

Computer Architecture
Lecture Week 6 (Part II)

Latch Circuit

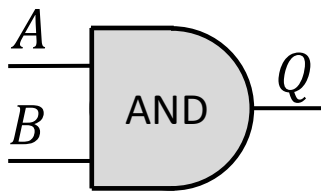
Artem Burmyakov

October 10, 2025

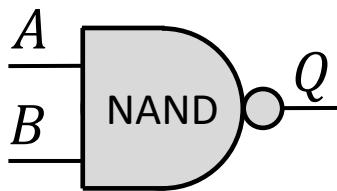


INNOPOLIS
UNIVERSITY

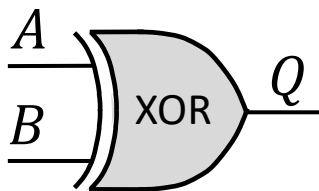
Recap: Basic Logic Gates



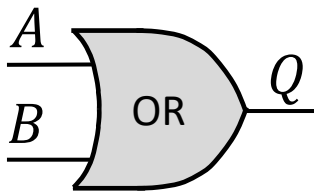
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



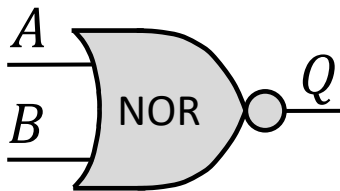
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



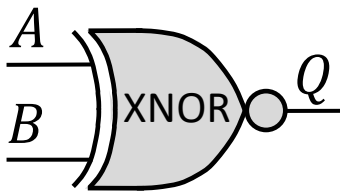
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

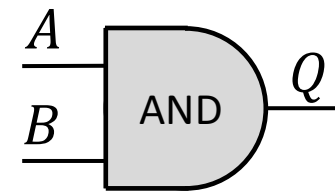


A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

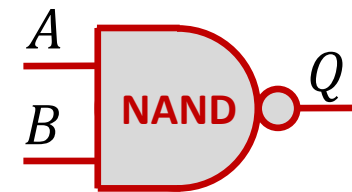


A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

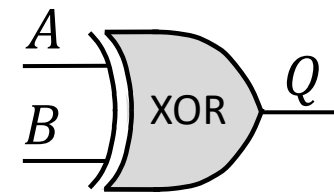
Recap: Universal Logic Gates



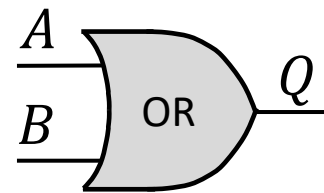
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



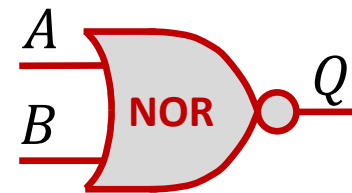
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



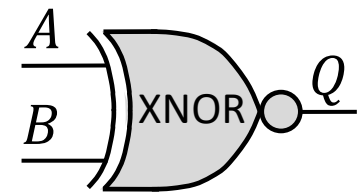
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

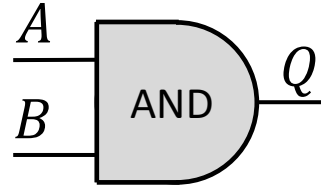


A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

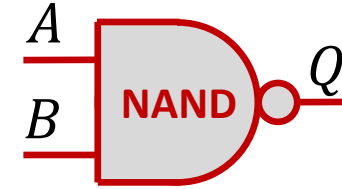
Recap: Universal Logic Gates

Used to construct
memory circuits:

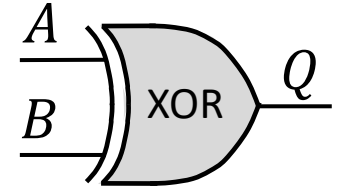
latches and flip-flops



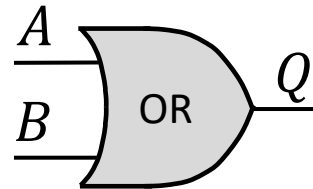
A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



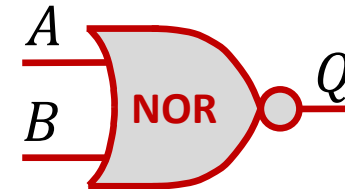
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0



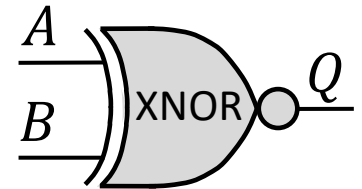
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

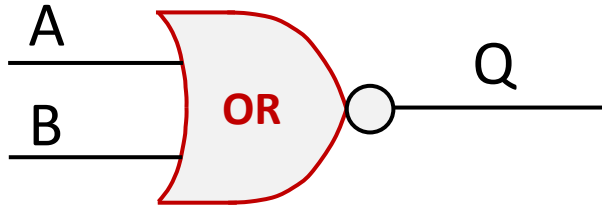


A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0



A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

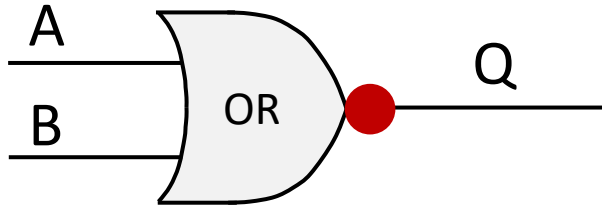
Recap: NOR Gate



NOR = OR + NOT

A	B	OR
1	1	1
1	0	1
0	1	1
0	0	0

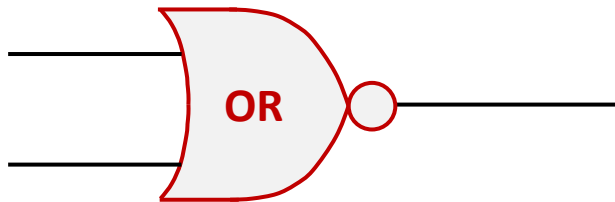
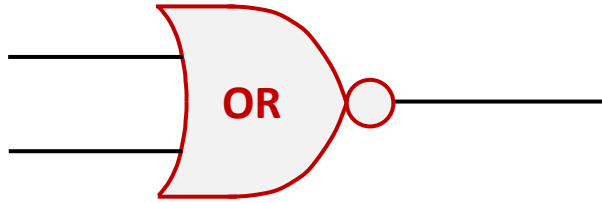
Recap: NOR Gate



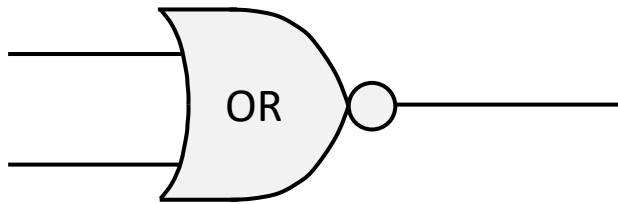
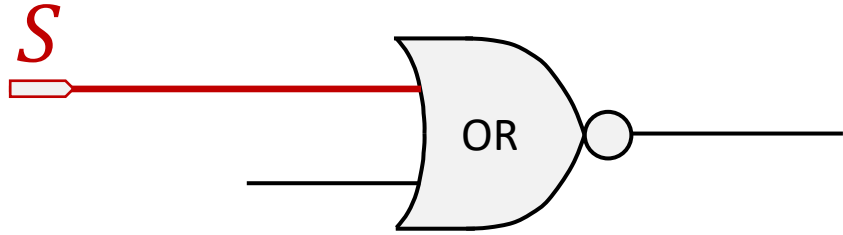
NOR = OR + NOT

A	B	OR	NOR
1	1	1	0
1	0	1	0
0	1	1	0
0	0	0	1

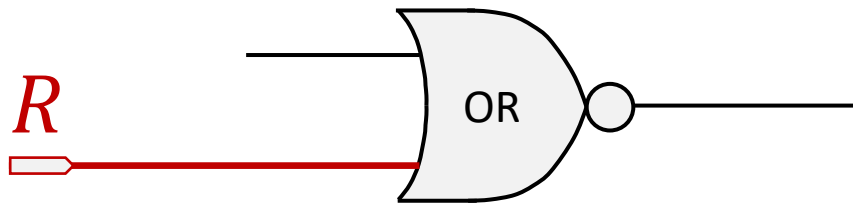
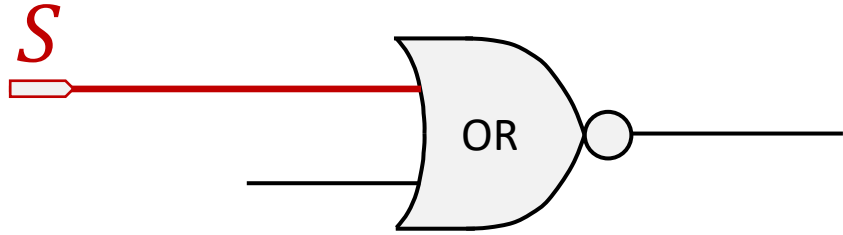
Circuit Construction: NOR Gates



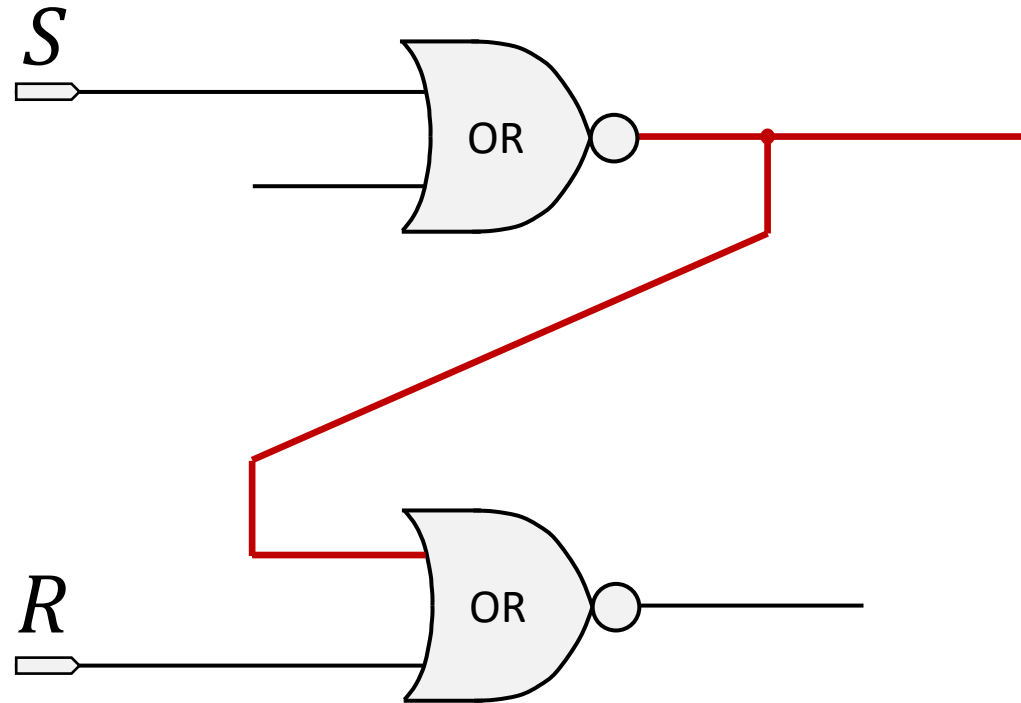
Circuit Construction: “S” Input Pin



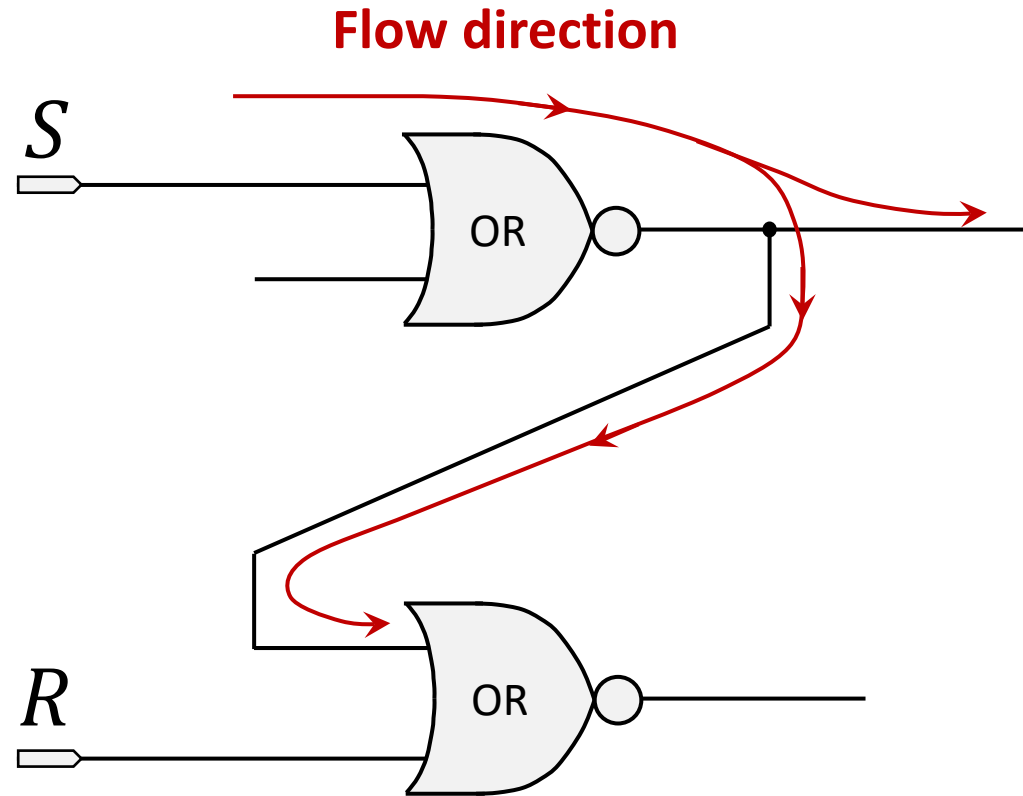
Circuit Construction: “S” and “R” Input Pins



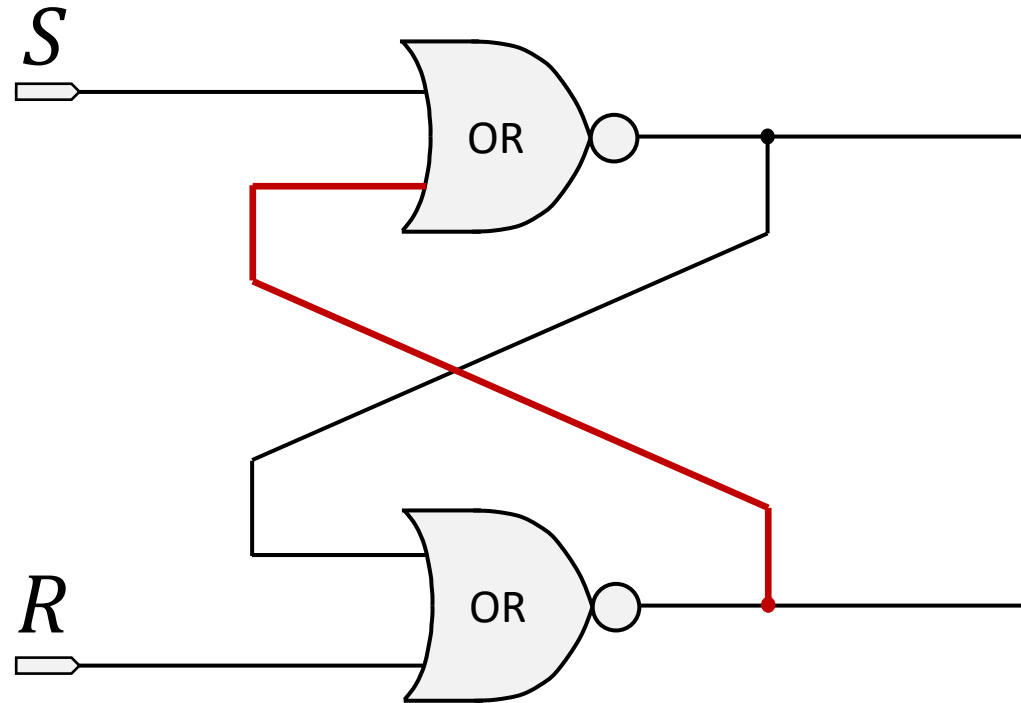
Circuit Construction: Interconnection of NOR Gates



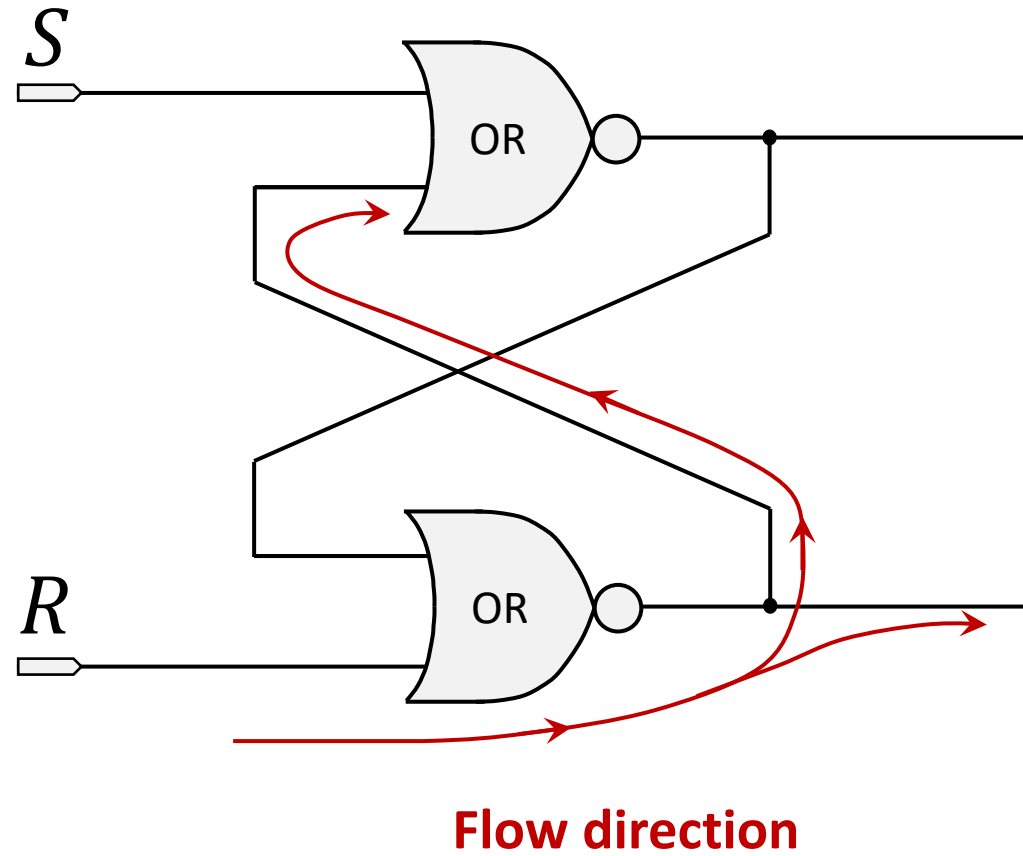
Circuit Construction: Interconnection of NOR Gates



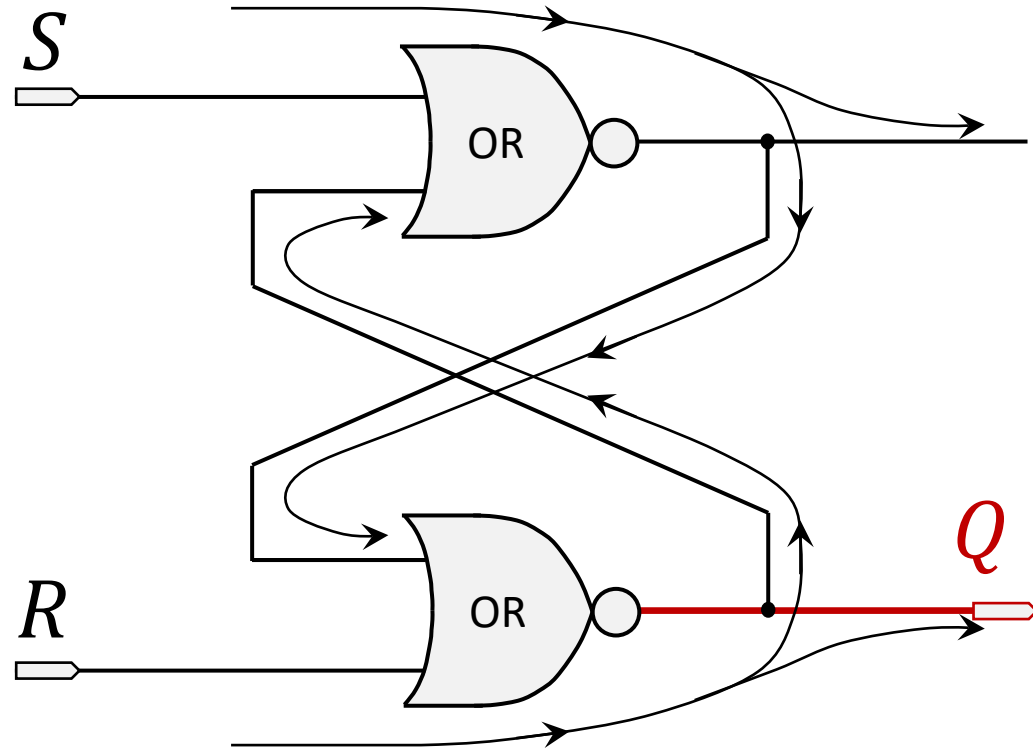
Circuit Construction: Cross-Coupled Interconnection of NOR Gates



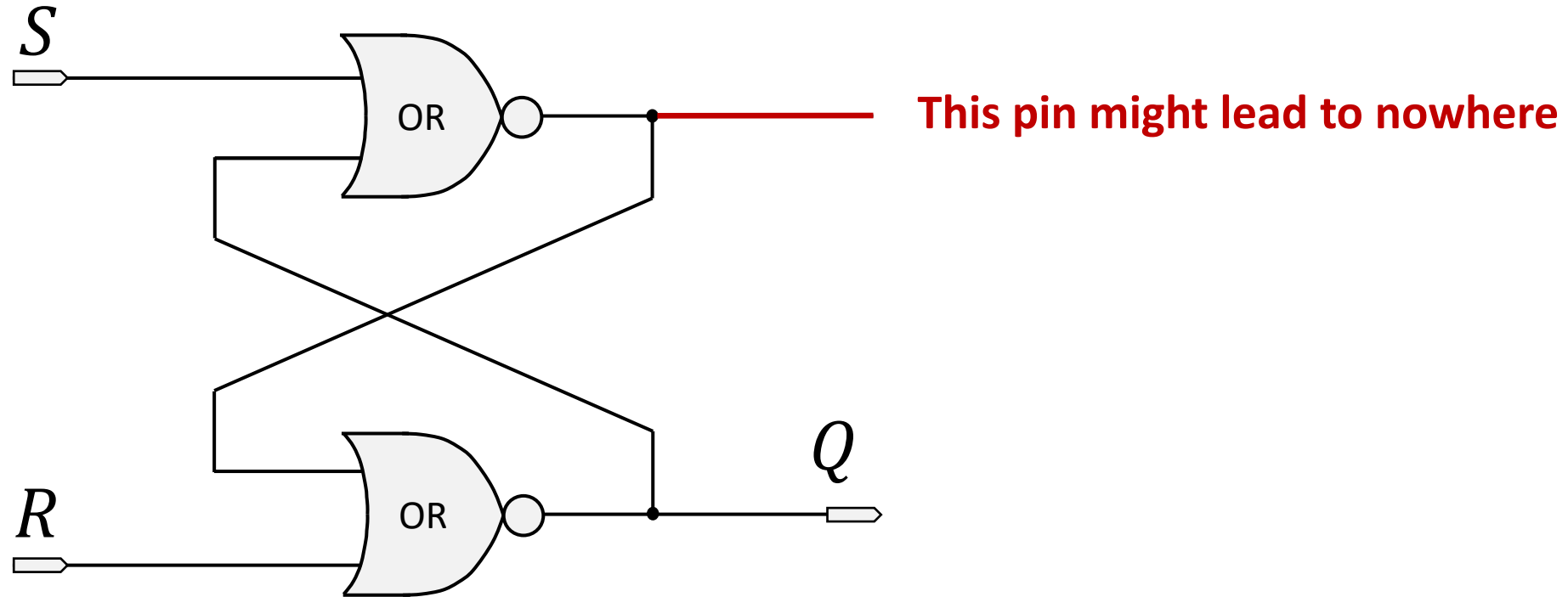
Circuit Construction: Cross-Coupled Interconnection of NOR Gates



Circuit Construction: “Q” Output Pin

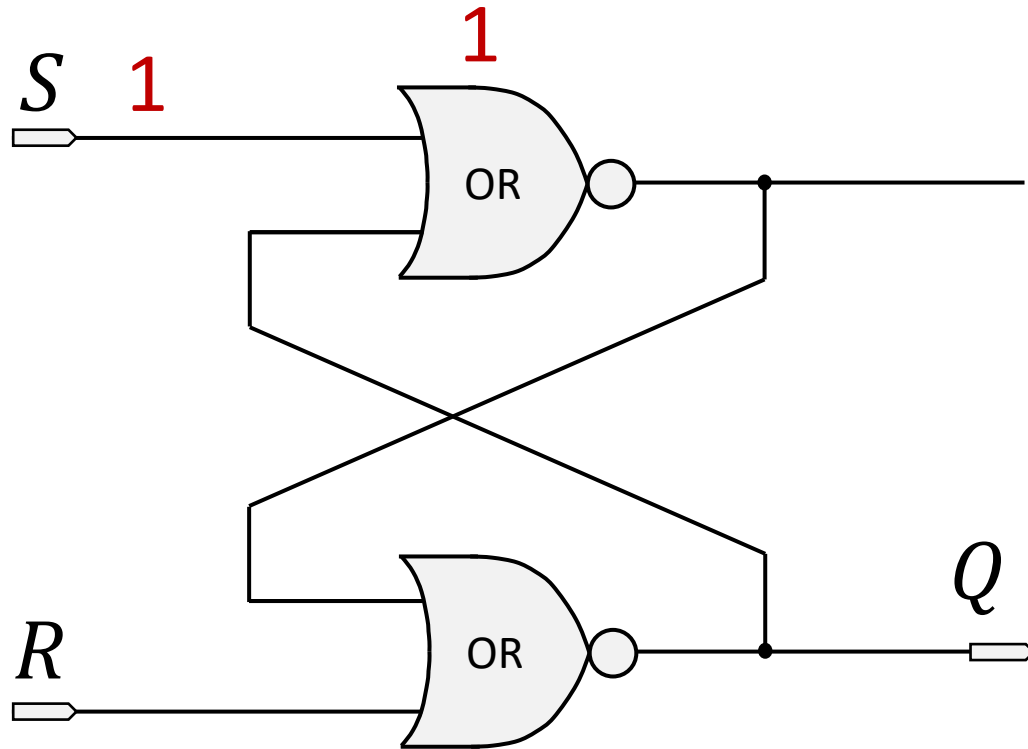


Circuit Construction: “Q” and Idle Output Pins



Circuit Behavior Analysis

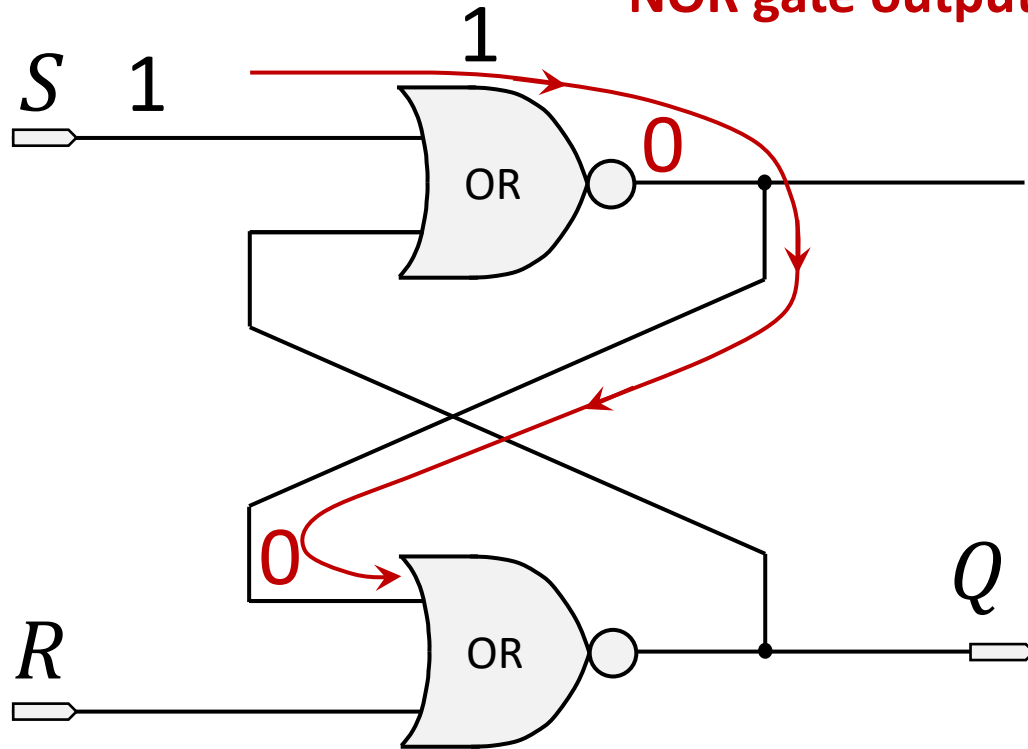
OR gate outputs "1" if at least one input is "1"



