Sairam INSTITUTIONS

Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

YEAR	SEM
II	III

CS8351

DIGITAL PRINCIPLES AND SYSTEM DESIGN

UNIT NO.4

4.2 DESIGN OF ASYNCHRONOUS SEQUENTIAL CIRCUITS

Version: 1.XX

4.1 DESIGN OF FUNDAMENTAL MODE SEQUENTIAL CIRCUITS

The design of an asynchronous sequential circuit starts from the statement of the problem and concludes in a logic diagram. There are a number of design steps that must be carried out in order to minimize the circuit complexity and to produce a stable circuit without critical races.

The design steps are as follows:

1. State the design specifications.
2. Obtain a primitive flow table from the given design specifications.
3. Reduce the flow table by merging rows in the primitive flow table.
4. Assign binary state variables to each row of the reduced flow table to obtain the transition table. The procedure of state assignment eliminates any possible critical races.
5. Assign output values to the dashes associated with the unstable states to obtain the output maps.
6. Simplify the Boolean functions of the excitation and output variables and draw the logic diagram.

1. Design a gated latch circuit with inputs, G (gate) and D (data), and one output, Q. Binary information present at the D input is transferred to the Q output when G is equal to 1. The Q output will follow the D input as long as G= 1. When G goes to 0, the information that was present at the D input at the time of transition occurred is retained at the Q output. The gated latch is a memory element that accepts the value of D when G= 1 and retains this value after G goes to 0, a change in D does not change the value of the output Q.

Soln:

Step 1:

From the design specifications, we know that Q= 0 if DG= 01

and Q= 1 if

DG= 11 because D must be equal to Q when G= 1.

When G goes to 0, the output depends on the last value of D. Thus, if the transition is from 01 to 00 to 10, then Q must remain 0 because D is 0 at the time of the transition from 1 to 0 in G.

If the transition of DG is from 11 to 10 to 00, then Q must remain 1. This information results in six different total states, as shown in the table.

State	Inputs		Output	Comments
	D	G	Q	
a	0	1	0	D= Q because G= 1
b	1	1	1	D= Q because G= 1
c	0	0	0	After state a or d
d	1	0	0	After state c
e	1	0	1	After state b or f
f	0	0	1	After state e

Step 2: A primitive flow is a flow table with only one stable total state in each row. It has one row for each state and one column for each input combination.

	DG 00	01	11	10
a	c, –	ⓐ, 0	b, –	–, –
b	–, –	a, –	ⓑ, 1	e, –
c	ⓒ, 0	a, –	–, –	d, –
d	c, –	–, –	b, –	ⓓ, 0
e	f, –	–, –	b, –	ⓔ, 1
f	ⓕ, 1	a, –	–, –	e, –

Primitive flow table

The primitive flow table has only stable state in each row. The table can be reduced to a smaller number of rows if two or more stable states are placed in the same row of the flow table. The grouping of stable states from separate rows into one common row is called merging

	DG 00	01	11	10
a	c, –	ⓐ, 0	b, –	–, –
c	ⓒ, 0	a, –	–, –	d, –
d	c, –	–, –	b, –	ⓓ, 0

	DG 00	01	11	10
b	–, –	a, –	ⓑ, 1	e, –
e	f, –	–, –	b, –	ⓔ, 1
f	ⓕ, 1	a, –	–, –	e, –

States that are candidates for merging

Thus, the three rows a, c, and d can be merged into one row. The second row of the reduced table results from the merging of rows b, e, and f of the primitive flow table.

Reduced table- 1

	DG 00	01	11	10
a, c, d	ⓒ, 0	ⓐ, 0	b, –	ⓓ, 0
b, e, f	ⓕ, 1	a, –	ⓑ, 1	ⓔ, 1

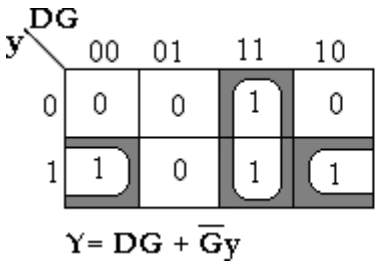
The states c & d are replaced by state a, and states e & f are replaced by state b

Step 4:

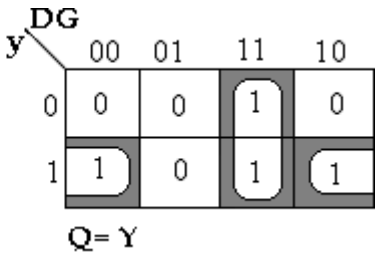
Reduced table- 2

Assign distinct binary value to each state. This assignment converts the flow table into a transition table. A binary state assignment must be made to ensure that the circuit will be free of critical races.

Assign 0 to state a, and 1 to state b in the reduced state table.



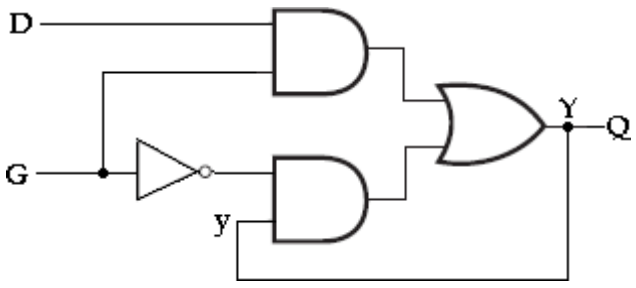
$Y = DG + \bar{G}y$



$Q = Y$

Transition table and output map

Step 5:



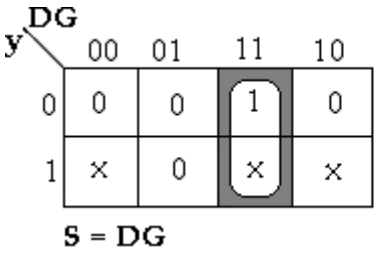
Gated-Latch Logic diagram

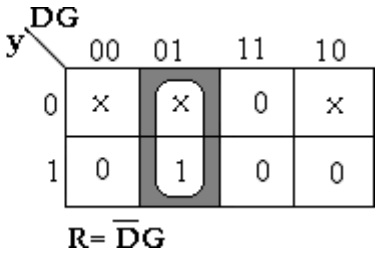
The diagram can be implemented also by means of an SR latch. Obtain the Boolean function for S and R inputs.

