CHAPTER - 11

Design of Sequential circuit

Introduction:

→ Design problem normally starts with description of input output relation & ends with a circuit diagram having sequential and combinatorial logic elements.

→ The word description is first converted to a state transition diagram followed by preparation of state synthesis table.

→ For flip flop based implementation, excitation tables are used to generate design equations through k-map.

→ The final circuit diagram is developed from these equations.

Part-A Design of Synchronous Sequential Circuit:

11.1 Model Selection:

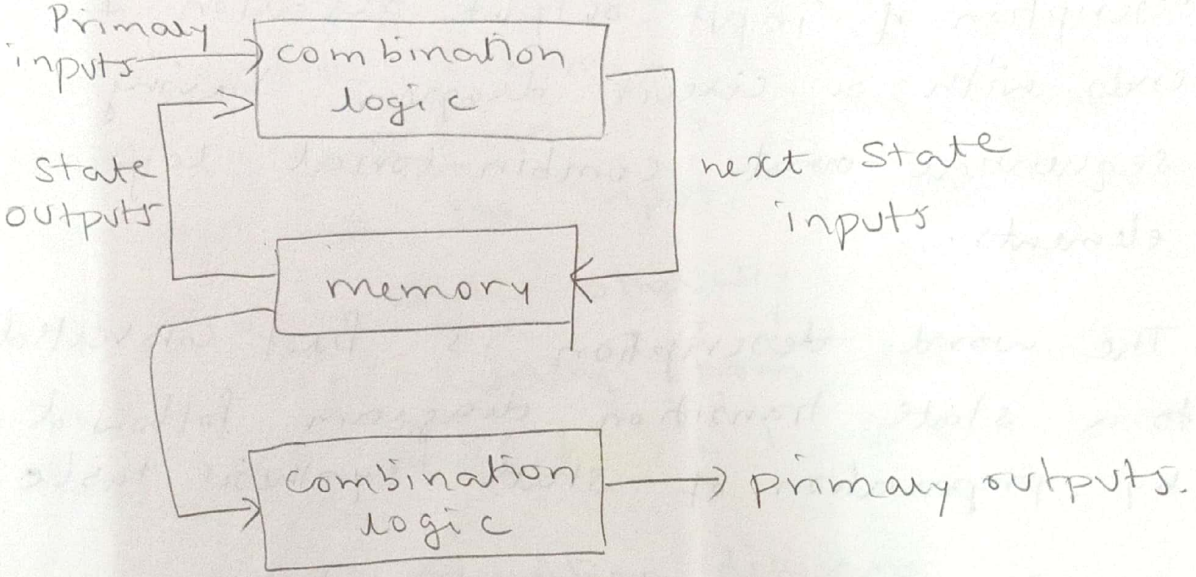
→ There are two distinct models by which a synchronous sequential logic circuit can be designed.

→ In moore model the output depends only on present state and not on input.

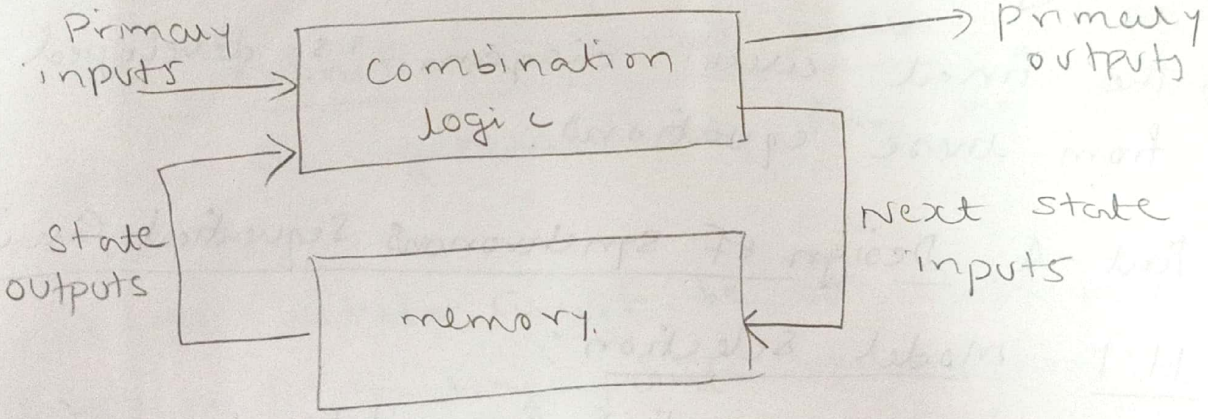
→ In mealy model the output is derived from present state as well as input

→ usually mealy model requires less number of states and thereby less hardware to solve any problem.

moore model.



mealy model.



⇒ Disadvantage with mealy → glitches appear at the output.

→ Advantage with moore → output remains stable over entire clock period and changes only when there occurs a state change.

→ Depending on application requirement, we choose one of these two models or a mixed model.

Problem:

→ Design a sequence detector that receives binary data stream at its input, x and signals → when a combination 011 arrives at the input by making its output, Y high otherwise Y = Low.

→ Data is coming from left, the first bit identified is 1, second 1, third 0.

11.2 State transition diagram:

→ The first step is to convert this word description to state transition diagram.

State definitions: moore model:

→ output is generated only from the state variables

→ let the detector circuit be at state **a** when initialized. (starting point of detection).

→ Then if 1st bit is detected properly the circuit should be at a different state **b**.

→ similarly we need two more states c & d to represent detection of 2nd and 3rd bit in proper order.

→ when detector circuit is at state d,

output Y is asserted and kept high as long as circuit remains in state d.

→ For other states detector output Y=0.

→ In moore model each state and output is defined within a circle in state transition diagram.

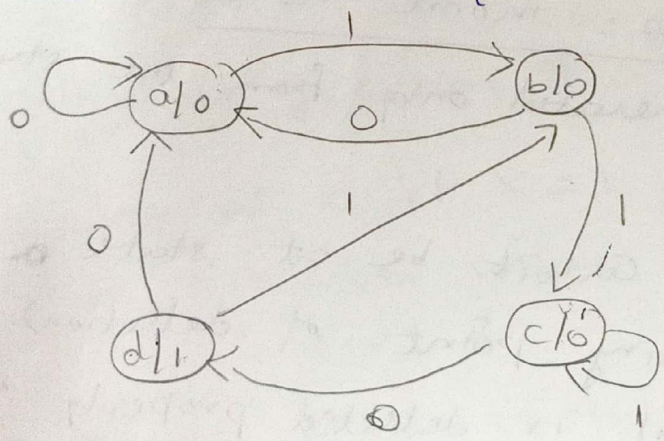
→ Format → S/Y.

S = symbol or memory values

Y = output.

→ An arrow sign marks state transition following an input value 0 or 1 that is written along the path.

→ Let x = binary input data



→ The above diagram shows the state transition diagram following moore model.

→ The circuit is initialized with state a

→ If input data X=1, the first bit of the sequence to be detected is considered detected & circuit goes to state B

→ If $x=0$ then it remains at state a to check next bit that arrives.

→ If at state b, the circuit receives $x=1$, then first two bit of the pattern is considered detected and it moves to state c.

→ But at state b, if it receives $x=0$ then detection has to start afresh as we need all the three bits of 011 to match.

→ Thus the detector goes back to initial state a.

→ At state c, if circuit receives $x=0$. Then input bit stream = 011 & the circuit goes to state d and signals detection of pattern at state d.

→ However at c if $x=1$, the detector is in a situation where it has received 111 in order.

→ It stays at c so that if next arriving bit $x=0$, it should signal sequence detection.

→ At d if the circuit continues sequence detection job, receiving $x=1$ it goes to state b.

→ This ensures detection of 011 second time.

State definition: Mealy model:

→ Since the output can be derived using the state as well as input we need three different states for 3 bit sequence detector circuit following mealy model.

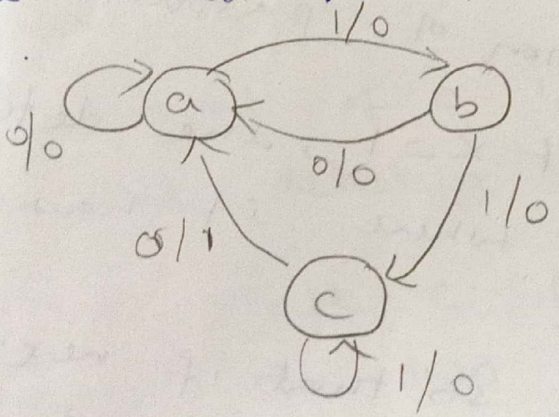
→ The three states say a, b, c represents none, 1st & 2nd bit detection.

→ when the circuit is at state c if the input is as per the pattern the output is generated in state c itself with proper logic combination of input.