Circuit Behavior Analysis

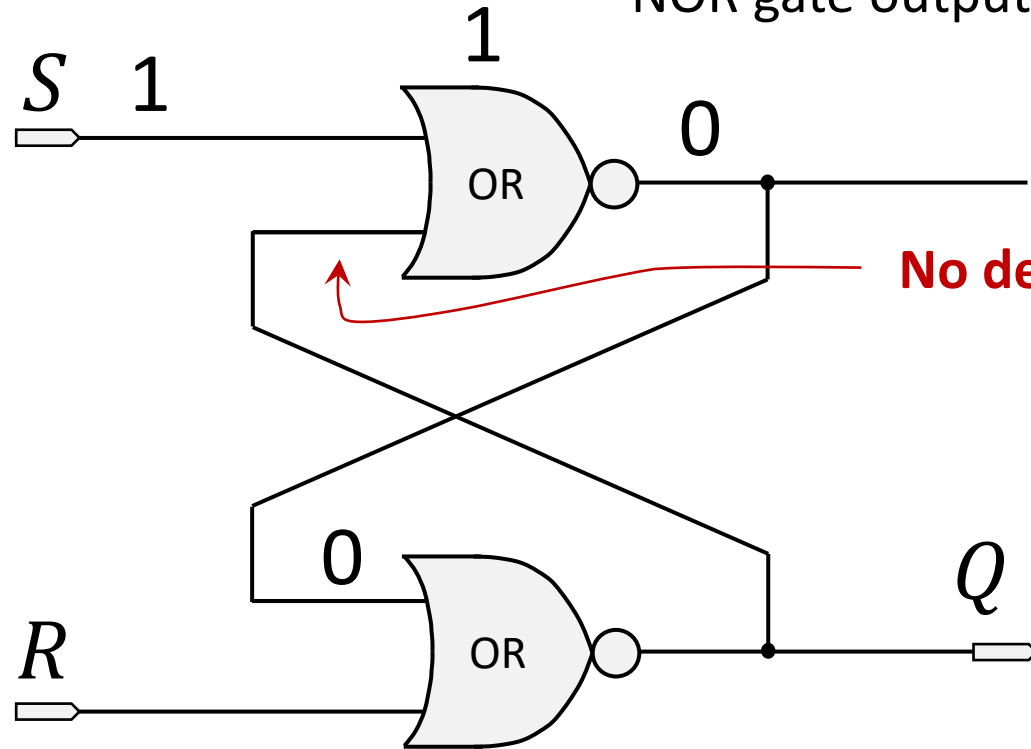
OR gate outputs "1" if at least one input is "1"

NOR gate outputs "0" if at least one input is "1"



Circuit Behavior Analysis

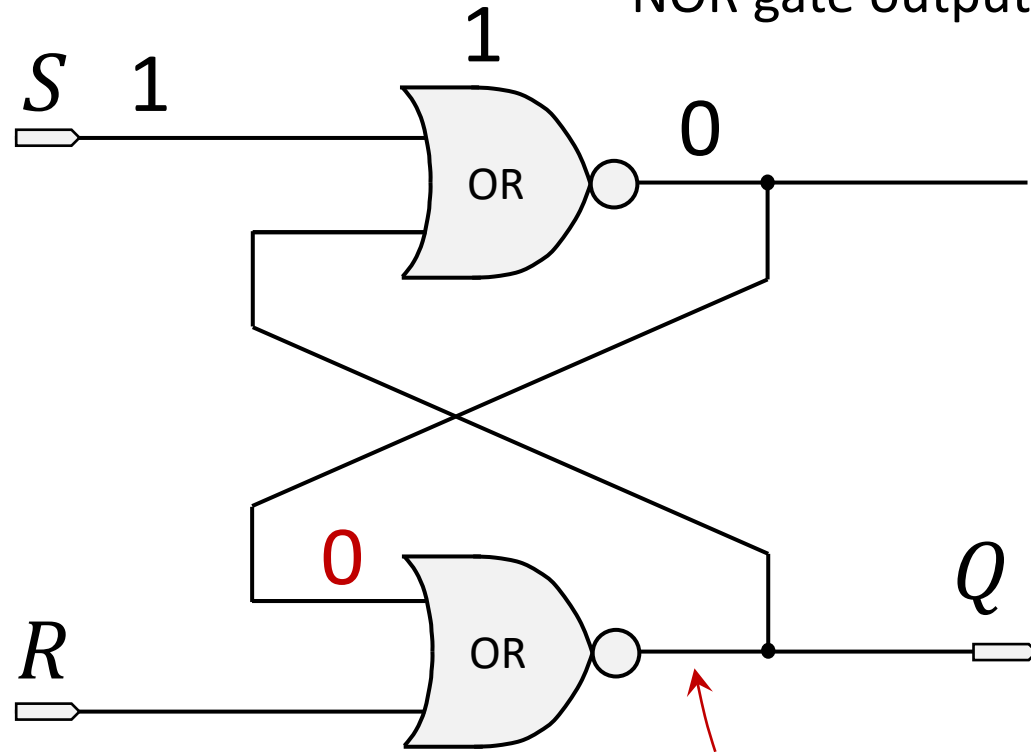
OR gate outputs "1" if at least one input is "1"
NOR gate outputs "0" if at least one input is "1"



No dependency on this input in case S=1

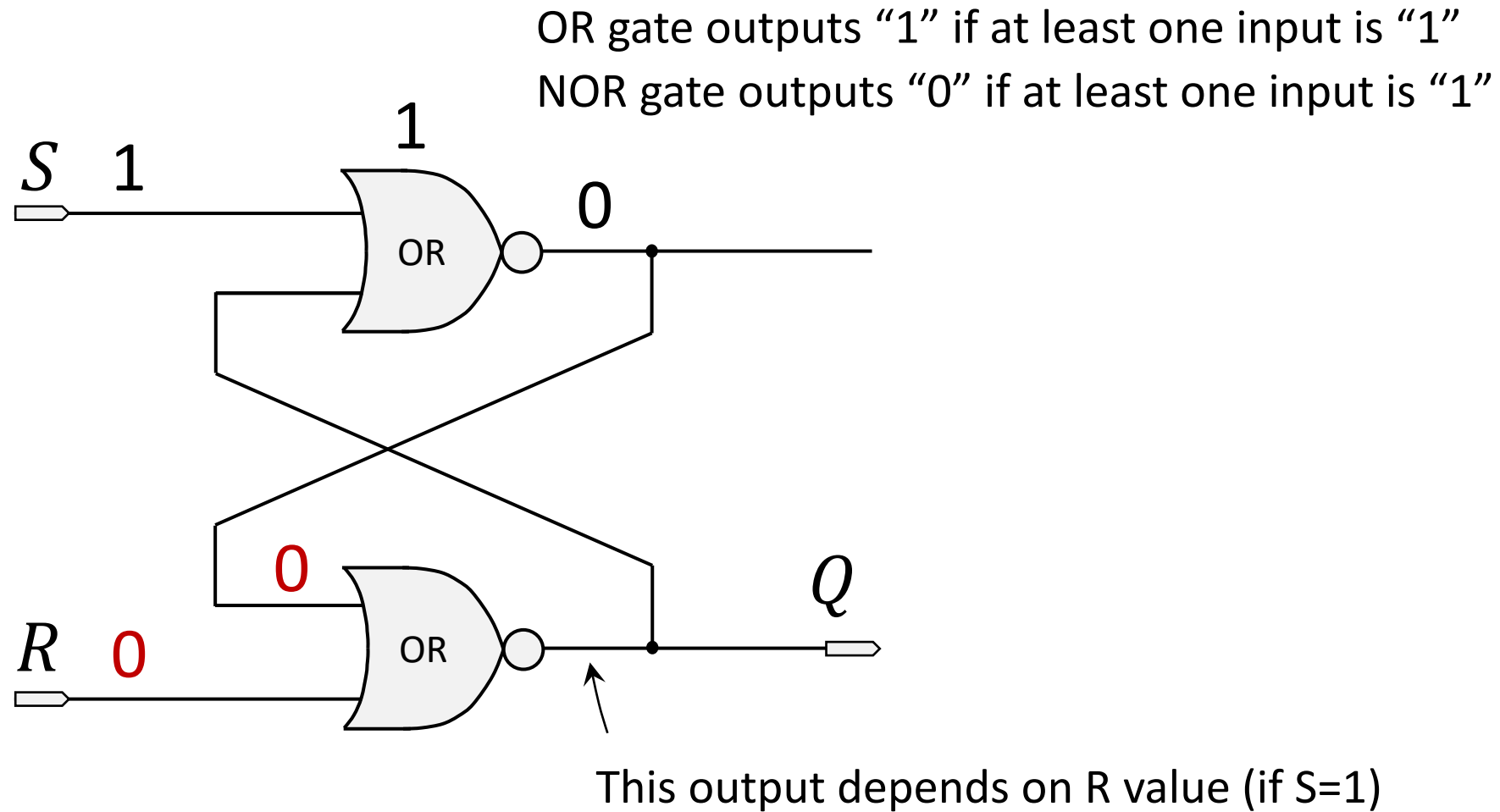
Circuit Behavior Analysis

OR gate outputs "1" if at least one input is "1"
NOR gate outputs "0" if at least one input is "1"



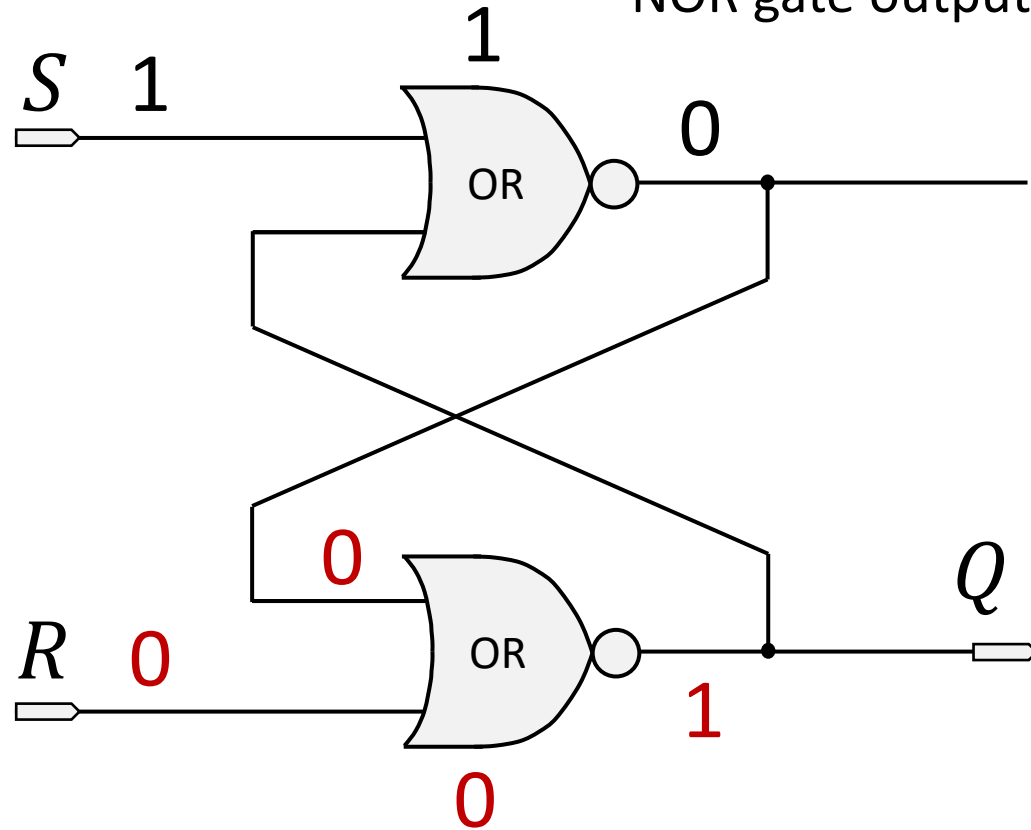
This output depends on R value in case S=1

Circuit Behavior Analysis



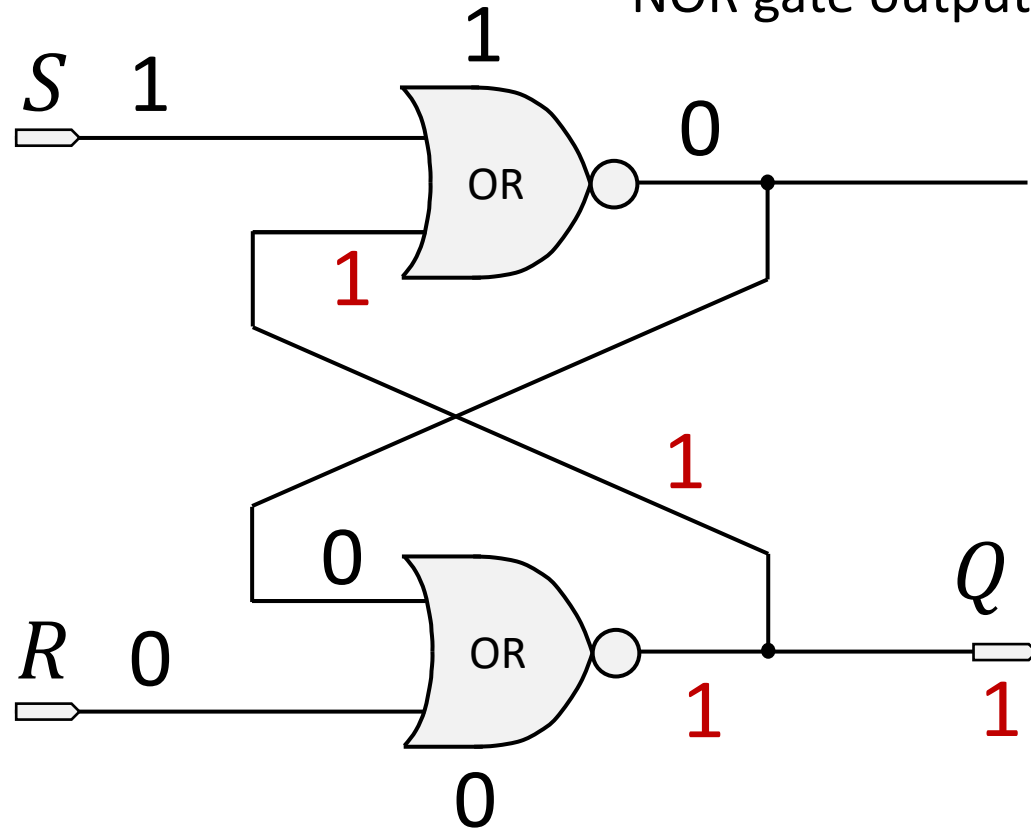
Circuit Behavior Analysis

OR gate outputs "1" if at least one input is "1"
NOR gate outputs "0" if at least one input is "1"

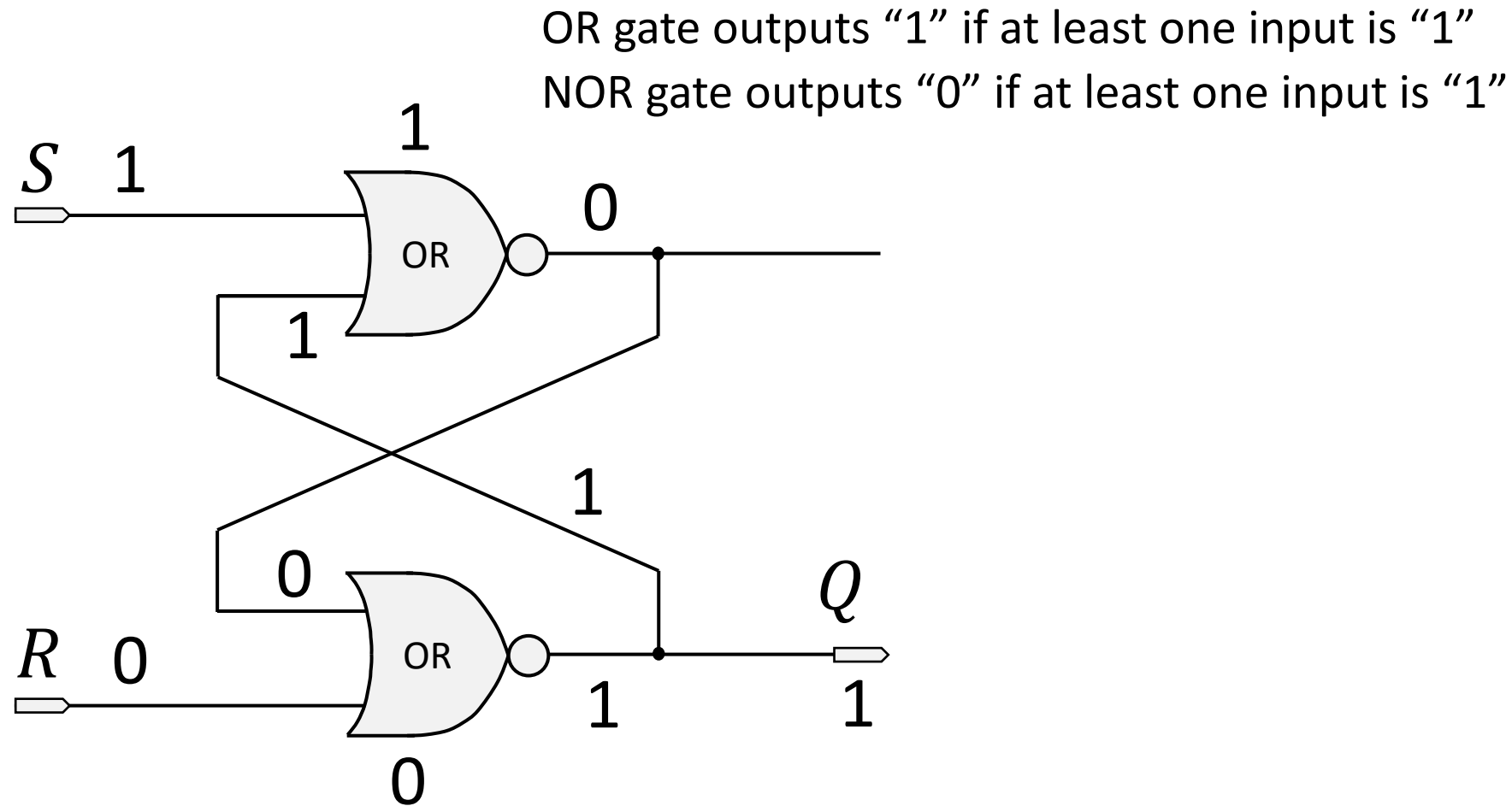


Circuit Behavior Analysis

OR gate outputs "1" if at least one input is "1"
NOR gate outputs "0" if at least one input is "1"

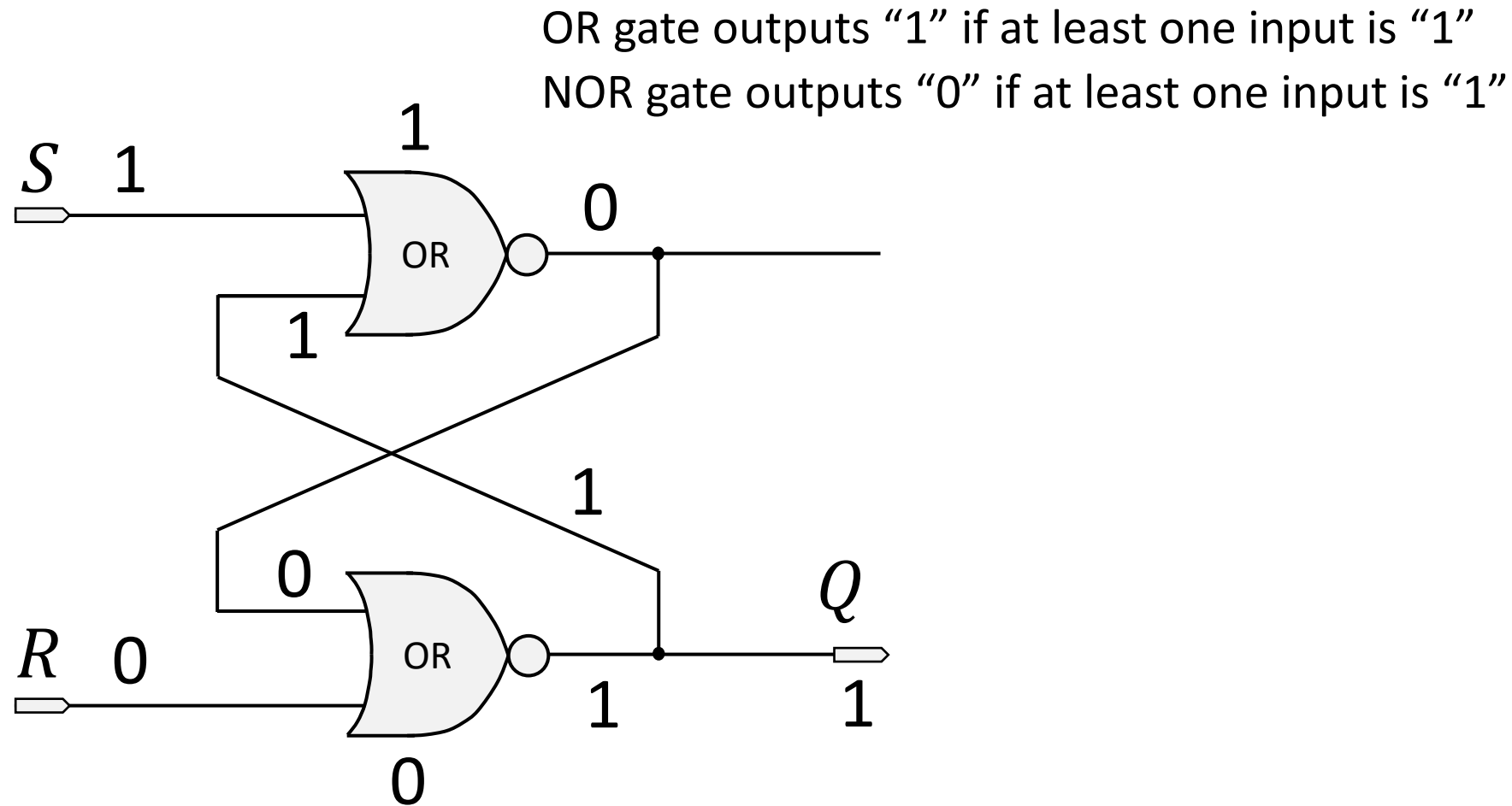


Circuit Behavior Analysis



Does Q depend on the order of S and R arrival (e.g. R arrives first, and S arrives next) ?

Circuit Behavior Analysis

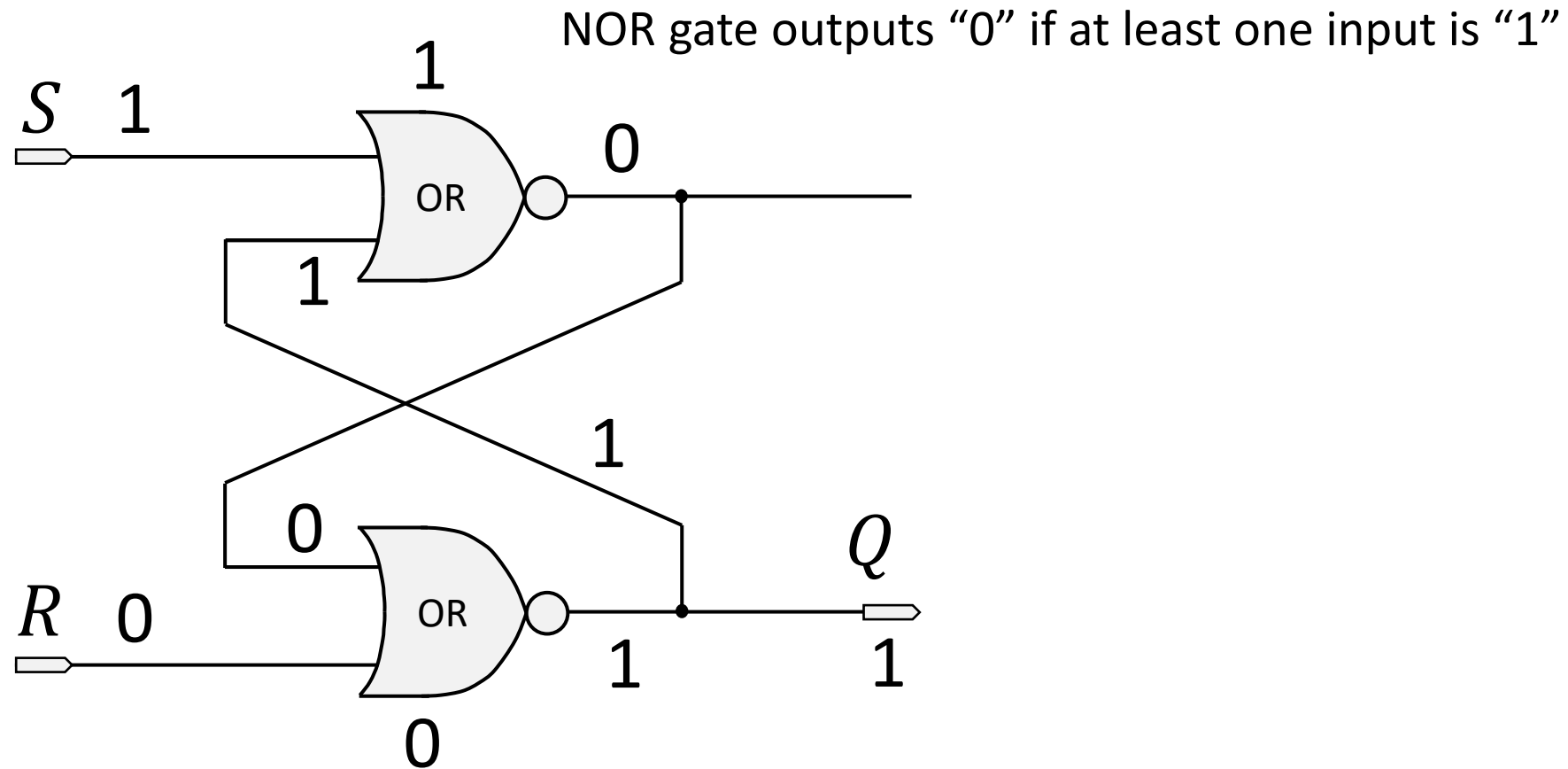


Does Q depend on the order of S and R arrival (e.g. R arrives first, and S arrives next) ?