y	Y	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

SR Latch excitation table

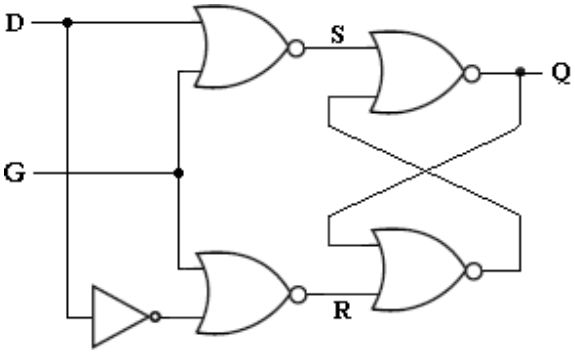
From the information given in the transition table and from the latch excitation table conditions, we can obtain the maps for the S and R inputs of the latch

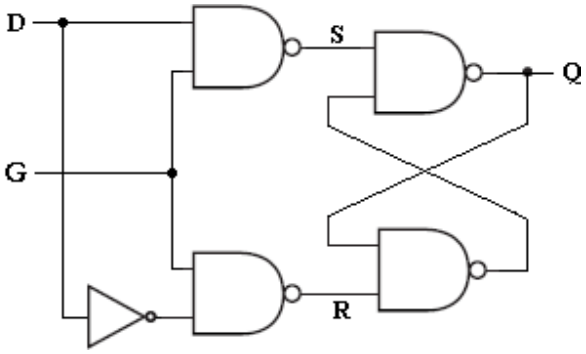




The logic diagram consists of an SR latch using NOR latch and the gates required to implement the S and R Boolean functions. With a NAND latch, we must use the complemented values for S and R.

S' = (DG)' and R' = (D'G)'





Logic diagram with NOR latch **Logic diagram with NAND latch**

2. Design a negative-edge triggered T flip-flop. The circuit has two inputs, T (toggle) and G (clock), and one output, Q. the output state is complemented if T= 1 and the clock changes from 1 to 0 (negative-edge triggering). Otherwise, under any other input condition, the output Q remains unchanged

Step 1:

Starting with the input condition TC= 11 and assign it to a. The circuit goes to state b and output Q complements from 0 to 1 when C changes from 1 to 0 while T remains a 1.

Another change in the output occurs when the circuit changes from state c to state d. In this case, T=1, C changes from 1 to 0, and the output Q complements from 1 to 0. The other four states in the table do not change the output, because T is equal to 0. If Q is initially 0, it stays at 0, and if initially at 1, it stays at 1 even though the clock input changes.

State	Inputs		Output	Comments
	T	G	Q	
a	1	1	0	Initial output is 0
b	1	0	1	After state a
c	1	1	1	Initial output is 1
d	1	0	0	After state c
e	0	0	0	After state d or f
f	0	1	0	After state e or a
g	0	0	1	After state b or h
h	0	1	1	After state g or c

Specifications of total states

Step 2: Merging of the flow table

The information for the primitive flow table can be obtained directly from the condition listed in the above table. We first fill in one square in each row belonging to stable state in that row as listed in the table.Then we enter dashes in those squares whose input differs by two variables from the input corresponding to the stable state.

The unstable conditions are then determined by utilizing the information listed under the comments in the above table

STEP : 3 COMPATIBLE TABLE

TC

	00	01	11	10
a	–, –	f, –	(a), 0	b, –
b	g, –	–, –	c, –	(b), 1
c	–, –	h, –	(c), 1	d, –
d	e, –	–, –	a, –	(d), 0
e	(e), 0	f, –	–, –	d, –
f	e, –	(f), 0	a, –	–, –
g	(g), 1	h, –	–, –	b, –
h	g, –	(h), 1	c, –	–, –

Primitive flow table

The rows in the primitive flow table are merged by first obtaining all compatible pairs of states. This is done by means of the implication table.

	a	b	c	d	e	f	g
b	a,c x						
c	X	b,d x					
d	b,d x	X	a,c x				
e	b,d x	e,g x b,d x	f,h x	✓			
f	✓	e,g x a,c x	f,h x a,c x	✓	✓		
g	f,h x	✓	b,d x	e,g x b,d x	X	e,g x f,h x	
h	f,h x a,c x	✓	✓	d,e x c,f x	e,g x f,h x	X	✓

Implication table

The implication table is used to find the compatible states. The only difference is that when comparing rows, we are at liberty to adjust the dashes to fit any desired condition. The two states are compatible if in every column of the corresponding rows in the primitive flow table, there are identical or compatible pairs and if there is no conflict in the output values.

A check mark ( ) designates a square whose pair of states is compatible. Those states that are not compatible are marked with a cross (x). The remaining squares are recorded with the implied pairs that need further investigation.

The squares that contain the check marks define the compatible pairs: (a, f) (b, g) (b, h) (c, h) (d, e) (d, f) (e, f) (g, h)

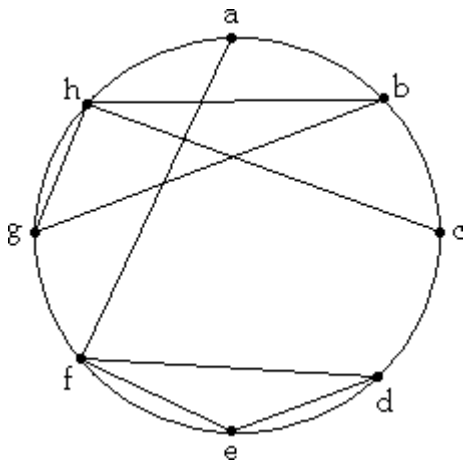
Step 4: Maximal compatibles

Having found all the compatible pairs, the next step is to find larger set of states that are compatible. The ***maximal compatible*** is a group of compatibles that contain all the possible combinations of compatible states. The maximal compatible can be obtained from a merger diagram

The **merger diagram** is a graph in which each state is represented by a dot placed along the circumference of a circle. Lines are drawn between any two corresponding dots that form a compatible pair. All possible compatibles can be obtained from the merger diagram by observing the geometrical patterns in which states are connected to each other.

