CSEE 3827: Fundamentals of Computer Systems, Spring 2022

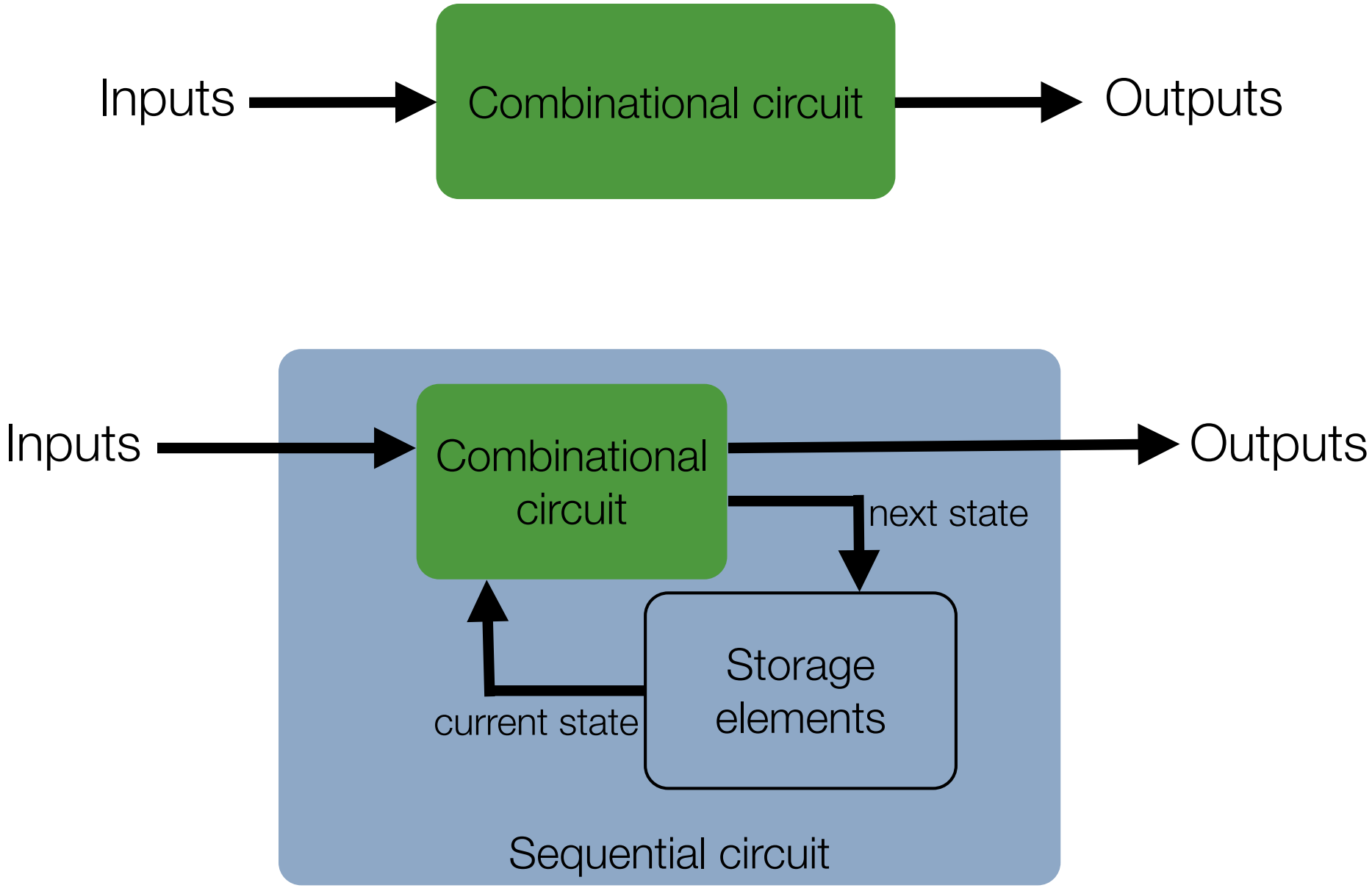
Lecture 5

Prof. Dan Rubenstein (danr@cs.columbia.edu)

Agenda (M&K 5.1-5.3, 5.6)

- Sequential Circuit Design
  - Latches
  - Flip-Flops
  - Timing Issues

Combinational v. sequential logic

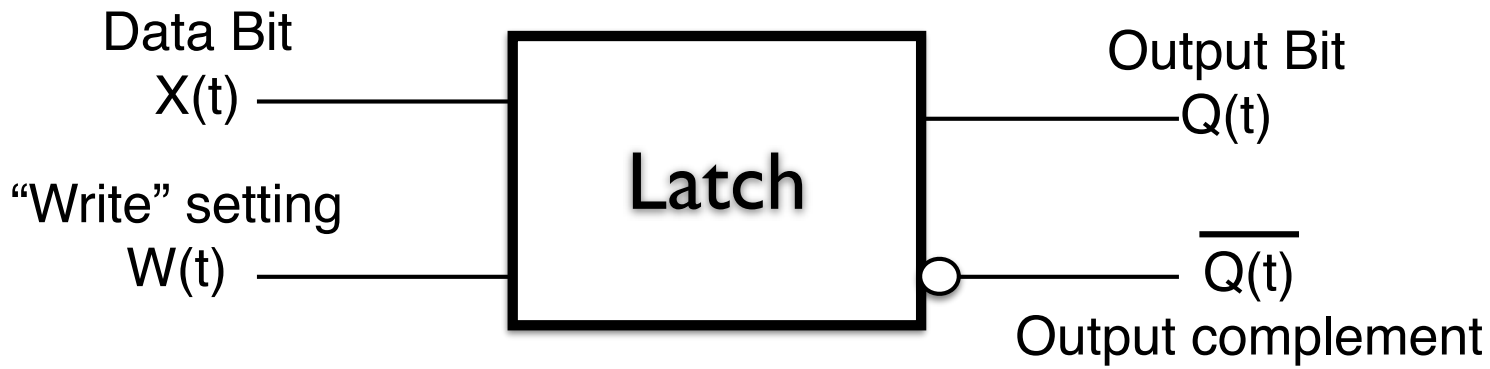


Latches

State of a circuit

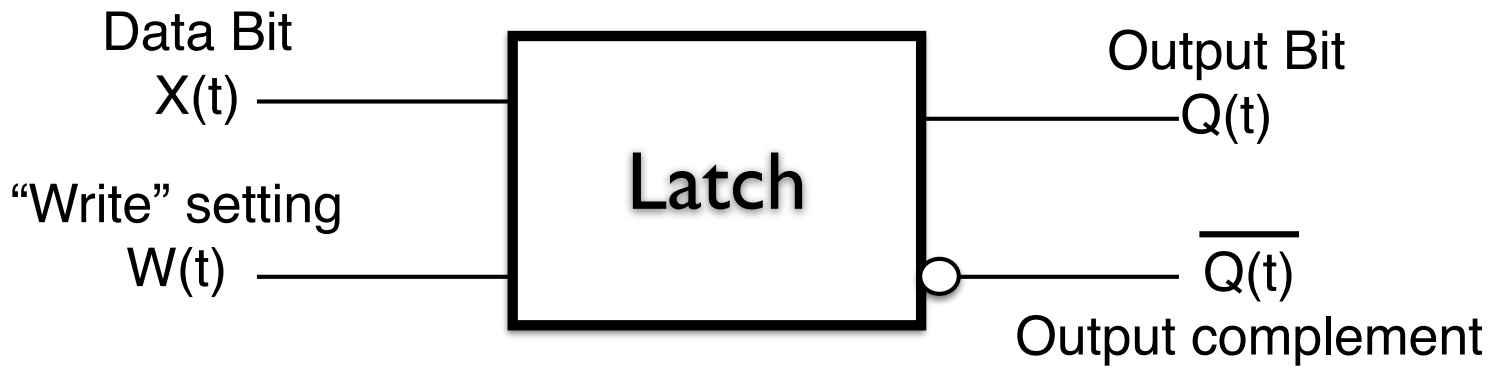
- Q: What is the simplest circuit with state?
- A: Hold a single bit of state where
  - can set the state to a new value (0 or 1)
  - hold the state to the current value

Intuitive Latch Design

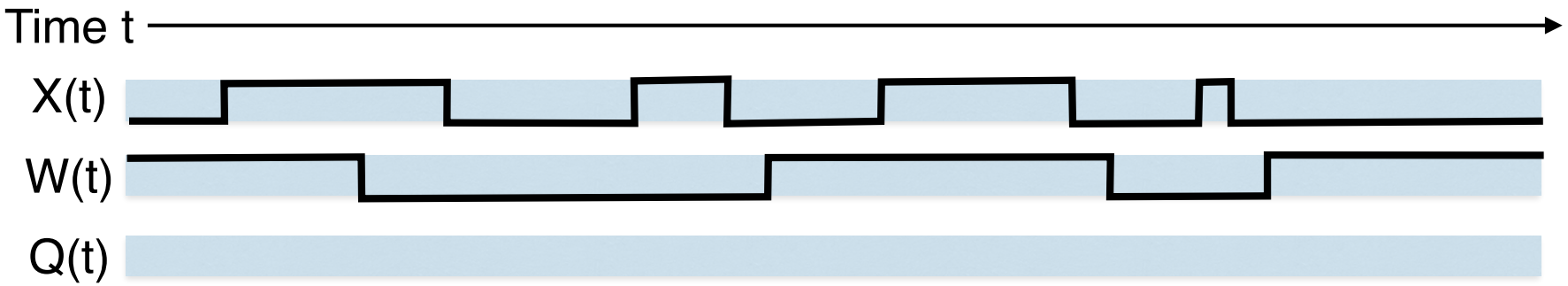


- W(t) = 1, write X(t) onto the (1-bit) memory of the latch
- W(t) = 0, ignore X(t), hold the latch memory fixed
- Q(t) = X(s) where s is the last time (before t) where W(s)=0
  - Latch is **holding** its value when W=0

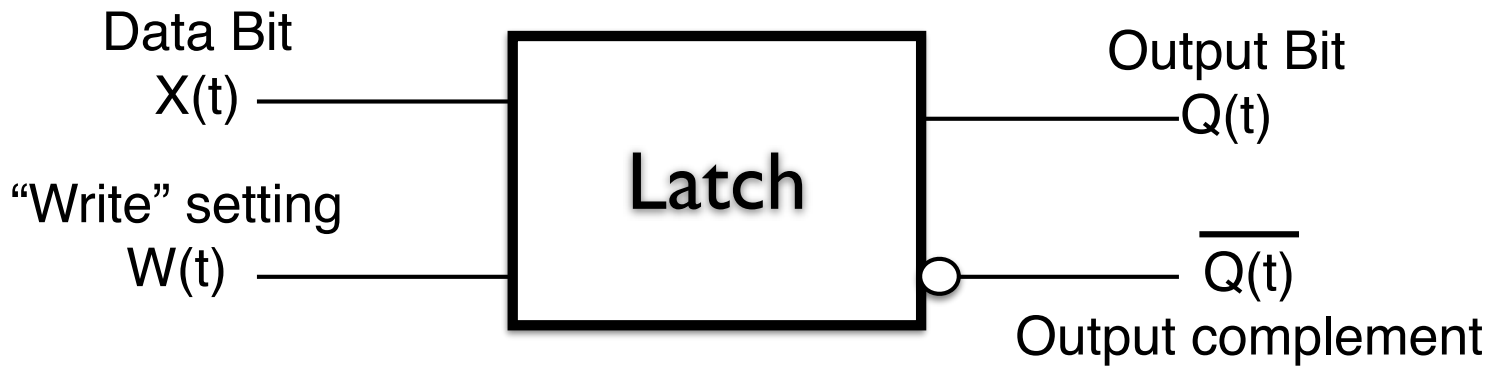
Intuitive Latch Design



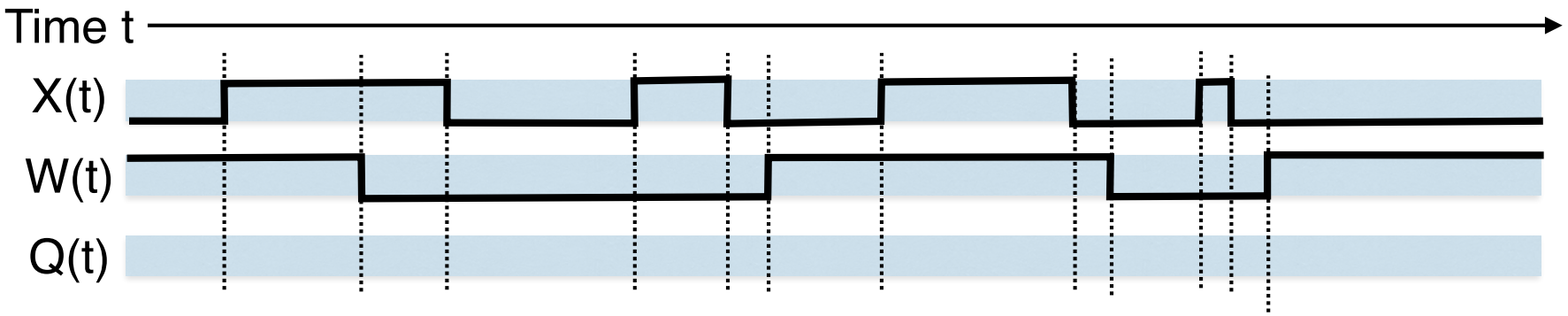
- W(t) = 1, write X(t) onto the (1-bit) memory of the latch
- W(t) = 0, ignore X(t), hold the latch memory fixed
- Q(t) = X(s) where s is the last time (before t) where W(s)=0
  - Latch is **holding** its value when W=0



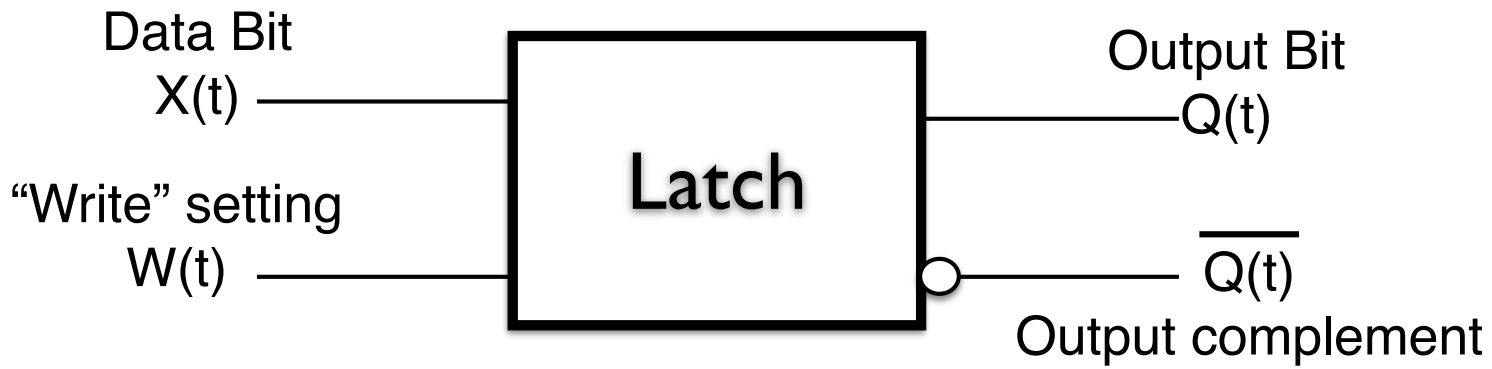
Intuitive Latch Design



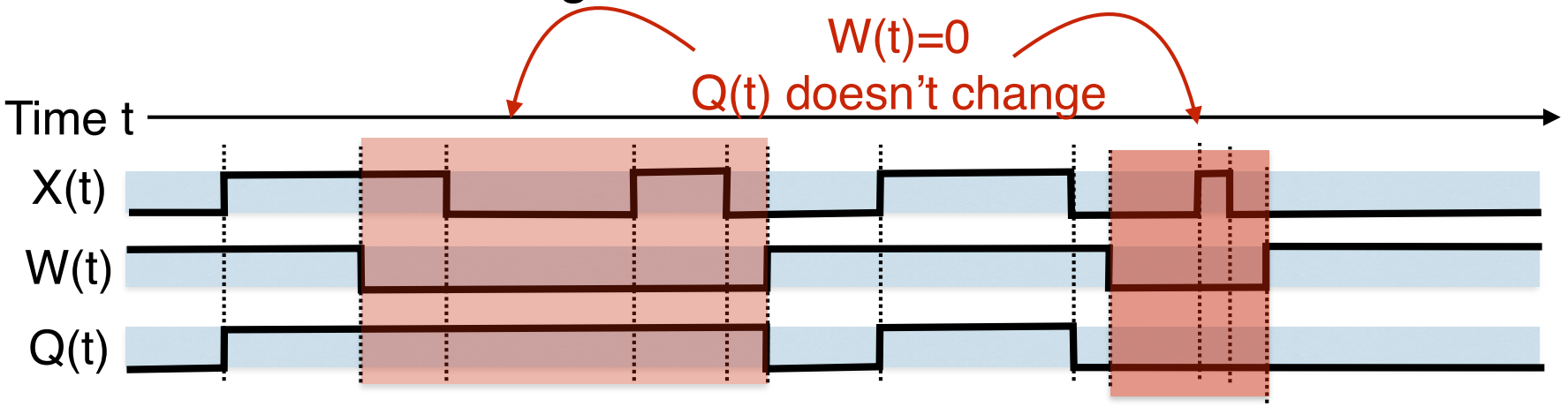
- W(t) = 1, write X(t) onto the (1-bit) memory of the latch
- W(t) = 0, ignore X(t), hold the latch memory fixed
- Q(t) = X(s) where s is the last time (before t) where W(s)=0
  - Latch is **holding** its value when W=0



Intuitive Latch Design



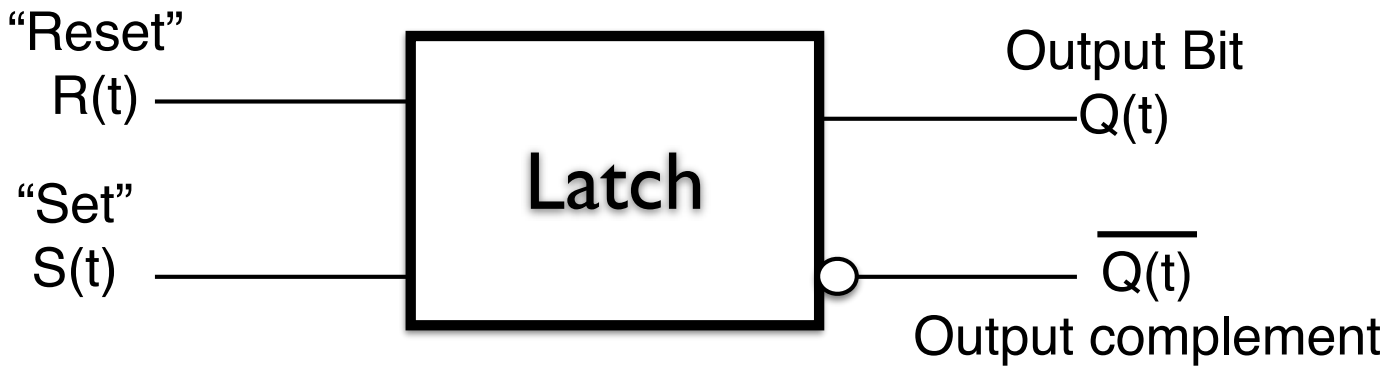
- W(t) = 1, write X(t) onto the (1-bit) memory of the latch
- W(t) = 0, ignore X(t), hold the latch memory fixed
- Q(t) = X(s) where s is the last time (before t) where W(s)=0
  - Latch is **holding** its value when W=0



Constructing Latches

- Important points:
  - Can store and hold a bit
  - Can be set to a new value or told to hold current value
  - Inputs to latch might be formulated differently, but same concept

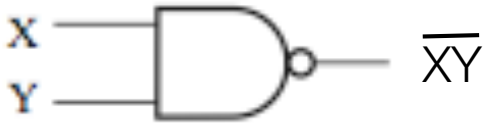
1st Latch We build…



- R=0, S=1, write 1 into the latch
- R=1, S=0, write 0 into the latch
- R=S=0, latch "holds" its value
- R=S=1: do not use!!!

