Digital Logic Design

Hajar Falahati

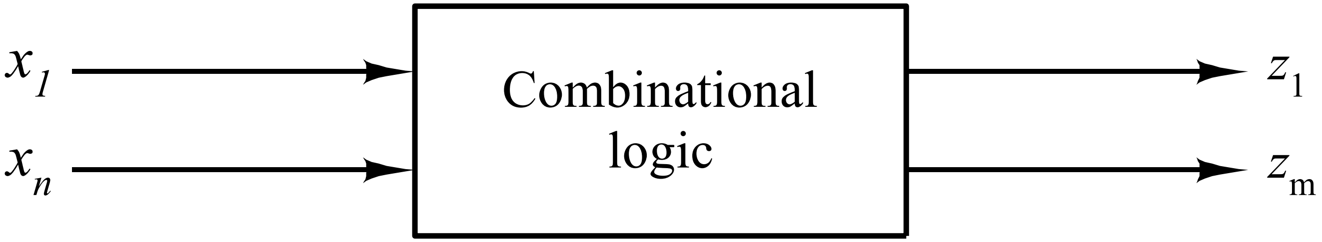
Department of Computer Engineering
IRAN University of Science and Technology

hfalahati@iust.ac.ir

Combinational Logics

- **Always** **Current inputs** produce the **output**
- **Output** is **independent** of
  - **Sequence of inputs**
  - **Time of applying inputs**

- => **Combinational logics** are memory less
  - **Memory-less circuits** do not contain any **feedback** lines



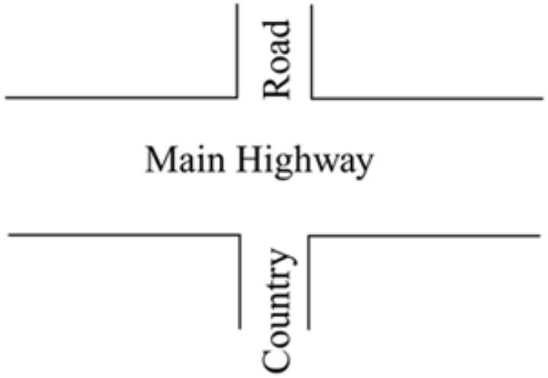
Outline

- Introduction to sequential logics
- Memory devices
  - Latch
  - Flip Flops (FF)

Sequential Logics

Traffic Signal Controller

- What is the functionality of a traffic signal controller?
- Inputs
  - Signal X
    - Detect cars waiting on the country road
    - X = 1 if there are cars on the country road; otherwise, X= 0

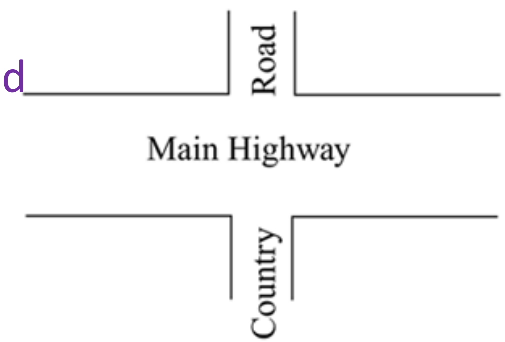


- Output
  - Manages the traffic signals for both roads
  - Gets highest priority to the main road
    - The highway traffic signal becomes green
  - If there is a car on the country road
    - The country-road traffic signal becomes green

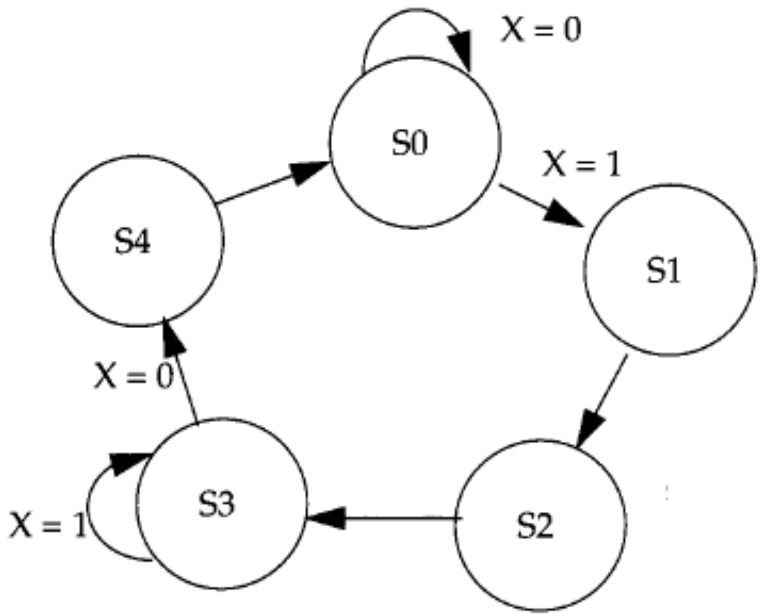
Traffic Signal Controller: More Detail

- Traffic Signal for main highway gets highest priority
  - Cars are continuously present on the main highway
  - Main highway signal **remains** green by default

- Traffic signal for the country road
  - Must turn green only long enough to let the cars on the country road go
  - **As soon as** there are no cars on the country road
    - Country road traffic signal turns yellow and then red
    - Traffic signal on the main highway turns green again
  - There is a **sensor** to detect cars waiting on the country road
    - Sends a signal X as input to the controller.
    - $X = 1$ if there are cars on the country road; otherwise, $X = 0$.



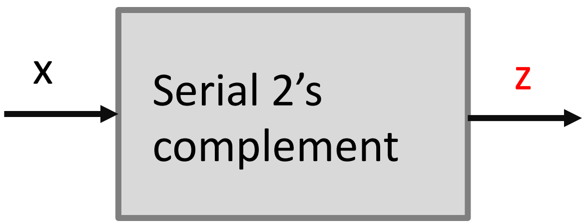
Sample Circuit 1: Traffic Signal Controller (cont’d)



State	Signals
S0	*Hwy* = *G* *Cntry* = *R*
S1	*Hwy* = *Y* *Cntry* = *R*
S2	*Hwy* = *R* *Cntry* = *R*
S3	*Hwy* = *R* *Cntry* = *G*
S4	*Hwy* = *R* *Cntry* = *Y*

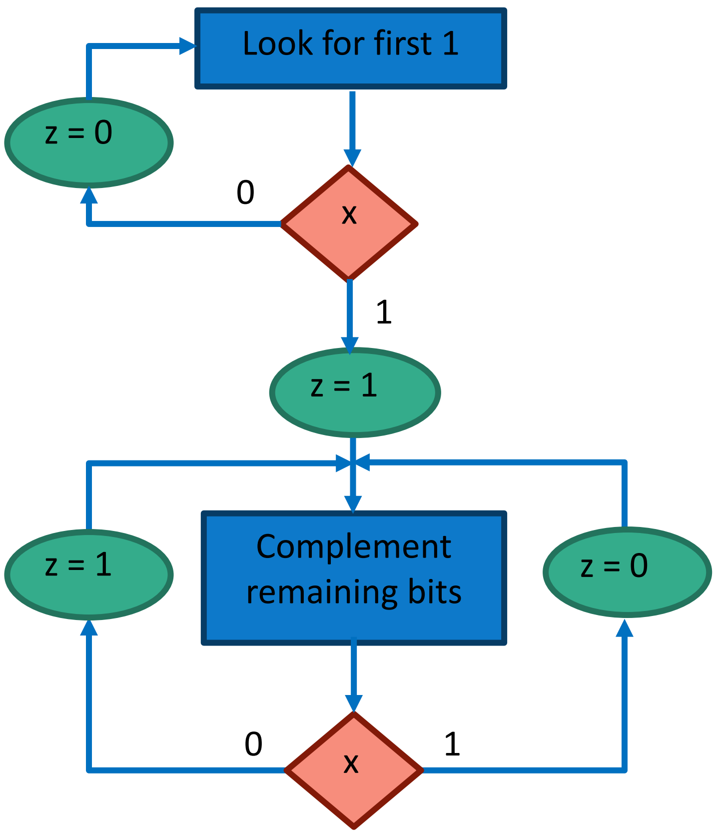
Sample Design 2

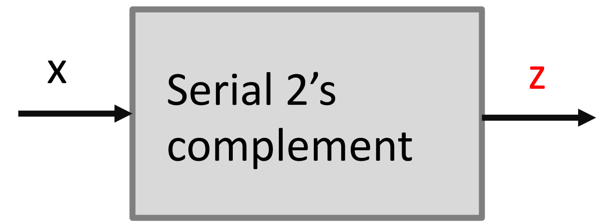
- Design a control unit for a serial 2's complement

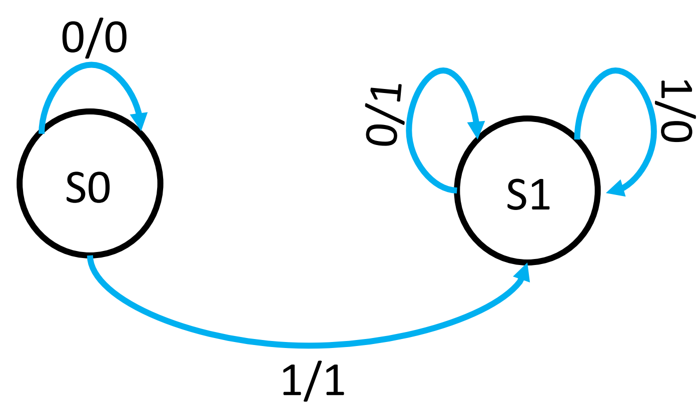


t	x	z
0	0	0
1	0	0
2	1	1
3	1	0

Serial 2’s Complement







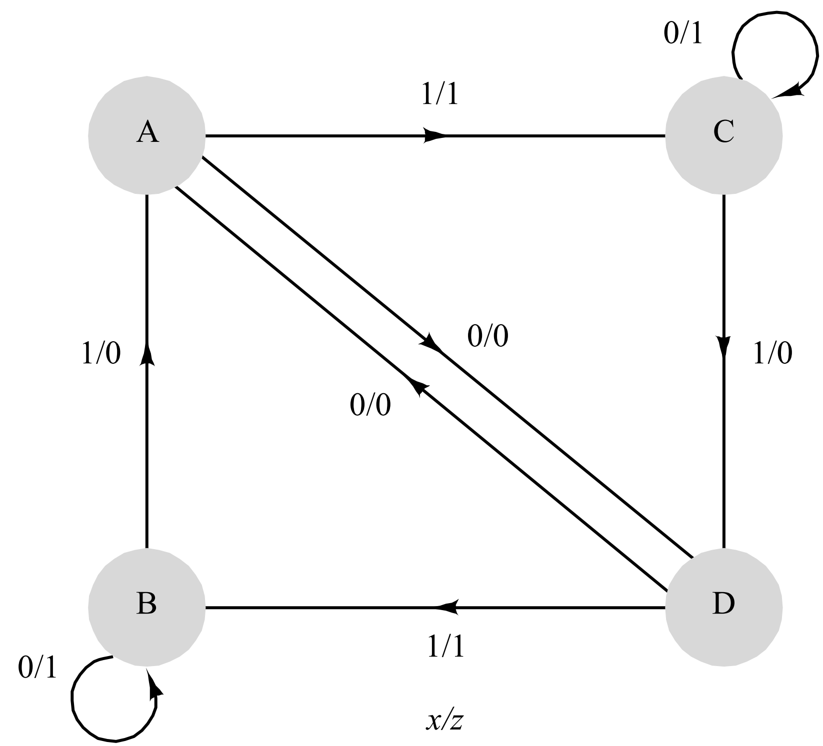
Sample Circuit 3

- What is the functionality of this circuit?
  - **Input =0**
  - **Input =1**

- Each line shows a pair (input, output)

