University Of Diyala
College Of Engineering
Department of Computer Engineering

Digital System Design II
Asynchronous Sequential Logic

Dr. Yasir Al-Zubaidi

Third stage

2021

Outline

- **Asynchronous Sequential Circuits**
- Analysis Procedure
- Circuits with Latches
- Design Procedure
- Reduction of State and Flow Tables
- Race-Free State Assignment
- Hazards
- Design Example

Sequential Circuits

- Consist of a combinational circuit to which storage elements are connected to form a feedback path
- Specified by ***a time sequence of inputs, outputs,*** and ***internal states***
- Two types of sequential circuits:
  - Synchronous
  - Asynchronous

primary difference

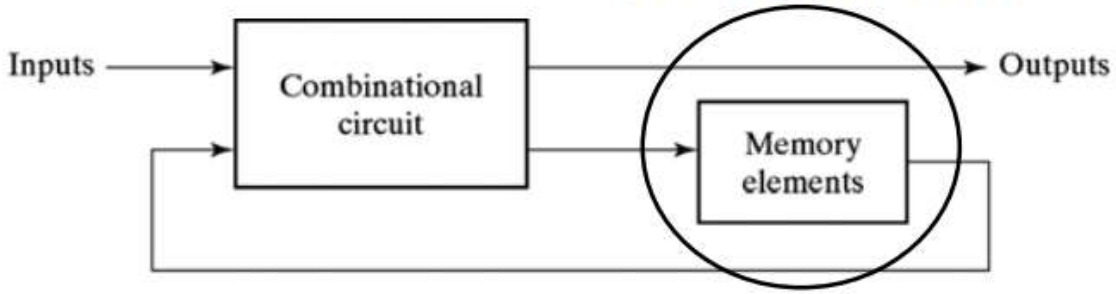
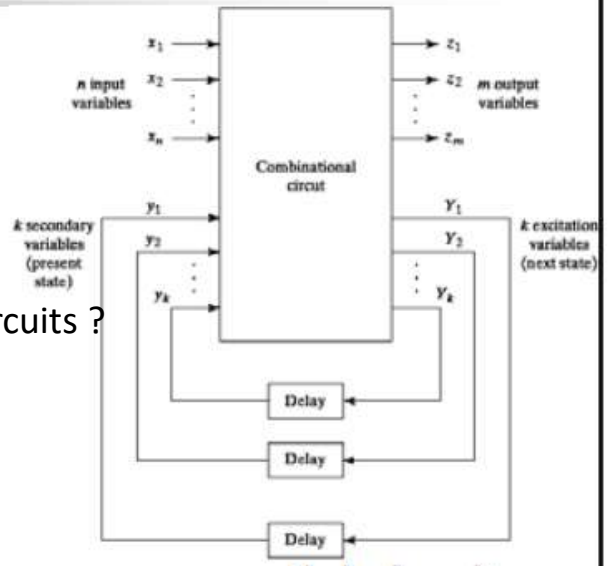


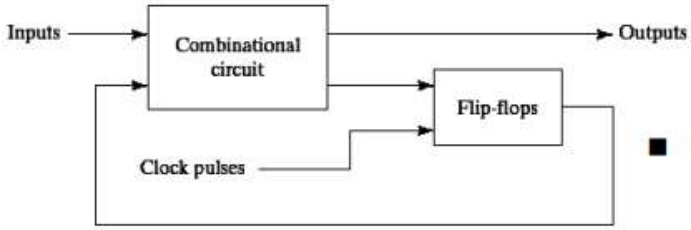
Fig. 5-1 Block Diagram of Sequential Circuit

Synchronous vs. Asynchronous

- **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



Why Asynchronous Circuits ?