RECALL: Universal gates: NAND, NOR

x	y	$z = \overline{xy}$
0	0	1
0	1	1
1	0	1
1	1	0



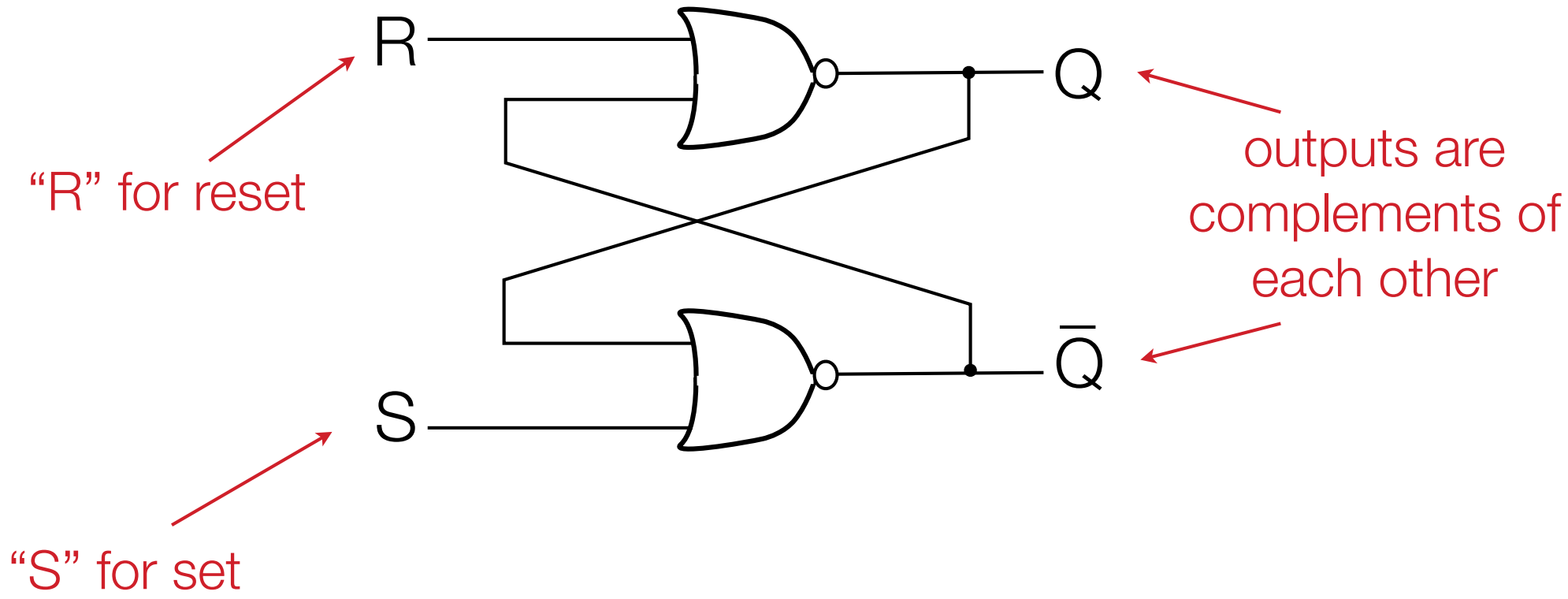
Note: the "o" in a circuit represents a NOT (inverter)

x	y	$z = \overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

Different from "●" which represents wire connector

SR latch

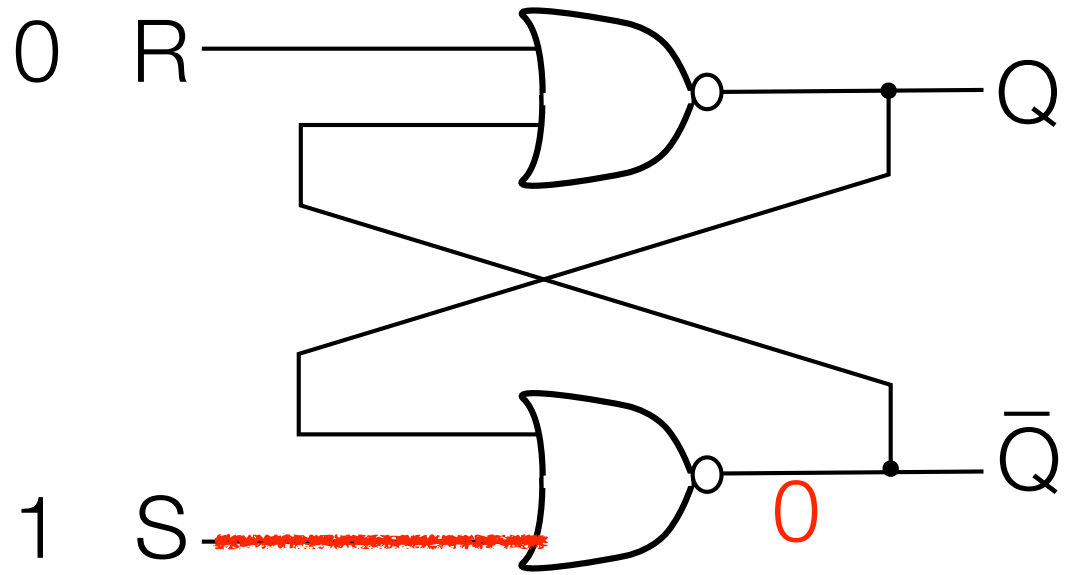
- Latch constructed of cross-coupled NOR gates



- What's so new? The wires "loop back" (output feeding back into circuit)

- Note: symmetrical around X axis (swap values for R & S and Q & $\overline{Q}$ swap values)

SR latch: set example



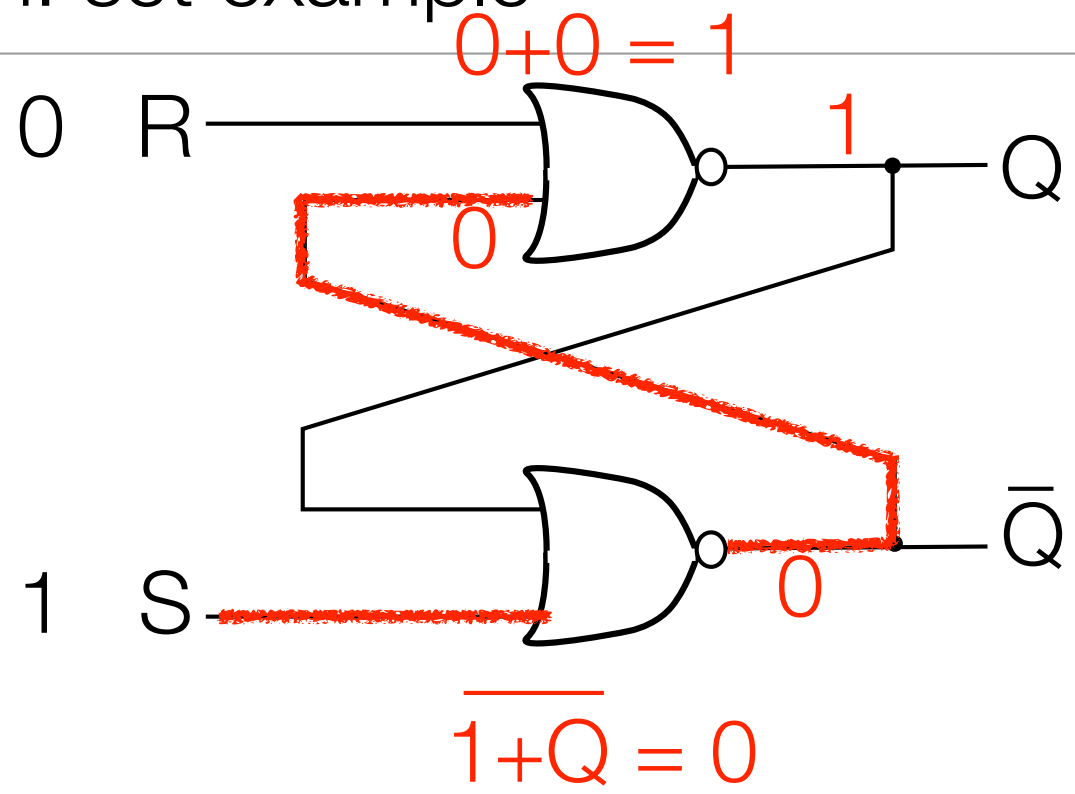
$\overline{1+Q} = 0$

R	S	Q	$\overline{Q}$
0	0		
0	1		
1	0		
1	1		

NOR

x	y	$z = \overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

SR latch: set example



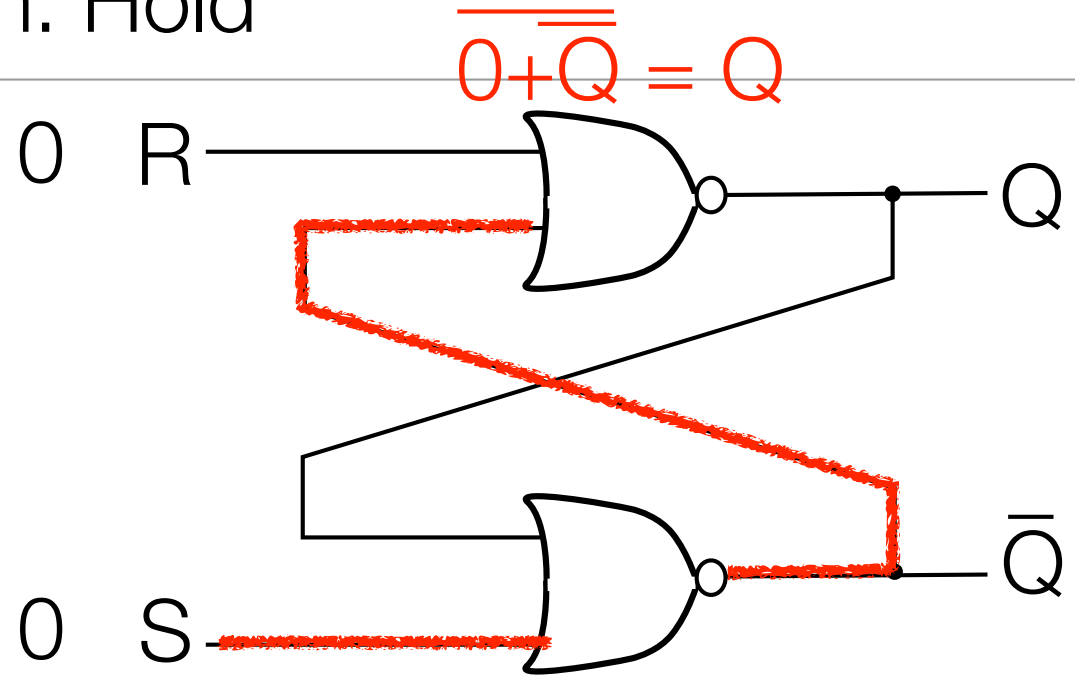
R	S	Q	$\overline{Q}$
0	0		
0	1	1	0
1	0	0	1
1	1		

By symmetry along horizontal cut of latch

NOR

x	y	$z = \overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

SR latch: Hold



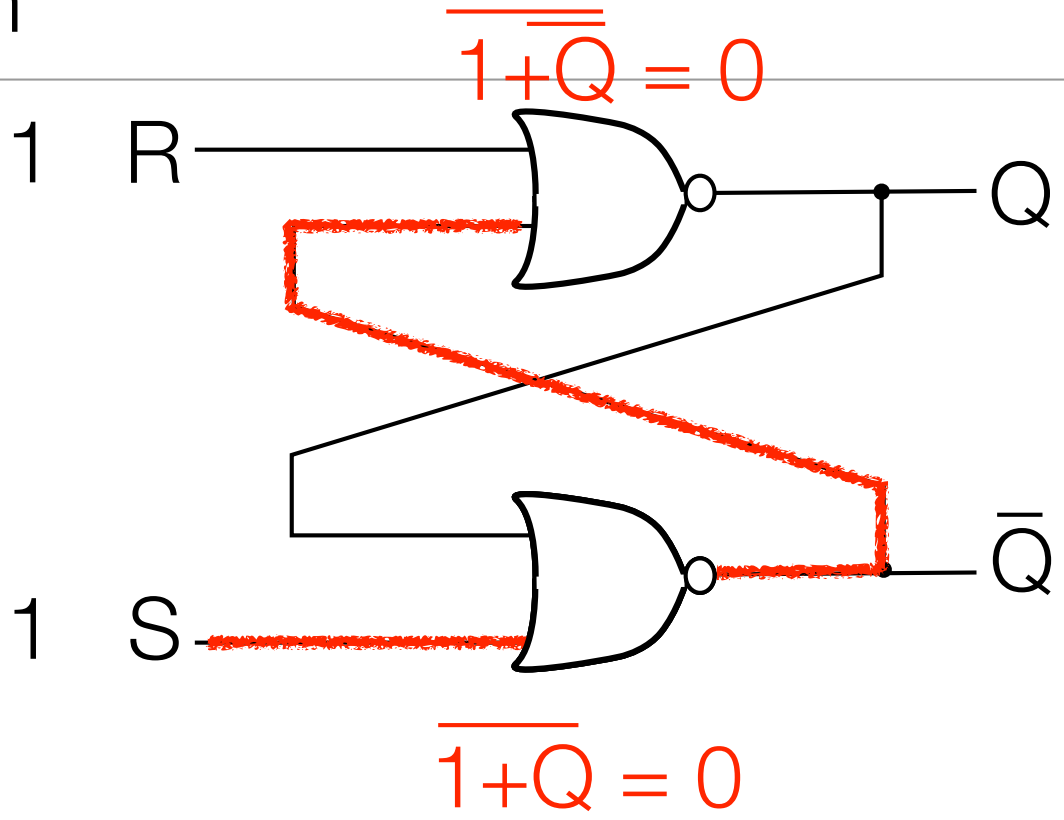
R	S	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	1	0
1	0	0	1
1	1		

Hold previous value

NOR

x	y	$z = \overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0

SR latch

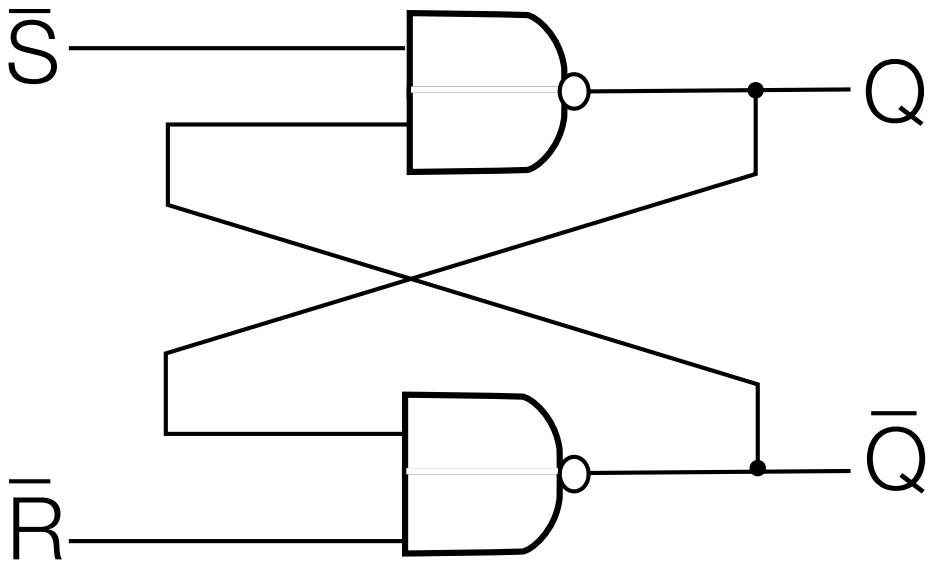


R	S	Q	$\overline{Q}$	
0	0	Q	$\overline{Q}$	No change
0	1	1	0	Set (Q=1)
1	0	0	1	Reset (Q=0)
1	1	0	0	Bad - do not use!

