Unit 4

Asynchronous Sequential Logic

Syllabus - Analysis and Design of Asynchronous Sequential Circuits - Reduction of State and Flow Tables - Race free State Assignment - Hazards

* Asynchronous Sequential Circuits

→ In synchronous sequential circuits, the change of internal state occurs in response to synchronized clock pulses.

→ Asynchronous sequential circuits do not use clock pulses, but rather there is a change in the internal state, when there is a change in the input variables.

→ The memory elements in asynchronous sequential circuits are unclocked flip flops or time-delay elements.

→ An asynchronous sequential circuit is similar to a combinational circuit with feedback.

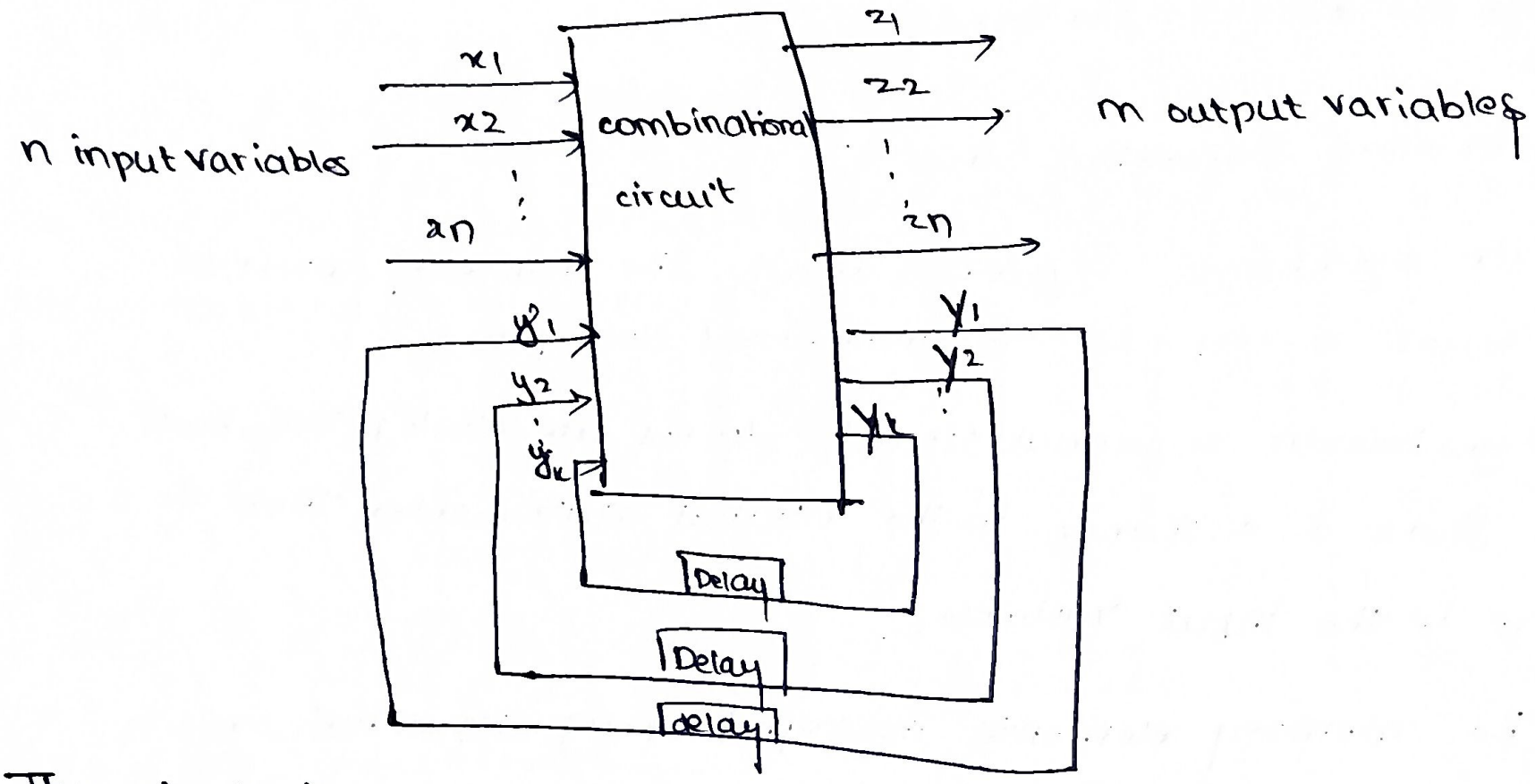
→ The design of these circuits is much more difficult because of the timing problems involved in the feedback path.

→ Care must be taken to ensure that each new state keeps the circuit in a stable condition even though a feedback path exists.

* Applications

→ used when speed of operation is important, especially where the digital system must respond quickly w/o having to wait for a clock pulse.

* Block Diagram



The circuit has:

n input variables, m output variables and k internal states.

→ The delay element is visualized by providing a short-term memory for the sequential circuit

The present state variables are called **secondary variables**

The next state variables are called **excitation variables**.

Working of the circuit

→ When an input variables changes in value, the y secondary variables do not change instantaneously.

→ It takes a certain amount of time for the signal to

propagate from the input terminals, through the combinational circuit, to the y excitation variables.

* Conditions for Stability

→ To ensure the stability of the circuit,

$$y_i = Y_i$$

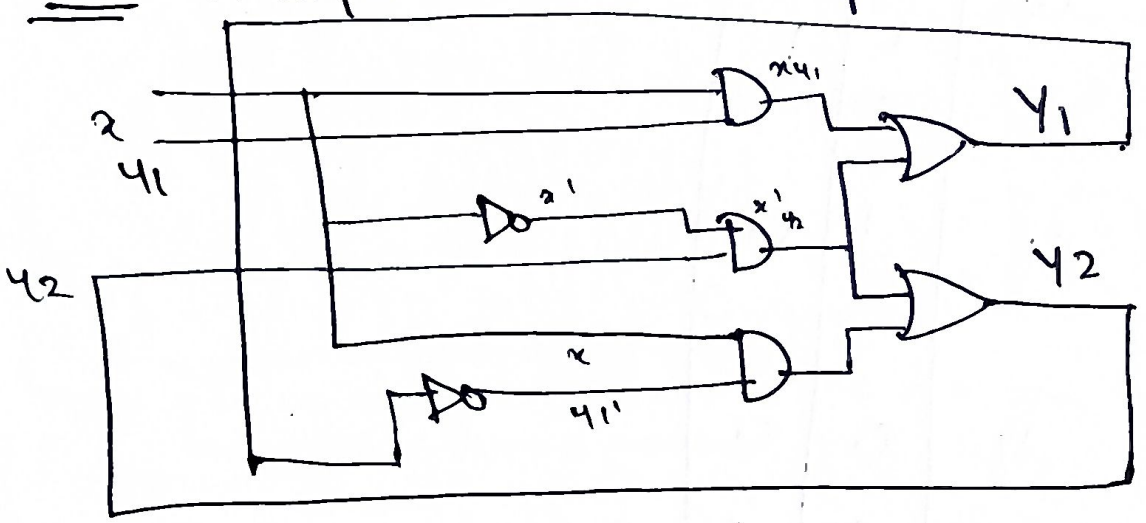
i^{th} secondary variable = i^{th} excitation variable

→ The circuit must be allowed to attain a stable state before the input is changed into a new value.

→ The simultaneous change of 2 or more variables is prohibited.

* Analysis Procedure

Ex1 Analyze the following asynchro



Step 1: Obtain the Boolean expressions of the excitation variables as a function of the input & secondary vars (present state)

$$Y_1 = x y_1 + x' y_2$$

$$Y_2 = x y_1' + x' y_2$$

Step2: Derive the values of Y_1 and Y_2 from the Boolean expressions to fill in the map

Present state		Input						Y_1	Y_2
y_1	y_2	x	x'	xy_1	$x'y_2$	Y_1'	xY_1'	$xY_1 + x'Y_2$	$xY_1' + x'Y_2$
0	0	0	1	0	0	1	0	0	0
0	0	1	0	0	0	1	1	0	1
0	1	0	1	0	1	1	0	1	1
0	1	1	0	0	0	1	1	0	1
1	0	0	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	1	0
1	1	0	1	0	1	0	0	1	1
1	1	1	0	1	0	0	0	1	0

Step3: Draw maps for the excitation variables

Y_1Y_2 \ x	0	1
00	0 (0)	0 (1)
01	1 (2)	0 (3)
11	1 (6)	1 (7)
10	0 (4)	1 (5)

$Y_1 = xY_1 + x'Y_2$

Y_1Y_2 \ x	0	1
00	0 (0)	1 (1)
01	1 (2)	1 (3)
11	1 (6)	0 (7)
10	0 (4)	0 (5)

$Y_2 = xY_1' + x'Y_2$

Step4: Transition table, and identification of stable states.

y_1y_2 \ x	0	1
00	(00)	01
01	11	(01)
11	(11)	10
10	00	(10)

Step 5: Constructing a flow table

A flow table is similar to a transition table, but for the fact that the internal states are symbolized by letters rather than binary numbers.

Here, let

a : 00
b : 01
c : 11
d : 10

y_1y_2 \ x	0	1
00	(a)	b
01	c	(b)
11	(c)	d
10	a	(d)

Note: Primitive Flow Table : flow table has only stable state per raw.

Eg? Analyze the asynchronous sequential circuit, which are expressed by the following ~~Boolean expressions~~ flow table

$Y = \overline{x + xz}$

(Case to convert a flow table into a circuit)

Flow Table

	x_1x_2 00	01	11	10
a	(a), 0	(a), 0	(a), 0	b, 0
b	a, 0	a, 0	(b), 1	(b), 0

Step 1: ~~Flow table into transition~~

Assign boolean values for alphabets

x_1x_2	00	01	11	10
a (0)	(0), 0	(0), 0	(0), 0	1, 0
b (1)	0, 0	0, 0	(1), 1	(1), 0

$a = 0$

$b = 1$

Step 2: Split flow table into transition table and output map

Transition Table

x_1x_2	00	01	11	10
0	(0)	(0)	(0)	1
1	0	0	(1)	(1)

Output Map

x_1x_2	00	01	11	10
0	0	0	0	0
1	0	0	1	0

Step 3: Group stable states to obtain boolean expression

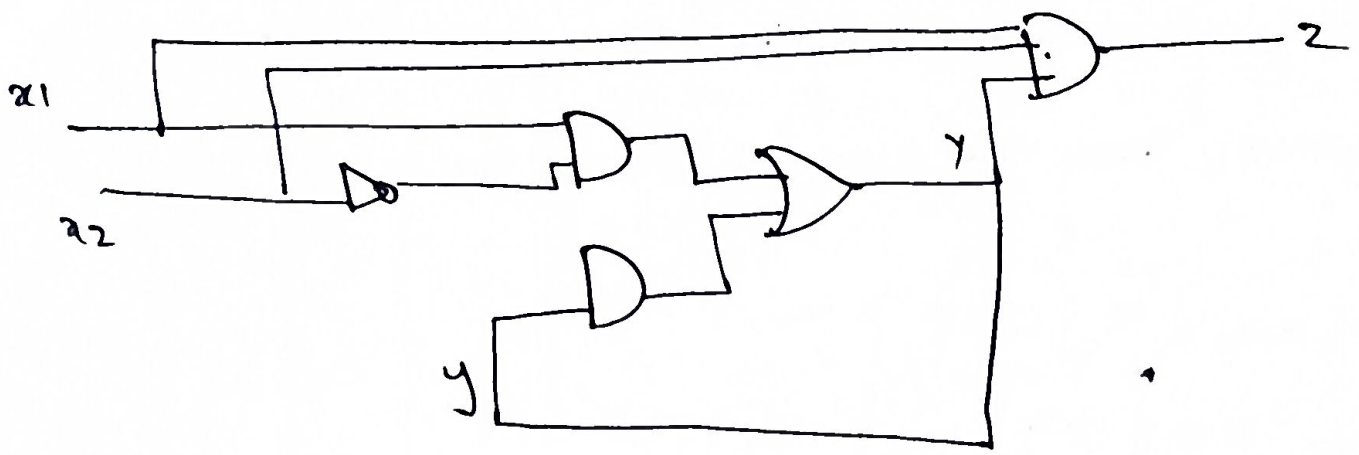
	00	01	11	10
0	0	0	0	1
1	0	0	1	1

$Y = x_1x_2' + x_1y$

	00	01	11	10
0	0	0	0	0
1	0	0	1	0

$Z = x_1x_2y$

Step 4: Draw the corresponding circuit



* Race Conditions

→ A race condition is said to exist in an asynchronous sequential circuit when 2 or more binary state variables change in response to a change in an input variable.

