End Term Exam DLD Solutions Jan-Feb 2023

Q.1 (a) (i) $(225.225)_{10} = (\_)_2$

2	225	
2	112	1
2	56	0
2	28	0
2	14	0
2	7	0
2	3	1
2	1	1
	0	1

$(225)_2 = (11100001)_2$

~~(0.225)~~

$0.225 \times 2 = 0.45$
$0.45 \times 2 = 0.9$
$0.9 \times 2 = 1.8$
$0.8 \times 2 = 1.6$
$0.6 \times 2 = 1.2$
$0.2 \times 2 = 0.4$
$0.4 \times 2 = 0.8$
$0.8 \times 2 = 1.6$
$0.6 \times 2 = 1.2$
$0.2 \times 2 = 0.4$

$(0.225)_2 = (0.0011100110)$

$\Rightarrow (225.225)_{10} = (11100001.0011100110)_2$

(bi) $(2AC5.D)_{16} = (\_)_{10}$

$(2AC5.D)_{16} = 5\times16^0 + 12\times16^1 + 10\times16^2 + 2\times16^3 + 13\times16^{-1}$
$= 5 + 192 + 2560 + 8192 + 0.8125$
$= (10949.8125)_{10}$

(iii) $(CAFE)_{16} = (\_)_8$

$(CAFE)_{16} = (1100\ 1010\ 1111\ 1110)_2$
$= (145376)_8$

0	0	0	-0
0	0	1	-1
0	1	0	-2
0	1	1	-3
1	0	0	-4
1	0	1	-5
1	1	0	-6
1	1	1	-7

Q.1 (b) (i) $AB + A\bar{C} + A\bar{B}C\,(AB + C)$

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

$= AB + A\bar{C} + A\bar{B}C$ $(\because \bar{B} \cdot B = 0)$

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

$= A(B+\bar{B})(B+C) + A\bar{C}$ $[x + yz = (x+y)(x+z)]$

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

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

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

$= AB + A$

$= A(B+1)$

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

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

~~$\overline{AB} \cdot \overline{ABC} +$~~

$= \overline{AB(1+C)} + AB + A \cdot AB$

$= \overline{AB} + AB$

$= 1$

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

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

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

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

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

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

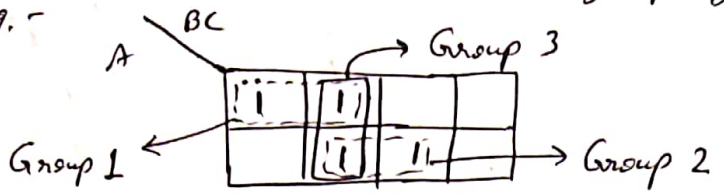
Q.1 (c) • Implicants are minterms in SOP's or maxterms in POS's of a Boolean Function. For eg:- Considering a Boolean function, $F = AB + ABC + BC$.

Implicants are AB, ABC, BC.

• Prime Implicants :- A prime implicant is a product term obtained by combining the maximum possible no. of adjacent squares in the ~~map~~ K-map.

In other words, all possible groups formed in K-map.

For eg.-

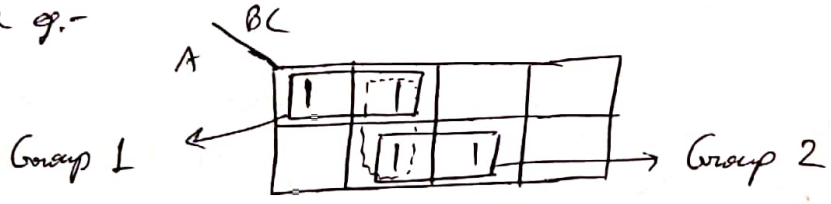


Here, No. of prime implicants = 3

- Essential Prime Implicant :- If a minterm in a square is covered by only one prime implicant, that prime implicant is said to be essential.

In other words, essential prime implicants are those prime implicants that always appear in the final solution in K-map.

For eg.-



Here, No. of essential prime implicants = 2

Q.1 (d) Design procedure for designing the combinational circuits involves the following steps :-

1.) The problem is stated.
2.) The no. of available input variables and required output variables is determined.
3.) The input and output variables are assigned letter symbols.
4.) The truth table that defines the required relationships between inputs and outputs is derived.
5.) The simplified Boolean function for each output is obtained.
6.) The logic diagram is drawn.

9.1 (e)

	Combinational Circuits	Sequential Circuits
①	Output depends only on the present values of input	output depends on the present and past values of input.
②	Feedback path is not used in combinational circuits	Feedback path is used for sequential circuits.
③	Memory element is not present.	Memory element is present.
④	Clock is not used.	clock is used.
⑤	Circuit is simple	Circuit is complex.
⑥	Examples :- Adders, Subtractors, Code converters, comparators, MUX, DEMUX, decoder, encoder, etc	Examples :- Flip-flops, Counters, registers, etc

9.2 (a) The number system has different bases ~~and~~ (or radix) and the most common radix are decimal, binary, octal and hexa-decimal.

- If the number system representing the digit from 0-9, then the base of the system is 10.
- If the number system representing the digit in the form of 0 and 1, then the base of the system is 2.
- If the number system representing the digit from 0-7, then the base of the system is 8.
- If the number system representing the digit from 0-9 and A-F, then the base of the system is 16.

(i) 101011 - 111000

Minuend : 101011

subtrahend : 111000

2's complement of subtrahend : 001000

~~Difference~~ Addition : 110011

∴ 101011 - 111000 = - 11101

111000 → 000111 + 1 = 001000

As there is no carry, we take 2's complement of 110011 i.e 001100 + 1 = 001101

(ii) 1110 − 110010

Minuend : 1110 — 110010

Subtrahend : 110010 — 001101

2's complement of subtrahend : 001110 — 001101 + 1 = 001110

Addition : 011100

As there is no carry in addition, we take 2's complement of 011100 i.e ~~011100~~ 100011 + 1 = 100100

∴ 1110 − 110010 = − 100100

(iii) 11010 − 1101

Minuend : 11010

subtrahend : 01101

2's complement of subtrahend : 10011

Addition : ①01101

01101 → 10010 + 1 = 10011

After dropping the carry, we get the result as 01101

∴ 11010 − 1101 = 01101

8.2(b) Absorption Law : It states that

A + A·B = A (OR absorption Law) and A·(A+B) = A (AND absorption Law)

Proof:

A + A·B = A·1 + A·B = A(1+B) = A

A·(A+B) = (A+0)·(A+B) = A + 0·B = A

Consensus Law : It states that

AB + A'C + BC = AB + A'C

(A+B)·(A'+C)·(B+C) = (A+B)·(A'+C)

Proof: ① $AB + A'C + BC$

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

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

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

$= AB + A'C$

② $(A + B)(A' + C)(B + C)$

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

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

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

$= (A + B)(A' + C)$ [Considering $A + B = x$, $A' + C = y$] (and apply absorption law)

8.3(a) Error Detection and Error Correction Codes

The complexity level of a memory array may cause occasional errors in storing and retrieving the binary information. The reliability of a memory unit may be improved by employing error-detecting and correcting codes.

