MODULE-5 **Hours-10**

Sequential Design - I:

Introduction, Mealy and Moore Models, State Machine Notation, Synchronous Sequential Circuit Analysis

Recommended readings:

1. Donald D Givone, "Digital Principles and Design ", Tata McGraw

Hill Edition, 2002.

Units-6.1, 6.2, 6.3

Mealy and Moore Type Finite State Machines

Objectives

There are two basic ways to design clocked sequential circuits. These are using:

1. Mealy Machine, which we have seen so far.
2. Moore Machine.

The objectives of this lesson are:

1. Study Mealy and Moore machines
2. Comparison of the two machine types
3. Timing diagram and state machines

Mealy Machine

- In a Mealy machine, the outputs are a function of the present state and the value of the inputs as shown in Figure 1.
- Accordingly, the outputs may change asynchronously in response to any change in the inputs.

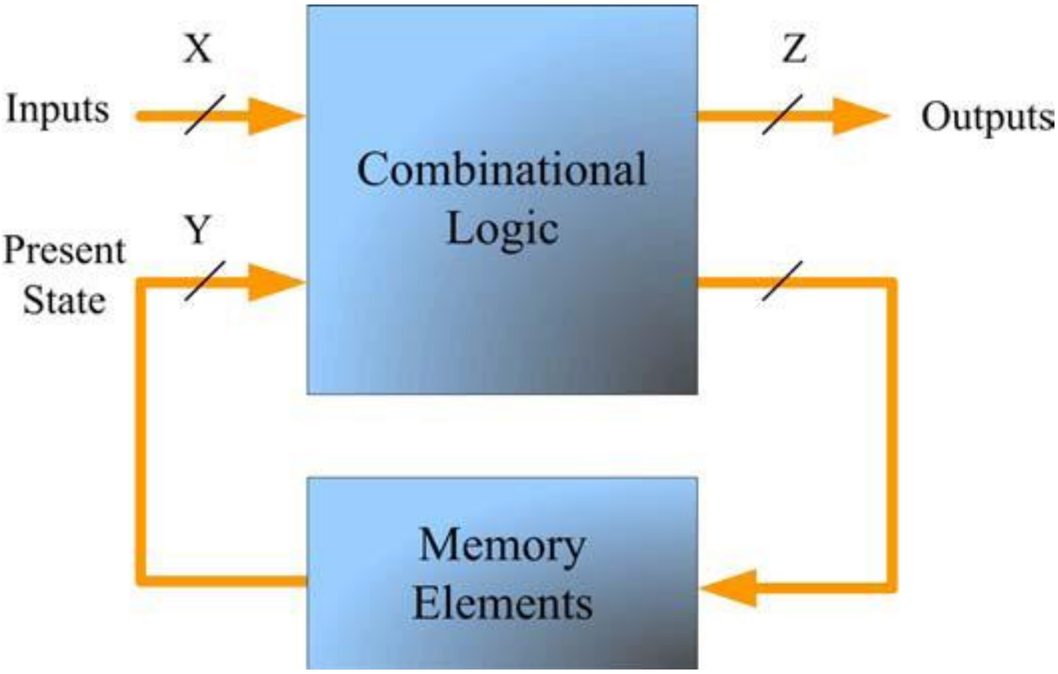


Figure 1: Mealy Type Machine

Mealy Machine

In a Moore machine the outputs depend only on the present state as shown in Figure 2.

A combinational logic block maps the inputs and the current state into the necessary flip-flop inputs to store the appropriate next state just like Mealy machine.

However, the outputs are computed by a combinational logic block whose inputs are only the flip-flops state outputs.

The outputs change synchronously with the state transition triggered by the active clock edge.

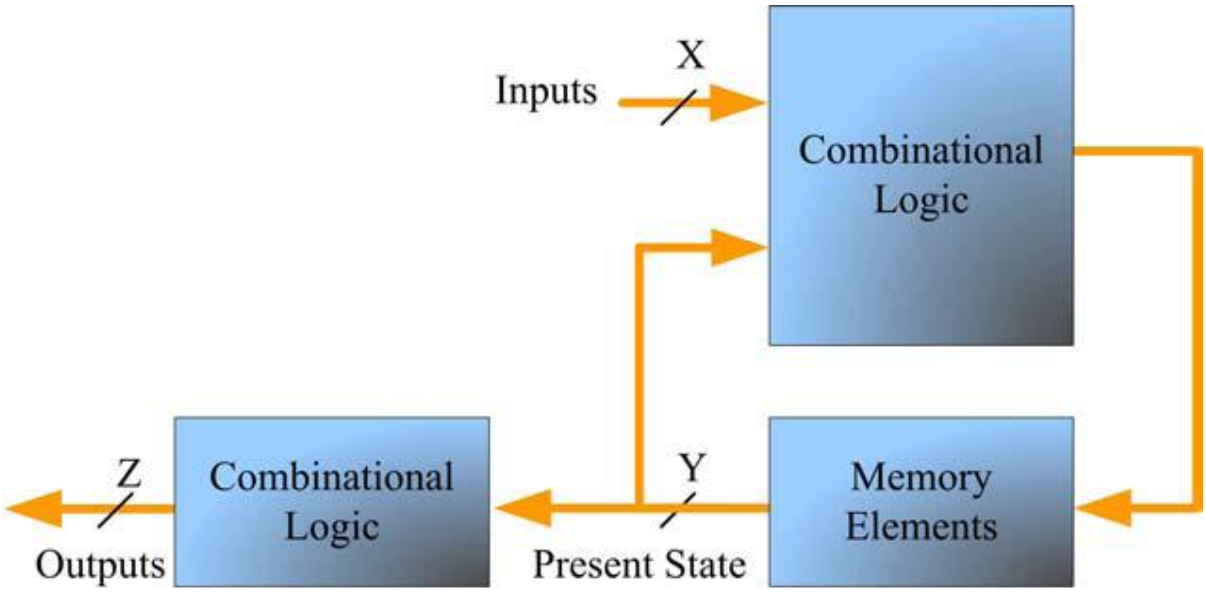


Figure 2: Moore Type Machine

Comparison of the Two Machine Types

Consider a finite state machine that checks for a pattern of '10' and asserts logic high when it is detected.

The state diagram representations for the Mealy and Moore machines are shown in Figure 3.

The state diagram of the Mealy machine lists the inputs with their associated outputs on state transitions arcs.

The value stated on the arrows for Mealy machine is of the form Zi/Xi where Zi represents input value and Xi represents output value.

A Moore machine produces a unique output for every state irrespective of inputs.

- Accordingly the state diagram of the Moore machine associates the output with the state in the form state-notation/output-value.

- The state transition arrows of Moore machine are labeled with the input value that triggers such transition.

- Since a Mealy machine associates outputs with transitions, an output sequence can be generated in fewer states using Mealy machine as compared to Moore machine. This was illustrated in the previous example.

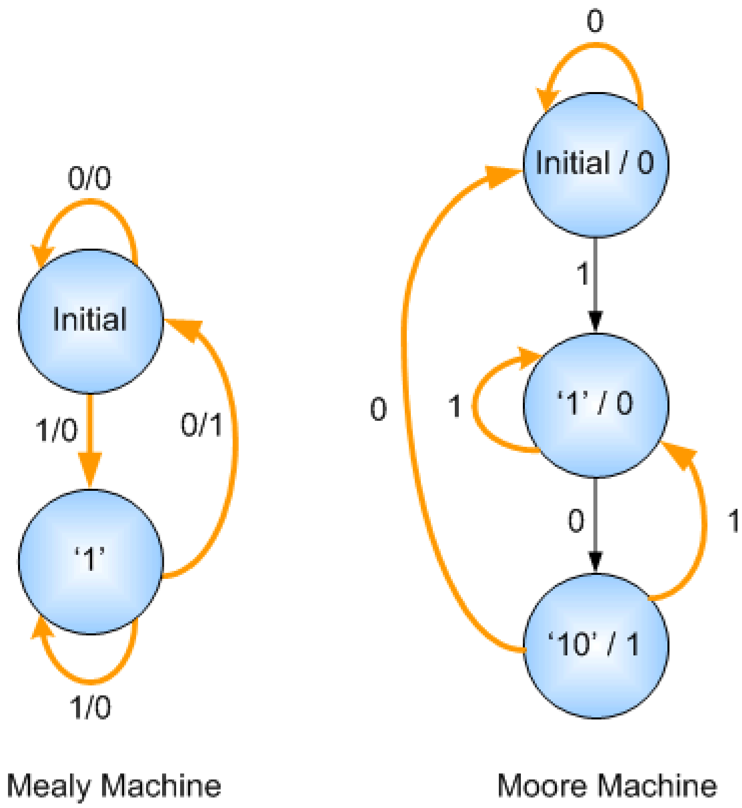


Figure 3: Mealy and Moore State Diagrams for '10' Sequence Detector

Timing Diagrams

 To analyze Mealy and Moore machine timings, consider the following problem. A state-machine outputs ‘1’ if the input is ‘1’ for three consecutive clocks.

