Q.1 Solve following :

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

Ans.: IP = 1762 FP = 0.46

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

0.46	
× 2	
0.92	0
× 2	
1.84	1
× 2	
1.68	1
× 2	
1.36	1

$(0.46)_{10} = (0111)_2$

$(1762)_{10} = (11011100010)_2$

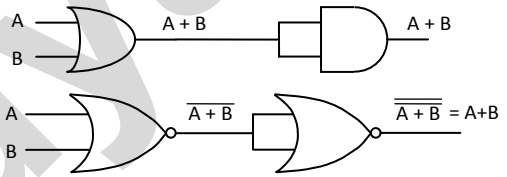
$\therefore$ $(1762.46)_{10} = (11011100010.0111)_2$

$(1762.46)_{10} = (3342.34)_8$

$(1762.46)_{10} = (6E2.7)_6$

Q.1(b) Prove OR-AND configuration is equivalent to NOR-NOR configuration. **[3]**

Ans.: Consider an example to show that OR–AND configuration is equivalent to NOR–NOR configuration



$\therefore$ LHS = RHS

Q.1(c) Perform following subtraction $(52)_{10} - (65)_{10}$ using 2's complement method. **[3]**

Ans.: 52 → 0110100

− 65 → 1000001 $\xrightarrow{\text{2's comp}}$ 0111111

$\therefore$ 52 → 0110100

\+ 2's C (65) → 0111111

C 0 1110011

$\because$ Carry bit C = 0 ($\therefore$ Answer is negative)

$\therefore$ = – (2's comp of 1110011)

= – $(0001101)_2$ = – 13

Q.1(d) Explain the term prime implicant. **[2]**

Ans.:
- Any single 1 or group of 1's that can be grouped together on the k–map of a function F represents a product term (P) called an Implicant.
- A prime implicant (PI) is a product term P that cannot be combined with another product term to eliminate a variable.
- A prime implicant that includes one or more distinguished one cells.
- The term prime implicant is useful in boolean algebra. K–map and Quine-McClusky.

Q.1(e) Construct Hamming code for 1010. **[3]**

Ans.:

B_1	B_2	B_3	B_4
1	0	1	0

No. of bits, n = 4

Value of k must be chosen to satisfy the following equation

$2^k \geq n + 4 + 1$

$\therefore \quad 2^k \geq n + 5$

Minimum value of k for which it is satisfied is k =3, Hence, 3 parity bits are attached to each of the BCD code for constructing hamming code.

$P_1 = B_1 + B_2 + B_4 = 1 + 0 + 0 = 1 \rightarrow 1$

$P_2 = B_1 + B_3 + B_4 = 1 + 1 + 0 = 2 \rightarrow 0$

$P_3 = B_2 + B_3 + B_4 = 0 + 1 + 0 = 1 \rightarrow 1$

For 4–bit code 3 parity bits P_1, P_2 and P_3 are appended in locations 1,2,4 respectively.

P_1	P_2	B_1	P_3	B_2	B_3	B_4
1	0	1	1	0	1	0

Q.1(f) Find 8's complement of the numbers $(37)_8$ and $(301)_8$ **[3]**

Ans.: $(37)_8 \rightarrow$ 7's complement $\rightarrow$

$$\begin{array}{rr} & 7\ 7 \\ - & \underline{3\ 7} \\ & 4\ 0 \end{array}$$

8's complement = 40 + 1 = 41

$(301)_8 \rightarrow$ 7's complement $\rightarrow$

$$\begin{array}{rr} & 7\ 7\ 7 \\ - & \underline{3\ 0\ 1} \\ & 4\ 7\ 6 \end{array}$$

8's complement = 476 + 1 = 477

Q.1(g) Add $(22)_{10}$ to $(56)_{10}$ in BCD. **[3]**

Ans.: $(22)_{10}$ BCD 0010 0010

$+ (56)_{10} \Rightarrow + \ 0101 \ 0110$

0111 1000

$\Downarrow$

$(78)_{10}$

Q.2(a) Obtain the minimal expression using Quine Mc-Cluskey method [10]

$F(A,B,C,D) = \sum m(1,5,6,12,13,14) + d(2,4)$

Ans.: Step 1: Determine prime Implicant

Min terms	Binary Representation			
1	0	0	0	1
d_2	0	0	1	0
d_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

Grouping Minterms	Binary Representation			
(1, 5)	0	–	0	1
(d_4, 5)	0	1	0	–
(d_2, 6)	0	–	1	0
(d_4, 6)	0	1	–	0
(d_4, 12)	–	1	0	0
(5, 13)	–	1	0	1
(12, 13)	1	1	0	–
(6, 14)	–	1	1	0
(12, 14)	1	1	–	0