(a) Block diagram

(b) Timing diagram of clock pulses

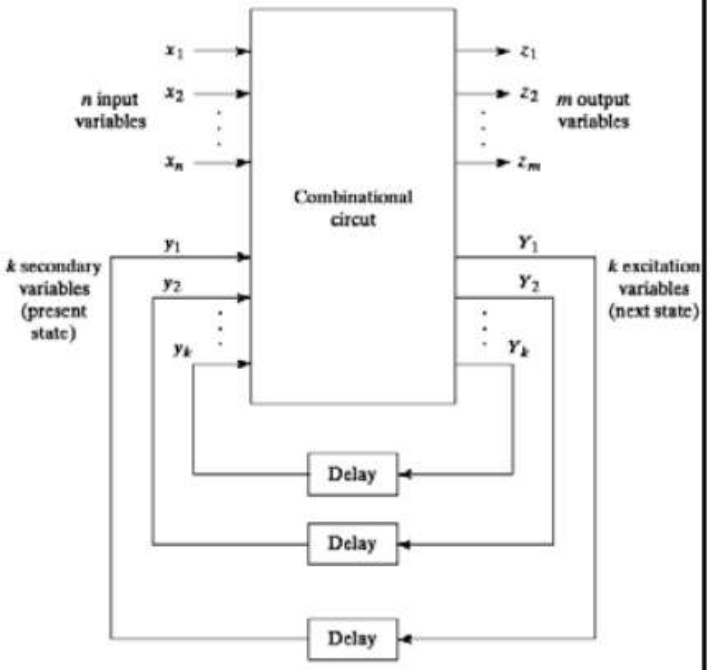
- **Synchronous sequential circuits**
  - Synchronized by a ***periodic*** train of clock pulses
  - Much easier to design (preferred design style)

Why Asynchronous Circuits ?

- Used when speed of operation is important
  - Response quickly without waiting for a clock pulse
- Used in small independent systems
  - Only a few components are required
- Used when the input signals may change independently of internal clock
  - Asynchronous in nature
- Used in the communication between two units that have their own independent clocks
  - Must be done in an asynchronous fashion

Definitions of Asyn. Circuits

- Inputs / Outputs
- Delay elements:
  - Only a short term memory
  - May not really exist due to original gate delay
- Secondary variable:
  - Current state (small y)
- Excitation variable:
  - Next state (big Y)
  - Have some delay in response to input changes



Operational Mode

- **Steady-state condition:**
  - **Current states and next states are the same**
  - **Difference between Y and y will cause a transition**
- **Fundamental mode:**
  - **No simultaneous changes of two or more variables**
  - **The time between two input changes must be longer than the time it takes the circuit to a stable state**
  - **The input signals change one at a time and only when the circuit is in a stable condition**

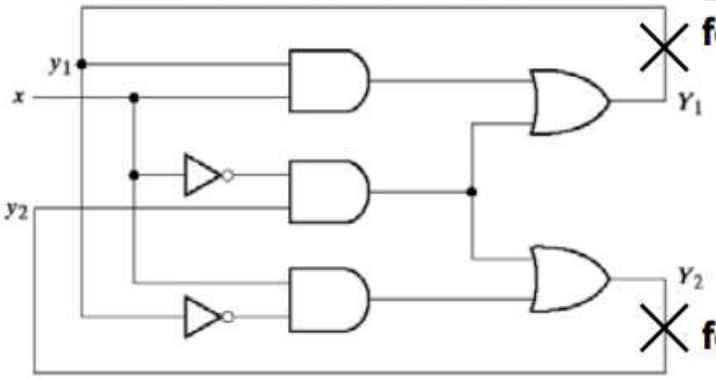
Outline

- Asynchronous Sequential Circuits
- **Analysis Procedure**
- Circuits with Latches
- Design Procedure
- Reduction of State and Flow Tables
- Race-Free State Assignment
- Hazards
- Design Example

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 (y_i) and output (Y_i) 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