		Input x 0	1
Present state	*A*	*D*/0	*C*/1
	B	*B*/1	*A*/0
	C	*C*/1	*D*/0
	D	*A*/0	*B*/1

(a)

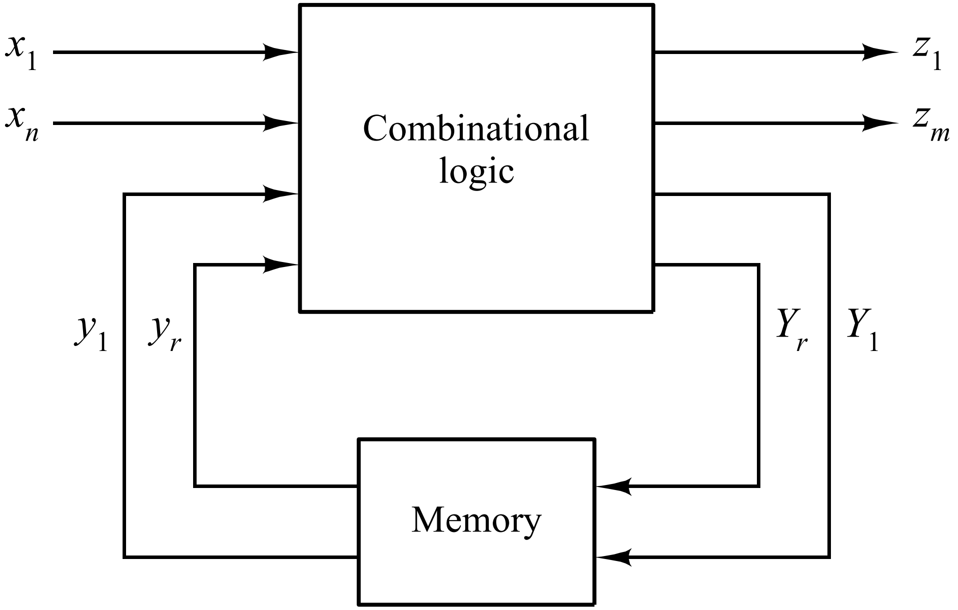


(b)

Combinational Logics + Memory

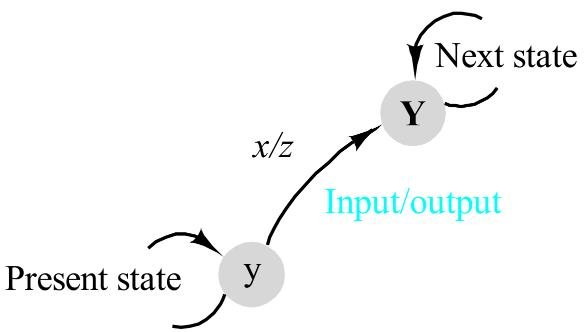
- **Output** of some logics **changes** by
  - **Sequence of inputs**
  - **Time of applying inputs**

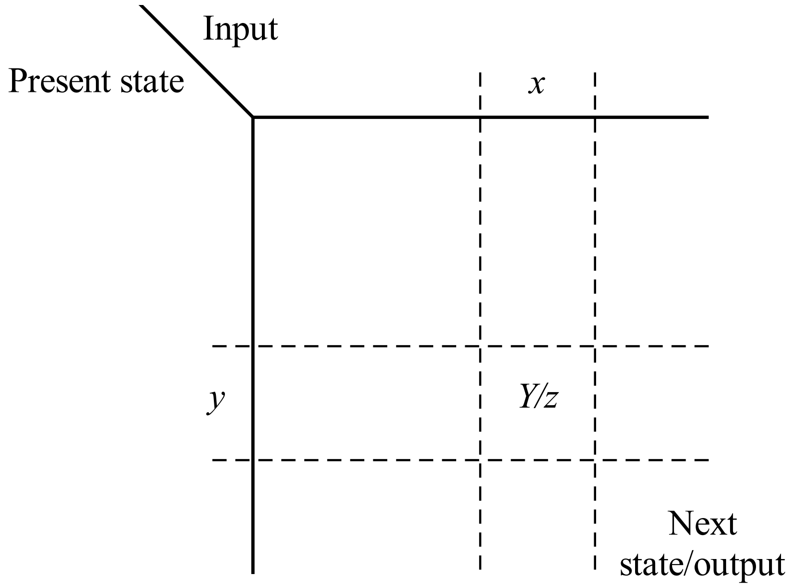
- => They **have memory**
- They **contain feedback lines**

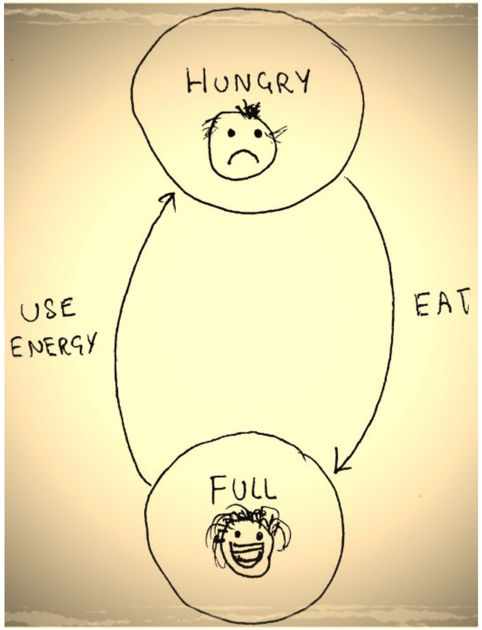


State

- Produced by previous inputs
- Information stored in a memory



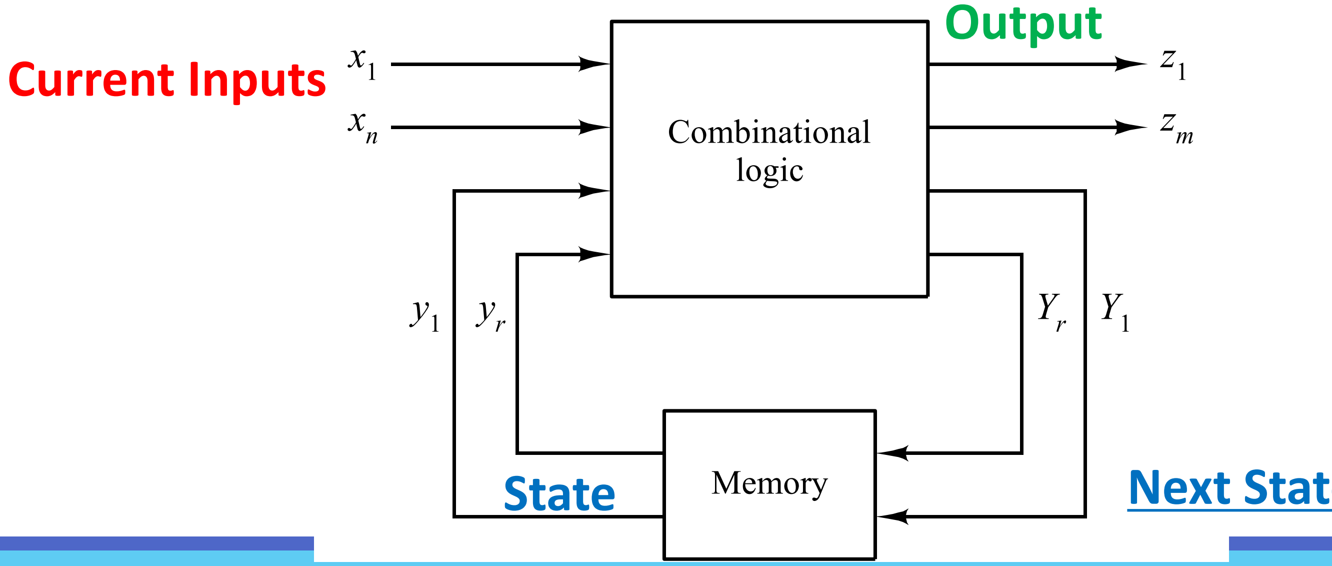




Sequential Logic

- Produce **OUTPUTS** by considering
  - **Current inputs**
  - **Current states (state)**
    - **Current outputs generated by previous inputs**

- Next State outputs are inputs to storage elements



Memory Devices

Memory Unit

- An element which **stores <u>one bit</u>** data

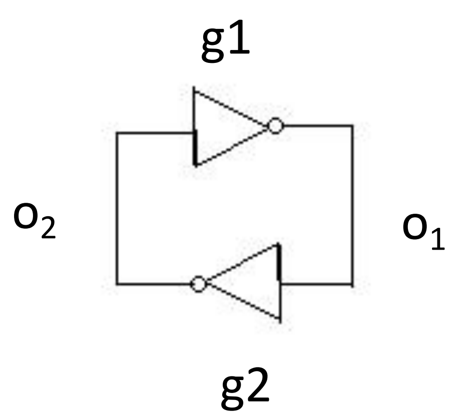
- How to store one bit data?
  - A system should have **two stable states <u>at least</u>**

- Stable state
  - Systems **stays** in this state
  - Does **not** need to any **outside signal**