→ When there are unequal delays, a race condition may cause the state variables to change in an unpredictable manner.

Non-Critical Race : If the final stable state that the circuit reaches does not depend on the order in which the state variables change.

Foreg

(i)

y_1y_2 \ x	0	1
00	(00)	11
01		11
11		(11)
10		11

possible transitions

00 → 11

00 → [01] → 11

00 → [10] → 11

□ indicates an unstable state.

(ii)

y_1y_2 \ x	0	1
00	(00)	11
01		01
11		(01)
10		11

possible transitions

00 → 01

00 → [10] → [11] → 01

00 → 11 → 01

What I understand ?

In an unstable state, only a single bit can change

Critical Race : If it is possible to end up in two or more different stable states, depending on the order in which the state variables change, then the race is said to be a critical race.

y_1y_2 \ x	0	1
00	(00)	11
01		(01)
11		(11)
10		(10)

possible transitions

$00 \rightarrow 01$

$00 \rightarrow 10$

$00 \rightarrow 11$

y_1y_2 \ x	0	1
00	(00)	11
01		11
11		(11)
10		(10)

$00 \rightarrow 11$

$00 \rightarrow 10$

$00 \rightarrow \boxed{01} \rightarrow 11$

* Avoiding races

→ By making a proper binary assignment to the state variables

→ The circuit can be redirected through a unique sequence of unstable states, back to a stable state. This is called a **cycle.**

Example :

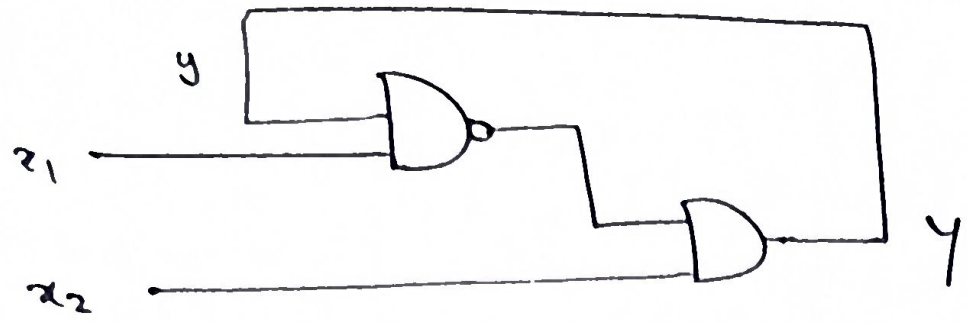
y_1y_2 \ x	0	1
00	(00)	01
01		11
11		10
10		(10)

$00 \rightarrow \boxed{01} \rightarrow \boxed{11} \rightarrow 10$

Note : cycles should end in a stable state, otherwise will oscillate infinitely

Ex3 in Analysis (including stability considerations)

Given Circuit :



Step1: Boolean exp

$$Y = (x_1 y)' \cdot x_2$$

$$= (x_1' + y')x_2$$

$$\boxed{Y = x_1'x_2 + y'x_2}$$

Step2: obtaining values for excitation variable

y	x_1	x_2	x_1'	y'	$x_1'x_2$	$y'x_2$	$x_1'x_2 + y'x_2$ (Y)
0	0	0	1	1	0	0	0
0	0	1	1	1	1	1	1
0	1	0	0	1	0	0	0
0	1	1	0	1	0	1	1
1	0	0	1	0	0	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0

Step3: Map

y \ x_1x_2	00	01	11	10
0	0 (0)	1 (1)	1 (3)	0 (2)
1	0 (4)	1 (5)	0 (7)	0 (6)

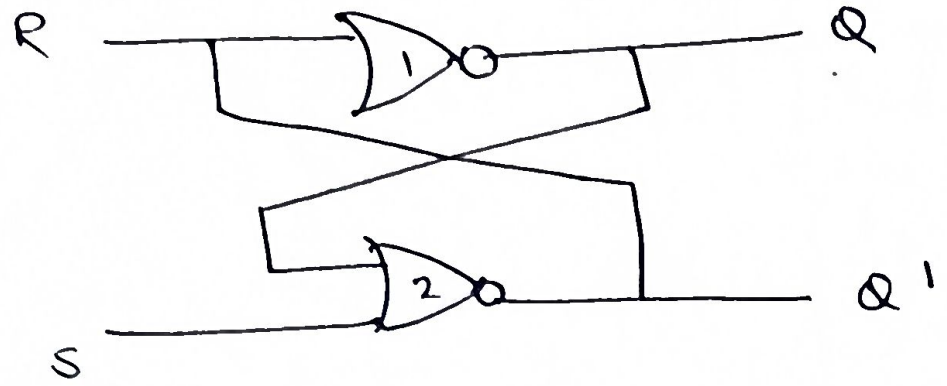
Step4: Identify stable state

y \ x_1x_2	00	01	11	10
0	0	1	1	0
1	0	1	0	0

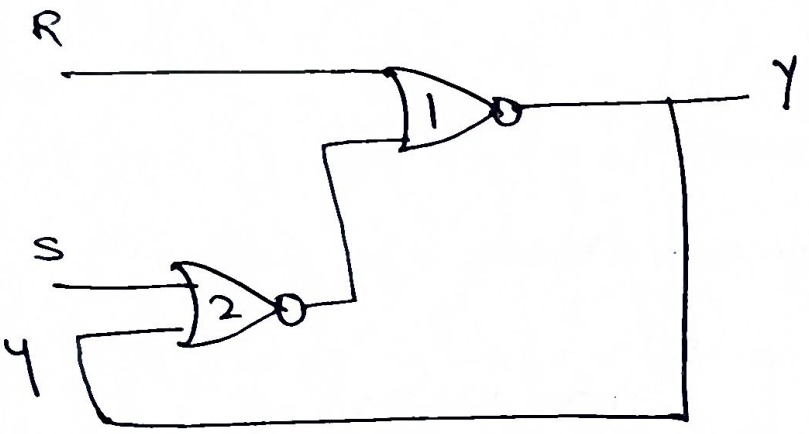
Note that when $x_1x_2 = 11$, there is no stable state. $\Rightarrow$ y & Y are diff.
If y=0, then Y=1, which causes a transition to the second row, and back again. The state variable oscillates indefinitely

* Analysis of Asynchronous Sequential Circuit with Latches

→ Consider an SR - Latch made from 2 cross-coupled NOR gates.



→ Redrawing the circuit



Step1 : Obtaining the Boolean expression

$$Y = ((S+Y)' + R)'$$

$$Y = \cancel{S'Y' + R} \quad (S+Y).R'$$

$$\boxed{Y = SR' + R'Y}$$

Truth table of SR Latch

S R	Q Q'
0 1	0 1
0 0	0 1
1 0	1 0
0 0	1 0
1 1	0 0

forbidden state

→ In order to prevent the forbidden state from occurring, the condition $SR = 0$ can be set.

$$Y = SR' + R'y$$

SR' can be rewritten as:

$SR' + 0$

$SR' + SR = S(R + R')$

$= S$

$$\Rightarrow \boxed{Y = S + R'y} \quad \text{when } SR = 0$$

Step2: Obtaining the excitation & variable's values

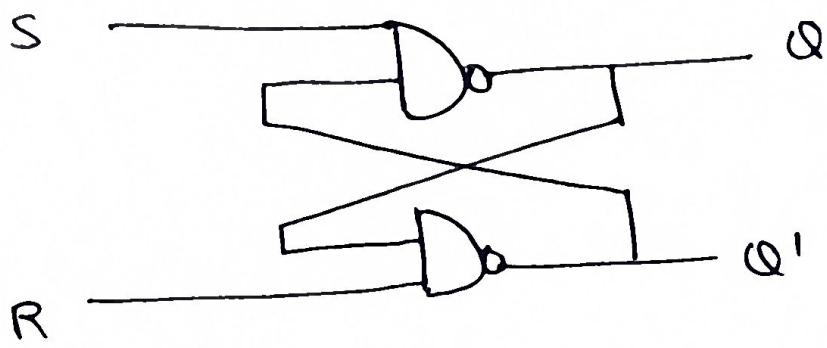
y	S	R	SR'	R'	R'y	SR' + R'y
0	0	0	0	1	0	0
0	0	1	0	0	0	0
0	1	0	1	1	1	1
0	1	1	0	0	0	0
1	0	0	0	1	1	1
1	0	1	0	0	0	0
1	1	0	1	1	1	1
1	1	1	0	0	0	0

Step3: Transition Table

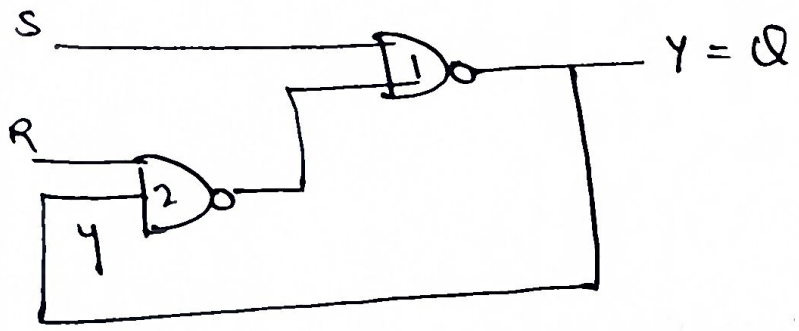
y \ SR	00	01	11	10
0	0 (0)	0 (1)	0 (3)	1 (2)
1	1 (4)	0 (5)	0 (7)	1 (6)

Analysis using an 'SR' Latch

→ Consider an SR latch implemented with NAND gates



→ ⓑ Redrawing the circuit



Step 1: Obtaining the Boolean function.

$$Y = ((Ry)'.S)'$$

$$Y = Ry + S'$$

$$\boxed{Y = S' + Ry}$$

Truth Table

S	R	Q	Q'
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1

→ forbidden

To avoid the forbidden state, the condition to be applied is $\boxed{S'R' = 0}$

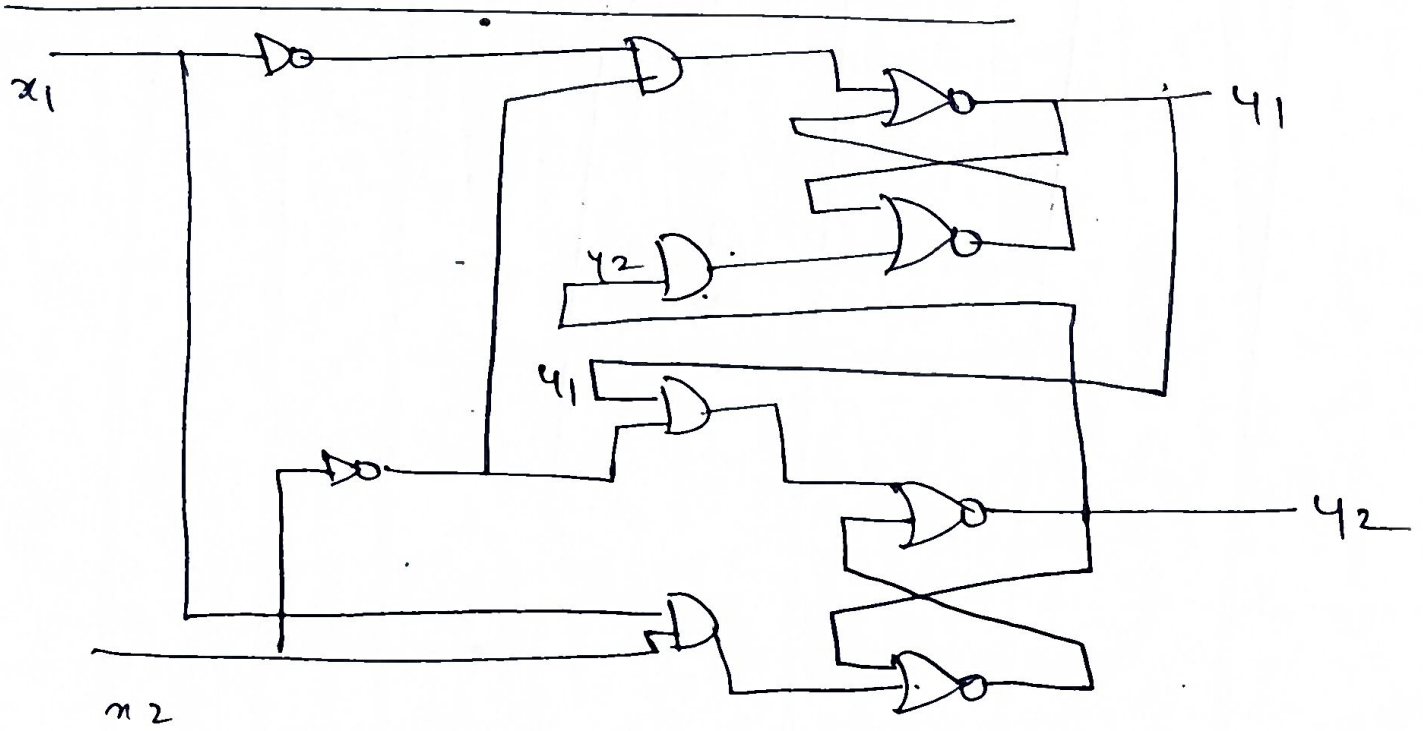
Step2: Obtaining the excitation variable's values

y	S	R	S'	Ry	y S'+Ry
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	1	1

Step3: Drawing the transition table

y \ SR	00	01	11	10
0	1 (0)	1 (1)	0 (3)	0 (2)
1	1 (4)	1 (5)	1 (7)	0 (6)

