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

Internal Assessment Test 2 –October 2019

Sub:	Analog and Digital Electronics					Sub Code:	18CS33	Branch:	CSE
Date:	15/10/2019	Duration:	90 min's	Max Marks:	50	Sem/Sec:	3rd / A,B,C		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	**Explain the operation of a read-only memory (ROM). Realize Hexadecimal-to- ASCII Code Converter using ROM**	[10]	CO4	L2,L3

Answer:

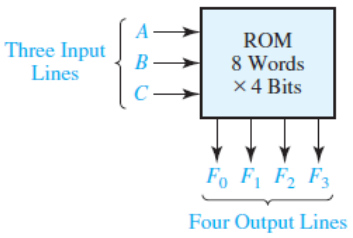
A read-only memory (ROM) consists of an array of semiconductor devices that areinterconnected to store an array of binary data. And can be read anytime and cannot be changed under normal operating conditions.

Figure (a) shows a ROM which has three input lines and four output lines.Figure (b) shows a typical truth table which relates the ROM inputs and outputs.

1. For each combination of input values on the three input lines, the corresponding 4bitadderttern of 0's and 1's appears on the ROM output lines.

For example, if the combination *ABC*=010 is applied to the input lines, the 4bitadderttern *F*0*F*1*F*2*F*3 =0111 appears on the output lines.

2. Each of the output 4bitaddertterns that is stored in the ROM is called a *word*.
3. Because the ROM has three input lines, we have 2^3 eight different combinations of input values.
4. Each input combination serves as an *address* which can select one of the eight words stored in the memory.
5. Because there are four output lines, each word is four bits long, and the size of this ROM is8 words x 4 bits.
6. A ROM which has *n*input lines and *m* output lines contains an array of 2^n words, and each word is *m* bits long.
7. The input lines serve as an address to select one of the 2^nwords.
8. When an input combination is applied to the ROM, the 4bitadderttern of 0's and 1's which is stored in the corresponding word in the memory appears at the output lines.
9. Typical sizes for commercially available ROMs range from 32 words x 4 bits to 512K words x 8 bits, or larger.

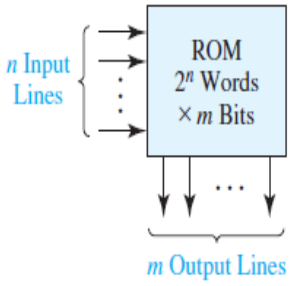


(a) Block diagram

A B C	F_0 F_1 F_2 F_3
0 0 0	1 0 1 0
0 0 1	1 0 1 0
0 1 0	0 1 1 1
0 1 1	0 1 0 1
1 0 0	1 1 0 0
1 0 1	0 0 0 1
1 1 0	1 1 1 1
1 1 1	0 1 0 1

Typical Data Stored in ROM (2^3 words of 4 bits each)

(b) Truth table for ROM



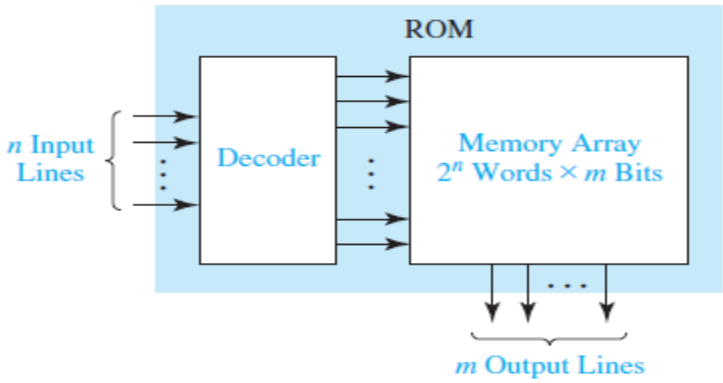
n Input Variables	*m* Output Variables
00 · · · 00	100 · · · 110
00 · · · 01	010 · · · 111
00 · · · 10	101 · · · 101
00 · · · 11	110 · · · 010
⋮	⋮
11 · · · 00	001 · · · 011
11 · · · 01	110 · · · 110
11 · · · 10	011 · · · 000
11 · · · 11	111 · · · 101

Typical Data Array Stored in ROM (2^n words of *m* bits each)

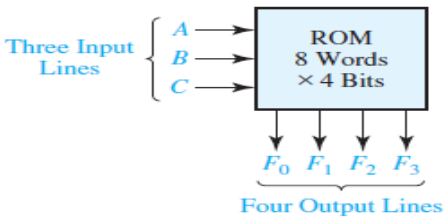
Basic ROM structure:

A ROM basically consists of a decoder and a memory array, as shown in Figure below When a 4bitadderttern of *n* 0's and 1's is applied to the decoder inputs, exactly one of the 2^n decoder outputs is 1. This decoder output line selects one of the words in the memory array, and the bit 4bitadderttern stored in this word is transferred to the memory output lines

FIGURE 9-19
Basic ROM Structure



Internal Architecture of ROM
Example: 8 words x 4 bits



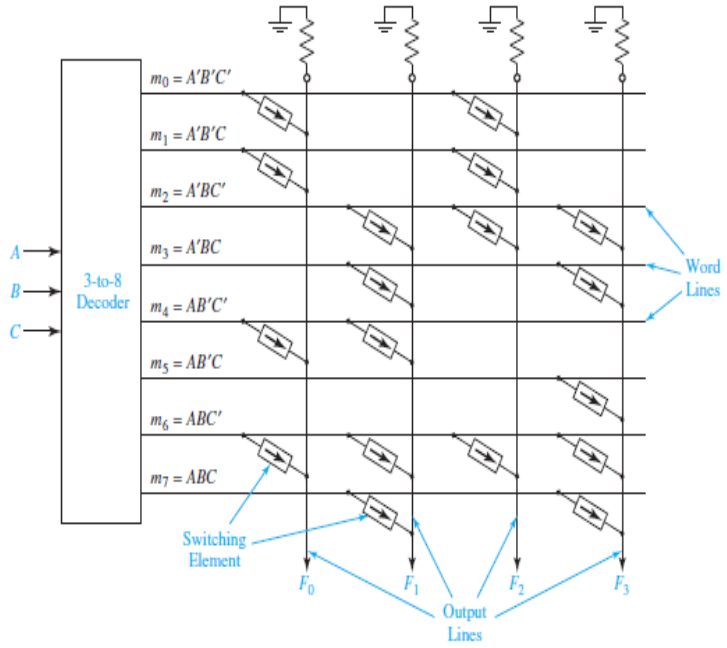
(a) Block diagram

A	*B*	*C*	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

Typical I
Stored i
ROM
(2^3 word
4 bits ea

(b) Truth table for ROM

FIGURE 9-20
An 8-Word × 4-Bit ROM



1. Internal structure of the 8-word x4-bit ROM shown in Figure above
2. The decoder generates the eight minterms of the three input variables.
3. The memory array forms the four output functions by ORing together selected minterms.
4. A switching element is placed at the intersection of a *word line* and an *output line* if the corresponding minterm is to be included in the output function; otherwise, the switching element is omitted (or not connected).
5. If a switching element connects an output line to a word line which is 1, the output line will be 1. Otherwise, the pulldown resistors at the top of Figure above cause the output line to be 0.
6. So the switching elements which are connected in this way in the memory array effectively form an OR gate for each of the output functions.

For example,
*m*0, *m*l, *m*4, and *m*6 are ORed together to form *F*0. Figure above shows the equivalent OR gate. In general, those minterms which are connected to output line *F* by switching elements are ORed together to form the output *Fi*.
Thus, the ROM in Figure above generates the following functions:

$$F_0 = \Sigma\, m(0,1,4,6) = A'B' + AC'$$
$$F_1 = \Sigma\, m(2,3,4,6,7) = B + AC'$$
$$F_2 = \Sigma\, m(0,1,2,6) = A'B' + BC'$$
$$F_3 = \Sigma\, m(2,3,5,6,7) = AC + B$$

Realize Hexadecimal-to- ASCII Code Converter using ROM

FIGURE 9-22
Hexadecimal-to-ASCII Code Converter

Input $W\ X\ Y\ Z$	Hex Digit	ASCII Code for Hex Digit $A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$
0 0 0 0	0	0 1 1 0 0 0 0
0 0 0 1	1	0 1 1 0 0 0 1
0 0 1 0	2	0 1 1 0 0 1 0
0 0 1 1	3	0 1 1 0 0 1 1
0 1 0 0	4	0 1 1 0 1 0 0
0 1 0 1	5	0 1 1 0 1 0 1
0 1 1 0	6	0 1 1 0 1 1 0
0 1 1 1	7	0 1 1 0 1 1 1
1 0 0 0	8	0 1 1 1 0 0 0
1 0 0 1	9	0 1 1 1 0 0 1
1 0 1 0	A	1 0 0 0 0 0 1
1 0 1 1	B	1 0 0 0 0 1 0
1 1 0 0	C	1 0 0 0 0 1 1
1 1 0 1	D	1 0 0 0 1 0 0
1 1 1 0	E	1 0 0 0 1 0 1
1 1 1 1	F	1 0 0 0 1 1 0

