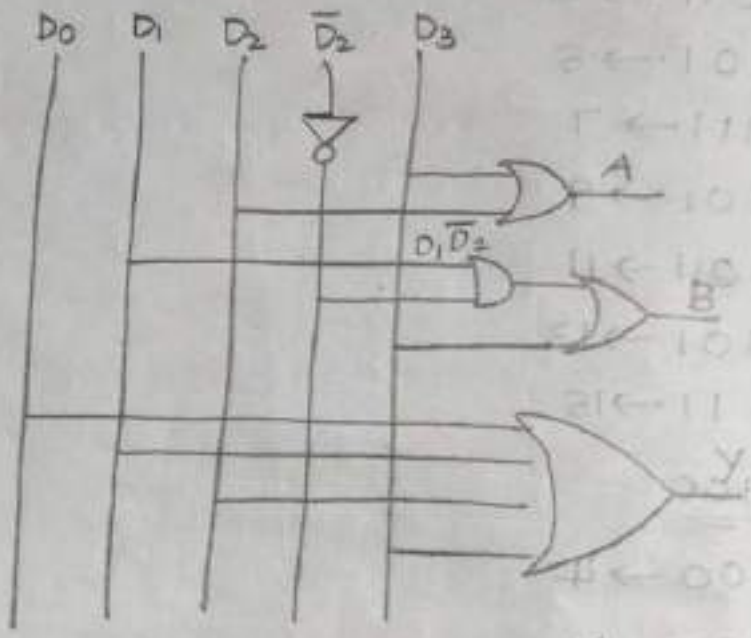


$B = D_1\bar{D}_2 + D_3$



$y = D_0 + D_1 + D_2 + D_3$

Ckt diagram:

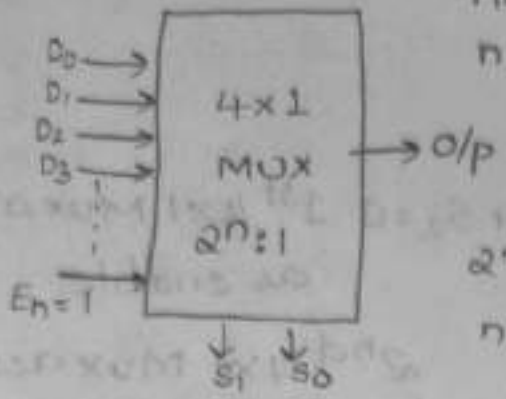


Multiplexer:

A Multiplexer (or) data selector is a ckt that accepts several inputs and allows only one of them to the output the input to the output is controlled by selection inputs, It is many inputs single output.

decoder - $n:2^n$
encoder - $2^n:n$
Mux - $2^n:1$
DeMux - $1:2^n$

4:1 MUX



Here,
n = selection lines

$2^n \rightarrow$ i/p's

$n \rightarrow$ o/p's

Truth Table

En	S_1	S_0	Y
0	0	0	0
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3

$$y = \overline{S_1}\,\overline{S_0}\,D_0 + \overline{S_1}\,S_0\,D_1 + S_1\,\overline{S_0}\,D_2 + S_1\,S_0\,D_3$$

2^n:1 MUX

If n=1 ⇒ 2:1 MUX

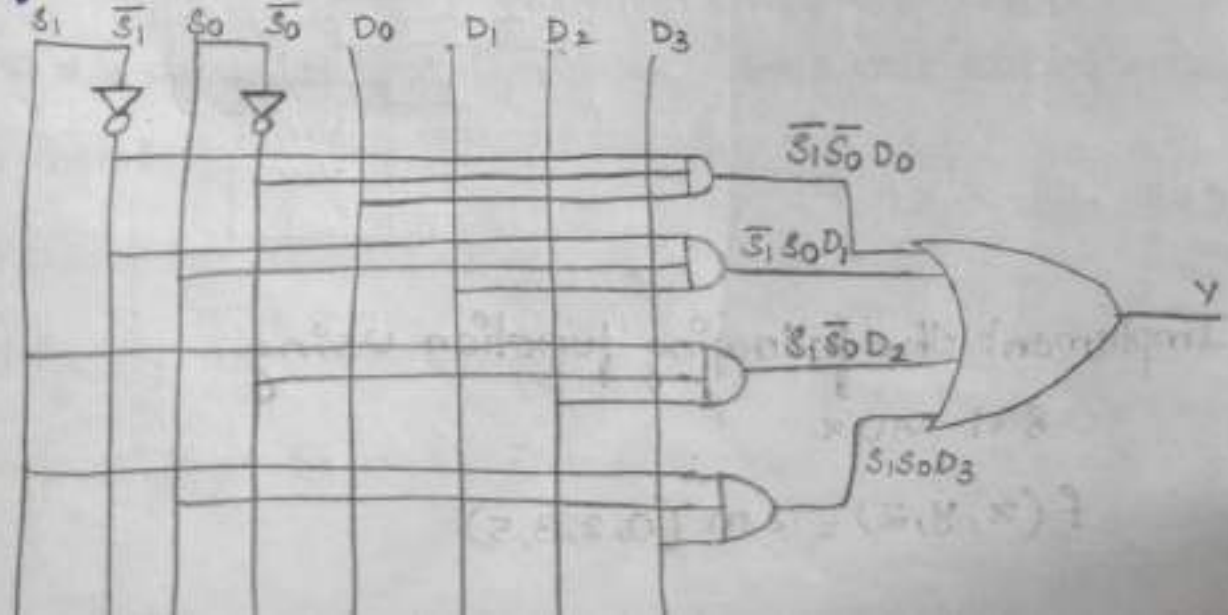
If n=2 ⇒ 4:1 MUX

If n=3 ⇒ 8:1 MUX

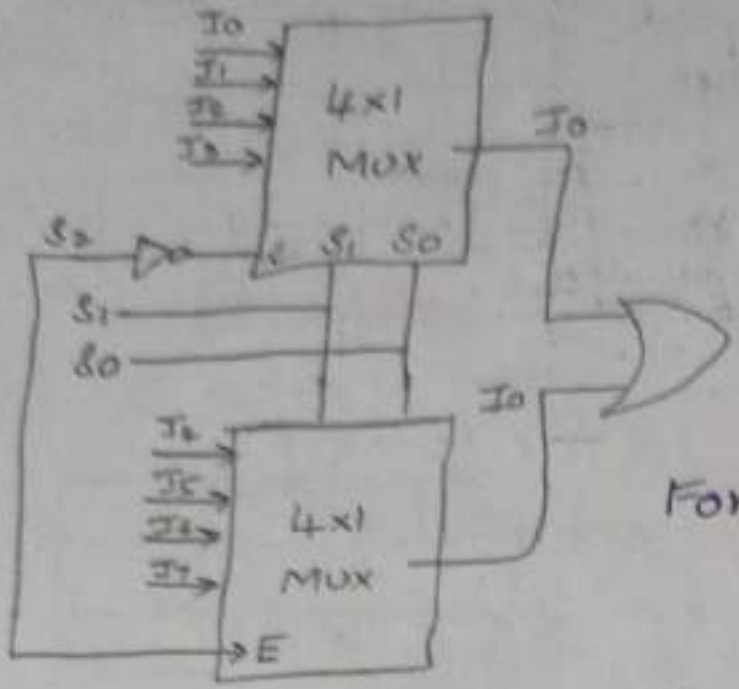
If n=4 ⇒ 16:1 MUX

If n=5 ⇒ 32:1 MUX

If n=6 ⇒ 64:1 MUX



Design 8x1 Mux using 4x1 Muxes



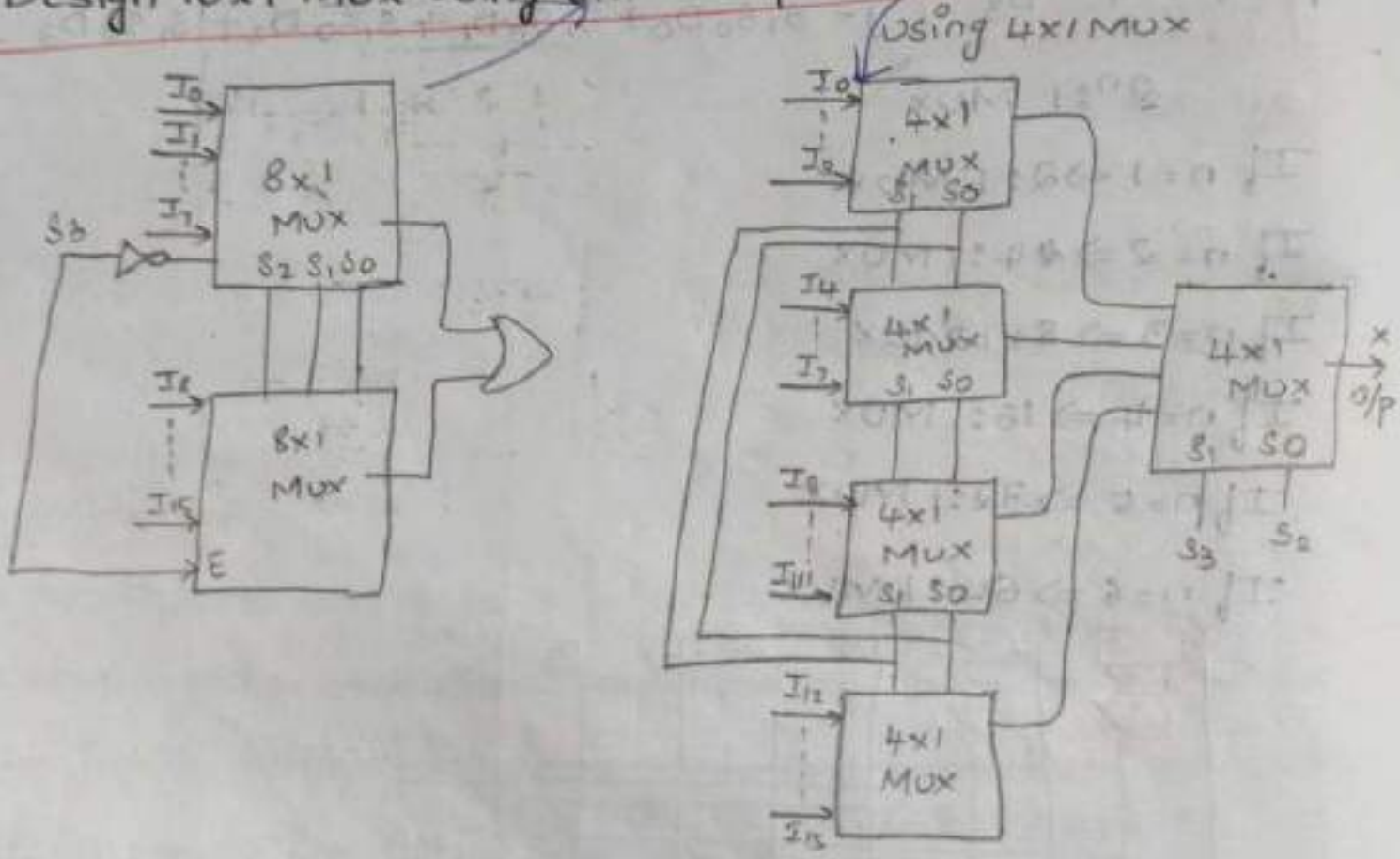
For $S_2 = 0$ 1st 4x1 Mux acts as enable

2nd 4x1 Mux acts as disable

For $S_2 = 1$ 1st 4x1 Mux acts as disable

2nd 4x1 Mux acts as enable

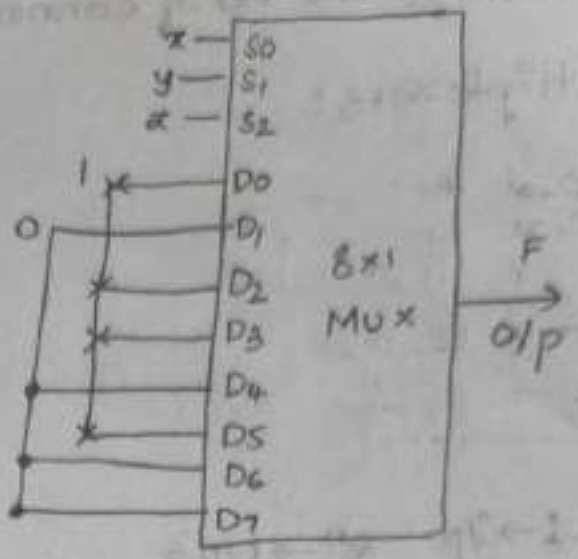
Design 16x1 Mux using 8x1 Mux & 4x1 Mux



Implement the following function using 8x1 Mux

$$f(x, y, z) = \Sigma m\,(0, 2, 3, 5)$$

S_0	S_1	S_2	F
x	y	z	
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



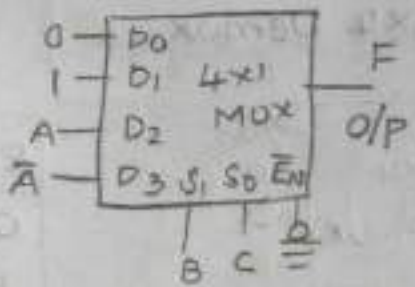
Implement the following function using 4×1 MUX

$f(A,B,C) = \Sigma m(1,3,5,6)$

data selector — Selection lines

min	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

	D_0	D_1	D_2	D_3
$\bar{A}$	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	$\bar{A}$



Applications of Multiplexers:

→ It is used as data selector to select one out of many data inputs

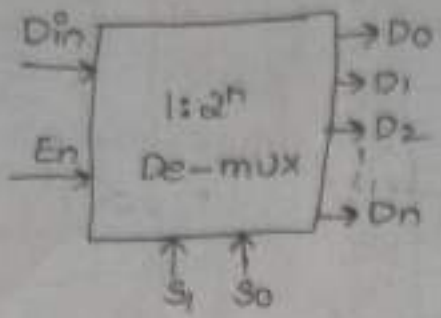
→ It is used for simplification of logic design

→ In designing the combinational circuits.

→ In the digital to analog convertors (used in)

→ To minimize the no of connections

De-Multiplexers:



1 → i/p 2^n → o/p's

n → selection lines

If n = 1 ⇒ 1:2 → De-mux

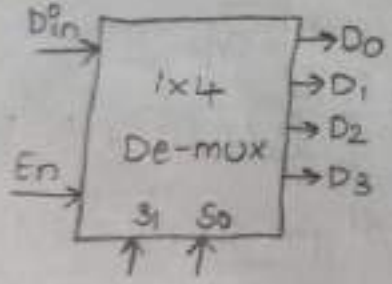
If n = 2 ⇒ 1:4 → De-mux

If n = 3 ⇒ 1:8 → De-mux

If n = 4 ⇒ 1:16 → De-mux

If n = 5 ⇒ 1:32 → De-mux

Design 1×4 Demux

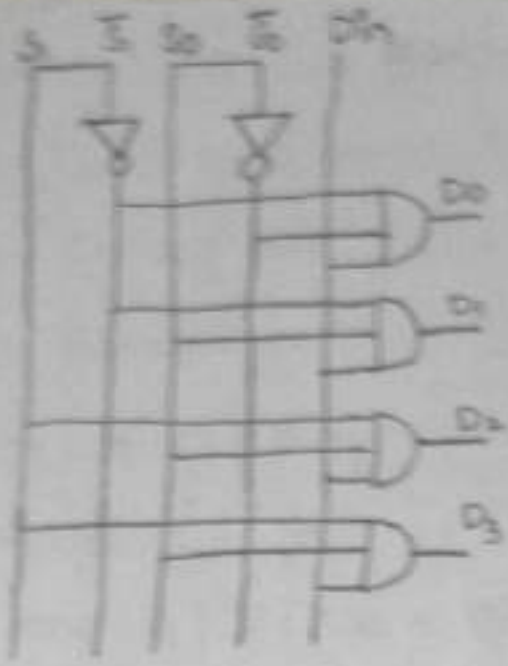


i/p's				o/p's			
En	S_1	S_0	Din	D_0	D_1	D_2	D_3
0	x	x	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	1	0	1	0	0
1	1	0	1	0	0	1	0
1	1	1	1	0	0	0	1

ckt diagram:

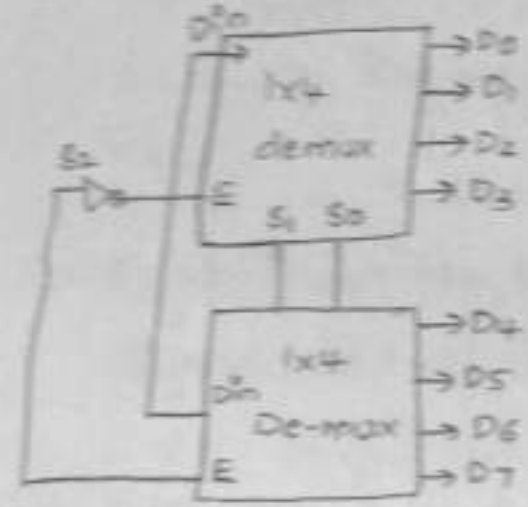
$D_0 = \overline{S_1}\,\overline{S_0}\,Din$, $D_1 = \overline{S_1}\,S_0\,Din$

$D_2 = S_1\,\overline{S_0}\,Din$, $D_3 = S_1 S_0\,Din$



logic diagram for 1:4 Demux

Design 1×8 De-mux by using ② $\frac{1}{2}$×4 De-mux



If $S_2=0 \Rightarrow E_n=1$ for 1st 1×4 De-mux enable (ON)

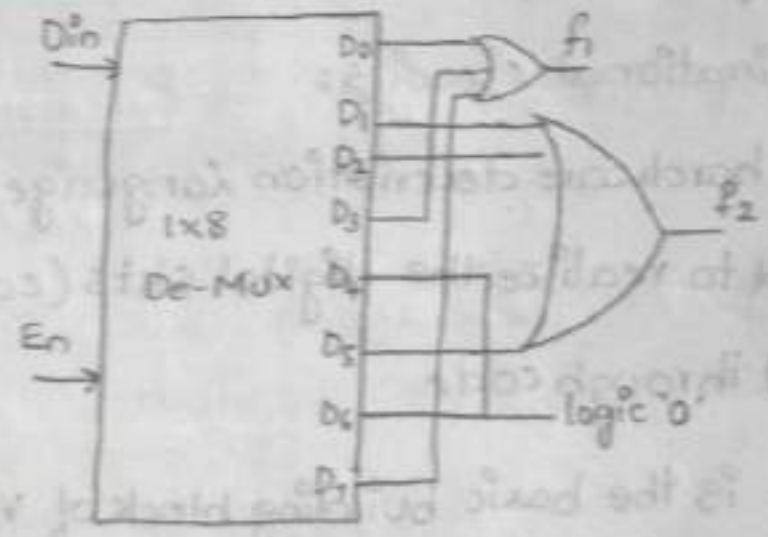
$E_n=0$ for 2nd 1×4 De-mux disable (OFF)

If $S_2=1 \Rightarrow E_n=0$ for 1st 1×4 Demux disable

$E_n=1$ for 2nd 1×4 De-mux enable (ON)

Implement $f_1(A,B,C)=\Sigma m(0,3,7)$

$f_2(A,B,C)=\Sigma m(1,2,5)$

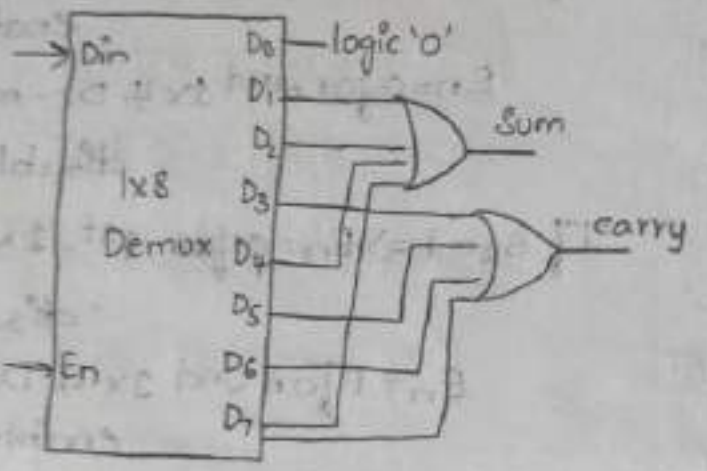


Implement full Adder by using De-mux

S_0	S_1	S_2	Sum	Carry
0	0	0	0	0
0	0	1	1(1)	0
0	1	0	1(2)	0
0	1	1	0	1(3)
1	0	0	1(4)	0
1	0	1	0	1(5)
1	1	0	0	1(6)
1	1	1	1(7)	1(7)

$Sum = \Sigma m(1,2,4,7)$

$carry = \Sigma m(3,5,6,7)$



present State:

Data stored by the mem

HDL for combinational circuits:

The verilog hardware description language (HDL)

→ It is used to realize the digital ckts (combinational & sequential) through code.

→ The module is the basic building block of verilog HDL.

→ A module can be described in 3 ways:

1. Gate-level modeling
2. Data flow
3. Behaviour

Ex 1: write verilog code for Half adder using Gate level model

```
module HA (S, C, A, B);
    output S, C;
    XOR f1 (S, A, B);
    and f2 (C, A, B);
    end module
```

Ex 2: write verilog code for halfadder by using dataflow model

```
module half (s, c, a, b);
    input a, b;
    output s, c;
    assign s = a^b;
    assign c = a&b;
    endmodule
```

Ex 3: write verilog code for half adder by using Behaviour model.

```
Module half (s, c, a, b);
    input a, b;
    out reg s, c;
    always@(*)
    begin
        s = a^b;
        c = a&b;
    end
    end module
```

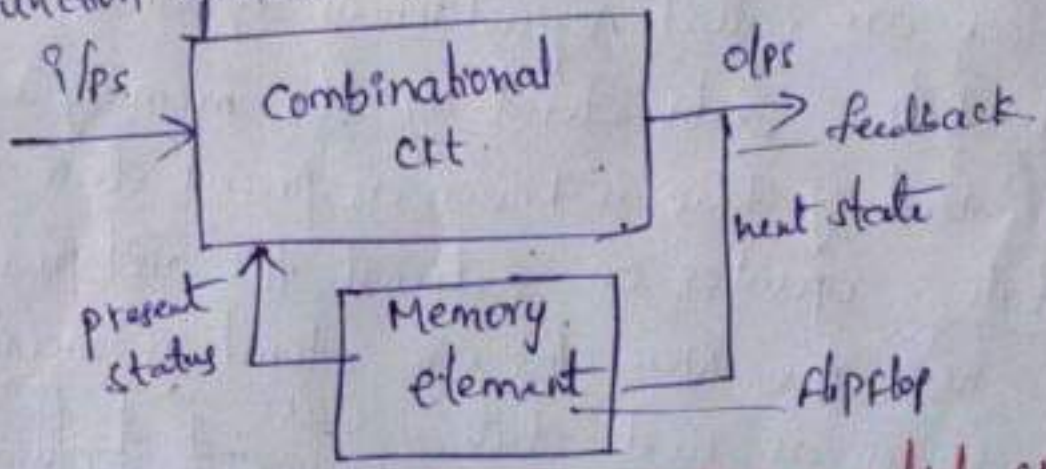
UNIT-IV

Sequential Logic Ckts.

Sequential Machines Fundamentals

Introduction

→ In combinational ckt don't use any memory
→ In some applications require o/p's to be generated that are not only dependent on the present i/p & past i/p..
So in this case we require the prefer sequential ckt.
→ Counters, shift registers, serial adders, sequence generators, logic function generators, etc. are examples of sequential ckts



Comparison b/w Combinational ckt & Sequential ckt

Combinational ckt	Sequential ckt
1) o/p variables at any instant of time are dependent only on the present i/p variables	Here dependent on present & past i/p variables
2) Memory is not required	Memory required.
3) It is faster because delay b/w i/p & o/p is due to propagation delay of gates.	slower than combinational ckt
4) ckts are easy to design	ckts are comparatively harder to design

Classification of sequential ckts.

Sequential ckts may be classified as

1) Synchronous sequential ckts
2) Asynchronous sequential ckts.

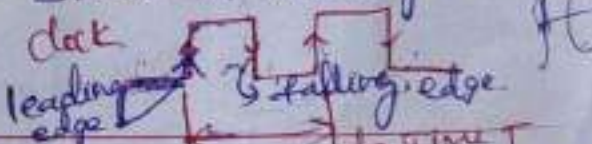
depending upon timing of their signals.

Synchronous sequential ckts :- sequential ckts which are controlled by a clock are called Synchronous sequential ckts. These ckts will be active only when clock signal is present.

Asynchronous sequential ckts :- ckts which aren't controlled by clock are called Asynchronous sequential ckts.

present state :- data stored by the memory element at any given instant of time is known as present state of sequential ckt

Next state :- operates on external i/p's and present state to produce new o/p's. New o/p's are stored in memory element and known as next state.



Synchronous sequential ckts	Asynchronous
1) In synchronous ckts, Memory elements are clocked flip-flop	In Asynchronous ckts Memory elements are either unclocked flip-flops or time delay elements.
2) In syn ckts, the change in i/p signals can affect memory element upon Activation of clock signal	2) In Asynch ckts change in signals can affect memory element at any instant of time.
3) The Maximum operation speed of clock depends on time delay involved.	3) Because of absence of clk This ckts operate faster than Synchron ckts
4) Easier to design.	4) More difficult to design.

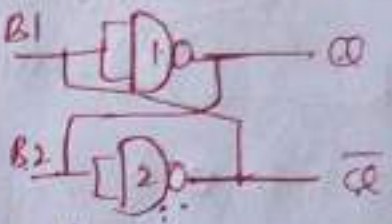
One bit Memory Cell

Flip-flop is popularly known as a basic digital memory ckt. It has two stable states namely logic 1 state and logic 0 state

→ It can be designed either using NOR & NAND gates.

→ Basic ckt Here NAND 1 & 2 gates are basically as Inverters. This ckt is known as cross coupled Inverter.

→ o/p of gate 1 connected to the i/p of gate 2 and o/p of gate 2 is connected to i/p of gate-1



→ Assume o/p gate 1 ie $Q=1$, $\therefore B_1=B_2=1$
o/p of gate 2 ie $\bar{Q}=0$, $B_1=0$

1) The o/p's Q & $\bar{Q}$ are always complementary.
2) $Q=1$, is referred to as set state & $Q=0$ " Reset "
3) This ckt store 1 bit of digital information called as 1-bit Memory Cell.
4) The 1-bit information stored in the ckt is locked (or) latched in the ckt is also called as latch.

Modified ckt for 1-bit Memory cell

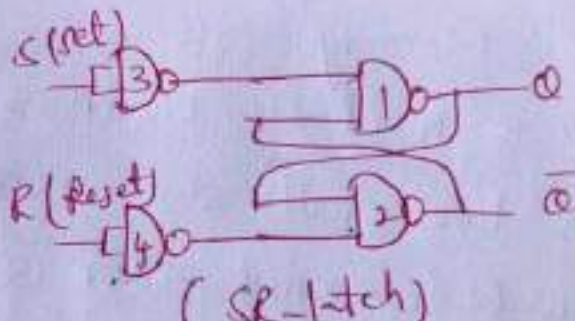
→ when the power is switched on the ckt switches to one of the stable states ie. $Q=1$ (or) $Q=0$

→ And it is n't possible to predict the state.

→ Thus we can't store/enter the desired digital information in it

→ By replacing Inverters 1 & 2 with 2-i/p NAND gates. we can use other i/p terminal of the NAND gates to enter the desired digital information. Cross coupled.

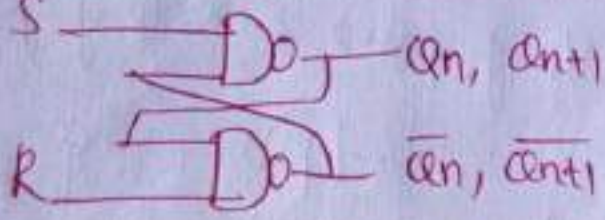
i/p for gate 3 is S & i/p for gate 4 is R. Hence latch is also called SR latch.



NAND

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

SR latch Using NAND gates



S	R	Qn	Qn+1	state
0	0	0	x	indeterminent
0	0	1	x	
0	1	0	1	set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	No change
1	1	1	1	

S=1, R=0, Q=0, $\bar{Q}$=1
S=0, R=1 Q=1, $\bar{Q}$=0
S=1, R=0 Q=0, $\bar{Q}$=1
S=0, R=1, Q=1, $\bar{Q}$=0
S=0, R=0, Q=1, $\bar{Q}$=1
S=1, R=1, Q=0, $\bar{Q}$=0
S=0, R=0, Q=1, $\bar{Q}$=1
S=1, R=1, Q=0, $\bar{Q}$=0
S=0, R=0 Q=1, $\bar{Q}$=1

Qn → present state before applied i/p
Qn+1 → next state after applied i/p

SR latch using NOR gates

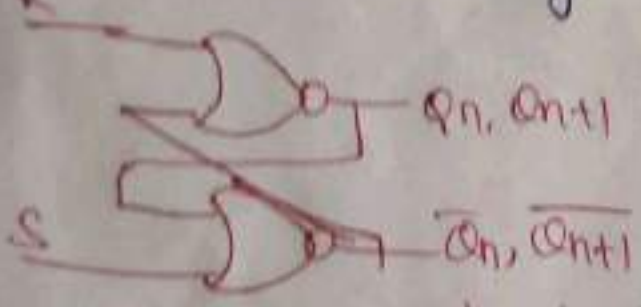
NOR gate Mandatory

SR latch using NOR gates

Here R first then set only in NOR gate Mandoratory because our requirement. we need to require functionality

So first we give to R & then S.

→ In case first S is given but this case we can't get require functionality.



NOR gate Truthtable

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

If Any one of the o/p is 1 o/p will zero (0).

Q=1 Q=1 Reset Q=0

S=1, R=0, Q=1, $\bar{Q}$=0

S=0, R=1, Q=0, $\bar{Q}$=1

S=1, R=1, Q=0, $\bar{Q}$=0

S=1, R=0, Q=1, $\bar{Q}$=0

S=0, R=0 Q=1 $\bar{Q}$=0

S=0, R=1, Q=0, $\bar{Q}$=1

S=1, R=1, Q=0, $\bar{Q}$=0

S=0, R=0, Q=0, $\bar{Q}$=1

S	R	Q_n	Q_{n+1}	state
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	x	Indeterminent condition
1	1	1	x	

Here we can't apply statement from table Consider previous o/p Q value as q the i/p for R gate So previous Q=0, Q=0 present both are same No change

Q=0, $\bar{Q}$=0

Q ≠ $\bar{Q}$ = Indeterminent

Gated SR latch

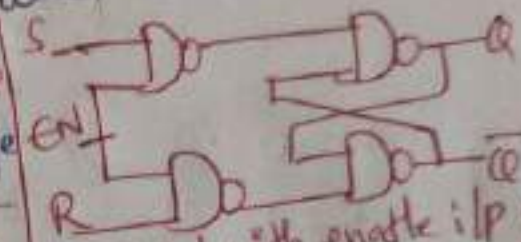
In SR latch we have seen that o/p changes occur immediately after the i/p changes occur ie. latch is sensitive to its S & R i/ps at all times.

→ It can easily be modified to create a latch that is sensitive to these i/ps only when an enable i/p is Active.

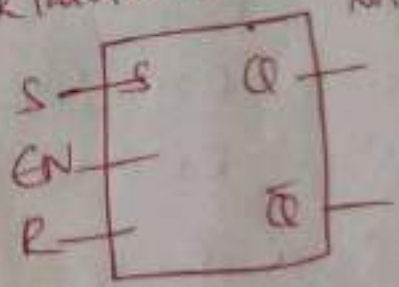
→ Such latch with enable i/p is known as gated SR latch.

→ ckt behaves like a SR latch when EN=1. and retains its previous state when EN=0.

EN	S	R	Q_n	Q_{n+1}	State
0	0	0	0	0	No change
0	0	0	1	1	
1	0	0	0	0	Reset
1	0	1	1	0	
1	0	1	0	0	
1	1	0	0	1	set
1	1	0	1	1	
1	1	1	0	X	Invalid/Indeterminate
1	1	1	1	X	



SR latch with enable i/p using NAND gates

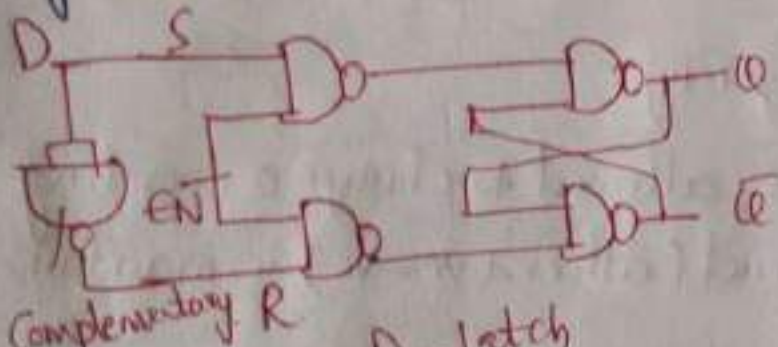


D Latch

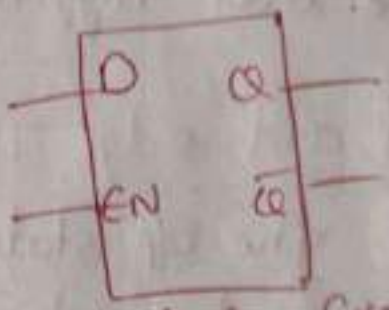
Truth table of the SR latch we can realize that when both i/p's are same the o/p either doesn't change or it is invalid.

00 → No change } many practical applications - these i/p's
11 → Invalid } conditions are not required.

→ These i/p's are avoided by making them complementary of each other. This modified SR latch is known as D latch.