An Example of Transition Table



$$Y_1 = xy_1 + x'y_2$$

$$Y_2 = xy'_1 + x'y_2$$

$$Y = Y_1Y_2$$

inputs

current states

y_1y_2	$x=0$	$x=1$
00	0	0
01	1	0
11	1	1
10	0	1

(a) Map for $Y_1 = xy_1 + x'y_2$

y_1y_2	$x=0$	$x=1$
00	0	1
01	1	1
11	1	0
10	0	0

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

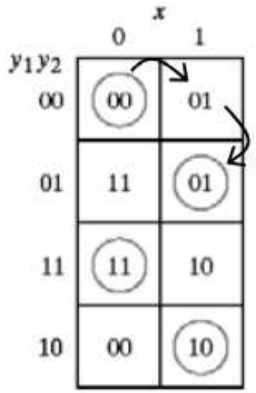
y_1y_2	$x=0$	$x=1$
00	(00)	01
01	11	(01)
11	(11)	10
(10)	00	(10)

(c) Transition table

stable !!

State Table

- When input x changes from 0 to 1 while y=00:
  - Y changes to 01 → unstable
  - y becomes 01 after a short delay → stable at the second row
  - The next state is Y=01
- Each row must have ***at least one*** stable state
- Analyze each state in this way can obtain its state table



(c) Transition table

Present State		Next State			
		X=0		X=1	
0	0	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	1	1	1	1	0

y_1y_2x : total state

4 stable total states: 000,011, 110,101

Flow Table

- Similar to a transition table except the states are represented by ***letter symbols***
- Can also include the output values
- Suitable to obtain the logic diagram from it
- Primitive flow table: ***only one*** stable state in each row (ex: 9-4(a))

	x = 0	x = 1
a	(a)	b
b	c	(b)
c	(c)	d
d	a	(d)

(a) Four states with one input

	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

(b) Two states with two inputs and one output

Equivalent to 9-3(c) if a=00, b=01, c=11, d=10

Flow Table to Circuits

- **Procedure to obtain circuits from flow table:**
  - **Assign to each state a distinct binary value (convert to a transition table)**
  - **Obtain circuits from the map**
- **Two difficulties:**
  - **The binary state assignment (to avoid race)**
  - **The output assigned to the unstable states**

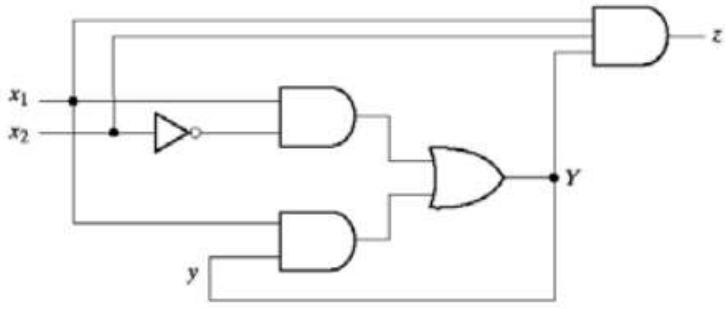
Ex: from the flow table 9-4(b)

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

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

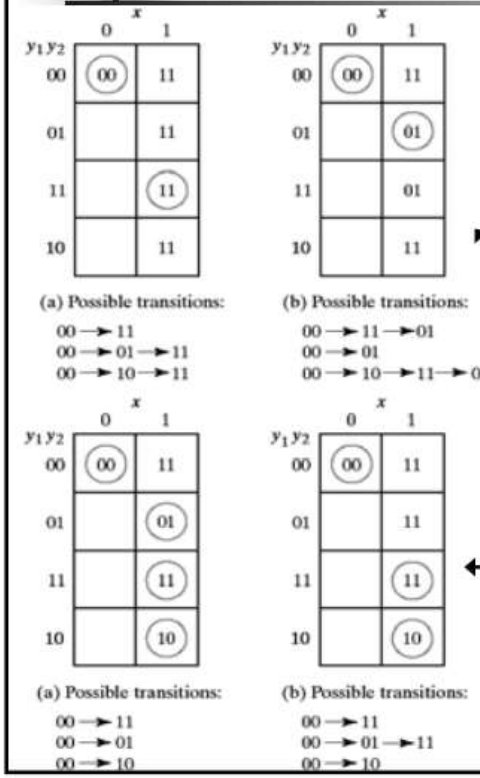
y \ x_1x_2	00	01	11	10
0	0	0	0	0
1	0	0	1	0