Other Types of Latches

$\bar{S}\bar{R}$ latch

- Latch constructed of cross-coupled **NAND** gates



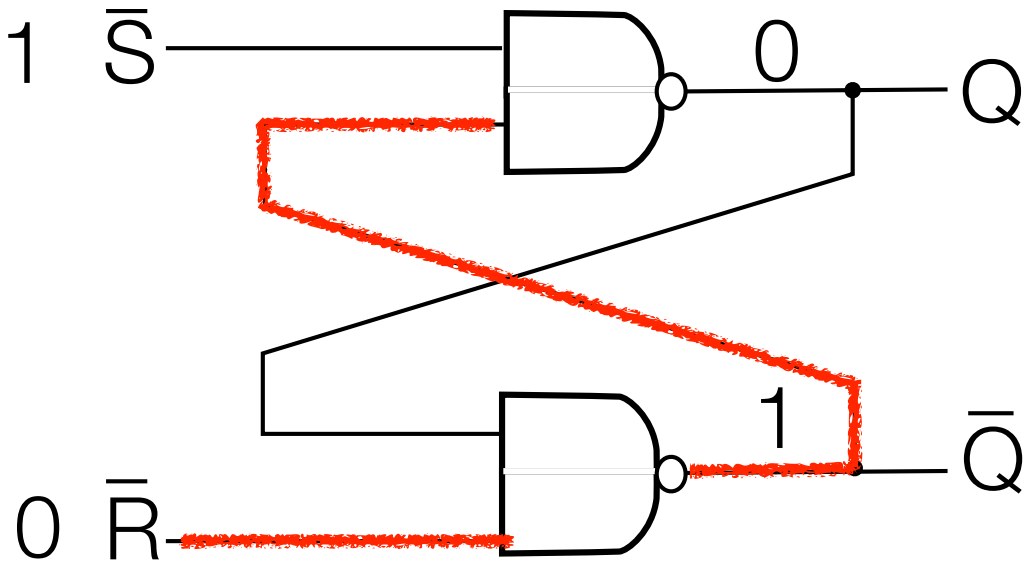
$\bar{R}$	$\bar{S}$	Q	$\bar{Q}$
0	0	1	1
0	1	0	1
1	0	1	0
1	1	Q	$\bar{Q}$

$\bar{R}$ and $\bar{S}$ have reverse behavior of SR latch, so R and S have same behaviors

Bad - do not use!

$\bar{S}\bar{R}$ latch

- Latch constructed of cross-coupled **NAND** gates



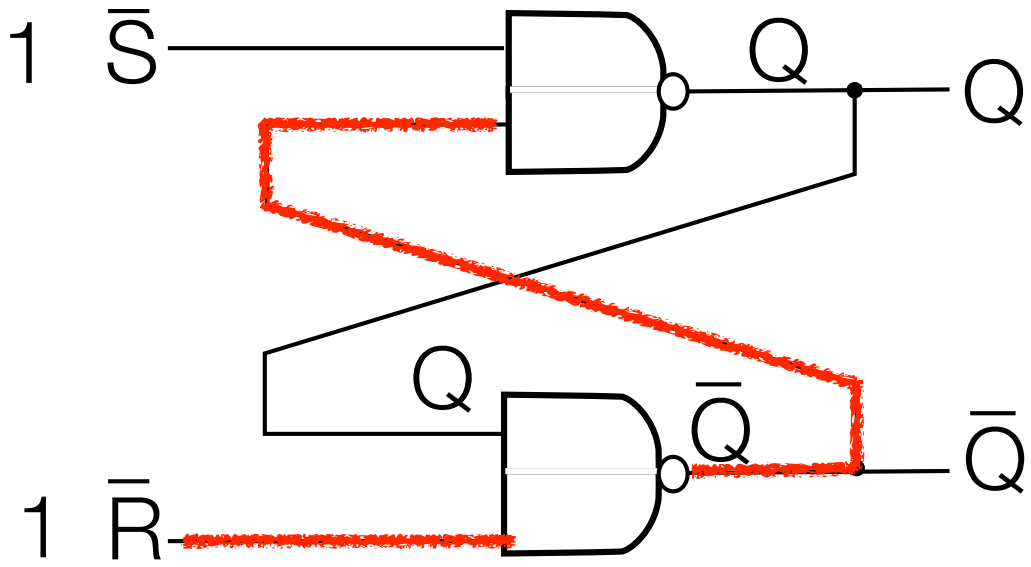
$\bar{R}$	$\bar{S}$	Q	$\bar{Q}$
0	0	1	1
0	1	0	1
1	0	1	0
1	1	Q	$\bar{Q}$

$\bar{R}$ and $\bar{S}$ have reverse behavior of SR latch, so R and S have same behaviors

Bad - do not use!

$\bar{S}\bar{R}$ latch

- Latch constructed of cross-coupled **NAND** gates

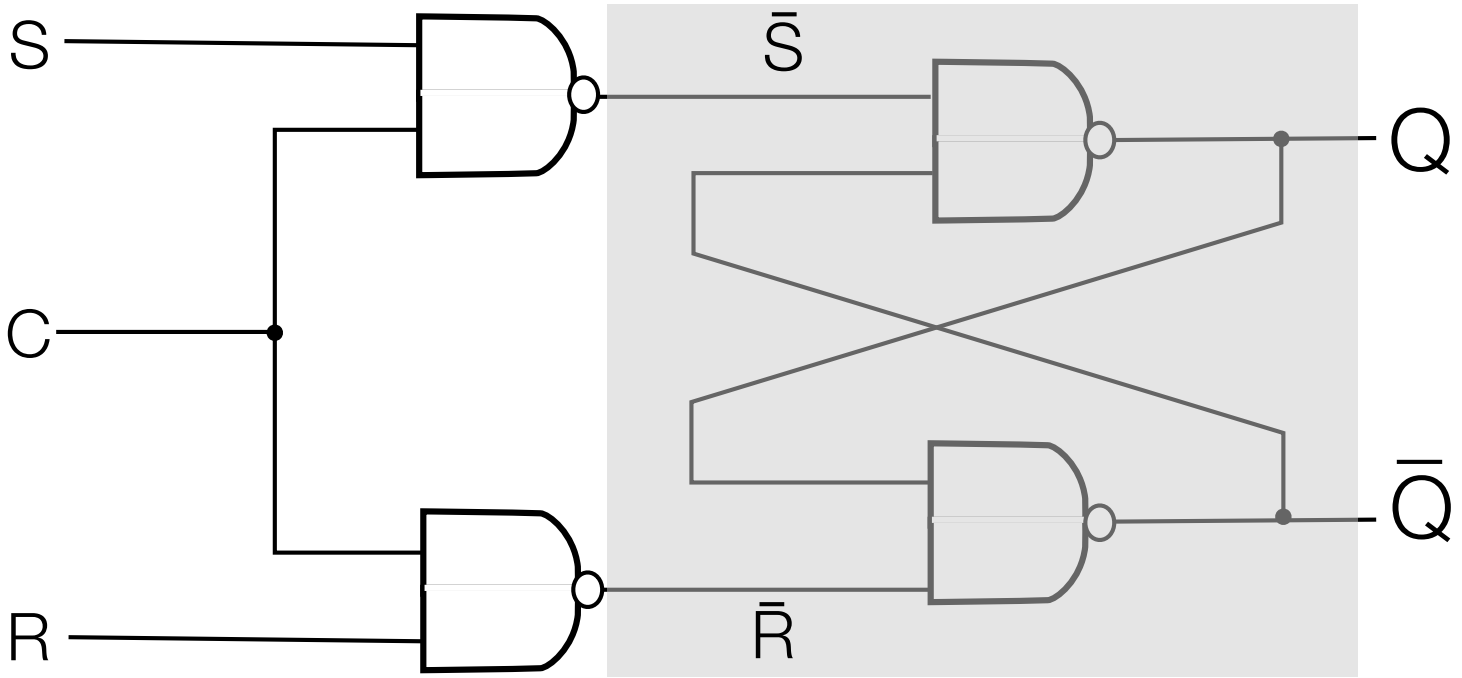


$\bar{R}$	$\bar{S}$	Q	$\bar{Q}$
0	0	1	1
0	1	0	1
1	0	1	0
1	1	Q	$\bar{Q}$

$\bar{R}$ and $\bar{S}$ have reverse behavior of SR latch, so R and S have same behaviors

Bad - do not use!

SR Latch with control

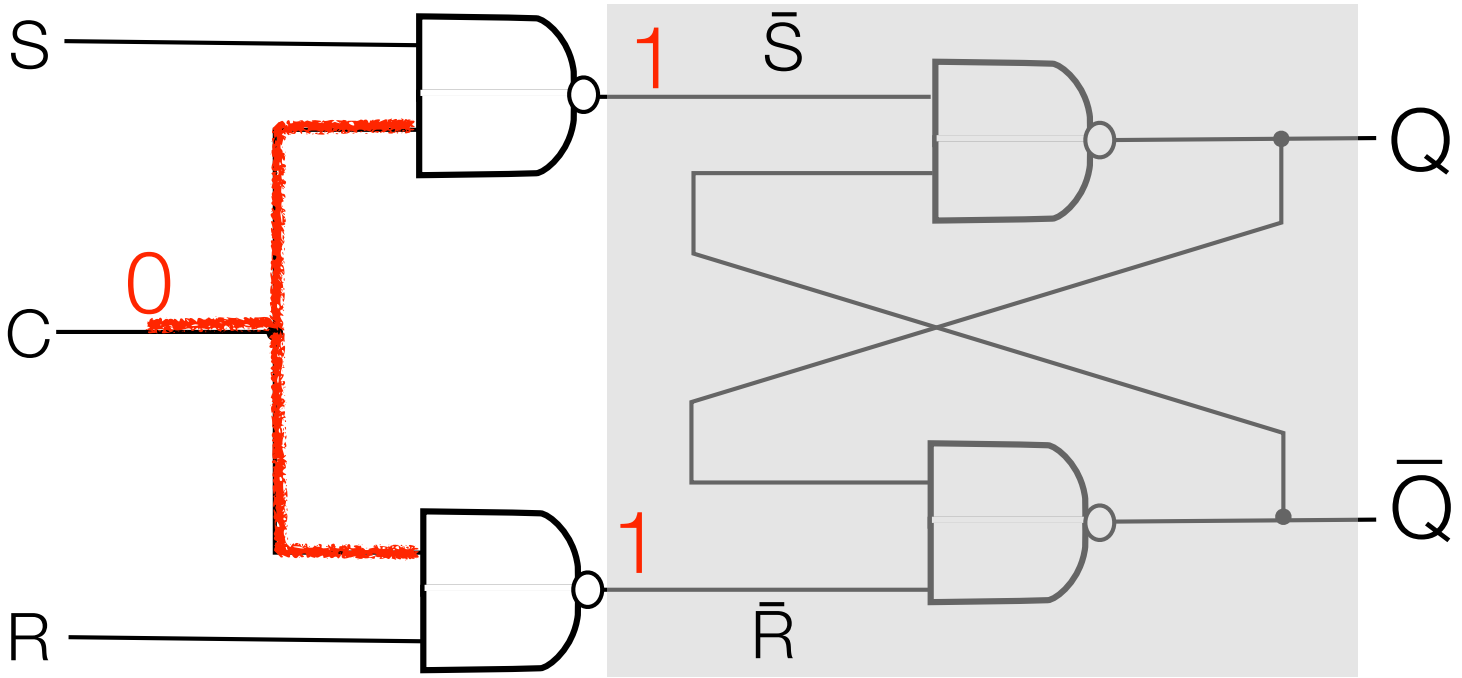


$\bar{S}\bar{R}$ latch

C	R	S	Q	$\bar{Q}$
0	X	X	Q	$\bar{Q}$
1	0	0	Q	$\bar{Q}$
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

- C=0, $\bar{S}\bar{R}$ latch receives $\bar{S}$=1, $\bar{R}$=1, values “hold”
- C=1, first set of NAND gates invert S & R inputs to $\bar{S}$ & $\bar{R}$

SR Latch with control

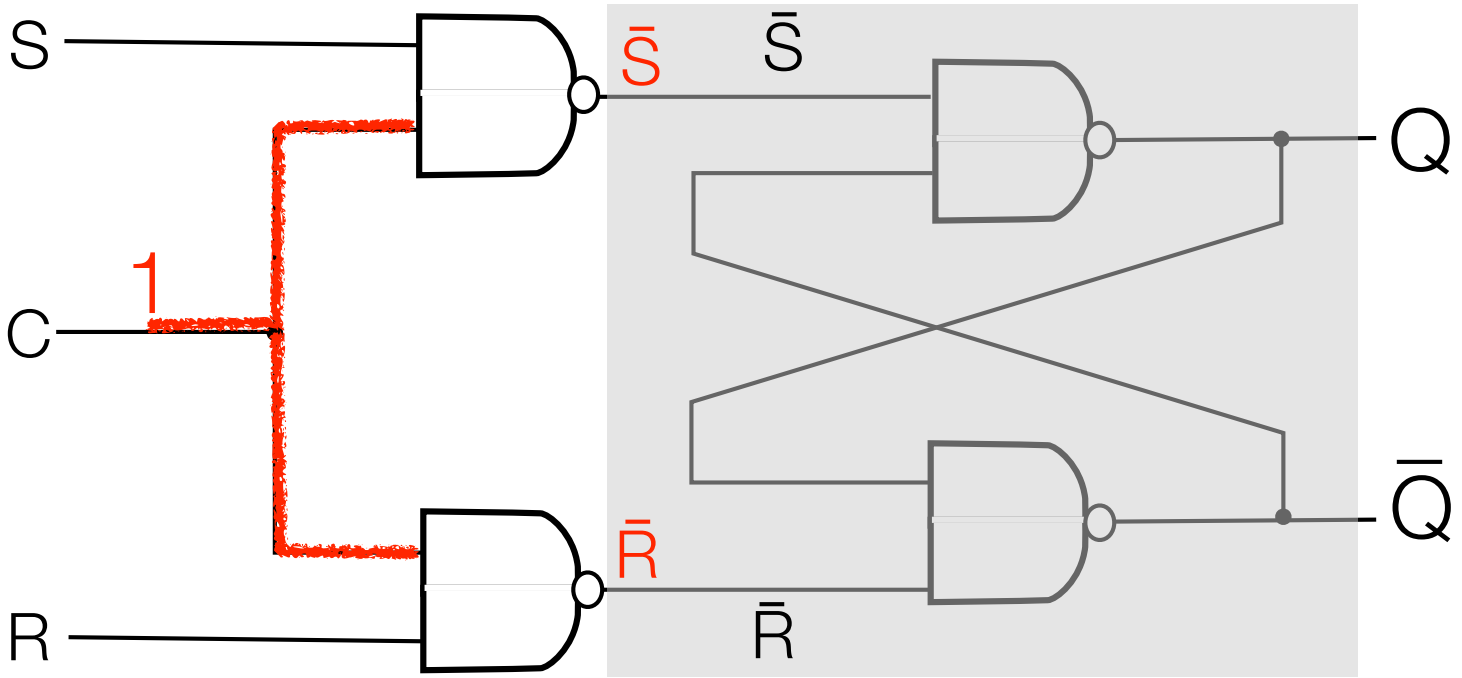


$\bar{S}\bar{R}$ latch

C	R	S	Q	$\bar{Q}$
0	X	X	Q	$\bar{Q}$
1	0	0	Q	$\bar{Q}$
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

- C=0, $\bar{S}\bar{R}$ latch receives $\bar{S}$=1, $\bar{R}$=1, values “hold”
- C=1, first set of NAND gates invert S & R inputs to $\bar{S}$ & $\bar{R}$

SR Latch with control

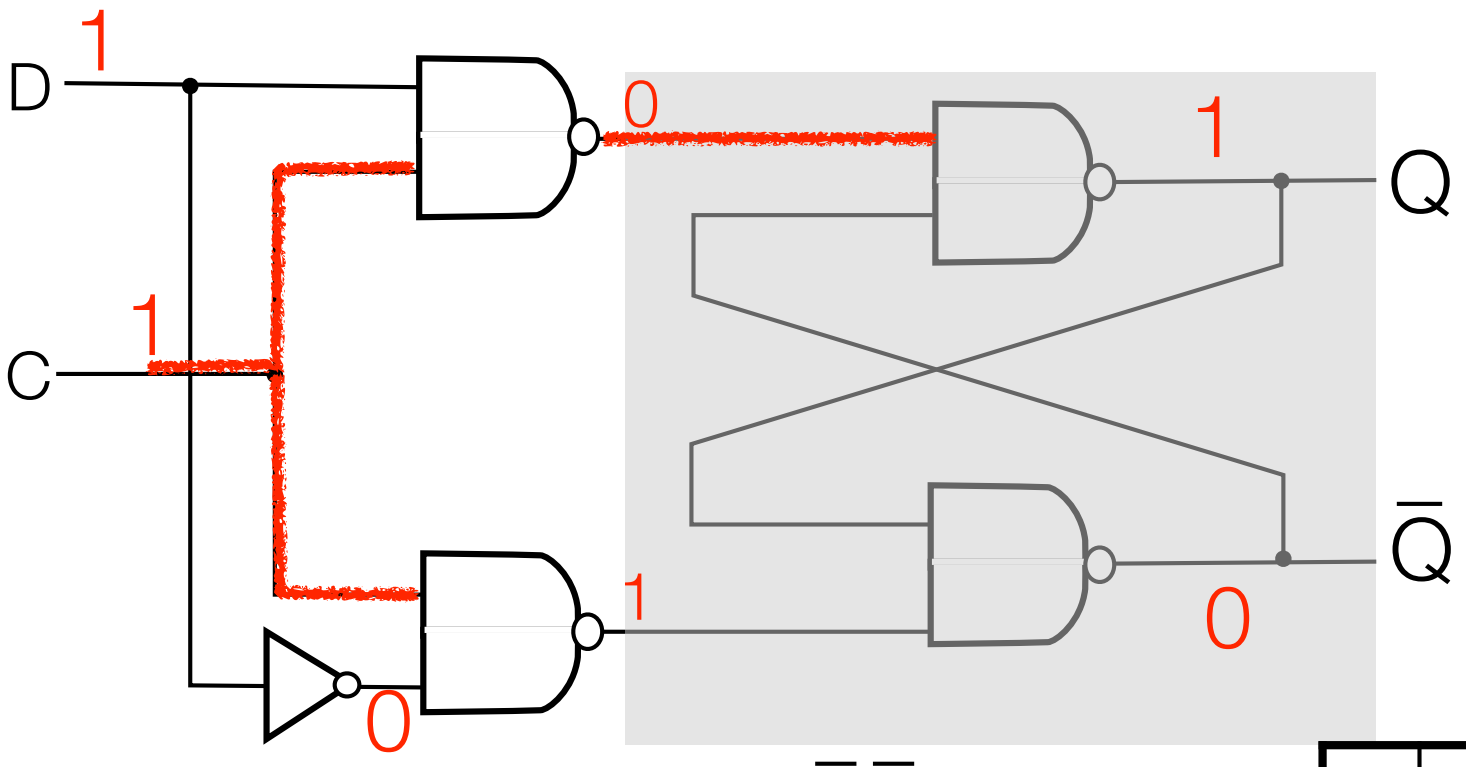


$\bar{S}\bar{R}$ latch

C	R	S	Q	$\bar{Q}$
0	X	X	Q	$\bar{Q}$
1	0	0	Q	$\bar{Q}$
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