D-latch



logic Symbol for D-latch

EN	D	Q_n	Q_{n+1}	State
0	x	x	Q_n	No change
1	0	x	0	Reset
1	1	x	1	Set

Table

Latches & Flip Flops

Latch	Flipflop
→ Latch is level mode	→ Flip-flop is pulse mode
→ Bistable states	→ Bistable states.
→ latches are controlled by enable signal and they are level triggered.	→ Flipflops are controlled by pulse(or clk and they are Edge triggered..

Level and Edge Triggering.

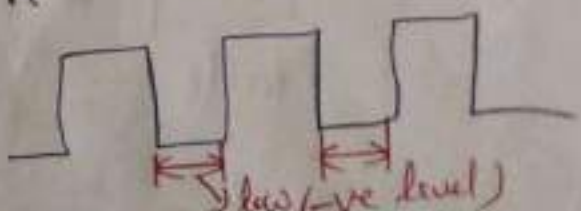
Level The o/p state is allowed to change according to I/ps when active level (either +ve -ve) is maintained at the enable I/p.

+ve level (positive level)

the o/p of latch responds to the I/p changes only when its enable I/p is 1 (High)

Negative (-ve) level.

The o/p of latch responds to the I/p changes only when its enable I/p is 0 (low)



Edge Triggering

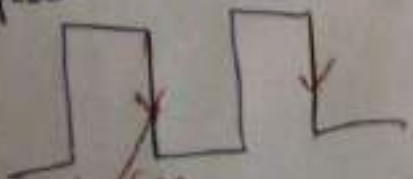
positive edge

o/p responds to the changes in the I/p only at the +ve edge of the clk pulse at the clk I/p

(leading +ve Edge.

Negative Edge Triggering

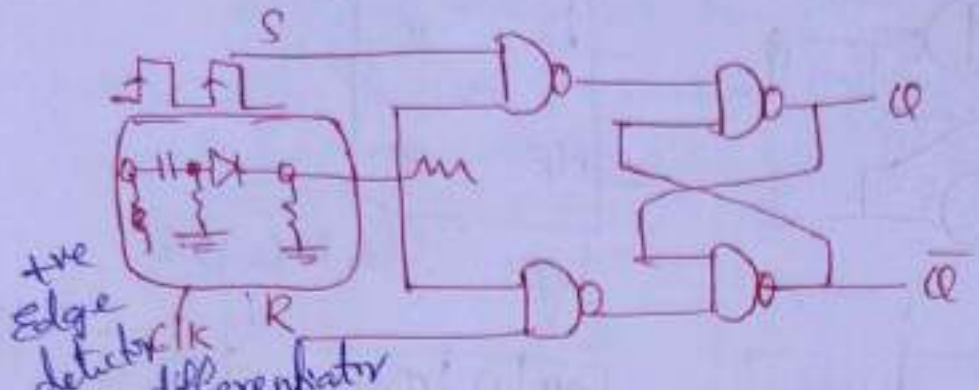
-ve Edge of the clk pulse at the clk I/p.



following -ve Edge.

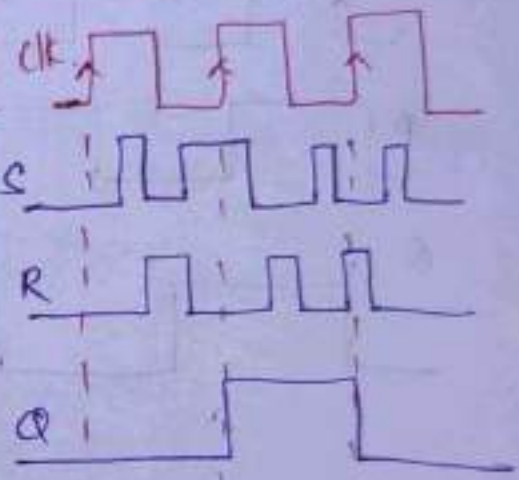
Clocked SR flip flop

+ve Edge Triggered SR FF



Symbol

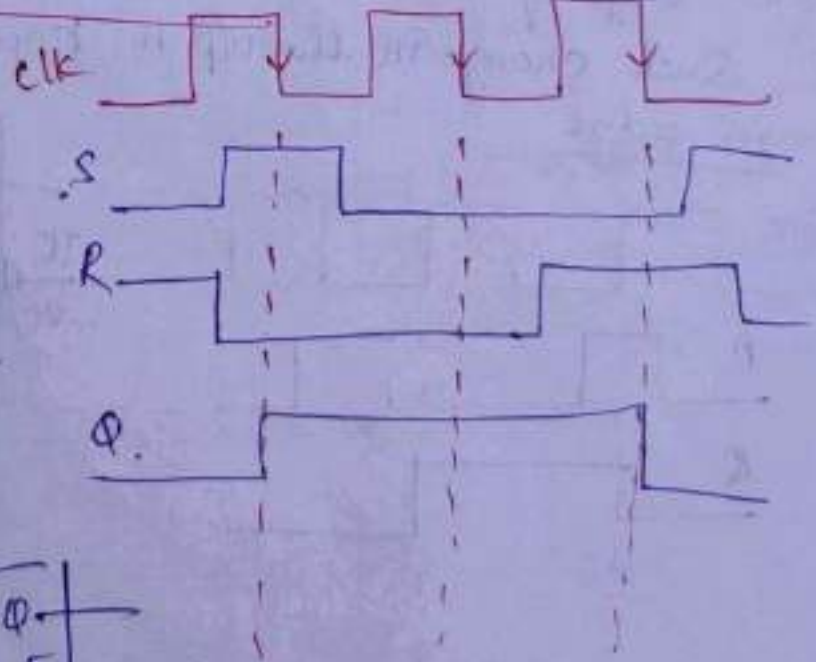
clk	S	R	Q_n	Q_{n+1}	state.
↑	0	0	0	0	No change
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	X	Indeterminate
↑	1	1	1	X	
0	X	X	0	0	No change
0	X	X	1	1	

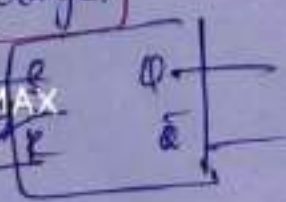


-ve Edge Triggered Clocked SR FF

clk	S	R	Q_n	Q_{n+1}	state.
↓	0	0	0	0	No change
↓	0	0	1	1	
↓	0	1	0	0	Reset
↓	0	1	1	0	
↓	1	0	0	1	Set
↓	1	0	1	1	
↓	1	1	0	X	Indeterminate (or) Invalid
↓	1	1	1	X	
0	X	X	0	0	No change.
0	X	X	1	1	

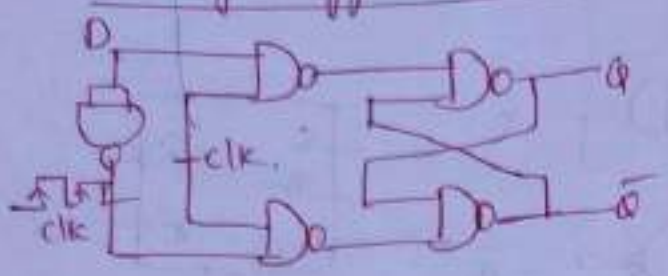
waveforms



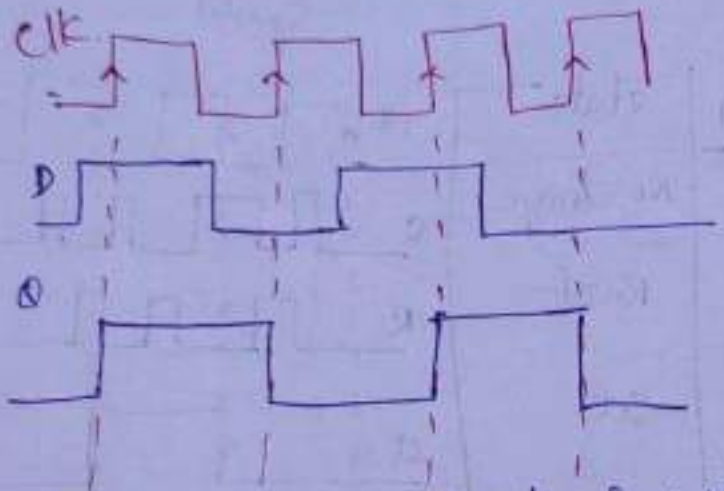


Clocked D FF (Flip-flop)

+ve Edge Triggered D FF

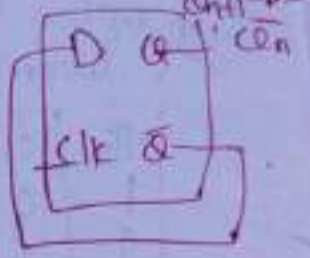


Logic Symbol for DFF.



CP	D	Q_{n+1}
↑	0	0
↑	1	1
0	X	Q_n

Characteristics table

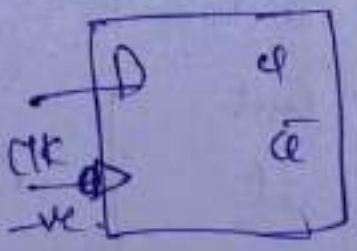


→ $Q_{n+1} = D$. Here the o/p Q_{n+1} is delayed by one clk period. This is called as delay FF.

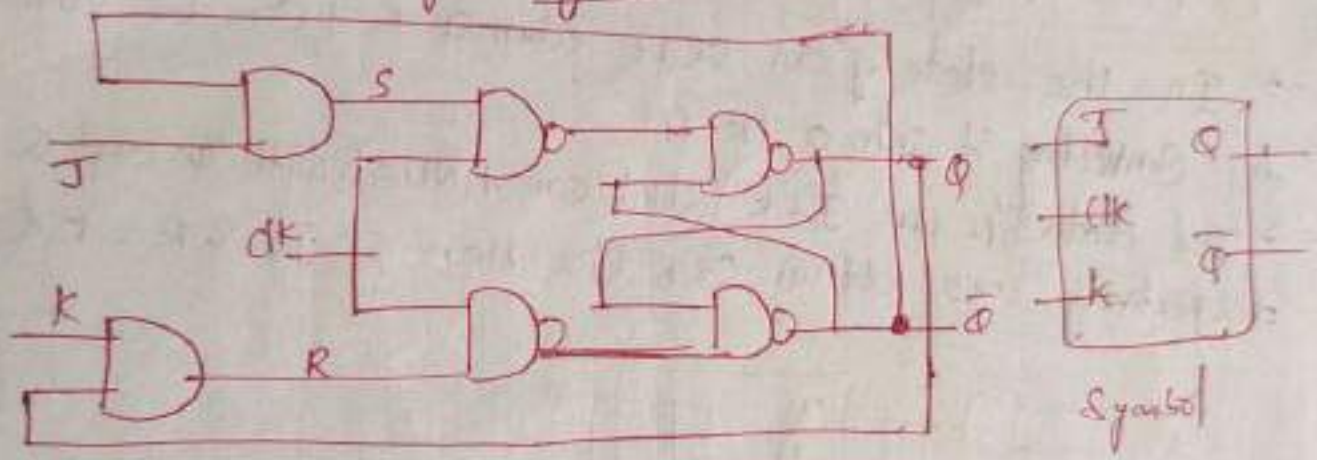
→ If we connect the $\bar{Q}$ o/p of DFF to its D i/p as o/p of DFF will change either from 0 to 1 (or) from 1 to 0 at every +ve edge of the DFF.

→ Such change in the o/p is known as toggling of the FF o/p

-ve edge



clocked JK FF by using AND & SR FF



Symbol

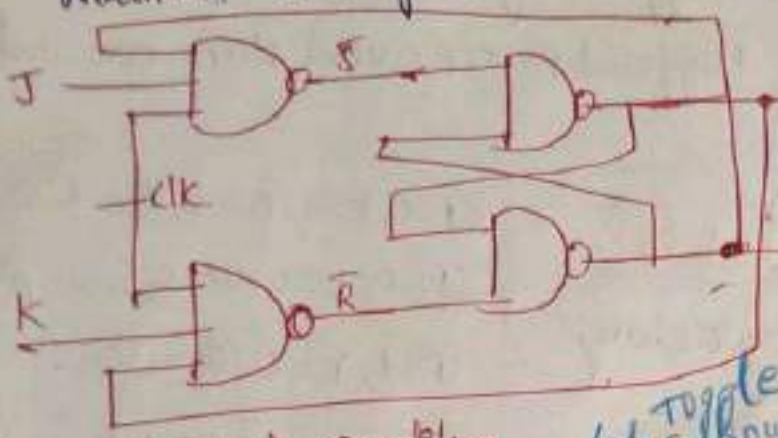
Qn	J	K	Qn+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

=

J	K	(Qn+1)
0	0	Qn
0	1	0
1	0	1
1	1	$\overline{Qn}$

JK Flip-Flop using NAND gates

→ To eliminate invalid state in SRFF using JK
→ modified ckt of JK FF which has only NAND gates.



NC
J=0, K=0 Q=0, $\bar{Q}$=1 (No change)
J=0, K=1, Q=0, $\bar{Q}$=1
J=1, K=0, Q=1, $\bar{Q}$=0
J=0, K=0, Q=1, $\bar{Q}$=0
J=0, K=1, Q=0, $\bar{Q}$=1
J=1, K=0, Q=1, $\bar{Q}$=0 (Set 1)
J=1, K=1, Q=1, $\bar{Q}$=0 Toggle (o/p continuously changing)

Race-Around Condition.

→ In JKFF when J=K=1 the o/p toggles (means 0 to 1 or 1 to 0)
→ Consider Q=0, J=K=1 after a time interval Δt equal to the propagation delay through two NAND gates in series the o/p will change to Q=1.
→ After another time interval of Δt the o/p will change back to Q=0
→ this toggling will continue until the FF is enabled & J=K=1
→ At the end of clk pulse the FF is disabled and the value of Q is uncertain.
→ this situation is referred to as the Race-around Condition.
→ when $t_p \geq \Delta t$ by keeping $t_p < \Delta t$ we can avoid race around condition.
→ we can keep $t_p < \Delta t$ by keeping the duration of edges less than Δt
→ more practical method for over coming this difficulty is the use of Master-slave configuration.

1) clk on time < propagation delay } } Impossible
Race can be avoid

2) If FF is made to toggle only one in clk period

3) By using +ve edge triggered FF to avoid Race around Condition by using

clk	J	K	Qn	Qn+1	State
↑	0	0	0	0	No change
	0	0	1	1	
	0	1	0	0	Reset
	0	1	1	0	
	1	0	0	1	Set
	1	0	1	1	
	1	1	0	1	Toggle (Race)
	1	1	1	0	

J=0, K=1, Q=0, $\bar{Q}$=1 Reset

J=0, K=0, Q=0, $\bar{Q}$=1 No change

J=1, K=0, Q=1, $\bar{Q}$=0 Set

J=0, K=0 Q=1, $\bar{Q}$=0

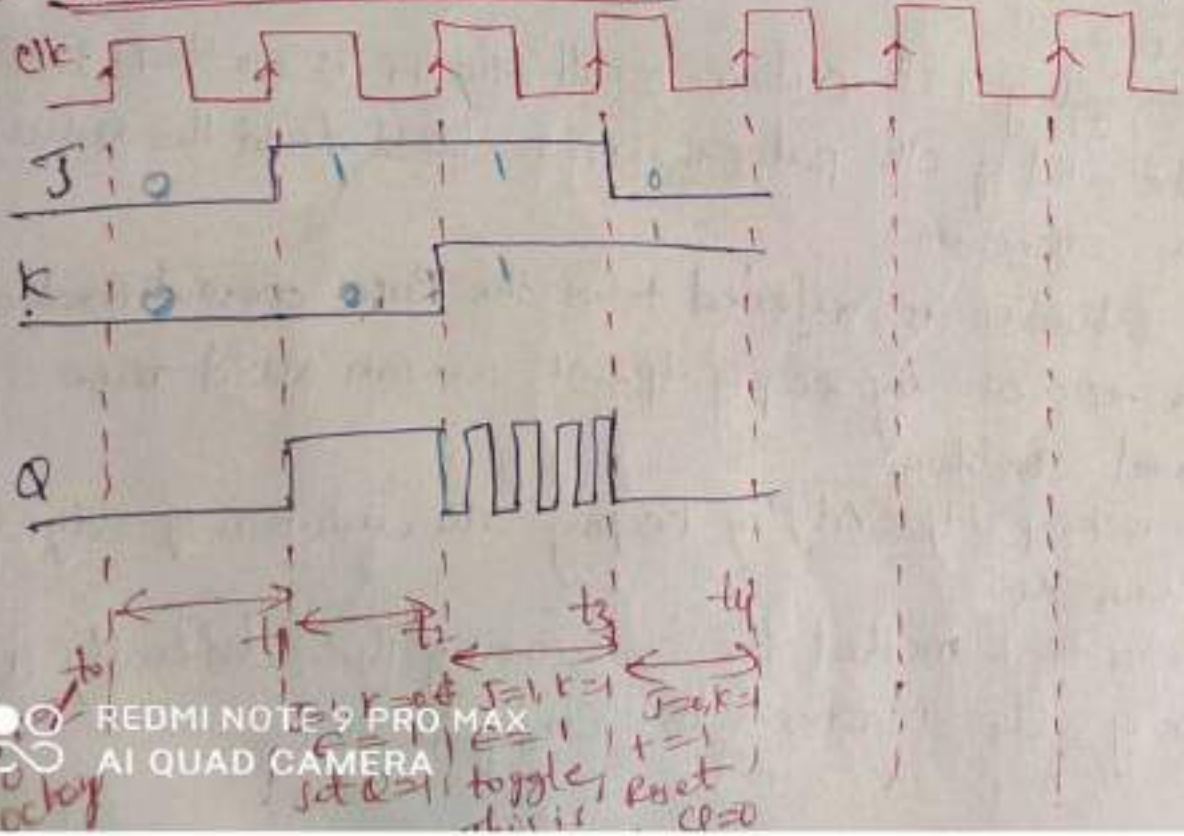
J=1, K=1, Q=0, $\bar{Q}$=1

J=0, K=0, Q=0, $\bar{Q}$=1

J=1, K=1, Q=1, $\bar{Q}$=0 Toggle

(Race) (0 to 1, 1 to 0)

vi) Race-around Condition.



UNIT - V

Memories and Asynchronous sequential Logic

Memories.

A Memory unit is a device to which binary information is transferred for storage and from which information is retrieved when needed for processing. when data processing takes place, information from memory is transferred to selected registers in the processing unit

→ Binary information received from an I/p device is stored in memory and information transferred to an o/p device is taken from memory.

→ Memory unit is a collection of cells capable of storing a large quantity of binary information.

→ There are 2 types of memories that are used in digital systems RAM (Random-access memory) & read-only memory (ROM)

→ RAM stores new information for later use. The process of transferring the stored information out of memory is referred to as a memory read operation.

→ RAM can perform both write & read operations.

→ ROM can perform only the Read operation. This means that suitable binary information is already stored inside memory and can be retrieved or read at any time. However, that information cannot be altered by writing

→ ROM is a programmable logic device (PLD) the binary information that is stored within such a device is specified in some fashion and then embedded within the hardware in a process referred to as programming the device. The word programming here refers to a hardware procedure which specifies the bits that are inserted into the hardware configuration of the device.

A Memory unit is a device to which binary information is transferred for storage and from which information is available when needed for processing.

It is a collection of cells capable of storing a large quantity of binary information.

→ In digital system there are 2 types of memories.
RAM & ROM.

→ It is used to store data that is currently accessed by cpu.
→ The process of storing new information into memory is referred to as a memory write operation.
→ The process of transferring the stored information out of memory is referred to as memory read operation.
→ It performs both read/write operation so it is also called as r/w memory.
→ It is a volatile in nature ie. if power goes off information is lost.
→ Memory stores binary information in groups of bits called word.
→ Communication b/w a memory and its environment is achieved through data i/p & o/p lines, address selection lines and control lines. that specify the direction of transfer.

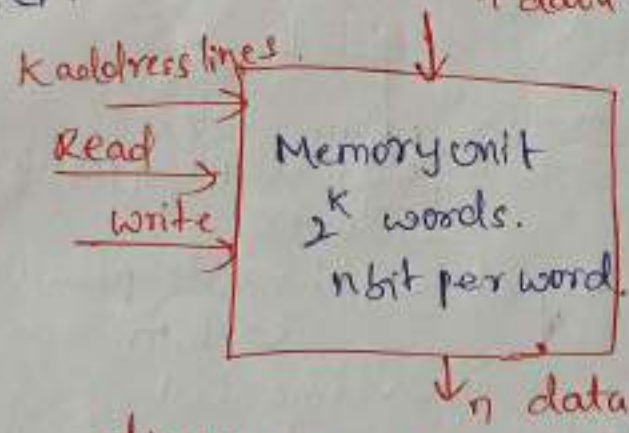
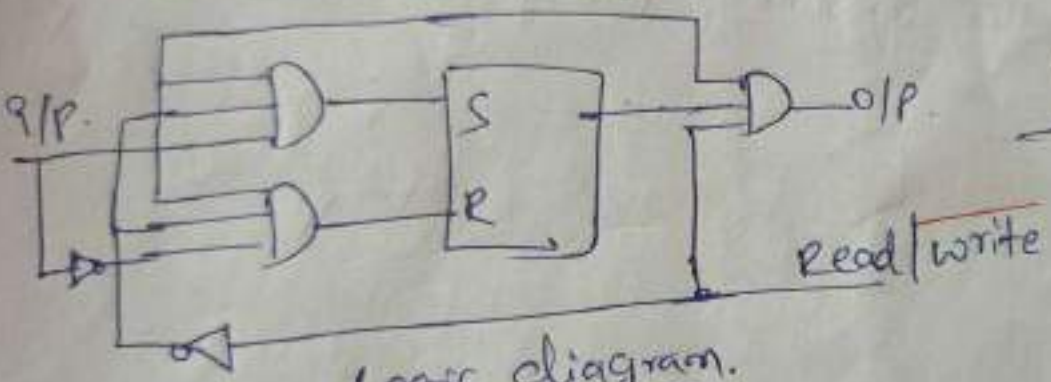


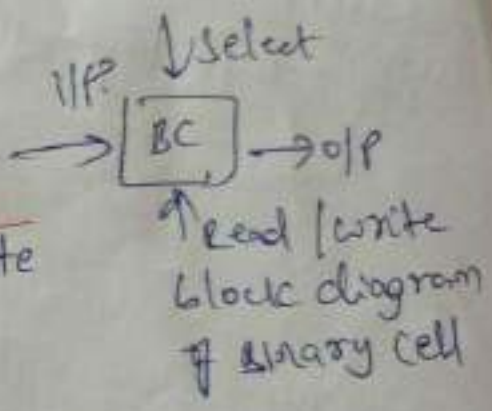
diagram of Memory Unit.

→ n data i/p lines provide the information to be stored in memory and n data o/p lines supply the information coming out of memory. The k address lines specify the particular word chosen among the many available.
→ The 2 control lines i/ps specify the direction of transfer desired. The write i/p causes binary data to be transfered into the memory, and the Read i/p causes binary data to be transferred out of memory.

Memory decoding.



Logic diagram.



Block diagram of Binary Cell

Binary Cell is a basic building block of RAM.

Read/write = 1, binary cell select Read mode

Read/write = 0, " " write mode.

Diagram of 4x4 RAM.

Each word in memory is assigned with identification number called as address ($0 - 2^{k-1}$)

→ Consider for ex: a memory unit with a capacity of 1k words of 16 bits each since $1K = 1,024 = 2^{10}$ and 16 bits constitute 2 bytes, we can say that the memory can accommodate 1,024 = 2k bytes.

Memory address

Binary	Decimal
0000000000	0
0000000001	1
0000000010	2
⋮	⋮
1111111101	1021
1111111110	1022
1111111111	1023.

Memory Content

1011010101010110
1010101111001001
0000111010100110
⋮
1001110100010100
0000110100011110
1101111000100101

Contents of a 1024 × 16 Memory.

the address is determined from the relationship $2^k \geq m$, where m is the total no of words and k is the no of address bits needed to satisfy the relationship.

Write & Read operations.

Write operation → Transfering a new word to be store into memory.

→ Apply the binary address to the desired word to address line.
→ Apply the data bits to be stored in memory to data i/p lines
→ Activate the write i/p.

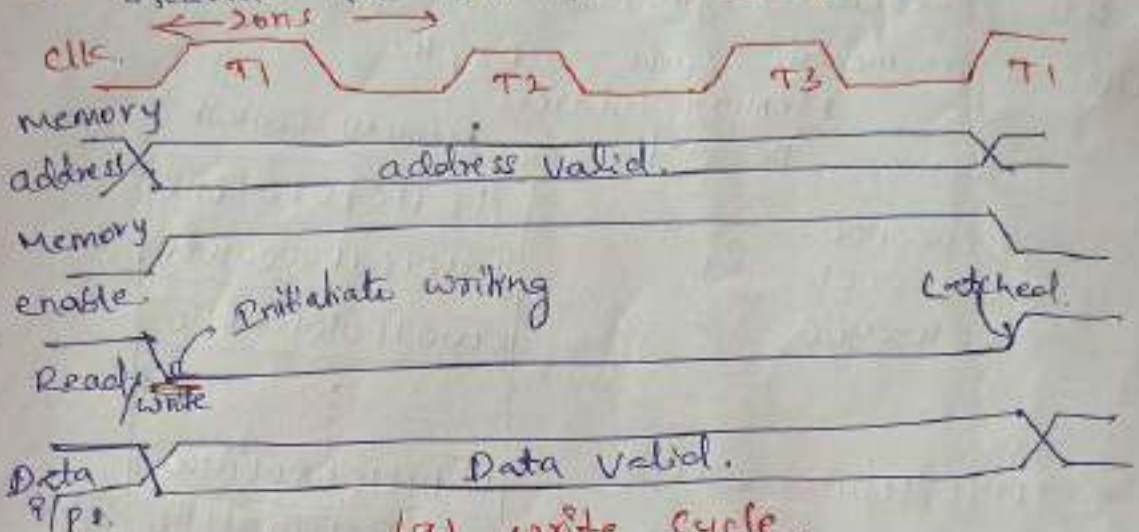
Read operation) data from specified address of memory through data lines is transfered out

→ Apply the binary address of the desired word to the addl lines

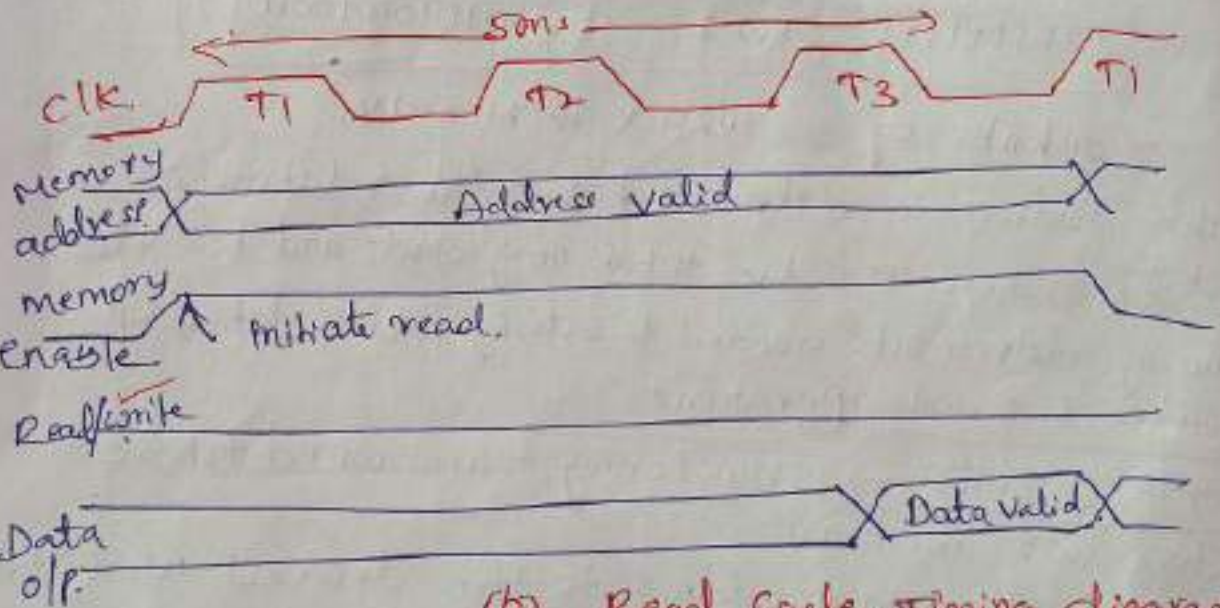
→ Activate the read p/p.

Memory Cycle (Read/write)

Timing diagrame.



(a) write cycle.



(b) Read Cycle Timing diagram.

→ Access time of memory is the time required to select a word and read it. the cycle time of memory is the time required to complete a write operation.

Semi Conductor Memories.

- RAM (Volatile Memory) — easily erased
  - Static RAM
  - Dynamic RAM
- ROM (Non Volatile)
  - Masked programmable
  - PROM
  - EPROM
  - EEPROM

Integrated ckt RAM units are available in two possible operating modes static & dynamic.

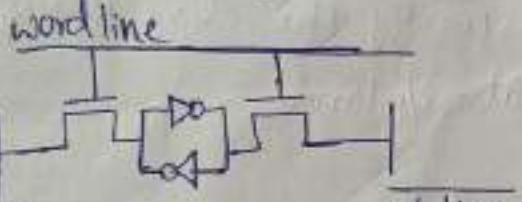
RAM (Random Access Memory)

RAM is volatile memory so when power is off, Data of memory will get lost. [data of RAM will erased]

→

SRAM

→ Static RAM
→ It is made up of Flipflop.

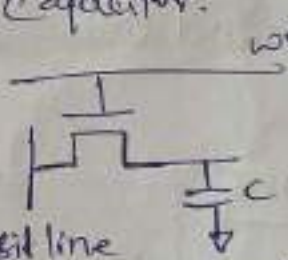


→ Read/write
→ faster
→ No need of periodic refresh
→ High cost
→ Structure density is higher
→ It needs more power
→ It generates more heat
→ It is used in cpu cache

DRAM

Dynamic RAM.

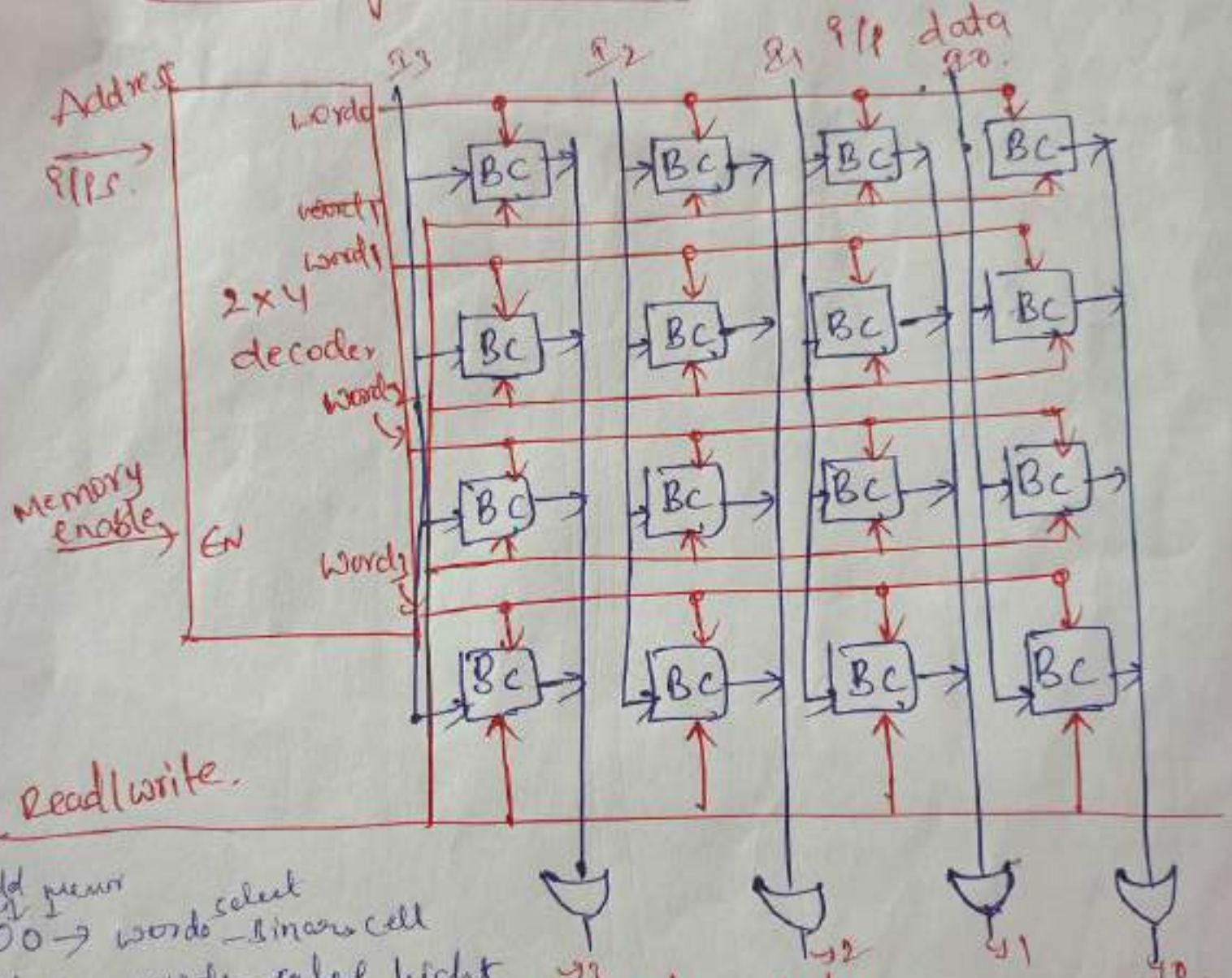
It is made up of transistor & capacitor.



