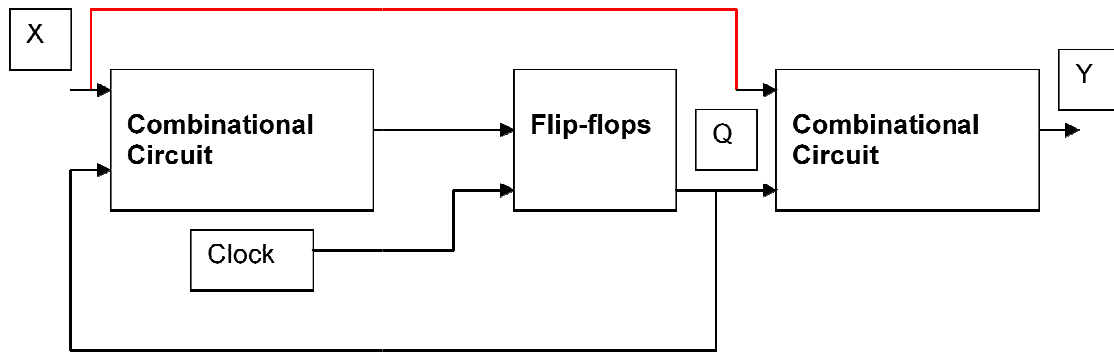


## Design of asynchronous sequential circuits

Digital electronics is classified into combinational logic and sequential logic. Combinational logic output depends on the inputs levels, whereas sequential logic output depends on stored levels and also the input levels.

The memory elements are devices capable of storing binary info. The binary info stored in the memory elements at any given time defines the state of the sequential circuit. The input and the present state of the memory element determines the output. Memory elements next state is also a function of external inputs and present state. A sequential circuit is specified by a time sequence of inputs, outputs, and internal states.



There are two types of sequential circuits. Their classification depends on the timing of their signals:

- Synchronous sequential circuits
- Asynchronous sequential circuits

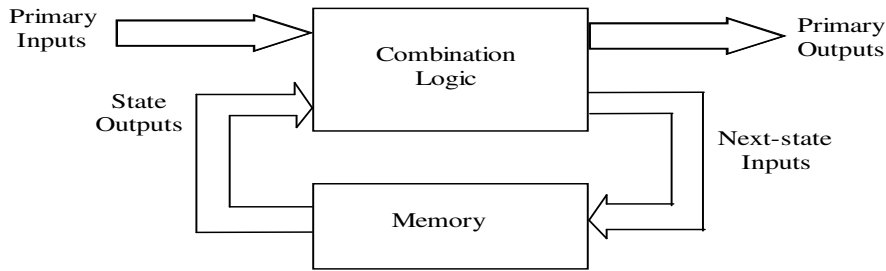
This is a system whose outputs depend upon the order in which its input variables change and can be affected at any instant of time. Gate-type asynchronous systems are basically combinational circuits with feedback paths. Because of the feedback among logic gates, the system may, at times, become unstable. Consequently they are not often used.

## MODEL SELECTION

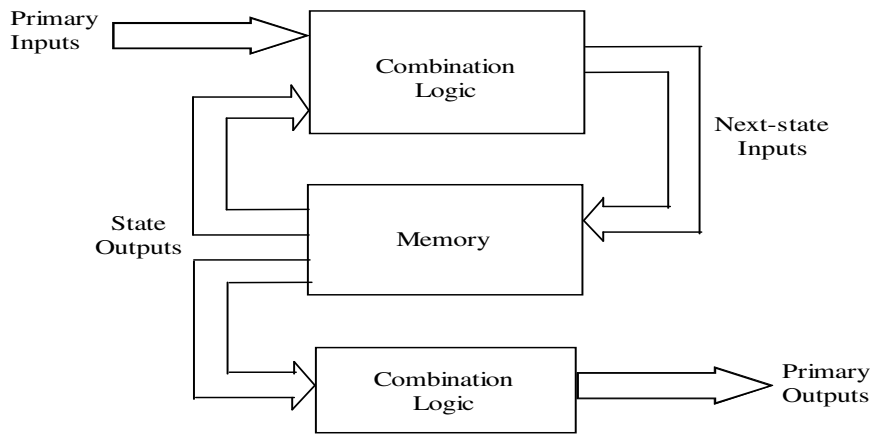
There are two distinct models by which a synchronous sequential logic circuit can be designed.

