KLS Vishwanathrao Deshpand Institute of Technology

Department of Electronics and Communication Engineering

Subject: Digital System Design using Verilog

Scheme and solution of Jan/Feb 2023

Subject code: 21EC32

Staff: Prof. Rohini Kallur

Staff Incharge
Prof. Rohini Kallur

Dr. Mahendra.M.Dixit
HOD, ECE

Head of the Department
Dept. of Electronic & Communication Engg.
KLS V.D.I.T., HALIYAL (U.K.)

Dr. Shreeniva. S
Dean, Academics

Dean, Academics
KLS VDIT, HALIYAL

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGG
KLS VDIT HALIYAL

CBCS SCHEME

USN | | | | | | | | | | |

21EC32

Third Semester B.E. Degree Examination, Jan./Feb. 2023
Digital System Design using Verilog

Time: 3 hrs. Max. Marks: 100

Note: ***Answer any FIVE full questions, choosing ONE full question from each module.***

Module-1

1 a. What are combinational circuits? Give example. Explain combinational circuit with block diagram. **(04 Marks)**

b. Define canonical form representation and solve the following equation using canonical form
i) $P = f(a,b,c) = ab' + ac' + bc$
ii) $G = f(w,x,y,z) = w'x + yz'$. **(08 Marks)**

c. Simplify the following Boolean function using K – Map
i) $D = f(x, y, z) = \Sigma m(0, 2, 4, 6)$
ii) $K = f(a, b, c) = \Sigma m(1, 2, 3, 6,7)$. **(08 Marks)**

OR

2 a. Define K-Map solve the following expression using K – Map.
i) $K = f(w, x, y, z) = \Sigma m(0, 1, 4, 5, 9, 11, 13, 15)$
ii) $D = f(a, b, c, d) = \Sigma m(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$. **(10 Marks)**

b. Define Quine-McClusky method and solve the following Boolean expression using Quine-McClusky method.
i) $D = f(a, b, c, d) = \Sigma m(0, 1, 2, 3, 6, 7, 8, 9, 14, 15)$
ii) $K = f(w, x, y, z) = \Sigma m(1, 3, 13, 15) + \Sigma d(8, 9, 10, 11)$. **(10 Marks)**

Module-2

3 a. Explain binary Adders with K–map and logical representation of equations for SUM and CARRY. **(06 Marks)**

b. Explain carry look ahead Adder with General and Sigma block. **(06 Marks)**

c. Explain working of decimal adder with neat block diagram (take example of BCD addition). **(08 Marks)**

OR

4 a. What are comparator circuits? Explain 2-bit magnitude comparators. **(08 Marks)**

b. Realize the Boolean expression using 3 : 8 decoder and two OR gates
i) $f_1(x_2, x_1, x_0) = \Sigma m(1, 2, 4, 5)$
ii) $f_2(x_2, x_1, x_0) = \Sigma m(1, 5, 7)$. **(06 Marks)**

c. Implement $D = (w, x, y, z) = \Sigma m(0, 1, 2, 4, 5, 7, 8, 9, 12, 13)$ using 8 : 1 MUX. **(06 Marks)**

Module-3

5 a. Write a note on Master Slave JK Flip-Flops with function table and timing diagram. **(08 Marks)**

b. What are Edge Triggered Flip-Flops. Explain positive edge Triggered and negative edge Triggered Flip-Flops. **(06 Marks)**

c. Write characteristic equation for : i) JK Flip-Flop ii) SR Flip-Flop. **(06 Marks)**

OR

6 a. Define Counters. Explain Binary Ripple counter with neat diagram. **(08 Marks)**

b. What are Registers? Explain any two classification registers with neat block diagram. **(06 Marks)**

c. Design synchronous MOD-6 counter using clocked JK Flip-Flops for sequences : 0 – 2 – 3 – 6 – 5 – 1. **(06 Marks)**

Module-4

7 a. Define HDL and types of HDL. Give structure of verilog module with example. **(06 Marks)**

b. Explain verilog logical operators with example. **(06 Marks)**

c. i) Write a note on verilog Data type

ii) Write verilog code for 8 × 1 MUX. **(08 Marks)**

OR

8 a. Give classification of Styles(Types) of description with example. **(08 Marks)**

b. Write verilog code for Full Adder. **(06 Marks)**

c. Write a note on Arithmetic and shift, Rotate relational operators with example. **(06 Marks)**

Module-5

9 a. Write a note on structure of Behavioural Description with example. **(08 Marks)**

b. Write a note on Signal Assignment and Variable Assignment with example. **(06 Marks)**

c. Write a note on sequential statement with example. **(06 Marks)**

OR

10 a. Write a verilog code for 2 × 1 MUX using if ELSE STATEMENT. **(06 Marks)**

b. Explain structural description with example. **(08 Marks)**

c. Explain structural description of 3-bit Ripple Carry Adder. **(06 Marks)**

* * * * *

Module 1

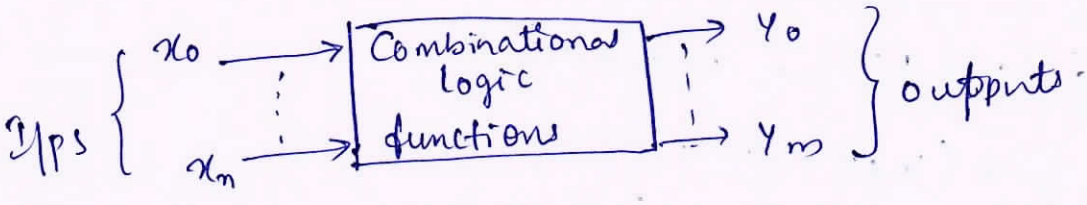
1)a) What are combinational circuits? Give example. Explain combinational circuit with block diagram (4M)

Soln Combinational logic deals with the techniques of combining the basic gates into circuits that perfor some desired function.

Logic circuits without feedback from output to the input, constructed from a functionally comple gate set, are said to be combinational circuits.

Example: Decoder, Encoders, multiplexers

Combinational circuit



Let X be the set of all input variables $\{x_0, x_1, \ldots x_n\}$ and Y be the set of all output variables $\{Y_0, Y_1, \ldots Y_m\}$. The combination function be F.

The output variables Y_0 through Y_m are not not fed back to the input.

$$Y = F(X)$$

1) b) Define Canonical form representation and solve the following equation using Canonical form

i) $P = f(a,b,c) = a\bar{b} + a\bar{c} + bc$

ii) $Q = f(w,x,y,z) = \bar{w}x + y\bar{z}$ (8M)

Soln: Canonical is a word used to describe a condition of a switching equation.

i) $P = f(a,b,c) = a\bar{b} + a\bar{c} + bc$

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

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

$bc = bc(a+\bar{a}) = abc + \bar{a}bc$

∴ The final Canonical SOP form is

$$P = f(a,b,c) = a\bar{b}c + a\bar{b}\bar{c} + ab\bar{c} + abc + \bar{a}bc$$

ii) $Q = f(w,x,y,z) = \bar{w}x + y\bar{z}$

$\bar{w}x = \bar{w}x(y+\bar{y})(z+\bar{z})$

$= (\bar{w}xy + \bar{w}x\bar{y})(z+\bar{z})$

$\bar{w}x = \bar{w}xyz + \bar{w}xy\bar{z} + \bar{w}x\bar{y}z + \bar{w}x\bar{y}\bar{z}$

$y\bar{z} = y\bar{z}(w+\bar{w})(x+\bar{x})$

$= (wy\bar{z} + \bar{w}y\bar{z})(x+\bar{x})$

$y\bar{z} = wxy\bar{z} + wx\bar{y}\bar{z} + \bar{w}xy\bar{z} + \bar{w}\bar{x}y\bar{z}$

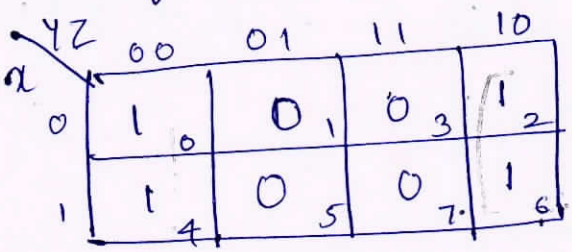
$$Q = f(w,x,y,z) = \bar{w}xyz + \bar{w}x\bar{y}\bar{z} + wxy\bar{z} + \bar{w}\bar{x}y\bar{z} + \bar{w}xy\bar{z} + \bar{w}x\bar{y}z + w\bar{x}y\bar{z}$$

1)c) Simplify the following using K-map

i) $D = f(x,y,z) = \Sigma m(0,2,4,6)$

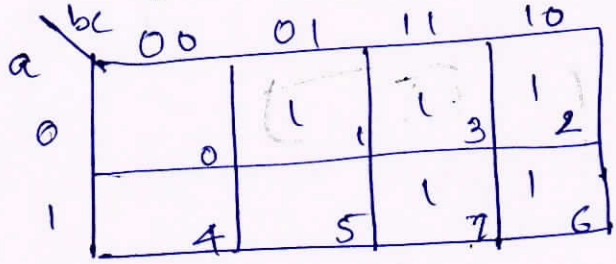
ii) $K = f(a,b,c) = \Sigma m(1,2,3,6,7)$ (8M)

i) $D = f(x,y,z) = \Sigma(0,2,4,6)$



$D = f(x,y,z) = \bar{z}$

ii) $K = f(a,b,c) = \Sigma m(1,2,3,6,7)$

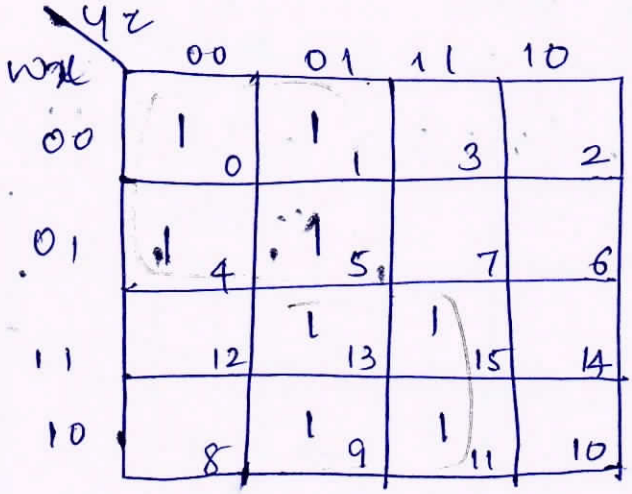


$K = f(a,b,c) = b + \bar{a}c$

2)a) Define K-map. Solve the following expression using K-Map

i) $K = f(w,x,y,z) = \Sigma m(0,1,4,5,9,11,13,15)$

ii) $D = f(a,b,c,d) = \Sigma m(0,1,2,4,5,6,8,9,12,13,14)$ (10M)