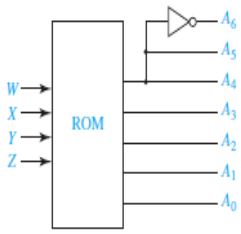


FIGURE 9-23
ROM Realization of Code Converter

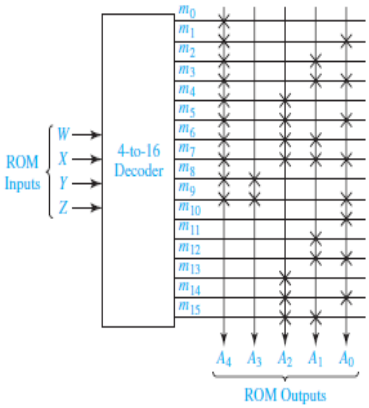


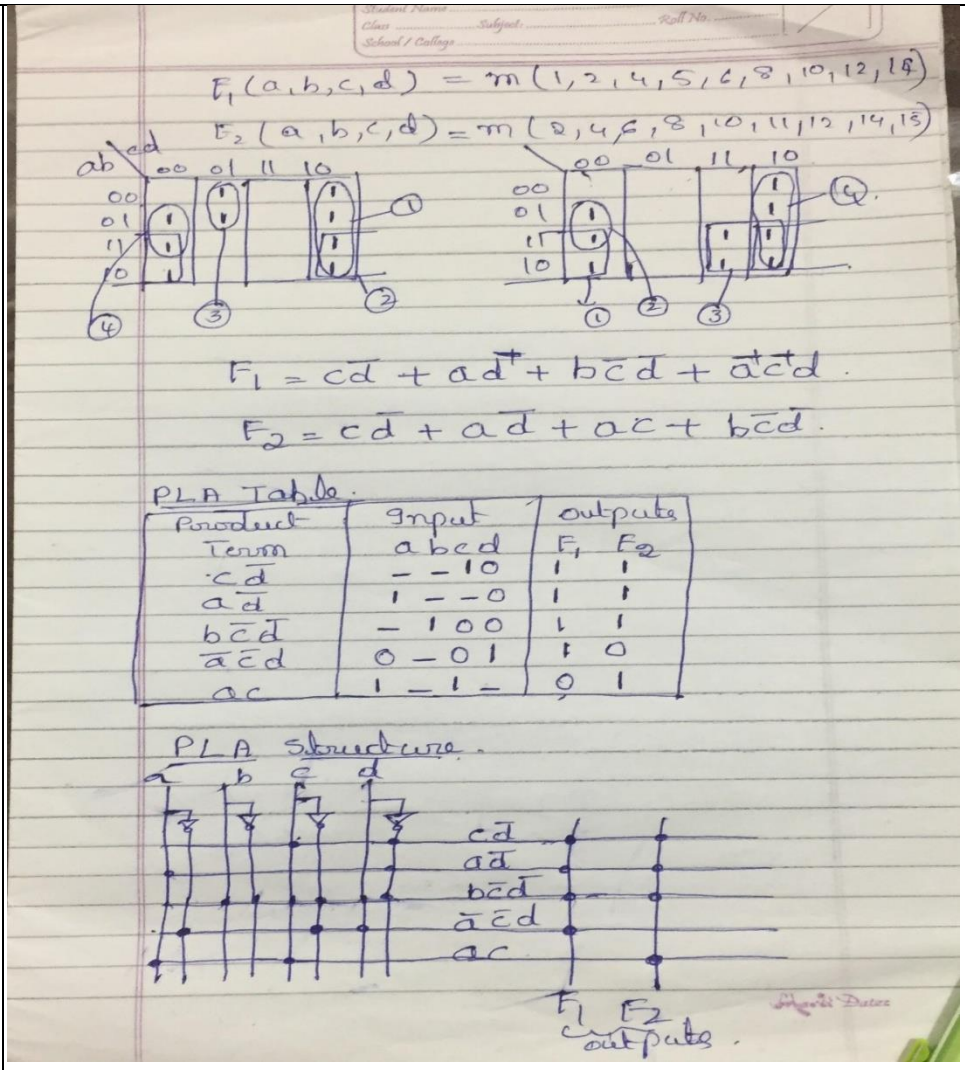
Figure shows the truth table and logic circuit for the converter.

1. Because $A5 = A4$, and $A6 = A4$, the ROM needs only five outputs.
2. Because there are four address lines, the ROM size is 16 words by 5 bits.
3. Columns $A4A3A2A1A0$ of the truth table are stored in the ROM

Figure 9-23 shows an internal diagram of the ROM.

1. The switching elements at the intersections of the rows and columns of the memory array are indicated using X's.
2. An X indicates that the switching element is present and connected, and no X indicates that the corresponding element is absent or not connected.