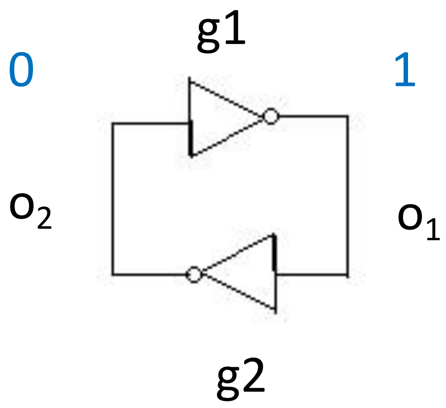
- Sample
  - Put a glass on the table

Back to Back Inverters

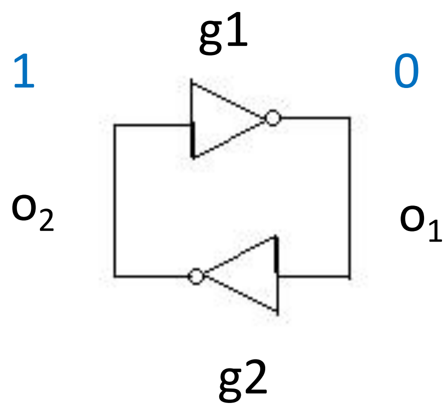
- Consider this circuit
- What is its functionality?
  - Consider O_2 as the output



Back to Back Inverters: Functionality



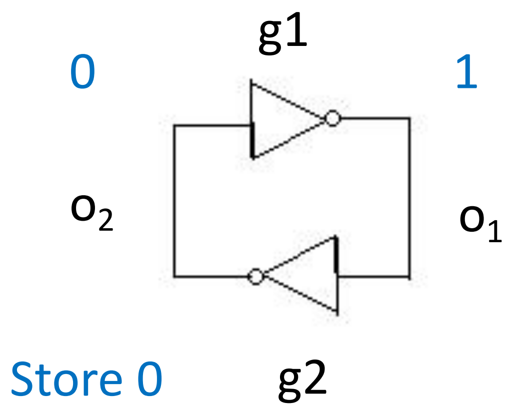
Store 0

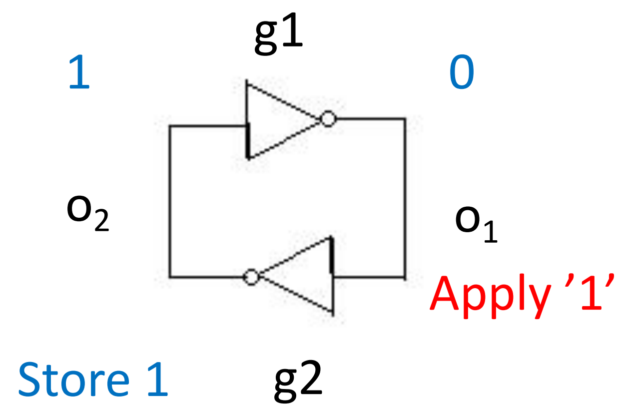


Store 1

Back to Back Inverters: How to update?

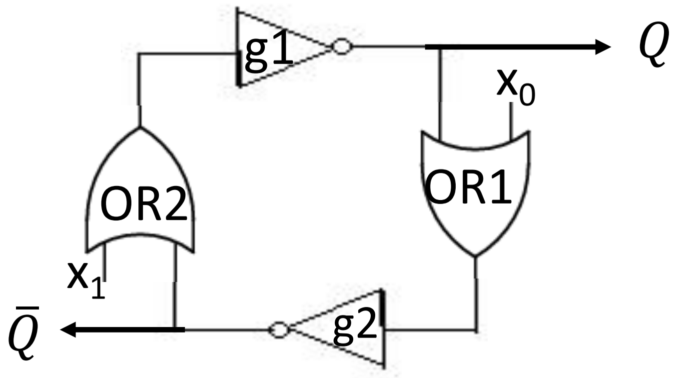
- How to update?
  - Apply ‘1’
- How to write?
  - We can not write
- Why?
  - We can not change the output of a logic gate
  - => Damage the circuit
  - => Incorrect functionality





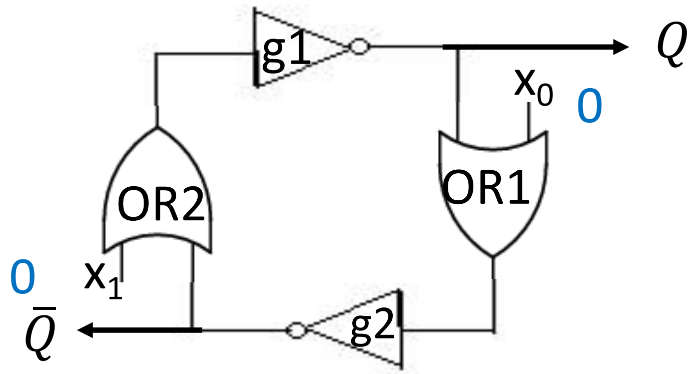
Back to Back Inverters: How to write?

- How to write?
  - Insert two OR gate

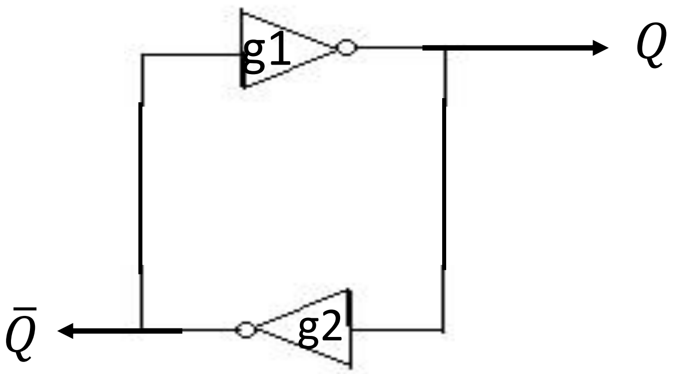


Back to Back Inverters + 2 OR Gate: Functionality

- $x_0 = x_1 = 0$
- Q?

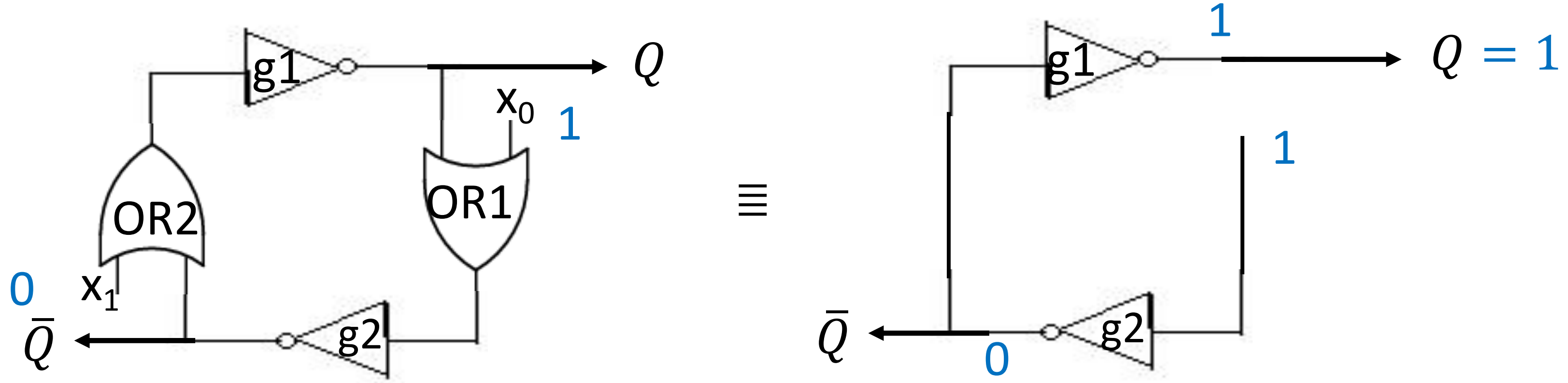


≡



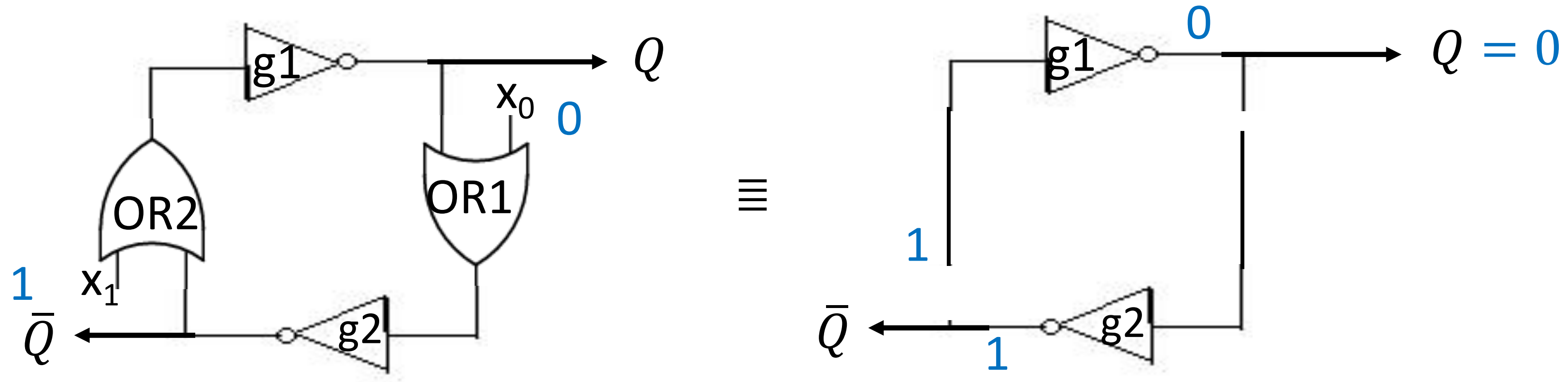
Back to Back Inverters + 2 OR Gate: Functionality (cont’d)

- $x_0 = 1$
- $x_1 = 0$
- Q?



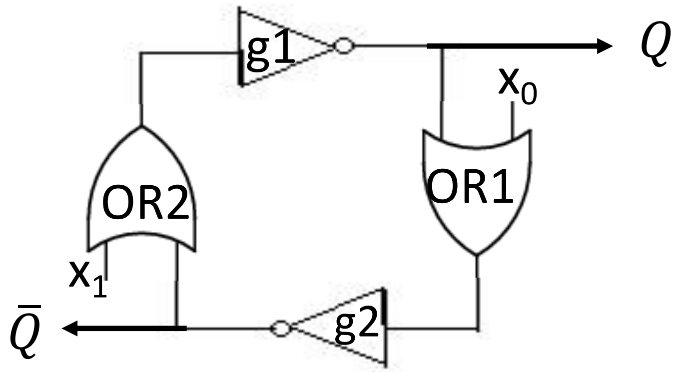
Back to Back Inverters + 2 OR Gate: Functionality (cont'd)

- $x_0 = 0$
- $x_1 = 1$
- Q?