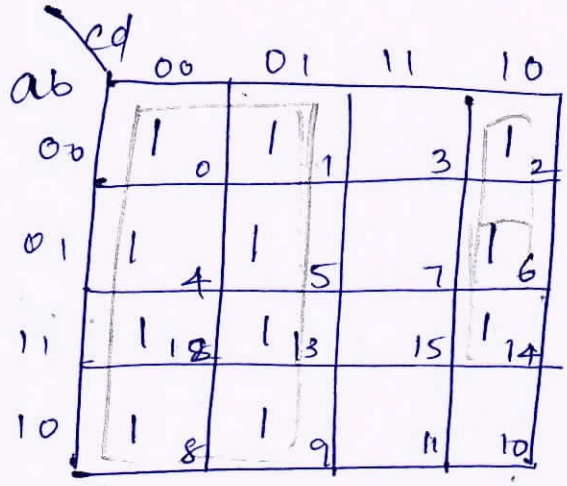


$K = f(w,x,y,z)$

$= \bar{w}\bar{y} + w\bar{z}$

Radha

ii) $D = f(a,b,c,d) = \Sigma m(0,1,2,4,5,6,8,9,12,13,14)$



$D = f(a,b,c,d)$

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

2b) Define Quine – McCluskey method and solve the following Boolean expression using Quine-McCluskey method.

i) $D = f(a,b,c,d) = \Sigma m(0,1,2,3,6,7,8,9,14,15)$

ii) $K = f(w,x,y,z) = \Sigma m(1,3,13,15) + \Sigma d(8,9,10,11)$

10 Marks

Soln:- Quine – McCluskey minimization technique is an algorithm that uses the same Boolean Algebra postulates that were used with Karnaugh maps but in a form suitable for a computer solution.

i) $D = f(a,b,c,d) = \Sigma m(0,1,2,3,6,7,8,9,14,15)$

Step 1: Group	Minterm	Variables a	b	c	d
0	0	0	0	0	0
1	1	0	0	0	1
	2	0	0	1	0
	8	1	0	0	0
2	3	0	0	1	1
	6	0	1	1	0
	9	1	0	0	1

Group	minterm	Variables a	b	c	d
3	7	0	1	1	1
	14	1	1	1	0
4	15	1	1	1	1

Step 2:

Group	Minterms	Variables a	b	c	d
0	0, 1	0	0	0	—
0	0, 2	0	0	—	0
0	0, 8	—	0	0	0
1	1, 3	0	0	—	1
1	1, 9	—	0	0	1
1	2, 3	0	0	1	—
1	2, 6	0	—	1	0
1	8, 9	1	0	0	—
2	3, 7	0	—	1	1
2	6, 7	0	1	1	—
2	6, 14	—	1	1	0
3	7, 15	—	1	1	1
3	14, 15	1	1	1	—

Step 3:

Group	Minterms	Variables a	b	c	d	
0	0, 1, 2, 3	0	0	–	–	$\bar{a}\bar{b}$
0	0, 1, 8, 9	–	0	0	–	$\bar{b}\bar{c}$
1	2, 3, 6, 7	0	–	1	–	$\bar{a}c$
2	6, 7, 14, 15	–	1	1	–	bc

Step 4:

PI terms	terms	Minterms 0	1	2	3	6	7	8	9	14	15
$\bar{a}\bar{b}$	0, 1, 2, 3	X	X	X	X						
$\bar{b}\bar{c}$	0, 1, 8, 9	X	X					X	X		
$\bar{a}c$	2, 3, 6, 7			X	X	X	X				
bc	6, 7, 14, 15					X	X			X	X

Step 5:

$D = f(a, b, c, d) = \bar{b}\bar{c} + bc$

ii) $k = f(w, x, y, z) = \Sigma m(1, 3, 13, 15) + \Sigma d(8, 9, 10, 11)$

Group	Minterms	Variables w	x	y	z
1	1	0	0	0	1
1	8*	1	0	0	0
2	3	0	0	1	1
2	9*	1	0	0	1
2	10*	1	0	1	0
3	11*	1	0	1	1
3	13	1	1	0	1
4	15	1	1	1	1

Group	minterms	minterms w	x	y	z
1	1, 9*	—	0	0	1
1	1, 3	0	0	—	1
1	8, 9*	1	0	0	—
1	8*, 10*	1	0	—	0
2	3, 11*	—	0	1	1
2	9*, 11*	1	0	—	1
2	10*, 11*	1	0	1	—
3	11*, 15*	1	—	1	1
3	13*, 15*	1	1	—	1

Group	Minterms	w	x	y	z
1	1, 3, 9*, 11*	—	0	—	1
1	8*, 9*, 10*, 11*	1	0	—	—
2	9*, 11*, 13*, 15*	1	—	—	1

Prime implicant Table

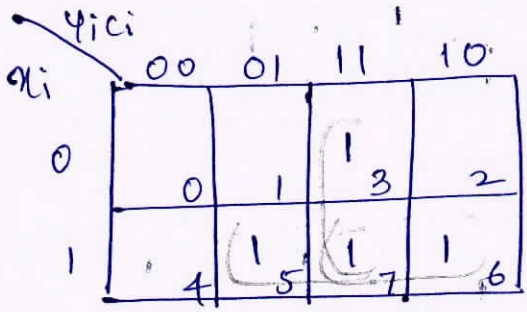
PI terms	Decimal	1	13	13	15
$\bar{x}z$	1, 3, 9, 11	✱	✱		
wz	9*, 13, 11*, 15			✱	✱

$$K = f(w, x, y, z) = \bar{x}z + wz$$

3.a) Explain Binary Adders with K-map and logical representation of equations for Sum and Carry. 06 Marks

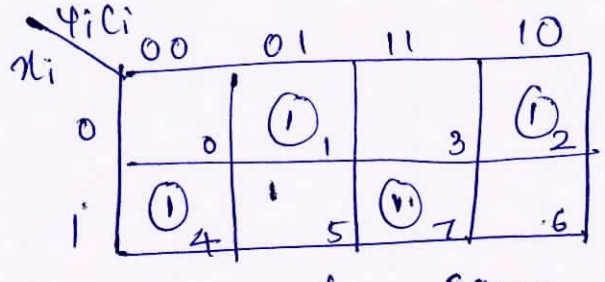
Soln: Full adder

x_i	y_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



K-map for carry, C_{i+1}

$$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$$



K-map for sum, S_i

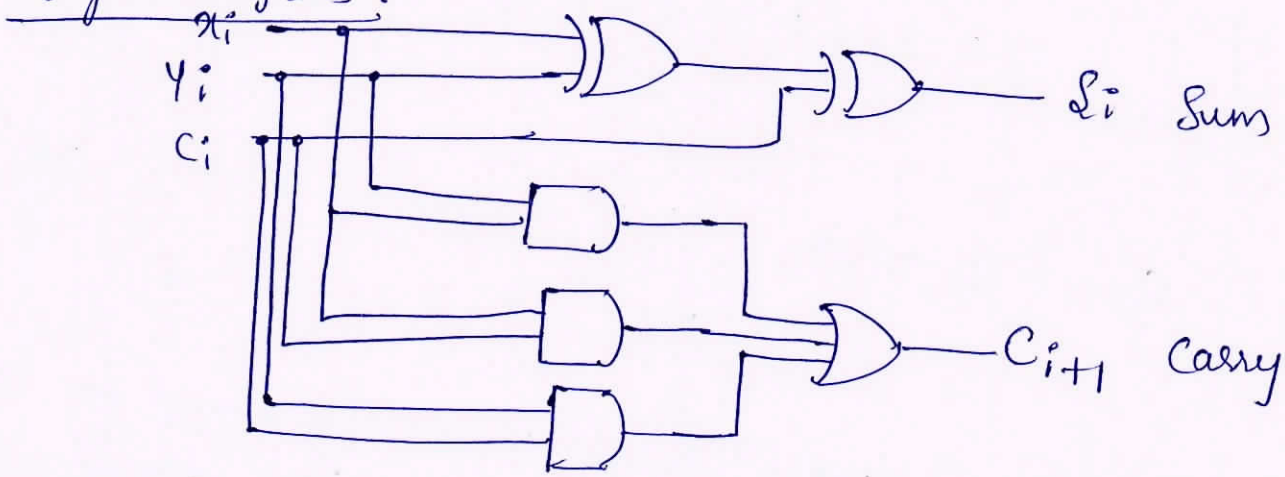
$$S_i = \bar{x_i}\,\bar{y_i}\,c_i + \bar{x_i}\,y_i\,\bar{c_i} + x_i\,\bar{y_i}\,\bar{c_i} + x_i\,y_i\,c_i$$

$$S_i = c_i(\bar{x_i}\,\bar{y_i} + x_i y_i) + \bar{c_i}(\bar{x_i}\,y_i + x_i\,\bar{y_i})$$

$$= c_i\,(\overline{x_i \oplus y_i}) + \bar{c_i}\,(x_i \oplus y_i)$$

$$\boxed{S_i = c_i \oplus x_i \oplus y_i}$$

Logic diagram:



3)b) Explain carry look ahead adder with General Sigma block

06 Marks.

Carry equation of full adder can be written as,

$$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$= x_i y_i + (x_i + y_i)\,c_i$$

$$C_{i+1} = g_i + p_i c_i$$

$g_i = x_i y_i$ = Carry generate function

$P_i = (x_i + y_i)$ = Carry propogate function.