- C=0, $\bar{S}\bar{R}$ latch receives $\bar{S}$=1, $\bar{R}$=1, values “hold”
- C=1, first set of NAND gates invert S & R inputs to $\bar{S}$ & $\bar{R}$

D Latch with control (our “intuitive latch” with data & write)

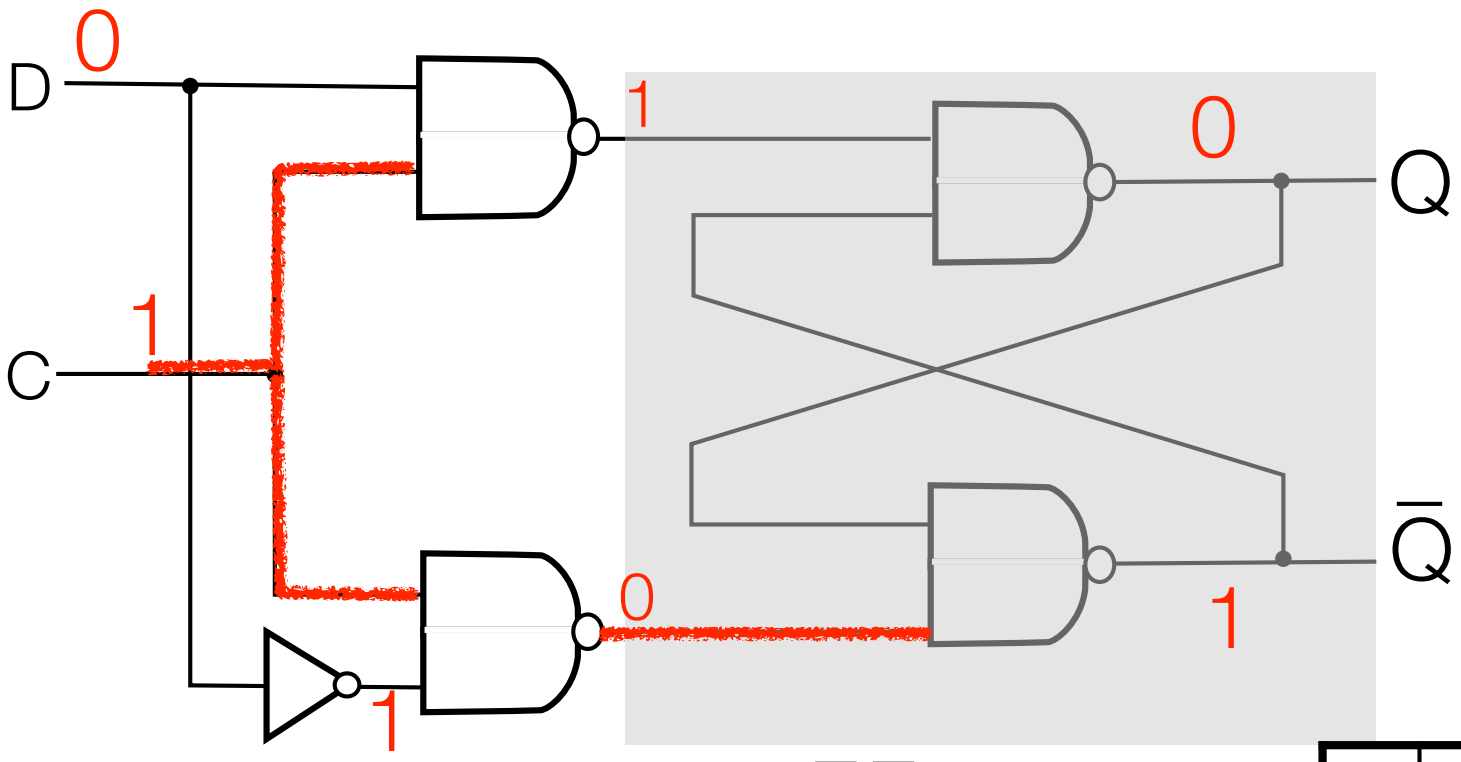


$\bar{S}\bar{R}$ latch

C	D	Q	$\bar{Q}$
0	X	Q	$\bar{Q}$
1	0	0	1
1	1	1	0

- With the control (C), no reason to ever have S=R
- C=0, latch holds value, C=1, Q=D

D Latch with control (our “intuitive latch” with data & write)

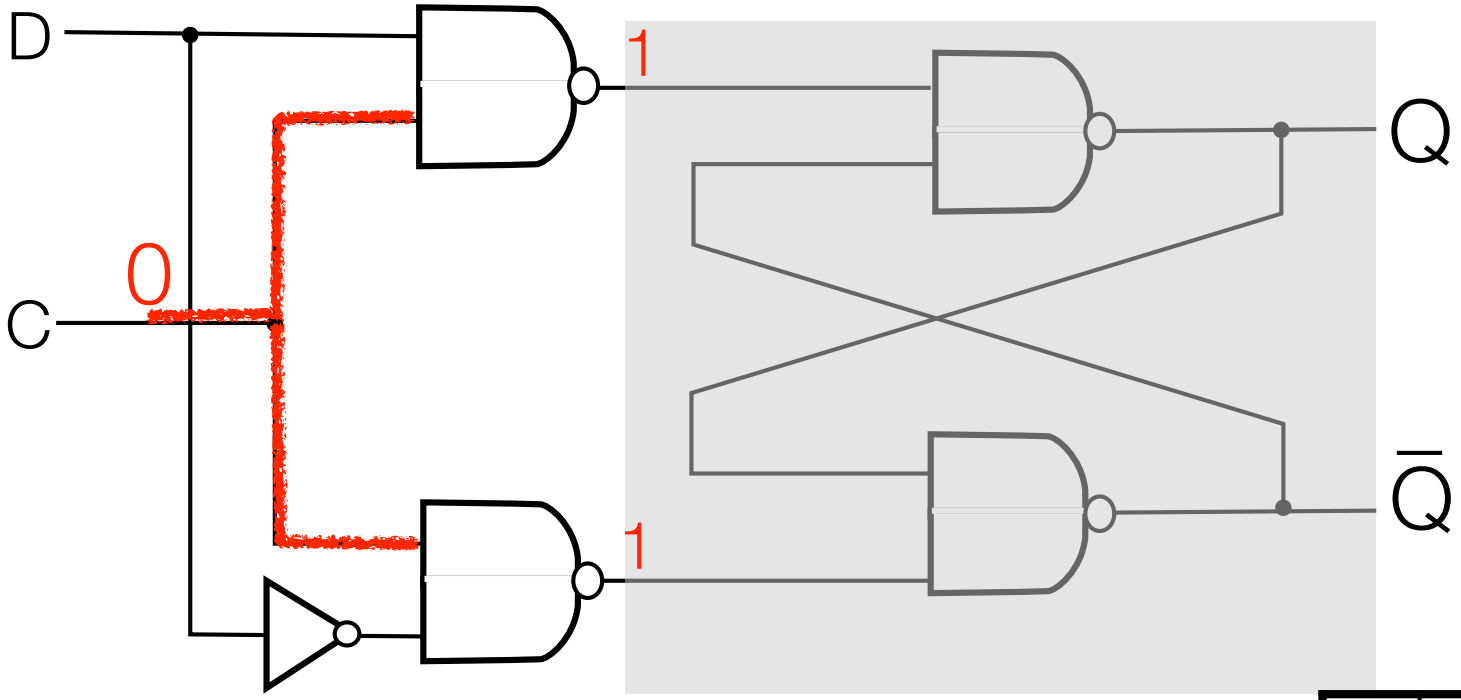


$\bar{S}\bar{R}$ latch

C	D	Q	$\bar{Q}$
0	X	Q	$\bar{Q}$
1	0	0	1
1	1	1	0

- With the control (C), no reason to ever have S=R
- C=0, latch holds value, C=1, Q=D

D Latch with control (our “intuitive latch” with data & write)

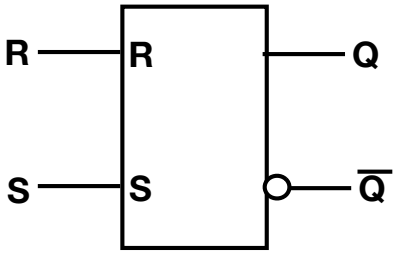


C	D	Q	$\bar{Q}$
0	X	Q	$\bar{Q}$
1	0	0	1
1	1	1	0

- With the control (C), no reason to ever have S=R
- C=0, latch holds value, C=1, Q=D

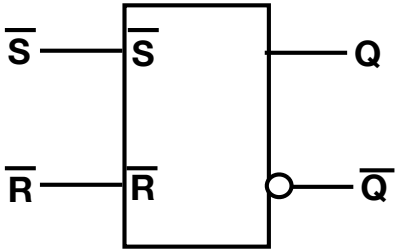
Circuit diagrams for latches

SR Latc



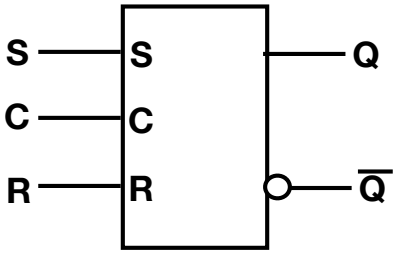
R	S	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	1	0
1	0	0	1
1	1	0	0

$\overline{S}\overline{R}$ Latch



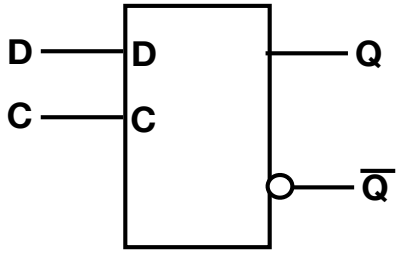
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$
0	0	0	0
0	1	1	0
1	0	0	1
1	1	Q	$\overline{Q}$

SR Latch with control



C	R	S	Q	$\overline{Q}$
0	X	X	Q	$\overline{Q}$
1	0	0	Q	$\overline{Q}$
1	0	1	1	0
1	1	0	0	1
1	1	1	0	0

D Latch with control



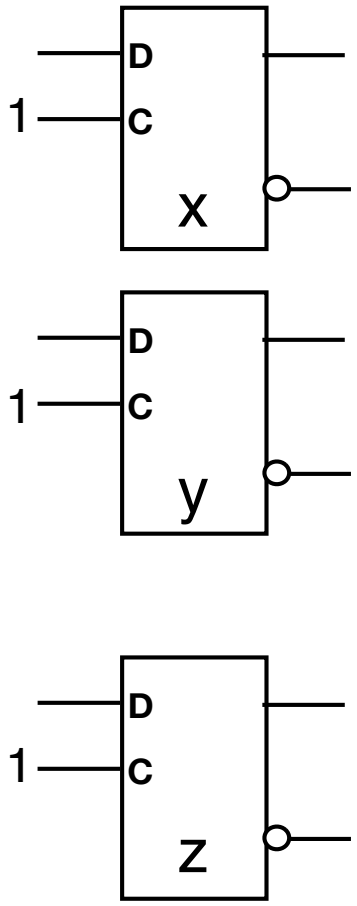
C	D	Q	$\overline{Q}$
0	X	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

Problem with Latches: No clocking

Where we are, where we are headed

- latches are circuits that can store "state"
  - set the latch to a value (0 or 1)
  - put the latch in a "same value" mode to hold the value
- Some computations use previously computed state
  - e.g., consider following pseudocode

```
while (){
  y = x̄ ⊕ ȳ ⊕ z
  z = ȳ+z̄
  x = new input
  output = x+z
}
```



Use 3 D-latches to store the 3 bits of state (x,y,z)

Where we are, where we are headed

- latches are circuits that can store “state”
  - set the latch to a value (0 or 1)
  - put the latch in a “same value” mode to hold the value
- Some computations use previously computed state
  - e.g., consider following pseudocode:

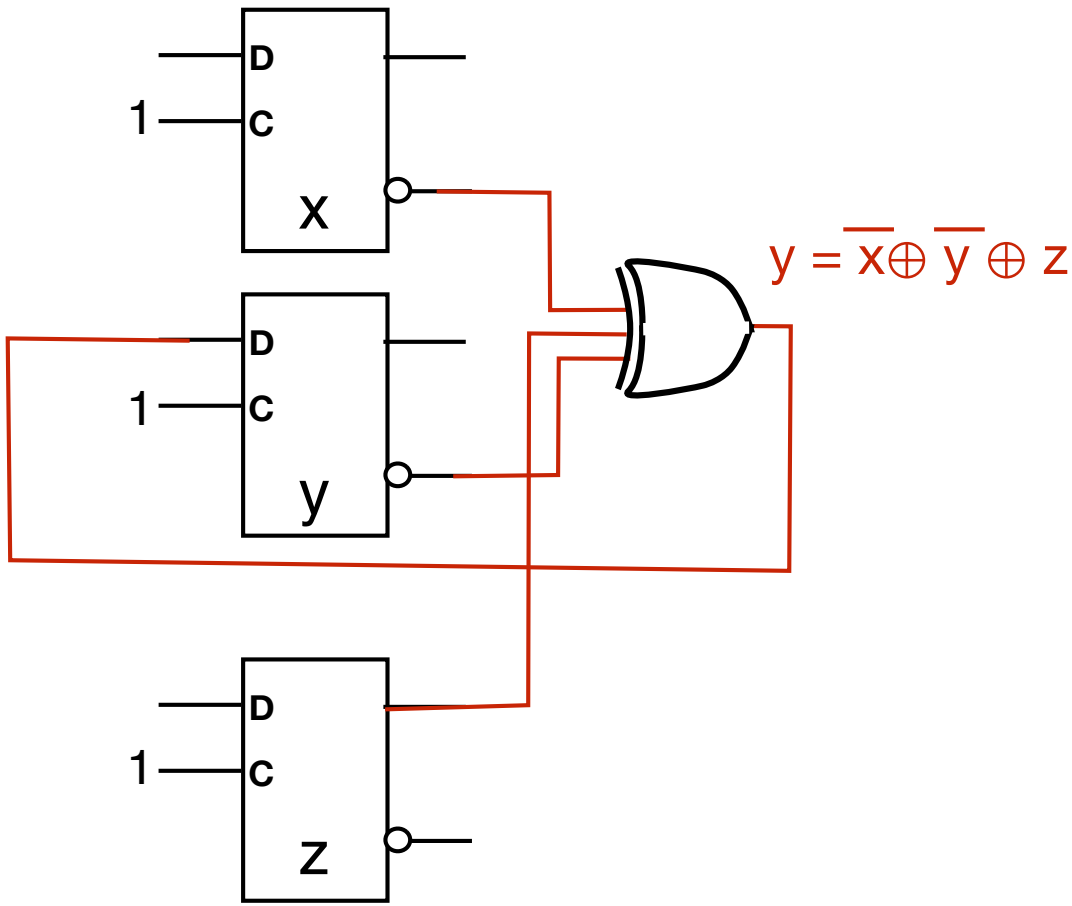
```
x=y=z=0
while (){
  y = x̄ ⊕ ȳ ⊕ z
  z = ȳ+z̄
  x = new input
  output = x+z
}
```

Iteration	x	y	z	new x	out x+z
0	0	0	0	1	1
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Where we are, where we are headed

- latches are circuits that can store "state"
  - set the latch to a value (0 or 1)
  - put the latch in a "same value" mode to hold the value
- Some computations use previously computed state
  - e.g., consider following pseudocode

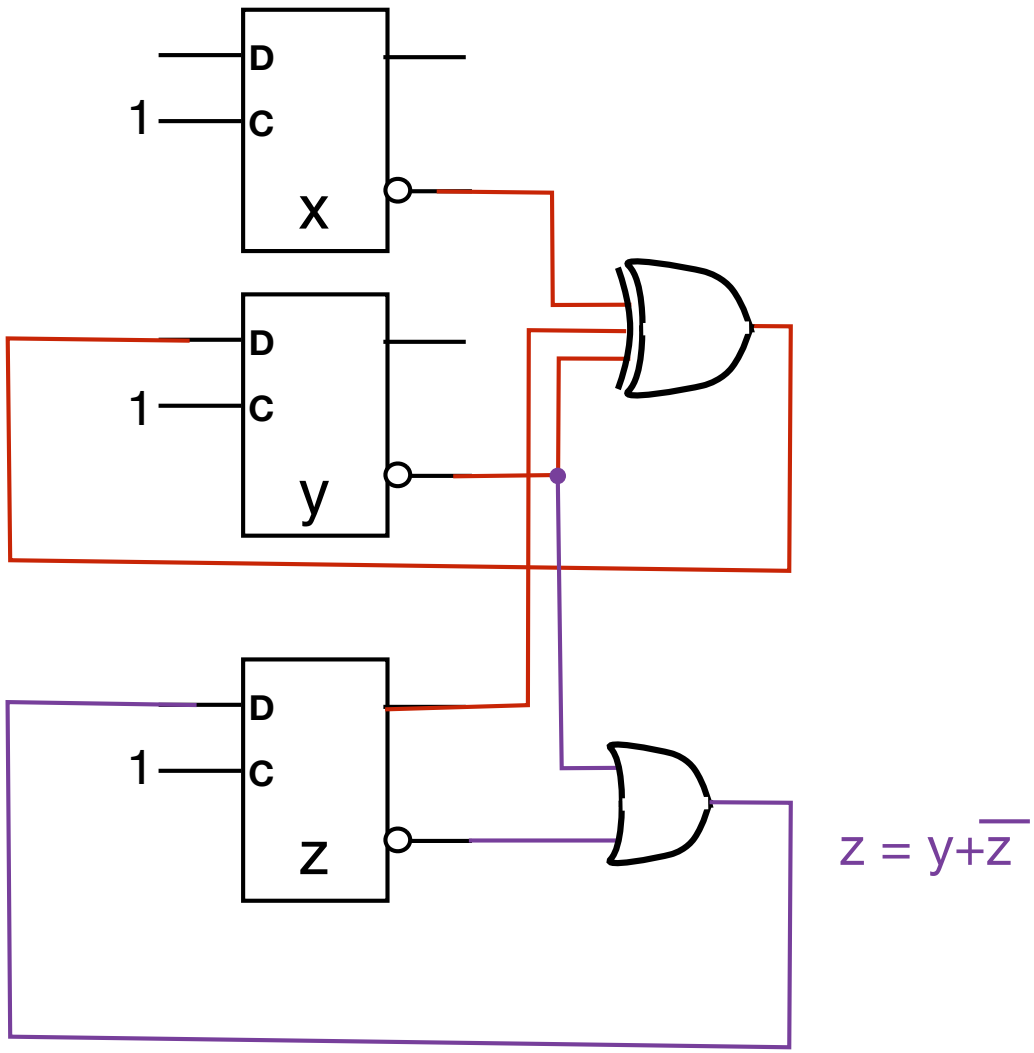
```
while (){
  y = x̄ ⊕ ȳ ⊕ z
  z = ȳ + z̄
  x = new input
  output = x+z
}
```