* Example for Analysis using an SR Latch



Step 1 : Obtaining the boolean expressions for the latch inputs and the excitation variable

$S_1 = x_1 y_2$ $\quad S_2 = x_1 x_2$

$R_1 = x_1' x_2'$ $\quad R_2 = x_2' y_1$

$Y_1 = S_1 + R_1' y_1$

$Y_2 = S_2 + R_2' y_2$

~~$Y_1 = x_1 y_2 + (x_1' x_2') y_1$~~

$Y_1 = x_1 y_2 + (x_1 + x_2) y_1$

~~$Y_1 = x_1 y_2 + x_1' y_1 + x_2' y_1$~~

$$\boxed{Y_1 = x_1 y_2 + x_1 y_1 + x_2 y_1}$$

$Y_2 = x_1 x_2 + (x_2 + y_1') y_2$

$$\boxed{Y_2 = x_1 x_2 + x_2 y_2 + y_1' y_2}$$

Step 2: Compute Y_1 and Y_2 values. (Draw correct table)

Inputs come up

y_1	y_2	x_1	x_2	S_1 $x_1 y_2$	S_2 $x_1 x_2$	R_1 $x_1' x_2'$	R_2 $x_2' y_1$	x_1'	x_2'
0	0	0	0	0	0	1	0	1	1
0	0	0	1	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0
0	1	0	0	0	0	1	0	1	1
0	1	0	1	0	0	0	0	1	0
0	1	1	0	1	0	0	0	0	1
0	1	1	1	1	1	0	0	0	0
1	0	0	0	0	0	1	1	1	1
1	0	0	1	0	0	0	0	1	0
1	0	1	0	0	0	0	1	0	1
1	0	1	1	0	1	0	0	0	0
1	1	0	0	0	0	1	1	1	1
1	1	0	1	0	0	0	0	1	0
1	1	1	0	1	0	0	1	0	1
1	1	1	1	1	1	0	0	0	0

Step 3 : Draw a map, and identify stable states

y_1y_2 \ x_1x_2	00	01	11	10
00	(00)	(00)	01	(00)
01	(01)	(01)	11	11
11	00	(11)	(11)	10
10	00	10	11	(10)

* **Latch Excitation Table**

A table that lists the required inputs for S and R, for each of the possible transitions to y to Y.

for an SR Latch, the truth table is :

$Y = S + R'y$

$Y = S + R'y$.

S	R	R'	Present State (y)	Next State
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	X
1	1	0	1	X

Excitation Table

y	Y	S R
0	0	0 X
0	1	1 0
1	0	0 1
1	1	X 0

* Implementation

→ process of making a sequential circuit given the transition table

The transition table is:

y \ x1x2	00	01	11	10
0	(0)	(0)	(0)	1
1	0	0	(1)	(1)

Step 2 Derive boolean expressions for S & R, after drawing a map

for the S input

y \ x1x2	00	01	11	10
0	0	0	0	1
1	0	0	X	X

y	x1	x2
0	1	0
1	1	0

$S = x_1 x_2'$

excitation table

y	Y	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

For the R input

y \ x1x2	00	01	11	10
0	X	X	X	0
1	1	1	0	0

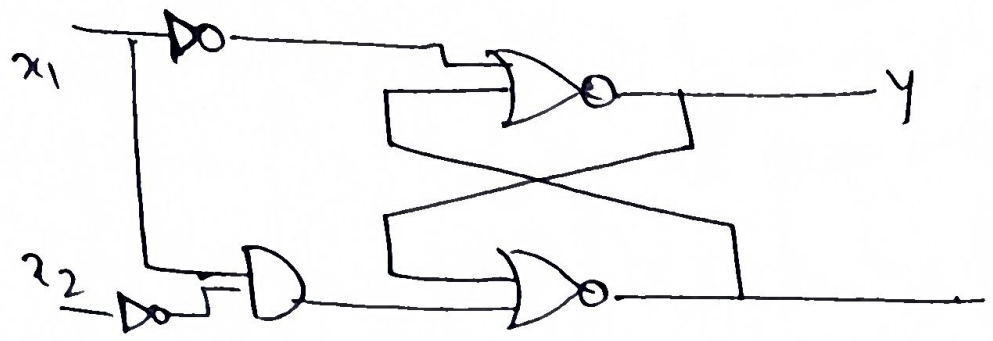
y	x1	x2
0	0	0
0	0	1
1	0	0
1	0	1

x_1'

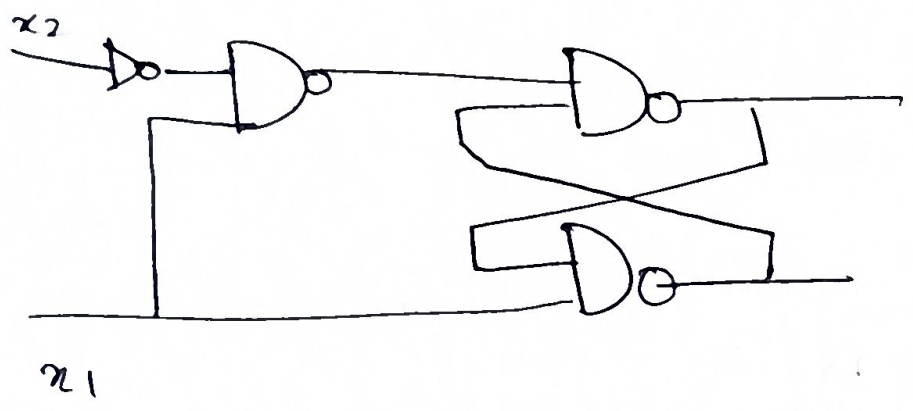
$R = x_1'$

Step 3: Draw the logic circuit

(i) with a NOR latch



(ii) with a NAND latch



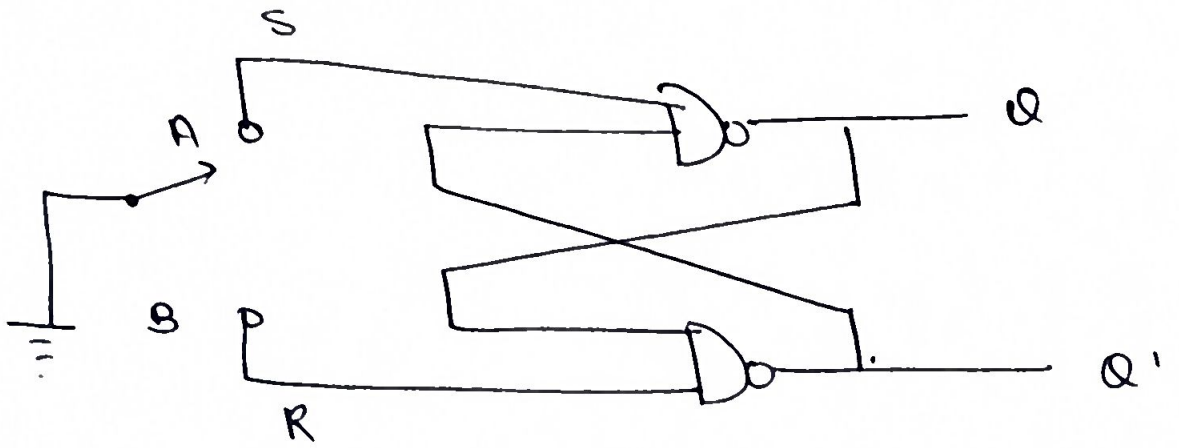
* Debounce Circuit

→ A debounce circuit is a circuit which removes the series of pulses that result from a contact bounce, and produces a single smooth transition of the binary signal from 0 to 1 or 1 to 0.

→ This circuit can be made by connecting a single pole, double throw switch to an SR latch.

→ The contact center is connected to the ground, and provides a signal equal to logic 0.

→ When one of the 2 contacts A or B is not connected to the ground through the switch, it behaves like a logic 1 signal



Working

When the switch rests in position A

$\Rightarrow S = 0$

$R = 1$

$Q = 1$

$Q' = 0$

When the switch is moved to position B,

R goes to 0.

$S = 1$

$Q = 0$

$Q' = 1$

After the switch makes an initial contact with B, it bounces several times.

Even if there is bouncing the latch o/p would not be affected due to the fact that there is never complete contact w/ the ground.

When the switch goes back to A, $S = 0$ and $Q = 1$

There will again be a smooth transition of the o/p even if there are contact bounces.

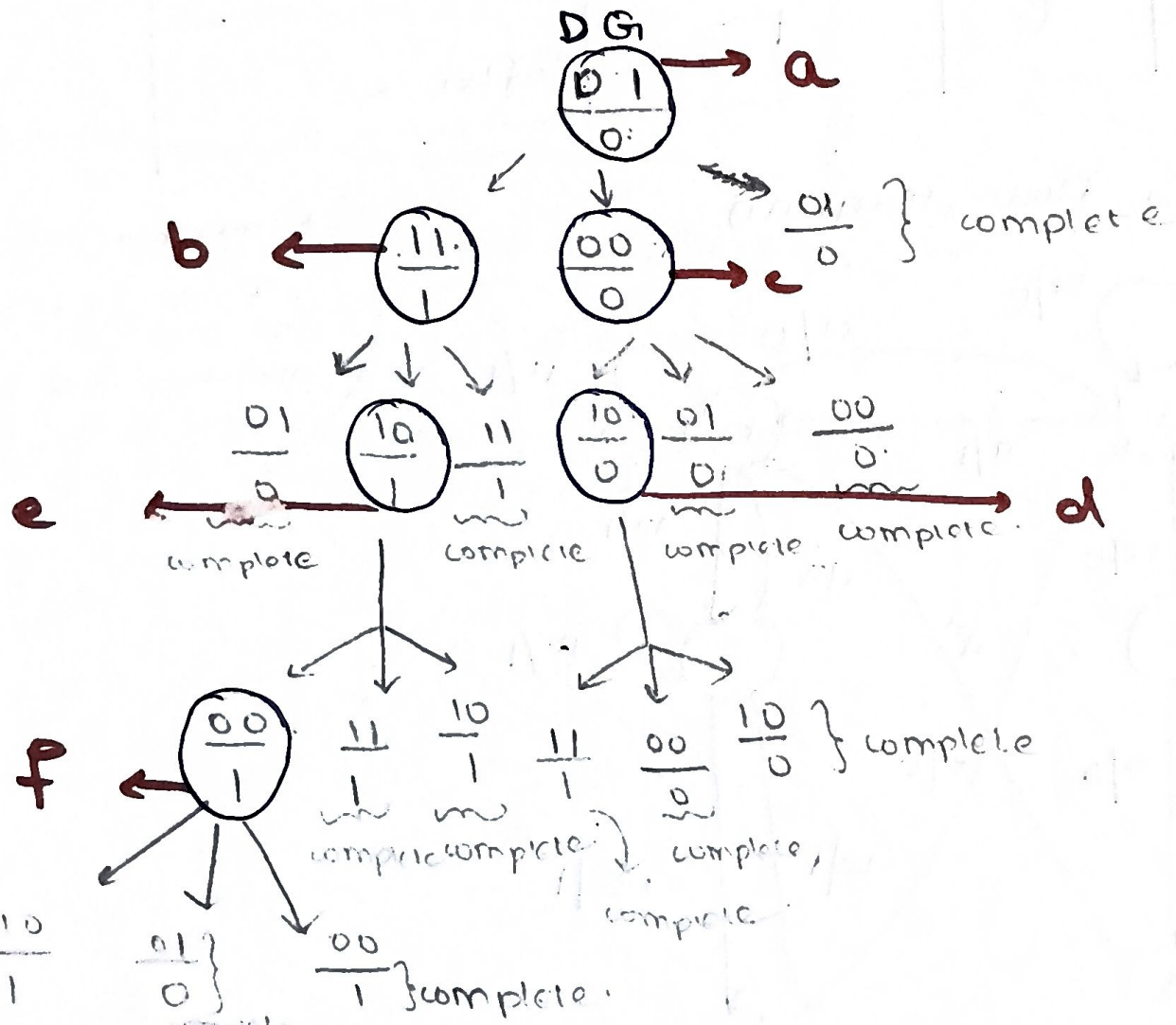
* Design Procedure

Q. Design a gated latch circuit with 2 i/p gates G and D and 1 o/p Q. The information present at D is transferred to Q when G = 1. The Q output will follow the D input as long as G = 1. When G goes to 0, the information present at the D input, at the time the transition occurred is retained.

