



National University

of Computer and Emerging Sciences Chiniot-Faisalabad Campus



EE1005 – Digital Logic Design Assignment – 5 Part – 1 Spring 2024

Instructor: Muhammad Adeel Tahir

Sections: BS-SE 2A, BS-SE 2B, BS-CS 2F

Maximum Marks: 150+110 = 260

Due Dates:

Part 1: 30th April 2024

Part 2: 10th May 2024

- Partially or fully **copied assignments** will be marked as **zero**.
- Only **handwritten** solution on **A4 page** will be accepted.
- Submission on the GCR by the deadline is **Compulsory**.
- Late submissions are not allowed. In case of late submission, assignment will not be accepted.
- Clearly indicate all the calculations in your solution. No points will be awarded in case of missing calculations.
- You can submit your assignment **during the class** on due date. But submitting on GCR as mentioned is compulsory.
- **Proper calculations including k-map and circuit diagram labelling at each output, simplifications if any are to be implemented, missing steps will receive zero marks in that question straight away.**
- **Only eligible handwriting will be checked, the question shall be liable to receiving a 0 if the not readable at all.**
- **A viva of this assignment will take place and hence not being able to explain your questions will lead to 0 in that specific question.**

Question 1:**(2+5+5+3= 15 marks)**

Design a combinational circuit that can detect whether the 3-bit input binary number is even, odd, or prime number.

- (a) Find the number of inputs, and outputs.
- (b) Draw the truth table for the design problem
- (c) Implement the circuit using 3×8 Decoder.
- (d) Explain the design implementation in your own words. Lack of understanding of the problem, and incorrect explanation will lead to 0 in the question.

Solution:

(2) Using $n=3$ bits, we can represent 0 to $2^n - 1$

0 to $2^3 - 1$

0 to 7

(a) Truth table:-

Decimal	D_2	D_1	D_0	even	odd	Prime
0	0	0	0	1	0	0
1	0	0	1	0	1	1
2	0	1	0	1	0	1
3	0	1	1	0	1	1
4	1	0	0	1	0	0
5	1	0	1	0	1	1
6	1	1	0	1	0	0
7	1	1	1	0	1	1

Even numbers - 0, 2, 4, 6

Odd numbers - 1, 3, 5, 7

Prime numbers - 1, 2, 3, 5, 7

used to fill
the truth
table

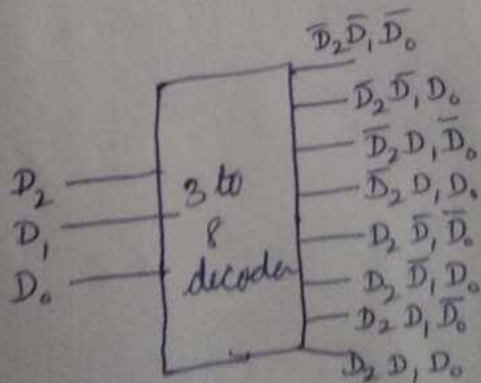
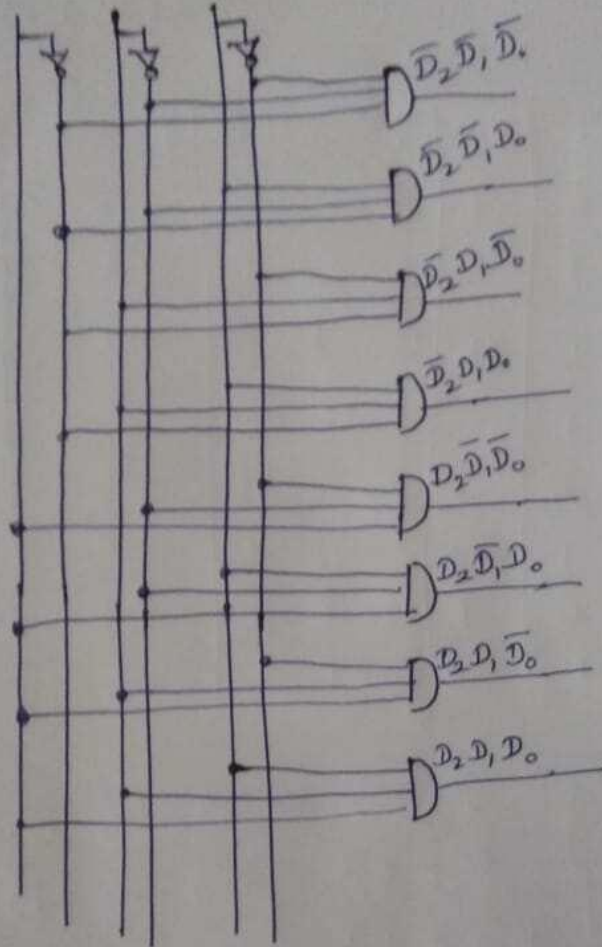
$$\text{Even numbers} = \bar{D}_2 \bar{D}_1 \bar{D}_0 + \bar{D}_2 D_1 \bar{D}_0 + D_2 \bar{D}_1 \bar{D}_0 + D_2 D_1 \bar{D}_0$$

$$\text{Odd numbers} = \bar{D}_2 \bar{D}_1 D_0 + \bar{D}_2 D_1 D_0 + D_2 \bar{D}_1 D_0 + D_2 D_1 D_0$$

$$\text{Prime numbers} = \bar{D}_2 \bar{D}_1 D_0 + \bar{D}_2 D_1 \bar{D}_0 + \bar{D}_2 D_1 D_0 + D_2 \bar{D}_1 D_0 + D_2 D_1 D_0$$

3 to 8 decoder:-

D_2 D_1 D_0



3 to 8 decoder
using blocks

Question 2:**(5+8+2=15 marks)**

Design a combinational circuit that takes 3-bit input and at the output it multiplies it by 3 and adds 1 to have the final output. Design this circuit using only **2 × 4 decoders** and basic logic gates if necessary.

- Properly label and fill the truth table in neat and clean manner for this design.
- Design the circuit diagram for this problem.
- Explain the approach in your own words (5-8 lines max). *Wrong explanation leads to 0.*

Solution:

The input is a 3-bit number (0 – 7).

The maximum output can be $(7 \times 3) + 1 = 22$, so we need 5 bits at the output.

Input Number	Input Bits			Output Number	Output Bits				
	x	y	z		A_4	A_3	A_2	A_1	A_0
0	0	0	0	$(0 \times 3) + 1 = 1$	0	0	0	0	1
1	0	0	1	$(1 \times 3) + 1 = 4$	0	0	1	0	0
2	0	1	0	$(2 \times 3) + 1 = 7$	0	0	1	1	1
3	0	1	1	$(3 \times 3) + 1 = 10$	0	1	0	1	0
4	1	0	0	$(4 \times 3) + 1 = 13$	0	1	1	0	1
5	1	0	1	$(5 \times 3) + 1 = 16$	1	0	0	0	0
6	1	1	0	$(6 \times 3) + 1 = 19$	1	0	0	1	1
7	1	1	1	$(7 \times 3) + 1 = 22$	1	0	1	1	0