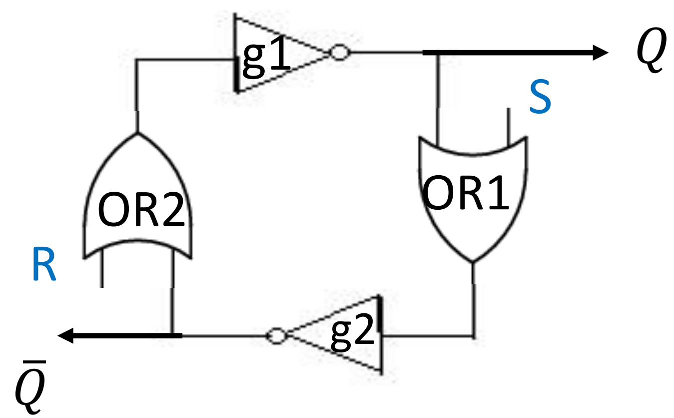


Back to Back Inverters + 2 OR Gate: Functionality: Summary

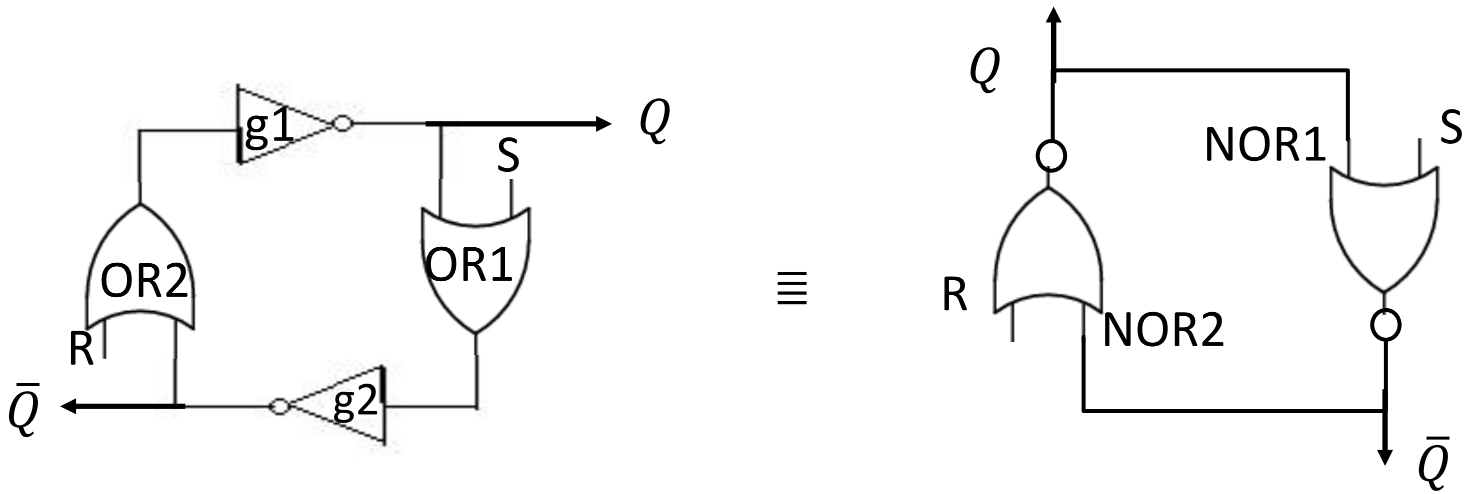
- x_1 x_0
  - 00 => Hold the value
  - 10 => Write 0
  - 01 => Write 1



- x_0
  - Set input (S)
- x_1
  - Reset input (R)



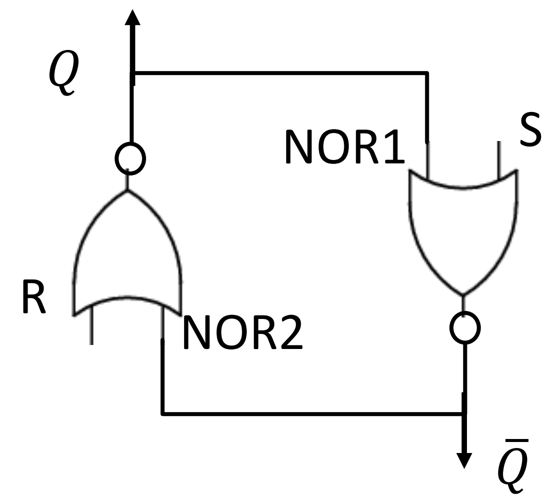
Back to Back Inverters + 2 NOR Gate



Set-Reset Latch (S-R Latch)

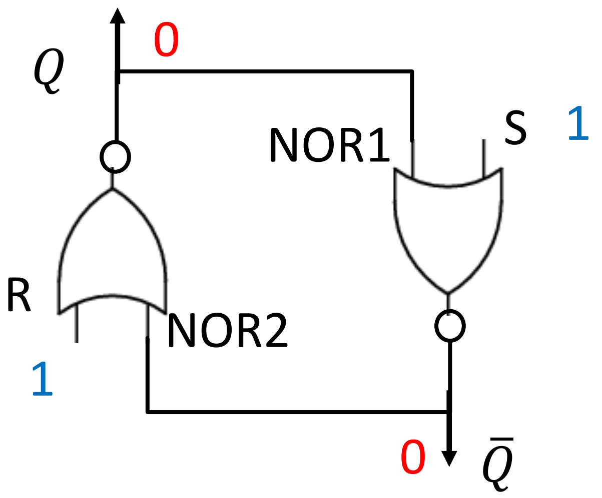
- A circuit can store one bit data
- Two inputs
  - R
  - S

R	S	Q	$\bar{Q}$	
0	0	Q	$\bar{Q}$	Hold the value
0	1	1	0	Set the value
1	0	0	1	Reset the value
1	1	Invalid		Invalid



S-R Latch: RS =11?

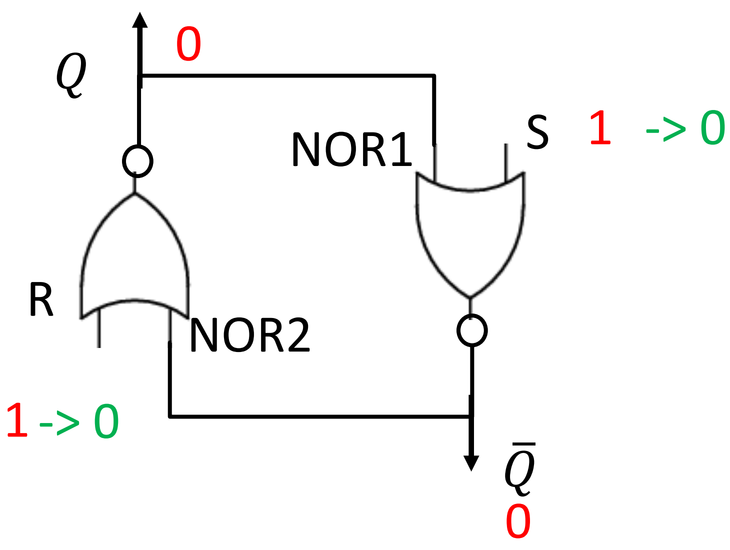
- $RS = 11$
  - $Q = \bar{Q} = 0$
  - **Invalid input**



S-R Latch: $RS = 11 \rightarrow RS = 00$?

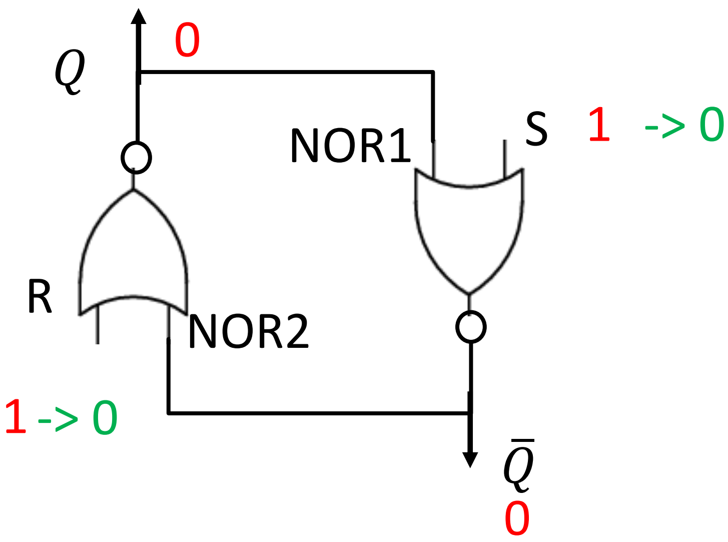
- $RS = 11 \rightarrow RS = 00$
  - $Q = ?$
  - $\bar{Q} = ?$

- $RS = 00$
  - Hold the value!
  - **Does it hold the invalid value?**



S-R Latch: $RS = 11 \rightarrow RS = 00$? (cont'd)

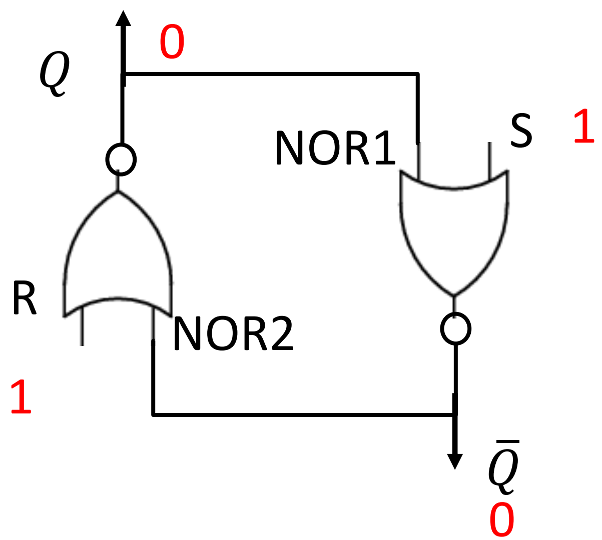
- $RS = 11 \rightarrow RS = 00$
- There are two conditions
  - $RS = 11 \rightarrow RS = 01 \rightarrow RS = 00$
  - $RS = 11 \rightarrow RS = 10 \rightarrow RS = 00$



- Why?
  - Race
- Race
  - When **two inputs change** at the **same time**
  - In **reality**, their values **do not become updated** at the **same time**

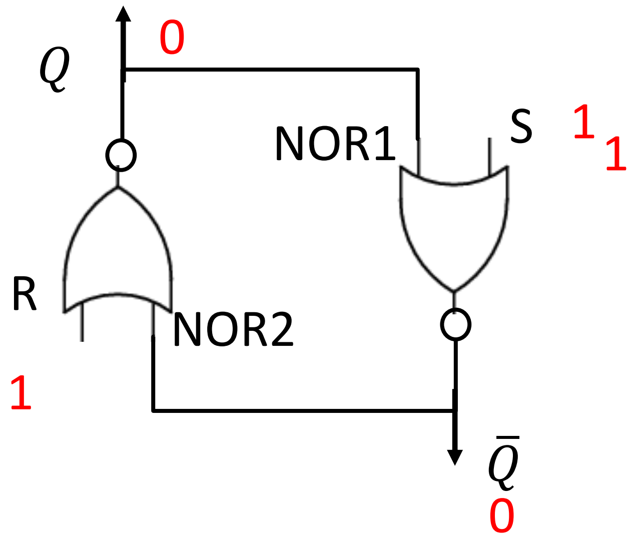
S-R Latch: $RS = 11 \rightarrow RS = 00$? (cont'd)