In *Mealy Model*, the output is derived from present state as well as input.

In *Moore Model* the output depends only on present state and not on input.



( a )



( b )

- a) Mealy model
- b) Moore model

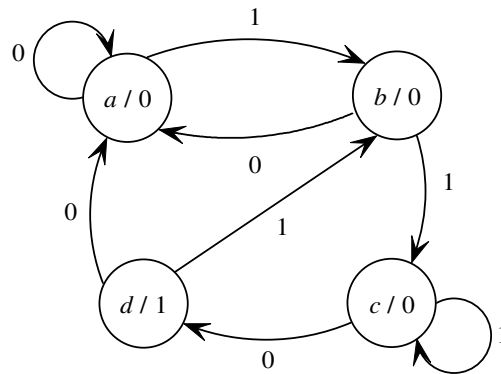
## STATE TRANSITION DIAGRAM

- **STATE DEFINITIONS: MOORE MODEL** Since, the output is generated only from the state variables let us see how many of them are necessary. Let the detector circuit be at state *a* when initialized. State *a* can also be considered as one where none of the bit in input sequence is properly detected or the starting point of detection. Then if 1st bit is detected

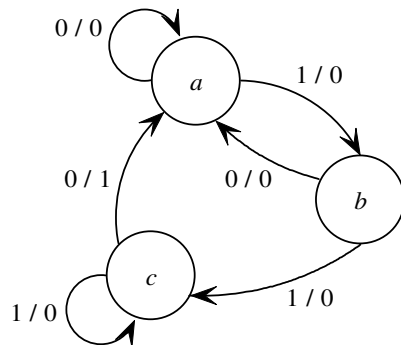
properly the circuit should be at a different state say,  $b$ . Similarly, we need two more states say,  $c$  and  $d$  to represent detection of 2nd and 3rd bit in proper order. When the detector circuit is at state  $d$ , output  $Y$  is asserted and kept high as long as circuit remains in state  $d$  signaling sequence detection. For other states detector output,  $Y=0$ .

- **STATE TRANSITION DIAGRAM: MOORE MODEL** In Moore Model each state and output is defined within a circle in state transition diagram in the format  $s/Y$  where  $s$  represents a symbol or memory values identified with a state and  $Y$  represents the output of the circuit. An arrow sign marks state transition following an input value 0 or 1 that is written along side

**State transition diagram of sequence detector: Moore model**



**State transition diagram of sequence detector : Mealy Model**



The next step in design process is to develop state synthesis table, also called circuit excitation table or simply state table from state transition diagram. Note that for  $m$  number of memory elements we can have up to  $2^m$  number of different states in a circuit. Once we decide how many memory elements are to be used we go for state assignment

### STATE ASSIGNMENT

Here, we allocate each state a binary combination of memory values. For the given problem, both Moore and Mealy model require minimum two flip-flops (say A and B) to define their states (4 for Moore and 3 for Mealy). 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

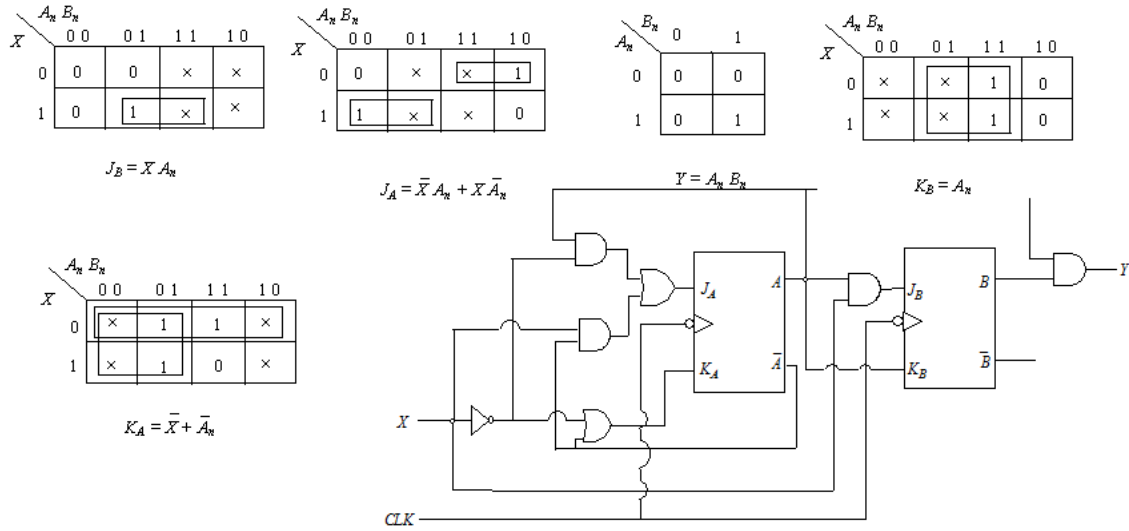
Note that Mealy Model does not use state d.