[we have Read/write operation]
→ Slower
→ It needs periodic refresh of capacitor.
→ low cost.
→ Structure density is lower
→ It needs less power
→ It generates less heat.
→ It is used in main memory of Computer.

Read/write = 0, " " write mode.

Diagram of 4x4 RAM.



Add mem
i1 i0
00 → word0 select - binary cell
01 → word1 - select wicht
10 → word2 - BC sebel activated
11 → word3 BC activated

ROM Structure Random decoders + programmable OR gate
= AND gates (fixed) + programmable OR gates.

ROM Size $2^x \times y$ → no of O/Ps
↓
no of I/Ps

Ex finding ROM size.

Ex If I/P = 8, O/P = 4, then what is the size of ROM

$x = 8$, $y = 4$, size = $2^x \times y \Rightarrow 2^8 \times 4 = 1024$ bits

$= \left(\frac{1024}{8}\right)$ bytes. $= 128$ bytes [bits are converted into byte dividing by 8]

ROM classifications

PROM → Programmable Read only Memory.
EPROM → Erasable "
EEPROM → Electrically "Erasable"
Flash → Memory
Mask ROM.

Rom classifications.

1) PROM programmable ROM	2) EPROM Erasable PROM	3) EEPROM Electrically EPROM	4) Flash memory	5) Mask ROM.
→ Initally it is Empty	By uv rays we can earase data.	we can erase data Electrically many times.	Speed of erase is faster than EEPROM	It is ROM only.
→ then we can program it once	→ After that we can reprogram this memory	→ Efficiency of memory will decay with respect to earse. Erase happens bit by bit.	→ Data erase is done 'block by block	→ It is programmed by chip manufacture → This ROM is masked off during photolithography

SPLD (Simple Programmable Logic device) Integrated ckt that contains a large no of gates (AND OR)

AND array followed by OR array

1) PROM — fixed program
2) PLA — prgm prgm
3) PAL — prgm fixed.

→ used to configure different types of PLD using PLD

PROM AND OR

Design a combinational ckt using ROM. The ckt accept a 3bit number and o/p a binary number equal to the square of the i/p number.

2bit o/p = 4
3bit o/p = 9
4bit o/p = 16
5bit o/p = 25
6bit o/p = 36
7bit o/p = 49

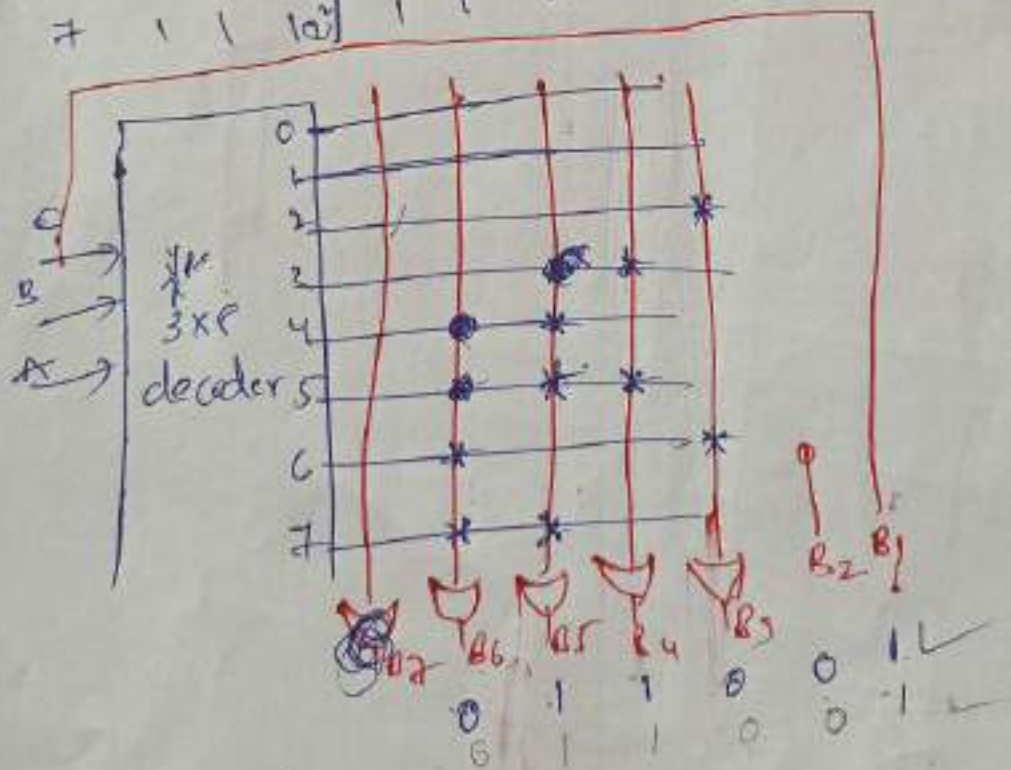
$2^5 = 32$ (0–31)
$2^6 = 64$ (0–63)

	A	B	C	B_6 (32)	B_5 (16)	B_4 (8)	B_3 (4)	B_2 (2)	B_1 (1)
0	0	0	0 (0)	0	0	0	0	0	0
1	0	0	1 (1)	0	0	0	0	0	1
2	0	1	0 (2)	0	0	0	1	0	0
3	0	1	1 (3)	0	0	1	0	0	1
4	1	0	0 (4)	0	1	0	0	0	0
5	1	0	1 (5)	0	1	1	0	0	1
6	1	1	0 (6)	1	0	0	1	0	0
7	1	1	1 (7)	1	1	0	0	0	1

Here the values are zero discount. we can reduce the no of gates for implementation

$[B_1 = C]$

Check the o/p. Consider i/p 101



PLA [Programmable Logic Array]

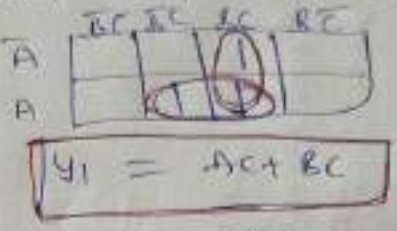
Programmable Logic Array as programmable AND gates and programmable OR gates

→ It is fixed architecture logic device with programmable AND gates and programmable OR gates

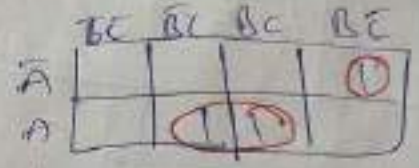
Steps to program PLA

Step1 :- Find boolean function from truth table

A	B	C	y_1	y_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1



$y_1 = AC + BC$

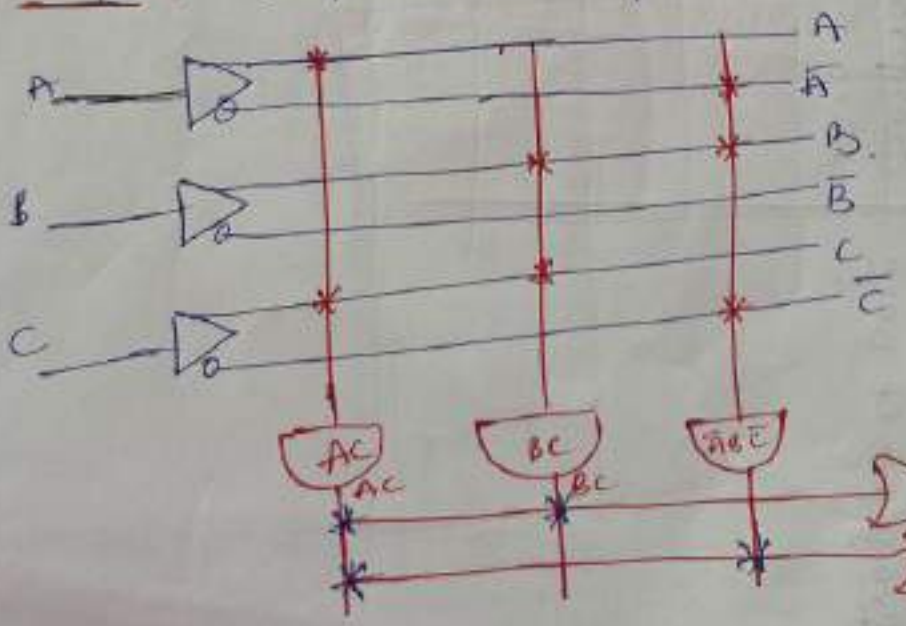


$y_2 = AC + \bar{A}B\bar{C}$

Step2 :- Identify no of i/p buffer

We have 3 i/p variables, so we need no of i/p buffers =3

Step3 :- Implementation of boolean function in PLA



$y_1 = AC + BC$

$y_2 = AC + \bar{A}B\bar{C}$

no of prgm AND gates = no of product terms in boolean fun.

Programmable Array Logic :-

In its Architecture includes programmable AND gates & fixed OR gates.

→ It is most commonly used PLD.

→ It has programmable AND gates and fixed OR gates.

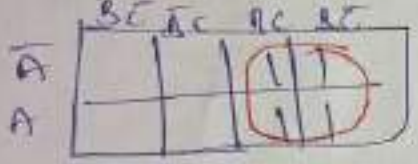
→ It provides simple structure & It has issues regarding flexibility.

Ex:

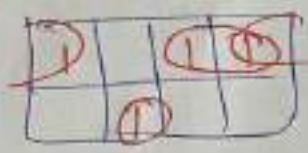
$x = \Sigma m(2, 3, 6, 7)$

$y = \Sigma m(0, 2, 3, 5)$

$Z = \Sigma m(1, 6, 7)$

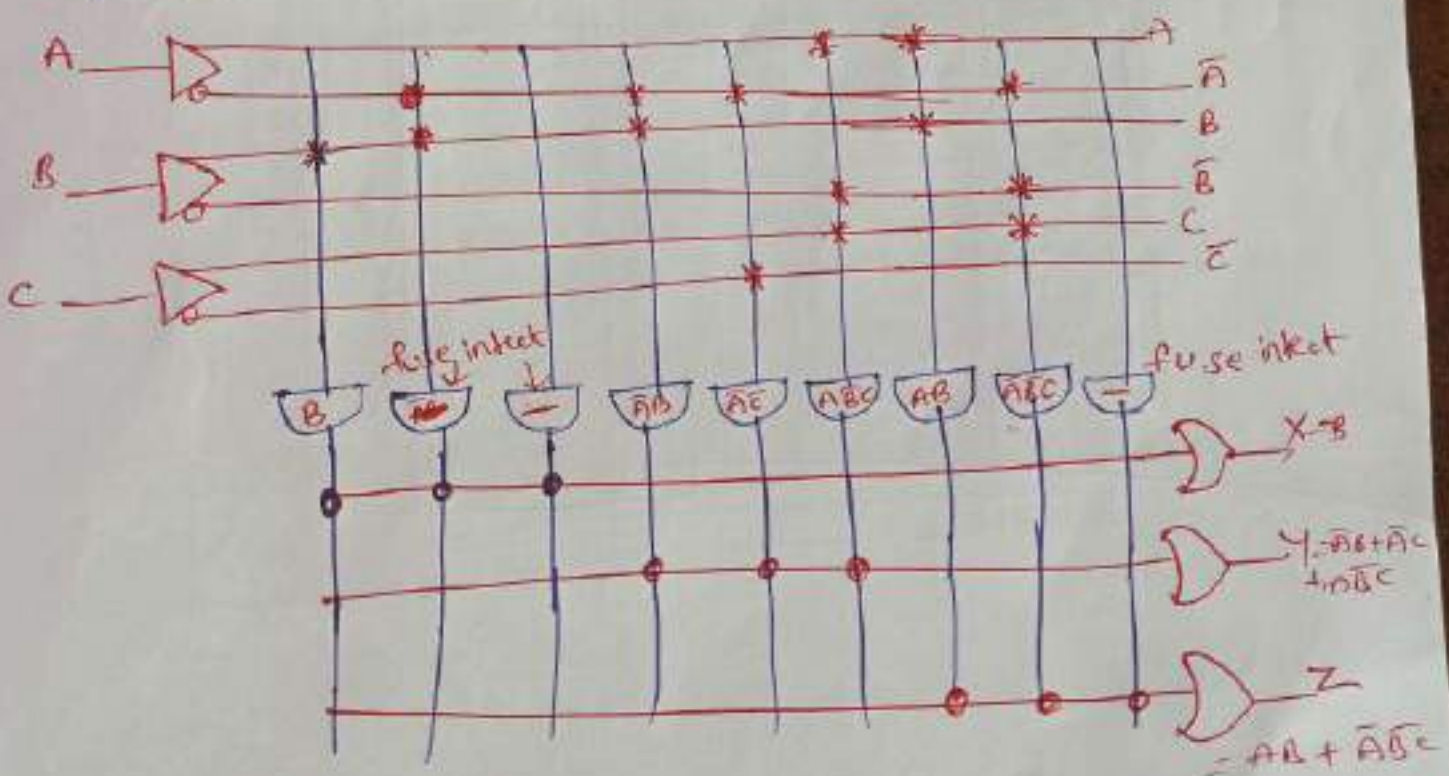


$X = B$



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

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



Hazards

Hazards

Unwanted switching transients that may appear at the o/p of a ckt are called Hazards.

→ Hazards cause the ckt to malfunction.

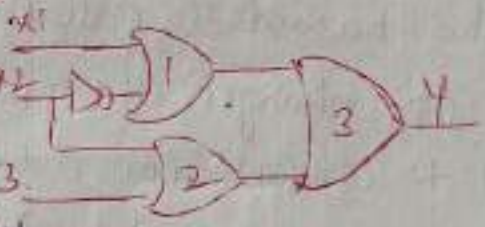
→ The main cause of hazards is the different propagation delays at different paths.

→ Hazards occur in combinational ckt where may cause a temporary false o/p value.

→ when such combinational ckts are used in asynchronous sequential ckts they may result in transition to wrong stable state.

Hazards in Combinational ckts

→ Assume Initially i/ps x_1 & $x_3 = 0$ &
$x_2 = 1$.



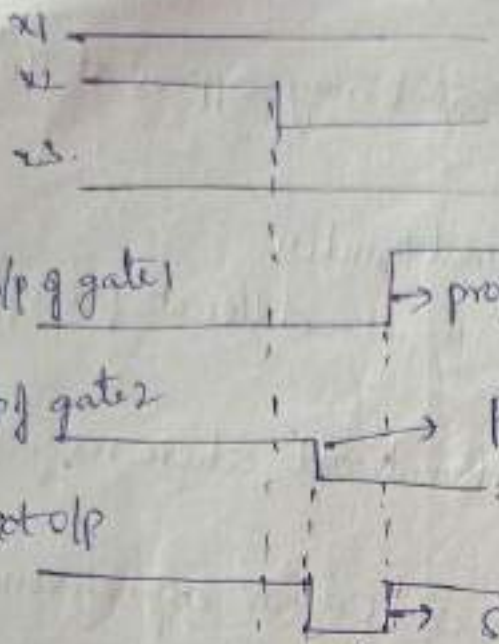
This case o/p of gate1 = 0
o/p of gate2 = 1 & o/p of ckt = 0

→ Now change in $x_2 = 0$
o/p of gate1 = 1
gate2 = 0.

→ leaving o/p at 1 however the o/p momentarily goes to '0' if the propagation delay through the Inverter is taken into considerations.

→ the delay in the inverter causes the o/p of gate 2 to change to '0' before the o/p of gate 1 changes to 1.

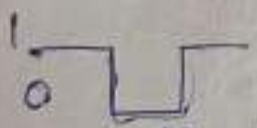
→ In this case both i/ps of gate 3 are equal to '0' causing the o/p to go to '0' for the short time equal to the propagation delay of the Inverter.



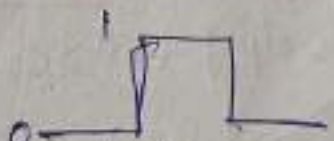
→ In combinational ckt, if dp goes momentarily '0' when it should remain a 1. the hazard as static-1 hazard.

→ o/p goes to momentarily 1 when it should remain a '0' The hazard is called as static-0 hazard.

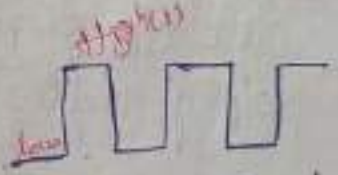
→ o/p changes 3 (or) more times when it should change from 1 to 0. (or) from 0 to 1.



static-1.



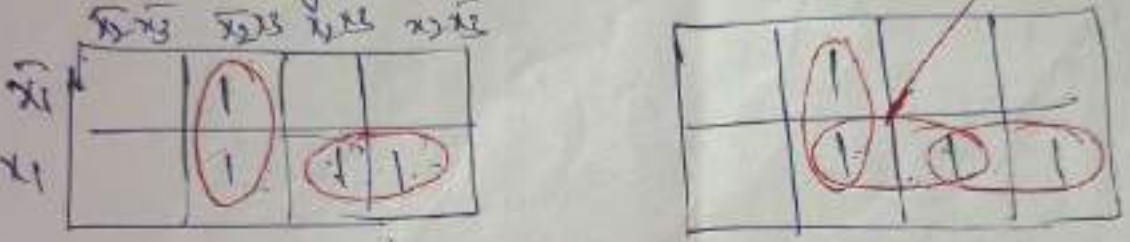
static-0


Dynamic hazard

2 and Hazard free Realization

Hazards can be eliminated by enclosing two minterms (or) maxterms in eqn

→ for ex: if the ckt has minterms $x_1x_2 + \bar{x}_2x_3$ then these two minterms must be enclosed by introducing another minterm x_1x_3.



$\bar{x}_2x_3 + x_1x_2$ $\quad$ $y = x_1x_2 + \bar{x}_2x_3 + x_1x_3$

eliminating hazards.

Ex: obtain hazard free ckt.

$f(A, B, C, D) = \Sigma m(1, 3, 6, 7, 13, 15)$

$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

enclosing

$\bar{A}\bar{B}D + \bar{A}BC + ABD + \bar{A}CD + BCD$

Hazards in sequential ckts [Essential Hazards]

In synchronous sequential ckts with combinational ckt design hazards are of no concern since momentary erroneous signals are not generally troublesome

In synchronous sequential ckts If false output is feedback to the I/p then ckt enters into wrong stable state.

Essential hazards
Unequal propagation delays originating at same point - overcome by adjusting the amount of propagation delay.

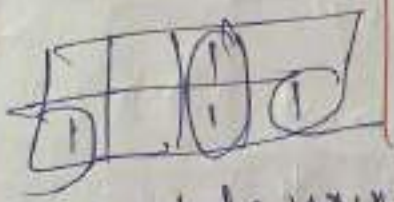
Let us consider the ckt

$Y = x_1x_2 + x_2'y$

Transition table

y \ x_1x_2	00	01	11	10
0	(0)	(0)	1	(0)
1	(1)	0	(1)	(1)

Output map

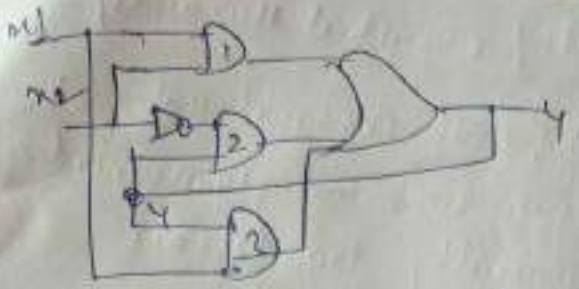


If ckt is in total stable state $y x_1 x_2 = 111$ & x_2 changes $1 \to 0$ the next stable state should be 110.

However because of the hazard o/p y may go to 0 momentarily if the false signal feedback into gate before the o/p of the inverter goes to 1, the o/p of gate 2 will remain at 0 and the ckt will switch to the incorrect total stable state 010.

This malfunction can be eliminated by adding an extra gate to the ckt. ie including redundant term

$$Y = x_1x_2 + x_2'y + x_1y$$



Implementation with SR Latches

Another way to avoid static hazards in asynchronous Seq ckt is to implement the ckt with SR latches. A momentary 1 signal applied to S or R i/ps of a NAND latch will have no effect on the state of the latch. Ily a momentary '0' signal applied to S & R i/ps of NOR latch will have no effect on state of the latch.

Consider a NAND SR latch with the following Ex

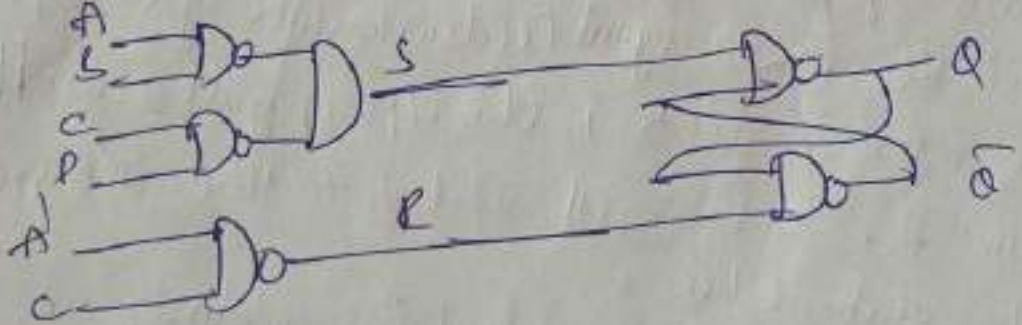
$S = AB + CD$

$R = A'C$

∴ It is a NAND latch we must apply complemented values to the i/p's ie

$S = (AB + CD)' = (AB)'(CD)'$

$R = (A'C)'$



Error detecting and correcting codes:

When information is transmitted from once circuit to another circuit and error may be occured this means '0' maly be changed to '1' and '1 changed to '0' that is 1 to 0 and 0 to 1.

→ one of the method to detect the error is parity bit method. A parity bits is used for detecting errors during transmission of binary information.

→ parity bit is an extra bit included with a binary message to make no of 1's either even or odd.

Even parity:

In even parity the added parity bit will make the total no of 1's and even count.

Odd parity:

In odd parity the added parity bit will make the total no of 1's and odd count.

Hamming code:

Hamming code provides detection of a bit error and also identifies which bit is in error so that it can be corrected this hamming code is called either detecting and correcting code

Number of parity bits:

No of parity bits depend on the no of information bikit

is determined by $2^p \geq x+p+1$:

Here x is the no of information bits.

p is the no of parity bits

Ex: Four 4 bit message 1011 find the hamming code consider even parity?

Sol 1 0 1 1

$x = 4$

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

if $p=1$ $2^1 \geq 4+1+1$ not satisfied

$p=2$ $2^2 \geq 4+2+1$ not satisfied

$p=3$ $2^3 \geq 4+3+1$ condition satisfied

$p = 3$

Total = x+p

Total no of bits = x+p

4+3

T = 7

locating parity bits:

3 parity bits are numbered in ascending power 2.

That is 2^2 2^1 2^0

↓ ↓ ↓

4 2 1

p^4 p^2 p^1

Assigning values to parity bits:

For P_1 LSB should be 1

For P_2 Middle bit should be 1

For P_4 third bit should be 1

D_7	D_6	D_5	P_4	D_3	P_2	P_1
7	6	5	4	3	2	1
111	110	101	100	011	010	001
1	0	1	0	1	0	1

1st bit position $P_1 \Rightarrow (1,3,5,7) \Rightarrow (111) \Rightarrow$ Total no of 1's is odd

$P_1 = 1$

2nd bit position $P_2 \Rightarrow (2,3,6,7) \Rightarrow (11) \Rightarrow$ Total no of 1's is even

$P_2 = 0$

3rd bit position 4 $P_3 \Rightarrow (4,5,6,7) \Rightarrow (11) \Rightarrow$ Total no of 1's is even

$P_4 = 0$

Hamming bit 1010101

Determine the single error correcting code for the information code 10111 for odd parity?

Sol 10111

$x = 5$

To find parity bits

$2^P \geq x + P + 1$

if $P=1$ $2^1 \geq 5+1+1$ not satisfied

$P=2$ $2^2 \geq 5+2+1$ not satisfied

$P=3$ $2^3 \geq 5+3+1$ not satisfied

$P=4$ $2^4 \geq 5+4+1$ condition satisfied

$P = 4$

Total $x+P$

$= 5+4 = 9$

locating parity bits

2^3 2^2 2^1 2^0
↓ ↓ ↓ ↓
8 4 2 1
P^8 P^4 P^2 P^1

D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
9	8	7	6	5	4	3	2	1
1001	1000	0111	0110	0101	0100	0011	0010	0001
1	0	0	1	1	1	1	1	0

1st bit position $P_1 \Rightarrow (1,3,5,7,9) \Rightarrow (1,1,1) \Rightarrow$ Total no of 1's odd

$P_1 = 0$

2nd bit position $P_2 \Rightarrow (2,3,6,7) \Rightarrow (11) \Rightarrow$ Total no of 1's even

$P_2 = 1$

3rd bit position $P_3^4 \Rightarrow (4,5,6,7) \Rightarrow (1,1) \Rightarrow$ Total no of 1's even

$P_3^4 = 1$

4th bit position $P_8^4 \Rightarrow (8,9) \Rightarrow (1) \Rightarrow$ Total no of 1's odd

$P_8 = 0$

Hamming code 100111110

Encode the information character 01101110101 according to the 15 bit Hamming code for even parity?

01101110101

x=11

To find parity bit

$2^P \geq x+p+1$

if p=1 $2^1 \geq 11+1+1$ not satisfied

$P=2$ $2^2 \geq 11+2+1$ not satisfied

$P=3$ $2^3 \geq 11+3+1$ not satisfied

$P=4$ $2^4 \geq 11+4+1$ condition satisfied

$P=4$

Total $= x+P$

$= 11+4$

$= 15$

2^4	2^3	2^2	2^1	2^0
↓	↓	↓	↓	↓
16	8	4	2	1
P^{16}	P^8	P^4	P^2	P^1

D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
0	1	1	0	1	1	1	1	0	1	0	1	1	1	0

1st bit position $P_1 \Rightarrow (1,3,5,7,9,11,13,15) \Rightarrow (1111) \Rightarrow$ Total no of 1's

even $P_1 = 0$

2nd bit position $P_2 \Rightarrow (2,3,6,7,10,11,14,15) \Rightarrow (11111) \Rightarrow$ Total no of 1's

odd $P_2 = 1$

3rd bit position $P_4 \Rightarrow (4,5,6,7,12,13,14,15) \Rightarrow (1111) \Rightarrow$ Total no of 1's

odd $P_4 = 1$

4th bit position $P_8 \Rightarrow (8,9,10,11,12,13,14,15) \Rightarrow (11111) \Rightarrow$ Total no of 1's

odd $P_8 = 1$

Hamming code 0110111101011110

Assume that the even parity having code in example 0110011 is transmitted and that 0100011 is recieved The reciever does not know what was transmitted determine bit location where error has occured using recieving code

0100011

$x=7$

To find parity bits

$2^P \geq x+P+1$

if P=1 $2^1 \geq 7+1+1$ not satisfied

P=2 $2^2 \geq 7+2+1$ not satisfied

P=3 $2^3 \geq 7+3+1$ not satisfied

P=4 $2^4 \geq 7+4+1$ condition satisfied

P= P=4

Total = x+P

~~= 7+4 =~~

Total = 7 = x+P

D_7	D_6	D_5	P_4	D_3	P_2	P_1
7	6	5	4	3	2	1
111	110	101	100	011	010	001
0	1	0	0	0	1	1

$P_1 \Rightarrow (1,3,5,7) \Rightarrow (1) \Rightarrow$ Total no of 1's odd

$P_1=1$

$P_2 \Rightarrow (2,3,6,7) \Rightarrow (11) \Rightarrow$ Total no of 1's even

$P_2=0$

$P_4 \Rightarrow (4,5,6,7) \Rightarrow (1) \Rightarrow$ Total no of 1's odd

$P_4=1$

$P_4\ P_2\ P_1$

1 0 1 $\Rightarrow$ 5th position is error

given Hamming code: 7 6 5 4 3 2 1 — 0 1 0 0 0 1 1

0110011

Hamming code 10110101 is recieved correct if any errors there are 4 parity bits & odd parity is used.

10110101

$x = 9$

To find parity bits

$2^P \geq x + P + 1$

if $P=1$ $\quad 2^1 \geq 9+1+1$ not satisfied

$P=2$ $\quad 2^2 \geq 9+2+1$ not satisfied

$P=3$ $\quad 2^3 \geq 9+3+1$ not satisfied

$P=4$ $\quad 2^4 \geq 9+4+1$ condition satisfied

Total $= 9 = x + P$

D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
9	8	7	6	5	4	3	2	1
1001	1000	0111	0110	0101	0100	0011	0010	0001
1	0	1	1	0	1	1	0	1

$P_1 \Rightarrow (1,3,5,7,9) \Rightarrow (1111) \Rightarrow$ Total no of 1's even

$P_1 = 1$

$P_2 \Rightarrow (2,3,6,7) \Rightarrow (111) \Rightarrow$ Total no of 1's odd $\quad P_2 = 0$

$P_4 \Rightarrow (4,5,6,7) \Rightarrow (1111) \Rightarrow$ Total no of 1's even

$P_4 = 0$

$P_8 \Rightarrow (8,9) \Rightarrow (1) \Rightarrow$ Total no of 1's odd

$P_8 = 0$

$P_8\ P_4\ P_2\ P_1$

$0\ 0\ 0\ 1 \Rightarrow$ 1st position

given Hamming code 10110110(1)

101101100

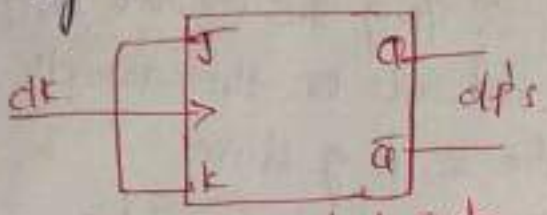
Determine which bit any error in the even parity hamming code character 1100111 decode the message. 1st

Hamming code sequence for 11 bit message and correct if it is necessary even parity.

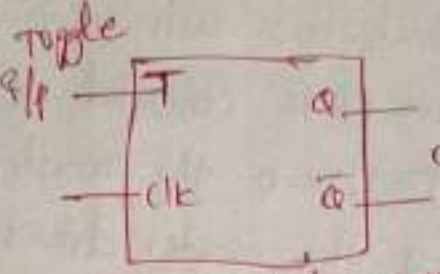
10100101110101011 6th

Toggle Flip-flop (T-FF)

Toggle FF is a basically a J-K ff with J & K terminals permanently connected together. It has only one i/p denoted by T.

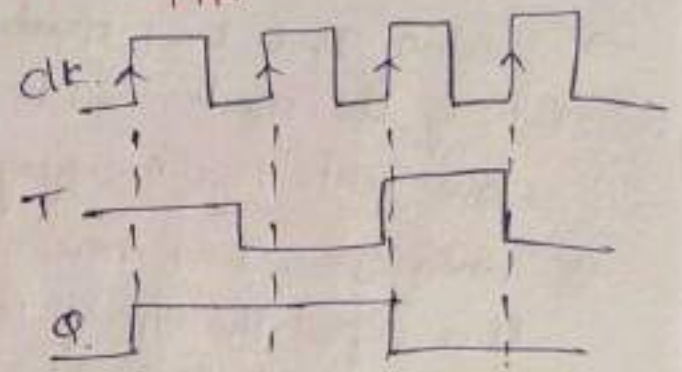


JK FF is converted into T FF

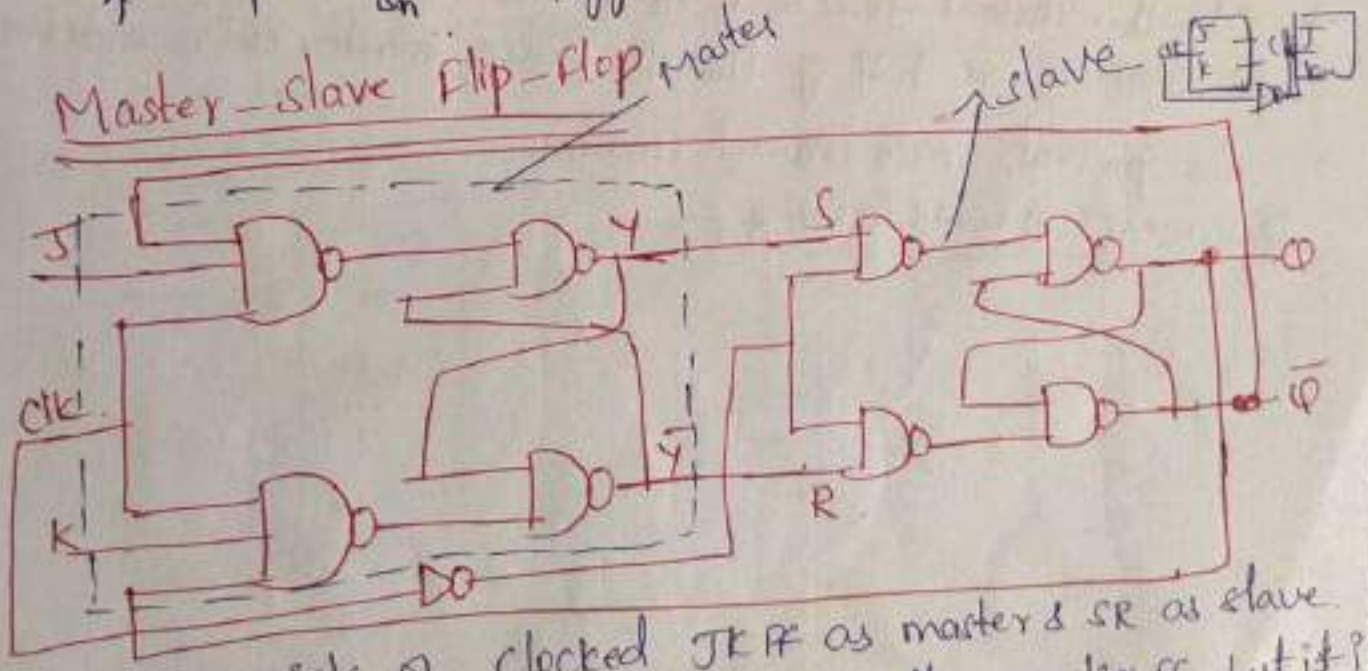


logic Symbol of +ve edge Triggered TFF

Clk	T	Qn+1	State
↑	0	Qn	No change
↑	X	Qn	No change
1	X	Qn	No change
0	X	Qn	No change
↑	1	Q̄n	Toggle



Master-slave Flip-flop



→ It consists of clocked JK FF as master & SR as slave.
→ clk signal is connected directly to the master FF. but it is connected through inverter to the slave FF

The information present at the J & K i/ps is transmitted to the o/p of master FF on the +ve clk pulse and it is held there until the -ve clk pulse occur.

