Design of Synchronous Sequential Circuits

Design Problems

- Starts with a word description of input output relation and ends with a circuit diagram.
- Word description is converted into first state transition diagram, followed by synthesis table and finally circuit diagram.
- There are two approaches of state machine design namely:
  1. Moore Model
  2. Mealy Model

Moore and Mealy Model

- In Moore Model, *the output depends only a present state only.* While in the Mealy machine, *the output depends on the present state and the input.*
- In Moore model, the output remains stable over entire clock period and changes only when there occurs a state change at clock trigger based on the input available at that time. Usually, the mealy machine requires less number of states and thereby less hardware to solve any problem.
- However there is one important disadvantage associated with mealy circuit; the input transients, glitches etc. (if any) are directly conveyed to the output. Also if we want output transitions to be synchronized while input can change at any time, Mealy machine is not preferred.

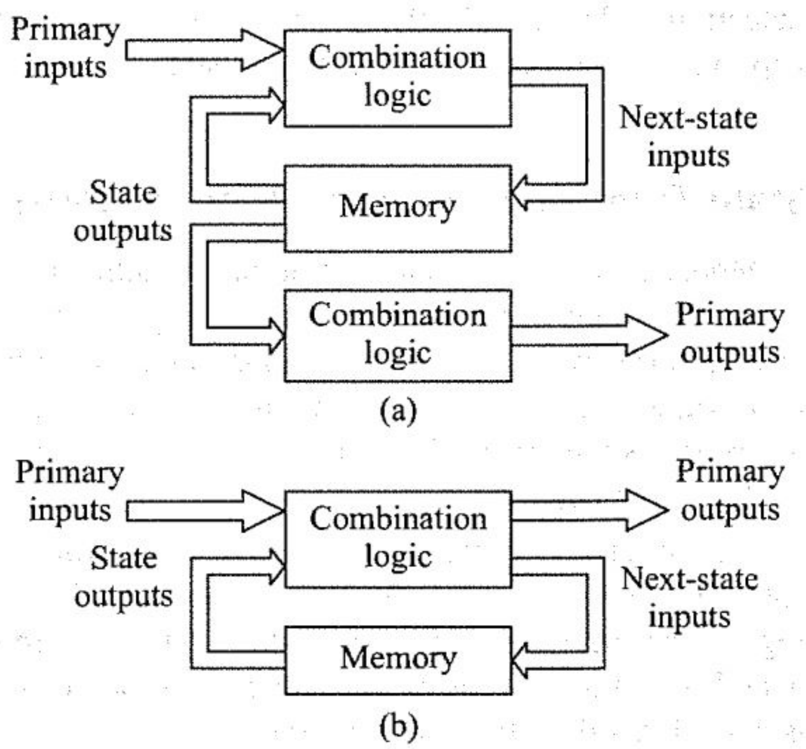
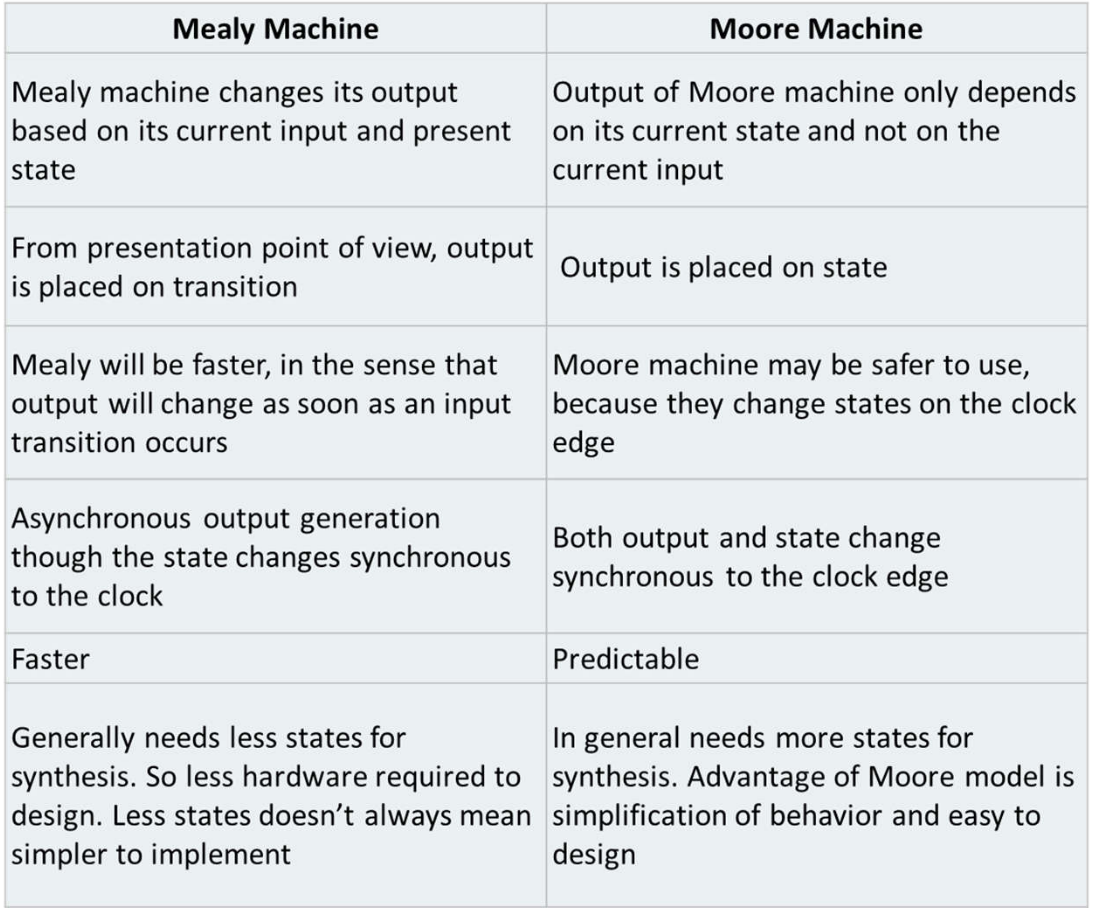


Figure: (a) Moore Model, (b) Mealy Model

Differences between Mealy and Moore Machine



Mealy Machine	Moore Machine
Mealy machine changes its output based on its current input and present state	Output of Moore machine only depends on its current state and not on the current input
From presentation point of view, output is placed on transition	Output is placed on state
Mealy will be faster, in the sense that output will change as soon as an input transition occurs	Moore machine may be safer to use, because they change states on the clock edge
Asynchronous output generation though the state changes synchronous to the clock	Both output and state change synchronous to the clock edge
Faster	Predictable
Generally needs less states for synthesis. So less hardware required to design. Less states doesn't always mean simpler to implement	In general needs more states for synthesis. Advantage of Moore model is simplification of behavior and easy to design

Question:

- Design a sequence detector that receives binary data stream at its input, X and signals when a combination '011' arrive at the binary by making its output Y, high which otherwise remains low. Consider data is coming from left, i.e. the first bit to be identified is 1, second and third 0 from the input time sequence.