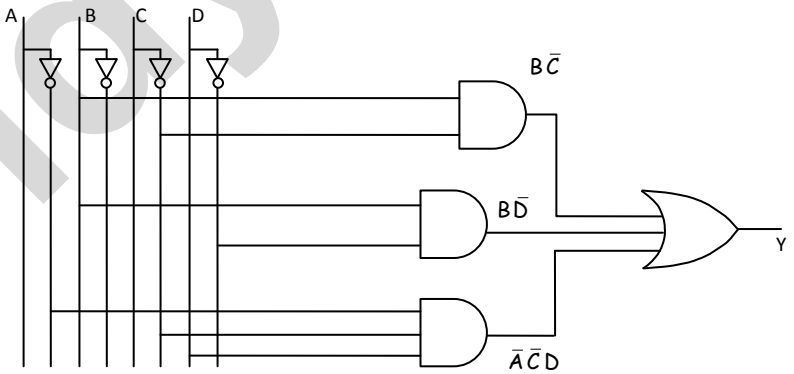
Grouping of Minterms :

Minterms	Binary Representation			
(d_4, 5, 12, 13)	–	1	0	–
(d_4, 6, 12, 14)	–	1	–	0

Step 2: Prime Implicant table

Prime Implicant \ Minterms	1	5	6	12	13	14	d_2	d_4
(d_4, 5, 12, 13)		X		X	X			X
(d_4, 6, 12, 14)			X	X		X		X
(1, 5)	X	X						

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

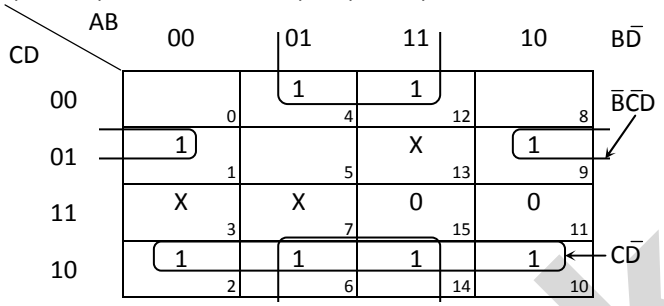


Q.2(b) Simplify the following equation using K map to obtain SOP equation and realize the minimum equation using only NAND gates. **[10]**

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

Ans: Realize using only NAND gates

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



From k–map the minimized equation is,

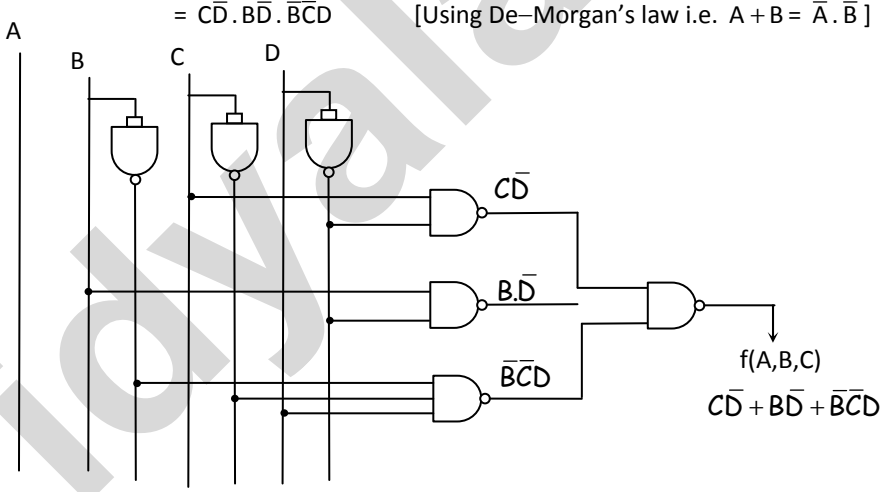
$F(A,B,C,D) = C\bar{D} + B\bar{D} + \bar{B}\bar{C}D$

For implementing using only NAND gates.

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

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

$= \overline{\overline{C\bar{D}} \cdot \overline{B\bar{D}} \cdot \overline{\bar{B}\bar{C}D}}$ [Using De–Morgan's law i.e. $\overline{A+B} = \bar{A} \cdot \bar{B}$]



Implementation of given equation only NAND gates.