→ After which it is allowed to pass through to the o/p of slave FF.

→ o/p of slave FF is connected as a 3rd i/p of the master JK FF.

→ when J=1, K=0 the master stet sets on the +ve clk. the high Y o/p of the master drives the S i/p of slave.

→ So -ve clk slave sets, copying the action of master.

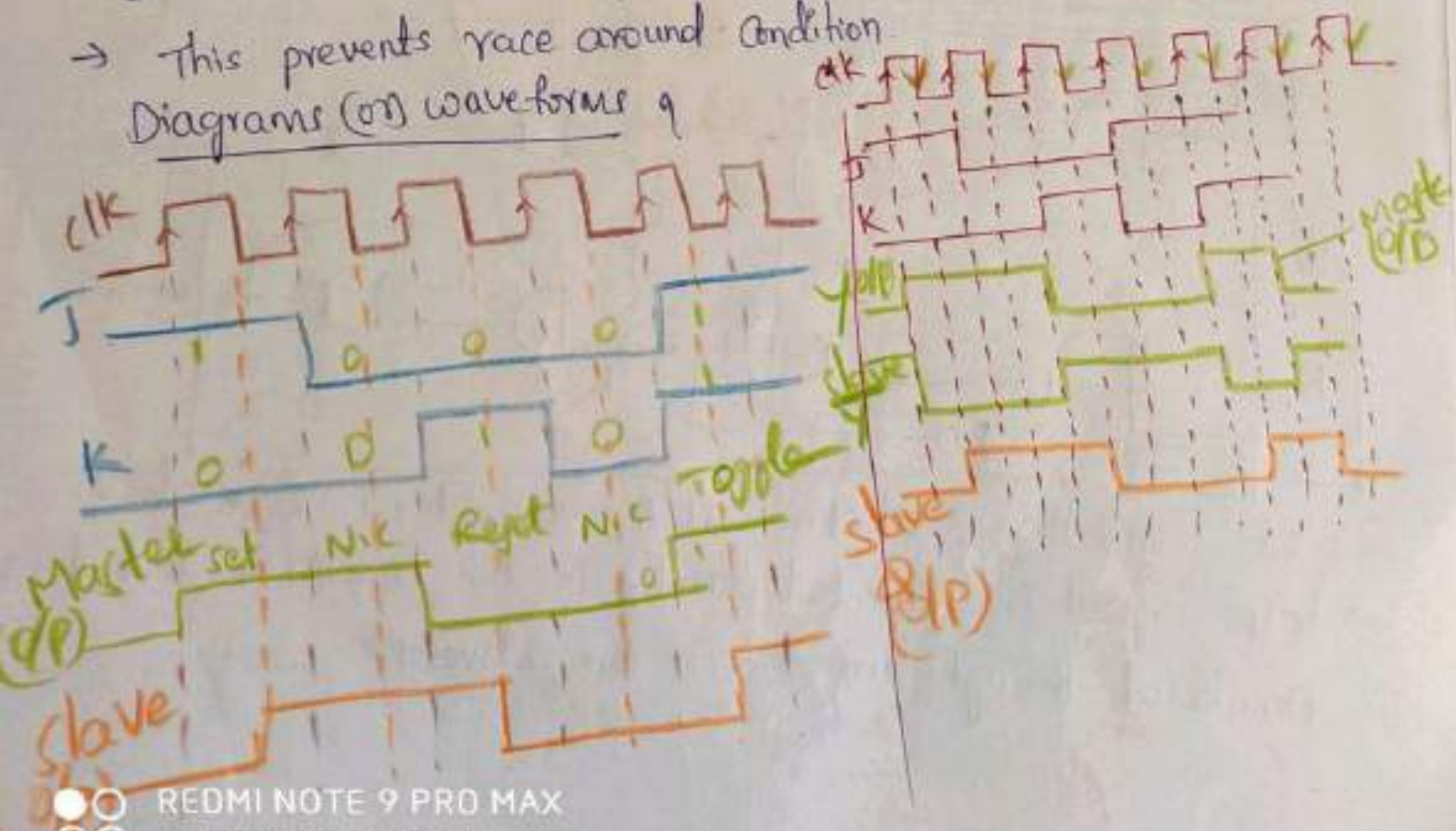
→ when J=0 K=1 master resets on the +ve clk the high $\bar{Y}$ o/p of the master goes to the R i/p of slave so. At(-ve) clk slave resets, again copying the action of the master.

→ when J=1, K=1 master toggles on the +ve clk and slave then copies the o/p of master on the -ve clk.

→ At this instant feed back i/p to the master FF are complemented but as it is -ve half of the clk pulse master FF is inactive.

→ This prevents race around condition

Diagrams (or) waveforms :



Characteristic and excitation table for JK FF

Truth table

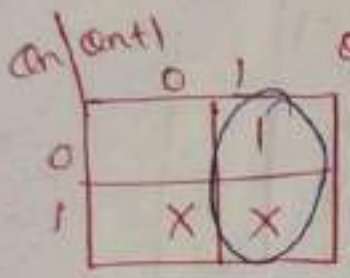
Clk	J	K	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	$\bar{Q}_n$ (toggle)

Characteristic table

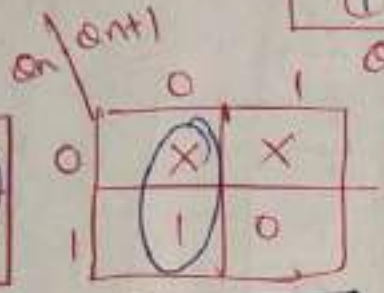
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Excitation table

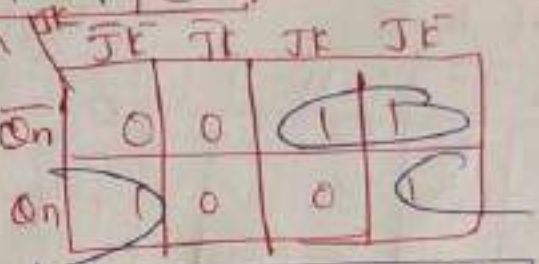
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



For $J = Q_{n+1}$



For $K = \bar{Q}_{n+1}$



$Q_{n+1} = \bar{Q}_n J + Q_n \bar{K}$

SR flip-flop

Clk	S	R	Q_{n+1}
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	Invalid

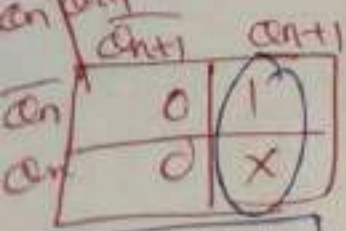
Truth table

Characteristic table

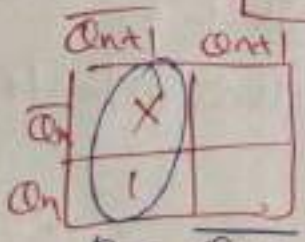
Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	X
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	X

Excitation

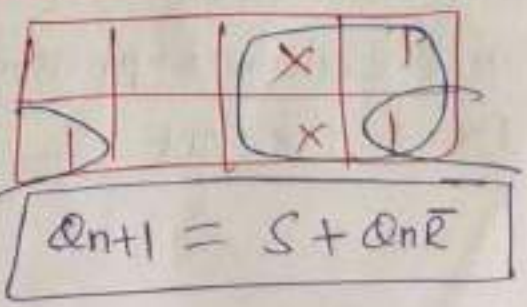
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0



$S = Q_{n+1}$



$R = \bar{Q}_{n+1}$



$Q_{n+1} = S + Q_n \bar{R}$

D FF

Truth table

Clk	D	Qn+1
0	X	Qn
1	0	0
1	1	1

Circuit table

Qn	D	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

Excitation table

Qn	Qn+1	D
0	0	0
0	1	1
1	0	0
1	1	1

$D = Q_{n+1}$

T FF

Truth table

Clk	T	Qn+1
0	X	Qn
1	0	Qn
1	1	$\bar{Q_n}$ (trig)

Circuit table

Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

Excitation table

Qn	Qn+1	T
0	0	0
0	1	1
1	0	1
1	1	0

Q_n \ T	T	T
$\bar{Q_n}$		1
Q_n	1	

Q_n \ Q_{n+1}	Q_{n+1}	Q_{n+1}
$\bar{Q_n}$		1
Q_n	1	

$Q_{n+1} = \bar{Q_n}T + Q_n\bar{T}$

$Q_{n+1} = T \oplus Q_n$

$T = Q_n \oplus Q_{n+1}$

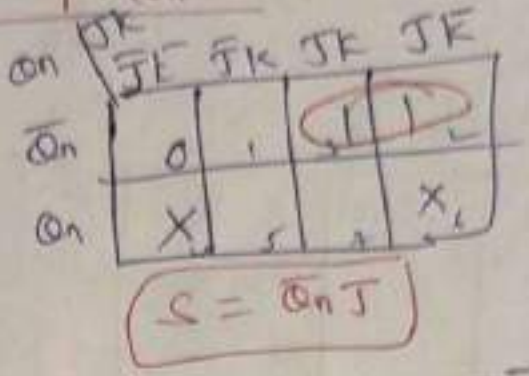
Flip-Flop Conversion steps

1) Identify available and required FF
2) Make characteristic table for required FF
3) Make excitation " available FF
4) Write boolean expression for available FF
5) Draw the ckt

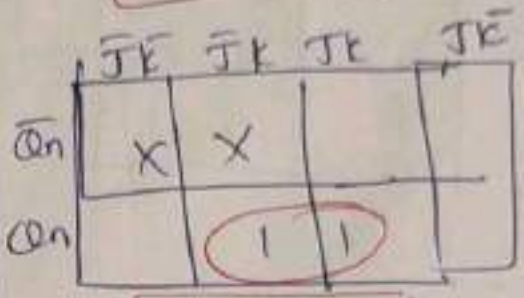
① SR flip-flop to JK flip-flop Conversion.

Required ff = JK
available ff = SR.

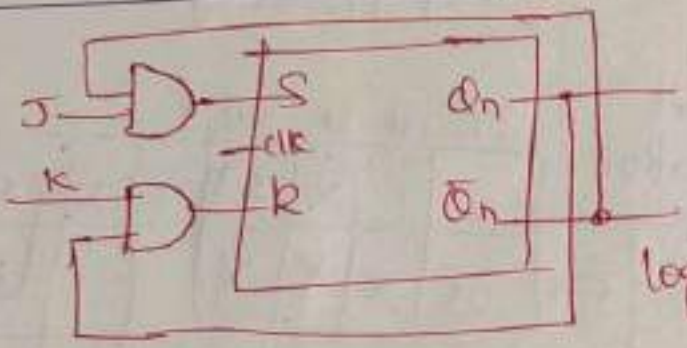
Q_n	J	K	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	0	0	X
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	1	X	0
1	0	1	0	0	1
1	1	0	1	X	0
1	1	1	0	0	1



$S = \bar{Q}_n J$



$R = Q_n K$

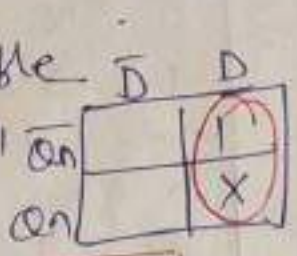


logic diagram of JK

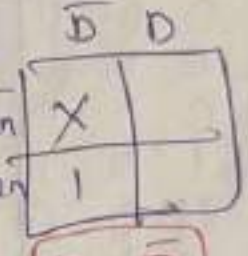
② SR to D

Required D characteristic table
Available SR excitation table

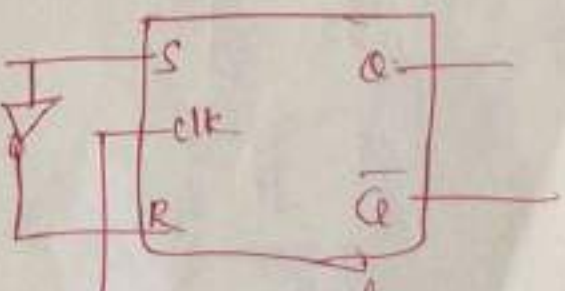
Q_n	D	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	1	0
1	0	0	0	1
1	1	1	X	0



$S = D$



$R = \bar{D}$



D ff logic diagram.

③ SR TO TFF

Required T characteristic. Karnaugh's S & R.
Available SR excitation

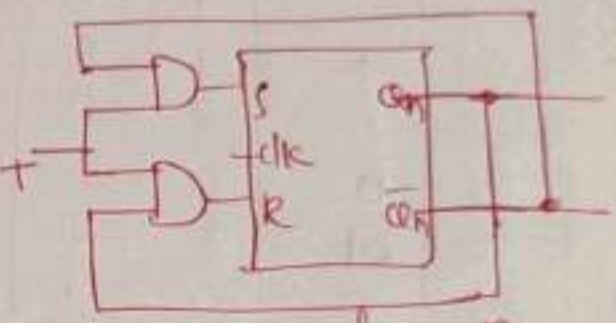
Q_n	T	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	1	0
1	0	1	X	0
1	1	0	0	1

Q_n \ T	$\bar{T}$	T
$\bar{Q}_n$		1
Q_n	X	

$S = T\bar{Q}_n$

Q_n \ T	$\bar{T}$	T
$\bar{Q}_n$	X	
Q_n		1

$R = TQ_n$



T FF diagram

④ JK FF to SR FF

required SR FF chart
Available JK excitation.

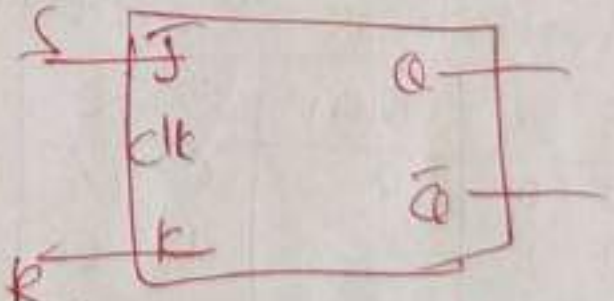
Q_n	S	R	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	0	0	X
0	1	0	1	1	X
0	1	1	X	X	X
1	0	0	1	X	0
1	0	1	0	X	1
1	1	0	1	X	0
1	1	1	X	X	X

Q_n \ SR	$\bar{S}\bar{R}$	$\bar{S}R$	SR	$S\bar{R}$
$\bar{Q}_n$			X	1
Q_n	X	X	X	X

$J = S$

Q_n \ JK	$\bar{J}\bar{K}$	$\bar{J}K$	JK	$J\bar{K}$
$\bar{Q}_n$	X	X	X	X
Q_n		1	X	

$K = R$

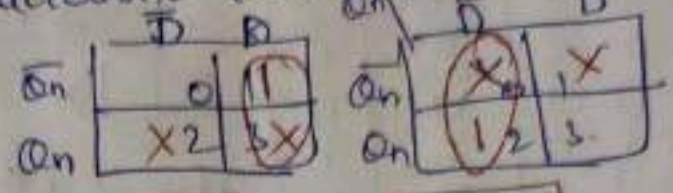


logic diagram for SR

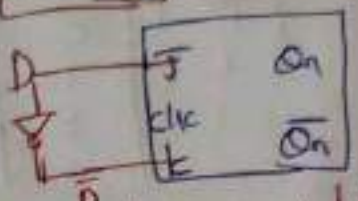
(5) Jk to D

Available Jk - Excitation table.
Required D - Characteristic table.

Qn	D	Qn+1	J	k
0	0	0	0	X
0	1	1	1	X
1	0	0	X	1
1	1	1	X	0



$J = D$ $\quad$ $K = \bar{D}$

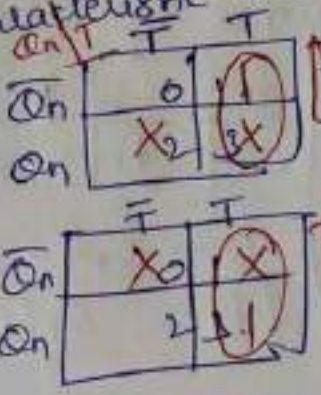


logic symbol for D

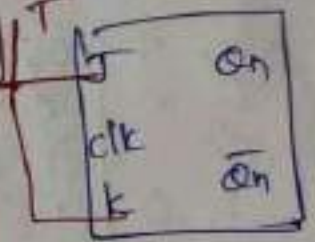
(6) Jk to T

Available Jk - Excitation table
Required T - Characteristic table

Qn	T	Qn+1	J	k
0	0	0	0	X
0	1	1	1	X
1	0	1	X	0
1	1	0	X	1



$J = T$ $\quad$ $K = T$

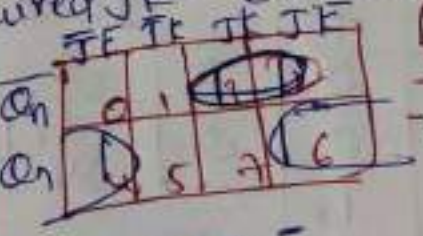


logic diagram for T from JK FF.

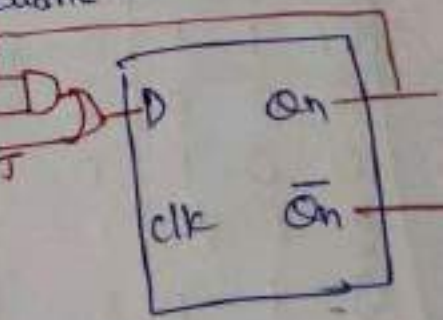
(7) D TO JK

Available D - Excitation table
Required Jk - Characteristic table.

Qn	J	k	Qn+1	D
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1(2)
0	1	1	1	13
1	0	0	1	14
1	0	1	0	0
1	1	0	1	16
1	1	1	0	0



$D = J + Q_n\bar{K}$



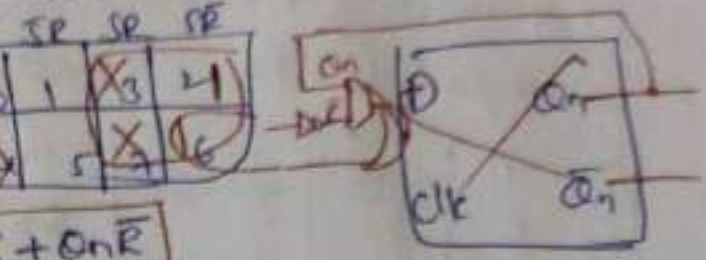
logic diagram for J from DFF.

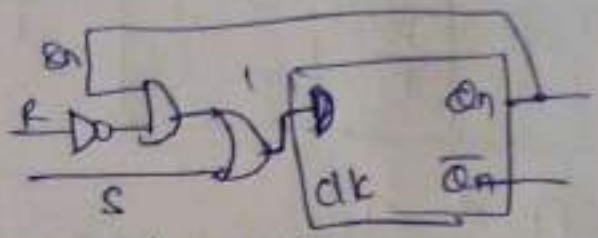
(8) D to SR — Available D – Excitation table.
Required SR – Characteristic table.

Q_n	S	R	Q_{n+1}	D
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	X	X
1	0	0	1	1
1	0	1	0	0
1	1	0	1	1
1	1	1	X	X

	$\bar{S}\bar{R}$	$\bar{S}R$	SR	$S\bar{R}$
$\bar{Q}_n$	0	1	X 3	4
Q_n	1 2	5	X 7	6

$D = S + Q_n\bar{R}$





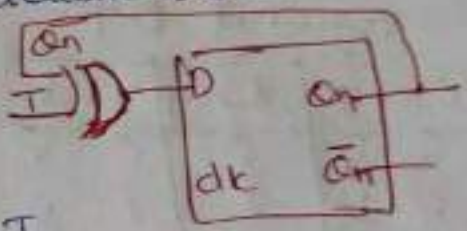
(9) D to T — Available D – Excitation table
Required T – Characteristic table.

Q_n	T	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

	$\bar{T}$	T
$\bar{Q}_n$	0	1 2
Q_n	1	3

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

$D = Q_n \oplus T$

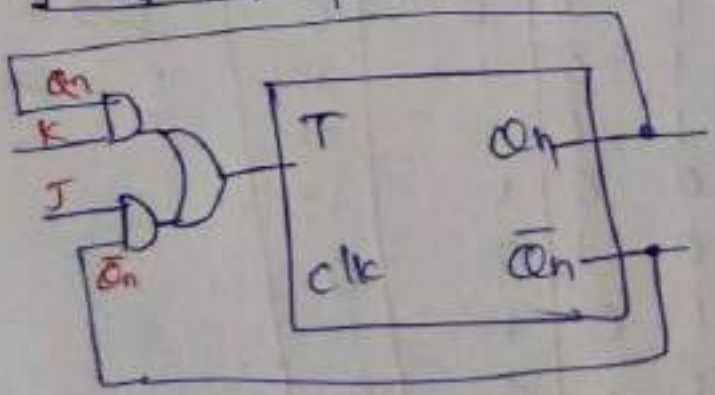


(10) T to JK — Available T – Excitation table.
Required JK – Characteristic table.

Q_n	J	K	Q_{n+1}	T
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	0	1

	$\bar{J}\bar{K}$	$\bar{J}K$	JK	$J\bar{K}$
$\bar{Q}_n$	0	1	3	2
Q_n	4	5	7	6

$T = Q_nK + \bar{Q}_nJ$



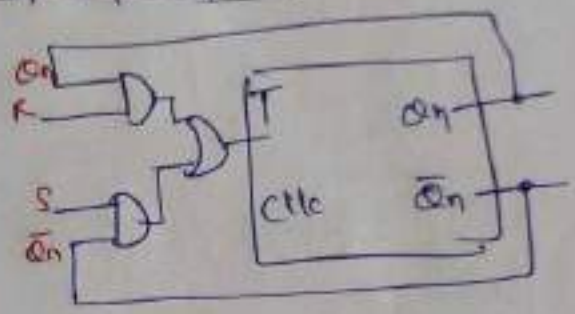
(11) T to SR

Available T – Excitation table.
Required SR – characteristic table.

Q_n	S	R	Q_{n+1}	T
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	X	X
1	0	0	1	0
1	0	1	0	1
1	1	0	1	0
1	1	1	X	X

	$\bar{S}\bar{R}$	$\bar{S}R$	$S\bar{R}$	SR
$\bar{Q}_n$	0	1	X_2	D_2
Q_n	4	t_5	X_7	6

$T = Q_n R + \bar{Q}_n S$



(12) T to D

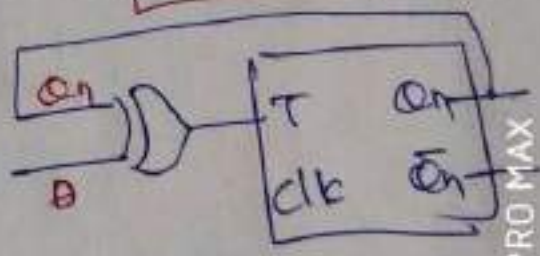
Available T – Excitation table
Required D – characteristic table.

Q_n	D	Q_{n+1}	T
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

	$\bar{D}$	D
$\bar{Q}_n$	0	1
Q_n	2	3

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

$T = Q_n \oplus D$



Flip-Flop Timing

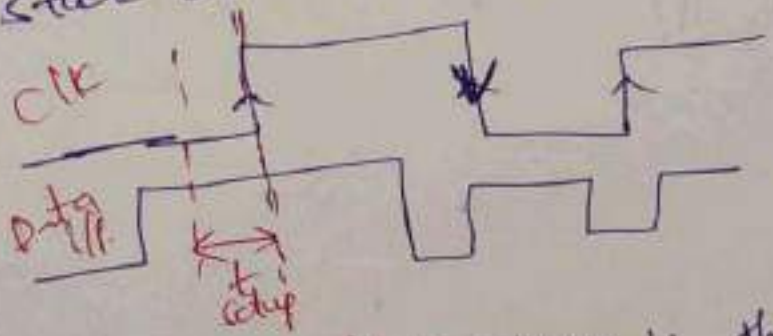
In designing digital systems with Flip-flops we have to consider 3 important parameters.

1) Setup time 2) Hold time 3) propagation Delay

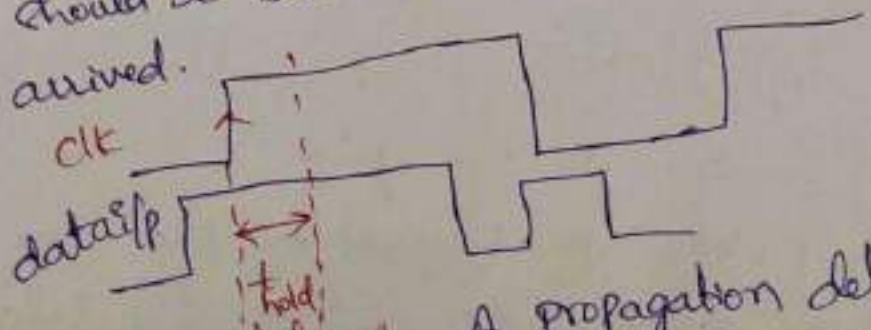
① **Setup time (ts)**: Setup time is the minimum time required to maintain a constant voltage level for data after excitation i/ps of the flip-flop device.

(or)

setup time is the minimum time for which data should be stable at the i/p before the active edge of clock arrive.



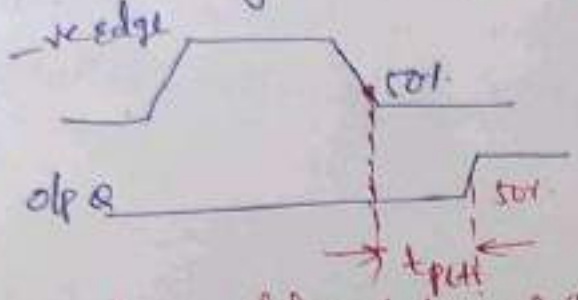
② **Hold time (th)** Time is the minimum time for which the data should be stable at the i/p after the active edge of clock has arrived.



③ **propagation delay**: A propagation delay is the time required to change the o/p after application of the i/p. Several categories of propagation delay are important in the operation of flip-flop

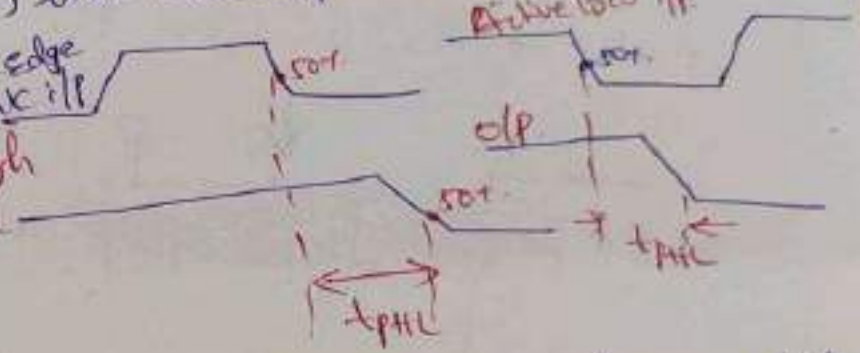
Propagation delay (t_{pLH}): It is measured from the triggering edge of the clock pulse (or) the present i/p to the Low-to-High transition of the o/p.

Response changing from low to high is called t_{pLH}



Propagation delay (t_{pHL}): It is measured from the triggering edge of the clk pulse (or) the clear i/p to the High-to-low transition of the o/p

Response changing from high to low is called t_{pHL}



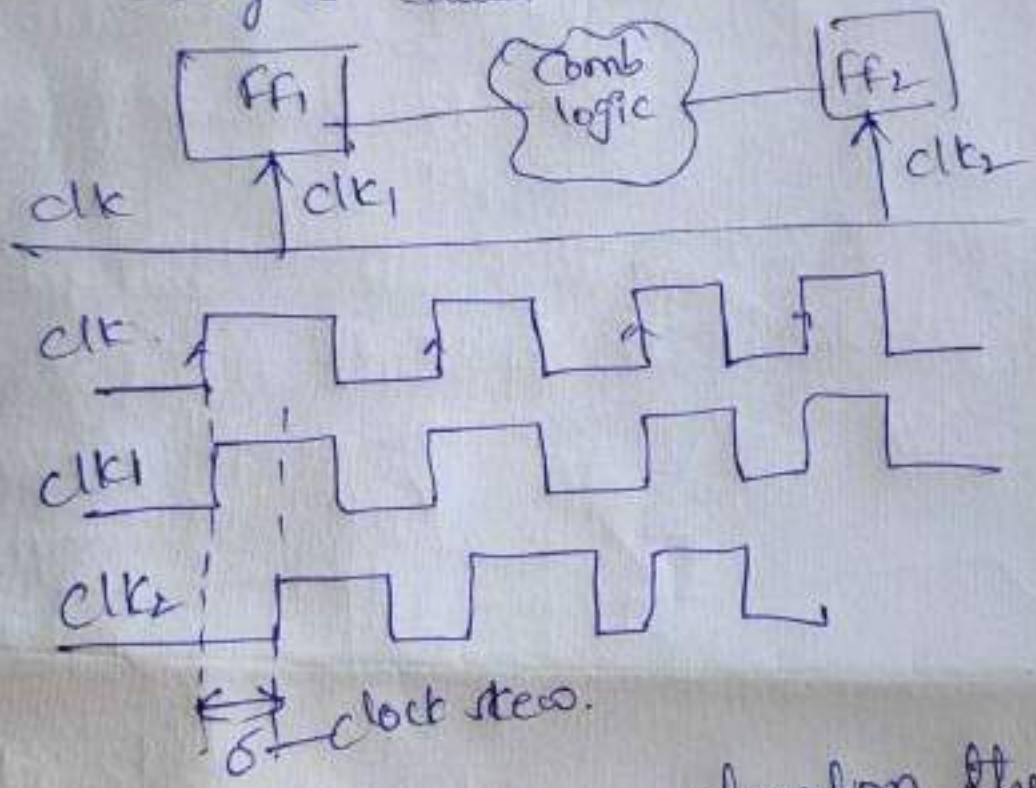
Clock skew

→ One of the most common timing problems in synchronous ckt is clock skew.

→ In many digital ckts, the o/p of the one ff is connected either directly (or) through logic gates to the i/p of another ff. and both flip-flops are triggered by the same clk signal.

→ The propagation delay of a ff and (or) the delays of the intervening gates make it difficult to predict precisely when the changing state of one ff will be experienced at another.

The clock signal which is applied simultaneously to all the ff in a synchronous system may undergo varying degrees of delay caused by wiring b/w components and arrive at the clk i/p of different ffs at different times. This delay is called clock skew.



clk&dat are in same direction then it is called +ve clock skew.

clk & data are in opposite direction then it is called -ve clock skew.

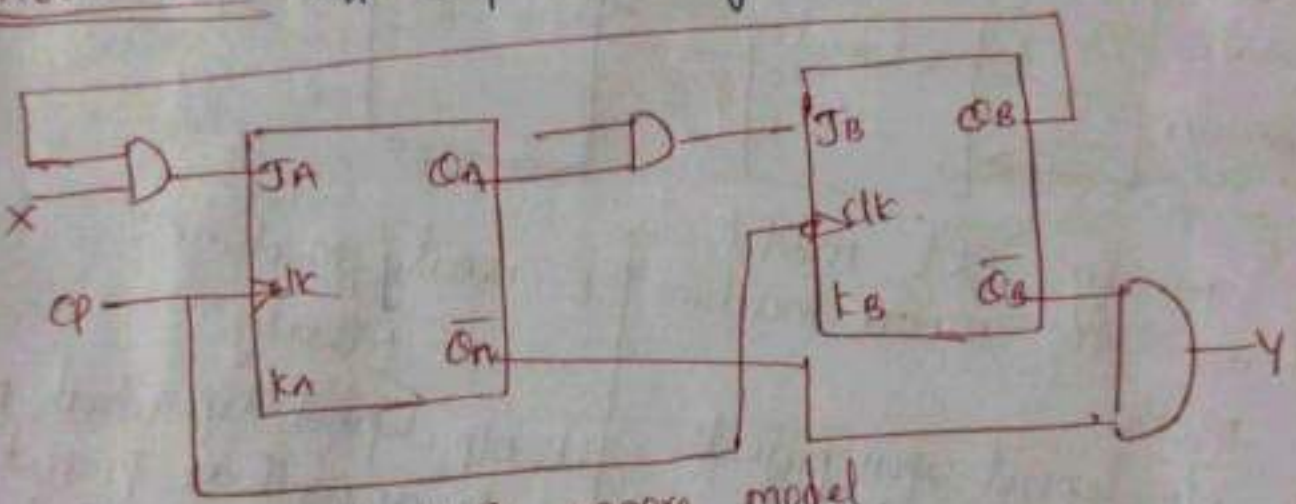
Sequential ckts Design and Analysis

→ clocked ffs are used as memory elements, which change their individual states in synchronism with the periodic clock signal.