The most common error-detection scheme is the parity-bit. A parity bit is generated & stored along with the data word in memory. The parity of the word is checked after reading it from the memory. The data word is accepted if the parity sense is correct. If the parity checked results in an inversion, an error is detected, but it cannot be corrected.

An error-correcting code generates multiple check bits that are stored with the data word in memory. Each check bit is a parity over a group of bits in the data word. When the word

is read from memory, the associated parity bits are also read from memory and compared with a new set of check bits generated from the read data. If the check bits compare, it signifies that no error has occurred. If the check bits do not compare with the stored parity, they generate a unique pattern, called a syndrome, that can be used to identify the bit in error. A single error occurs when a bit changes in value from 1 to 0 or from 0 to 1 during the write or read operation. If the specific bit in error is identified, then the error can be corrected by complementing the erroneous bit. One of the most common error-correcting codes is Hamming Code.

⇒ 7-bits Hamming Code is received as 0101101.

- In 7-bits Hamming Code, 4-bits (D_3, D_5, D_6 & D_7) represents data-bits and 3-bits (P_1, P_2 & P_4) represents parity-bits.
- The 7-bit Hamming Code is represented as follows :-

Bit Position	1	2	3	4	5	6	7
Hamming Code Representation	P_1	P_2	D_3	P_4	D_5	D_6	D_7
Received Hamming Code	0	1	0	1	1	0	1

The check-bits are evaluated as follows :-

- C_1 = XOR of bits (1, 3, 5, 7) = $0 \oplus 0 \oplus 1 \oplus 1 = 0$
  i.e Even number of 1's which means 'no' error is present.
- C_2 = XOR of bits (2, 3, 6, 7) = $1 \oplus 0 \oplus 0 \oplus 1 = 0$
  i.e Even number of 1's which means 'no' error is present.

- C_4 = XOR of bits (4, 5, 6, 7) = $1 \oplus 1 \oplus 0 \oplus 1 = 1$

i.e. Odd number of 1's which means error is present.

$\therefore C = C_4 C_2 C_1 = 100$ and this represents the decimal number 4.

Hence, a single-bit error is present at the 4th position of received Hamming code 0101101.

Therefore, correct code is 0100101.

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

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

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

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

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

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

$= \sum m(1, 4, 5, 6, 7)$

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

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

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

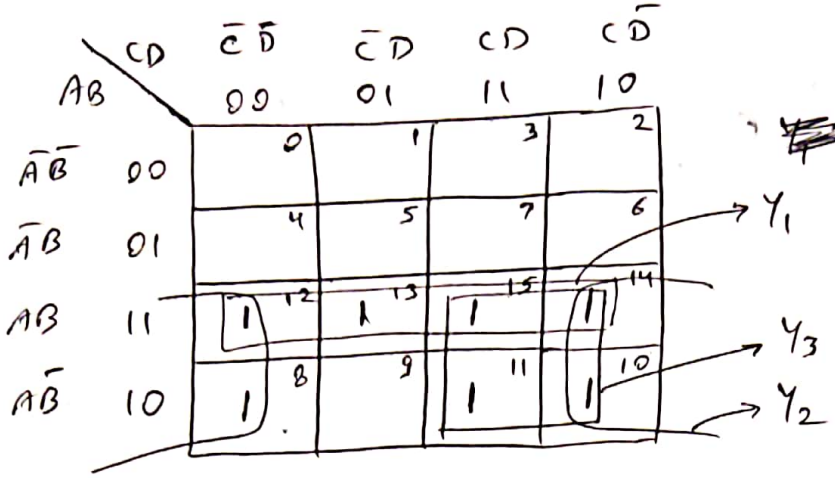
$= (A + \bar{B} + C\bar{C})(A + C + B\bar{B})$

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

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

$= \prod M(0, 2, 3)$

Q.4 (a) $Y = ABCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C + AB$



$Y_1 = AB$

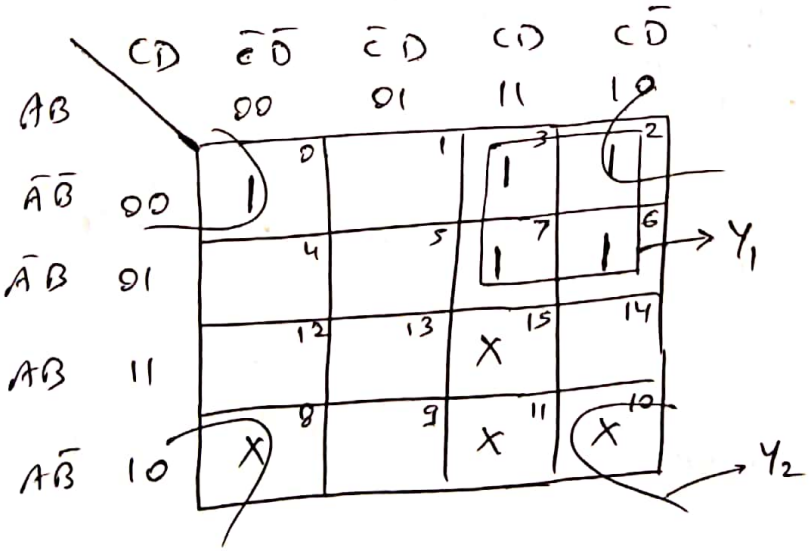
$Y_2 = A\bar{D}$

$Y_3 = AC$

Simplified expression is given as :-

$$Y = Y_1 + Y_2 + Y_3$$
$$= AB + A\bar{D} + AC$$
$$= A(B + C + \bar{D})$$

Q.4 (b) (i) $Y = \sum m(0, 2, 3, 6, 7) + \sum d(8, 10, 11, 15)$



$Y_1 = \bar{A}C$

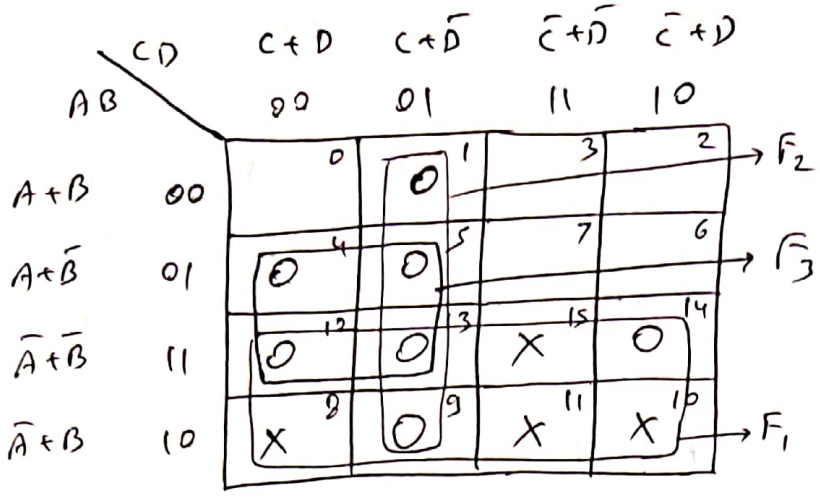
$Y_2 = \bar{B}\bar{D}$

$Y = Y_1 + Y_2$

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

(ii) For POS, let the Boolean function be F.