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

Ans.: Step 1: Truth table of full adder

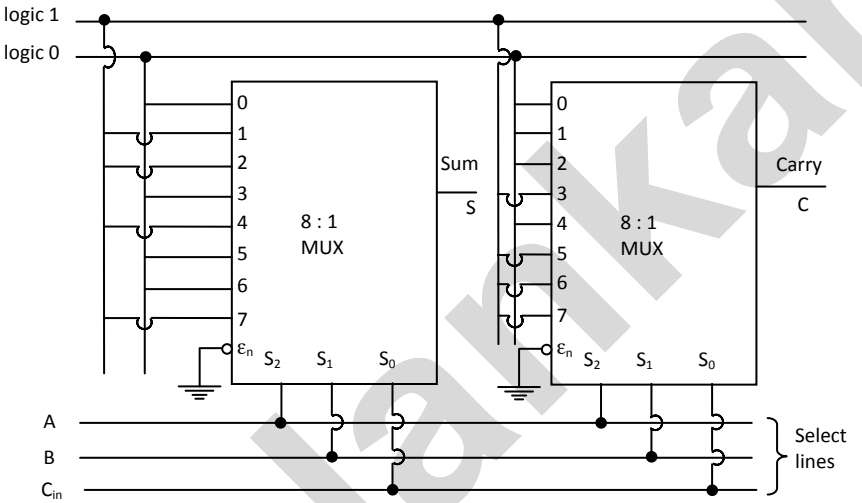
	Inputs			Outputs	
	A	B	C_{in}	S	C
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1

4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

$S = \sum m\ (1, 2, 4, 7)$

$C = \sum m\ (3, 5, 6, 7)$

Step 2: Multiplexer Implementation



Q.3(b) What is race around condition? How to overcome it? [10]

Ans.:

- The difficulty of both inputs 1 (S = R = 1) being not allowed in an S-R flip-flop is eliminated in a J-K flip-flop by using the feedback connected from outputs to inputs of gates G_3 and G_4 as shown in figure 1.

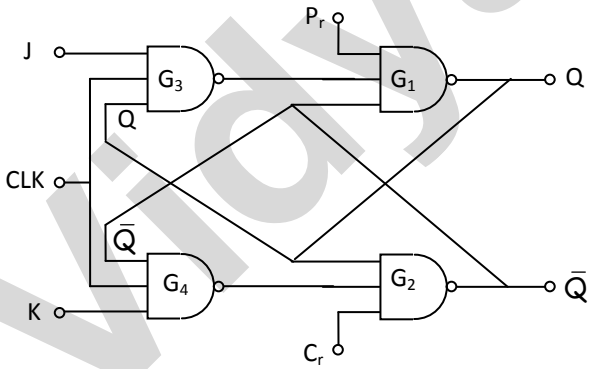


Fig.1: J–K FLIP–FLOP

Truth Table of J–K FLIP FLOP

Input		Output
J_n	K_n	Q_{n+1}
0	0	Q_n
1	0	1
0	1	0
1	1	$\bar{Q}_n$

- The above truth table assumes that the inputs do not change during the clock pulse (CLK = 1), which is not true because of the feedback connections.

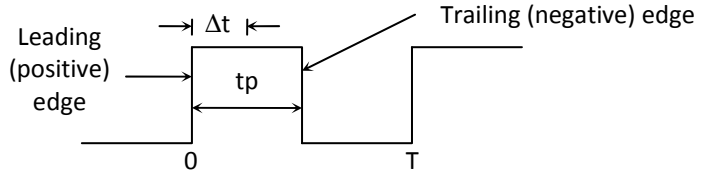


Fig.2 : A clock pulse

- Consider that inputs are J = K = 1 and Q = 0 and a pulse shown in figure 2 is applied at the clock input.
- After a time interval Δt equal to propagation delay through two NAND gates in series, output will change to Q = 1 (as seen in Truth Table).
- Now, when J = K = 1 and Q = 1 and after another time interval of Δt the output will change back to Q = 0. Hence, for duration tp of clock pulse, output will oscillate back and forth between 0 and 1. At the end of clock pulse, the value of Q is uncertain. This situation is referred to as the race–around condition.
- The race–around condition can be avoided if $tp < \Delta t < t$. But this is difficult because of very small propagation delays in ICs. Practical method for overcoming this is use of master–slave (M–S) configuration.
- A master slave J–K FLIP FLOP is a cascade of two S–R FLIP FLOPs, with feedback from the outputs of the second to inputs of first as shown in figure below :