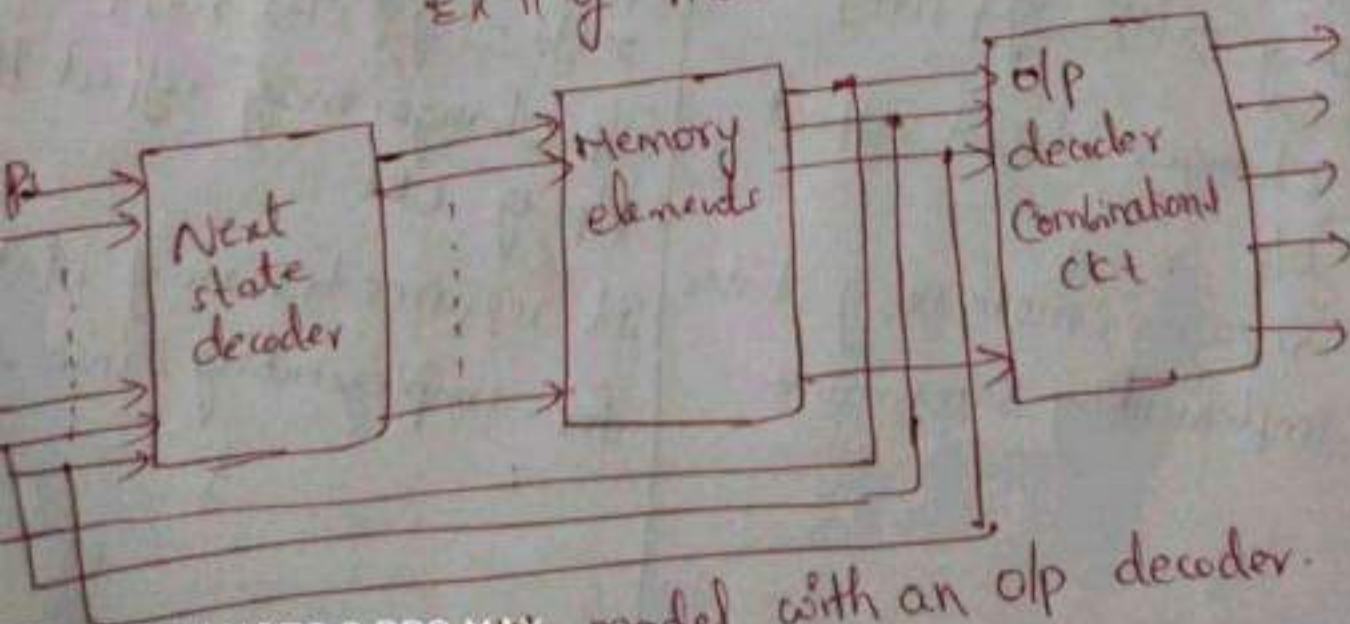
→ change in states of ffs and change in state of the entire ckt occurs at the transition of the clock signal.

→ synch (or) clocked sequential ckts are represented by two models

moore ckt o/p depends only on the present state of the ffs



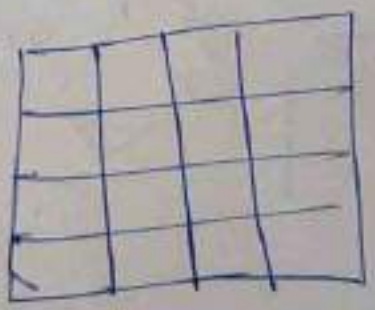
Ex 11 of moore model.

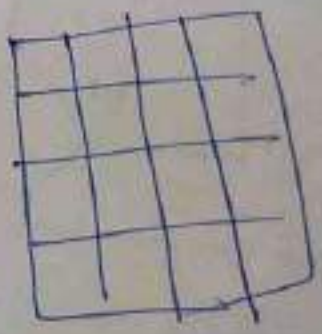


moore ckt model with an o/p decoder.

Design 4bit Synchronous Counter using JK FF.

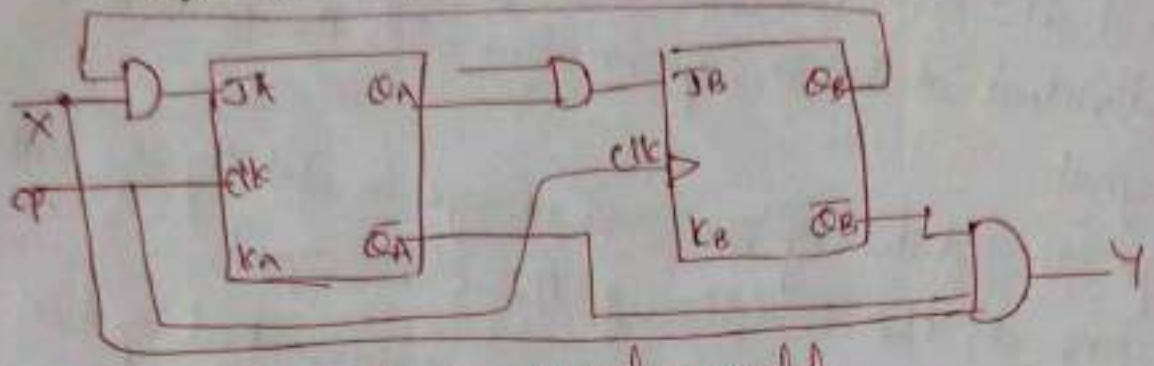
Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	0	1	1	1	1	0	0	X	0	0	X	X	0	X	1
1	1	0	0	1	1	0	1	X	0	0	X	0	X	1	X
1	1	0	1	1	1	1	0	X	0	0	X	1	X	X	1
1	1	1	0	1	1	1	1	X	0	0	X	X	0	1	X
1	1	1	1	0	0	0	0	X	1	1	X	X	1	X	1



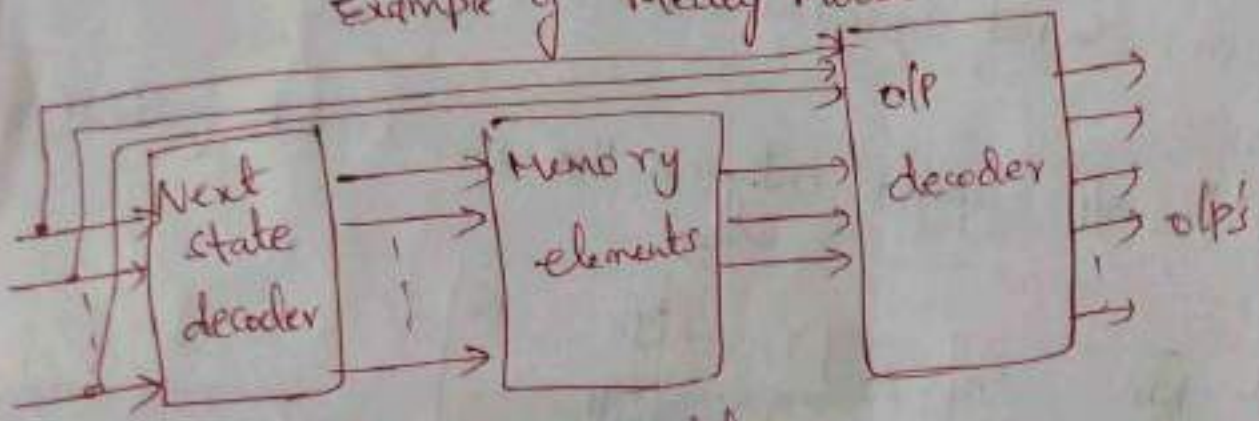


Mealy ckt o/p depends on both the present state & the ip's of the FFs & on the i/ps

→ Sequential ckts are also called as **Finite state Machine**



Example of Mealy model



Mealy ckt model.

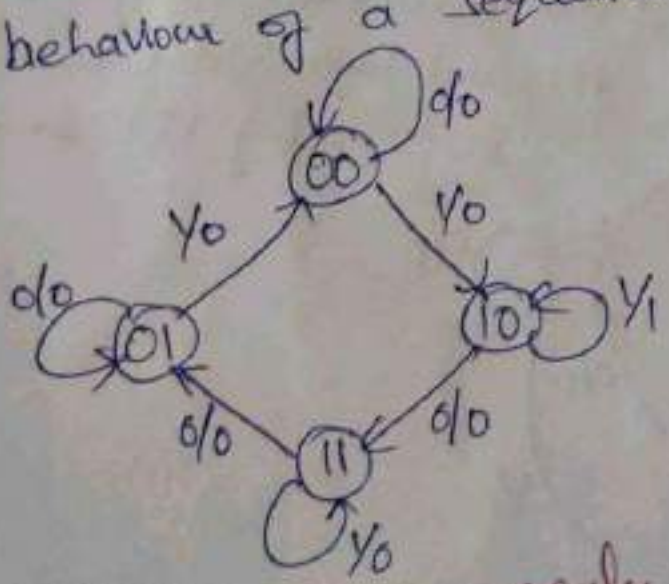
Difference b/w moore machine & mealy machine

Moore	Mealy
1) Its o/p depend upon present state only $z(t) = g\{s(t)\}$	Its o/p depends upon both present state as well as present i/p. $z(t) = g\{s(t), x(t)\}$
2) I/p changes doesn't affect the o/p	I/p changes may affect the o/p of ckt
3) It requires more no of states for implementing same func	It requires less no of states for implementing same func

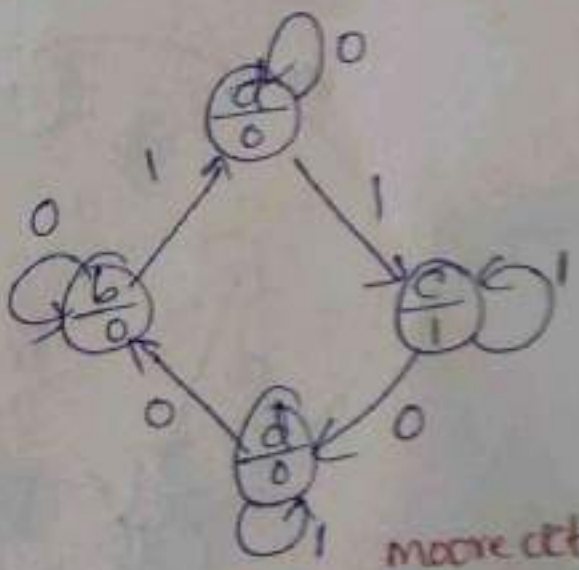
Design procedure for clocked sequential ckt.

1) A state diagram (or) timing diagram is given which describes the behaviour of the ckt that is to be designed.
2) obtain the state table.
3) The no. of states can be reduced by state reduction method.
4) Do state assignment (If required)
5) Determine the no. of flip-flops required and assign letter symbol.
6) Decide the type of flip-flop to be used.
7) Derive the ckt excitation table from state table obtain the expression for ckt o/p and FF o/p.
8) Implement the ckt.

State diagram. It is a pictorial representation of a behaviour of a sequential ckt



mealy ckt.



moore ckt

State table

translate the information contained in the state diagram into a tabular form.

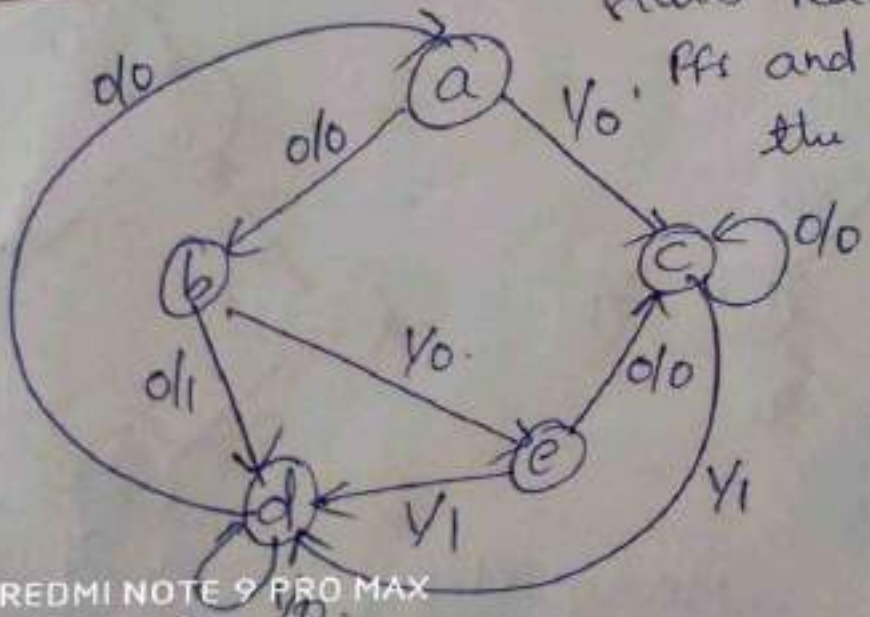
present state	Next state		o/p	
	x=0	x=1	x=0	x=1
AB	AB	AB	Y	Y
a	a, 0	c, 0	0	0
b	b, 0	a, 0	0	0
c	d, 0	c, 1	0	1
d	b, 0	d, 0	0	0

present state	Next state		o/p
	x=0	x=1	Y
a	a	c	0
b	b	a	0
c	d	c	1
d	b	d	0

moore ckt table

State reduction

The reduction in redundant states reduce the no. of required FFs and logic gates, reducing the cost of the final ckt.

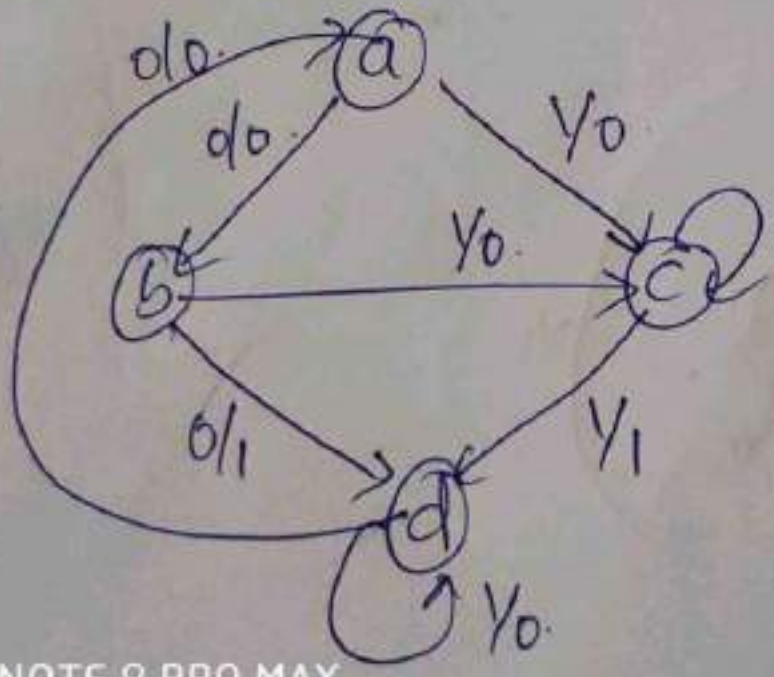


State table for given state diagram

Present state	Next state		o/p	
	x=0,	x=1	x=0,	x=1
a	b	c	0	0
b	d	e	1	0
c	c	d	0	1
d	a	d	0	0
e	c	d	0	1

Reduced state table.

Present state	Next state		o/p	
	x=0	x=1	x=0	x=1
a	b	c	0	0
b	d	c	1	0
c	c	d	0	1
d	a	d	0	0



Reduced state diagram.

State Assignment

=> During design process states are often denoted symbolically

=> Some state assignments (choosing binary representation) result in simple ckts than other state assignments (or)

The process of assigning binary values to the states of the sequential machine is known as state assignment.

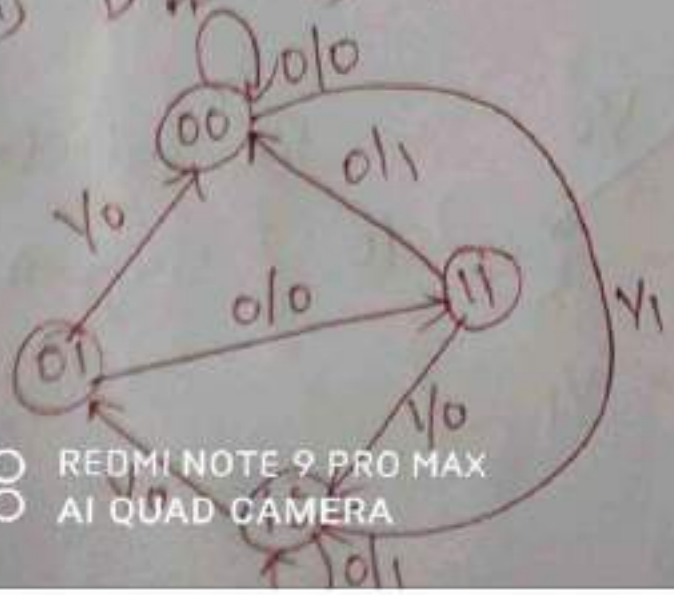
Rules for state assignment

Rule1 States having the same NEXT STATE for a given I/P condition should have assignments which can be grouped into logically adjacent cells in K-map

Rule2 states that are the next states of a single state should have assignments which can be grouped into logically adjacent cells in a K-map

A sequential ckt has one I/P & one o/p. The state diagram is shown in fig design the sequential ckt with

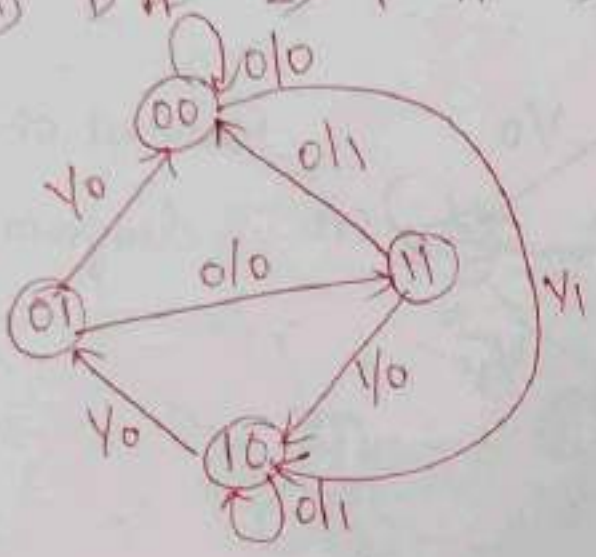
a) D FF 2) T FF 3) RS & 4) JK FF



Rule1 states having the same NEXT STATE for a given I/P condition should have assignments which can be grouped into logically adjacent cells in K-map

Rule2 states that are the next states of a single state should have assignments which can be grouped into logically adjacent cells in a K-map

A sequential ckt has one I/p & one o/p. The state diagram is shown in fig design the sequential ckt with
a) D FF 2) T FF 3) RS & 4) JK FF



state table

present state		Next state		o/p	
		x=0	x=1	x=0,	x=1
A	B	AB	AB	Y	Y
0	0	00	10	0	1
0	1	11	00	0	0
1	0	10	01	1	0
1	1	00	10	1	0

There are no equivalent states ∴ no reduction

→ State table shows that ckt goes to 4 states

Required. 2 FF = no. of states = $2^m = 2^2 = 4$, m-no of FF)

-TWO FFs are required A & B

1) Design using D FFs

given state table

excitation table of D FF

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Present state A	B	X	Next state A^+	B^+	flip flop i/ps D_A	D_B	o/p Y
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

Q_n + Q_{n+1} = D_n

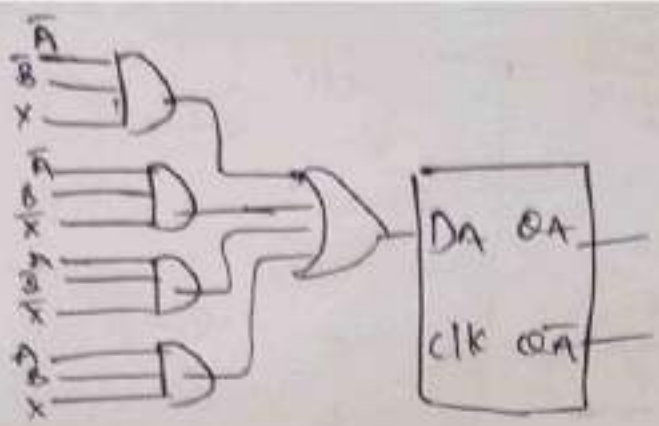
-Apply the Excitation table of D FF to A, At columns it will give the D_A o/p then apply to B, B^+ it gives the D_B value.

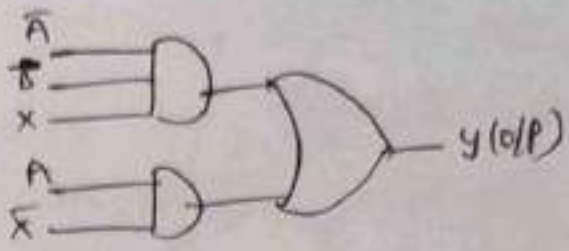
$D_A = \bar{A}B\bar{X} + A\bar{B}\bar{X} + AB\bar{X} + ABX$

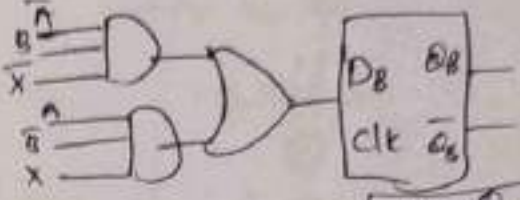
$D_B = \bar{A}B\bar{X} + A\bar{B}X$

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

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







Sequential ckt for given State diagram by using D ff

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

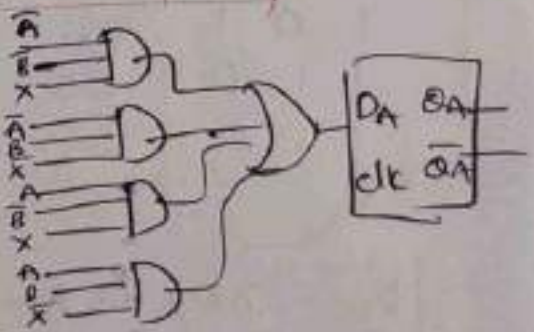
By using T ff

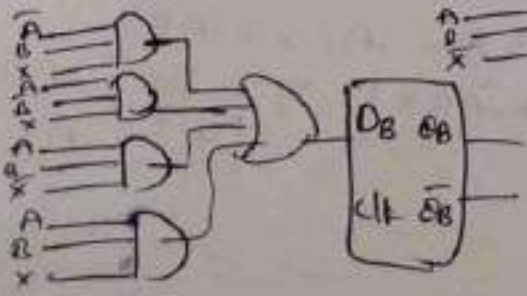
present state (i/p's)			Next state		Flip-flops i/ps		o/p
A	B	X	A^+	B^+	T_A	T_B	Y_1
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1
0	1	0	1	1	1	0	0
0	1	1	0	0	0	1	0
1	0	0	1	0	0	0	1
1	0	1	0	1	1	1	0
1	1	0	0	0	1	1	1
1	1	1	1	0	0	1	0

$T_A = \bar{A}\bar{B}X + \bar{A}B\bar{X} + A\bar{B}X + AB\bar{X}$

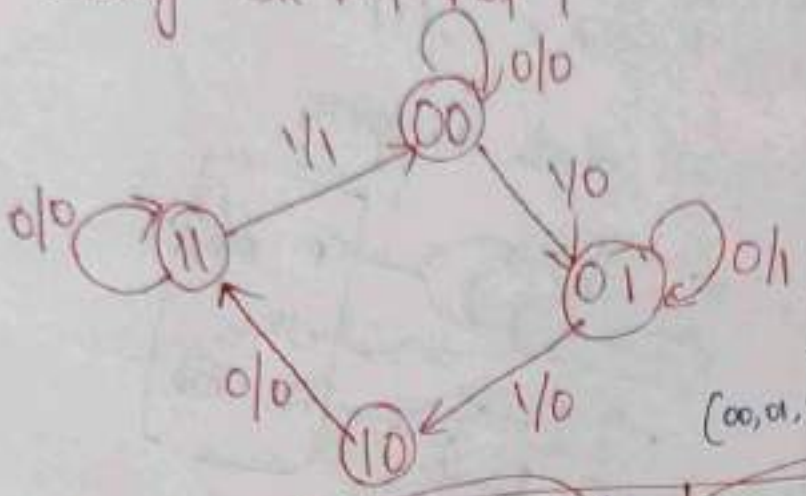
$T_B = \bar{A}BX + A\bar{B}X + AB\bar{X} + ABX$

$o/p = \bar{A}\bar{B}X + A\bar{B}\bar{X} + AB\bar{X}$





Design sequential ckt for given state diagram by using SR flip-flop ?



Excitation table of SR

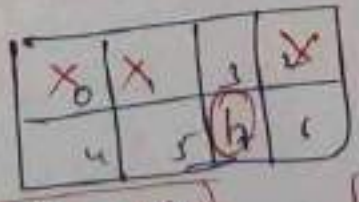
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

No. of flip-flops depend upon the no. of states
(00, 01, 10, 11) 4 states $= 2^M = 2^2$
$M = 2$ [2 flip-flops are required]

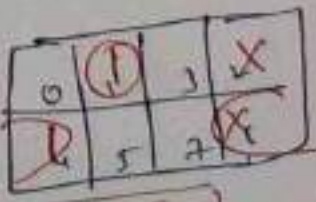
present state A	B	I/p X	Next state Q_{n+1} A	B	S_A	R_A	S_B	R_B	O/p(y)
0	0	0	0	0	0	X(6)	0	X(0)	0
0	0	1	0	1	0	X(1)	1(9)	0	0
0	1	0	0	1	0	X(2)	X(6)	0	1(2)
0	1	1	1	0	1(3)	0	0	1(3)	0
1	0	0	1	1	X(4)	0	1(4)	0	0
1	0	1	1	0	X(5)	0	0	X(5)	1(5)
1	1	0	1	1	X(6)	0	X(6)	0	0
1	1	1	0	0	0	1(7)	0	1(7)	1(7)

Excitation table of SR flip-flop is applied to A & A+ column then we will get S_A R_A values. Similarly apply to B & B+ column then you will get S_B, R_B values.

$S_A = \bar{A}BX$



$R_A = ABX$



$S_B = A\bar{X} + \bar{A}BX$

$O/P = AX + \bar{A}BX$

$R_B = BX + AX$

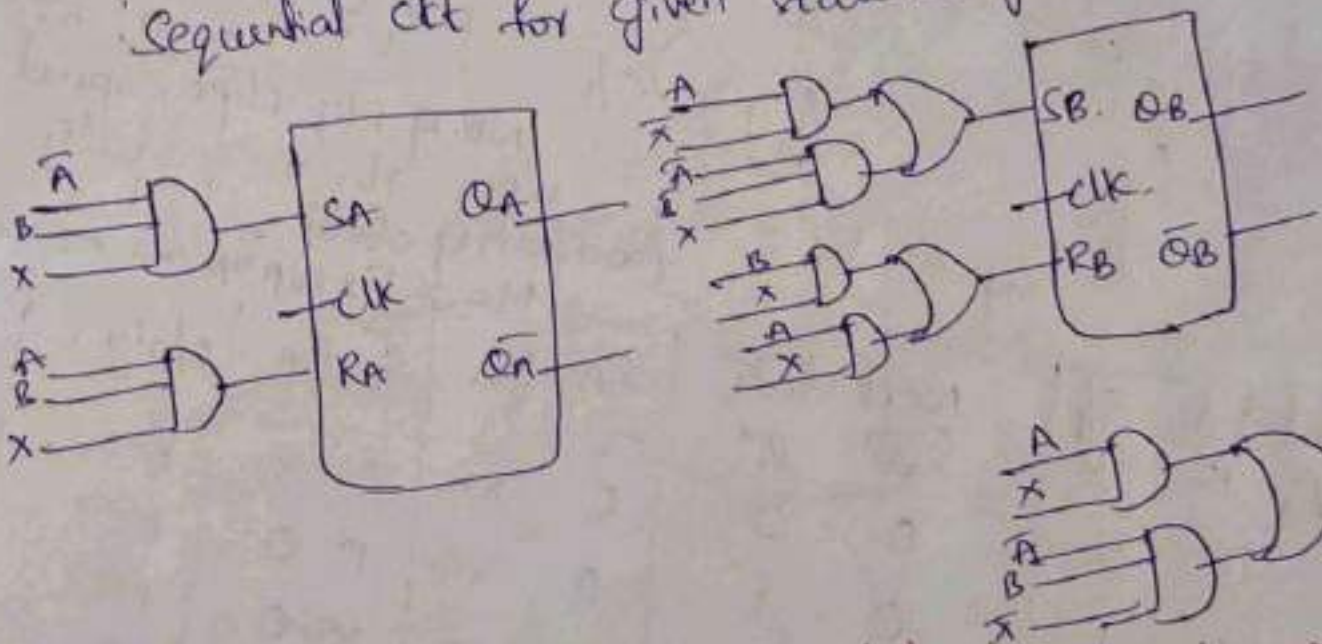
$S_A = \bar{A}BX$, $S_B = A\bar{X} + \bar{A}\bar{B}X$

$R_A = ABX$

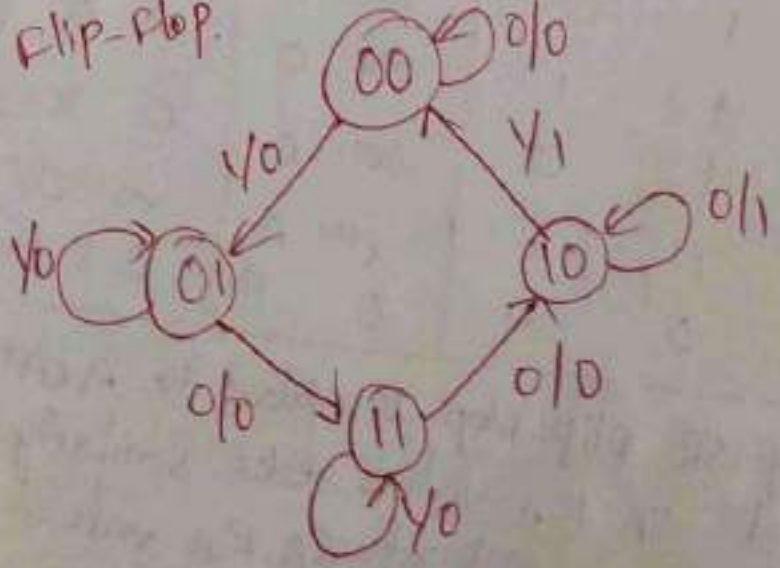
$R_B = BX \oplus AX$

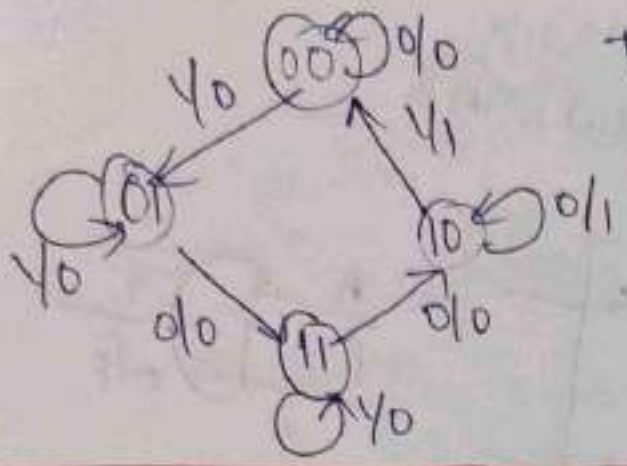
$R_B = BX + AX$, $o/p = AX + \bar{A}B\bar{X}$

Sequential ckt for given state diagram.



Design sequential ckt for given state diagram by using JK Flip-flop.





Total state (00,01,10,11) 4. we required

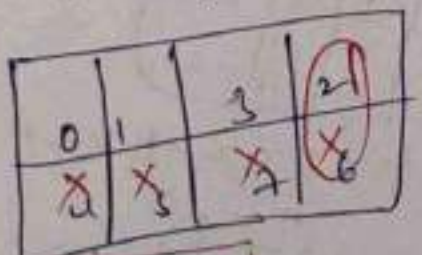
$4 = 2^{n} = 2^{n+1} \Rightarrow n = 2$ (no of flipflops = 2 FF are required)

Excitation table of J,K

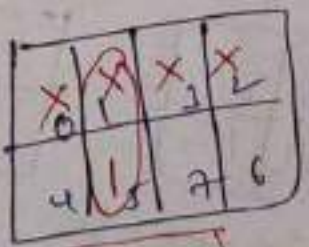
Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Present state A	B	i/p X	Next state A^+	B^+	J_A	K_A	J_B	K_B	o/p (y)
0	0	0	0	0	0	X_0	0	X_0	0
0	0	1	0	1	0	X^1	1	X^1	0
0	1	0	0	1	1^2	X^2	X^2	0	0
0	1	1	0	1	0	X^3	X^3	0	0
1	0	0	1	0	X^4	0	0	X^4	1^4
1	0	1	1	0	X^5	1^5	0	X^5	1^5
1	1	0	0	0	X^6	0	X^6	1^6	0
1	1	1	1	1	X^7	0	X^7	0	0

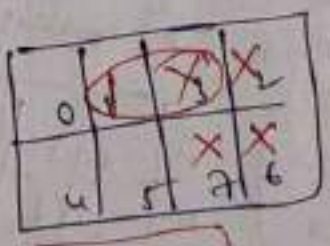
Apply excitation table of JK to A & A^+ colum. then we will get $J_A K_A$ value. and same table apply to B & B^+ colum then we will get $J_B K_B$ values.