No; you can check it by hand

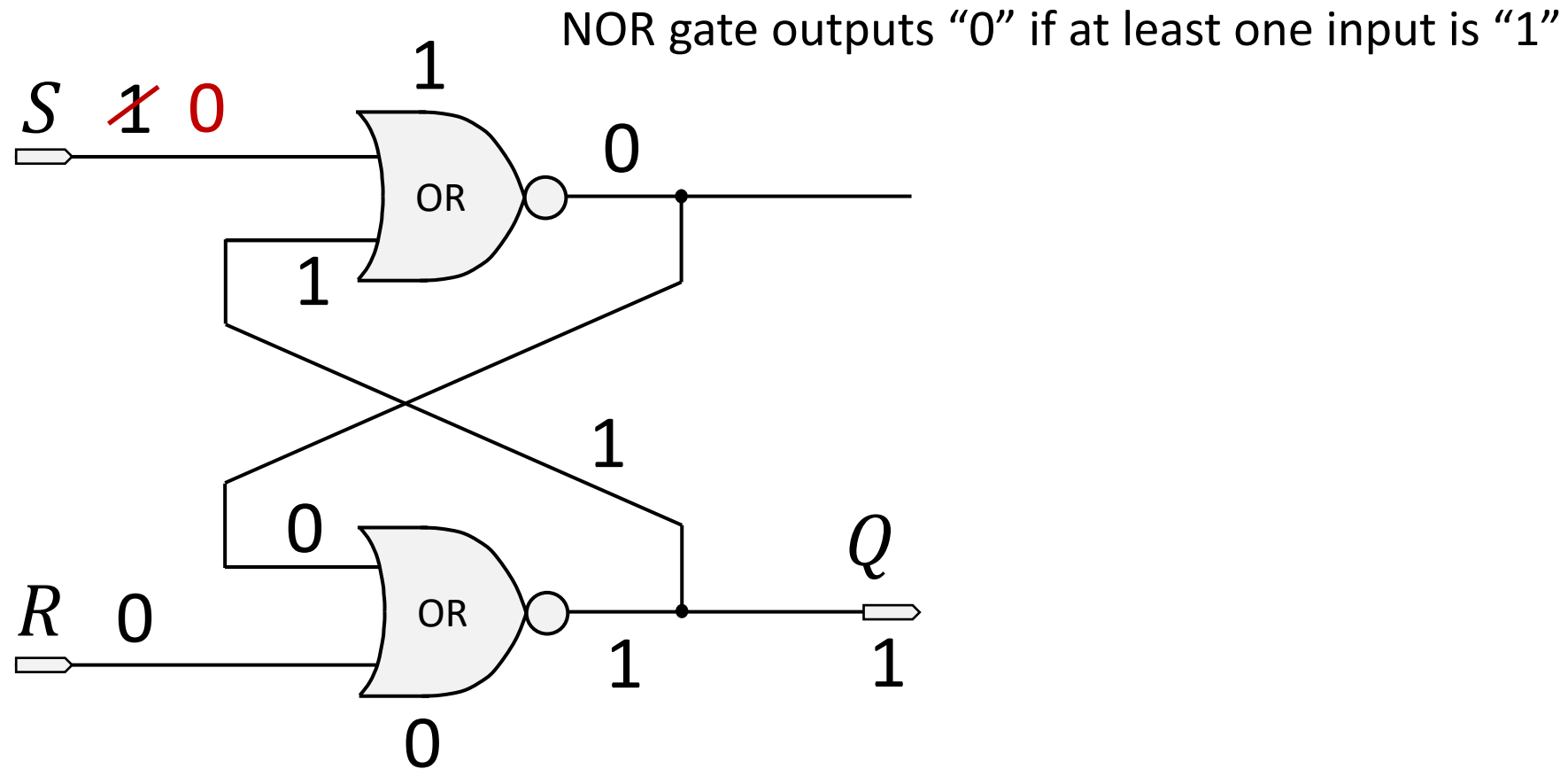
Circuit Behavior Analysis

Assume that S changes from “1” to “0”



Circuit Behavior Analysis

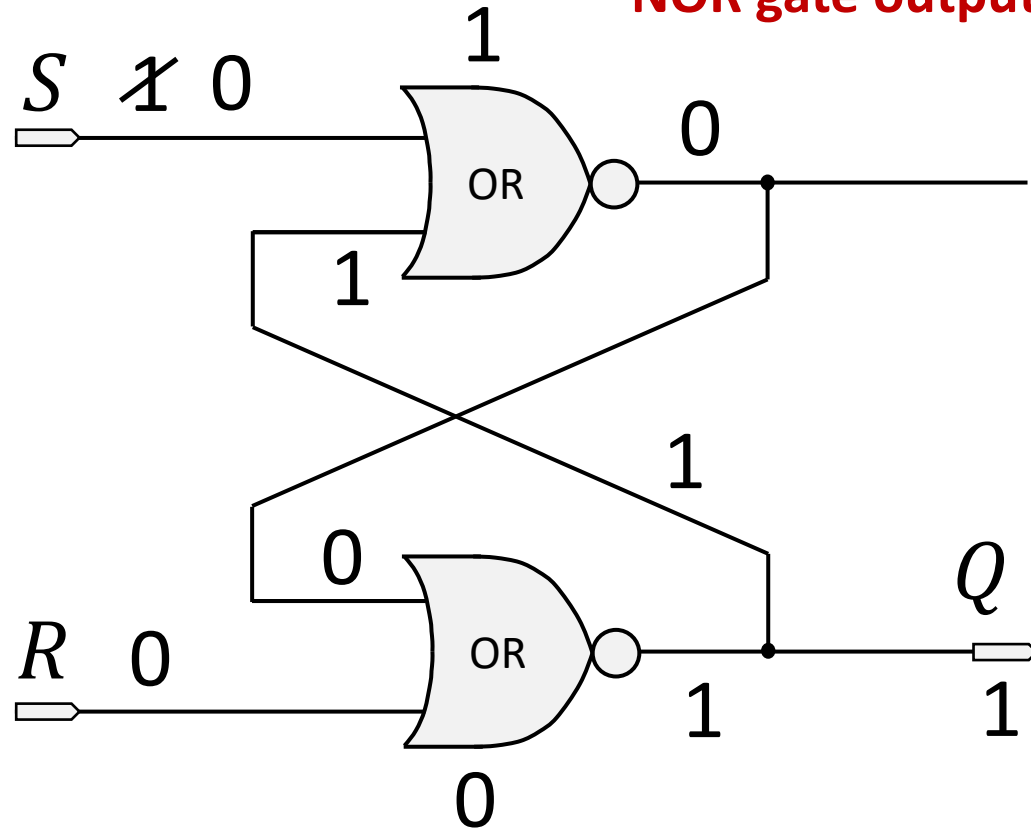
Assume that S changes from "1" to "0"



Circuit Behavior Analysis

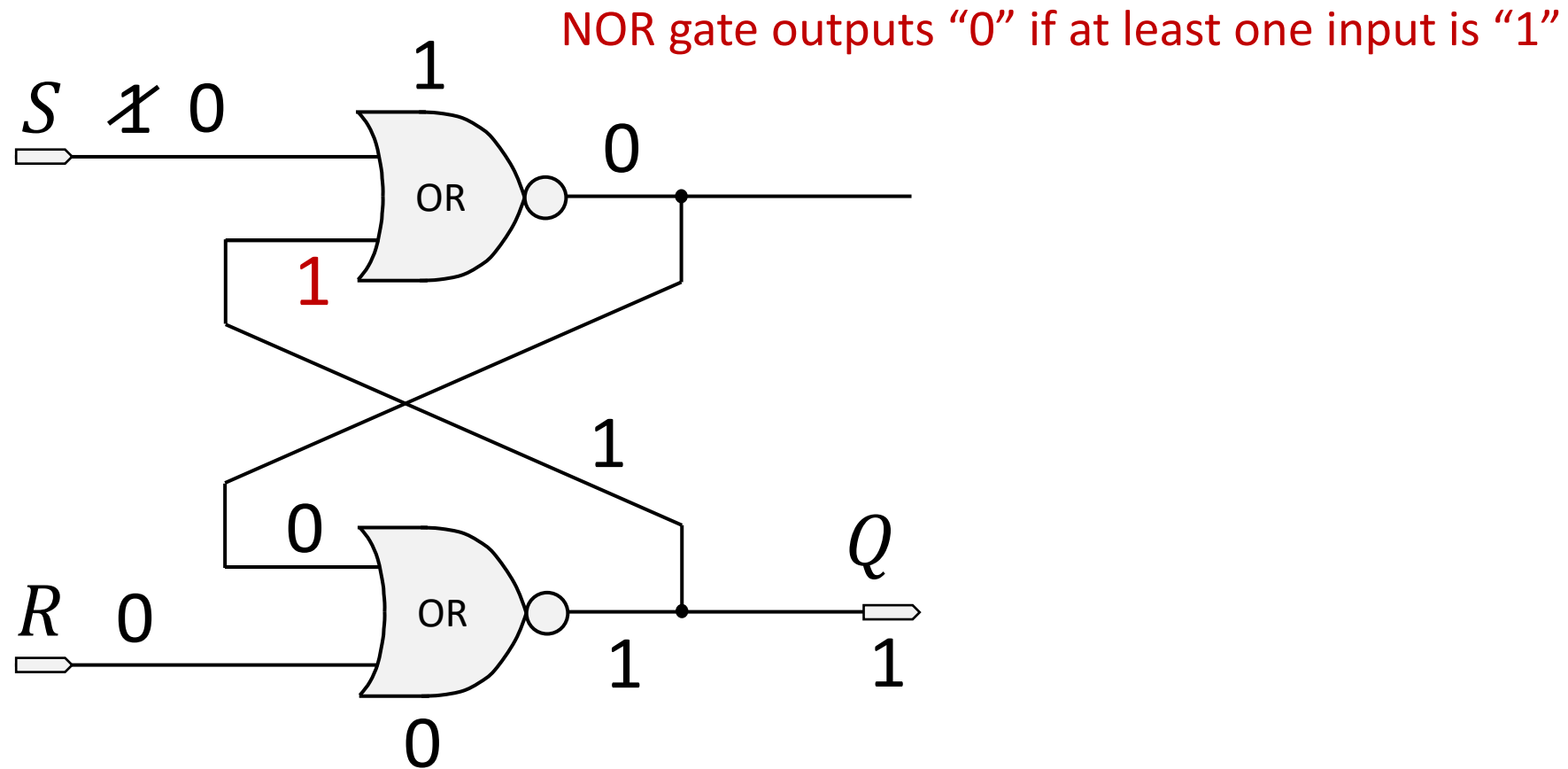
Assume that S changes from "1" to "0"

NOR gate outputs "0" if at least one input is "1"



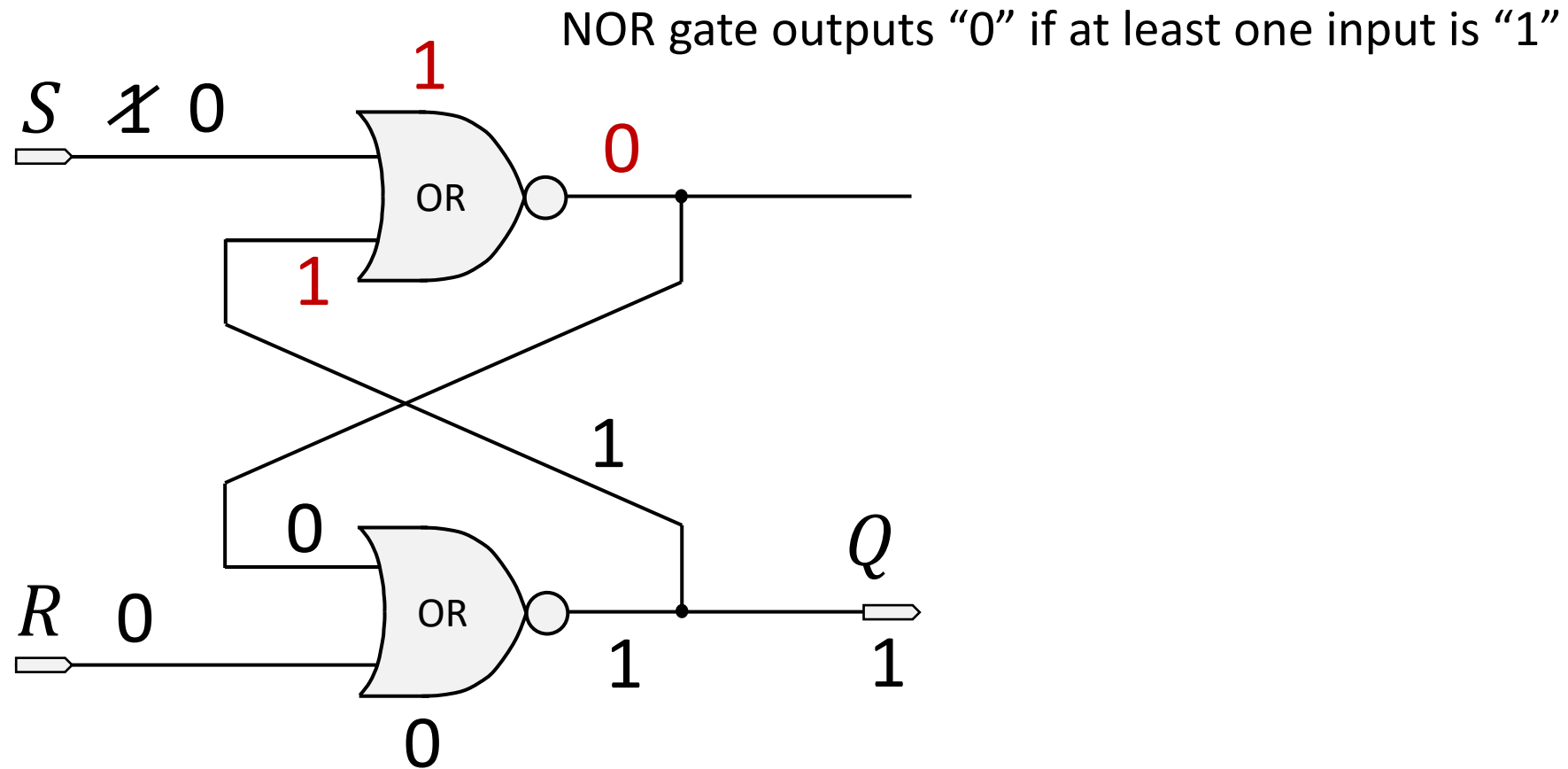
Circuit Behavior Analysis

Assume that S changes from "1" to "0"



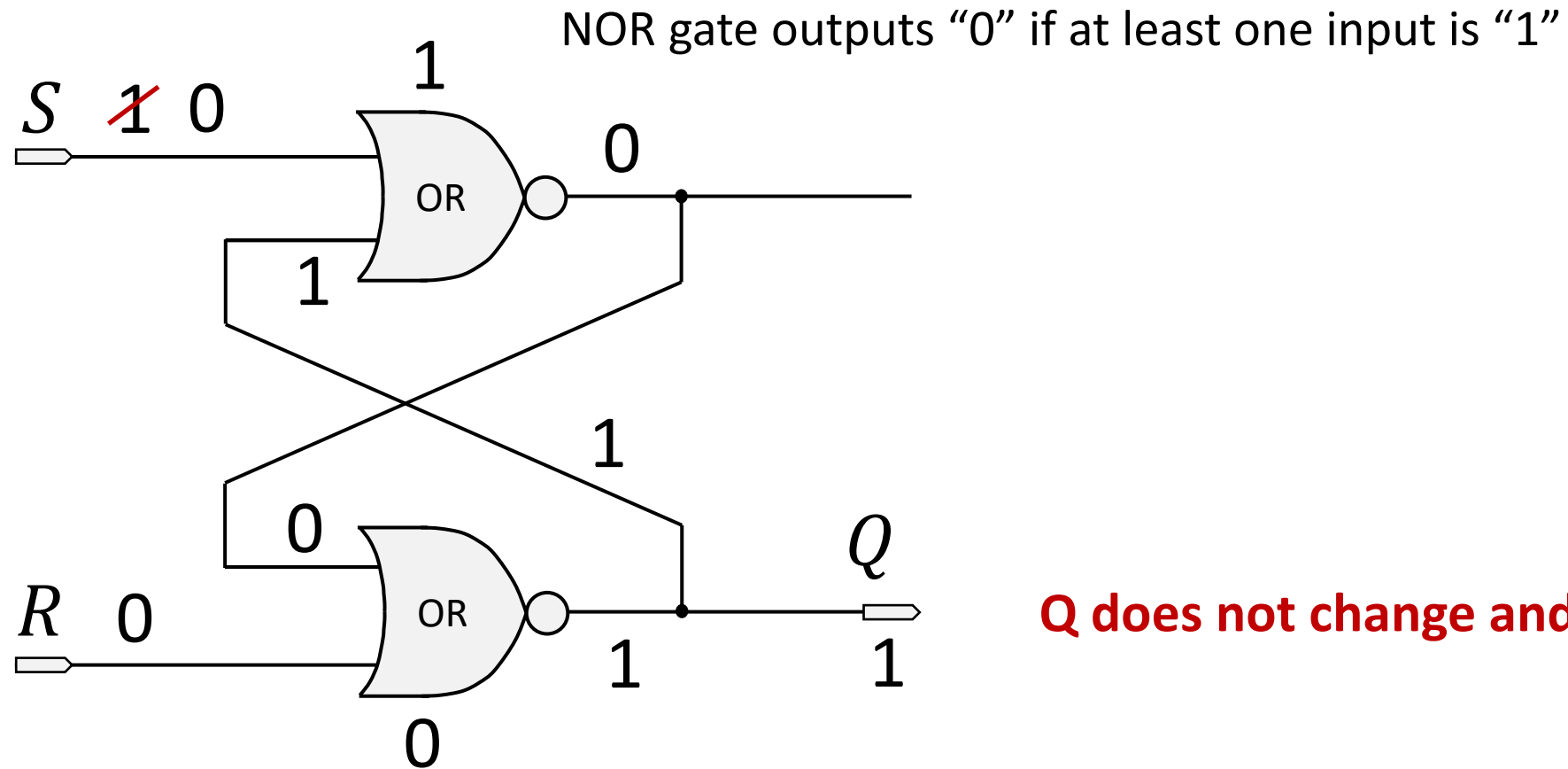
Circuit Behavior Analysis

Assume that S changes from "1" to "0"



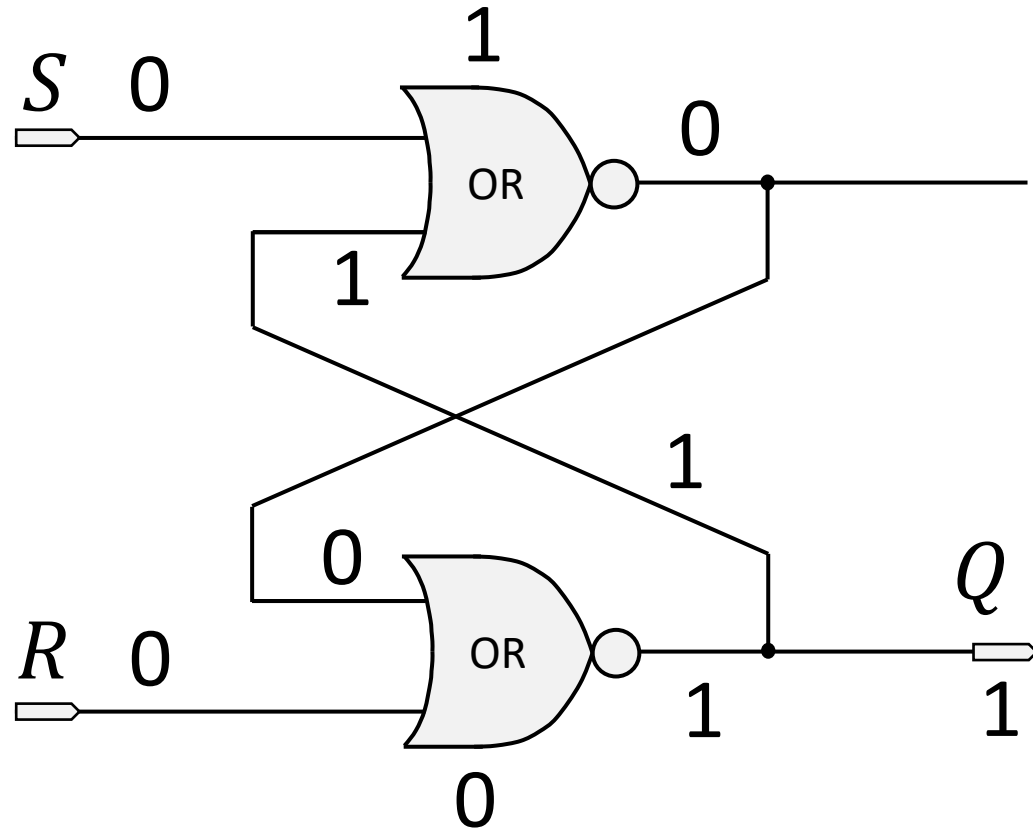
Circuit Behavior Analysis

Assume that S changes from “1” to “0”



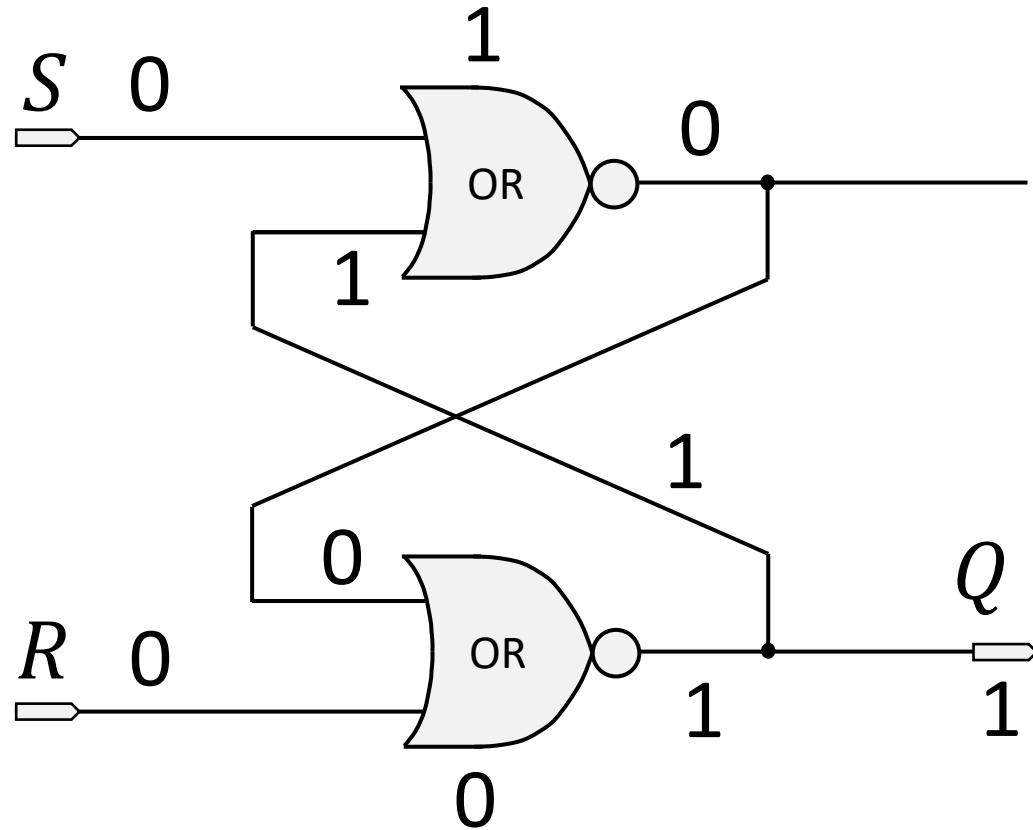
Q does not change and remains “1”

Circuit Behavior Analysis



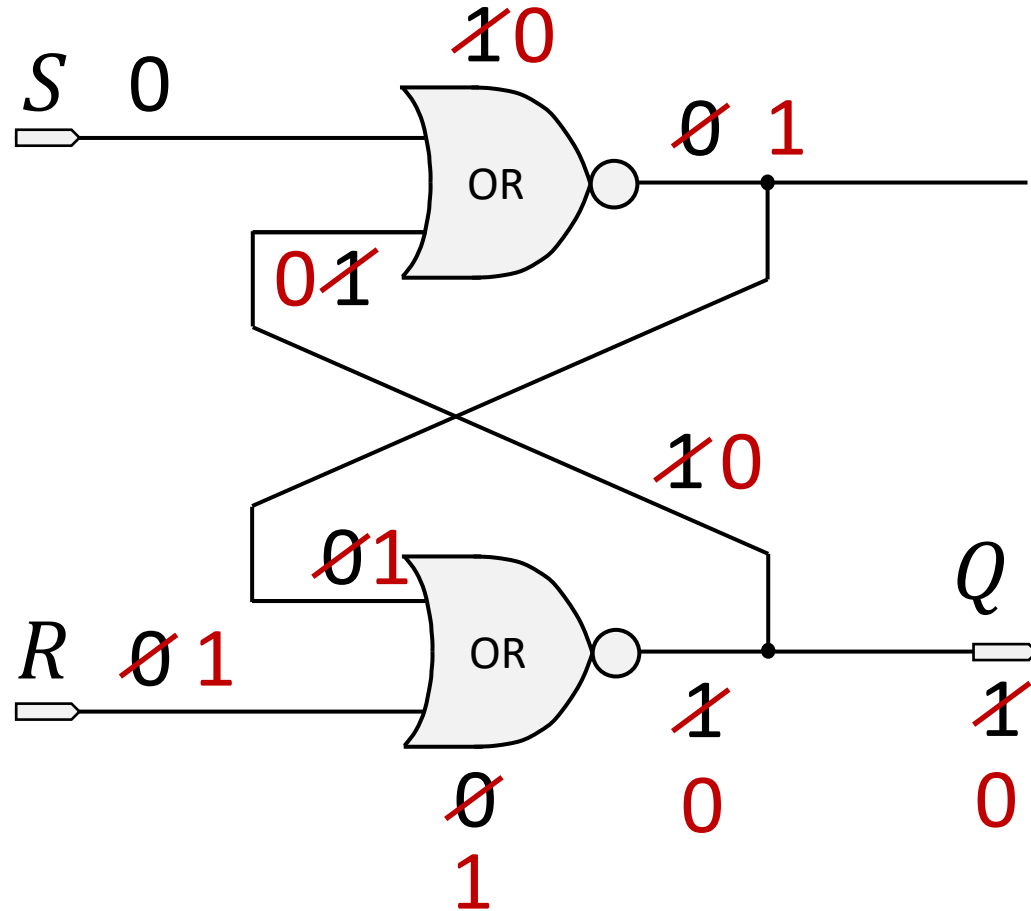
Circuit Behavior Analysis

Next, assume that R changes from “0” to “1”



Circuit Behavior Analysis

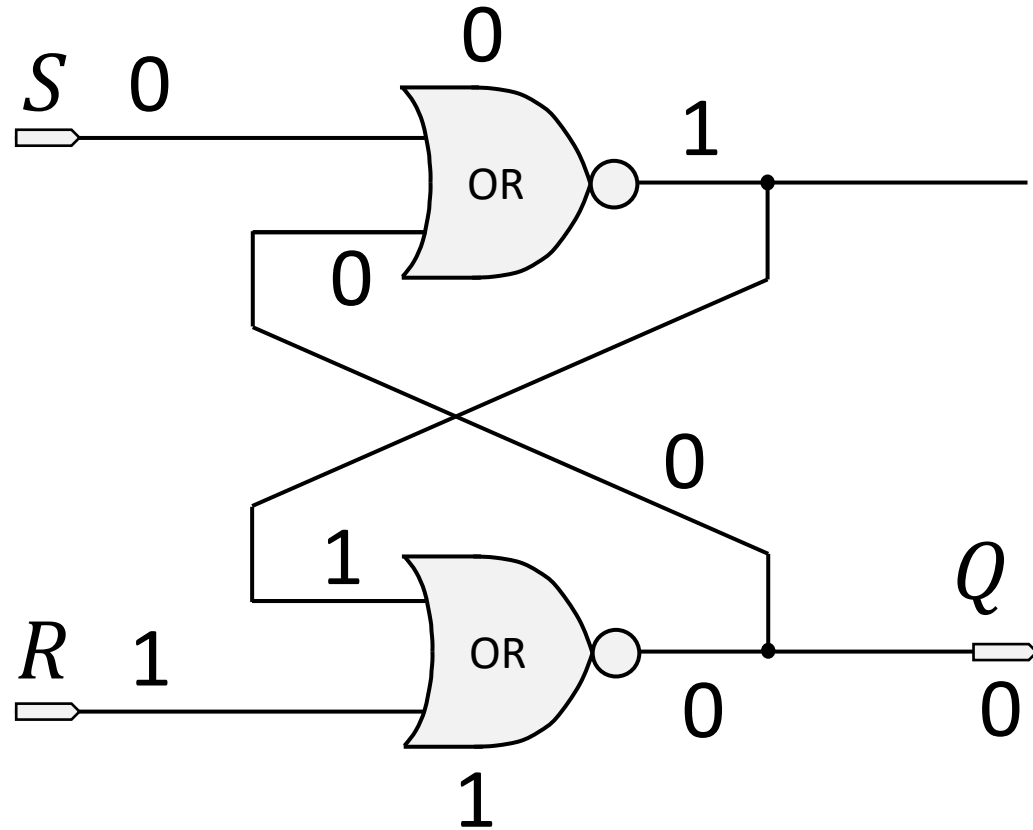
Next, assume that R changes from “0” to “1”



Q changes to “0”

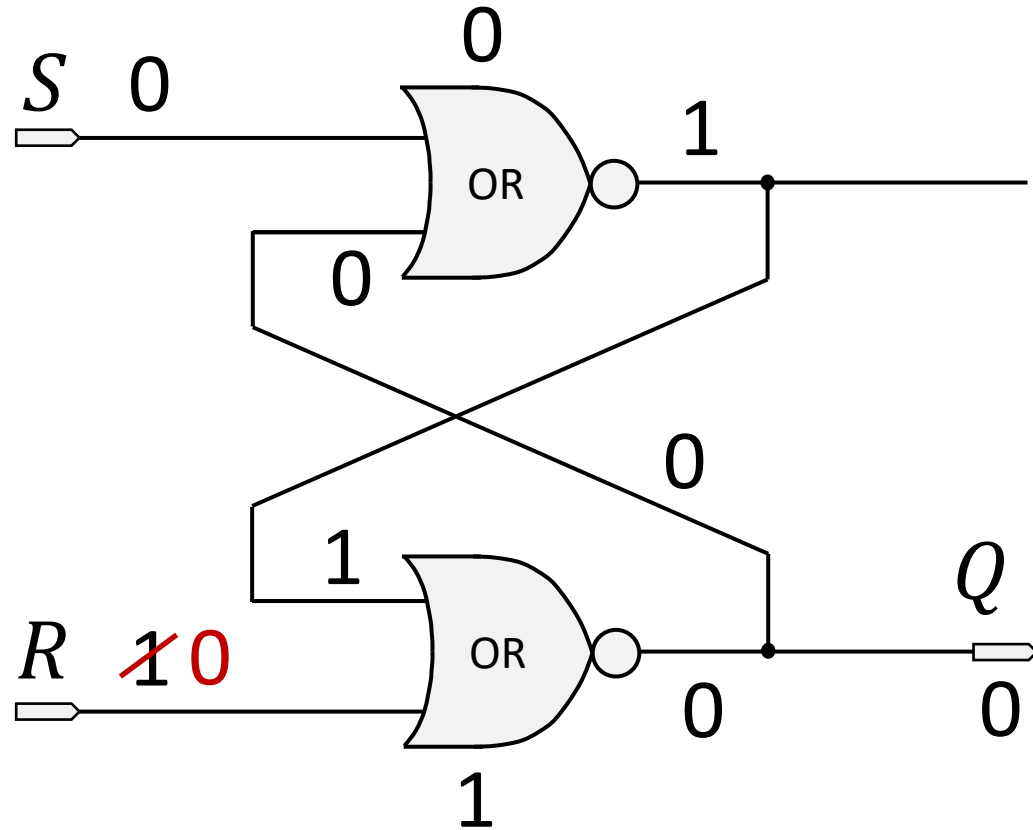
Circuit Behavior Analysis

Finally, let R to change from “1” back to “0”



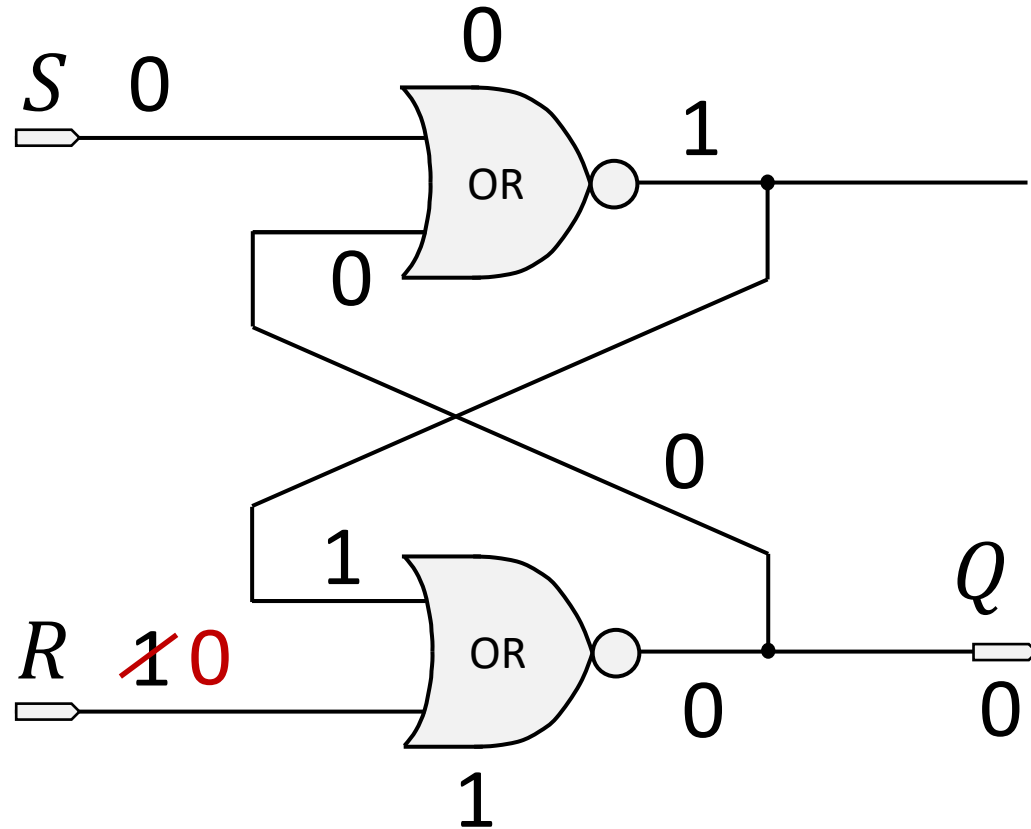
Circuit Behavior Analysis

Finally, let R to change from “1” back to “0”