$$F = \prod M(1, 4, 5, 9, 12, 13, 14) + \sum d(8, 10, 11, 15)$$



$F_1 = \bar{A}$

$F_2 = C + \bar{D}$

$F_3 = \bar{B} + C$

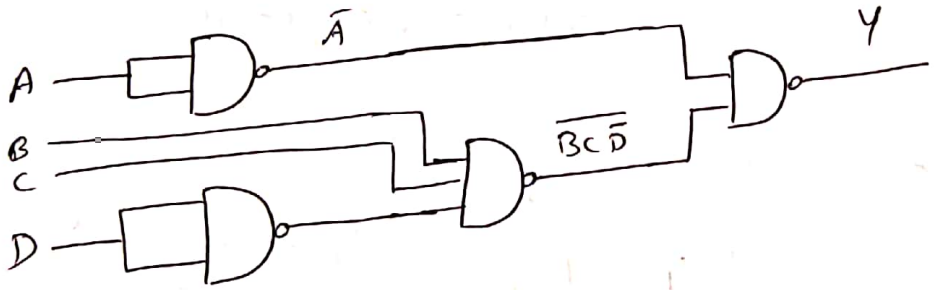
$\therefore F = F_1 \cdot F_2 \cdot F_3$

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

Q.5(a) $Y = A + BC\bar{D}$ using NAND gates

$= \overline{\overline{A + BC\bar{D}}}$

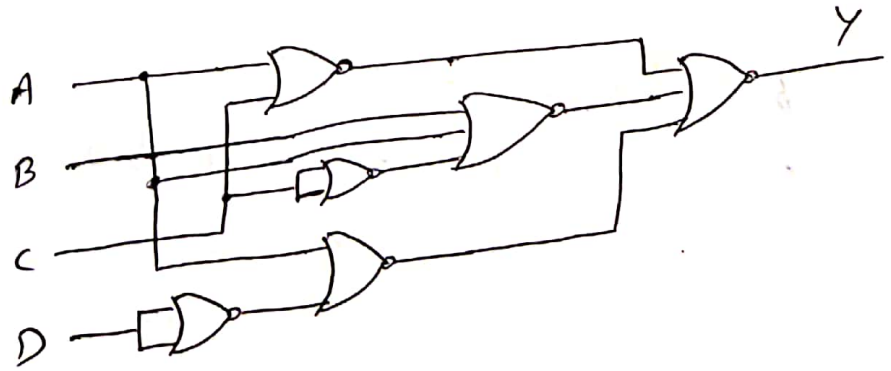
$= \overline{\bar{A} \cdot \overline{BC\bar{D}}}$



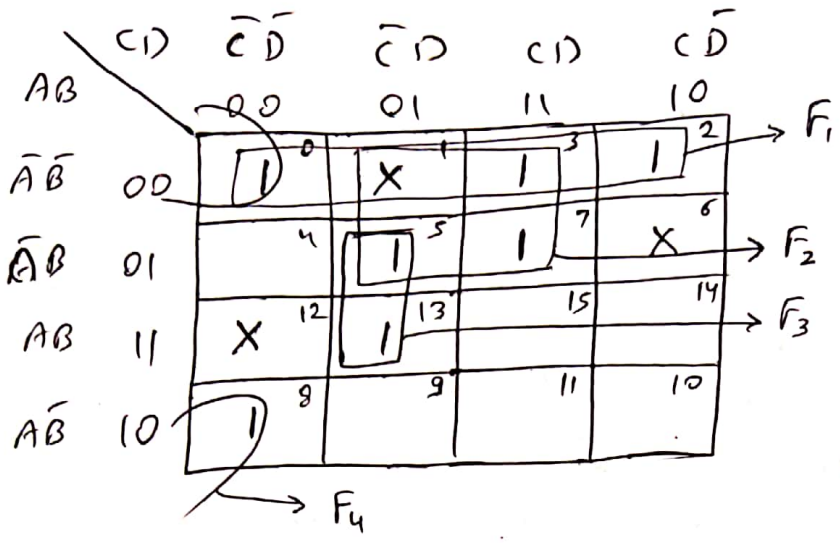
(b) $Y = (A+C)(A+\bar{D})(A+B+\bar{C})$ using NOR gates

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

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



Q. 5 (b) $F(A,B,C,D) = \sum(0,2,3,5,7,8,13) + \sum d(1,6,12)$



$F_1 = \bar{A}\bar{B}$

$F_2 = \bar{A}D$

$F_3 = B\bar{C}D$

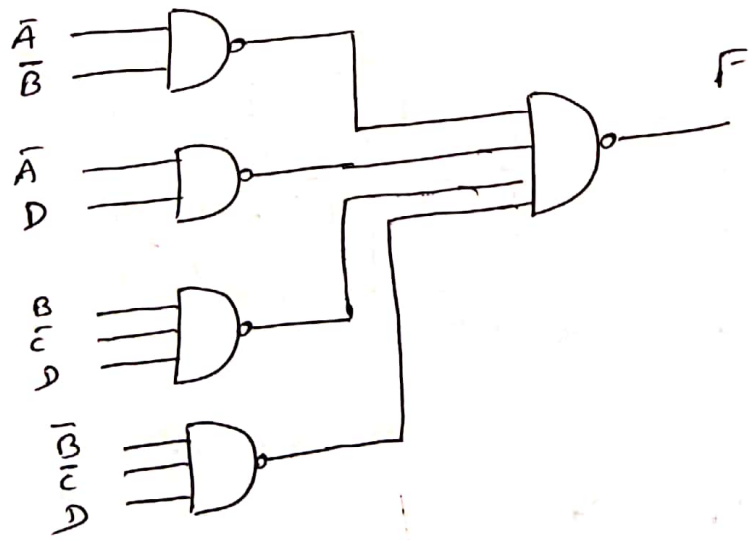
$F_4 = \bar{B}\bar{C}\bar{D}$

$F = F_1 + F_2 + F_3 + F_4$

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

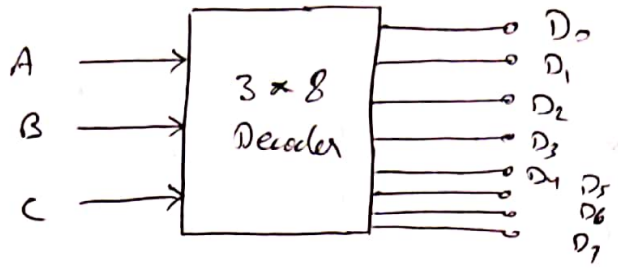
$= \overline{\bar{A}\bar{B} + \bar{A}D + B\bar{C}D + \bar{B}\bar{C}\bar{D}}$

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



Q.6 (a) (i) **3 to 8 Decoder**

Block diagram of 3 to 8 decoder is given as-



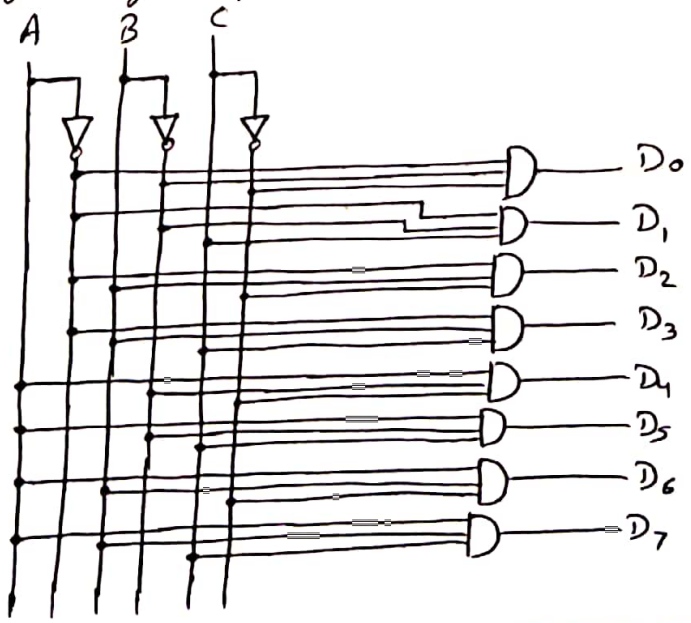
Here, in 3 to 8 decoder, we have-

- No. of inputs = 3 with variables A, B and C.
- No. of possible input combinations = $2^3 = 8$
- No. of outputs = 8 and indicated as $D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7$.