2	**Find a minimum two-level NOR gate circuit to realize *F*1 and *F*2. Use as many common gates as possible.** $F1(a, b, c, d) = m(1, 2, 4, 5, 6, 8, 10, 12, 14)$ $F2(a, b, c, d) = m(2, 4, 6, 8, 10, 11, 12, 14, 15)$ **Realize *F*1 and *F*2 using a PLA. Give the PLA table and internal connection diagramfor the PLA.** **Answer:**	[10]	CO4	L3

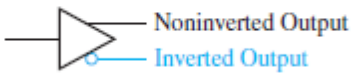


3	**Explain the operation of a Programmable Array logic . Implement Full Adder using PAL**	[10]	CO4	L2,L3

Answer:

1. The PAL (programmable array logic) is a special case of the programmable logic array in which the AND array is programmable and the OR array is fixed.
2. The basic structure of the PAL is the same as the PLA shown in Figure below. Because only the
3. AND array is programmable, the PAL is less expensive than the more general PLA,and the PAL is easier to program.

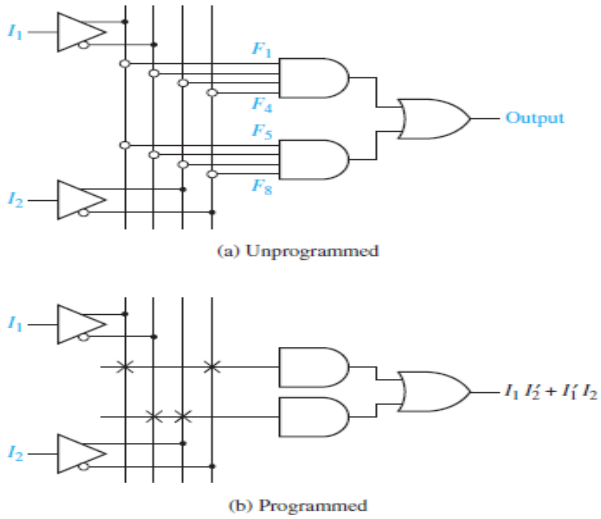
Figure 9-28(a) represents a segment of an unprogrammedPAL.The symbol



A buffer is used because each PAL input must drive many AND gate inputs. When the PAL is programmed, some of the interconnection points are programmed to make the desired connections to the AND gate inputs. Connections to the AND gate inputs in a PAL are represented by X's as shown:

For example:
To realize the function $I1I2 + I1\ I2$.

FIGURE 9-28
PAL Segment



Note:
we must simplify logic equations and try to fit them into one (or more) of the available PAL.
Unlike the more general PLA, the AND terms cannot be shared among two or more OR gates; therefore, each function to be realized can be simplified by itself **without regard to common terms.**
For a given type of PAL the number of AND terms that feed each output OR gate is fixed and limited.

<u>Implement Full Adder using PAL</u>

Implement a full adder. The logic equations for the full adder are

$$Sum = X'Y'C_{in} + X'YC'_{in} + XY'C'_{in} + XYC_{in}$$

$$C_{out} = XC_{in} + YC_{in} + XY$$

FIGURE 9-29
Implementation of a Full Adder Using a PAL

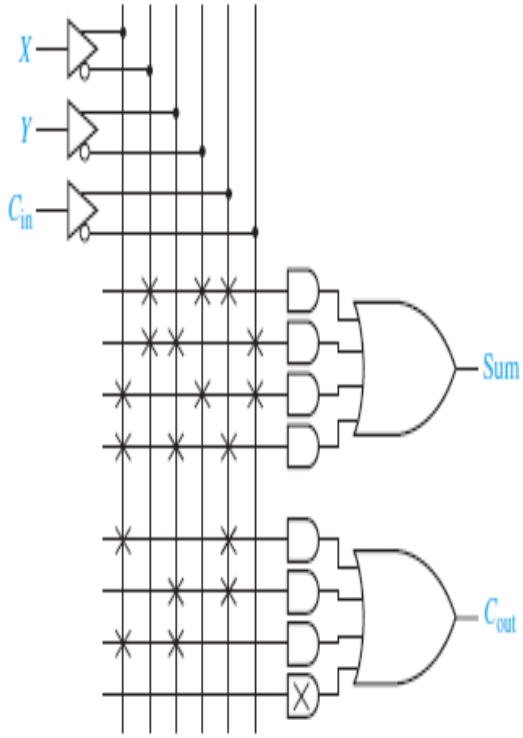


Figure above shows a section of a PAL where each OR gate is driven by four AND gates. The X's on the diagram show the connections that are programmed into the PAL to implement the full adder equations

4 (a) **Write a VHDL module for a combinational circuit using data flow model**
a) 4:1 Multiplexer
b) 8 : 3 Encoder