(b) Map for output
$z = x_1x_2y$



(c) Logic diagram

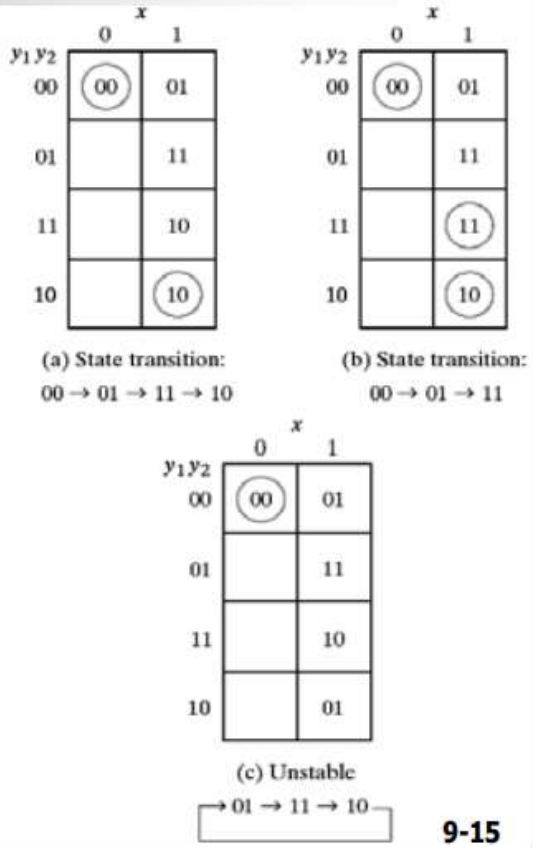
Race Conditions



- **Race condition:**
  - ***two or more*** **binary state variables will change value when one input variable changes**
  - **Cannot predict state sequence if unequal delay is encountered**
- **Non-critical race:**
  - **The final stable state *does not* depend on the change order of state variables**
- **Critical race:**
  - **The change order of state variables will result in *different* stable states**
  - **Should be avoided !!**

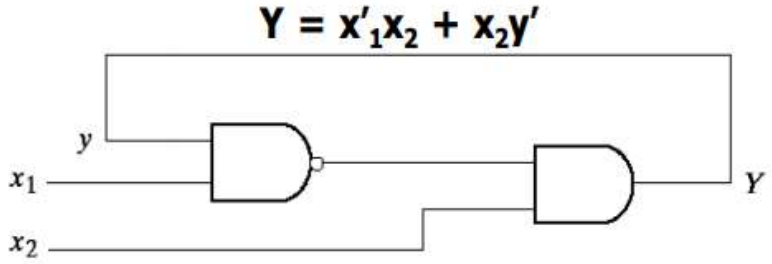
Race-Free State Assignment

- Race can be avoided by proper state assignment
  - Direct the circuit through intermediate unstable states with a unique state-variable change
  - It is said to have a ***cycle***
- Must ensure that a cycle will terminate with a stable state
  - Otherwise, the circuit will keep going in unstable states
- More details will be discussed in Section 9-6



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 → unstable
  - Ex: when x_1x_2=11 in Fig. 9-9(b), Y and y are never the same



(a) Logic diagram

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

(b) Transition table

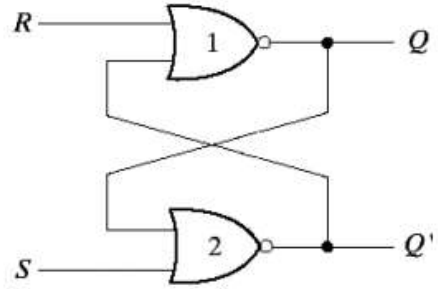
Outline

- Asynchronous Sequential Circuits
- Analysis Procedure
- **Circuits with Latches**
- Design Procedure
- Reduction of State and Flow Tables
- Race-Free State Assignment
- Hazards
- Design Example