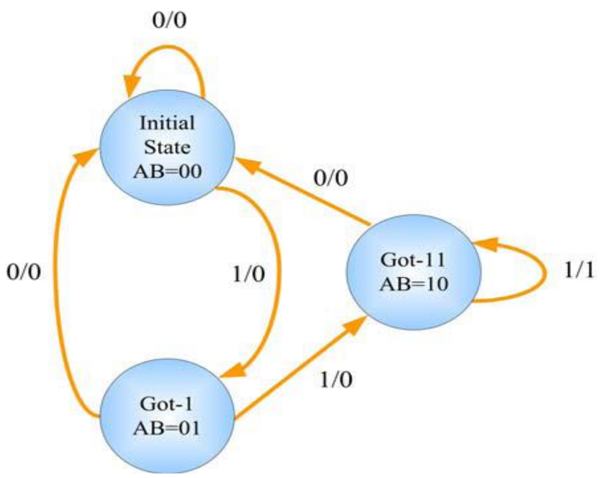


Figure 4: Mealy State Machine for '111' Sequence Detector

Mealy State Machine

 The Mealy machine state diagram is shown in Figure 4.

 Note that there is no reset condition in the state machine that employs two flip-flops. This means that the state machine can enter its unused state ‘11’ on start up.

 To make sure that machine gets resetted to a valid state, we use a ‘Reset’ signal.

 The logic diagram for this state machine is shown in Figure 5. Note that negative edge triggered flip-flops are used.

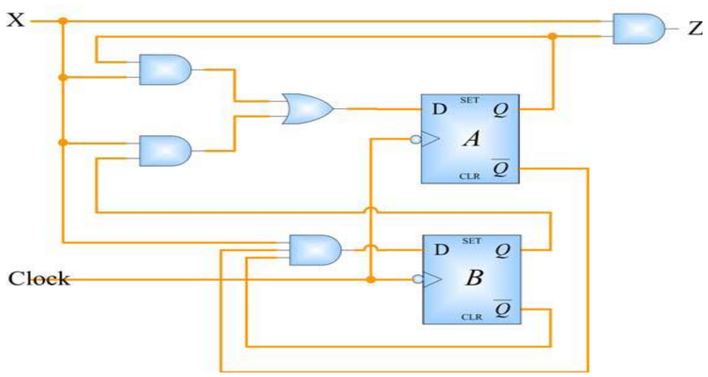


Figure 5: Mealy State Machine Circuit Implementation

 Since the output in Mealy model is a combination of present state and input values, an unsynchronized input with triggering clock may result in invalid output, as in the present case.

 Consider the present case where input 'x' remains high for sometime after state 'AB = 10' is reached. This results in 'False Output', also known as 'Output Glitch'.

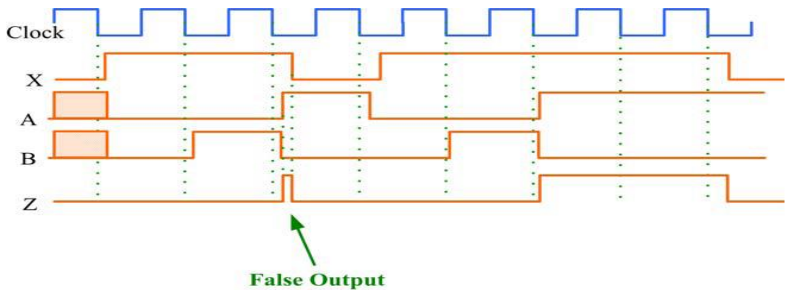


Figure 6: Timing Diagram for Mealy Model Sequence Detector

Moore State Machine

- The Moore machine state diagram for ‘111’ sequence detector is shown in Figure 7.
- The state diagram is converted into its equivalent state table (See Table 1).
- The states are next encoded with binary values and we achieve a state transition table (See Table 2).

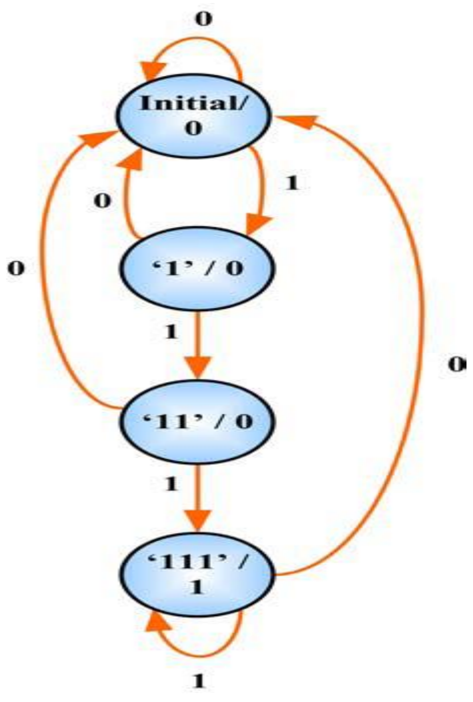


Figure 7: Moore Machine State Diagram

Table 1: State Table

Present	Next State		Output
Present State	Next State		Output
	x = 0	x = 1	Z
Initial	Initial	Got-1	0
Got-1	Initial	Got-11	0
Got-11	Initial	Got-111	0
Got-111	Initial	Got-111	1

Table 2: State Transition Table and Output Table

Present State	Next State x = 0	Next State x = 1	Output Z
Initial	Initial	Got-1	0
Got-1	Initial	Got-11	0
Got-11	Initial	Got-111	0
Got-111	Initial	Got-111	1

- We will use JK and D flip-flops for the Moore circuit implementation. The excitation tables for JK and D flip-flops (Table 3 & 4) are referenced to tabulate excitation table (See Table 5).

Table 3: Excitation Table for JK flip-flop

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table 4: Excitation Table for D flip-flop

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

Table 5: Excitation Table for the Moore Implementation

Inputs of Comb.Circuits			Next State		Outputs of Comb.Circuit			Output
Present State		Input			Flip-flop Inputs			
A	B	X	A	B	J_A	K_A	D_B	Z
0	0	0	0	0	0	X	0	0
0	0	1	0	1	0	X	1	0
0	1	0	0	0	0	X	0	0
0	1	1	1	0	1	X	0	0
1	0	0	0	0	X	1	0	0
1	0	1	1	1	X	0	1	0
1	1	0	0	0	X	1	0	1
1	1	1	1	1	X	0	1	1

- Simplifying Table 5 using maps, we get the following equations:
  - o $\mathbf{J_A = X.B}$
  - o $\mathbf{K_A = X'}$
  - o $\mathbf{D_B = X(A + B)}$
  - o $\mathbf{Z = A . B}$
- Note that the output is a function of present state values only.
- The circuit diagram for Moore machine circuit implementation is shown in Figure 8.
- The timing diagram for Moore machine model is also shown in Figure 9.
- There is no false output in a Moore model, since the output depends only on the state of the flop flops, which are synchronized with clock. The outputs remain valid throughout the logic state in Moore model.

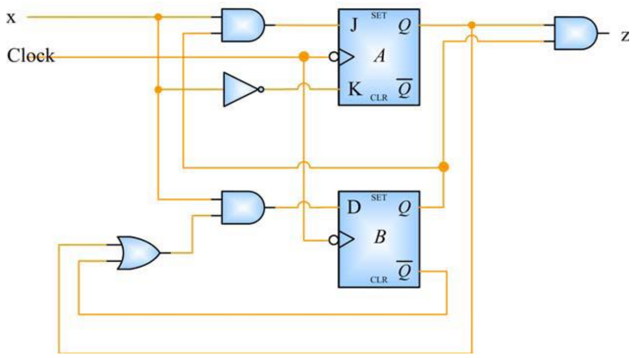


Figure 8: Moore Machine Circuit Implementation for Sequence Detector.

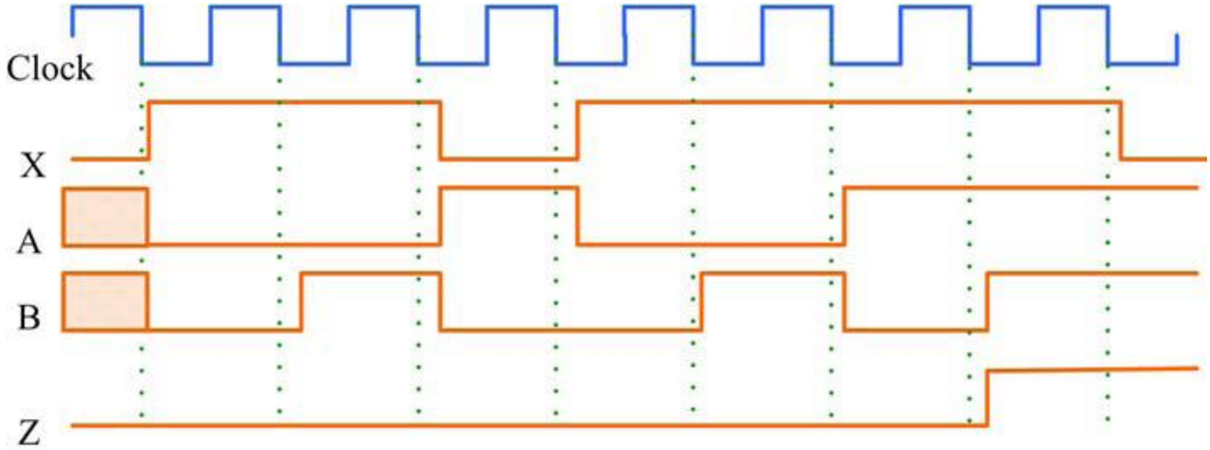


Figure 9: Timing Diagram for Moore Model Sequence Detector.