Ans. Two inputs : G & D

G = 1 ⇒ o/p = D

G = 0 ⇒ o/p = previous state

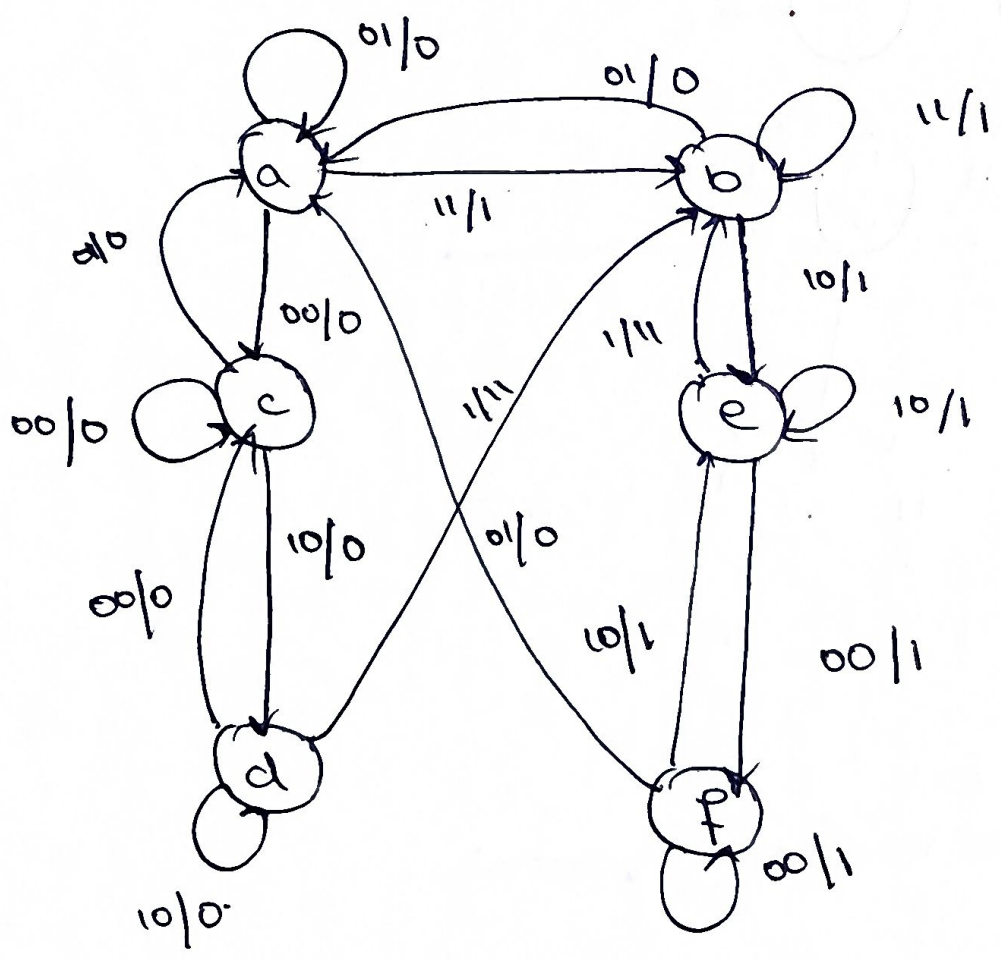


Assign each unique transition an alphabet.

Step 3: Consolidate flow diagram into a state table

State	Inputs		Outputs	Comments
	D	G	Q	
a	0	1	0	
b	1	1	1	
c	0	0	0	After a or d
d	1	0	0	after c
e	1	0	1	after b or f
f	0	0	1	after e

Step 2: Draw a flow diagram



Remember as cf adeb and no 3 or 6

Step 4: Draw a primitive flow table

		00	01	11	10
01	a	c, -	(a, 0)	b, -	-, -
11	b	-, -	a, -	(b, 1)	e, -
00	c	(c, 0)	a, -	-, -	d, -
10	d	c, -	-, -	b, -	(d, 0)
10	e	f, -	-, -	b, -	(e, 1)
00	f	(f, 1)	a, -	-, -	e, -

Note: 2 bit changes not allowed, use -, - in such cases

Look at comments in the state table

(i) f comes after e, and not c

(ii) e does not come after d, so d is the other possibility

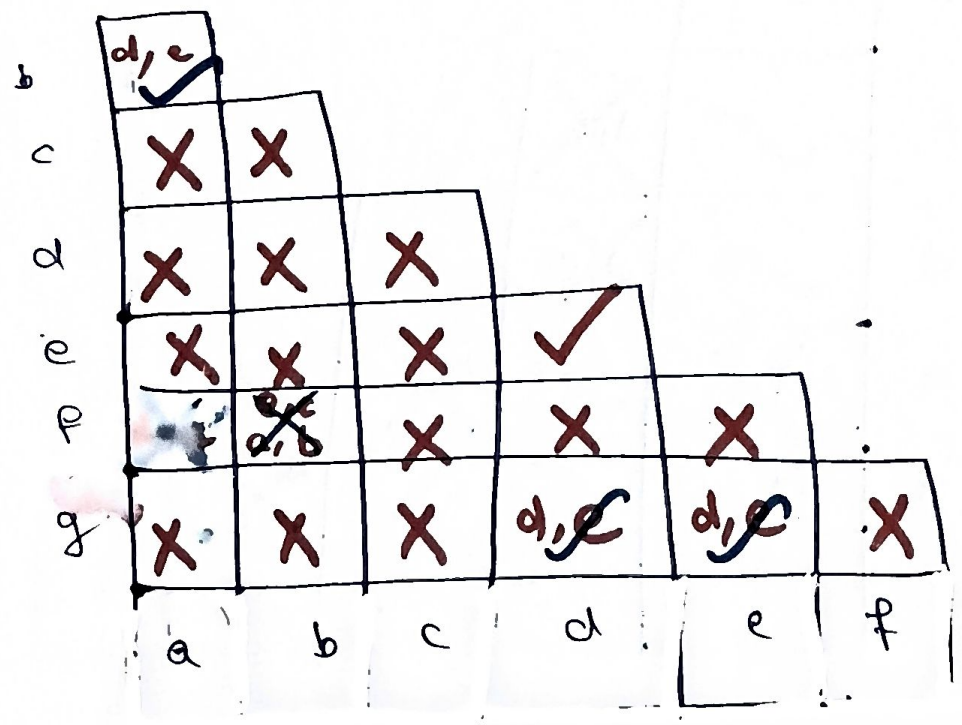
* Reduction of State

Construction of implication tables

Ex1 Construct an implication table and reduce the states for the following state table.

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
a	d	b	0	0
b	e	a	0	0
c	g	f	0	1
d	a	d	1	0
e	a	d	1	0
f	c	b	0	0
g	a	e	1	0

Step1:



Step2

Write when equivalent states occur

(a,b), (d,g), (e,d), (e,g)

⇒ (a,b), (deg)

∴ The final grouping becomes

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

Step3 : Merge flow table

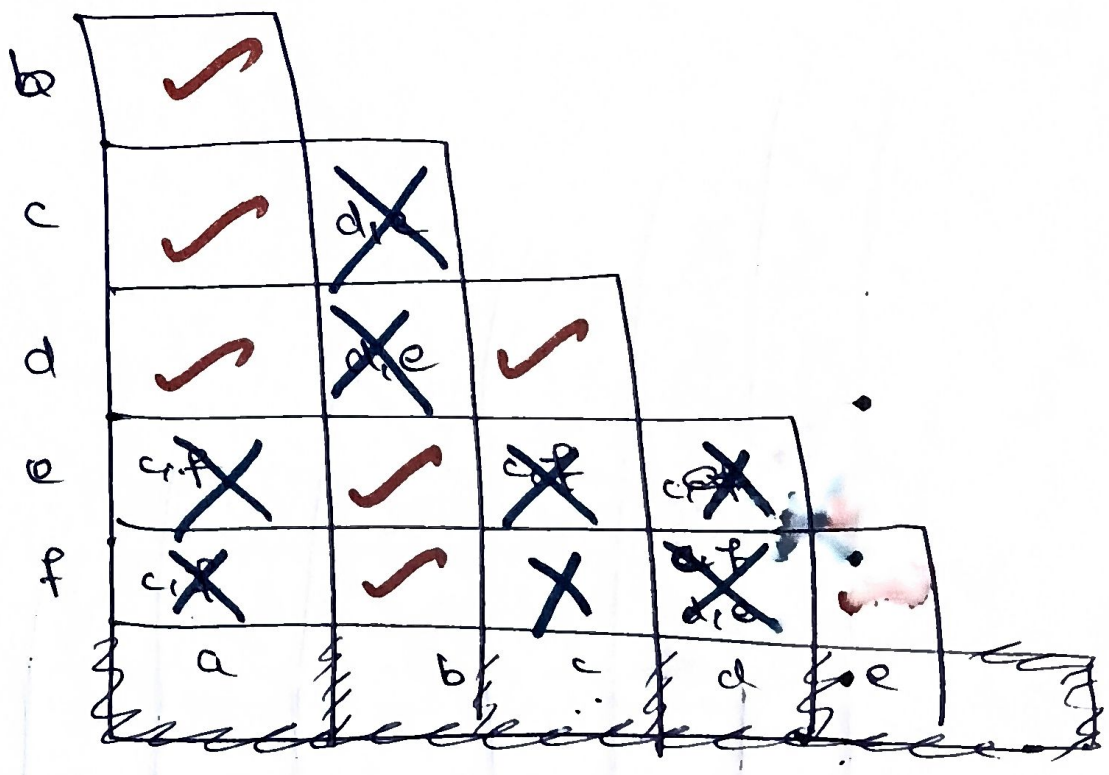
Replace b with a
e,g with d

Present State	Next State		Output
	x=0	x=1	x=1
a	d	a	
c	d	f	
d	a	d	
f	c	a	

* State Reduction from the flow table

Ex2 Flow table.