$$A_4 = \Sigma(5, 6, 7)$$

$$A_1 = \Sigma(2, 3, 6, 7)$$

$$A_3 = \Sigma(3, 4)$$

$$A_0 = \Sigma(0, 2, 4, 6)$$

$$A_2 = \Sigma(1, 2, 4, 7)$$

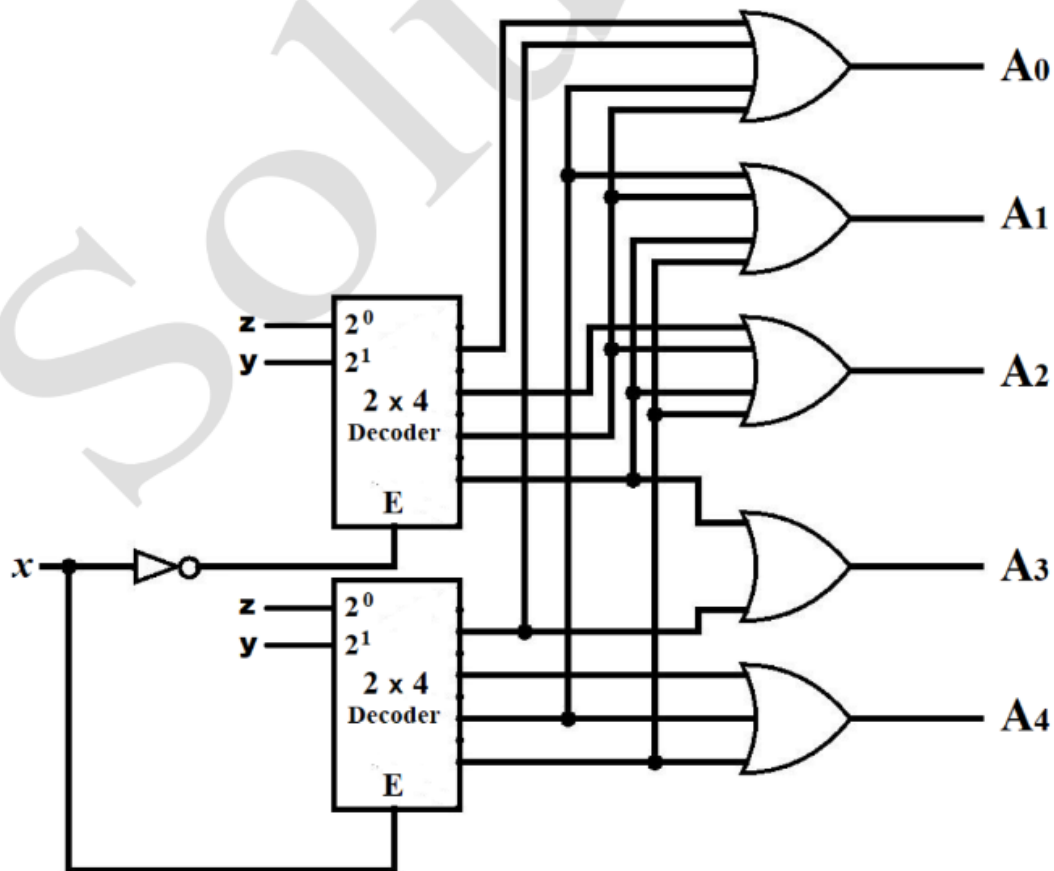
$$A_4 = \Sigma(5, 6, 7)$$

$$A_1 = \Sigma(2, 3, 6, 7)$$

$$A_3 = \Sigma(3, 4)$$

$$A_0 = \Sigma(0, 2, 4, 6)$$

$$A_2 = \Sigma(1, 2, 4, 7)$$



Question 3:

(10+10+5= 25)

Design a Circuit for the Space Colony Defense Game Using a 8×1 Multiplexer.

Objective:

Your task is to design an electronic circuit for a game called "Space Colony Defense." In this game, there are four defense systems positioned around a space colony. Each defense system can either activate (HIGH) or deactivate (LOW) based on the threat level detected by its sensors. The colony is considered safe if at least three out of the four defense systems activate.

Requirements:

1. Defense Systems and Activation:

- There are four defense systems positioned around the space colony.
- Each defense system can either activate (HIGH) or deactivate (LOW) based on the threat level detected.

2. Safety Indicator:

- The game must include a "safe zone indicator" that turns ON if the colony is considered safe. For the purpose of this game, define "safe" as at least three out of the four defense systems being activated.
- If the colony is not considered safe, the indicator should remain OFF, indicating that the colony is at risk.

3. Circuit Design:

- Use an 8 X 1 Multiplexer (MUX) to determine whether the colony is safe based on the status of the defense systems.
- You may use basic logic gates if necessary to assist in the design.
- Determine how the outputs from the defense systems will control the selection lines of the MUX to achieve the desired outcome.

4. Output Explanation:

- Clearly explain how the MUX and any additional logic gates you use contribute to the final decision of turning the safe zone indicator ON or OFF.

Implement the following:

- a) A truth table that outlines how different combinations of defense system statuses affect the safe zone indicator.
- b) A schematic diagram of the circuit, clearly labeling each input and output carefully.
- c) A detailed explanation of how the circuit processes the defense system statuses to control the safe zone indicator.

Note: Ensure your design is clear and well-documented, as you will need to explain how it works later. In case the writing is not readable a straight 0 shall be awarded.

Solution:

Here we have 4 inputs and 1 output.

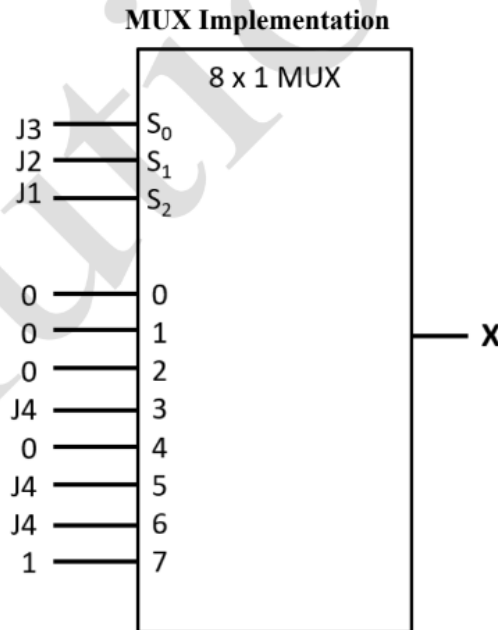
As we are using 8×1 Multiplexer, so there will be 3 selection lines and 8 data lines.

J1, J2, and J3 will act as selection lines.

We will divide the truth table in 8 equal parts and for each part we will represent S in terms of J4

Truth Table

J1	J2	J3	J4	S
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	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



Question 4:

(5+3+2 = 10 marks)

Design a digital circuit that takes in an input consisting of three binary digits. The circuit should process the input and produce an output made up of two binary digits, known as B1 and B0. The purpose of this output is to represent the number of '1's that are present in the input.

For example, if the input is '101', the output should be '10'.

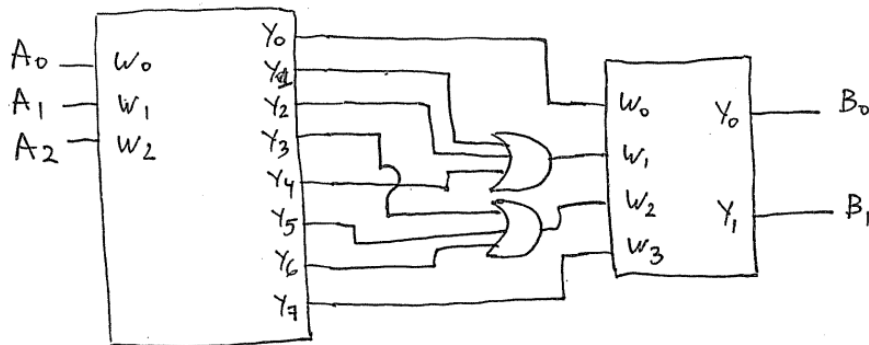
- Implement the truth table of this circuit
- Implement the design using a one 3 × 8 Decoder, OR gates, and one 4-2 encoder.
- Explain the role of each component of design (i.e the use of decoder, encoder, and the OR gate)
- Draw the circuit diagram/design implementation in a neat and clean handwriting, clearly mention the inputs, outputs and connections and label them carefully, incase the diagram is not understandable, the question will receive no marks.
- Explain how the design is working in a clear and concise manner (5-8 points step by step)

Solution:

Truth Table:

A2	A1	A0	Number of 1's	B1	B0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0	2	1	0
1	1	1	3	1	1

Design Implementation:



Design Explanation:

- Using a 3-8 Decoder:** The 3-8 decoder will take the 3-bit input A and generate 8 outputs, each corresponding to one of the combinations of A.
- Connecting Outputs to OR Gates:**
 - Connect the outputs of the decoder that correspond to having one '1' in the input (positions 001, 010, 100) to an OR gate to generate B_0 .
 - Connect the outputs of the decoder that correspond to having two '1's in the input (positions 011, 101, 110) to another OR gate to generate part of B_1 .
 - Connect the output of the decoder that corresponds to having three '1's in the input (position 111) directly to B_1 as well.
- Using a 4-2 Encoder:**
 - The outputs of the two OR gates and the direct output from the decoder (for three '1's) are then fed into a 4-2 encoder.
- Labeling:**
 - Label all inputs (A_2 , A_1 , A_0) and outputs (B_1 , B_0).
 - Label intermediate connections and components (decoder outputs, OR gate inputs/outputs, encoder inputs/outputs).