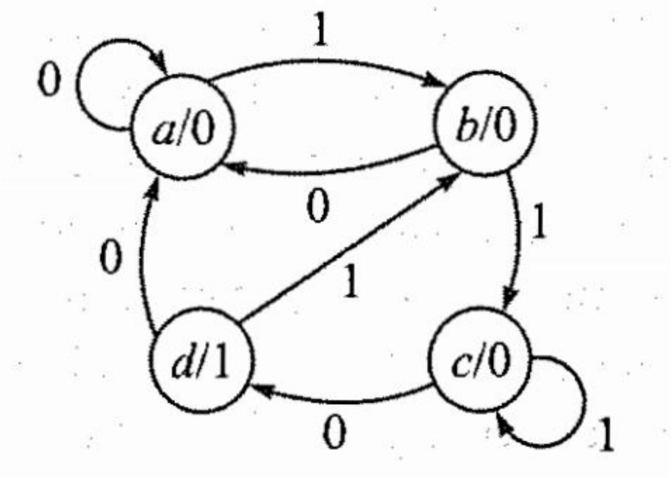


Figure: State Transition Diagram for Moore Model

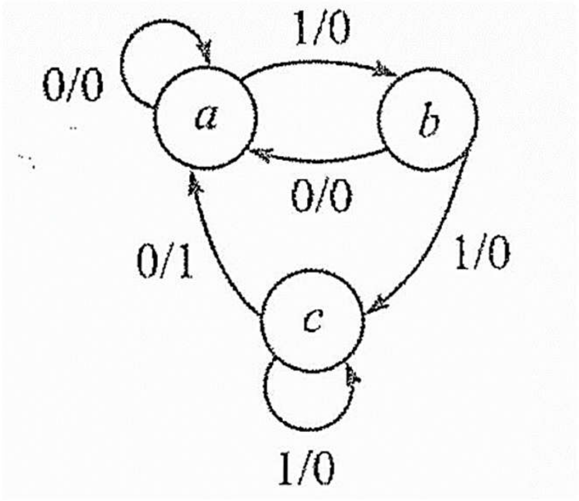
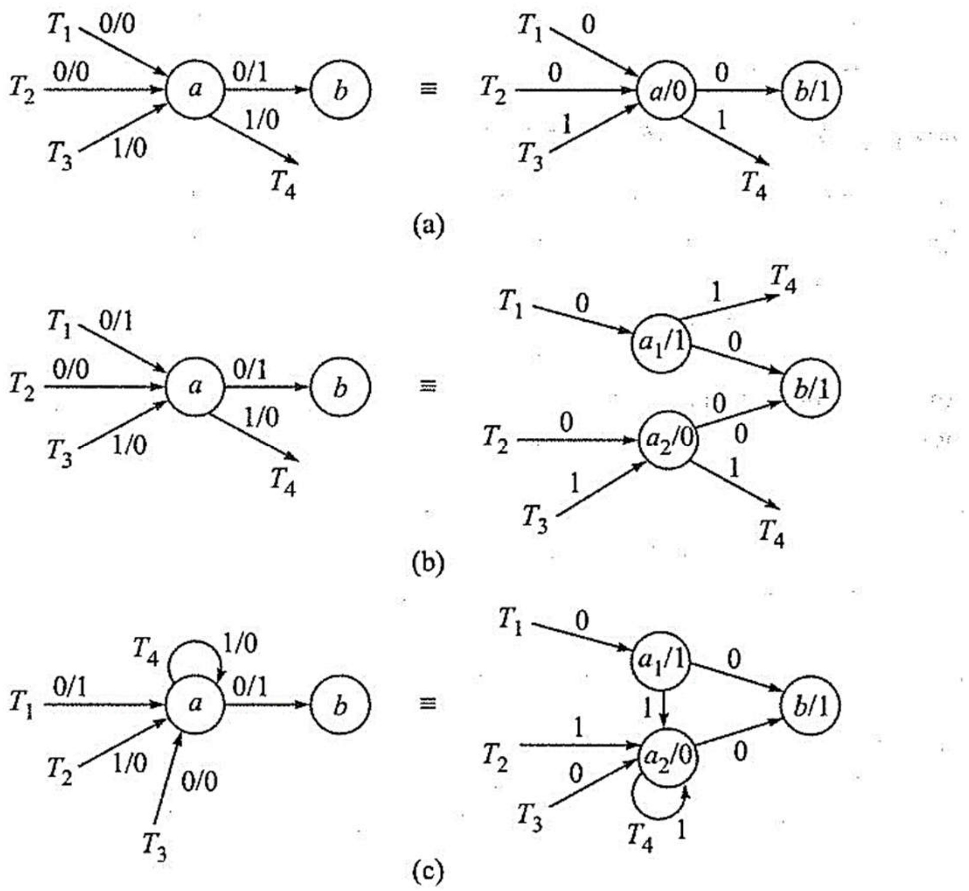


Figure: State Transition Diagram for Mealy Model

Conversion between Mealy and Moore model



- Here, T_1, T_2, T_3 represent paths leading to state a.
- The path T_4 leads from state a when input is *1*.

Rule of conversion

1. 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 that output becomes state output. → (a)
2. If there are more than one output in Mealy model we need as many intermediate state variable. → (b)
3. State transition that loop within a particular state. → c

State Reduction

- 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.
- Two state reduction techniques:

1. Row Elimination Method
2. Implication Table Method

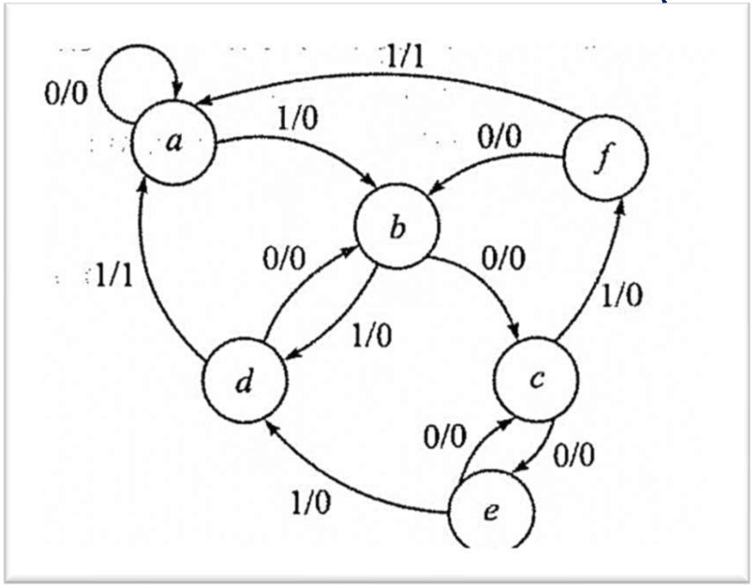
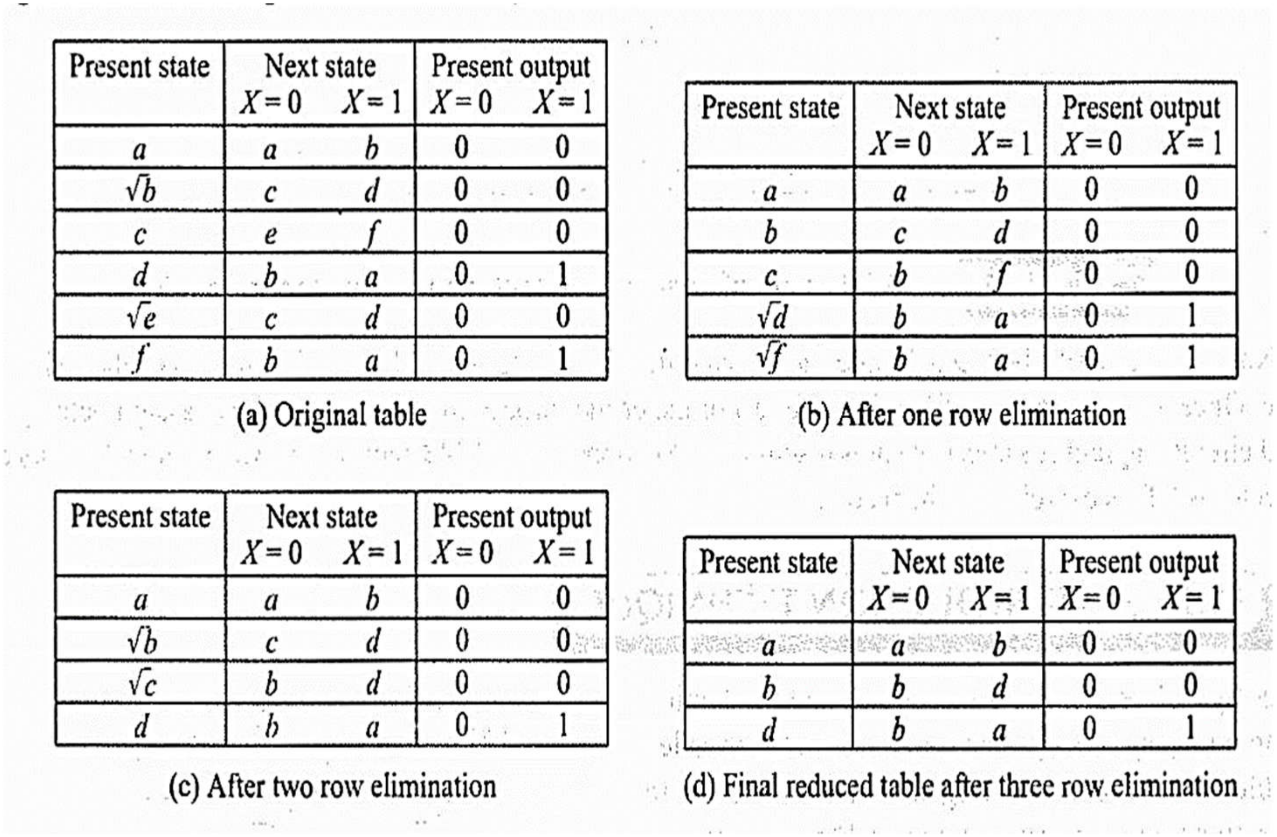