(b) **Explain the application of the S-R latch is for debouncing switches.**

[5+5] CO5 L3

Answer:

a) 4:1 Multiplexer

```
Library ieee;
useieee.std_logic_1164.all;

entity mux is
port(S1,S0,D0,D1,D2,D3:in bit; Y:out bit);
end mux;

architecture data of mux is
begin
   Y<=(not S0 andnot S1 and D0)or
(S0 andnot S1 and D1)or
(not S0 and S1 and D2)or
(S0 and S1 and D3);
end data;
```

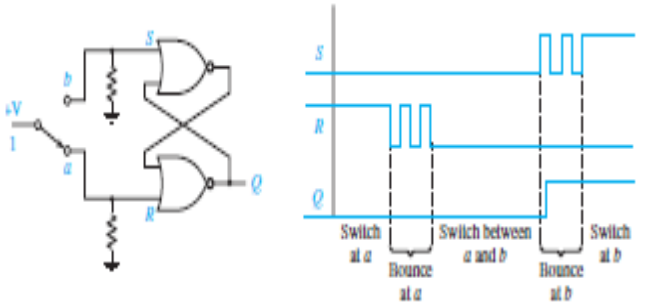
b) 8 : 3 Encoder

```
library ieee;
useieee.std_logic_1164.all;

entity encis
   port(i0,i1,i2,i3,i4,i5,i6,i7:in bit; o0,o1,o2:out
```

```
bit);
end encis;

architecture adder4 of encis
begin
   o0<=i4 or i5 or i6 or i7;
   o1<=i2 or i3 or i6 or i7;
   o2<=i1 or i3 or i5 or i7;
end adder4;
```

b)

FIGURE 11-9
witch Debouncing
with an S-R Latch



When a mechanical switch is opened or closed, the switch contacts tend to vibrate or bounce open and closed several times before settling down to their final position.

- This produces a noisy transition, and this noise can interfere with the proper operation of a logic circuit. This can avoided by contacting to SR latch.

1. The input to the switch in Figure above is connected to a logic 1 (+V).The pull-down resistors connected to contacts *a* and *b* assure that when the switch is between *a* and *b* the latch inputs *S* and *R* will always be at a logic 0, and the latch output will not change state.
2. The timing diagram shows what happens when the switch is flipped from *a* to*b*.
   a. As the switch leaves *a*, bounces occur at the *R* input; when the switch reaches *b*, bounces occur at the *S* input.
   b. After the switch reaches *b*, the first time *S* becomes 1, after a short delay the latch switches to the *Q* =1 state and remains there.
   c. Thus *Q* is free of all bounces even though the switch contacts bounce.

This debouncing scheme requires a *double throw* switch that switches between two contacts;it will not work with a *single throw* switch that switches between one contact and open.

5 **Write a structural VHDL module for a 4 bit adder.** [10] CO5 L3

Answer:

Full Adder

```
Library ieee;
useieee.std_logic_1164.all;

entity fa is port(a,b,c:in bit;sum,carry:out bit);
end fa;

architecture data of fa is
begin
sum<= a xor b xor c;
carry<=((a and b)or(b and c)or(a and c));
end data;
```

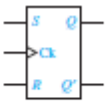
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity 4bitadder is
port(a :in STD_LOGIC_VECTOR(3downto0);
b :in STD_LOGIC_VECTOR(3downto0);
ca :out STD_LOGIC;
sum :out STD_LOGIC_VECTOR(3downto0)
);
end 4bitadder;