Question 5:**(10 marks)**

Using a decoder and external gates, design the combinational circuit defined by the following three Boolean functions:

$$F1 = x'y'z' + xz$$

$$F2 = xy'z' + x'y$$

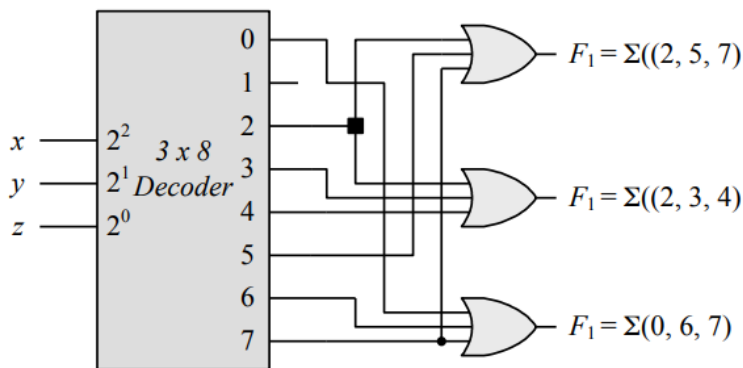
$$F3 = x'y'z' + xy$$

Solution:

$$F_1 = x(y + y')z + x'y'z' = xyx + xy'z + x'y'z' = \Sigma(2, 5, 7)$$

$$F_2 = xy'z' + x'y = xy'z' + x'y'z + x'yz' = \Sigma(2, 3, 4)$$

$$F_3 = x'y'z' + xy(z + z') = x'y'z' + xyz + xyz' = \Sigma(0, 6, 7)$$

**Question 6:****(5+5+5 = 15 marks)**

Implement a Full Adder Using a 3-to-8 Decoder and Minimal Logic Gates

Part A: Truth Table

- **Task:** Draw the truth table for a full adder. Include all possible combinations of inputs and the corresponding outputs for sum and carry.
- **Expected Output:** A complete truth table with columns for inputs C_i, X_i, Y_i and outputs S_i (sum) and C_{i+1} (carry output).

Part B: Schematic Diagram

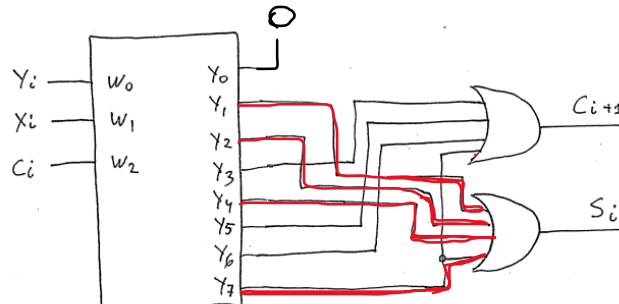
- **Task:** Create a schematic diagram showing how to implement the full adder using a single 3-to-8 decoder and the minimal number of gates.
- **Expected Output:** A clear and labeled diagram that includes:
 - The decoder with labeled inputs and outputs.
 - The connections from the decoder outputs to the gates.
 - The gates that produce the sum and carry outputs.

Part C: Implementation Details (5+5)

- **Task:** Explain how the outputs of the decoder are used to derive the sum and carry outputs using the logic gates.
- **Expected Output:** A detailed explanation of:
 - Which decoder outputs are connected to which gate.
 - How these connections implement the sum and carry functions of the full adder.

Solution:

C_i	X_i	Y_i	C_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Implementation Details:

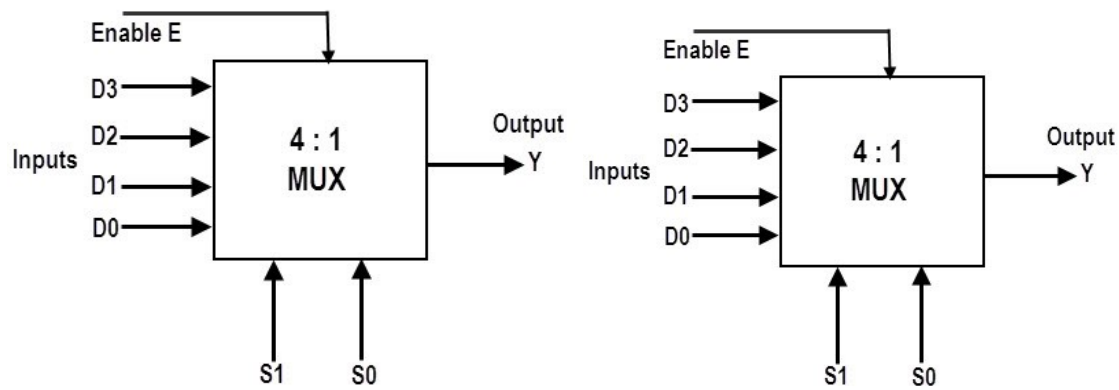
Using a 3-to-8 Decoder:

- **Decoder Outputs:** The decoder has three inputs and eight outputs. Each output corresponds to one of the eight possible combinations of the inputs.
- **Connection to OR Gates:**
 - **Sum (S) Calculation:** Connect outputs Y1, Y2, Y4, Y7 of the decoder to an OR gate. These outputs represent the cases where the sum is 1.
 - **Carry Out (C_{i+1}) Calculation:** Connect outputs Y3, Y5, Y6, Y7 to another OR gate. These outputs represent the cases where the carry out is 1.

Question 7

(5+5 = 10)

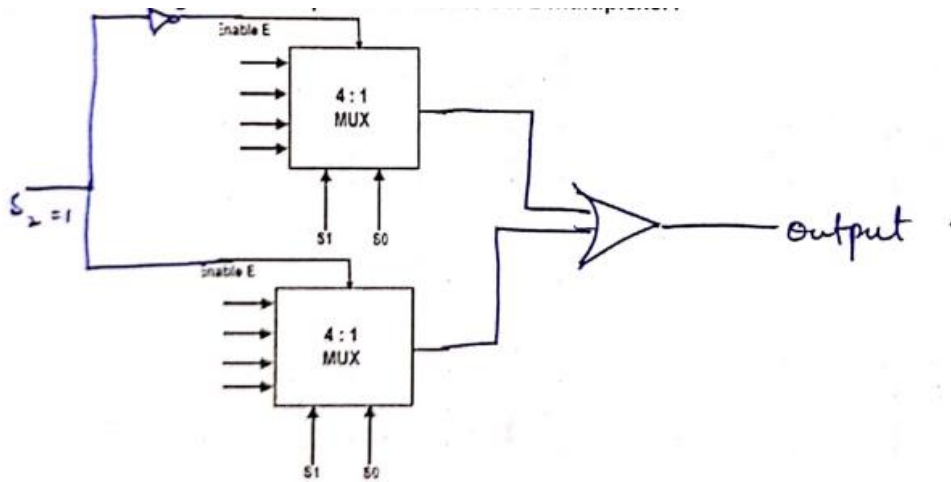
- Use the following 4×1 MUX to create a 8×1 MUX. Label the diagram neatly on the paper.

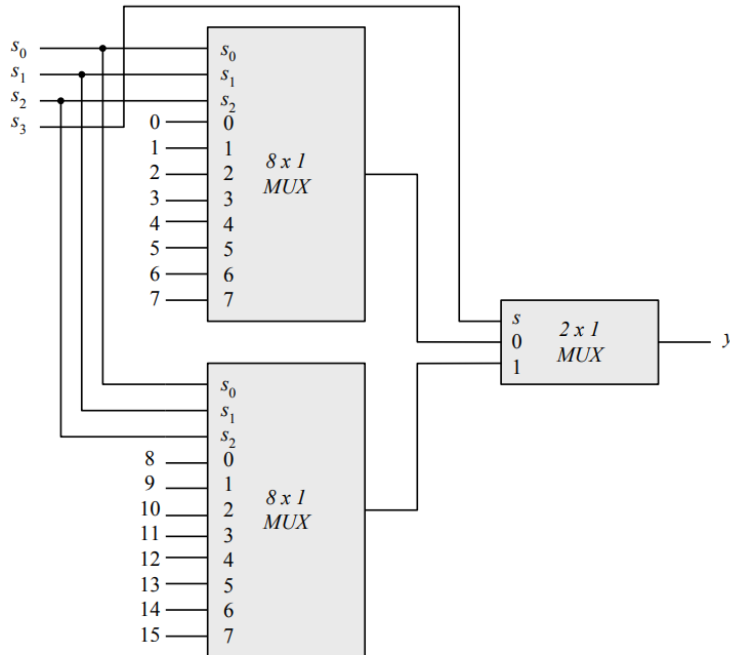


- Construct a 16 X 1 multiplexer with two 8 X 1 and one 2 X 1 multiplexers. Use block diagrams.