Truth Table for 3 to 8 Decoder is given as-

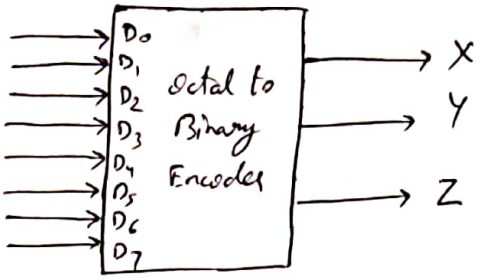
Inputs			Outputs								Minterms
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
0	0	0	1	0	0	0	0	0	0	0	$D_0 = \bar{A}\bar{B}\bar{C}$
0	0	1	0	1	0	0	0	0	0	0	$D_1 = \bar{A}\bar{B}C$
0	1	0	0	0	1	0	0	0	0	0	$D_2 = \bar{A}B\bar{C}$
0	1	1	0	0	0	1	0	0	0	0	$D_3 = \bar{A}BC$
1	0	0	0	0	0	0	1	0	0	0	$D_4 = A\bar{B}\bar{C}$
1	0	1	0	0	0	0	0	1	0	0	$D_5 = A\bar{B}C$
1	1	0	0	0	0	0	0	0	1	0	$D_6 = AB\bar{C}$
1	1	1	0	0	0	0	0	0	0	1	$D_7 = ABC$

Logic Diagram for 3 × 8 decoder is shown below :-



(ii) Octal to Binary Encoder

Block Diagram of Octal to Binary Encoda is given as-



In Octal to Binary encoder, we have-

- No. of inputs = 8 with variables $D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7$.
- No. of outputs = 3 and indicated as X, Y, Z.
- Only one input has one value at any given time.
- Each input corresponds to each octal digit and output generates corresponding Binary code.

Truth Table for Octal to Binary Encoder is given as -

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

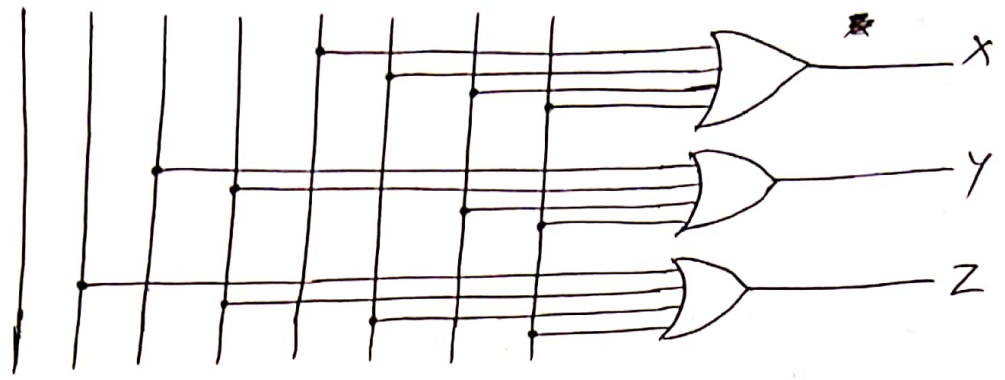
From the truth table, we get -

$$X = D_4 + D_5 + D_6 + D_7$$

$$Y = D_2 + D_3 + D_6 + D_7$$

$$Z = D_1 + D_3 + D_5 + D_7$$

Logic Diagram for Octal to Binary Encoder is shown below:-



8.6(b) 2-Bit Magnitude Comparator

Let the 2-bit numbers be $A = A_1 A_0$ and $B = B_1 B_0$.

Case 1: For $A > B$

If $A_1 = 1$ and $B_1 = 0$, then $A > B$

OR

If A_1 and B_1 coincide, and $A_0 = 1$ and $B_0 = 0$, then $A > B$

∴ The logical expression for $A > B$ is given as-

$$G = A_1 \bar{B_1} + (A_1 \odot B_1) A_0 \bar{B_0}$$

Case 2: For $A < B$

If $A_1 = 0$ and $B_1 = 1$, then $A < B$

OR

If A_1 and B_1 coincide, and $A_0 = 0$ and $B_0 = 1$, then $A < B$.