Recommended question and answer –unit-7

Jan -2009

Q.7 b) *Give output function, excitation table and state transition diagram by analyzing the sequential circuit shown in Fig.* 7.
(12)

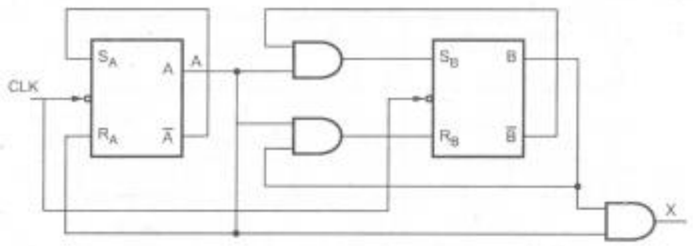


Fig. 7

Ans. : The excitation equations are :

$S_A = \overline{A}$ $S_B = A\overline{B}$

$R_A = A$ $R_B = AB$

The output equation is

$X = AB$

Evaluation of excitation and output expressions :

A	B	S_A	R_A	S_B	R_B	X
0	0	1	0	0	0	0

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

Excitation table :

Present state		Excitation S_A R_A, S_B R_B for input	Output X
A	B		
0	0	1 0, 0 0	0
0	1	1 0, 0 0	0
1	0	0 1, 1 0	0
1	1	0 1, 0 1	1

Transition table :

Present state		Next state		Output X
A	B	A^+	B^+	
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	0	1

Let the states be S_0 = 00, S_1 = 01, S_2 = 10 and S_3 = 11

State table :

Preset state	Next state	Output X
S_0	S_2	0
S_1	S_3	0
S_2	S_1	0
S_3	S_0	1

State diagram :

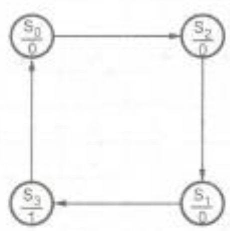


Fig. 7 (a)

Jan -2008

7.For the state machine M1 shown in Fig. 19 , obtain

i) State table

ii) Transition table

iii) Excitation table for T flip-flop

iv) Logic circuit for T excitation realization

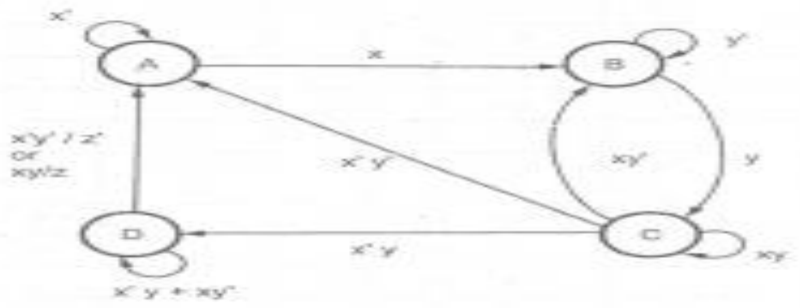


Fig. 19

Ans. : State table

Present state	Next state				Excitation table for T-FF		
	$\bar{X}\,\bar{Y}$	$\bar{X}Y$	$X\bar{Y}$	XY	RS	NS	T
	0 0	0 1	1 0	11	Q_1	Q_2	
A	A, 0	A, 0	B, 0	B, 0	0	0	0
B	B, 0	C, 0	B, 0	C, 0	0	1	1
C	A, 0	D, 0	B, 0	C, 0	1	0	1
D	A, 0	D, 0	D, 0	A, 0	1	1	0
Let	A = 00	B = 01	C = 10	D = 11			

Transition table

	Input		PS		NS		T_2	T_1
			P	Q	P^{+1}	Q^{+1}		
A	0	0	0	0	0	0	0	0
	0	1	0	0	0	0	0	0
	1	0	0	0	1	0	1	0
	1	1	0	0	1	0	1	0
B	0	0	0	1	0	1	0	0
	0	1	0	1	1	0	1	1
	1	0	0	1	0	1	0	0
	1	1	0	1	1	0	1	1
C	0	0	1	0	0	0	1	0
	0	1	1	0	1	1	0	1
	1	0	1	0	0	1	1	1
	1	1	1	0	1	0	0	0
D	0	0	1	1	0	0	1	1
	0	1	1	1	1	1	0	0
	1	0	1	1	1	1	0	0
	1	1	1	1	0	0	1	1

Using K-map we find the T_1 and T_2

For T_2

T_2 XY \ PQ	00	01	11	10
00	0	0	1	1
01	0	1	0	0
11	1	1	1	0
10	1	0	0	1

Fig. 20

$$T_2 = \overline{X}\,\overline{X}P + Y\overline{P}Q + XY\overline{P} + XYQ + X\overline{Y}\,\overline{Q}$$
$$= \overline{X}\,\overline{Y}P + Y\overline{P}Q + XY(\overline{P} + Q) + X\overline{Y}\,\overline{Q}$$

T_1 XY \ PQ	00	01	11	10
00	0	0	1	0
01	0	1	0	1
11	0	1	1	0
10	0	0	0	1

Fig. 21

$$T_1 = \overline{X}\,\overline{Y}PQ + Y\overline{P}Q + XYQ + X\overline{Y}P\overline{Q} + \overline{X}YP\overline{Q}$$
$$= \overline{X}\,\overline{Y}PQ + YQ(\overline{P} + X) + P\overline{Q}\,(\overline{X}Y + X\overline{Y})$$
$$= \overline{X}\,\overline{Y}PQ + YQ(\overline{P} + X) + P\overline{Q}\,(X \oplus Y)$$

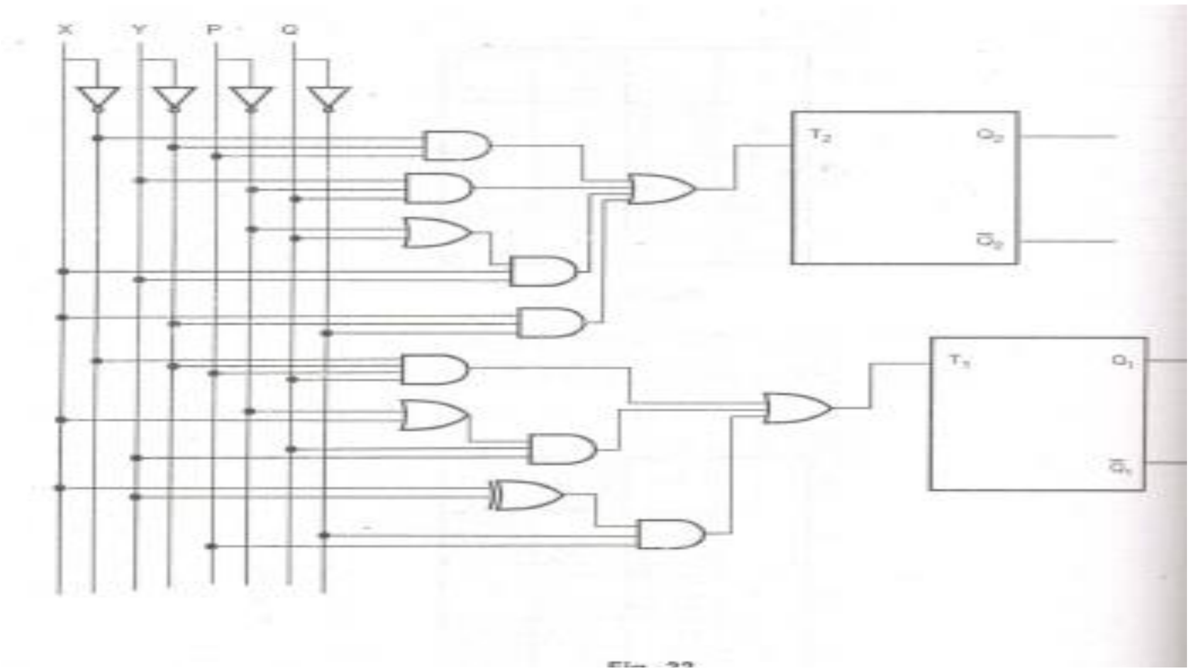


Fig. 22

Aug-2009

c.State the rules for state assignments.

Ans. : Rules for state assignments

There are two basic rules for making state assignments.

Rule 1: States having the same NEXT STATES for a given input condition should have assignments which can be grouped into logically adjacent cells in a K-map.

Fig. 12 shows the example for Rule 1. As shown in the Fig. 12, there are four states whose next state is same. Thus states assignments for these states are 100, 101,110 and 111, which can be grouped into logically adjacent cells in a K-map.

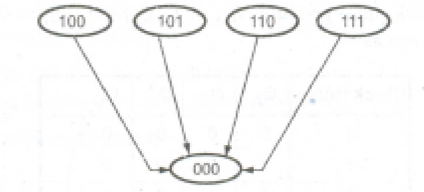


Fig. 12 Example of using Rule 1

Rule 2: States that are the NEXT STATES of a single state should have assignment which can be grouped into logically adjacent cells in a K-map.