Solution:

1)





2)

Question 8

(5+5+5 = 15 marks)

Design a circuit that tests if a 4-bit binary number is a prime number in the decimal system. Assume that the input is represented by the signals a, b, c, d (a is most significant bit and d is the least significant bit) and the output is represented by F . Hint: 0 and 1 are not Prime.

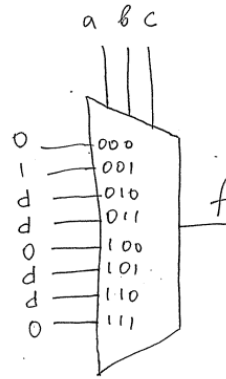
You will implement the following:

1. Draw the truth table for the function $F(a, b, c, d)$
2. Implement the function using one 8-to-1 multiplexer and a minimal number of extra AND, OR, and NOT gates:
 - You are required to clearly label all inputs, pins, and outputs in your implementation.
3. Draw the truth table for the function again:
 - After redrawing the truth table, implement the function using one 4-to-1 multiplexer and a minimal number of extra AND, OR, and NOT gates.
 - Again, all inputs, pins, and outputs must be clearly labeled.

Solution:

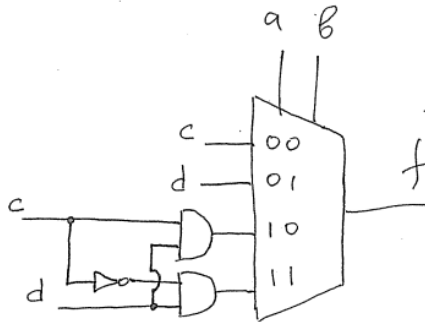
a)

#	a	b	c	d	f	
0	0	0	0	0	0	} 0
1	0	0	0	1	0	
2	0	0	1	0	1	} 1
3	0	0	1	1	1	
4	0	1	0	0	0	} d
5	0	1	0	1	1	
6	0	1	1	0	0	} d
7	0	1	1	1	1	
8	1	0	0	0	0	} 0
9	1	0	0	1	0	
10	1	0	1	0	0	} d
11	1	0	1	1	1	
12	1	1	0	0	0	} d
13	1	1	0	1	1	
14	1	1	1	0	0	} 0
15	1	1	1	1	0	



a)

#	a	b	c	d	f	
0	0	0	0	0	0	} c
1	0	0	0	1	0	
2	0	0	1	0	1	
3	0	0	1	1	1	
4	0	1	0	0	0	} d
5	0	1	0	1	1	
6	0	1	1	0	0	
7	0	1	1	1	1	
8	1	0	0	0	0	} c.d
9	1	0	0	1	0	
10	1	0	1	0	0	
11	1	0	1	1	1	
12	1	1	0	0	0	} $\bar{c} \cdot d$
13	1	1	0	1	1	
14	1	1	1	0	0	
15	1	1	1	1	0	



Question 9**(5+5+5 = 15 marks)**

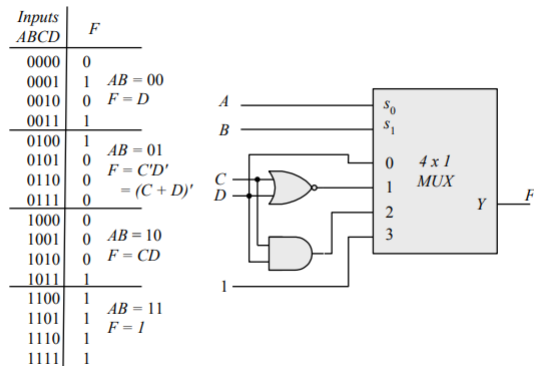
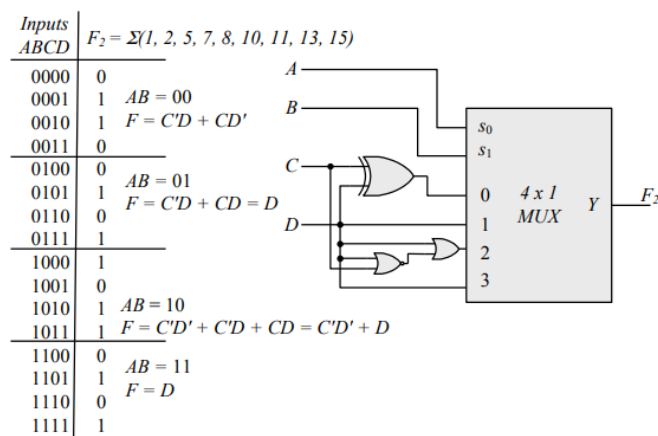
Implement the following using 4×1 MUX and external gates, connect A and B to the selection lines. The input requirements for the four data lines will be a function of variables C and D. These values are obtained by expressing F as a function of C and D for each of the four cases when AB= 00, 01,10,11. The functions may have to be implemented with external gates:

a) $F(A,B,C,D) = \sum (1,3,4,11,12,13,14,15)$

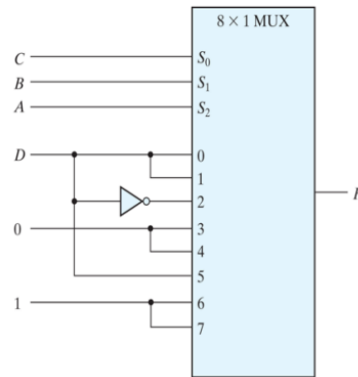
b) $F(A,B,C,D) = \sum (1,2,4,7,8,9,10,11)$

Implement the following using 8×1 MUX and external gates if required, a truth table with neat and clean diagram is necessary.

a) $F(A, B, C, D) = \sum (1, 3, 4, 11, 12, 13, 14, 15)$

Solution:**(a)****(b)** $F = \sum(1, 2, 5, 7, 8, 10, 11, 13, 15)$ **Part 2:**

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
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	0
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



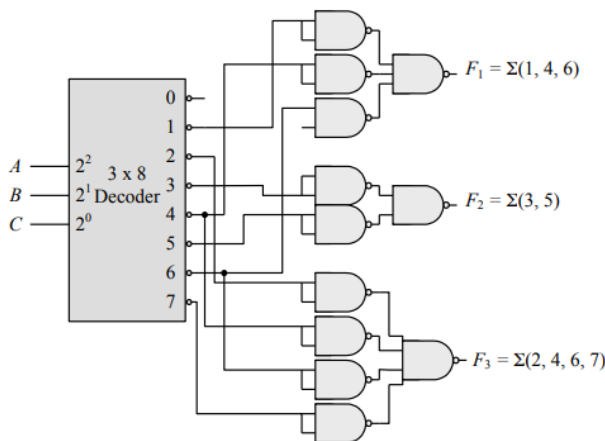
Question 10

(5 marks)

A combinational circuit is specified by the following three Boolean functions: Implement the circuit with a decoder constructed with NAND gates and NAND or AND gates connected to the decoder outputs. Use a block diagram for the decoder. Minimize the number of inputs in the external gates.

