MUMBAI UNIVERSITY

SEMESTER 3

Digital Logic Design And Analysis

DECEMBER 2017-CBCS

Q.1(a) Convert $(1762.46)_{10}$ into octal,binary and hexadecimal. [3]

Ans :

(i) $(1762.46)_{10} = (X)_8$ i.e Decimal to octal

		Remainder
8	1762	2
	220	4
	27	3
	3	3

$\therefore (1762)_{10} = (3342)_8$

Fractional part :

0.	46	
	8	
3	68	
	8	
5	44	
	8	
3	52	

$\therefore (0.42)_{10} = (0.353)_8$

$\therefore (1762.46)_{10} = (3342.353)_8$

(ii) $(1762.46)_{10} = (Y)_2$ i.e Decimal to binary

2	1762	0
	881	1
	440	0
	220	0
	110	0
	55	1
	27	1
	13	1
	6	0
	3	1
	1	1

$\therefore (1762)_{10} = (11011100010)_8$

Fractional part :

0.	46
	2
0	92
	2
1	84
	2
1	68

$\therefore (0.42)_{10} = (0.011)_2$

$\therefore (1762.46)_{10} = (11011100010.011)_2$

(iii) $(1762.46)_{10} = (Z)_{16}$ i.e Decimal to hexadecimal

		Remainder
16	1762	2
	110	E
	6	6

$\therefore (1762)_{10} = (6E2)_{16}$

Fractional part :

0.	46
	16
7	36
	16
5	76
	16
C	16

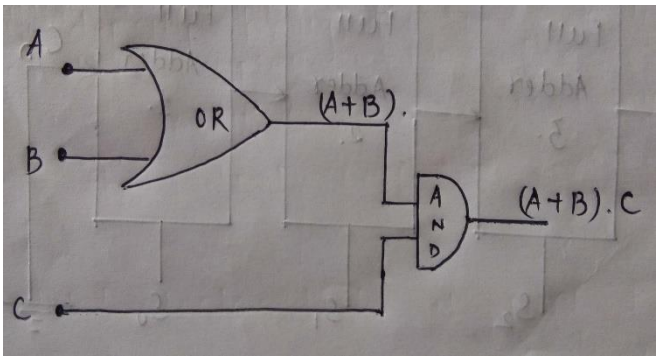
$\therefore (0.42)_{10} = (0.75C)_2$

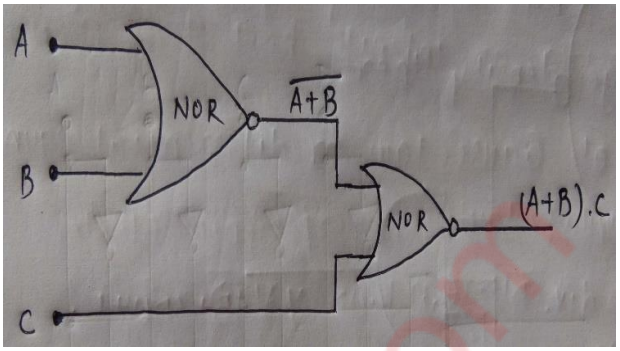
$\therefore (1762.46)_{10} = (6E2.75C)_2$

(b) Prove OR-AND configuration is equivalent to NOR-NOR configuration. **[3]**

Ans :

The expression for OR gate output is y=A+B.
The expression for AND gate output is y=A.B
The expression for NOR gate output is y=$\overline{A+B}$





From this we can prove that OR-AND configuration is same as NOR-NOR Configurations.

(c) Perform substration using 16's complement. **[4]**

(i) $(\mathbf{CB}10)_{16} - (971)_{16}$

(ii) $(426)_{16} - (\mathbf{DBA})_{16}$

Ans : (i) $(971)_{16}$ 15's complement is :

$(\mathbf{CB}10)_{16} - (971)_{16} = (\mathbf{CB}10)_{16} + (971)_{16} = C19F$

$(\mathbf{CB}10)_{16} - (971)_{16}$ = C19F

(ii) $(\mathbf{DBA})_{16}$ 15's complement is : 245

426 + 245 = −994 IN HEXA DECIMAL

$(426)_{16} - (\mathbf{DBA})_{16}$= -994

(d) Find 8's complement of following numbers.

(i) $(\mathbf{27})_8$ **(i)** $(\mathbf{321})_8$ **[2]**

Ans : 8's complement is 1 + 7's complement of number.

To calculate 7's complement substract each digit of number from 7

(i) $(\mathbf{27})_8$= (77-27) + 1 = 50 + 1 =51

$\therefore (\mathbf{27})_8 = (\mathbf{51})_8$

(ii) $\therefore (\mathbf{321})_8 = 777 - 321 = 456 + 1 = 457$

$\therefore (\mathbf{321})_8 = (\mathbf{457})_8$

(e) Perform following substraction $(52)_{10} - (65)_{10}$ using 2's complement Method. **[2]**

Ans : $(52)_{10} = (110100)_2$

$(65)_{10} = (1000001)_2$

2's complement of $(65)_{10}$ is (0111110)+1=0111111

$\therefore (52)_{10} - (65)_{10} = (110100)_2 - (0111111)_2 = -1101$

We can represent -1101 in decimal as -13 .

$$\therefore \quad (52)_{10} - (65)_{10} = (-13)_{10}$$

(f) Write hamming code for 1010. **[2]**

Ans : Given code W=(1 0 1 0)

Hamming code is given by ,

X=W.G

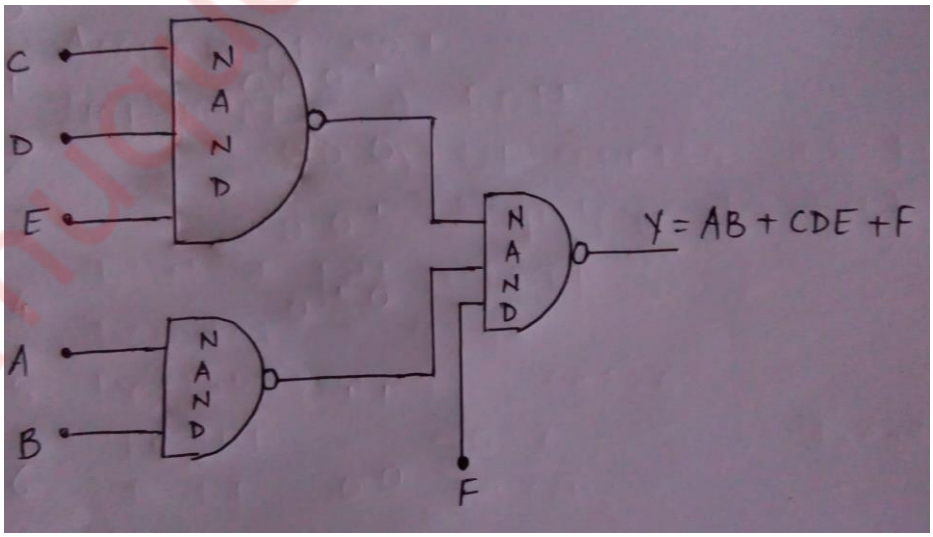
$$X=(\,1 \quad 0 \quad 1 \quad 0\,) \cdot \begin{bmatrix} 1 & 0 & 00 & 1 & 10 \\ 0 & 1 & 00 & 1 & 01 \\ 0 & 0 & 10 & 0 & 1\ 1 \\ 0 & 0 & 01 & 1 & 1\ 1 \end{bmatrix}$$

X = (1 0 1 0 0 0 1)

(g) Implement the following Boolean equation using NAND gates only.

Y= AB+CDE+F **[2]**

Ans : In this y can be implemented using 3 input nand and 2 input nand gates.



(h) Explain the term prime implicant. **[2]**

Ans :

i) A group of related 1's (implicant) on a Karnaugh map which is not subsumed by any other implicant in the same map. Equivalently (in terms of Boolean algebra), a product term which is a "minimal" implicant in the sense that removing any of its literals will yield a product term which is not an implicant (on a Karnaugh map it would appear "maximal").

ii) A group of related 0's (implicant) on a Karnaugh map which is not subsumed by any other implicant (of 0's) in the same map.

Q.2(a) Design a 4-bit ripple adder. **[10]**

Ans :

(i) Multiple full adder circuits can be cascaded in parallel to add an N-bit number. For an N- bit A parallel A adder, there must be N number of full adder circuits.

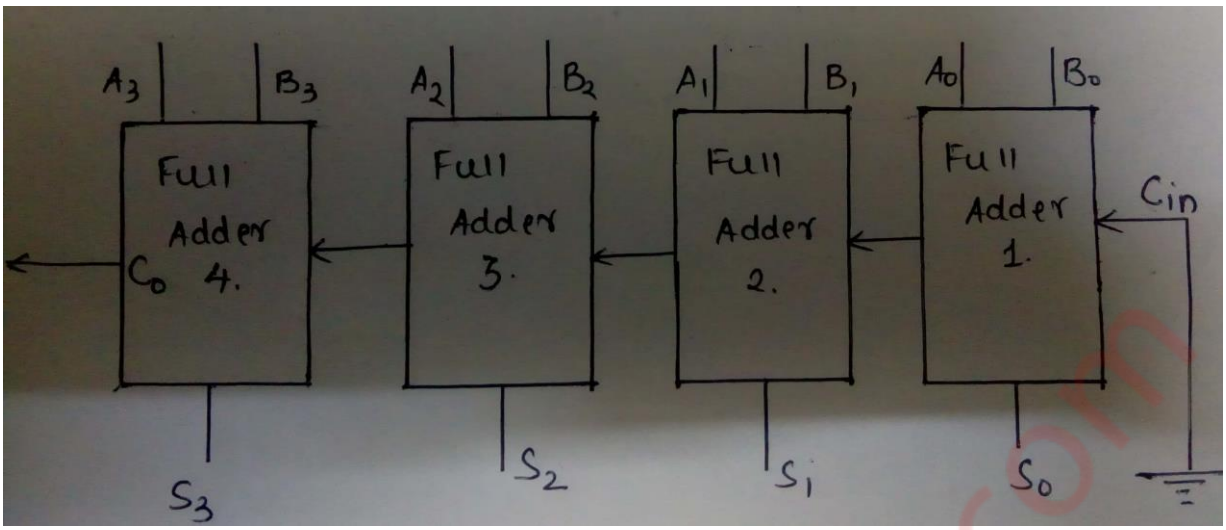
(ii) A ripple carry adder is a logic circuit in which the carry-out of each full adder is the carry in of the succeeding next most significant full adder. It is called a ripple carry adder because each carry bit gets rippled into the next Â stage.

(iii)In a ripple carry adder the sum and carry out bits of any half adder stage is not valid until the carry in of that stage occurs.

(iv)Propagation delays inside the logic circuitry is the reason behind this. Propagation delay is time elapsed between the application of an input and occurance of the corresponding output.

(v)Consider a NOT gate, When the input is “0” the output will be “1” and vice versa. The time taken for the NOT gate’s output to become “0” after the application of logic “1” to the NOT gate’s input is the propagation delay here.

(v)Similarly the carry propagation delay is the time elapsed between the application of the carry in signal and the occurance of the carry out (Cout) signal.



(vi) Sum out S0 and carry out Cout of the Full Adder 1 is valid only after the propagation delay of Full Adder 1.

(vii) In the same way, Sum out S3 of the Full Adder 4 is valid only after the joint propagation delays of Full Adder 1 to Full Adder 4. In simple words, the final result of the ripple carry adder is valid only after the joint propogation delays of all full adder circuits inside it.

(b) Obtain the minimal expression using QuineMc-Cluskey method.

F(A,B,C,D)=∑m(1,5,6,12,13,14) + **d**(2,4) **[10]**

Ans :

Step 1: Write down the binary equivalent of the given expressions (including the don't care):

	A	B	C	D
1	0	0	0	1
2	0	0	1	0
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0

Step 2:
Create the following table in which the binary number with equal number of 1's are grouped together. For e.g. the first group contains binary numbers with only one number of 1's, second group contains binary numbers with two 1's and so on.

GROUP	MINTERM	BINARY REPRESENTATION				
		A	B	C	D	
1	m1	0	0	0	1	√
	d2	0	0	1	0	√
	d4	0	1	0	0	√
2	m5	0	1	0	1	√
	m6	0	1	1	0	√
	m12	1	1	0	0	√
3	m13	1	1	0	1	√
	m14	1	1	1	0	√

Step 3: In the next step compare every element of group 1 with elements in group 2, elements of group 2 with 3, etc. In general compare elements of nth group with elements of n+1th group. If only 1 element is changing in the comparison the mark it with an underscore. Put a tick mark against both the elements if the elements are present in the comparison. For e.g. in m1 and m5 the A, C, D bits are same but only B bit changes, so an underscore is put against it.

GROUP	MATCHED PAIRS	BINARY REPRESENTATION				
		A	B	C	D	
1	m1 m5	0	_	0	1	
	d2 m6	0	_	1	0	
	d4 m5	0	1	0	_	√

	d4 m6	0	1	_	0	√
	d4 m12	_	1	0	0	√
2	m5 m13	_	1	0	1	√
	m6 m14	_	1	1	0	√
	m12 m13	1	1	0	_	√
	m12 m14	1	1	_	0	√

Step 4: In this step again match the values of nth group with n+1th group and mark tick against the ones that gets matched, followed by underscore to the single bits that change. Since m1 m5, d2 m6 are not getting matched with any other no tick marks is put against them.

GROUP	MATCHED PAIRS	BINARY REPRESENTATION			
		A	B	C	D
1	d4 m5 m12 m13	_	1	0	_
	d4 m6 m12 m13	_	1	_	0

Step 5: Now make the last table which contains the prime implicant and the min terms involved (Prime implicants are the ones that do not get matched). Write them in their alphabet form with 0 bit represented as bars. In this step write all the numbers which were given in the question (not the don't care ones). Put a cross below the numbers which contain the minterms.

Prime Implicant	Minterms Involved	1	5	6	12	13	14
BC	4, 5, 12, 13		×		×	×	
BD	4, 6, 12, 14			×	×		×
ACD	1, 5	×	×				
ACD	2, 6			×			

If the column of the numbers contain only one cross then the prime implicant belonging to the single cross belonging row is considered. In this case the columns of 1, 13 and 14 contain only 1 crosses so the row belonging to them is considered which are BC, BD, ACD. These are the reduced answer.

Therefore the final answer is:

$$Y = BC + BD + ACD$$

$$\therefore F(A,B,C,D) = BC + BD + ACD$$

Q.3(a) Implement full adder using 8:1 multiplexer. [10]

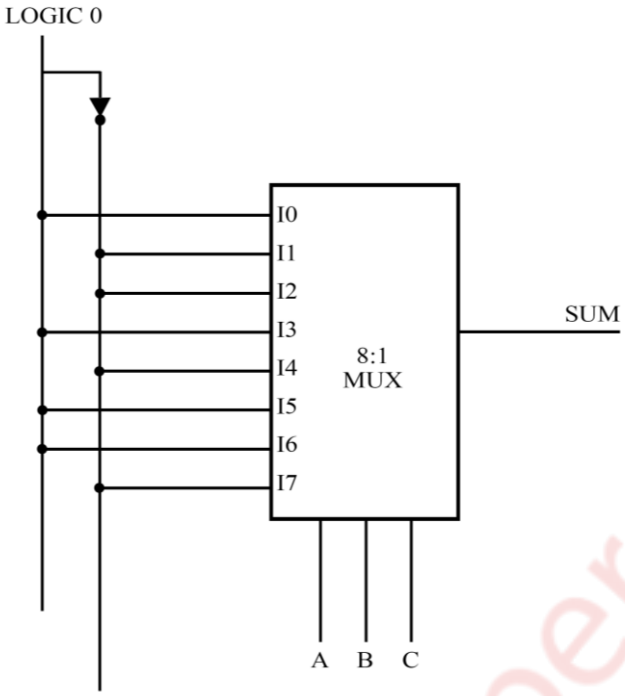
Ans :

3 bit binary adder is adder circuit which performs bit by bit addition and gives output as sum and carry .The truth table for a full adder :

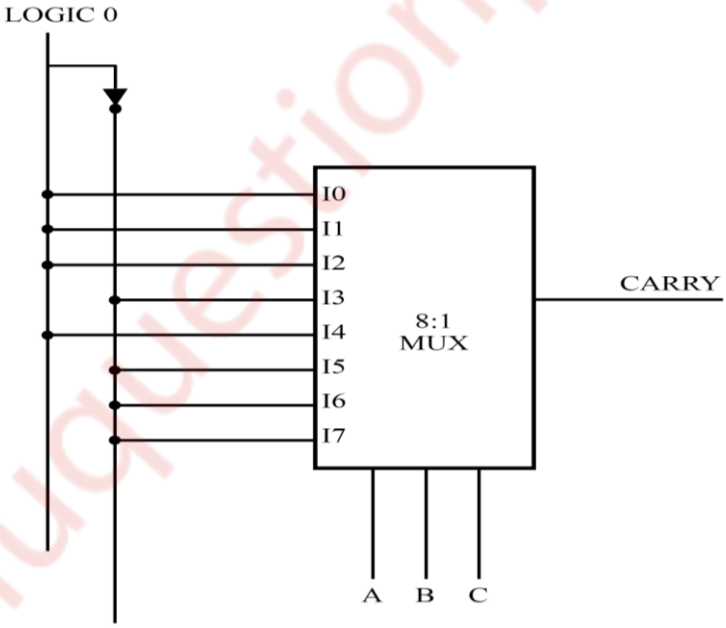
A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

We will require two 8:1 multiplexer to implement a full adder. One for sum and one for carry.

In 8:1 MUX we have 3 selection lines, assigning them A, B, C.For sum :



For carry :



(b) Implement the following functions using demultiplexer.

F1(A,B,C)=∑m(0,3,7) F2(A,B,C)=∑m(1,2,5) [5]

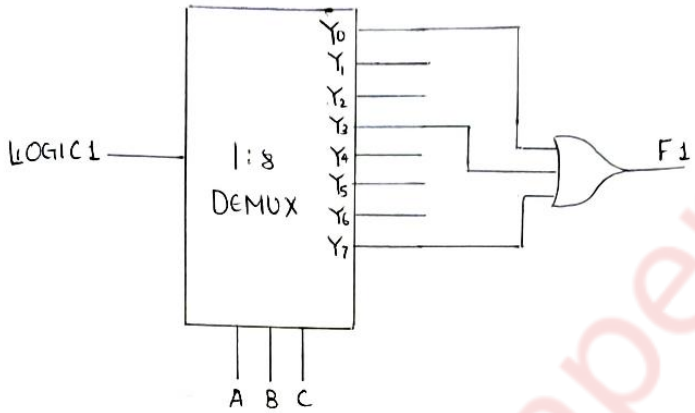
Ans :

(1) F1 (A, B, C) = Σm(0, 3, 7)

Number of variables = 3

Number of select lines = 3

Number of lines = 2^3 = 8

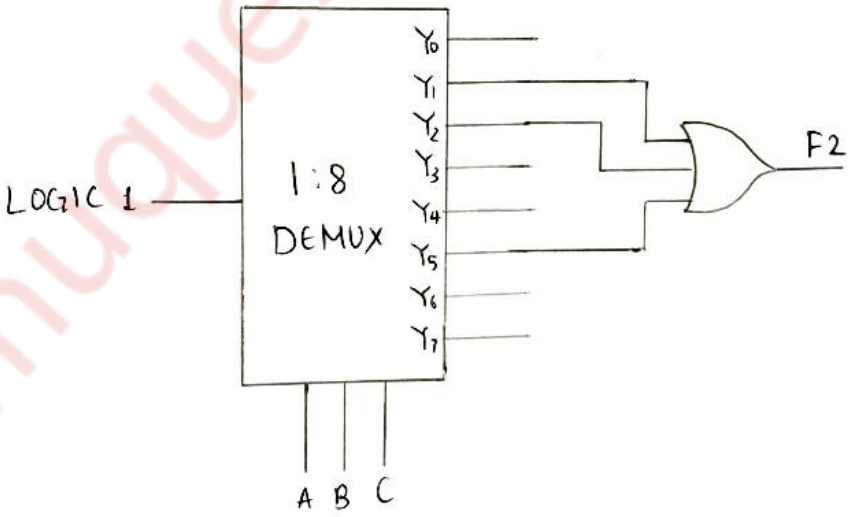


(2) F2 (A, B, C) = Σm(1, 2, 5)

Number of Variables = 3

Number of Select Lines = 3

Number of Lines = 2^3 = 8



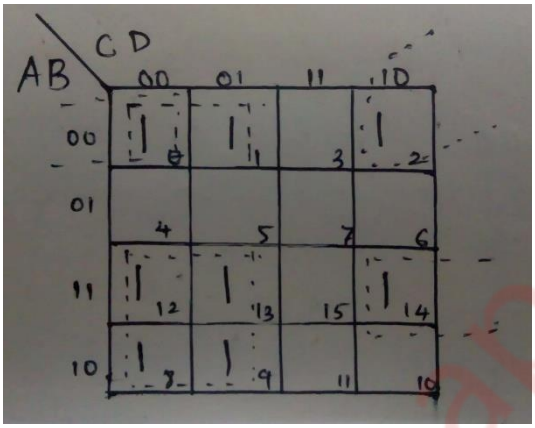
(c) Simplify F(A,B,C,D)=∏M(3,4,5,6,7,10,11,15) and implement using minimum number of gates. **[5]**

Ans :

We can write same function using sum of product way ,

F(A,B,C,D)=∏m(0,1,2,8,9,12,13,14)

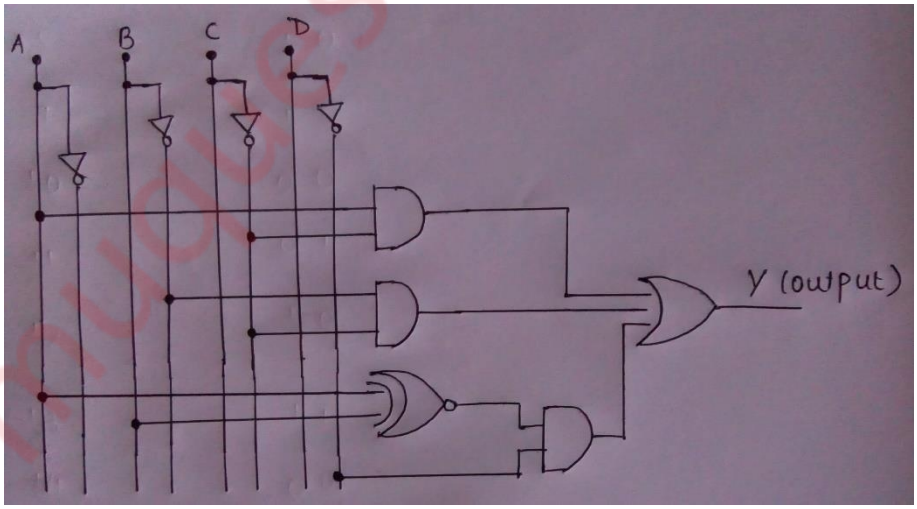
K-map is given by



The expression after grouping :

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

Circuit Diagram :



Q.4(a) Compare TTL and CMOS logic withn respect to fan in,fan out Propogation delay,power consumption,noise margin,current and Voltage parametes. [5]

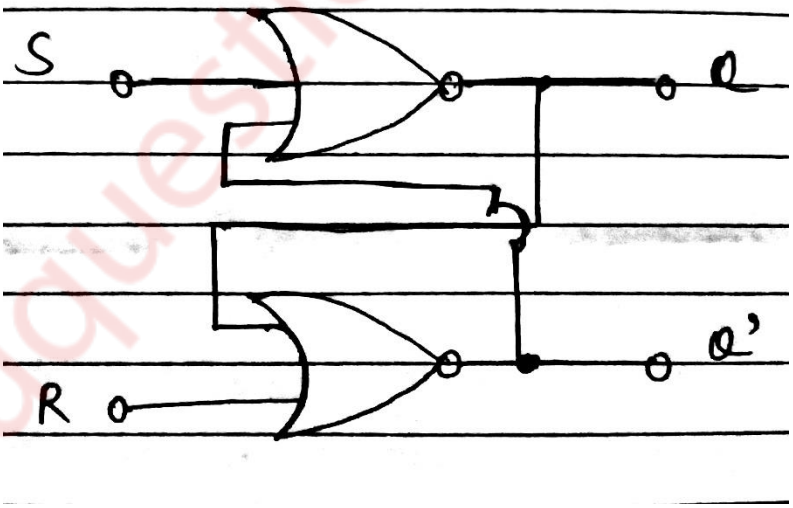
Ans :

Parameter	TTL	CMOS
Propagation Delay	1 to 200nsec	1.5 to 33nsec
Power consumption	Relatively High	Depends on Vcc
Noise Margin	0.3-0.5	0.3Vcc
Current	High (0.2-2mA)	Low (1uA)
Voltage	5V	3-18V
Fan in	High fan in	Low fan in
Fan out	Approx. 10	>50
Transistor used	BJT	FET
Type	Current controlled Current devices	Voltage controlled current device

(b) Draw the circuit of SR flip flop using two NOR gates and write the Architecture body for the same using structural modelling. [5]

Ans :

SR flip flop using two NOR gate :



SR FLIP FLOP

Architectural Body :

```
begin
PROCESS(CLOCK)
 variable tmp: std_logic;
 begin
 if(CLOCK='1' and CLOCK'EVENT) then
 if(S='0' and R='0')then
 tmp:=tmp;
 elsif(S='1' and R='1')then
 tmp:='Z';
 elsif(S='0' and R='1')then
 tmp:='0';
 else
     tmp:='1';
 end if;
 end if;
 Q <= tmp;
 QBAR <= not tmp;
end PROCESS;
end behavioral;
```

(c) Explain 1-digit BCD adder. **[10]**

Ans :

i) BCD adder adds two BCD digits and produces output as a BCD digit. A BCD or Binary Coded Decimal digit cannot be greater than 9.

ii)The two BCD digits are to be added using the rules of binary addition. If sum is less than or equal to 9 and carry is 0, then no correction is needed. The sum is correct and in true BCD form.

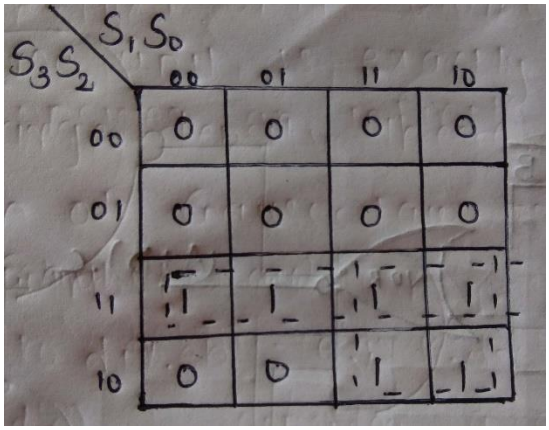
iii)But if sum is greater than 9 or carry =1, the result is wrong and correction must be done.

iv)The wrong result can be corrected adding six (0110) to it.

v)For implementing a BCD adder using a binary adder circuit IC 7483, additional combinational circuit will be required, where the Sum output S3−S0S3−S0 is checked for invalid values from 10 to 15.

vi) Truth Table :

S3	S2	S1	S0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



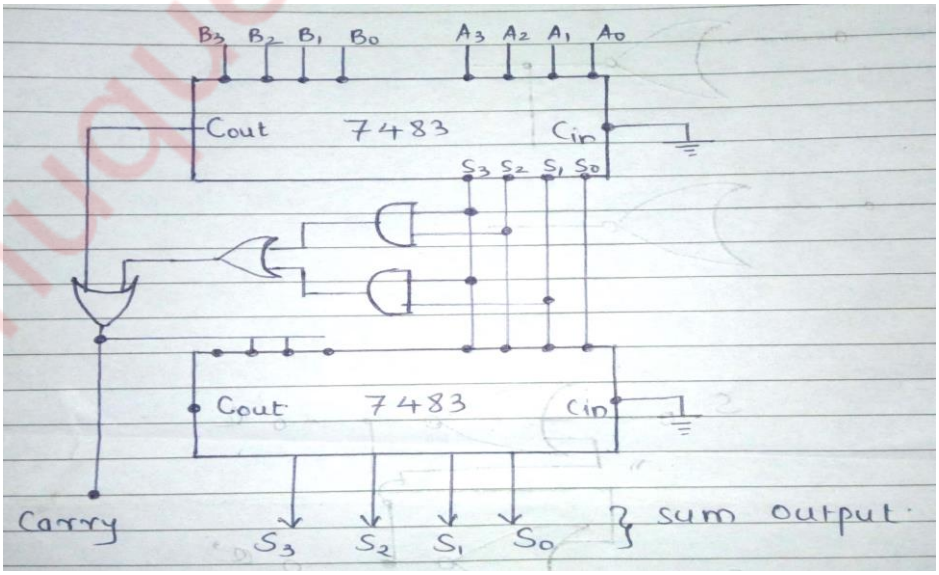
vi) The Boolean expression is, Y=S3S2+S3S1Y=S3S2+S3S1

vii)The BCD adder is shown below. The output of the combinational circuit should be 1 if Cout of adder-1 is high. Therefore Y is ORed with Cout of adder 1.

viii)The output of combinational circuit is connected to B1B2 inputs of adder-2 and B3=B1+0B3=B1+0 as they are connected to ground permanently. This makes B3B2B1B0B3B2B1B0 = 0110 if Y' = 1.

ix)The sum outputs of adder-1 are applied to A3A2A1A0A3A2A1A0 of adder-2.

x)The output of combinational circuit is to be used as final output carry and the carry output of adder-2 is to be ignored.