Figure: A State Transition Diagram

Row Elimination Method



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

(b) After one row elimination

Present state	Next state		Present output	
	$X=0$	$X=1$	$X=0$	$X=1$
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		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

(d) Final reduced table after three row elimination

Figure: Tables Showing Row Elimination Steps

- For b and c except for next state as $X=0$ 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.

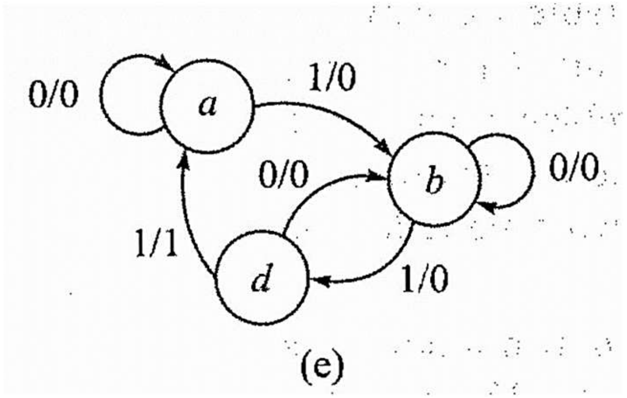


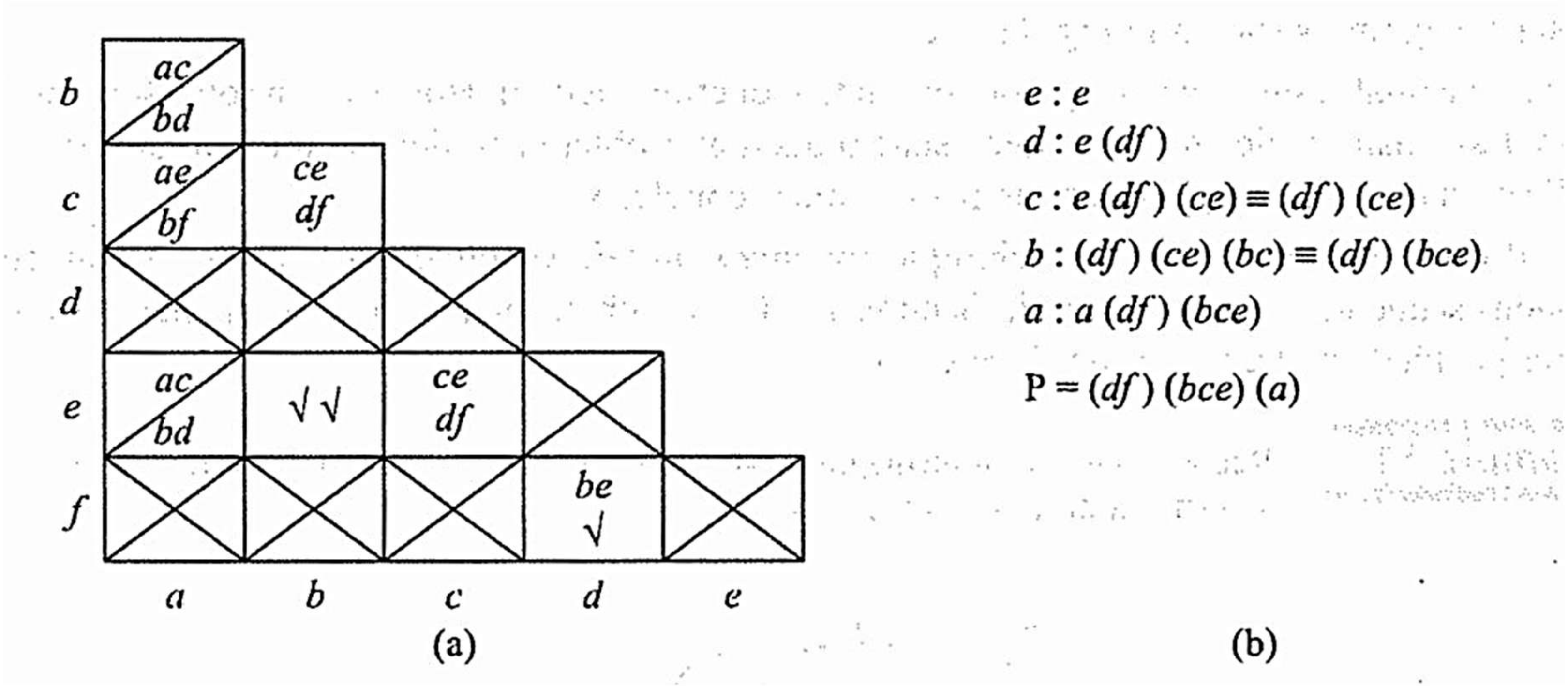
Figure: Reduced State Diagram

Implication Table Method

- Implication table provides a more systematic approach towards 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.
- The cross-point in an implication table is the location where a row and a column meet.
- Here the equivalence between the states crossing other are tested.
- We use the state table derived from the state transition diagram to fill up implication table.

Exaple

See the methods in the class



	a	b	c	d	e
b	~~ac~~ ~~bd~~				
c	~~ae~~ ~~bf~~	ce df			
d	✗	✗	✗		
e	~~ac~~ ~~bd~~	√ √	ce df	✗	
f	✗	✗	✗	be √	✗

(a)

e : *e*

d : *e* (*df*)

c : *e* (*df*) (*ce*) ≡ (*df*) (*ce*)

b : (*df*) (*ce*) (*bc*) ≡ (*df*) (*bce*)

a : *a* (*df*) (*bce*)

P = (*df*) (*bce*) (*a*)

(b)

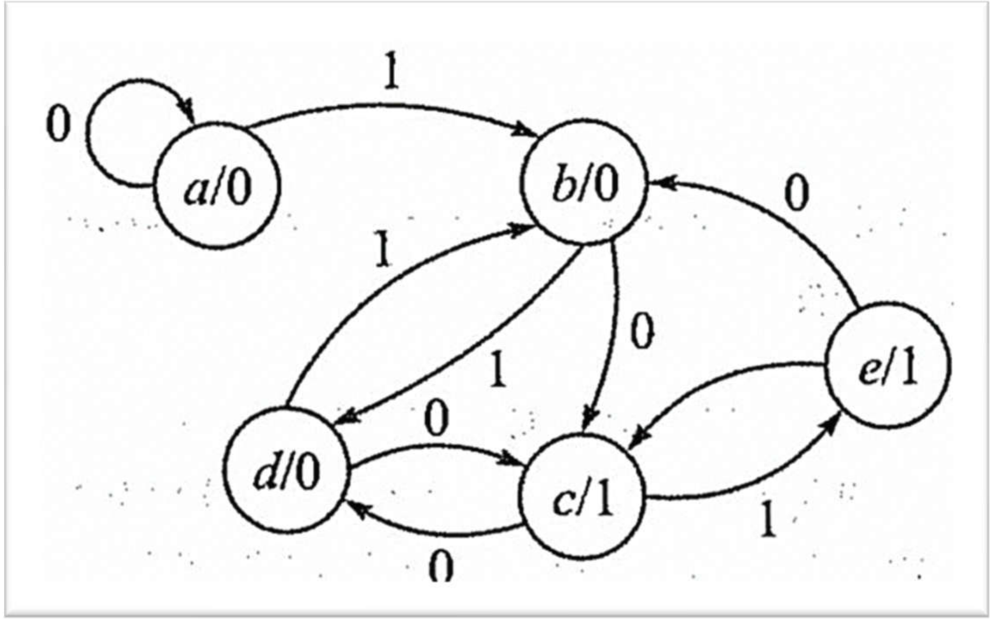
Figure: Implication Table Method of State Reduction; (a) Implication Table, (b) Partition Table