∴ The logical expression for $A < B$ is given as-

$$L = \bar{A_1} B_1 + (A_1 \odot B_1) \bar{A_0} B_0.$$

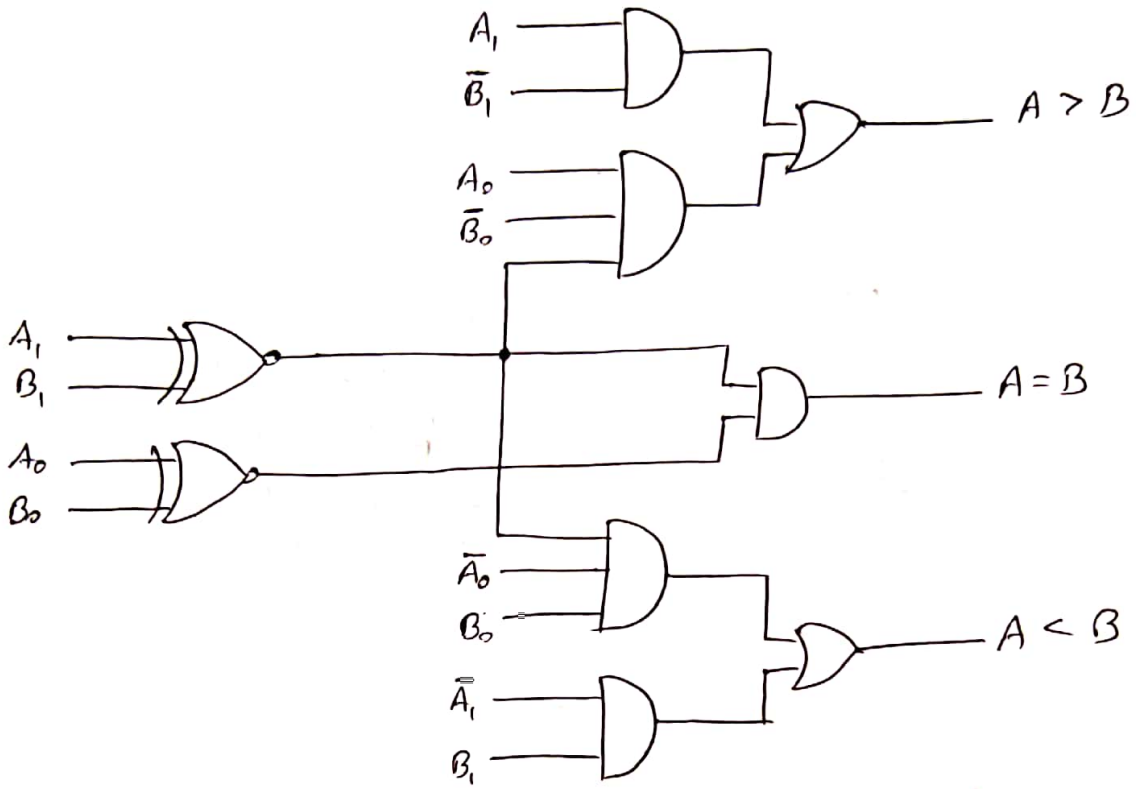
Case 3: For $A = B$

If A_1 and B_1 coincide, and A_0 and B_0 coincide, then $A = B$.

∴ The logical expression for $A = B$ is given as-

$$E = (A_1 \odot B_1).(A_0 \odot B_0)$$

The logic diagram for 2-bit comparator circuit is shown below :-



Q.7 (a) $F(A, B, C, D) = \sum m(2, 5, 8, 9, 10, 14, 15)$

(i) **Using 8:1 MUX**

Consider 3 data select lines as B, C, D and one data input as A.

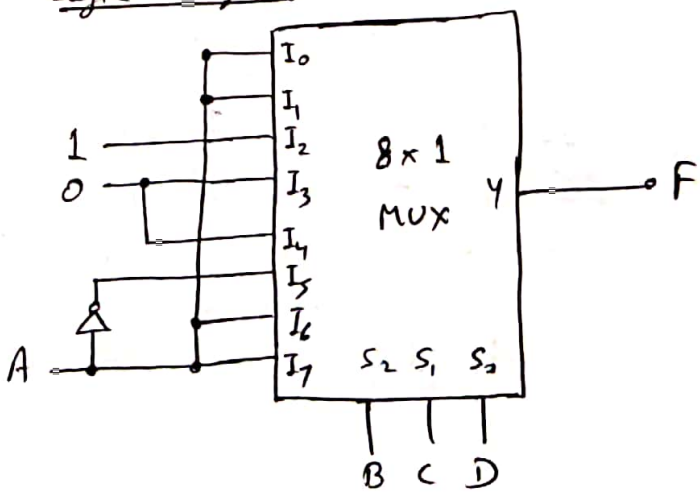
Truth Table

Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	0	0	1
3	0	0	1	0
4	0	1	0	0
5	0	1	1	1
6	0	1	0	0
7	0	1	1	0
8	1	0	0	1
9	1	0	1	1
10	1	0	0	1
11	1	0	1	0
12	1	1	0	0
13	1	1	1	0
14	1	1	0	1
15	1	1	1	1

Implementation Table

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	(2)	3	4	(5)	6	7
A	(8)	(9)	(10)	11	12	13	(14)	(15)
	A	A	1	0	0	A'	A	A

Logic Diagram



(i) Using 4:1 MUX (Method 1)

Considering 2 data select lines as C and D, and 2 data inputs as A and B.

Implementation Table

	I_0	I_1	I_2	I_3
$A'B'$	0	1	(2)	3
$A'B$	4	(5)	6	7
AB'	(8)	(9)	(10)	11
AB	12	13	(14)	(15)

From the table, we get

$I_0 = AB'$

$I_1 = A'B + AB' = A \oplus B$

$I_2 = A'B' + AB' + AB$

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

$= A'B' + A$

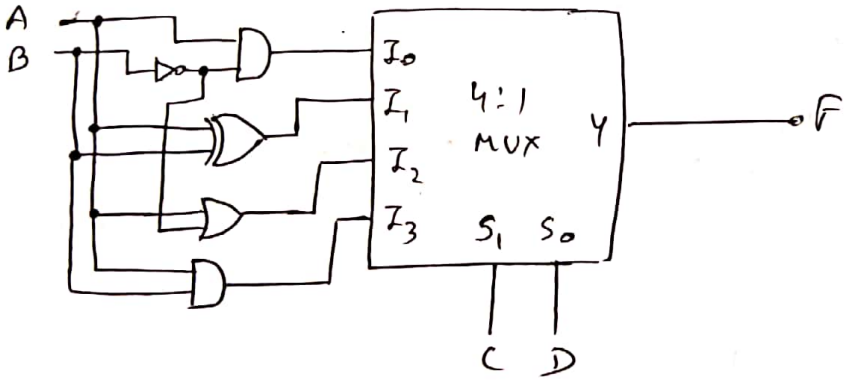
$= A + A'B'$

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

$= A + B'$

$I_3 = AB$

Logic Diagram



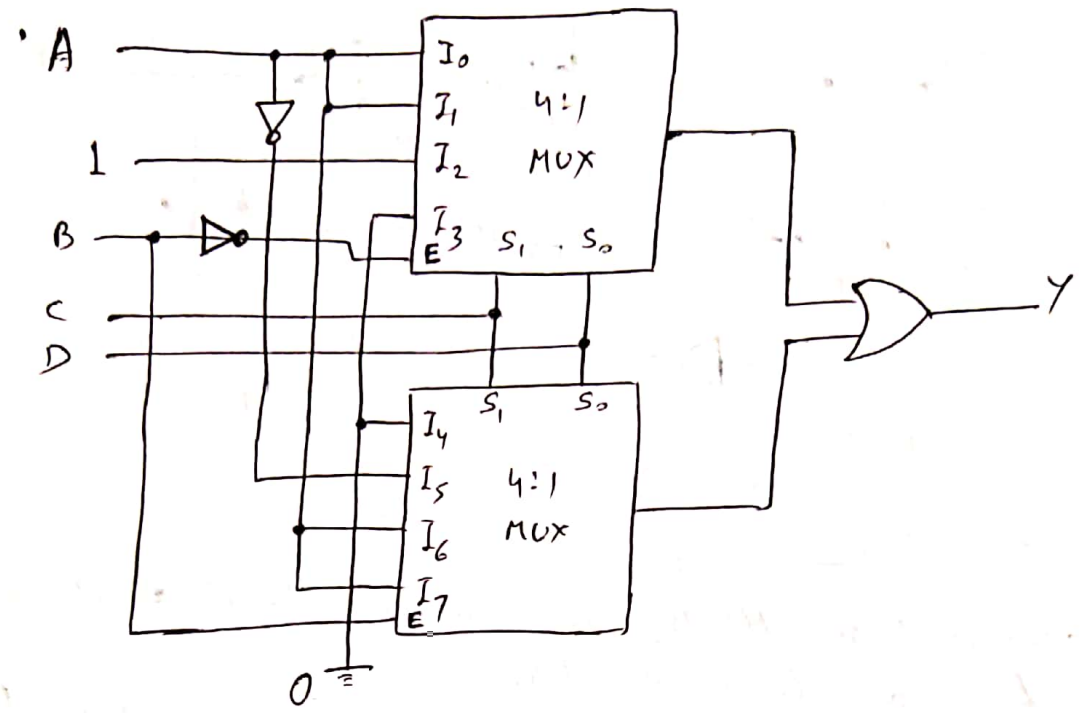
Method 2

Considering 2 data select lines as C and D, one enable input as B and one data input as A.