State transition diagram: Mealy model:

→ The output is written by the side of input along arrow path in the format x/y where x and y represent i/p & o/p.

→



→ The circuit is initialized with state a. If it receives input X = 0, it stays at a else goes to b that signifies first bit is detected properly.

→ In both cases output Y = 0 signifying no detection.

→ At state b, if x = 0, the circuit returns to initial state a i.e. no bit in given order is detected and if x = 1 goes to state c, signifying two bits in order are detected.

→ In both cases Y = 0.

→ Now when at c, if input received is 0 then all the three bits of the Pattern are received properly and sequence detection can be signalled through Y = 1

→ Also the circuit goes to initial state a and prepares for a new set of detection.

→ At state c, if x = 1 then the sequence received is 111.

→ An arrival of 0 in next clock can make the detection '110' possible.

→ so at state c if x = 1 it is considered as two bits, "11" have been detected properly and the circuit remains at state c.

→ The output at that time is Y = 0 since sequence is not fully detected.

Conversion of models :

→ T_1, T_2, T_3 represent paths leading to state a.

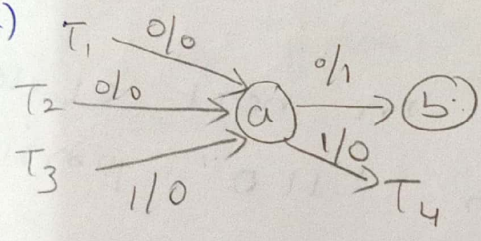
→ The path T_4 leads from state a when 'input is 1.

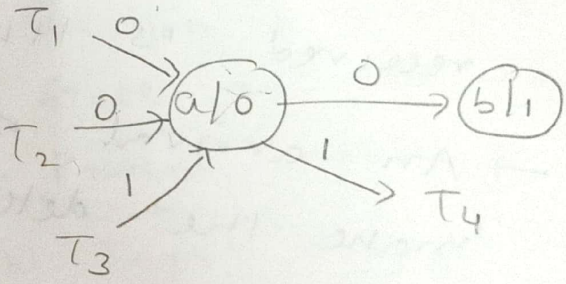
→ If 'input is 0, state a leads to state b and there are no other paths reaching b.

→ The rule of conversion is as follows.

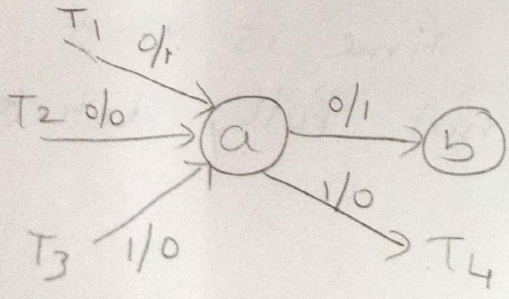
→ If all the transitions in a mealy model to a particular state are associated with only one type of output then in corresponding moore model that output becomes state output.

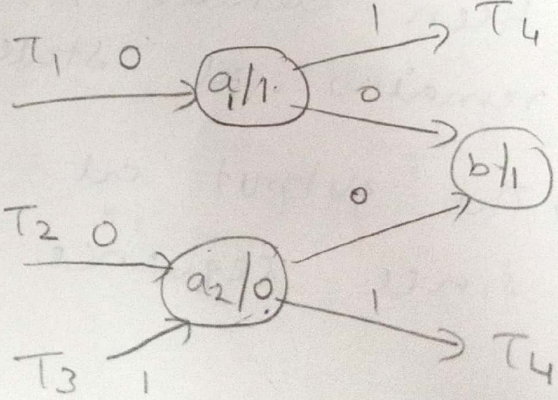
(a)





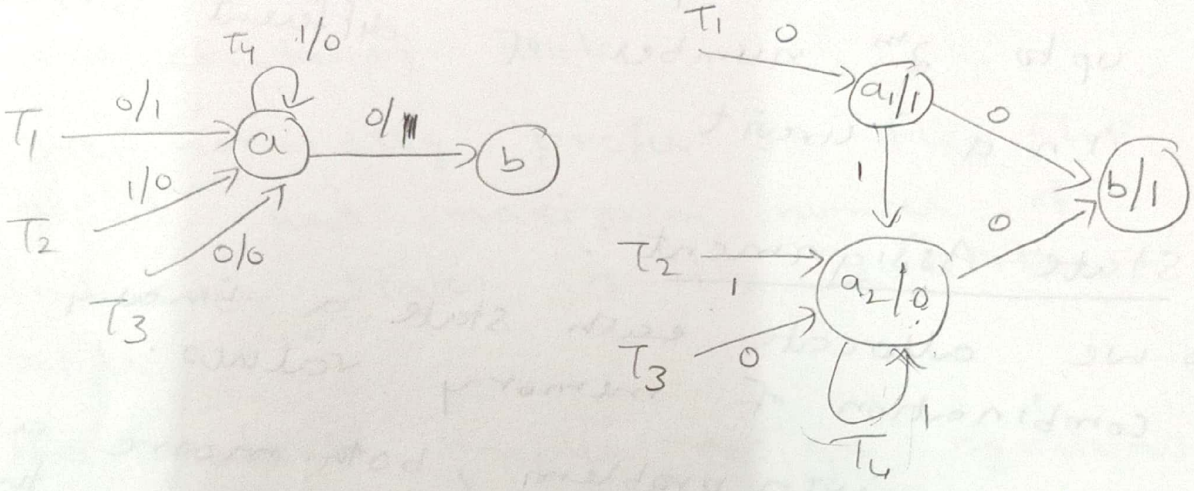
→ If there is more than one output in mealy model we need as many intermediate state variables, as shown in (b)





→ In fig-c it is shown how to treat transitions that loop within a particular state.

→ The reverse of this is applied in converting moore to mealy model.



Comparision of moore & mealy ckt model:

moore	mealy
a) Its output is a function of present state only	(b) Its output is a function of present state as well as Present input.
) Input changes does not affect the output	(b) Input changes may affect the output of the circuit
) moore circuit requires more number of states for implementing same function	(c) It requires less number of states for implementing same function.

11.3 State synthesis table:

→ Next step is to develop state synthesis table also called circuit excitation table, or state table.

→ For m memory elements we can have upto 2^m number of different states in a circuit.

State Assignment:

→ we allocate each state a binary combination of memory values.

→ For the given problem, both moore and mealy models require minimum two flip flops (say A & B).

→ let the state assignment be as follows.

a : B = 0 A = 0

b : B = 0 A = 1

c : B = 1 A = 0

d : B = 1 A = 1

→ mealy model does not use stated.

→ Assignment can be done in any order, and proceed with the design

State synthesis table:

→ Flip flops are commonly used for this purpose

→ we use diffent types of flip flops

→ Each has a unique characteristic equation and excitation table

→ we normally prefer JK flip flop as it has maximum number of dont care states in its excitation table

→ This leads to simpler design equations.

moore model:

→ we have four possible states for the moore model of the sequence detector problem.

→ There can be two diffrent types of inputs.

Present state		present input	Next state		o/p	$J_B K_B$	$J_A K_A$
B_n	A_n	x_n	B_{n+1}	A_{n+1}	Y_n		
0	0	0	0	0	0	0 X	0 X
0	0	1	0	1	0	0 X	1 X
0	1	0	0	0	0	0 X	X 1
0	1	1	1	0	0	1 X	X 1
1	0	0	1	1	0	X 0	1 X
1	0	1	1	0	0	X 0	0 X
1	1	0	0	0	1	X 1	X 1
1	1	1	0	1	1	X 1	X 0

Mealy model

Present state		Present input	Next state		Present output				
B_n	A_n	x_n	B_{n+1}	A_{n+1}	y_n	J_B	K_B	J_A	K_A
0	0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	0	x	1	x
0	1	0	0	0	0	0	x	x	1
0	1	1	1	0	0	1	x	x	1
1	0	0	0	0	1	x	1	0	x
1	0	1	1	0	0	x	0	0	x

11.4 Design equations & circuit diagram.

moore model :–

→ output is generated by AND operation on two flip flop outputs & does not use x

mealy model :

→ output directly uses input information.

moore model

$J_B = x A_n$

$J_A = \bar{x} B_n + x \bar{B}_n$

$K_B = A_n$

$K_A = \bar{x} + \bar{B}_n$

$Y = A_n B_n$

mealy model

$J_B = x A_n$

$K_B = \bar{x}$

$J_A = x \bar{B}_n$

$K_A = 1$

$Y = \bar{x} B_n$

map & cht diagram – 11.4 (page 397), 11.5 (page-398)

and output are obtained in similar way. Note that, output is generated by AND operation on two flip-flop outputs and does not use X.