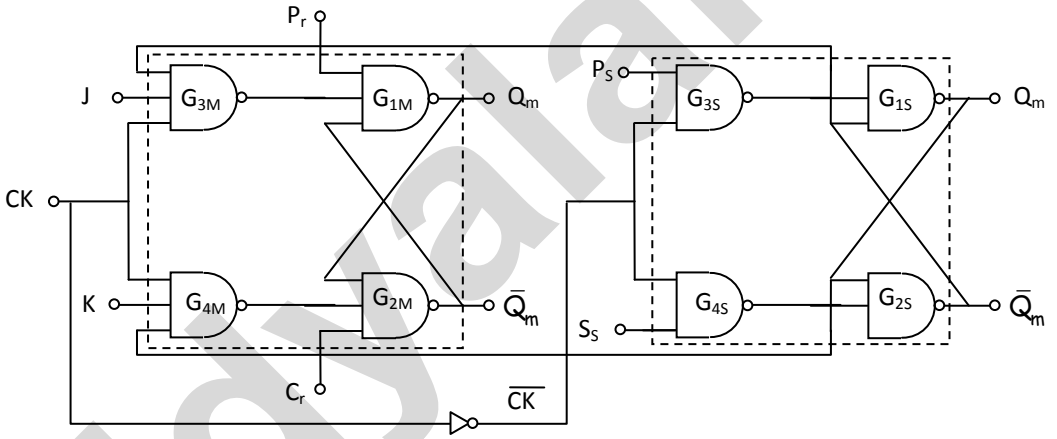


Fig.3: Master Slave J–K FLIP–FLOP

- Positive clock pulses are applied to first FLIP –FLOP and clock pulses are inverted before these are applied to the second FLIP–FLOP.
- When CK=1, the first FLIP–FLOP is enabled and the outputs Q_m and $\overline{Q}_m$ respond to inputs J and K. At this time, the second FLIP–FLOP is inhibited because its clock is LOW ($\overline{CK}$ =0)
- When CK goes LOW ($\overline{CK}$ =1), the first FLIP–FLOP is inhabited and second FLIP–FLOP is enabled, because now its clock is HIGH ($\overline{CK}$ =1). Thus the outputs Q and $\overline{Q}$ follow the outputs Q_m and $\overline{Q}_m$ respectively.
- Since, the second FLIP–FLOP simply follows the first–one, it is referred to as the slave and first one as the master. Hence, this configuration is referred to as master–slave (M–S) FLIP–FLOP.

- In this circuit, the inputs to the gates G_3M and G_4m do not change during the clock pulse, hence the race–around condition does not exist.

Q.4(a) Compare TTL and CMOS logic with respect to fan in, fan out, propagation delay, power consumption, noise margin, current and voltage parameters. **[10]**

Ans.:

Parameter	TTL	CMOS
V_{1H} (min)	2 V	3.5 V
V_{1L} (max)	0.8 V	1.5 V
V_{0H} (min)	2.4 V	4.5 V
V_{0L} (min)	0.4 V	0.5 V
I_{1H}	40 μA	5 nA
I_{1L}	1.6 mA	5 nA
I_{0H}	400 μA	0.36 mA
I_{0L}	16 mA	0.4 mA
Basic Gates	NAND	NOR/NAND
Propagation Delay	10 ns	105 ns
Power Consumption	10 mW	0.1 mW
Fan Out	10	3 – 10
Operating Voltage	4.75 V – 5.25 V	3V – 15V

Q.4(b) Design 3 bit asynchronous counter and draw the timing diagram. **[10]**

Ans.: Counting Sequence of a 3–bit binary counter

Counter state	Count		
	Q_2	Q_1	Q_0
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

- Consider the count sequence shown in above table. The number of states in this sequence is 8 which requires 3 FLIP–FLOPs ($2^3 = 8$) and Q_2, Q_1 and Q_0 are the outputs of these FLIP–FLOPs. Assuming master–slave FLIP–FLOPs.
- The output Q_0 of the least–significant FLIP–FLOP, change for every clock pulse. This can be achieved by using T–Type FLIP–FLOP with $T_0 = 1$. The output Q_1 makes a transition (from 0 to 1 or 1 to 0) whenever Q_0 changes from 1 to 0.
- Thus, if Q_0 is connected to the clock input of next T–type FLIP–FLOP, FF1 with $T_1 = 1$, Q_1 will change whenever Q_0 goes from to 0 (falling edge of clock pulse.)

- Similarly, Q_2 makes a transition whenever Q_1 goes from 1 to 0 and this can be achieved by connecting Q_1 to the clock input of the most-significant FLIP–FLOP, FF2 (and T_2 =1). The resulting circuit is shown in fig. 1 below :

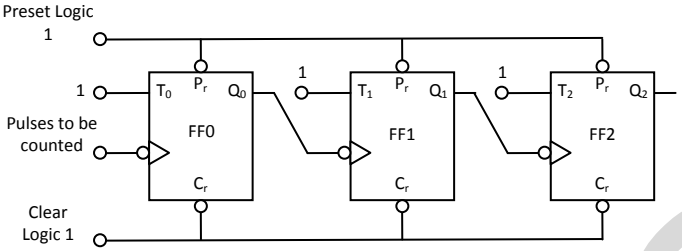


Fig.1: 3-Bit binary Counter

- The waveforms of the outputs of the FLIP FLOPs are shown in figure below :