architecture adder4 of 4bitadder is
Component fa is
port(a :in STD_LOGIC;
b :in STD_LOGIC;
c :in STD_LOGIC;
sum :out STD_LOGIC;
ca :out STD_LOGIC
);
end component;
signal s :STD_LOGIC_VECTOR(2downto0);
signal temp:STD_LOGIC;
begin
temp<='0';
u0 :fa port map (a(0),b(0),temp,sum(0),s(0));
   u1 :fa port map (a(1),b(1),s(0),sum(1),s(1));
   u2 :fa port map (a(2),b(2),s(1),sum(2),s(2));
ue:fa port map (a(3),b(3),s(2),sum(3),ca);
end adder4;
```

6(a) **Explain the operation of master and slave S-R flip flop with truth table and timing diagram** [7] CO4 L2

Answer:

1. Figure below(a) shows an S-R flip-flop constructed from two S-R latches and gates.
2. This flip-flop changes state after the rising edge of the clock.
3. The circuit is often referred to as a master-slave flip-flop.
4. When CLK= 0, the *S* and *R* inputs set the outputs of the master latch to the appropriate value while the slave latch holds the previous value of *Q*.
5. When the clock changes from 0 to 1, the value of *P* is held in the master latch and this value is transferred to the slave latch.
6. The master latch holds the value of *P* while CLK =1, and, hence,*Q*does not change.
7. When the clock changes from 1 to 0, the *Q* value is latched in the slave, and the master can processnew inputs.

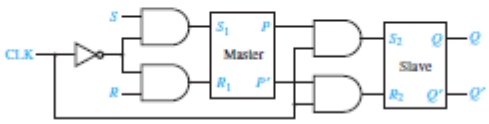
FIGURE 11-18
S-R Flip-Flop



Operation summary:

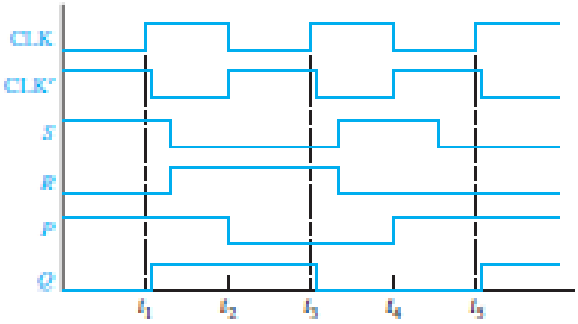
$S = R = 0$	No state change
$S = 1, R = 0$	Set Q to 1 (after active Ck edge)
$S = 0, R = 1$	Reset Q to 0 (after active Ck edge)
$S = R = 1$	Not allowed

FIGURE 11-19
S-R Flip-Flop Implementation and Timing



(a) Implementation with two latches

Figure below(b) shows the timing diagram. Initially, S= 1 and Q changesto 1 at $t1$.Then R = 1 and Q changes to 0 at $t3$.



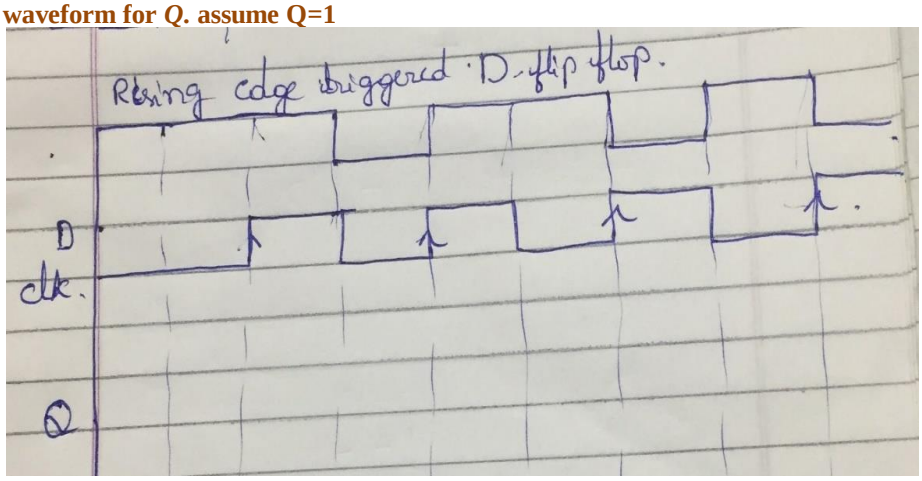
(b) Timing analysis

1. This flip-flop appears to operate just like an edge-triggered flip-flop, but there is a subtle difference.
2. For a rising-edge-triggered flip-flop the value of the inputs is sensed at the rising edge of the clock, and the inputs can change while the clock is low.
3. For the master-slave flip-flop, if the inputs changewhile the clock is low, the flip-flop output may be incorrect.

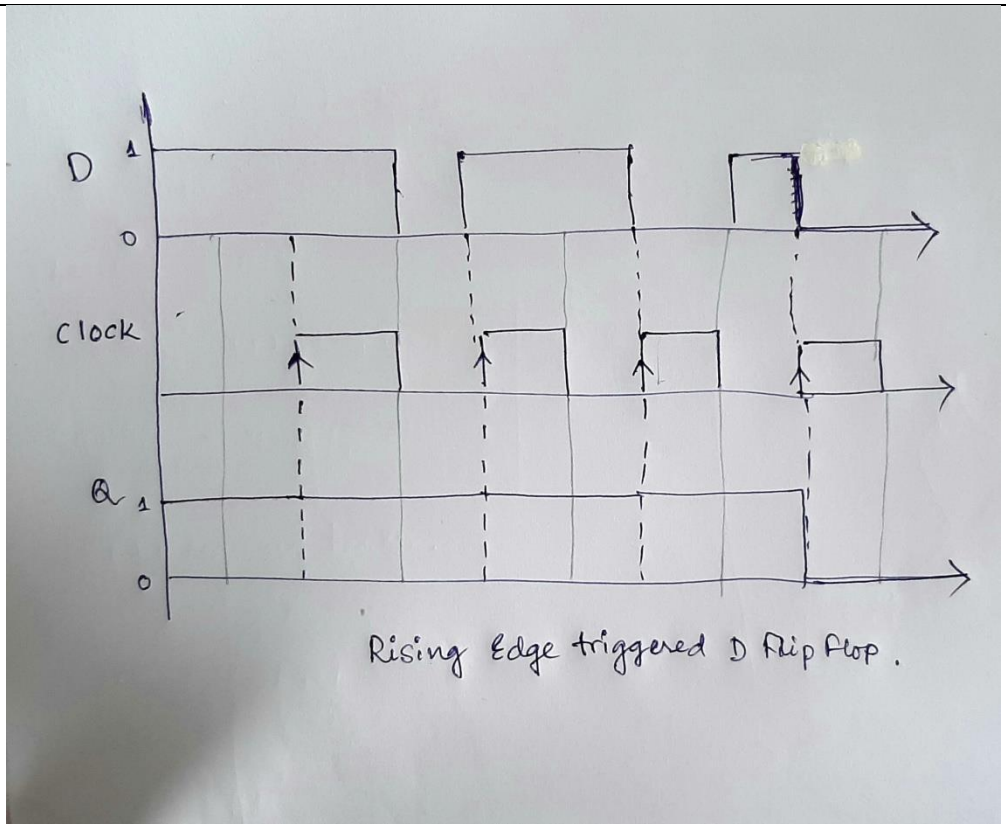
For example,

- In figure above (b) at $t4$, S =1 and R =0, so P changes to 1.
- Then S changes to 0 at $t5$, but P does not change, so at $t5$, Q changes to 1 after the rising edge of CLK.
- However, at $t5$, S =R =0, so the state of Q should not change
- Solve this problem if we only allow the S and R inputs to change while the clock is high.

(b) **Given a rising-edge-triggered D flip-flop with the following inputs, sketch the waveform for *Q*. assume Q=1** [3] CO4 L3