- $F_1(A, B, C) = \Sigma(1, 4, 6)$
- $F_2(A, B, C) = \Sigma(3, 5)$
- $F_3(A, B, C) = \Sigma(2, 4, 6, 7)$

Solution:

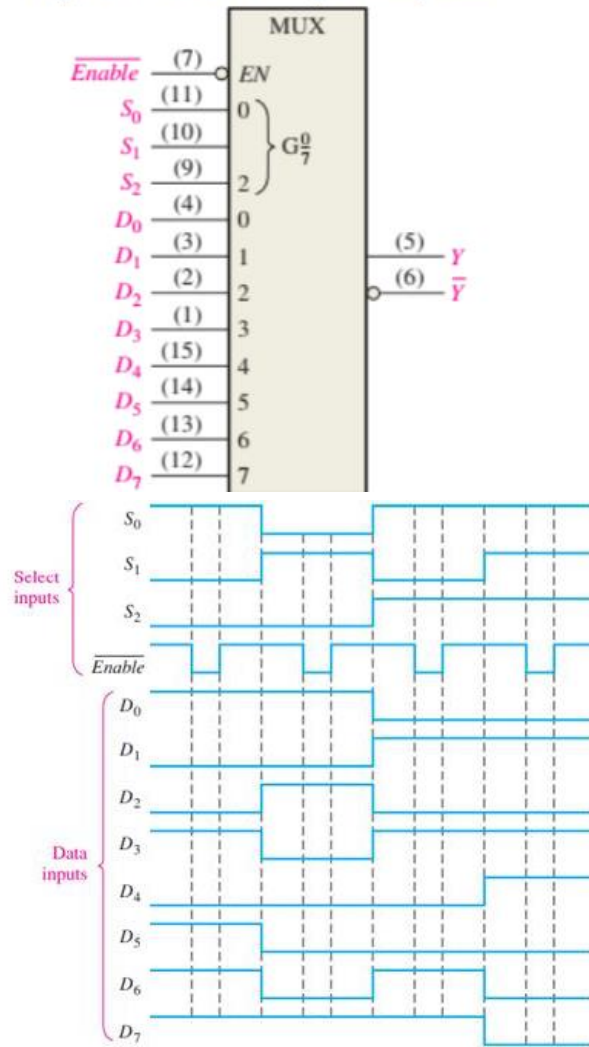


Question 11

(5 marks)

- The waveforms in Figure below are observed on the inputs of a 8-input multiplexer. Sketch the Y output waveform.

Figure 1: 74HC151 8-Input Multiplexer



- For the 4-bit comparator in figure below, plot the output waveforms for $A > B$, $A < B$, and $A = B$ for each time interval. The outputs are active-HIGH

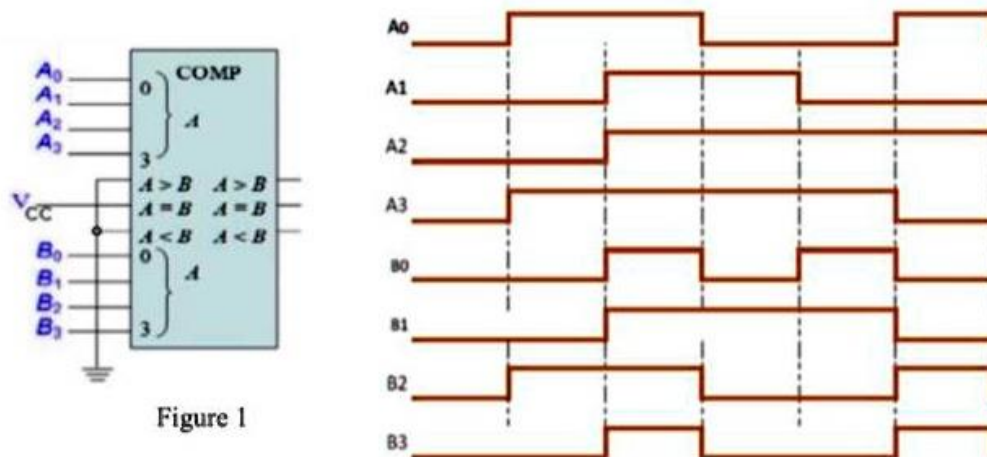
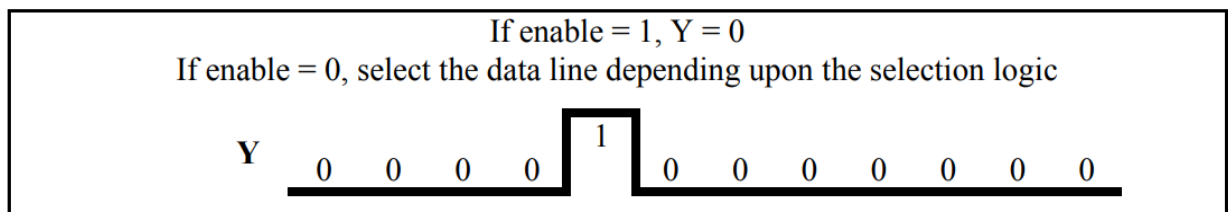
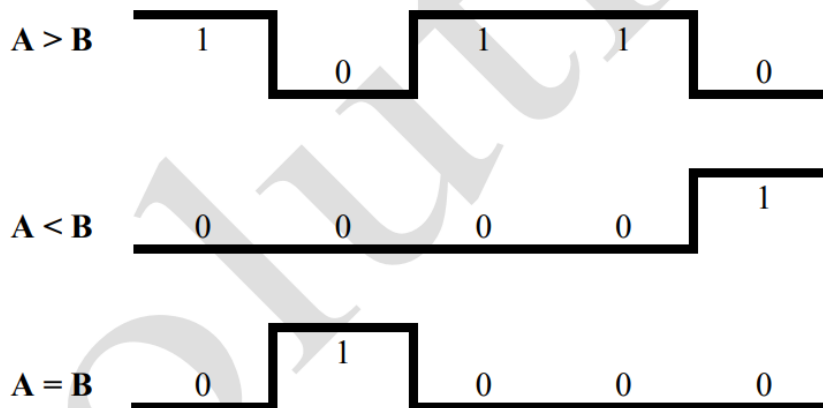


Figure 1

Solution:



a)

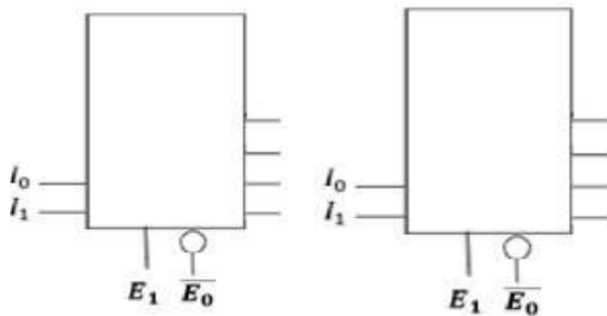


b)

Question 12

(2.5+2.5=5 marks)

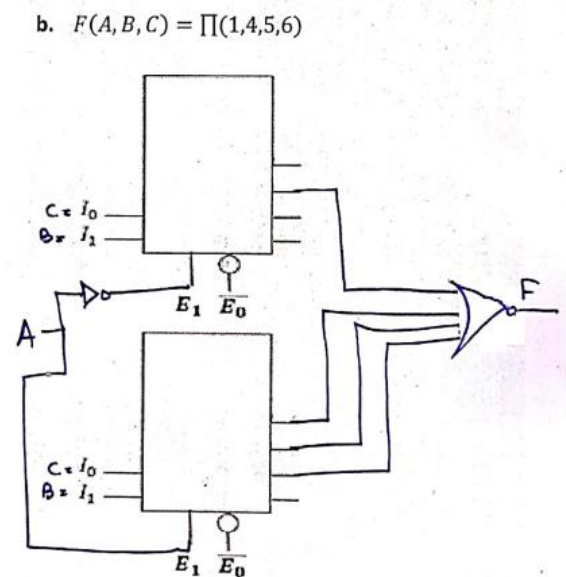
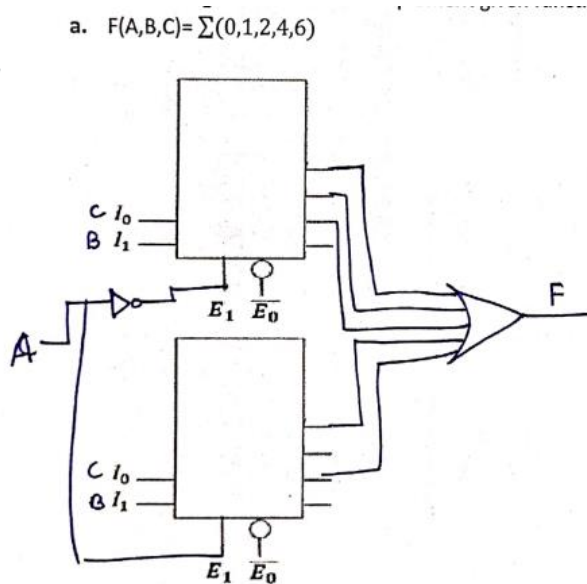
Use the following 2x4 decoders to implement given functions separately. Note: You are supposed to use the following block diagrams for each part and implement it by connecting them properly.