	00	01	11	10
a	c,-	(a),0	b,-	-,-
b	-,-	a,-	(b),-	e,-
c	(c),0	a,-	-,-	d,-
d	c,-	-,-	b,-	(d),0
e	f,-	-,-	b,-	(e),1
f	(f),1	a,-	-,-	e,-



Equivalent states

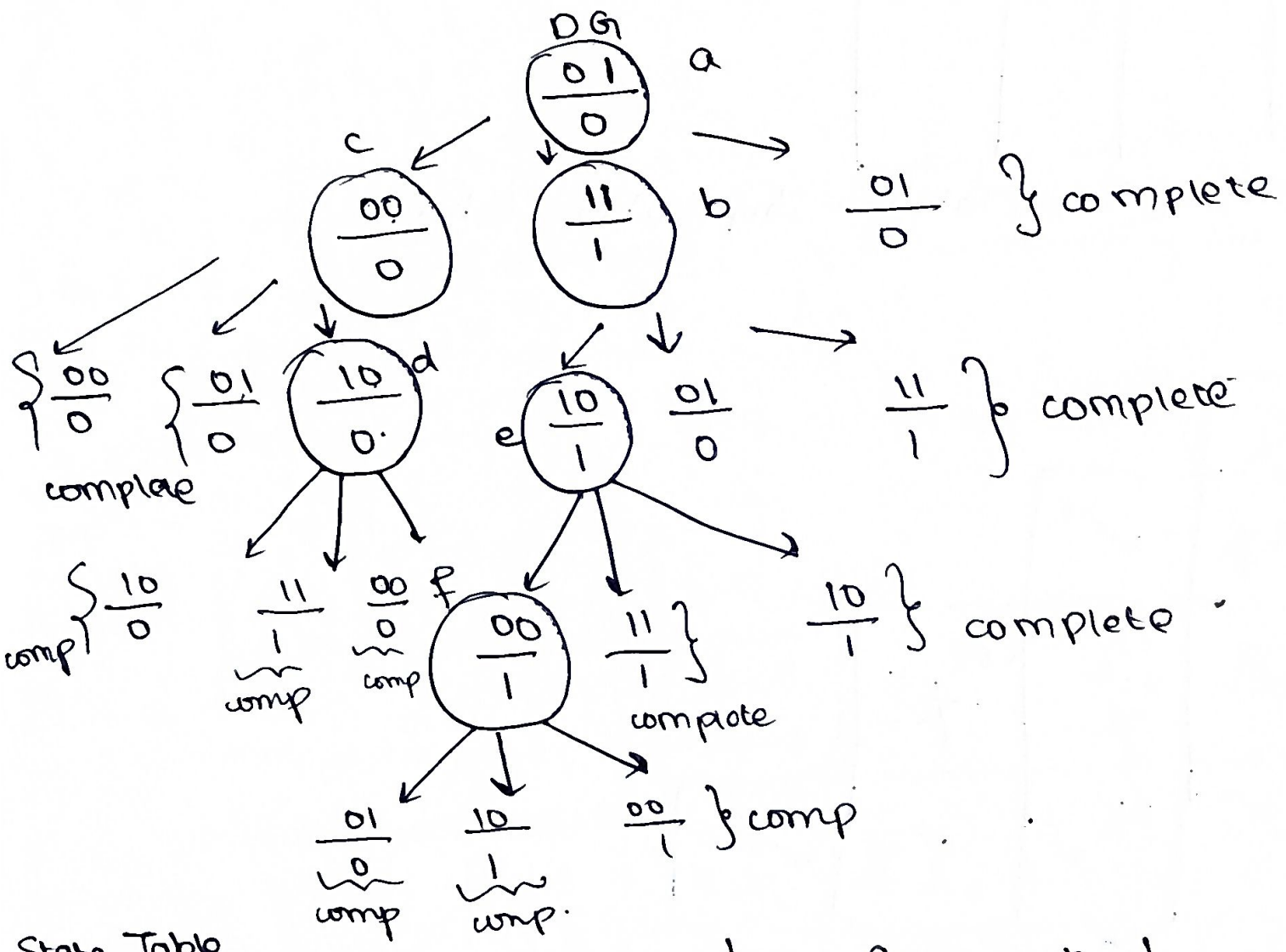
(a,b) (a,c) (a,d) (b,e) (b,f) (c,d) (e,f)

Asynchronous Design Examples

Example 1 : Design a gated latch circuit with 2 inputs G → gate and D → data and one output Q. The gated latch is a memory element that has the value of D when G = 1 and retains the value when G goes to 0. (when G goes to 0, there is no change in the output)

Solution : G = 1 ⇒ O/P = D

G = 0 ⇒ O/P = previous state



State Table

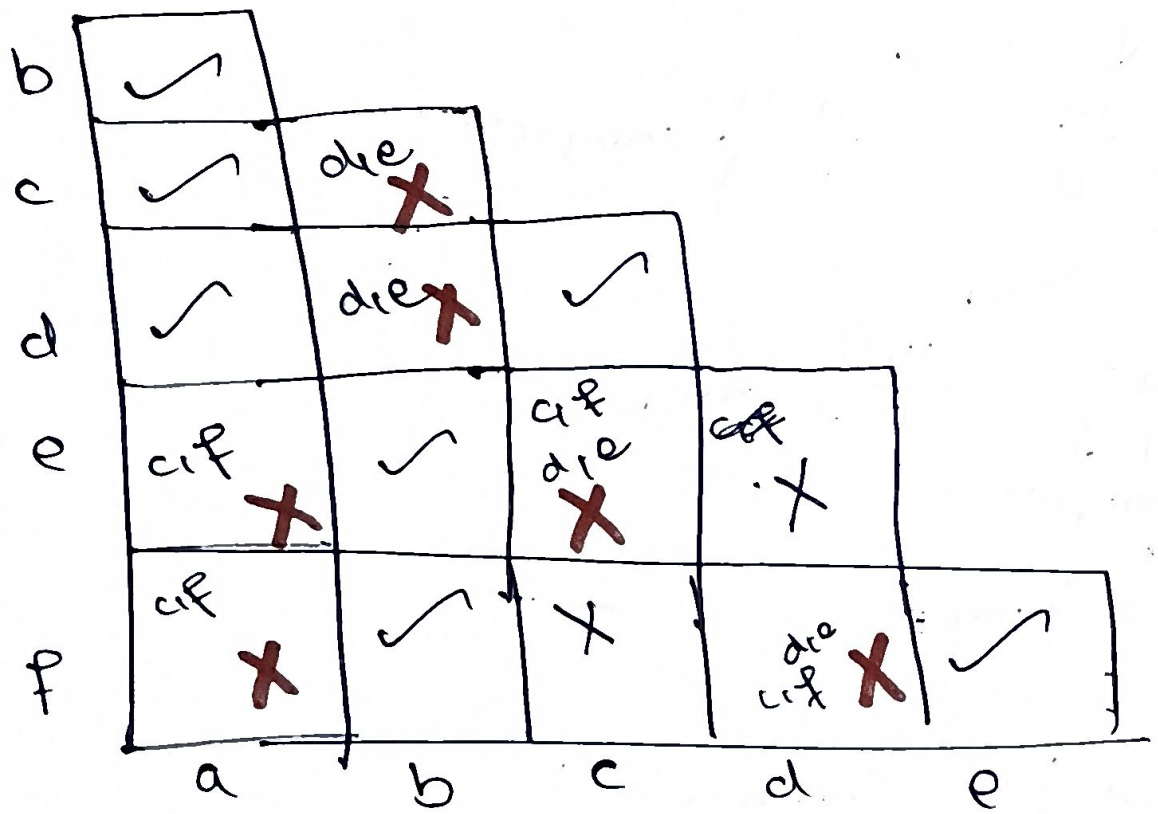
State	Input	Output	Comments
a	01	0	D = Q as G = 1
b	11	1	D = Q as G = 1
c	00	0	after state a or d.
d	10	0	after c
e	10	1	after b or f
f	00	1	after e

Primitive Flow Table

DG

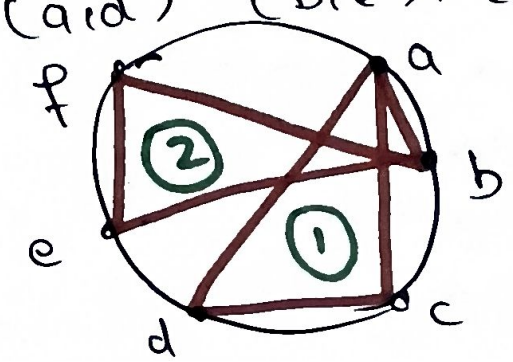
		00	01	11	10
01	a	c_1-	(a), 0	b_1-	$-_1-$
11	b	$-_1-$	a_1-	(b), 1	e_1-
00	c	(c), 0	a_1-	$-_1-$	d_1-
10	d	c_1-	$-_1-$	b_1-	(d), 0
10	e	f_1-	$-_1-$	b_1-	(e), 1
00	f	(f), 1	a_1-	$-_1-$	e_1-

Reduction of Primitive Flow Table - Implication Table



Compatible Pairs

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



find closed paths

$(a, c, d) = a$
$(b, e, f) = b$
} maximum compatible pairs

Splitting of primitive flow table

	00	01	11	10
a	c, -	(a), 0	b, -	-, -
c	(c), 0	a, -	-, -	d, -
d	c, -	-, -	b, -	(d), 0

	00	01	11	10
b	-, -	a, -	(b), 1	e, -
e	f, -	-, -	b, -	(e), 1
f	(f), 1	a, -	-, -	e, -

merge 2 diagrams

	00	01	11	10
acd	c, 0	a, 0	b, -	d, 0
bef	f, 1	a, -	b, 1	e, 1

apply equalities obtained from the max comp. pairs

	00	01	11	10
a	a, 0	a, 0	b, ~~0~~	a, 0
b	b, 1	a, -	b, 1	b, 1

assign state values

	00	01	11	10
a	a, 0	a, 0	b, 0	a, 0
b	b, 1	a, 1	b, 1	b, 1

Note : if the states are in the form

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

let D = 0 and G = 1

K-Map Simplification

(input values)

	DG 00	01	11	10
y 0	0	0	1	0
1	1	0	1	1

$Y = DG + G'y$

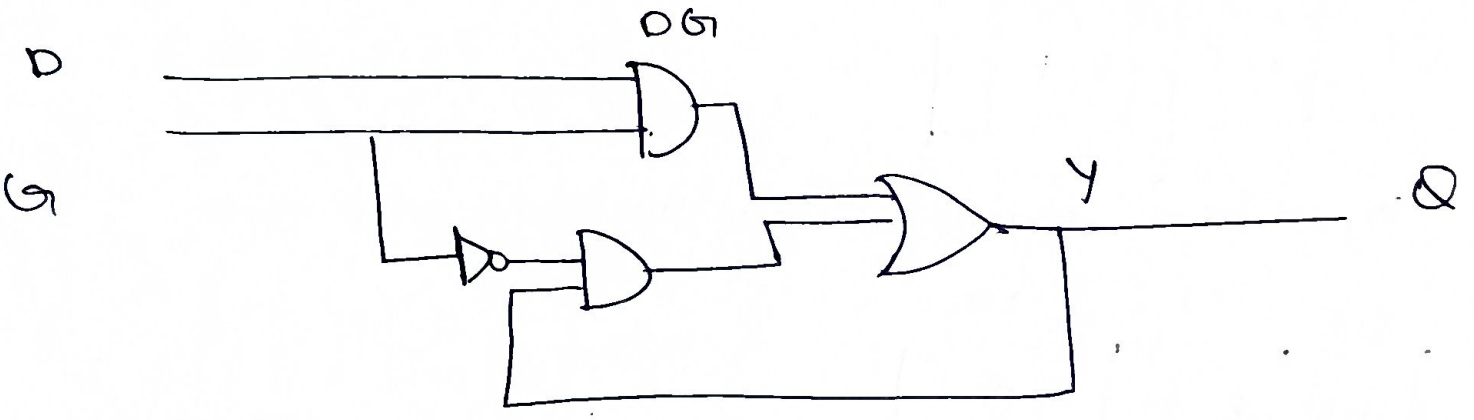
y	D	G
1	0	0
1	x	0
		G'y

y	D	G
0	1	1
1	1	1
		DG

	DG 00	01	11	10
y 0	0	0	0	0
1	1	1	1	1

$Q = y$

Combinational Circuit Implementation



SR Latch Implementation

Zy \ DG	00	01	11	10
0	0	0	1	0
1	1	0	1	1

Latch Excitation Table

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

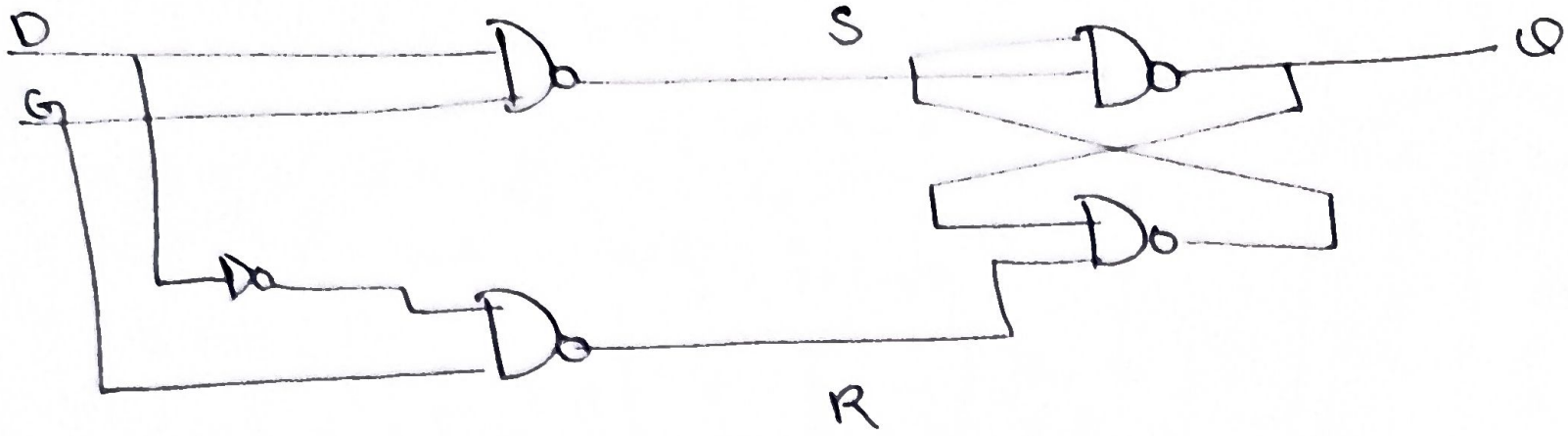
y \ DG	00	01	11	10
0	0	0	1	0
1	x	0	x	x

$S = DG$

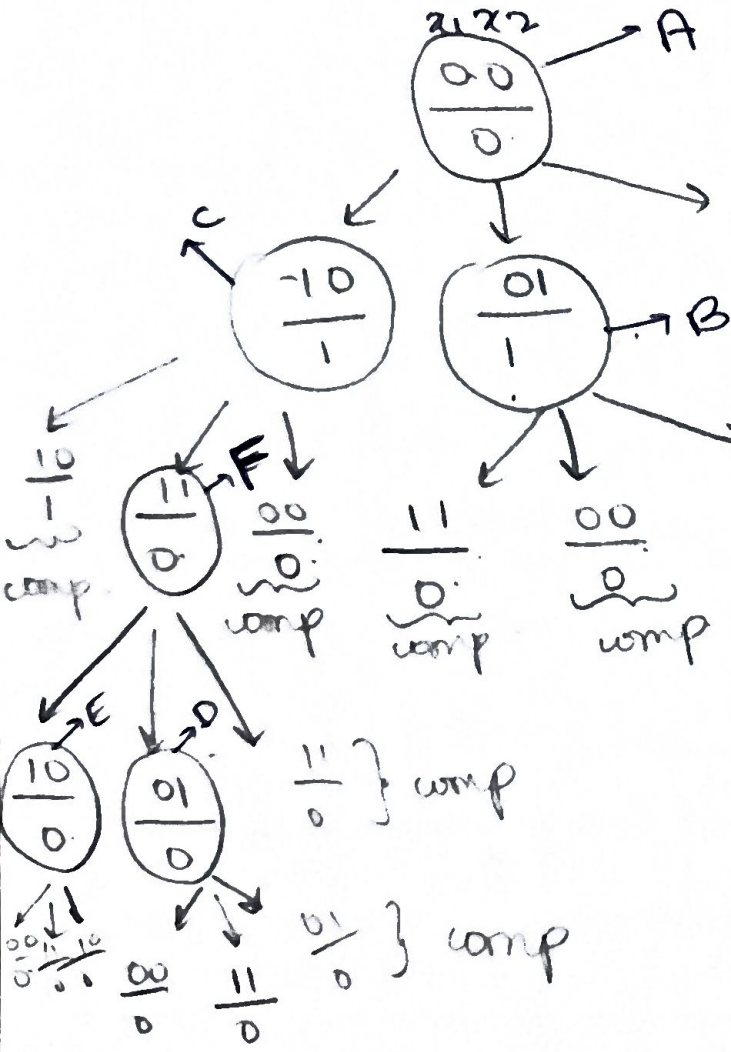
y \ DG	00	01	11	10
0	x	x	0	x
1	0	1	0	0

$R = D'G$

Latch Diagram



Example 2 : Design an asynchronous sequential circuit w/ 2 inputs x_1 and x_2 and one output z. Initially both inputs are equal to 0. When x_1 or x_2 becomes 1, the output z becomes 1. When the second input also becomes 1, the output changes to zero. The output stays at 0 until the circuit goes back to the initial state.



State Table

State	Input	Output
a	00	0
b	01	1
c	10	1
d	01	0
e	10	0
f	11	0

primitive flow table

x_1x_2	00	01	11	10
a	(a) 0	b, -	-, -	c, -
b	a, -	(b), 1	f, -	-, -
c	a, -	-, -	f, -	(c), 1
d	a, -	(d) 0	f, -	-, -
e	a, -	-, -	f, -	(e), 0
f	-, -	d, -	(f), 0	e, -

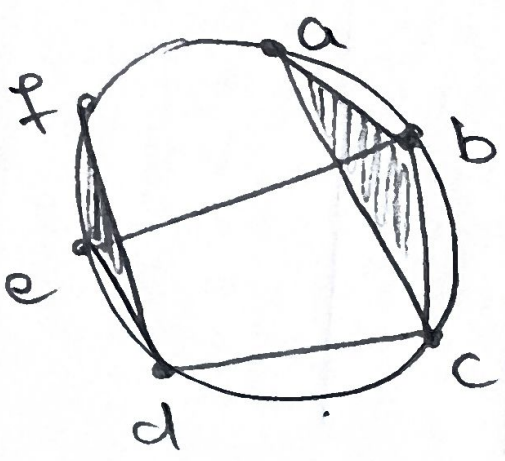
Implication Table

	a	b	c	d	e
b	✓				
c	✓	✓			
d	b,d X	X	✓		
e	c,e X	✓	X	✓	
f	~~X~~ b,d c,e.	X b,d	c,e X	✓	✓

compatible pairs

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

maximum compatible pairs



(a,b,c) = a

(d,e,f) = d.

separated Flowtable

	00	01	11	10
a	a,0	b,-	-,-	c,-
b	a,-	b,1	f,-	-,-
c	a,-	-,-	f,-	c,1

	00	01	11	10
d	a,-	d,0	f,-	-,-
e	a,-	-,-	f,-	e,0
f	-,-	d,-	f,0	e,-

merged

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

a,b,c = a and d,e,f = d.

F \ x_1x_2	00	01	11	10
	a,0	a,1	d, 1	a,1
	a,0	d,0	d,0	d,0

a=0

d=1

K-Map Simplification

F \ x_1x_2	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$F x_1 x_2$: 0 1 1 / 1 1 1 → $x_1 x_2$

$F x_1 x_2$: 1 0 1 / 1 1 1 → $F x_2$

$F x_1 x_2$: 1 1 1 / 1 1 0 → $F x_1$

$= F x_1 + x_1 x_2 + F x_2$

check discrepancies

Z = F \ x_1x_2	00	01	11	10
0	0	1	1	1
1	0	0	0	0

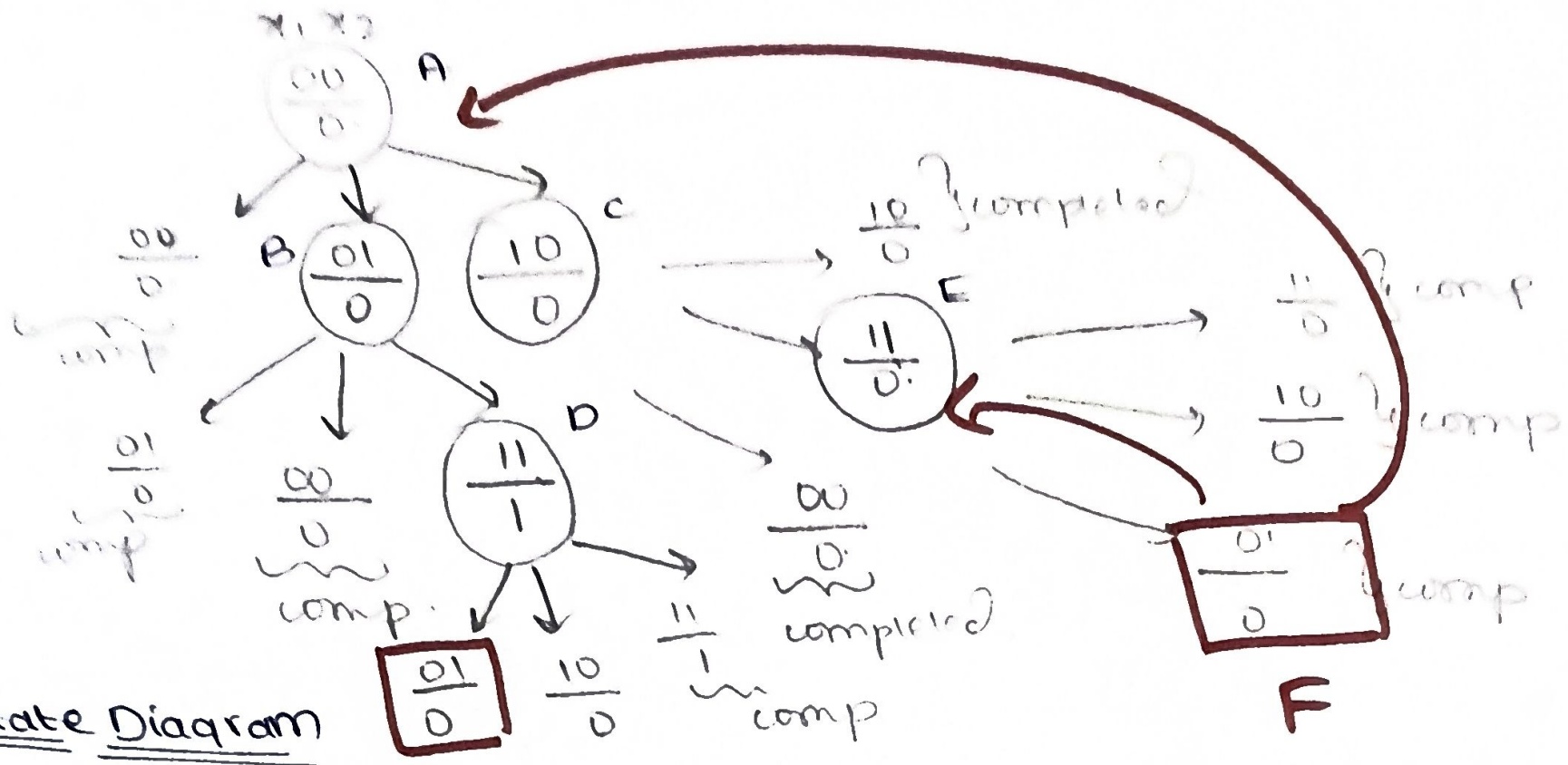
$F x_1 x_2$: 0 0 1 / 0 1 1 → $F' x^2$

$F x_1 x_2$: 0 1 1 / 0 1 0 → $F' x_1$

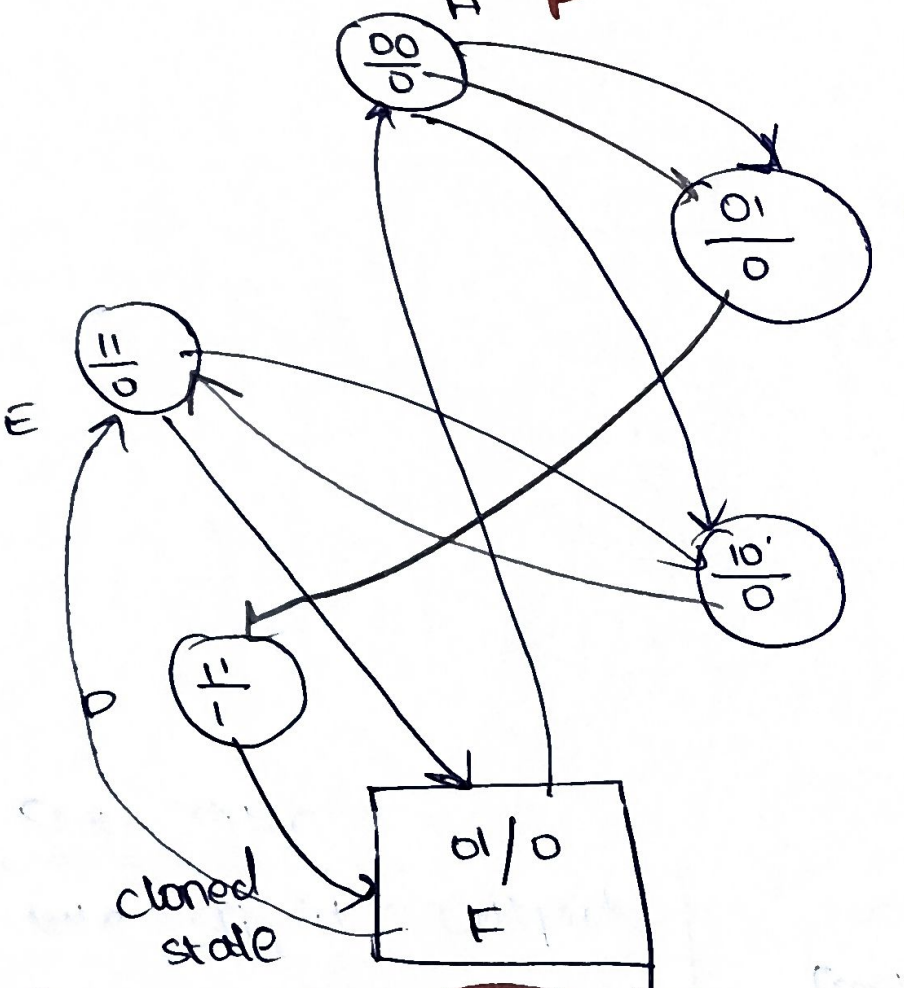
Draw ckt diagram

Example3: Design an asynchronous sequential circuit which which has 2 inputs x_1 & x_2 and one output z. The circuit gives an output whenever the input sequence is (0,0), (0,1) and (1,1) in that order.