Fig. 13 shows the example for Rule 2. As shown in the Fig. 13 for state 000, there are four next states. These states are assigned as 100, 101, 110 and 111 so that they can

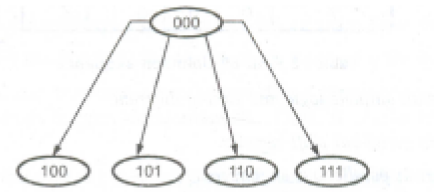


Fig. 13 Example of using Rule 2

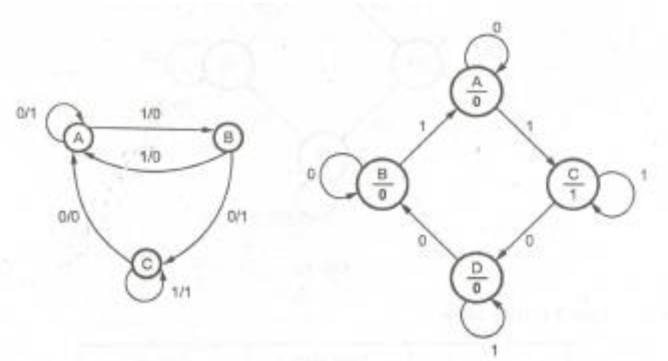
be grouped into logically adjacent cells in a K-map and table shows the state table with assigned statt

Present State	Next state		Output	
	X = 0	X = 1	X = 0	X = 1
00	01	10	0	0
01	11	10	1	0
10	10	11	0	1
11	00	11	0	0

Table 4 State table with assigned states

Aug 2008

7. b) *Construct the state table for the following state diagram.*



Ans. :

Input x	Present state	Next state	Output Y
0	A	A	1
0	B	C	1
0	C	A	0
1	A	B	0
1	B	A	0
1	C	C	1

Input	Present state	Next state	Output
0	A	A	0
0	B	B	0
0	C	D	1
0	D	B	0
1	A →	C	0
1	B →	A	0
1	C →	C	1
1	D →	D	0

Aug-2007

applied to the J and K inputs of flip-flop C. Whenever both QA and QB are HIGH, the output of the AND gate makes the J and K inputs of flip-flop C HIGH, and flip-flop C toggles on the following clock pulse. At all other times, the J and K inputs of flip-flop C are held LOW by the AND gate *qutput*, and flip-flop does not change state.

b) *Explain the different types of shift register. 5150, SIPO, PIPO, PISO with relevant circuit diagram.* . [10]

Sol. :SISO Shift Register:

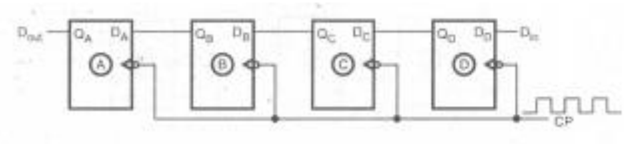


Fig. 13 Shift-left register

Fig. 13 shows serial in serial out shift-left register.

We will illustrate the entry of the four bit biIlary number 1111 into the register, beginning with the left-most bit.

Initially, register is cleared. SO

QAQBQCQO = 0000

a) When data 1 1 1 1 is applied serially, Le.

left-most 1 is applied as Din'

Din = 1, QAQBQCQo = 00 0 0

The arrival of the first falling clock edge sets the right-most flip-flop, and the stored word becomes, .

QAQBQCQO = 000 1

b) When the next negative clock edge hits, the Q1 flip-flop sets and the register contents become,

QAQBQCQO = 001 1

c) The third negative clock edge results in,

QAQBQCQO = *b* 1 1 1

d) The fourth falling clock edge gives,

QAQBQCQo = 1 1 1 1

SIPO Shift Register: In this case, the data bits are entered into the register in the same manner as discussed in the last section, i.e. serially. But the output is taken in parallel. Once the data are stored, each bit appears on its respective output line and all bits are available simultaneously, instead of on a bit-by-bit basis as with the serial output as shown in Fig. 14.

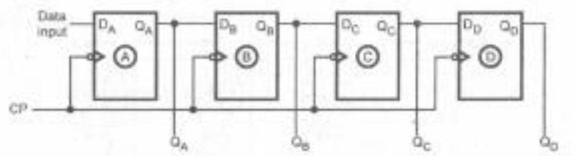


Fig. 14 A serial in parallel out shift register

PIPO **Shift Register :** From the third and second types of registers, **it** is cleared that how to enter the data LT\ parallel i.e. all bits simultaneously into the register and how to take data out in parallel from the register. In 'parallel in parallel out register', there is simultaneous entry of all data bits and the bits appear on parallel outputs simultaneously. Fig. 15 shows this type of register.

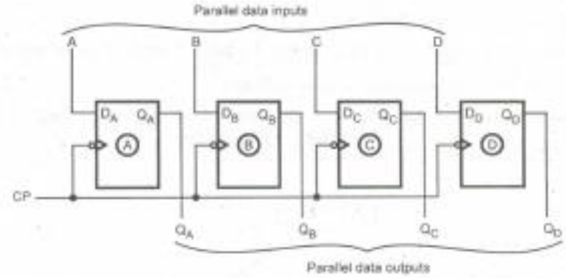


Fig. 15 Parallel in parallel out shift register

PISO Shift Register : In this type, the bits are entered in parallel i.e simultaneously into their respective stages on parallel lines.

Fig. 16 illustrates a four-bit parallel in serial out register. There are four input iines *XAI* XIY Xc, XD for entering data in parallel into the register. SHIFT/LOAD is the control input which allows shift or loading data operation of the register. When SHIFT/LOAD is low, gates G" G2, G3 are enabled, allowing each input data bit to be applied to D input of its respective flip-flop. When a clock pulse is applied, the flip-flops with D = 1 will SET and those with D = 0 will RESET. Thus all four bits are stored simultaneously.

When SHIFT/LOAD is high, gates G1, G1, G3 are disabled and gates G4' *Gy* G6 are enabled. This allows the data bits to shift .left from one stage to the next. The OR gates at the D-inputs of the flip-flops allow either the parallel data entry operation or shift o!,eration, depending on which AND gates are enabled by the level on the SHIFT/LOAD input.

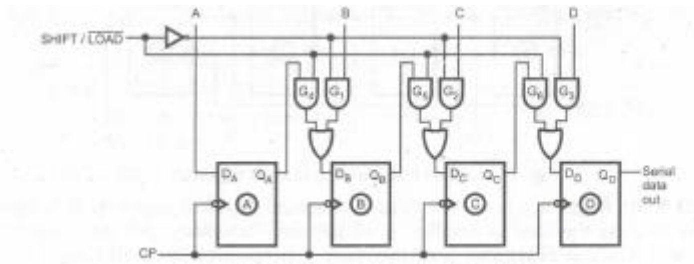


Fig. 16 Parallel in serial out shift register

Unit 8: 6 Hours

Sequential Design - II:

Construction of state Diagrams, Counter Design

Recommended readings:

1. Donald D Givone, "Digital Principles and Design ", Tata McGraw Hill Edition, 2002.

Units- 6.4, 6.5

Design of Synchronous Sequential Circuits

Objectives

1. Design of synchronous sequential circuits with an example.

2. Construction of state diagrams and state tables/

3. Translation of State transition table into excitation table.

4. Logic diagram construction of a synchronous sequential circuit

Sequential Circuit Design Steps

- The design of sequential circuit starts with verbal specifications of the problem (See Figure 1).

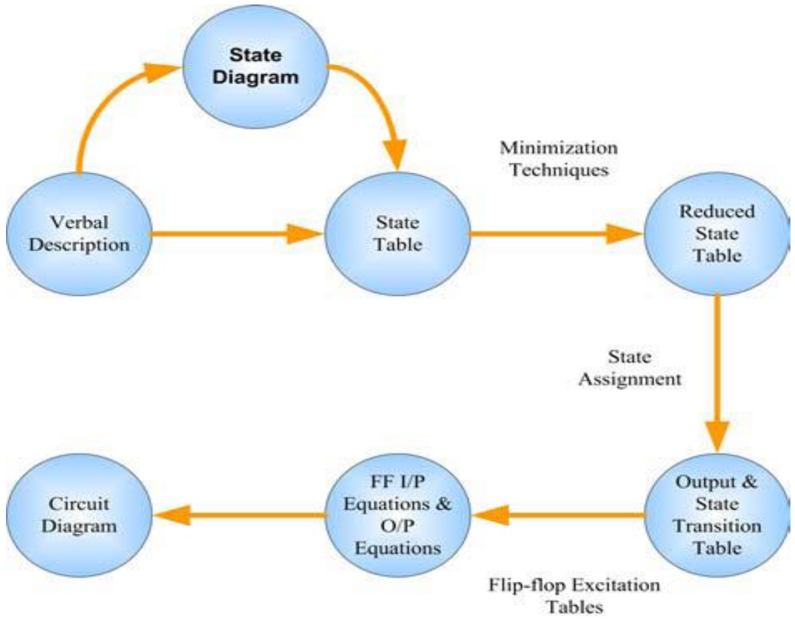


Figure 1: Sequential Circuit Design Steps

The next step is to derive the state table of the sequential circuit. A state table represents the verbal specifications in a tabular form.

In certain cases state table can be derived directly from verbal description of the problem.

In other cases, it is easier to first obtain a state diagram from the verbal description and then obtain the state table from the state diagram.

A state diagram is a graphical representation of the sequential circuit.

In the next step, we proceed by simplifying the state table by minimizing the number of states and obtain a reduced state table.

The states in the reduced state table are then assigned binary-codes. The resulting table is called output and state transition table.