$J_A = B\bar{X}$

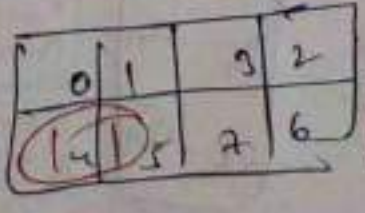


$K_A = \bar{B}X$



$J_B = \bar{A}X$

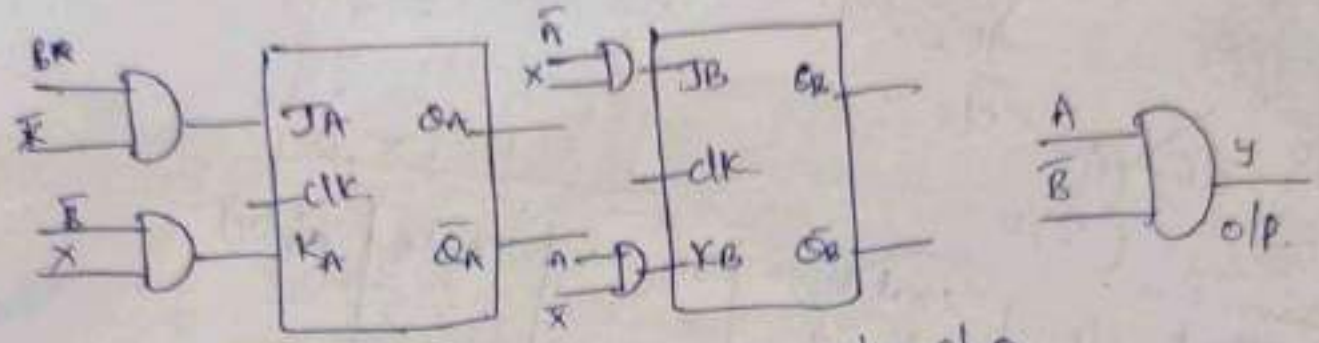
$K_B = A\bar{X}$



o/p(y) = $A\bar{B}$

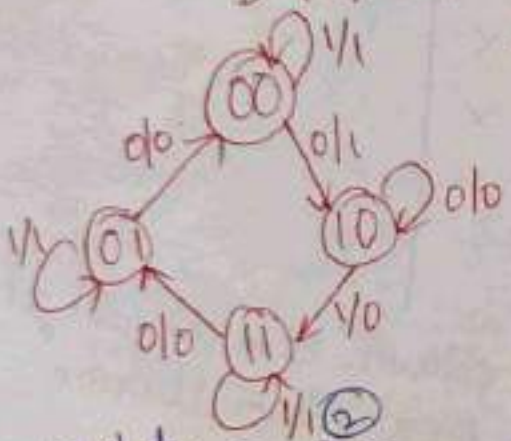
$J_A = B\bar{x}$ $J_B = \bar{A}x$

$K_A = \bar{B}x$ $K_B = Ax$, $o/p(y) = A\bar{B}$



Sequential ckt by using JK flip-flop

Design sequential ckt for given state diagram by using D flip-flop



present state A B	x	next state A^+ B^+	D_A	D_B	o/p (y)
0 0	0	1 0	1	0	1
0 0	1	0 0	0	0	1
0 1	0	0 0	0	0	0
0 1	1	0 1	0	1	1
1 0	0	1 0	1	0	0
1 0	1	1 1	1	1	0
1 1	0	0 1	0	1	0
1 1	1	1 1	1	1	1

Excitation of D FF

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

$D = Q_{n+1}$

$D_A = A^+$

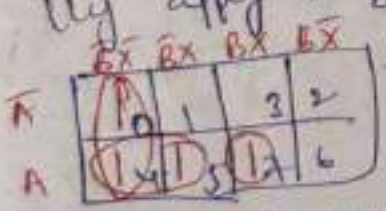
$D_B = B^+$

4 states = 2 =

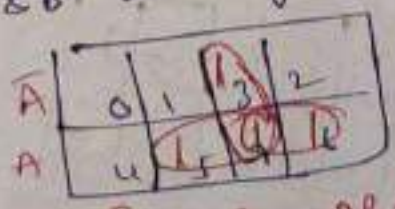
No. of FF = 2 FF required

Apply Excitation table of D FF to A & A^+ Colum you will get D_A value

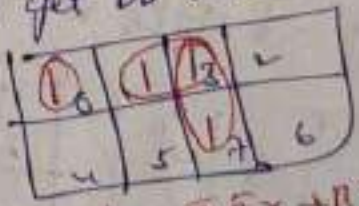
By apply to B & B^+ value you will get D_B value



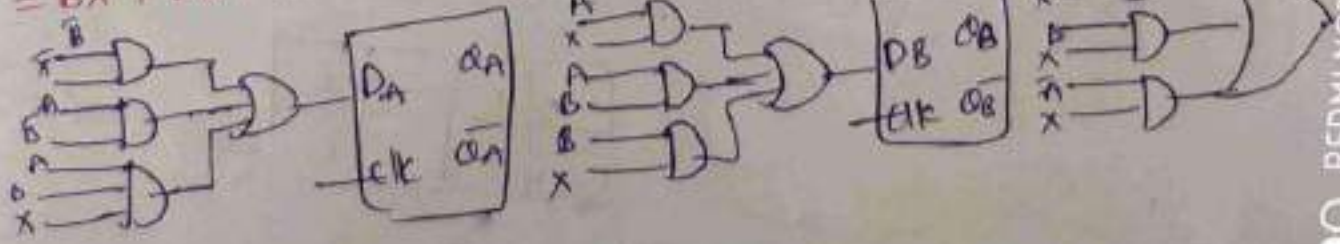
$D_A = \bar{B}\bar{x} + A\bar{B} + ABx$



$D_B = Ax + A\bar{B} + Bx$

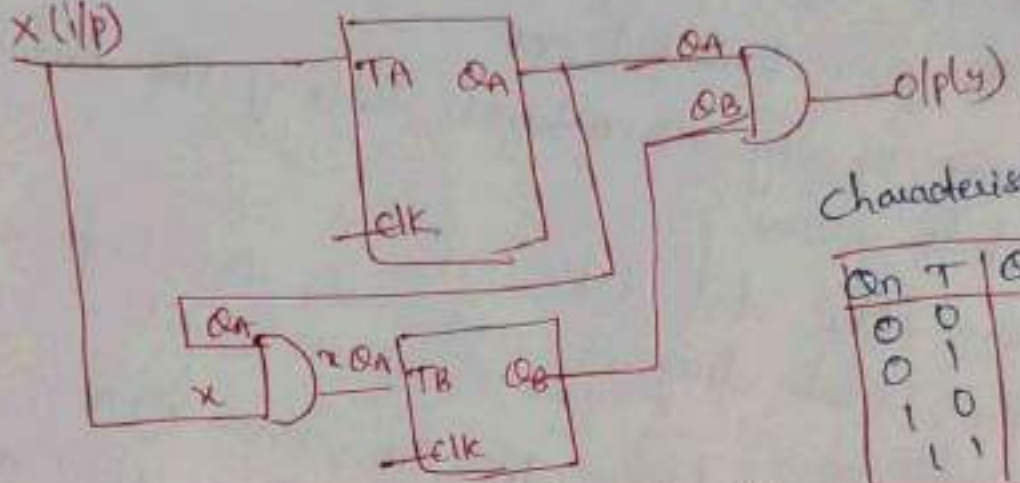


$o/p = \bar{A}\bar{B}x + B\bar{x} + A\bar{x}$



Analysis of sequential ckt

Construct the state table & state diagram for the mealy sequential ckt ? (7)



Characteristic of TFF

Qn	T	Qn+1	T_A
0	0	0(Qn)	
0	1	1($\bar{Q}_n$)	
1	0	1(Qn)	
1	1	0($\bar{Q}_n$)	

$T_A = X$, $T_B = xQ_A$, $o/p(y) = Q_A Q_B$

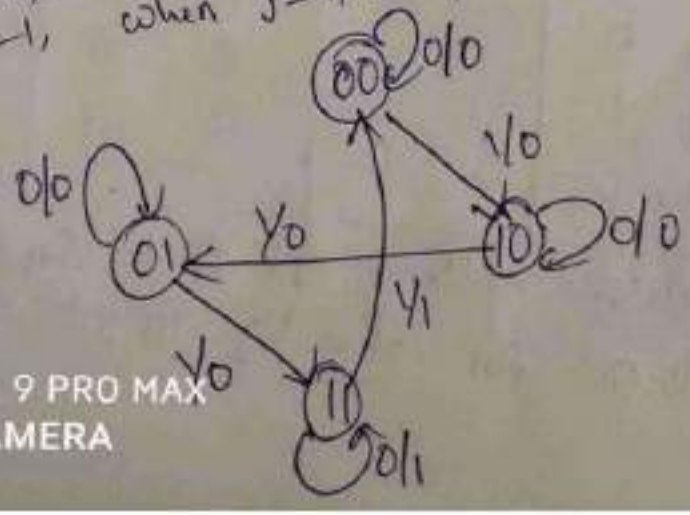
present state A	B	x	$x = T_A$	$(xQ_A) = T_B$	Next state Q_A^+	Q_B^+	o/p(y)
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0
0	1	0	0	0	0	1	0
0	1	1	1	0	1	1	0
1	0	0	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	1	0	0	1
1	1	1	1	1	0	1	1

Apply characteristic table of TFF to A & T_A column then we will get Q_A^+ value ily apply to B & T_B column you will get Q_B^+

From characteristic table of TFF

$T_A = 0$, when $J=0, K=0$, $Q_A^+ = Q_A$,

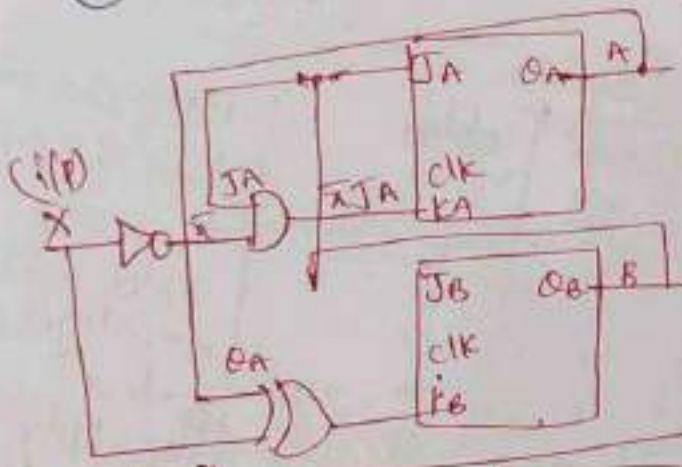
$T_A = 1$, when $J=1, K=1$, $Q_A^+ = \bar{Q}_A$



Analysis of Sequential Ckt

Steps

1. Determine the flip-flop i/p eqns and the o/p eqns from the given sequential ckt
2. ~~Derive~~ By using characteristic table of FFs find the Next state values.
3. Draw the state table
4. Draw state diagram. Construct the state diagram for given mode sequential ckt



$J_A = Q_B$, $K_A = \bar{x} J_A$

$J_B = \bar{x}$ $K_B = x \oplus Q_A$

$= \bar{x} Q_A + x \bar{Q}_A$

Characteristic table of JK

Q_n	J	K	Q_{n+1}
0	0	0	0 (Q_n)
0	0	1	0 (Q_n)
0	1	0	1
0	1	1	1 ($\bar{Q}_n$)
1	0	0	1 (Q_n)
1	0	1	0
1	1	0	1
1	1	1	0

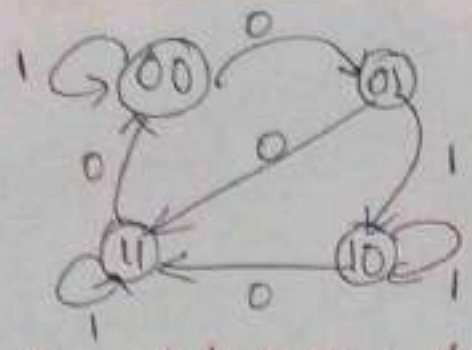
Q_n J K | Q_{n+1}

Q_A J_A K_A | Q_A^+

Q_B J_B K_B | Q_B^+

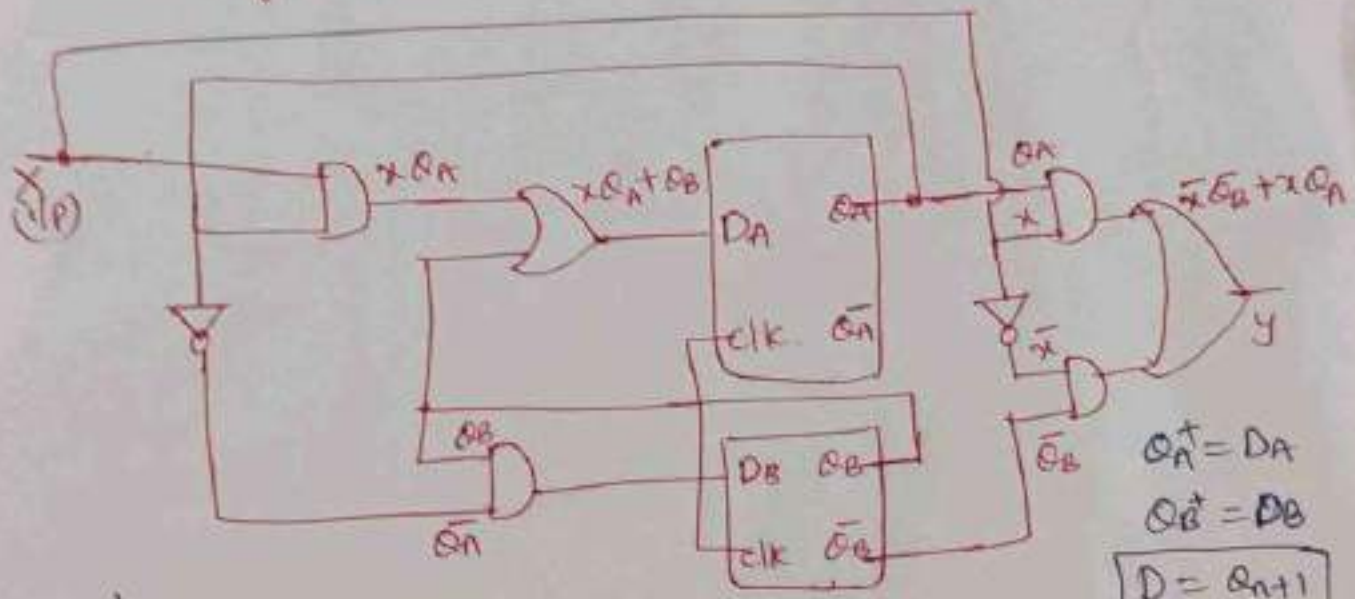
Present state Q_A	Q_B	i/p x	J_A	K_A	J_B	K_B	Next state Q_A^+	Q_B^+
0	0	0	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1
0	1	0	1	1	0	1	1	0
0	1	1	1	0	0	1	1	0
1	0	0	0	0	0	0	1	0
1	0	1	0	0	1	1	1	1
1	1	0	1	1	0	1	0	0
1	1	1	1	0	0	1	1	0

apply characteristic table of JK FF to Q_A & J_A column then we will get Q_A^+ value. By apply to Q_B & $J_B = Q_B^+$



state diagram.

Construct the state table & state diagram from given mealy sequential ckt.



$Q_A^+ = D_A$

$Q_B^+ = D_B$

$D = Q_{n+1}$

$Q_A^+ = D_A = xQ_A + Q_B$ $D_B =$

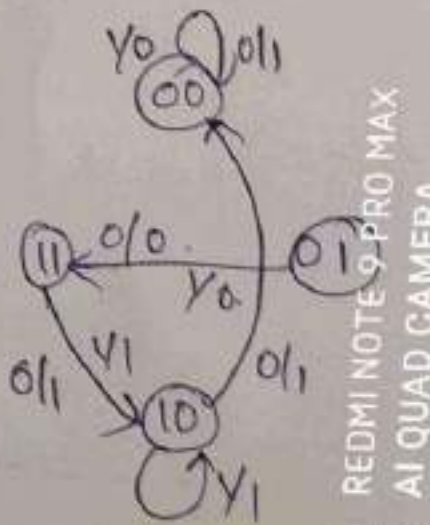
$Q_B^+ = D_B = \bar{x}\bar{Q}_A Q_B$, $y(O/P) = \bar{x}\bar{Q}_B + xQ_A$

P.S Q_A	P.S Q_B	I/P x	N.S $D_A = Q_A^+$	$D_B = Q_B^+$	$y(O/P)$
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	0	1

state table

$Q_A^+ = D_A = xQ_A + Q_B$
$= 0.0 + 0 = 0$.

$Q_B^+ = D_B = \bar{x}\bar{Q}_A Q_B = 1.0 = 0$



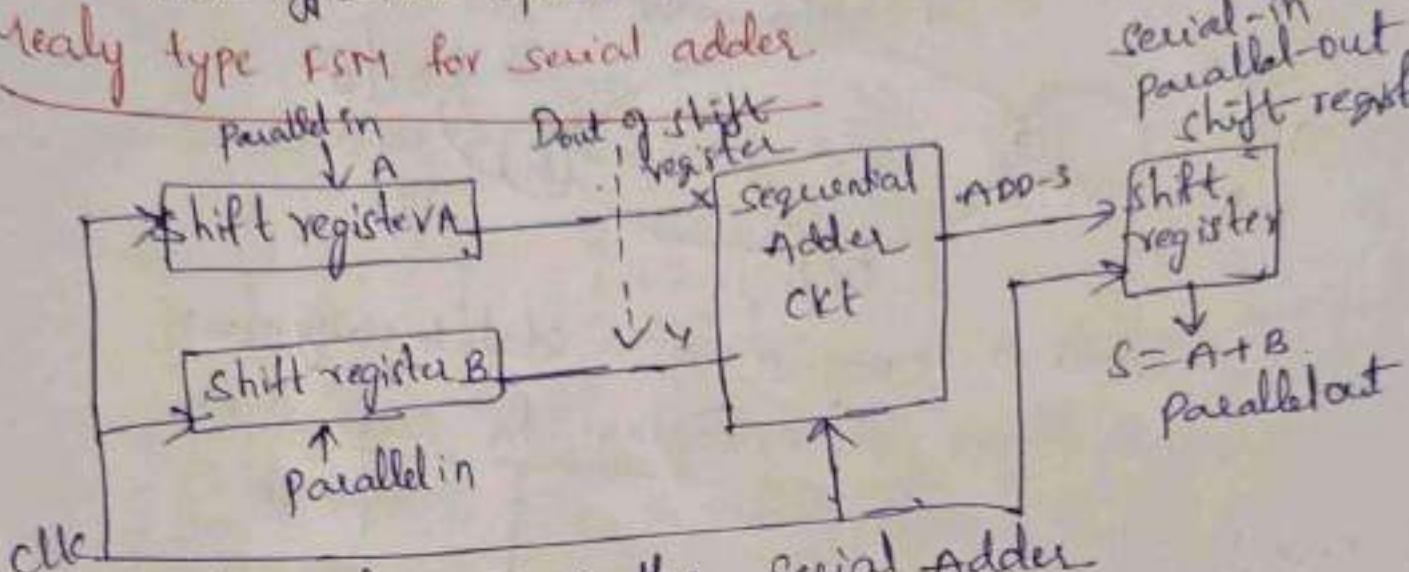
state diagram

Serial adder

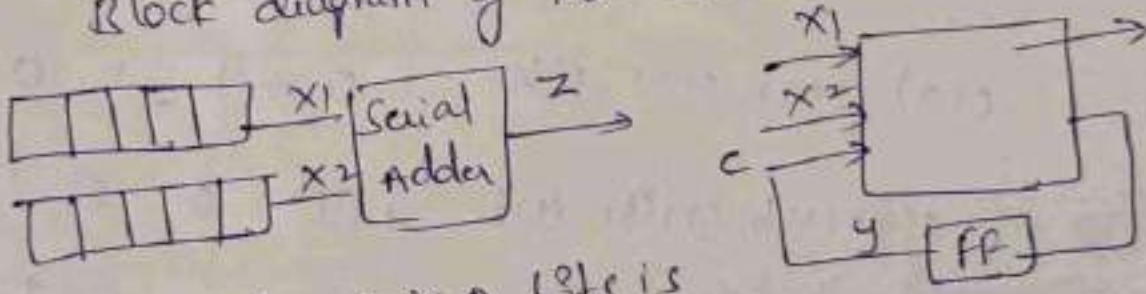
→ Serial adder is a ckt in which bits are added a pair at a time when speed is not of great important.

→ It's cost-effective option to use serial adder.

Mealy type FSM for serial adder



Block diagram of the Serial Adder



→ each clk cycle pair of bits is added by the Seq" ckt used as an Adder.

→ we have to consider stating carry of previous bit addition while adding the current pair of bits C=0 either 1

→ o/p value ADD-S depend on both the state (carry-in) & previous I/P X & Y

→ state a, carry-in is 0 we get

ADD-S=0 carry-in=0, for XY=00

ADD-S=1, carry-in =0, for XY=01(or)10

ADD-S=0, carry-in =1, for XY=11

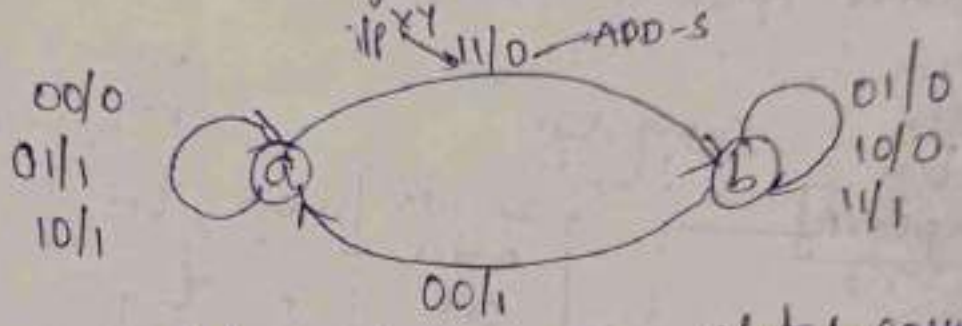
→ For XY=11, carry-in becomes 1 and we have to change state for other values of X & Y carry-in remains 0 and hence state doesn't change

→ In state b, Carry-in is 1, we get

ADDS = 1 Carry-in = 0, for XY = 00

ADDS = 0, Carry-in = 1 for XY = 01 (or) 10

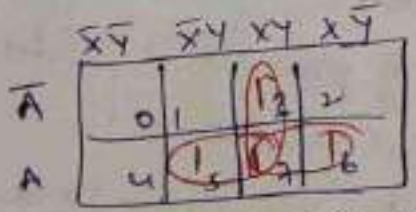
ADDS = 1, Carry in = 1, for XY = 11



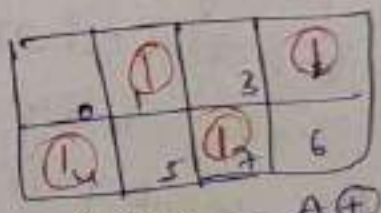
state a carry-in = 0. state b carry in = 1

Mealy type state diagram for Serial Adder

present state A	Next state XY = 00	01	10	11	O/p ADD-S 00	01	10	11
(0)(a)	0(a)	0(a)	0(a)	(1)(b)	0	1	1	0
(1) (b)	0(a)	(1)b	(1)(b)	(1)b	1	0	0	1



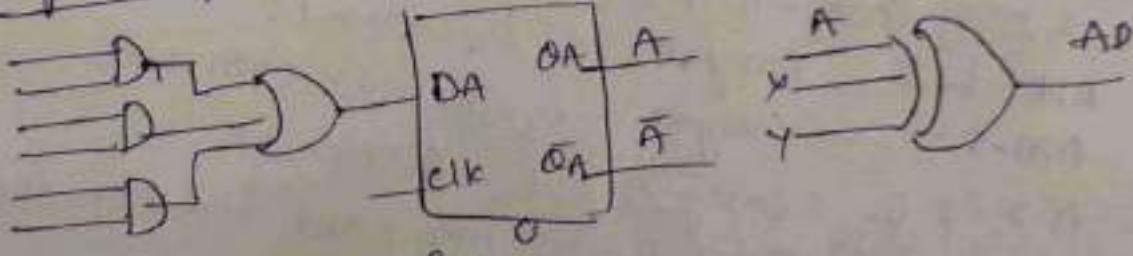
$A_t = XY + AX + AY$



ADD-S = A ⊕ X ⊕ Y.

A	B	X	A^+	B^+	d/p
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	1	1

Logic diagram.



Data Transmission in Shift Registers

→ Register is nothing but to store more than 1 bit we require registers.

→ FF is nothing but a binary cell capable of storing one bit information and can be connected together to perform counting operations the group of FF is called counter.

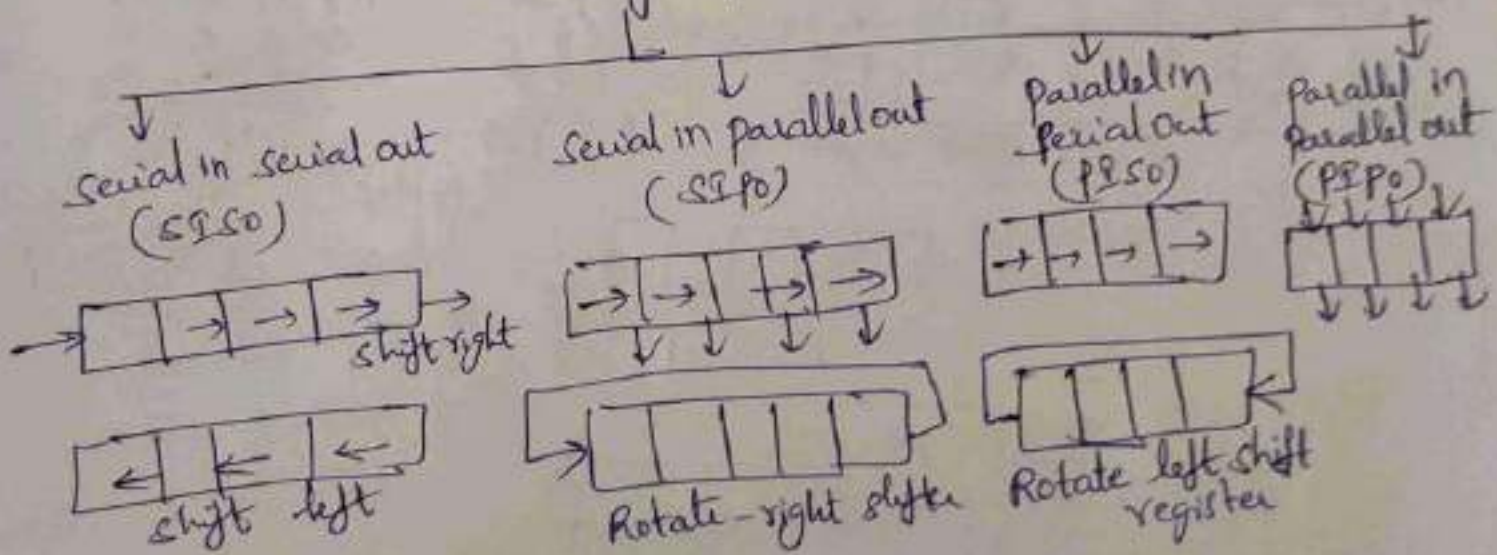
→ Group of FF can be used to store a word which is called register.

→ Increase the storage capacity interms of no. of bits we have to use a group of FF is called register.

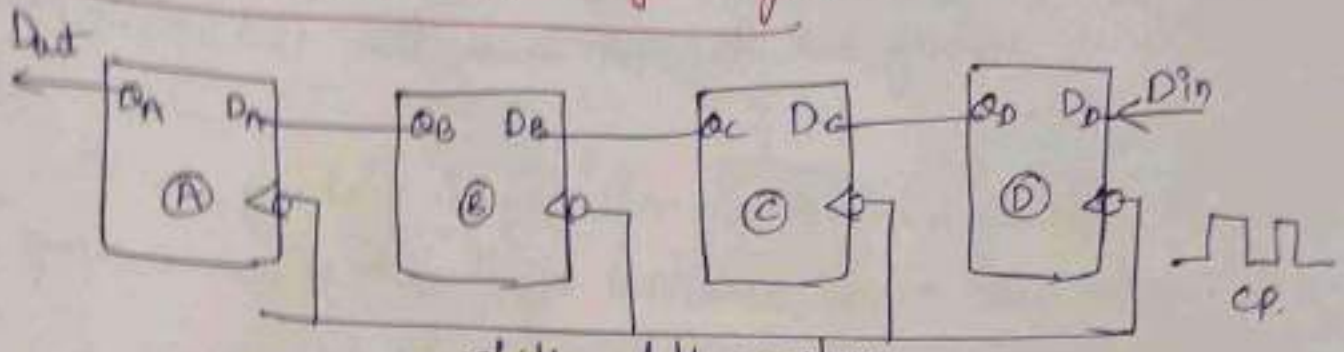
Register Classification.

→ Register may be classified based on the way in which data are entered and taken out from a register.

modes of operation



Serial-in serial-out shift register.



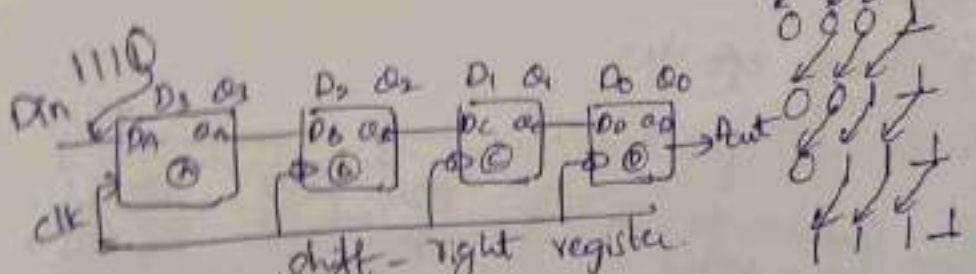
shift-left register.

entry of 4 bit binary number 1111 into the register beginning with left most bit initially register is cleared. ∴ we have

$Q_A Q_B Q_C Q_D = 0000$

a) when data 1111 is applied serially, ie left most 1 is applied is Din

$D_{in} = 1$, $Q_A Q_B Q_C Q_D = 0000$

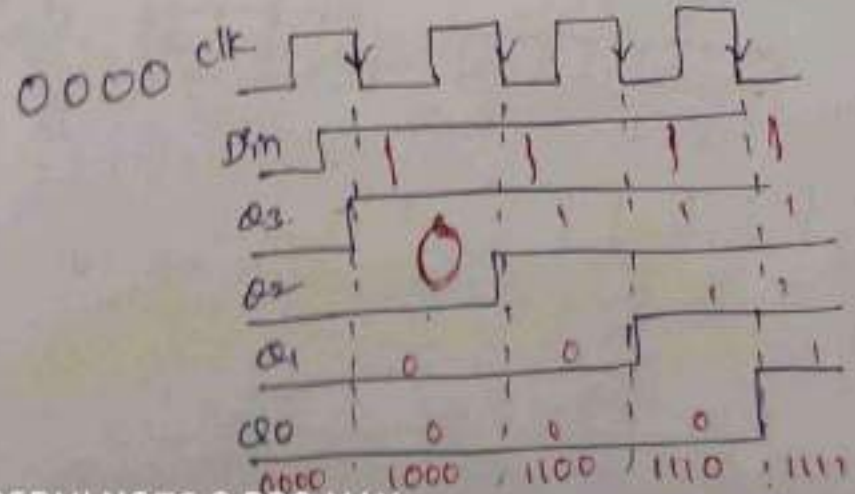


we need 4 clk pulses to transmit 4 bits ie 1111

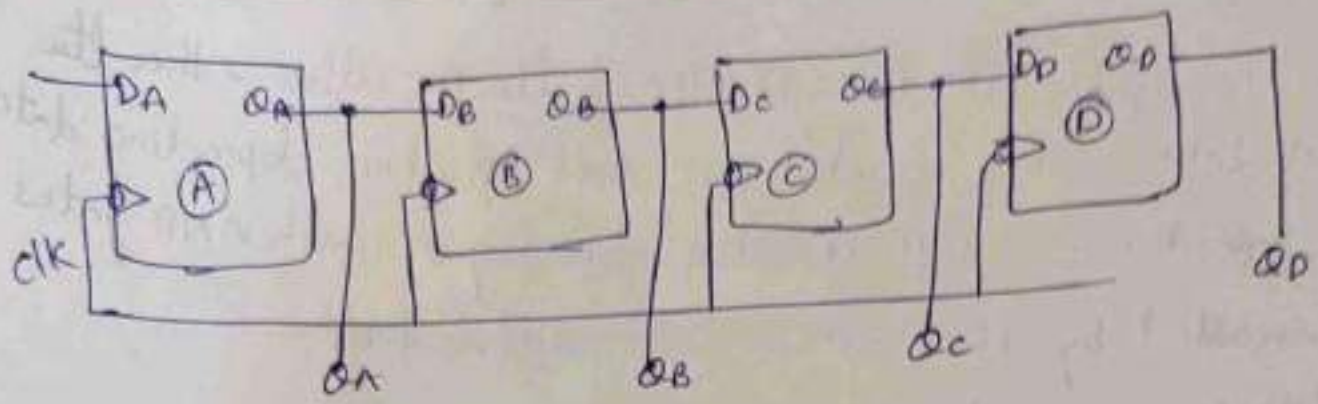
MSB LSB

Q_3 Q_2 Q_1 Q_0

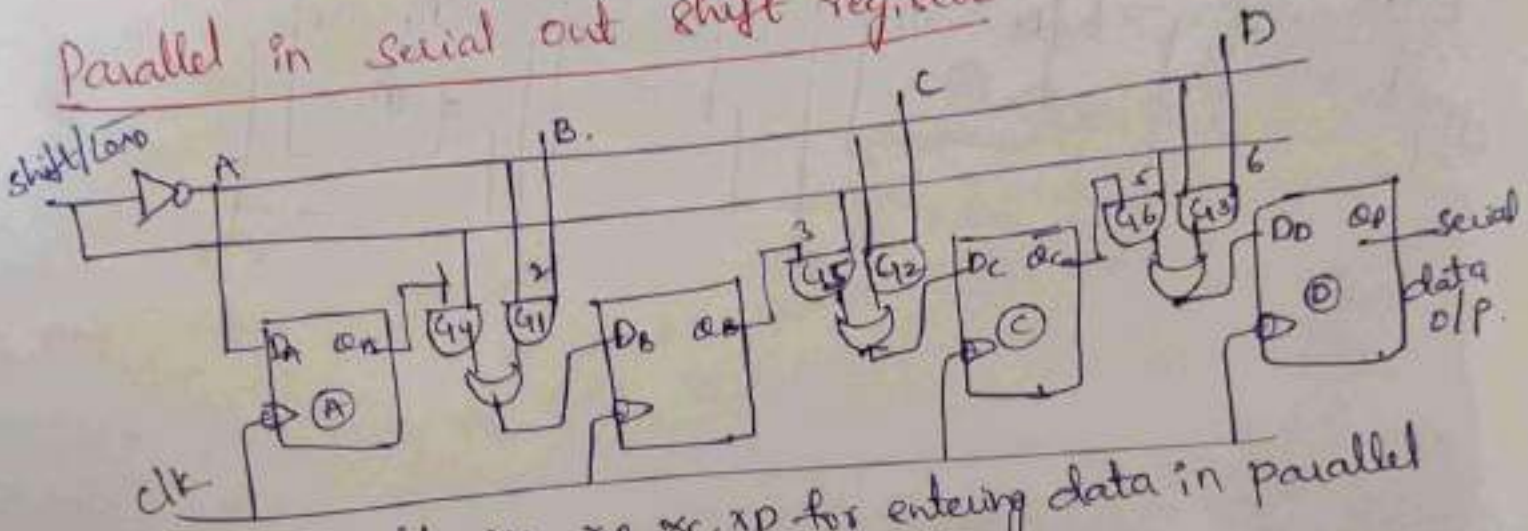
0 0 0 0
1 0 0 0
1 1 0 0
1 1 1 0
1 1 1 1



Serial in parallel out shift register.