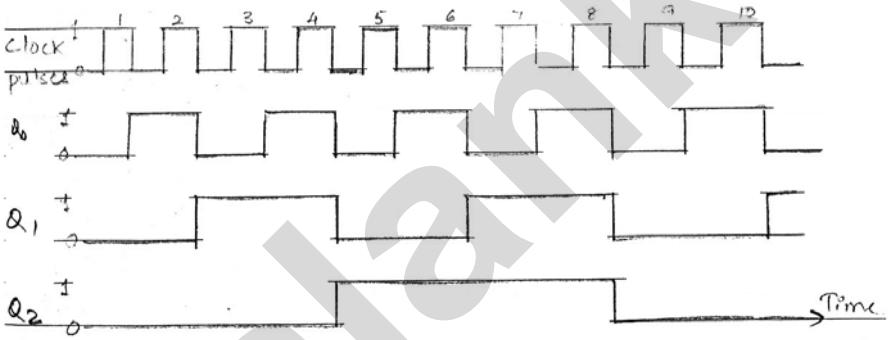


Fig.2: Output waveforms of counter of fig.1

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

Ans.: JK flip to SR flip flop

Step 1: Truth table for SR flip–flop

Input		Output
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	*

Step 2: Excitation table for JK flip flop

Input		Output	
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

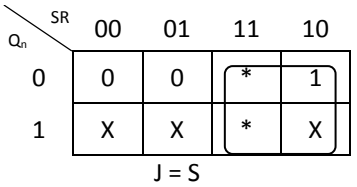
Step 3: Combine both table

Input		Present State	Next State	Output	
S	**R**	**Q_n**	**Q_{n+1}**	**J**	**K**
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X

1	0	1	1	X	0
1	1	0	*	*	*
1	1	1	*	*	*

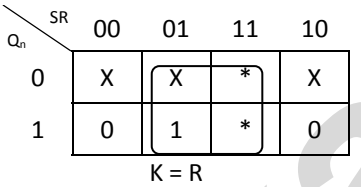
Step 4: K–map implementation for J & K

K-map for J



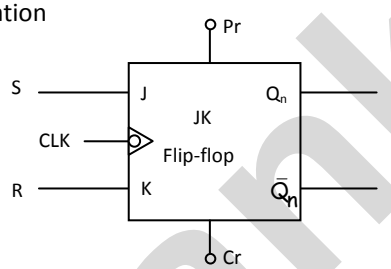
J = S

K-map for K



K = R

Step 5: Flip–flop implementation



JK flip flop to D flip flop

Step 1: Truth table for D flip flop

Input	Output
D	**Q_{n+1}**
0	0
1	1

Step 2: Excitation table for JK flip flop

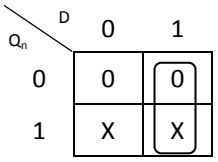
Inputs		Outputs	
Q_n	**Q_{n+1}**	**J**	**K**
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Step 3: Combine both tables

Input	Present State	Next State	Outputs	
D	**Q_n**	**Q_{n+1}**	**J**	**K**
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0

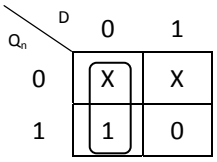
Step 4: K-map for J & K

K-map for J



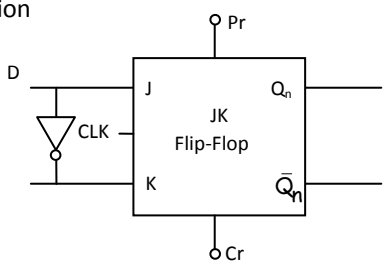
J = D

K-map for K



K = $\bar{D}$

Step 5: Flip flop implementation



Q.5(b) Explain the features of VHDL and its modeling styles. [10]

Ans.:

- VHDL can be used as an exchange medium between chip vendors and CAD tool users.
- VHDL supports hierarchy, that is, a digital system can be modeled as a set of interconnected components.
- VHDL supports flexible design methodologies : top–down, bottom–up, mixed.
- VHDL is not technology–specific, but it is capable of supporting technology-specific features. It can also support various hardware technologies.
- VHDL supports both synchronous & asynchronous timing models.
- Various digital modeling techniques such as finite–state machine descriptions, algorithmic descriptions and Boolean equations can be modeled using VHDL.
- VHDL is an IEEE and ANSI standard, and thus models described using this language are portable.
- VHDL has elements that make large scale design modeling–easier eg: components , functions, procedures and packages.
- VHDL when used for systems design, it allows behavior of required system to be modeled and simulated before synthesis tools translate design into real hardware.
- VHDL allows description of concurrent system.
- The internal details of an entity are specified by an architecture body using any of the following modeling styles :

1) As a set of interconnected components (to represent structure)
2) As a set of concurrent assignment statements (to represent data flow)
3) As a set of sequential assignment statements (to represent behavior)
4) Any combination of above three.