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

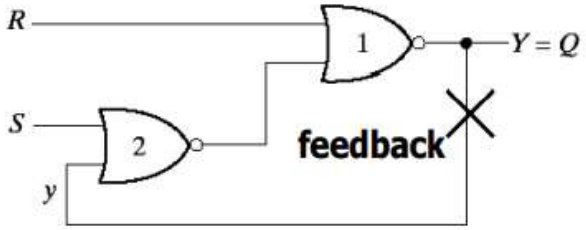
SR Latch with NOR Gates



(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 $SR = 0$ → *

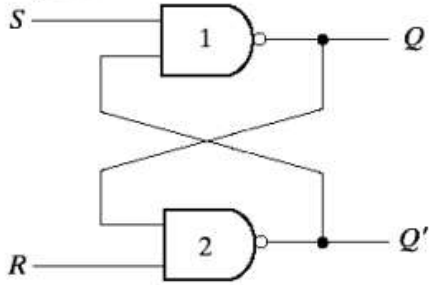
(d) Transition table

S=1, R=1 (SR = 1) should not be used

⇒ SR = 0 is normal mode

should be carefully checked first

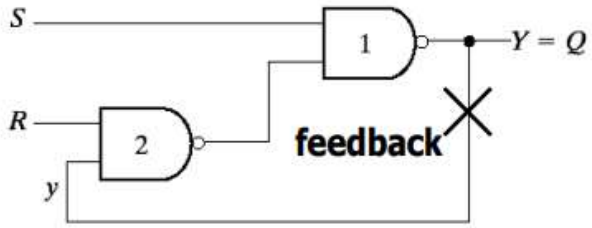
SR Latch with NAND Gates



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

*** should be carefully checked first**

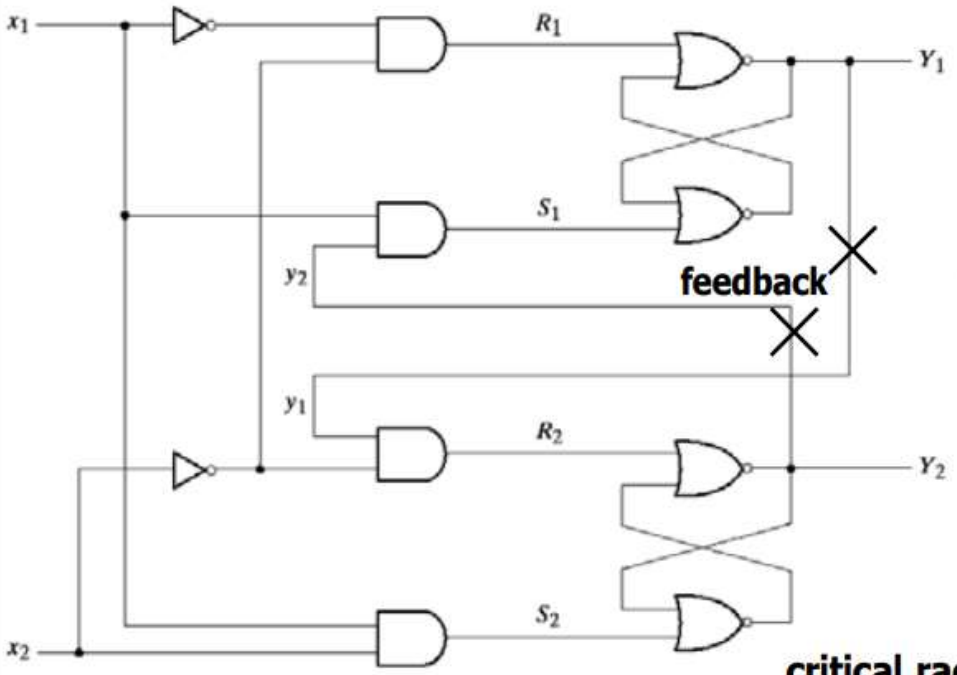
Analysis Procedure

- **Procedure to analyze an asynchronous sequential circuits with SR latches:**
  1. Label each latch output with Y_i and its external feedback path (if any) with y_i
  2. Derive the Boolean functions for each S_i and R_i
  3. Check whether $SR=0$ (NOR latch) or $S'R'=0$ (NAND latch) is satisfied
  4. Evaluate $Y=S+R'y$ (NOR latch) or $Y=S'+Ry$ (NAND latch)
  5. Construct the transition table for $Y=Y_1Y_2\ldots Y_k$
  6. Circle all stable states where $Y=y$

Analysis Example

$S_1=x_1y_2$ $R_1=x'_1x'_2 \Rightarrow S_1R_1 = x_1y_2x'_1x'_2 = 0$ (OK)
$S_2=x_1x_2$ $R_2=x'_2y_1 \Rightarrow S_2R_2 = x_1x_2x'_2y_1 = 0$ (OK)

$Y_1=S_1 + R'_1y_1$
$=x_1y_2 + (x_1+x_2)y_1$
$=x_1y_2+x_1y_1+x_2y_1$
$Y_2=S_2 + R'_2y_2$
$=x_1x_2 + (x_2+y'_1)y_2$
$=x_1x_2+x_2y_2+y'_1y_2$



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

critical race !!

Implementation Procedure