Parallel in serial out shift register



→ There are i/p X_A, X_B, X_C, X_D for entering data in parallel into the register.

→ Shift/$\overline{load}$ is the control i/p which allows shift (or) loading data operation of the register.

→ when shift/$\overline{load}$ is low, gates G_1 G_2 G_3 are enabled allowing each i/p data bit to be applied to D i/p of respective FF.

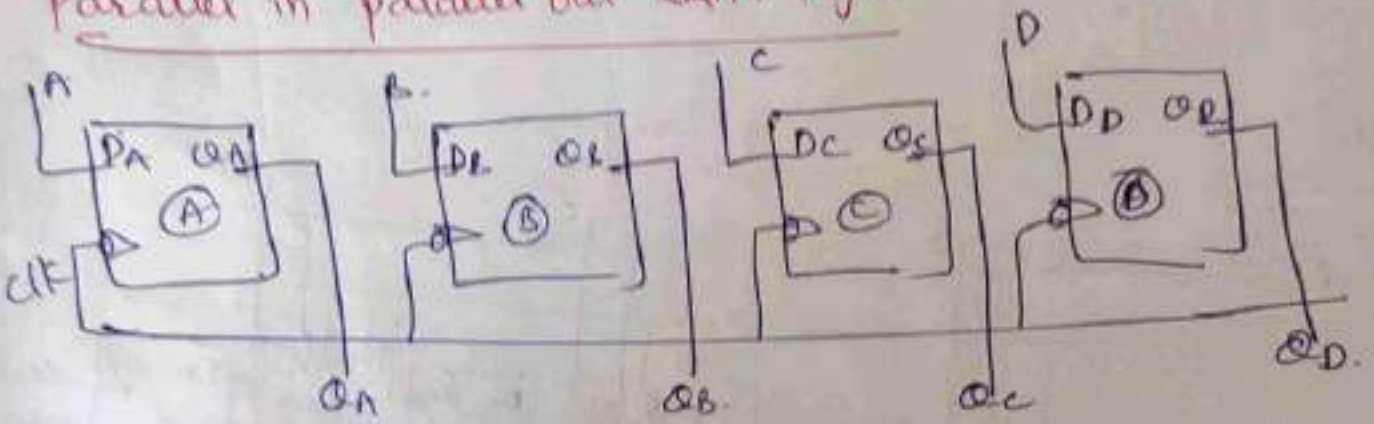
→ when clock pulse is applied the FFs with D=1 will SET and those with D=0 will RESET thus all 4 bits are stored simultaneously.

→ when shift/$\overline{load}$ is high, gates G_1 G_2 G_3 are disabled and gates G_4, G_5, G_6 are enabled

This allows the data bits to shift right from one state to the next

→ The OR gates at the D-I/Ps of the FFs allow either the parallel data entry operation (OR) shift operation, depending data entry operation (OR) shift operation depending on which AND gates are enabled by the level on the shift/load I/P.

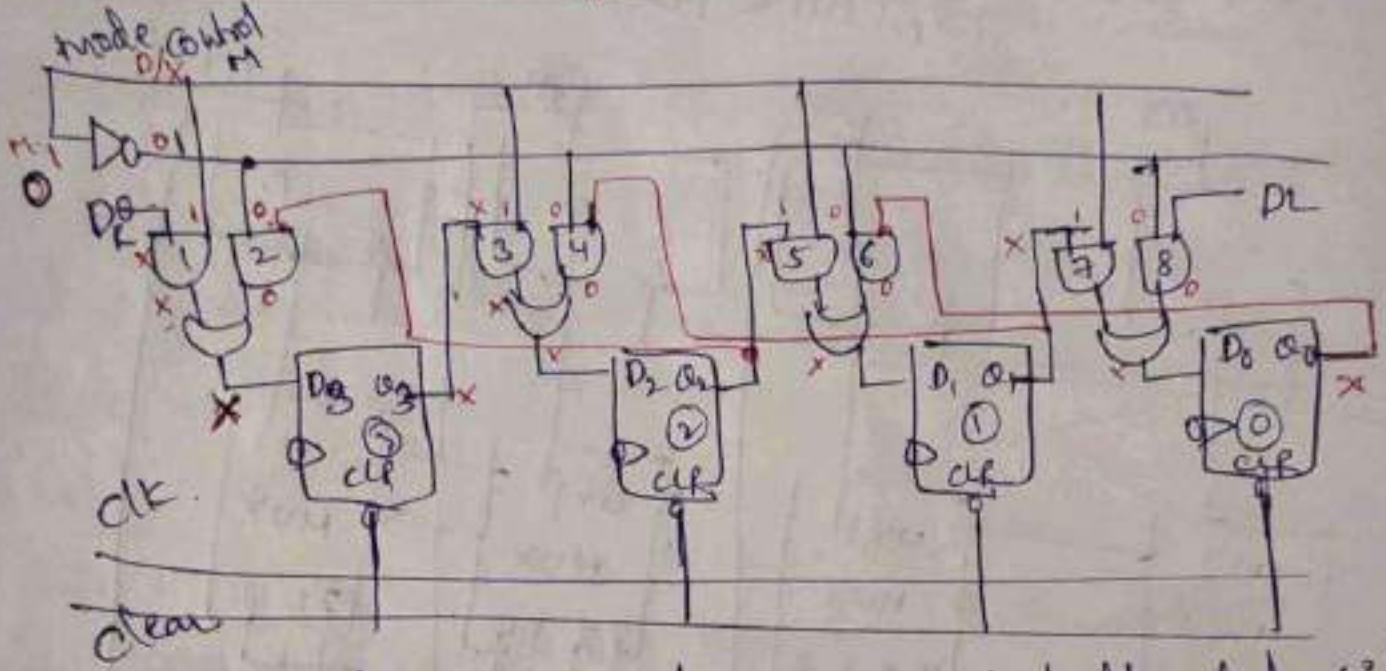
Parallel in parallel out shift register.



Applications of shift Register.

1) parallel to serial conversion
2) Serial to parallel Conversion
3) Dealy line
4) shift Register Counters
5) Sequence generator

Bidirectional shift Register.



A Bidirectional shift Register is one which the data bits can be shifted from left to right (or) from Right to left

→ Mode control M=1 it act as shift Right
M=0 it act as shift left

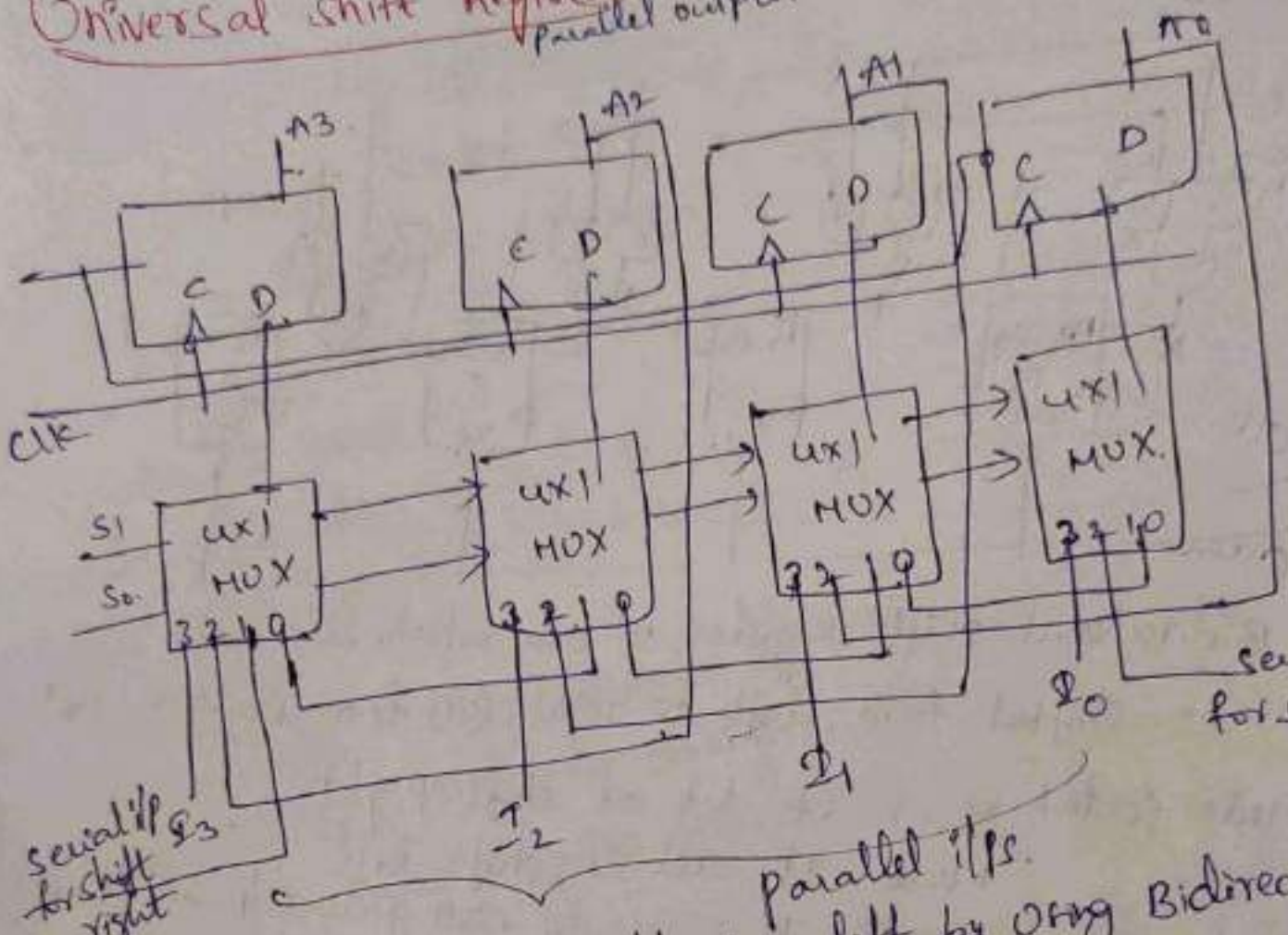
→ M=1, Any one of the i/p is zero for AND gate o/p will be 0

M=1, AND gate 1, 2 3 4 – enable
5, 6 7 8 – Disable

→ M=0 AND gate 1 2 3 4 – disable
5 6 7 8 – enable.

Universal shift Register

Parallel outputs



→ shift left, right & Right & left by using Bidirectional Parallel i/ps.

S1	S0	
0	0	→ No change
0	1	→ shift left
1	0	→ shift right
1	1	→ parallel load.

Counters

A Digital Counter is a set of FFs whose states change in response to pulses applied at the I/P to the counter.

→ The FFs are interconnected such that their combined state at any time is the binary equivalent of the total no of pulses that have occured upto that time.

→ A counter is used to count pulses.

→ A counter can also be used as a frequency divider to obtain waveforms with frequencies that are specific fractions of the clk frequency.

→ They are also used to perform the timing functions as in digital watches to create time delay to produce non-sequential binary counts, to generate pulse trains, and to act as frequency counters.

→ Counters may be asynchronous counters (or) synchronous counters.

→ Asynchronous counters are also called ripple counters. The ripple counter is the simplest type of counter.

→ The easier to design and requires the least amount of hardware.

→ In ripple counters the FFs within the counter are not made to change the states at exactly the same time.

→ Because the FFs are not triggered simultaneously the clk doesn't directly control the time at which every stage changes state.

→ An asynchronous counter uses FFs to perform a counting function.
→ The actual hardware used is usually J-K FFs connected in toggle-mode i.e with Js & Ks connected to logic 1
→ The asynchronous counter has a disadvantage in so-far as the unwanted spikes are concerned.
→ This limitations is overcome in parallel counters
→ The asynchronous counter is called ripple counter.

Asynchronous Counters	Synchronous Counters
1) In this type of counter FFs are connected in such a way that the o/p of first FF drives the clk for the 2nd FF, the o/p of 2nd the clk of the 3rd so on.	In this type of counter there is no connection b/w the o/p of first FF and clk i/p of next FF and so on.
2) All the FFs are not clocked simultaneously.	2) All the FFs are clocked simultaneously
3) Design and implementation is very simple even for more no of states.	3) Design and implementation becomes tedious and complex as the no of state increases.
4) Main drawbacks of the counters is their low speed as the clk is propagated through a No. of FFs before it reaches the last FF	4) clk is applied to all FFs simultaneously the total propagation delay is equal to the propagation delay of only one FF Hence they are faster

→ A counter may be an up-counter (or) down counter.

→ An up-counter is a counter which counts in the upward direction ie. 0, 1, 2, 3 ... N

→ A down-counter is a counter which counts in downward direction ie. N, N-1, N-2, ... 1, 0.

→ Each of the counters counts of the counter is called state of the counter.

→ The no. of states through which the counter passes before returning to the starting state is called the modulus of the counter.

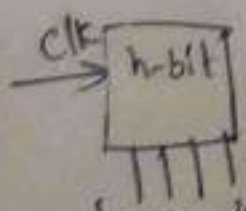
Design of Asynchronous Counter.

1) No. of states through counter passes.
2) Decide no. of bits for Ripple counter.
3) state diagram.
4) Truth table for Reset logic
5) K-map simplification.
6) Logic diagram.

Asynchronous Counters

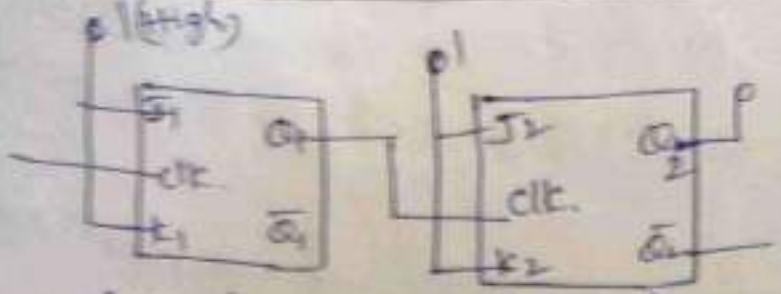
→ Counter consists of a series connection of complementing FFs with the clp of.

log binary counter can count is $2^n - 1$ Ex: 2 bit counter can count is $2^2 - 1 = 3$ (11)

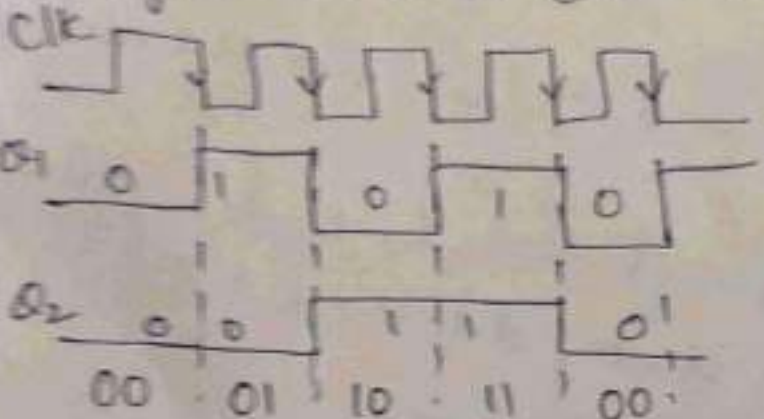


+ve (or) -ve Edge triggered n-bit counter

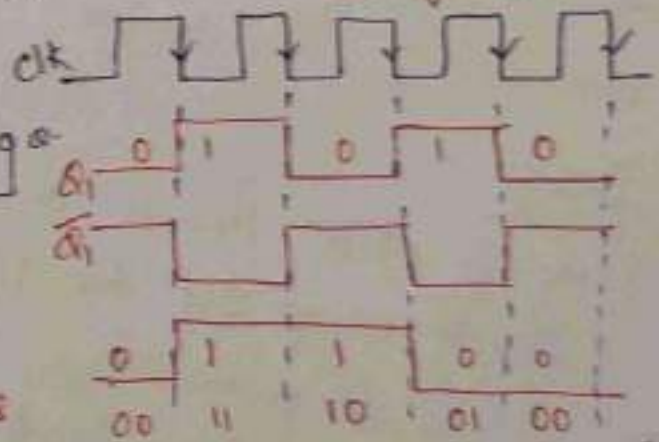
After reaching the maximum count the counter resets to '0' on arrival of the next CLK pulse and it starts counting again.



Asynchronous 2-bit Counter using Negative Edge triggered

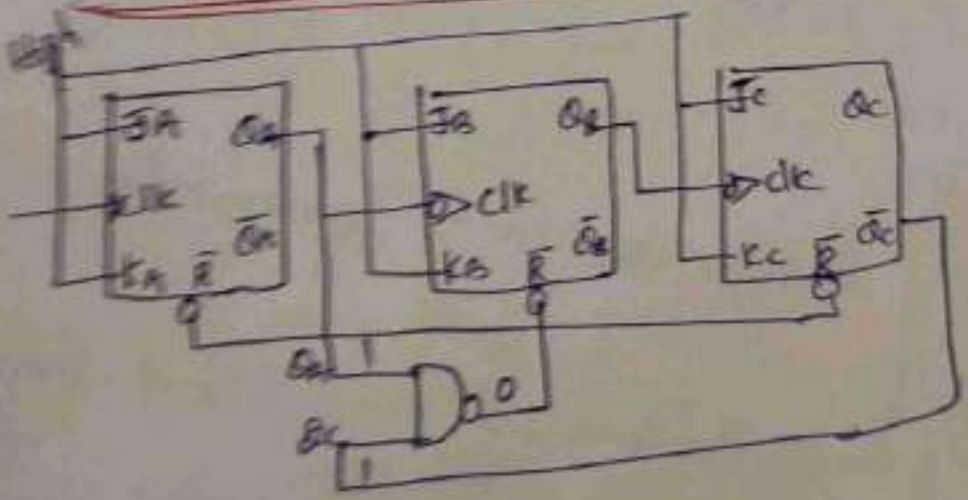


Timing diagram

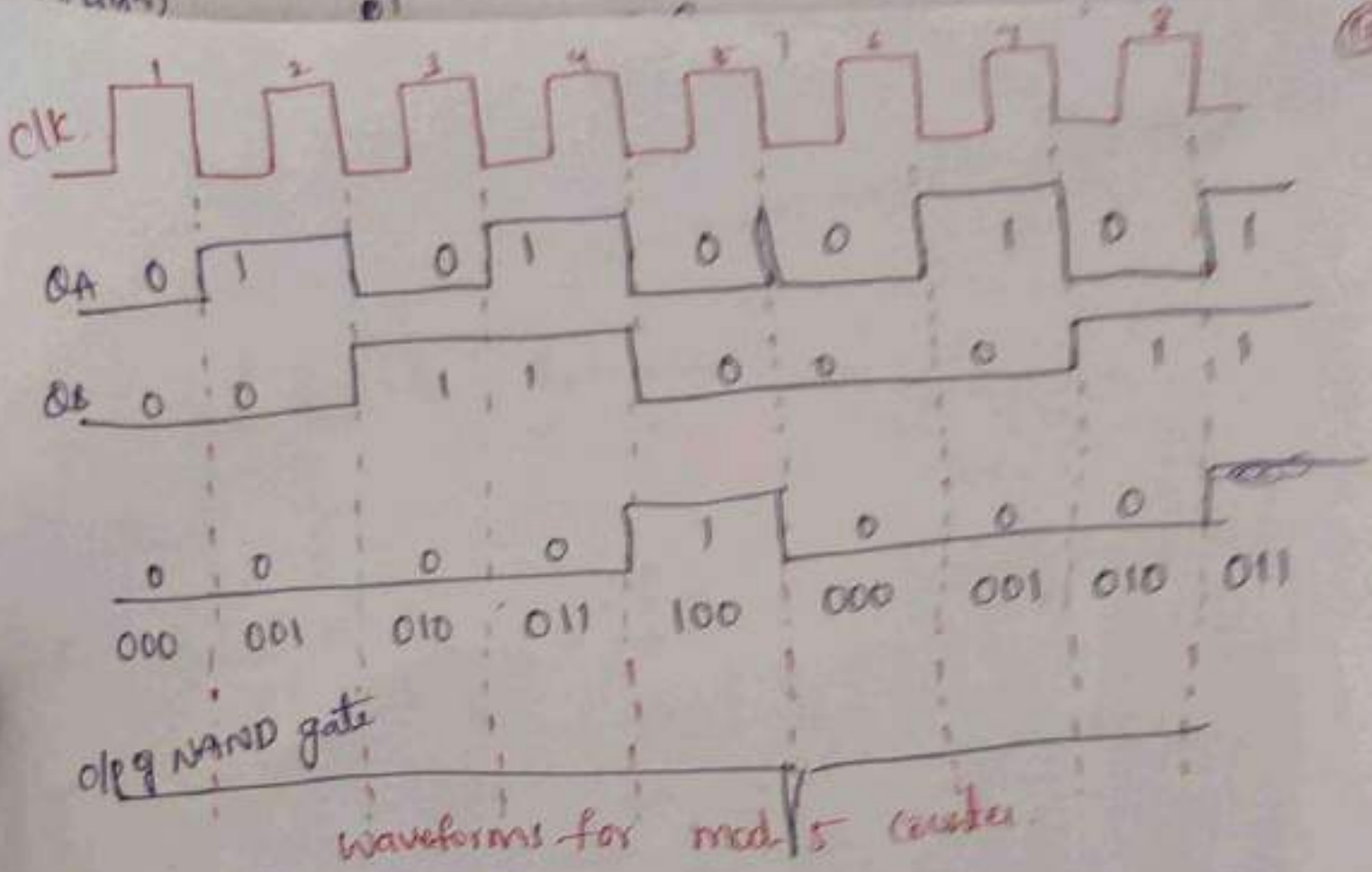


Two-bit down Counter

Design mod-5 Asynchronous Counter



Clk	C	B	A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
	1	0	1



waveforms for mod-5 counter.

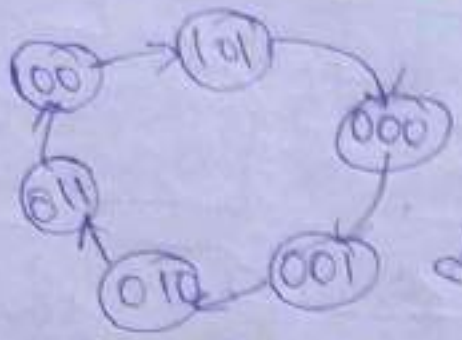
State diagram

Design of synchronous counters:

1) obtain no of flip-flops required
2) state diagram
3) choice of FFs and excitation table
4) Minimal expressions for excitations
5) Logic diagram.

Design of a synchronous mod-6 counter using JK.

1) mod-6 counter is sequence.
000 001 010 011 100 101, 000 It has six states.
It requires $n=3$, FFs, $2^3=8$ states. $N \geq 2^n$
$6 \geq 2^3$



state diagram

000 → 0
001 → 1
010 → 2
011 → 3
100 → 4
101 → 5
000 → 6

Excitation table

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Type of FF & Excitation table

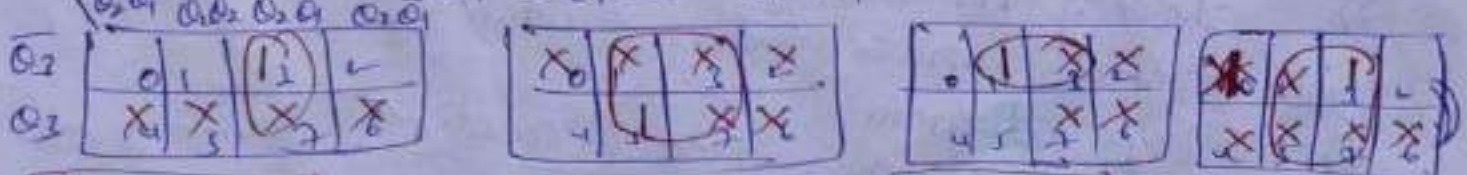
State table

Present state Q_3 Q_2 Q_1	Next state Q_3^+ Q_2^+ Q_1^+	J_3 K_3	J_2 K_2	J_1 K_1
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
1 1 0	X X X	X X	X X	X X
1 1 1	X X X	X X	X X	X X

Apply Excitation table of JK to Q_3 & Q_2 + colum then we will get J_3 K_3 values.

By apply to Q_2 & Q_2^+ → J_2 K_2

Q_1 & Q_1^+ → J_1 K_1



$J_3 = Q_2 Q_1$

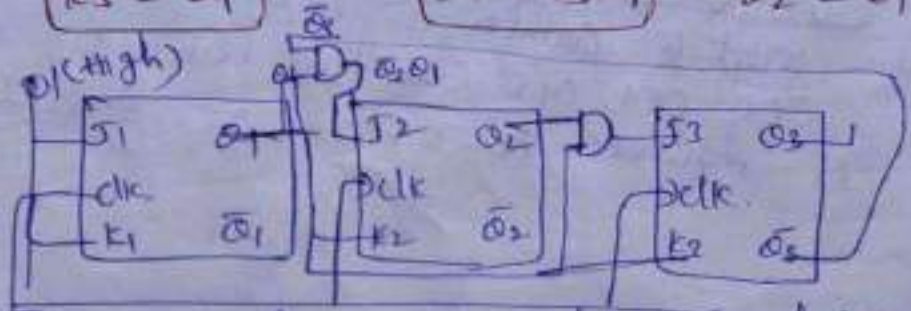
$K_3 = Q_1$

$J_2 = \bar{Q}_3 Q_1$

$K_2 = Q_1$

$J_1 = 1$

$K_1 = 1$



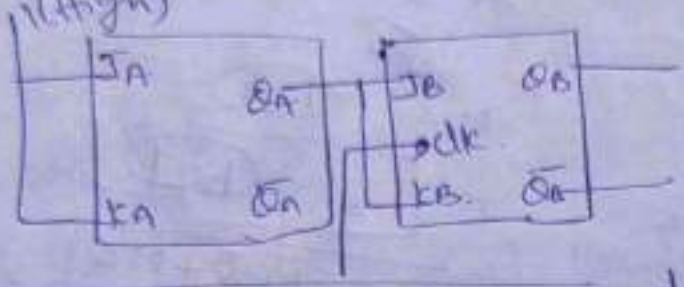
Logic diagram of synchronous mod-6 counter using JK FF

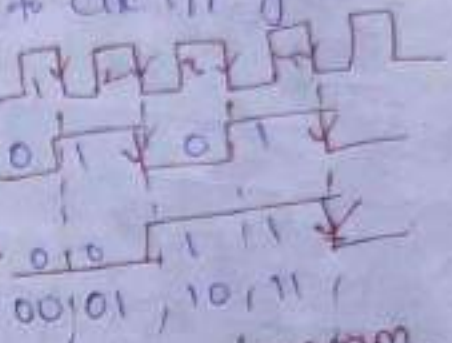
Synchronous Counter

→ Each flip-flop in the counter is triggered at the same time the counter is called as synchronous counter.

→ clk signal is connected in parallel to clk i/p of both FFs

→ (1(High))





2 bit synchronous counter. Timing diagram.

→ Q_A o/p of first stage is used for drive the JK i/p of 2nd stage

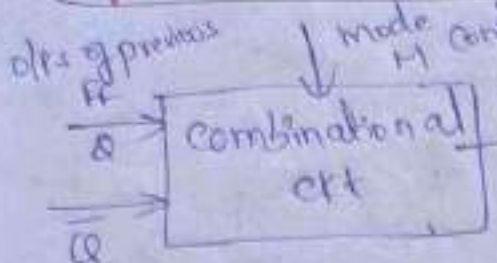
→ Let us see operation ① First clk pulse we assume $Q_A = Q_B = 0$. +ve edge FFA will toggle because $J_A = K_A = 1$. FFB o/p will remain zero. because $J_B = K_B = 0$.

② After 1st clk $Q_A = 1$, $Q_B = 0$ –ve edge In this both FFs will toggle because they both have a toggle condition ($J_A = K_A = J_B = K_B = 1$)

③ After 2nd clk $Q_A = 0$, $Q_B = 1$ –ve edge triggering of 3rd pulse FFA toggle $Q_A = 1$ FFB $Q_B = 1$

④ leading edge of 4 clk. JK both are toggles $Q_A = Q_B = 0$. & counter recycled back to its original state

Asynchronous up/down Counter.

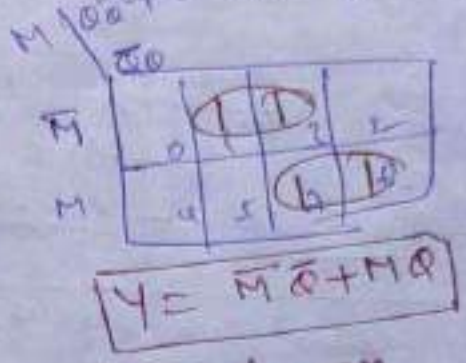


1) A mode control i/p is used to select either up(00) down mode
2) Combinational ckt is required b/w each pair of FFs.

inputs M	Q	$\bar{Q}$	o/p Y	
0	0	0	0	
0	0	1	1	(1)
0	1	0	0	(2)
0	1	1	1	
1	0	0	0	
1	0	1	0	(4)
1	1	0	1	(5)
1	1	1	1	

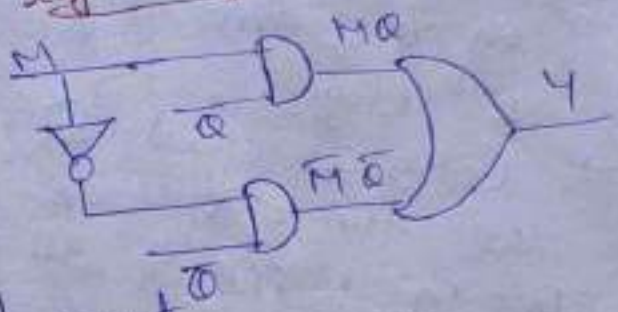
M=0 : $Y = \bar{Q}$ for down counting

M=1 : Y = 0 for up counting.