- A line represents a compatible pair
- A triangle constitutes a compatible with three states

An n-state compatible is represented in the merger diagram by an n-sided polygon with all its diagonals connected.



Diagram

The merger diagram is obtained from the list of compatible pairs derived from the implication table. There are eight straight lines connecting the dots, one for each compatible pair. The lines form a geometrical pattern consisting of two triangles connecting (b, g, h) & (d, e, f) and two lines (a, f) & (c, h). The maximal compatibles are:

(a, f) (b, g, h) (c, h) (d, e, f)

TC

	00	01	11	10
a, f	e, –	(f), 0	(a), 0	b, –
b, g, h	(g), 1	(h), 1	c, –	(b), 1
c, h	g, –	(h), 1	(c), 1	d, –
d, e, f	(e), 0	(f), 0	a, –	(d), 0

Reduced Flow table

The reduced flow table is drawn. The compatible states are merged into one row that retains the original letter symbols of the states. The four compatible set of states are used to merge the flow table into four rows.

	TC 00	01	11	10
a	d, –	(a), 0	(a), 0	b, –
b	(b), 1	(b), 1	c, –	(b), 1
c	b, –	(c), 1	(c), 1	d, –
d	(d), 0	(d), 0	a, –	(d), 0

Final Reduced Flow table

Here we assign a common letter symbol to all the stable states in each merged row. Thus, the symbol f is replaced by a; g & h are replaced by b, and similarly for the other two rows.

Step 5: State Assignment and Transition table

Find the race-free binary assignment for the four stable states in the reduced flow table. Assign a= 00, b= 01, c= 11 and d= 10.

Substituting the binary assignment into the reduced flow table, the transition table is obtained. The output map is obtained from the reduced flow table.

Transition Table and Output Map

y1y2 \ TC	00	01	11	10
00	10	(00)	(00)	01
01	(01)	(01)	11	(01)
11	01	(11)	(11)	10
10	(10)	(10)	00	(10)

Transition table

y1y2 \ TC	00	01	11	10
00	0	0	0	X
01	1	1	1	1
11	1	1	1	X
10	0	0	0	0

Output map Q= y_2

Logic Diagram:

y_1y_2 \ TC	00	01	11	10
00	1	0	0	0
01	0	0	1	0
11	0	X	X	X
10	X	X	0	X

$S_1 = y_2TC + y'_2T'C'$

y_1y_2 \ TC	00	01	11	10
00	0	X	X	X
01	X	X	0	X
11	1	0	0	0
10	0	0	1	0

$R_1 = y_2T'C' + y_2'TC$

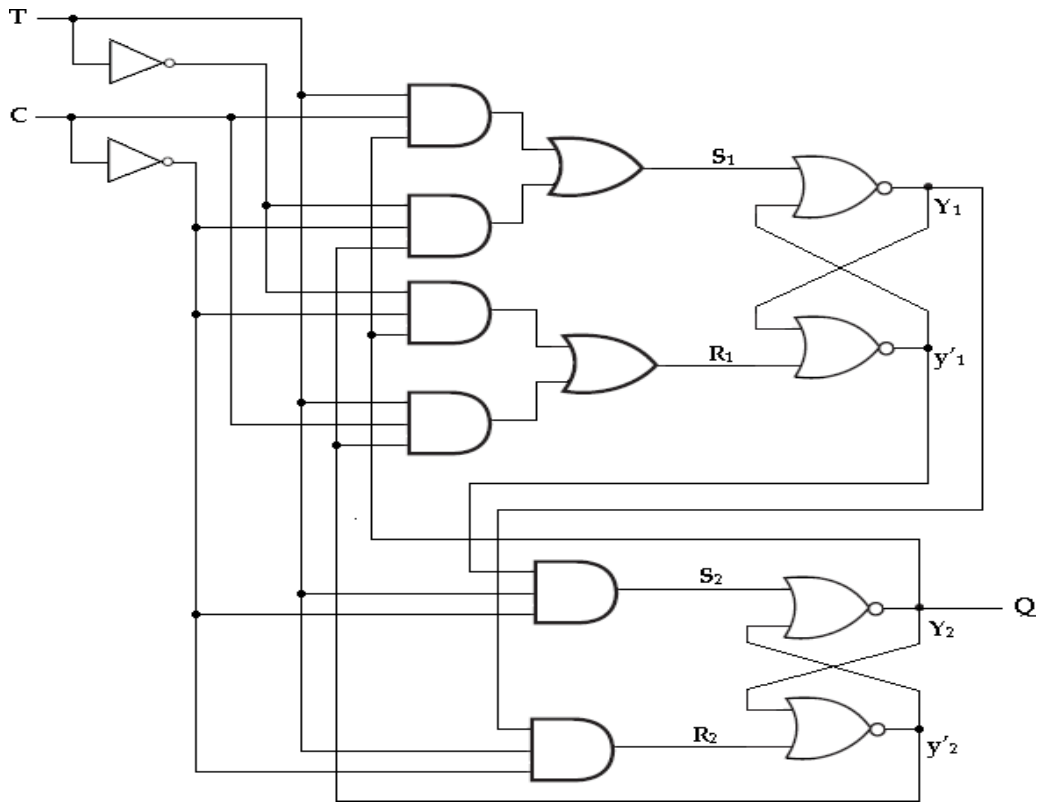
y_1y_2 \ TC	00	01	11	10
00	0	0	0	1
01	X	X	X	X
11	X	X	X	0
10	0	0	0	0

$S_2 = y'_1TC'$

y_1y_2 \ TC	00	01	11	10
00	X	X	X	0
01	0	0	0	0
11	0	0	0	1
10	X	X	X	X

$R_2 = y_1TC'$

Maps for Latch inputs



3. Develop a state diagram and primitive flow table for a logic system that has two inputs, X and Y, and a single output X, which is to behave in the following manner. Initially, both inputs and output are equal to 0. Whenever X= 1 and Y= 0, the Z becomes 1 and whenever X= 0 and Y= 1, the Z becomes 0. When inputs are zero, i.e. X= Y= 0 or inputs are one, i.e. X= Y= 1, the output Z does not change; it remains in the previous state. The logic system has edge triggered inputs without having a clock. The logic system changes state on the rising edges of the two inputs. Static input values are not to have any effect in changing the Z output.