Where we are, where we are headed

- latches are circuits that can store "state"
  - set the latch to a value (0 or 1)
  - put the latch in a "same value" mode to hold the value
- Some computations use previously computed state
  - e.g., consider following pseudocode

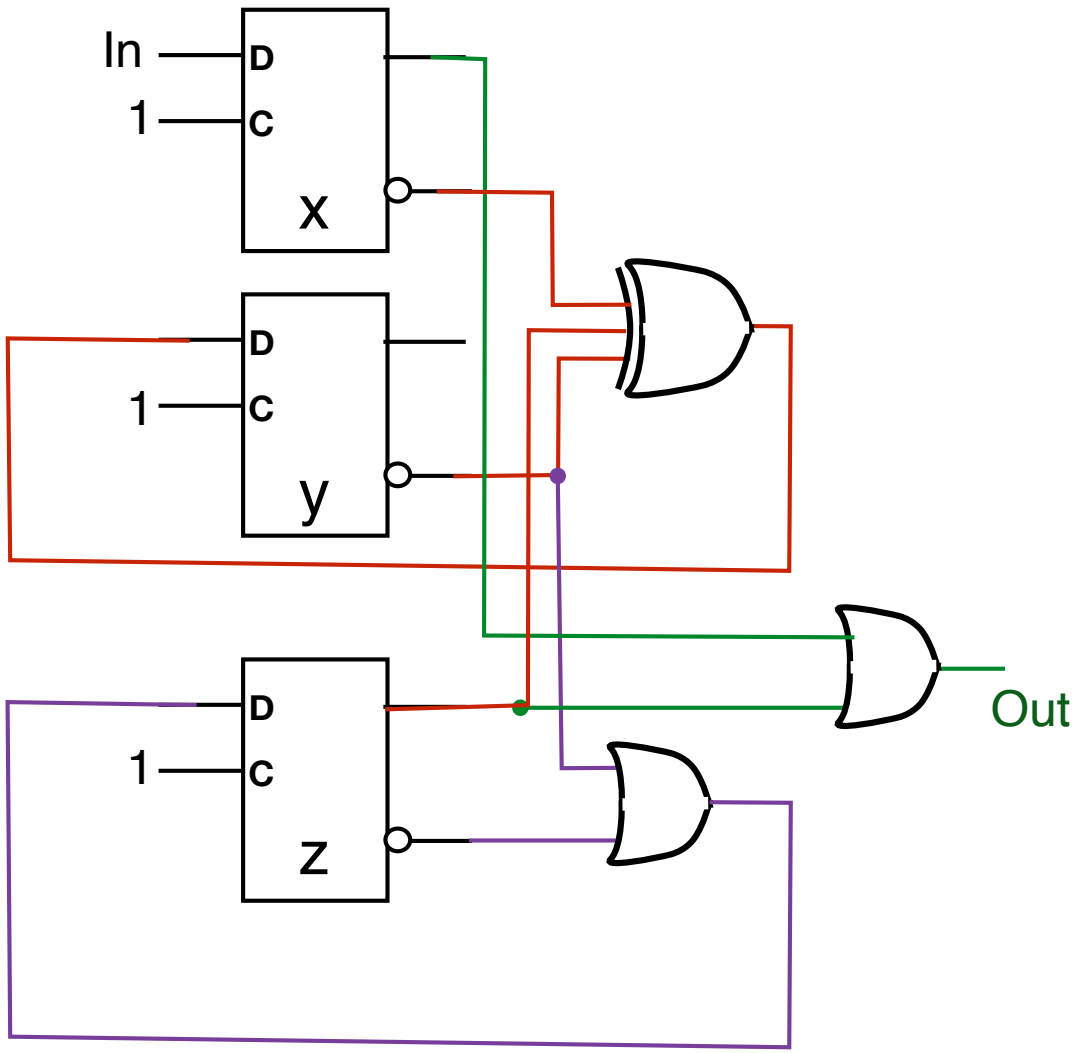
```
while (){
  y = x̄ ⊕ ȳ ⊕ z
  z = ȳ+z̄
  x = new input
  output = x+z
}
```



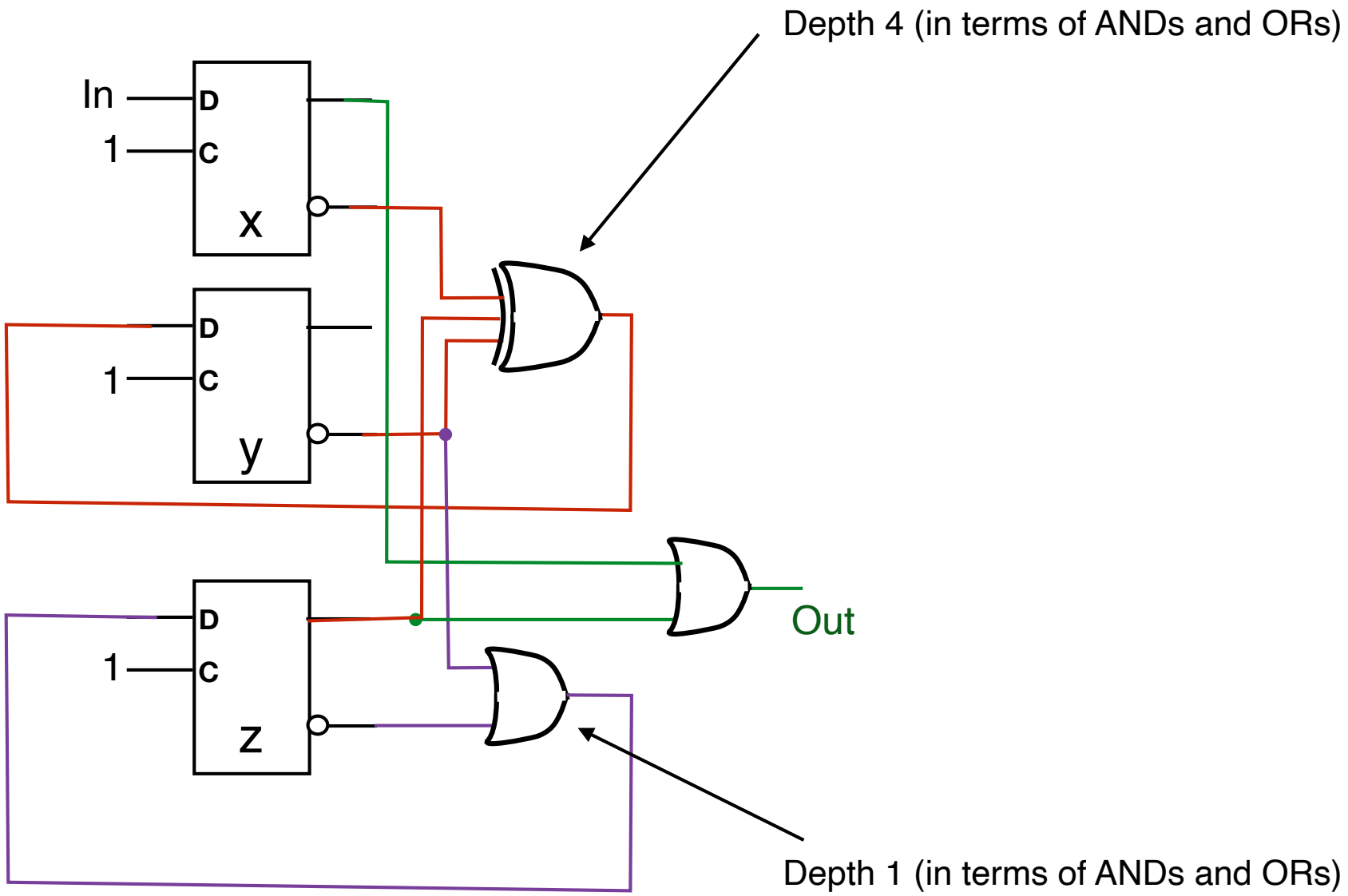
Where we are, where we are headed

- latches are circuits that can store "state"
  - set the latch to a value (0 or 1)
  - put the latch in a "same value" mode to hold the value
- Some computations use previously computed state
  - e.g., consider following pseudocode

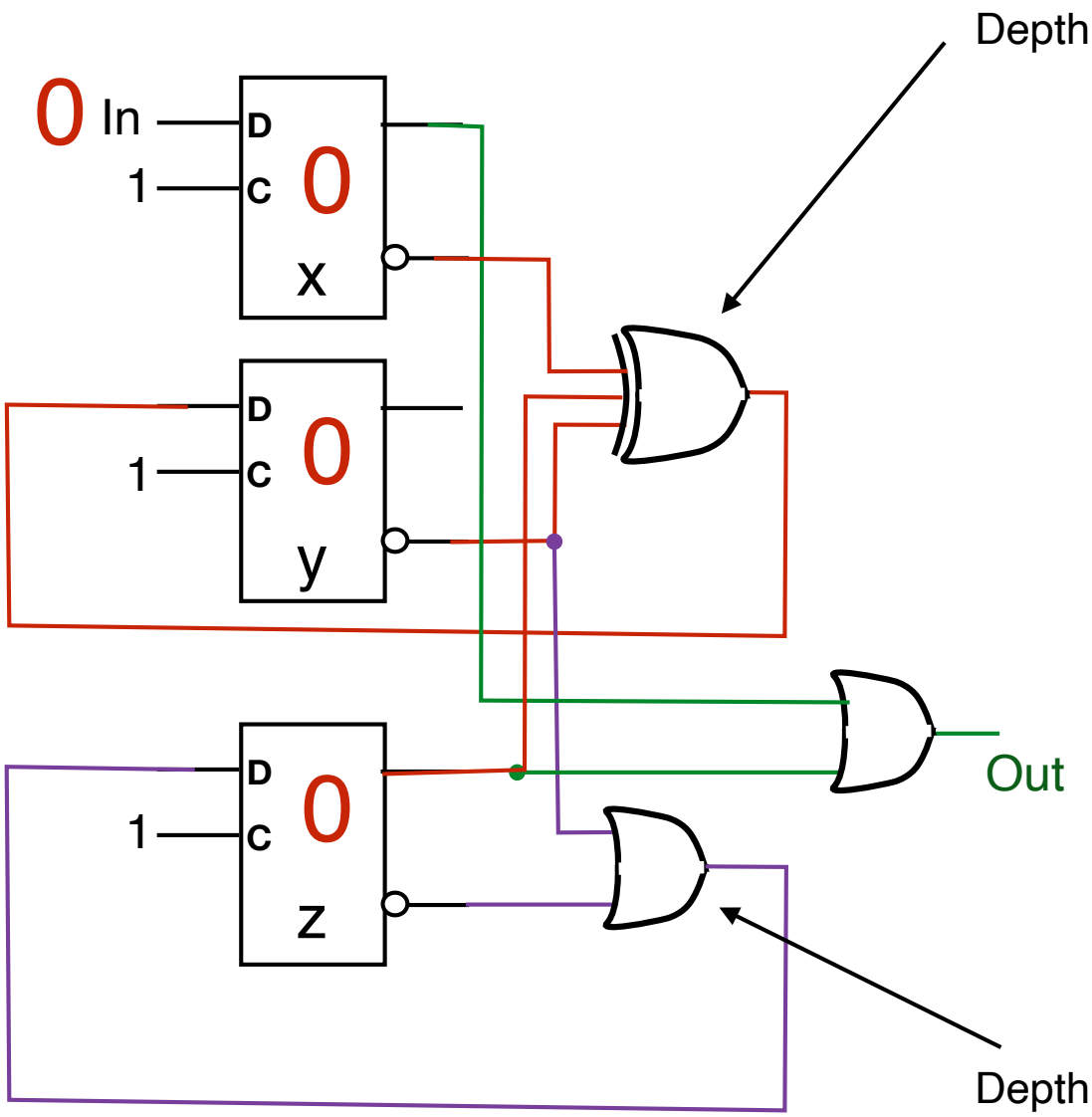
```
while (){
  y = x̄⊕ ȳ ⊕ z
  z = ȳ+z̄
  x = new input
  output = x+z
}
```



Where we are, where we are headed

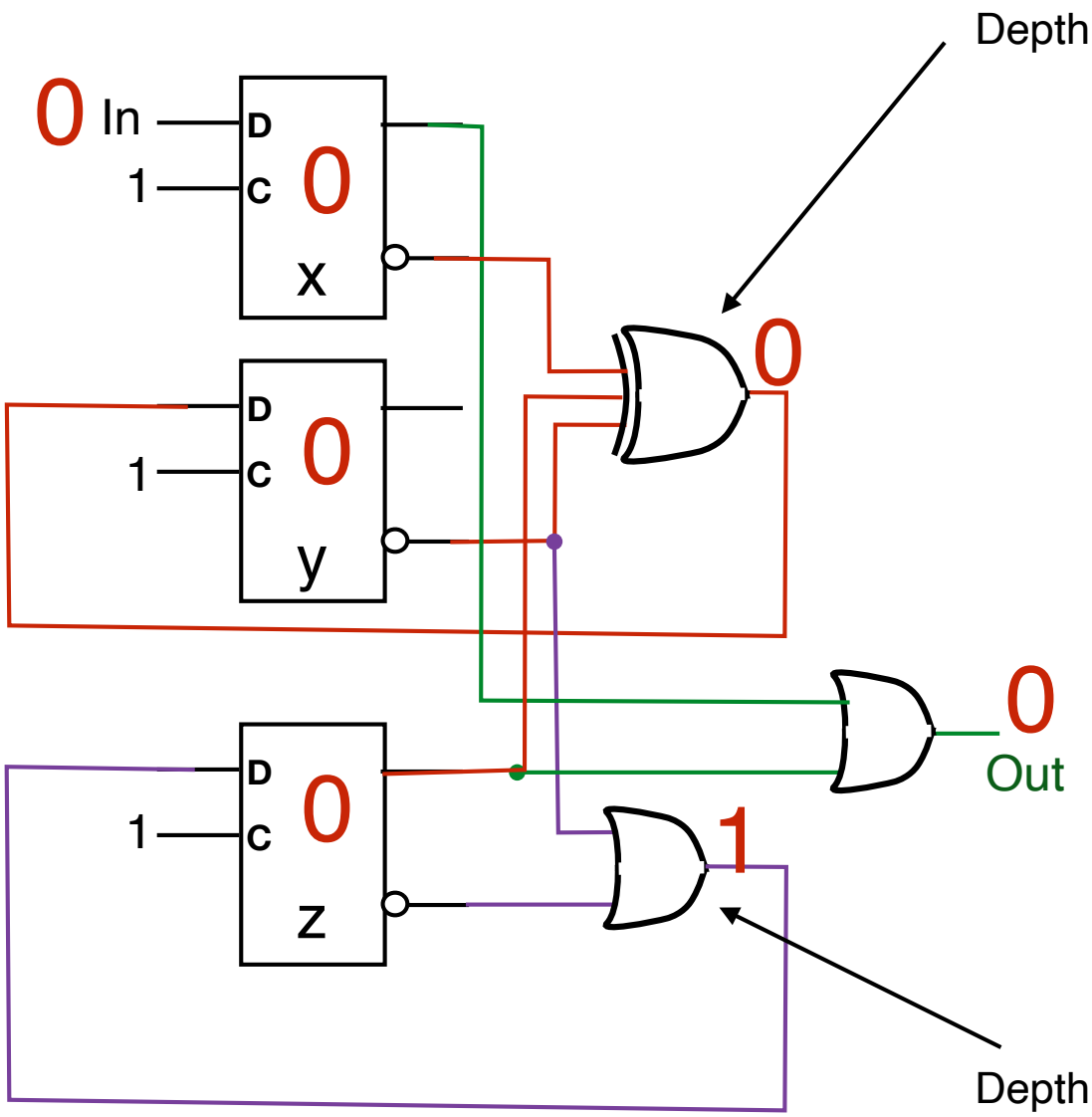


Where we are, where we are headed



Iteration	x	y	z	new x	out x+z
0	0	0	0	1	1
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0

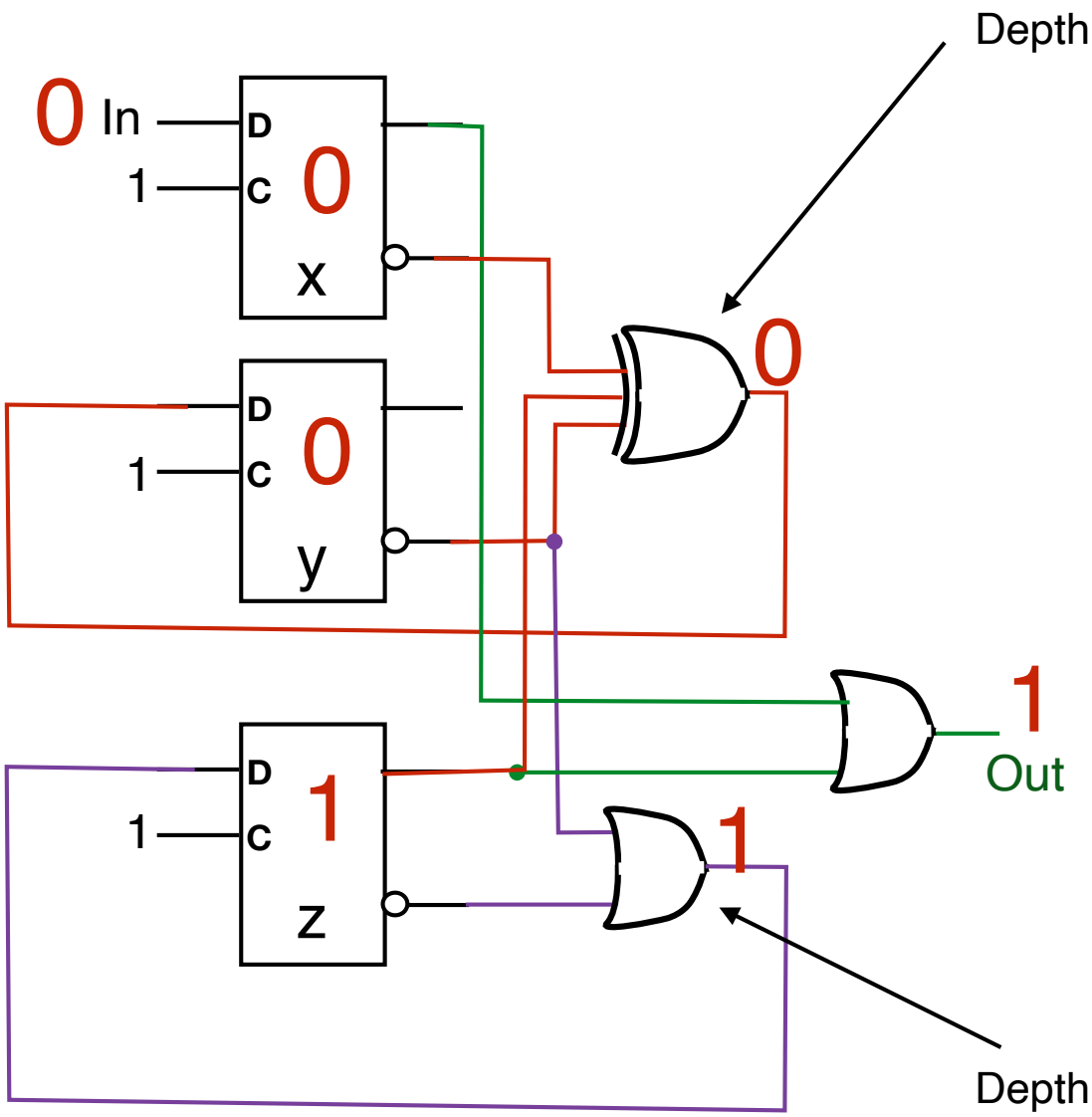
Where we are, where we are headed



Iteration	x	y	z	new x	out x+z
0	0	0	0	1	1
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Initial state stable