Answer:



7(a)	**Derive the characteristic equations for the following latches and flip-flops in product of-sums form.** **(a) S-R latch or flip-flop** **(b) Gated D latch** **(c) D flip-flop** **(d) J-K flip-flop** **(e) T flip-flop** **Answer:** **(a) S-R latch or flip-flop**	[5]	CO4	L2,L3

FIGURE 11-8
Derivation of Q^+ for an S-R Latch

RQ \ S	0	1
00	0	1
01	1	1
11	0	X
10	0	X

(a) Q^+ map

S	R	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	1	0	– } Inputs not allowed
1	1	1	–

(b) Truth table

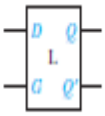
$Q^+ = S + R'Q$ $(SR = 0)$

Q+ -next state Q –present state

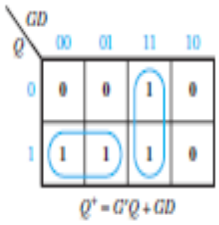
Next state Q+ output 1 when S=1 or when R=0 and Q=1.

(b) Gated D latch

FIGURE 11-12
Symbol and Truth Table for Gated Latch



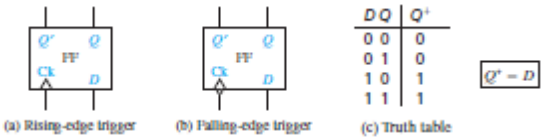
G D Q	Q⁺
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	1
1 1 1	1



Next state Q+ output 1 when G=0 and Q=1 or G=1 and D =1

(c) D flip-flop

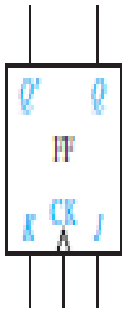
FIGURE 11-13
D Flip-Flops



Next state Q+ output 1 when D=1

(d) J-K flip-flop

FIGURE 11-20
J-K Flip-Flop
(Q Changes on the Rising Edge)



(a) J-K flip-flop

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 1 0	1
1 1 1	0

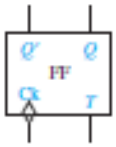
$Q^+ = JQ' + K'Q$

(b) Truth table and characteristic equation

Next state Q+ output 1 when J=1 and Q-0 or K=0 and Q=1

(e) T flip-flop

FIGURE 11-22
T Flip-Flop



(a)

T Q	Q⁺
0 0	0
0 1	1
1 0	1
1 1	0

(b)

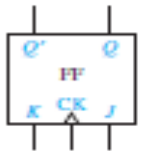
$Q^+ = T'Q + TQ' = T \oplus Q$

Next state Q+ output 1 when T=1 and Q=0 or T=0 and Q=1

(b) **Explain operation of JK flip flop with truth table and timing diagram.** [5] CO4 L2

Answer:

FIGURE 11-20
J-K Flip-Flop
(Q Changes on the Rising Edge)

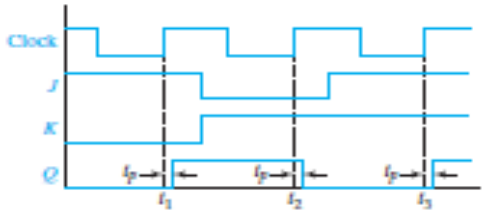


(a) J-K flip-flop

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 1 0	1
1 1 1	0

$$Q^+ = JQ' + K'Q$$

(b) Truth table and characteristic equation



(c) J-K flip-flop timing

The J-K flip-flop (Figure above is an extended version of the S-R flip-flop. The J-K flip-flop has three inputs—*J*, *K*, and the clock (CK).

- The *J* input corresponds to *S*, and *K* corresponds to *R*.
- That is, if *J*= 1 and *K* = 0, the flip-flop output is set to *Q* =1 after the active clock edge; and if *K*= 1 and *J*=0, the flip-flop output is reset to *Q* = 0after the active edge.
- Unlike the S-R flip-flop, a 1 input may be applied simultaneouslyto *J* and *K*, in which case the flip-flop changes state after the active clock edge
- .When*J*=*K*= 1, the active edge will cause *Q* to change from 0 to 1, or from 1 to 0.