→ The output carry function for the ith stage is given by,

$$C_{i+1} = g_i + p_i c_i$$

Dattu

$$C_1 = g_0 + P_0 C_0$$

$$C_2 = g_1 + P_1 C_1$$

$$= g_1 + P_1 (g_0 + P_0 C_0)$$

$$C_2 = g_1 + P_1 g_0 + P_1 P_0 C_0$$

$$C_3 = g_2 + P_2 C_2$$

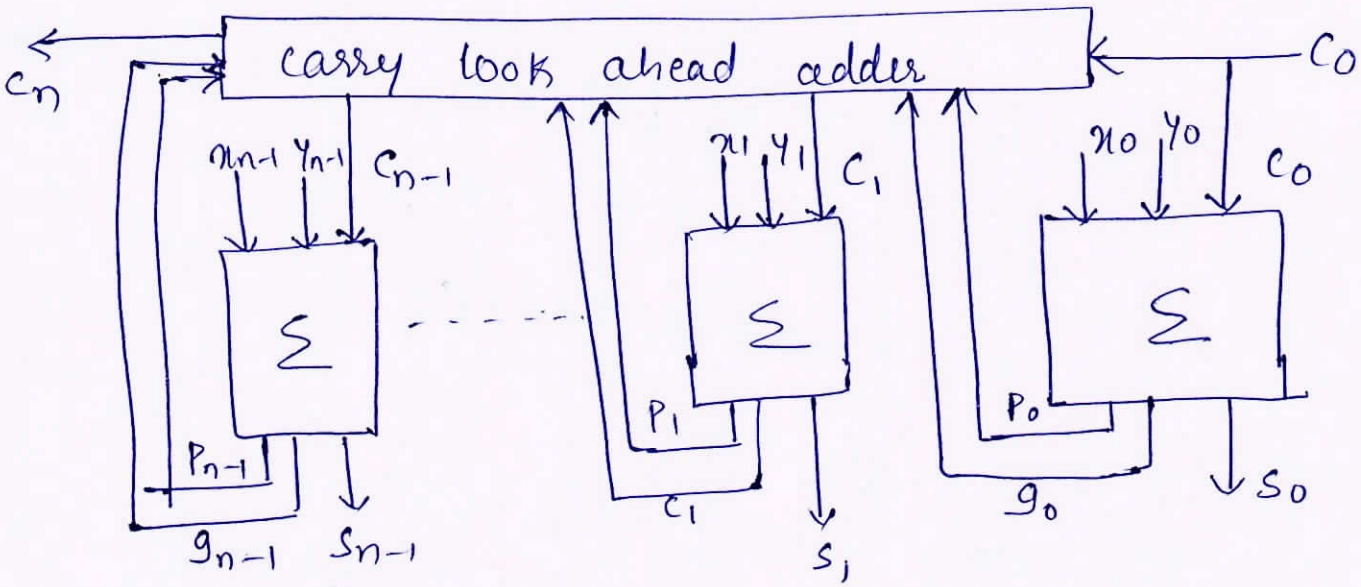
$$= g_2 + P_2 (g_1 + P_1 g_0 + P_1 P_0 C_0)$$

$$C_3 = g_2 + P_2 g_1 + P_2 P_1 g_0 + P_2 P_1 P_0 C_0$$

$$\vdots$$

$$C_{i+1} = g_i + P_i g_{i-1} + P_i P_{i-1} g_{i-2} + \cdots + P_i P_{i-1} \cdots P_1 g_0 + P_i P_{i-1} \cdots P_0 C_0 \quad \text{— (x)}$$

Parallel adders whose realisations are based on the above equations are called carry look ahead adders



Fig(a) carry look ahead adder General organization

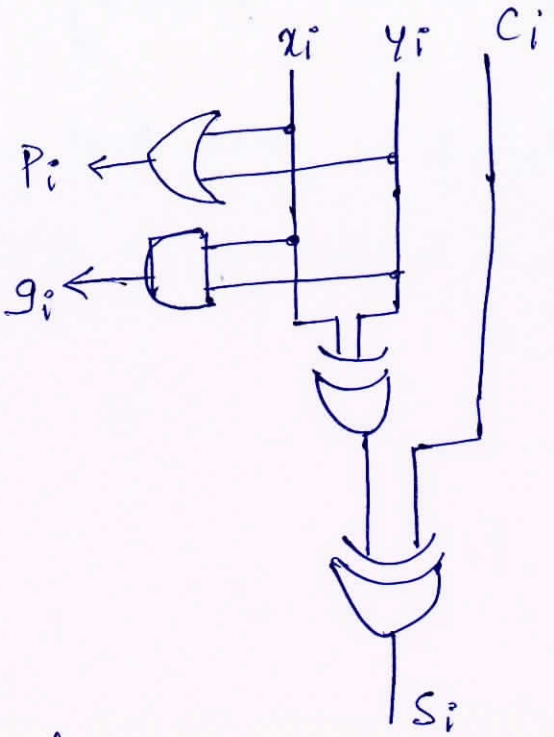
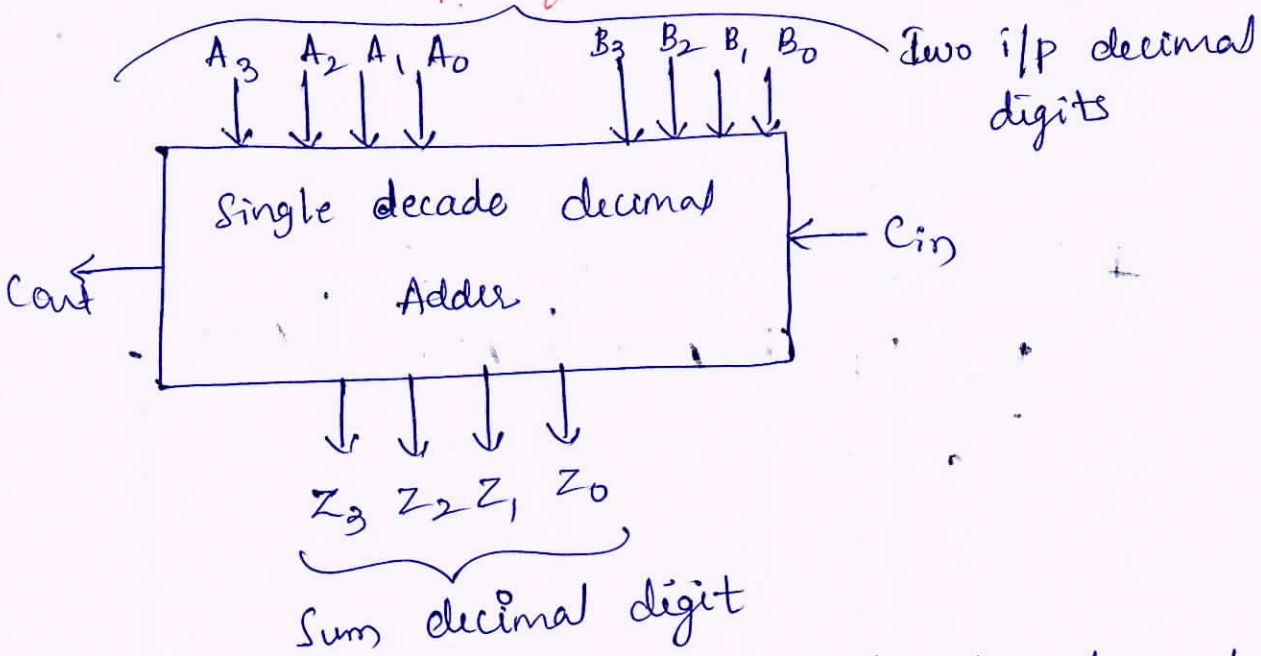


fig (b) Sigma blocks

3)c) Explain working of decimal adder with a neat block diagram (take example of BCD addition) 08 Marks

Soln:



The general form of a single decade decimal adder i.e, an adder corresponding to a single order digit position is given in the above figure.

→ Here two decimal digits serving as operands are denoted by $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ and

a carry denoted by C_{in} is also appearing as an input.

The outputs are $Z_3 Z_2 Z_1 Z_0$ and Cout

if one of I/p is 1001, & another i/p is 1000

$$\begin{array}{r} 1001 \\ +\ 1000 \\ \hline (1)\ 0001 \\ \hline \text{Cout}\ \ Z_3 Z_2 Z_1 Z_0 \end{array}$$

4)a) what are comparator circuits? Explain 2-bit magnitude comparators?

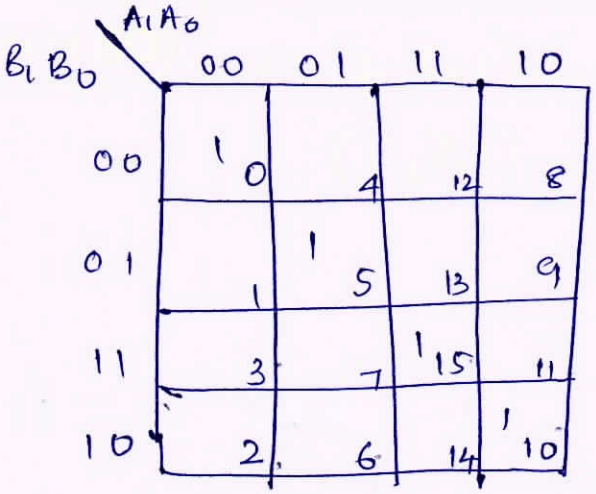
Soln: Comparator is a circuit to compare the magnitudes of two binary numbers for the purpose of establishing whether one is greater than, equal to or less than the other.