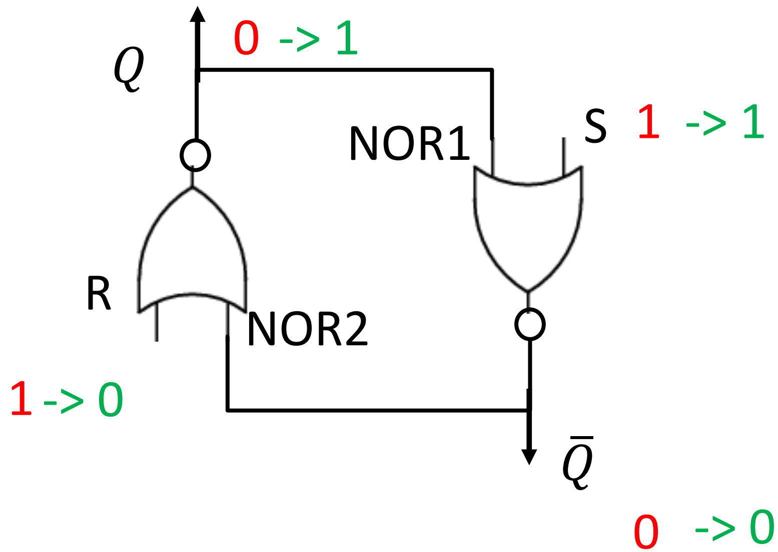
- $RS = 11 \rightarrow RS = 01 \rightarrow RS = 00$



S-R Latch: $RS = 11 \rightarrow RS = 00$? (cont'd)

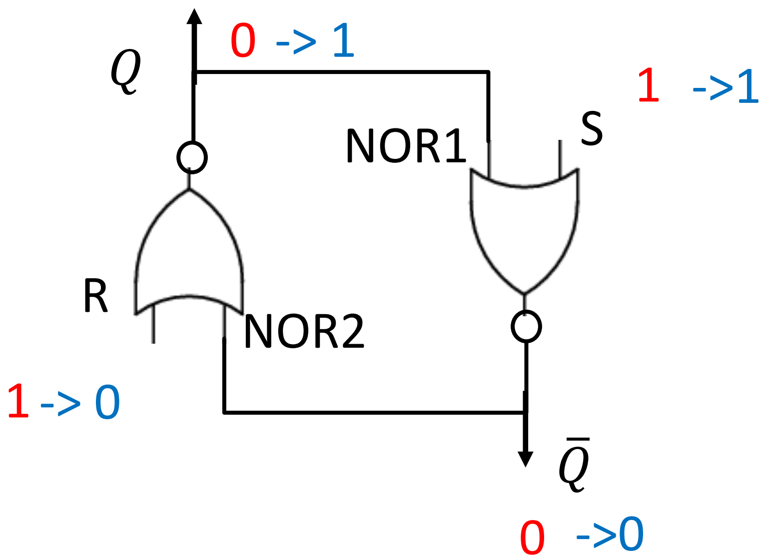
- $RS = 11 \rightarrow RS = 01 \rightarrow RS = 00$

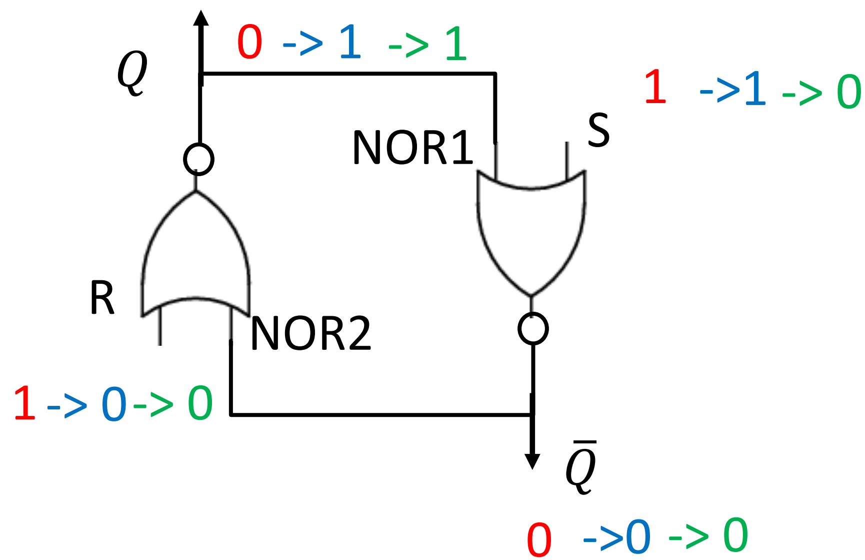




S-R Latch: $RS = 11 \rightarrow RS = 00$? (cont'd)

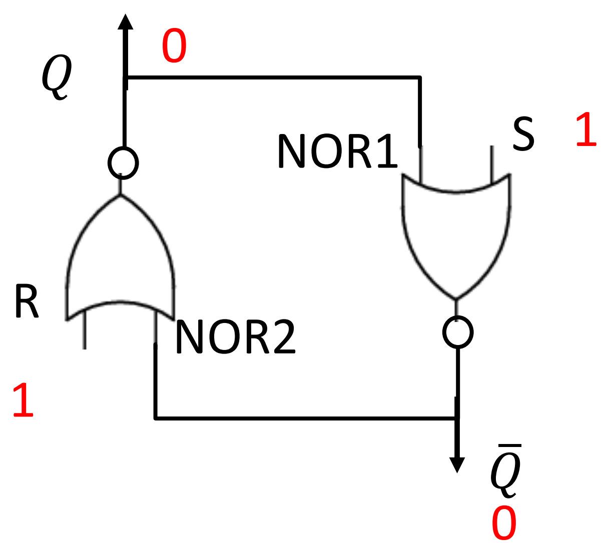
- $RS = 11 \rightarrow RS = 01 \rightarrow RS = 00$





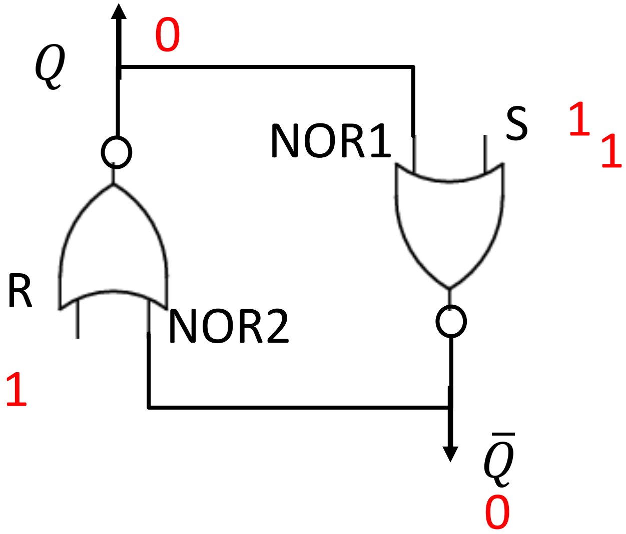
S-R Latch: $RS = 11 \rightarrow RS = 00$? (cont'd)

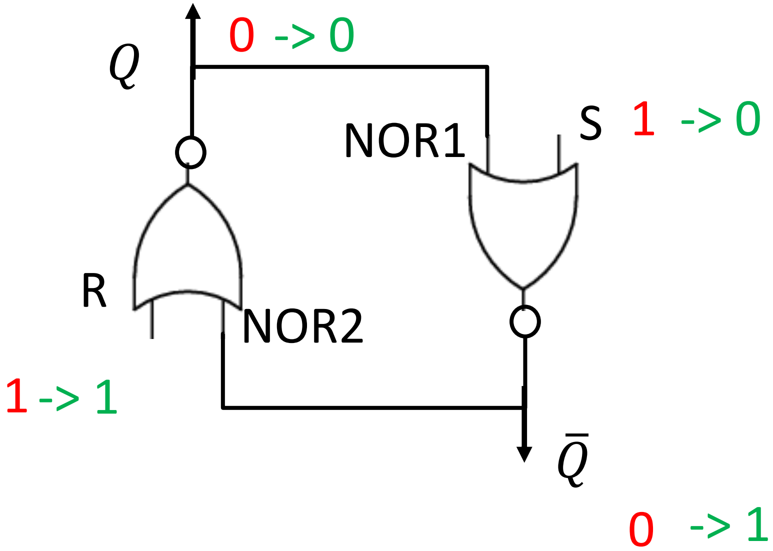
- $RS = 11 \rightarrow RS = 10 \rightarrow RS = 00$



S-R Latch: $RS = 11 \rightarrow RS = 00$? (cont'd)

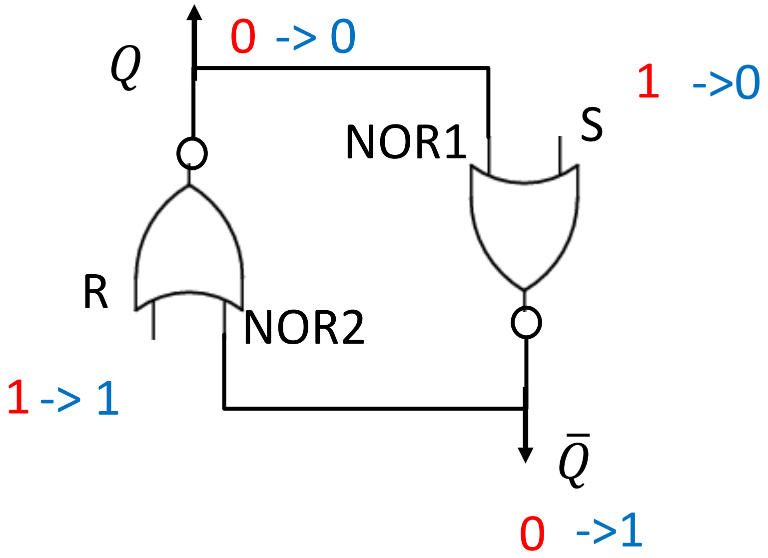
- $RS = 11 \rightarrow RS = 10 \rightarrow RS = 00$