Implementation Table

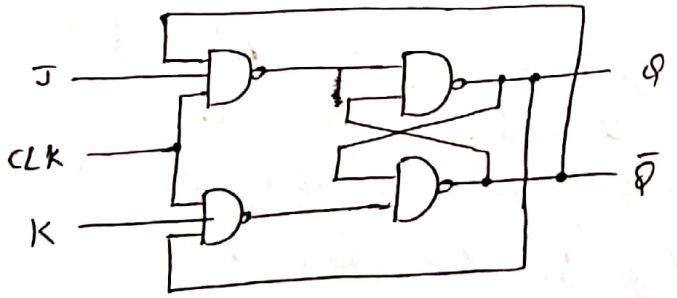
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	0	1	(2)	3	4	(5)	6	7
A	(8)	(9)	(10)	11	12	13	(14)	(15)
	A	A	1	0	0	A'	A	A

Logic Diagram



Q. 8 (a) Race Around Condition in JK Flip-Flop

The logic diagram of JK flip-flop along with Truth Table is given as-



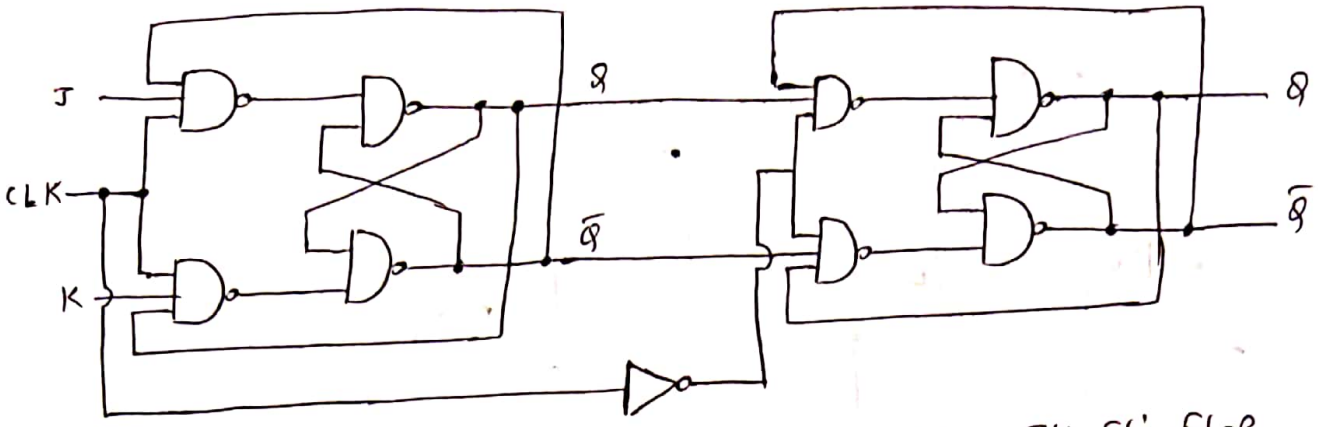
Logic Diagram of Flip-flop

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Truth Table

- For JK flip flop, if J = K = 1, and if CLK = 1 for a long period of time, then output Q will ~~change~~ toggle as long as CLK remains high which makes the output unstable or uncertain.
- This is called a race-around condition in JK flip-flop.
- The circuit used to overcome race around conditions is called Master-slave JK flip-flop by making the CLK=1 for very less duration.

The logic diagram of Master Slave Flip-flop is shown below :-



Master JK Flip Flop

Slave JK Flip Flop

Working of a Master Slave Flip-Flop

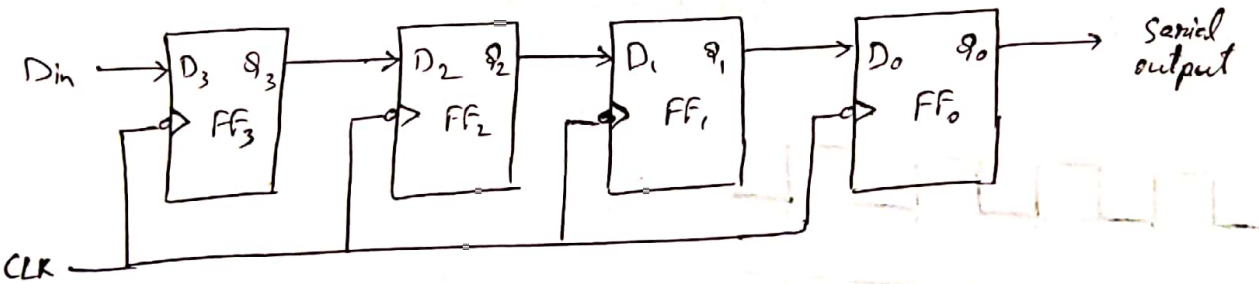
- When the clock pulse goes HIGH, the slave is isolated as its clock pulse goes LOW due to an inverter.
- When CLK goes back to '0', information is transmitted from the master flip-flop to slave flip flop and output is obtained.
- As the master flip-flop is ~~triggered~~ positive triggered, it responds first and the slave later.
- The master goes to the ~~input~~ 'K' input of the slave when both inputs $J=0$ and $K=1$, and also $Q'=1$. In this case the slave copies the master as the clock forces the slave to RESET.
- The master goes to the 'J' input of the slave when both inputs $J=1$ and $K=0$, and also $Q=1$. In this case, the clock is SET due to the negative transition of the clock.
- There is a state of TOGGLE when both inputs $J=1$ and $K=1$. on the negative transition of the clock, slave toggles, and on the positive transition of the clock, master toggles.
- Both the flip-flops are disabled when both $J=0$ and $K=0$, and Q is unchanged.

Q.8(b) Shift Register

As we know, one flip-flop can store 1-bit of information. In order to store multiple bits of information, we require multiple flip-flops. The group of flip-flops which are used to store the binary data is known as Register. And if the register is capable of shifting bits either towards right hand side or towards left hand side is known as Shift Register. An 'N' bit shift register contains 'N' flip-flops.