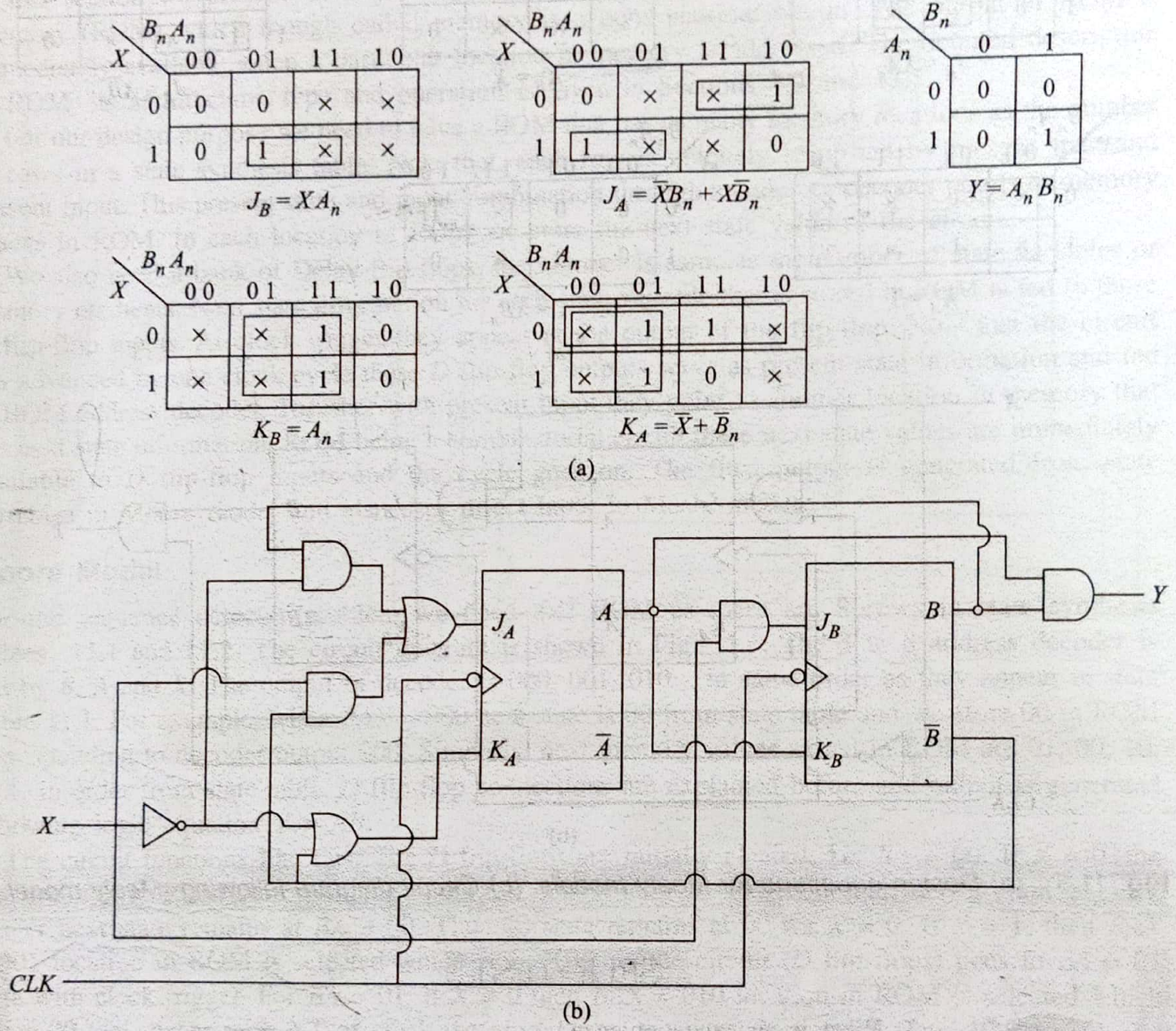


Fig. 11.4 *(a) Design equations for Moore model. (b) Circuit diagram following Moore model.*

Mealy Model

Using state synthesis table corresponding to Mealy model (Table 11.2) we can fill six positions in each Karnaugh map (Fig. 11.5a). Locations $B_nA_nX = 110$ and $B_nA_nX = 111$ are filled with don't care(×) conditions as such a combination never occur in the detector circuit if properly initialized. The design equations are obtained from these Karnaugh maps from which circuit diagram is drawn as shown in Fig. 11.5b. Note that in this circuit, output directly uses input information.

11.5 Implementation using Rom:

→ Rom → memory → combinational circuit.

→ output of Rom is immediately available when a particular location in memory is addressed.

→ For our design purpose we need to have a Rom that has as many memory locations as the number of rows in a state synthesis table

→ Each row is uniquely identified by Present state and present input.

→ This present state and input combination through an address decoder points to memory spaces in Rom.

→ Information is stored in flip flops

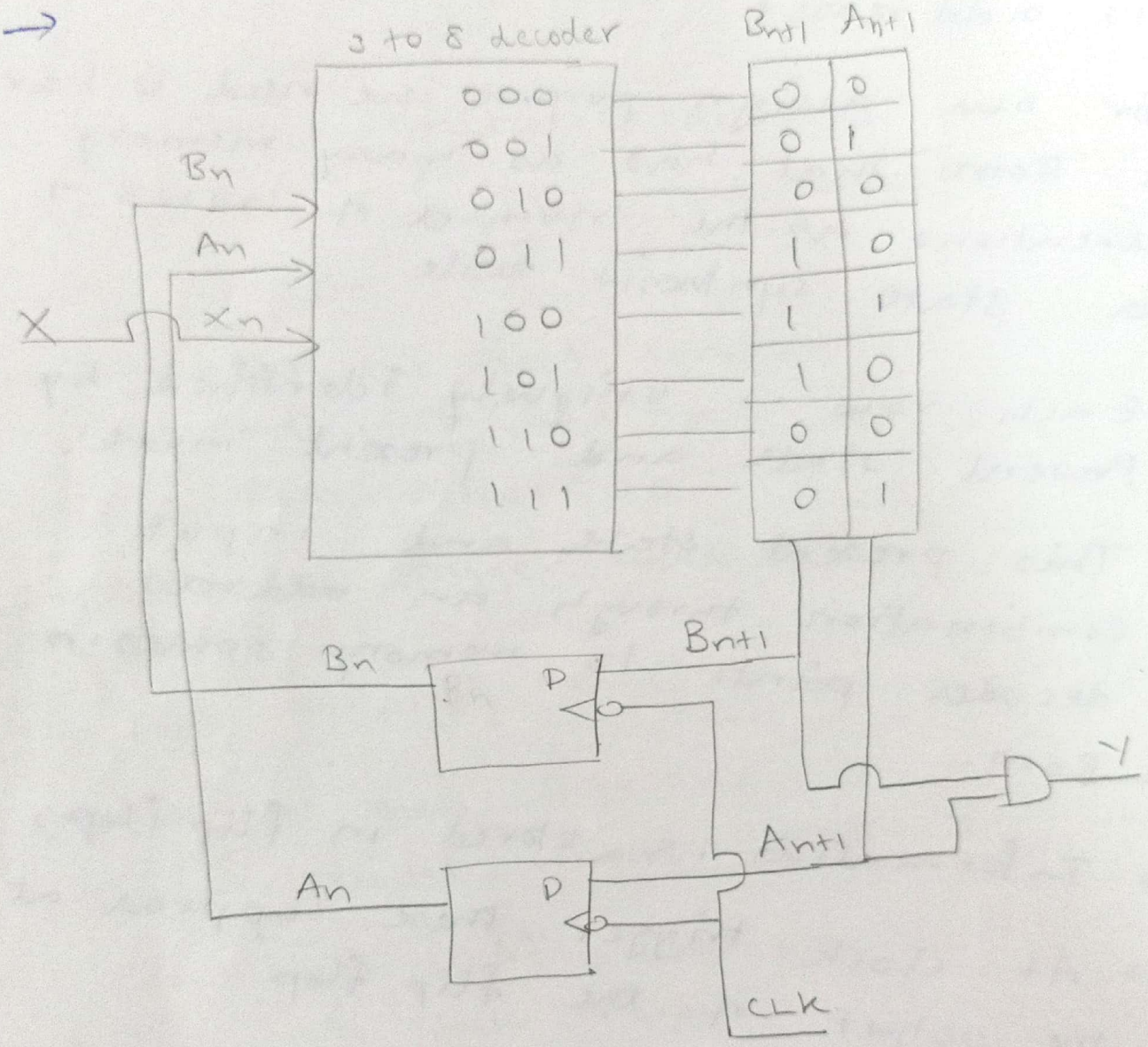
→ At clock trigger these appear at the output of the flip flop.

→ The final output is generated from state variables in moore model and also uses direct input in mealy model.