Circuit Behavior Analysis

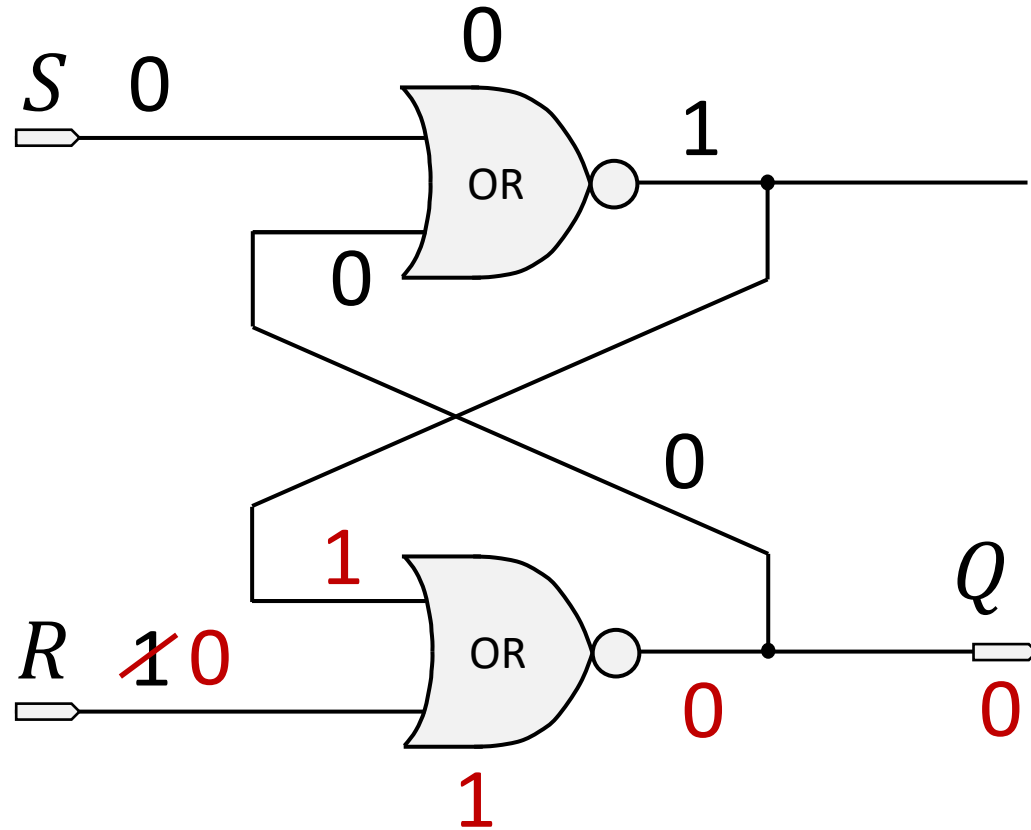
Finally, let R to change from “1” back to “0”



NOR gate outputs “0” if at least one input is “1”

Circuit Behavior Analysis

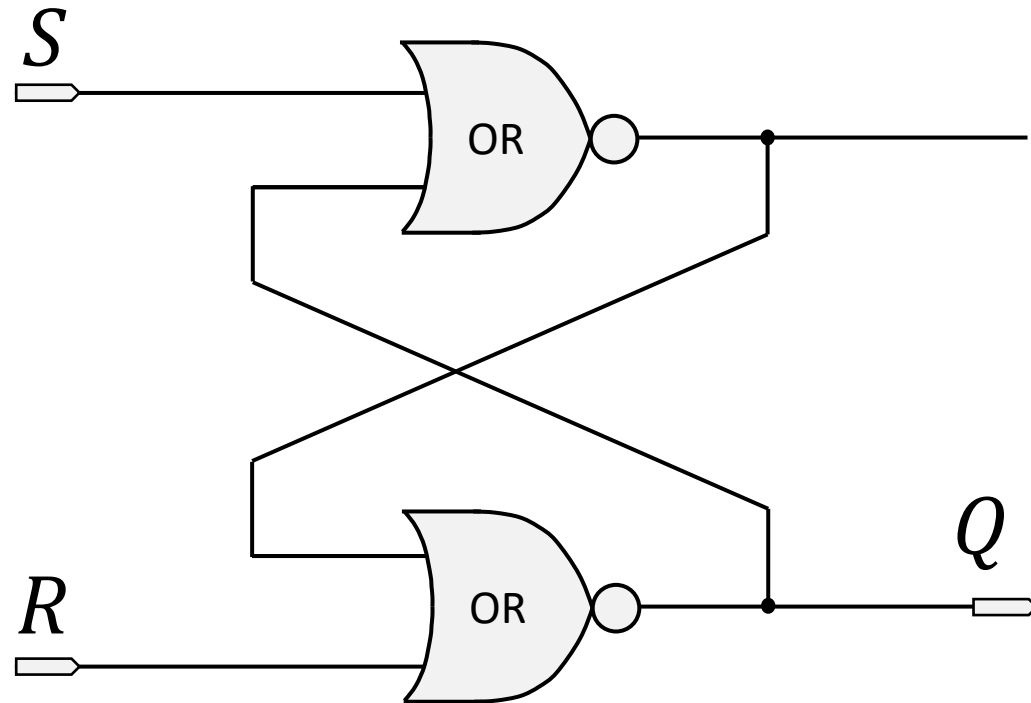
Finally, let R to change from “1” back to “0”



Q does not change

Circuit Behavior Analysis

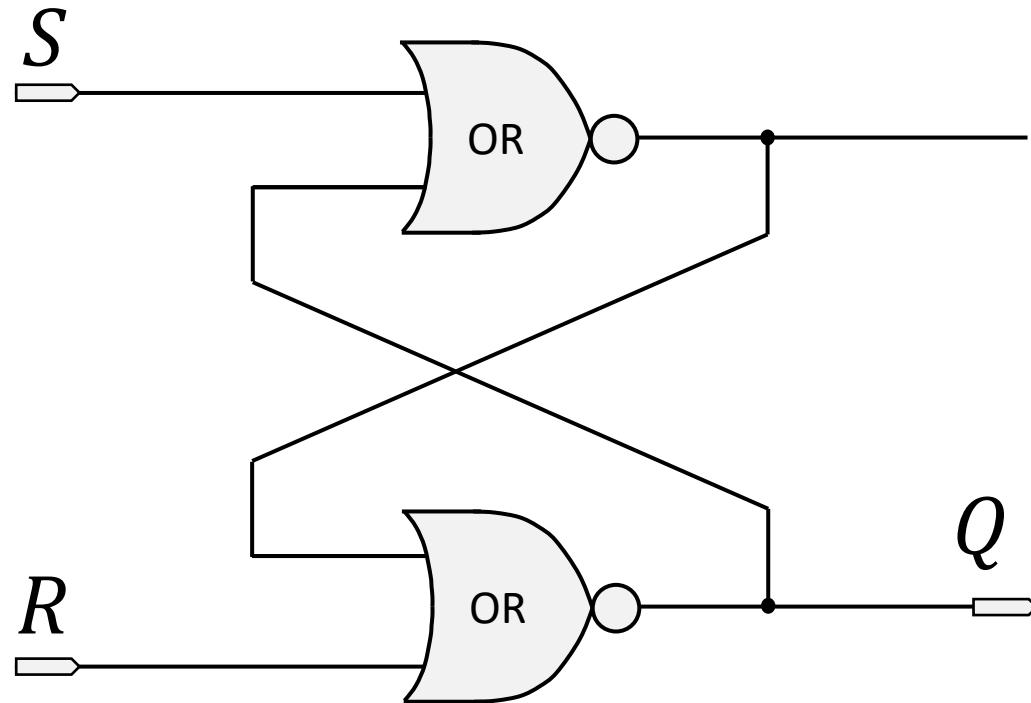
$S = 1, R = 0: \quad Q = 1$



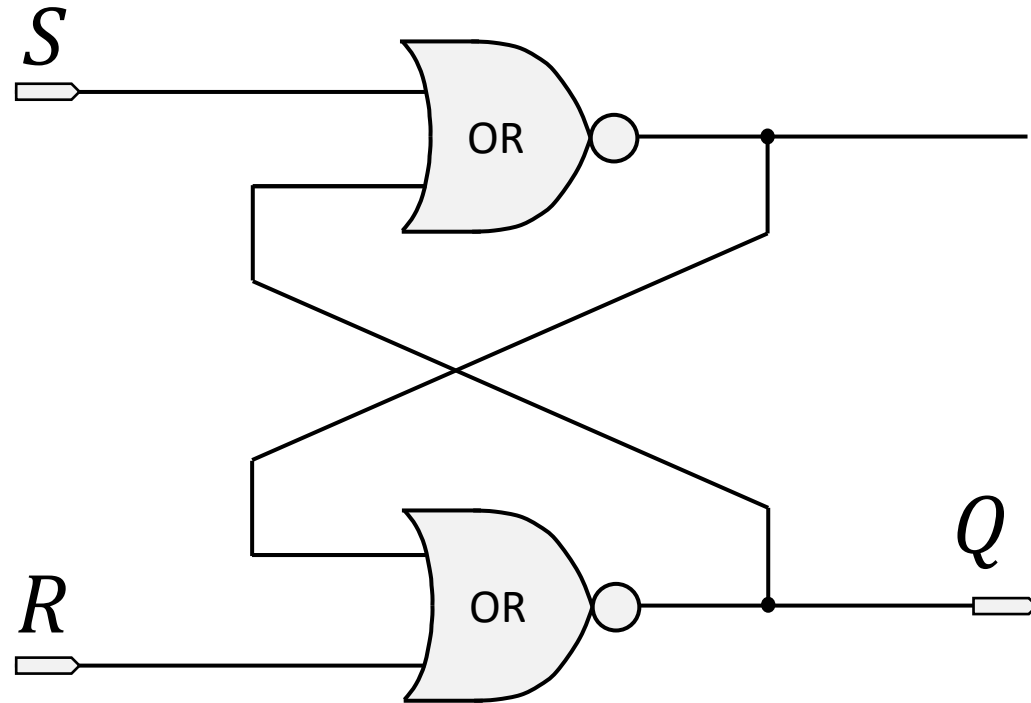
Circuit Behavior Analysis

$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$



Circuit Behavior Analysis

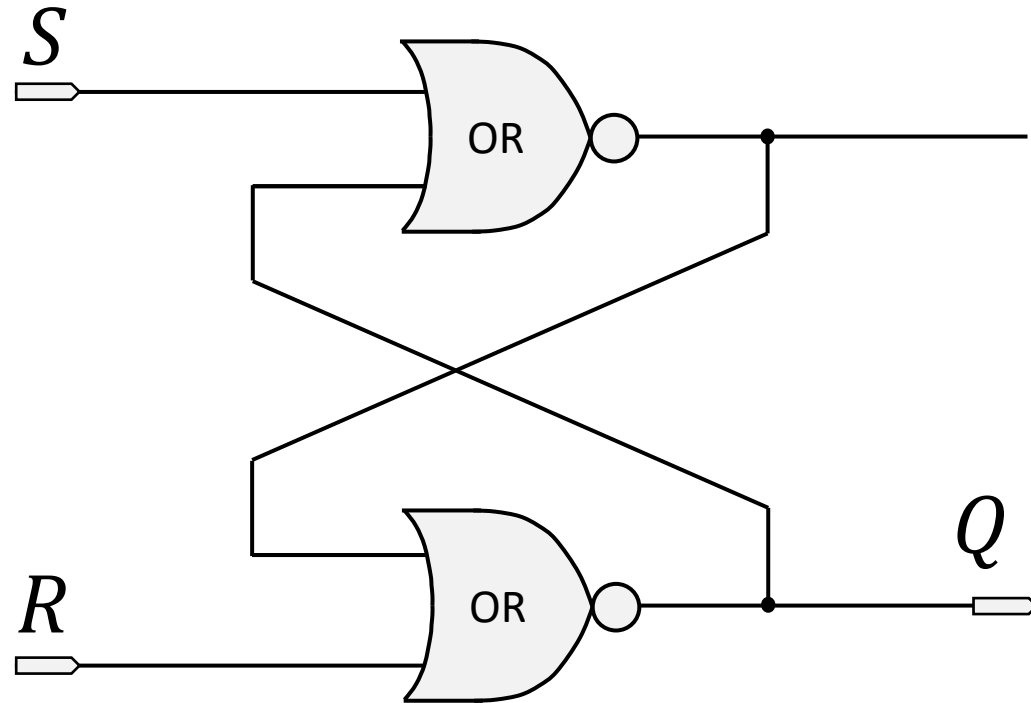


$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$

$S = 0, R = 0:$ Q is stored by circuit

Circuit Behavior Analysis



$S = 1, R = 0: \quad Q = 1$

$S = 0, R = 1: \quad Q = 0$

$S = 0, R = 0: \quad Q$ is stored by circuit

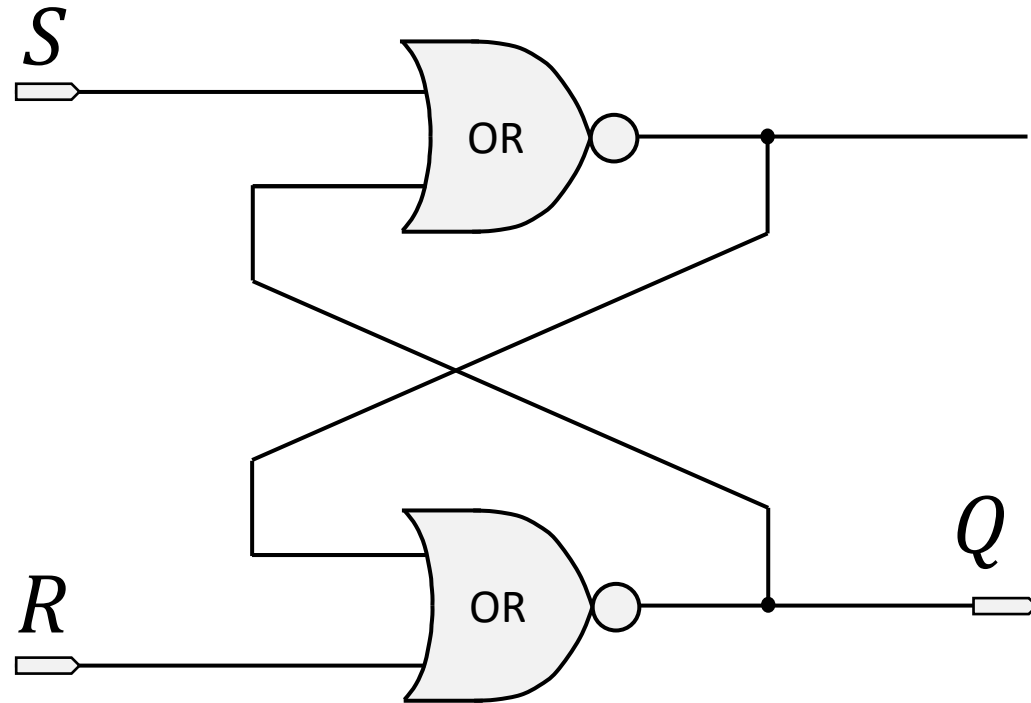
S

R

Q

t

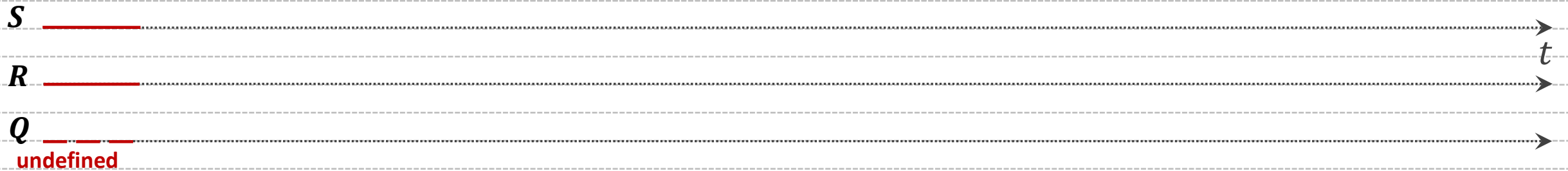
Circuit Behavior Analysis



$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$

$S = 0, R = 0:$ Q is stored by circuit

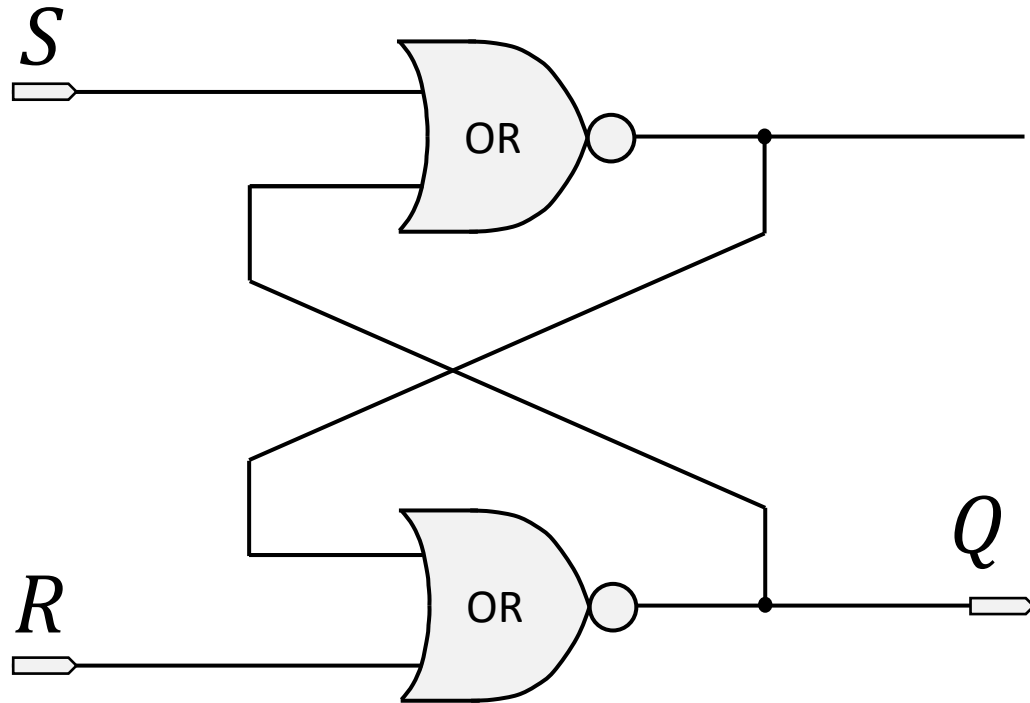


Circuit Behavior Analysis

$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$

$S = 0, R = 0:$ Q is stored by circuit

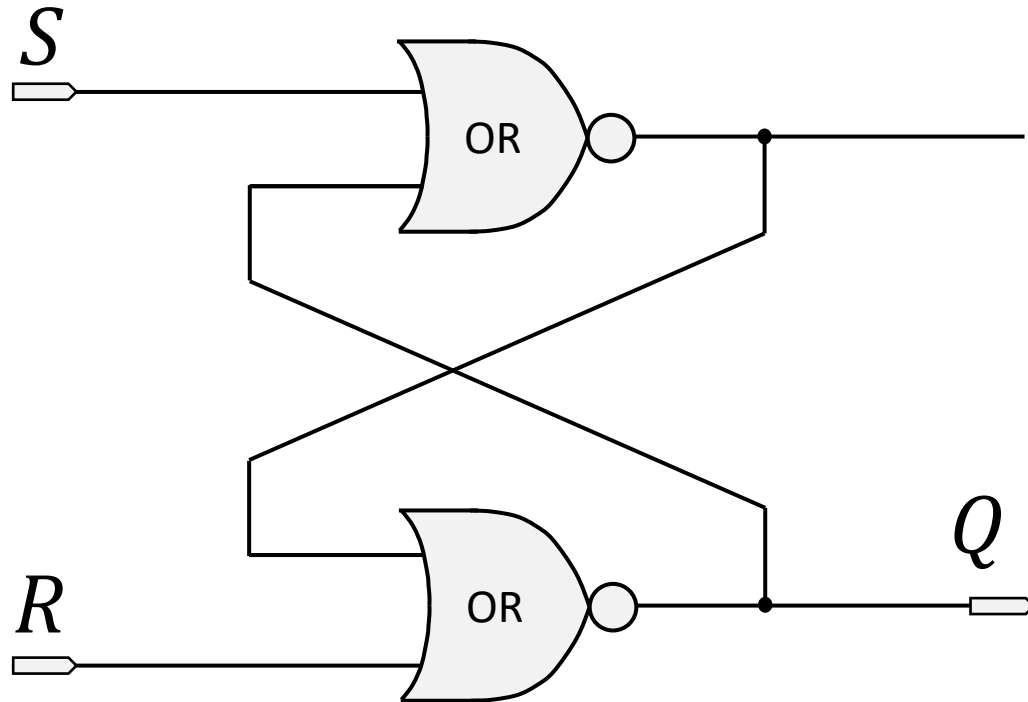


Circuit Behavior Analysis

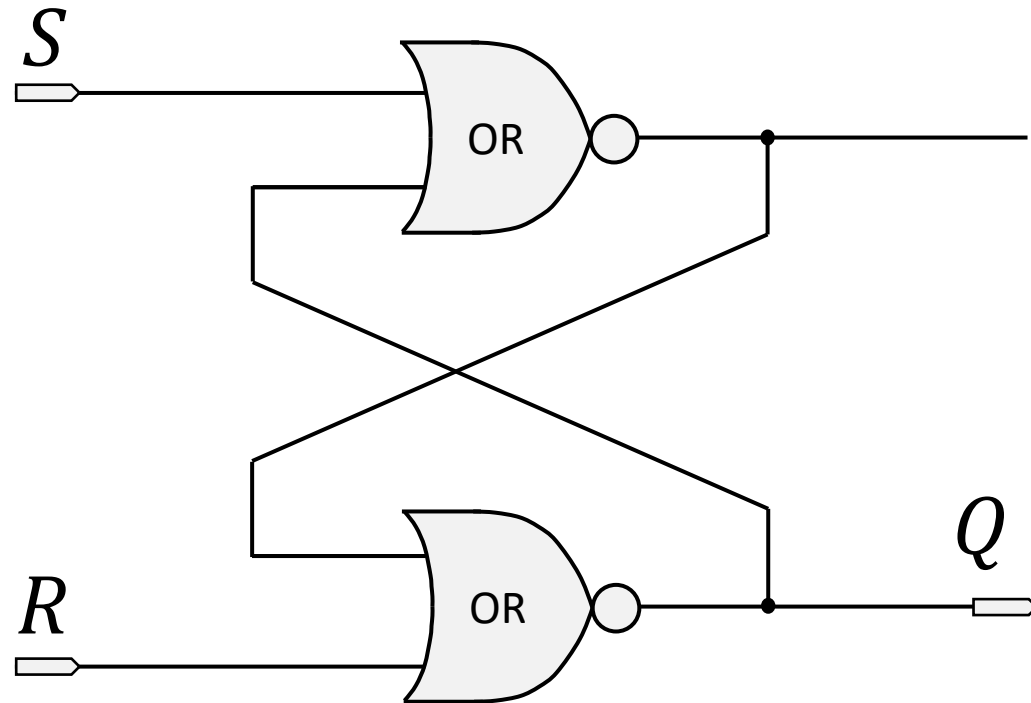
$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$

$S = 0, R = 0:$ Q is stored by circuit



Circuit Behavior Analysis



$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$

$S = 0, R = 0:$ Q is stored by circuit

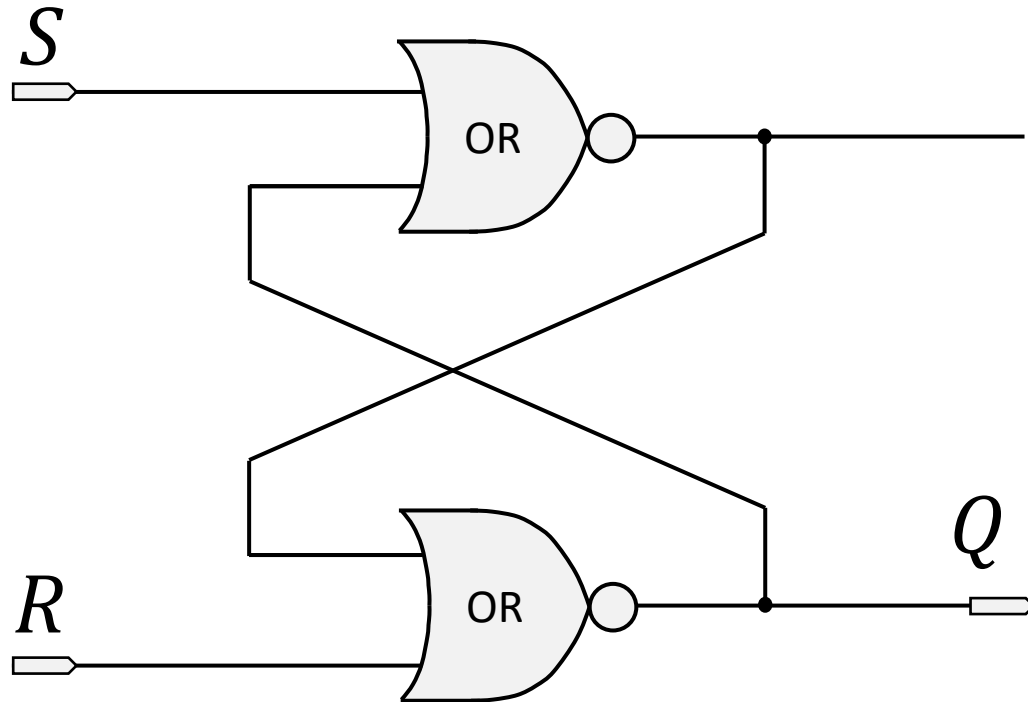


Circuit Behavior Analysis

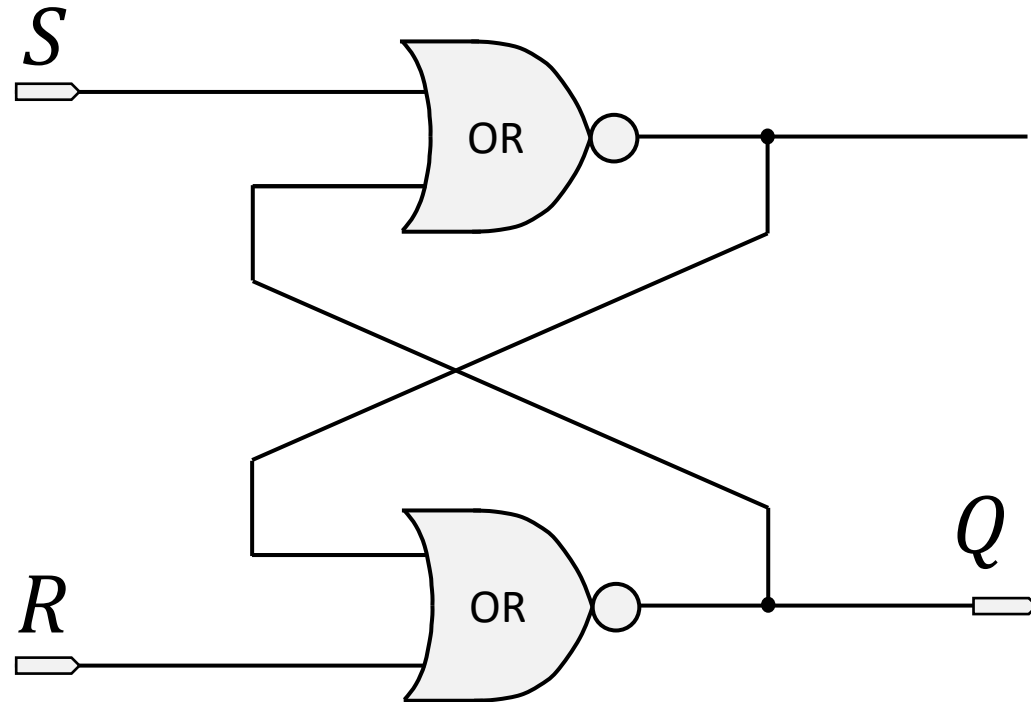
$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$