From the state transition table and using flip-flop's excitation tables, flip-flops input equations are derived. Furthermore, the output equations can readily be derived as well.

Finally, the logic diagram of the sequential circuit is constructed.

An example will be used to illustrate all these concepts.

Sequence Recognizer

A sequence recognizer is to be designed to detect an input sequence of '1011'. The sequence recognizer outputs a '1' on the detection of this input sequence. The sequential circuit is to be designed using JK and D type flip-flops.

A sample input/output trace for the sequence detector is shown in Table 1.

Table 1: Sample Input/Output Trace

Input	0	1	1	0	1	0	1	1	0	1	1	1	0	1	0	1	1	1	0	0
Output	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0

We will begin solving the problem by first forming a state diagram from the verbal description.

A state diagram consists of circles (which represent the states) and directed arcs that connect the circles and represent the transitions between states.

In a state diagram:

1. The number of circles is equal to the number of states. Every state is given a label (or a binary encoding) written inside the corresponding circle.

2. The number of arcs leaving any circle is 2^n, where n is the number of inputs of the sequential circuit.

3. The label of each arc has the notation x/y, where x is the input vector that causes the state transition, and y is the value of the output during that present state.

4. An arc may leave a state and end up in the same or any other state.

Before we begin our design, the following should be noted

1. We do not have an idea about how many states the machine will have.

2. The states are used to “remember” something about the history of past inputs. For the sequence 1011, in order to be able to produce the output value 1 when the final 1 in the sequence is received, the circuit must be in a state that “remembers” that the previous three inputs were 101.

3. There can be more than one possible state machine with the same behavior.

.

Deriving the State Diagram

Let us begin with an initial state (since a state machine must have at least one state) and denote it with **‘S0’** as shown in Figure 2 (a).

Two arcs leave state **‘S0’** depending on the input (being a 0 or a 1). If the input is a 0, then we return back to the same state. If the input is a 1, then we have to remember it (recall that we are trying to detect a sequence of 1011). We remember that the last input was a one by changing the state of the machine to a new state, say **‘S1’**. This is illustrated in Figure 2 (b).

‘S1’ represents a state when the last single bit of the sequence was one. Outputs for both transitions are zero, since we have not detected what we are looking for.

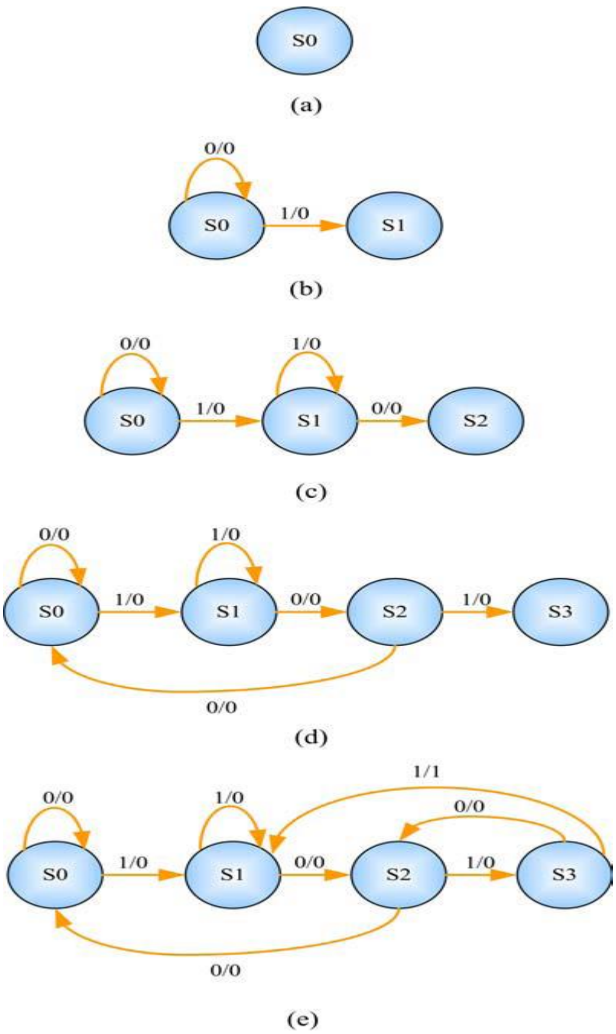
Again in state **‘S1’,** we have two outgoing arcs. If the input is a 1, then we return to the same state and if the input is a 0, then we have to remember it (second number in the sequence). We can do so by transiting to a new state, say **‘S2’**. This is illustrated in Figure 2 (c).

Note that if the input applied is ‘1’, the next state is still **‘S1’** and not the initial state **‘S0’**. This is because we take this input 1 as the first digit of new sequence. The output still remains 0 as we have not detected the sequence yet.

State **‘S2’** represents detection of ‘10’ as the last two bits of the sequence. If now the input is a ‘1’, we have detected the third bit in our sequence and need to remember it. We remember it by transiting to a new state, say **‘S3’** as shown in Figure 2 (d). If the input is ‘0’ in state **‘S2’** then it breaks the sequence and we need to start all over again. This is achieved by transiting to initial state **‘S0’**. The outputs are still 0.

In state **‘S3’**, we have detected input sequence ‘101’. Another input 1 completes our detection sequence as shown in Figure 2 (e). This is signaled by an output 1. However we transit to state **‘S1’** instead of **‘S0’** since this input 1 can be counted as first 1 of a new sequence. Application of input 0 to state **‘S3’** means an input sequence of 1010. This implies the last two bits in the sequence were 10 and we transit to a state that remembers this input sequence, i.e. state **‘S2’**. Output remains as zero.

Figure 2: Deriving the State Diagram of the Sequence Recognizer



Deriving the State Table

A state table represents time sequence of inputs, outputs, and states in a tabular form. The state table for the previous state diagram is shown in Table 2.

The state table can also be represented in an alternate form as shown in Table 3.

Here the present state and inputs are tabulated as inputs to the combinational circuit. For every combination of present state and input, next state column is filled from the state table.

The number of flip-flops required is equal to [$\log_2$(number of states)].

Thus, the state machine given in the figure will require two flip-flops □$\log_2(4)$□=2. We assign letters A and B to them.

Table 2: State Table of the Sequence Recognizer

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
S0	*S0*	*S1*	0	0
S1	*S2*	*S1*	0	0
S2	*S0*	*S3*	0	0
S3	*S2*	*S1*	0	1

Table 3: Alternative Format of Table 2

Inputs of Combinational Circuit		Next State	Output
Present State	Input		
S0	0	*S0*	0
S0	1	*S1*	0
S1	0	*S2*	0
S1	1	*S1*	0
S2	0	*S0*	0
S2	1	*S3*	0
S3	0	*S2*	0
S3	1	*S1*	1

State Assignment

The states in the constructed state diagram have been assigned symbolic names rather than binary codes.

It is necessary to replace these symbolic names with binary codes in order to proceed with the design.

In general, if there are **m** states, then the codes must contain **n** bits, where $2^n \geq m$, and each state must be assigned a unique code.

There can be many possible assignments for our state machine. One possible assignment is show in Table 4.

Table 4: State Assignment

State	*Assignment*
S0	00
S1	01
S2	10
S3	11

 The assignment of state codes to states results in state transition table as shown.

 It is important to mention here that the binary code of the present state at a given time **t** represents the values stored in the flip-flops; and the next-state represents the values of the flip-flops one clock period later, at time **t+1**.

Table 5: State Transition Table

Inputs of Combinational Circuit		Next State	Output
Present State	Input		
A B	*X*	*A B*	*Y*
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	1 0	0
0 1	1	0 1	0
1 0	0	0 0	0
1 0	1	1 1	0
1 1	0	1 0	0
1 1	1	0 1	1

General Structure of Sequence Recognizer

 The specifications required using JK and D type flip-flops.

Referring to the general structure of sequential circuit shown in Figure 3, our synthesized circuit will look like that as shown in the figure. Observe the feedback paths.

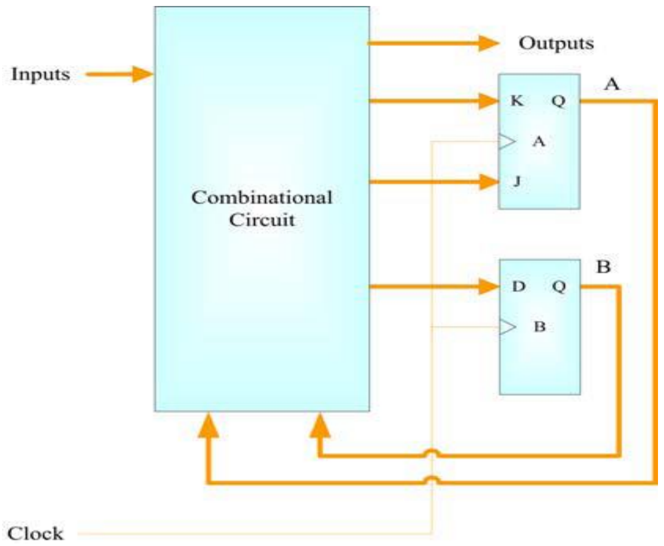


Figure 3: General Structure of the Sequenc Recognizer

- What remains to be determined is the combinational circuit which specifies the external outputs and the flip-flop inputs.

- The state transition table as shown can now be expanded to construct the excitation table for the circuit.