- **Procedure to implement an asynchronous sequential circuits with SR latches:**
  1. **Given a transition table that specifies the excitation function $Y = Y_1 Y_2 \ldots Y_k$, derive a pair of maps for each S_i and R_i using the latch excitation table**
  2. **Derive the Boolean functions for each S_i and R_i**
     (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 Example

Excitation table: list the required S and R for each possible transition from y to Y

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

(a) Transition table

$Y = x_1x'_2 + x_1y$

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

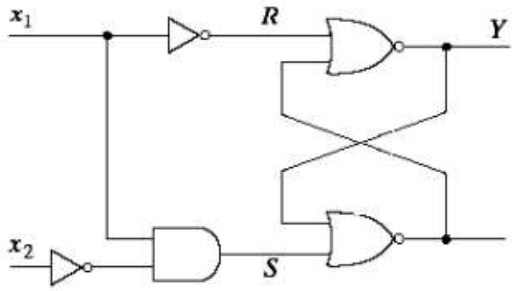
(b) Latch excitation table

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

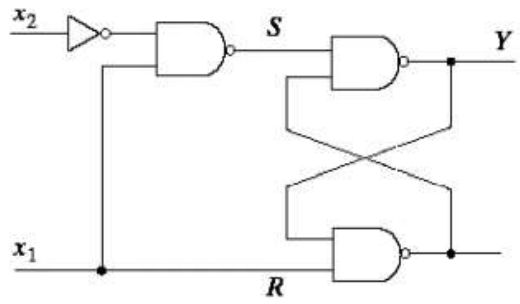
(c) Map for $S = x_1x'_2$

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

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



(e) Circuit with NOR latch



(f) Circuit with NAND latch

y = 1 (outside) → 0 (inside)

∴ S=0, R=1 from excitation table

Debounce Circuit

- Mechanical switches are often used to generate binary signals to a digital circuit
  - It may vibrate or bounce several times before going to a final rest
  - Cause the signal to oscillate between 1 and 0
- A debounce circuit can remove the series of pulses from a contact bounce and produce a single smooth transition
  - Position A (SR=01) → bouncing (SR=11) → Position B (SR=10)

    Q = 1 (set) → Q = 1 (no change) → Q = 0 (reset)