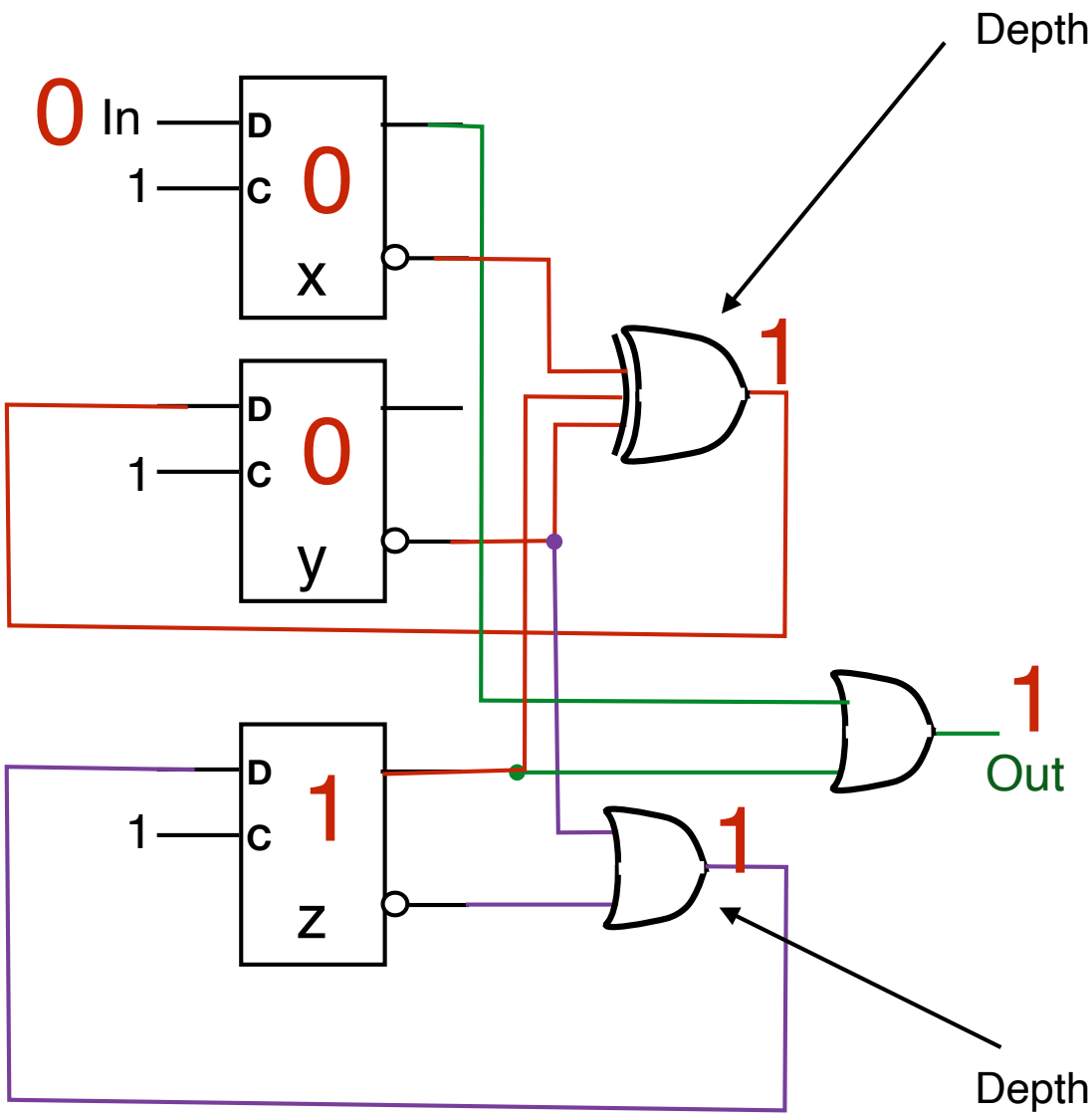
Where we are, where we are headed



Iteration	x	y	z	new x	out x+z
0	0	0	0	1	1
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Initial state stable

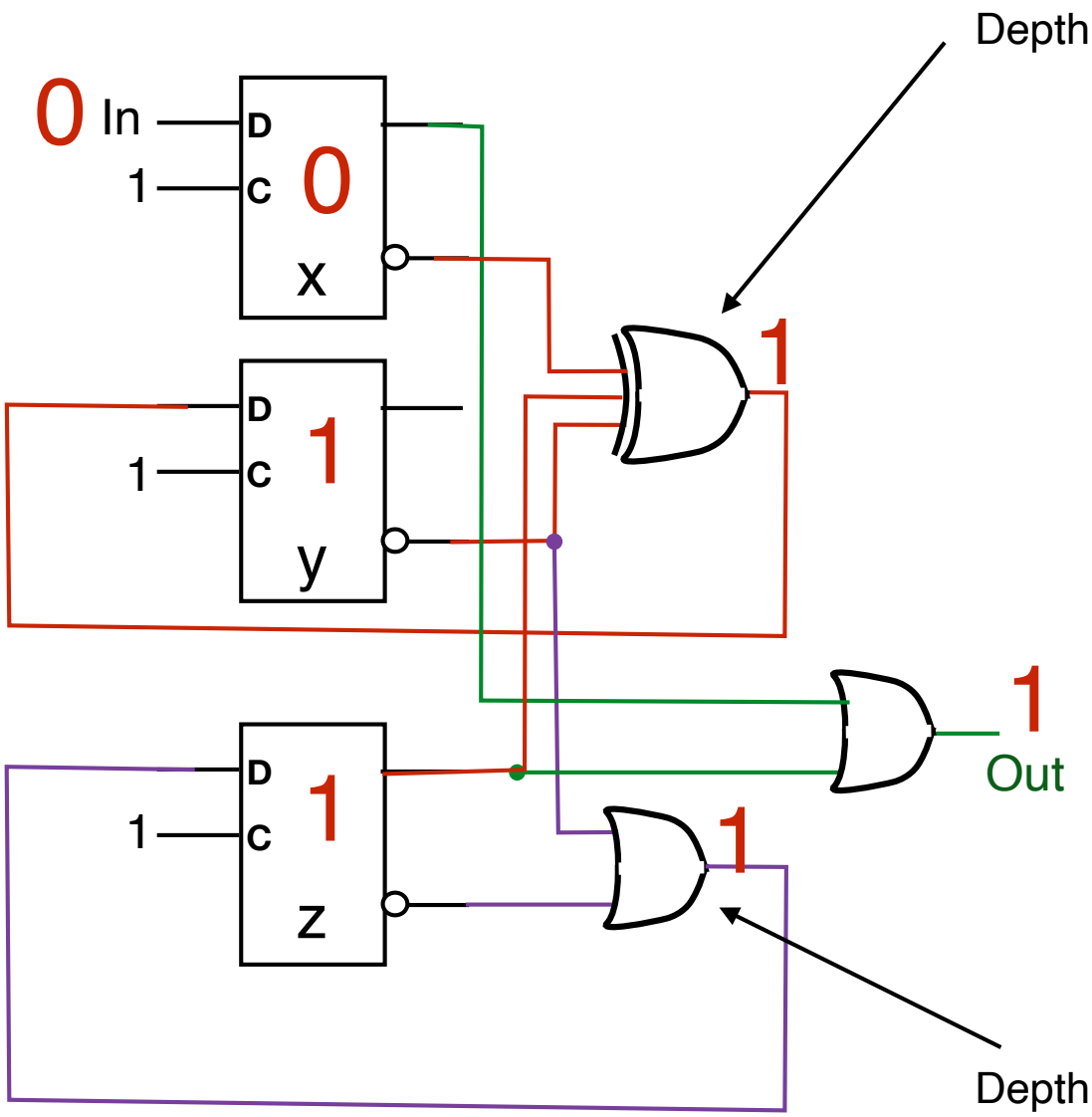
Where we are, where we are headed



Iteration	x	y	z	new x	out x+z
0	0	0	0	1	1
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Initial state stable

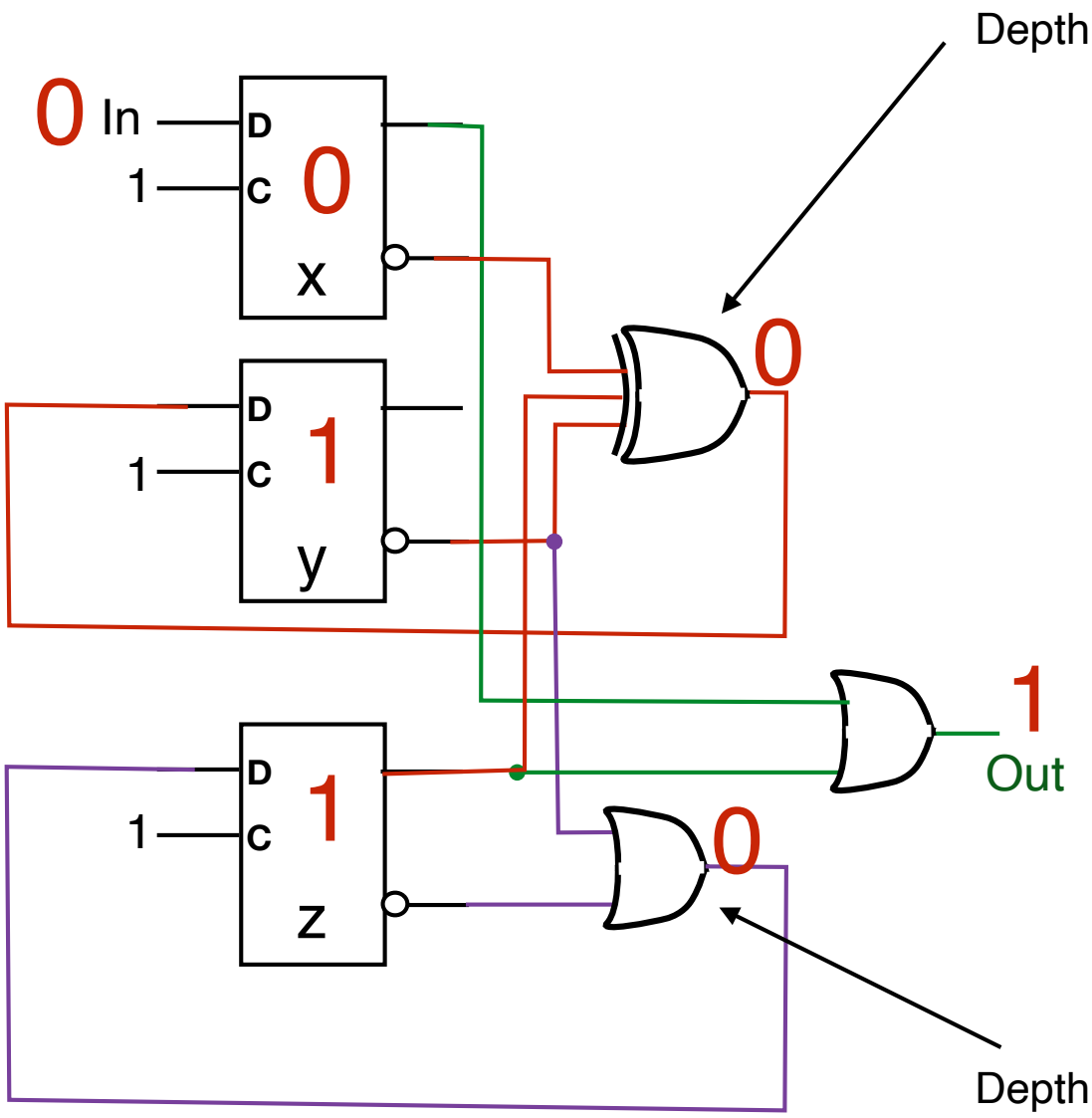
Where we are, where we are headed



Iteration	x	y	z	new x	out x+z
0	0	0	0	1	1
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Initial state stable

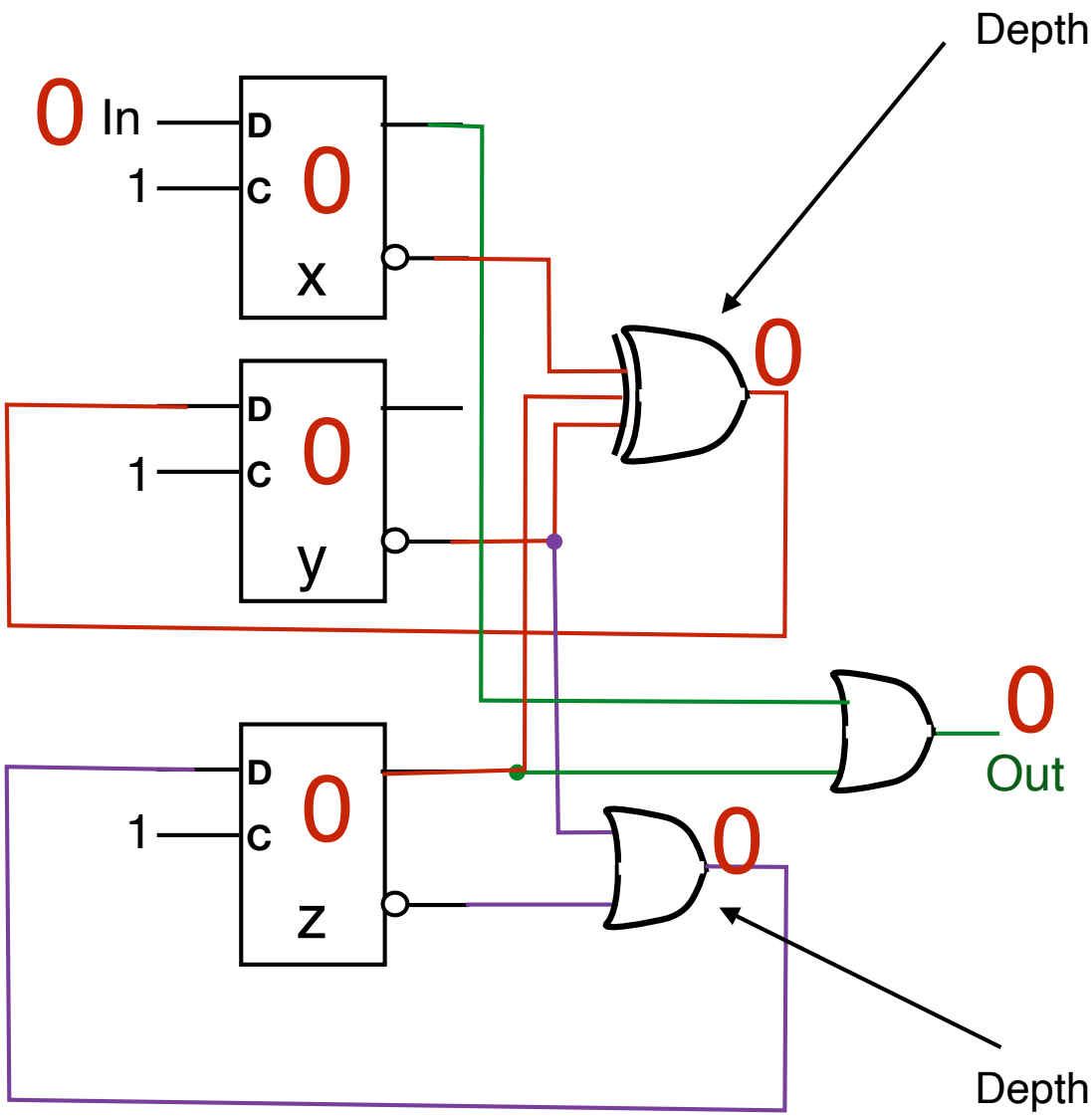
Where we are, where we are headed



Iteration	x	y	z	new x	out x+z
0	0	0	0	1	1
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Initial state stable

Where we are, where we are headed



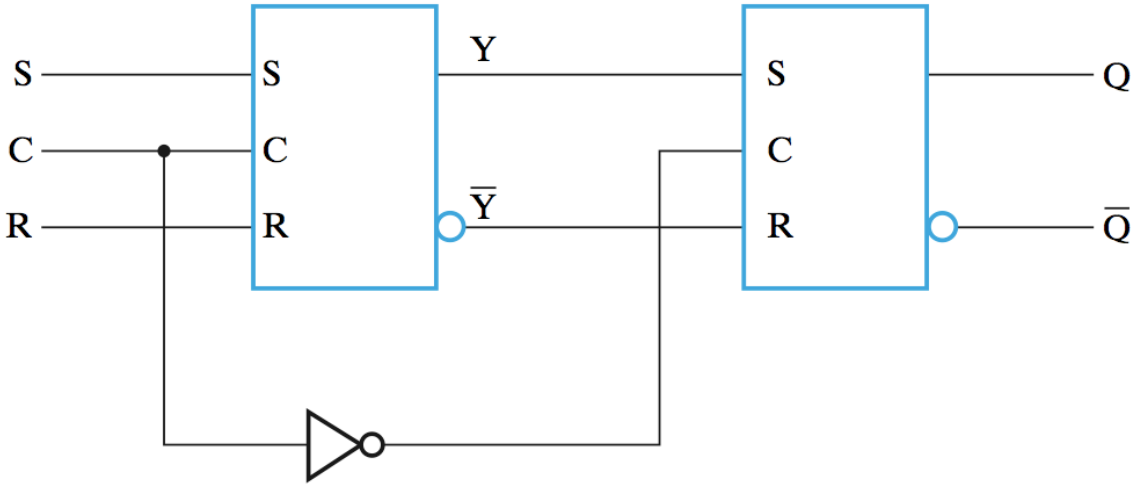
Iteration	x	y	z	new x	out x+z
0	0	0	0	1	1
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0