2-Bit magnitude Comparator →

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

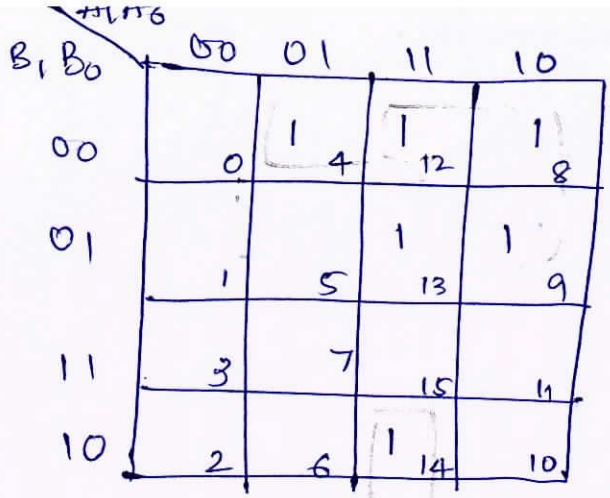
Two-bit magnitude comparator

Inputs				Outputs		
A_1	A_0	B_1	B_0	$A>B$	$A=B$	$A<B$
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0



(a) $A = B$

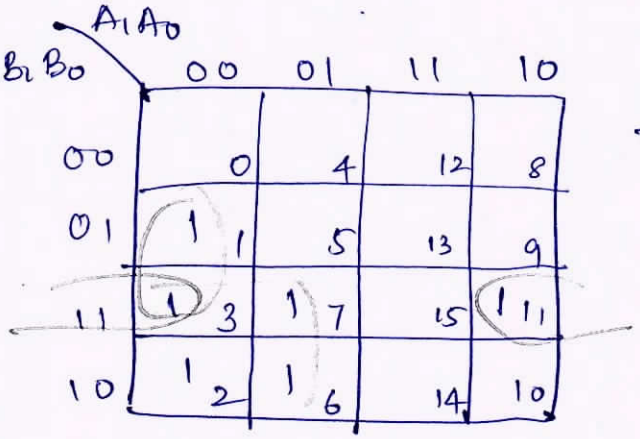
$A = B = f(A_1, A_0, B_1, B_0) = \Sigma(0, 5, 10, 15)$



$A \geq B = f(A_1, A_0, B_1, B_0)$

$= \Sigma(4, 8, 9, 12, 13, 14)$

(b) $A > B$



$A < B = f(A_1, A_0, B_1, B_0)$

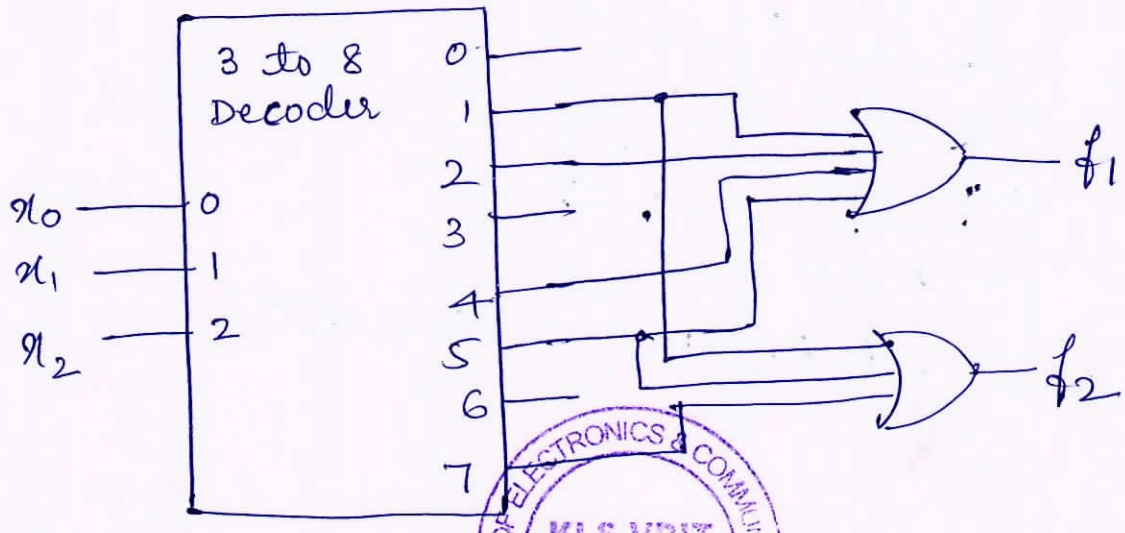
$= \Sigma(1, 2, 3, 6, 7, 11)$

(c) $A \leq B$

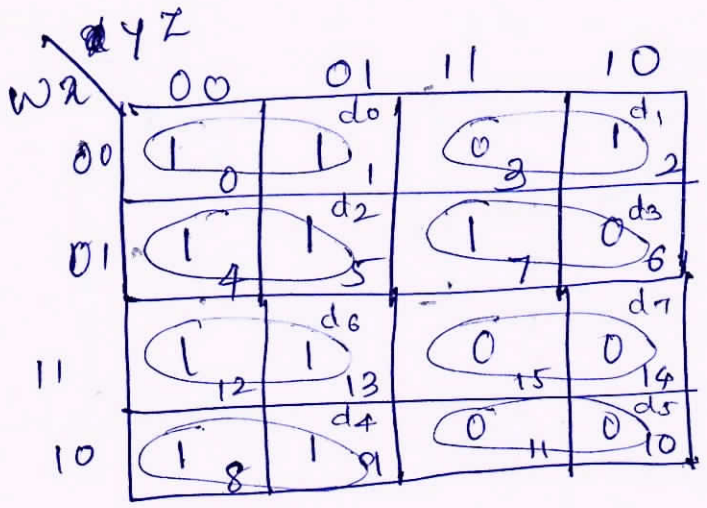
4)b) Realize the Boolean expression using 3:8 decoder and two OR gates.

i) $f_1(x_2, x_1, x_0) = \Sigma m(1, 2, 4, 5)$ (06 Marks)

ii) $f_2(x_2, x_1, x_0) = \Sigma m(1, 5, 7)$



4)c) Implement $D = f(w,x,y,z) = \Sigma m(0,1,2,4,5,7,8,9,12,13)$ using 8:1 MUX — 06 Marks



$d_0 = 1$

$d_1 = \overline{z}$

$d_2 = 1$

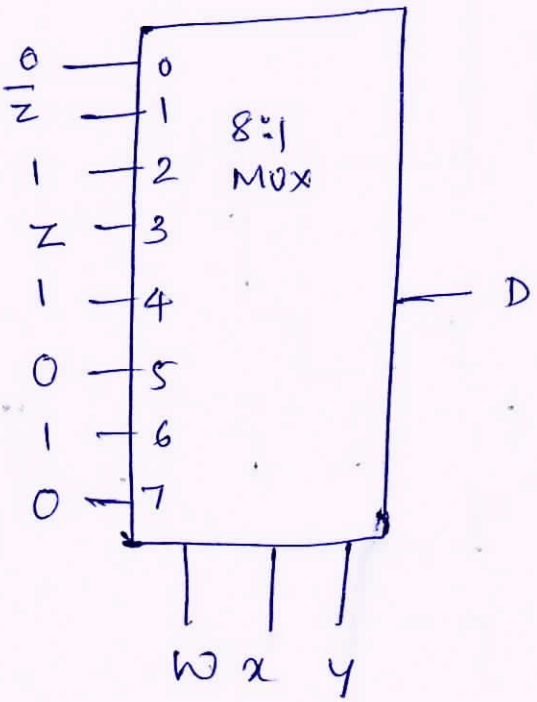
$d_3 = z$

$d_4 = 1$

$d_5 = 0$

$d_6 = 1$

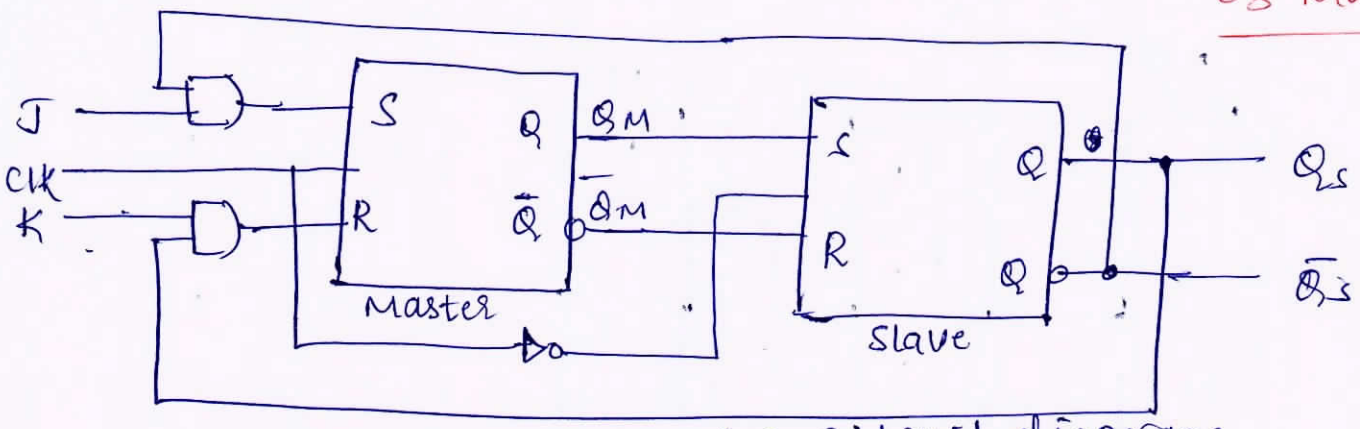
$d_7 = 0$



Pradhan

5)a) Write a note on Master Slave JK flip-flop with function table and timing diagrams.