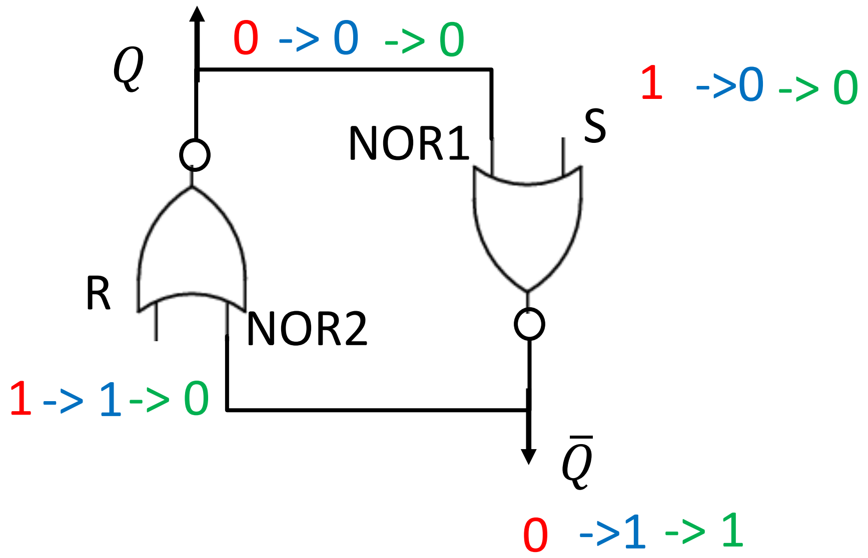




S-R Latch: $RS = 11 \rightarrow RS = 00$? (cont'd)

- $RS = 11 \rightarrow RS = 10 \rightarrow RS = 00$



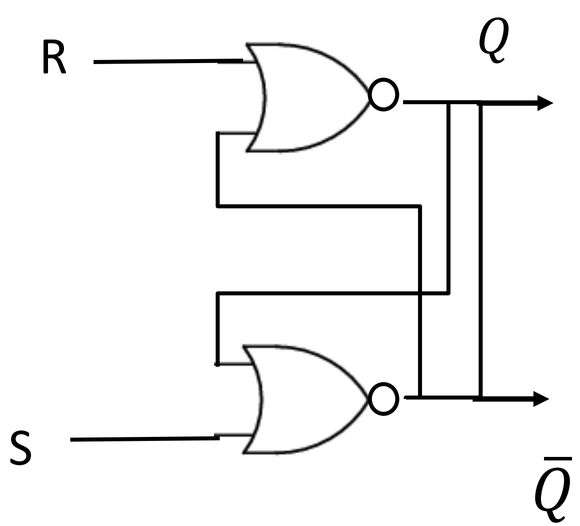


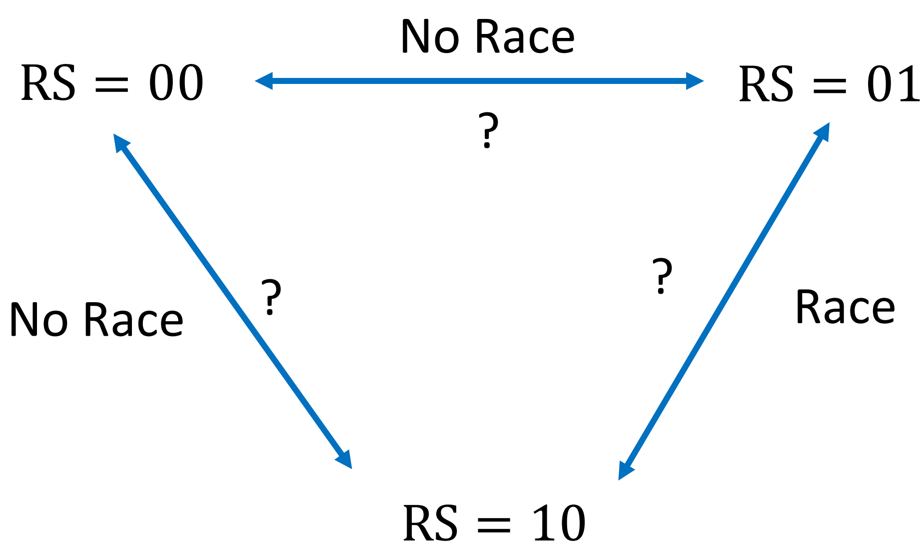
S-R Latch: Critical Race

- $RS = 11 \rightarrow RS = 00$
- $RS = 11 \rightarrow RS = 01 \rightarrow RS = 00$
  - $R\ win\ the\ race$
  - =>$Q\bar{Q} = 00 \rightarrow Q\bar{Q} = 10 \rightarrow Q\bar{Q} = 10$
- $RS = 11 \rightarrow RS = 10 \rightarrow RS = 00$
  - S $win\ the\ race$
  - =>$Q\bar{Q} = 00 \rightarrow Q\bar{Q} = 01 \rightarrow Q\bar{Q} = 01$
- Critical race
  - Affect the functionality
  - Winner set the output

S-R Latch: Race

- Is there any race?
- $R\ win\ the\ race$
  - $RS = 01 \rightarrow RS = 11 \rightarrow RS = 10$
- S $win\ the\ race$
  - $RS = 01 \rightarrow RS = 00 \rightarrow RS = 10$





S-R Latch: Non-Critical Race

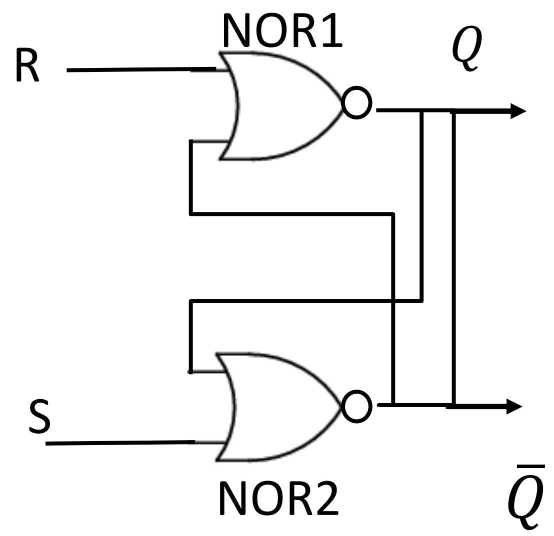
- $R\ win\ the\ race$
  - $RS = 01 \rightarrow RS = 11 \rightarrow RS = 10$
  - $=> Q\bar{Q} = 10 \rightarrow Q\bar{Q} = 00 \rightarrow Q\bar{Q} = 01$

- S $win\ the\ race$
  - $RS = 01 \rightarrow RS = 00 \rightarrow RS = 10$
  - $=> Q\bar{Q} = 10 \rightarrow Q\bar{Q} = 10 \rightarrow Q\bar{Q} = 01$

- Non-critical race
  - In **both cases**, **final output** is the same
  - Winner set the output

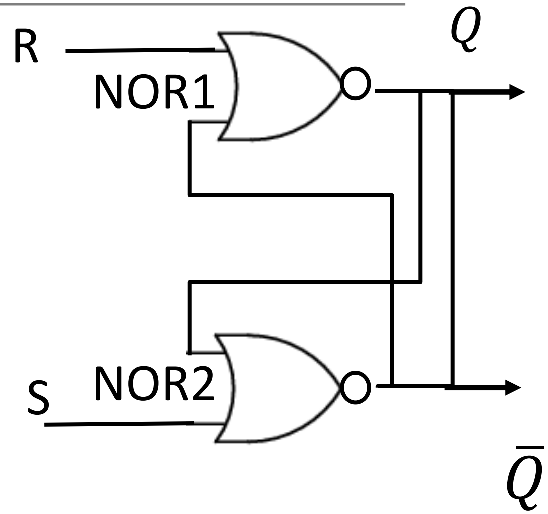
S-R Latch: NAND

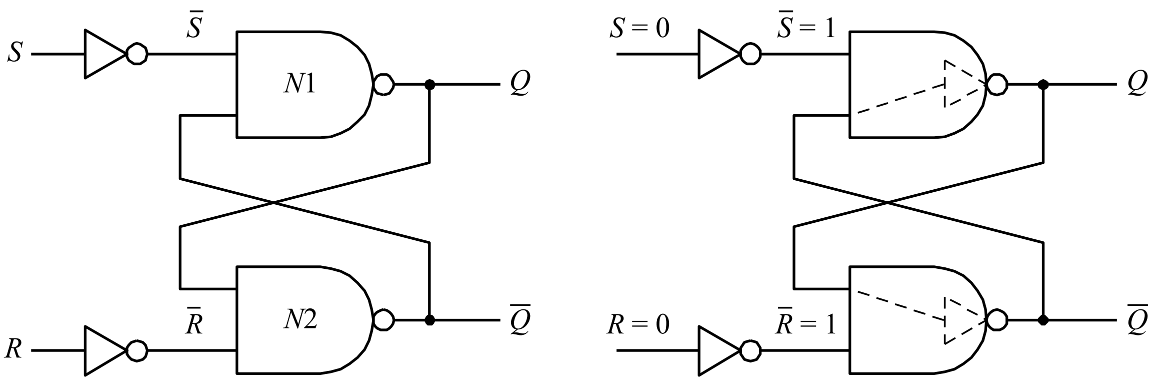
- Implement S-R lacth with NAND gates?



NAND S-R Latch

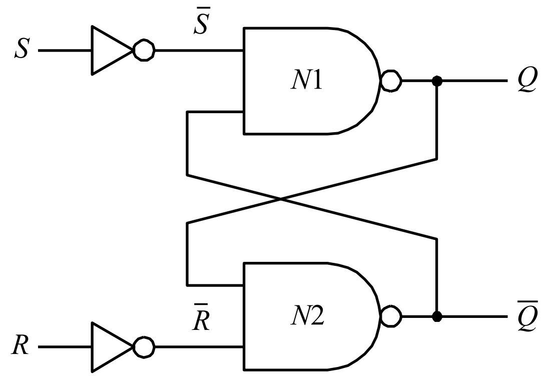
- NOR 1
  - $\overline{R+\bar{Q}} = \bar{R}\,.\,Q$
- NOR 2
  - $\overline{S+Q} = \bar{S}.\bar{Q}$



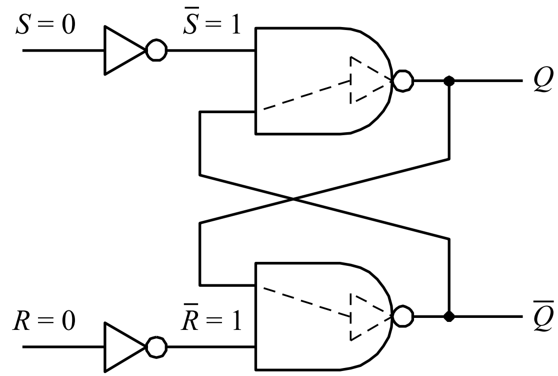


(a) (b)