Moore model:

→ For the sequence detector problem we need 8×2 ROM as there are 8 rows in the state synthesis table.

→



→ The 3 to 8 decoder (addr decoder) is fed by B, A, and x.

→ The output of the decoder is 000, 001 ... in same order as they appear in state table.

→ For ex when BAx = 000 next state is 00 from state table.

Mealy model :

→ Rom based solution of mealy model uses state synthesis table in the same way as moore model.

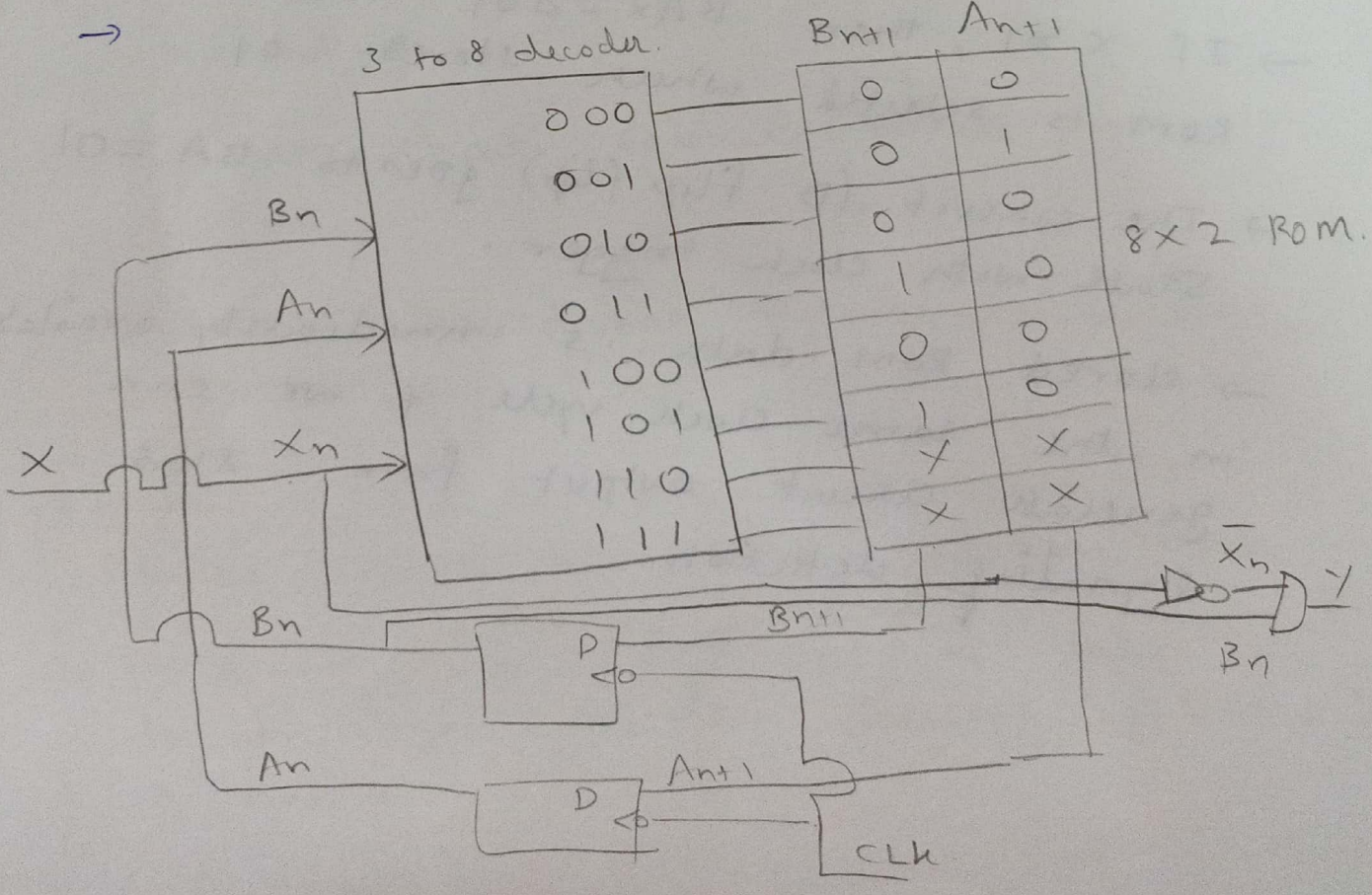
→ Rom locations are selected by present state and input as appears in state table and next state value fills corresponding Rom locations.

→ Delay flip flops banks are used in the same way but final output is generated from D flip flop outputs.

→ In moore model we have used Rom outputs directly to generate sequence detector output.

→ Rom size = 6 × 2

→



11.6 Algorithmic State machine :

→ Algorithmic state machine (Asm) is a flow chart like representation (Asm chart) of the algorithm a state machine performs.

→ For relatively more complex diagrams though more compact in representation ~~has~~ state transition diagram has certain disadvantages.

→ For relatively more complex problem where number of inputs and states are higher the state diagram space becomes so crowded that it is difficult to read

→ The other advantage of Asm chart is that, it handles implementation issues with more ease offering better timing information.

→ In Asm chart, square boxes represents a state

→ Diamond shaped box represents decision to be taken

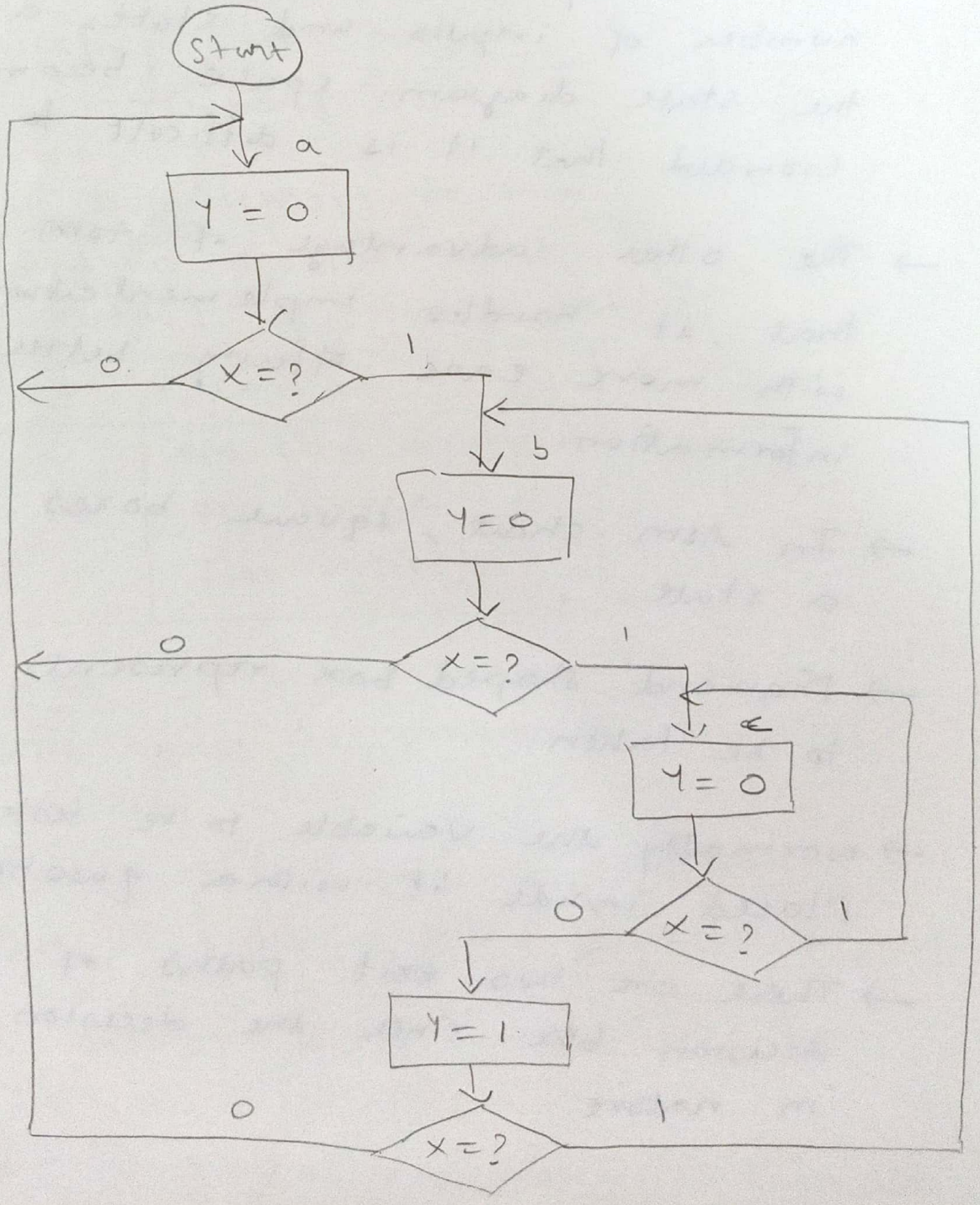
→ Normally the variable to be tested is placed inside it with a question mark

→ There are two exit paths of this decision box since the decision is binary in nature

→ For mealy model oval shaped boxes are used to describe the output that depends on present state as well as present input.