Solution $(0,0) \rightarrow (0,1) \rightarrow (1,1)$



State Diagram



Assume that there is a cloned state F, which is the same as B, since the problem does not state if $\frac{11}{1}$ can go back to B or if E can go to B

~~state table~~

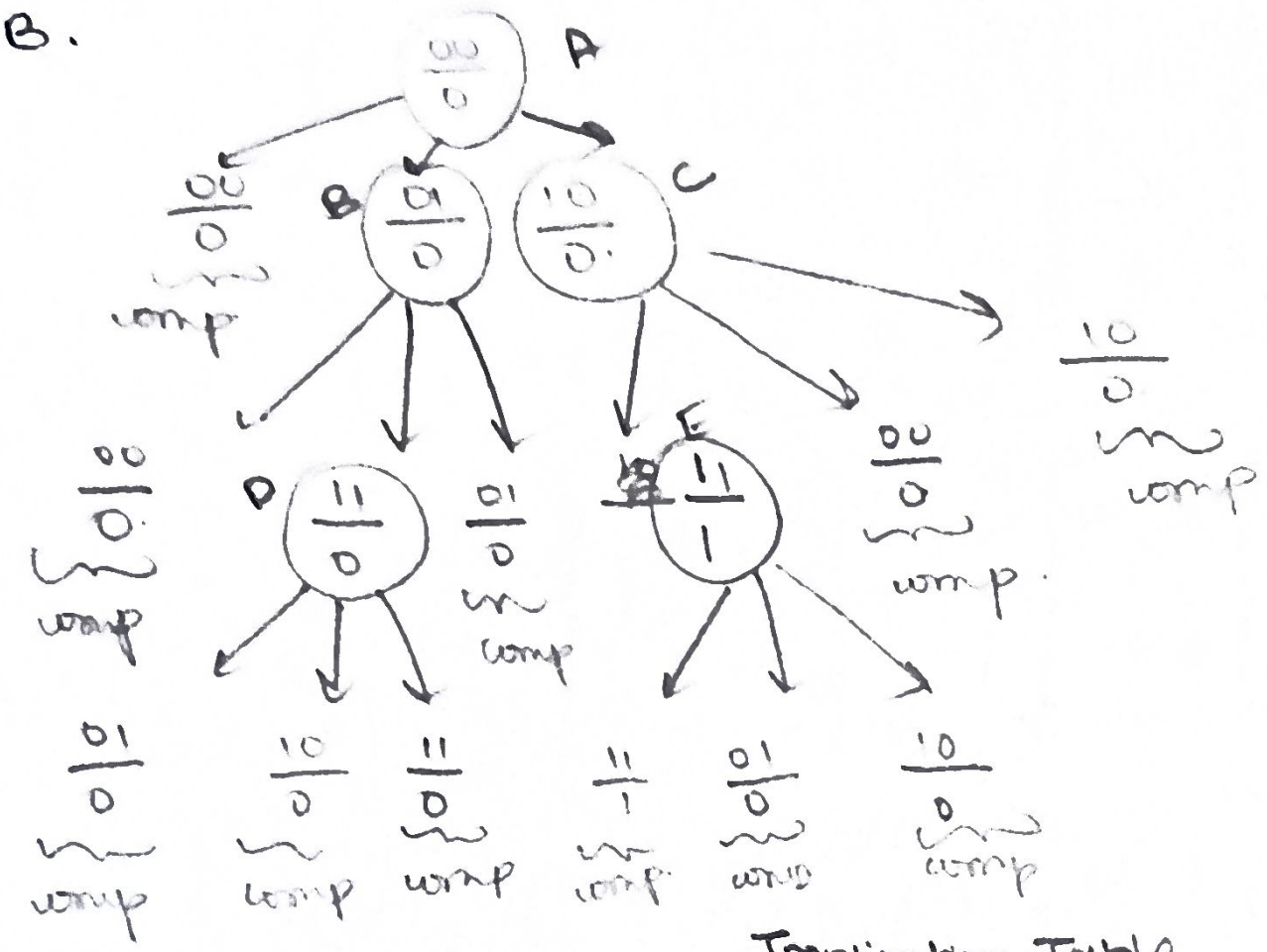
primitive flow table

	00	01	11	10
a	(a), 0			
b				
c				
d				
e				
f				

Example 4: Design a circuits with inputs A and B to give an output $z=1$, when $AB=1$, but only if A becomes 1 before

AB → AB
10 → 11
1.

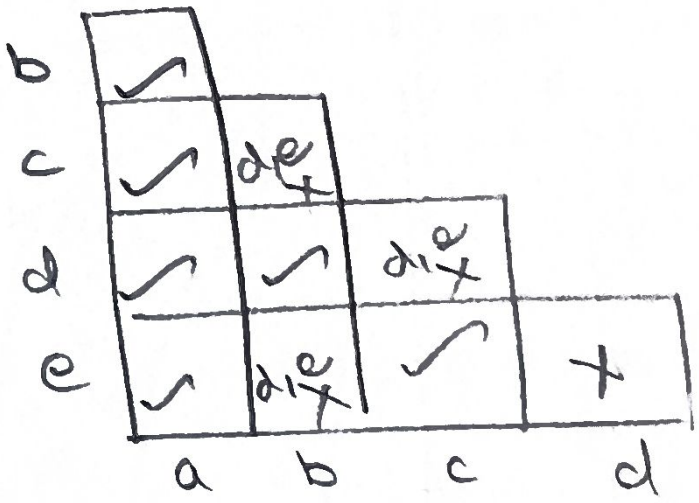
B.



State Table

State	Input	Output
a	00	0
b	01	0
c	10	0
d	11	0
e	11	1

Implication Table



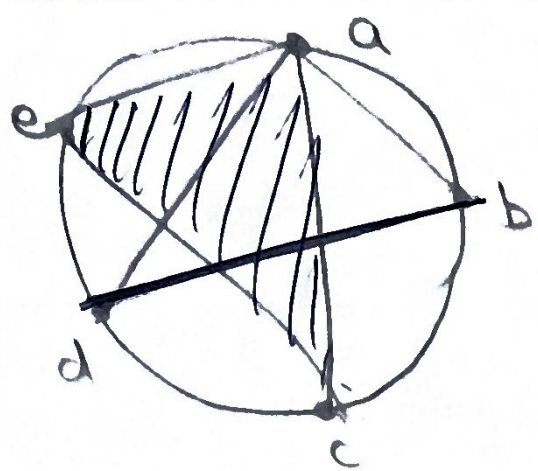
Primitive Flow Table

	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, -

Compatible Pairs

(a,b), (a,c), (a,d), (a,e)
(b,d), (c,e)

Maximum Compatible Pair



eliminate. AB
eliminate DA

consider (a,c,e) = a
and b,d = b.

i.e

AB /	00	01	11	10
a	a,0	b,-	-,-	c,-
c	a,-	-,-	e,-	c,0
e	-,-	b,-	e,1	c,-

b.d

AB /	00	01	11	10
b	a,-	b,0	d,-	-,-
d	-,-	b,-	d,0	c,-

⇓

merge.

a,0	b,-	e,1	c,0
a,-	b,0	d,0	c,-

⇓.

a,0	b,-	a,1	a,0
a,-	b,0	b,0	a,-

X-Map Simplification

F / AB	00	01	11	10
0	0	1	0	0
1	0	1	1	0

Output / AB	00	01	11	10
F 0	0	X	1	0
F 1	X	0	0	X

F A B
0 0 1
1 0 1

A'B

F A B
1 0 1
1 1 1

F B

F A B
0 0 1
0 y 1

F'B

Input = A'B + F B

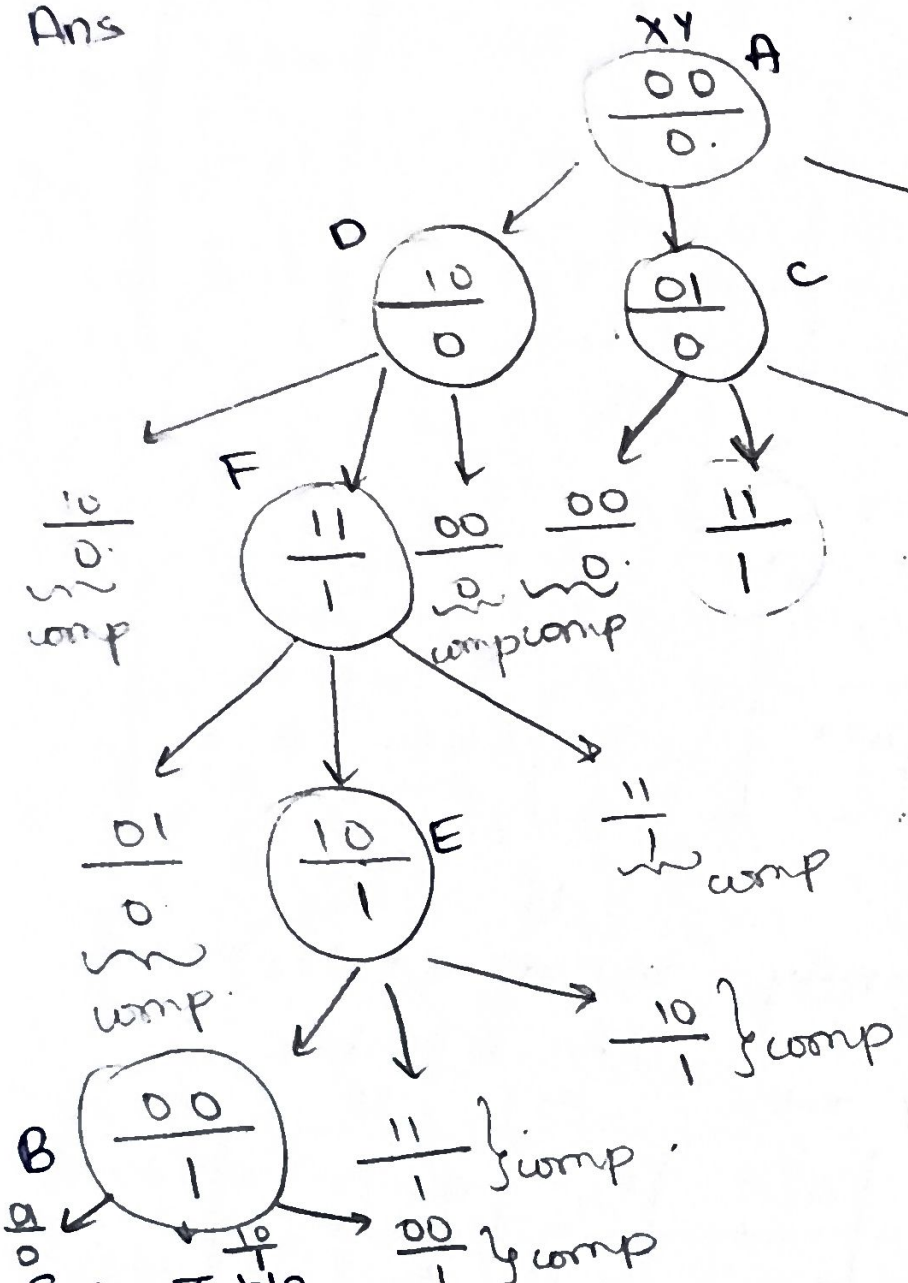
∴

Input = A'B + F B
O/p = F'B

Draw circuit diagram

Example 6 : Design an asynchronous sequential circuit with 2 inputs X and Y and with one output Z. Whenever Y is 1, the input X is transferred to Z. When Y is zero, the output does not change for any change in X.