Exercise

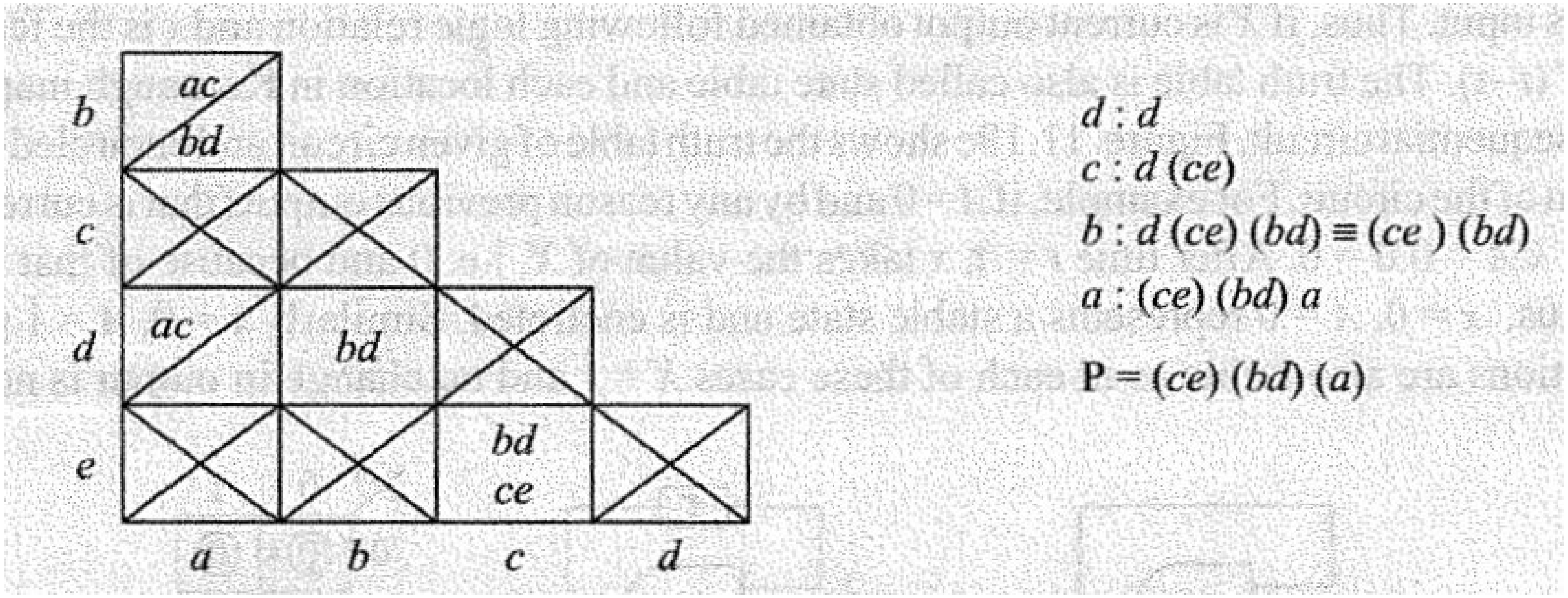
- Reduce the given state transition diagram by:

i. Row elimination method

ii. Implication table method



Ans:



	a	b	c	d
b	ac bd			
c	☒	☒		
d	ac	bd	☒	
e	☒	☒	bd ce	☒

$d : d$

$c : d\,(ce)$

$b : d\,(ce)\,(bd) \equiv (ce)\,(bd)$

$a : (ce)\,(bd)\,a$

$P = (ce)\,(bd)\,(a)$

(a) (b)

Figure: Implication Table Method of State Reduction; (a) Implication Table, (b) Partition Table

Design of Asynchronous Sequential Circuit

Introduction

- Asynchronous sequential circuit, also called Event Driven Circuit, does not have any clock to trigger change of state.
- States changes are triggered by change in input signal i.e. the Circuit behavior is determined by signal at any instant in time and the order in which input signal change.
- Design of this circuit is very complex and have several constraints to be taken care of, which is not required for synchronous circuit.

Where we use Asynchronous Sequential Circuit?

- These are used when speed of operation is important; as they are independent of internal clock pulse, they are operate quickly, so they are used in Quick response time.
- Used in the communication between two units having their own independent clocks.
- Used when we require the better external input handling.

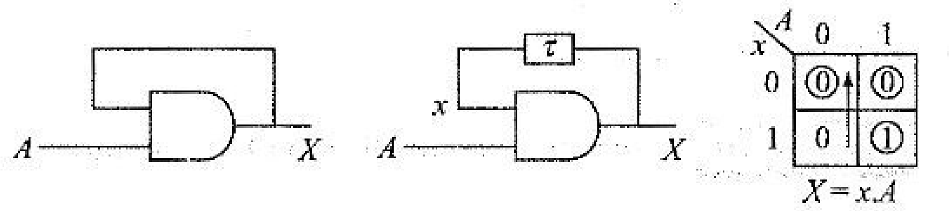
Drawbacks

- More difficult to design
- Though they have a faster performance, their output is uncertain.

Analysis of Asynchronous Sequential Circuit

AND Gate

- The two input AND gate with output feedback an one input is shown in figure ‘a’.
- The circuit is redrawn as shown in figure ‘b’ that includes the effect of propagation delay of the gate say, τ, the finite time after which gate reacts to input.
- Thus, if *X* is current output obtained following logic relation and *x* is the feedback output we write, $x = X(t - \tau)$
- In the given truth table (state table), encircled state indicate stable conditions of the circuit.



x \ A	0	1
0	⓪	⓪
1	0	①

$X = x.A$

(a) AND gate

(b) AND gate with propagation delay

(c) Truth table

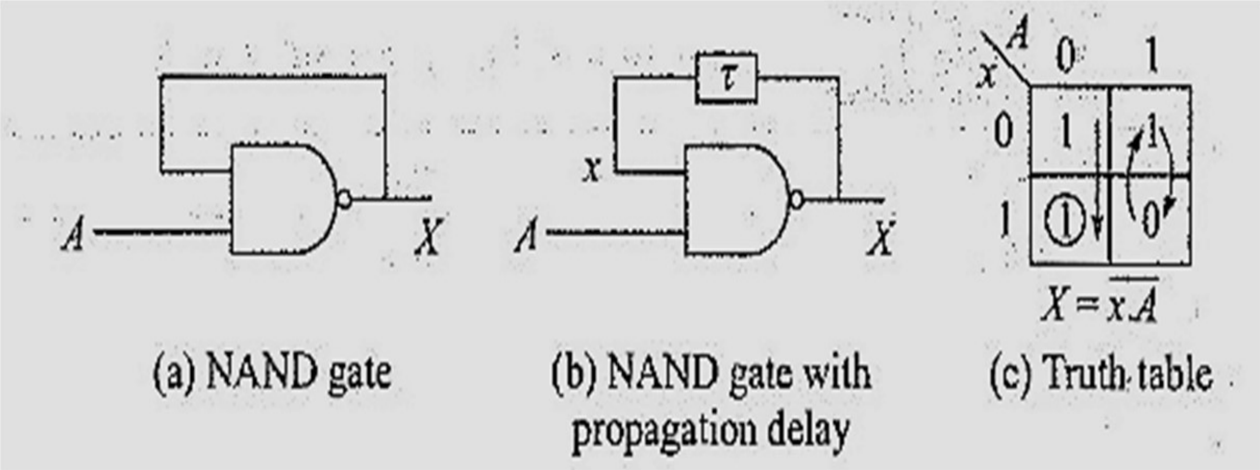
Cases:

- If $A = 0,$ and by any reason previous output (that is currently feedback) $x = 0$ then, $X = x.A = 0.0 = 0$. after time $t = \tau,$ x takes the value of X i.e. 0 and because of that the output X does not change. Thus, $x = 0$, $A = 0$, represent a stable state and is encircled. Similar for the case of $x = 0$ and $A = 1,$ and $x = 1$ and $A = 1$. *In each of these cases, $X = x$ and no change in output is necessary.*
- Consider $A = 1$ and $x = 1$, then $X = 1$. the circuit is held at this stable state. Then when we change the value of $A = 0$, the circuit moves one step to left i.e. from $x = 1$ and $A = 1$ *to* $x = 1, A = 0$ position because x, the feedback input takes finite time, τ to react. At this position, $X= 0$. therefor after τ , x become 0, i.e. we move up by one position in K map to position, a stable one. Thus we find $x = 1,$ $A = 0$ position is unstable.
- ***For any state, if $x = X$ then the circuit is stable if $x \neq X$, it is unstable***

Analysis of Asynchronous Sequential Circuit

NAND Gate

- For $A = 1$, there is no stable state and $x = X'$, for both $x = 0$ and $x = 1$. *Thus there is oscillation between $x = 0$, $A = 1$ and $x = 1$, $A = 1$ state*.



(a) NAND gate

(b) NAND gate with propagation delay

x \ A	0	1
0	1	1
1	1	0

$X = \overline{xA}$

(c) Truth table

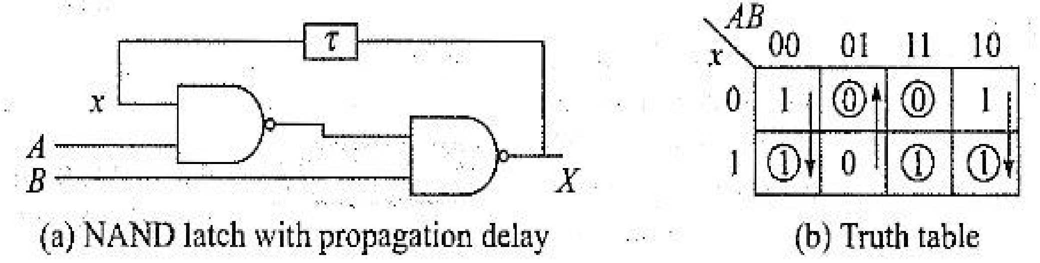
Figure: Two input NAND gate with output feedback

Analysis of Asynchronous Sequential Circuit

Two Input NAND Latch

Constraint to be followed in asynchronous circuit:

- Though there can be more than one input feeding the circuit, at a time only one input variable can change. The other input can change only when the circuit is stabilized following the previous input change. The time required the circuit is in the order of propagation delay of gate i.e. in nanosecond order.
- Similarly, if there are two or more output variables only one output variable can change at any instant, as propagation delays in different paths are different.



x \ AB	00	01	11	10
0	1	⓪ ↑	⓪	1
1	① ↓	0	①	① ↓

(a) NAND latch with propagation delay

(b) Truth table

Figure: Two input NAND Latch

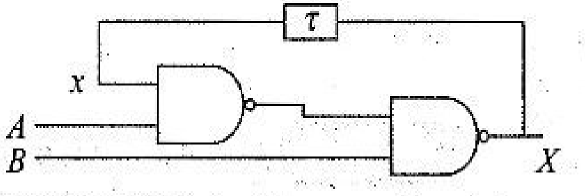
- For any given combination of *x, A, B* if, *X* = *x*, the circuit is stable otherwise not.
- Stables states are encircled and arrows show the movements from transient states.

Transition Table of NAND Latch

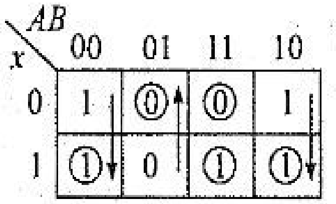
Analysis

i. Input AB change from 00 to 01.

ii. Input AB changes from 10

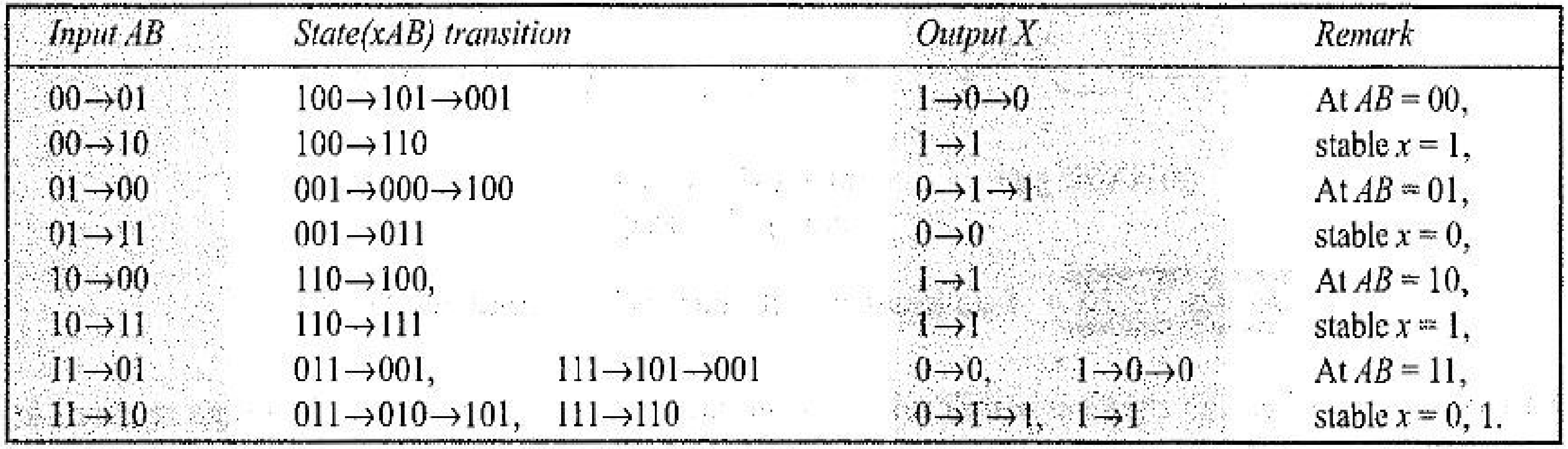


(a) NAND latch with propagation delay



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

(b) Truth table



Input AB	*State(xAB) transition*		*Output X*		*Remark*
00→01	100→101→001		1→0→0		At AB = 00,
00→10	100→110		1→1		stable x = 1,
01→00	001→000→100		0→1→1		At AB = 01,
01→11	001→011		0→0		stable x = 0,
10→00	110→100,		1→1		At AB = 10,
10→11	110→111		1→1		stable x = 1,
11→01	011→001,	111→101→001	0→0,	1→0→0	At AB = 11,
11→10	011→010→101,	111→110	0→1→1,	1→1	stable x = 0, 1.

Example: Analysis of given Mealy Machine

- Consider $x = X(t-\ \tau)$ *where* τ *is the cumulative propagation delay from input side up to X.*
- For all possible combinations of *xAB* we get *X* and *Y*
- States where $X = x$ are stable and encircle.