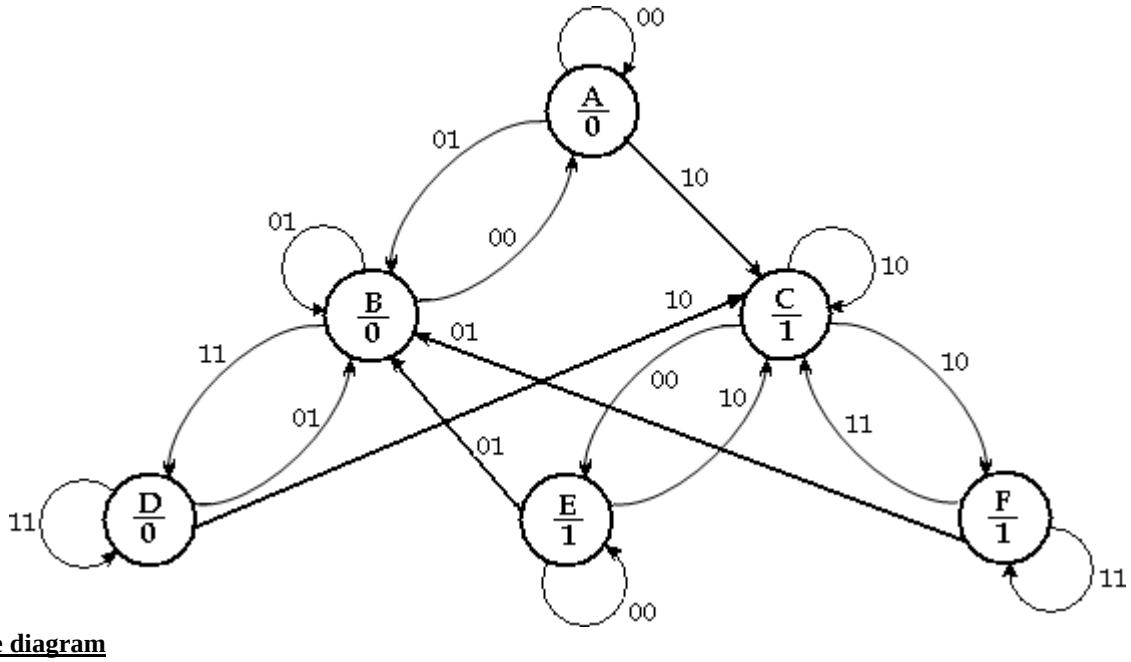
Soln:

The conditions given are,

- Initially both inputs X and Y are 0. When X= 1, Y= 0; Z= 1
- When X= 0, Y= 1; Z= 0
- When X= Y= 0 or X= Y= 1, then Z does not change, it remains in the previous state.

Step 1:

The above state transitions are represented in the state diagram as,



State diagram

Step 2:

A primitive flow table is constructed from the state diagram. The primitive flow table has one row for each state and one column for each input combination. Only one stable state exists for each row in the table. The stable state can be easily identified from the state diagram. For example, state A is stable with output 0 when inputs are 00, state C is stable with output 1 when inputs are 10 and so on.

We know that both inputs are not allowed to change simultaneously, so we can enter dash marks in each row that differs in two or more variables from the input variables associated with the stable state. For example, the first row in the flow table shows a stable state with an input of 00. Since only one input can change at any given time, it can change to 01 or 10, but not to 11. Therefore we can enter two dashes in the 11 column of row A.

The remaining places in the primitive flow table can be filled by observing state diagram. For example, state B is the next state for present state A when input combination is 01; similarly state C is the next state for present state A when input combination is 10.

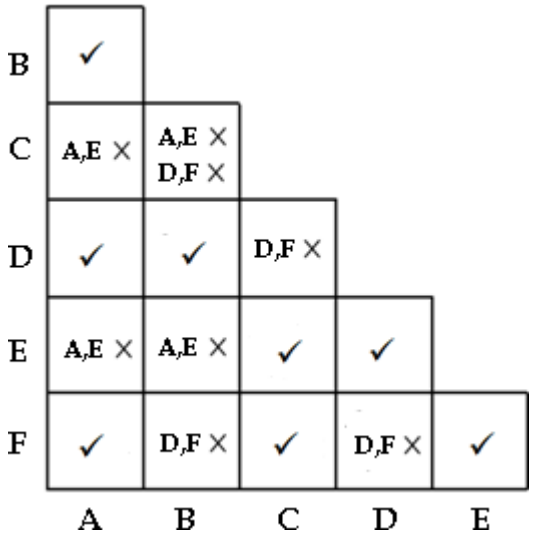
Step 3:

Primitive flow table

	XY 00	01	11	10
A	(A), 0	B, –	–, –	C, –
B	A, –	(B), 0	D, –	–, –
C	E, –	–, –	F, –	(C), 1
D	–, –	B, –	(D), 0	C, –
E	(E), 1	B, –	–, –	C, –
F	–, –	B, –	(F), 1	C, –

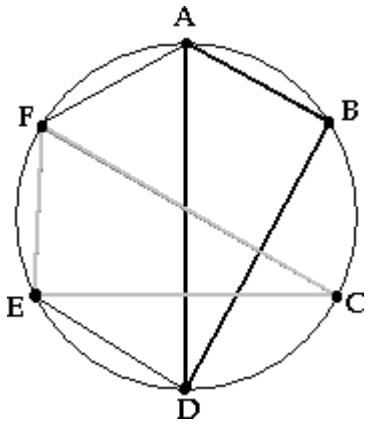
The rows in the primitive flow table are merged by first obtaining all

compatible pairs of states. This is done by means of the implication table.