Initial state stable

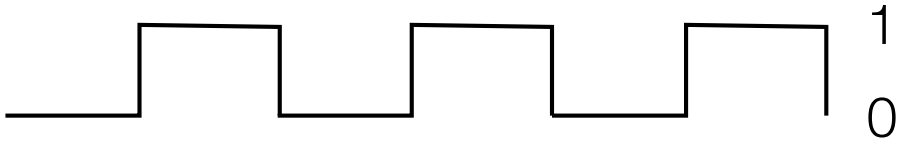
Flip Flops

SR Flip-Flop

2 SR Latches with control



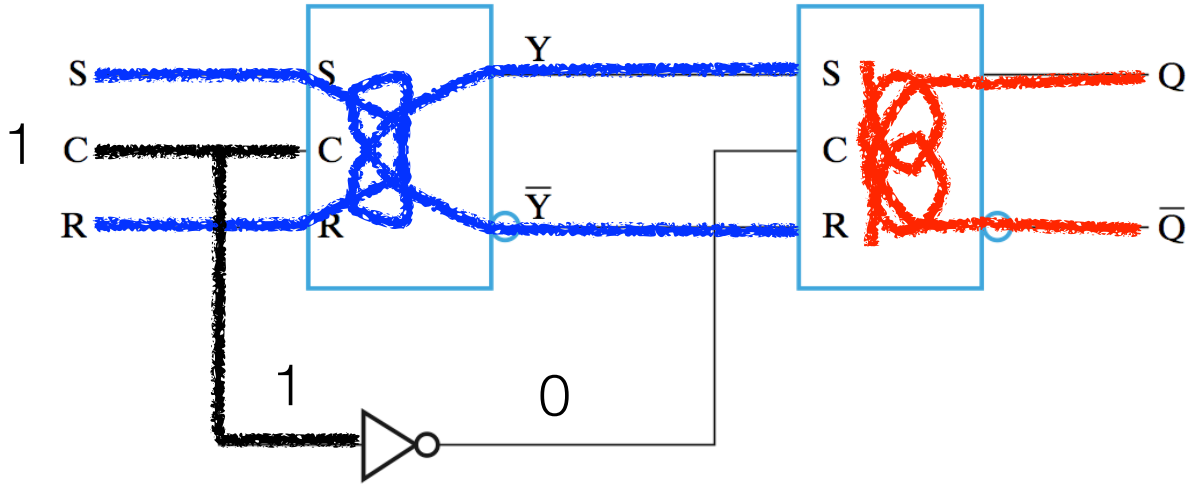
Clock:



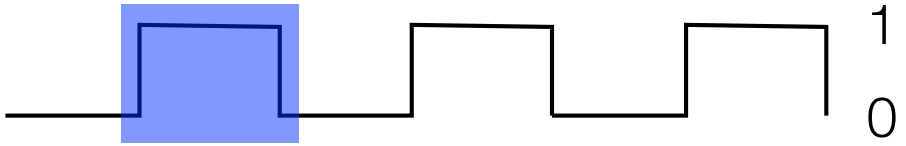
- C (Control) is fed a clock pulse (alternates between 0 and 1 with fixed period)
  - C=1: Leader latch “on”, Follower latch “off”
    - New S & R inputs read into leader
    - Previous Q values still emitted (not affected by new S&R inputs)
  - C=0: Leader latch “off”, Follower latch “on”
    - Changing S & R inputs has no effect on Lead (or Follower) latch
    - S&R inputs from last time C=1 stored safely in Leader and transferred into Follower

SR Flip-Flop

2 SR Latches with control



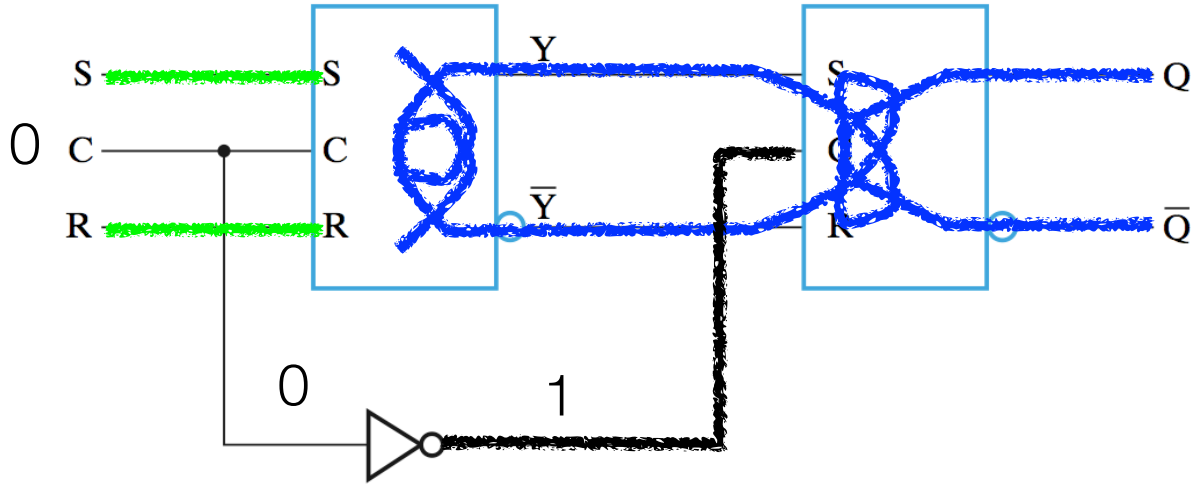
Clock:



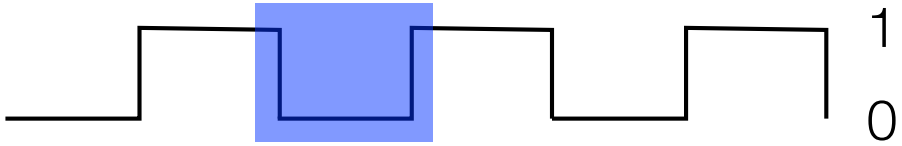
- C (Control) is fed a clock pulse (alternates between 0 and 1 with fixed period)
  - C=1: Leader latch “on”, Follower latch “off”
    - New S & R inputs read into leader
    - Previous Q values still emitted (not affected by new S&R inputs)
  - C=0: Leader latch “off”, Follower latch “on”
    - Changing S & R inputs has no effect on Lead (or Follower) latch
    - S&R inputs from last time C=1 stored safely in Leader and transferred into Follower

SR Flip-Flop

2 SR Latches with control

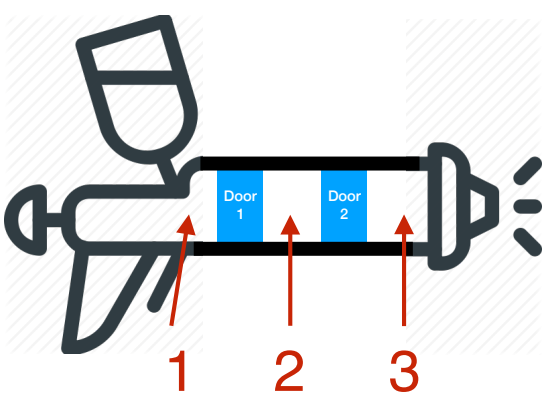


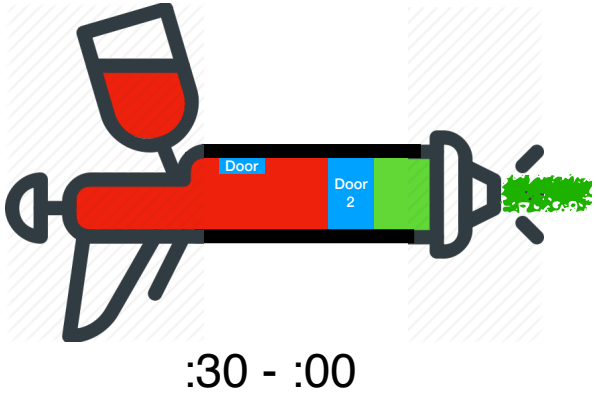
Clock:



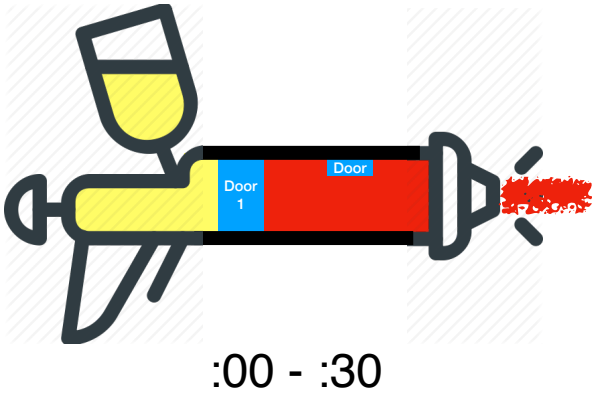
- C (Control) is fed a clock pulse (alternates between 0 and 1 with fixed period)
  - C=1: Leader latch “on”, Follower latch “off”
    - New S & R inputs read into leader
    - Previous Q values still emitted (not affected by new S&R inputs)
  - C=0: Leader latch “off”, Follower latch “on”
    - Changing S & R inputs has no effect on Lead (or Follower) latch
    - S&R inputs from last time C=1 stored safely in Leader and transferred into Follower

Like the Paint Gun from HW#0 (problem 10)

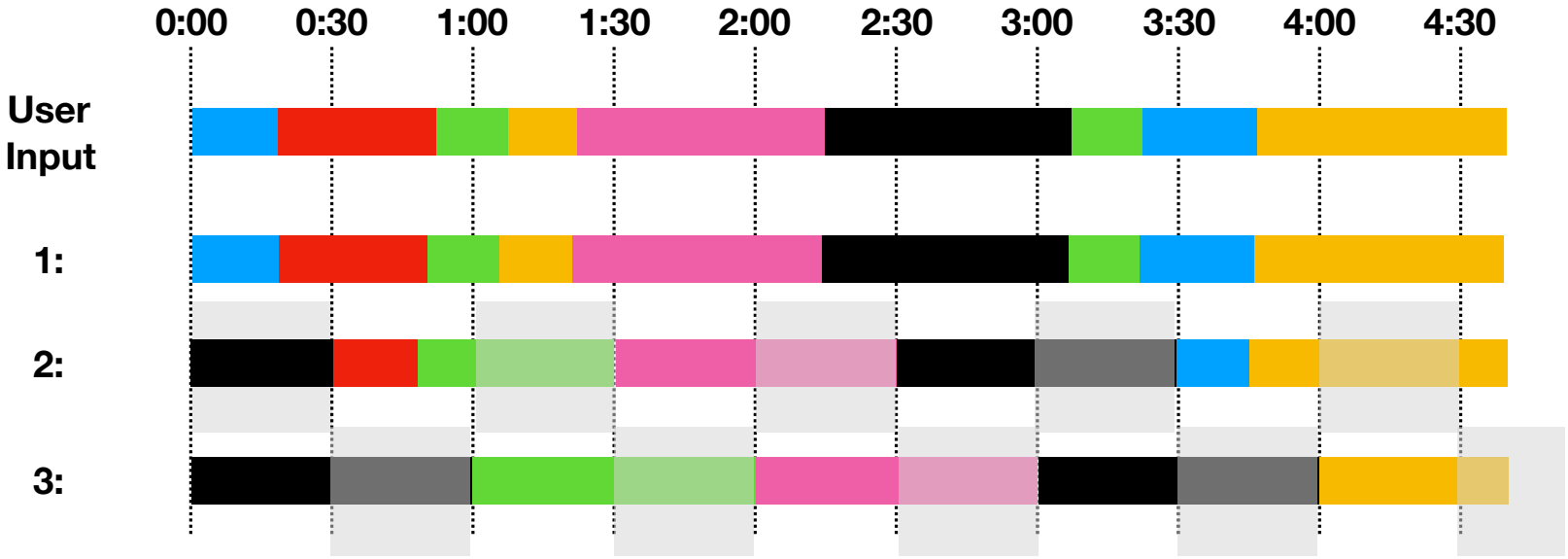




:30 - :00



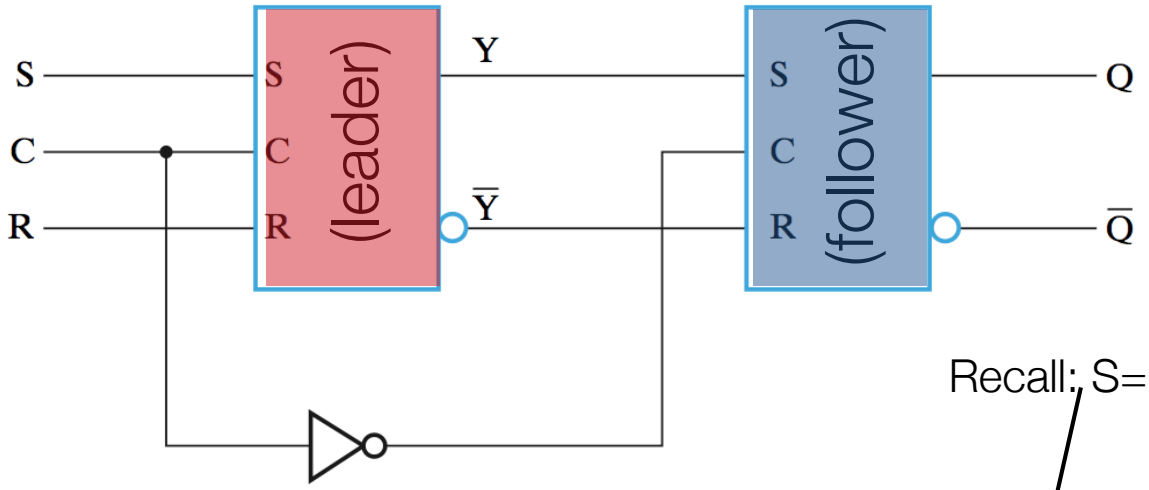
:00 - :30



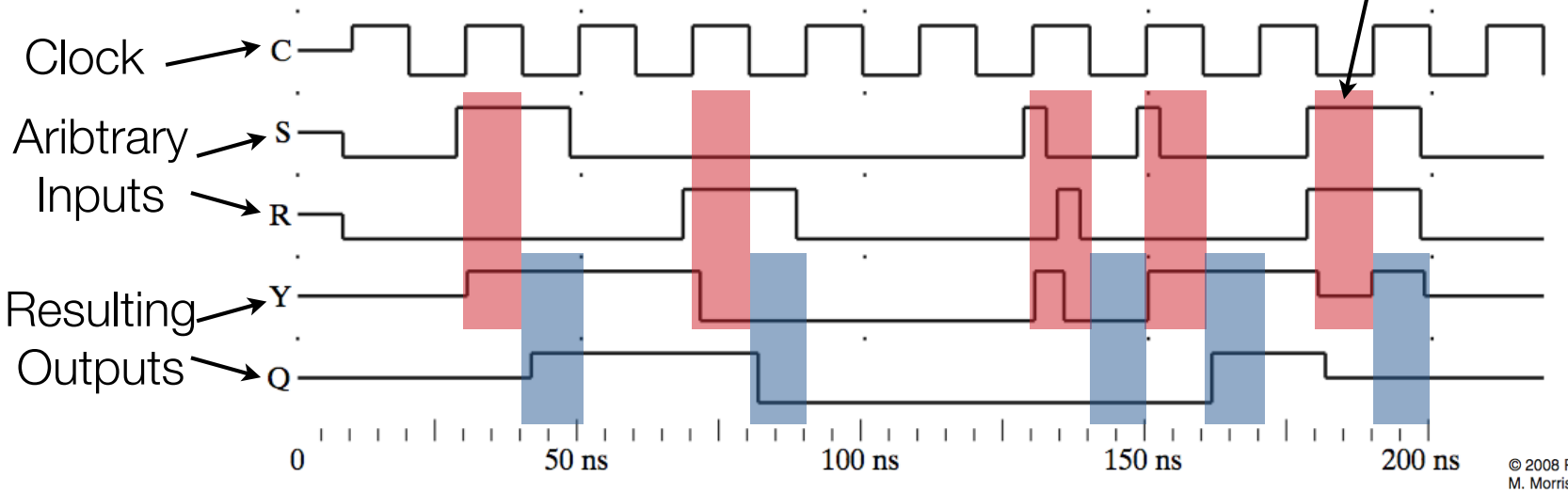
SR Leader-Follower flip-flop

Internal state (Y) updated when CLK=1

External state (Q) updated when CLK=0

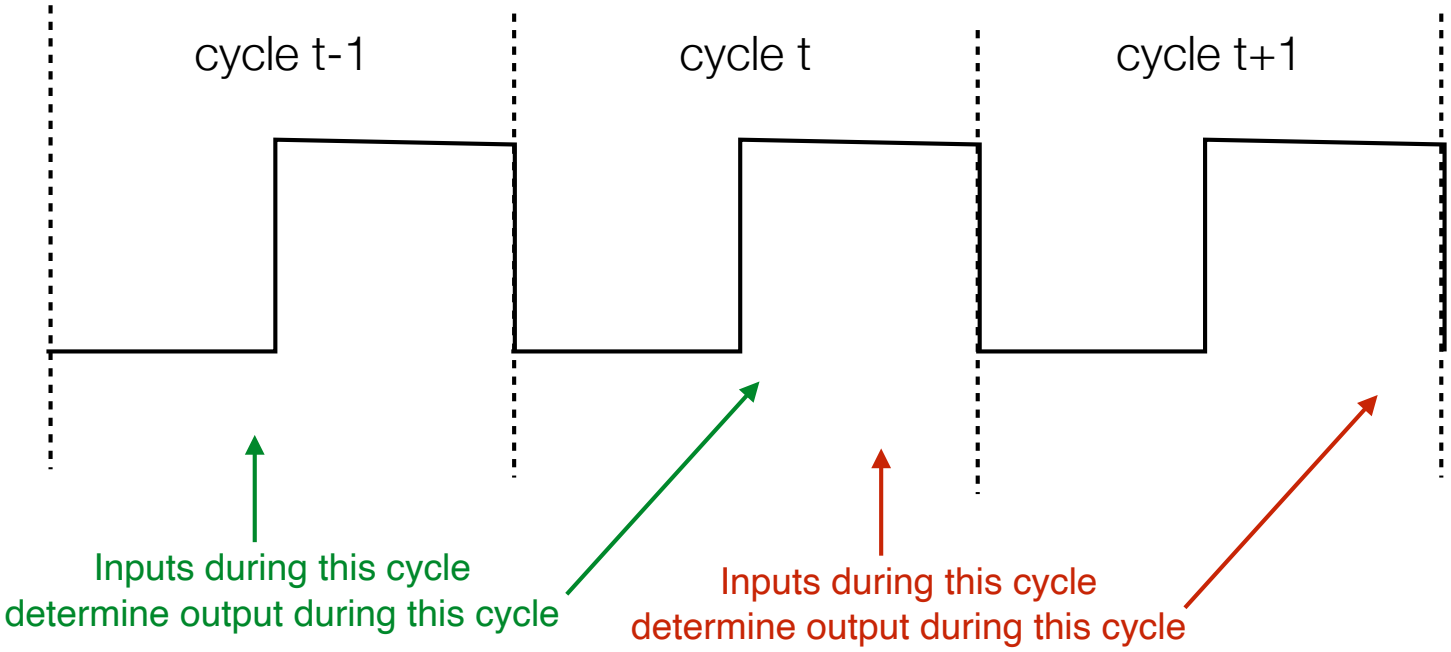


Recall: S=R=1 is "bad"