→ Circles are used to denote start, stop of the algorithm

→ Asm chart for sequence detector problem is shown below (moore model)



Vending Machine Problem:

→ Task: Design a synchronous logic control unit of a vending machine.

→ machine can take only two types of coins of denomination 1 & 2, in any order.

→ It delivers only one product that is priced Rs. 3.

→ On receiving Rs 3 the product is delivered by asserting an output $x = 1$ which otherwise remains 0.

→ There are two sensors to sense the denomination of the coins that give binary output as shown below.

I	I	Coin
0	x	No coin dropped
1	0	one rupee
1	1	Two rupees

ASM chart:

→ The ASM chart is prepared following mealy model.

→ The initial state when no coin is deposited is designated as state a.

→ sensor output I=0 indicates no coin is deposited. (X=0, Y=0)

→ X → no product is delivered, Y → coin returned.

→ If I = 1, controller tests J.

→ If J = 0 it goes to state b that indicates represents Re 1 is received.

→ If J = 1, it goes to state c indicating RS 2 is received.

→ If I = 1 & J = 0 the machine has received two Re 1 coins in succession and should move to state c.

→ I = 1, J = 1 means a RS 2 coin is received, following Re 1. total of RS-3.

→ Hence the product is delivered X = 1 and circuit goes to initial state.

→ At state c if on testing I = 1 that is a coin is deposited, the controller tests J

→ If J = 0 Re 1 is deposited and a total of RS 3 is received.

→ The product is delivered by X = 1 and circuit goes to initial state a.

→ If J = 1 then RS 2 is received totaling RS-4.

→ Then Re 1 is returned by asserting Y = 1, also the product delivered X = 1. and controller moves to initial state a.

State assignment & state synthesis table

→ The state assignment is same as with the previous example.

→ With this example a D flip flop has been used.

→ State assignment :

BA → value	state
00	a
01	b
10	c
11	not used

→ State synthesis table :

Present state		Input		Next state		output		D_B	D_A
B_n	A_n	I	J	B_{n+1}	A_{n+1}	X	Y		
0	0	0	0	0	0	0	0	0	0
		0	1	0	0	0	0	0	0
		1	0	0	1	0	0	0	1
		1	1	1	0	0	0	1	0
0	1	0	0	0	1	0	0	0	1
		0	1	0	1	0	0	0	1
		1	0	1	0	0	0	1	0
		1	1	0	0	1	0	0	0
1	0	0	0	1	0	0	0	1	0
		0	1	1	0	0	0	1	0
		1	0	0	0	1	0	0	0
		1	1	0	0	1	1	0	0

Design equations from k-map & cht diagram :-

$$D_B = \bar{I} B_n + I \bar{J} A_n + I J \bar{B}_n A_n$$

$$D_A = \bar{I} A_n + I \bar{J} \bar{B}_n \bar{A}_n$$

$$X = I B_n + I J A_n$$

$$Y = I J B_n.$$

→ circuit diagram & Rom implementation are as shown in xerox

→ Example 11.3 (page 407)

11.7 State reduction Technique :-

→ In the design of sequential logic circuit state reduction techniques play an important role, more so for complex problems.

→ 1) Row elimination method
2) Implication table method

Row Elimination method :-

→ we first prepare a state table where at any given state the next state and present output are written for each combination of input.

→ Refer text for problems on state reduction.

→ Refer text for asynchronous x not in syllabus

IJ \ B_nA_n	00	01	11	10
00	0	0	×	1
01	0	0	×	1
11	1	0	×	0
10	0	1	×	0

$D_B = \bar{I}B_n + I\bar{J}A_n + IJ\bar{B}_n\bar{A}_n$

IJ \ B_nA_n	00	01	11	10
00	0	1	×	0
01	0	1	×	0
11	0	0	×	0
10	1	0	×	0

$D_A = \bar{I}A_n + I\bar{J}\bar{B}_n\bar{A}_n$

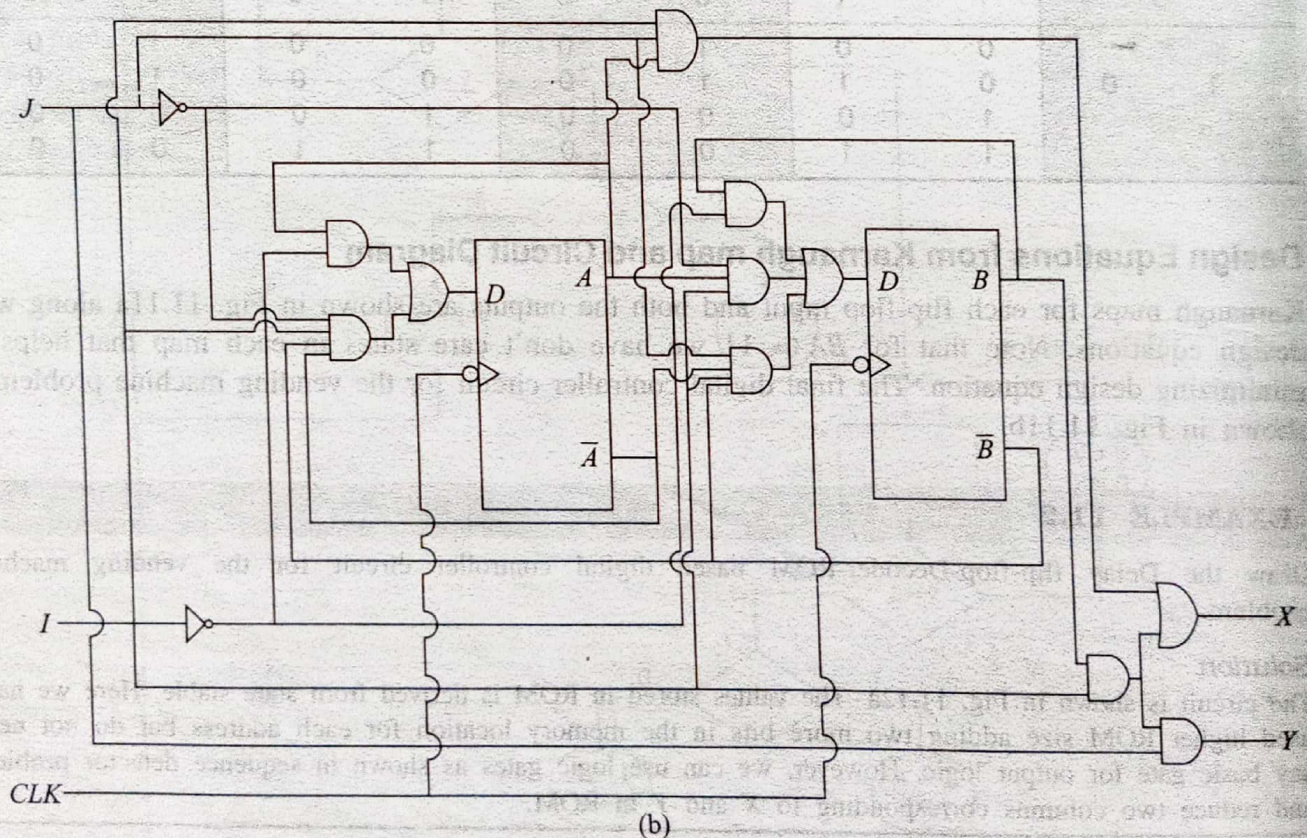
IJ \ B_nA_n	00	01	11	10
00	0	0	×	0
01	0	0	×	0
11	0	1	×	1
10	0	0	×	1

$X = IB_n + IJA_n$

IJ \ B_nA_n	00	01	11	10
00	0	0	×	0
01	0	0	×	0
11	0	0	×	1
10	0	0	×	0

$Y = IJB_n$

(a)



(b)

Fig. 11.11 *(a) Design equation. (b) Circuit diagram for vending machine problem: Mealy model.*