The squares that contain the check marks ( ) define the compatible pairs: (A, B) (A, D) (A, F) (B, D) (C, E) (C, F) (D, E) (E, F)

Step 4:
The merger diagram is obtained from the list of compatible pairs derived from the implication table. There are eight straight lines connecting the dots, one for each compatible pair. The lines form a geometrical pattern consisting of two triangles connecting (A, B, D) & (C, E, F) and two lines (A, F) & (D, E). The maximal compatibles are:
(A, B, D) (C, E, F) (A, F) (D, E)



Merger diagram
Closed covering condition:

The condition that must be satisfied for merging rows is that the set of chosen compatibles must cover all the states and must be closed. The set will cover all the states if it includes all the states of the original state table. The closure condition is

satisfied if there are no implied states *or* if the implied states are included within the set. A closed set of compatibles that covers all the states is called a *closed covering.*

If we remove (A, F*)* and (D, E), we are left with a set of two compatibles:

(A, B, D) (C, E, F)

All six states from the primitive flow table are included in this set. Thus, the set satisfies the covering condition.

The reduced flow table is drawn as below.

	XY 00	01	11	10
A,B,D	(A),0	(B),0	(D),0	C,–
C,E,F	(E),1	B,–	(F),1	(C),1

Reduced flow table

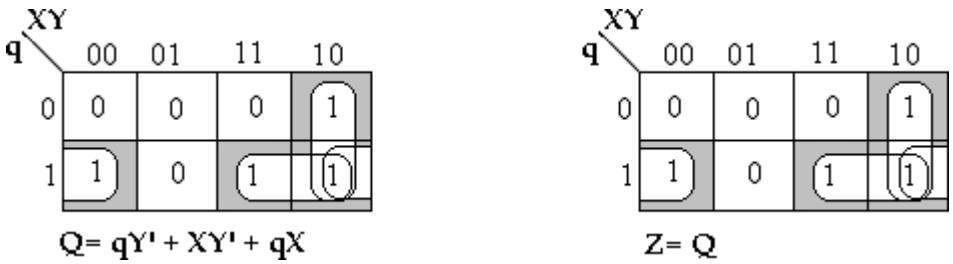
Here we assign a common letter symbol to all the stable states in each merged row. Thus, the symbol B & D is replaced by A; E & F are replaced by C.

	XY 00	01	11	10
A	(A),0	(A),0	(A),0	C,–
C	(C),1	A,–	(C),1	(C),1

Step 5:

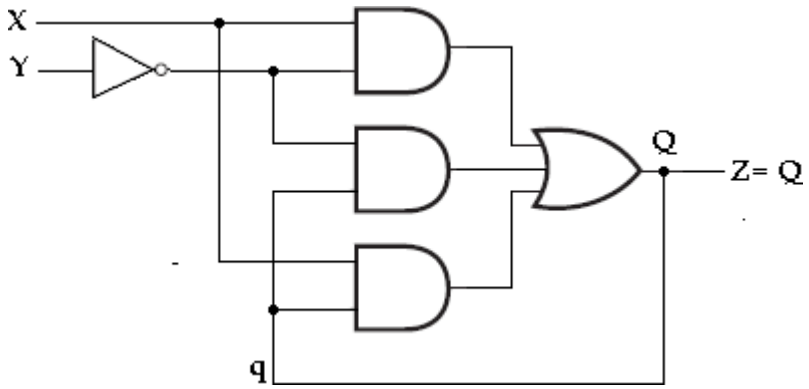
Find the race-free binary assignment for the four stable states in the reduced flow table. Assign A= 0 and C= 1

Substituting the binary assignment into the reduced flow table, the transition table is obtained. The output map is obtained from the reduced flow table.



Transition table and output map

Step 6:



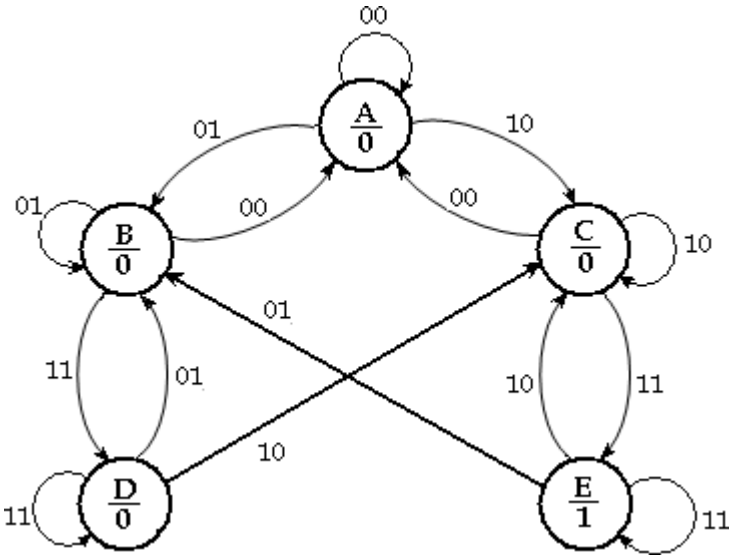
Gated-Latch Logic diagram

4. Design a circuit with inputs X and Y to give an output Z= 1 when XY= 11 but only if X becomes 1 before Y, by drawing total state diagram, primitive flow table and output map in which transient state is included.