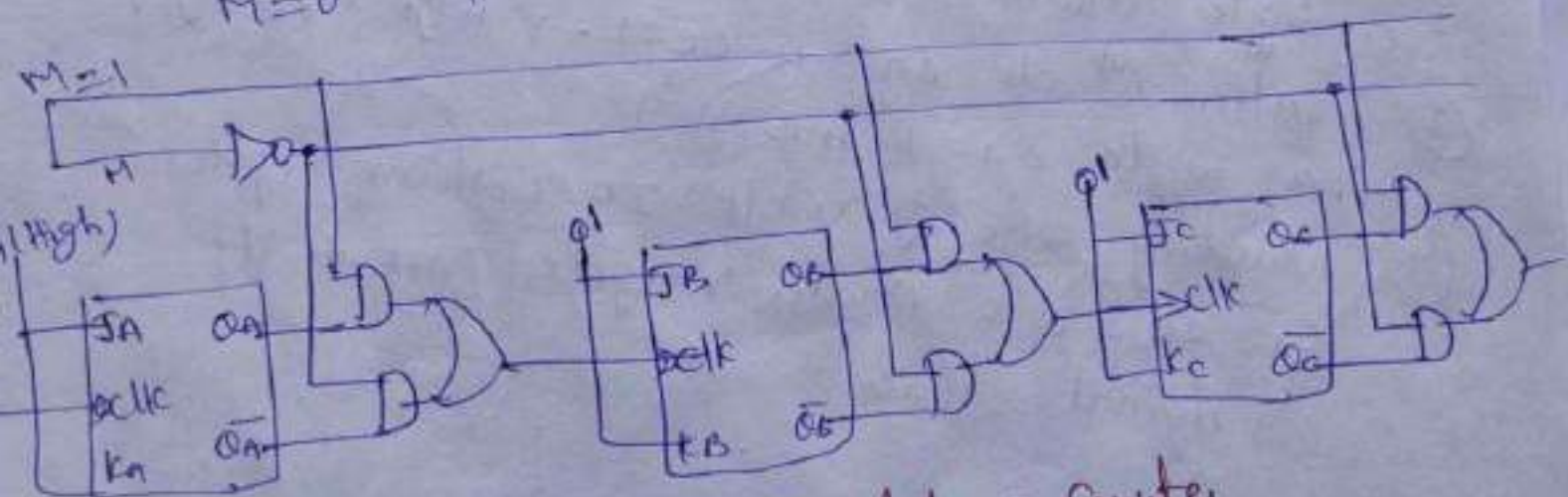


$$Y = \bar{M}\bar{Q} + MQ$$

logic diagram



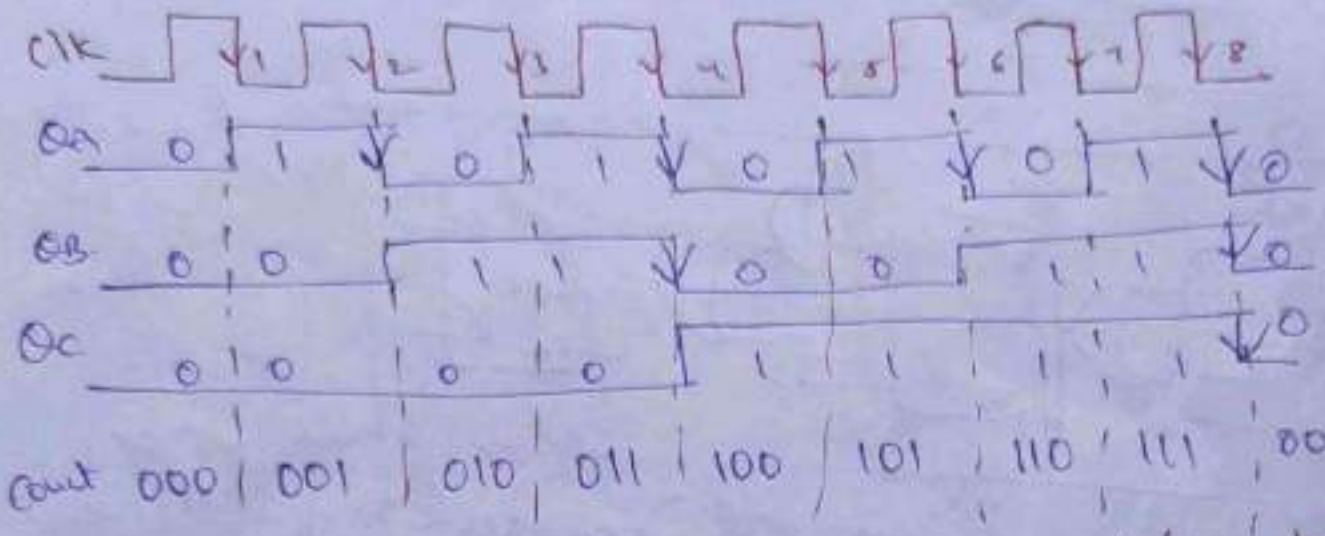
3 bit up/down counter that will count when M=1 it will count from 000 upto 111
M=0 " 111 down to 000.



3-bit Asynchronous up/down Counter

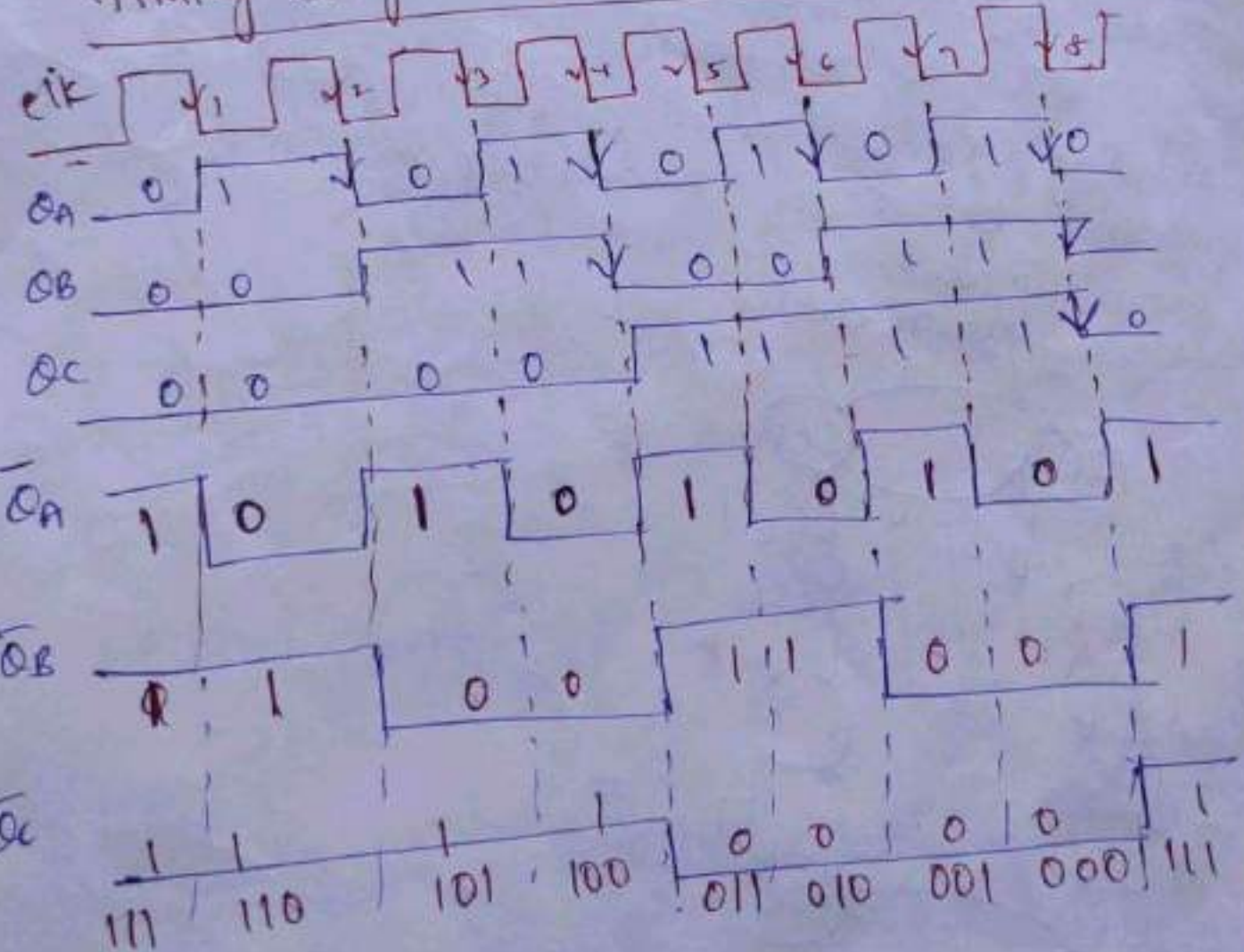
Timing diagram for 3 bit up/down ripple counter.

up/down = 1

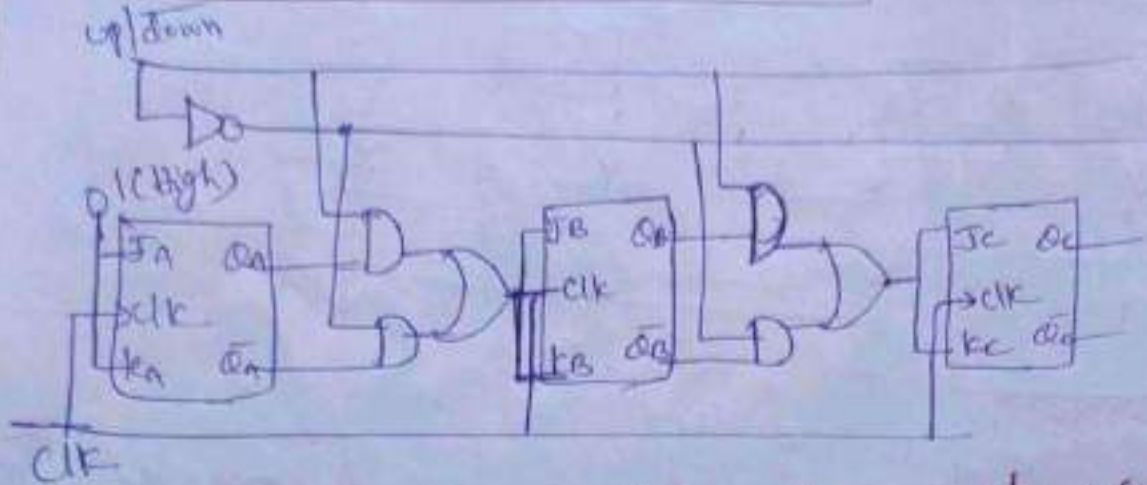


up/down = 0

Timing diagram for 3 bit down ripple (Asynchronous) counter



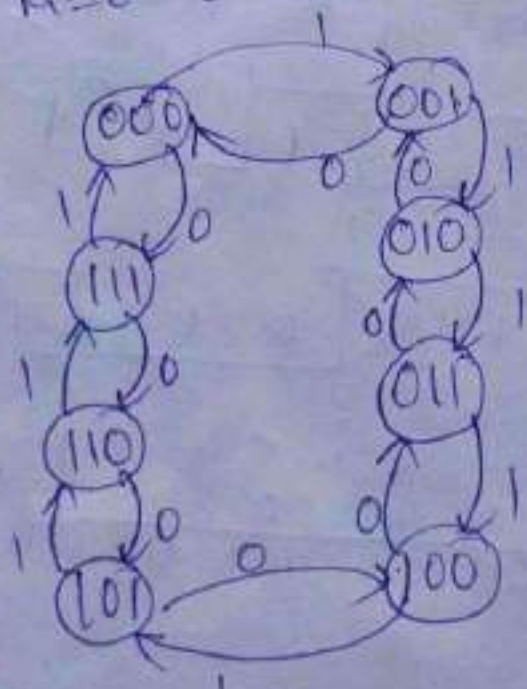
3 bit Synchronous up/down Counter.



Design of synchronous 3 bit up-down counter using JK FF.

3 bit Counter requires 3 FFs. It has 8 states
000, 001, 010, 011, 100, 101, 110, 111

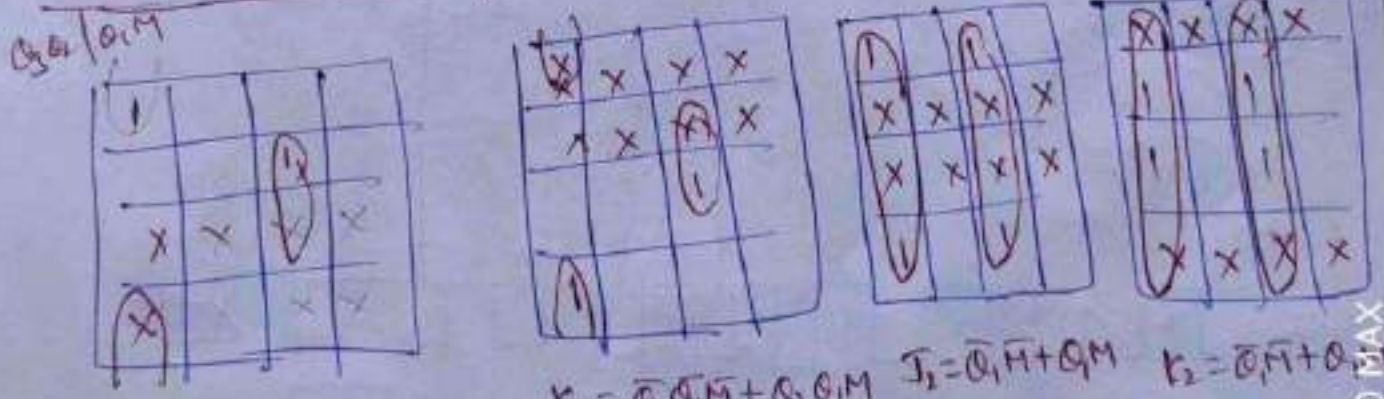
→ For selecting up & down modes, a control (or) mode signal M is required. Let us say it counts up when mode signal M=1 up counter.
M=0 down counter.



State diagram.

State table

Present state Q_3	Q_2	Q_1	mode M	Q_3^+	Q_2^+	Q_1^+	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	1	0	X	0	X	1	X
0	0	0	1	0	1	1	0	X	1	X	1	X
0	0	1	0	0	1	0	0	X	X	0	1	X
0	0	1	1	0	1	1	0	X	X	0	1	X
0	1	0	0	0	1	1	0	X	X	0	X	0
0	1	0	1	0	1	1	0	X	X	0	X	0
0	1	1	0	1	0	0	1	X	X	1	X	1
0	1	1	1	0	1	1	0	X	X	0	X	0
1	0	0	0	1	0	1	X	0	0	X	1	X
1	0	0	1	1	0	1	X	0	0	X	1	X
1	0	1	0	1	1	0	X	0	1	X	X	1
1	0	1	1	1	1	0	X	0	1	X	X	1
1	1	0	0	1	1	1	X	0	X	0	1	X
1	1	0	1	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	0	X	1	X	1	X	1
1	1	1	1	0	0	0	X	1	X	1	X	1

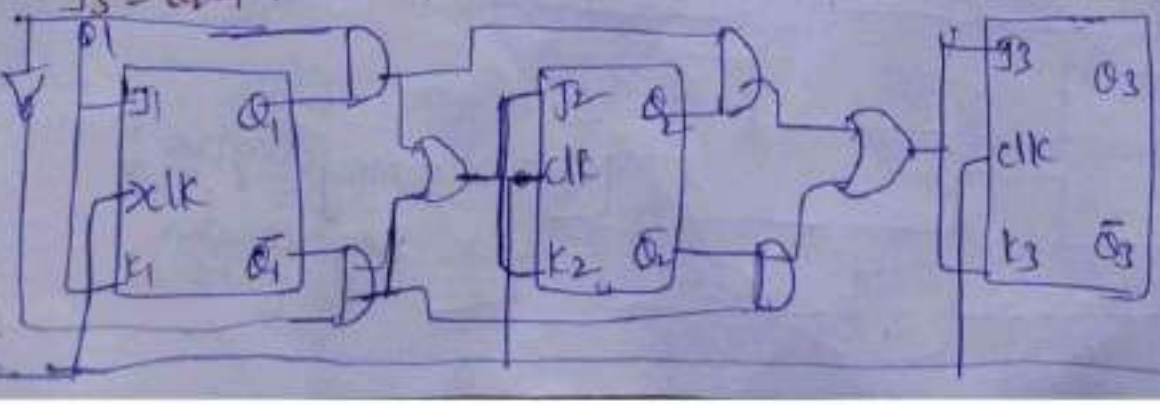


$J_3 = \bar{Q}_3\bar{Q}_1\bar{M} + Q_2Q_1M$

$K_2 = \bar{Q}_3Q_1\bar{M} + Q_2Q_1M$

$J_1 = \bar{Q}_1\bar{M} + Q_1M$

$K_2 = \bar{Q}_1\bar{M} + Q_1M$



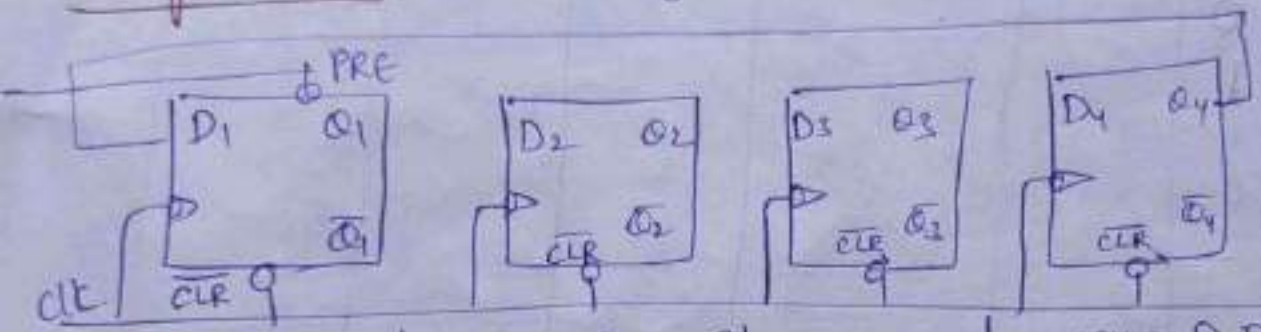
... counter.

Shift Register counters

→ One of the applications of shift register is that they can be arranged to form several types of counters.

→ shift register counters are obtained from serial-in, serial-out shift registers by providing feedback from the o/p of last FF to the i/p of first FF. These devices are called counters.

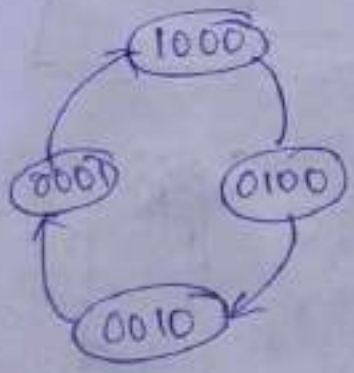
Ring counter (Twisted ring (or) Johnson counter)



logic diagram of 4bit ring counter using D FF

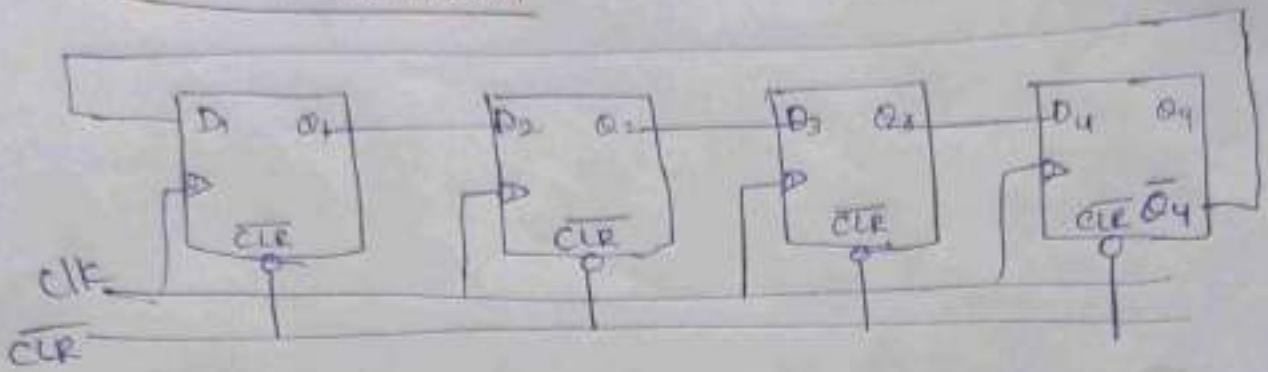
Q_1	Q_2	Q_3	Q_4	After clk pulse
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7

no. of states = no. of FF
PR = 0, Q = 1
CLR = 0, Q = 0

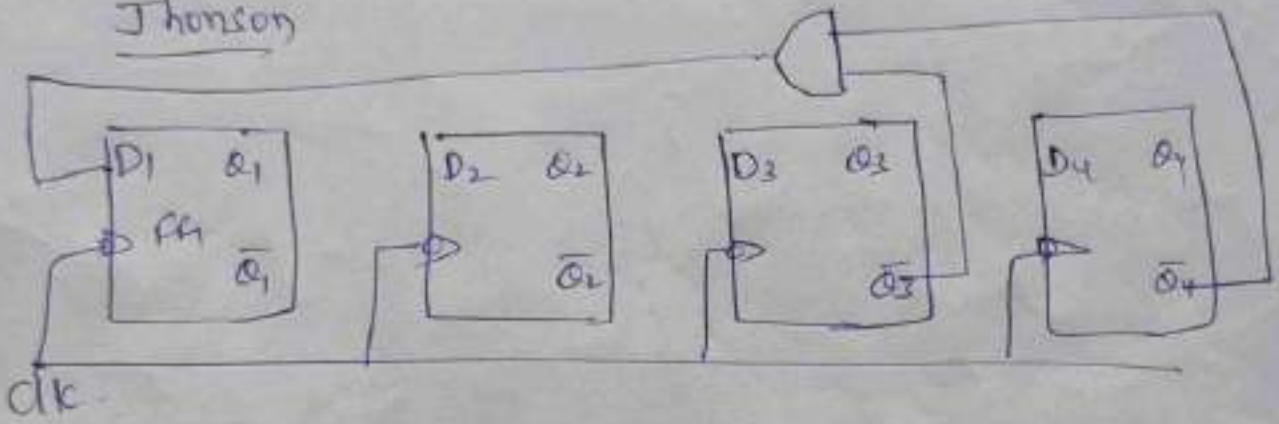


Timing diagram of a 4bit ring counter

Twisted Johnson

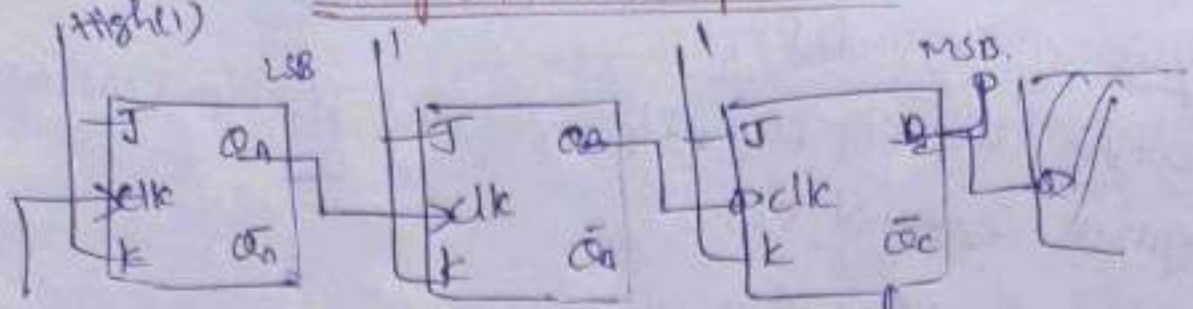


Jhonson

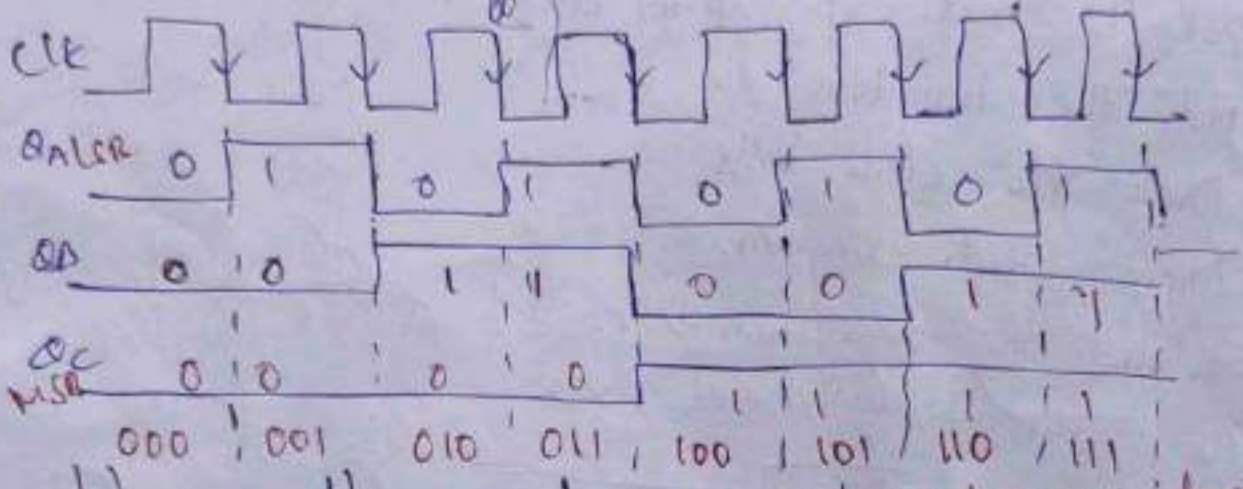


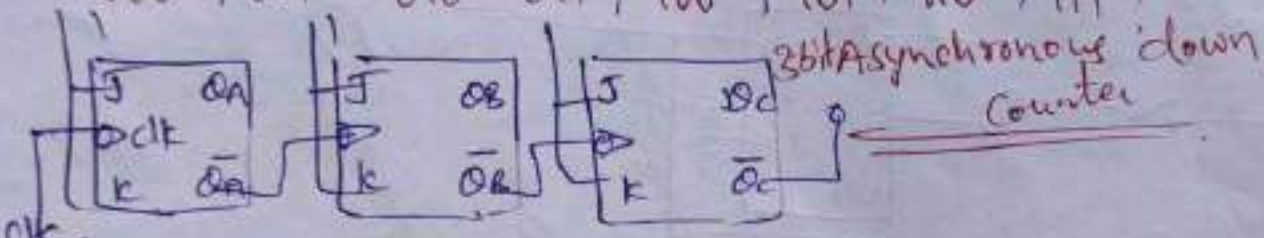
CLR	clk	Q_1	Q_2	Q_3	Q_4	
↑	X	0	0	0	0	— 1
:	↓	1	0	0	0	— 2
:	↓	1	1	0	0	— 3
:	↓	1	1	1	0	— 4
:	↓	1	1	1	1	— 5
:	↓	0	1	1	1	— 6
:	↓	0	0	1	1	— 7
:	↓	0	0	0	1	— 8
	↓	0	0	0	0	

3 bit Asynchronous up-counter.

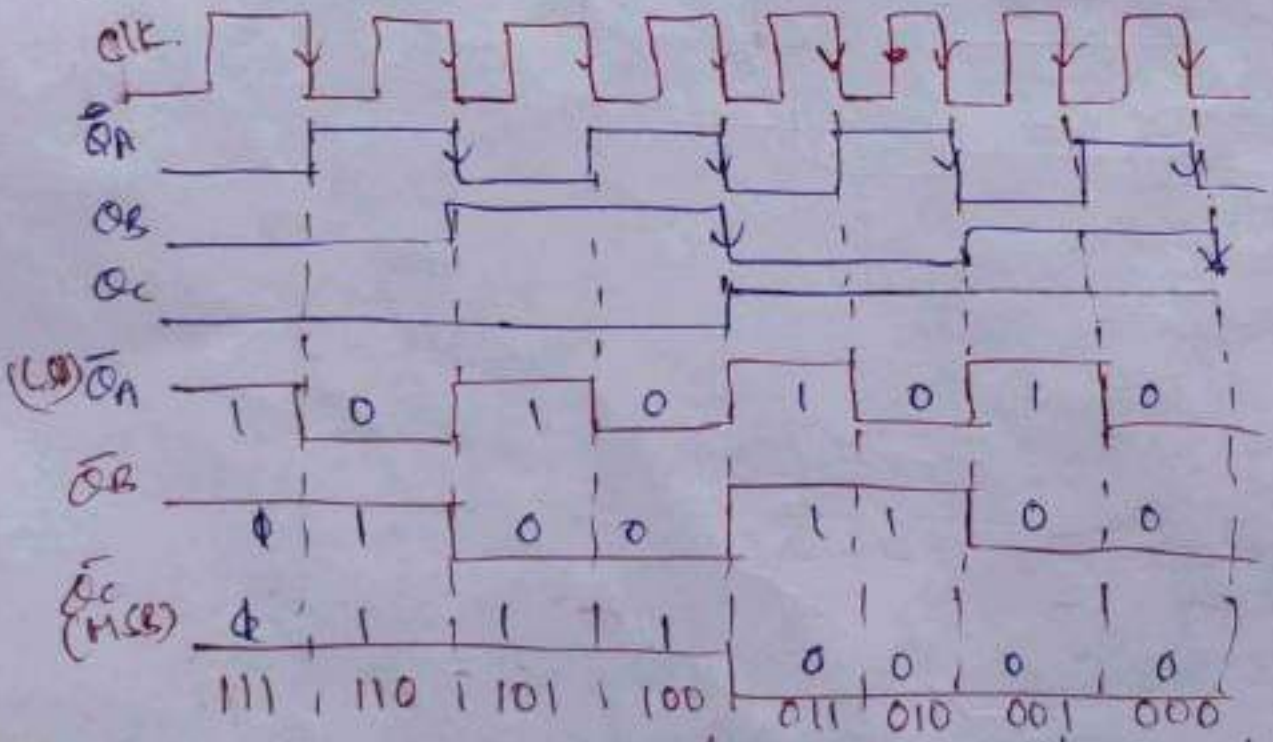


3 bit Asynchronous up counter.





3 bit Asynchronous down Counter



Timing diagram for 3 bit Asynchronous down Counter.

By using eqns Draw state table, state diagram & Sequential ckt for given eqns

$T_1 = Q_1Q_2 + Q_2x$

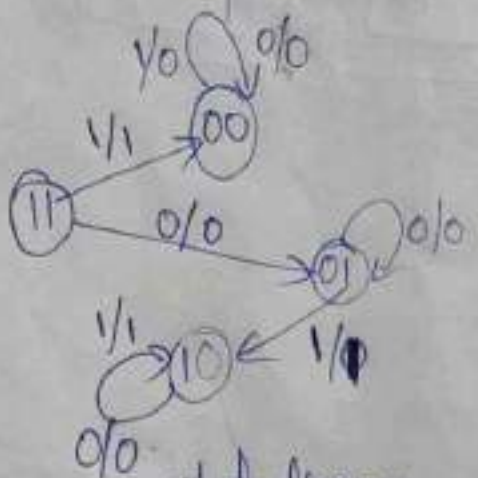
$T_2 = Q_1x + Q_2x + Q_1Q_2x$

$o/p_2 = (Q_1 + Q_2)x$

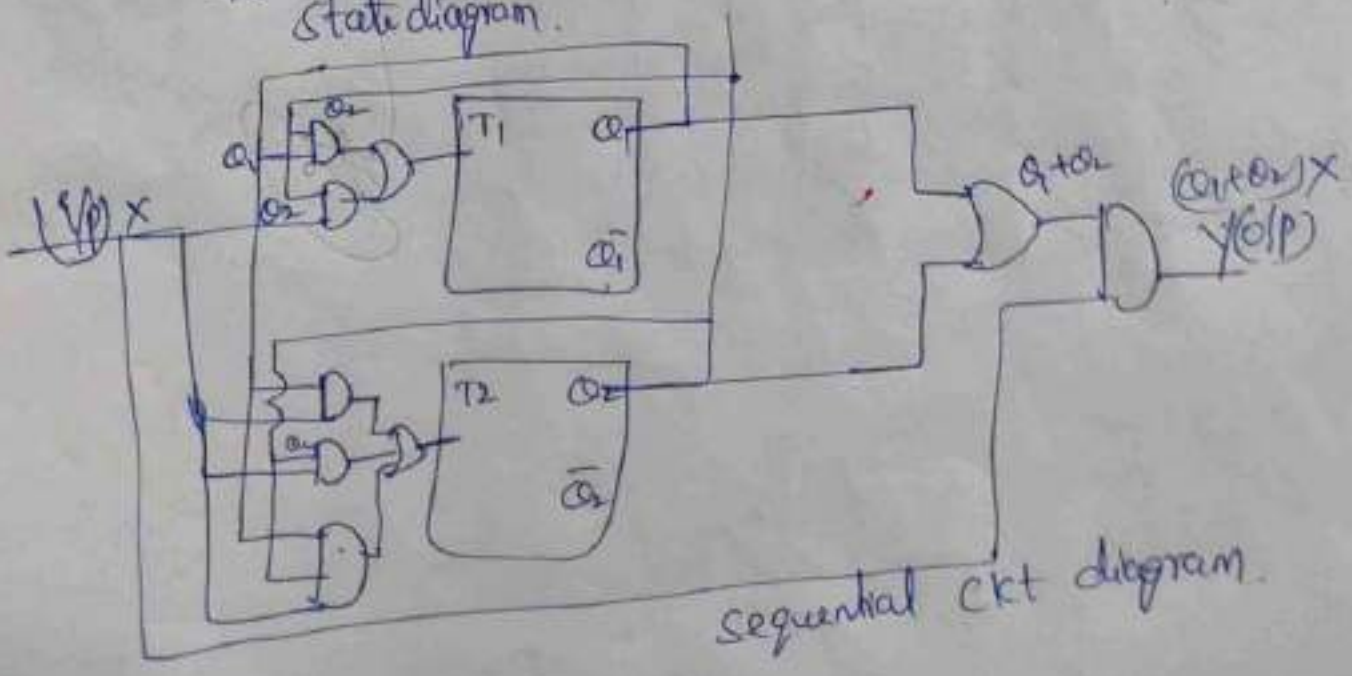
Q_n	T	Q_{n+1}
0	0	0 (Q_n)
0	1	1 ($\bar{Q}_n$)
1	0	1 (Q_n)
1	1	0 ($\bar{Q}_n$)

Q_1	Q_2	x	$T_1 = Q_1Q_2 + Q_2x$	$T_2 = Q_1x + Q_2x + Q_1Q_2x$	Q_1^+	Q_2^+	$o/p = (Q_1+Q_2)x$
0	0	0	0+0=0	0+0+0=0	0	0	0
0	0	1	0+0=0	0+0+0=0	0	0	0
0	1	0	0+0=0	0+0+0=0	0	1	0
0	1	1	0+1=1	0+1+0=1	1	0	1
1	0	0	0+0=0	0+0+0=0	1	0	0
1	0	1	0+0=0	0+0+0=0	1	0	1
1	1	0	1+0=1	0+0+0=0	0	1	0
1	1	1	1+1=1	1+1+1=1	0	0	1

state table



state diagram.



sequential ckt diagram.