### STATE SYNTHESIS TABLE

The next design step is to decide what kind of memory elements are to be used for our design. Flip-flops are commonly used for this purpose. We normally choose *JK* flip-flop as it has maximum number of don't care states in its excitation table and that leads to simpler design equations

### State synthesis table for Moore Model

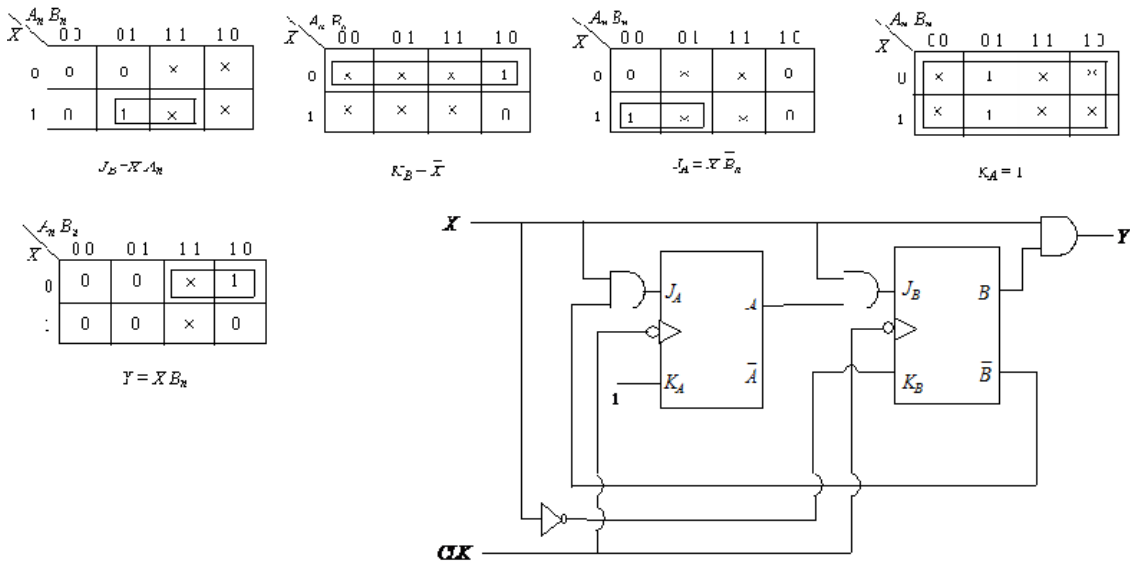
| Present State |       | Input | Next State |           | Output |       |       |       |       |
|---------------|-------|-------|------------|-----------|--------|-------|-------|-------|-------|
| $B_n$         | $A_n$ | $X_n$ | $B_{n+1}$  | $A_{n+1}$ | $Y_n$  | $J_B$ | $K_B$ | $J_A$ | $K_A$ |
| 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     | 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     |