Soln:

Step 1:

The state diagram can be drawn as,



State table

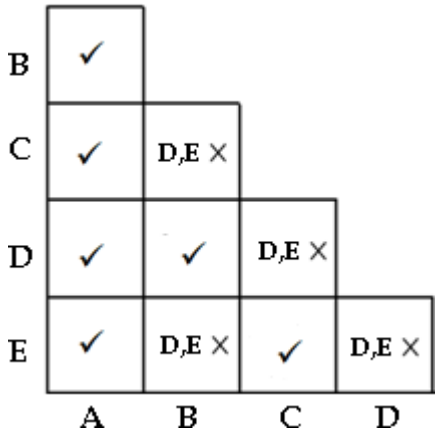
A primitive flow table is constructed from the state table as,

	XY 00	01	11	10
A	(A),0	B,–	–,–	C,–
B	A,–	(B),0	D,–	–,–
C	A,–	–,–	E,–	(C),0
D	–,–	B,–	(D),0	C,–
E	–,–	B,–	(E),1	C,–

Primitive flow table

Step 3:

The rows in the primitive flow table are merged by first obtaining all compatible pairs of states. This is done by means of the implication table.

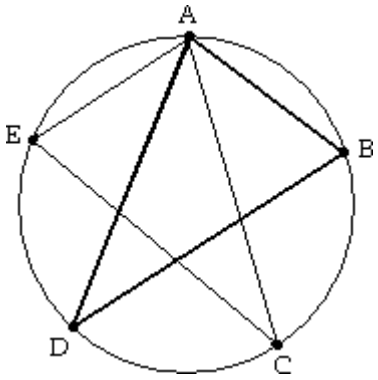


The squares that contain the check marks () define the compatible pairs: (A, B) (A, C) (A, D) (A, E) (B, D) (C, E)

Step 4:

The merger diagram is obtained from the list of compatible pairs derived from the implication table. There are six straight lines connecting the dots, one for each compatible pair. The lines form a geometrical pattern consisting of one triangle connecting (A, B, D) & a line (C, E). The maximal compatibles are:

(A, B, D) (C, E)



The reduced flow table is drawn as below.

	XY 00	01	11	10
A, B, D	(A), 0	(B), 0	(D), 0	C, –
C, E	A, –	B, –	(E), 1	(C), 0

Reduced flow table

Here we assign a common letter symbol to all the stable states in each merged row. Thus, the symbol B & D is replaced by A; E is replaced by C.

	XY 00	01	11	10
A	(A), 0	(A), 0	(A), 0	C, –
C	A, –	A, –	(C), 1	(C), 0

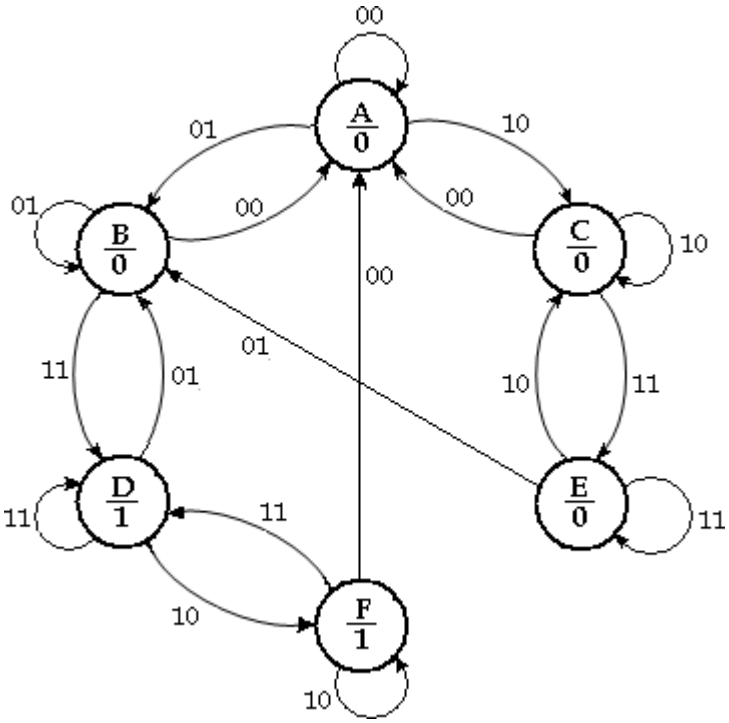
Transition table

5. Design a circuit with primary inputs A and B to give an output Z equal to 1 when A becomes 1 if B is already 1. Once Z= 1 it will remain so until A goes to 0. Draw the total state diagram, primitive flow table for designing this circuit.

Soln:

Step 1:

The state diagram can be drawn as,



State diagram

A primitive flow table is constructed from the state table as,

	AB 00	01	11	10
A	(A), 0	B, –	–, –	C, –
B	A, –	(B), 0	D, –	–, –
C	A, –	–, –	E, –	(C), 0
D	–, –	B, –	(D), 1	F, –
E	–, –	B, –	(E), 0	C, –
F	A, –	–, –	D, –	(F), 1