$S = 0, R = 0:$ Q is stored by circuit



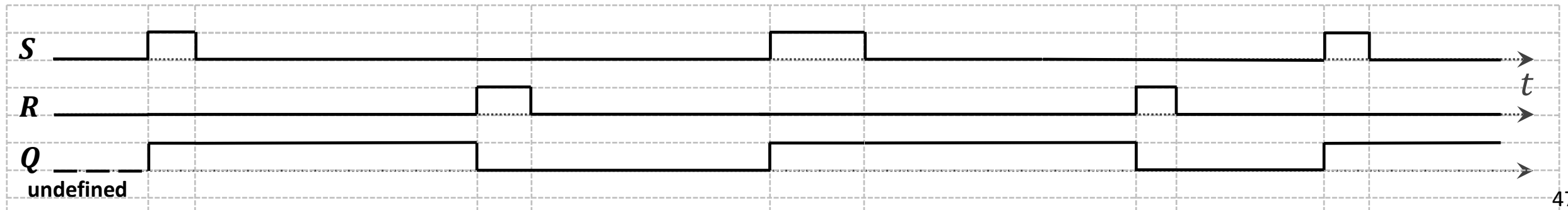
Circuit Behavior Analysis



$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$

$S = 0, R = 0:$ Q is stored by circuit

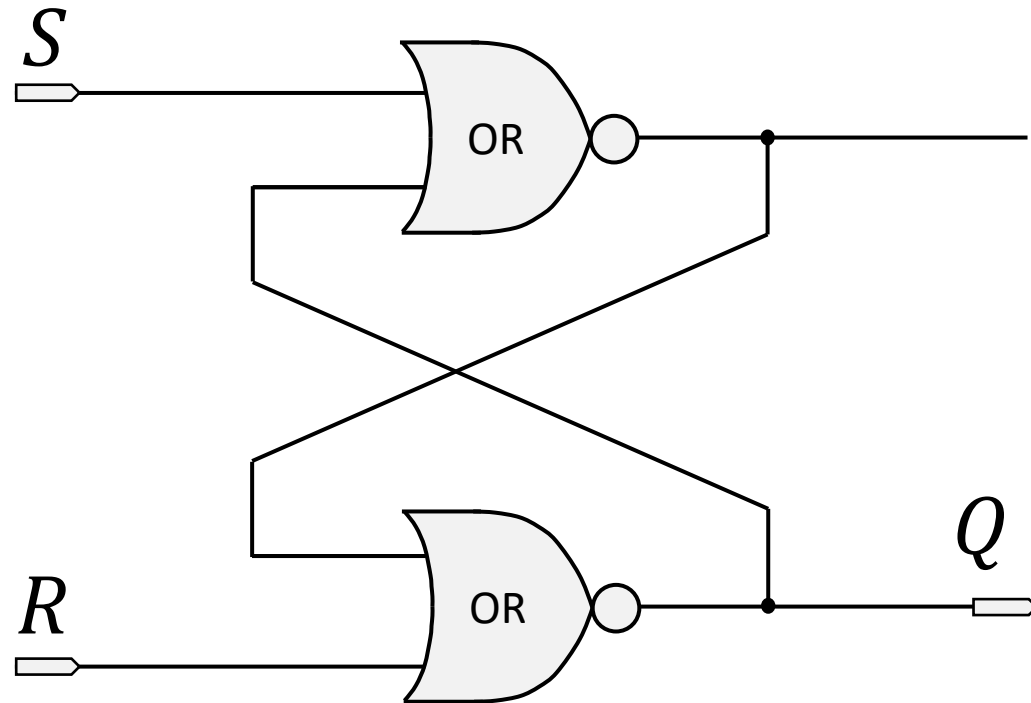


Circuit Behavior Analysis

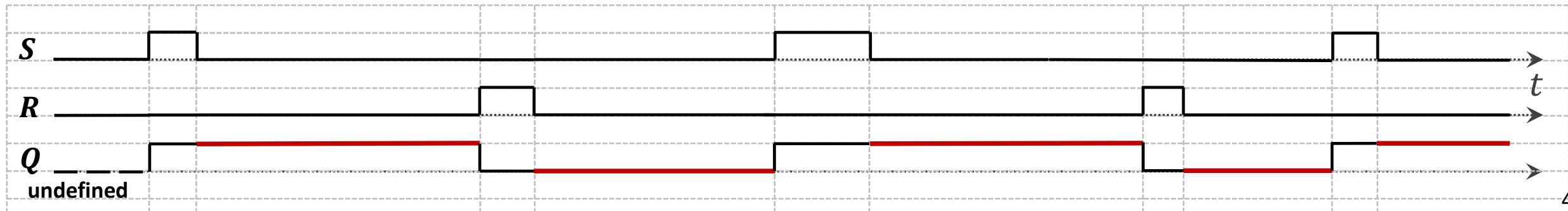
$S = 1, R = 0: \quad Q = 1$

$S = 0, R = 1: \quad Q = 0$

$S = 0, R = 0: \quad Q \text{ is stored by circuit}$

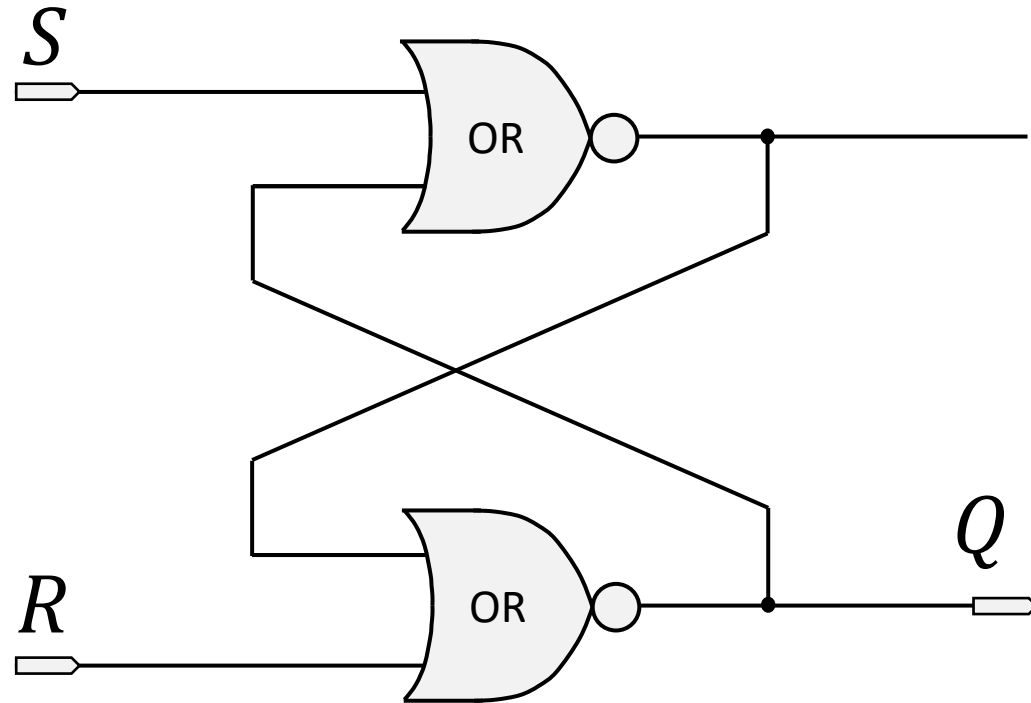


**The value of Q is stored by a circuit for some time
(due to complex physics processes)**



S/R Latch Circuit

Stores 1 data bit



$$S = 1, R = 0: \quad Q = 1$$

$$S = 0, R = 1: \quad Q = 0$$

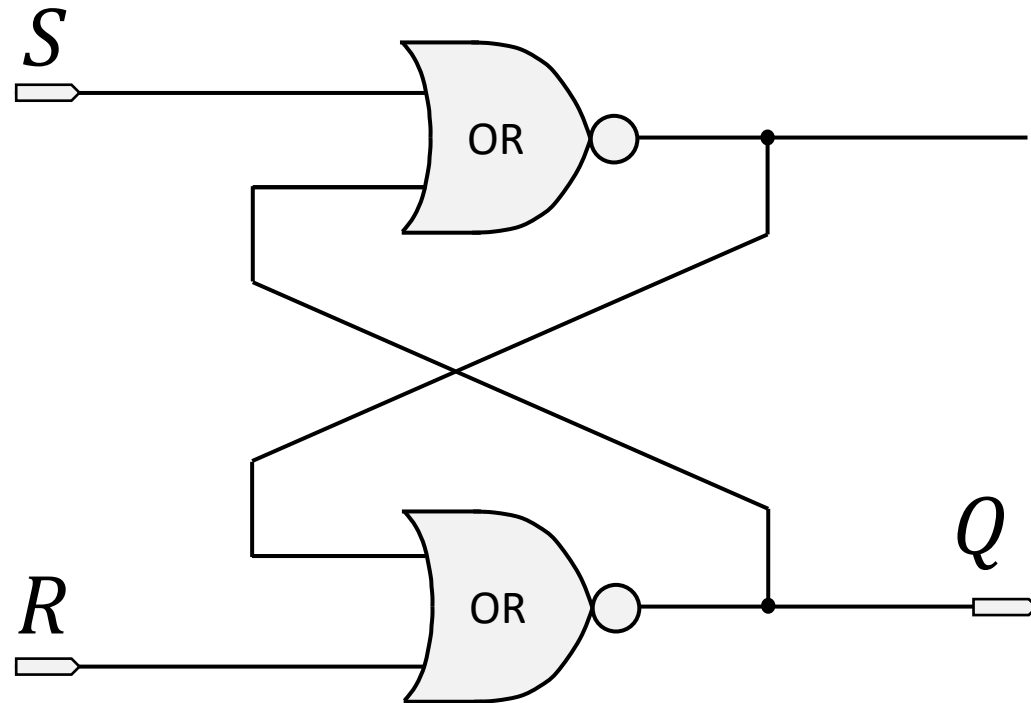
$$S = 0, R = 0: \quad \text{Q is stored by circuit}$$

The value of Q is stored by a circuit for some time
(due to complex physics processes)



Set/Reset Latch Circuit

Stores 1 data bit



$$S = 1, R = 0: \quad Q = 1$$

$$S = 0, R = 1: \quad Q = 0$$

$$S = 0, R = 0: \quad Q \text{ is stored by circuit}$$

Interpretation:

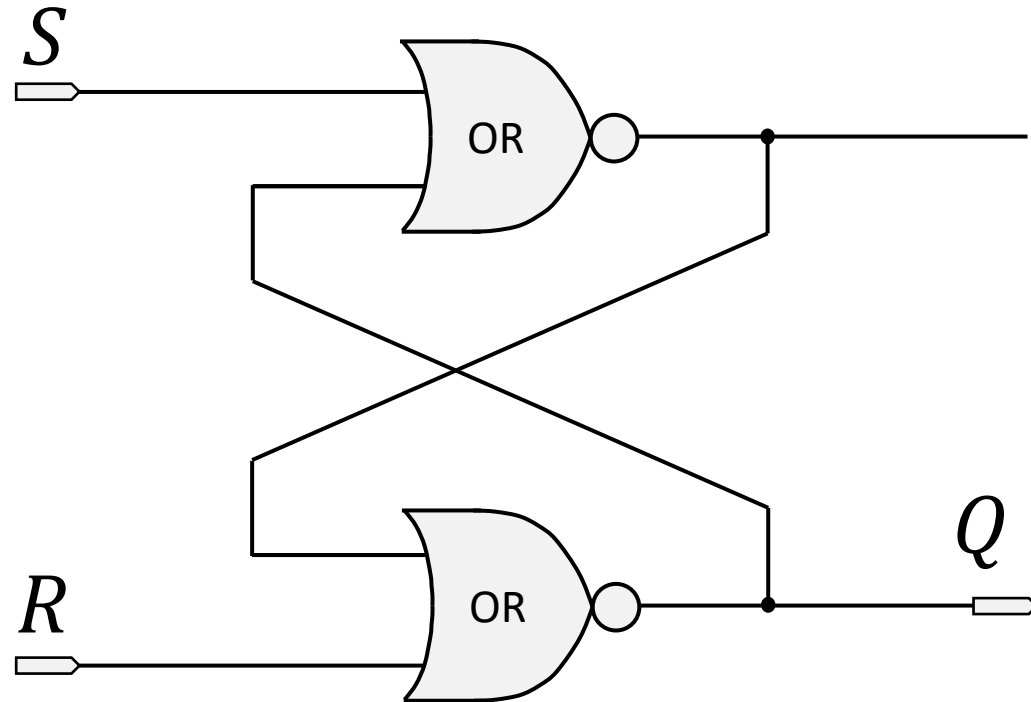
S – “set” pin

R – “reset” pin

Q – stored value

Set/Reset Latch Circuit

Stores 1 data bit



$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$

$S = 0, R = 0:$ Q is stored by circuit

$S = 1, R = 1:$ An illegal combination

Interpretation:

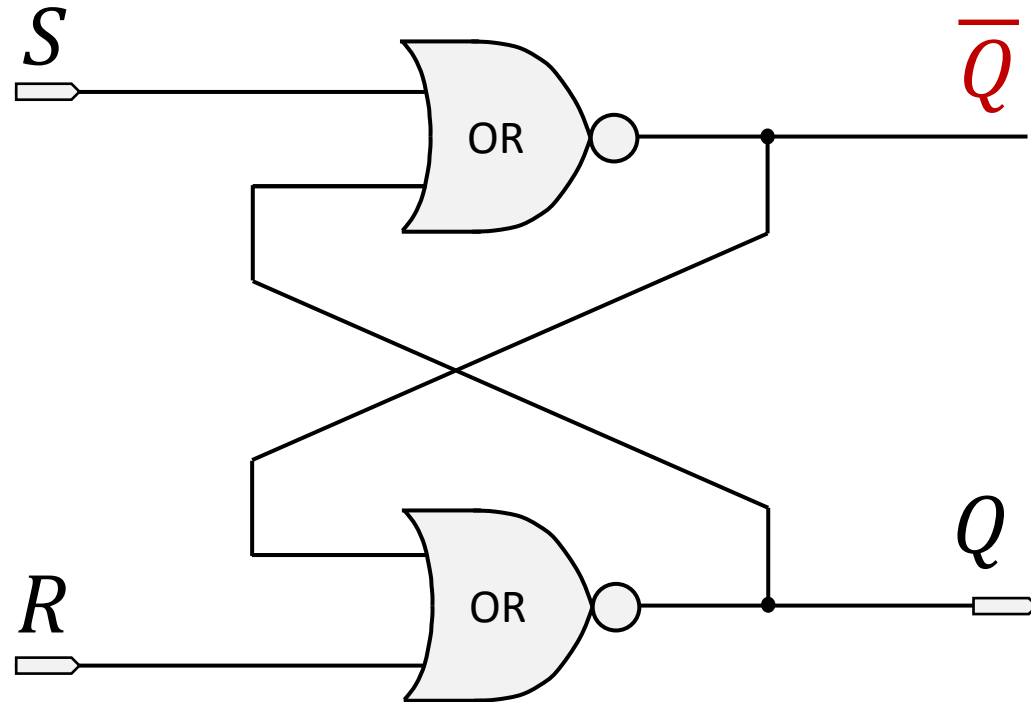
S – “set” pin

R – “reset” pin

Q – stored value

Set/Reset Latch Circuit

Stores 1 data bit



$S = 1, R = 0:$ $Q = 1$

$S = 0, R = 1:$ $Q = 0$

$S = 0, R = 0:$ Q is stored by circuit

$S = 1, R = 1:$ An illegal combination

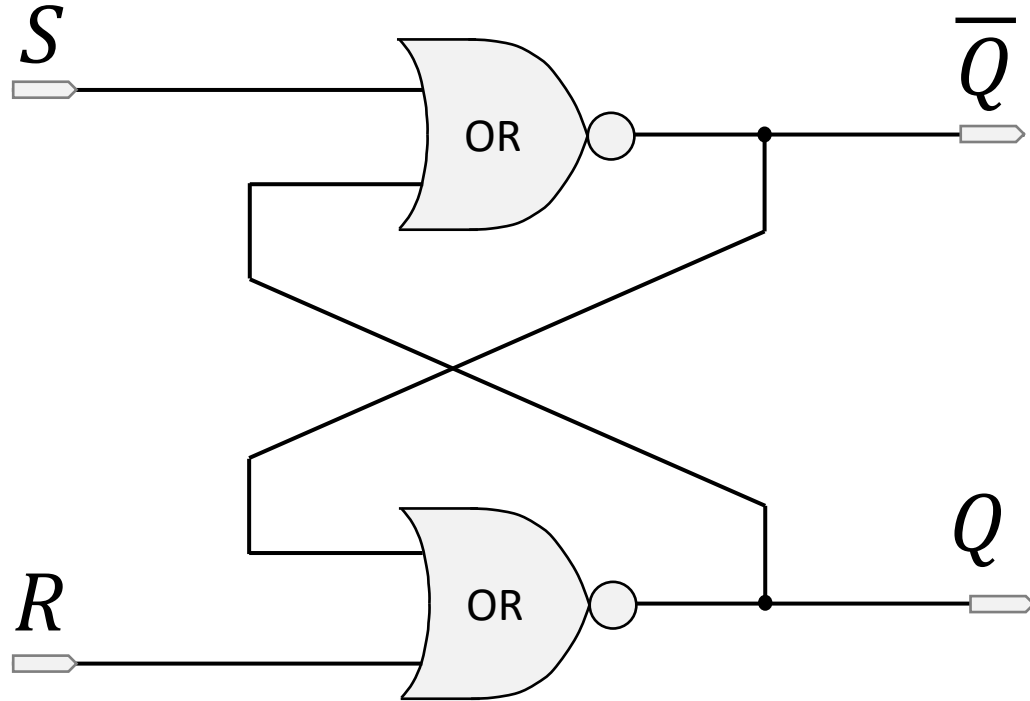
Interpretation:

S – “set” pin

R – “reset” pin

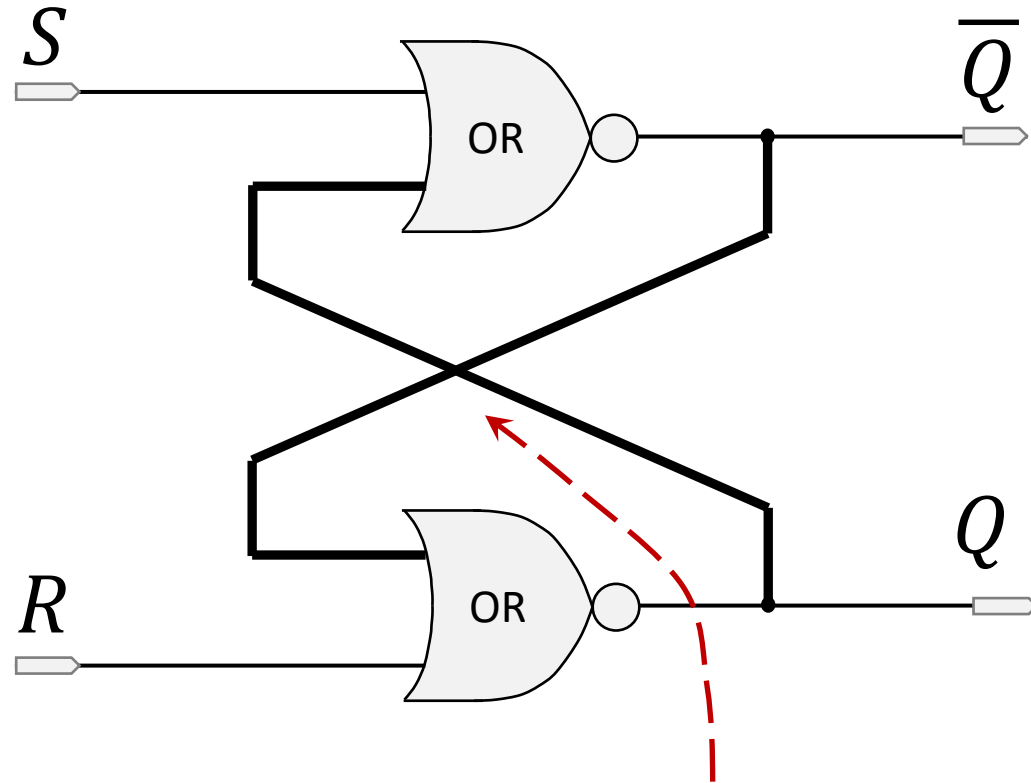
Q – stored value

Set/Reset Latch Circuit: Truth Table



S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal inputs

Set/Reset Latch: Cross-Coupled Interconnection – Its Feature

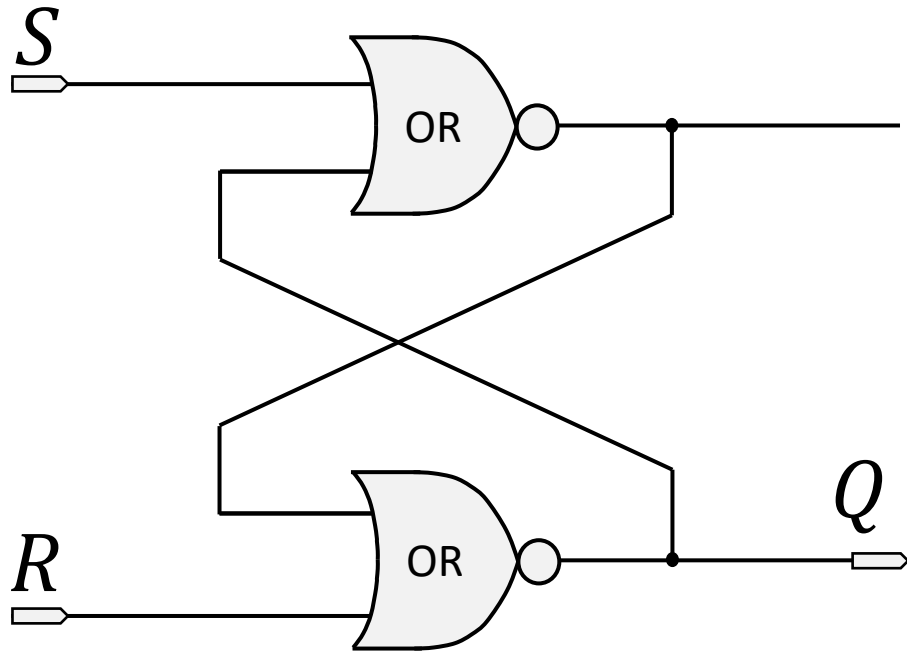


S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal inputs

The output of one gate serves as an input to another gate

Bistable Multivibrator (Latch)

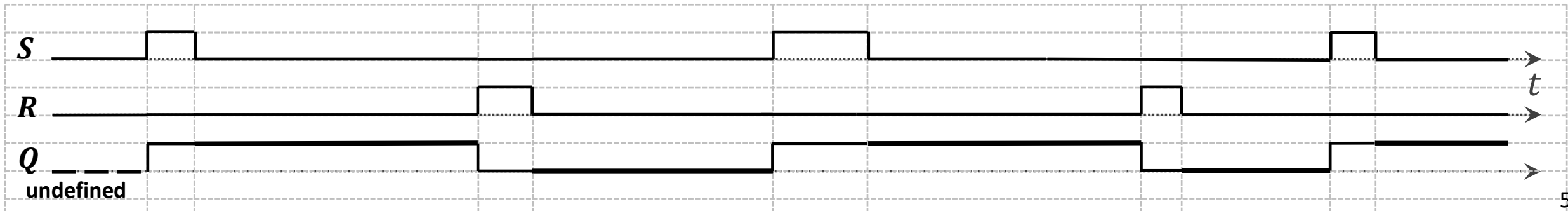
Circuit



$$S = 1, R = 0: \quad Q = 1$$

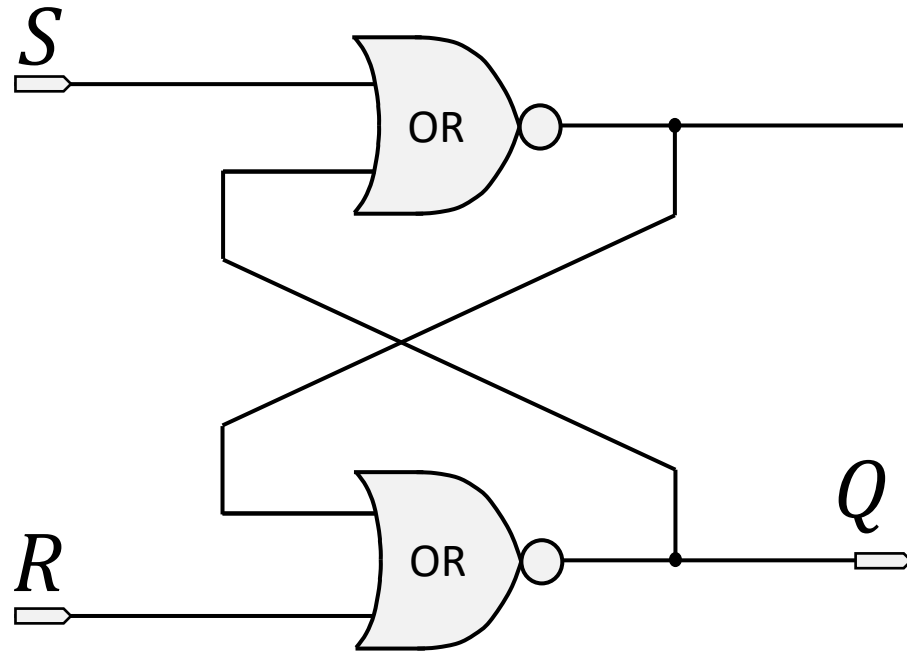
$$S = 0, R = 1: \quad Q = 0$$

$$S = 0, R = 0: \quad Q \text{ is stored by circuit}$$



Bistable Multivibrator (Latch)

Circuit

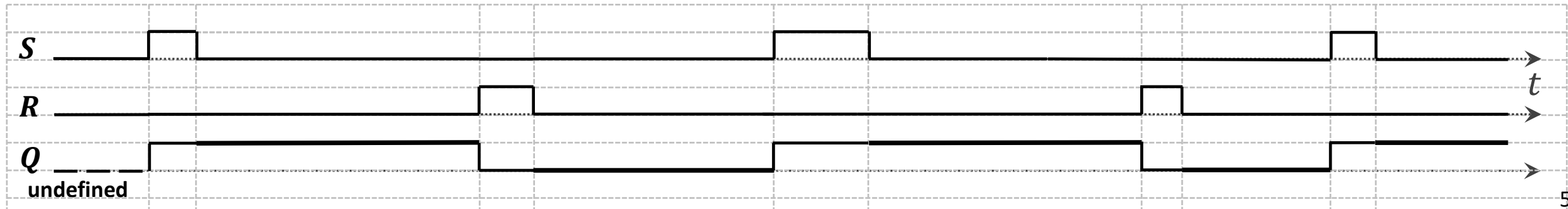


- **Stable in either state**

$$S = 1, R = 0: \quad Q = 1$$

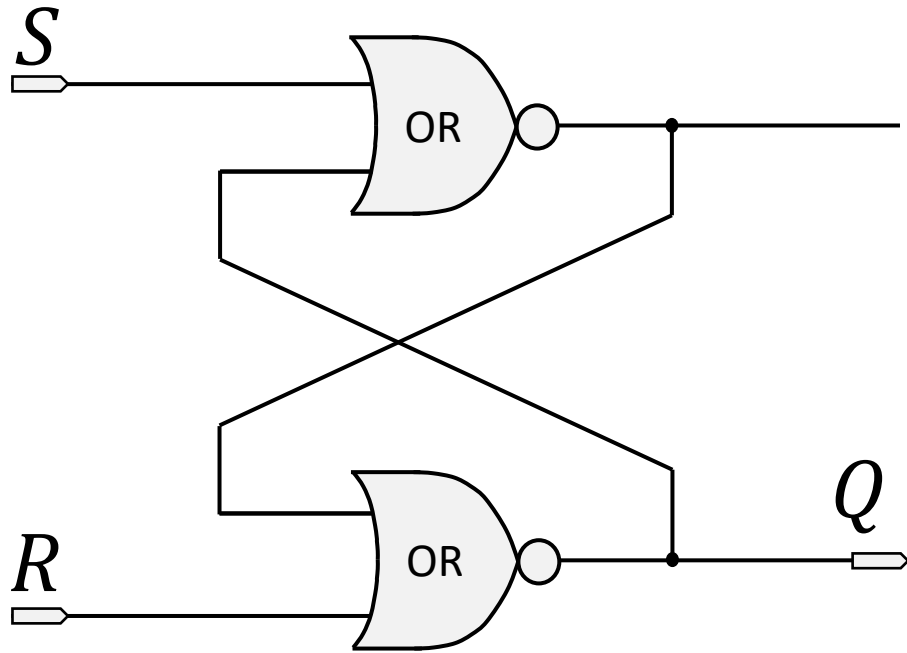
$$S = 0, R = 1: \quad Q = 0$$

$$S = 0, R = 0: \quad Q \text{ is stored by circuit}$$



Bistable Multivibrator (Latch)