Q.5 (a) Convert JK flip flop to SR flip flop and D filp flop. **[10]**

*Ans: **(A) JK Flip flop to SR flip flop :***

1.Truth table of SR FF:

S	R	Q	$\bar{Q}$
0	0	X	X
0	1	0	1
1	0	1	0
1	1	-	-

2.Excitation Table of JK FF:

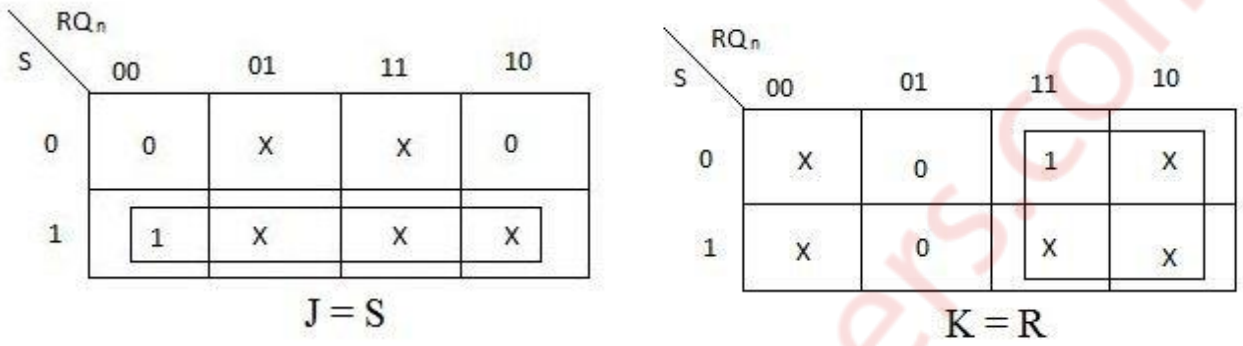
Qn	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

3.Conversion Table:

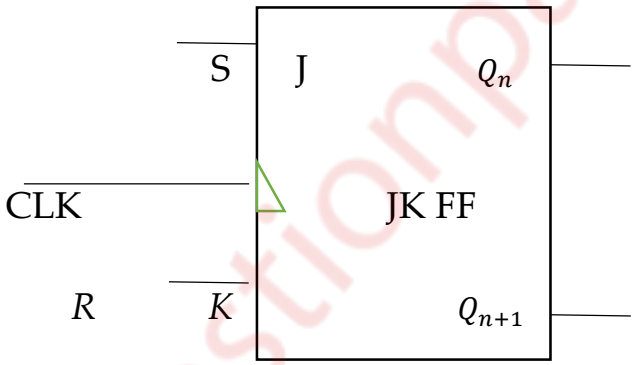
S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	X
0	1	1	1	X	0
0	0	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	1	1	1	X	0
1	0	-	-	X	X
1	1	-	-	X	X

4. *Logical expressions for the inputs :*

Simplify the logical expressions for the inputs of the given flip-flop (J and K) in terms of the inputs of the desired flip-flop (S and R) and the flip-flop's present-state, Q_n.This can be done by following any logical simplification technique like that of the K-map.



5. *Circuit Diagram :*



(B) JK TO D FF :

1.Truth Table of D flip flop :

D	Q	$\bar{Q}$
0	0	0
0	1	0
1	0	1
1	1	1

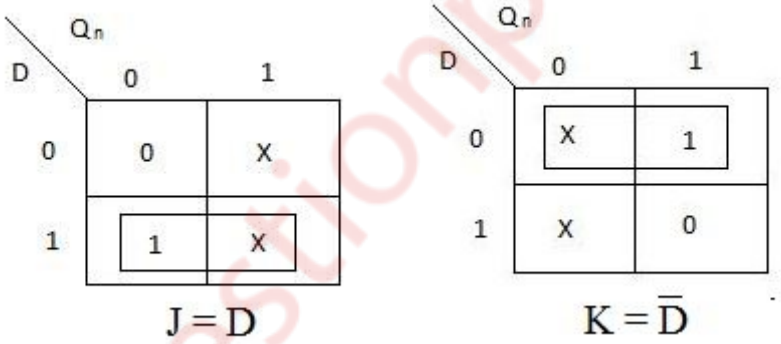
2.Excitation Table of JK FF:

Qn	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

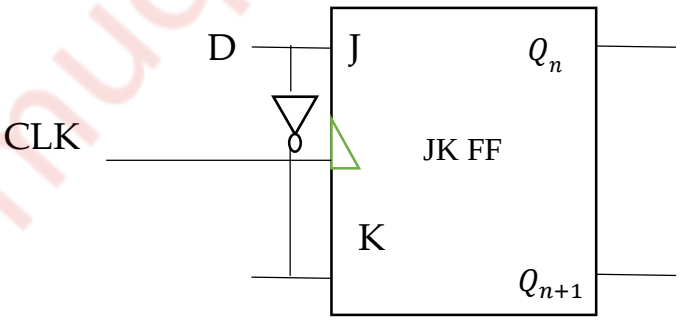
3.Conversion Table :

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

4.Use a K-map to obtain the logical expressions for the inputs J and K in terms of D and Qn.



5. Circuit Diagram:



(b) Design 3 bit synchronous counter using T flip flops. [10]

Ans : (i) A synchronous counter, in contrast to an asynchronous counter, is one whose output bits change state simultaneously, with no ripple.

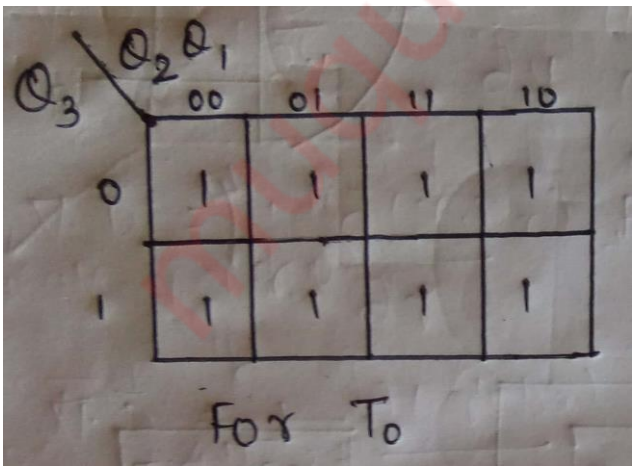
(ii) The only way we can build such a counter circuit from T flip-flops is to connect all the clock inputs together, so that each and every flip-flop receives the exact same clock pulse at the exact same time.

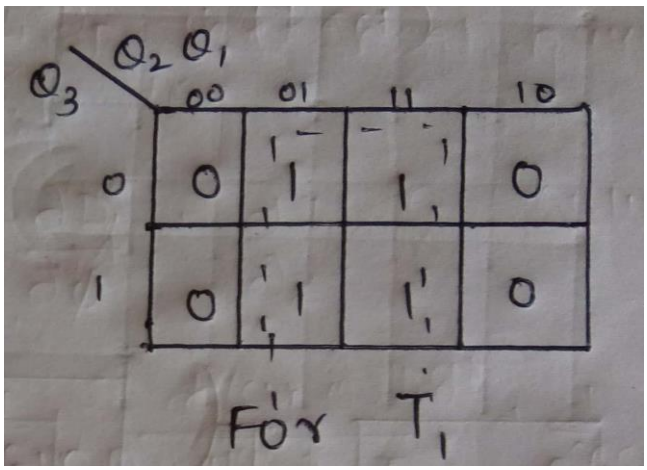
(iii) For an up counter that means it will count from value 0 to 7 for interval difference of 1.

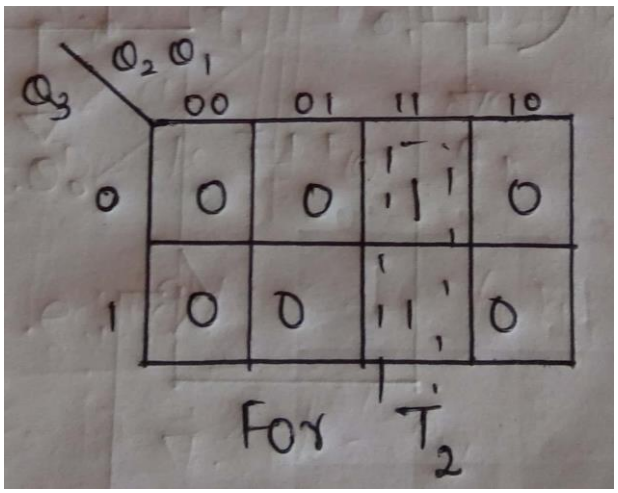
(iv) Truth table for up counter is given by ,

Present State	Next State	T2	T1	T0
000	001	0	0	1
001	010	0	1	1
010	011	0	0	1
011	100	1	1	1
100	101	0	0	1
101	110	0	1	1
110	111	0	0	1
111	000	1	1	1

(v) From this truth table :





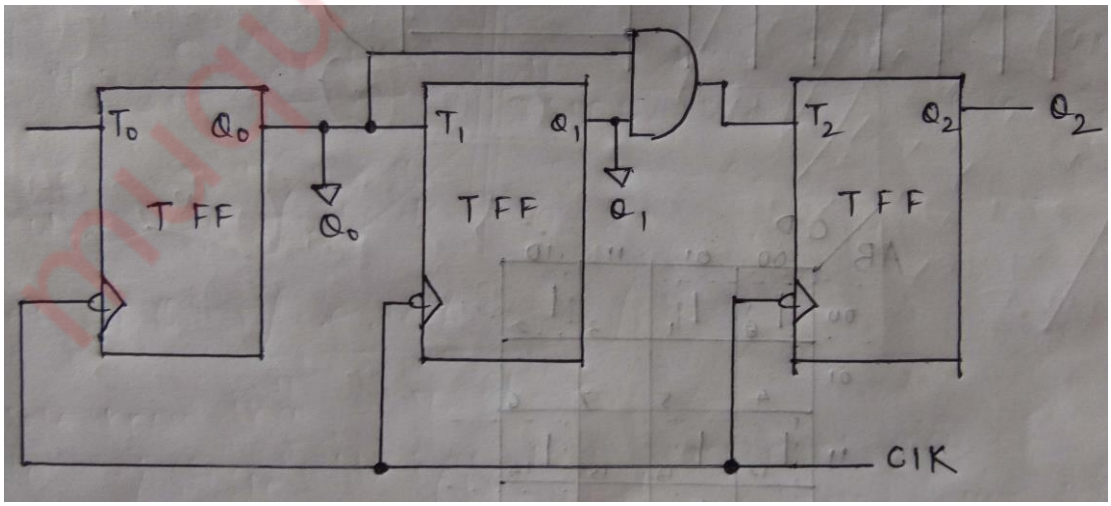


(vii)The 3-bit Synchronous binary up counter contains three T flip-flops & one 2-input AND gate. All these flip-flops are negative edge triggered and the outputs of flip-flops change (affect) synchronously. The T inputs of first, second and third flip-flops are 1, Q0Q0 & Q1Q0Q1Q0 respectively.

(viii)The output of first T flip-flop **toggles** for every negative edge of clock signal.

(ix)The output of second T flip-flop toggles for every negative edge of clock signal if Q0Q0 is 1.

(x)The output of third T flip-flop toggles for every negative edge of clock signal if both Q0Q0 & Q1Q1 are 1.
(Here we can write Q3Q2Q1=Q2Q1Q0).



Q.6 Write short note on (any four) **[20]**

(a) State table :

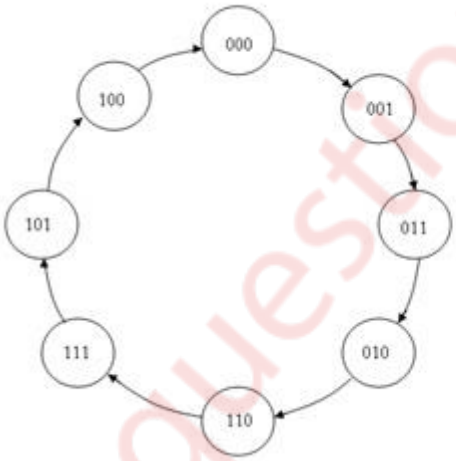
Ans :

i) The state table representation of a sequential circuit consists of three sections labeled *present state*, *next state* and *output*.

Ii)The present state designates the state of flip-flops before the occurrence of a clock pulse.

iii) The next state shows the states of flip-flops after the clock pulse, and the output section lists the value of the output variables during the present state.

iv) In addition to graphical symbols, tables or equations, flip-flops can also be represented graphically by a state diagram.
v) In this diagram, a state is represented by a circle, and the transition between states is indicated by directed lines (or arcs) connecting the circles.



Present State			Next State		
Q2	Q1	Q0	Q2	Q1	Q0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0

(b) ALU IC 74181

Ans :

i)The **74181** is a bit slice arithmetic logic unit (ALU), implemented as a 7400 series TTL integrated circuit.

ii)The first complete ALU on a single chip, it was used as the arithmetic/logic core in the CPUs.

iii) The 74181 is a 7400 series medium-scale integration (MSI) TTL integrated circuit, containing the equivalent of 75 logic gates and most commonly packaged as a 24-pin DIP.

iv) The 4-bit wide ALU can perform all the traditional add / subtract / decrement operations with or without carry, as well as AND / NAND, OR / NOR, XOR, and shift.

v) Many variations of these basic functions are available, for a total of 16 arithmetic and 16 logical operations on two four-bit words. Multiply and divide functions are not provided but can be performed in multiple steps using the shift and add or subtract functions.

vi)Shift is not an explicit function but can be derived from several available functions; e.g., selecting function "A plus A" with carry (M=0) will give an arithemetic left shift of the A input.

(c) Sequence generator :

Ans :

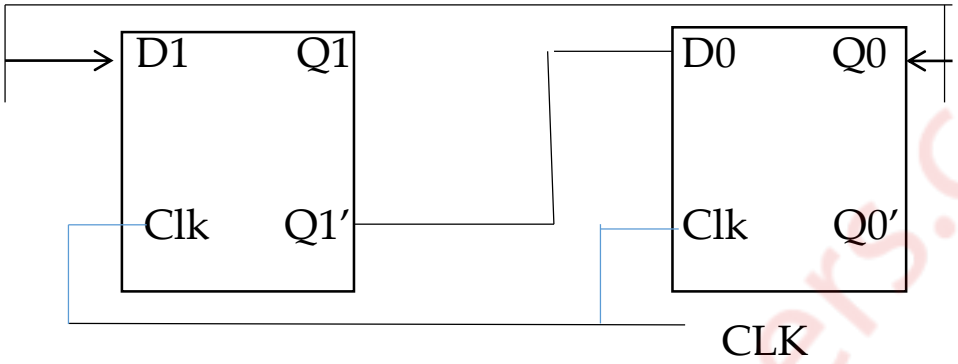
i)There are counters which pass through a definite number of states in a pre-determined order. For example, a 3-bit up-counter counts from 0 to 7 while the same order is reversed in the case of 3-bit down counter.

ii) These circuits when suitably manipulated can be made to count till an intermediate level also. This means that instead of counting till 7, we can terminate the process by resetting the counter just at, say, 5. Such counters are then known as mod-N counters.

iii)We can design sequence generator using some D flip flops.
It will generate sequence from some given input binary digit.

iv) Sequence generator :

Q_1	Q_0	Q_1*	Q_0*	D_1	D_0
0	0	0	1	0	1
0	1	1	1	1	1
1	1	1	0	1	0
1	0	0	0	0	0



Sequence Generator

(d) Data flow modelling
Ans :

i)The process of identifying, modeling and documenting how data moves around an information system. Data flow modeling examines processes (activities that transform data from one form to another), data stores (the holding areas for data), external entities (what sends data into a system or receives data from a system, and data flows (routes by which data can flow).

ii)Data flow modeling is one of the foundations of the Structured Systems Analysis and Design Method.

iii)Data flow modeling also is called data flow diagramming.

(e) 4 bit ring counter

Ans :

i) A ring counter is a Shift Register (a cascade connection of flip-flops) with the output of the last flip flop connected to the input of the first.

ii)It is initialised such that only one of the flip flop output is 1 while the remander is 0.

iii)The 1 bit is circulated so the state repeats every n clock cycles if n flip-flops are used. The MOD of a counter is the number of unique states. The MOD of the n flip flop ring counter is n.

iv)It can be implemented using D-type flip-flops (or JK-type flip-flops).

v) 4-bit Ring Counter :

Truth Table :

CLK	Q_3	Q_2	Q_1	Q_0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0

DLDA

(CBCGS MAY 2018)

Q1] a) Write the entity declaration in VHDL for NOR gate.

(02)

Solution :-

NOR Gate:
A logic gate whose output logic is '1' if and only if all of its inputs are logic '0'.
Entity declaration of NOR gate

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity nor_df is
    Port (a: in STD_LOGIC;
          b: in STD_LOGIC;
          c: out STD_LOGIC);
end nor_df;
architecture Behavioral of nand_df
is
begin
c<=a nor b;
```

Q1] b) Add $(22)_{10}$ to $(56)_{10}$ in BCD.

(02)

Solution :-

BCD of $(22)_{10}$ is 00100010
BCD of $(56)_{10}$ is 01010110

```
    0 0 1 0 0 0 1 0
+   0 1 0 1 0 1 1 0
-------------------
    0 1 1 1 1 0 0 0
```

$(22)_{10} + (56)_{10} = 01111000$

Q1] c) Convert decimal 57 into binary, base 7 and hexadecimal.

(02)

Solution :-

Decimal 57 to binary

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

Decimal 57 to base 7

7	57	
7	8	1
7	1	1
	0	1

Decimal 57 to hexadecimal

16	57	
16	3	9
	0	3

Q1] d) Construct hamming code for 1010.

(02)

Solution :-

7 bit Hamming code is given as

D7	D6	D5	P4	D3	P2	P1

Given 4 bit is 1010 therefore it can be written as

1	0	1	P4	0	P2	P1

P1: For P1 consider P1, D3, D5, D7

D3, D5, D7 = 011

For even parity set P1 = 0

P1: For P2 consider P2, D3, D6, D7
D3, D6, D7 = 001
For even parity set P2 = 1
P1: For P1 consider P4, D5, D6, D7
D5, D6, D7 = 011
For even parity set P4 = 0

1	0	1	0	0	1	0

Therefore, hamming code for 1010 is given as 1010010.

Q1] e) Perform subtraction using 2's complement for $(10)_{10}$ - $(7)_{10}$.

(02)

Solution :-

Binary representation of 7 is 0111
1's complement of 7 is 1000
2's complement of 7 is 1's complement + 1

```
    1 0 0 0
+       1
-----------
    1 0 0 1
```

Now adding 10 and 2's complement of 7

```
    1 0 1 0
+   1 0 0 1
-----------
  1 0 0 1 1
```

Discard carry 1
Therefore **$(10)_{10}$ - $(7)_{10}$ = 0011 = $(3)_{10}$.**

Q1] f) State and prove De Morgan's law.

(02)

Solution :-

Theorem 1:

Statement: The complement of the product of two or more variables is equal to the sum of the components of the variables.
Logical expression: $\overline{A.B} = \overline{A} + \overline{B}$
NAND is equal to BUBBLED OR.

A	B	A.B	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

From the above truth table we can see that $\overline{A.B} = \overline{A} + \overline{B}$ Hence proved.

Theorem 2 :-

Statement: The complement of the sum of two or more variables is equal to the product of the complement of the variables
Logical expression: $\overline{A + B} = \bar{A}.\bar{B}$
NOR is equal to BUBBLED AND.

A	B	A + B	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

From the above truth table we can see that $\overline{A + B} = \bar{A}.\bar{B}$ Hence proved.

Q1] g) Convert $(77)_{10}$ into Excess 3 code.

(02)

Solution :-

Excess 3 code is 3 added to BCD.
BCD of 77 is 0111 0111 and 0f 3 is 0011

```
  0 1 1 1 0 1 1 1
+ 0 0 1 1 0 0 1 1
-----------------
  1 0 1 0 1 0 1 0
```

Excess 3 code of 77 is 10101010

Q1] h) Perform addition of $(34)_8$ and $(62)_8$.

(02)

Solution :-

$(34)_8 + (62)_8$

```
    3 4
+   6 2
-------
  1 1 6
```

In the above sum 6 + 3 is greater than 7 hence can't be represented as octal therefore we need to divide it with 8

```
    1
  ------
8 | 9
  - 8
  ---
    1
```

Remainder is the sum and the quotient is carry.
Therefore, we get $(34)_8 + (62)_8 = (116)_8$.

Q1] i) Find 8's complement of the number $(37)_8$ and $(301)_8$.

(02)

Solution :-

To find 8's complement we first need to find 7's complement and then add 1 to it.
7's complement of 37 is obtain by subtracting 37 from 77

i)