1) Structural Style of modeling :

- In this, an entity is described as a set of interconnected components. Such as model for HALF_ADDER entity is shown below :

```
entity HALF_ADDER is
port(A,B : in BIT, SUM, CARRY : out BIT):
and HALF_ADDER;
architecture HA_STRUCTURE of HALF ADDER is
component XOR2
      port (X,Y : in BIT; Z :out BIT);
end component;
component AND2
      port (L,M : in BIT; N : out BIT);
```

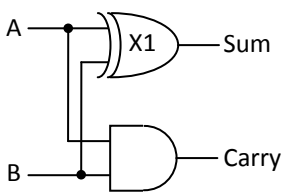


Fig.: Half adder circuit

```
end component;
begin
    X1 : XOR2 port map (A,B, SUM);
    A1 : AND2 port map (A, B, CARRY)
end HA_structure
```

- Here half adder is described as an interconnection of an XOR gate design entity and an AND gate design entity.
- Design entity half–adder describes how XOR & AND gates are connected to implement a half adder. It is top–level entity that has structural style description.
- When components are used, each must be declared.
- A port map tells how a design entity is connected in enclosing architecture.

2) Data flow style of modeling :

- In this, the flow of data through entity is expressed using concurrent signal assignment statements.
- Consider following code for HALF ADDER.

```
architecture HA_CONCURRENT of HALF_ADDER is begin
    SUM  < = A XOR B after 8ns;
    CARRY < = A and B after 4ns;
end HA_CONCURRENT;
```

- First assignment statement describes how input data flows from inputs A and B through on XOR function to create sum. Second assignment describes how input data flows through an AND function to create carry.

3) Behavioral style of modeling :

- In this, it specifies behavior of entity as a set of statements that are executed sequentially in specified order.
- Architecture consists of a single process statement.
- Following the keyword process is a list of signalism parentheses called a sensitivity list, which enumerates exactly which cause process to be executed. Whenever there is an event on a signal in a process sensitivity list, the process is executed.

```
architecture behavior of half–adder is
begin
    ha : process (a,b)
begin
    if a = '1' then
    sum < = not b;
    carry_out < = b;
    else
    sum < = b;
    carry_out < = '0';
    end if;
    end process ha;
    end behavior;
```

Q.6 Write short notes on following (any FOUR) : [20]

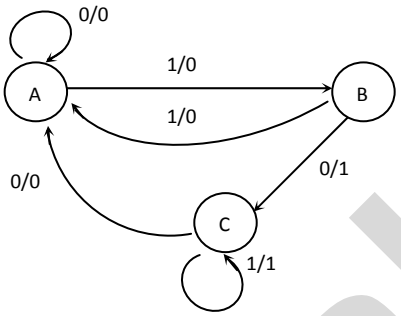
Q.6(a) Write short note on State table. [5]

Ans.:

- A state table or state transition table is a tabular form of representing the behavior of a sequential circuit.
- Each row of the table corresponds to a state of the circuit and each column corresponds to combinational of external inputs. The entries of the table denote the state transition (next state) and the outputs associated with these transitions.
- The no. of rows in the state table will be equal to the no. of states of the circuit, and the no. of columns will be same as the no. of combination of the inputs.
- A state table can be easily constructed from a state diagram. In fact, whatever information is available in the state diagram is also available in its state table and one can be obtained from the other.

Example, construct state table for following diagram :

State diagram :



State table :

Present state PS	Next state, output NS, Y	
	X = 0	X = 1
A	A, 0	B, 0
B	C, 1	A, 0
C	A, 0	C, 1

Q.6(b) Write short note on Sequence Generator. [5]

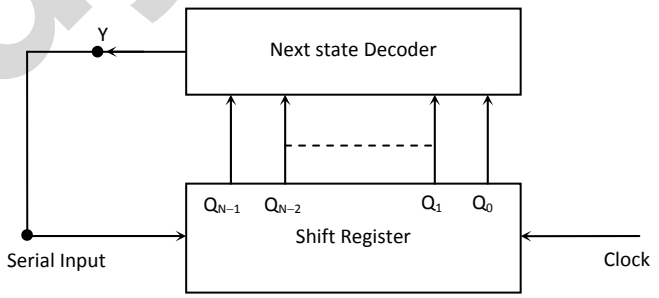
Ans.:

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

(i) Counters (ii) Random bit generators

(iii) Prescribed period and sequence generator (iv) Code generator

The basis structure of sequence generator is as follows :