Circuit

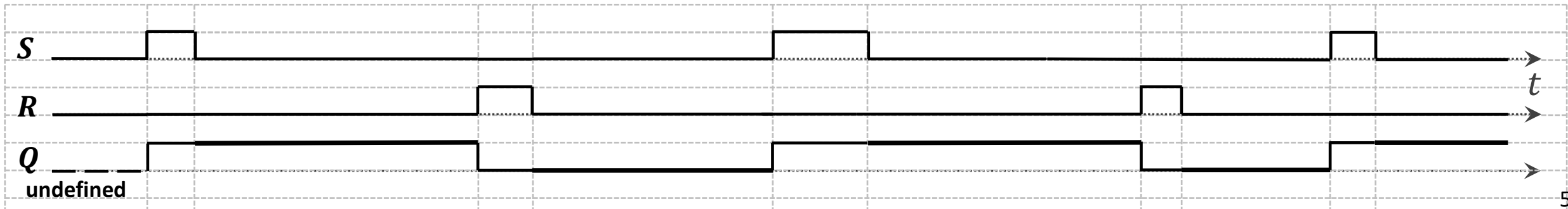


- Stable in either state;
- **Flipped from one state to the other by an external trigger pulse**

$$S = 1, R = 0: \quad Q = 1$$

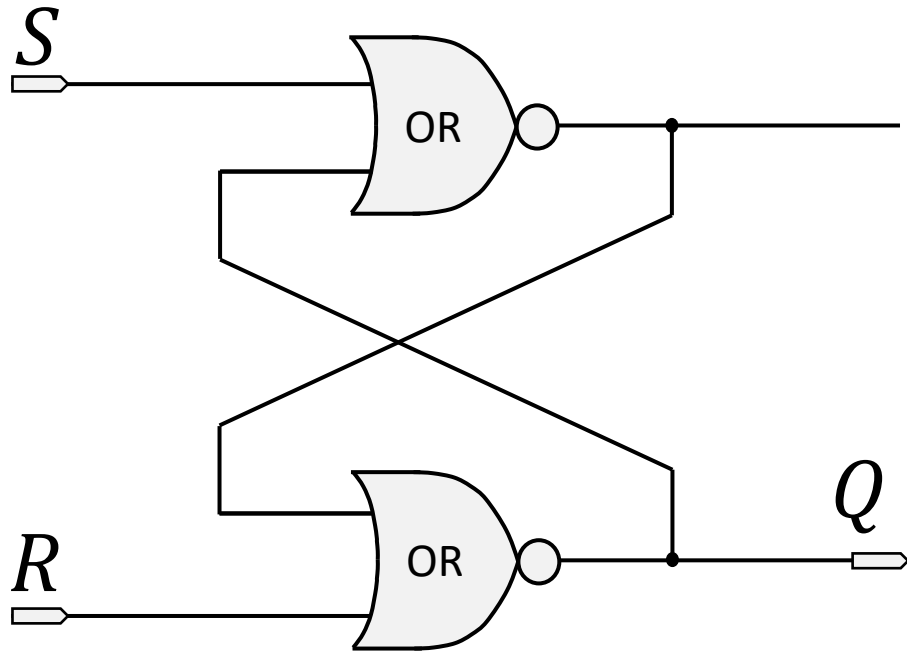
$$S = 0, R = 1: \quad Q = 0$$

$$S = 0, R = 0: \quad Q \text{ is stored by circuit}$$



Bistable Multivibrator (Latch)

Circuit

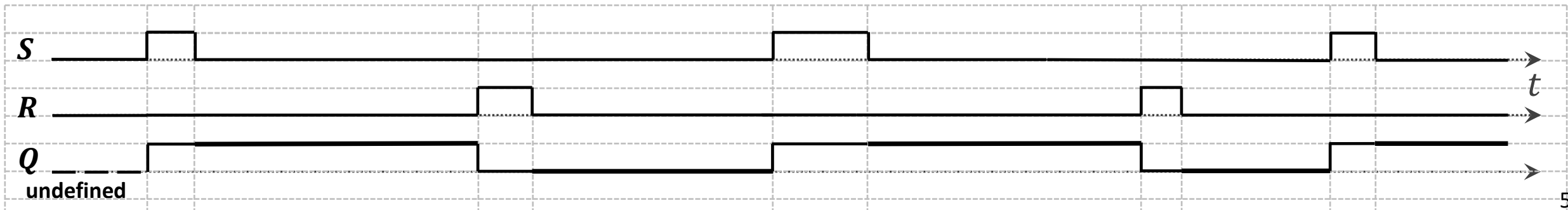


- Stable in either state;
- Flipped from one state to the other by an external trigger pulse;
- **Used to store one bit of data**

$$S = 1, R = 0: \quad Q = 1$$

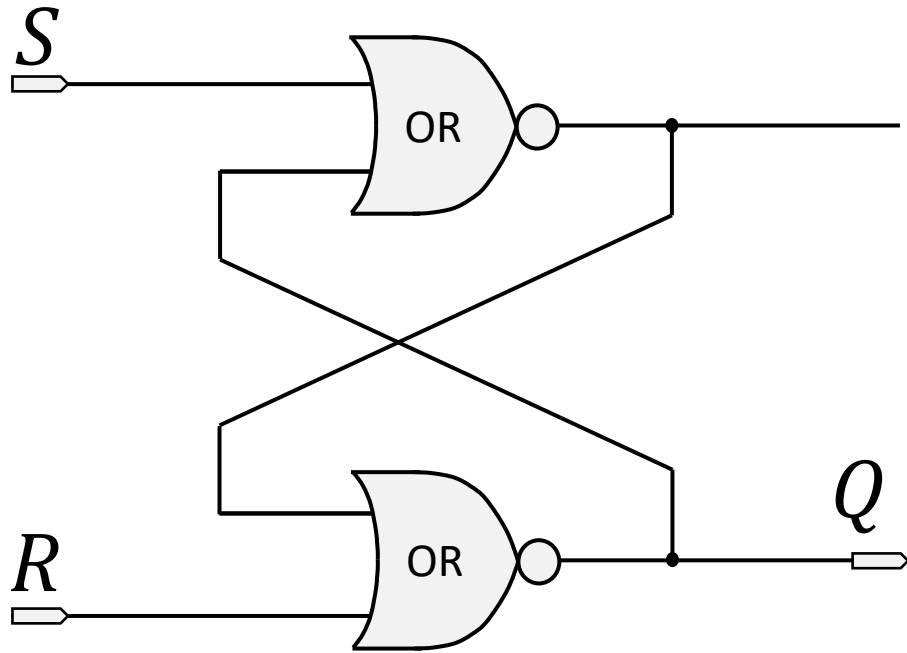
$$S = 0, R = 1: \quad Q = 0$$

$$S = 0, R = 0: \quad \mathbf{Q \text{ is stored by circuit}}$$



Bistable Multivibrator (Latch)

Circuit



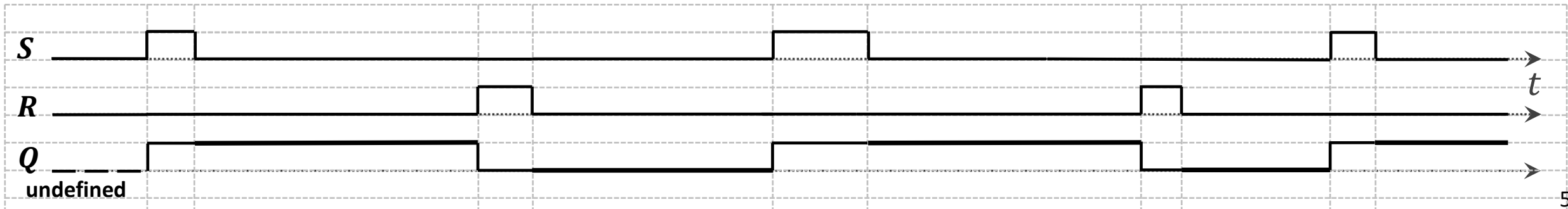
- Stable in either state;
- Flipped from one state to the other by an external trigger pulse;
- Used to store one bit of data;
- **Used in volatile computer memory**

(Wikipedia)

$$S = 1, R = 0: \quad Q = 1$$

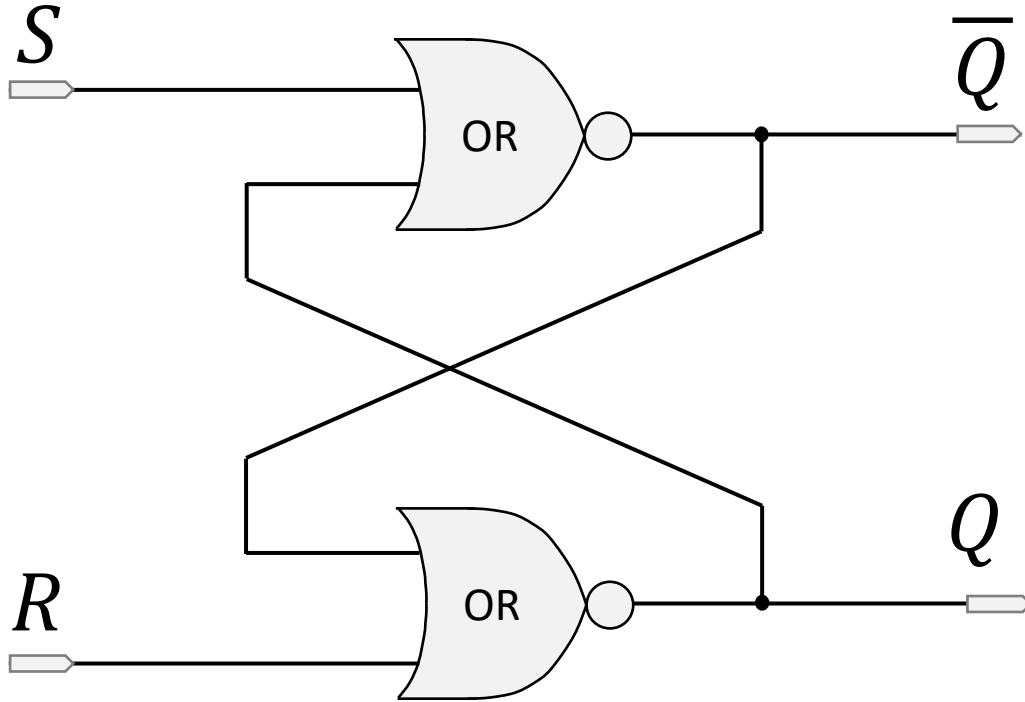
$$S = 0, R = 1: \quad Q = 0$$

$$S = 0, R = 0: \quad Q \text{ is stored by circuit}$$



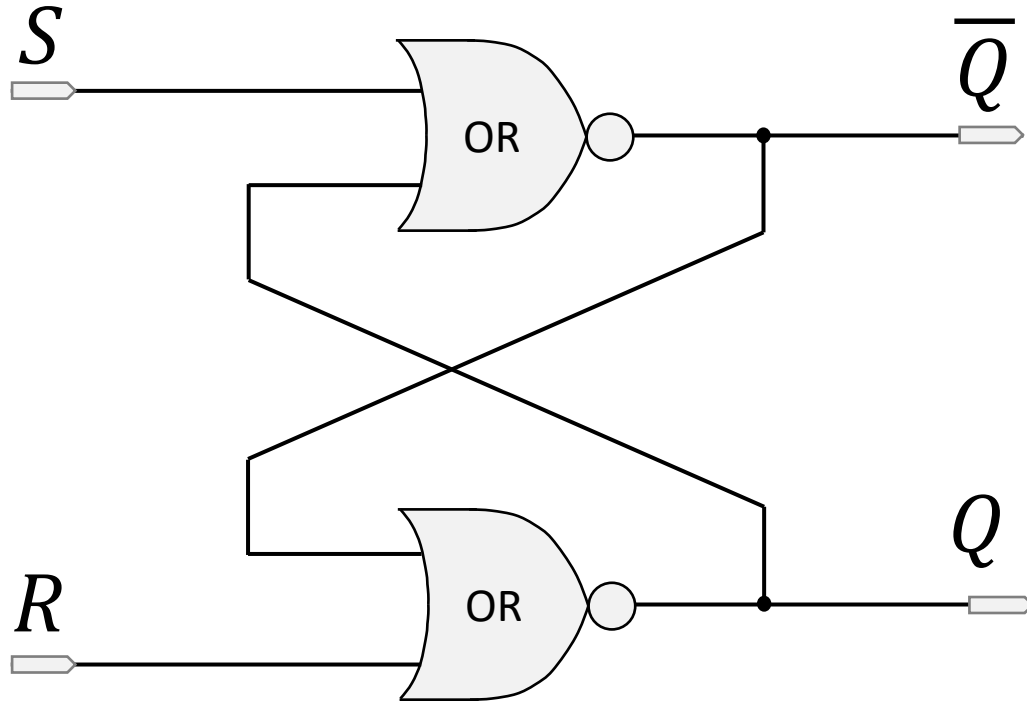
Set/Reset Latch: Multiple Implementations Available

1) Implementation by using NOR logic gates:

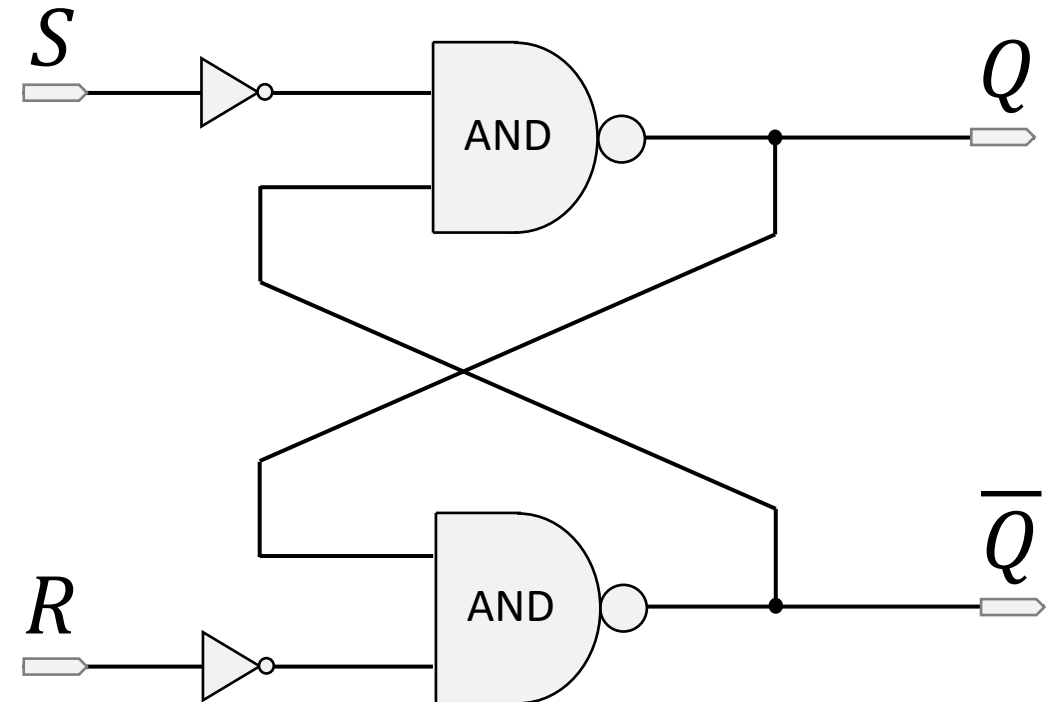


Set/Reset Latch: Multiple Implementations Available

1) Implementation by using NOR logic gates:

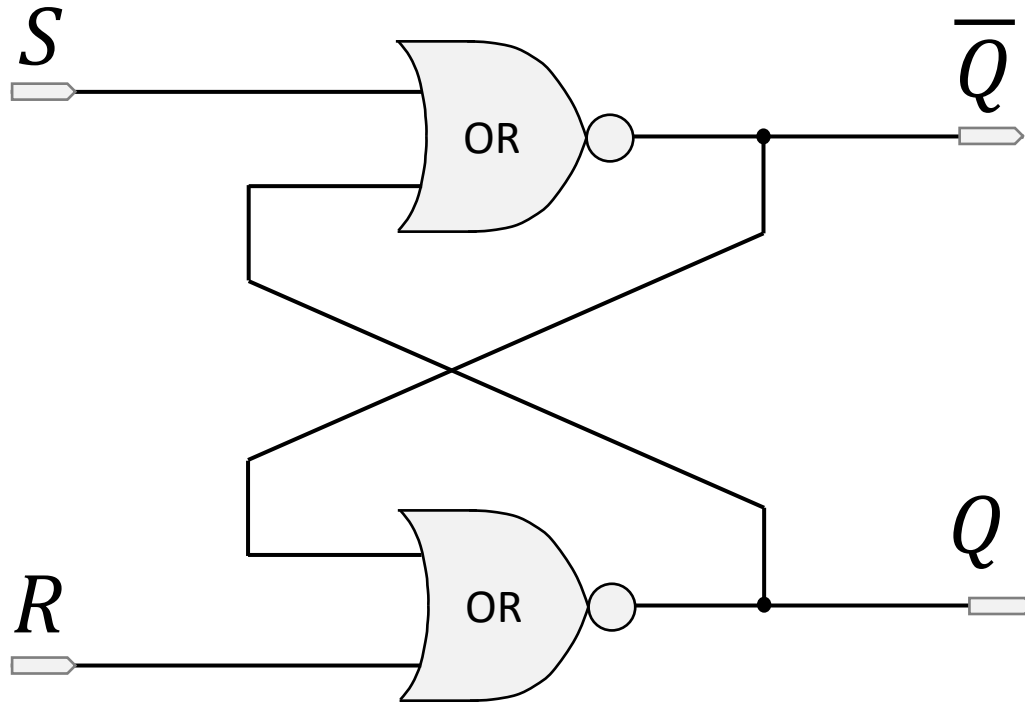


2) by using NAND logic gates:

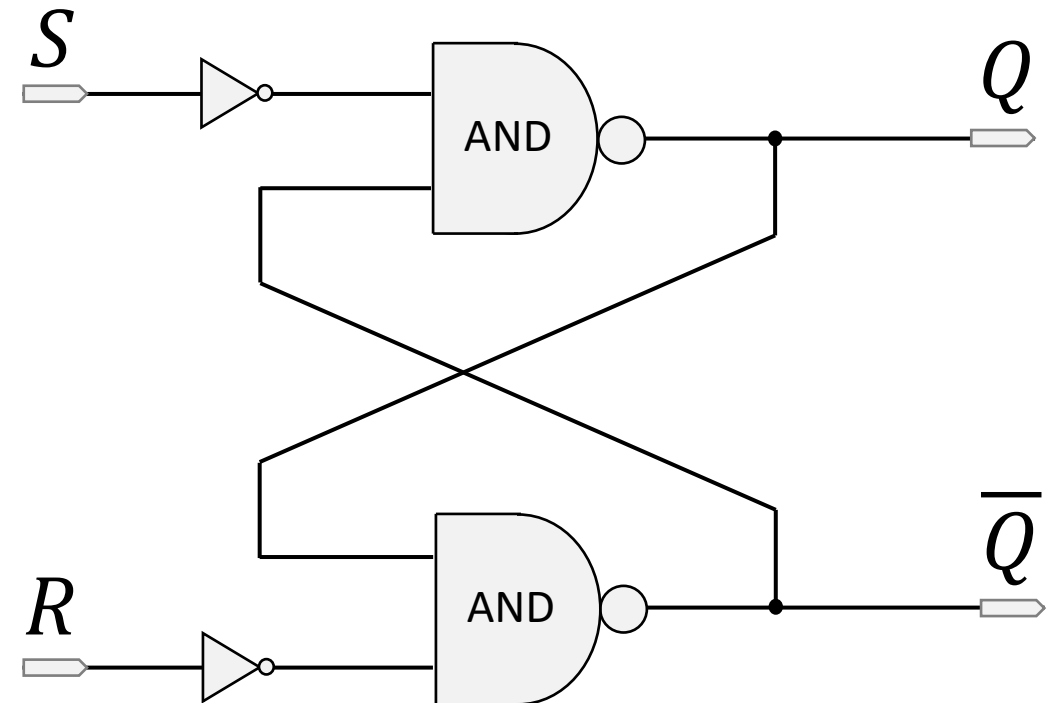


Set/Reset Latch: Multiple Implementations Available

1) Implementation by using NOR logic gates:

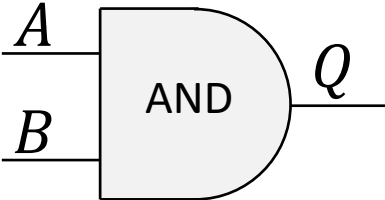
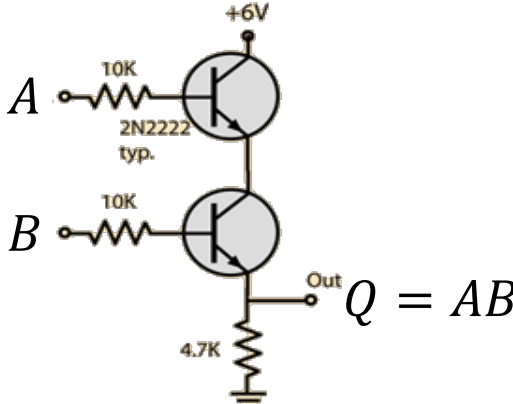
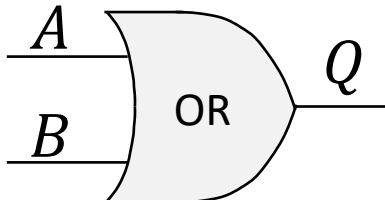
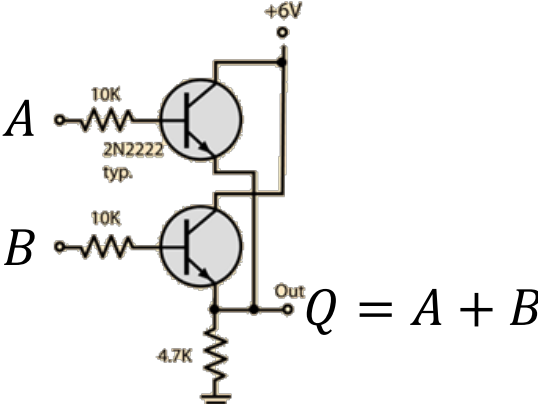
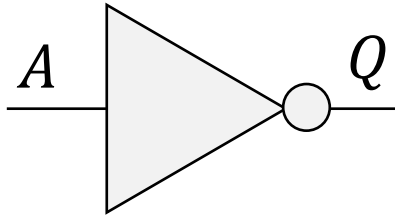
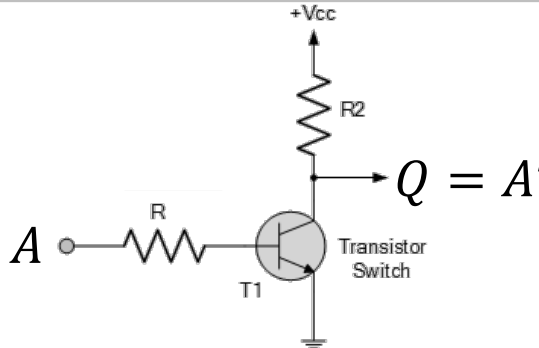


2) by using NAND logic gates:

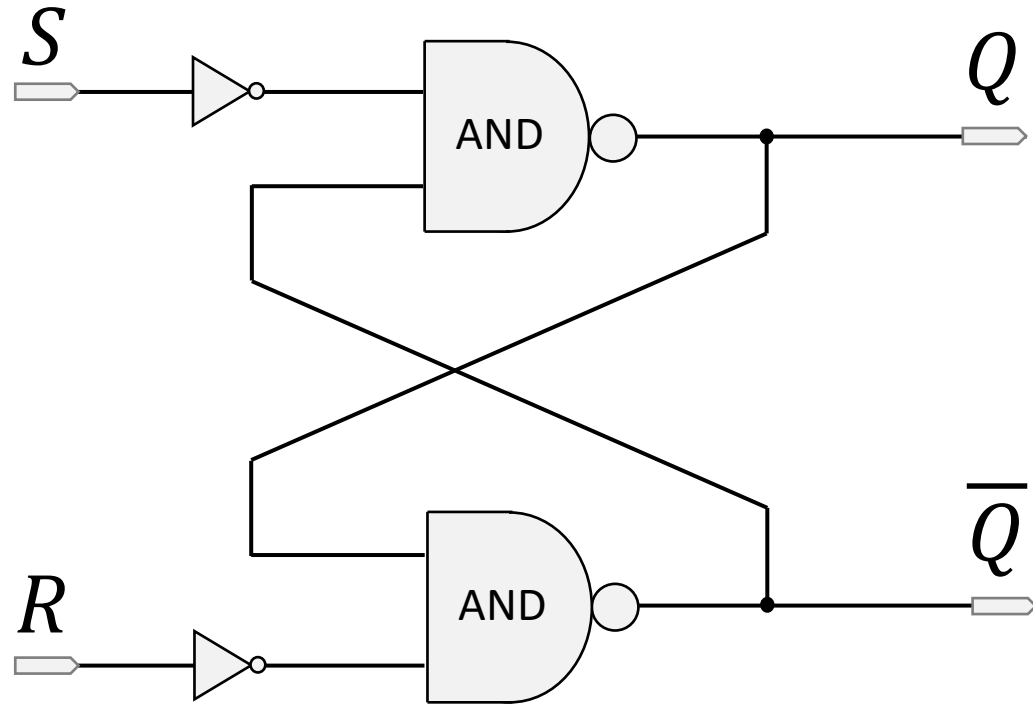


These representations are logical oversimplified representations, hiding many implementation details

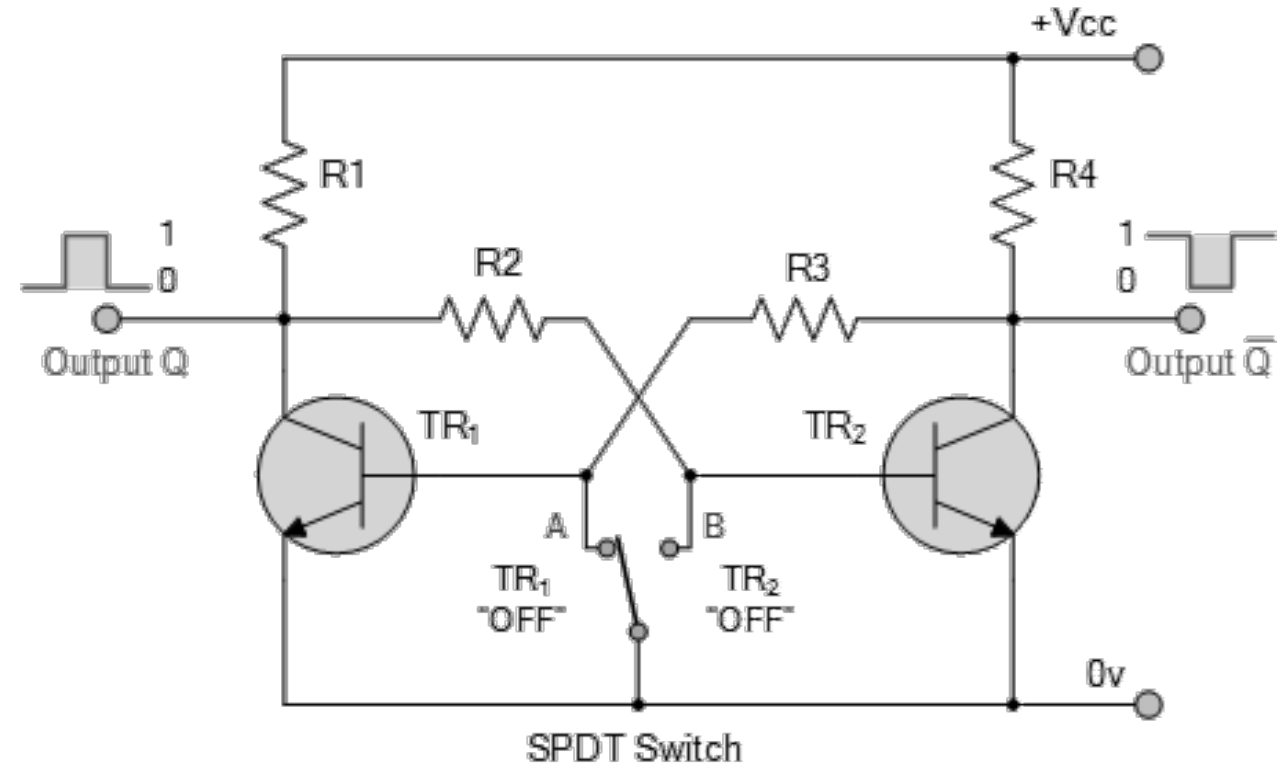
Recap: Each logic gate is an electronic circuit, implemented by using transistors

Logic Gate	Symbolic Representation	Truth Table	Implementation with Transistors															
AND		<table><tr><th>A</th><th>B</th><th>Q=AB</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Q=AB	0	0	0	1	0	0	0	1	0	1	1	1	
A	B	Q=AB																
0	0	0																
1	0	0																
0	1	0																
1	1	1																
OR		<table><tr><th>A</th><th>B</th><th>Q=A+B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	Q=A+B	0	0	0	1	0	1	0	1	1	1	1	1	
A	B	Q=A+B																
0	0	0																
1	0	1																
0	1	1																
1	1	1																
NOT		<table><tr><th>A</th><th>Q=A'</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	Q=A'	0	1	1	0										
A	Q=A'																	
0	1																	
1	0																	