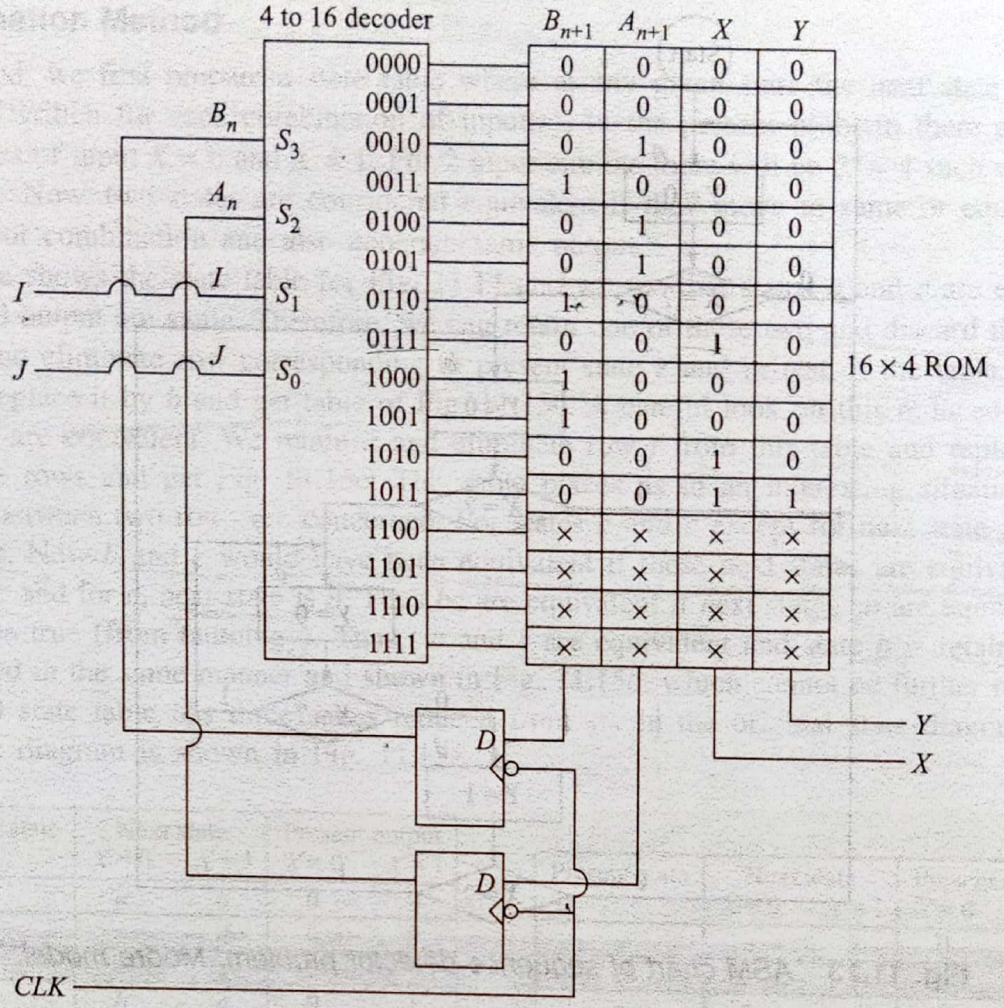


Fig. 11.12a *ROM implementation of vending machine problem.*

EXAMPLE 11.3

Draw state transition diagram of the Mealy model vending machine problem.

Solution

The diagram can easily be drawn from ASM chart (Fig. 11.10) and is shown in Fig. 11.12b.

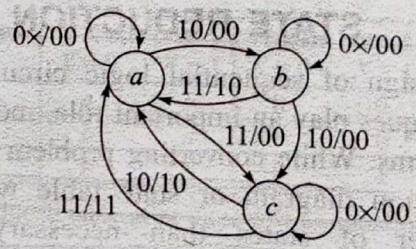


Fig. 11.12b *State transition diagram of vending machine problem.*

EXAMPLE 11.4

Draw ASM chart of the sequence detector problem described in Section 11.2 following Moore model.

Solution

In Fig. 11.13 we show the ASM chart of the sequence detector problem described in Section 11.2 following Moore model where X denotes the input data bit and Y the detector output.

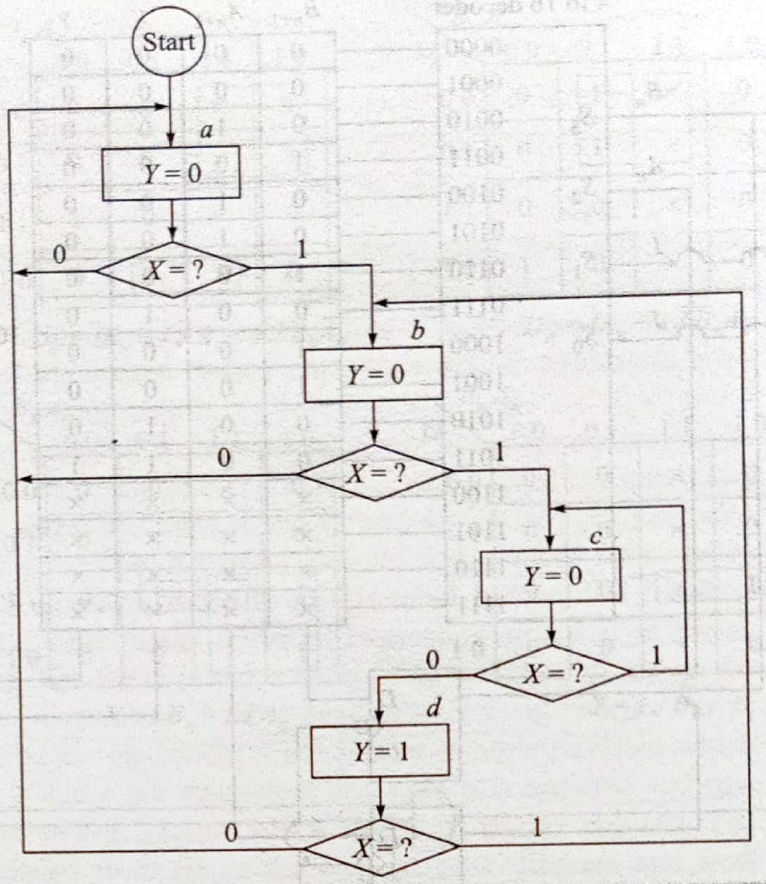


Fig. 11.13 *ASM chart of sequence detector problem: Moore model.*

Note the similarity between Moore model state transition diagram of Fig. 11.2a and ASM chart shown here. Once we arrive at the ASM Chart the rest of the design procedure starting from state assignment up to final circuit diagram is same as what is discussed in Section 11.6. ASM Chart for the Mealy model sequence detector is left as exercise for the reader.

11.7 STATE REDUCTION TECHNIQUE

In design of sequential logic circuit state reduction techniques play an important role, more so for complex problems. While converting problem statement to state transition diagram or state table we may use more number of states than necessary. On removing redundant states the clarity of the problem is enhanced. This also offers simpler solution and less hardware to implement a circuit. We explain two state reduction techniques through an example.

Let the state transition diagram drawn following a Mealy model is as shown in Fig. 11.14. The goal is to identify and remove redundant states, if any and obtain the reduced state diagram.

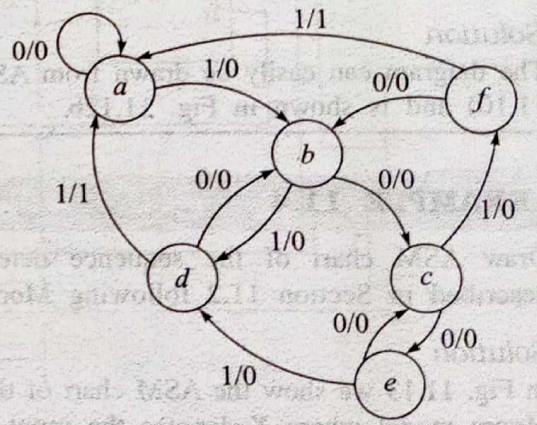


Fig. 11.14 *A state transition diagram.*

Row Elimination Method

In this method, we first prepare a state table where at any given state the next state and present output(s) are written for each combination of input(s). In the present problem there are only two possible values of input $X = 0$ and $X = 1$. For 2 input circuits there will be $2^2 = 4$ such combinations in this table. Now, two states are considered equivalent if they move to same or equivalent state for every input combination and also generate same output.

Fig. 11.15a shows the state table for Fig. 11.14 and we see that states b and e are equivalent as next state and output are same. Therefore, we can retain one of these two and discard the other. Let us retain b and eliminate row corresponding to present state e and in rest of the table, wherever e appears we replace it by b and get table of Fig. 11.15b. A careful look on this reduced table shows state d and f are equivalent. We retain d and eliminate row f from this table and replace f with d in rest of the rows and get Fig. 11.15c. This table places us in an interesting situation as far as equivalence between two rows are concerned. For states b and c except for next state at $X = 0$ the rest are same. Now b and c would have been equivalent if these next states are equivalent. For b, next state is c and for c, next state is b. Thus bc are equivalent if next states cb are equivalent which can always be true (from tautology). Thus, b and c are equivalent and state b is retained and row c is eliminated in the same manner and shown in Fig. 11.15d, which cannot be further reduced. The final reduced state table has three states reduced from six in the original state diagram and final reduced state diagram is shown in Fig. 11.15e.