```
    7 7
+   3 7
-------
    4 0      7's complement
+     1
-------
    4 1      8's complement
```

ii)
```
    7 7 7
+   3 0 1
-------------
    4 7 6      7's complement
+       1
-------------
    4 7 7      8's complement
```

Q1] j) Explain ASCII code in brief.

(02)

Solution :-

ASCII was developed by the American National Standards Institute (ANSI). Pronounced ask-ee, ASCII is the acronym for the **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange.
It is a code for representing 128 English characters as numbers.
Technically, ASCII is 7-bit representing only 128 characters (0-127). The range 0-31 are control characters, with 32-127 representing alphabetical characters from A to Z, numerals from 0 to 9 and punctuation marks (though not in that order).
For example, the ASCII code for uppercase M is 77.
Most computers use ASCII codes to represent text, which makes it possible to transfer data from one computer to another.
Text files stored in ASCII format are sometimes called ASCII files. Text editors and word processors are usually capable of storing data in ASCII format, although ASCII format is not always the default storage format.

Q2] a) Simplify the following equation using K map to obtain SOP equation and realize the minimum equation using only NAND gates.

$F(A,B,C,D) = \Sigma m(1,2,4,6,9,10,12,14) + d(3,7,13)$

(10)

Solution :-

Step 1: Draw K map

	00	01	11	10
00	0	1	X	1
01	1	0	X	1
11	1	X	0	1
10	0	1	0	1

Step 2: SOP Equation

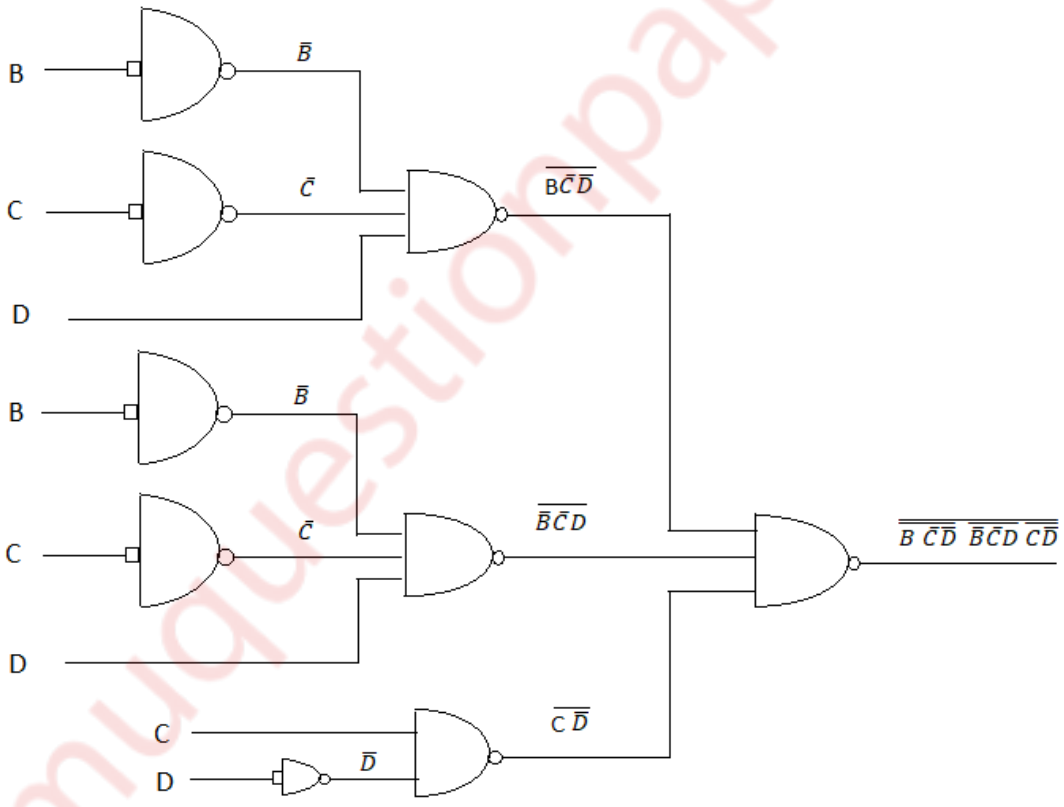
Y = B$\bar{C}\bar{D} + \bar{B}\bar{C}D$ + C $\bar{D}$

Step 3: NAND gate realization

Take double inversion

$Y = \overline{\overline{\mathrm{B}\bar{C}\bar{D} + \bar{B}\bar{C}D + \mathrm{C}\,\bar{D}}}$

Y = $\overline{\overline{B}\,\overline{\bar{C}\bar{D}}\;\overline{\bar{B}\bar{C}D}\,\overline{C\bar{D}}}$



Q2] b) Implement full adder using 8:1 mux.

(10)

Solution :-

Truth table of full adder

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

In full adder there are 3 input A, B and Cin and they are added and generates two output S and Cout. S is the sum of A, B and Cin and Cout is the carry generated by adding 3 input's.

Full Adder Implementation using 8:1 MUX

An 8:1 multiplexer consists of eight data inputs I0 to I7, three input select lines A, B and C and a single output line Y. Depending on the select lines combinations, multiplexer decodes the inputs.
Since the number of input data bits given to the MUX are eight therefore 3 bits (23=8) are needed to select one of the eight data bits. That is there is 3 select lines in 8:1 MUX.
Since in full adder there are 2 output, we need two 8:1 multiplexer. Output of 1st multiplexer is sum of adder and output of 2nd multiplexer is carry generated of full adder.

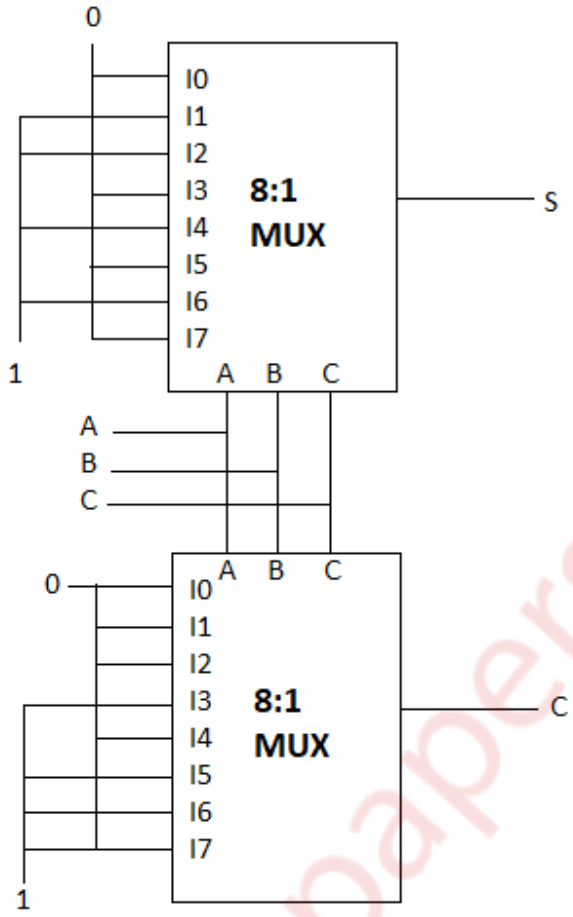


Fig. Full Adder using 8:1 MUX

As we can see in the above diagram there are 3 select lines A, B and C. there are 7 input I0 to I7. The input is connected to 0 or 1 depending upon the value given in truth table.
The select lines of both multiplexer is connected to same input.

Q3] a) Obtain the minimal expression using Quine Mc-Cluskey method

F(A,B,C,D) = Σm(1,2,3,5,6,10.11,13,14) + d(4,7)

(10)

Solution :-

Table 1

Group	Minterm	Binary Representation			
		A	B	C	D
0	M1	0	0	0	1
	M2	0	0	1	0
	M4*	0	1	0	0
1	m3	0	0	1	1
	m5	0	1	0	1

	m6	0	1	1	0
	m10	1	0	1	0
2	m13	1	1	0	1
	m14	1	1	1	0
	m7*	0	1	1	1

Table 2

Group	Minterm	Binary Representation			
		A	B	C	D
0	m1-m3	0	0	_	1
	m1-m5	0	_	0	1
	m2-m3	0	0	1	_
	m2-m6	0	_	1	0
	m2-m10	_	0	1	0
	m4-m5*	0	1	0	_
	m4-m6*	0	1	_	0
1	m3-m7*		0	_	1
	M5-m13	_	1	0	1
	m5-m7*	0	1	_	1
	m6-m14	_	1	1	0
	m6-m7*	0	1	1	_
	m10-m14	1	_	1	0

Table 3

Group	Minterm	Binary Representation			
		A	B	C	D
0	m1-m3-m5-m7	0	_	_	1
	M1-m5-m3-m7	0	_	_	1
	M1-m5-m10-m14	_	_	_	_
	M2-m3-m6-m7	0	_	1	_
	M2-m6-m3-m7	0	_	1	_
	M2-m6-m10-m14	_	_	1	0
	M2-m10-m5=m13	_	_	_	_
	M2-m10-m6-m14	_	_	1	0
	M4-m5-m6-m7	0	1	_	_
	M4-m6-m5-m7	0	1	_	_

PI	Decimal of PI	1	2	3	5	6	10	13	14	4	7
$\bar{A}D$	M1 m3 m5 m7	⊗		X	X						X
$\bar{A}C$	M2 m3 m6 m7		X	X		X					X
$C\bar{D}$	M2-m6-m10-m14		X			X	⊗		⊗		
$\bar{A}B$	M4-m5-m6=m7				X	X				⊗	X

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

Q3] b) What is race around condition? How to overcome it?

(10)

Solution:-

The race around condition occurs in JK flipflop when both the inputs are high that is J=K=1 normally in level triggered JK flipflop.

Fig.3.1 Race around condition

The above figure shows the race around condition in JK Flipflop when J=K=1.

This may lead to unpredictable output when clock disables and it is called as race around condition.

This can be avoided by

1. Using edge triggered JK flipflop
2. Using Master Slave JK flipflop

In edge triggered JK flipflop, race condition is avoided as the enable input of the clock remains active only for a short time.

In master slave flipflop master and slave are inverted clock so that only one will be active at a time. During the high of the clock, output will be toggled and will remain at the output of master but could not be followed by slave as it is disabled. So, output of slave remains same as earlier and toggled output remains at master. When clock is low master is deactivated and slave is enabled which only follows the output of master. So at the end of one complete clock we just have one toggled output and race around condition is avoided.

Q4] a) Design 3 bit asynchronous counter and draw the timing diagram.

(10)

Solution :-

A ripple counter is an asynchronous counter where only the first flip-flop is clocked by an external clock. All subsequent (next occurring) flip-flops are clocked by the output of the preceding flip-flop.

In asynchronous counter, a clock pulse drives FF0. Output of FF0 drives FF1 which then drives the FF2 flip flop. All J and K inputs are connected to Logic 1. Therefore, each flip flop will toggle with negative transition at its clock input.

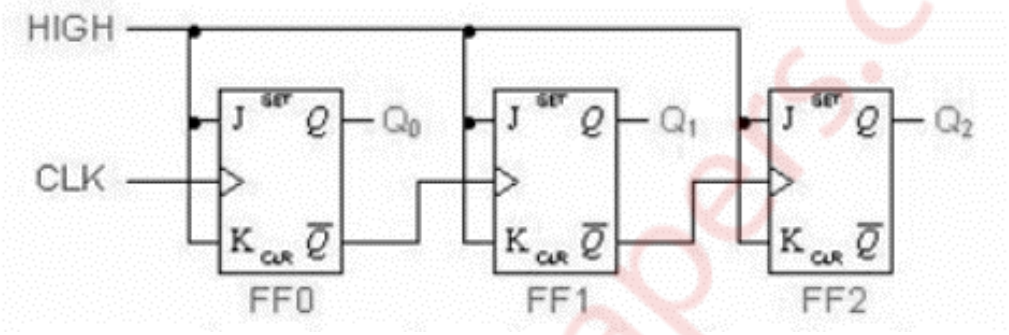


Fig. 4.1 bit Asynchronous counter

The 3 bit MOD-8 asynchronous counter consists of 3 JK flip flops. Overall propagation delay time is the sum of individual delays. Initially all flip flops are reset to produce 0. The output conditions is Q2 Q1 Q0 = 000.

When the first clock pulse is applied, the FF0 changes state on its negative edge. Therefore, Q2Q1Q0 = 001. On the negative edge of second clock pulse flip flop FF0 toggles. Its output changes from 1 to 0. This being negative change, FF1 changes state. Therefore, Q2 Q1 Q0 = 010. Similarly, the output of flipflop FF2 changes only when there is negative transition at its input when fourth clock pulse is applied.

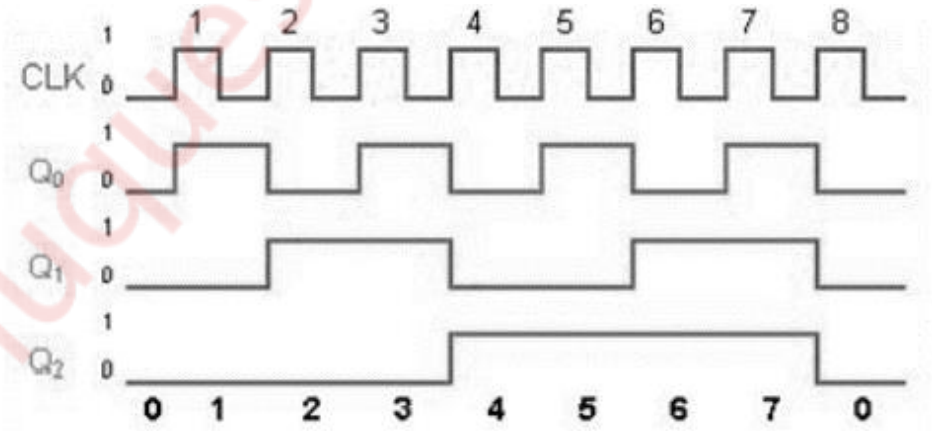


Fig. 4.2 Timing diagram

The output of the flip flops is a binary number equivalent to the number of clock pulses received. The output conditions are as shown in the truth table.

Counter State	Q2	Q1	Qo
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

On the negative edge of eighth pulse, counter is reset. The counter acts as a frequency divider. FF0 divides clock frequency by 2, FF1 divides clock frequency by 4, FF2 divides clock frequency by 8. If n flip flops are cascaded, we get 2n output conditions. The largest binary number counted by n cascaded flip flops has a decimal equivalent of 2n −1. MOD-8 counter has count of the largest binary number 111 which has decimal equivalent of 23−1=7.

Q4] b) Convert JK flipflop to SR flipflop and D flipflop.

(10)

Solution :-

JK to SR

a) Excitation Table

Input's		Present State	Next State	Flip Flop Input	
S	R	Qn	Qn+1	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	X	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	0	X	X	X
1	1	1	X	X	X

b) K- map simplification

For J

Qn \ SR	00	01	11	10
0	0	0	X	1
1	X	X	X	X

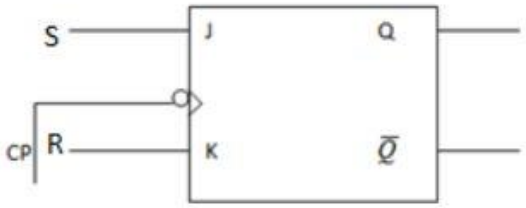
J = S

For K

Qn \ SR	00	01	11	10
0	X	X	X	0
1	0	1	X	X

K = R

c) Logic diagram



JK to D

a) Excitation Table

Input	Present State	Next State	Flip Flop Input	
D	Qn	Qn+1	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0

b) K-map simplification

For J

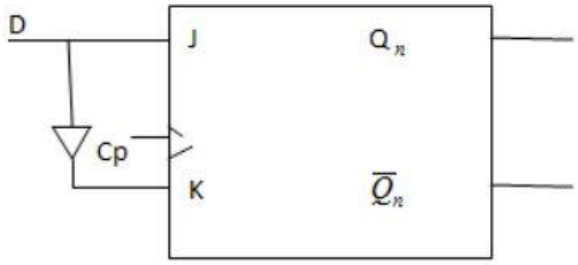
D \ Q_n	0	1
0	0	1
1	×	×

J = D

For K

D \ Q_n	0	1
0	×	×
1	1	0

c) Logic diagram



Q5] a) Compare TTL and CMOS with respect to different parameters.

(10)

Solution :-

PARAMETER	**TTL**	**CMOS**
Basic gate	NAND	INVERTER
Circuit	Transistor- transistor	Complementary MOS
Speed of operation	More	Less
Propagation delay per rate	4- 12 ns	50 ns
Power supply voltage	5V	3 to 15V
Power dissipation	P_D = 10 Mw	P_D = 0.1 mW
Wired collector capability	With passive pull up	With tristate output
Fan-out	10	50
Noise Immunity	Good	Very good to excellent
Speed power product	100 pJ	10.5 pJ
Available functions	Very high	High
Compatibility with other families	With DTL	No, but compatible with TTL for 5V supply
Switching speed	Faster than CMOS	Less than TTL
Circuits	It uses BJT's	It uses FET's
Noise immunity	Less than CMOS	Better than TTL
Power consumption	It consumes lot more power than CMOS chips.	Consumes less power than TTL chips.

Q5 b) Explain the features of VHDL and its modeling styles.

(10)

Solution:-

VHDL stands for very high-speed integrated circuit hardware description language. This language was first introduced in 1981 for the department of Defense (DoD) under the VHSIC program.
Features of VHDL are as follows:

1) Designs are organized hierarchically.
2) Each design element has:
   i) A well-defined interface.
   ii) A precise behavioral specification using either algorithmic description or hardware structural description.
3) Models concurrency, timing, and clocking:
   i) Handles asynchronous and synchronous circuits
   ii) Designs can be simulated.
4) Language is not case sensitive.
5) A formal language for specifying the behavior and structure digital circuit.

Modeling styles: Modeling Style means, that how we Design our Digital IC's in Electronics. With the help of modeling style we describe the Design of our Electronics.
Normally we use Three type of Modeling Style in VHDL -
• Data Flow Modeling Style.
• Behavior Modeling Style.
• Structural Modeling Style.
Data Flow Modeling Style – Dataflow system describes a system in terms of how data flows through the system. Data dependencies in the description match those in atypical hardware implementation. A dataflow description directly implies a corresponding gate level implementation.
Dataflow descriptions consist of one or more concurrent signal assignment situation
Behavior Modeling Style – A behavioral description describes a system's behavior or function in an algorithmic fashion.
Behavioral style is the most abstract style. The description is abstract in the sense that it does not directly imply a particular gate-level implementation.
Behavioral style consists of one or more process statements. Each process statement is a single concurrent statement that itself contains one or more sequential statements.
Sequential statements are executed sequentially statements by a simulator, the same as the execution of sequential statements in a conventional programming language.
Structural Modeling Style – In structural style of modelling, an entity is described as a set of interconnected components.
The top-level design entity's architecture describes the interconnection of lower level design entities. Each lower level entity can, in turn, be described as an interconnection of design entities at the next-lower level, and so on.

Structural style is most useful and efficient when a complex system is described as an interconnection of moderately complex design entities. This approach allows each design entity to be independently designed and verified before being used in the higher-level description.

Q6] a) Moore and Mealy machine. **(05)**

Solution :-

Moore Machines: Moore machines are finite state machines with output value and its output depends only on present state. It can be defined as (Q, q0, ∑, O, δ, λ) where:

- Q is finite set of states.
- q0 is the initial state.
- ∑ is the input alphabet.
- O is the output alphabet.
- δ is transition function which maps Q×∑ → Q.
- λ is the output function which maps Q → O.

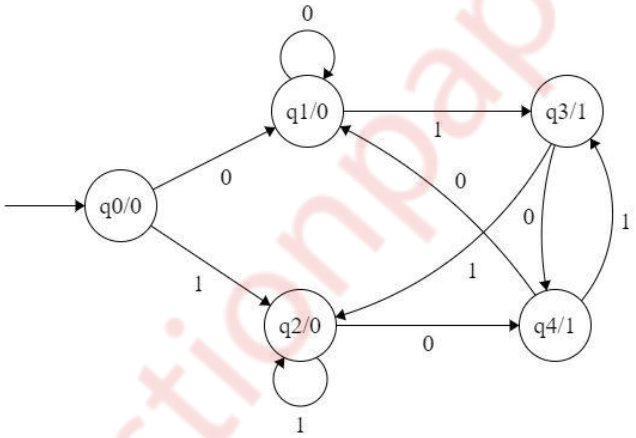


Figure 1

In the moore machine shown in Figure 1, the output is represented with each input state separated by /. The length of output for a moore machine is greater than input by 1.

- **Input:** 11
- **Transition:** δ (q0,11)=> δ(q2,1)=>q2
- **Output:** 000 (0 for q0, 0 for q2 and again 0 for q2)

Mealy Machines: Mealy machines are also finite state machines with output value and its output depends on present state and current input symbol. It can be defined as (Q, q0, ∑, O, δ, λ') where:

- Q is finite set of states.
- q0 is the initial state.
- ∑ is the input alphabet.
- O is the output alphabet.
- δ is transition function which maps Q×∑ → Q.
- 'λ' is the output function which maps Q×∑→ O.

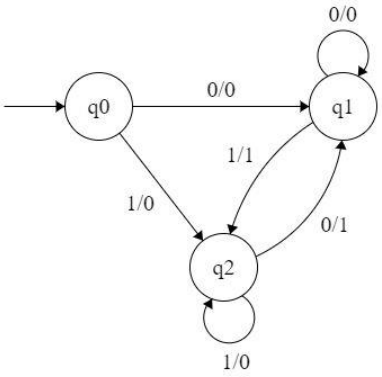


Figure 2

In the mealy machine shown in Figure 1, the output is represented with each input symbol for each state separated by /. The length of output for a mealy machine is equal to the length of input.

- **Input:**11
- **Transition:** δ (q0,11)=> δ(q2,1)=>q2
- **Output:** 00 (q0 to q2 transition has Output 0 and q2 to q2 transition also has Output 0)

Q6] b) Sequence generator.

(05)

Solution :-

the sequence generators are nothing but a set of digital circuits which are designed to result in a specific bit sequence at their output. There are several ways in which these circuits can be designed including those which are based on multiplexers and flip-flops.

A circuit which generates a prescribed sequence of bits, in synchronization with a clock, is referred to as a sequence generator such generator can be used as:

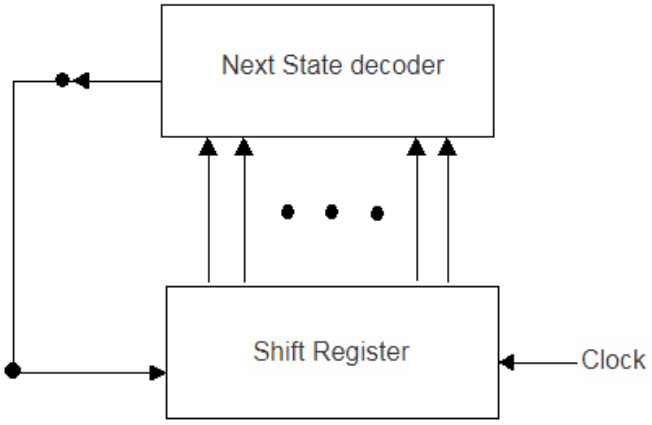
1] counters.

2] Random bit generators.

3] Prescribed period and sequence generators.

4] Code generator.

The basic structure of a sequence generator is shown in diagram below:



Basic Structure of a Sequence Generator.

The output Y of the next state decoder is function of QN-2, QN-1 …….. Q1, Q0. This system is similar to a ring counter or twisted ring counter which are special case of sequence generator, the design of the decoder will be clear.

Q6] c) Universal shift register.

(05)

Solution:-

A Universal shift register is a register which has both the right shift and left shift with parallel load capabilities. Universal shift registers are used as memory elements in computers. A Unidirectional shift register is capable of shifting in only one direction. A bidirectional shift register is capable of shifting in both the directions. The Universal shift register is a combination design of bidirectional shift register and a unidirectional shift register with parallel load provision.
n-bit universal shift register –
A n-bit universal shift register consists of n flip-flops and n 4×1 multiplexers. All the n multiplexers share the same select lines(S1 and S0)to select the mode in which the shift register operates. The select inputs select the suitable input for the flip-flops.
The working of the Universal shift register depends on the inputs given to the select lines.

The register operations performed for the various inputs of select lines are as follows:

S1	S0	REGISTER OPERATION
0	0	No changes
0	1	Shift Right
1	0	Shift Left

1	1	Parallel Load

Q6] d) Priority Encoder.

(05)

Solution :-

The priority encoders output corresponds to the currently active input which has the highest priority. So, when an input with a higher priority is present, all other inputs with a lower priority will be ignored.

A priority encoder provides n bits of binary coded output representing the position of the highest order active input of 2^ninputs. If two or more inputs are high at the same time, the input having the highest priority will take precedence.

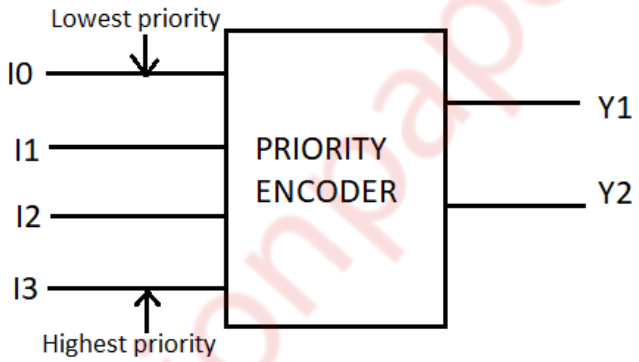


Fig.6.d Priority Encoder

A 4-to-2 priority encoder takes 4 input bits and produces 2 output bits as shown in figure 6.4. In this truth table, for all the non-explicitly defined input combinations (i.e. inputs containing 2, 3, or 4 high bits) the lower priority bits are shown as don't cares (X). Similarly, when the inputs are 0000, the outputs are not valid and therefore they are XX.

Truth table for 4 bit input to priority encoder:

I3	I2	I1	I0	Y1	Y0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
0	1	X	X	1	0

1	X	X	X	1	1

From this truth table, we use the Karnaugh Map to minimize the logic to the following Boolean expressions:

Y1 = I2 + I3

Y0 = I3 + I2 X I1

Q6] e) Carry look ahead adder.

(05)

Solution :-

In ripple carry adders, for each adder block, the two bits that are to be added are available instantly. However, each adder block waits for the carry to arrive from its previous block. So, it is not possible to generate the sum and carry of any block until the input carry is known. This disadvantage is overcomed in carry look ahead adder.

Carry Look-ahead Adder:
A carry look-ahead adder reduces the propagation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two-level logic.

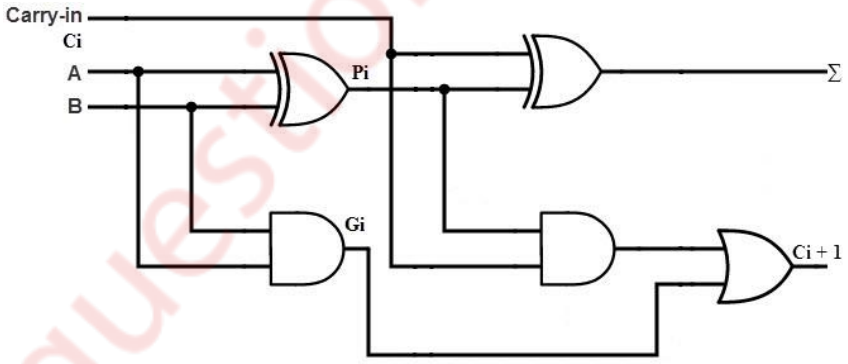


Fig 6.e Carry look ahead adder

We define two variables as 'carry generate' Gi and 'carry propagate' Pi then,

Pi = Ai ⊕ Bi

Gi = Ai x Bi

DIGITAL LOGIC DESIGN AND ANALYSIS (DLDA)

NOVEMBER--2018

Q1 (a) Convert decimal number 576.24 into binary, base-9, octal, hexadecimal system. **(04)**

Solution:
(i)Conversion to Binary

2	576	
2	288	0
2	144	0
2	72	0
2	36	0
2	18	0
2	9	0
2	4	1
2	2	0
2	1	0
	0	1

The binary equivalent of 576 is 1001000000.

0.24 x 2 = 0.48 ~ 0
0.48 x 2 = 0.96 ~ 0
0.96 x 2 = 1.92 ~ 1-
0.92 x 2 = 1.84 ~ 1
0.84 x 2 = 1.68 ~ 1

$(576.24)_{10} = (1001000000.00111)_{2}$

(ii)Conversion to Base-9

9	576	
9	64	0
9	7	1
	0	7

The base-9 equivalent of 576 is (710).

0.24 x 9 = 2.16 ~ 2
0.16 x 9 = 1.44 ~ 1
0.44 x 9 = 3.96 ~ 3
0.96 x 9 = 8.64 ~ 8

$(576.24)_{10}=(710.2138)_{9}$

(iii)Conversion to Octal

8	576	
8	72	0
8	9	0
8	1	1
	0	1

The octal equivalent of 576 is (1100).

0.24 x 8 = 1.92 ~ 1
0.92 x 8 = 7.36 ~ 7
0.36 x 8 = 2.88 ~ 2
0.88 x 8 = 7.04 ~ 7

$(576.24)_{10}= (1100.1727)_{8}$

(iv)Conversion to Hexadecimal

16	576	
16	36	0
16	2	4
	0	2

The Hexadecimal equivalent of 576 is (240).

0.24 x 16 = 3.84 ~ 3
0.84 x 16 = 13.44 ~ D
0.44 x 16 = 7.04 ~ 7
0.04 x 16 = 0.64 ~ 0

$(576.24)_{10} = (240.3D70)_{16}$

(b) Construct hamming code for 1010 using odd parity. (04)

Solution:

The given code is 1010 i.e. 4-bits.

∴ 3 Parity bits are required.

1	2	3	4	5	6	7
P1	P2	1	P3	0	1	0

Using Odd - parity,
P1 = (3, 5, 7) = (1, 0, 0) = 0
P2 = (3, 6, 7) = (1, 1, 0) = 1
P3 = (5, 6, 7) = (0, 1, 0) = 0
∴ Hamming code is as follows:

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

The Hamming code for the given code is 1010 is 0110010.

(c)Convert $(-89)_{10}$ to its equivalent sign magnitude, 1's Complement and 2's Complement Form. **(04)**

Solution:

The sign magnitude equivalent of $(-89)_{10}$ is as follows:

Sign bit

1	1	0	1	1	0	0	1

MSB magnitude

$(-89)_{10} = (11011001)_2$

The binary equivalent of $(-89)_{10}$ is $(11011001)_2$.
So, 1`s complement of $(-89)_{10}$ is $(00100110)_2$

2's complement of $(-89)_{10}$ is :

```
    0   0   1   0   0   1   1   0
+                               1
_____________________________________
    0   0   1   0   0   1   1   1
```

So, 2's complement of $(-89)_{10}$ is $(00100111)_2$.

(d)Perform $(BC5)_H$ - $(A2B)_H$ without converting to any other base. **(04)**

Solution:

```
    B   ¹²C   ¹⁵5
-   A    2    B
_____________________
```

1 9 A

Here, as 5 < B, so we borrow 16 from the previous value as the given values are in Hexadecimal format. 16 is then added to 5 ,thus it becomes 21 and 21 is subtracted from B. So, we get answer as A.

Since, we have borrowed from the previous value i.e C ,the value of C decreases by 1.So, it becomes B.

Thus, $(BC5)_H - (A2B)_H = (19A)_H$

(e) Prove De Morgan's theorem. **(04)**

Solution:

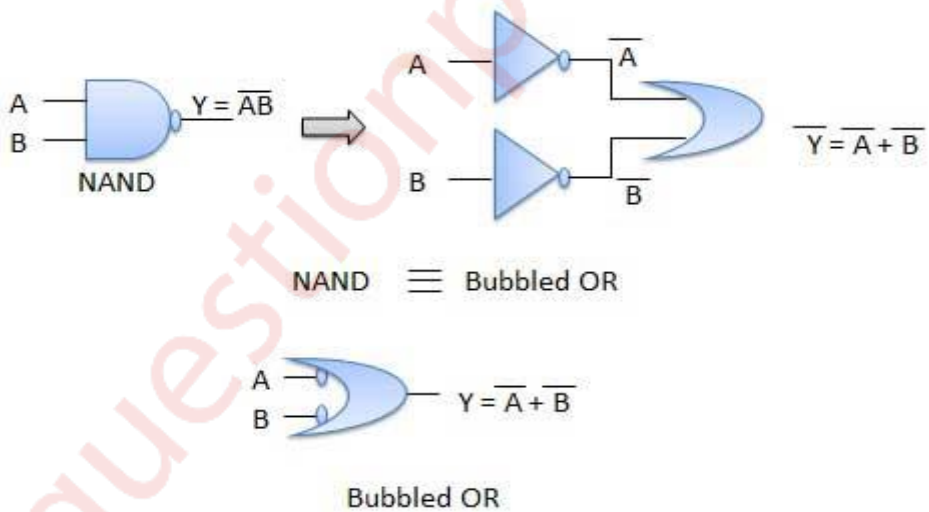
The two theorems suggested by De-Morgan and which are extremely useful in Boolean algebra are stated as follows:

Theorem 1: $\overline{AB} = \overline{A} + \overline{B}$

NAND = Bubbled OR:

This theorem states that the complement of a product is equal to the sum of individual complements.

This theorem can be verified using truth table as follows:



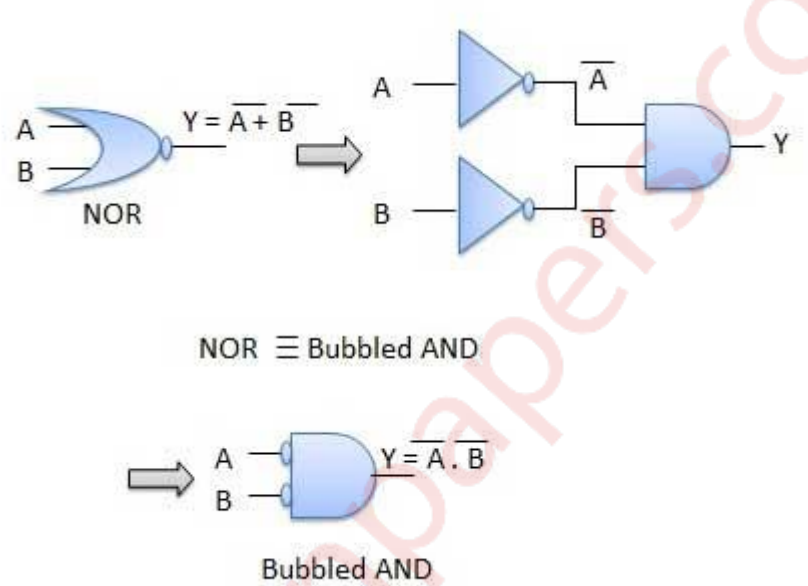
A	B	$\overline{AB}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

Thus we can see that $\overline{AB} = \overline{A} + \overline{B}$.

Theorem 2 : $A + B = A \cdot B$

NOR = Bubbled AND:

This theorem states that the complement of a sum is equal to the product of individual complements.



This theorem can be verified using truth table as follows:

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

Thus we can say that $\overline{A + B} = \overline{A} \cdot \overline{B}$.

Q.2(a)Given the logic expression $A + \overline{B}\ \overline{C} + A B \overline{D} + A B C D$ (10)

1. **Express it in standard SOP form.**
2. **Draw K-map and simplify**
3. **Draw logic diagram using NOR gates only.**

Solution:

The given expression is :

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

(i)Expressing the given expression in standard SOP form:

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

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

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

$\therefore A B C D + A B \overline{C} D + A B C \overline{D} + A B \overline{C}\ \overline{D} + A \overline{B} C D + A \overline{B}\ \overline{C} D + A \overline{B} C \overline{D} + A \overline{B}\ \overline{C}\ \overline{D} +$

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

The standard SOP form of the given expression is :

$\therefore \overline{A}\ \overline{B}\ \overline{C}\ \overline{D} + \overline{A}\ \overline{B}\ \overline{C} D + A \overline{B}\ \overline{C}\ \overline{D} + A \overline{B}\ \overline{C} D + A \overline{B} C \overline{D} + A \overline{B} C D + A B \overline{C}\ \overline{D} + A B \overline{C} D +$

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

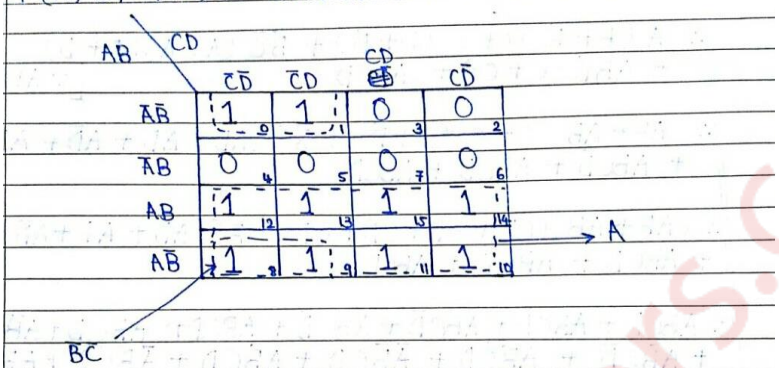
(ii)Using K-map to simplify the given expression :

From standard SOP form we get,
$F(A, B, C, D) = \Sigma m (0, 1, 8, 9, 10, 11, 12, 13, 14, 15)$

2. Using K-map to simplify the given expression.

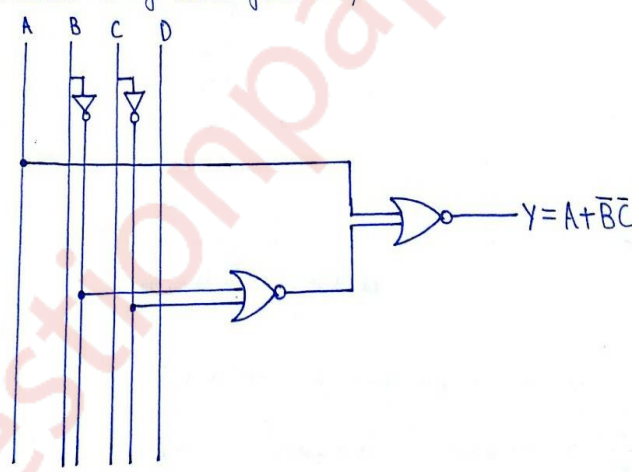
From standard SOP form we get

$F(A, B, C, D) = \Sigma m(0, 1, 8, 9, 10, 11, 12, 13, 14, 15)$



The simplified expression is $Y = A + \bar{B}\bar{C}$

3. Implementation using NOR gates only



Q.2(b)Reduce using Quine McCluskey method and realize the operation using only NAND gates. **(10)**

$F(A, B, C, D) = \pi M(0, 2, 3, 6, 7, 8, 9, 12, 13)$

Solution:

$F(A, B, C, D) = \pi M(0, 2, 3, 6, 7, 8, 9, 12, 13)$

$F(A, B, C, D) = \Sigma m(1, 4, 5, 10, 11, 14, 15)$

Step 1: Grouping the minterms according to the number of 1's

Group	Minterm	Binary representation			
		A	B	C	D
1	1	0	0	0	1
	4	0	1	0	0
2	5	0	1	0	1
	10	1	0	1	0
3	11	1	0	1	1
	14	1	1	1	0
4	15	1	1	1	1

Step 2: Grouping the minterms that form pairs

Group	Minterm	Binary representation			
		A	B	C	D
1	1-5	0	-	0	1
	4-5	0	1	0	-
2	10-11	1	0	1	-
	10-14	1	-	1	0
3	11-15	1	-	1	1
	14-15	1	1	1	-

Step 3: Grouping the minterms to form quad

Group	Minterm	Binary representation			
		A	B	C	D
1	10-11-14-15	1	-	1	-
	10-14-11-15	1	-	1	-

Step 4: Collecting all Prime Implicants

From Step 2, the prime implicants are $\bar{A}\,\bar{C}\,D$ and $\bar{A}\,B\,\bar{C}$.
From Step 3, we get the prime implicants as A C.

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

Step 5: Preparing the Prime Implicants Table

Prime Implicant (PI)	Decimal numbers (minterms)	Given Minterms
		1 4 5 10 11 14 15
$\bar{A}\,\bar{C}\,D$	1,5	X X
$\bar{A}\,B\,\bar{C}$	4,5	X X
A C	10,11,14,15	X X X X

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

Q.3(a)Design a 4-bit binary to gray code converter. (10)

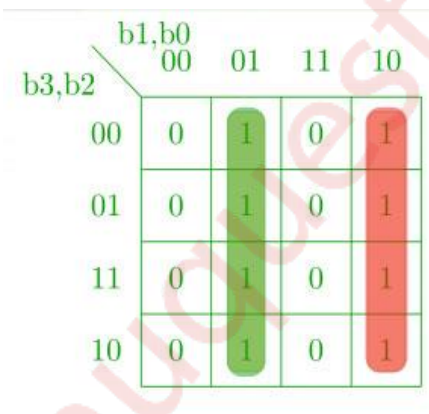
Solution:

The truth table showing binary inputs being converted to gray outputs is as follows:

BINARY INPUT				GRAY CODE OUTPUT			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

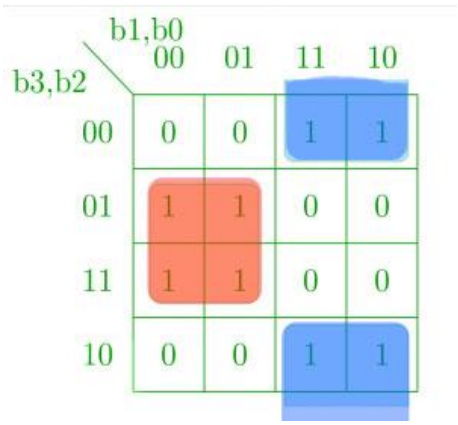
The K-maps for each gray output is as shown below.

K-map for g_0:



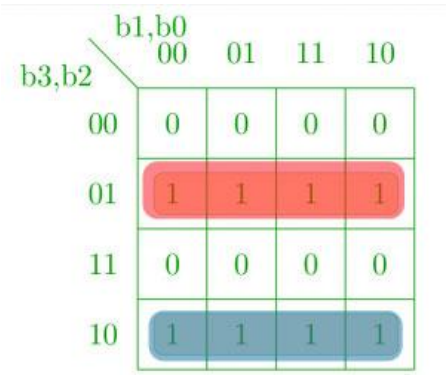
$\therefore g_0 = \overline{b1}\, b0 + b1\, \overline{b0}$

K-map for g_1:

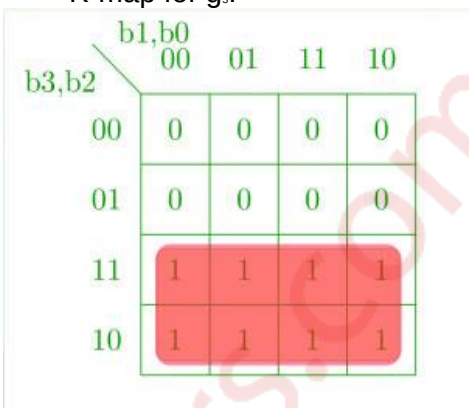


$\therefore g_1 = \overline{b2}\, b3 + b2\, \overline{b3}$

K-map for g_2:



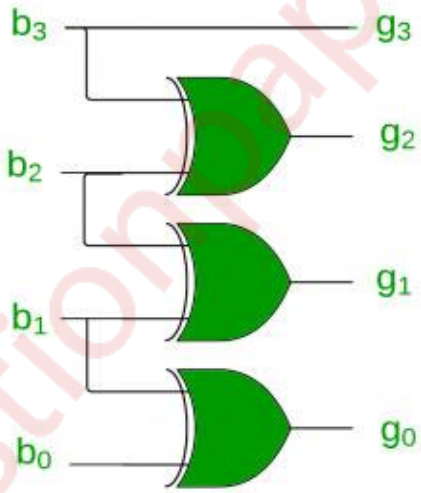
K-map for g_3:



$\therefore g_2 = \overline{b1}\, b2 + b1\, \overline{b2}$

$\therefore g_3 = b3$

The logic diagram of 4-bit Binary to Gray code converter is as shown below.



Q.3(b)Design a 4 bit BCD adder using IC 7483 and necessary gates. (10)

Solution:

1. A BCD adder adds two BCD digits and produces a BCD digit. BCD number cannot be greater than 9.
2. The two given BCD numbers are to be added using the rules of binary addition.
3. If sum is less than or equal to 9 and carry=0 then correction is necessary. The sum is correct and in the true BCD form.
4. But if sum is invalid BCD or carry=1, then the result is wrong and needs correction.
5. The wrong result can be corrected by adding six (0110) to it.
6. The 4 bit binary adder IC 7483 can be used to perform addition of BCD numbers.

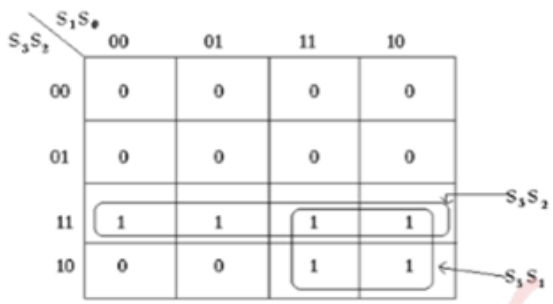
7. In this, if the four-bit sum output is not a valid digit, or if a carry C3 is generated then decimal 6 (0110 binary) is to be added to the sum to get the correct result.
8. Fig1 shows a 1-digit BCD adders can be cascaded to add numbers several digits long by connecting the carry-out of a stage to the carry-in of the next stage.
9. The output of combinational circuit should be 1 if the sum produced by adder 1 is greater than 9 i.e. 1001. The truth table is as follows:

I/P				O/P
S3	S2	S1	S0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Truth table for BCD numbers

Y=1 since the sum obtained is an invalid BCD number.

K-map:



The Boolean expression is
$Y = S_3S_2 + S_3S_1$

- The BCD adder is shown below. The output of the combinational circuit should be 1 if Cout of adder-1 is high. Therefore Y is 0 with Cout of adder 1.
- The output of combinational circuit is connected to B_1B_2 inputs of adder-2 and $B_3 = B_1 + 0$ as they are connected to ground permanently. This makes $B_3B_2B_1B_0 = 0110$ if Y' = 1.
- The sum outputs of adder-1 are applied to $A_3A_2A_1A_0$ of adder-2. The output of combinational circuit is to be used as final output carry and the carry output of adder-2 is to be ignored.

BLOCK DIAGRAM OF BCD ADDER

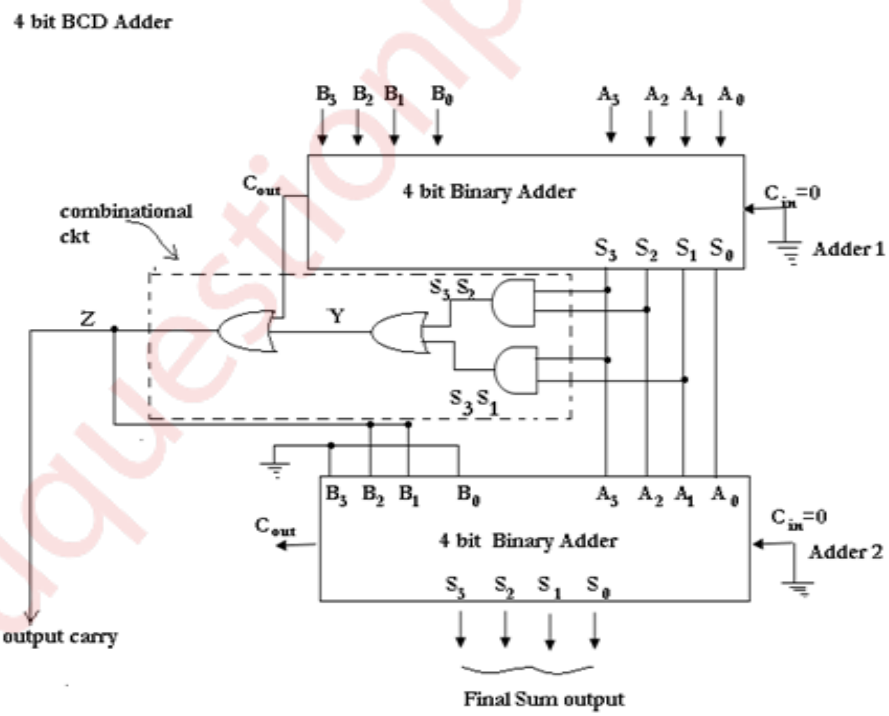


Fig5: 4-bit BCD adder

Operation of BCD Adder is as follows:
Case1: Sum ≤ 9 and carry = 0

- The output of combinational circuit Y' = 0. Hence $B_3B_2B_1B_0$ = 0 0 0 0 for adder-2.

- Hence output of adder-2 is same as that of adder-2

Case 2: Sum >9 and carry = 0

- If S3S2S1S0 of adder -1 is greater than 9, then output Y' of combinational circuits becomes 1.
- Therefore $B_3B_2B_1B_0$ = 0 1 1 0 (of adder-2).
- Hence six (0 1 1 0) will be added to the sum output of adder-1.
- We get the corrected BCD result at the output of adder-2.

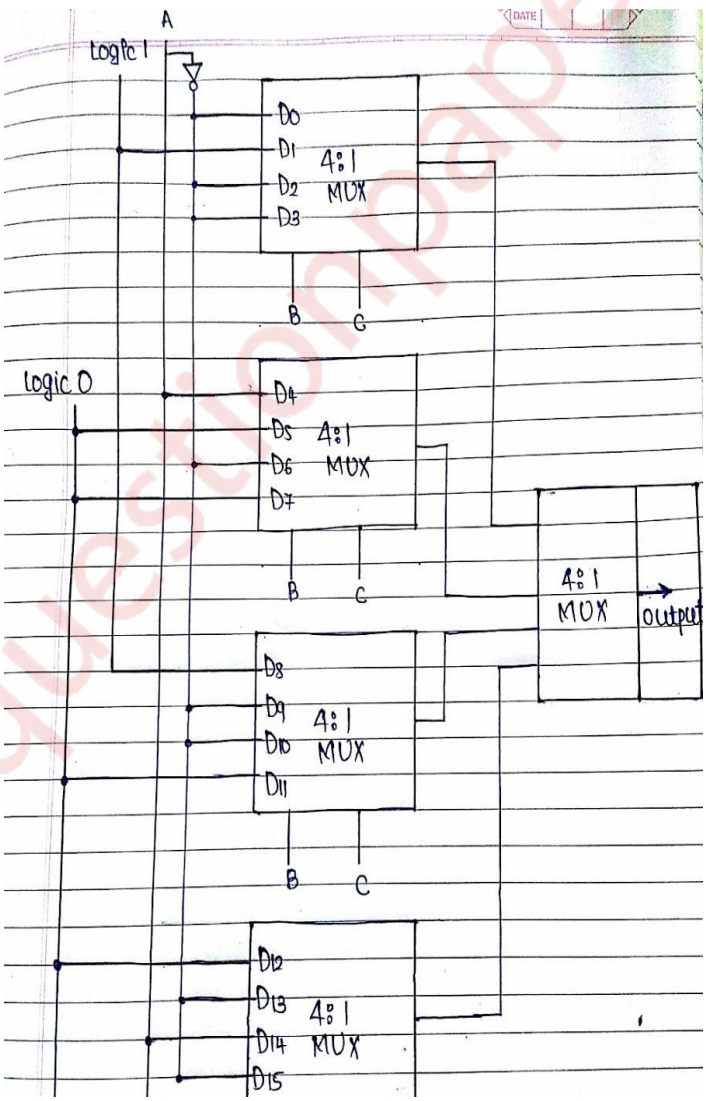
Case 3: Sum ≤ 9 but carry = 1

- As carry output of addere-1 is high, Y' = 1.
- Therefore $B_3B_2B_1B_0$ = 0 1 1 0 (of adder-2).
- Hence six (0 1 1 0) will be added to the sum output of adder-1.
- We get the corrected BCD result at the output of adder-2. Thus the Four bit BCD addition can be carried out using the binary adder.

Q.4(a)Implement the following logic function using all 4 : 1 multiplexers with the select inputs as 'B', 'C', 'D', 'E' only.
F(A, B, C, D, E) = Σ(0,1,2,3,6,8,9,10,13,15,17,20,24,30) (10)

Solution:

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
$\overline{A}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	$\overline{A}$	1	$\overline{A}$	$\overline{A}$	A	0	$\overline{A}$	0	1	$\overline{A}$	$\overline{A}$	0	0	$\overline{A}$	A	$\overline{A}$



Q.4(b)Convert a SR flip flop to JK flip flop. **(10)**

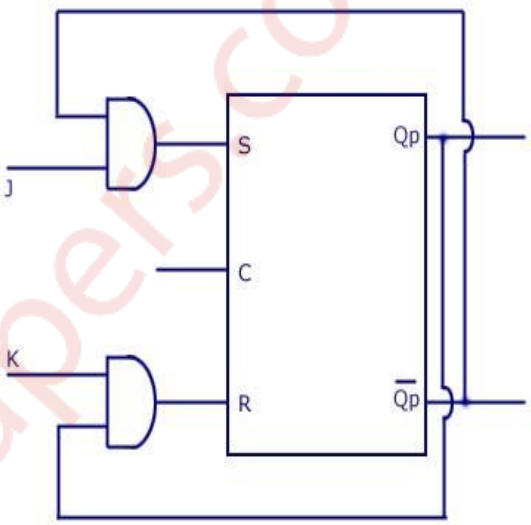
Solution:

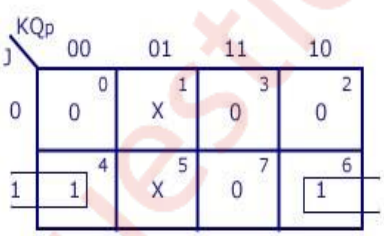
S-R Flip Flop to J-K Flip Flop

Conversion Table

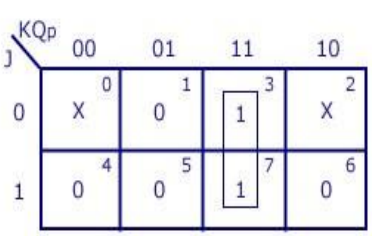
J-K Inputs		Outputs		S-R Inputs	
J	K	Qp	Qp+1	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

Logic Diagram





$S = \overline{J}Q_p$

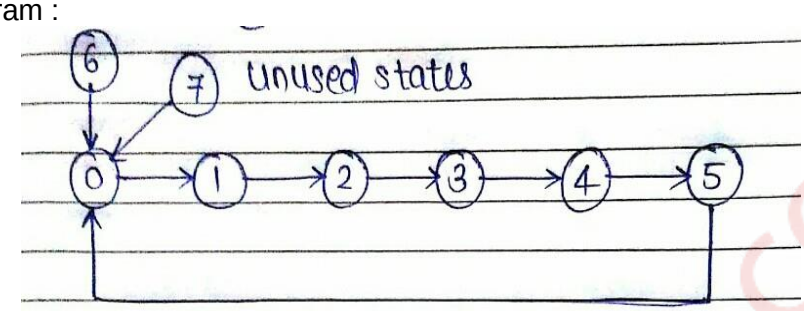


$R = KQ_p$

K-Map

Q.5(a)Design a mod-6 synchronous counter using T flip flop. (10)

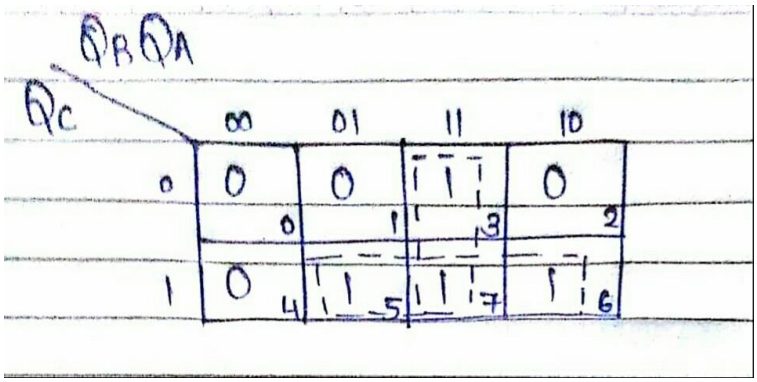
Solution:
(i)State Diagram :



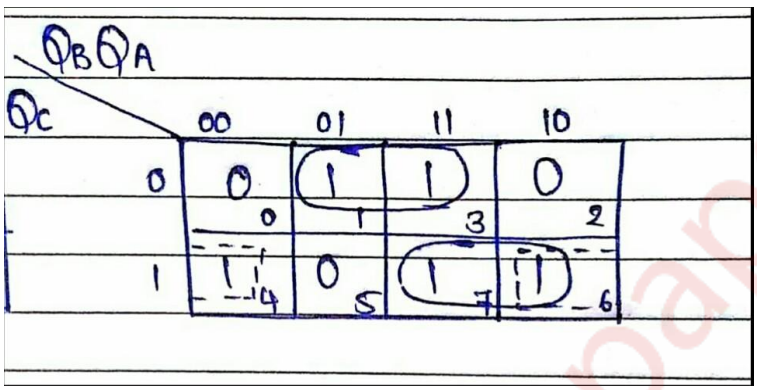
(ii)State Table :

Present State			Next State			Flip-flop inputs		
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	T_c	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	1	1
1	0	1	0	0	0	1	0	1
1	1	0	0	0	0	1	1	0
1	1	1	0	0	0	1	1	1

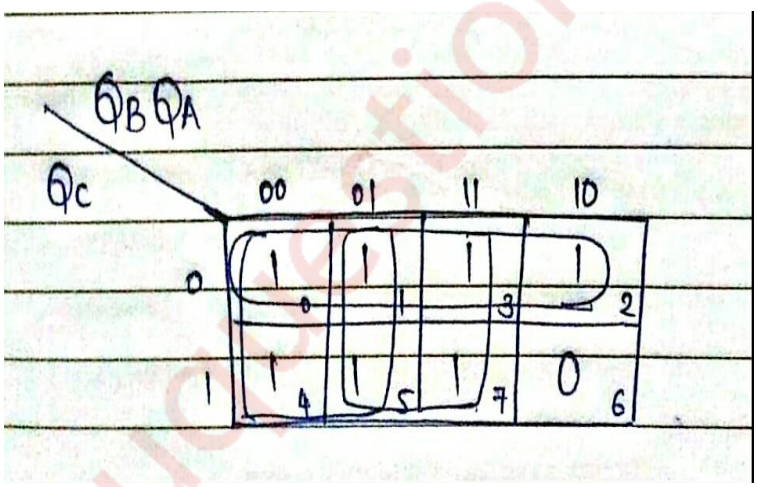
(iii)K map and simplification :



$T_C = Q_A Q_B + Q_B Q_C + Q_A Q_C$

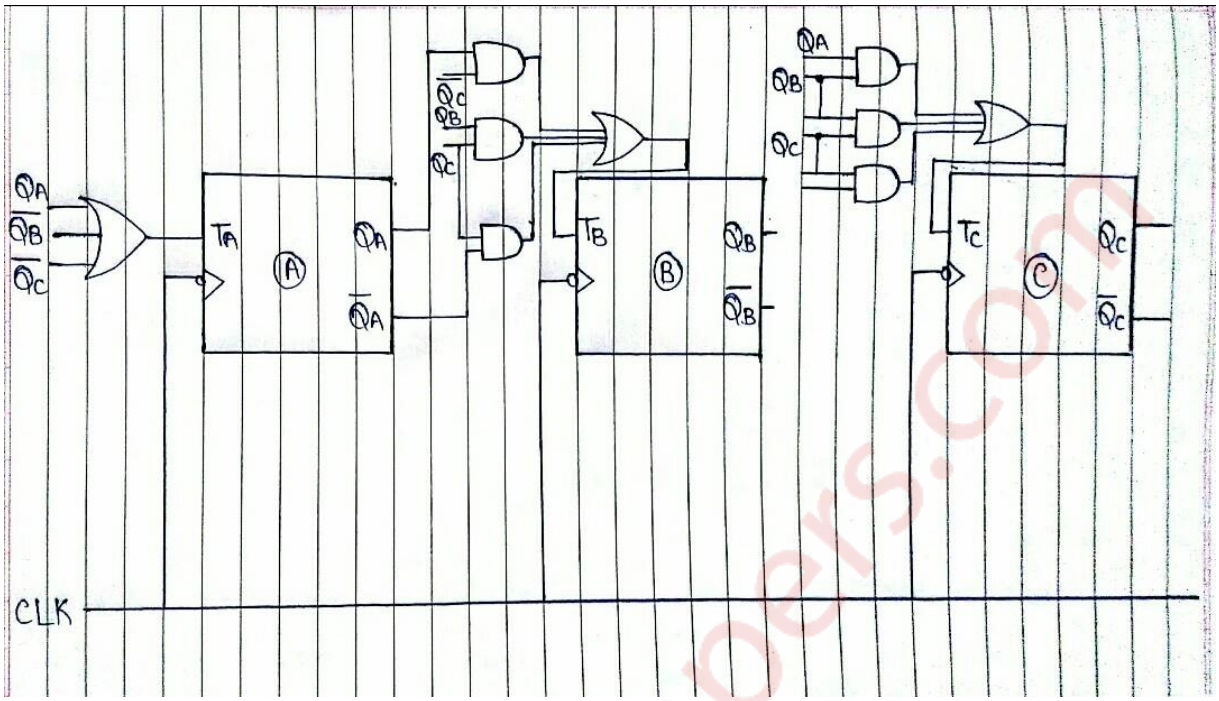


$T_B = Q_A \overline{Q_C} + Q_B Q_C + Q_C \overline{Q_A}$



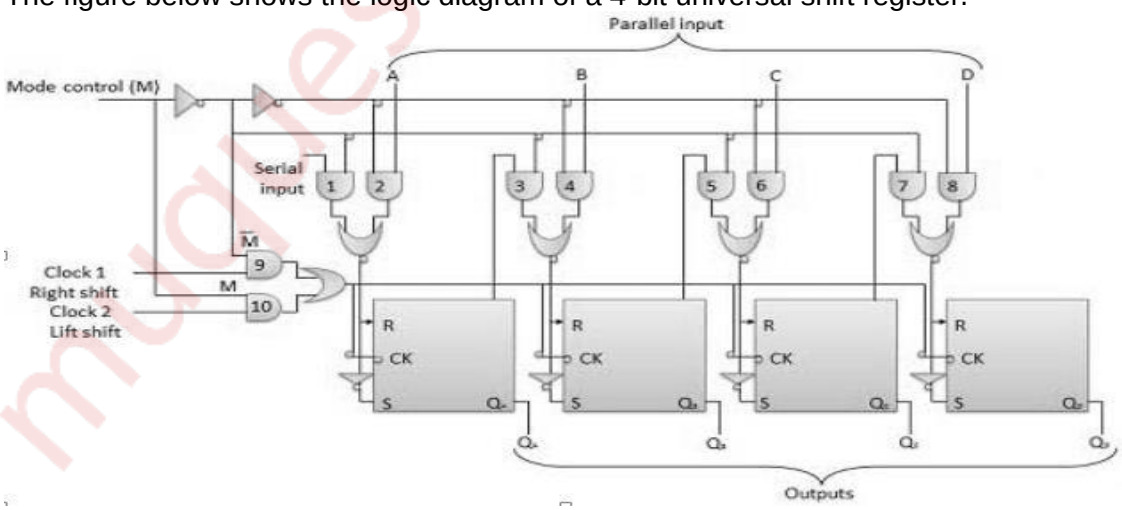
$T_A = \overline{Q_C} + \overline{Q_B} + Q_A$

(iv)Logic Diagram:



Q.5(b)Explain the operation of a 4-bit universal shift register. (10)

Solution:
A shift register which can shift the data in only one direction is called as a unidirectional shift register. A shift register which can shift the data in both directions is called as bi-directional shift register. Similarly, a shift register which can shift data in both directions i.e shift left or right as well as load it parallelly, is called as a universal shift register.
The figure below shows the logic diagram of a 4-bit universal shift register.



This shift register is capable of performing the following operations:
1.Parallel loading (parallel input parallel output)
2.Left shifting

3.Right shifting

The mode control input is connected to Logic 1 for parallel loading operation whereas it is connected to 0 for serial shifting.When mode pin is connected to ground, the universal shift register acts as a bi-directional register.For serial left operation, the input is applied to serial input which goes into AND gate-1 of the above figure.For serial right operation, the serial input is applied to D input (input of AND gate-8).The well known example of universal shift register is in the form of IC 7495.

Universal Shift Register IC 7495:

It is a 4 bit shift register with serial and parallel synchronous operating modes.Because of its capability to operate in all possible modes, it is called a universal shift register.

Some features of this chip are:

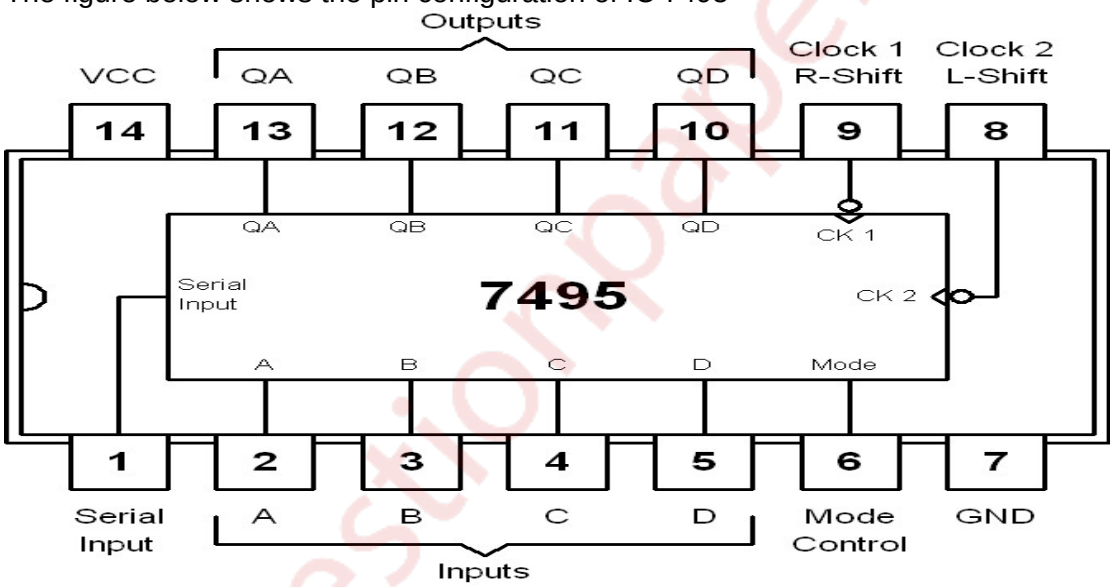
1.Synchronous shift left capacity.

2.Synchronous parallel loading is possible.

3.It has separate clock inputs,one for shift operation and the other for load operation.

4.The cascading of two or more 7495 ICs for more than 4-bits is possible.

The figure below shows the pin configuration of IC 7495



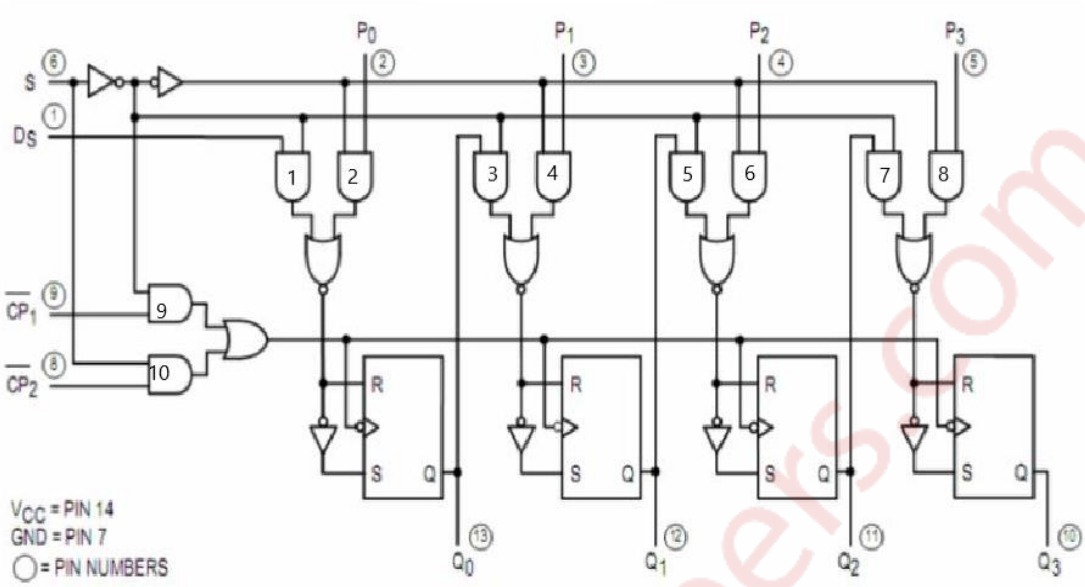
A,B,C,D are the inputs to four internal flip flops with A acting as LSB and D as MSB. QA through QD are the corresponding outputs.

IC 7495 is capable of performing the following operations:

1.Parallel loading (parallel input parallel output)

2.Left shifting

3.Right shifting

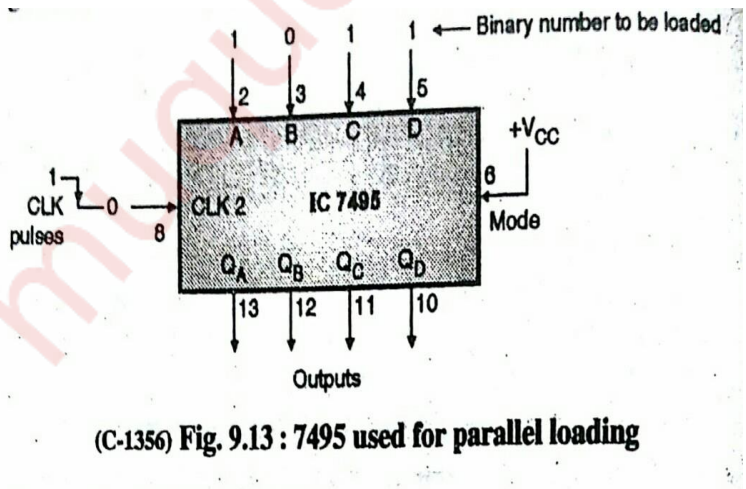


INTERNAL LOGIC DIAGRAM OF IC 7495

1.Parallel loading:

The connection diagram for IC 7495 in parallel input parallel output mode is as shown in the figure below.The mode control (M) is connected to logic 1. This will enable the AND gates 2, 4, 6, 8 as shown in internal logic diagram. The AND gates 1, 3, 5, 7 are disabled.This allows data transfer from the inputs A, B, C, D to the flip flops and disable
s the serial transfer of data. The 4-bit binary number which is to be loaded parallely is applied to the A, B, C, D inputs.

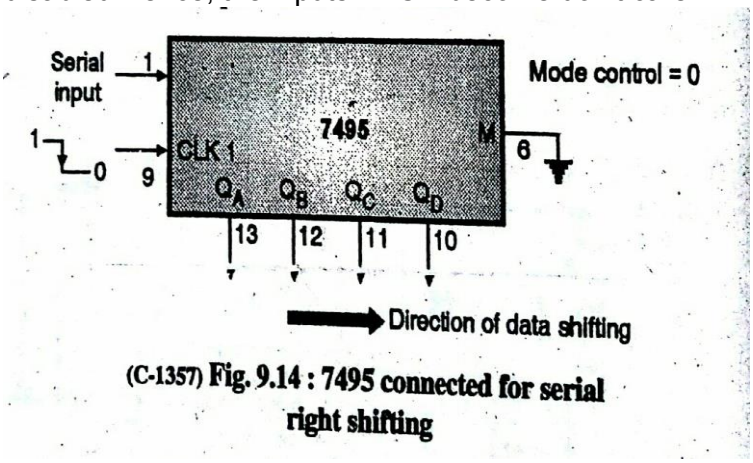
The clock applied at clock-2 input only will be paused through the flip flops because with M=1 and AND gate-10 is enabled and gate-9 is disabled. As soon as a falling edge of clock is applied, all flip-flops will change their status simultaneously and binary number applied to ABCD inputs will be loaded into the shift register. The unused inputs such as serial input and clock-1 can be left open or connected to ground because they are the don't care options for this mode.



(C-1356) Fig. 9.13 : 7495 used for parallel loading

2.Serial Shift Right Operation:

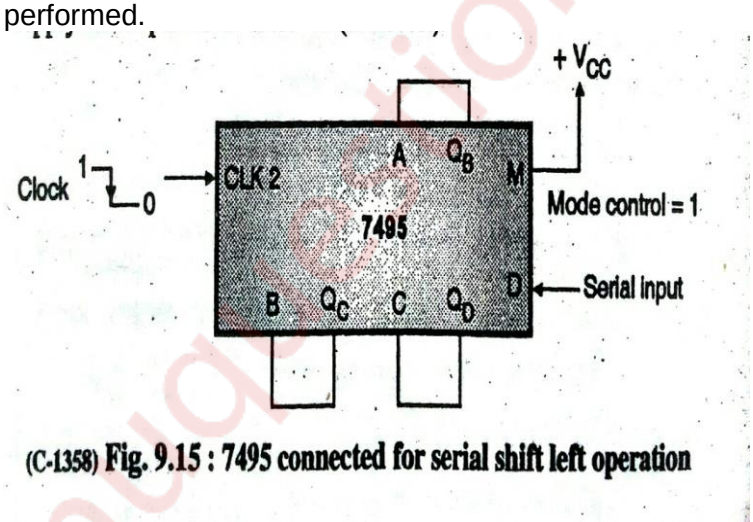
The connection for serial shift right mode is as shown in the figure below. Make mode control = 0, therefore AND gates 1, 3, 5, 7 will be enabled and AND gate 2, 4, 6, 8 will get disabled.Hence, the inputs ABCD become don't care.



(C-1357) **Fig. 9.14 : 7495 connected for serial right shifting**

3.Serial Shift Left Operation:

The connection diagram of 7495 for the shift left operation is as shown in the figure. Q_D is connected to C, Q_C to B and Q_B to A and the serial data is applied at input D.Mode control is connected to 1. Hence, the AND gates 2, 4, 6, 8 are enabled whereas 1, 3, 5, 7 are disabled. This will make the serial input (pin no. 1) a don't care input. The serial data is applied to D which will be routed through the enabled AND gates 2, 4, 6, 8 to facilitate the right shifting operation. As M = 1, AND gate - 10 is enabled and gate - 9 is disabled. So clock - 1 becomes a don't care input. Apply clock pulses to CLK - 2(shift left).Each high to low transition of clock will transfer data from D to Q_D, Q_D to Q_C , Q_C to Q_B ,Q_B to Q_A.Thus the shift left operation is performed.



(C-1358) **Fig. 9.15 : 7495 connected for serial shift left operation**

Q.6 Write Short notes on (any 2) **(20)**

(a)VHDL

Solution:

The long form of VHDL is Very High Speed Integrated Circuit (VHSIC) hardware description language.

VHDL is used to form a digital system at many levels of ideas ranging from algorithmic level to the gate level.

This language defines the syntax as well as simulation semantics for each language. It is a strong typed language which contains too many words to write.

VHDL is difficult to understand because it provides wide ranging of modelling capabilities but without learning the more complex features it is possible to incorporate a core subset of language which is simple and easy to understand.

Some Features of VHDL are:

1. Strongly typed language: Only LHS and RHS operators of the same type are allowed in VHDL.
2. Support hierarchies: Using VHDL, hierarchy can be represented. For example, full adder.In this case it is composed of half adder and OR gate.
3. VHDL supports for test and simulation of programs.
4. Concurrency: VHDL is a concurrent language which executes statements simultaneously in parallel.
5. VHDL supports different types of data modelling
   (i)Structural
   (ii)Data flow
   (iii)Behavioural
   (iv)Mixed

6. Supports sequential statement: VHDL can execute only one statement at a time in sequence only.
7. VHDL supports synchronous and asynchronous models.
8. VHDL can be used as a communication model between different CAD and CAE models.

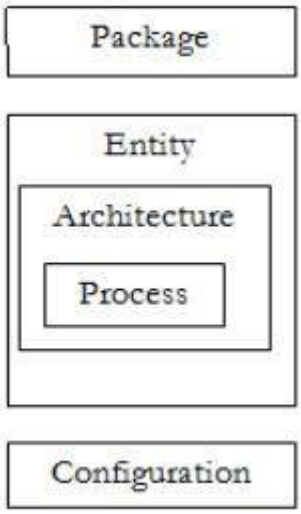
Structure of VHDL module:

Design units of VHDL code are independent components which are separately combined and stored in the library.

VHDL program is composed of the following design units:

1. Package (optional)
2. Entity
3. Architecture
4. Configuration (optional)

The diagram below shows the design units of VHDL.



Advantages of VHDL

1. VHDL allows designers to quickly develop designs requiring tens of thousands of logic gates.
2. VHDL supports multiple level of hierarchy and modular design methods.
3. VHDL allows user to pick any synthesis tool.
4. VHDL is multipurpose i.e. once calculation block is created then it can be used in many other projects.
5. For describing complex logic, VHDL provides powerful high level constructs.

Disadvantages of VHDL

VHDL is not a low level language i.e. gate level program. It is not suitable for verification of basic objects like gates. Because in VHDL these objects are readily available.

Applications of VHDL

1. It is used in electronic design automation to describe mixed signal system such as FPGA (Field Programmable Gate Arrays) and Integrated circuits.
2. VHDL can be used as a general purpose parallel programming language.
3. VHDL can also be used for design and simulation purposes.

(b)TTL and CMOS logic families:

TTL family:

The long form of TTL is Transistor Transistor Logic. It is a logic family consisting completely of transistors. It employs transistors with multiple emitters. The digital ICs in the TTL family use only transistors as their basic building block. TTL ICs were first developed in 1965 and they were known as “Standard TTL”. This version of TTL is not practically used now due to availability of advanced versions. Some characteristics of TTL family are as follows:

1. TTL is made up of BJTs (Bipolar Junction Transistor).
2. It has a high level noise margin of 0.4 V i.e. V_{NH} = 0.4 V.
3. It has a low level noise margin of 0.4 V i.e. V_{NL} = 0.4 V.
4. TTL family has less noise immunity than CMOS family.
5. The propagation delay of a standard TTL is 10nS.
6. It has a comparatively faster switching speed as compared to CMOS family.
7. It has a power dissipation of 10mW per gate i.e. P_D= 10mW.

8. TTL family has a speed power product of 100 pJ.
9. P_D of TTL does not depend on frequency.
10. TTL has a Fan Out capacity of 10.
11. In TTL, inputs can remain floating. The floating inputs are treated as logic 1s.
12. Since BJTs require more space, TTLs have a comparatively low component density.
13. Transistors are operated in saturated or cut off regions.
14. Power supply voltage of TTL is fixed which is equal to 5V.

CMOS family:

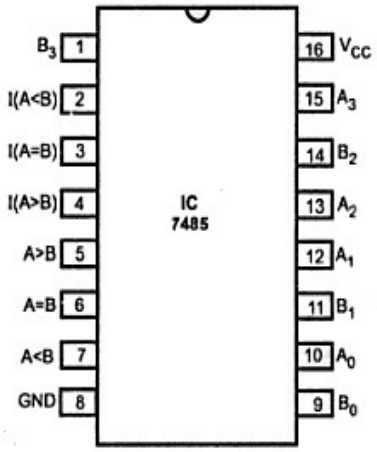
Complementary Metal Oxide Semiconductor (CMOS) is a technology for constructing integrated circuits, employing MOSFET transistors. CMOS technology is used in microprocessors, microcontrollers, static RAM, and other digital logic circuits. CMOS technology is also used for several analog circuits such as image sensors (CMOS sensor), data converters, and highly integrated transceivers for many types of communication. Some characteristics of CMOS logic family are as follows:

1. CMOS devices are made up of N-channel MOSFET and P-channel MOSFET.
2. It has a high level noise margin of 1.45 V i.e. V_{NH} = 1.45 V.
3. It has a low level noise margin of 1.45 V i.e. V_{NL} = 1.45 V.
4. CMOS family has better noise immunity than TTL family.
5. The propagation delay of a metal gate CMOS is 105nS.
6. It has a comparatively slower switching speed as compared to TTL family.
7. It has a power dissipation of 0.1 mW per gate i.e. P_D= 0.1 mW. Hence, it is used for battery backup applications.
8. CMOS family has a speed power product of 10.5 pJ.
9. P_D of CMOS depends on frequency.
10. CMOS has a Fan Out capacity of 50.
11. The unused inputs should be returned to GND or V_{DD}.They should never be left floating.
12. CMOS usually have a high component density than TTL as MOSFETs need less space while fabricating an IC.
13. MOSFETs are operated as switches i.e. in the ohmic region or cut off regions.
14. Power supply voltage of CMOS is flexible ranging from 3V to 15V.

(c)4-bit magnitude comparator

Solution:

A 4-bit comparator is used to compare two 4-bit words A (A_3- A_0) and B (B_3- B_0). IC 7485 is a four bit comparator in the integrated circuit form. It is possible to cascade more than one IC 7485 to compare words of almost any size.The figure below shows the pin configuration and the logic symbol of IC 7485.



(a) Pin diagram (IC 7485)

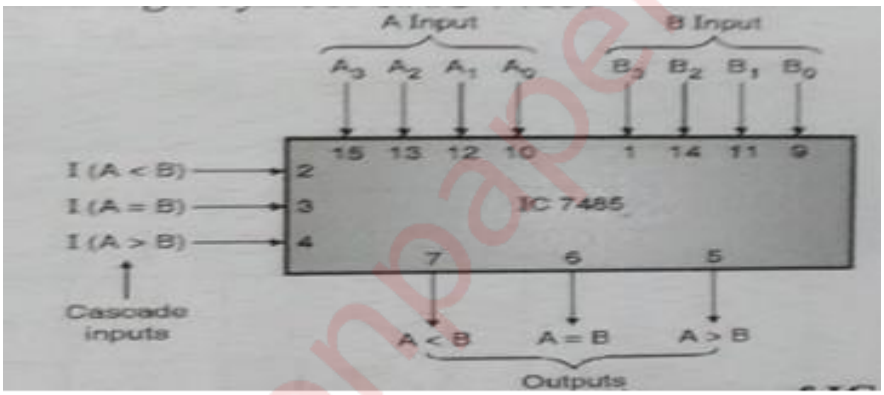


Fig13. Logic diagram of IC7485

The Pin names and their functions are as shown in the table below:

Pin name	Pin number	Function
A_0 to A_3	10,12,13,15	Binary input (opearand 1) Active high
B_0 to B_3	9,11,14,1	Binary input (opearand 2) Active high
I (A < B) I (A = B) I (A > B)	2 3 4	These lines are used for cascading a number of IC 7485 outputs of the previous stage are fed as inputs to this stage.
A < B A = B A > B	7 6 5	These are the outputs. When ICs 7485 are cascaded, these outputs are applied to cascading inputs of the next stage.

Comparing inputs				Cascading inputs			Outputs		
A3,B3	A2,B2	A1,B1	A0,B0	A>B	A<B	A=B	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	X	X	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L

DLDA

(CBCGS MAY 2019)

Q1] a) Convert (451.43)10 into octal, binary and hexadecimal and base 7. **(04)**

Solution :-

Decimal to octal

8	451	
8	56	3
8	7	0
	0	7

451 = 703

0.43 x 8 = 3.44 3
0.44 x 8 = 3.52 3
0.52 x 8 = 4.16 4
0.43 = (0.334)
$(451.43)_{10} = (703.334)_8$

Decimal to binary

2	451	
2	225	1
2	112	1
2	56	0
2	28	0
2	14	0
2	7	0
2	3	1
2	1	1
	0	1

451 = 111000011
0.43 x 2 = 0.86 0

0.86 x 2 = 1.72 1
0.43 = 0.01
$(451.43)_{10} = (111000011.01)_2$

Decimal to hexadecimal

16	451	
16	28	3
16	1	12
	0	1

451 = (1 12 3)
0.43 x 16 = 6.88 6
0.88 x 16 = 14.08 14
0.08 x 16 = 1.28 1
0.43 = (0.6 14 1)
$(451.43)_{10} = (1\ 12\ 3.6\ 14\ 1)_{16}$

Decimal to base 7

7	451	
7	64	3
7	9	1
7	1	2
	0	1

451 = (1 2 1 3)
0.43 x 7 = 3.01 3
0.01 x 7 = 0.07 0
0.43 = (0.30)
$(451.43)_{10} = (1213.30)_7$

Q1] b) Subtract using 1's and 2's complement method (73)10 – (49)10.

(04)

Solution :-

1's complement method
Binary representation of 49 is 0110001
1's complement of 49 is 1001110