Serial in - Serial - out (SISO) shift Register

The shift register, which allows serial input and produces serial output is known as SISO shift register. The block diagram of 4-bit SISO shift register is shown below:-



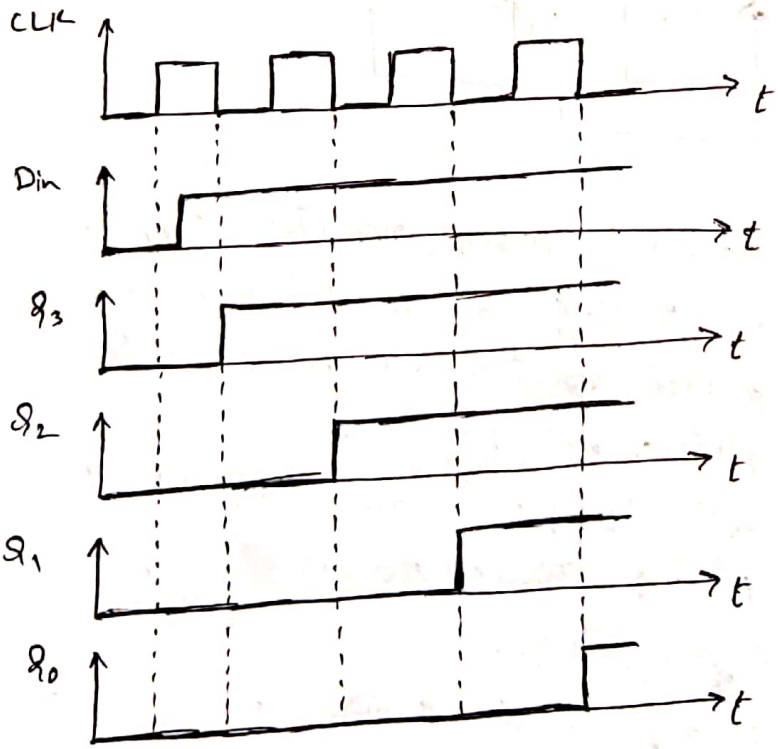
To understand how data is stored in a shift register, suppose '1111' needs to be stored. Initially, as the device is in RESET mode, the output of each register will be Low. Hence, the output of all the 4 registers will be '0'.

- Starting with LSB of the data '1111', assign $D_{in} = 1$. Here, $D_3 = 1$ which cause Q_3 to be 1. Hence, the overall output will be '1000' for first falling edge.
- When another data input bit is provided at D_3 i.e. $D_3 = 1$ again. This will cause $Q_3 = 1$ again and as Q_3 is provided as input to D_2, this will cause $Q_2 = 1$, while rest of all other outputs will be '0'. Hence, the overall output will be '1100' for second falling edge.

- Similarly, for third falling edge, the overall output will be '1110' and for fourth falling edge, the overall output will be '1111'.
- Thus, in this way shift register stores '1111'.
- The truth table for the 4-bit SISO shift register is shown below:-

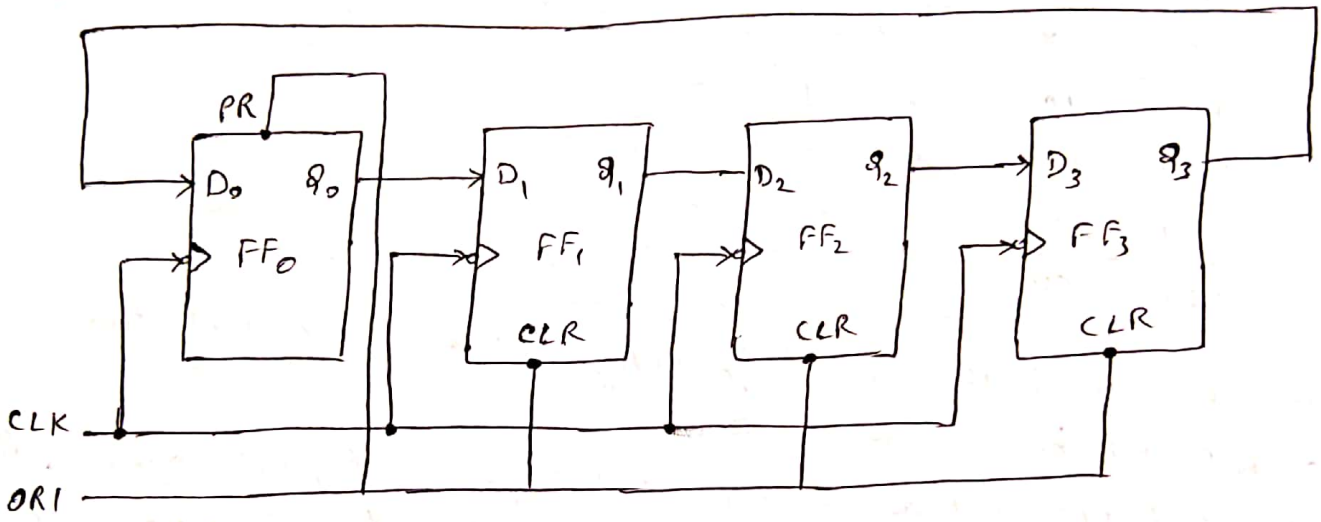
CLK	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1st Falling edge	1	0	0	0
2nd Falling edge	1	1	0	0
3rd Falling edge	1	1	1	0
4th Falling edge	1	1	1	1

- The waveform representation for 4-bit SISO shift register is shown as-



Q.9(a) 4-bit Ring Counter using D-Flip Flops

A ring counter is also known as SISO shift register counter, where the output of the flip-flop is connected to the input of the flip-flop which acts as a ring counter. The designing of the ring counter can be done by using 4 D-Flip-flops with a common clock signal and overriding input can be connected to pre-set and clear. The block diagram of 4-bit Ring counter using D-flip flops is shown below:-



From the above diagram,

- The number of states used is 4, where no. of states = No. of flip-flops used.

Pre-set or Clear: The main function of this is that if the input clock signal changes, then the output value is also changed.

Considering a condition where pre-set = '0 0 0 0', then the outputs obtained at each flip flop is as follows -

For FF_0, the output Q_0 is 1, where in other flip flops FF_1, FF_2 and FF_3, the outputs obtained at $Q_1 = Q_2 = Q_3 = 0$.

- Pre-set (PR) and clear (CLR) :- PR and CLR are used as ORI.

When PR = 0, then the output is 1.

When CLR = 0, then the output is 0.

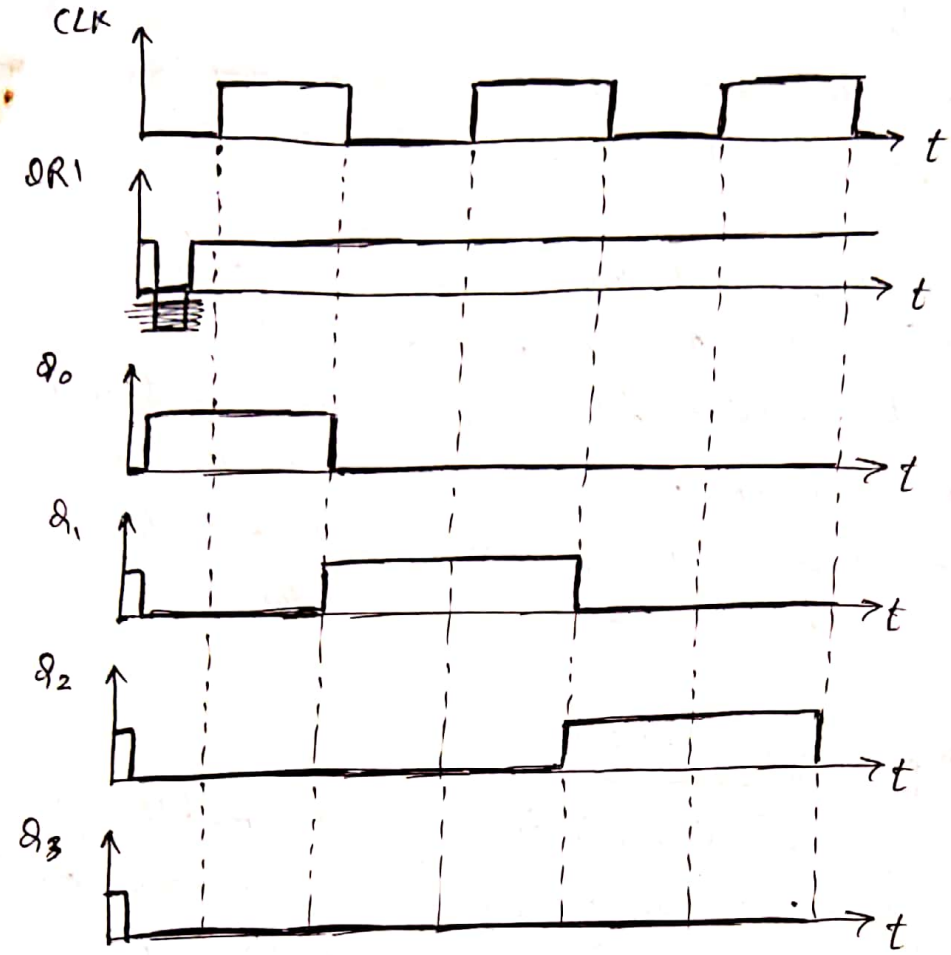
Both PR and CLR are active low signal that always works in value '0'.