Present state	Next state $X=0$	Next state $X=1$	Present output $X=0$	Present output $X=1$
a	a	b	0	0
$\sqrt{b}$	c	d	0	0
c	e	f	0	0
d	b	a	0	1
$\sqrt{e}$	c	d	0	0
f	b	a	0	1

(a) Original table

Present state	Next state $X=0$	Next state $X=1$	Present output $X=0$	Present output $X=1$
a	a	b	0	0
b	c	d	0	0
c	b	f	0	0
$\sqrt{d}$	b	a	0	1
$\sqrt{f}$	b	a	0	1

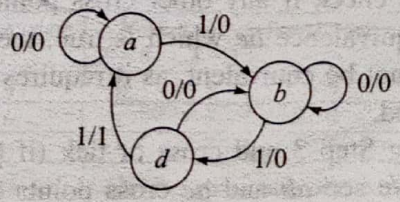
(b) After one row elimination

Present state	Next state $X=0$	Next state $X=1$	Present output $X=0$	Present output $X=1$
a	a	b	0	0
$\sqrt{b}$	c	d	0	0
$\sqrt{c}$	b	d	0	0
d	b	a	0	1

(c) After two row elimination

Present state	Next state $X=0$	Next state $X=1$	Present output $X=0$	Present output $X=1$
a	a	b	0	0
b	b	d	0	0
d	b	a	0	1

(d) Final reduced table after three row elimination

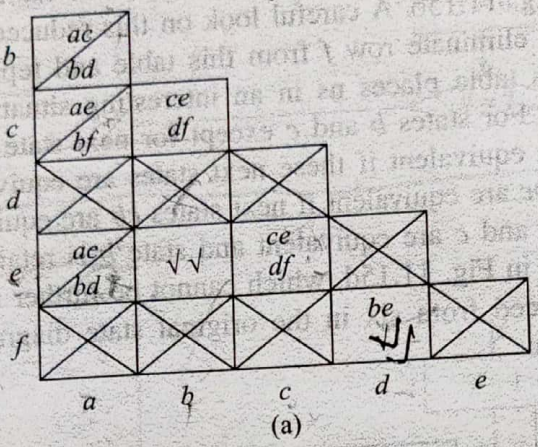


(e)

Fig. 11.15 *(a)–(d) Tables showing row elimination steps. (e) Reduced state diagram.*

Implication Table Method

Implication table provides a more systematic approach towards solution of a complex state reduction problem. For n states in the initial description we have $n-1$ rows in implication table and as many number of columns. Refer to implication table of Fig. 11.16a for the given state reduction problem. The cross-point in an implication table is the location where a row and a column meet. Here, the conditions for equivalence between the states crossing each other are tested. We use state table of Fig. 11.15a derived from state transition diagram to fill up implication table. The steps to be followed are given next.



(a)

$e : e$
$d : e\ (df)$
$c : e\ (df)\ (ce) \equiv (df)\ (ce)$
$b : (df)\ (ce)\ (bc) \equiv (df)\ (bce)$
$a : a\ (df)\ (bce)$
$P = (df)\ (bce)\ (a)$

(b)

Fig. 11.16 *Implication table method of state reduction. (a) Implication table. (b) Partition table.*

In **Step 1**, we identify the states, which cannot be equivalent, as their outputs do not match. This we denote by putting a double-cross in respective cross points. In this problem state d and f only have output = 1 for $X = 1$ unlike other states. Thus, intersection of d and f with others except themselves are double crossed.

In **Step 2**, for other cross points, we write necessary conditions for equivalence of intersecting states. As an example, let us look at intersection of states a and b. To get the necessary condition we refer to rows starting with a and b in state table of Fig. 11.15a. We find that at $X = 0$, a stays at a while b goes to c and at $X = 1$, a goes to b while b goes to d. Thus, a and b can only be equivalent if next states a and c are equivalent and also if b and d are equivalent. This is written at cross point of a and b in implication table. Note that output of a and b match, else, it would have got a double cross in Step 1. We similarly fill up other cross points and note that b and e are equivalent and does not require any equivalence between other states and a double tick mark is placed at that cross point.

In **Step 3**, we use relationships obtained in Steps 1 and 2, specially the ones represented by double cross and double tick mark and check if any other cross points can be crossed or ticked. Since df equivalence depends only on equivalence be which is true, they are equivalent and that cross point can be ticked. Similarly, ac cannot be equivalent, as it requires bf to be equivalent which is not true. Hence, ac intersection is crossed.

In **Step 4**, we keep repeating Step 3 and cross or tick (if possible) as many cross points in the implication table as possible. We see ab and ae cross points can be crossed as they need ac to be equivalent which is crossed in the previous step. With no further crossing and ticking possible the implication table is fully prepared and we go to Step 5.

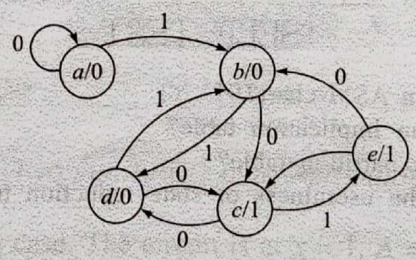
In **Step 5**, we check pairwise equivalence starting from rightmost column *e* of implication table. Since, the only cross point, representing *ef* equivalence along column *e* is crossed there is no equivalence possible at column *e* and we write *e* in Fig. 11.16b in the first place. In column *d*, we find *df* are equivalent and along *d* in Fig. 11.16b the same is written. In column *c*, we find *ce* are equivalent as *df* is equivalent. In column *b*, equivalence between *be* and *bc* can be observed as *ce* and *df* are already considered equivalent and the same are written along *c* and *b*. Note that if *p* and *q* are equivalent and so are *q* and *r* then *p* and *q* are equivalent. In that case, *pqr* can form one group and any one of its members can represent the group. Since, column *a* does not give any equivalent pair the final partition table is represented as P = (*df*) (*bce*) (*a*) and has three partitions. Then three states are sufficient to solve this problem each representing one partition. If *d* represents any of *df* and *b* represents *bce* in Fig. 11.16b we get reduced state table as shown in Fig. 11.15d and corresponding reduced state diagram is shown in Fig. 11.15e.

Note that the final result is same by both the state reduction method. However, in row elimination method one has to draw many tables for a complex state reduction problem and depend a lot on observation power. The implication method being more systematic is more conclusive.

We shall discuss state reduction technique for *incompletely specified state table* in connection with asynchronous sequential circuit design in Section 11.10. In such problems some of the next states or output remains unspecified and treated as don't care condition.

EXAMPLE 11.5

Reduce state transition diagram (Moore Model) of Fig. 11.17a by (i) row elimination method and (ii) implication table method



(a)

Present state	Next state $X=0$	Next state $X=1$	Present output
a	a	b	0
$\sqrt{b}$	c	d	0
c	d	e	1
$\sqrt{d}$	c	b	0
e	b	c	1

(b) Original table

Present state	Next state $X=0$	Next state $X=1$	Present output
a	a	b	0
b	c	b	0
$\sqrt{c}$	b	e	1
$\sqrt{e}$	b	c	1

(c) Table after elimination of one row

Present state	Next state $X=0$	Next state $X=1$	Present output
a	a	b	0
b	c	b	0
c	b	c	1

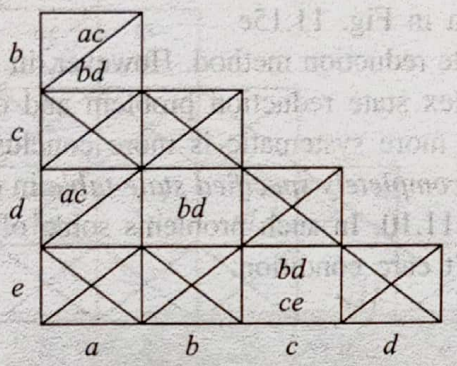
(d) Final reduced table after elimination of two rows

Fig. 11.17 *Reduction by row elimination method.*

Solution

(i) Refer to state table of Fig. 11.17b obtained from state transition diagram. Comparing row *b* and *d* we see they are equivalent because that needs no other consideration except equivalence between themselves. Retaining *b* and replacing *d* by *b* in rest of the table we get Table of Fig. 11.17c. There we find *c* and *e* are equivalent and we retain *c* and replacing *e* by *c* get Fig. 11.17d. We see no further reduction is possible and final reduced state table that has three states.