- Since we are designing the sequential circuit using JK and D type flip-flops, we need to correlate the required transitions in state transition table with the excitation tables of JK and D type-flip-flops.

- The functionality of the required combinational logic is encapsulated in the excitation table. Thus, the excitation table is next simplified using map or other simplification methods to yield Boolean expressions for inputs of the used flip-flops as well as the circuit outputs.

Deriving the Excitation Table

- The excitation table (See Table 6) describes the behavior of the combinational portion of sequential circuit.

Table 6: Excitation Table of the Sequence Recognizer

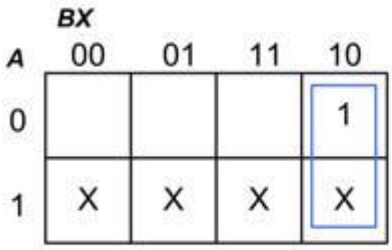
Present State		Input				Flip-flops Inputs		
A	B	X	A	B	Y	J_A	K_A	D_B
0	0	0	0	0	0	0	X	0
0	0	1	0	1	0	0	X	1
0	1	0	1	0	0	1	X	0
0	1	1	0	1	0	0	X	1
1	0	0	0	0	0	X	1	0
1	0	1	1	1	0	X	0	1
1	1	0	1	0	0	X	0	0
1	1	1	0	1	1	X	1	1

- For deriving the actual circuitry for the combinational circuit, we need to simplify the excitation table in a similar way we used to simplify truth tables for purely combinational circuits.

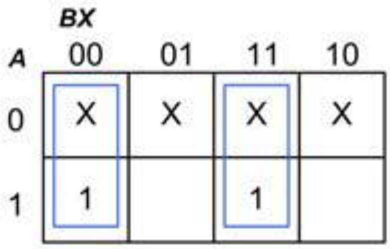
- Whereas in combinational circuits, our concern were only circuit outputs; in sequential circuits, the combinational circuitry is also feeding the flip-flops inputs. Thus, we need to simplify the excitation table for both outputs as well as flip-flops inputs.

- We can simplify flip-flop inputs and output using K-maps as shown in Figure 4.

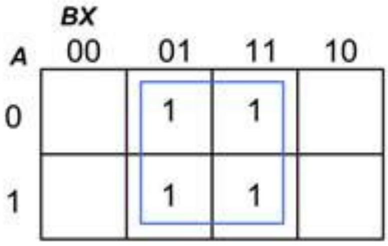
- Finally the logic diagram of the sequential circuit can be made as shown in Figure 5.



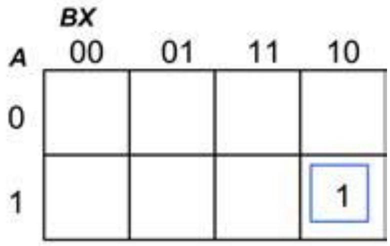
JA = BX'



KA = BX + B'X'



DB = X



Y = ABX'

Figure 4: Input Equations of the Sequence Recognizer

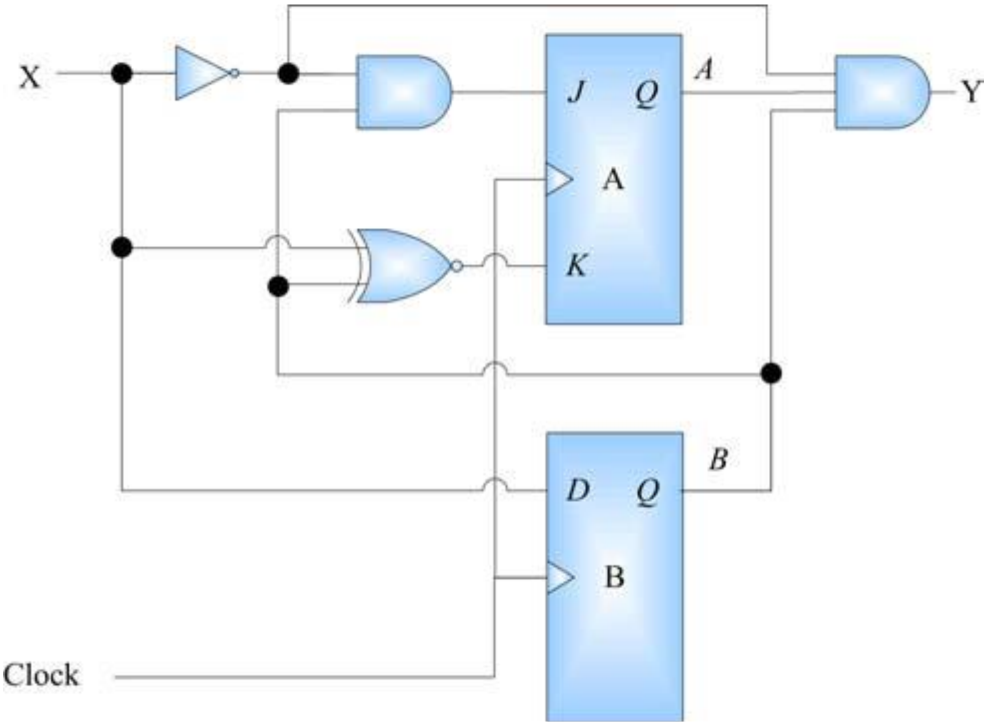


Figure 5: Circuit Diagram of the Sequence Recognizer

Recommended question and answer –unit-8

Q.8 a) *Design a cyclic mod-6 synchronous binary counter ? state diagram, transition table using JK flip-flop.* (10)

Ans. : Design of a synchronous mod-6 counter using clocked JK flip-flops The counter with n flip-flops has maximum mod number 2". For example, 3-bit binary counter is a mod 8 counter. This basic counter can be modified to produce MOD numbers less than 2" by allowing the counter to skip states those are normally

part of counting sequence. Let us design mod-6 counter using clocked JK flip-flops.

Step 1 : Find number of flip-flops required to build the counter : Flip-Flops required are: 2n >= N.

Here N = 6 :. n = 3

i.e. three flip-flops are required.

Step 2 : Write an excitation table for JK flip-flop.

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

Table 2

Step 3 : Determine the transition table.

Present state			Next state			Flip-flop inputs					
Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	0	0	0	x	1	0	x	x	1
1	1	0	x	x	x	x	x	x	x	x	x
1	1	1	x	x	x	x	x	x	x	x	x

Step 4 : K-map simplification for flip-flop inputs.

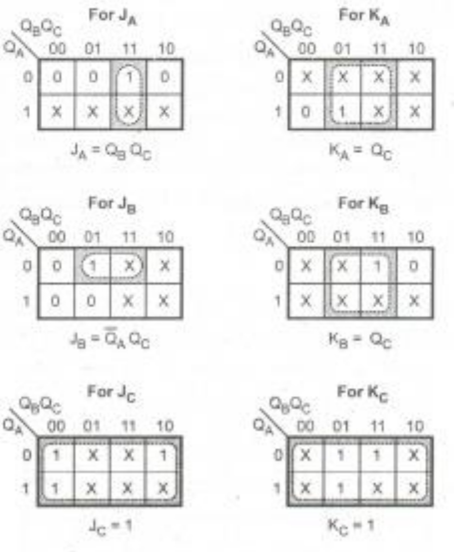


Fig. 8

Step 5 : Implement the counter.

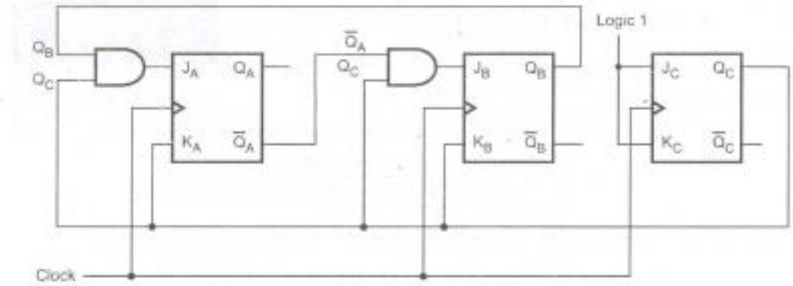


Fig. 9 Implementation of mod-6 synchronous counter

Note : To avoid crossing of lines and to have better clarity the circuit can also be represented as shown in Fig. 10. Here, instead of actual connections only signal names are specified.