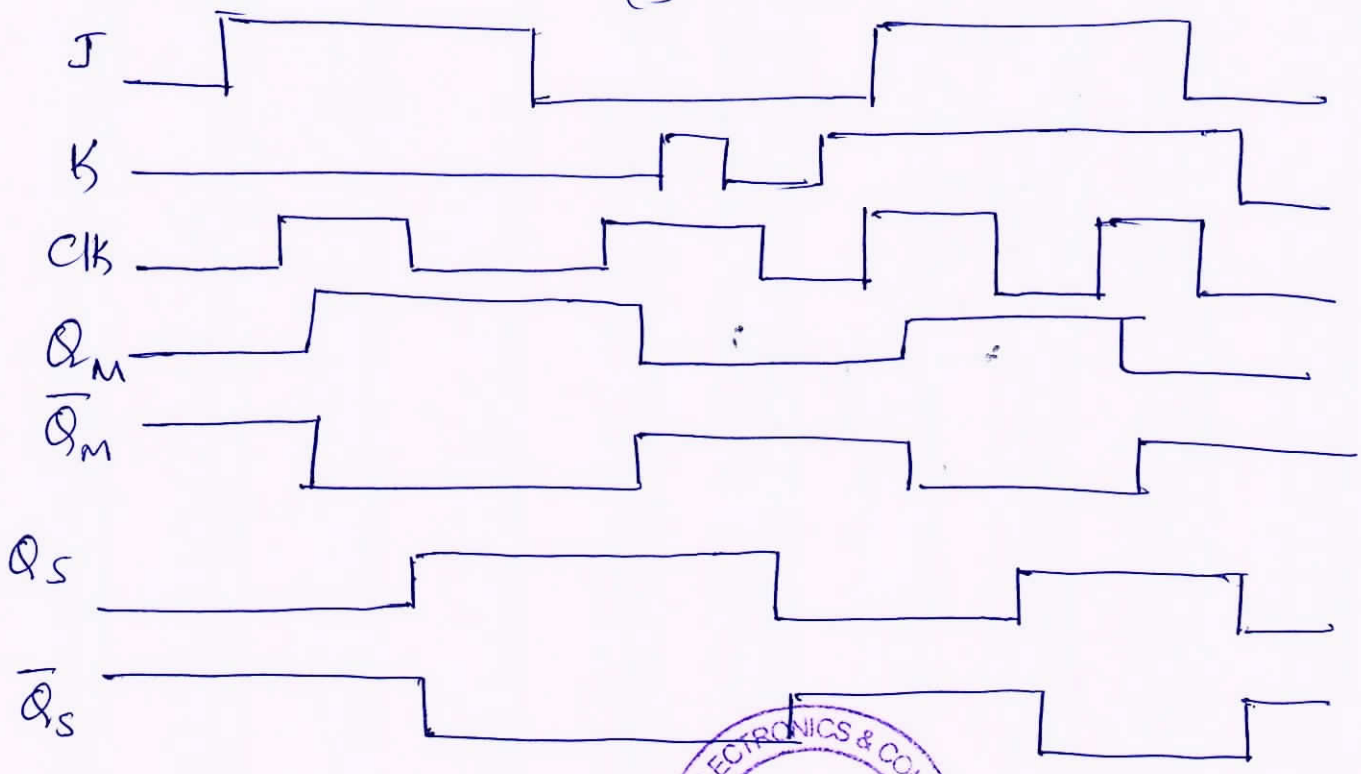
08 Marks



(a) Circuit diagram

Inputs			o/ps	
J	K	CLK	Q	Q^+
0	0	⊓	Q	$\bar{Q}$
0	1	⊓	0	1
1	0	⊓	1	0
1	1	⊓	$\bar{Q}$	Q
X	X	0	Q	$\bar{Q}$

(b) Truth table



Timing diagram

* The master slave JK flip-flop, on the other hand, does allow its two information input lines to be simultaneously 1. This results in the toggling of the o/p of the flip-flop.

If the present state is 0, then the next state is 1, while if the present is 1, then the next state is 0.

* Assume the master slave JK flip flop of figure above is in its 1-state, the control signal i.e the clock is 0 and that $J = K = 1$. Thus the master and slave latches are both in the 1-state with $Q = Q_S = 1$ & $\overline{Q} = \overline{Q}_S = 0$.

* Now assume that master slave JK flip flop is in its 0-state, again $J = K = 1$ and the control signal i.e the clock is low. Thus $Q = Q_S = 0$ and $\overline{Q} = \overline{Q}_S = 1$

* Assume the master slave JK flip flop is in its 1-state when the clock is low. Thus $Q = Q_S = 1$ & $\overline{Q} = \overline{Q}_S = 0$.

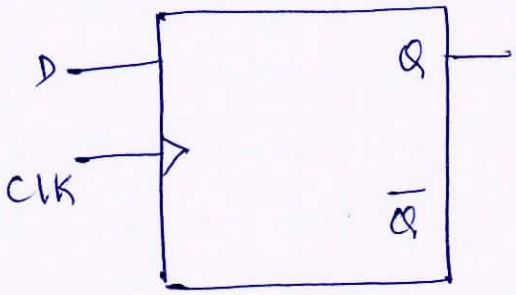
* On the other hand, if the master slave latches are in their 0-states when $J = 1, K = 0$ and the clock is low, then $Q = Q_M = Q_S = 0$ & $\overline{Q} = \overline{Q}_M = \overline{Q}_S = 1$

In figure (c) two symbols are shown for the master slave JK flip-flop.

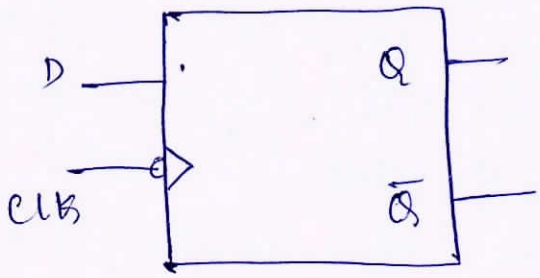
5.b) what are Edge triggered flip-flops. Explain Positive edge triggered and negative edge triggered flip-flops. 06 Marks

Ans: Edge triggered flip flops →

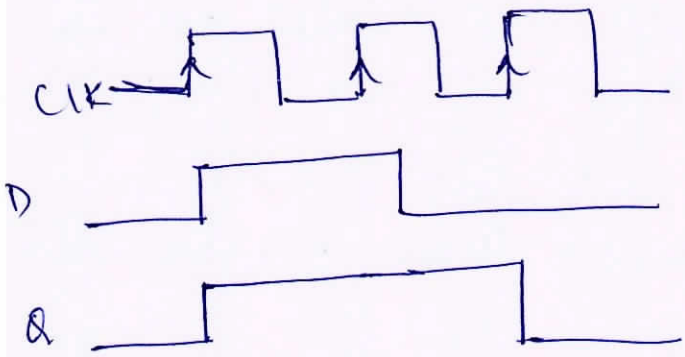
flip flop is a bistable device, with inputs, that remains in a given state as long as power is applied and until input signals are applied to cause its output to change. Positive and negative edge triggered flip flops get triggered on positive and negative edge of ~~flip flop~~ clock signal.

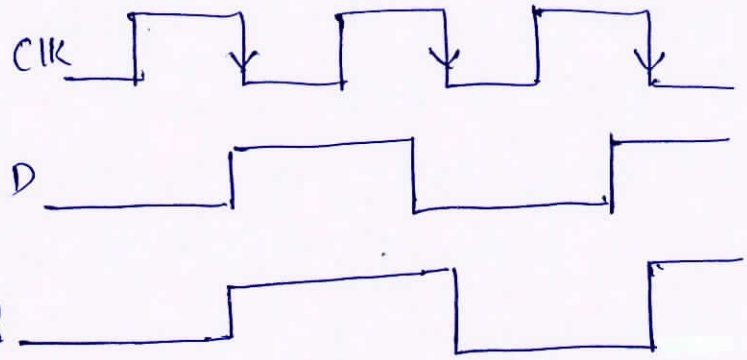


Positive edge triggered flip flop



negative edge triggered flip flop





(b) waveforms

5-C) Write characteristics equation for
i) JK flip-flop ii) SR flip-flop 06 Marks

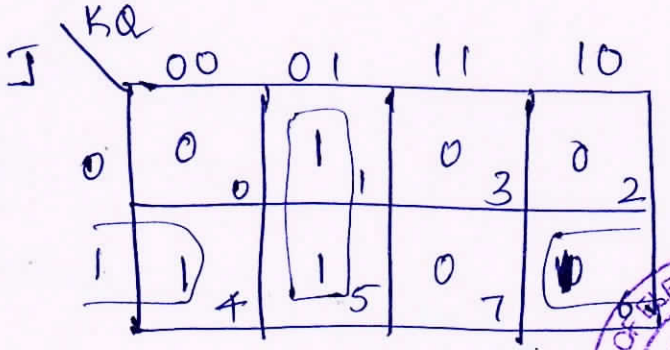
Soln: i) JK flip-flop

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\bar{Q}$

(a) Flip flop function tables

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	0	0	1
1	1	1	0

(b) Flip-flop next state tables



Characteristic Equation

$Q^+ = J\bar{Q} + \bar{K}Q$

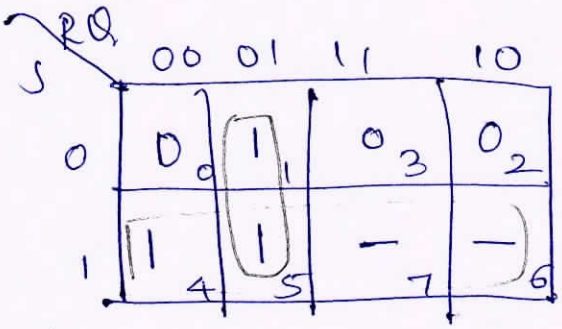
ii) SR flip-flop

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	—

(a) function table

S	R	Q	Q^+	
0	0	0	0	I/p n allowe
0	0	1	1	
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	1	0	—	
1	1	1	—	