State synthesis table for Mealy Model

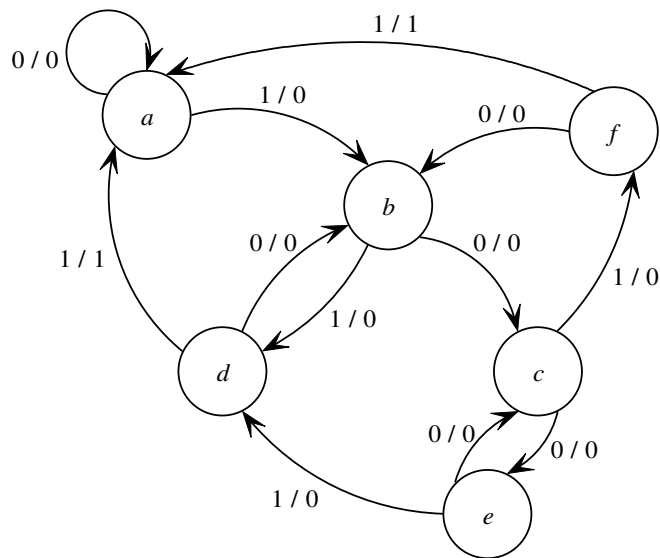
| Present State |       | Present Input<br>$X_n$ | Next State |           | Present Output<br>$Y_n$ | $J_B$ $K_B$ |   | $J_A$ $K_A$ |  |
|---------------|-------|------------------------|------------|-----------|-------------------------|-------------|---|-------------|--|
| $B_n$         | $A_n$ |                        | $B_{n+1}$  | $A_{n+1}$ |                         |             |   |             |  |
| 0             | 0     | 0                      | 0          | 0         | 0                       | ×           | 0 | ×           |  |
| 0             | 0     | 1                      | 0          | 1         | 0                       | ×           | 1 | ×           |  |
| 0             | 1     | 0                      | 0          | 0         | 0                       | ×           | × | 1           |  |
| 0             | 1     | 1                      | 1          | 0         | 0                       | 1           | × | 1           |  |
| 1             | 0     | 0                      | 0          | 0         | 1                       | ×           | 0 | ×           |  |
| 1             | 0     | 1                      | 1          | 0         | 0                       | ×           | 0 | ×           |  |

## MEALY MODEL



## State Reduction Technique

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



| Present State | Next State |     | Present Output |     |
|---------------|------------|-----|----------------|-----|
|               | X=0        | X=1 | X=0            | X=1 |
| a             | a          | b   | 0              | 0   |
| ↯ b           | c          | d   | 0              | 0   |
| c             | e          | f   | 0              | 0   |
| d             | b          | a   | 0              | 1   |
| ↯ e           | c          | d   | 0              | 0   |
| f             | b          | a   | 0              | 1   |

Original table

| Present State | Next State |     | Present Output |     |
|---------------|------------|-----|----------------|-----|
|               | X=0        | X=1 | X=0            | X=1 |
| a             | a          | b   | 0              | 0   |
| b             | c          | d   | 0              | 0   |
| c             | b          | f   | 0              | 0   |
| ↯ d           | b          | a   | 0              | 1   |
| ↯ f           | b          | a   | 0              | 1   |

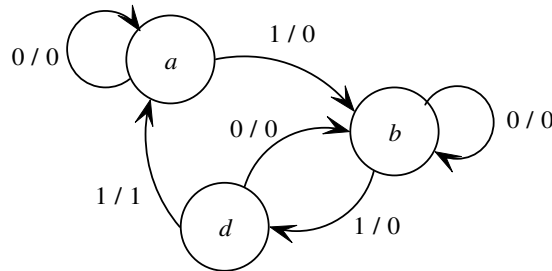
After one row elimination

| Present State | Next State |     | Present Output |     |
|---------------|------------|-----|----------------|-----|
|               | X=0        | X=1 | X=0            | X=1 |
| a             | a          | b   | 0              | 0   |
| ↯ b           | c          | d   | 0              | 0   |
| ↯ c           | b          | d   | 0              | 0   |
| d             | b          | a   | 0              | 1   |

After two row elimination

| Present Stat | Next State |     | Present Output |     |
|--------------|------------|-----|----------------|-----|
|              | X=0        | X=1 | X=0            | X=1 |
| a            | a          | b   | 0              | 0   |
| b            | b          | d   | 0              | 0   |
| d            | b          | a   | 0              | 1   |