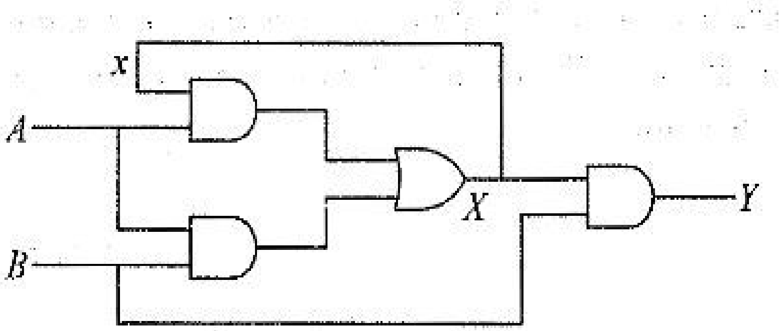
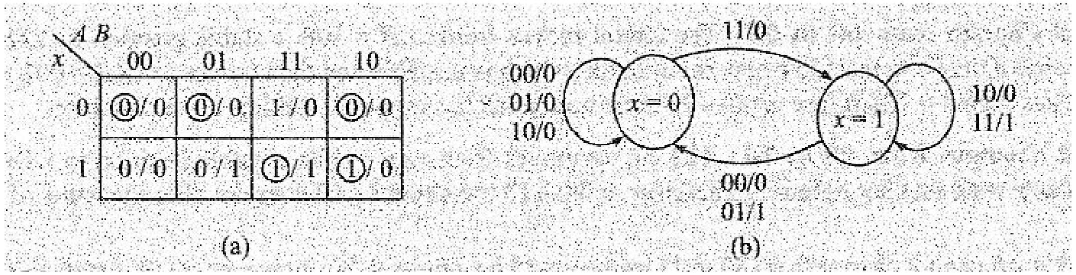


Figure: An asynchronous sequential circuit: Mealy model



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

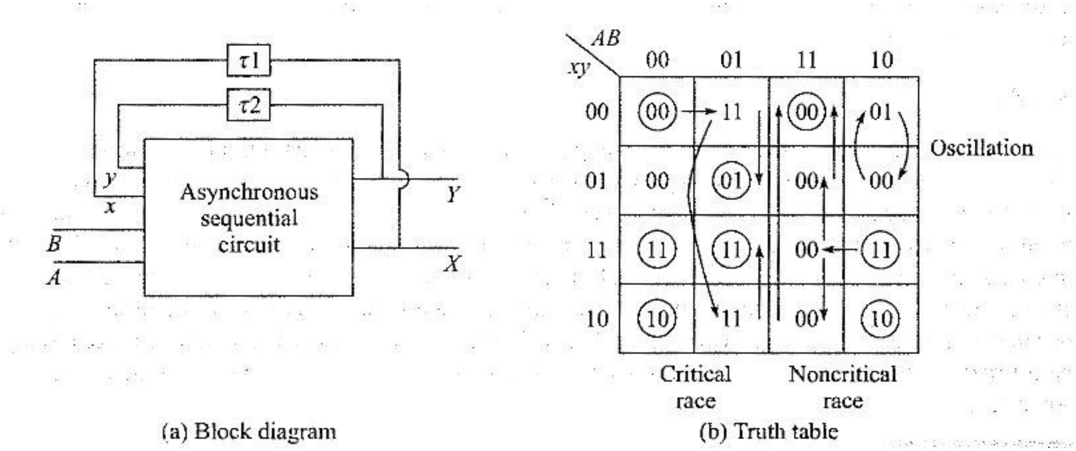
a) Karnaugh map, b) state diagram for asynchronous circuit given

Problems with Asynchronous Sequential Circuits

- Oscillation
- Critical Race
- Hazards

Example

- Consider an asynchronous sequential circuit with two inputs A and B and two outputs, X and 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 and y cannot change simultaneously but with time delays τ1 and τ2 respectively and we can write x = X(t-τ1) and y = Y(t-τ2).
- The stable states are encircled with circuit where xy = XY. But there are certain major problems with this truth table.



(a) Block diagram

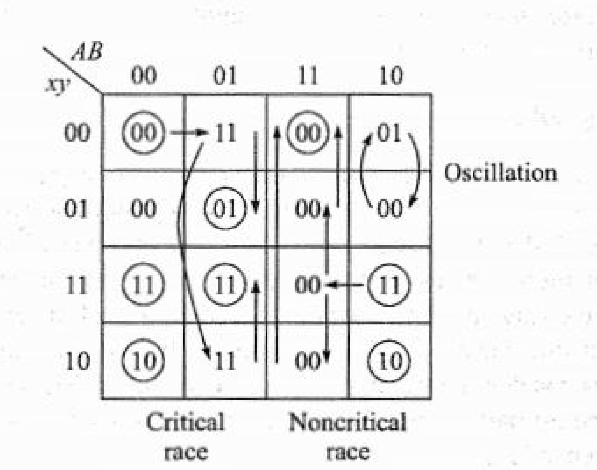
xy \ AB	00	01	11	10
00	00	11	00	01
01	00	01	00	00
11	11	11	00	11
10	10	11	00	10

(b) Truth table

Figure: (a) Block Diagram; (b) Truth Table; of 2 inputs and 2 outputs asynchronous system

Oscillation

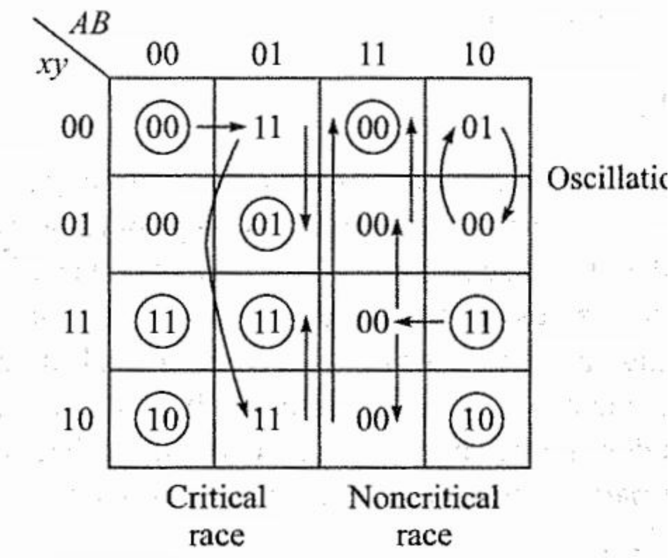
- Consider the stable state xyAB = 0000, where x = X and y = y;
- if the input AB changes from 00 to 10, the circuit goes to xyAB = 0010 state and then output XY = 01. This is transient because xy ≠ XY.
- After time τ2, y take the value of Y=1 and the circuit goes to xyAB=0110 where output XY=00.
- This again is transient state and after another propagation delay of τ2, the circuit goes through xyAB = 0010. Thus, the circuit oscillates between state 0010 and 0110 and the output Y oscillates between 0 and 1 with time gap τ2.
- In asynchronous sequential circuits for any given input, transitions between two unstable states like these are to be avoided to remove oscillation.



xy \ AB	00	01	11	10
00	00	11	00	01
01	00	01	00	00
11	11	11	00	11
10	10	11	00	10

Critical Race

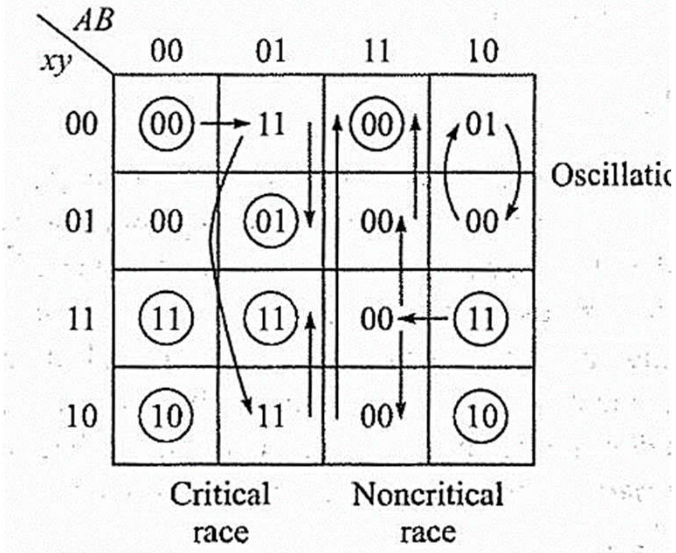
- One of the major problems in asynchronous circuits
- This occurs when an input change tries to modify more than one output.
- Consider state xyAB = 0000. Now, if AB changes to 01, the circuit moves to xyAB = 0001 where XY = 11. Now depending upon which of τ1 and τ2 lower, xy moves from 00 to either 01, or 10.
- If τ1 is lower, x changes earlier and the circuit goes to xyAB = 1001 which is unstable state with output XY = 11; the circuit moves to the state xyAB = 1101 which is a stable state and final output XY = 11.
- If τ2 is lower, y changes earlier and the circuit goes to xyAB = 0101, a stable state and final output is 01.
- Thus, depending upon propagation delays in feedback path, the circuit settles at two different states generating two different set of outputs. Such a situation is called critical race condition and is to avoided in asynchronous sequential circuit.