a. $F(A,B,C) = \sum(0,1,2,4,6)$

b. $F(A, B, C) = \prod(1,4,5,6)$

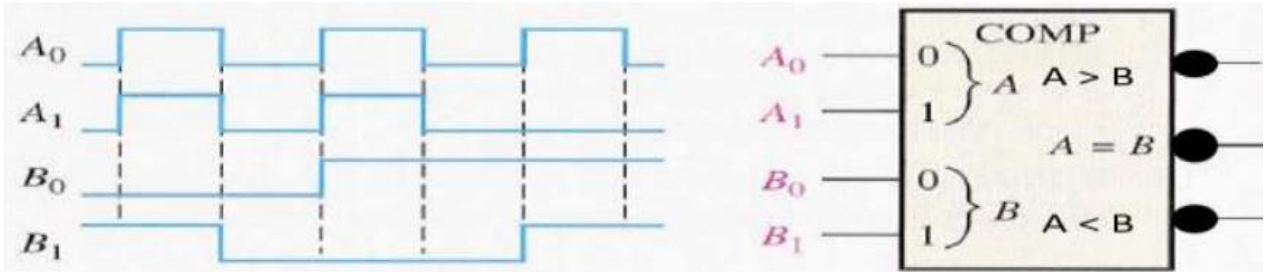
Solution:



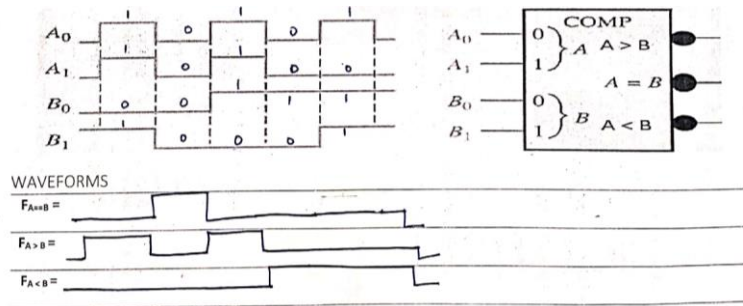
Question 13

(5 marks)

Apply the Waveform give below to the comparator circuit. Determine the output waveforms of $A=B$, $A>B$, $A<B$ which are active low enable.



Solution:



EE1005 – Digital Logic Design
Assignment – 5
Part – 2
Spring 2024

Sequential Circuits

Deadline: 10th May 2024

Total Marks: 110

Note: Incase if the explanation is wrong, copied/plagiarized and not correct, that question will receive 0 marks.

Max Limit: 5-8 lines

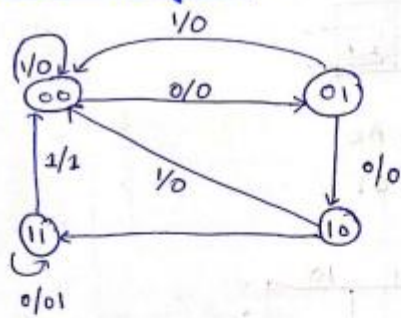
Question 1

[3 + 2 + 3 + 2 = 10 Marks]

Design a sequential circuit by using D Flip Flop to detect the sequence of three or more 0's in an input sequence of bits. The output of the circuit should be 1 when three or more 0's appears at the input, otherwise the output should be 0. You can complete your design by constructing:

1. State Diagram
2. State Table
3. Flip Flop Input Equations
4. Circuit Diagram
5. Explain the design in your own words

• State Diagram:-



• State Table:-

Present state		Input	Next state		Output
A	B	X	A(t+1)	B(t+1)	y
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
1	0	0	1	1	0
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

• Flip-Flop Input equations:-

	Bx			
	00	01	11	10
A				
0				1
1	1			1

$$A(t+1) = Bx' + Ax'$$

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

	Bx			
	00	01	11	10
A				
0	1			
1	1			1

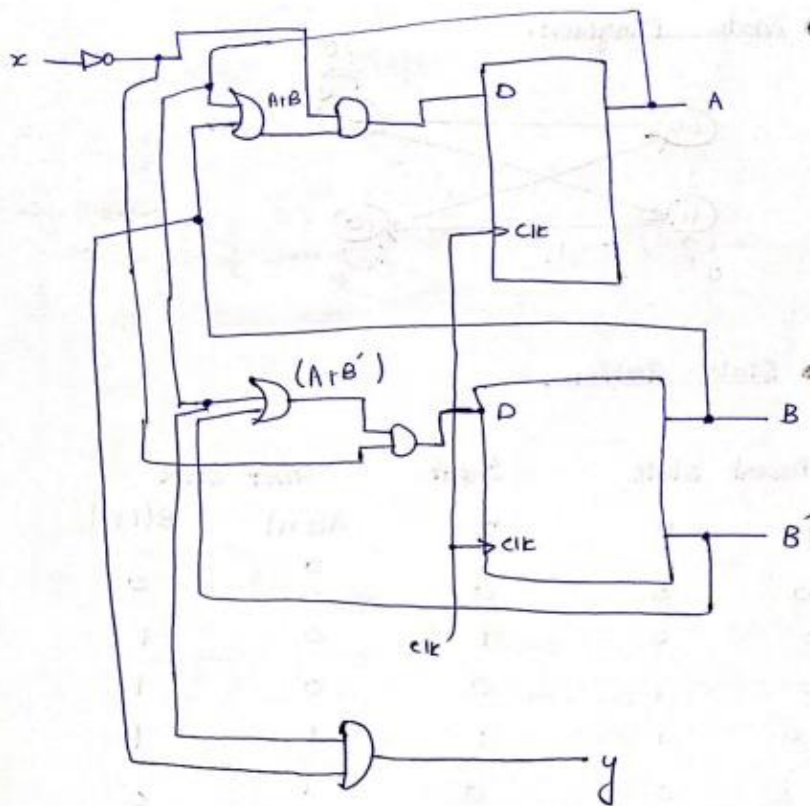
$$B(t+1) = Bx' + Ax'$$

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

	Bx			
	00	01	11	10
A				
0				
1			1	1

$$y = AB$$

• Circuit Diagram:-



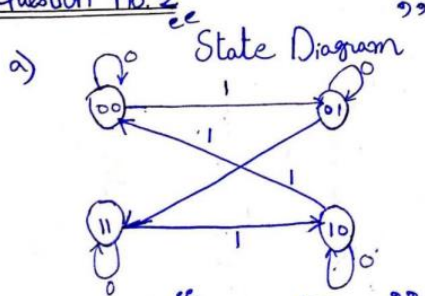
Question 2

[3 + 2 + 3 + 2 = 10 Marks]

Design a sequential circuit with two D flip-flops A and B, and one input x. When $x = 0$, the state of the circuit remains the same. When $x = 1$, the circuit goes through the state transitions from 00 to 01, to 11, to 10, back to 00, and repeats. Complete your design by constructing:

1. State Diagram
2. State Table
3. Flip Flop Input Equations
4. Circuit Diagram
5. Explain your design in your own words.