(b) flip-flop next state tables



K-map

Characteristic Equation:

$$Q^+ = S + \bar{R}Q$$

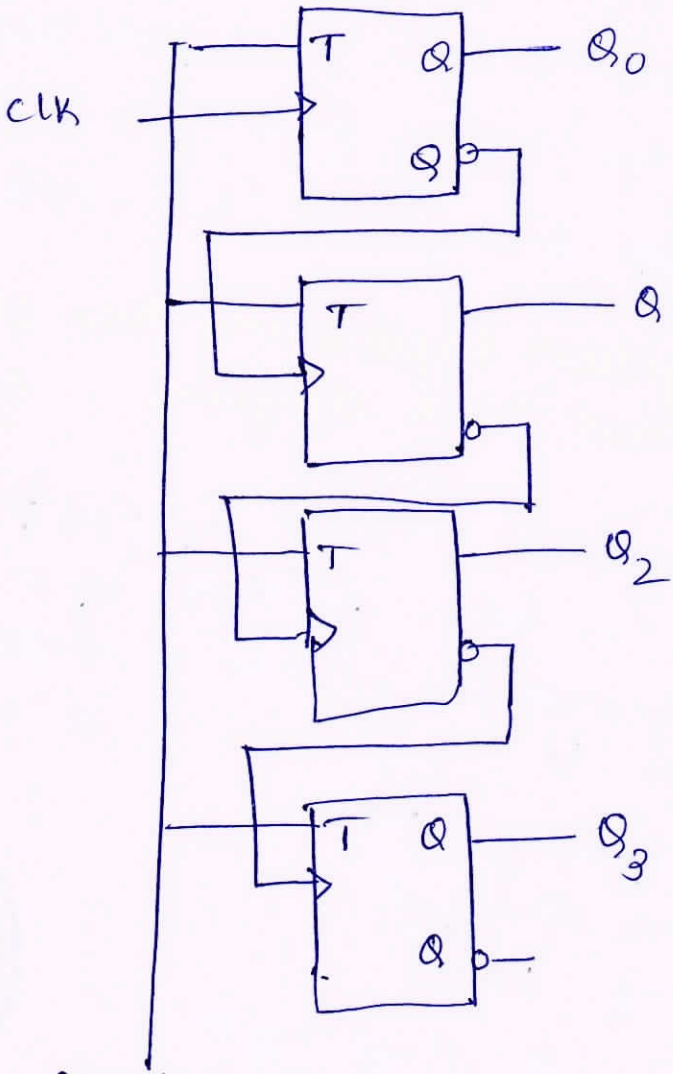
6.a) Define counters. Explain Binary Ripple counter with neat diagram. 08 Marks

Soln: Counter is another example of a register. Its primary function is to produce a specified o/p pattern sequence. For this reason it is also called as Pattern generator.

Binary ripple counter

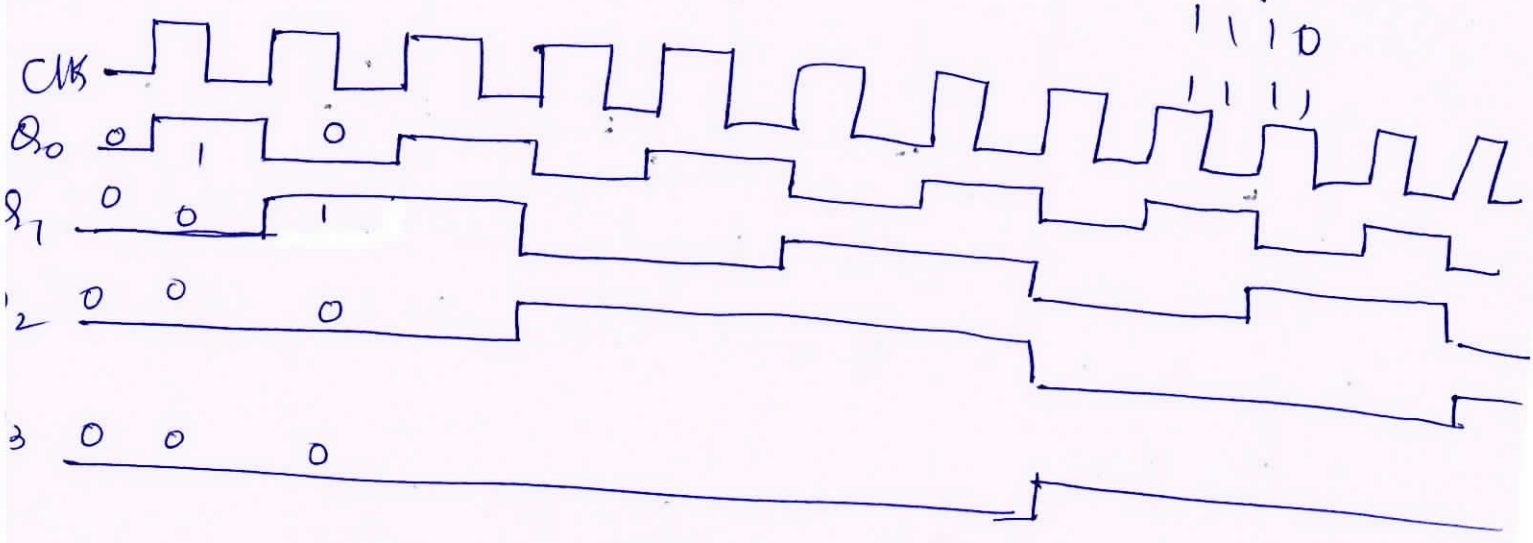
The figure below shows four bit binary up counter implemented with positive edge triggered T flip flops. In this way, each positive transition from logic 0 to logic 1 on the clock terminal causes the counter

flip-flop to toggle.



(a) logic diagram

Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1



Figures illustrate the counters behaviour - The counter is assumed to be initially in its 0000 state and the count enable signal is logic 1. upon the occurance of the positive edge of the first count pulse, the Q_0 flip-flop changes to its 1-state.

6)b) What are registers? Explain any two classification Registers with neat block diagram. 06 Marks

Soln: Register is simply a collection of flip flops taken as an entity. The basic function of a register is to hold information within a digital system so as to make it available to the logic elements during the computing process.

Shift registers are classified into

SISO, SIPO, PISO, PIPO.

Serial In serial out shift register

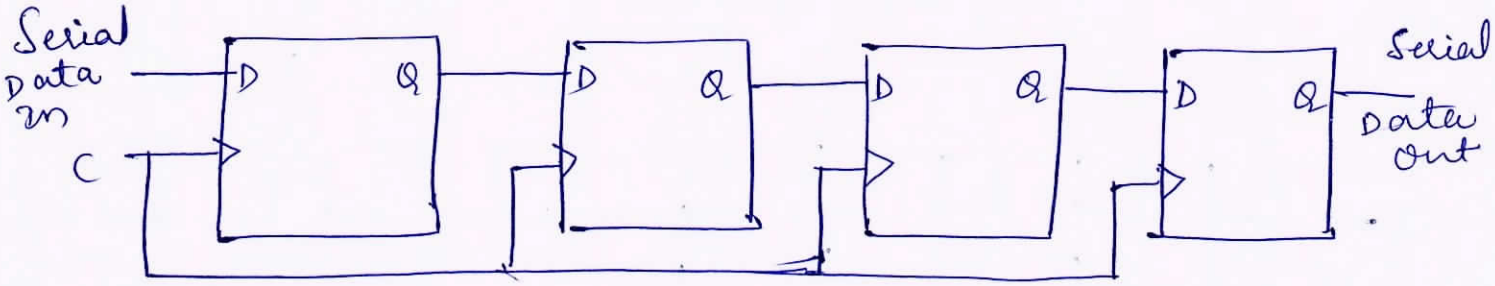
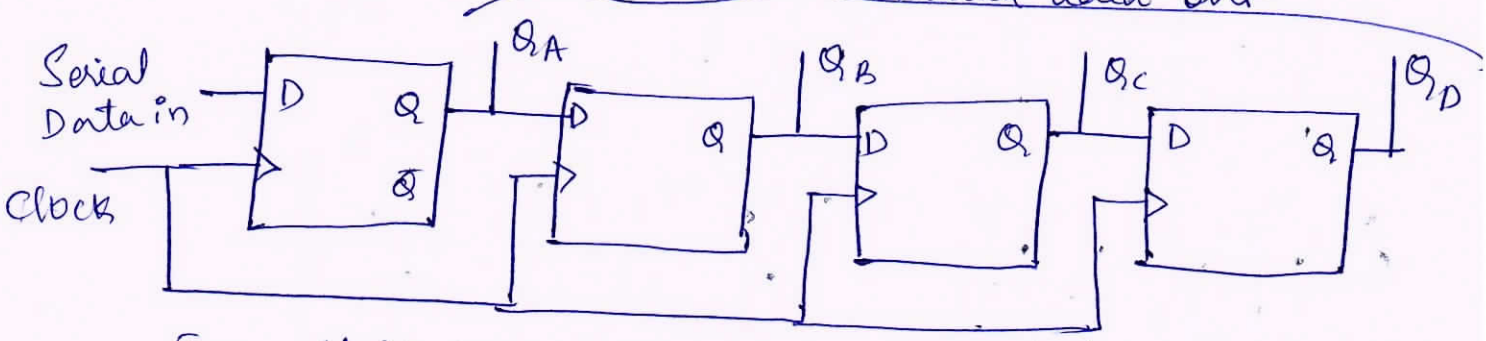


Figure above illustrates the serial in serial out shift register constructed from positive edge triggered D flip-flops.