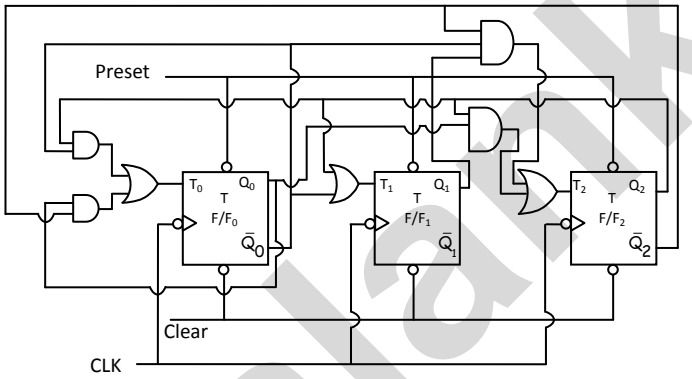
Example, obtain the following sequence generator using T Flip–flop

2 – 6 – 5 – 3 – 1 – 0 – 2

Truth table :

	State	Present State			Next State			i/p to flip-flop		
		Q_2	Q_1	Q_0	Q_{2+1}	Q_{1+1}	Q_{0+1}	T_2	T_1	T_0
Used state	2	0	1	0	1	1	0	1	0	0
	6	1	1	0	1	0	1	0	1	1
	5	1	0	1	0	1	1	1	1	0
	3	0	1	1	0	0	1	0	1	0
	1	0	0	1	0	0	0	0	0	1
	0	0	0	0	0	1	0	0	1	0
Unused state	4	1	0	0	×	×	×	×	×	×
	7	1	1	1	×	×	×	×	×	×

Circuit Diagram of sequence generator using T flip-flop :



Q.6(c) Write short note on 4-bit ring counter. [5]

Ans.: A ring counter is a shift register (a cascade connection of flip–flops) with the output of the last flip flop will be connected to the input of the first.

4-bit ring counter using D Flip flop :

Step 1: No of required flip flop

$m = 2^N$

m = No. of states

N = No. of flip flops

$m = 2^4 = 16$ states (0 to 15)

$\therefore$ N = 4 (No. of flip flops)

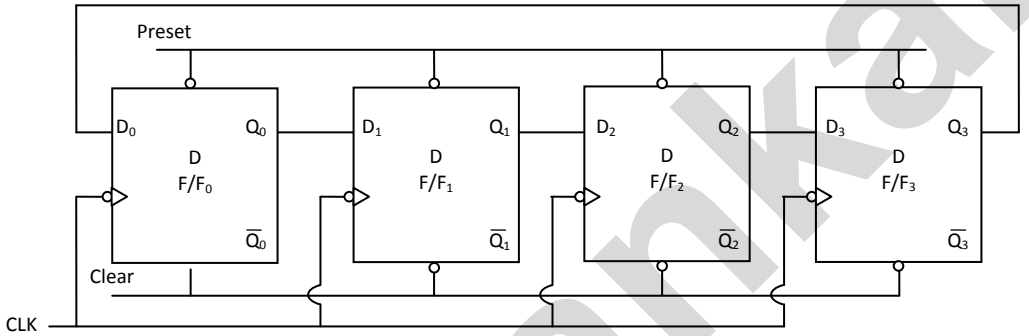
Step 2: Excitation table of D flip flop

Inputs		Outputs
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Step 3: Truth table for ring counter

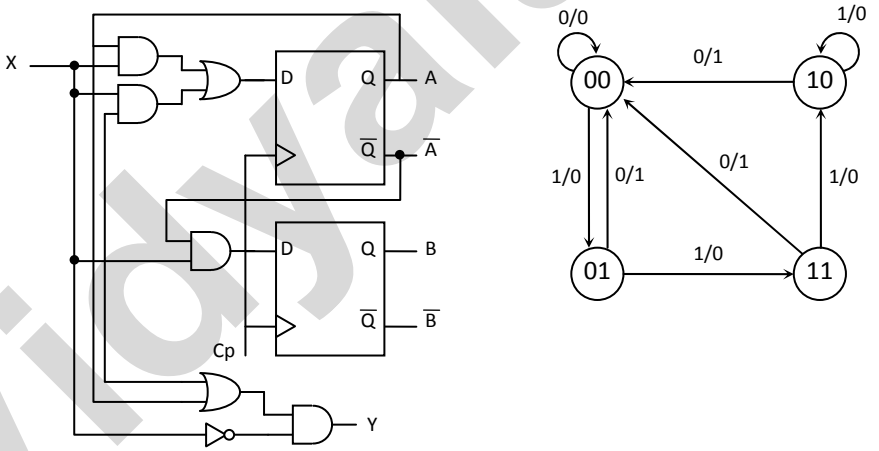
States	Present States				Next States				Input			
	Q_3	Q_2	Q_1	Q_0	Q_{3+1}	Q_{2+1}	Q_{1+1}	Q_{0+1}	D_3	D_2	D_1	D_0
1	0	0	0	1	0	0	1	0	0	0	1	0
2	0	0	1	0	0	1	0	0	0	1	0	0
4	0	1	0	0	1	0	0	0	1	0	0	0
8	1	0	0	0	0	0	0	1	0	0	0	1