Question no. 2



b) "State Table"

Present State		Input	Next State	
A	B		A	B
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

"Equations"

c)

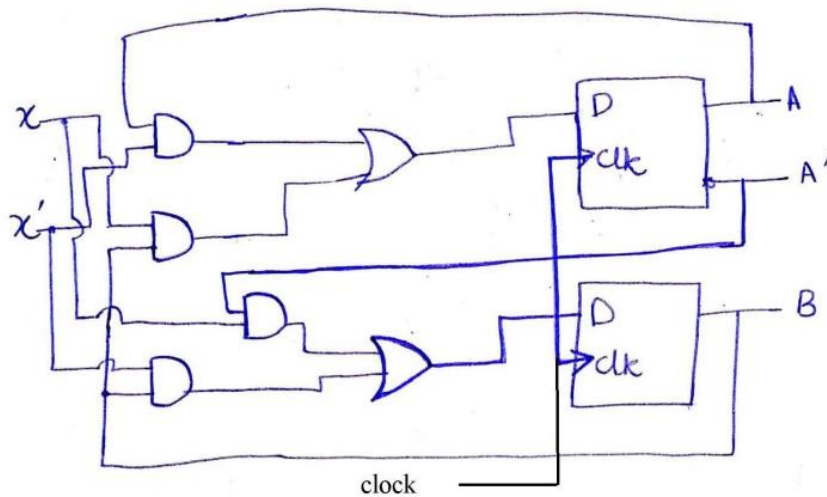
A \ Bx	00	01	11	10
0	0	0	1	0
1	1	0	1	1

$$D_A = A'x + Bx$$

A \ Bx	00	01	11	10
0	0	1	1	1
1	0	0	0	1

$$D_B = A'x + Bx'$$

"Circuit Diagram"



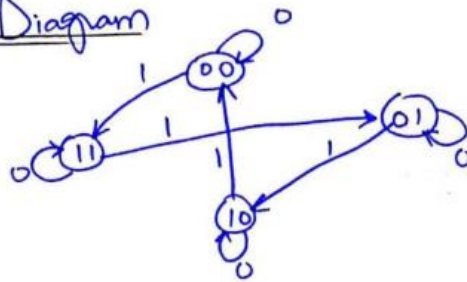
Question 3

[3+2 +2+3+2 =12 Marks]

Design a sequential circuit with two JK flip-flops A and B, and one input x . When $x = 0$, the state of the circuit remains the same. When $x = 1$, the circuit goes through the state transitions from 00 to 11, to 01, to 10, back to 00, and repeats. Complete your design by finding:

1. State Diagram
2. State Table
3. Flip Flop Inputs
4. Flip Flop Input Equation
5. Circuit Diagram
6. Explain the design in your own words

State Diagram



State Table and FF inputs.

Present State		Input x	Next State		FF inputs			
A	B		A	B	JA	KA	JB	KB
0	0	0	0	0	0	x	0	x
0	0	1	1	1	1	x	1	x
0	1	0	0	1	0	x	x	0
0	1	1	1	0	1	x	x	1
1	0	0	1	0	x	0	0	x
1	0	1	0	0	x	1	0	x
1	1	0	1	1	x	0	x	0
1	1	1	0	1	x	1	x	0

Flip flop input equations.

JA

A \ Bx	00	01	11	10
0	0	1	1	0
1	x	x	x	x

$JA = x$

JB

A \ Bx	00	01	11	10
0	0	1	x	x
1	0	0	x	x

$JB = A'x$

KA

A \ Bx	00	01	11	10
0	x	x	x	x
1	0	1	1	0

$KA = x$

KB

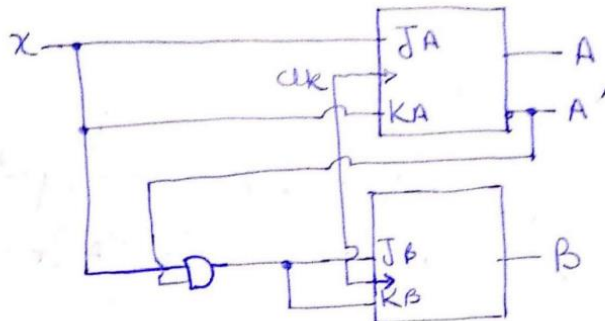
A \ Bx	00	01	11	10
0	x	x	1	0
1	x	x	0	0

$KB = A'x$

$$K_A = x$$

$$K_B = A'x$$

Logic Diagram

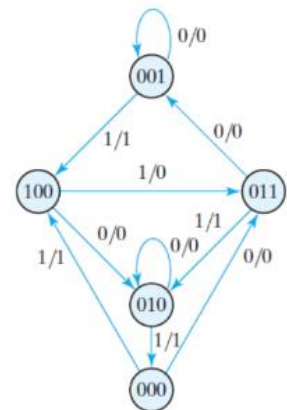


Question 4

[10 + 12 + 12 = 34 Marks]

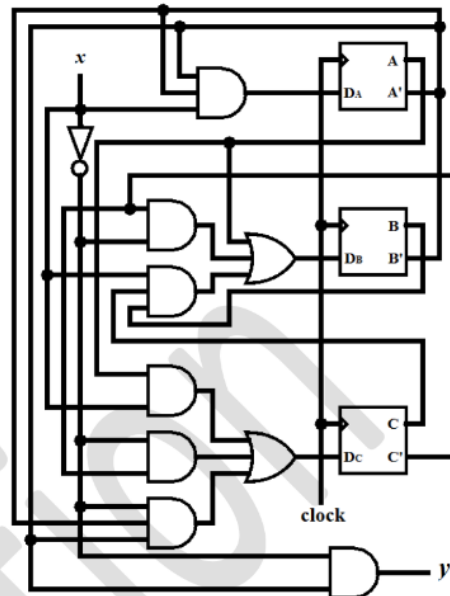
A sequential circuit has three flip-flops A, B, C ; one input x; and one output y. The state diagram is shown in figure below. The circuit is to be designed by treating the unused states as don't-care conditions. **Explain each design in your own words.**

- Use D Flip Flops for design
- Use JK Flip Flops for design
- Use T Flip Flops for design



i. With D Flip Flop

Present State			Input	Next State			Output
A	B	C		A	B	C	
0	0	0	0	0	1	1	0
0	0	0	1	1	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	1	0	0	1
0	1	0	0	0	1	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	1	0	1
1	0	0	0	0	1	0	0
1	0	0	1	0	1	1	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



AB \ Cx	00	01	11	10
00		1	1	
01				
11	X	X	X	X
10			X	X

$$D_A = A'B'x$$

AB \ Cx	00	01	11	10
00	1			
01	1		1	
11	X	X	X	X
10	1	1	X	X

$$D_B = A + C'x' + BCx$$

AB \ Cx	00	01	11	10
00	1			1
01				1
11	X	X	X	X
10		1	X	X

$$D_C = Ax + Cx' + A'B'x'$$

AB \ Cx	00	01	11	10
00	1	1		
01	1	1		
11	X	X	X	X
10			X	X

$$y = A'B'x$$

ii. With JK Flip Flop

Present State			Input	Next State			Output	Flip Flop Inputs					
A	B	C	x	A	B	C	y	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	1	0	0	X	1	X	1	X
0	0	0	1	1	0	0	1	1	X	0	X	0	X
0	0	1	0	0	0	1	0	0	X	0	X	X	0
0	0	1	1	1	0	0	1	1	X	0	X	X	1
0	1	0	0	0	1	0	0	0	X	X	0	0	X
0	1	0	1	0	0	0	1	0	X	X	1	0	X
0	1	1	0	0	0	1	0	0	X	X	1	X	0
0	1	1	1	0	1	0	1	0	X	X	0	X	1
1	0	0	0	0	1	0	0	X	1	1	X	0	X
1	0	0	1	0	1	1	0	X	1	1	X	1	X
1	0	1	0	X	X	X	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X

By using k-maps the equations for J_A , K_A , J_B , K_B , J_C , K_C and y can be found.

$$J_A = B'x$$

$$K_A = 1$$

$$J_B = A + C'x'$$

$$K_B = C'x + Cx'$$

$$J_C = Ax + A'B'x'$$

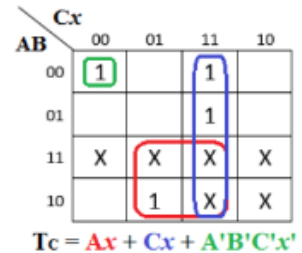
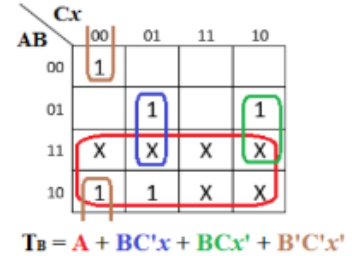
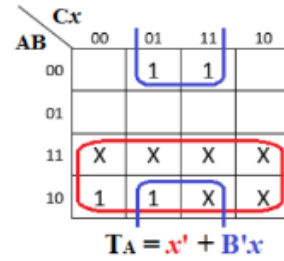
$$K_C = x$$

$$y = A'x$$

The above equations can be used to draw the circuit diagram

iii. With T Flip Flops

Present State			Input	Next State			Output	Flip Flop Inputs		
A	B	C	x	A	B	C	y	T _A	T _B	T _C
0	0	0	0	0	1	1	0	0	1	1
0	0	0	1	1	0	0	1	1	0	0
0	0	1	0	0	0	1	0	0	0	0
0	0	1	1	1	0	0	1	1	0	1
0	1	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	1	0	1	0
0	1	1	0	0	0	1	0	0	1	0
0	1	1	1	0	1	0	1	0	0	1
1	0	0	0	0	1	0	0	1	1	0
1	0	0	1	0	1	1	0	1	1	1
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X



The above equations can be used to draw circuit diagram.