```
  1 0 0 1 1 1 0
+ 1 0 0 1 0 0 1
---------------
  1 0 0 1 0 1 1 1
+               1
---------------
  1 0 0 1 1 0 0 0
```

Therefore **$(73)_{10}$ - $(49)_{10}$ = 11000 = $(24)_{10}$.**

2's complement method.

Binary representation of 49 is 0110001

1's complement of 49 is 1001110

2's complement of 49 is 1's complement + 1

```
  1 0 0 1 1 1 0
+             1
---------------
  1 0 0 1 1 1 1
```

Now adding 73 and 2's complement of 7

```
    1 0 0 1 0 0 1
+   1 0 0 1 1 1 1
-----------------
  1 00 1 1 0 0 0
```

Discard carry 1

Therefore **$(73)_{10}$ - $(49)_{10}$ = 11000 = $(24)_{10}$.**

Q1] c) Perform (52)10 - (68)10 in BCD using 9's complement.

(04)

Solution :-

First subtract 68 from 99 that is find 9's complement of 68

```
    9 9
-   6 8
-------
    3 1
```

We then add 52 and 9's complement of 68

```
    5 2
```

+ 3 1

8 3

(52)10 - (68)10 = (83)

Q1] d) State De Morgan's theorem. Prove OR-AND configuration is equivalent to NOR-NOR configuration. **(04)**

Solution :-

Theorem 1:

Statement: The complement of the product of two or more variables is equal to the sum of the components of the variables.
Logical expression: $\overline{A.B} = \overline{A} + \overline{B}$
NAND is equal to BUBBLED OR.

A	B	A.B	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

From the above truth table we can see that $\overline{A.B} = \overline{A} + \overline{B}$ Hence proved.

Theorem 2 :-

Statement: The complement of the sum of two or more variables is equal to the product of the complement of the variables
Logical expression: $\overline{A + B} = \bar{A}.\bar{B}$
NOR is equal to BUBBLED AND.

A	B	A + B	$\overline{A + B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

From the above truth table we can see that $\overline{A + B} = \bar{A}.\bar{B}$ Hence proved.

Q1] e) Encode the data bits 111010001 using Hamming code.

(04)

Solution :-

For given data of 9 bit, hamming code is given as

D13	D12	D11	D10	D9	P8	D7	D6	D5	P4	D3	P2	P1

Given 9 bit is 111010001 therefore it can be written as

1	1	1	0	1	P8	0	0	0	P4	1	P2	P1

P1: For P1 consider P1, D3, D5, D7, D9, D11, D13
D3, D5, D7, D9, D11, D13 = 100111
For even parity set P1 = 0
P2: For P2 consider P2, D3, D6, D7, D10, D11
D3, D6, D7, D10, D11 = 10001
For even parity set P2 = 0
P4: For P1 consider P4, D5, D6, D7, D12, D13
D5, D6, D7 = 00011
For even parity set P4 = 0
P8: For P1 consider P8, D9, D10, D11, D12, D13
D9, D10, D11, D12, D13= 10111
For even parity set P4 = 0

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

Therefore, hamming code for 111010001 is given as 1110100000100.

Q1] f) Explain SOP and POS and solve the following using K-Map
F(A,B,C,D)=πM(1,3,5,6,7,10,11)+d(2,4)

(04)

Solution :-

The sum-of-products (SOP) form is a method (or form) of simplifying the Boolean expressions of logic gates. In this SOP form of Boolean function representation, the variables are operated by AND (product) to form a product term and all these product terms are ORed (summed or added) together to get the final function.

The product of sums form is a method (or form) of simplifying the Boolean expressions of logic gates. In this POS form, all the variables are ORed, i.e. written as sums to form sum terms.

F(A,B,C,D)=πM(1,3,5,6,7,10,11)+d(2,4)

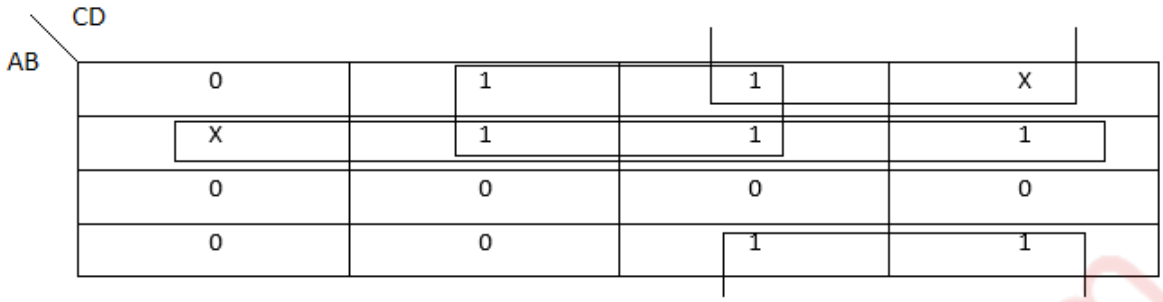


Fig.1.1 K map representation.

Equation form is

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

Q1] g) Explain lockout condition. How can it be avoided?

(04)

Solution :-

In counters, Lockout condition is that condition wherein a counter gets onto a forbidden state and rather than coming out of it to another acceptable state or initial state, the counter switches to another forbidden state and gets stuck up in the cycle of forbidden states only.
If by chance, the counter happens to find itself in any one of the unused states, its next state would not be known. It may just be possible that the counter might go from one unused state to another and never arrive at a used state. A counter whose unused states have this feature is said to suffer from LOCK OUT.

To avoid lock out and make sure that at the starting point the counter is in its initial state or it comes to its initial state within few clock cycles, external logic circuitry is to be provided and so we design the counter assuming the next state to be the initial state, from each unused states.
In order to prevent lock out condition, use PRESET option and clear option in a flip flop and hence the registers will be cleared and we can go for the smooth functioning of counter circuits.

Q2] a) Reduce equation using Quine McCluskey method and realize circuit using (10) basic gates.
-F(A,B,C,D) = Σm (1,5,6,12,13,14) + d(2,4)

(10)

Solution :-

Table 1

Group	Minterm	Binary Representation			
		A	B	C	D
0	m1	0	0	0	1
	m2*	0	0	1	0
	m4*	0	1	0	0
1	m5	0	1	0	1
	m6	0	1	0	1
	m12	0	1	1	0
2	m13	1	1	0	1
	m14	1	1	1	0

Table 2

Group	Minterm	Binary Representation			
		A	B	C	D
0	m1-m5	0	_	0	1
	m2-m6*	0	_	1	0
	m4-m5*	0	1	0	_
	m4-m12*	_	1	0	0
1	M5-m13	_	1	0	1
	M6-m14	_	1	1	0
	M12-m13	1	1	0	_
	M12-m14	1	1	_	0

Table 3

Group	Minterm	Binary Representation			
		A	B	C	D
0	M4-m5-m12-m13*	_	1	0	_
	M4-m12-m5-m13*	_	1	0	_
	M4-m12-m6-m14*	_	1	_	0

PI	Decimal of PI	1	5	6	12	13	14	2	4
$B\bar{C}$	M4 m5 m12 m13		⊗		X	⊗			X
$B\bar{D}$	M4 m6 m12 m14			⊗	X		⊗		X

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

Q2] b) Design 4-bit BCD subtractor using IC 7483.

(10)

Solution :-

4 Bit BCD subtractor

IC 7483 performs the addition of two 4-bit BCD numbers A3 A2 A1 A0 and B3 B2 B1 B0 and carry input to give the output S3 S2 S1 S0 and carry out. So for adding the two numbers A3 A2 A1 A0 and B3 B2 B1 B0, the two numbers are given to input terminals 1,3,8,10 and 16,4,7,11 of the IC 7483 and carry in the terminal 13 is set to zero. To subtract two numbers by two's complement method, we are adding the 2's complement of the second number to each of the four bits of the first numbers. The final carry is neglected and the difference is taken from S3 S2 S1 S0.

In the circuit we set mode control such that when the mode control is zero, addition is performed and subtraction is performed when the mode control is one. We use XOR gates to feed the input so that when control is one, the complement of each of the four bits are fed and mode control is zero, the input as such is fed.

In 1's complement subtraction, the complement of the subtraction is taken and added with the other number. The final carry is then added to LSB of the result. In case there is no carry, the complement of the result is taken and this will be a negative number. This indicates that, subtraction is performed from a smaller number.

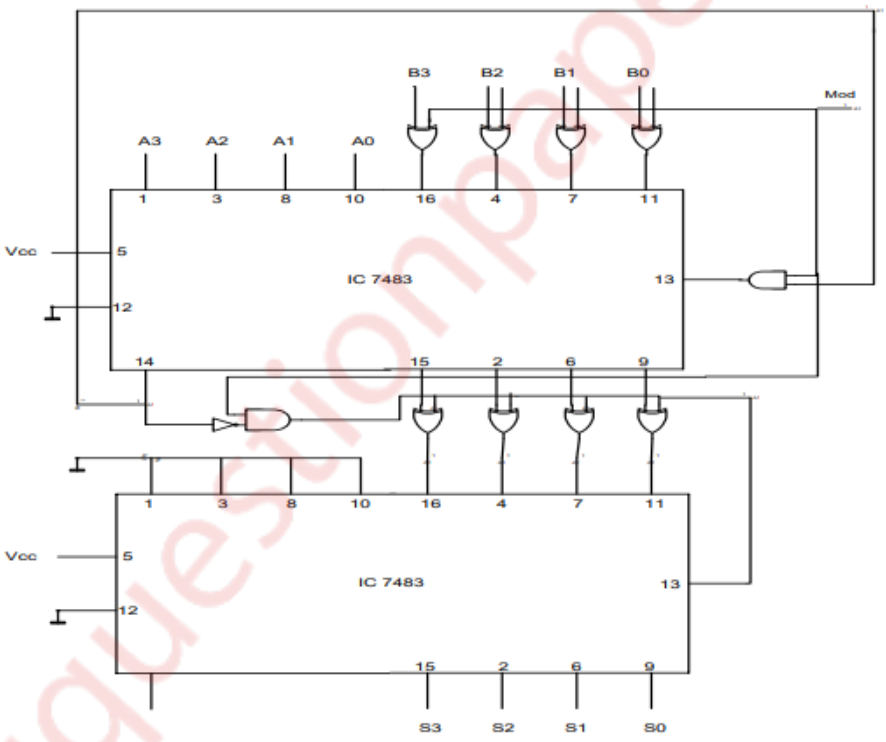


Fig 2 - 4 bit BCD subtractor using IC 7483

Q3] a) Implement the following using only one 8:1 Mux.
F(A,B,C,D) = Σm (0,2,4,6,8,10,12,14)

(05)

Solution :-

Step 1: Draw a table

	D0	D1	D2	D3	D4	D5	D6	D7
A	⓪	1	②	3	④	5	⑥	7
A	⑧	9	⑩	11	⑫	13	⑭	15

If both the values in the column are included than it is connected to 1. If both the values of the column are not included than it is connected to input 0. So, for the given question the mux drawn is given below.

Step 2: MUX implementation

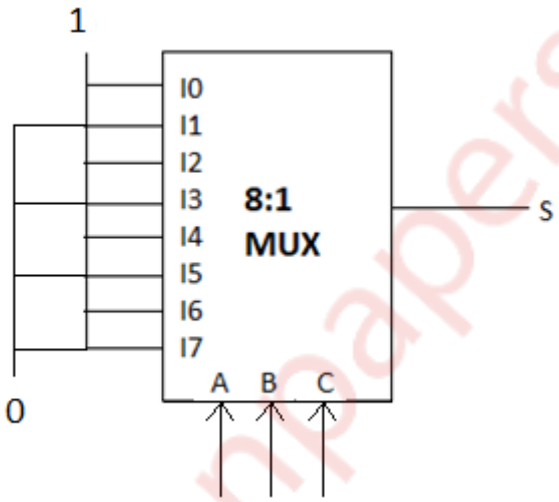


Fig. 3a 8:1 MUX

Q3] b) Design a Full Subtractor using only NAND gates.

(05)

Solution:-

Truth table of full subtractor

X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0

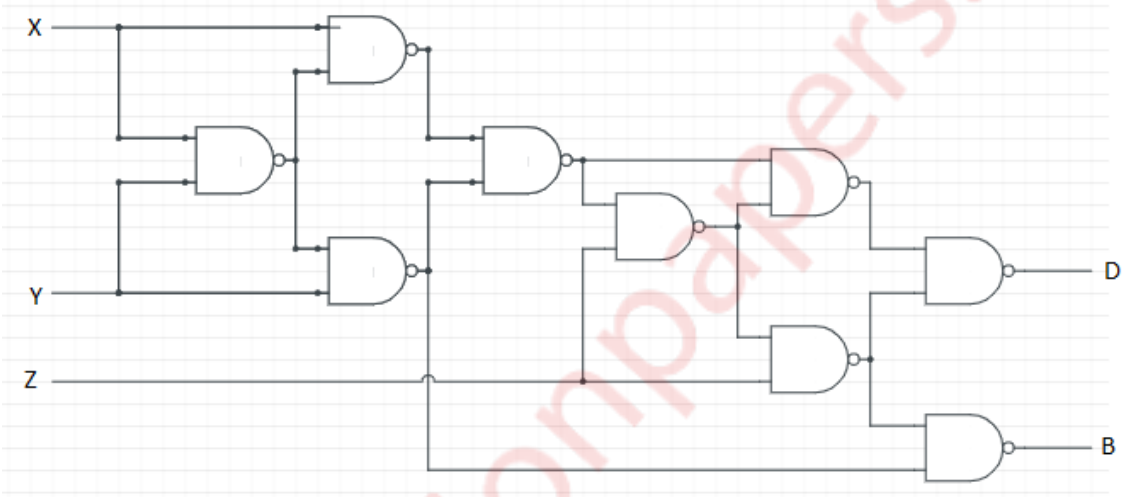
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$D = X \oplus Y \oplus Z$

X \ YZ	00	01	11	10
0	0	1	1	1
1	0	0	1	0

$B = \overline{X}Y + \overline{X}\,\overline{Y}\,Z + XYZ$

Representation of full subtractor using NAND gates.



Q3] c) Design a logic circuit to convert 4-bit gray code to its corresponding BCD code. **(10)**

Solution:-

Step1: Write the truth table of Gray code and its equivalent binary code is as shown in table:

Gray Code Input				Binary Code Output			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

Step2: Write K – map for each binary output and get simplified expression The K – map for various binary outputs and the corresponding simplified expression are given below:

For output B3

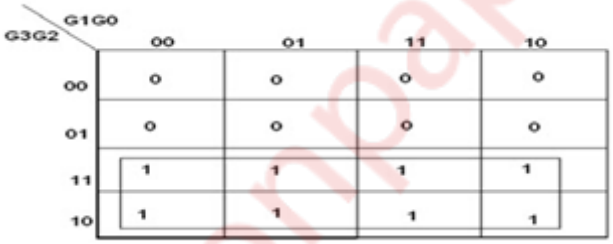


Fig.3.1 K map for B3

Simplified expression B3=G3
For output B2

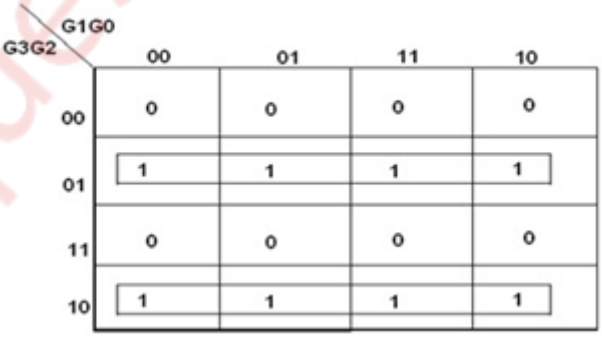


Fig 3.2 K map for B2

B2 = G(3)($\overline{G2}$) + G(2)($\overline{G3}$)
=G3 $\oplus$ G2

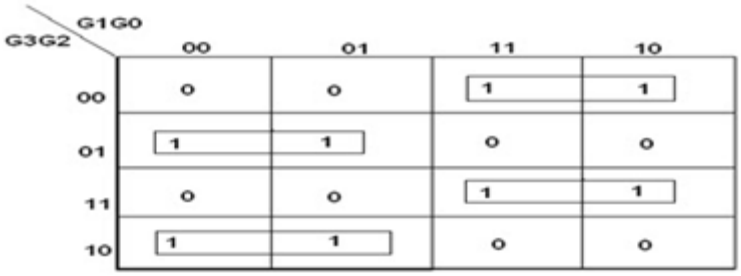


Fig 3.3 K map for B1

B1 = G1 $\overline{G2\ G3}$ + G2 $\overline{G3\ G2}$ + G3 G2 G1 + G3 $\overline{G1\ G2}$

= $\overline{G1}$ (G2⊕G3) + G1 (G2⊕G3)

= $\overline{G1}X + \overline{G1}X$ Where X = G2 ⊕ G3

= G1 ⊕ G2 ⊕ G3

For output B0

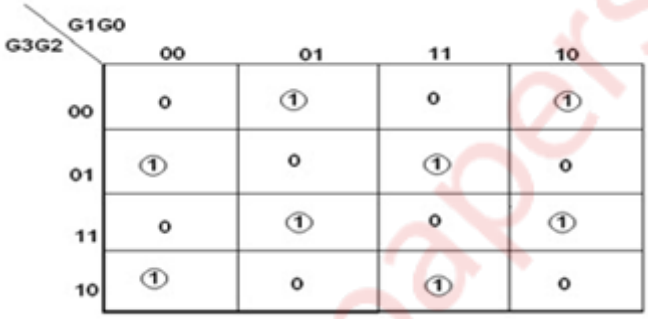


Fig 3.4 K map for B0

B0 = G0 $\overline{G1\ G2\ G3}$ + G2 $\overline{G3\ G1\ G0}$ + G1 $\overline{G3\ G2\ G0}$ + G1 $\overline{G3\ G2\ G0}$ + G2 $\overline{G3}$ G1 G0 + G2 $\overline{G1}$ G3 G0 + G2 $\overline{G0}$ G3 G1 + G3 $\overline{G1\ G2\ G0}$ + G0 $\overline{G2}$ G3 G1

= $\overline{G0\ G1}$ (G2⊕G3) + G0 $\overline{G1}$ ($\overline{G3 \oplus G2}$) + G0 $\overline{G1}$ ($\overline{G3 \oplus G2}$) + G1 $\overline{G0}$ ($\overline{G3 \oplus G2}$)

= (G2⊕G3) + ($\overline{G1 \oplus G0}$) +(G1⊕G0) + ($\overline{G3 \oplus G2}$)

= $\overline{Y}$ X + Y $\overline{X}$

Where X = (G2⊕G3) and Y=(G1⊕G0)

= (G2⊕G3) ⊕ (G1⊕G0)

= G2⊕G3⊕G1⊕G0

Step3: Realization

The Gray to Binary code converter is as shown in figure given below

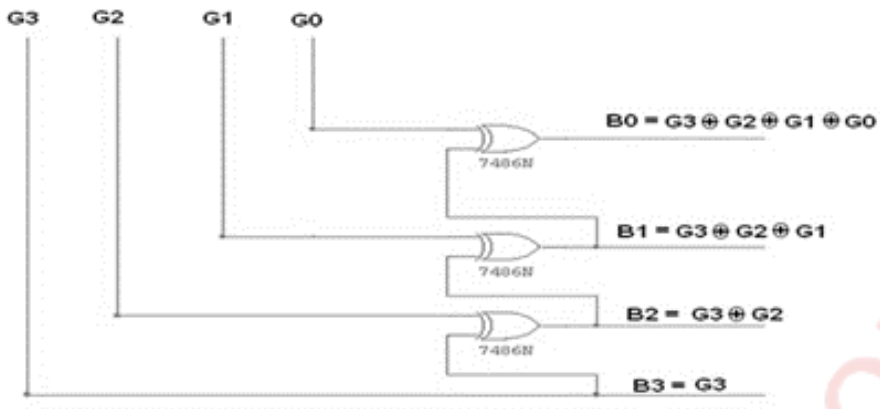


Fig. 3.5 Gray code to binary converter

Q4] a) Compare different logic families with respect to fan in, fan out, speed, Propogation delay and power dissipation.

(05)

Solution :-

Parameter	ECL	DTL	TTL	IIL	CMOS
Fan in	-	-	12-14	-	Greater than 10
Fan out	25	Medium (8)	High (10)	8-11	50
Speed X Power	100	300	100	Less than 1	70
Propagation Delay (ns)	1-2	30	10	25-2500	70
Power Dissipation (mW)	40-45	8-12	10	5-25	01

Q4] b) Implement 3 bit binary to gray code converter using Decoder.

(05)

Solution :-

Truth table:-

A	B	C	X	Y	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Formulae:-

Let b2 b1 b0 be the 3-bit binary number and g2 g1 g0 be its equivalent gray code.

Then,

$g2 = b2$

$g1 = b2 \oplus b1$

$g0 = b1 \oplus b0$

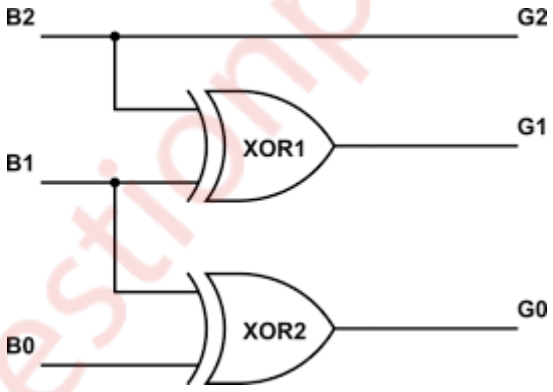


Fig. 4.1 Binary to gray code converter.

Q4] c) Explain 4 bit bidirectional shift register.

(10)

Solution :-

- The binary information (data) in a register can be moved from stage to stage within the register or in or out of the register upon applications of clock pulses.
- This type of bit movement or shifting is essential for certain application like arithmetic and logic operations used in Microprocessors.

- This group of registers are called as 'Shift Register'

4 –bit Bidirectional Shift Register:

- Bidirectional shift register allows shifting of data either to left or to the right side.
- It can be implemented using logic gates circuitry that enables the transfer of data from one stage to the next stage to the right or to the left, depend on the level of control line.
- The RIGHT/LEFT is the control input signal which allows data shifting either towards right or towards left.
- A high on this line enables the shifting of data towards right and low enables it towards left.
- When RIGHT/LEFT is high, gates G1, G2, G3 and G4 are enabled.
- The state of Q output of each flip flop is passed through the D input of the following flip flop.
- When the pulse arrives, the data are shifted one place to the right.
- When the RIGHT/LEFT signal is low, gates G5, G6, G7 are enabled.
- The Q output of each flip-flop is passed through the D input of the preceding flip-flop.

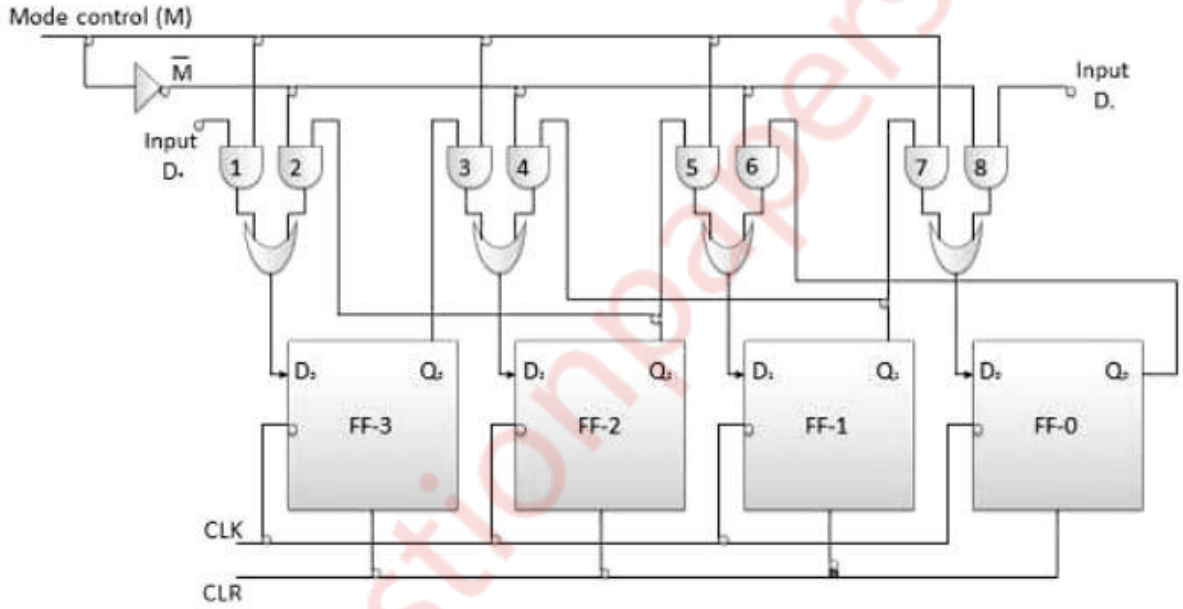


Fig.4.s2 - 4 bit Bidirectional Shift Register

Q5] a) Design mod 13 synchronous counter using T flipflop.

(10)

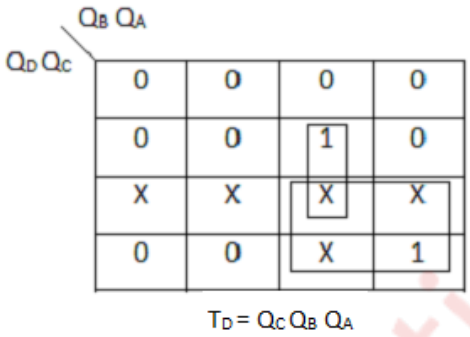
Solution :-

A synchronous counter is one which has the same clock input for all its flip flops. A MOD 13 synchronous counter counts from 0000 to 1101. Hence it will require four T flip flops. Synchronous counters are designed by using excitation table to determine the combinational logic of inputs to each flip flop. The excitation table for all the four T flip flops is shown:

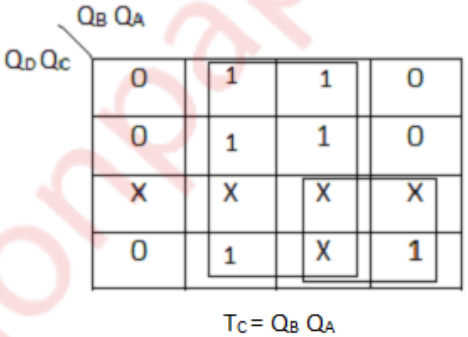
Present state				Next state				T Flipflop			
Q_D	Q_C	Q_B	Q_A	Q_D^+	Q_C^+	Q_B^+	Q_A^+	T_D	T_C	T_B	T_A
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	1	0	1	0	0	0	1	1
1	0	1	0	0	0	0	0	1	0	1	0
1	0	1	1	X	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X

Fig 5.a Excitation table for MOD 13 synchronous counter

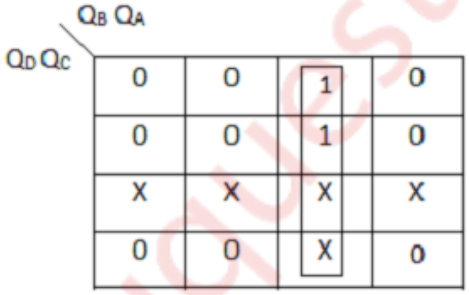
From the above excitation table, we can draw k maps to determine input to every flipflop



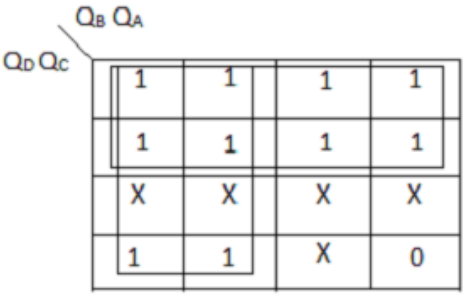
$T_D = Q_C\,Q_B\,Q_A$



$T_C = Q_B\,Q_A$



$T_B = Q_A + Q_D\,Q_B$



$T_A = \overline{Q_D} + \overline{Q_B}$

Four equations for four T flipflop are obtained. Using them, the MOD 13 synchronous counter is designed as follows:

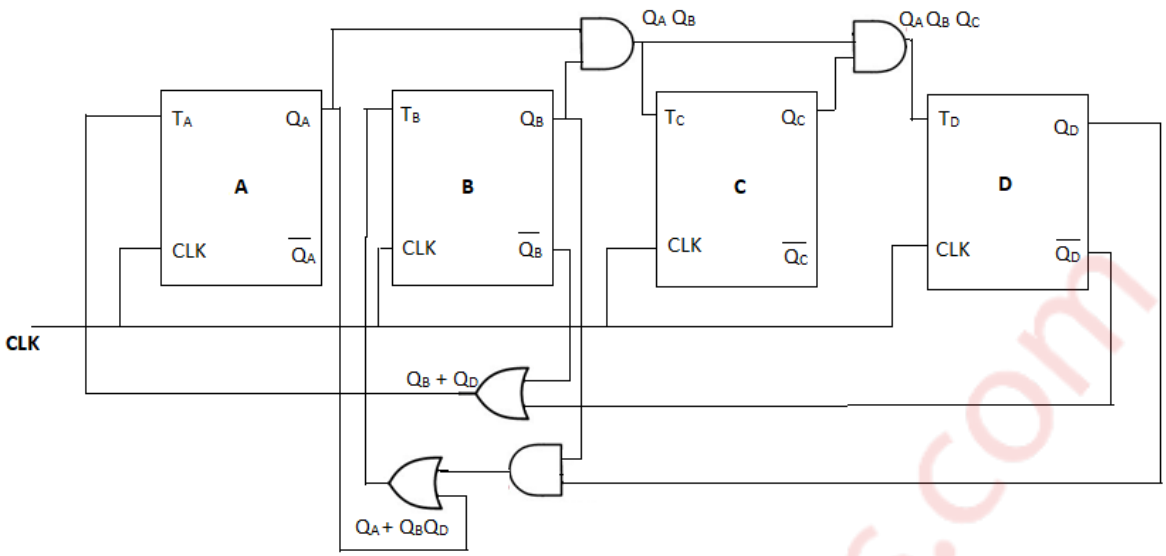


Fig 5.2 MOD 13 synchronous counter using T flipflop

Q5 b) Convert SR flipflop to JK flipflop and D flipflop.

(10)

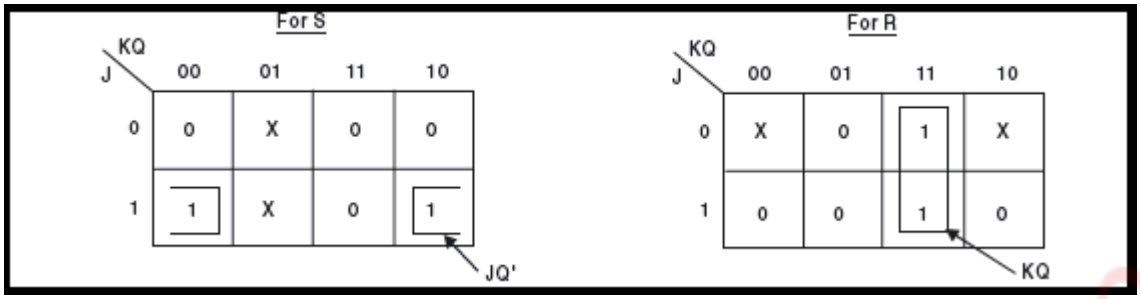
Solution:-

SR to JK

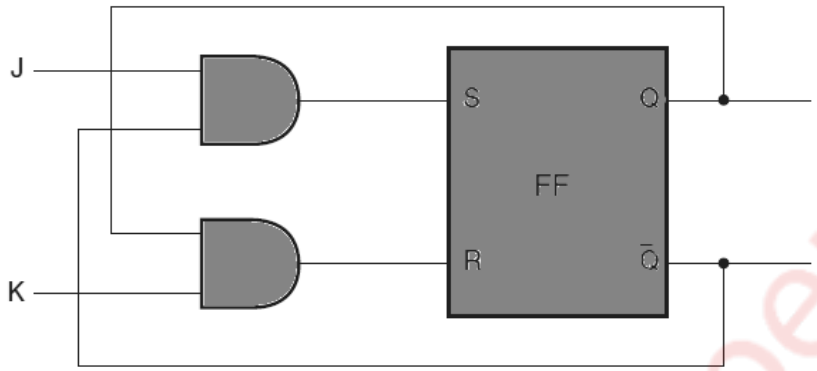
a) Excitation Table

Input's		Present State	Next State	Flip Flop Input	
J	K	Qn	Qn+1	S	R
0	0	0	0	0	X
0	1	0	0	0	X
1	0	0	1	1	0
1	1	0	1	1	0
0	0	1	0	0	1
0	1	1	0	0	1
1	0	1	1	X	0
1	1	1	1	X	0

b) K- map simplification



c) Logic diagram

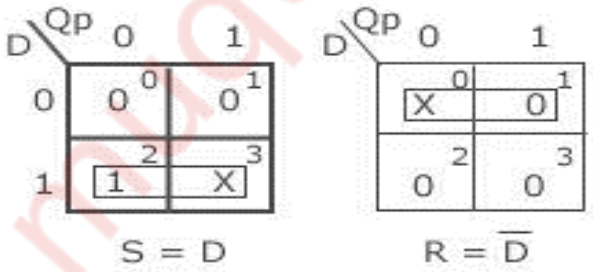


SR to D

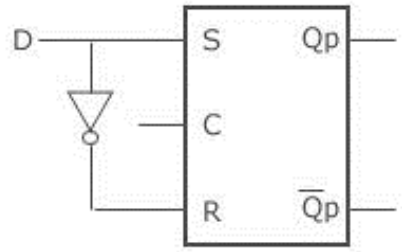
a) Excitation Table