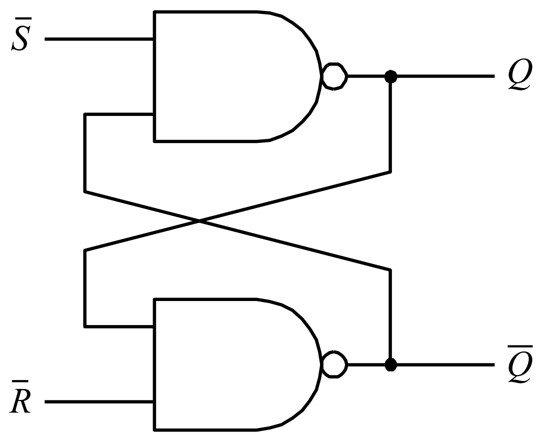
NAND S-R Latch (cont’d)



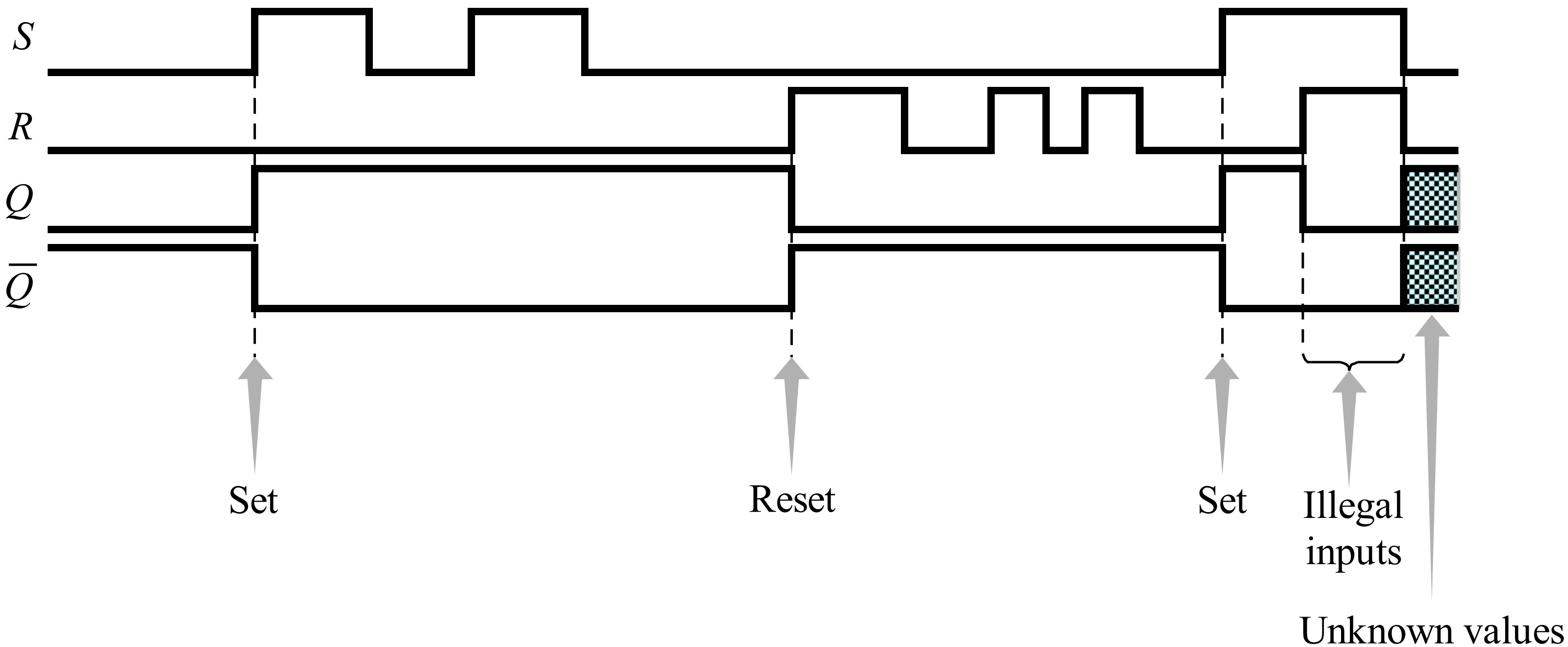
(a)



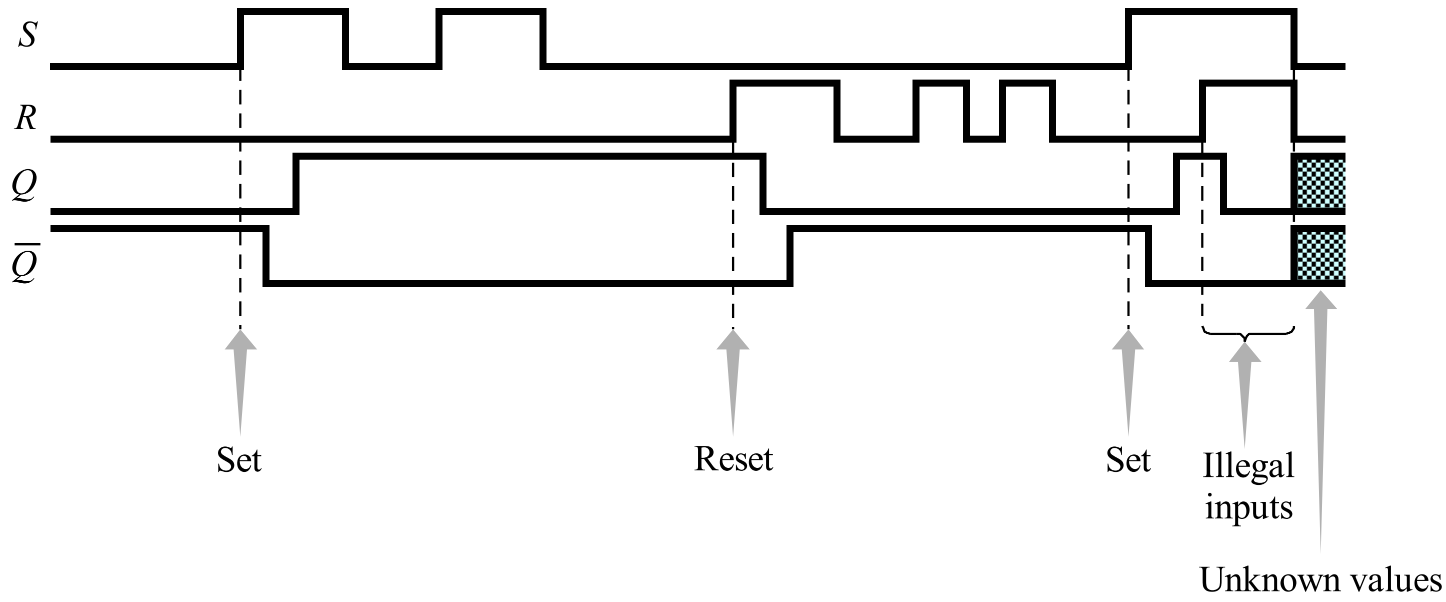
(b)



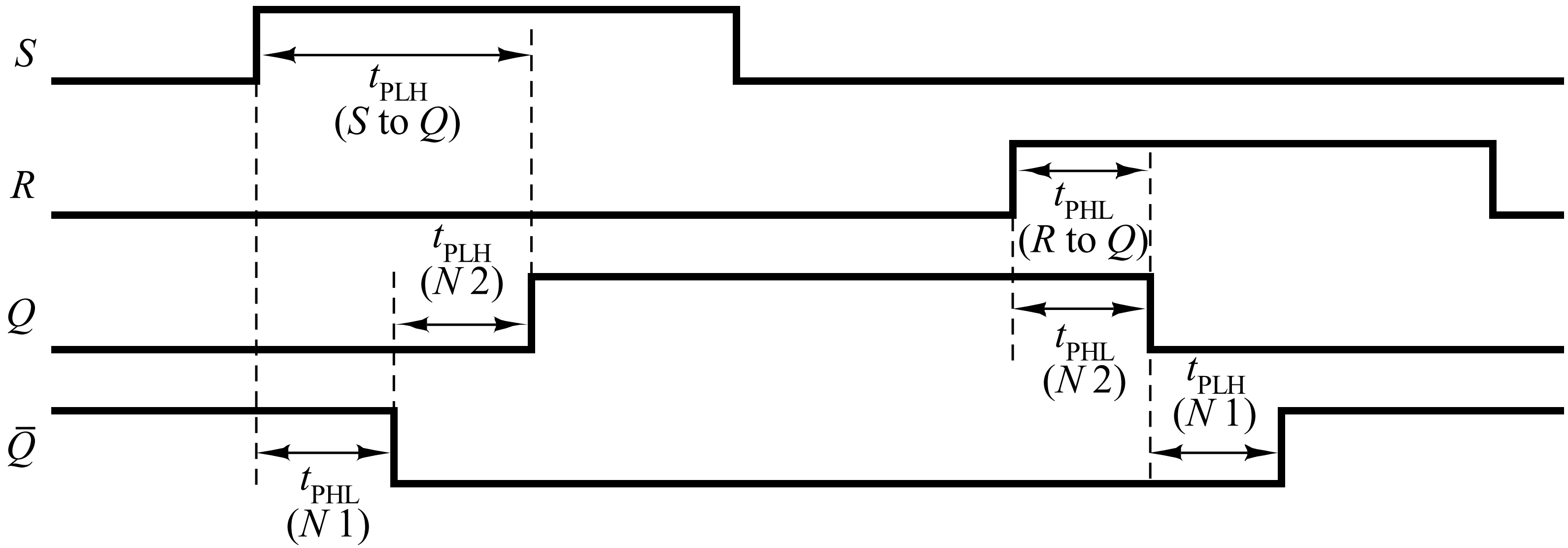
S-R Latch: Timing Diagram 1



S-R Latch: Timing Diagram 2

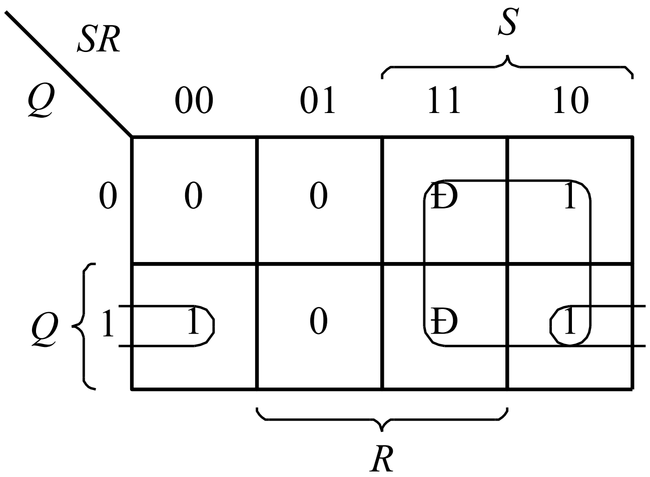


S-R Latch: Propagation Delays



S-R Latch: Characteristics

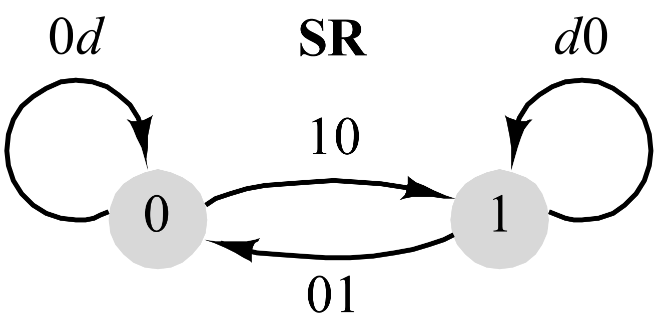
Excitation inputs S	R	Present state Q	Next state Q^*	
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	×	Not allowed
1	1	1	×	



$$Q^* = S + R'Q$$

S-R Latch: Characteristics (cont’d)