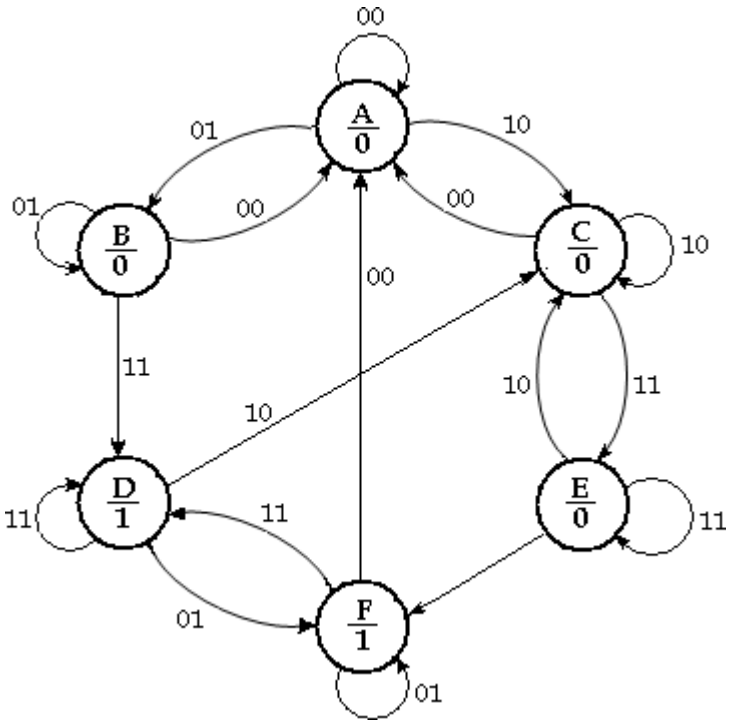
Primitive flow table

6. Design an asynchronous sequential circuit that has two inputs X_2 and X_1 and one output Z. When X_1= 0, the output Z is 0. The first change in X_2 that occurs while X_1 is 1 will cause output Z to be 1. The output Z will remain 1 until X_1 returns to 0.

Soln:

Step:1

State diagram



Step : 2

A primitive flow table is constructed from the state table as,

	x_2x_1 = 00	01	11	10
A	(A), 0	B, –	–, –	C, –
B	A, –	(B), 0	D, –	–, –
C	A, –	–, –	E, –	(C), 0
D	–, –	F, –	(D), 1	C, –
E	–, –	F, –	(E), 0	C, –
F	A, –	(F), 1	D, –	–, –

Primitive flow table

Step :3

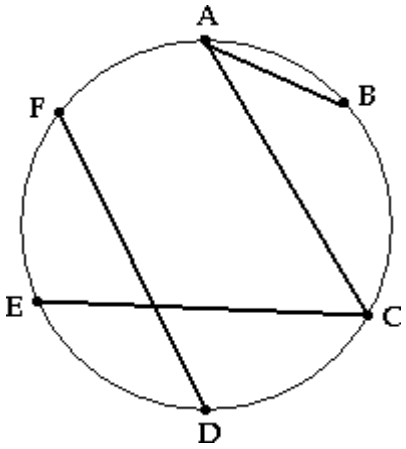
The rows in the primitive flow table are merged by obtaining all compatible pairs of states. This is done by means of the implication table

	A	B	C	D	E
B	✓				
C	✓	D,E ×			
D	B,F ×	B,F ×	D,F ×		
E	B,F ×	B,F × D,E ×	✓	D,E ×	
F	B,F ×	B,F ×	D,E ×	✓	D,E ×

The squares that contain the check marks (✓) define the compatible pairs: (A, B) (A, C) (C, E) (D, F)

Step 4:

The merger diagram is obtained from the list of compatible pairs derived from the implication table. There are four straight lines connecting the dots, one for each compatible pair. It consists of four lines (A, B), (A, C), (C, E) and (D, F).



The maximal compatibles are:

Merger diagram

(A, B) (C, E) (D, F)

This set of maximal compatible covers all the original states resulting in the reduced flow table.

The reduced flow table is drawn as below

	X_2X_1 00	01	11	10
A,B	(A),0	(B),0	D,–	C,–
C,E	A,–	F,–	(E),0	(C),0
D,F	A,–	(F),1	(D),1	C,–

Flow table

Here we assign a common letter symbol to all the stable states in each merged row. Thus, the symbol B is replaced by A; E is replaced by C and F is replaced by D.

Step 5:

Reduced Flow table

	X_2X_1 00	01	11	10
A	(A),0	(A),0	D,–	C,–
C	A,–	D,–	(C),0	(C),0
D	A,–	(D),1	(D),1	C,–

Find the race-free binary assignment for the four stable states in the reduced

flow table. Assign A= S0, C= S1 and D= S2.

Now, if we assign S0= 00, S1 = 01 and S2 = 10, then we need one more state S3= 11 to prevent critical race during transition of S0 to S1 or S2 to S1. By introducing S3 the transitions S1 to S2 and S2 to S1 are routed through S4.
Thus after state assignment the flow table can be given as

Present State	Next state for Inputs X_2X_1, Output			
F_2F_1	00	01	11	10
$S_0 \rightarrow 00$	(S0), 0	(S0), 0	S2, –	S1, –
$S_1 \rightarrow 01$	S0, –	S3, –	(S1), 0	(S1), 0
$S_2 \rightarrow 10$	S0, –	(S2), 1	(S2), 1	S3, –
$S_3 \rightarrow 11$	–, –	S2, –	–, –	S1, –

Flow table with state assignment

Substituting the binary assignment into the reduced flow table, the transition table is obtained. The output map is obtained from the reduced flow table

Present State		Next state for Inputs X_2X_1, Output			
F_2	F_1	00	01	11	10
0	0	(00), 0	(00), 0	10, –	01, –
0	1	00, –	11, –	(01), 0	(01), 0
1	0	00, –	(10), 1	(10), 1	11, –
1	1	–, –	10, –	–, –	01, –

K- Map simplification:

For F_2^+

F_2F_1 \ X_2X_1	00	01	11	10
00	0	0	1	0
01	0	1	0	0
11	X	1	X	0
10	0	1	1	1

$F_2^+ = \overline{F}_1X_2X_1 + F_1\overline{X}_2X_1 + F_2X_1 + F_2\overline{F}_1X_2$

For F_1^+

F_2F_1 \ X_2X_1	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	X	0	X	1
10	0	0	0	1