Question 5

[3 + 8 + 3 = 14 Marks]

For the following state table:

Present State	Next State		Output	
	x = 0	x = 1	x = 0	x = 1
a	f	b	0	0
b	d	c	0	0
c	f	e	0	0
d	g	a	1	0
e	d	c	0	0
f	f	b	1	1
g	g	h	0	1
h	g	a	1	0

- Draw the corresponding state diagram.
- Tabulate the reduced state table.
- Draw the state diagram corresponding to the reduced state table

Solution:

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	f	b	0	0
b	d	c	0	0
c	f	e	0	0
d	g	a	1	0
e	d	c	0	0
f	f	b	1	1
g	g	h	0	1
h	g	a	1	0

States b and e are equivalent, and d and h are also equivalent. So, removing e and h, and then replacing e and h with b and d respectively in remaining table we get.

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	f	b	0	0
b	d	c	0	0
c	f	b	0	0
d	g	a	1	0
f	f	b	1	1
g	g	d	0	1

States a and c are equivalent, removing c and replacing c with a in remaining table, we get.

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	f	b	0	0
b	d	a	0	0
d	g	a	1	0
f	f	b	1	1
g	g	d	0	1

There are no more equivalent states, so this is the reduced state table.

State diagrams can be drawn from the given and reduced state tables easily.

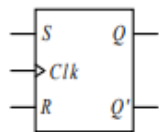
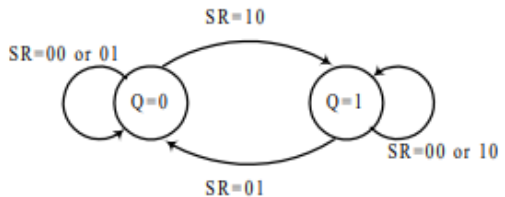
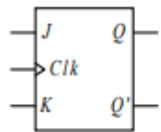
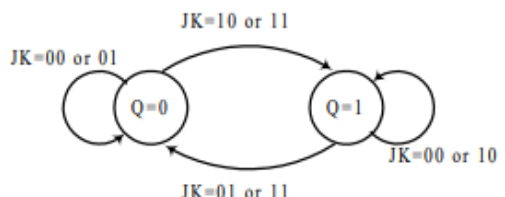
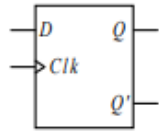
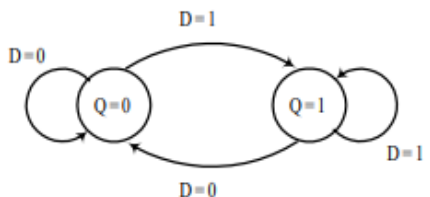
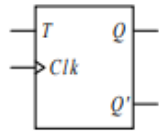
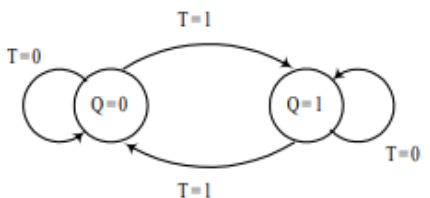
Question 6

[3*5=15 Marks]

There are basically four main types of flip-flops: SR, D, JK, and T. The major differences in these flip-flop types are in the number of inputs they have and how they change state. For each of these flip-flop types implement the following:

- Characteristic Table
- State Diagram and Characteristic equations
- Excitation table.

Solution:

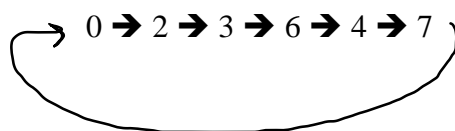
Name / Symbol	Characteristic (Truth) Table	State Diagram / Characteristic Equations	Excitation Table																																																								
SR 	<table><tr><th>S</th><th>R</th><th>Q</th><th>Q_{next}</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>×</td></tr><tr><td>1</td><td>1</td><td>1</td><td>×</td></tr></table>	S	R	Q	Q _{next}	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	×	 <p>$Q_{next} = S + R'Q$ $SR = 0$</p>	<table><tr><th>Q</th><th>Q_{next}</th><th>S</th><th>R</th></tr><tr><td>0</td><td>0</td><td>0</td><td>×</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>×</td><td>0</td></tr></table>	Q	Q _{next}	S	R	0	0	0	×	0	1	1	0	1	0	0	1	1	1	×	0
S	R	Q	Q _{next}																																																								
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	×																																																								
Q	Q _{next}	S	R																																																								
0	0	0	×																																																								
0	1	1	0																																																								
1	0	0	1																																																								
1	1	×	0																																																								
JK 	<table><tr><th>J</th><th>K</th><th>Q</th><th>Q_{next}</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	J	K	Q	Q _{next}	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	 <p>$Q_{next} = J'K'Q + JK' + JKQ'$ $= J'K'Q + JK'Q + JK'Q' + JKQ'$ $= K'Q(J' + J) + JQ'(K' + K)$ $= K'Q + JQ'$</p>	<table><tr><th>Q</th><th>Q_{next}</th><th>J</th><th>K</th></tr><tr><td>0</td><td>0</td><td>0</td><td>×</td></tr><tr><td>0</td><td>1</td><td>1</td><td>×</td></tr><tr><td>1</td><td>0</td><td>×</td><td>1</td></tr><tr><td>1</td><td>1</td><td>×</td><td>0</td></tr></table>	Q	Q _{next}	J	K	0	0	0	×	0	1	1	×	1	0	×	1	1	1	×	0
J	K	Q	Q _{next}																																																								
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	Q _{next}	J	K																																																								
0	0	0	×																																																								
0	1	1	×																																																								
1	0	×	1																																																								
1	1	×	0																																																								
D 	<table><tr><th>D</th><th>Q</th><th>Q_{next}</th></tr><tr><td>0</td><td>×</td><td>0</td></tr><tr><td>1</td><td>×</td><td>1</td></tr></table>	D	Q	Q _{next}	0	×	0	1	×	1	 <p>$Q_{next} = D$</p>	<table><tr><th>Q</th><th>Q_{next}</th><th>D</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	Q	Q _{next}	D	0	0	0	0	1	1	1	0	0	1	1	1																																
D	Q	Q _{next}																																																									
0	×	0																																																									
1	×	1																																																									
Q	Q _{next}	D																																																									
0	0	0																																																									
0	1	1																																																									
1	0	0																																																									
1	1	1																																																									
T 	<table><tr><th>T</th><th>Q</th><th>Q_{next}</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	T	Q	Q _{next}	0	0	0	0	1	1	1	0	1	1	1	0	 <p>$Q_{next} = TQ' + T'Q = T \oplus Q$</p>	<table><tr><th>Q</th><th>Q_{next}</th><th>T</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	Q	Q _{next}	T	0	0	0	0	1	1	1	0	1	1	1	0																										
T	Q	Q _{next}																																																									
0	0	0																																																									
0	1	1																																																									
1	0	1																																																									
1	1	0																																																									
Q	Q _{next}	T																																																									
0	0	0																																																									
0	1	1																																																									
1	0	1																																																									
1	1	0																																																									

[Solution](#)

Question 7

[10+5=15 Marks]

Design a synchronous counter by using T-Flip Flop(s) that counts in the following sequence:



1. Take the unused states as don't care
2. Modify the above counter to make it self-correcting (in case of an invalid input, the counter should move to the possible next state, for example if input is 1, then the next state should be 2).
3. Explain how design 1 and 2 are different and how did you construct it?

Solution:

