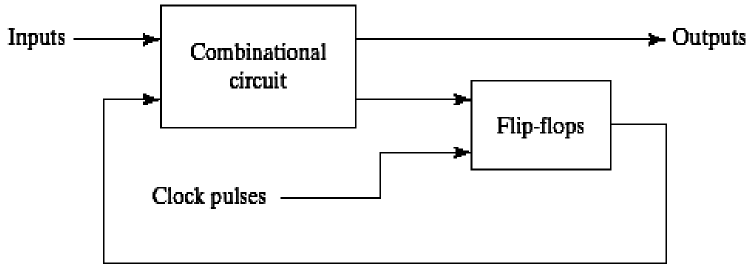
LECTURE 18

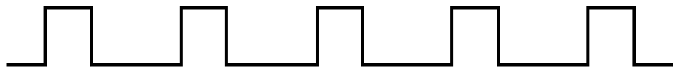
Digital Logic Families

Synchronous Sequential Circuits

- The change of internal state occurs in response to the synchronized clock pulses.
- The memory elements are flip-flops.



(a) Block diagram



Asynchronous Sequential Circuits

Asynchronous sequential circuits

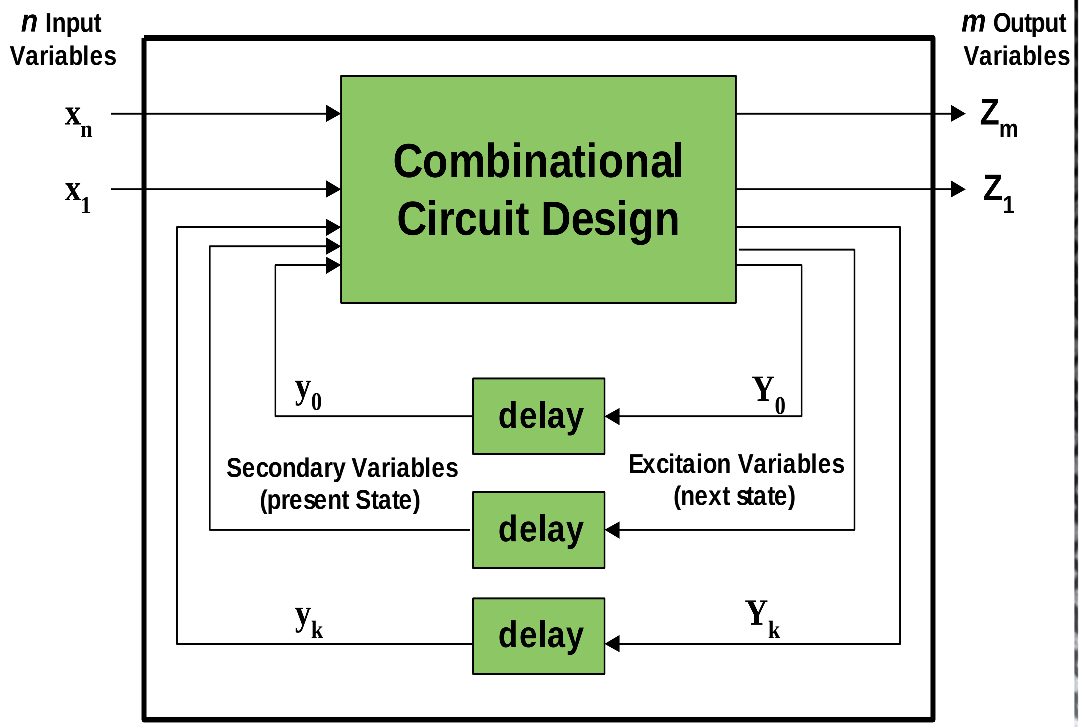
- Internal states can change at any instant of time when there is a change in the input variables

- No clock signal is required

- Have better performance but hard to design due to timing problems

-The memory elements are either unclocked FF's or time-delay elements.

-The design of these circuits is more difficult than the design of synchronous circuits due to the timing problem.



n Input Variables
x_n
x_1
Combinational Circuit Design
m Output Variables
Z_m
Z_1
y_0
delay
Y_0
Secondary Variables (present State)
delay
Excitaion Variables (next state)
y_k
delay
Y_k

Why Asynchronous Circuits?

1- Accelerate the speed of the machine (no need to wait for the next clock pulse).

2-Used when the input signals change independently of the clock pulses.

3- Simplify the circuit in the small independent circuits.

4- Used to communicate two circuits each have its own clock.

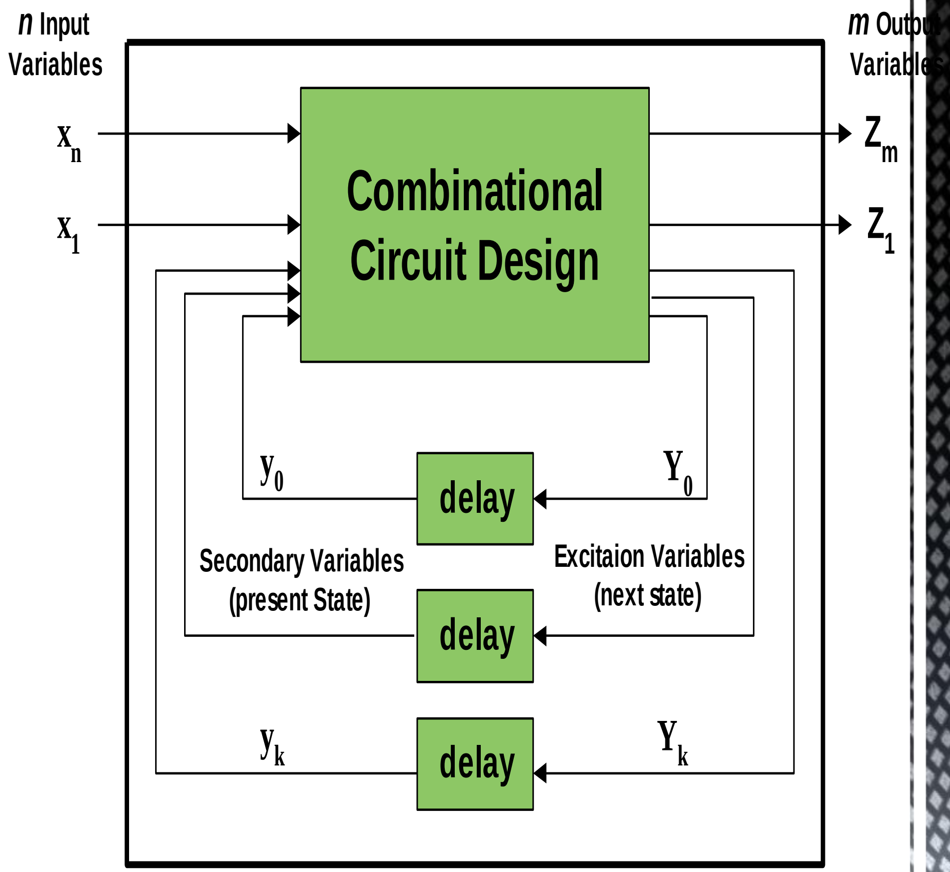
Asynchronous Circuits

• The delay elements provide short-term memory for the sequential circuits.

• Present state variables [y1..yk] are called secondary variables

• Next state variables [Y1..Yk] are called excitation variables.

• When an input variable changes, it takes a certain time to propagate through the combinational circuit to change Y, and then Y takes a certain time to propagate through the delay element to become a new state.



n Input Variables
x_n
x_1
Combinational Circuit Design
m Outpu Variable
Z_m
Z_1
y_0
delay
Y_0
Secondary Variables (present State)
Excitaion Variables (next state)
delay
y_k
delay
Y_k

Asynchronous Circuits

- The circuit reaches a steady-state condition when yi = Yi for i=1,2,…, K.

- Stable System:

  for a given value of input variables, the system is stable if the circuit reaches a steady state condition.

- Fundamental-mode operation:

  this mode assumes that the one input signal changes at a time and only when the circuit is in stable condition.

- The time between two input changes must be longer than the time it takes the circuit to reach a stable state.

Analysis Procedure

The analysis consists of obtaining a table or a diagram that describes the sequence of internal states and outputs as a function of changes in the input variables.

Transition Table

Flow Table

Stability Consideration

Transition Table

Transition table is useful to analyze an asynchronous circuit from the circuit diagram Procedure to obtain transition table:

1. Determine all feedback loops in the circuits

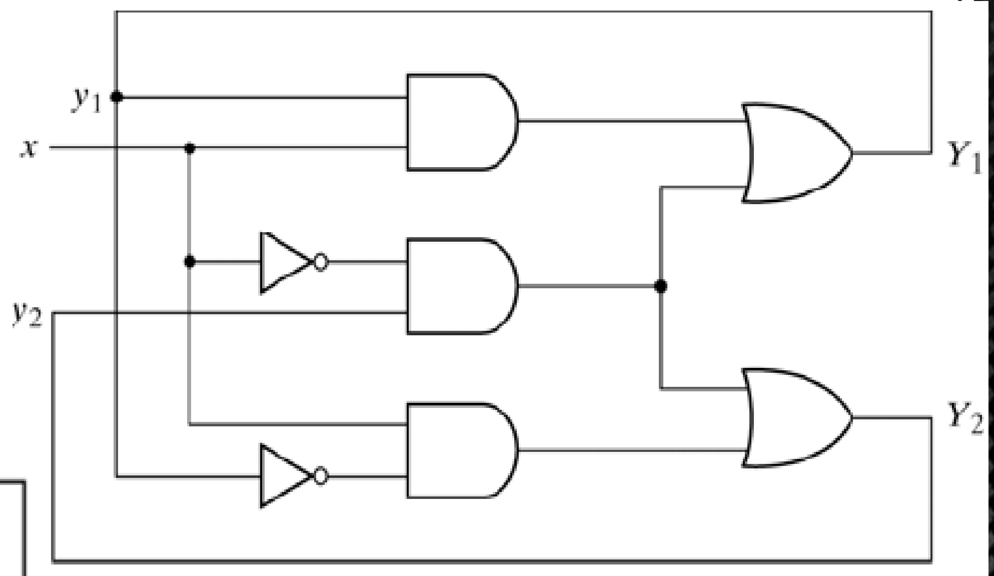
2. Mark the input (yi) and output (Yi) of each feedback loop

3. Derive the Boolean functions of all Y’s

4. Plot each Y function in a map and combine all maps into one table

5. Circle those values of Y in each square that are equal to the value of y in the same row

Transition Table



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

(a) Map for
$Y_1 = xy_1 + x'y$

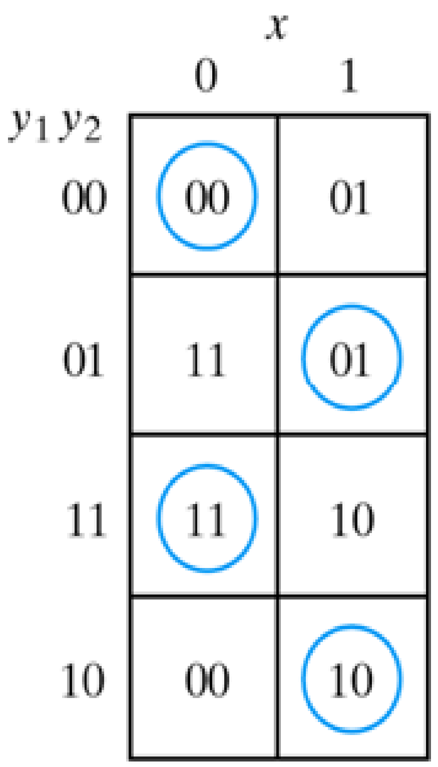
y_1y_2 \ x	0	1
00	0	1
01	1	1
11	1	0
10	0	0

(b) Map for
$Y_2 = xy'_1 + x'y_2$

Y1 = xy1 + x’y2
Y2 = xy’1 + x’y2

e

- -If y=00 and x= 0 $\Rightarrow$ Y ==00 (Stable)
- -If x changes from 0 to 1 while y=00, the circuit changes Y to 01 which is temporary unstable condition (Y != y)
- -As soon as the signal propagates to make Y = 01, the feedback path causes a change in y to 01. (transition form the first row to the second row)
- -If the input repeatedly alternates between 0 and 1, the circuit will repeat the sequence of states



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

$$\begin{tikzcd}00 \arrow[r] & 01 \arrow[r] & 11 \arrow[r] & 10 \arrow[lll, bend right]\end{tikzcd}$$

Transition Table

In an asynchronous sequential circuit, the internal state can change immediately after a change in the input.

It is sometimes convenient to combine the internal state with input

value together and call it the **Total State of the circuit.**

(Total state = Internal state + Inputs)

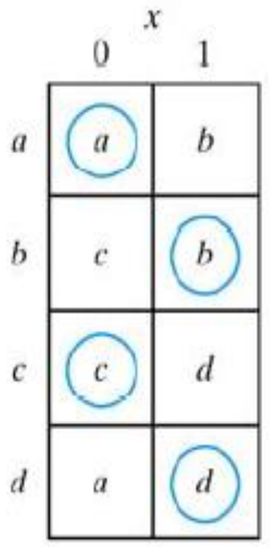
In the last example , the circuit has

- 4 **stable total states: (y1y2x= 000, 011, 110, and 101)**
- 4 **unstable total states: (y1y2x= 001,010,111, and 100)**

Flow Table

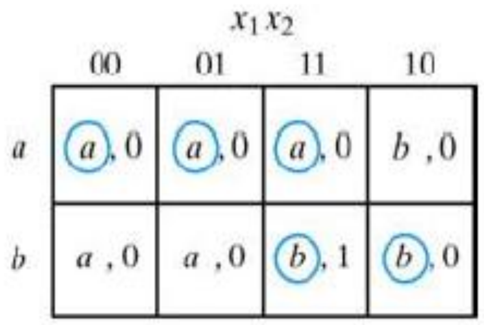
- A flow table is similar to a transition table except that the internal state are symbolized with letters rather than binary r

- It also includes the output values of the circuit for each stable state.



	x = 0	1
a	ⓐ	b
b	c	ⓑ
c	ⓒ	d
d	a	ⓓ

(a) Four states with one input



	x_1x_2 = 00	01	11	10
a	ⓐ, 0	ⓐ, 0	ⓐ, 0	b , 0
b	a , 0	a , 0	ⓑ, 1	ⓑ, 0

(b) Two states with two inputs and one output

Flow Table

- In order to obtain the circuit described by a flow table, it is necessary to convert the flow table into a transition table from which we can derive the logic diagram .

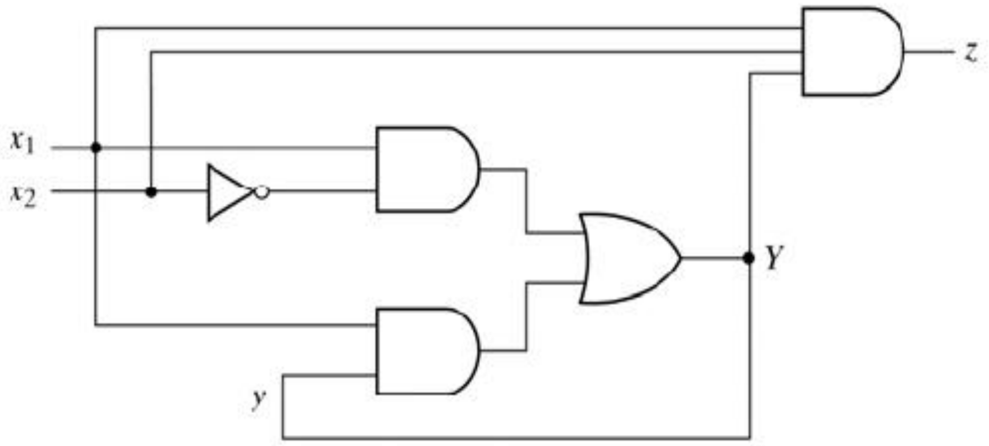
- This can be done through the assignme of a distinct binary value to each state.

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

(a) Transition table
$Y = x_1x'_2 + x_1y$

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

(b) Map for output
$z = x_1x_2y$



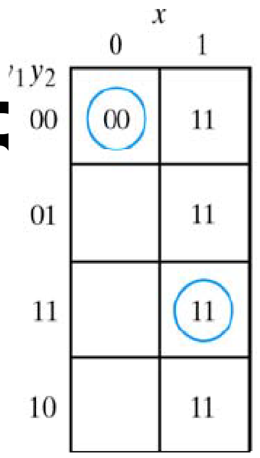
(c) Logic diagram

Two or more binary state variables will change value when one input variable changes.

Cannot predict state sequence if unequal delay is encountered.

<u>Non-critical race</u>: The final stable state does not depend on the change order of state variables

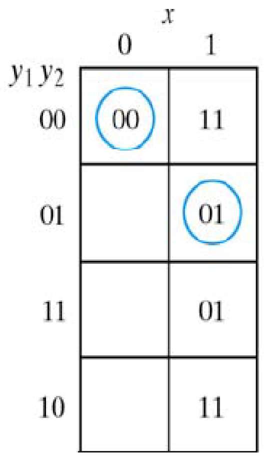
<u>Critical race</u>: The change order of state variables will result in different stable states Should be avoided !!



y_1y_2 \ x	0	1
00	$\textcircled{00}$	11
01		11
11		$\textcircled{11}$
10		11

(a) Possible transitions:

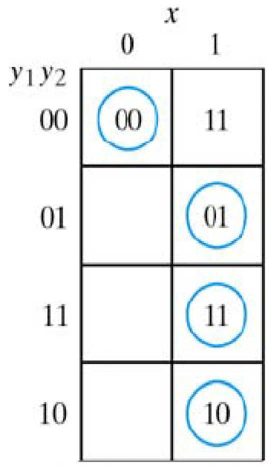
$00 \rightarrow 11$
$00 \rightarrow 01 \rightarrow 11$
$00 \rightarrow 10 \rightarrow 11$



y_1y_2 \ x	0	1
00	$\textcircled{00}$	11
01		$\textcircled{01}$
11		01
10		11

(b) Possible transitions:

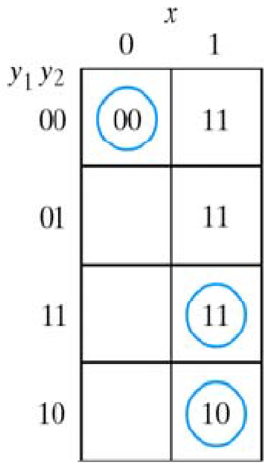
$00 \rightarrow 11 \rightarrow 01$
$00 \rightarrow 01$
$00 \rightarrow 10 \rightarrow 11 \rightarrow 01$



y_1y_2 \ x	0	1
00	$\textcircled{00}$	11
01		$\textcircled{01}$
11		$\textcircled{11}$
10		$\textcircled{10}$

(a) Possible transitions:

$00 \rightarrow 11$
$00 \rightarrow 01$
$00 \rightarrow 10$



y_1y_2 \ x	0	1
00	$\textcircled{00}$	11
01		11
11		$\textcircled{11}$
10		$\textcircled{10}$

(b) Possible transitions:

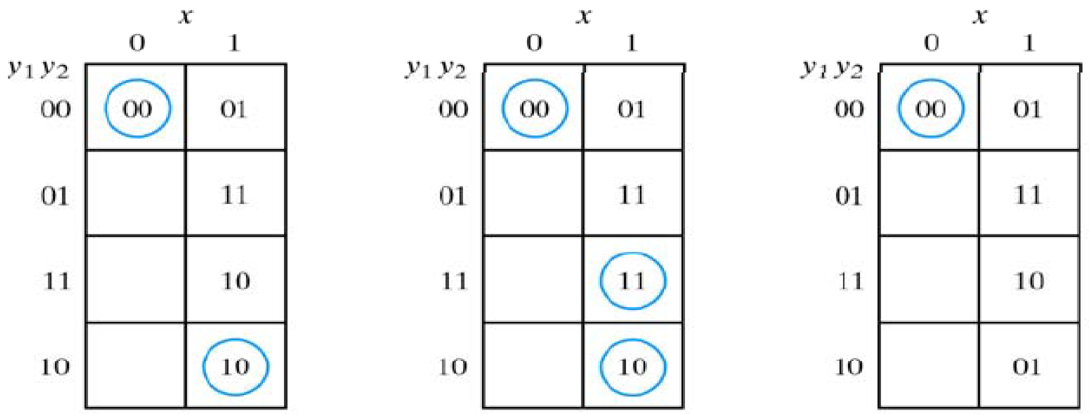
$00 \rightarrow 11$
$00 \rightarrow 01 \rightarrow 11$
$00 \rightarrow 10$

Race Solution

It can be solved by making a proper binary assignment to the state variables.

The state variables must be assigned binary numbers in such a way that only one state variable can change at any one time when a state transition occurs in the flow table.

It will be discussed later.



(a) State transition: 00 → 01 → 11 → 10

(b) State transition: 00 → 01 → 11

(c) Unstable → 01 → 11 → 10

Stability Check

Asynchronous sequential circuits may oscillate between unstable states due to the feedback

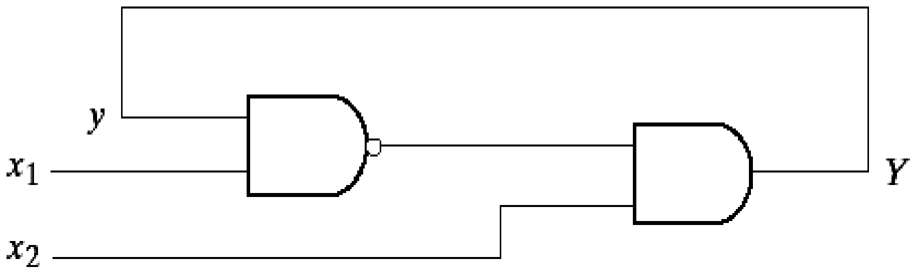
-Must check for stability to ensure proper operations

Can be easily checked from the transition table

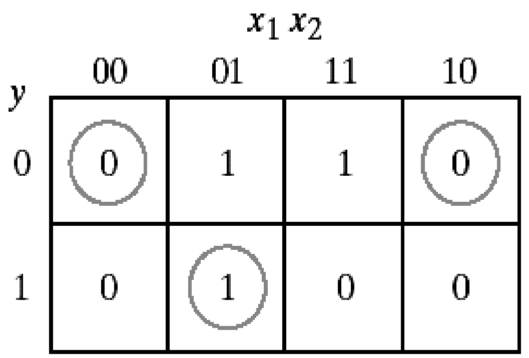
-Any column has no stable states $\Rightarrow$ unstable

-Ex: when x1x2=11 in Fig. 9-9(b), Y and y are never the same

Y = x'1x2 + x2y'



(a) Logic diagram



y \ $x_1 x_2$	00	01	11	10
0	$\textcircled{0}$	1	1	$\textcircled{0}$
1	0	$\textcircled{1}$	0	0

(b) Transition table

Latches in Asynchronous Circuits

-The traditional configuration of asynchronous circuits is using one or more feedback loops

- No real delay elements.

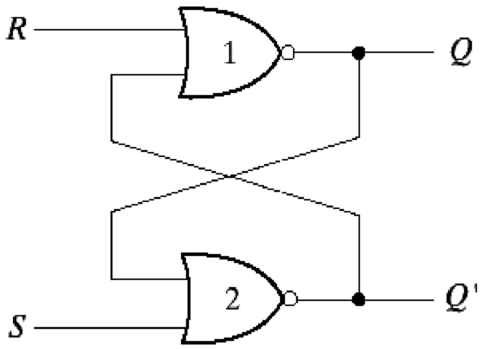
-It is more convenient to employ the SR latch as a memory element in asynchronous circuits

- Produce an orderly pattern in the logic diagram with the memory elements clearly visible.

-SR latch is also an asynchronous circuit

- Will be analyzed first using the method for asynchronous circuits.