xy \ AB	00	01	11	10
00	00	11	00	01
01	00	01	00	00
11	11	11	00	11
10	10	11	00	10

Non-critical Race

- Race can be non-critical too, in which case its presence does not pose any problem for the circuit behavior.
- Consider stable state xyAB = 1110; if the input AB changes to 11, the ciruit goes to xyAB = 1111 where output XY = 00.
- Again depending on propagation delays, xy becomes either 01 and 10. if xy = 01, then the circuit moves to xyAB = 0111 and then to 0011 and settles there. If xy = 10 then the transition path is 1111 → 1011 → 0011.
- In both case final state is 0011 and output is 00.
- Since, the race condition does not lead to two different states, it is termed as non-critical race.



xy \ AB	00	01	11	10
00	00	11	00	01
01	00	01	00	00
11	11	11	00	11
10	10	11	00	10

Hazards

- Static and dynamic hazards causes malfunctioning of asynchronous sequential circuit.
- Situations like Y = A + A’ and Y = AA’ are to be avoided for any input output combinations with the help of hazards covers in truth table.
- In circuit with feedback even when these hazards are adequately covered there can be another problem called essential hazard.
- This occurs when change in input does not reach one part of the circuit while from other part one output fed back to the input side becomes available.
- Essential hazard is avoided by adding delay, may be in the form of additional gates that does not change the logic level, in the feedback path.

Design Example

- 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 transitions 1 → 0. X remains high as long as this A = 1, B =0 are maintained.
- If any of A and B changes at this time output X goes low. It becomes high again when at A = 1, B goes from 1 to 0.

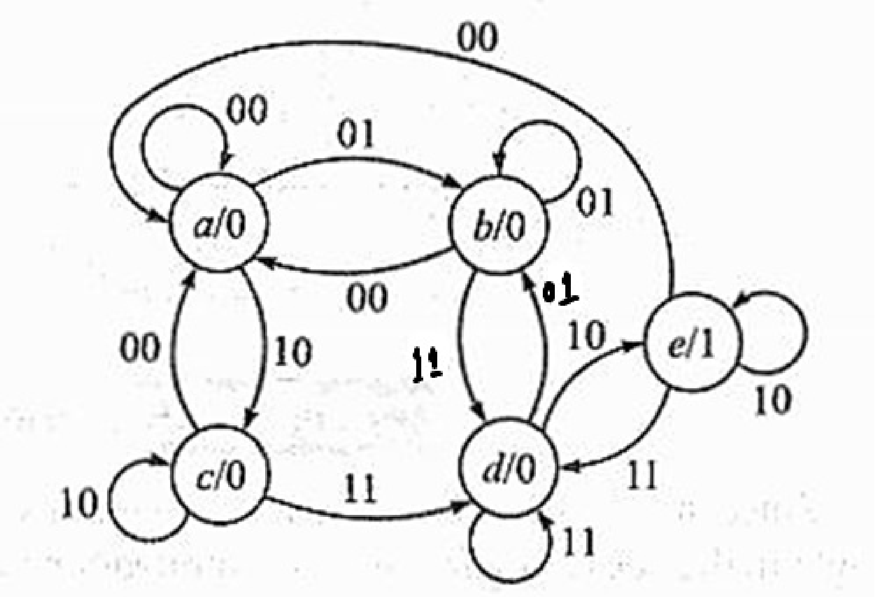


Figure: State Transition Diagram of the problem

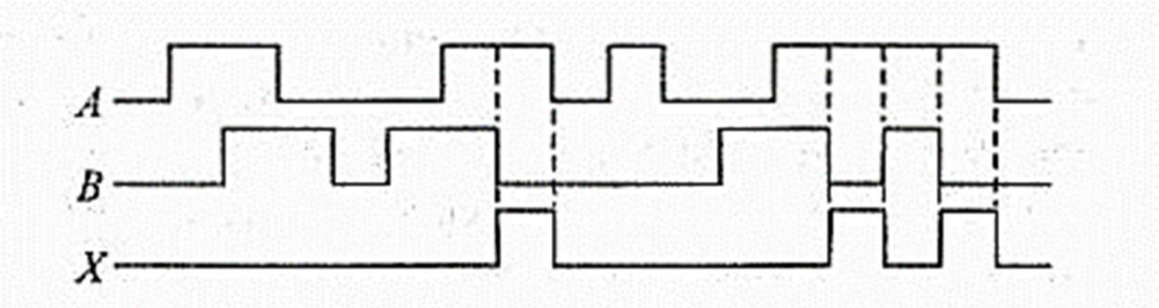
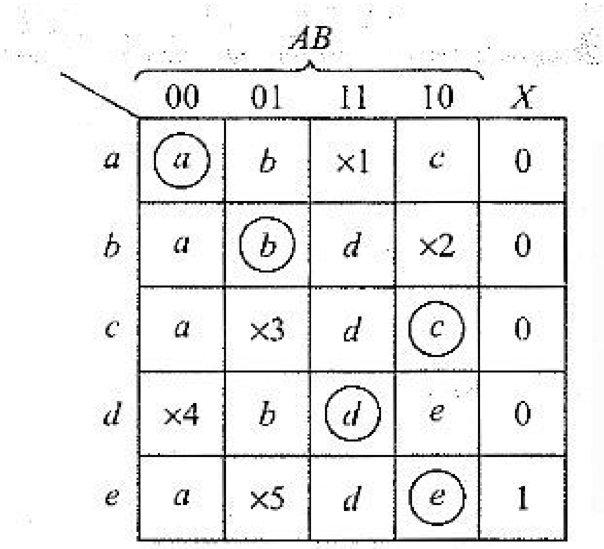


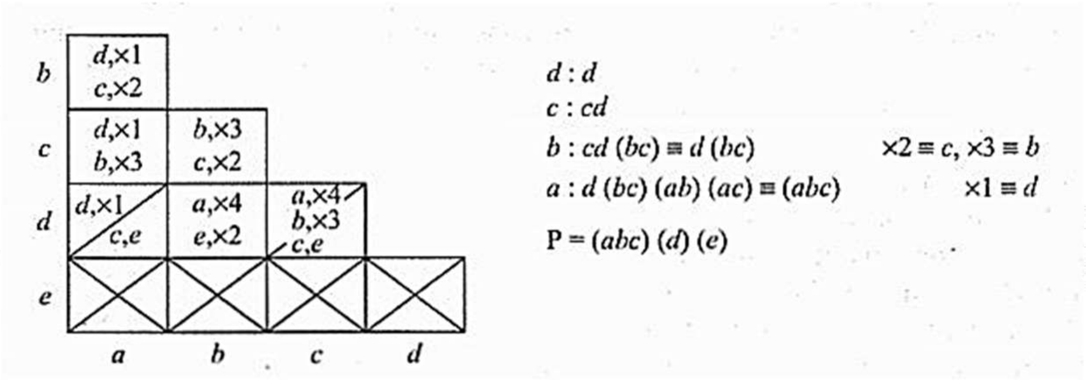
Figure: Timing Diagram of the problem

Design Example



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

Primitive Table or primitive flow table or flow table



b	d,×1 c,×2			
c	d,×1 b,×3	b,×3 c,×2		
d	d,×1 c,e	a,×4 e,×2	a,×4 b,×3 c,e	
e	☒	☒	☒	☒
	a	b	c	d