Step 4: Implementation of 4 bit ring counter using D flip flop



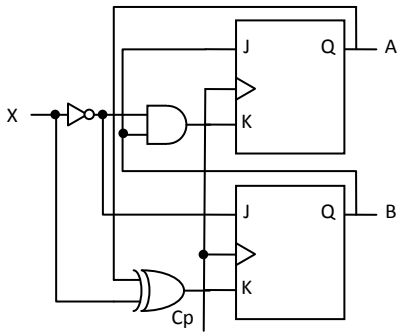
Q.6(d) Write short note on Moore and Mealy machine. [5]

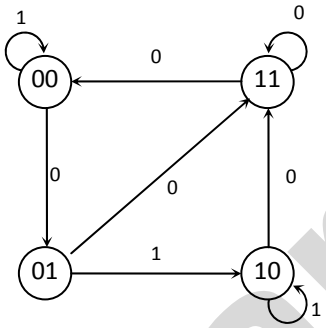
Ans.: Mealy machine :



- In Mealy model, outputs are functions of both the present state and inputs. An example of mealy model is shown in above figure.
- Output y is function of both input x and present state of A and B.
- The state table of a Mealy model sequential circuit must include an output section that is a function of both present state and inputs.
- In a Mealy model, outputs may change if inputs change during the clock-pulse period. In order to synchronized Mealy circuit, inputs of sequential circuit must be synchronized with clock and outputs must be sampled only during clock–pulse transition.

Moore Model :





State diagram of Moore model

- In Moore model, outputs are a function of present state only. An example of Moore model is shown above.
- Here the outputs are taken from the Flip–flops and are a function of present state only. The outputs of a Moore model can be a combination of flip–flop variables such as A $\oplus$ B. This output is a function of the present state only even though it requires an additional XOR gate to generate it.
- In a Moore model, the outputs of sequential circuit are synchronized with clock because they depends an only flip flop outputs that are synchronized with clock.

Q.6(e) Write short note on Priority encoder. [5]

Ans.:

- A priority encoder is an encoder circuit that includes the priority function. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having highest priority will take precedence.
- The truth table of a 4–input priority encoder is given as

Truth table of a Priority encoder :

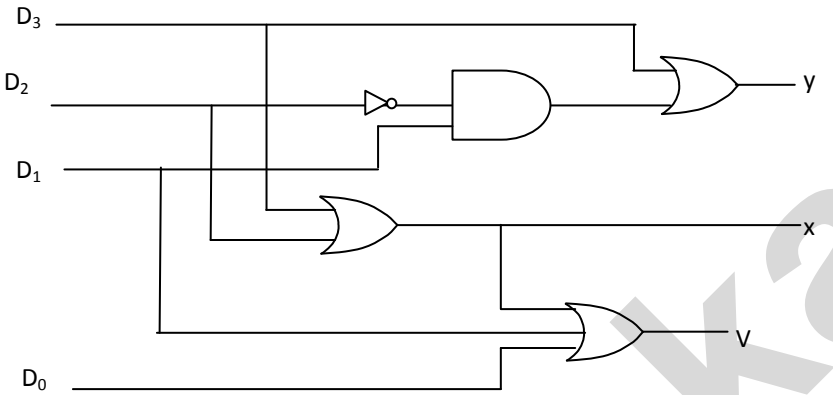
Inputs					Output		
D_0	D_1	D_2	D_3		x	y	v
0	0	0	0		X	X	0
1	0	0	0		0	0	1
X	1	0	0		0	1	1
X	X	1	0		1	0	1
X	X	X	1		1	1	1

- The X's are don't care conditions that designate the fact that binary value may be equal either to OD_1. Input D_3 has highest priority; so when this input is 1, the output for xy is 11 (binary 3).
- D_2 has the next priority level. The output is 10 if D_2 =1 provided that D_3=0; regardless of the values of other two lower–priority inputs.
- The output for D1 is generated only if where –priority inputs are 0.
- A valid –output indicator (v) is set to 1 only when one or more of the inputs are equal to 1. If all inputs are 0, V is equal to 0 and other & outputs of circuit are not used.
- The priority encoder is implemented using following Boolean functions.

$x = D_2 + D_3$

$y = D_3 + D_1 + \bar{D}_2$

$v = D_0 + D_1 + D_2 + D_3$



❑❑❑❑❑