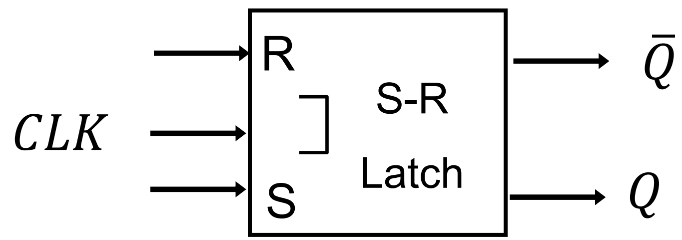
Excitation inputs *S*	Excitation inputs *R*	Present state *Q*	Next state *Q**	
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	×	Not allowed
1	1	1	×	



Clocked S-R Latch (CSR Latch)

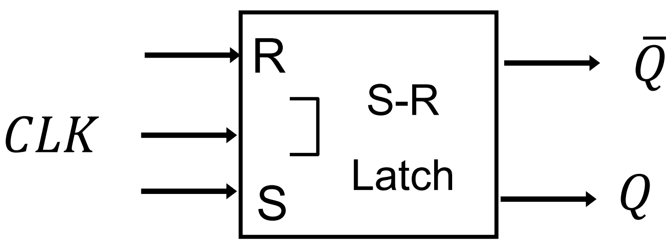
- Change all the Latched at the same time
  - Use a clock signal, CLK
  - A latch is active when CLK is 1 or 0
- Clocked S-R
- Flip Flop

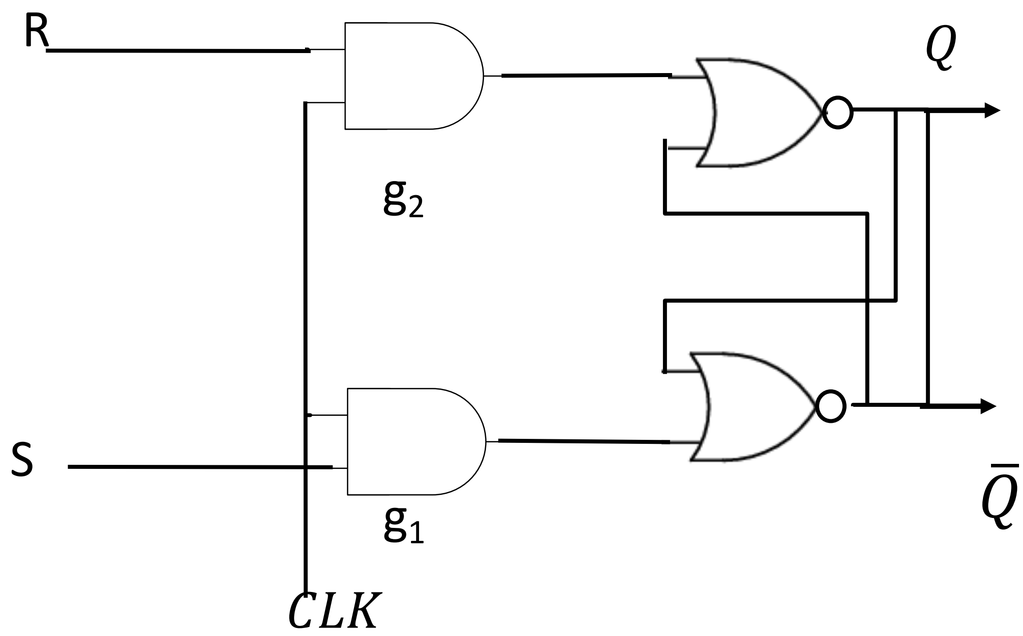
CLK	R	S	Q	$\bar{Q}$	
0	X	X	Q	$\bar{Q}$	Hold
1	0	0	Q	$\bar{Q}$	Hold
1	0	1	1	0	Set
1	1	0	0	1	Reset
1	1	1	Invalid		Invalid



Clocked S-R Latch (cont’d)

CLK	R	S	Q	$\bar{Q}$	
0	X	X	Q	$\bar{Q}$	Hold
1	0	0	Q	$\bar{Q}$	Hold
1	0	1	1	0	Set
1	1	0	0	1	Reset
1	1	1	Invalid		Invalid

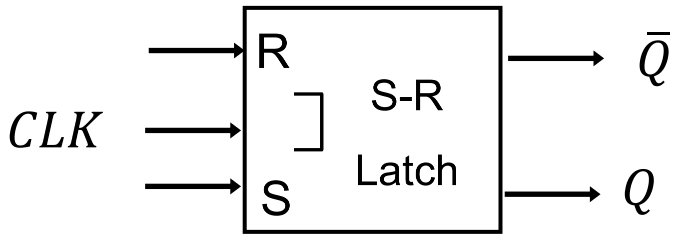


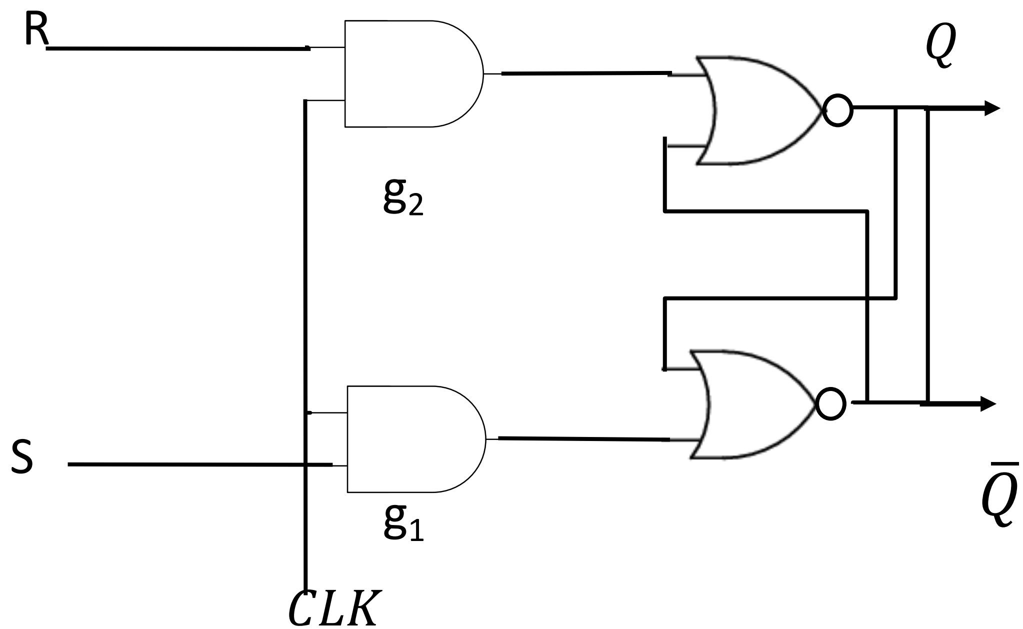


Clocked S-R Latch (cont'd)

CLK	R	S	Q	$\bar{Q}$	
0	X	X	Q	$\bar{Q}$	Hold
1	0	0	Q	$\bar{Q}$	Hold
1	0	1	1	0	Set
1	1	0	0	1	Reset
1	1	1	Invalid		Invalid

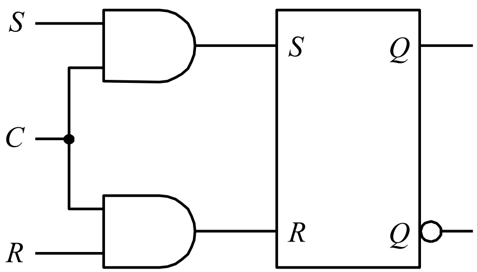
Clocked S-R Flip Flop



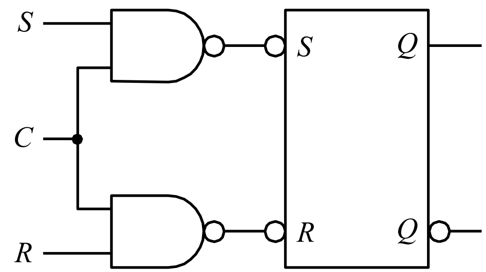


دانشگاه علم و صنعت ایران
Iran University of Science & Technology
IUST

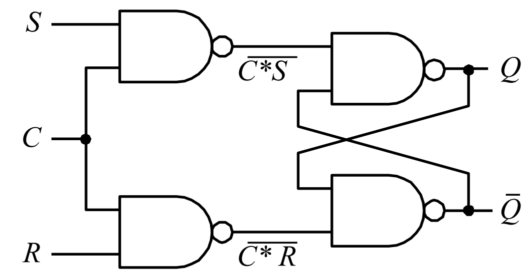
S-R Flip Flop (S-R FF)



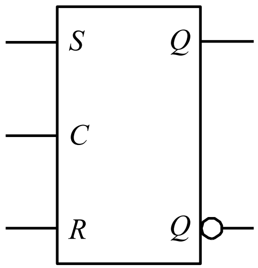
(a)



(b)



(c)



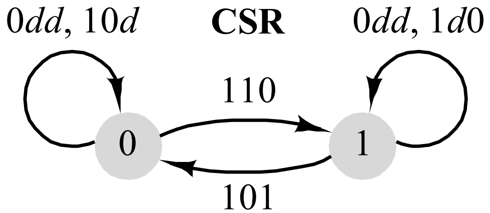
(d)

S-R Flip Flop (S-R FF): characteristics

- C is CLK

Enable inputs C	Excitation inputs S	R	Present state Q	Next state Q^*	
0	×	×	0	0	Hold
0	′	′	1	1	
1	0	0	0	0	No change
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	×	Not allowed
1	1	1	1	×	

(a)



(b)

$$Q^* = S.\,Clk + R'.\,Q + Clk'.\,Q$$

Thank You