Ans



State Table

pre-state	input	output
a	00	0
b	00	1
c	01	0
d	10	0
e	10	1
f	11	1

primitive flowtable

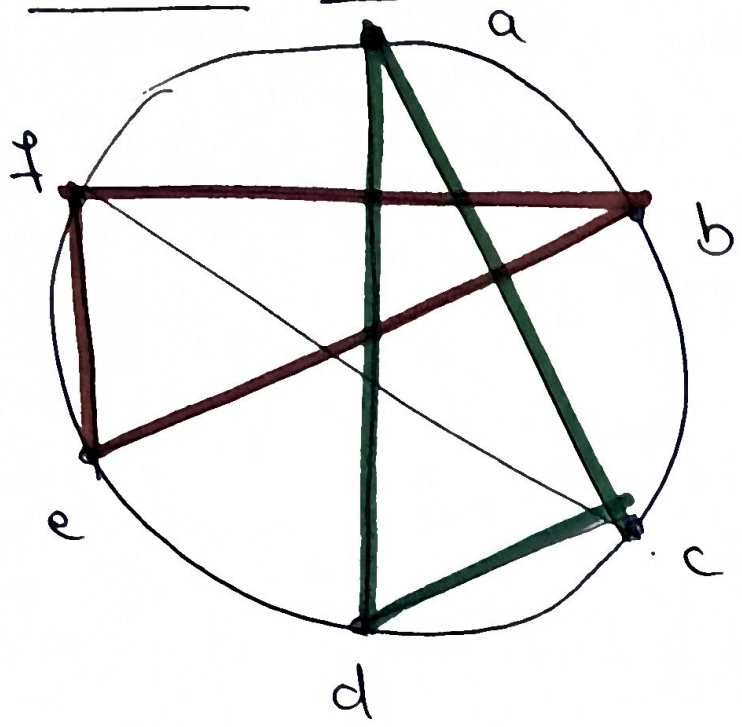
	00	01	11	10
a	(a), 0	c, -	-, -	d, -
b	(b), 0	c, -	-, -	e, -
c	a, -	(c), 0	f, -	-, -
d	a, -	-, -	f, -	(d), 0
e	b, -	-, -	f, -	(e), 1
f	-, -	c, -	(f), 1	e, -

Implication Table

compatible pairs

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

Merger Diagram



$(a, d, c) = a$

$(b, e, f) = b$

Reduced primitive flow table

	00	01	11	10
a	a, 0	c, -	- , -	d, -
c	a, -	c, 0	f, -	- , -
d	a, -	- , -	f, -	d, 0

⇓

	00	01	11	10
b	b, 1	c, -	- , -	e, -
e	b, -	- , -	f, -	e, 1
f	- , -	c, -	f, 1	e, -

⇒

merge

F \ xy	00	01	11	10
a	a, 0	c, 0	f, -	d, 0
b	b, 1	c, -	f, 1	e, 1

⇓

simplify - assign $(a, d, c) = a$ & $(b, e, f) = b$

state assignment

check for discrepancy

F \ xy	00	01	11	10
0	0, 0	a, 0	1, -	a, 0
1	1, 1	0, -	1, 1	1, 1

⇐

F \ xy	00	01	11	10
a	a, 0	a, 0	b, -	a, 0
b	b, 1	a, -	b, 1	b, 1

⇓

split into kmaps

Input

0	0	1	0
1	0	1	1

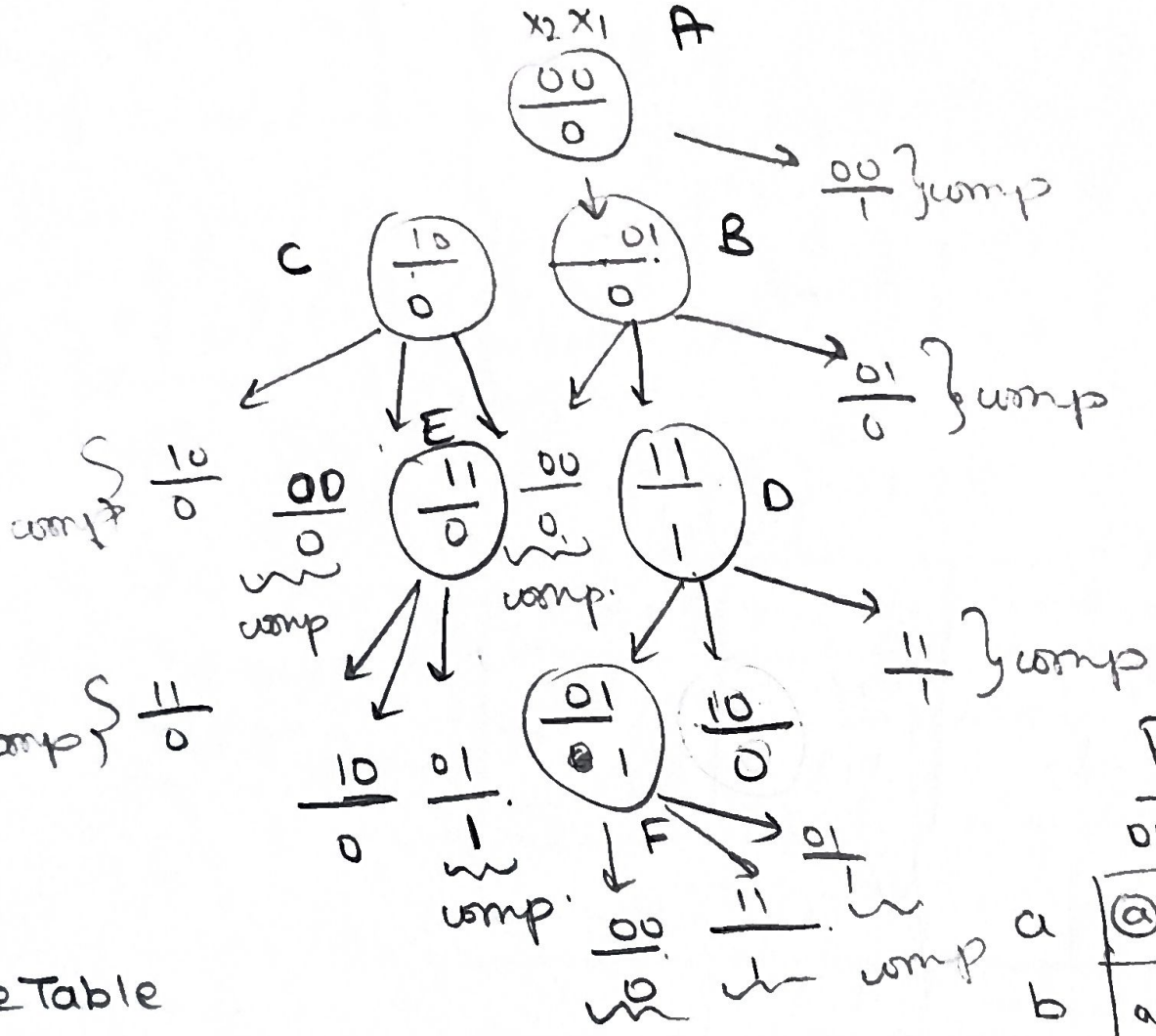
z

F \ xy	00	01	11	10
0	0	0	~	0
1	1	1	1	1

simplify & draw circuits

Ex 6

Design an asynchronous sequential circuit that has 2 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 occurs while x_1 is 1, will cause the output z to be 1. The output z will remain 1 until X_1 returns to 0.



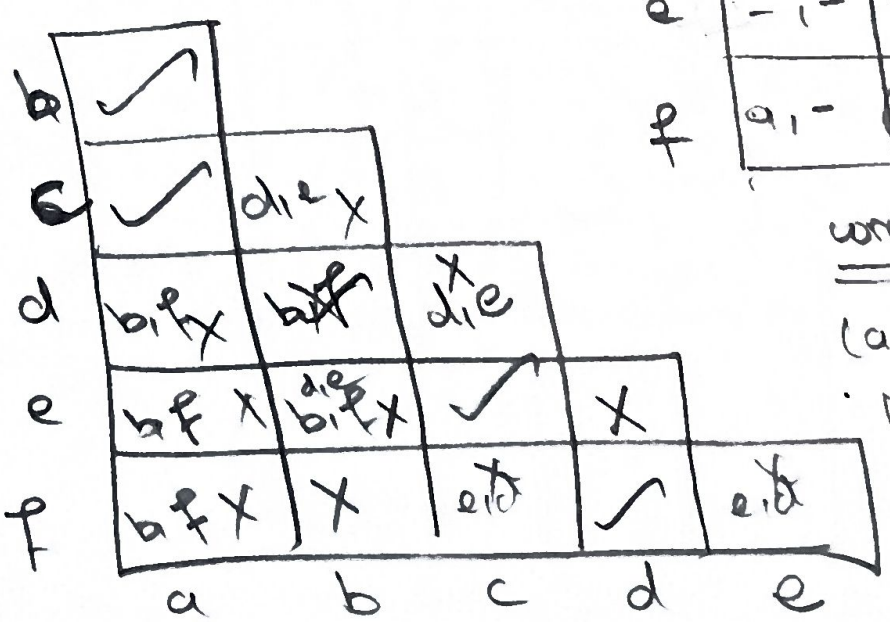
State Table

State	Input	Output
a	00	0
b	01	0
c	10	0
d	11	1
e	11	0
f	01	1

Primitive Flow Table

	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, -	-, -

Implication Table

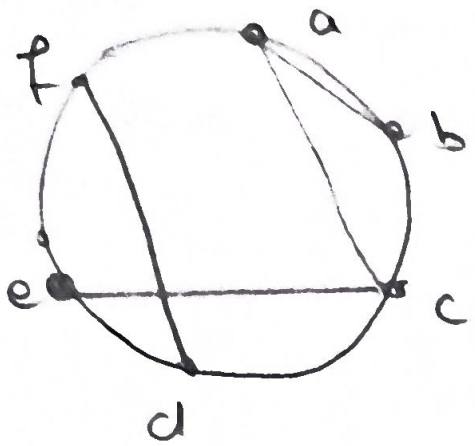


compatible pairs

(a, b), (a, c)
~~(b, d)~~, (c, e),
(d, f)

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

Merger diagram



a = b

d = f

c = e

	00	01	11	10
a	a,0	b,-	-,-	c,-
b	a,-	b,-	d,-	-,-

	00	01	11	10
d	-,-	f,-	d,1	c,-
f	a,-	f,1	d,-	-,-

	00	01	11	10
c	a,-	-,-	e,-	c,0
e	-,-	f,-	e,0	c,-

⇓ merge

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

a = 00

~~d = 01~~ c = 01

~~c = 10~~ d = 10

⇓

00	01	11	10
a,0	a,-	d,-	c,-
a,-	d,-	c,0	c,0
a,-	d,1	d,1	c,-

⇓

00	01	11	10
00,0	00,-	10,-	a,-
00,-	10,-	01,0	a,0
00,-	10,-	10,1	01,-

Hazards: unwanted switching transients that may

appear at the output

→ gives tempory wrong values

→ can accidentally transition to a wrong state

Types of Hazards

static 0 → goes to 1 when it should remain at 0

static 1 → goes to 0 when it should remain at 1

dynamic hazard → causes o/p to change 3 or more times when it should change from 1→0 or 0→1

essential hazard → feedback delay < delay of other inputs (cannot be handled by adding more gates)

To fix hazards: identify op exp
draw ckt, find values
plot in k map
merge disjoint values

To Do:

1. 3 Asynchronous design questions
2. Unit 1
3. Registers
4. SPLD Theory from unit 5
5. Go through everything again

4x11x2
11
110
421