S/R Latch Logical Representation

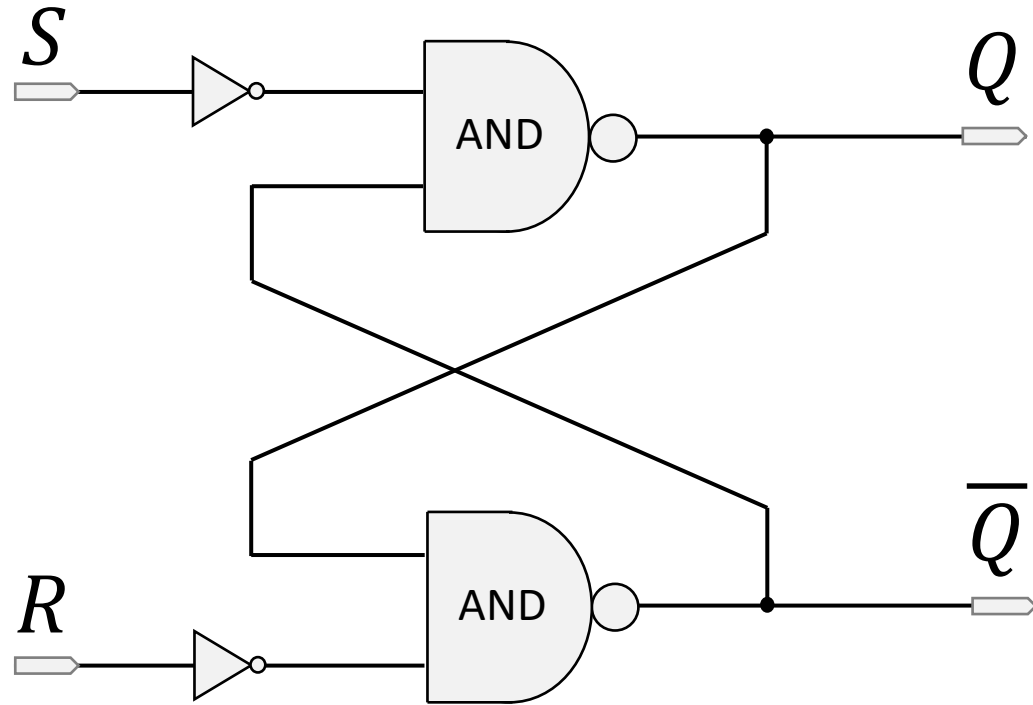


Hardware Implementation

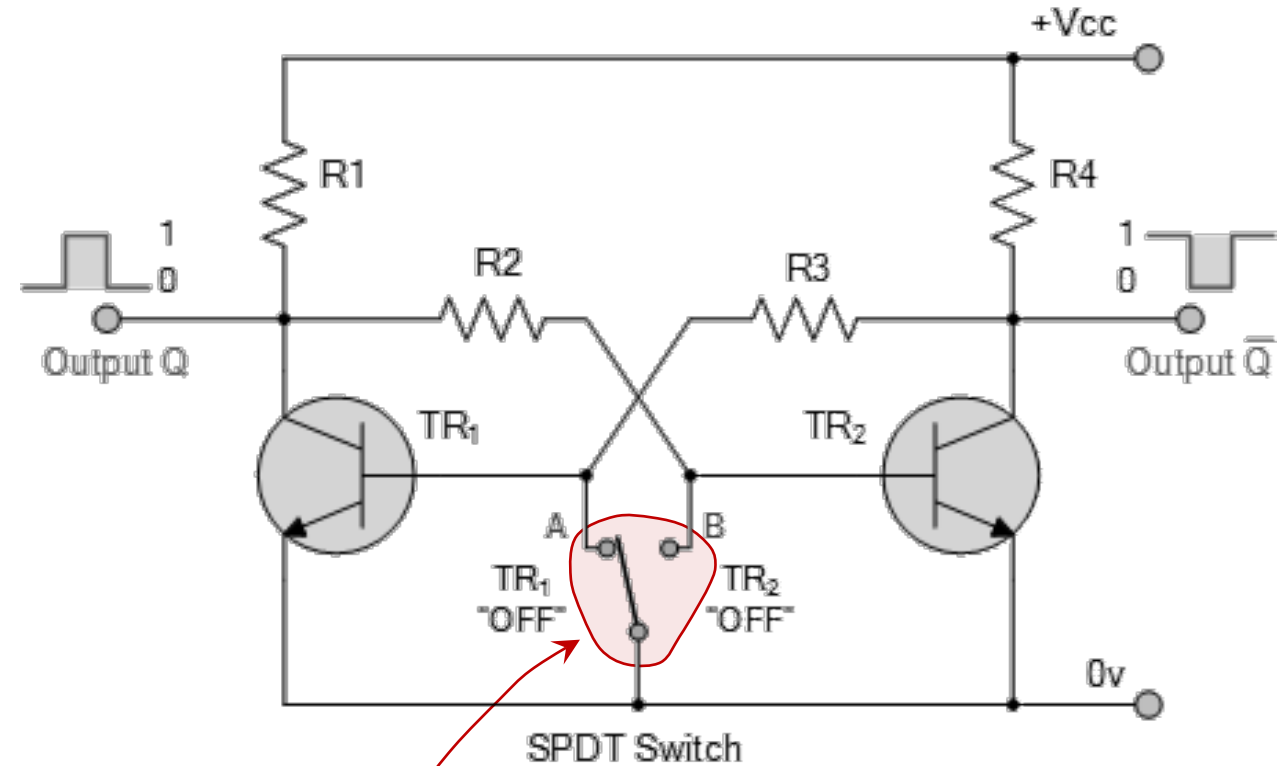


**Before usage, latch is initialized by one of these input combinations:
either $R = 1$ & $S = 0$, or $S = 1$ & $R = 0$;**

S/R Latch Logical Representation

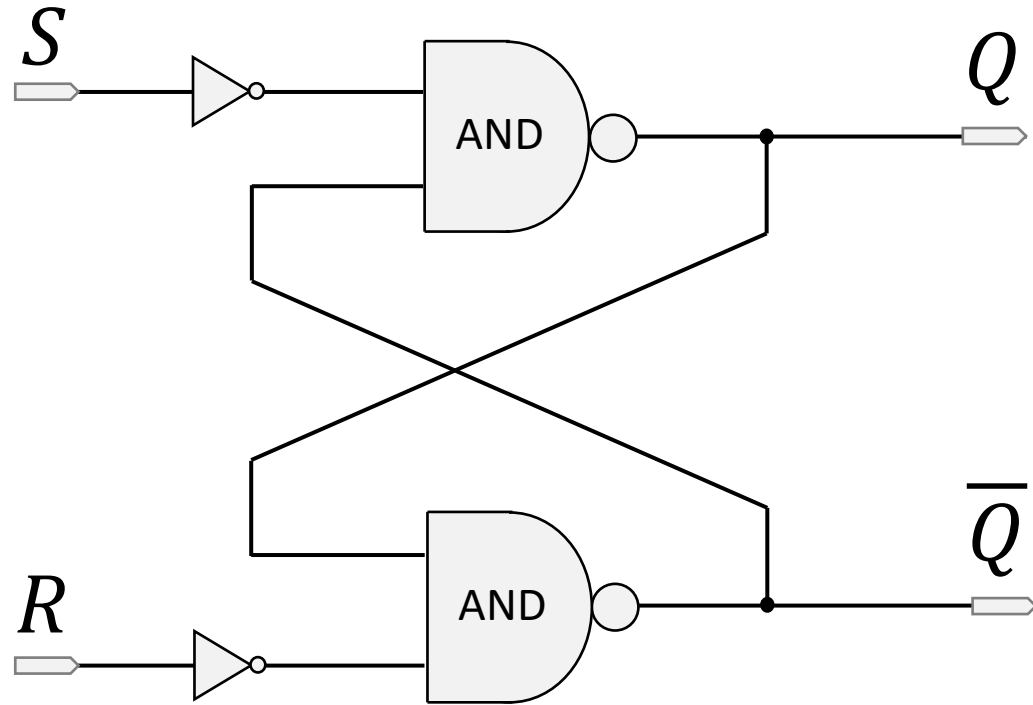


Hardware Implementation



**Switch to control
the input values of S and R
(assumed to have 3 positions)**

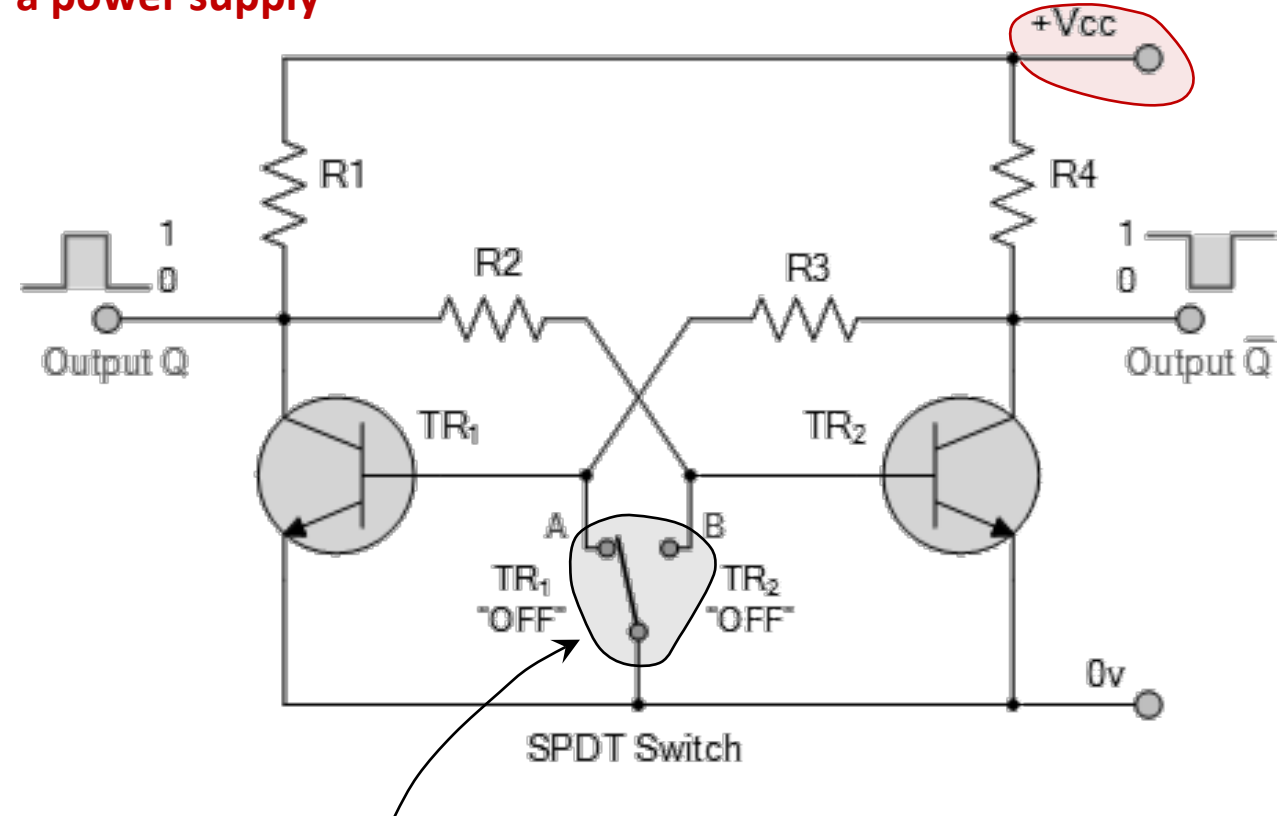
S/R Latch Logical Representation



Hardware Implementation

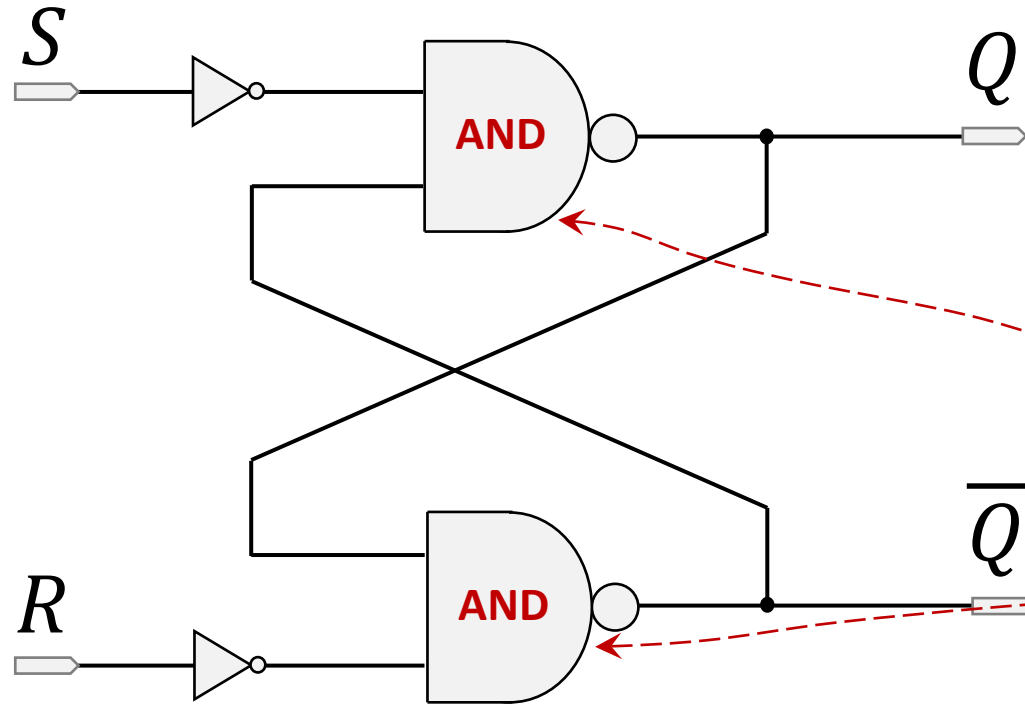
If S and R are "0"s, Q may be "1";
Latch is not an "endless engine", it has
a power supply

There is an external
power supply, to support
latch functioning

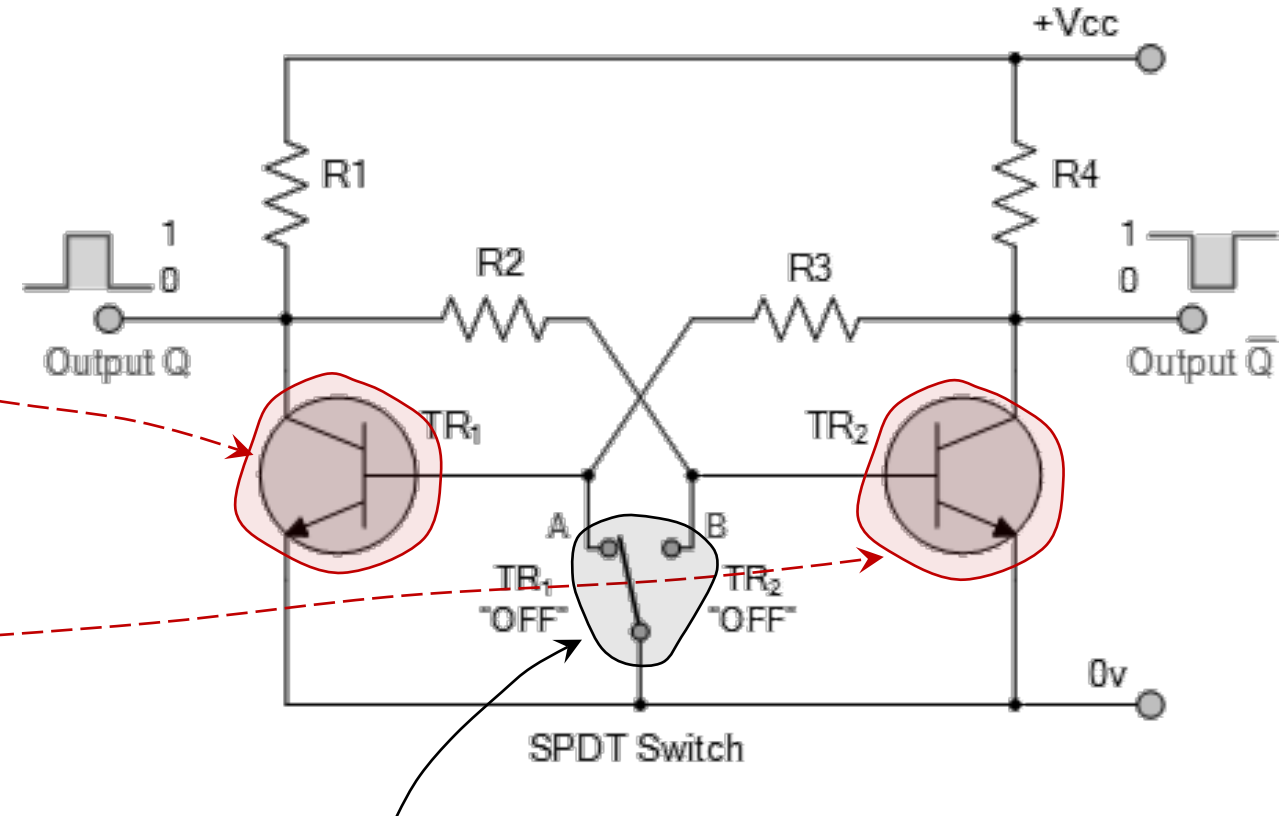


Switch to control
the input values of S and R
(assumed to have 3 positions)

S/R Latch Logical Representation

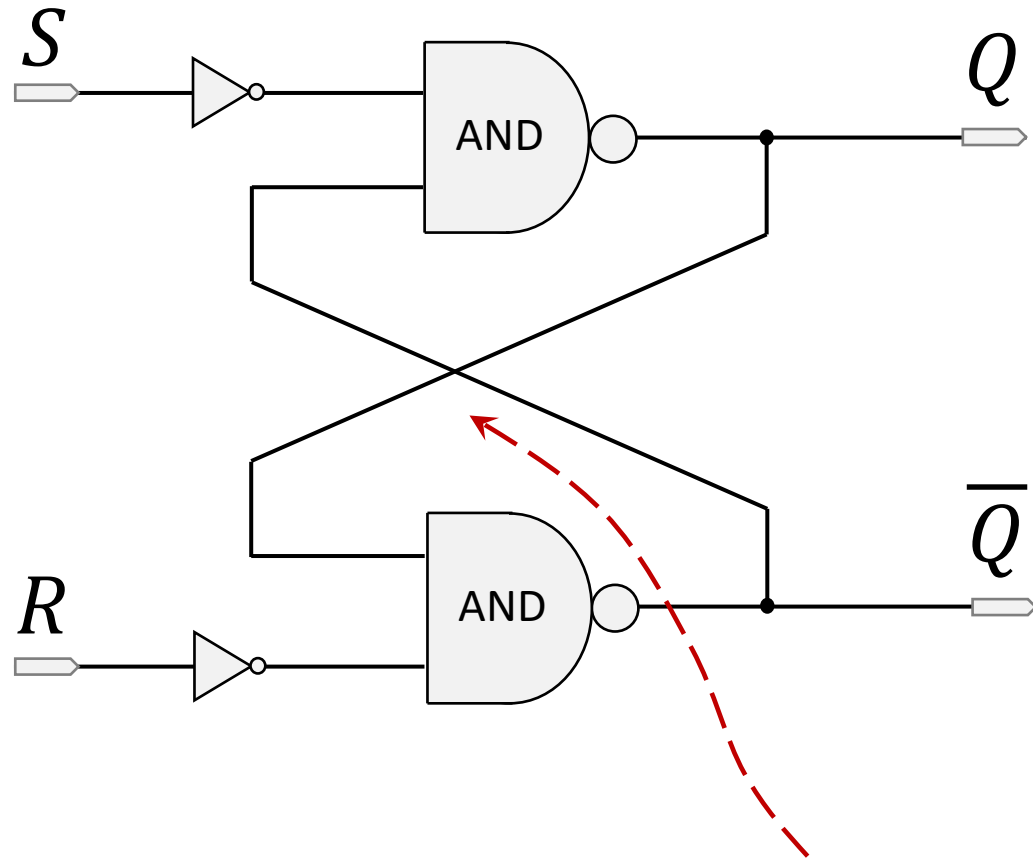


Hardware Implementation

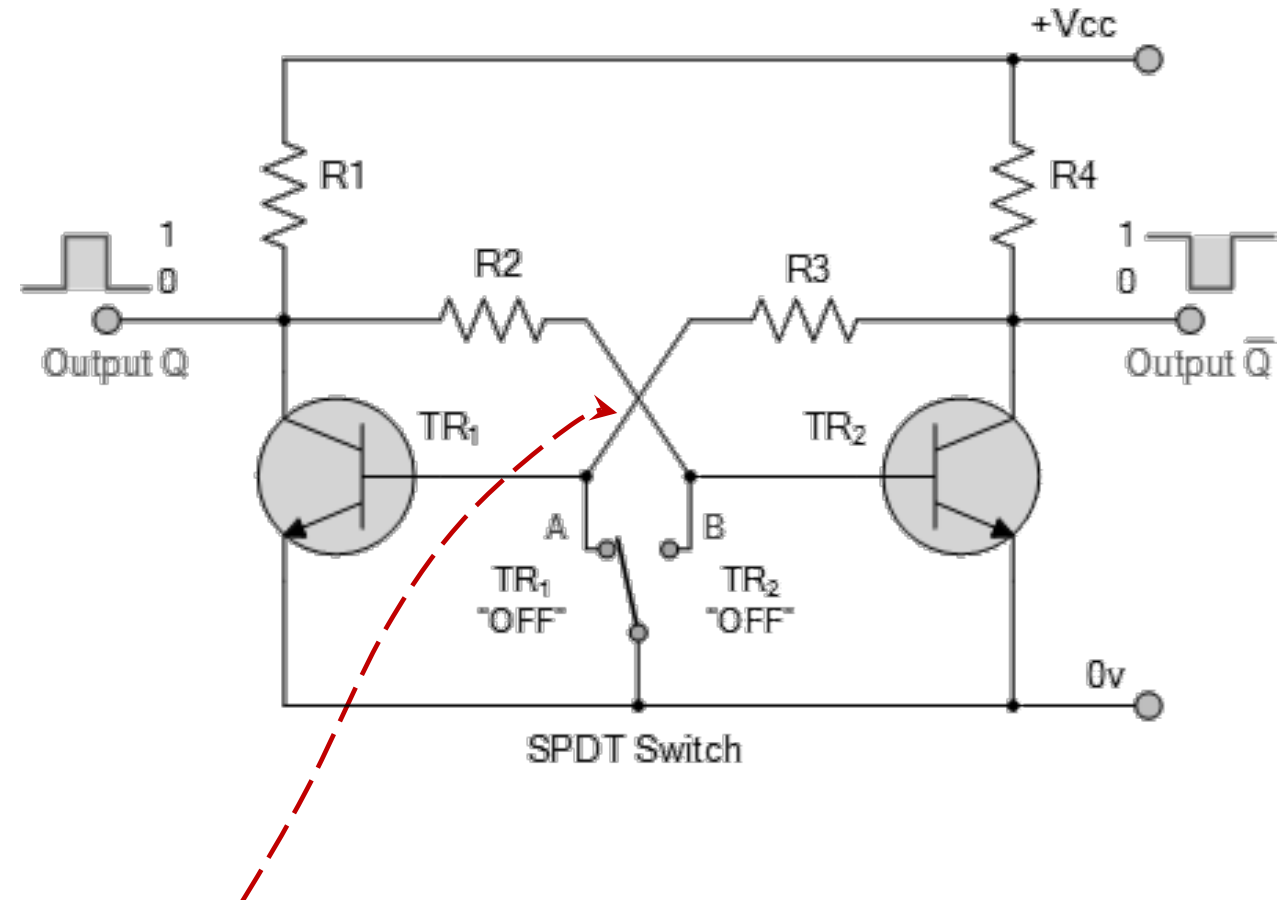


Switch to control
the input values of S and R
(assumed to have 3 positions)

S/R Latch Logical Representation



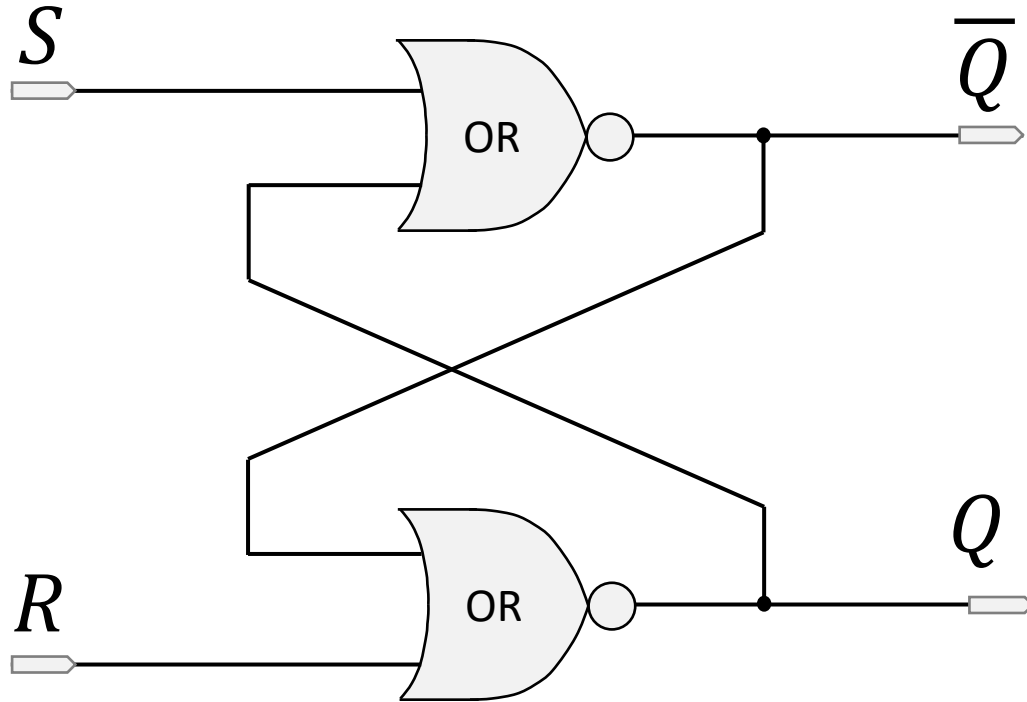
Hardware Implementation



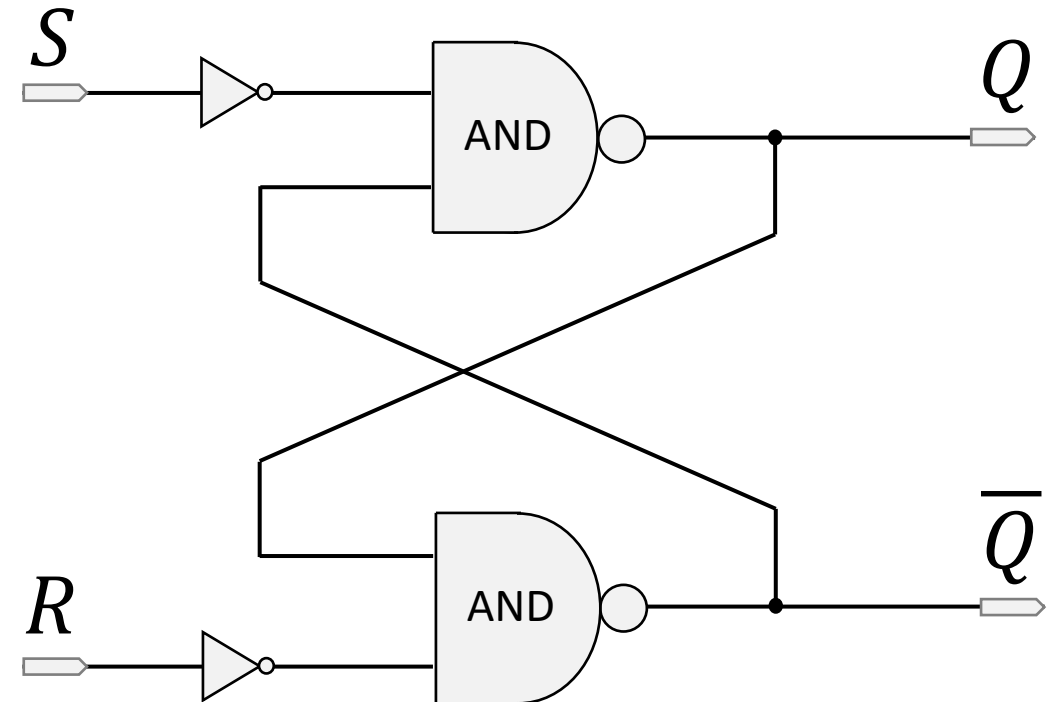
Cross-coupled interconnections

Insignificant Propagation Delay for Set/Reset Latch

1) NORs case:



2) NANDs case:

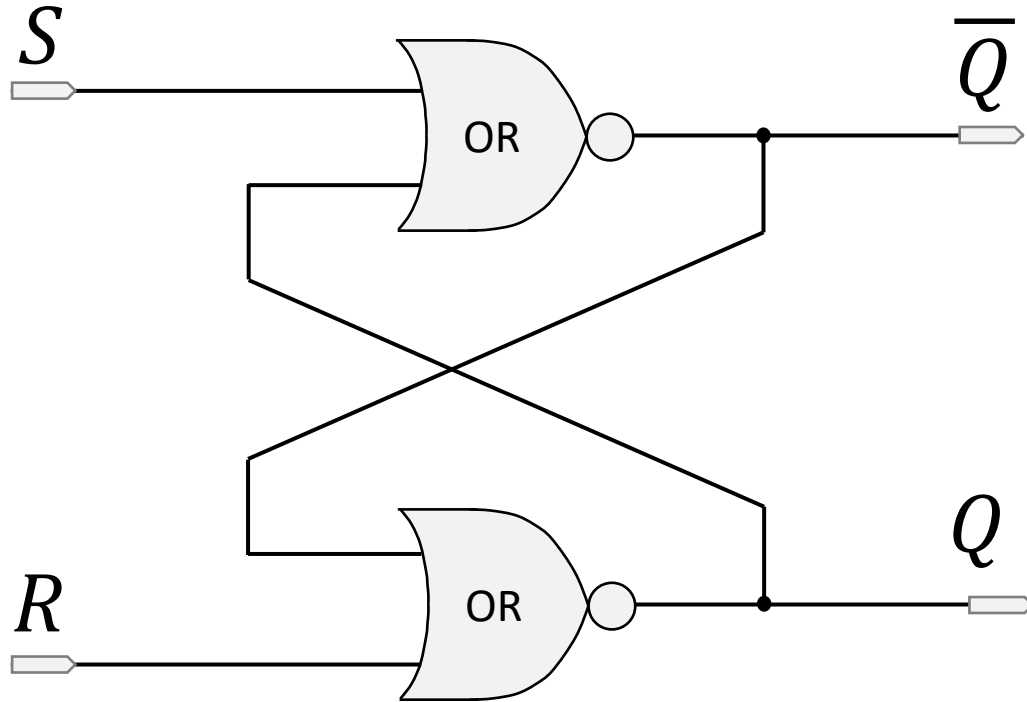


Propagation delay is low:

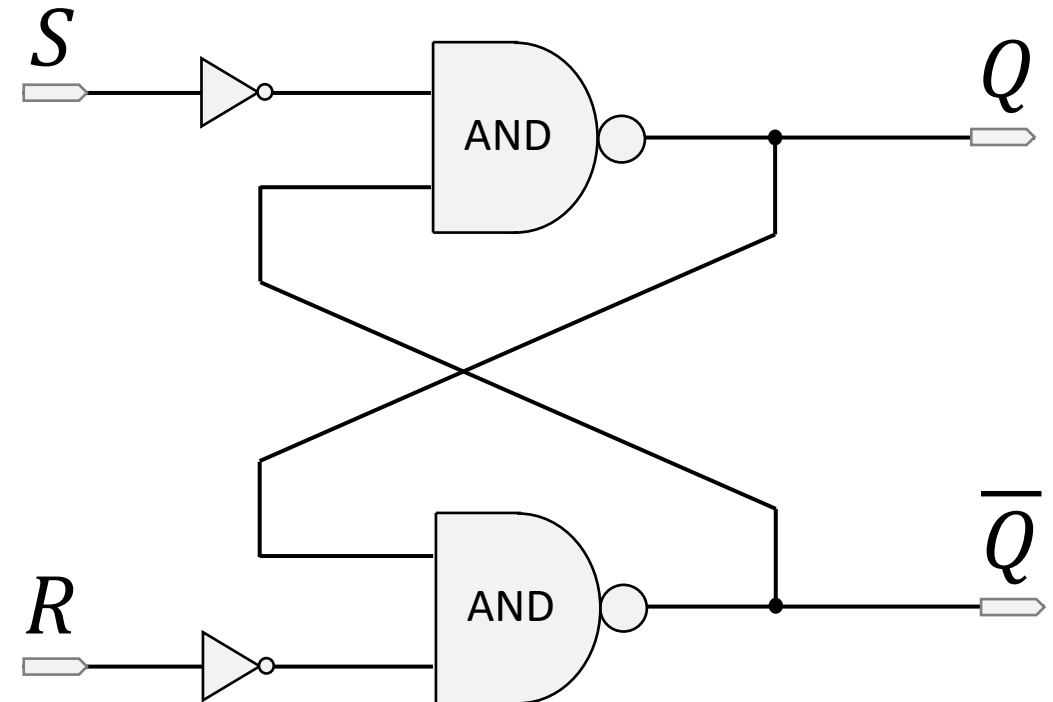
Output Q is updated almost immediately after the update of inputs

Insignificant Propagation Delay for Set/Reset Latch

1) NORs case:



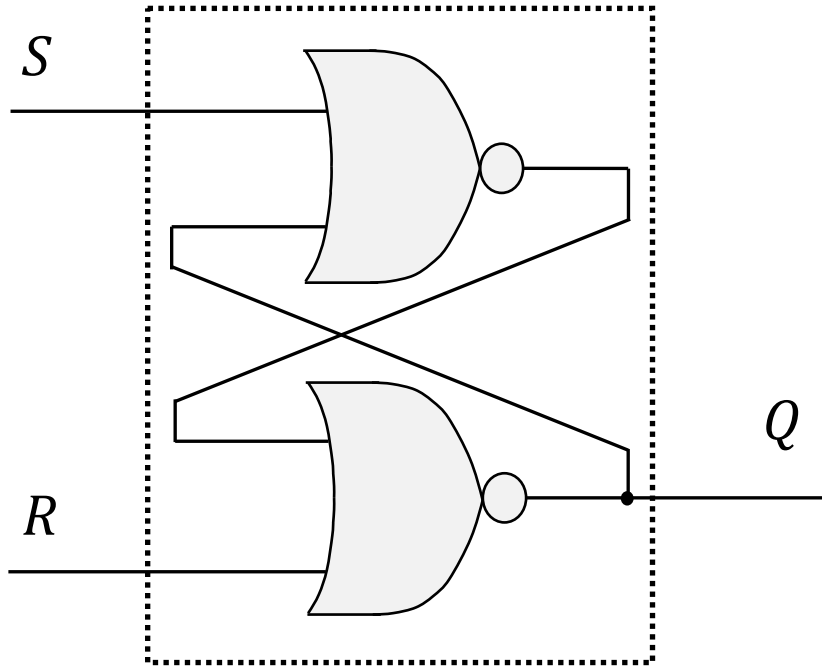
2) NANDs case:



Propagation delay is low

Set/Reset Latch is also called “transparent”:
Q is updated transparently (without a delay)

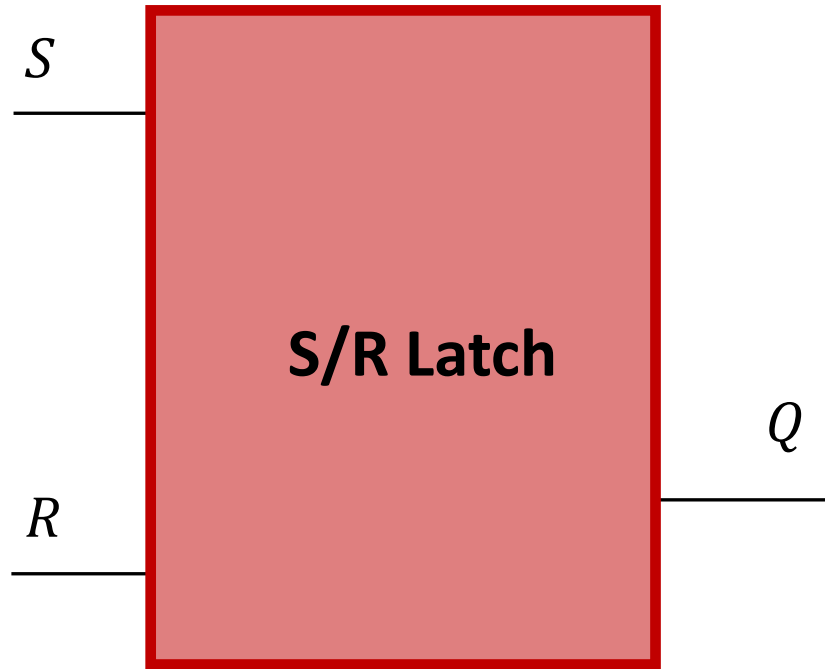
Transparent Set/Reset Latch: Behavior Table



Transparent S/R Latch

S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal inputs

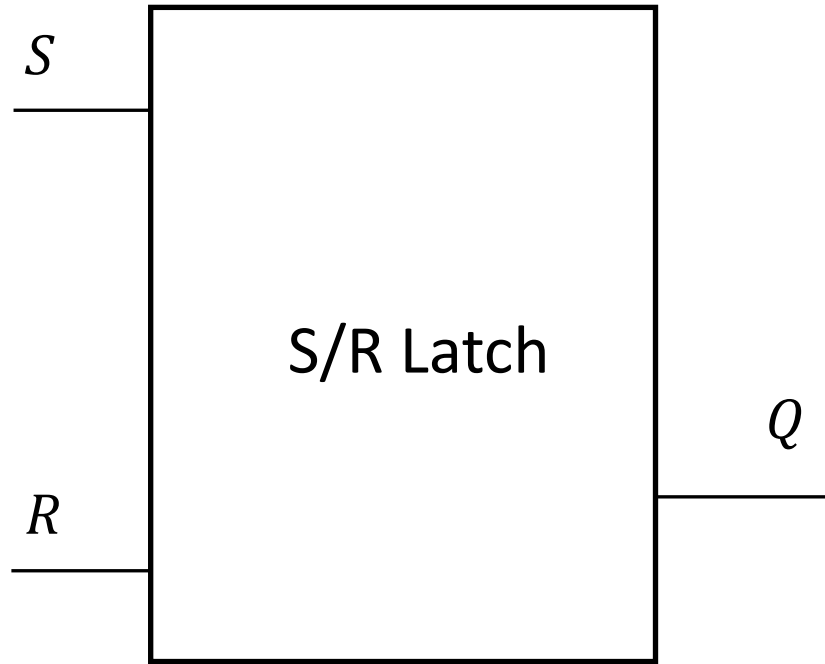
S/R Latch: Symbolic Representation



Transparent S/R Latch

S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal inputs

S/R Latch: Illegal Inputs Combination

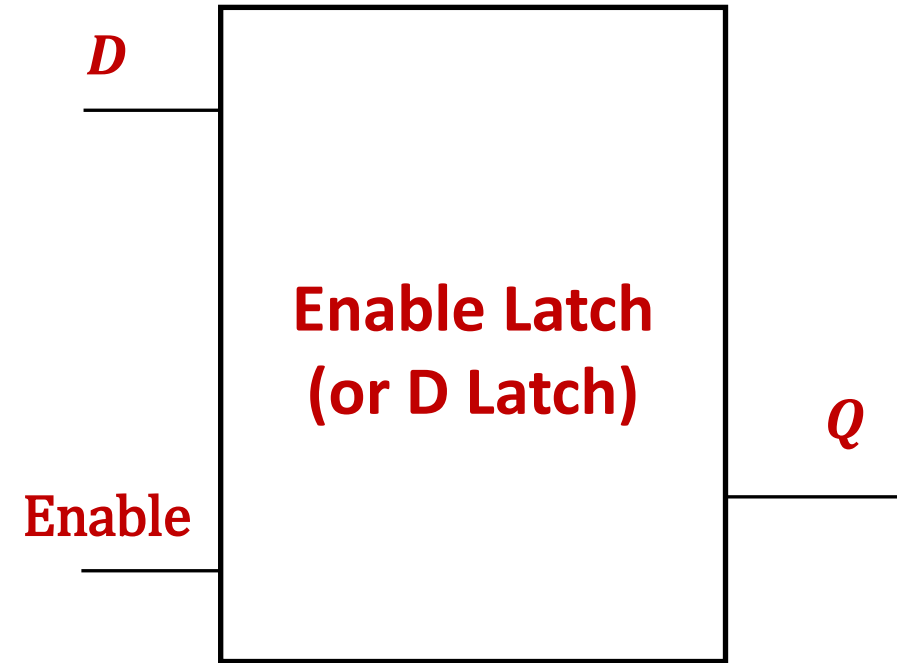
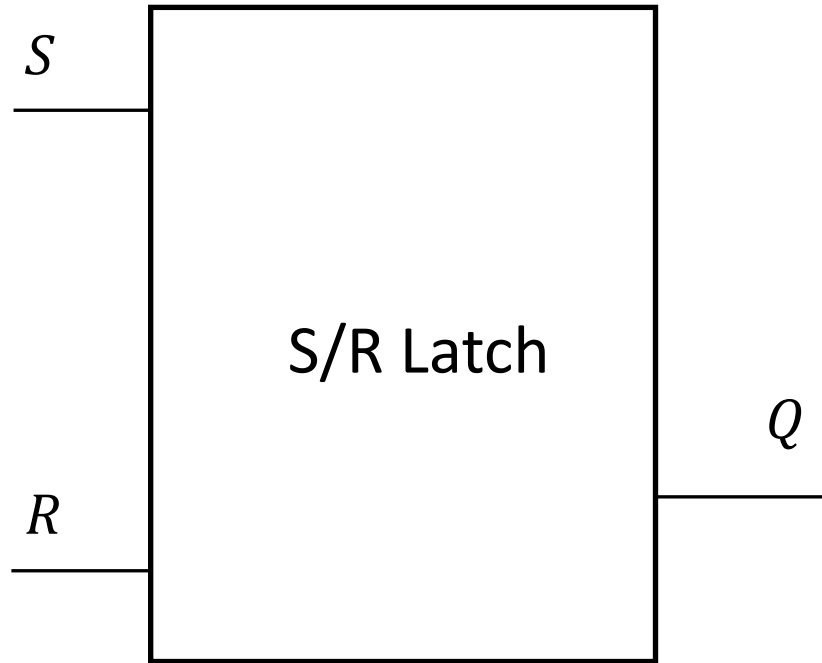


Transparent S/R Latch

S	R	Q
1	0	1
0	1	0
0	0	Q^{prev}
1	1	Illegal inputs

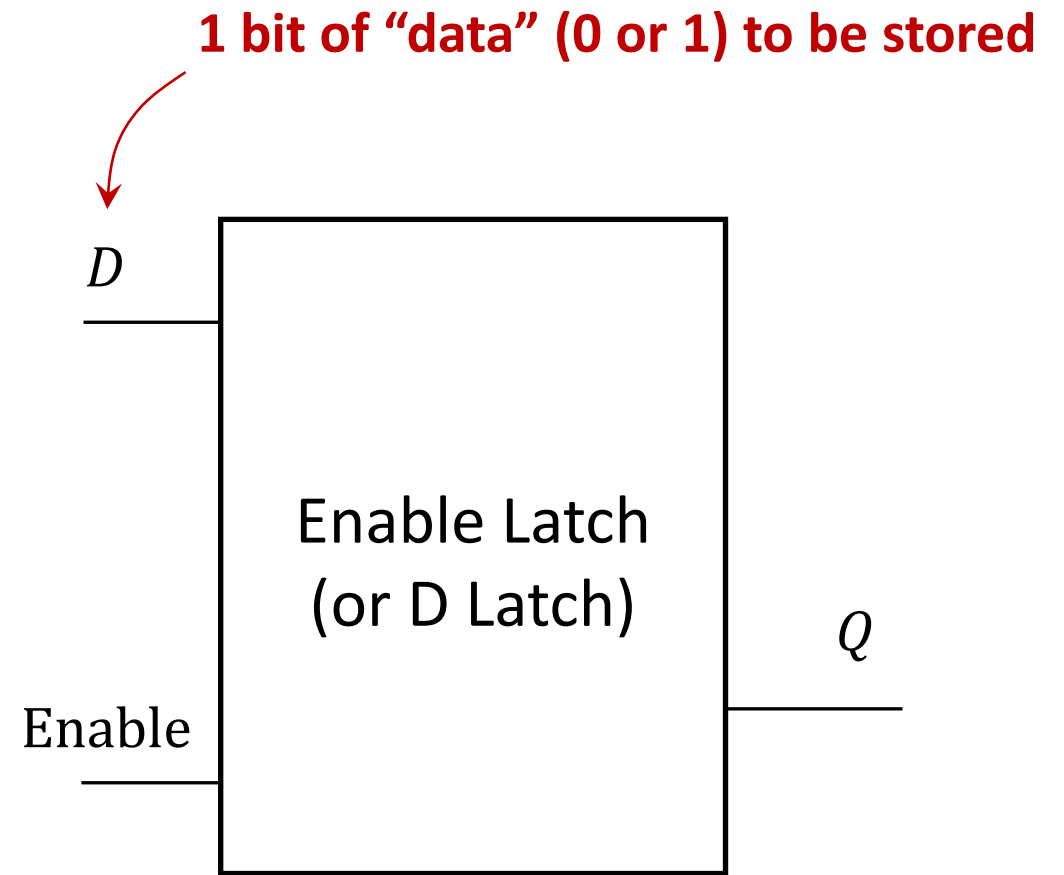
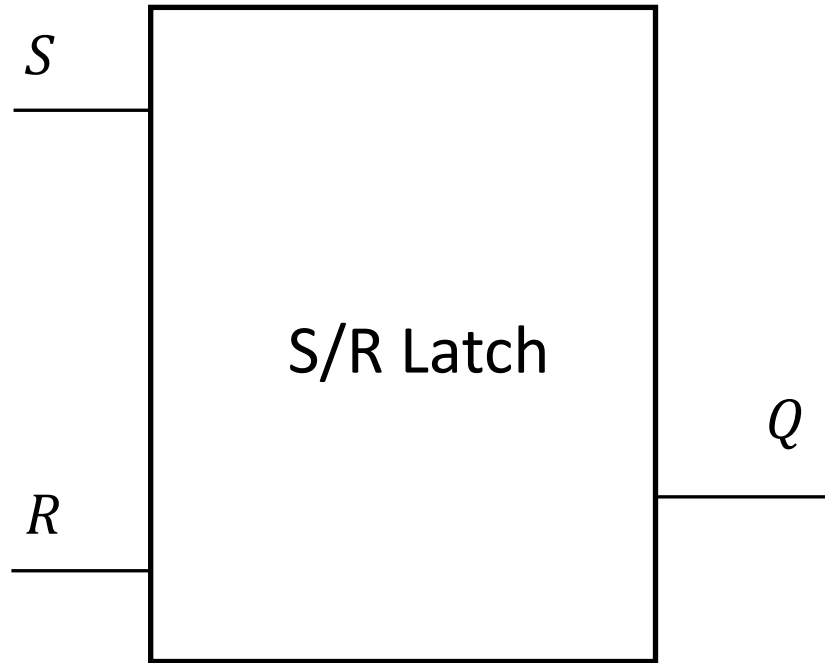
Enable Latch (or D Latch)

Solves illegal inputs problem



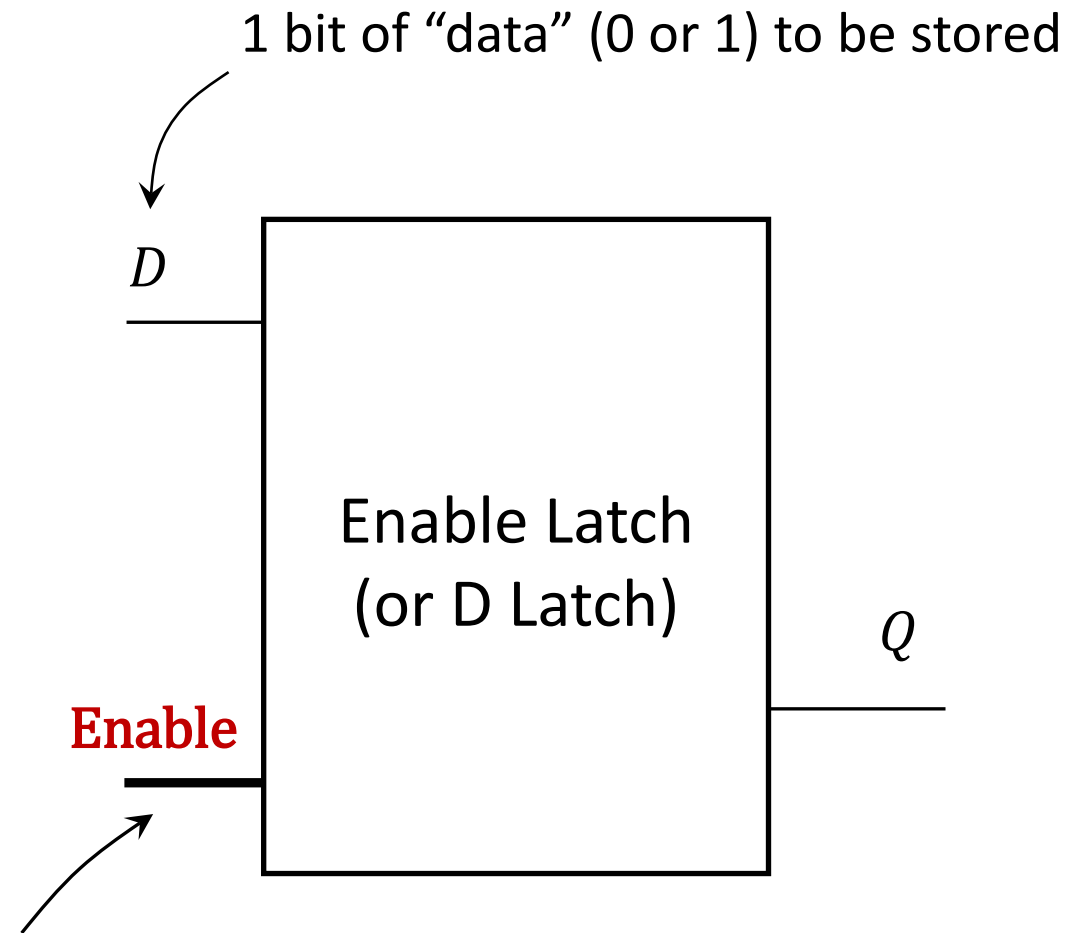
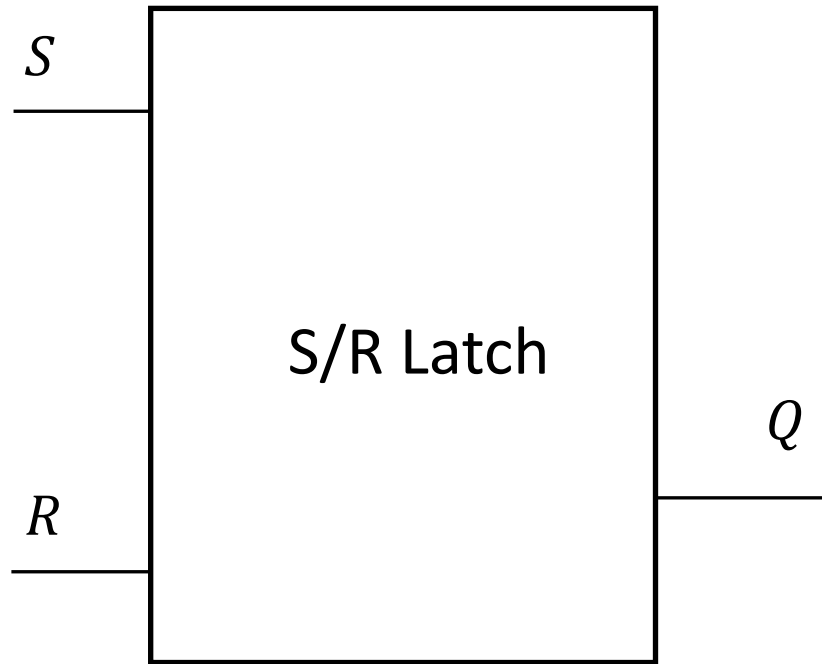
Enable Latch (or D Latch)

Solves illegal inputs problem



Enable Latch (or D Latch)

Solves illegal inputs problem

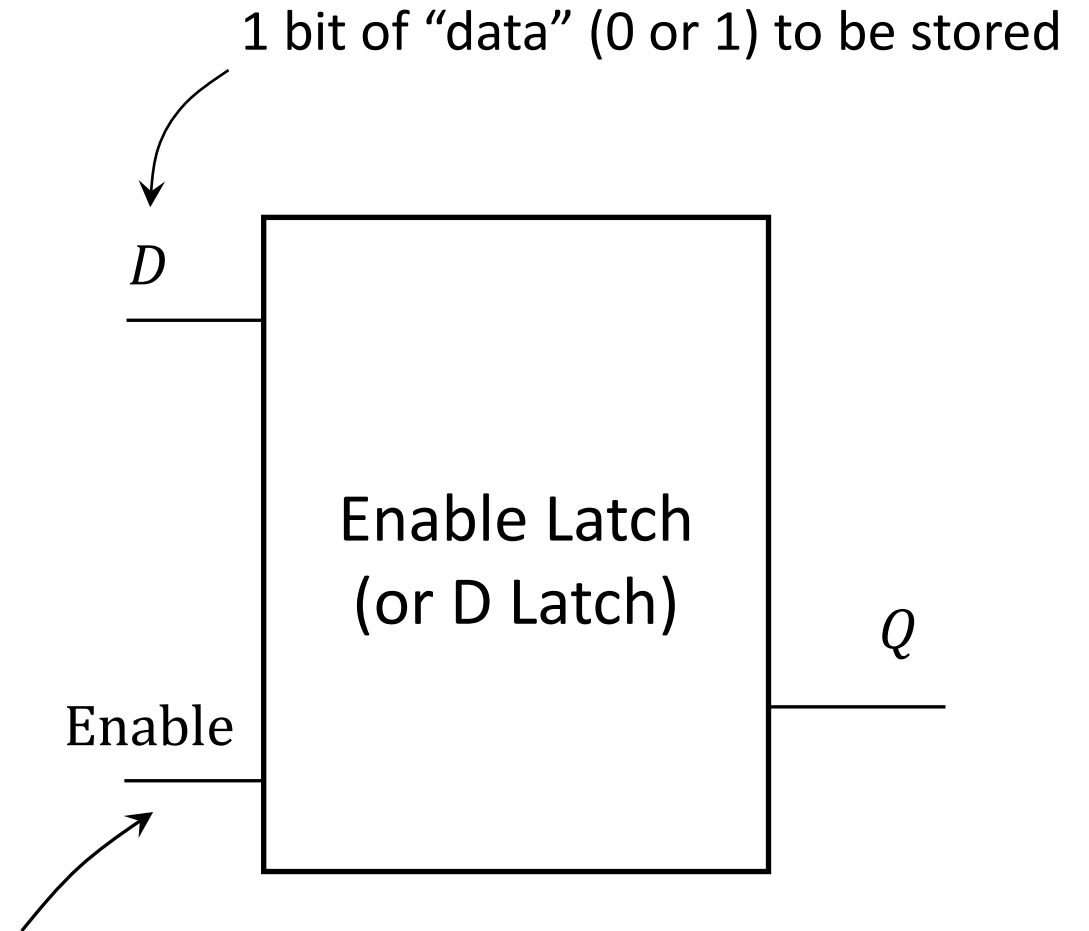
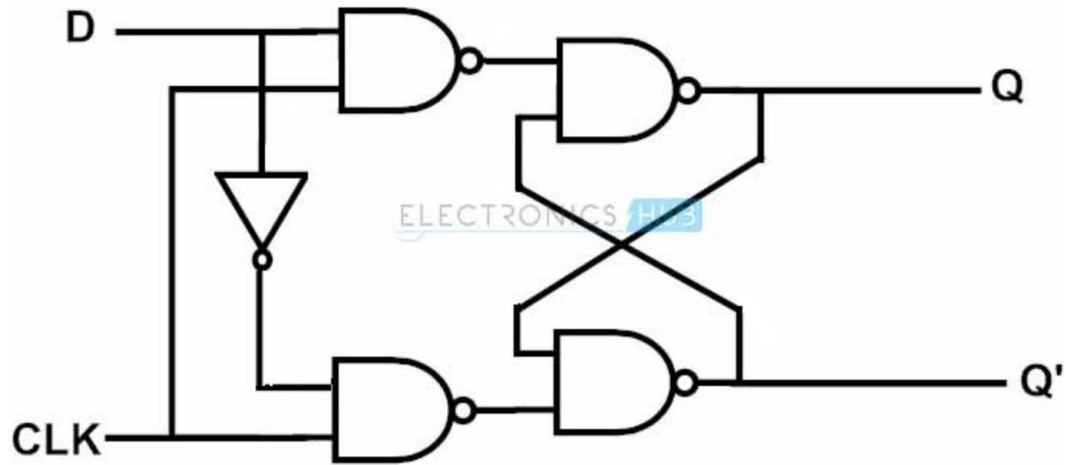


Enables data storage:

- if "1": D is stored;
- If "0": D is ignored

Enable Latch (or D Latch)

**Enable Latch implementation;
clock serves as "Enable" pin**



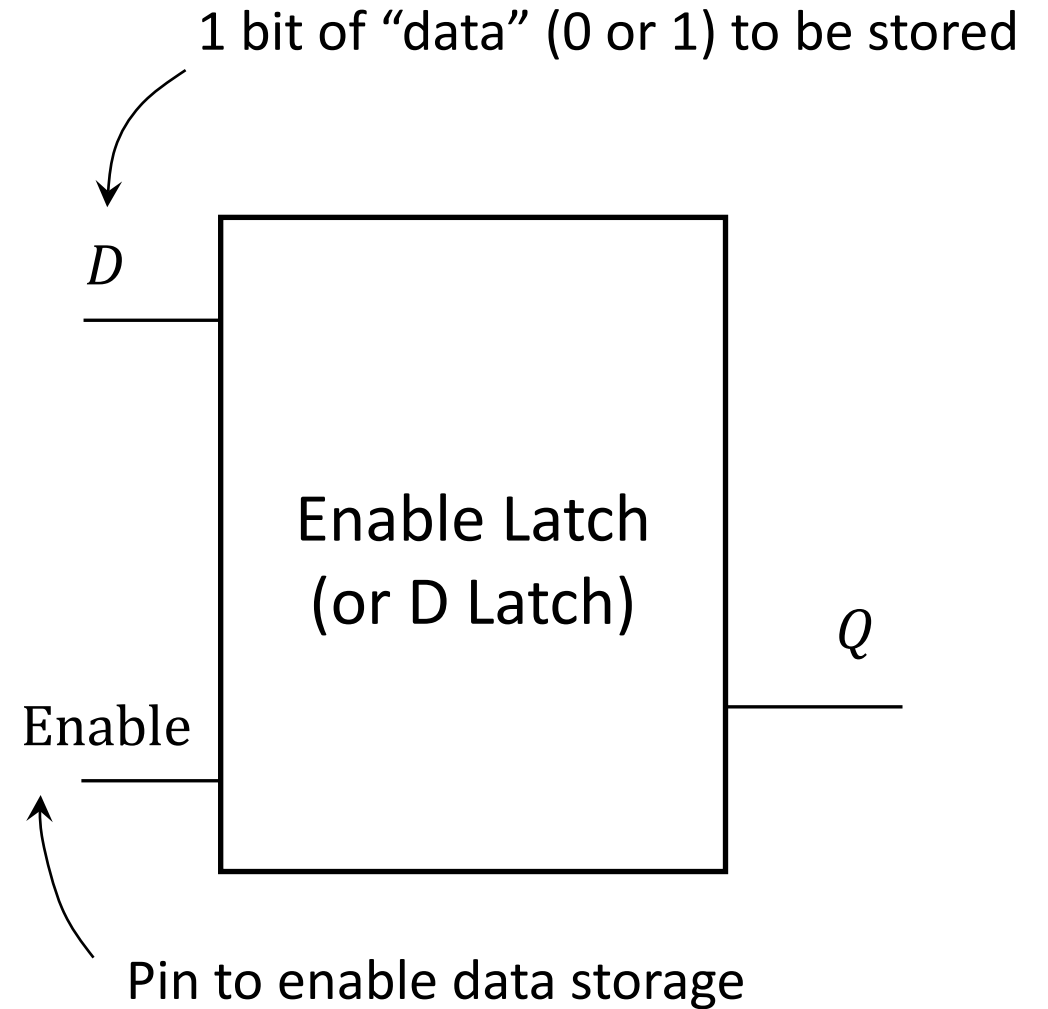
Enables data storage:

- if "1": D is stored;
- If "0": D is ignored

Enable Latch (or D Latch)

Advantage:

- No need to care about an invalid input combination (like $S=1, R=1$ is prohibited for S/R latch);
- All input combinations are allowed
- Simpler usage



Enable Latch (or D Latch)

Advantage:

- No need to care about an invalid input combination (like $S=1, R=1$ is prohibited for S/R latch);
- All input combinations are allowed
- Simpler usage

Data is stored by latch, until a different value “D” arrives, at a time when pin “Enable” is activated