Final reduced table after three row elimination



### IMPLICATION TABLE METHOD

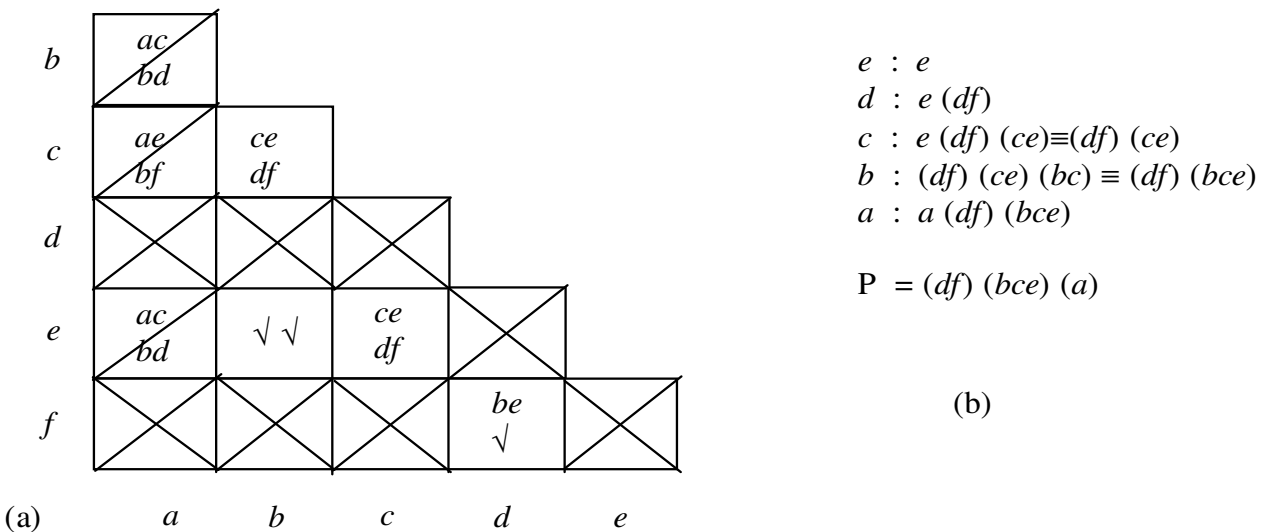
In **step 1**, we identify the states, which cannot be equivalent, as their outputs do not match. This we denote by putting a double-cross in respective cross points. In this problem state d and f only have output=1 for X=1 unlike other states. Thus, intersection of d and f with others except themselves 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. 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 get 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 step 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  $bf$  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 pair wise equivalence starting from rightmost column  $e$  of implication table.

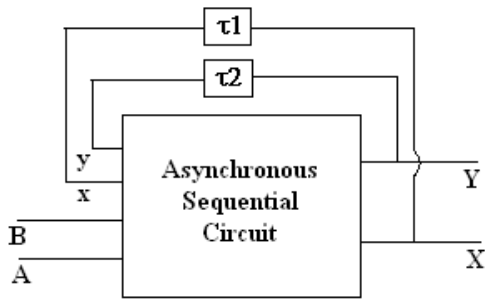


### PROBLEMS WITH ASYNCHRONOUS SEQUENTIAL CIRCUITS

Asynchronous circuit responds to all the transient values and problems like **oscillation, critical race, hazards** can cause major problem unless they are addressed at design stage. To explain these problems we take help of Truth Table shown in Fig where the circuit has two external inputs  $A, B$  and two outputs  $X, Y$ . Both the outputs are fed back to the input side in the form of  $x$  and  $y$  but with different propagation delays. Thus  $x, y$  cannot change simultaneously



but with time delays  $\tau_1$  and  $\tau_2$  respectively and we can write  $x=X(t-\tau_1)$  and  $y=(Y-\tau_2)$ .