Working

- Here, ORI is connected to Pre-set (PR) in FF_0 and it is connected to clear (CLR) in FF_1, FF_2 and FF_3. Thus, the output $Q_0 = 1$ and the rest of the flip flops generate output $Q_1 = Q_2 = Q_3 = 0$.
- The output $Q = 1$ at FF_0 is known as Pre-set 1 which is used to form the ring in the Ring Counter.

ORI	CLK	Q_0	Q_1	Q_2	Q_3	
LOW	X	1	0	0	0	→ PRESETED 1
HIGH	LOW	0	1	0	0	
HIGH	LOW	0	0	1	0	
HIGH	LOW	0	0	0	1	
HIGH	LOW	1	0	0	0	

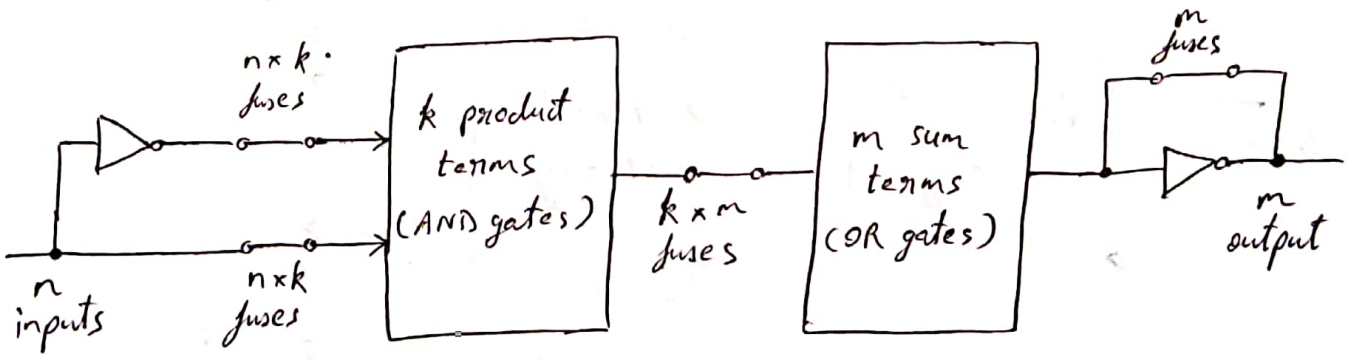
- This Preseted 1 is generated by making ORI Low and that time clock (CLK) becomes don't care.
- After that ORI is made to HIGH and apply clock pulse signal as the CLK is negative edge triggered. After that, at each clock pulse, the preseted 1 is shifted to the next flip-flop and thus forms a Ring.
- From the above table, we can say that there are 4-states in a 4-bit Ring Counter.
- The timing diagram is shown below :-



Q.9(b)

(i) Programmable Logic Array (PLA)

A block diagram of the PLA is shown below -



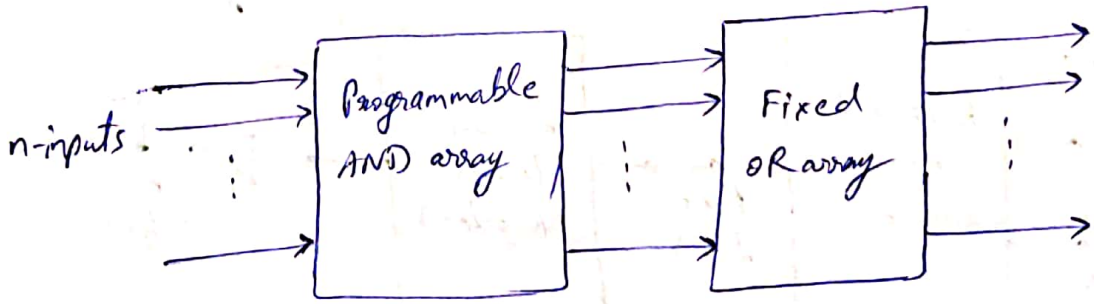
- PLA consists of 'n' inputs, 'm' outputs, 'k' product terms and 'm' sum terms. The product terms consitute a group of 'k' AND gates and the sum terms constitute a group of 'm' OR gates.

- Fuses are inserted between all 'n' inputs and their complement values to each of the AND gates. Fuses are also provided between the outputs of the AND gates and the inputs of the OR gates.
- Another set of fuses in the output inverts allows the output function to be generated either in the AND-OR form or in the AND-OR-INVERT form. With the inverter fuse in place, the inverter is by passed, giving an AND-OR implementation. With the fuse blown, the inverter becomes part of the circuit and the function is implemented in the AND-OR-INVERT form.

(ii) Programmable Array Logic (PAL)

The programmable array logic (PAL) is a programmable logic device (PLD) with a fixed OR array and a programmable AND array. Because only the AND gates are programmable, the PAL is easier to program, but is not flexible as the PLA.

The block diagram of PAL is shown below.



Each input has a buffer gate and an inverter gate. Buffer gate is added to the normal input, NOT gate is added to the inverted or complemented input.

8.7 (b)

ROM, which stands for Read Only Memory, stores the instructions required to start a computer. It is a non-volatile memory. Therefore, the data remains even when there is no continuous power flow. In other words, the data is permanent. PROM, EPROM and EEPROM are some types of ROM.

Difference b/w PROM, EPROM and EEPROM are as follows :-

S No.	PROM	EPROM	EEPROM
1.	Stands for programmable read only memory.	Stands for Erasable programmable read only memory.	Stands for Electrically Erasable read only memory.
2.	It is a ROM that can be modified only once by a user.	It is a programmable ROM that can be erased & reused.	It is a user-modifiable ROM that can be erased and re-programmed repeatedly through a normal electrical voltage.
3.	PROM can be reprogrammed only once	Can be reprogrammed using ultraviolet (UV) light.	Can be reprogrammed using electrical charge.
4.	Example: CD-R	Example: CD (RW)	Example: Pen-Drive