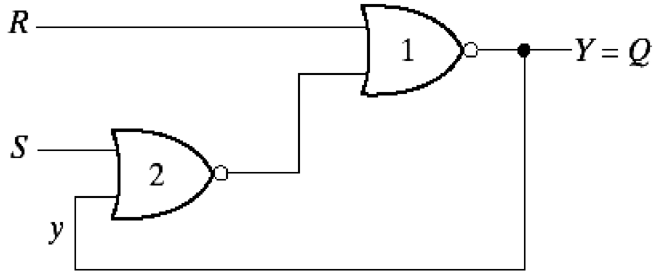
SR Latch with NOR Gates



(a) Crossed-coupled circuit

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(After SR = 10)
0	1	0	1	
0	0	0	1	(After SR = 01)
1	1	0	0	

(b) Truth table



(c) Circuit showing feedback

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

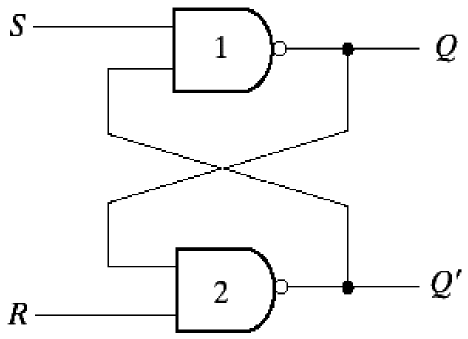
$Y = SR' + R'y$

$Y = S + R'y$ when $SR = 0$ $\Rightarrow$ *** should be carefull** checked first

(d) Transition table

S=1, R=1 (SR = 1) should not be used
⇒ SR = 0 is normal mode

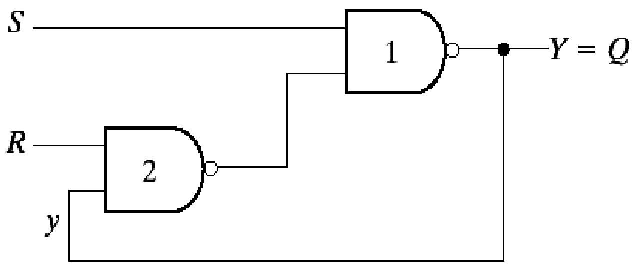
SR Latch with NAND Gates



(a) Crossed-coupled circuit

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(After SR = 10)
0	1	1	0	
1	1	1	0	(After SR = 01)
0	0	1	1	

(b) Truth table



(c) Circuit showing feedback

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

$Y = S' + Ry$ when $S'R' = 0$

(d) Transition table

S=1, R=1 (SR = 1) should not be used ⇒ SR = 0 is normal mode

*** should be carefull**
checked first

Analysis Procedure

Analysis Procedure for NOR latch based asynchronous circuit

(i) Label each latch o/p with Yi and feed back path with yi

(ii) Derive Boolean functions for Si and Ri

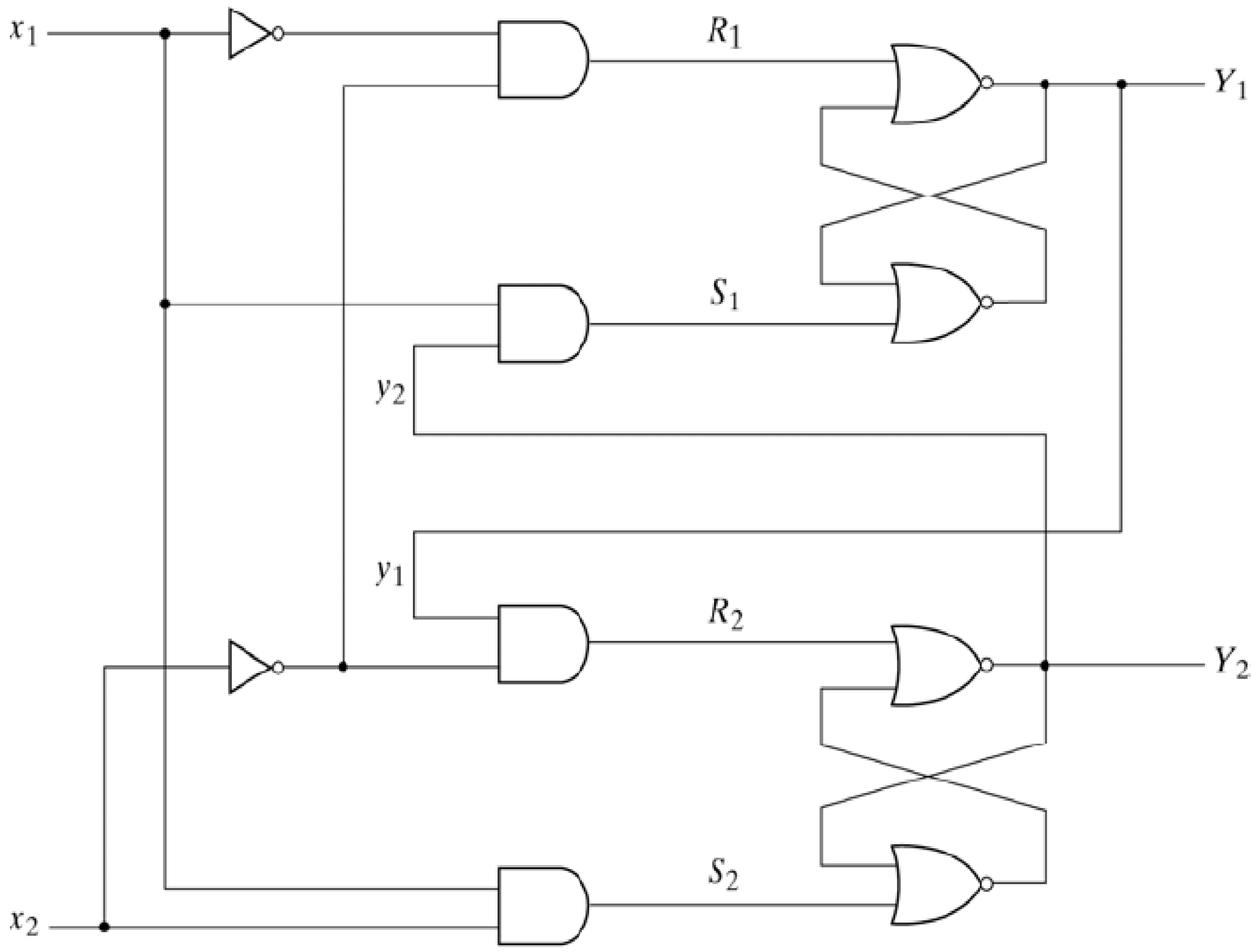
(iii) Check SR = 0 for each NOR latch

(iv) Evaluate Y = S + R'y for each latch

(v) Construct the transition table

(vi) Circle all stable states

Analysis Example



Analysis Example

The procedure for analyzing an asynchronous sequential circuit with SR latches can be summarized as follows:

1. Label each latch output with Yi and its external feedback path with yi for i=1,2,…,k

2. Derive the Boolean functions for the Si and Ri inputs in each latch.

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

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

Analysis Example

3. Check whether SR =0 for each NOR latch or whether S’R’ = 0 for each NAND latch. (if either of these two conditions is not satisfied, there is a possibility that the circuit may not operate properly)

$$S_1R_1 = x_1 y_2 x_1' x_2' = 0$$

$$S_2R_2 = x_1 x_2 x_2' y_1 = 0$$

4. Evaluate Y = S + R’y for each NOR latch or Y = S’ + Ry for each NAND latch.

$$Y_1 = S_1 + R_1' y_1 = x_1 y_2 + (x_1 + x_2) y_1 = x_1 y_2 + x_1 y_1 + x_2 y_2$$

$$Y_2 = S_2 + R_2' y_2 = x_1 x_2 + (x_2 + y_1') y_2 = x_1 x_2 + x_2 y_2 + y_1' y_2$$

Analysis Example

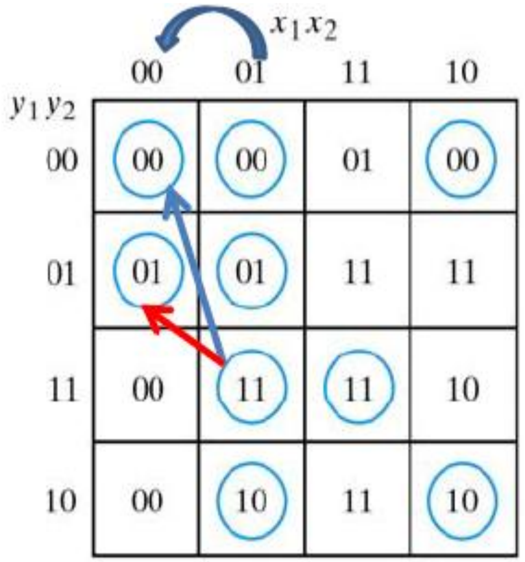
5. Construct a map, with the y's representing the rows and the x inputs representing the columns.
6. Plot the value of Y=Y1Y2…Yk in the map.
7. Circle all stable states such that Y=y. the result is then the transition table.

• The transition table shows that the circuit is **stable**

• Race Conditions: there is a **critical race** condition when the circuit is initially in total state y1y2x1x2 = <u>1101</u> and x2 changes from 1 to 0.

-The circuit should go to the total state <u>0000</u>.

-If Y1 changes to 0 before Y2, the circuit goes to total state <u>0100</u> instead of <u>0000</u>.



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

Transition Table

Implementation Procedure

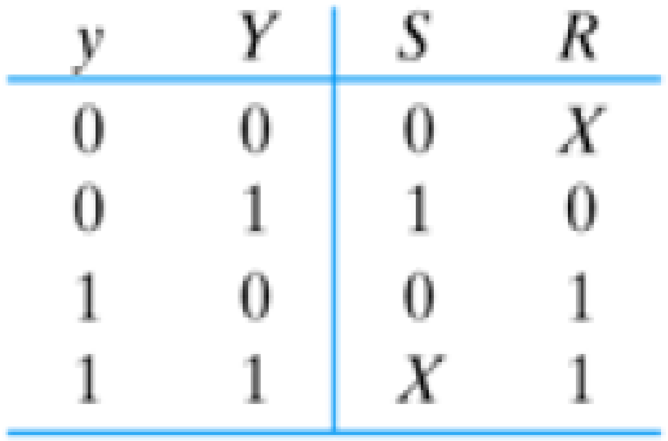
Procedure to implement an asynchronous sequential circuits with SR latches:

1. Given a transition table that specifies the excitation function Y = Y1Y2…Yk, derive a pair of maps for each Si and Ri using th latch excitation table
2. Derive the Boolean functions for each Si and Ri (do not to make Si and Ri equal to 1 in the same minterm square)
3. Draw the logic diagram using k latches together with the gates required to generate the S and R (for NAND latch, use the complemented values in step 2)

Implementation Procedure

Latch Excitation Table

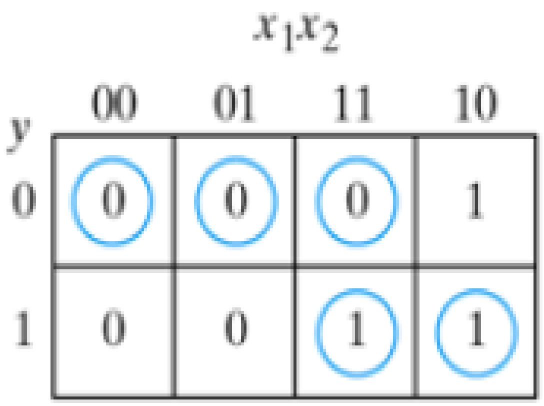
- During the implementation process, the transition table of the circuit is available and we wish to find the values of S and R .
- Excitation table: Lists the required inputs S and R for each of the possible transition from y to Y.



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

Implementation Example

- Given a transition table that specifies the excitation function Y=Y1Y2 cedure for implementing a circuit ımarized as follows:



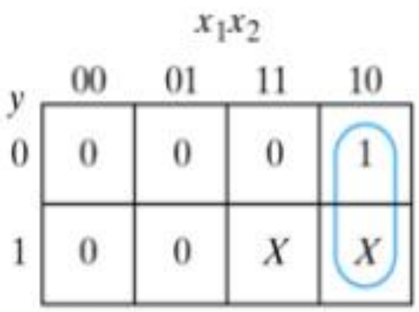
y \ x_1x_2	00	01	11	10
0	0	0	0	1
1	0	0	1	1

(a) Transition table

$Y = x_1x'_2 + x_1y$

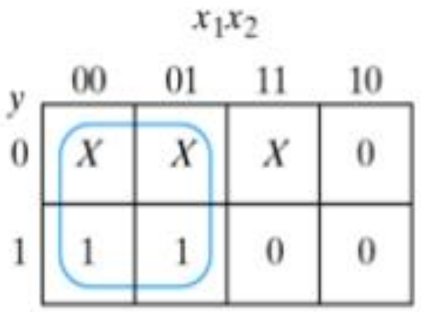
Implementation Example

1. Derive a pair of maps for Si and Ri for each I = 1, 2,…,k. (This is done by using the latch excitation table)



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

(c) Map for $S = x_1x'_2$



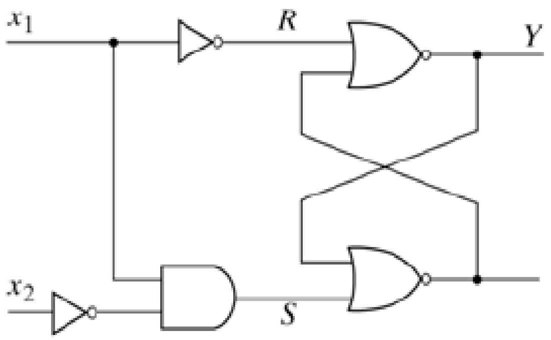
y \ x_1x_2	00	01	11	10
0	X	X	X	0
1	1	1	0	0

(d) Map for $R = x'_1$

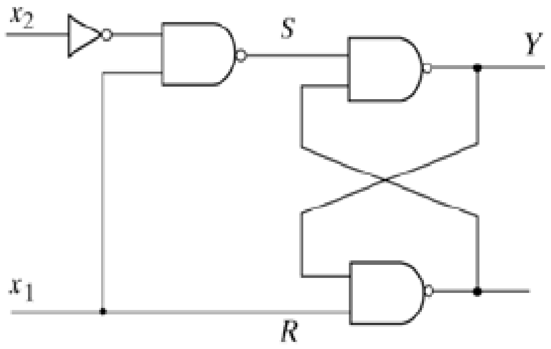
Implementation Example

	NOR Latch	NAND Latch
S=	$X_1X'_2$	$(X_1X'_2)'$
R=	X'_1	X_1

2. Draw the logic diagram, using k latches together with the gates required to generate the S and R Boolean functions obtained in step1 (for NAND latches, use the complemented val



(e) Circuit with NOR latch



(f) Circuit with NAND latch