(a) Block diagram

| AB \ xy | 00 | 01 | 11 | 10 |
|---------|----|----|----|----|
| 00      | 00 | 11 | 00 | 01 |
| 01      | 00 | 01 | 00 | 00 |
| 11      | 11 | 11 | 00 | 11 |
| 10      | 10 | 11 | 00 | 10 |

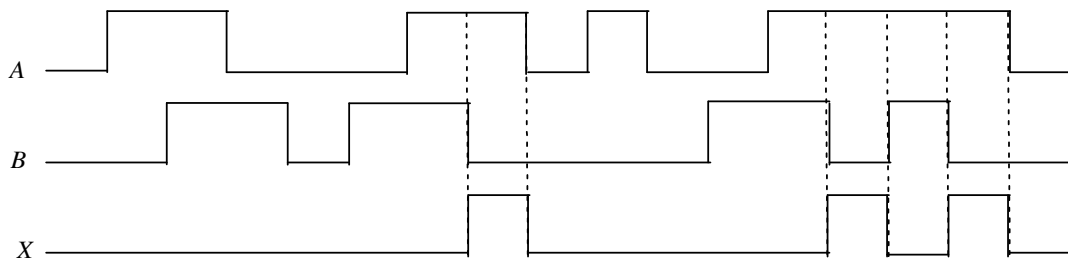
Oscillation

Critical Race    Noncritical Race

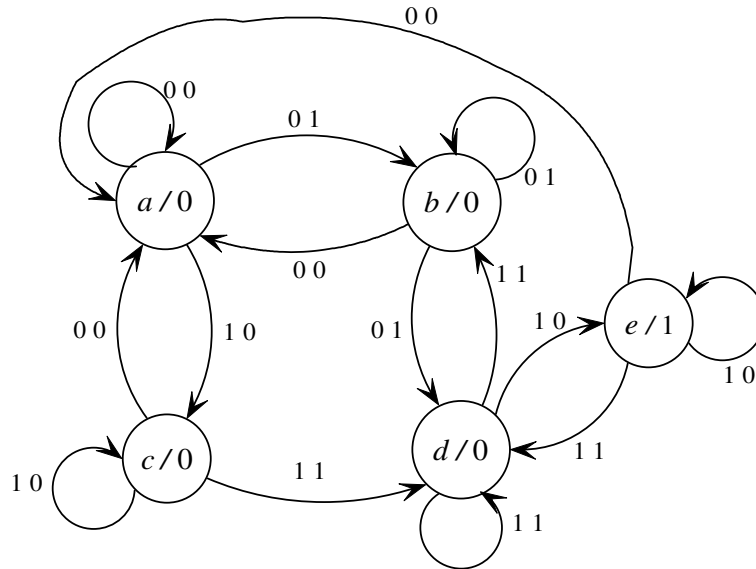
(b) Truth Table

- THE PROBLEM

A digital logic circuit is to be designed that has two inputs A, B and one output X. X goes high if at A=1, B makes a transition 1→0. X remains high as long as this A=1, B=0 are maintained. If any of A or B changes at this time output X goes low. It becomes high again when at A=1, B goes from 1 to 0. The timing diagram corresponding to this problem is shown in Figure



- From the problem statement we first develop a state transition diagram, say using Moore model. The state symbol and output at that state is shown together within a circle in this diagram Figure.