The Q o/p of each flip flop is connected to the D i/p of the flip-flop to its right. The control i/ps of all the flipflops are connected together to a common signal called clock.

thus upon the occurance of a positive edge of the clock signal, the content of each flip flop is shifted one position to the right.

Serial in parallel out shift register



In this case, o/ps are provided from each flip-flop. Once information is shifted into the register i.e serial-in, the information is available as a single entity i.e parallel out, at the flip-flop output terminals. Since information is transferred into this register serially and after an appropriate number of shifts, made available in parallel, this type of register provides for the serial to parallel conversion of information.

6)c) Design Synchronous MOD-6 counter using clocked JK flip-flops for sequences: 0-2-3-6-5-1

06 Marks

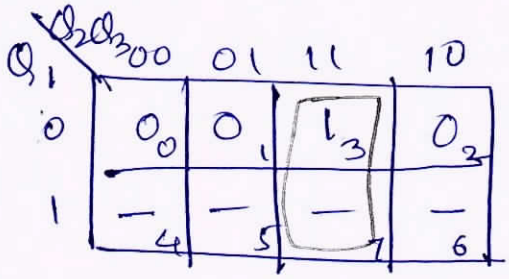
Soln:

Present state			Next state			Flip-Flop inputs					
Q_1	Q_2	Q_3	Q_1^+	Q_2^+	Q_3^+	J_1	K_1	J_2	K_2	J_3	K_3
0	0	0	0	1	0	0	—	1	—	0	—
0	1	0	0	1	1	0	—	—	0	1	—
0	1	1	1	1	0	1	—	—	0	—	1
1	1	0	1	0	1	—	0	—	1	1	—
1	0	1	0	0	1	—	1	0	—	—	0
0	0	1	0	0	0	0	—	0	—		

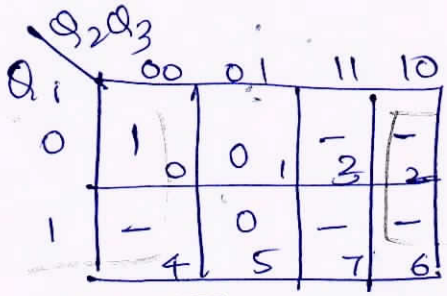
the

Excitation table of JK flip flop:

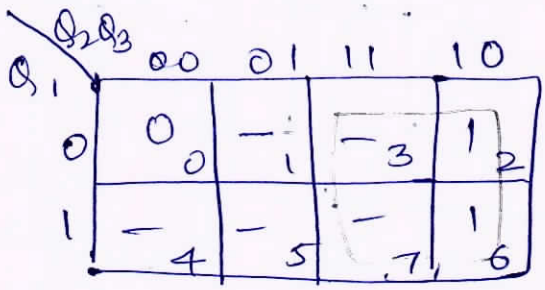
Q	Q^+	J	K
0	0	0	—
0	1	1	—
1	0	—	1
1	1	—	0



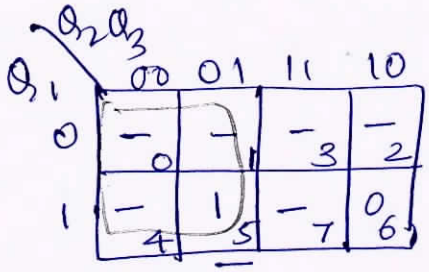
$J_1 = Q_2 Q_3$



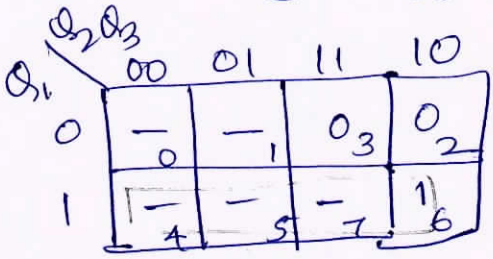
$J_2 = \overline{Q_3}$



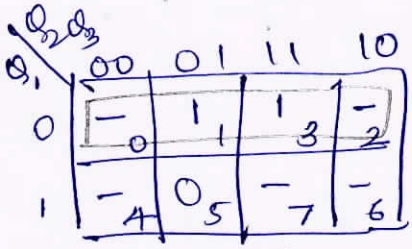
$J_3 = Q_2$



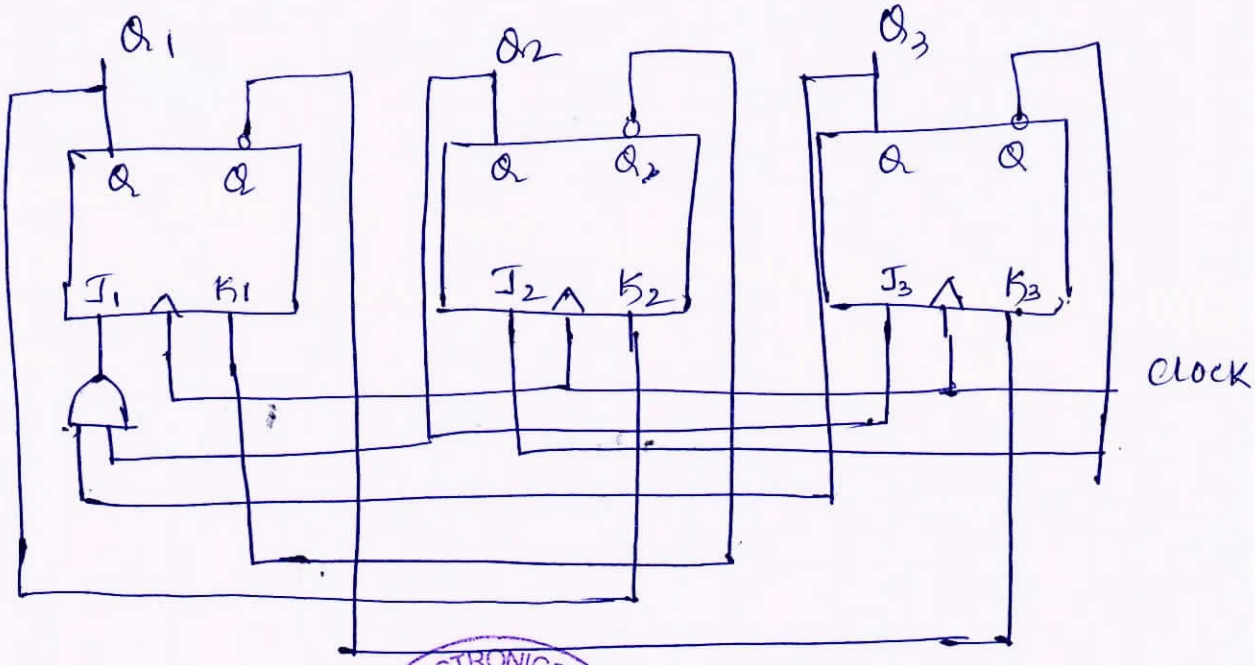
$K_1 = \overline{Q_2}$



$K_2 = Q_1$



$K_3 = \overline{Q_1}$



7.a) Define HDL and types of HDL. Give structure of Verilog module with example — 06 Marks

Soln: Hardware Description language (HDL) is a computer Aided Design (CAD) tool for the modern design and Synthesis of digital systems.

Types of HDL: 1) Verilog HDL 2) VHDL

Structure of Verilog module :-

```
module half-adder (I1, I2, O1, O2);
    input I1;
    input I2;
    output O1;
    output O2;
// Blank lines are allowed
    assign O1 = I1 ^ I2; // Statement 1
    assign O2 = I1 & I2; // Statement 2
endmodule
```

* Verilog is case sensitive.
* declaration of the module starts with a predefined word module, followed by a user selected name.
* The name of i/ps and o/ps follow the same guidelines as the module's name.
* The i/p, o/p ports are written inside paranthesis and are seperated by commas. The closing paranthesis is followed by a semicolon.
* SemiColon (;) is a line seperator.
* The double slashes (//) signifies a comment command.

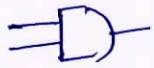
7.b) Explain Verilog logical operators with example.

06 marks

Soln: Verilog has extensive logical operators. These operators perform logical operations such as AND, OR & EX-OR.

* Verilog logical operators can be classified into three groups: Bitwise, Boolean logical and Reduction Operators

Bitwise Operators

Operator	Equivalent logic	operand type	Result type
&		Bit	Bit
|		Bit	Bit
~(&)		Bit	Bit
~(|)		Bit	Bit
^		Bit	Bit
~^		Bit	Bit
~		Bit	Bit

* Bitwise operators are operated on bits of the operands

Boolean logical operators

Operator	Operation	Number of operands
&&	AND	two
||	OR	two

Reduction operators

Operator	Operation	Number of operands
&	Reduction AND	one
|	Reduction OR	one
~&	Reduction NAND	one
~|	Reduction NOR	one
^	Reduction Ex-OR	one
~^	Reduction Ex-NOR	one
!	Negation	one

* These operators operate on a single operand.

7.C. i) write a note on Verilog Data Type
ii) write Verilog code for 8x1 MUX

08 Marks

Soln: i) Verilog Data Type

Verilog supports several data types including nets, registers, vectors, integer, real, parameters and arrays.

Nets:- Nets are declared by predfined word wire

Value	Definitions
0	logic 0
1	logic 1
X	unknown
Z	High impedance

Registers:

Registers store values untill they are updated.

Value	Definitions
0	logic 0
1	logic 1
X	unknown
Z	High impedance

Vectors :-

Vectors are multiple bits. which are declared by brackets [ ]

Integers :-

Integers are declared by the predefined word integer.

Real :-

Real numbers are declared with the predefined word real.

Parameters :-

Parameters represent global constants. They are declared by the predefined word Parameter.

Arrays :-

Verilog does not have a predefined word for array.

ii) Verilog code for 8x1 MUX

```
module mux8_1(en, a, sel, y);
  input en;
  input [7:0] a;
  output reg y;
  input [2:0] sel;
  always @(a, sel, en)
  begin
    if (en==1)
case (sel)
  3'd0 : y = a[0];
  3'd1 : y = a[1];
  3'd2 : y = a[2];
  3'd3 : y = a[3];
  3'd4 : y = a[4];
  3'd5 : y = a[5];
  3'd6 : y = a[6];
  3'd7 : y = a[7];
default : y = 1'bx;
end case
else y = 1'bz;
 end
endmodule
```

8.a) Give classification of styles (Types) of description with example — 08 Marks

Soln: The styles of description in verilog can be classified as: Behavioural, Structured, Switch level, data flow, mixed type/style or mixed language.

i) Behavioral discription

```
module half-add (I1, I2, O1, O2);
input I1, I2;
output O1, O2;
reg O1, O2;
always @ (I1, I2)
begin
  #10 O1 = I1 ^ I2;
  # 10 O2 = I1 & I2;
end
endmodule
```

iii) Structural discription

```
module System (a, b, Sum, Cout);
input a, b;
output Sum, Cout;
xor x1 (Sum, a, b);
and a1 (Cout, a, b);
endmodule
```

iii) Data flow discription

```
module halfadder (a, b, S, C);
input a, b;
output S, C;
assign S = a ^ b;
assign C = a & b;
endmodule
```

8.b) Write Verilog code for full adder.

Soln: 06 Marks.

```
module full_adder (x, y, Cin, S, Cout);
  input x, y, Cin;
  Output S, Cout;
  assign {Cout, S} = Cin + x + y;
endmodule
```

8.c) Write a note on Arithmetic and Shift, Rotate relational operators with example

Soln: 06 Marks

Shift and rotate operators are implemented in many applications.

Operation	Description	Operand A Before Shift	Operand A after Shift
A <<1	Shift A one Position left logical	1110	110x
A <<2	Shift A two Positions left logical	1110	10xx
A >>1	Shif A one Position right logical	1110	x111
A >>2	Shift A two Positions right logical	1110	xx11

9-a) Write a short note on structure of Behavioural Description with example.

08 Marks

Soln:

```
module half_add (I1, I2, O1, O2);
  input I1, I2;
  output O1, O2;
  reg O1, O2;
always @ (I1, I2)
 begin
  #10 O1 = I1 ^ I2;
  #10 O2 = I1 & I2;
 end
endmodule
```

* Program above shows a simple example of HDL code describing a half adder using behavioural descri-ption
* Referring to the above code, 'always' is the verilog behavioral statement. All verilog statements inside always are treated as concurrent, the same as in the data flow description.
* Any signal that is declared as an o/p should also be declared as a register (reg) if it appears inside always.

@Lathe

Q.6) write a note on Signal Assignment and Variable assignment with example.

06 marks.

Ans:- Assign statements are used to drive values on signals of type wire or a data type require the continuous assignment of a value.

* This concept is realized by the assign statement where any wire or other similar wire can be driving continuously with a value.

Syntax:

assign <net_expression> = [drive_strengths] [delay]
< expression of different signals or constant value]

Example:

```
assign a = x & y;

wire [1:0] a = x & y;

assign o = ~((a & b) | c ^ d);
```

* To illustrate the difference between signal & variable assignments statements in Verilog code, the behavioral description of a D-latch is written.

```
module D_latch (d, E, Q, Qb);
  input d, E;
  output Q, Qb;
  reg Q, Qb;
always @ (d, E)
  begin
    if (E == 1)
    begin
      Q = d;
      Qb = ~Q;
    end
end
endmodule
```


Qualified

9.c) Write a note On Sequential Statement with example. 06 Marks

Soln: Sequential Statements

1) If Statement

If is a sequential Statement that appears inside always or initial in verilog. It has several formats

```
if (Boolean Expression)
begin
   Statement 1;
   Statement 2;
end
else
begin
   Statement a;
   Statement b;
end
```

2) Else-if:

```
if (Boolean expression)
begin
   Statement 1; Statement 2;
end
else if (Boolean expression 2)
begin
   Statement i; Statement j;
end
else
begin
   Statement a; Statement b;
end
```


Chandhini

Example of If Statement :-

```
if (clk == 1)
begin
    temp = S1;
end
```

Example of else-if statement :-

```
if (Signal1 == 1)
temp = S1;
else if (Signal2 == 1)
  temp = S2;
else
temp = S3;
```

10.a) write a verilog code for 2×1 MUX using if else Statement — 06 marks

Soln:

```
module mux21 (A, B, SEL, Qbar, Y);
  input A, B, SEL, Qbar; Y;
  output Y;
   reg Y;
always @ (SEL, A, B, Qbar)
begin
    if ( Qbar == 0 & SEL == 1)
begin
      Y = B;
end
else if (Qbar == 0 & SEL == 0)
      Y = A;
else
```

```
    y = 1'bz;
  end
endmodule
```

10.b) Explain structural description with example 08 Marks.

Soln:

```
module adder (a, b, Sum, Cout);
 input a, b;
 Output Sum, Cout ;
 xor  x1 ( Sum, a, b);
/* x1 is an optional identifier; it can be omitted*
 and  a1 ( cout, a, b);
endmodule
```

* The above code describes a system using structural description. It has two inputs, a & b and two outputs Sum and Cout.
* Verilog has a large number of built in gates for example : xor, and, or, not

```
xor x1 ( sum, a, b);
```

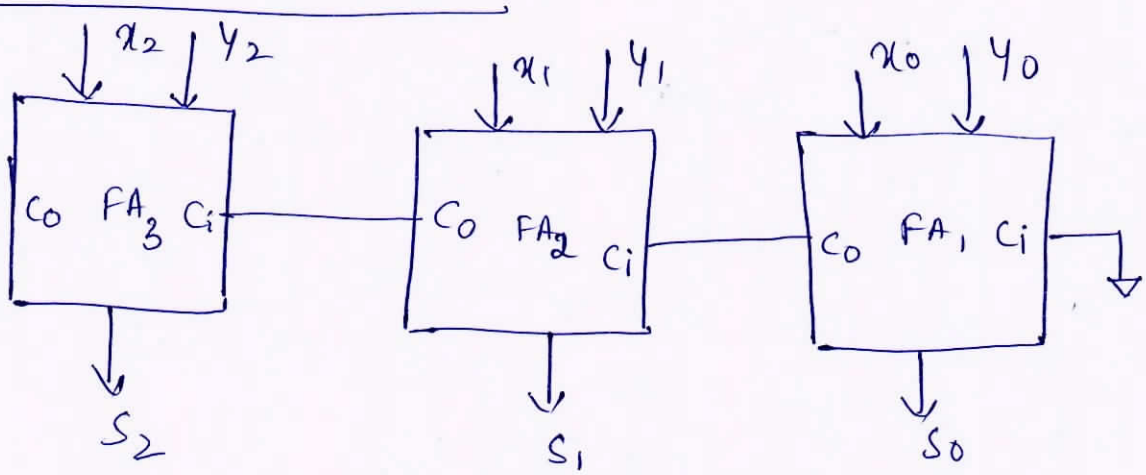
describes a two input XOR gate. The inputs are a & b and the output is Sum.

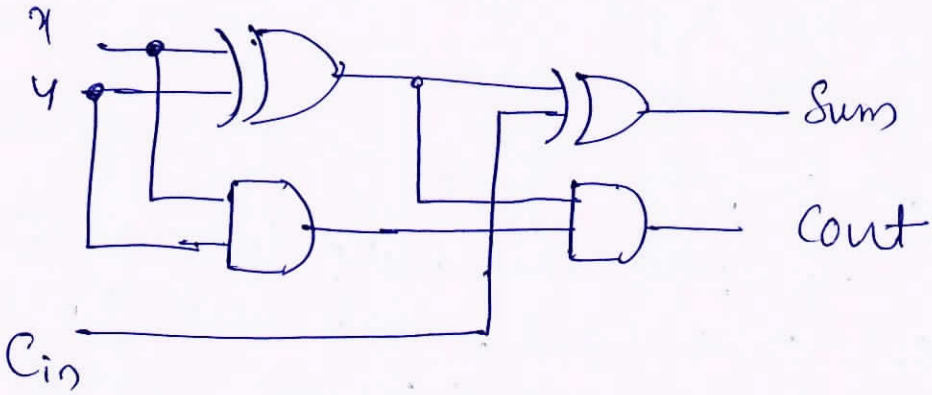
* Verilog has complete set of built in primitive gates
* Structural description can easily describe these components. On the other hand; it is hard to describe the digital logic of say harmonic Secretion in the blood. So we can use behavioural description in such case.

* Structural description simulates the system by describing its logical components. The components can be gate level such as AND gate, OR gate, or NOT Gate. or components can be in a higher logic level. Such as Register Transfer Level (RTL) or Processor level.
* It is more convenient to use structural description rather than behavioural description for system that required a specific design.

10.c. Explain structural description of 3-bit Ripple carry adder. 06 MARKS

Soln: 3-bit Ripple carry Adder





Prabhu

```
module fulladder (x, y, ci, s, co);
  input x, y, ci;
  output s, co;
wire w1, w2, w3;
  xor g1 (w1, x, y);
  xor g2 (s, w1, ci);
  and g3 (w2, w1, ci);
  and g4 (w3, x, y);
  or g5 (co, w2, w3);
endmodule

module ripple_adder (x, y, s, co);
  input [2:0] x, y;
  output [2:0] s;
  output co;
wire w1, w2;
 fulladder u1 (x[0], y[0], 1'b0, s[0], w1);
 full adder u2 (x[1], y[1], w1, s[1], w2);
 full adder u3 (x[2], y[2], w2, s[2], co);
 endmodule.
```