- The nextstatetable and characteristic equation for the J-K flip-flop are given in Figure above(b).
- Figure above (c) shows the timing for a J-K flip-flop.
- This flip-flop changes state ashort time (*tp*) after the rising edge of the clock pulse, provided that *J* and *K* haveappropriate values.

1. If *J* =1 and *K*= 0 when Clock = 0, *Q* will be set to 1 followingthe rising edge.
2. If *K*= 1 and *J* =0 when Clock _ 0, *Q* will be set to 0 after the risingedge.
3. Similarly, if *J* =*K*= 1, *Q* will change state after the rising edge.

Referring toFigure above(c),

- because*Q* =0, *J*= l, and *K*= 0 before the first rising clock edge, *Q*changes to 1 at *t*1.
- Because *Q*= 1, *J* = 0, and *K*= 1 before the second rising clockedge,*Q*changes to 0 at *t*2.
- Because *Q*= 0, *J*= 1, and *K* = 1 before the third risingclock edge,*Q*changes to 1 at *t*3.

8	**Explain edge triggered D Flip Flop with truth table and timing digram and why edge trigger is important.** **Answer:** A D flip-flop (Figure below has two inputs, *D* (data) and Ck (clock). The small arrowhead on the flip-flop symbol identifies the clock input.	[10]	CO4	L2

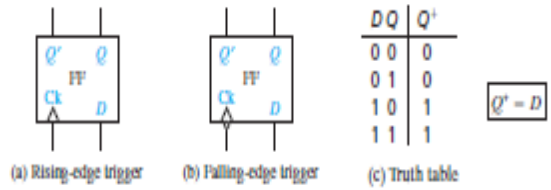
Unlike the *D* latch, the flip-flop output changes only in response to the clock, not to a change in *D*.

- If the output can change in response to a 0 to 1 transition on the clock input, we say that the
  flip-flop is triggered on the *rising edge* (or positive edge) of the clock.
- If the output can change in response to a 1 to 0 transition on the clock input, we say that the flipflop is triggered on the *falling edge* (or negative edge) of the clock.

An inversion bubble on the clock input indicates a *falling-edge trigger* (Figure below(b)), and no bubbleindicates a *rising-edge trigger* [Figure below (a)].

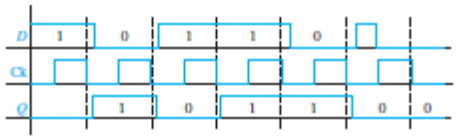
The term *active edge* refers to the clock edge (rising or falling) that triggers the flip-flop state change.

FIGURE 11-13
D Flip-Flops



‘

FIGURE 11-14
Timing for D Flip-Flop (Falling-Edge Trigger)



The state of a D flip-flop after the active clock edge (*Q*=) is equal to the input (*D*) before the active edge.

For example,

1. if*D* = 1 before the clock pulse, *Q* =1 after the active edge, regardless of the previous value of *Q*. Therefore, the characteristic equation is $Q^{+}= D$.
2. If *D* changes at most once following each clock pulse, the output of the flip-flop is the same as the *D* input, except that the output changes are delayed until after the active edge of the clock pulse, as illustrated in Figure above

<u>why edge trigger is important.</u>

To avoid race condition and unpredictable output.