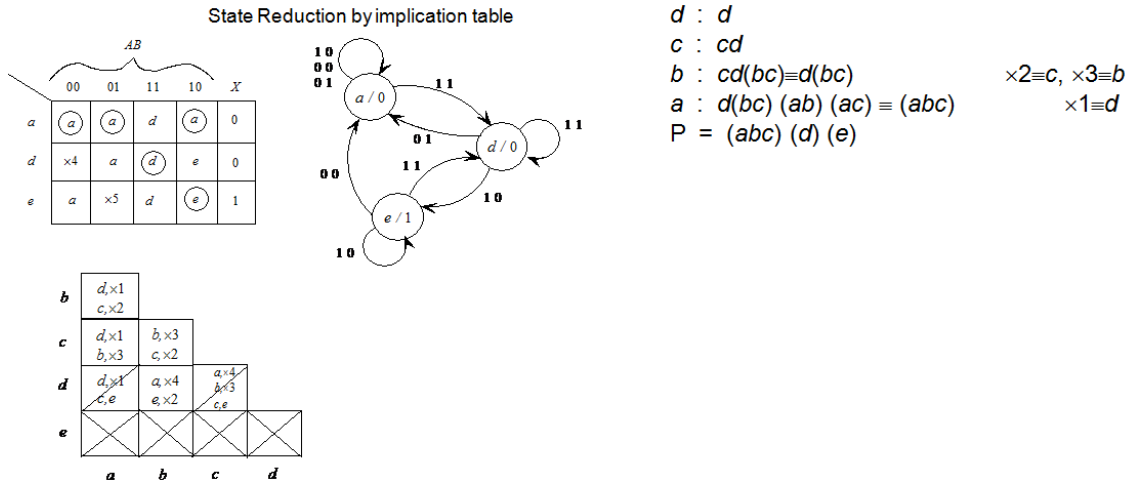
- **PRIMITIVE TABLE**

The next step is to form state table from state transition diagram. In this table if all the rows representing a state has only one stable state for all possible input combinations it is termed as *primitive table*, also called *primitive flow table* or simply *flow table*.

|   | AB  |     |     |     |   |
|---|-----|-----|-----|-----|---|
|   | 00  | 01  | 11  | 10  | X |
| a | (a) | b   | ×1  | c   | 0 |
| b | a   | (b) | d   | ×2  | 0 |
| c | a   | ×3  | d   | (c) | 0 |
| d | ×4  | b   | (d) | e   | 0 |
| e | a   | ×5  | d   | (e) | 1 |

- STATE REDUCTION

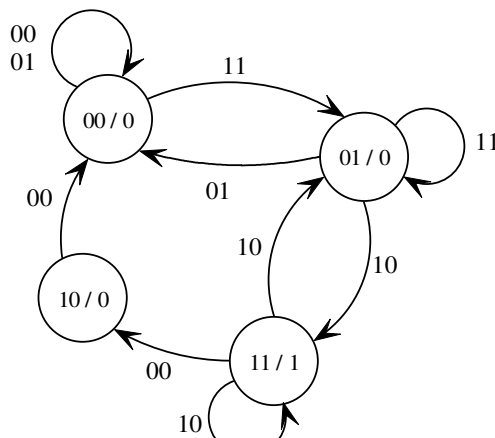
It is always useful to check state redundancy before going for actual circuit design. Removing redundant states helps in generating the circuit in a simpler way and with less hardware.



- STATE ASSIGNMENT

This step in asynchronous sequential circuit design has to be done very carefully so that a valid state transition does not require two or more output variables to change simultaneously which may lead to racing problem. In this problem there are three states in the reduced state diagram which needs two variable to represent them. Let us represent the states in this problem by two variables  $PQ$  in the following way

$a: 00$        $d: 01$        $e: 11$        $\phi: 10$



• DESIGN EQUATIONS AND CIRCUIT DIAGRAM

|      |    |    |    |    |
|------|----|----|----|----|
| $AB$ | 00 | 01 | 11 | 10 |
| $pq$ |    |    |    |    |
| 00   | 00 | 00 | 01 | 00 |
| 01   | x  | 00 | 01 | 11 |
| 11   | 10 | x  | 01 | 11 |
| 10   | 00 | x  | x  | x  |

(a)

|      |    |    |    |    |
|------|----|----|----|----|
| $AB$ | 00 | 01 | 11 | 10 |
| $pq$ |    |    |    |    |
| 00   | 0  | 0  | 0  | 0  |
| 01   | x  | 0  | 0  | 1  |
| 11   | 1  | x  | 0  | 1  |
| 10   | 0  | x  | x  | x  |

$P = q\bar{B}$

(b)

|      |    |    |    |    |
|------|----|----|----|----|
| $AB$ | 00 | 01 | 11 | 10 |
| $pq$ |    |    |    |    |
| 00   | 0  | 0  | 1  | 0  |
| 01   | x  | 0  | 1  | 1  |
| 11   | 0  | x  | 1  | 1  |
| 10   | 0  | x  | x  | x  |

$Q = qA + AB$   
 $= A(q + B)$

(c)

|     |   |   |
|-----|---|---|
| $Q$ | 0 | 1 |
| $P$ |   |   |
| 0   | 0 | 0 |
| 1   | 0 | 1 |

$X = PQ$

(d)