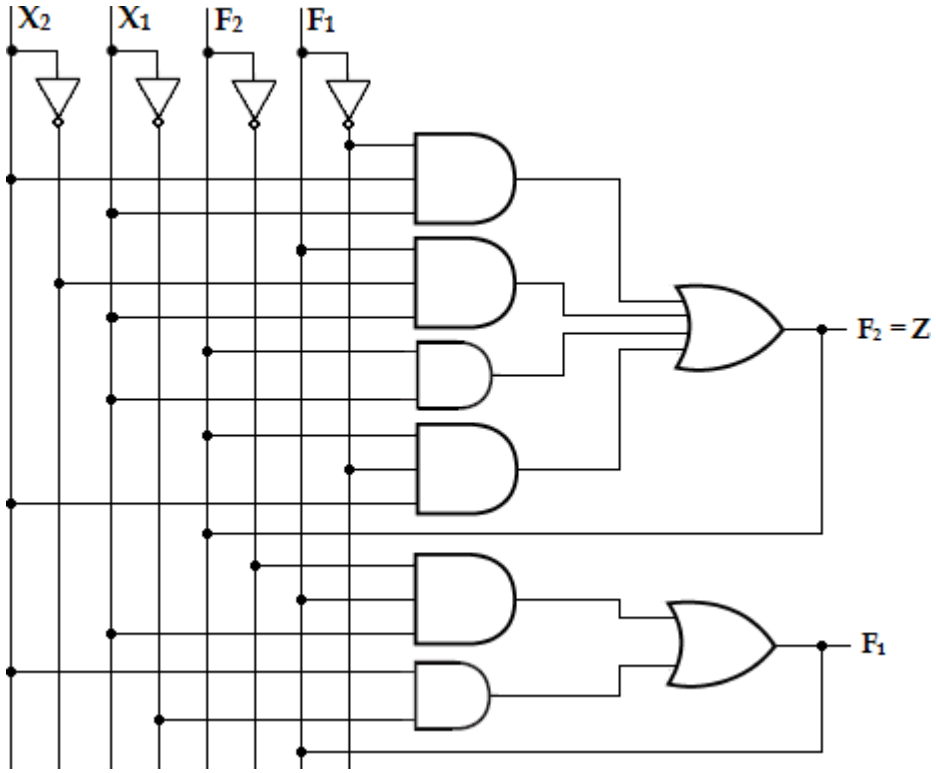
$F_1^+ = \overline{F}_2F_1X_1 + X_2\overline{X}_1$

For Z

F_2F_1 \ X_2X_1	00	01	11	10
00	0	0	X	X
01	X	X	0	0
11	X	X	X	X
10	X	1	1	X

$Z = F_2$

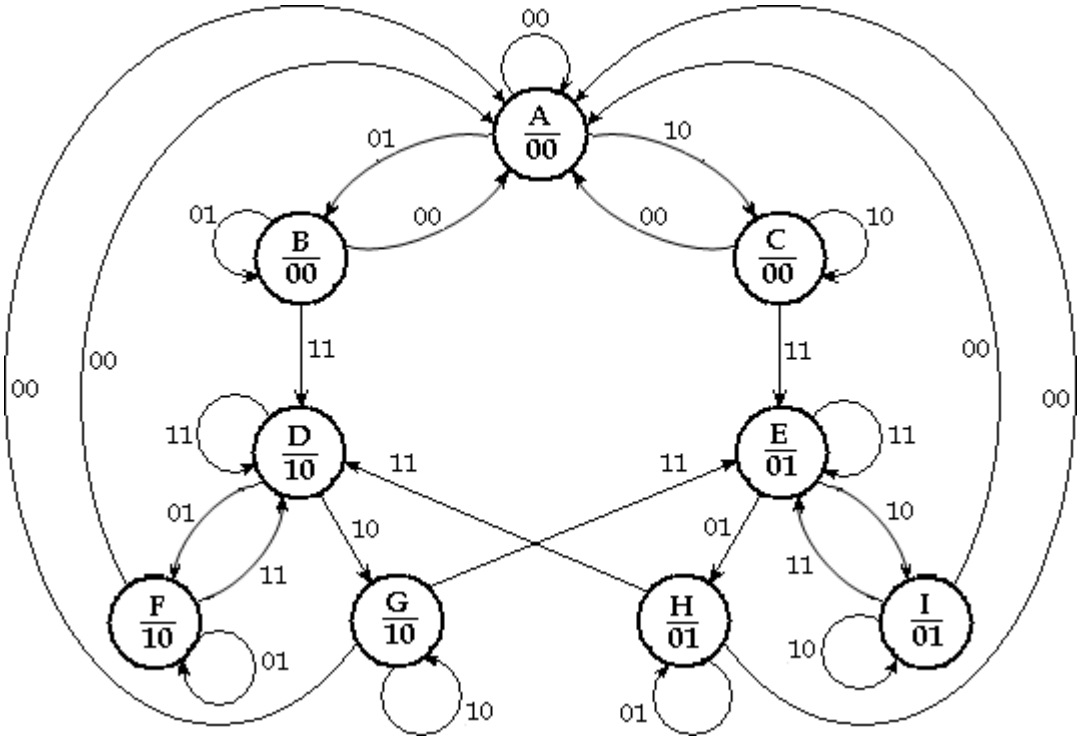
Logic Diagram:



7. Obtain a primitive flow table for a circuit with two inputs x_1 and x_2 and two outputs z_1 and z_2 that satisfies the following four conditions.
   i. When $x_1x_2 = 00$, output $z_1z_2 = 00$.
   ii. When $x_1 = 1$ and x_2 changes from 0 to 1, the output $z_1z_2 = 01$.
   iii. When $x_2 = 1$ and x_1 changes from 0 to 1, the output $z_1z_2 = 10$.
   iv. Otherwise the output does not change.

Soln:

The state diagram can be drawn as



Step 2: A primitive flow table is constructed from the state table as,

	X_1X_2 = 00	01	11	10
A	(A), 00	B, –	–, –	C, –
B	A, –	(B), 00	D, –	–, –
C	A, –	–, –	E, –	(C), 00
D	–, –	F, –	(D), 10	G, –
E	–, –	H, –	(E), 01	I, –
F	A, –	(F), 10	D, –	–, –
G	A, –	–, –	E, –	(G), 10
H	A, –	(H), 01	D, –	–, –
I	A, –	–, –	E, –	(I), 01