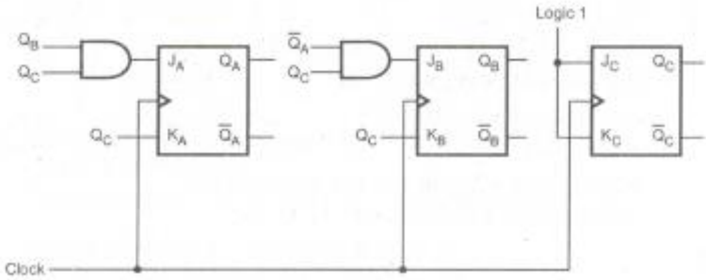


Fig. 10 Simplified representation of Fig. 9

Jan-2008

Q.8 a) *Construct a mealy state diagram that will detect a serial sequence of 10110.*WIr *the input pattern has been detected, cause an output* Z *to be asserted high.* I

Ans. :

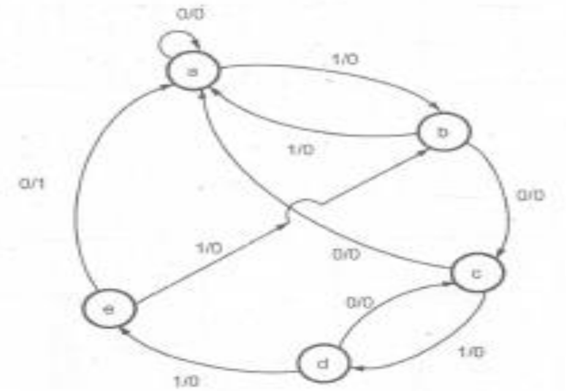


Fig. 23

State table

Present state	Next state		Output	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	a	0	0
c	a	d	0	0
d	c	e	0	0
e	a	b	1	0

Excitation table

Present state			Next state						Output	
			A^+	B^+	C^+	A^+	B^+	C^+		
A	B	C	x = 0			x = 1			x = 0	x = 1
0	0	0	0	0	0	0	0	1	0	0
0	0	1	0	1	0	0	0	0	0	0

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

K-map simplification

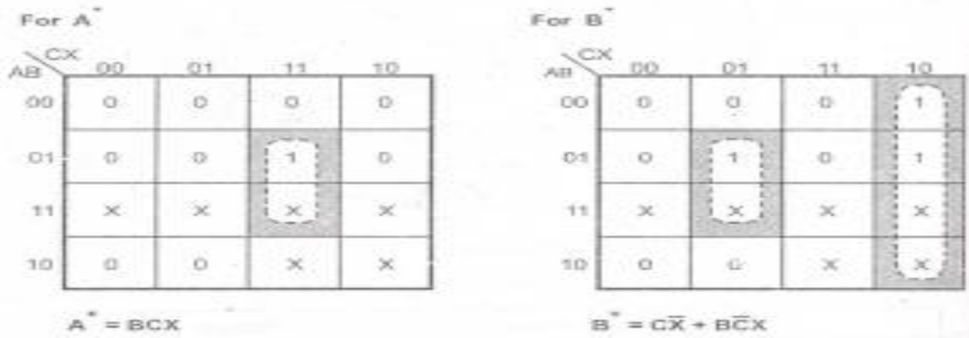
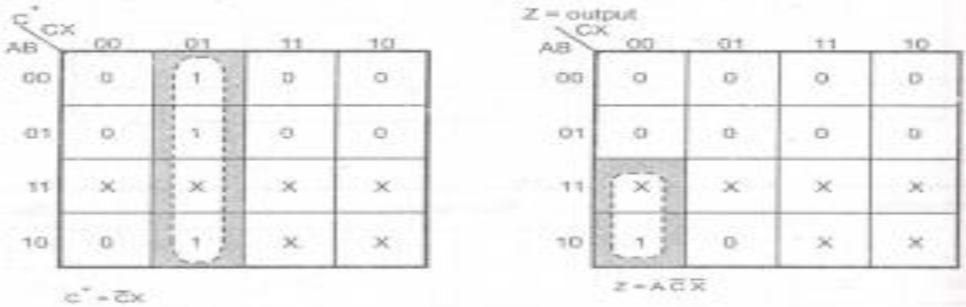
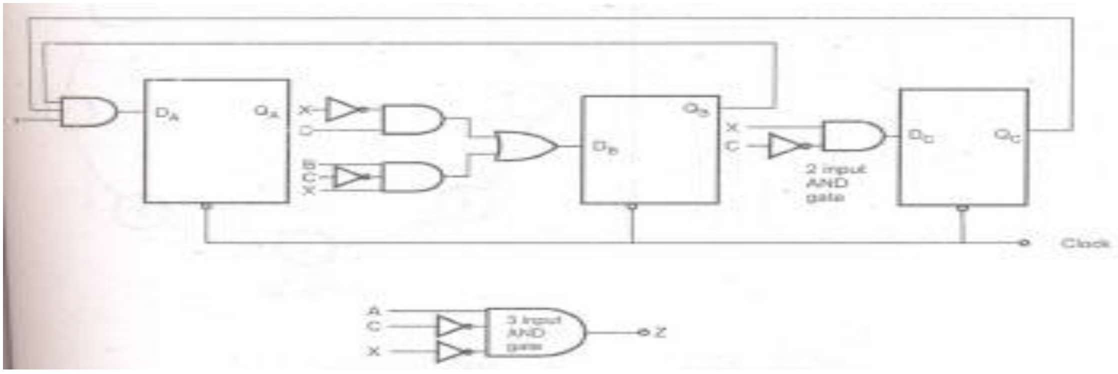


Fig. 24





b) *Design a cyclic modulo-8 synchronous counter using J-K flip-flop that will count the number of occurrences of an input; that is, the number of times it is a* 1. *The input variable* X *must be coincident with the clock to be counted. The counter is to count in binary.*

(12)

Ans. :

Present state			Next state			J_A	K_A	J_B	K_B	J_C	K_C
A	B	C	A_{+1}	B_{+1}	C_{+1}						
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

Excitation table

Present	Next	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

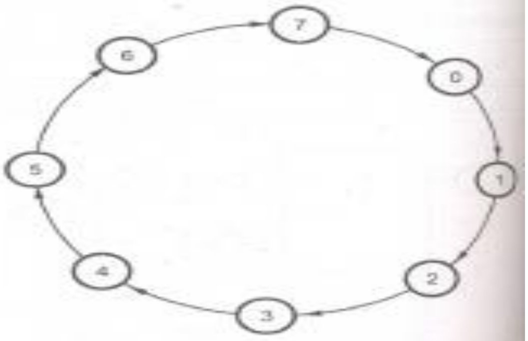


Fig. 27

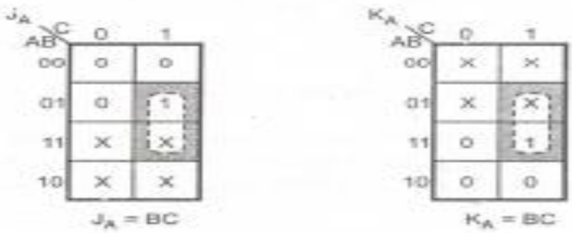


Fig. 28

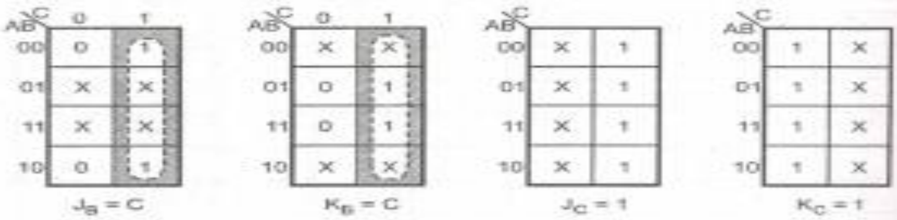


Fig. 29

Aug 2009

c.State the rules for state assignments.

Ans. : Rules for state assignments

There are two basic rules for making state assignments.

Rule 1: States having the same NEXT STATES for a given input condition should have assignments which can be grouped into logically adjacent cells in a K-map.

Fig. 12 shows the' example for Rule 1. As shown in the Fig. 12, there are four states whose next state is same. Thus states assignments for these states are 100, 101,110 and 111, which can be grouped into logically adjacent cells in a K-map.

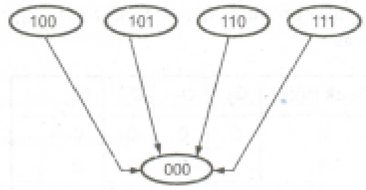


Fig. 12 Example of using Rule 1

Rule 2: States that are the NEXT STATES of a single state should have assignment which can be grouped into logically adjacent cells in a K-map.

Fig. 13 shows the example for Rule 2. As shown in the Fig. 13 for state 000, there are four next states. These states are assigned as 100, 101, 110 and 111 so that they can

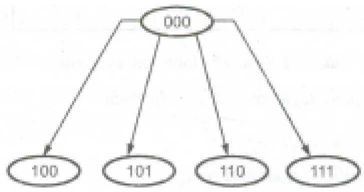


Fig. 13 Example of using Rule 2

be grouped into logically adjacent cells in a K-map and table shows the state table with assigned statt

Present State	Next state		Output	
	X = 0	X = 1	X = 0	X = 1
00	01	10	0	0
01	11	10	1	0
10	10	11	0	1
11	00	11	0	0

Table 4 State table with assigned states

Aug-2008

Q.8 a) *Design a· clocked sequential circuit that operates according to the state diagram* *shown. Implement the circuit using D flip-flop.* ***(12)***

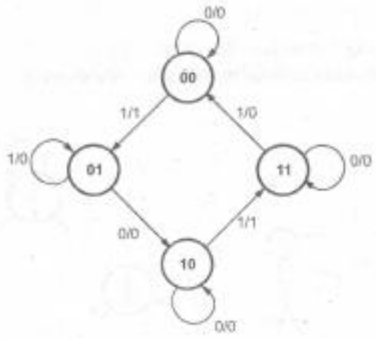


Fig. 12

Ans. : **Step 1 : State table**

Present state		Next state		Output	
		x = 0	x = 1	x = 0	x = 1
A	B	A^+ B^+	A^+ B^+	Y	Y
0	0	X X	0 1	X	1
0	1	1 0	0 1	0	0
1	0	1 0	1 1	0	1
1	1	1 1	0 0	0	0