Input	Present State	Next State	Flip Flop Input	
D	Qn	Qn+1	S	R
0	0	0	0	X
1	0	1	1	0
0	1	0	0	1
1	1	1	X	0

b) K-map simplification



c) Logic diagram



Q6] a) ALU

(05)

Solution :-

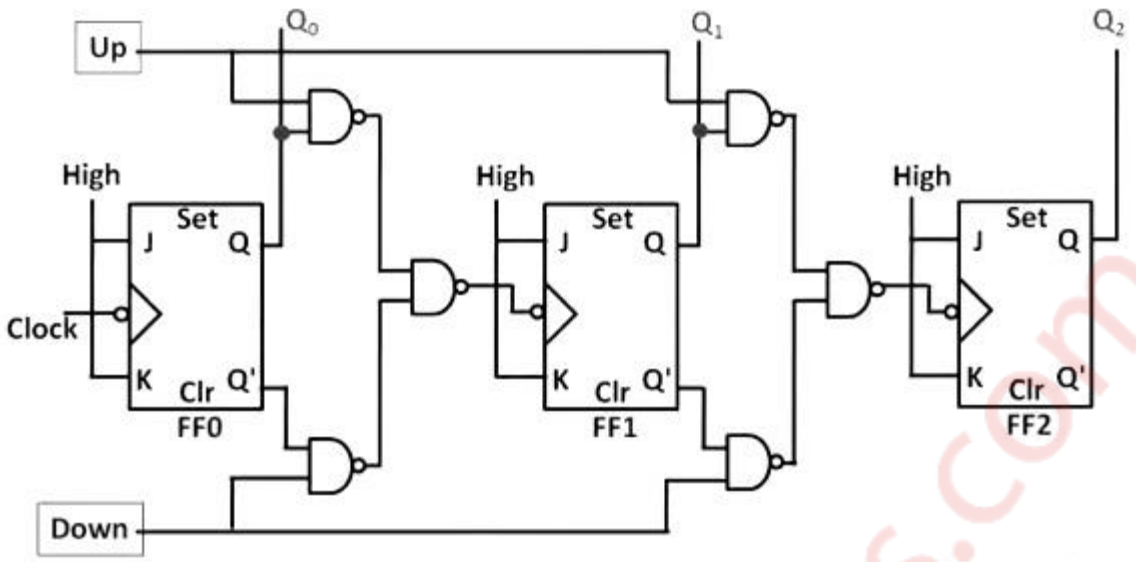
The arithmetic logic unit is that part of the CPU that handles all the calculations the CPU may need. Most of these operations are logical in nature. Depending on how the ALU is designed, it can make the CPU more powerful, but it also consumes more energy and creates more heat. Therefore, there must be a balance between how powerful and complex the ALU is and how expensive the whole unit becomes. This is why faster CPUs are more expensive, consume more power and dissipate more heat.
The main functions of the ALU are to do arithmetic and logic operations, including bit shifting operations. These are essential processes that need to be done on almost any data that is being processed by the CPU.
ALUs routinely perform the following operations:
Logical Operations: These include AND, OR, NOT, XOR, NOR, NAND, etc.
Bit-Shifting Operations: This pertains to shifting the positions of the bits by a certain number of places to the right or left, which is considered a multiplication operation.
Arithmetic Operations: This refers to bit addition and subtraction. Although multiplication and division are sometimes used, these operations are more expensive to make. Addition can be used to substitute for multiplication and subtraction for division.

Q6] b) 3 bit Up/Down Asynchronous Counter

(05)

Solution :-

Asynchronous counters are those whose output is free from the clock signal. Because the flip flops in asynchronous counters are supplied with different clock signals, there may be delay in producing output.
By adding up the ideas of UP counter and DOWN counters, we can design asynchronous up /down counter. The 3 bit asynchronous up/ down counter is shown below.



It can count in either ways, up to down or down to up, based on the clock signal input.
UP Counting
If the UP input and down inputs are 1 and 0 respectively, then the NAND gates between first flip flop to third flip flop will pass the non-inverted output of FF 0 to the clock input of FF 1. Similarly, Q output of FF 1 will pass to the clock input of FF 2. Thus, the UP /down counter performs up counting.
DOWN Counting
If the DOWN input and up inputs are 1 and 0 respectively, then the NAND gates between first flip flop to third flip flop will pass the inverted output of FF 0 to the clock input of FF 1. Similarly, Q output of FF 1 will pass to the clock input of FF 2. Thus, the UP /down counter performs down counting.
The up/ down counter is slower than up counter or a down counter, because the addition propagation delay will be added to the NAND gate network.

Q6] c) Octal to Binary Encoder

(05)

Solution:-

An encoder has 2n or fewer input lines, only one of which is in the "1" state at a particular time and an n-bit code is generated on "n" output lines depending upon which of the input is excited. In other words, an encoder is a circuit in which output lines generate the binary code corresponding to the input value.
An octal to binary encoder has 23 = 8 input lines D0 to D7 and 3 output lines Y0to Y2. Below is the truth table for an octal to binary encoder.

Input								Output	
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Fig.6.c Truth Table of octal to binary encoder.

From the truth table, the outputs can be expressed by following Boolean Function.
Y0 = D1 + D3 + D5 + D7
Y1 = D2 + D3 + D6 + D7
Y2 = D4 + D5 + D6 + D7
Above boolean functions are formed by ORing all the input lines for which output is 1. For instance Y0 is 1 for D1, D3, D5, D7 input lines. The encoder can therefore be implemented with OR gates whose inputs are determined directly from truth table.

Q6] d) 4-bit Universal shift register

(05)

Solution :-

A Universal shift register is a register which has both the right shift and left shift with parallel load capabilities. Universal shift registers are used as memory elements in computers. A Unidirectional shift register is capable of shifting in only one direction. A bidirectional shift register is capable of shifting in both the directions. The Universal shift register is a combination design of bidirectional shift register and a unidirectional shift register with parallel load provision.
The register operations performed for the various inputs of select lines are as follows:

S1	S0	REGISTER OPERATION
0	0	No changes
0	1	Shift Right
1	0	Shift Left
1	1	Parallel Load

4-bit universal shift register is built with four blocks each constituted of a 4X1 mux and a D-flipflop. All the blocks are essentially identical. Because all the multiplexers in the register are wired similarly,

The L inputs come through port 11, which is why the L inputs are readable only when S1S0 = 11. The feedback Q wires are connected at port 00, so that when S1S0 = 00 the output Q of the D-flipflops

feed back into the flipflops' inputs resulting in no total change in the register content. Port 01 is wired to facilitate right-shifts. In mode S1S0 = 01 only port 01 is active and it takes its value from the previous more significant flipflop and passes it down to the flipflop wired to its mux output. Lastly port 10 is wired to conduce to left-shifts. Being the only active port when S1S0 = 10, it remits the output of the less significant flipflop sourcing into it to the flipflop wired to its mux output. As a consequence of this wiring pattern where each block of the register is an exact replica of any other block, the selector switches are able to align the behavior of all the multiplexers simultaneously. This coincidence of behavior is what we refer to as mode behavior of the universal register.

Q6] e) VHDL

(05)

Solution :-

VHDL stands for very high-speed integrated circuit hardware description language. This language was first introduced in 1981 for the department of Defense (DoD) under the VHSIC program.

VHDL is a hardware description language that can be used a model a digital system at many levels of abstraction, ranging from the algorithmic level to the gate levels.

Structure of VHDL Module:

The Main component of VHDL module consist of following declaration.

1) Package
2) Entity
3) Architecture
4) Configuration

The following fig. shows the relationship of these basic blocks of VHDL program.

A design may include any number of packages, entity, and Architecture and configuration declaration.

The entity and architecture blocks are compulsory but packages and configuration blocks are optional.

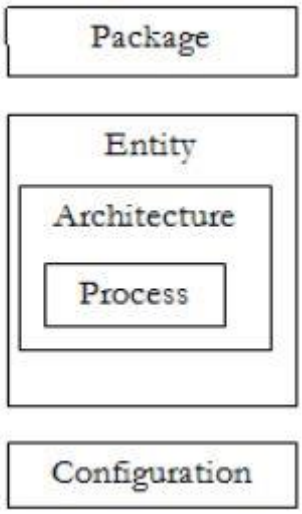


Fig 6.e Relationship of VHDL Design units

Features of VHDL are as follows:

1) Designs are organized hierarchically.
2) Each design element has:
   i) A well-defined interface.
   ii) A precise behavioral specification using either algorithmic description or hardware structural description.
3) Models concurrency, timing, and clocking:
   i) Handles asynchronous and synchronous circuits
   ii) Designs can be simulated.
4) Language is not case sensitive.
5) A formal language for specifying the behavior and structure digital circuit.

Q.P CODE:76197

MUMBAI UNIVERSITY

DIGITAL LOGIC DESIGN & ANALYSIS

SEMESTER 3 – CBCGS – DECEMBER 2019

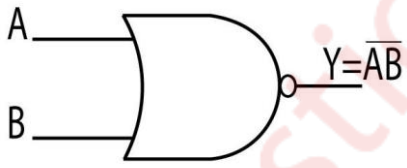
Q.1 a) What are Universal gates? Why they are called so ?Explain with suitable Example . [4M]

Ans : i) A universal gate is a gate which can implement any Boolean function without use any other gate type.

ii) The NAND and NOR gates are universal gates.

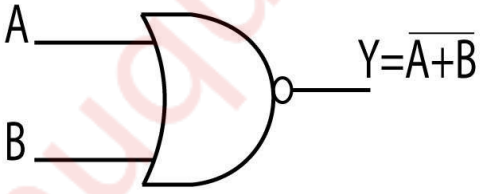
iii) AND and NOR are called universal gates because all the other gates like and,or,not,xor and xnor can be derived from it.

iv) NAND GATE :



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

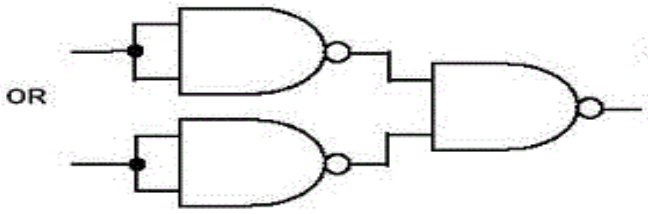
NOR GATE :



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

v) Example : OR gate using universal gate i.e NAND gate .

OR gate can be implemented using three NAND gate .



OR GATE USING NAND

b) Perform following substractions using 7's complement method.

a) $(\mathbf{20})_{\mathbf{5}}$ - $(\mathbf{14})_{\mathbf{5}}$

b) $(\mathbf{20})_{\mathbf{10}}$ - $(\mathbf{15})_{\mathbf{10}}$ [4M]

Ans : a) The given numbers are of base 5 .We should convert it into octal for Substraction process.

Using conversion table ,

$(20)_5 = (12)_8$ & $(14)_5 = (11)_8$

Here A = 12, B = 11.
Find A - B = ? using 7's complement
First find 7's complement of B = 11

Note : 7's complement of a number is obtained by subtracting all bits from 77.
7's complement of 11 is 77-11 = 66

Add it in A i.e in 12 ,

12 + 66 = 100

Here in 7s complement substraction we add carry in LSB

$$\therefore 00 + 1 = 01$$

$$\therefore (20)_5 - (14)_5 = (01)_8$$

b) let A = 20 and B = 15 are decimal value .

We need octal numbers for the 7s complement of any number.

$\therefore (20)_{10} = (24)_8$ & $(15)_{10} = (17)_8$

Applying 7's complement method on B ,

77 – 17 = 60 in octal

Add this number in A

$\therefore (24)_8 + (60)_8 = (104)_8$ Here carry is 1 . Add carry to LSB

$\therefore (20)_{10} - (15)_{10} = (5)_{10}$

c) Perform $(34)_{10} - (12)_{10}$ in BCD using 10's complement method . [4M]

Ans :

Here A = 34, B = 10.
Find A - B = ? using 10's complement
First find 10's complement of B = 10

Note : 10's complement of a number is 1 added to it's 9's complement number.
9's complement of 10 is

9 9
\- 1 0
_____
8 9

Now add 1 : 89 + 1 = 90

Now Add this 10's complement of B to A

$$\begin{array}{rrr} & 3 & 4 \\ + & 9 & 0 \\ \hline 1 & 2 & 4 \end{array}$$

Here carry is 1 ; carry is ignored .,

$$(34)_{10} - (12)_{10} = (24)_{10}$$

d) Explain lockout condition.How can it be avoided ? [4M]

Ans : i) Sometimes a counter may find itself in some unused states, this happen when if next state of some unused state is again some unused one and if by chance the counter happens to find itself in some unused state and never arrives at in used state then this condition is called “Lock Out”.

ii) To avoid lock out condition the unused states are introduced in front of used states 1,4,5,6 and 7.

iii) An additional circuit is require to ensure that “lock out” does not occur. The counter should be designed using the next state to be initial state from the unused state.

iii)An additional circuit is require to ensure that “lock out” does not occur. The counter should be designed using the next state to be initial state from the unused state.

iv) To avoid lock out condition the unused states are introduced in front of used states from the above state diagram the 1,4,5,6 and 7 is the sequence and unused states are 0,3 and 6 the states are introduced in front of us States 1,4,5 and 7 respectively.

e) If the 7 bit hamming code word received by receiver is 1011011,assuming the even parity,state whether the received code word is correct or not. If wrong locate the bit having error and extract corrected data. [4M]

Ans : The received data is 1101101

2^k -1 >= m+k ,

2^3 – 1 >= 4+3 ,

7=7

Odd C1=1011

Even C2 = 1001

Odd C4 =1101

Bit error = 1+4 = 5

The correct data 1001011

Q.2 a) Reduce using Quine McClusky Method & realize the operation using NOR gates only.

F = $\sum m(0, 1, 2, 8, 10, 11, 14, 15)$ **[10M]**

Ans : The given function contains min terms and truth table and Quine McClusky method is given

Quine McClusky Method :

Finding all prime implicants of the function.Use those prime implicants in a prime implicant chart to find the essential prime implicants of the function, as well as other prime implicants that are necessary to cover the function

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

∴ **y = B'D' + AC + A'B'C'**

Column 1

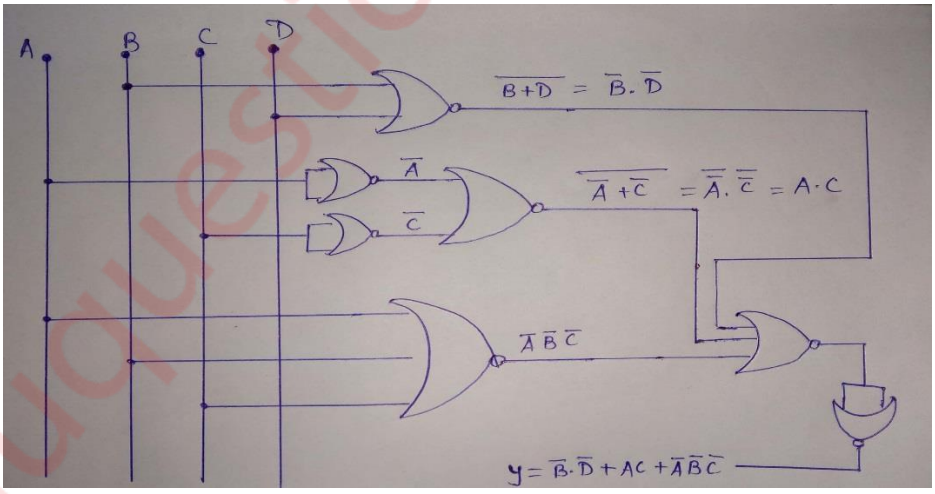
0	0000	✓
1	0001	✓
2	0010	✓
8	1000	✓
10	1010	✓
11	1011	✓
14	1110	✓
15	1111	✓

Column 2

(1,0)	000-	
(2,0)	00-0	✓
(8,0)	-000	✓
(10,2)	-010	✓
(10,8)	10-0	✓
(11,10)	101-	✓
(14,10)	1-10	✓
(15,11)	1-11	✓
(15,14)	111-	✓

0	(10,8,2,0)	-0-0
1	—	—
2	(15,14,11,10)	1-1-

Implementation :



b) Explain one digit BCD Adder . [10]

Ans : I)A BCD adder adds two BCD digits and produces output as a BCD digit. A BCD or Binary Coded Decimal digit cannot be greater than 9.

II)The two BCD digits are to be added using the rules of binary addition.

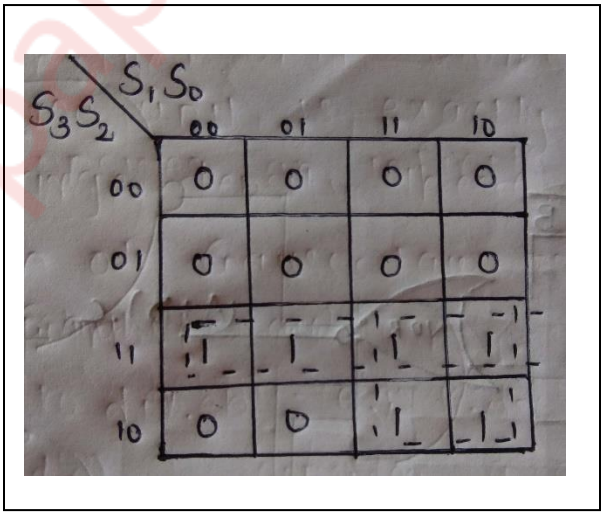
III) If sum is less than or equal to 9 and carry is 0, then no correction is needed. The sum is correct and in true BCD form.

IV) But if sum is greater than 9 or carry =1, the result is wrong and correction must be done. The wrong result can be corrected adding six (0110) to it.

V) For implementing a BCD adder using a binary adder circuit IC 7483, additional combinational circuit will be required, where the Sum output S3−S0S3−S0 is checked for invalid values from 10 to 15. The truth table and K-map for the same is as shown:

I/P				O/P
S3	S2	S1	S0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Table1. Truth table for BCD numbers



VI)The Boolean expression is, Y=S3S2+S3S1Y=S3S2+S3S1

VII)The BCD adder is shown below. The output of the combinational circuit should be 1 if Cout of adder-1 is high. Therefore Y is ORed with Cout of adder 1.

VIII)The output of combinational circuit is connected to B1B2 inputs of adder-2 and B3=B1+0B3=B1+0 as they are connected to ground permanently. This makes B3B2B1B0B3B2B1B0 = 0110 if Y' = 1.

IX) The sum outputs of adder-1 are applied to A3A2A1A0A3A2A1A0 of adder-2. The output of combinational circuit is to be used as final output carry and the carry output of adder-2 is to be ignored.

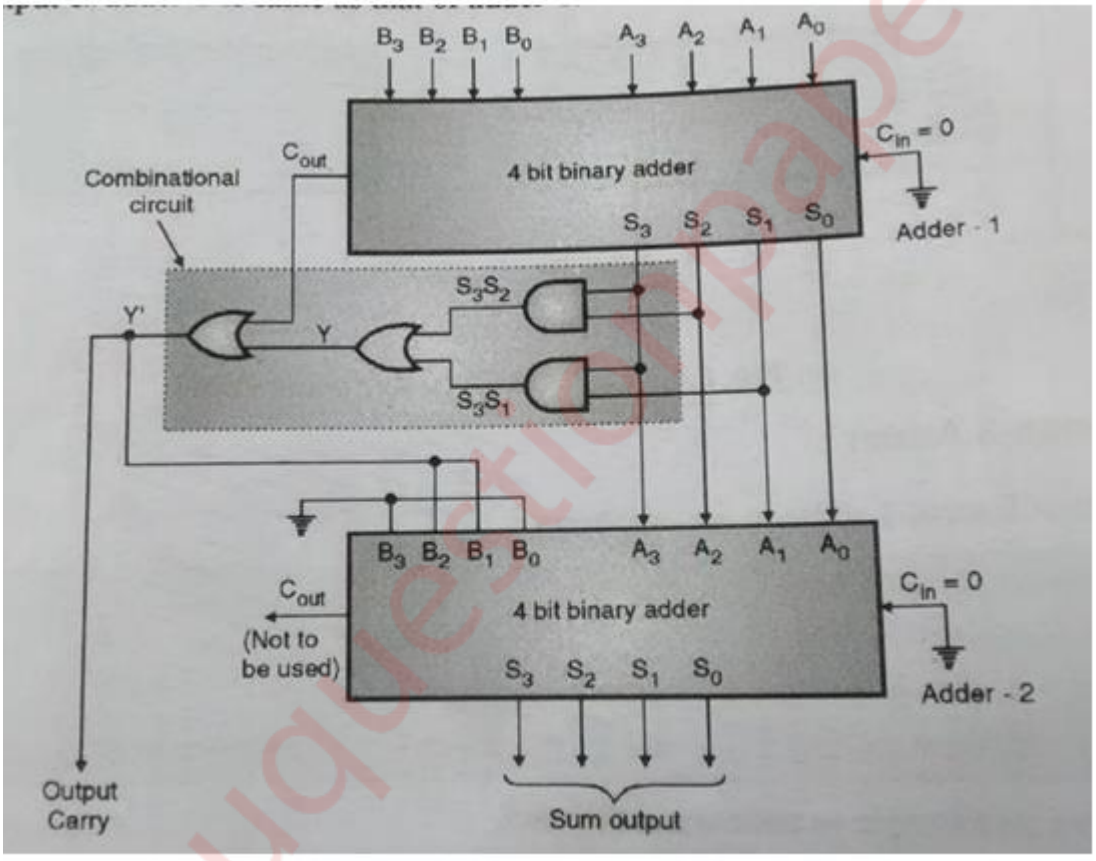


Fig. BCD addition using IC 7483

Example :