d : d

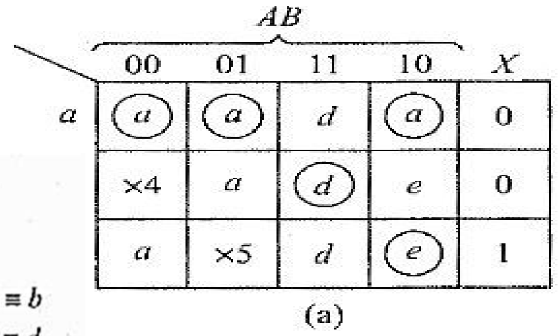
c : cd

b : cd (bc) ≡ d (bc) ×2 ≡ c, ×3 ≡ b

a : d (bc) (ab) (ac) ≡ (abc) ×1 ≡ d

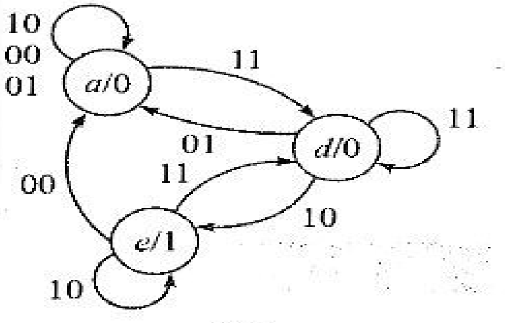
P = (abc) (d) (e)

State Reduction Using Implication Table



	AB 00	01	11	10	X
a	ⓐ	ⓐ	d	ⓐ	0
	×4	a	ⓓ	e	0
	a	×5	d	ⓔ	1

(a)



(b)

Reduced State Table and State Transition Diagram

Design Example

State Assignment:

a : 00

d : 01

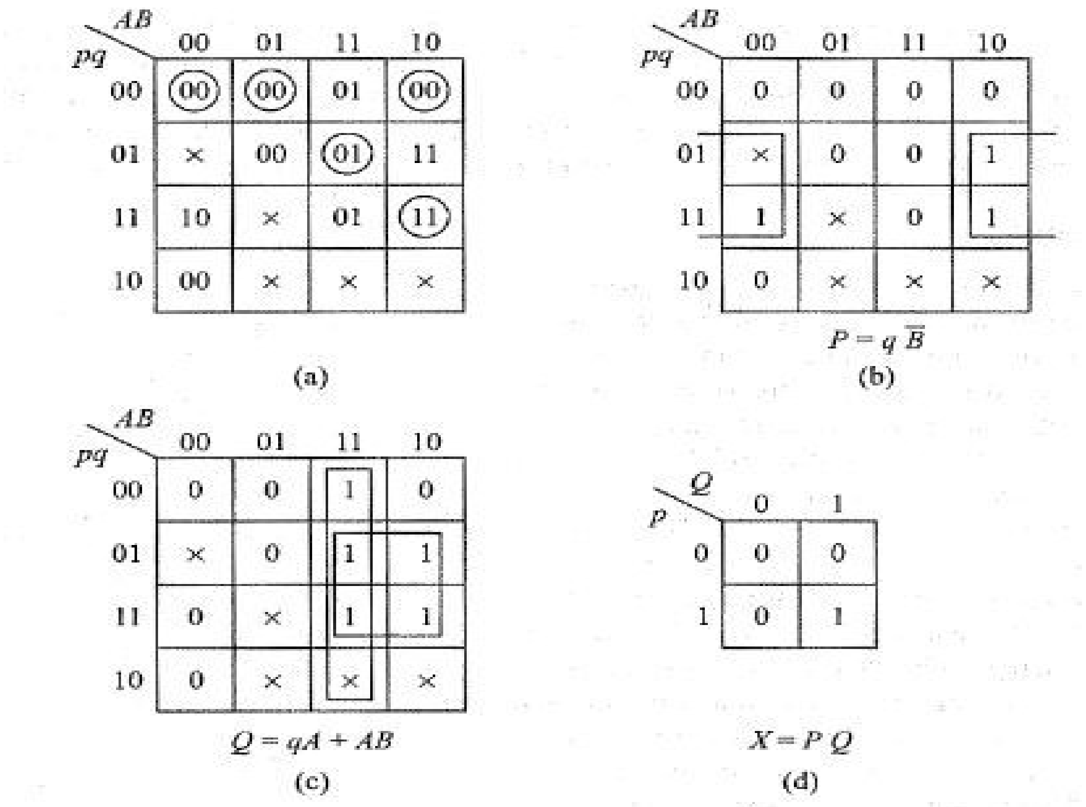
e : 11

∅ : 10

Primitive Table:

- In this table, if all the rows representing a state has only one stable state for all possible input combinations and it is termed as primitive table, or primitive flow table or simply flow table.
- It is prepared from state transition diagram.

Design Example



pq \ AB	00	01	11	10
00	00	00	01	00
01	×	00	01	11
11	10	×	01	11
10	00	×	×	×

(a)

pq \ AB	00	01	11	10
00	0	0	0	0
01	×	0	0	1
11	1	×	0	1
10	0	×	×	×

$P = q\,\overline{B}$

(b)

pq \ AB	00	01	11	10
00	0	0	1	0
01	×	0	1	1
11	0	×	1	1
10	0	×	×	×

$Q = qA + AB$

(c)

P \ Q	0	1
0	0	0
1	0	1

$X = P\,Q$

(d)

(a) Reduced State Diagram, (b) – (d) K-map and design equation

- We represent state variable by P and Q, the corresponding feedback variables are represented by p and q respectively.
- We use Karnaugh map to get expression of sate variables P and Q as a function of input A, B and feedback variables p and q.
- The equation of output is generated from P and Q

Design Example

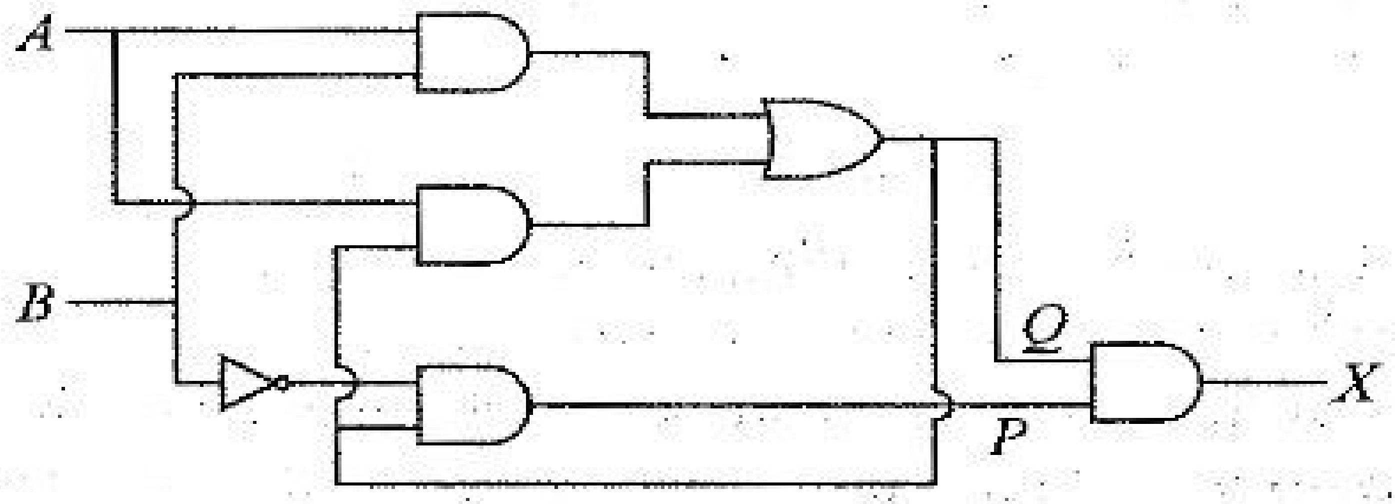


Figure: Circuit diagram of given problem