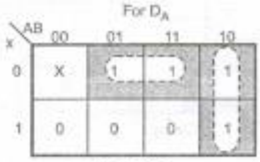
Step 2 : Design using D flip-flop

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

Present state		Next state				Output			
		x = 0		x = 1		x = 0		x = 1	
A	B	A^+	B^+	A^+	B^+	D_A	D_B	D_A	D_B
0	0	X	X	0	1	X	X	0	1
0	1	1	0	0	1	1	0	0	1
1	0	1	0	1	1	1	0	1	1
1	1	1	1	0	0	1	1	0	0

K-map simplification :

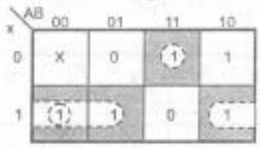
For D_A



$D_A = \bar{x}B + A\bar{B}$

Fig. 12 (a)

For D_B



$D_B = \bar{x}AB + x\bar{A} + x\bar{B}$

Fig. 12 (b)

For Y

x \ AB	00	01	11	10
0	X	0	0	0
1	1	0	0	1

$Y = x\overline{B}$

Fig. 12 (c)

Sequential circuit :

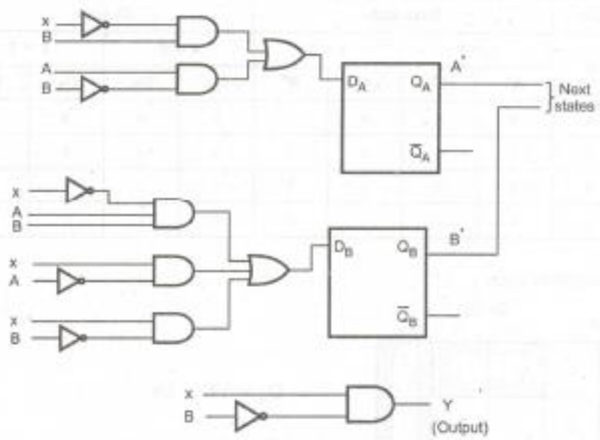


Fig. 12 (d)

b) *Design a counter using JK flip-flops whose counting sequence is 000, 001, 100, 110, 111, 101, 000 etc. by obtaining its minimal sum equations.* ***(8)***

Ans. :

Present state			Next State			J_1	K_1	J_2	K_2	J_3	K_3
C_n	B_n	A_n	C_{n+1}	B_{n+1}	A_{n+1}						
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	1	0	0	1	X	0	X	X	1

0	1	0	X	X	X	X	X	X	X	X	X
0	1	1	X	X	X	X	X	X	X	X	X
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	0	0	0	X	1	0	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	1	X	0	X	1	X	0

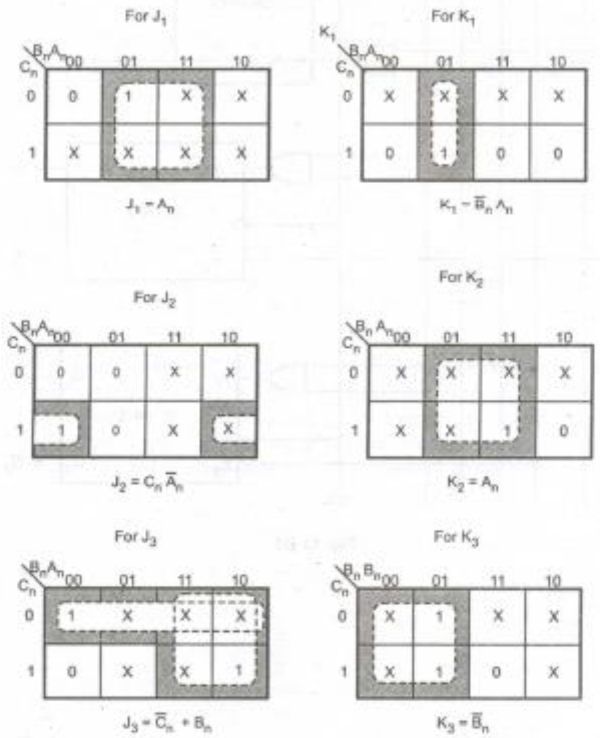


Fig. 13 (a)

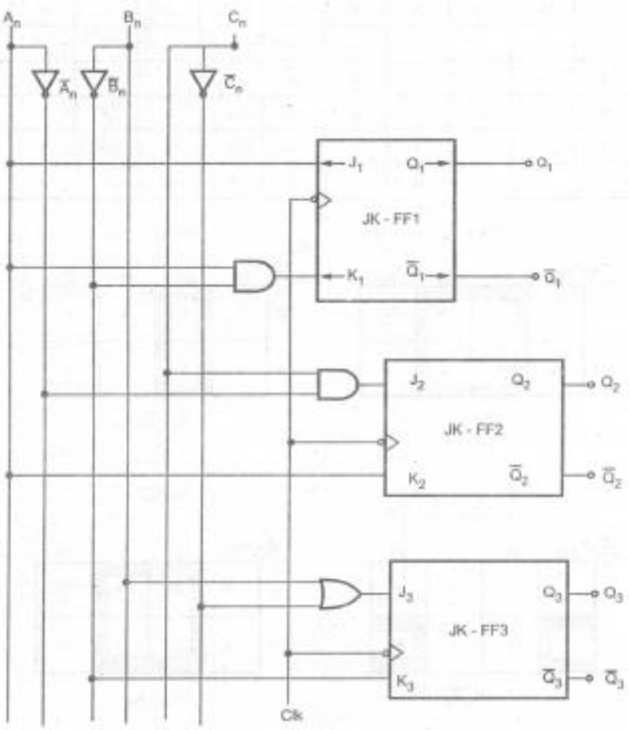


Fig. 13 (b)

Aug-2007

Q.8 a) *A combinational circuit is defined by the functions :*

$F_1 = \Sigma m(3,5,7)$, $F_2 = \Sigma m(4,5,7)$

Implement the circuit with a PLA having 3 inputs, 3 product terms and 2 outputs. **[8]**

Sol. : This topic is not included in the new syllabus.

b) *A sequential network has one input and one output. The state diagram is shown in Fig. 17. Design the sequential circuit with T flip flop.* **[12]**

Sol. :

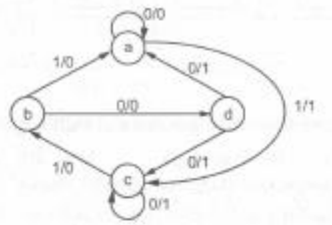


Fig. 17

Input	Present state		Next state		Flip-Flop inputs		Output
X	A	B	A^+	B^+	T_A	T_B	Z
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	1	1	1
1	0	0	1	0	1	0	1
1	0	1	0	0	0	1	0
1	1	0	0	1	1	1	0
1	1	1	1	0	0	1	0

Logic diagram

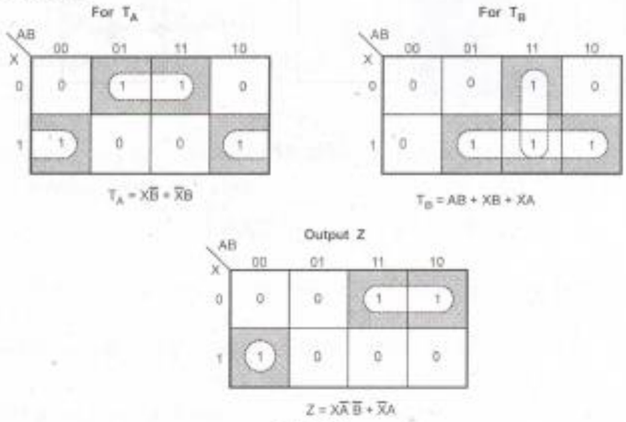


Fig. 18

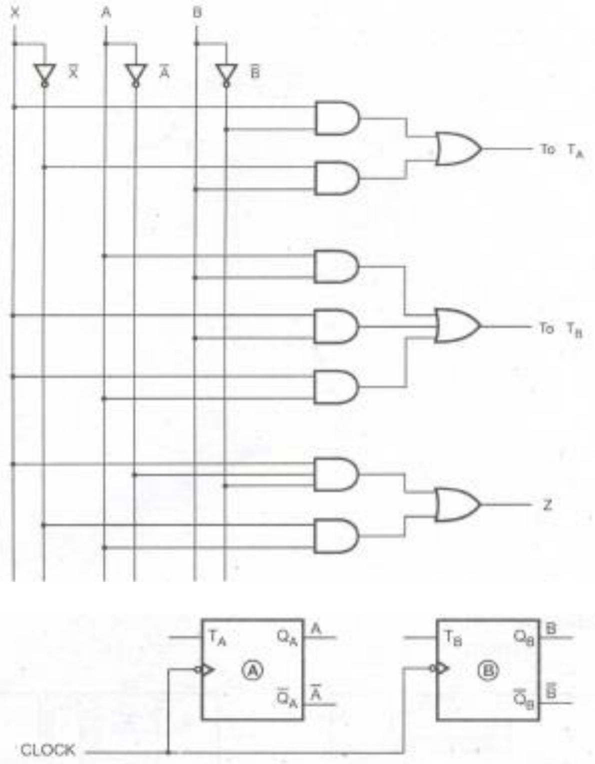


Fig. 19