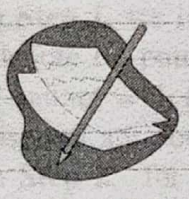
(ii) Refer to state transition diagram and state table developed from it. Implication table is shown in Fig. 11.18. The non-compliance of output makes cross-points *de, be, ae, cd, bc, ac* non-equivalent and hence double crossed. From this we find *ab* and *ad* cannot be equivalent as that requires *ac* to be equivalent which is not true. Finally moving columnwise starting from *d* we get partition table and final partition P has three groups. Hence, the number of states is reduced to 3 from 5 by this technique.



$d : d$
$c : d\,(ce)$
$b : d\,(ce)\,(bd) \equiv (ce)\,(bd)$
$a : (ce)\,(bd)\,a$
$P = (ce)\,(bd)\,(a)$

Fig. 11.18 *Reduction by implication table method.*

SELF-TEST

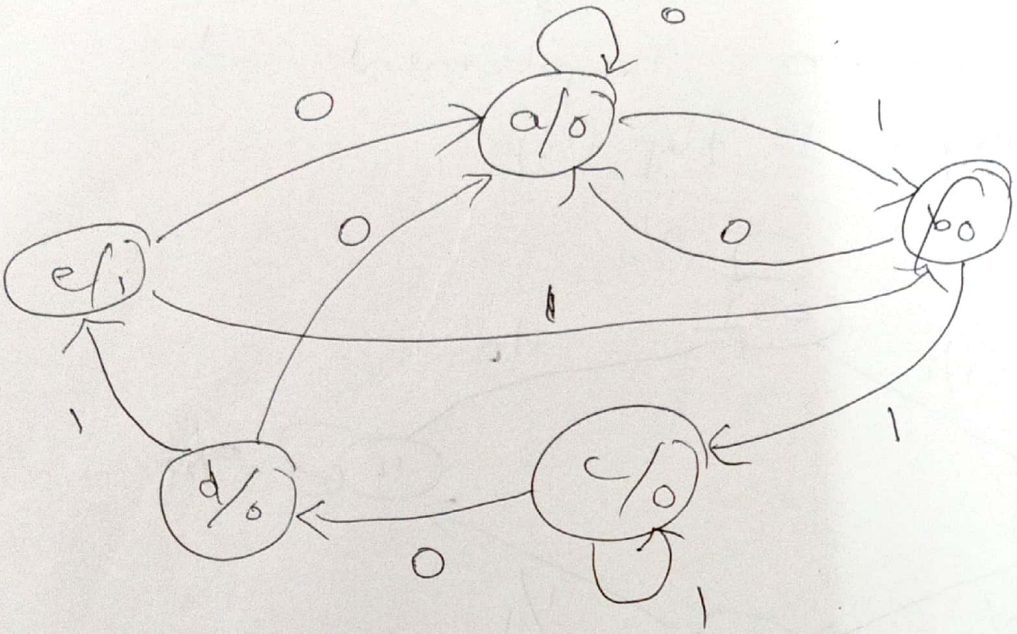


8. What is an ASM chart?
9. What is an implication table?
10. What is a partition table?
11. What is the usefulness of state reduction technique?

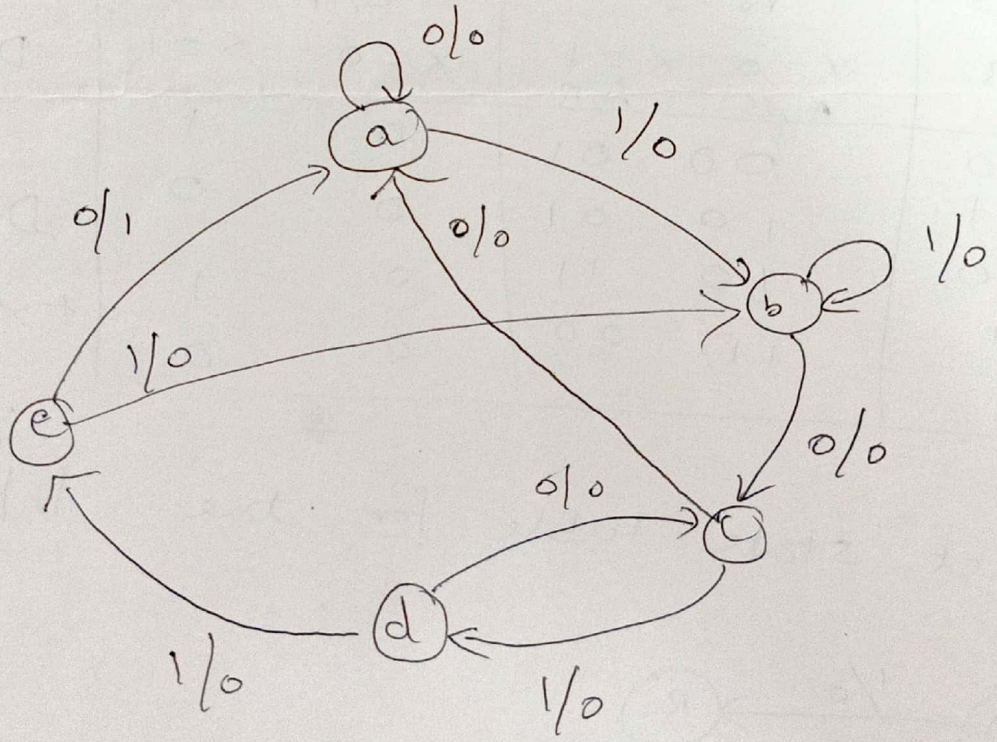
PART B: ASYNCHRONOUS SEQUENTIAL CIRCUIT

Asynchronous Sequential Circuit, also called Event Driven Circuit does not have any clock to trigger change of state. State changes are triggered by change in input signal. In ciock driven circuit all the memory elements change their states together. In spite of all the advantages it offers, there are certain limitations with such circuit. The most important being the speed of operation. This is limited by the clock frequency since, state change can only take place at time t = *n*T, where T = Time period of clock signal and inverse of frequency and *n* is an integer. If the input changes in a manner that warrants change in the state, it cannot do that immediately and wait till the next clock trigger comes. Asynchronous sequential circuit is a solution to this however, design of such circuit is very complex and has several constraints to be taken care of, which is not required for synchronous circuit. Here, we shall discuss *fundamental mode* of operation of asynchronous sequential circuit where output change depends on change in input level. There is another type of such circuit called *pulse mode* where output change is affected by edge of the input pulse.

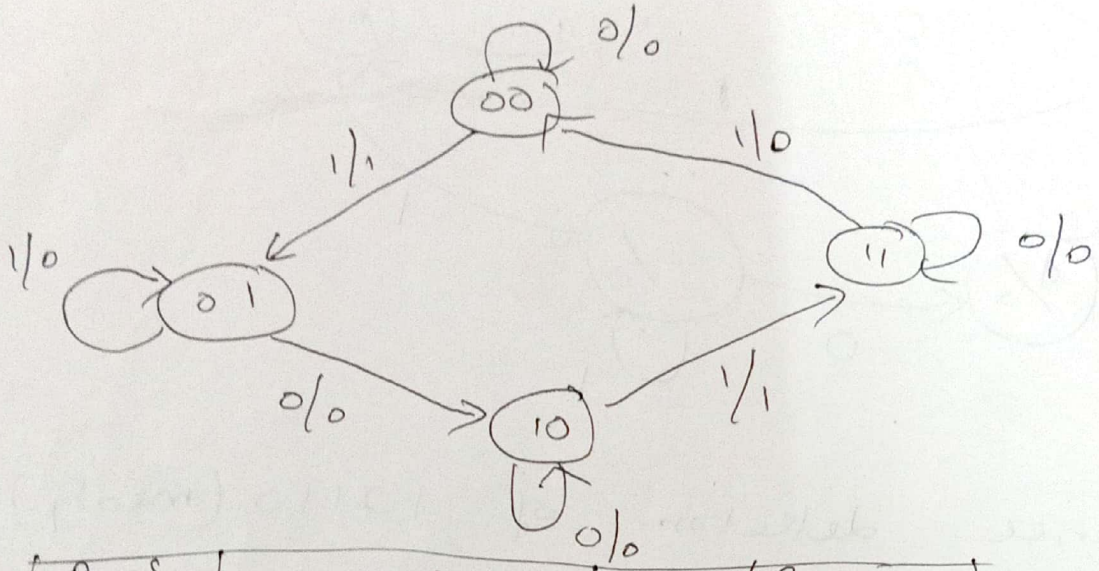
Sequence detector to detect sequence 1101 (moore)



Sequence detector of 10110 (mealy)



Design a clocked sequential circuit that operates according to the state diagram shown. Implement the circuit using D flip flop.



P.S	N.S		O/P	
A B	x=0 A,B	x=1 A,B	x=0	x=1
0 0	0 0	0 1	0	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

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

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

$Y = x\bar{B}$

Construct state table for the following