```
  1  1  1  1
 ------------
     0  1  1  1
+    1  0  0  1
 ------------
  1  0  0  0  0
 ------------
```

Thus,

Cout = 1

S3S2S1S0=0000
Hence, for adder, inputs will be
A3A2A1A0=0000
B3B2B1B0=0110
This will give final output as
Cout S3S2S1S0=10110

S3S2S1S0=0000

A3A2A1A0=0000
B3B2B1B0=0110

S3S2S1S0=10110.

Q.3 (a) Construct 32:1 MUX using 8:1 MUX only.Also comment about select lines used [10M]

Ans :

i) In electronics, a multiplexer (or mux), also known as a data selector, is a device that selects between several analog or digital input signals and forwards it to a single output line.

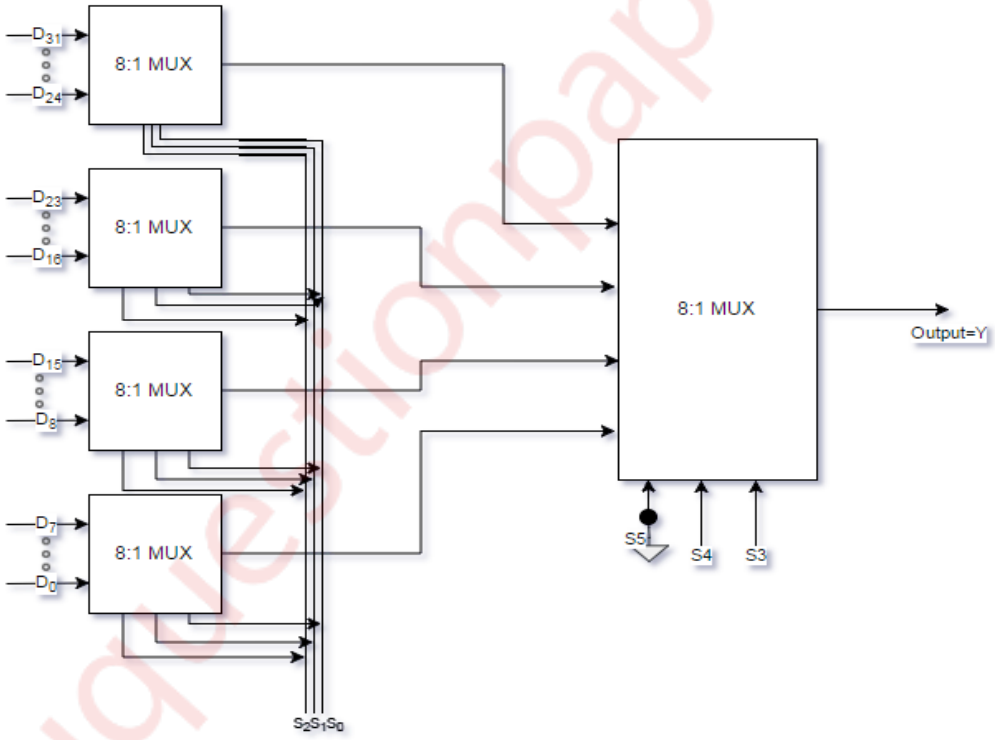
ii) The multiplexer is a combinational logic circuit designed to switch one of several input lines to a single common output line.

iii) Multiplexer are of different types .Example 2:1 MUX , 4:1 MUX , 8:1 MUX , 16:1 MUX ,32:1 MUX ,etc.

iv) We can implement a MUX using different types of MUX .For example , we can Construct 32:1 MUX using 8:1 MUX only.

V) Truth table and implementation of 32:1 MUX :

Select inputs					Output=Y
S_4	S_3	S_2	S_1	S_0	
0	0	0	0	0	D_0
⋮		⋮	⋮	⋮	⋮
0	0	1	1	1	D_7
0	1	0	0	0	D_8
⋮		⋮	⋮	⋮	⋮
0	1	1	1	1	D_{15}
1	0	0	0	0	D_{16}
⋮		⋮	⋮	⋮	⋮
1	0	1	1	1	D_{23}
1	1	0	0	0	D_{24}
⋮		⋮	⋮	⋮	⋮
1	1	1	1	1	D_{31}



Comment on select lines : i) From circuit diagram , we can see that S0 S1 S2 select lines are used to select MUX which are connected to the main data lines.

ii)There are select lines such as S5 S4 S3 used to select this multiple MUXes .

b) Solve the following using Kmap

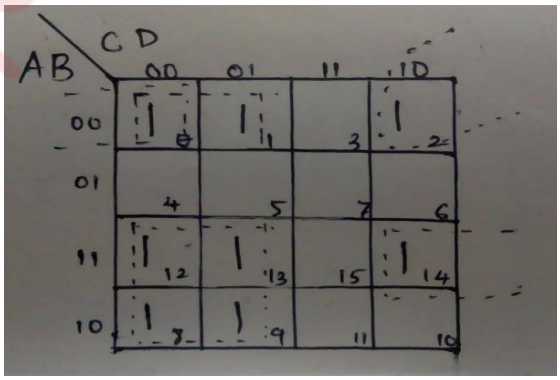
F(A,B,C,D) = $\pi M(3,4,5,6,7,10,11,15)$ **[5M]**

Ans : The given function contains max terms .

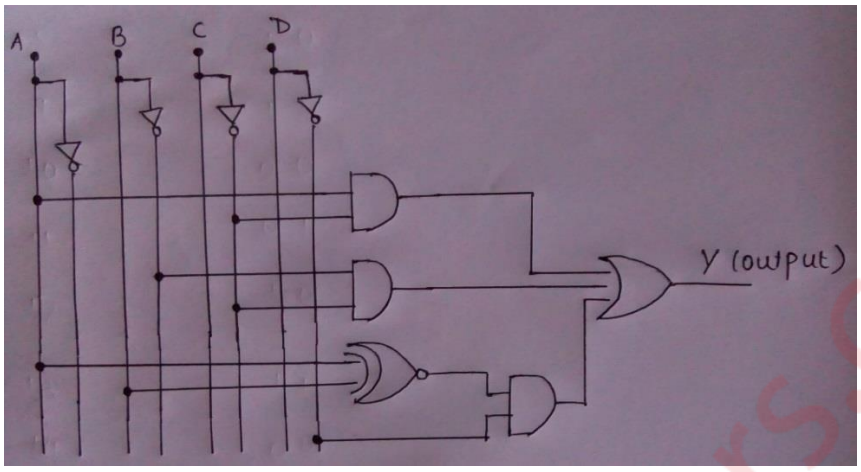
Truth table :

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

KMAP :



Circuit Diagram :



y = B'C' + AC' + A'B'D' + ABD'

c) Design full adder using half adders and few gates. [5M]

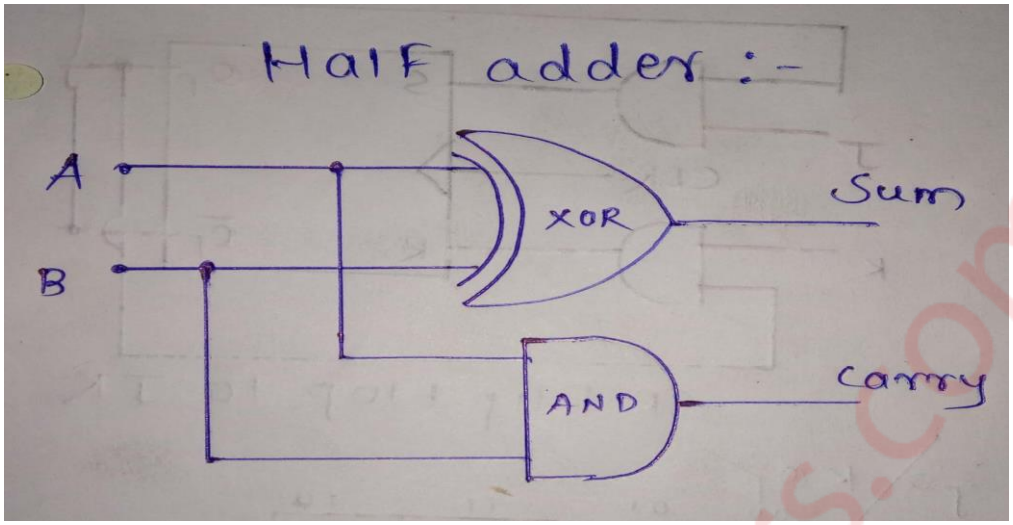
Ans : **Half Adder** : The addition of 2bits is called Half adder the input variebles are augent and addent bits and output variebles are sum&carry bits.

Full Adder : Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

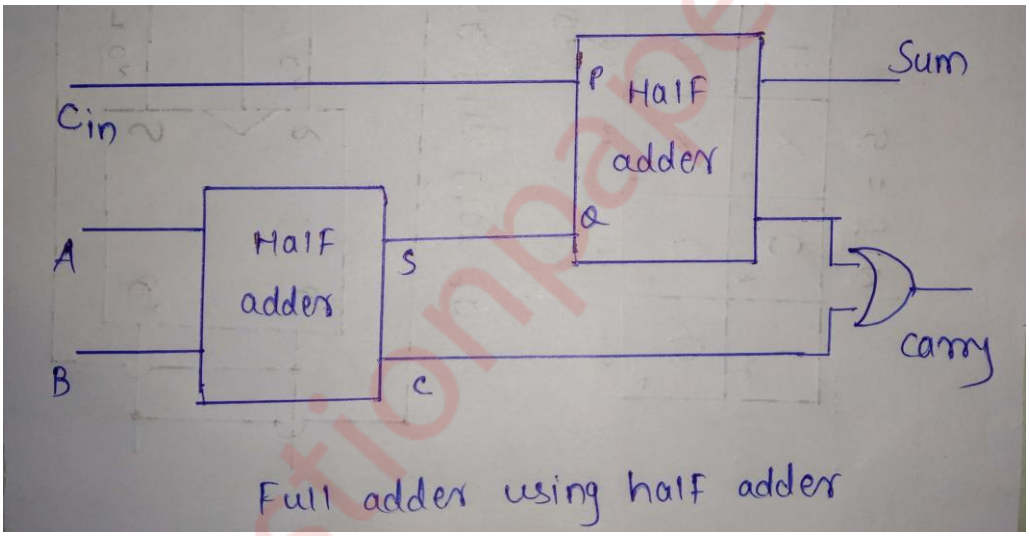
A	B	C	Y(SUM)	Y(CARRY)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A	B	Y(SUM)	Y(CARRY)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

HALF ADDER



Full adder using half adder circuit :



Q.4 a) Convert SR flip flop to JK flip flop and T flip flop. [10M]

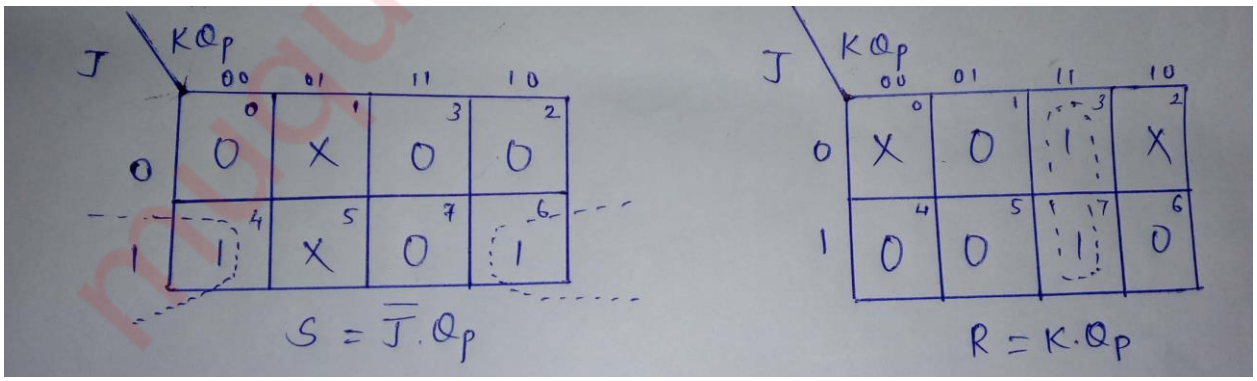
Ans : A) SR flip flop to JK flip flop :

I) The truth tables for the flip flop conversion are given below. The present state is represented by Qp and Qp+1 is the next state to be obtained when the J and K inputs are applied.

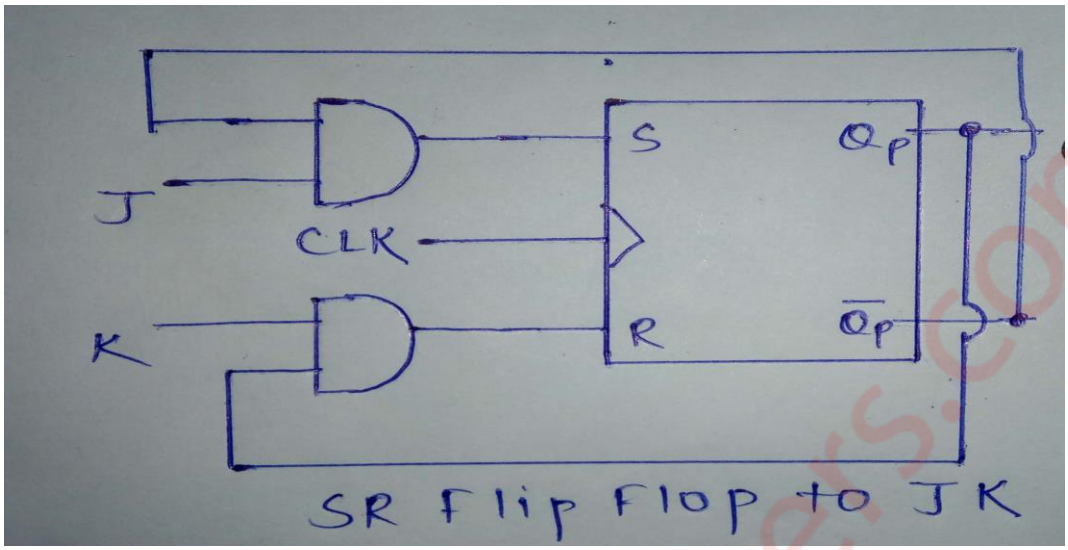
II)For two inputs J and K, there will be eight possible combinations. For each combination of J, K and Qp, the corresponding Qp+1 states are found. Qp+1 simply suggests the future values to be obtained by the JK flip flop after the value of Qp. The table is then completed by writing the values of S and R required to get each Qp+1 from the corresponding Qp. That is, the values of S and R that are required to change the state of the flip flop from Qp to Qp+1 are written.

Truth Table :

J	K	Q_P	Q_{P+1}	S	R
0	0	0	0	0	X
0	0	0	1	X	0
0	0	1	0	0	X
0	0	1	0	0	1
0	1	0	1	1	0
0	1	0	1	X	0
0	1	1	1	1	0
0	1	1	0	0	1



Circuit :



B) SR Flip flop to T flip flop :

Truth table of T flip flop :

T INPUT	PRESENT STATE	NEXT STATE
0	0	0
0	1	1
1	0	1
1	1	0

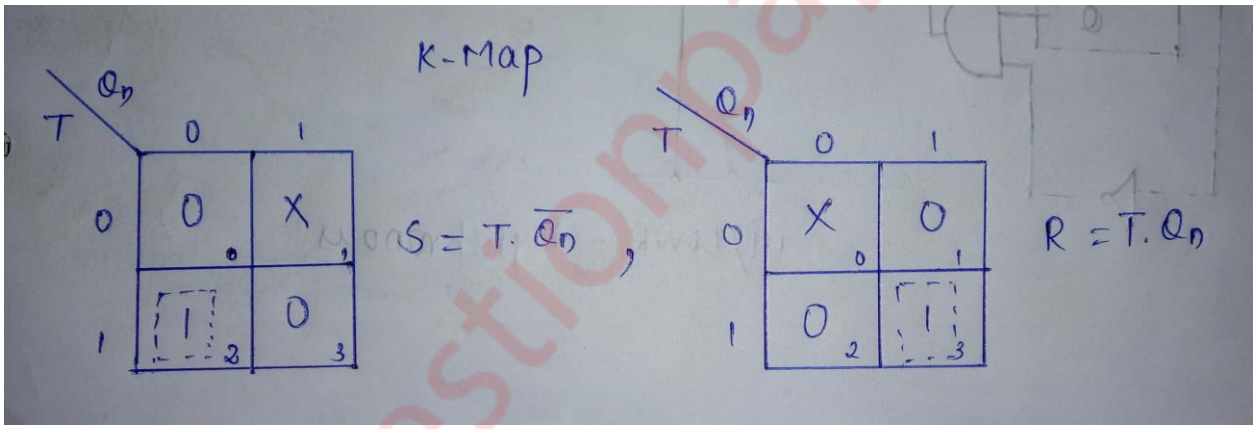
Excitation table of SR flip flop :

Present State	Next state	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	x	0

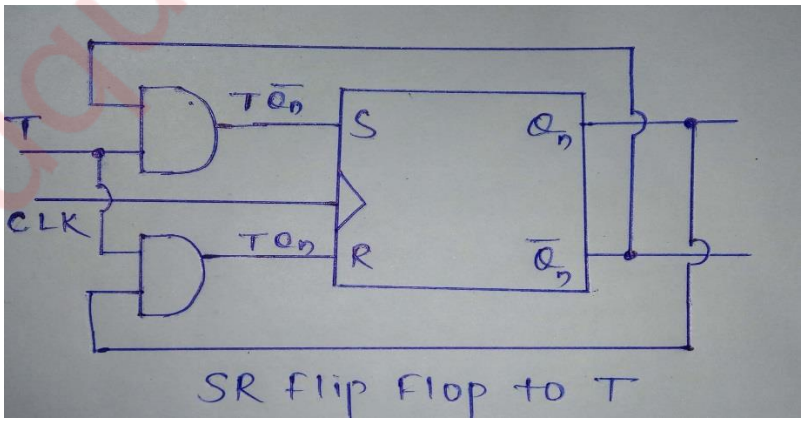
Combined table for conversion :

T	Present	Next state	S	R
0	0	0	0	x
0	1	1	x	0
1	0	1	1	0
1	1	0	0	1

1. Connect the S input to the output of a two-input AND gate which is driven by the user-provided input, T, and the negation of the flip-flop's present-state, $\bar{Q}_n$
2. Connect the R input to the output of a two-input AND gate which is driven by the user-defined input, T, and the present-state of the flip-flop, Q_n



Circuit :



b) Design 3-bit asynchronous up-down counter. [10M]

Ans :

i) As we know that in the up-counter each flip-flop is triggered by the normal output of the preceding flip-flop (from output Q of first flip-flop to clock of next flip-flop); whereas in a down-counter,

each flip-flop is triggered by the complement output of the preceding flip-flop (from output Q^ of first flip-flop to clock of next flip-flop).

Truth table :

State	Qc	Qb	Qa	State	Qc	Qb	Qa
0	0	0	0	7	1	1	1
1	0	0	1	6	1	1	0
2	0	1	0	5	1	0	1
3	0	1	1	4	1	0	0
4	1	0	0	3	0	1	1
5	1	0	1	2	0	1	0
6	1	1	0	1	0	0	1
7	1	1	1	0	0	0	0

i)For up down counting operation preceding flip-flop sometime it need input from output from output Q of first flip-flop to clock of next flip-flop for up-counting and sometimes from output Q^ of first flip-flop to clock of next flip-flop for down-counting. So in above circuit diagram it is shown clearly.

ii) As we know a flip-flop can hold single bit so for 3 bit operation it need three flip-flops.

iii)An inverter has been inserted in between the count-up control line and the count-down control line to ensure that the count-up and count-down cannot be simultaneously in the HIGH state.

iv)When the count-up/down line is held HIGH, the lower AND gates will be disabled and their outputs will be zero.

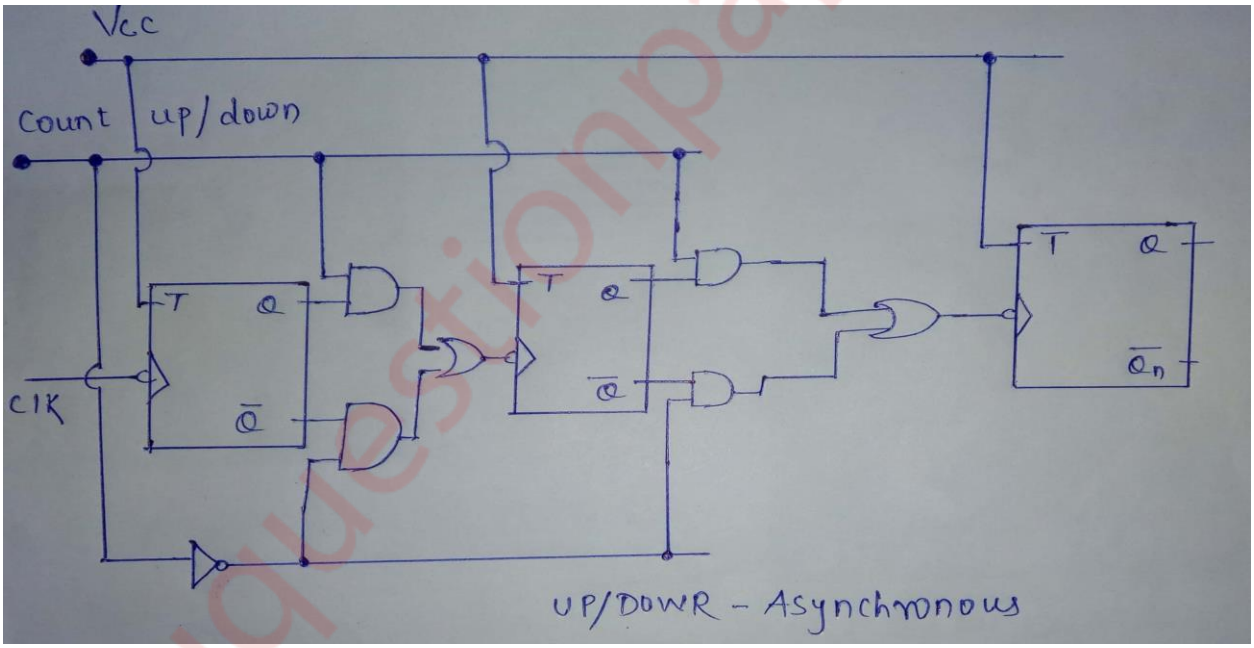
v) So they will not affect the outputs of the OR gates. At the same time the upper AND gates will be enabled. Hence, QA will pass through the OR gate and into the clock input of the B flip-flop.

vi)Similarly, QB will be gated into the clock input of the C flip-flop. Thus, as the input pulses are applied, the counter will count up and follow a natural binary counting sequence from 000 to 111.

vii)Similarly, with count-up/down line being logic 0, the upper AND gates will become disabled and the lower AND gates are enabled, allowing Q'A and Q'B to pass through the clock inputs of the following flip-flops.

viii)Hence, in this condition the counter will count in down mode, as the input pulses are applied.

Designed Circuit :



Q.5 a) Design 4-bit Binary to Gray Code Convertor. [10M]

Ans :

i) The logical circuit which converts the binary code to equivalent gray code is known as binary to gray code converter.

ii) The gray code is a non-weighted code. The successive gray code differs in one-bit position only that means it is a unit distance code.

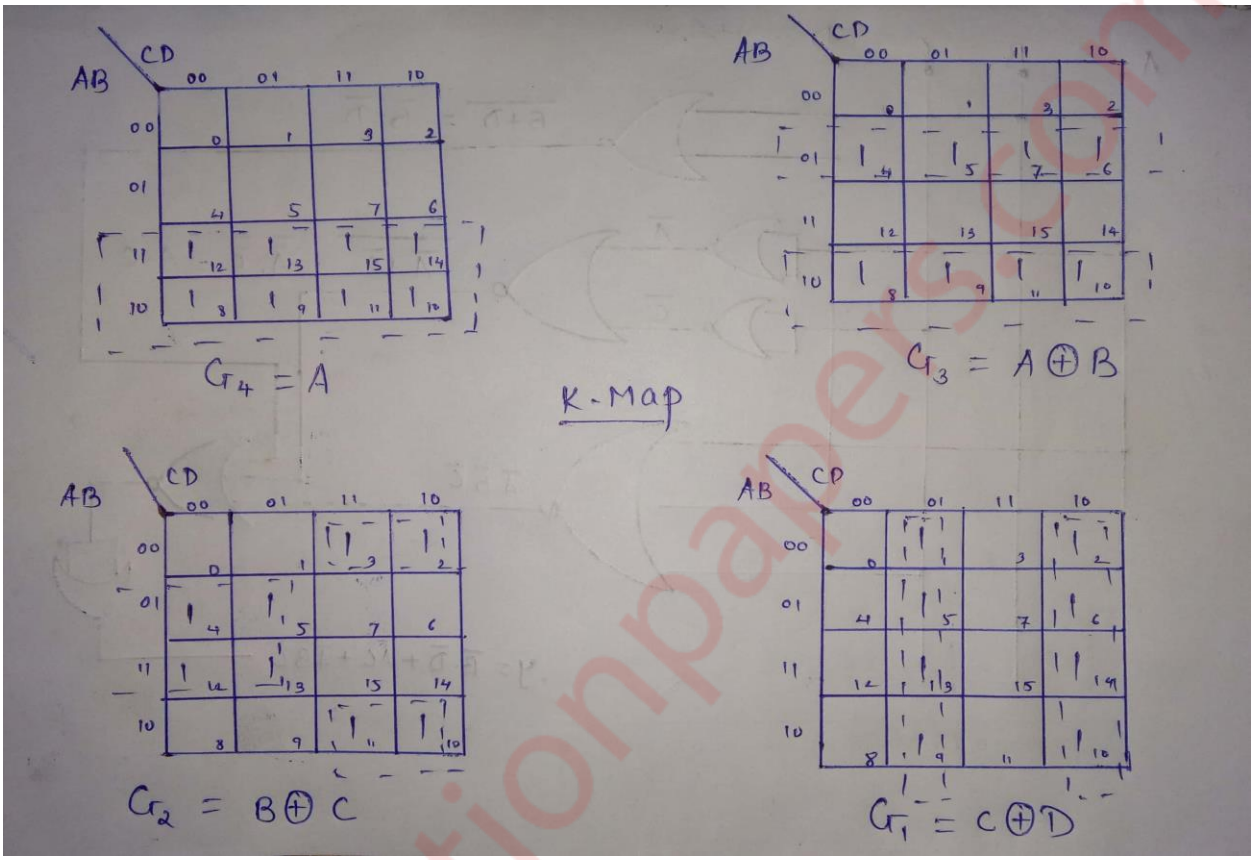
iii)It is also referred as a cyclic code. It is not suitable for arithmetic operations. It is the most popular of the unit distance codes. It is also a reflective code.

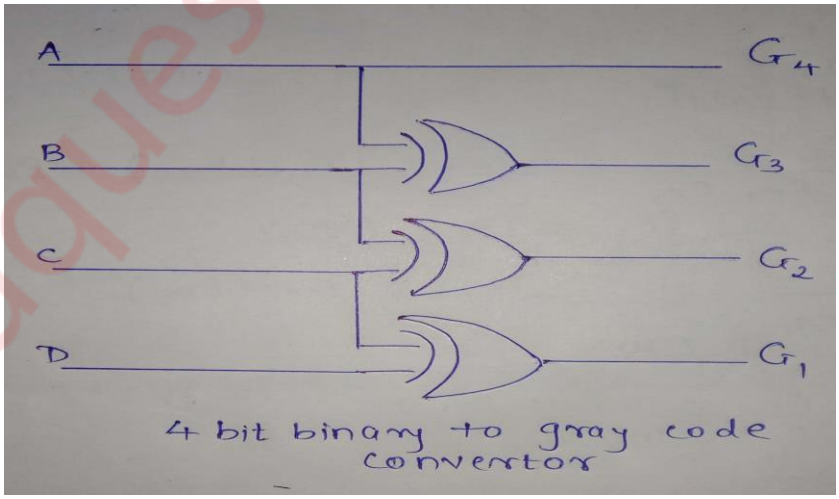
iv) An n-bit Gray code can be obtained by reflecting an n-1 bit code about an axis after 2n-1 rows and putting the MSB of 0 above the axis and the MSB of 1 below the axis. Reflection of the 4 bits binary to gray code conversion table is given below:

A	B	C	D	GRAY CODE
0	0	0	0	0000
0	0	0	1	0001
0	0	1	0	0011
0	0	1	1	0010
0	1	0	0	0110
0	1	0	1	0111
0	1	1	0	0101
0	1	1	1	0100
1	0	0	0	1100
1	0	0	1	1101
1	0	1	0	1111
1	0	1	1	1110
1	1	0	0	1010
1	1	0	1	1011
1	1	1	0	1001
1	1	1	1	1000

$G_4=\sum m(8,9,10,11,12,13,14,15)$, $G_3=\sum m(4,5,6,7,8,9,10,11)$

$G_2=\sum m(2,3,4,5,10,11,12,13)$, $G_1=\sum m(1,2,5,6,9,10,13,14)$





b) What is race around condition?How it is overcome in Master slave JK Flip flop? [5M]

Ans : Race around condition :

i) In jk flip flop there occurs a condition called race around when we put both j and k as 1.In race around Condition till the clock is high the output varies continuously from 0 to 1 &1 to 0.

ii) This condition is undesirable as it is of no use because the change in output is uncontrolled.

iii) In JK flip flop as long as clock is high for the input conditions J&K equals to the output changes or complements its output from 1–>0 and 0–>1.

iv) This is called toggling output or uncontrolled changing or racing condition. Consider above J&K circuit diagram as long as clock is high and J&K=11 then two upper and lower AND gates are only triggered by the complementary outputs Q and Q(bar). I.e. in any condition according to the propagation delay one gate will be enabled and another gate is disabled.

v) If upper gate is disabled then it sets the output and in the next lower gate will be enabled which resets the flip flop output.

vi) If the Clock On or High time is less than the propagation delay of the flip flop then racing can be avoided. This is done by using edge triggering rather than level triggering.

vii) If the flip flop is made to toggle over one clock period then racing can be avoided. This introduced the concept of Master Slave JK flip flop.

c) Design 1-Bit Magnitude comparator using logic gates. [5M]

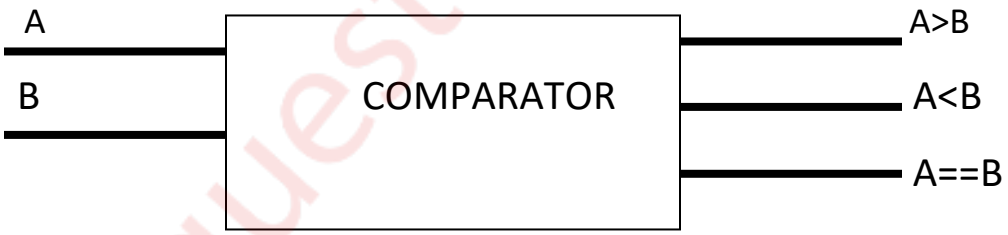
Ans : i) A comparator used to compare two bits is called a single bit comparator.

ii) It consists of two inputs each for two single bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers.

A	B	A>B	A<B	A==B
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

iii) A magnitude digital Comparator is a combinational circuit that compares two digital or binary numbers in order to find out whether one binary number is equal, less than or greater than the other binary number.

iv) We logically design a circuit for which we will have two inputs one for A and other for B and have three output terminals, one for A > B condition, one for A = B condition and one for A < B condition.

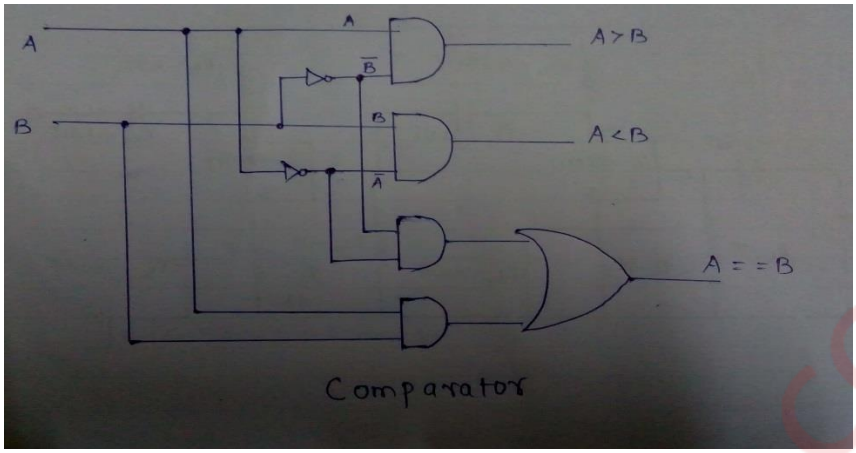


Logical expression :

A>B : AB'

A<B : A'B

A=B : A'B' + AB



Q. 6 Write a short note on any four **[20M]**

a) VHDL Modelling Styles

b) TTL and CMOS Logic Families.

c) SISO and PISO Shift Register

d) ALU

e) Twisted ring counter

Ans : A) VHDL Modelling Styles :

1] Structural modelling: in this type of modelling an entity is explained as a set of inter connected component's.

- It shows graphical representation of modules component's, with their inter connection.
- Structural modelling can be used to generate very high level or low level description in ckt.

2] Data flow modelling: In data flow modelling the data flow through the entity is expressed using con current signal assignment statements.

- Concurrent signal assignment statements are those in which appear outside of a process, there are event triggered.
- In signal assignment for assignment of a value to a signal symbol < = is used.

3] Behavioral style of modelling:

- The behavioral level of abstraction is the level of abstraction supported in VHDL.
- Process body must be included in every behavioral description.
- The behavioral style of modelling denotes the entity behavior as a set of statements which executes sequentially.

4] Mixed style of modelling: in a single structure body, we can mix the three modelling style is within an architecture body to represent structure we can use components installation statements, to represent data flow we can use concurrent signal assignment statements and to represent behavior, we can use process statements

B) **TTL and CMOS Logic Families :**

i) The transistor-transistor-logic (TTL) family was developed in the use of transistor switches for logical operations and defines the binary values as

0 V to 0.8 V = logic 0
2 V to 5 V = logic 1

ii) TTL is the largest family of digital ICs, but the CMOS family is growing rapidly. They are inexpensive, but draw a lot of power and must be supplied with +5 volts. Individual gates may draw 3 to 4 mA.

iii) The low power Schottky versions of TTL chips draw only 20% of the power, but are more expensive. Part numbers for these chips have LS in the middle of them.

iv) The complementary metal oxide semiconductor family (CMOS) has equivalents to most of the TTL chips.

v) CMOS chips are much lower in power requirements (drawing about 1 mA) and operate with a wide range of supply voltages (typically 3 to 18 volts).

vi) The CMOS model number will have a C in the middle of it, e.g., the 74C04 is the CMOS equivalent to the TTL 7404.

vii) A high drawback is extreme sensitivity to static electricity - they must be carefully protected from static discharges.

C) SISO and PISO Shift Register :

i) The Shift Register is another type of sequential logic circuit that can be used for the storage or the transfer of binary data

ii) Shift Registers are used for data storage or for the movement of data and are therefore commonly used inside calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format.

iii) Serial-in to Serial-out (SISO) - the data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.

The SISO shift register is one of the simplest of the four configurations as it has only three connections, the serial input (SI) which determines what enters the left hand flip-flop, the serial output (SO) which is taken from the output of the right hand flip-flop and the sequencing clock signal (Clk).

iv) Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.

The Parallel-in to Serial-out shift register acts in the opposite way to the serial-in to parallel-out one above. The data is loaded into the register in a parallel format in which all the data bits enter their inputs simultaneously, to the parallel input pins PA to PD of the register. The data is then read out sequentially in the normal shift-right mode from the register at Q representing the data present at PA to PD.

D) ALU :

i) An arithmetic logic unit (ALU) is a digital circuit used to perform arithmetic and logic operations.

ii) It represents the fundamental building block of the central processing unit (CPU) of a computer. Modern CPUs contain very powerful and complex ALUs. In addition to ALUs, modern CPUs contain a control unit (CU).

iii) Most of the operations of a CPU are performed by one or more ALUs, which load data from input registers. A register is a small amount of storage available as part of a CPU.

iv) The control unit tells the ALU what operation to perform on that data, and the ALU stores the result in an output register. The control unit moves the data between these registers, the ALU, and memory.

v) An ALU performs basic arithmetic and logic operations. Examples of arithmetic operations are addition, subtraction, multiplication, and division. Examples of logic operations are comparisons of values such as NOT, AND, and OR.

vi) All information in a computer is stored and manipulated in the form of binary numbers, i.e. 0 and 1. Transistor switches are used to manipulate binary numbers since there are only two possible states of a switch: open or closed.

vii) An open transistor, through which there is no current, represents a 0. A closed transistor, through which there is a current, represents a 1.

E**) Twisted Ring counter :**

i) A twisted ring counter, also called switch-tail ring counter, walking ring counter, Johnson counter, or Möbius counter, connects the complement of the output of the last shift register to the input of the first register and circulates a stream of ones followed by zeros around the ring.

ii)Ring counters are often used in hardware design (e.g. ASIC and FPGA design) to create finite-state machines .

iii) A binary counter would require an adder circuit which is substantially more complex than a ring counter and has higher propagation delay as the number of bits increases, whereas the propagation delay of a ring counter will be nearly constant regardless of the number of bits in the code.