Using a Flip-Flop

So to use a Flip-Flop



Excitation Table

S(t)	R(t)	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Don't Use

Example

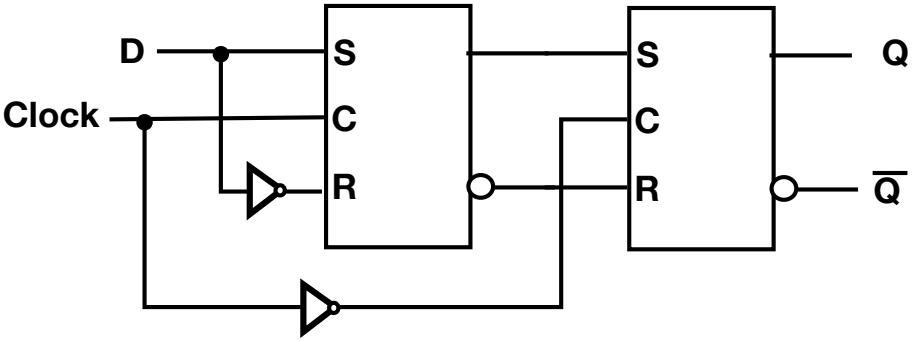
t	0	1	2	3	4	5	6	7	8
S(t)	0	0	1	1	0	1	0	0	0
R(t)	1	0	0	0	0	0	0	1	0
Q(t)	X	0 →	0	1	1 →	1	1 →	1	0

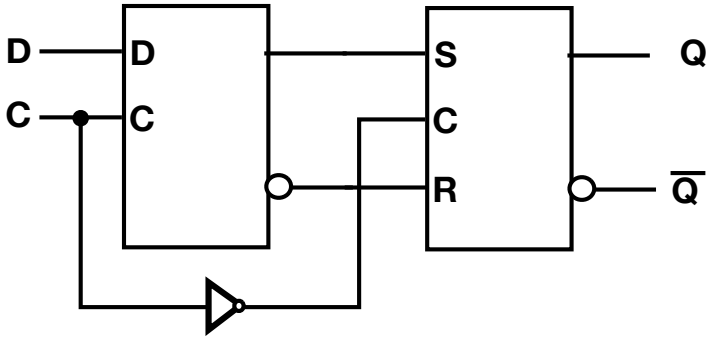
Squares of same color show dependencies between input and output

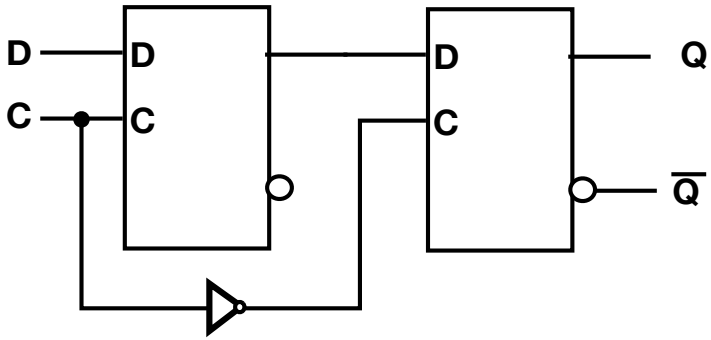
Other Types of Flip Flops (SR, D, T, JK)

D Flip-Flop

- Can build lots of ways - here are three



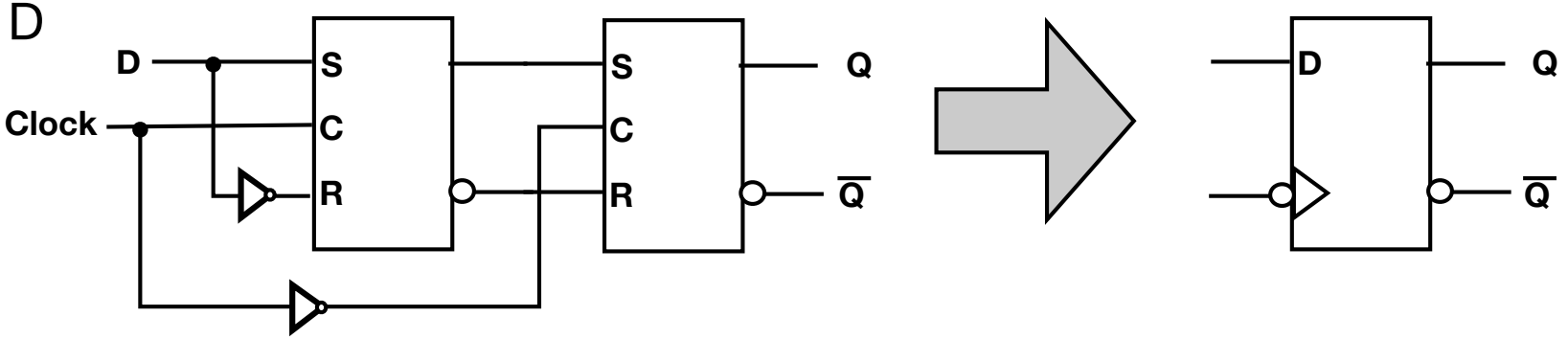




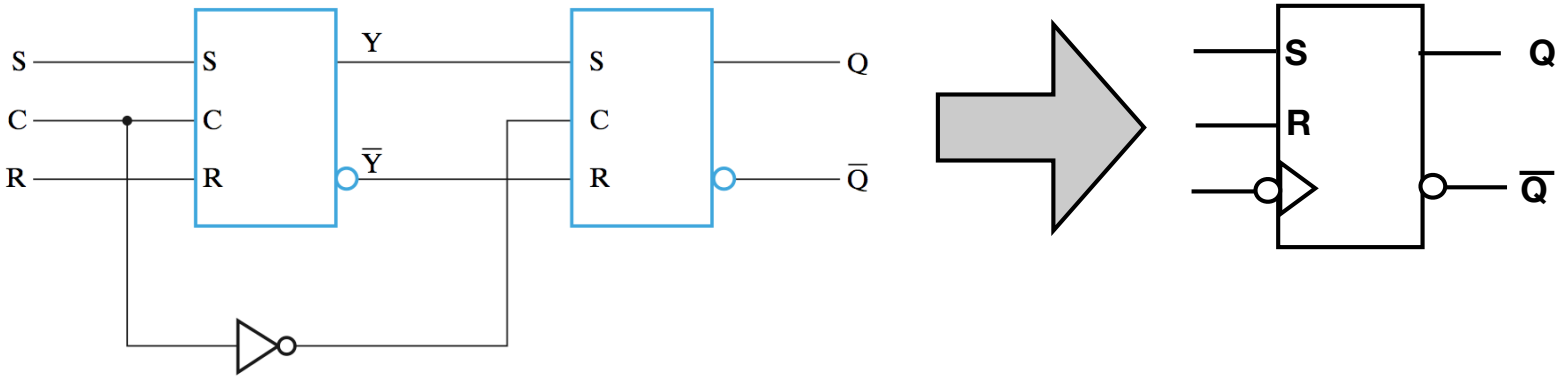
D(t)	Q(t+1)
0	0
1	1

Circuit Diagram for Flip-Flops

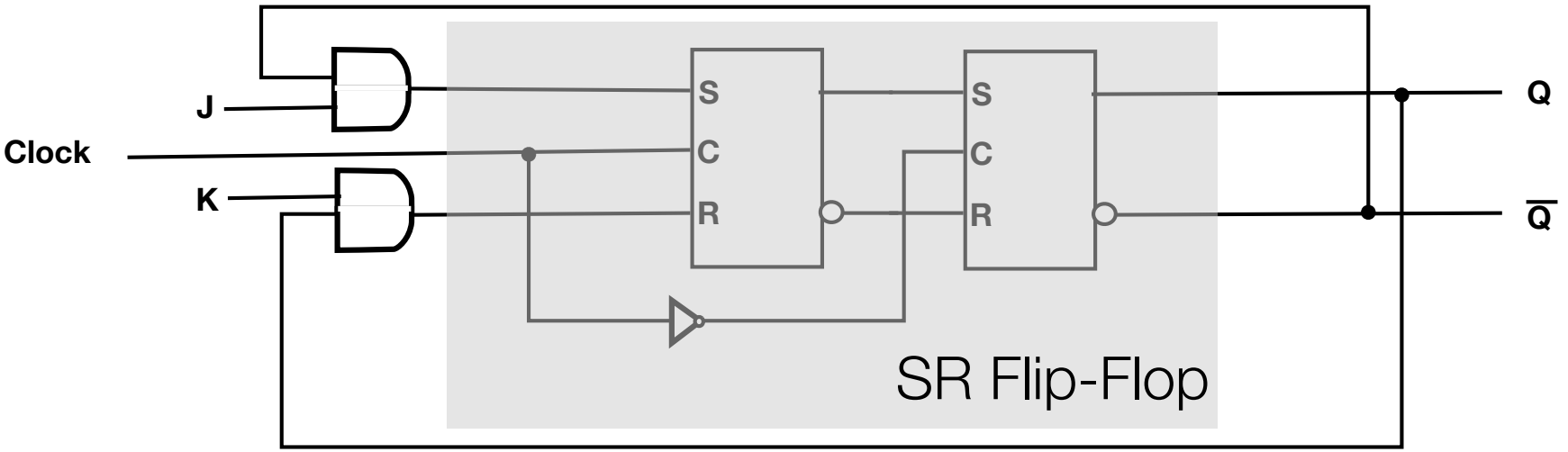
- D



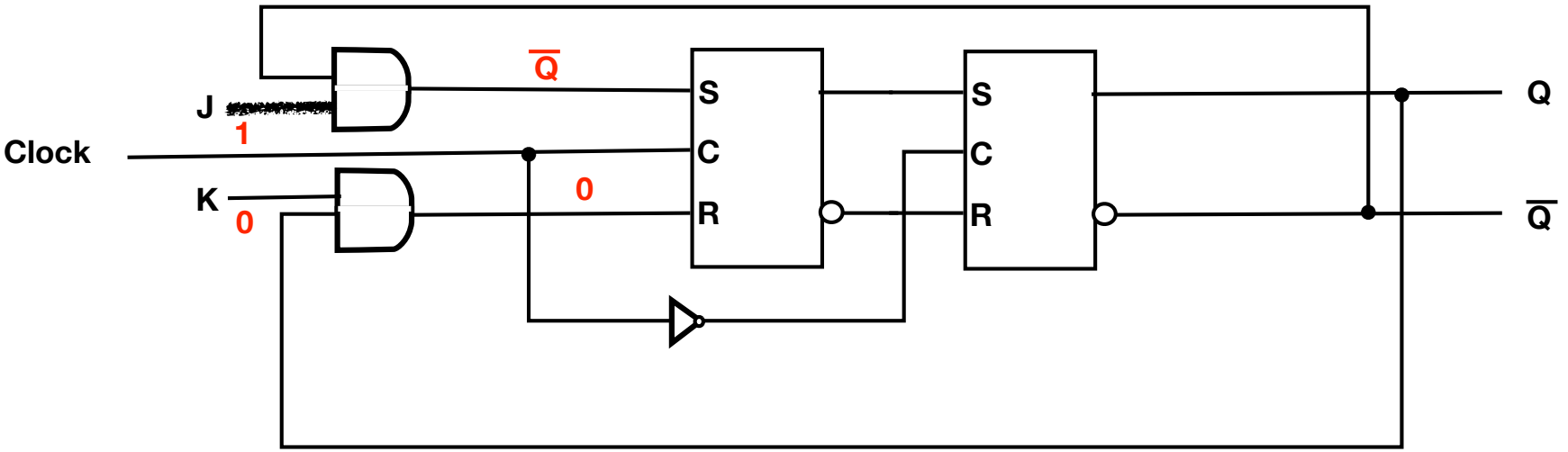
- SR



JK Flip Flop from SR F.F.

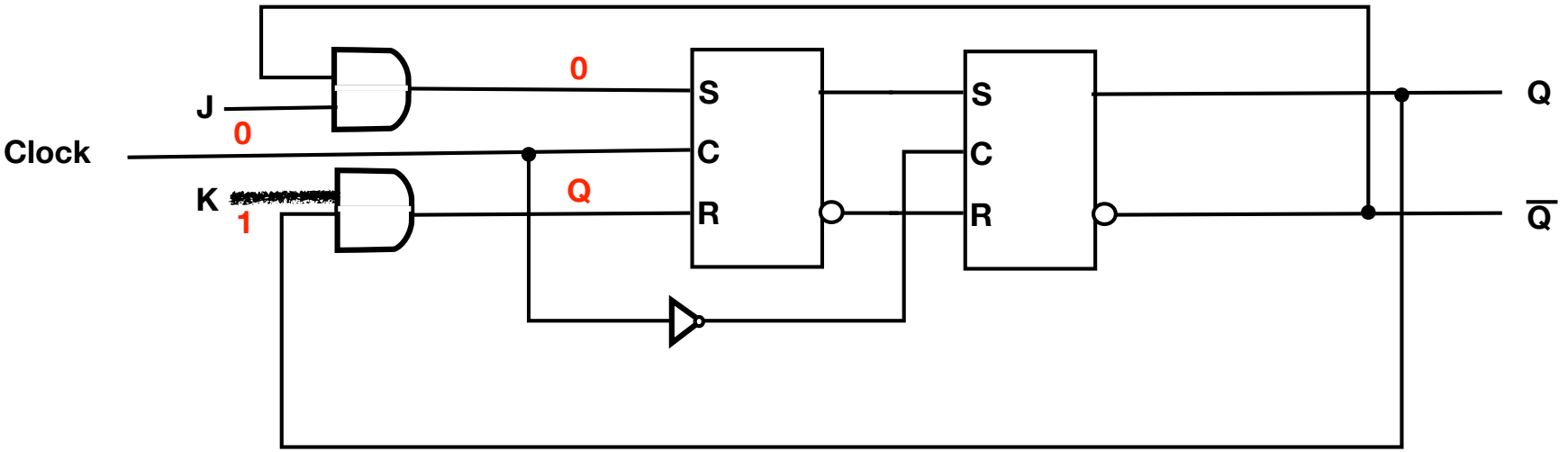


JK Flip Flop from SR F.F.



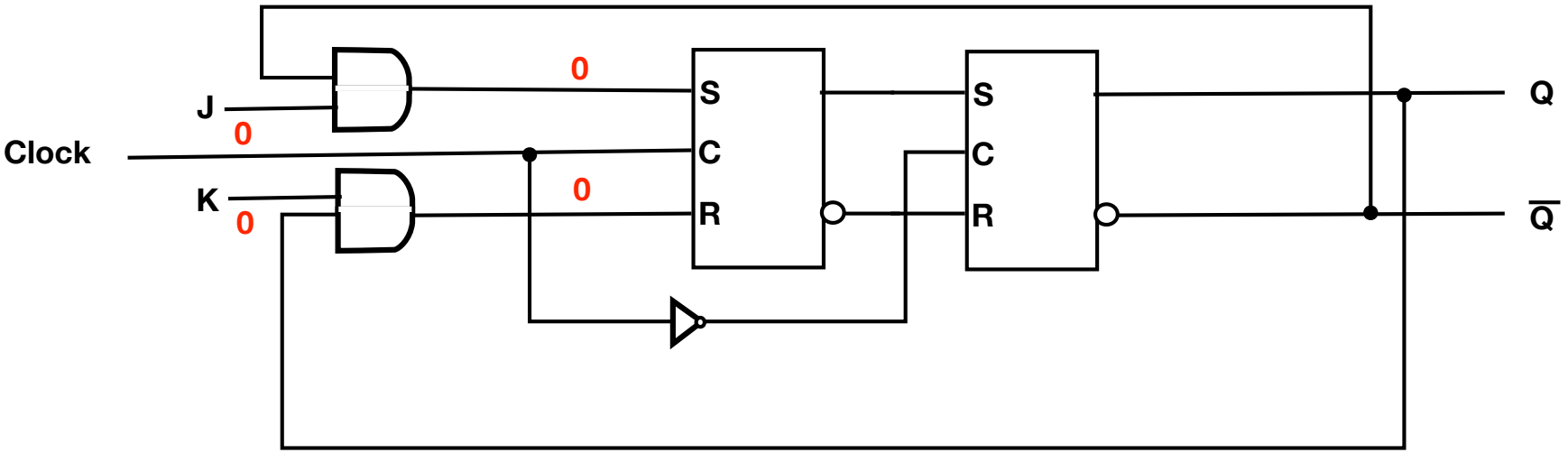
- J=1, K=0
  - Q(t-1)=1, $\overline{Q}$(t-1)=0 SR F.F. fed S=0, R=0, stays the same: Q(t)=1
  - Q(t-1)=0, $\overline{Q}$(t-1)=1, SR F.F. fed S=1, R=0, set: Q(t)=1
  - So regardless of Q(t-1) value, J=1, K=0 **sets** the JK F.F.: Q(t) = 1

JK Flip Flop from SR F.F.



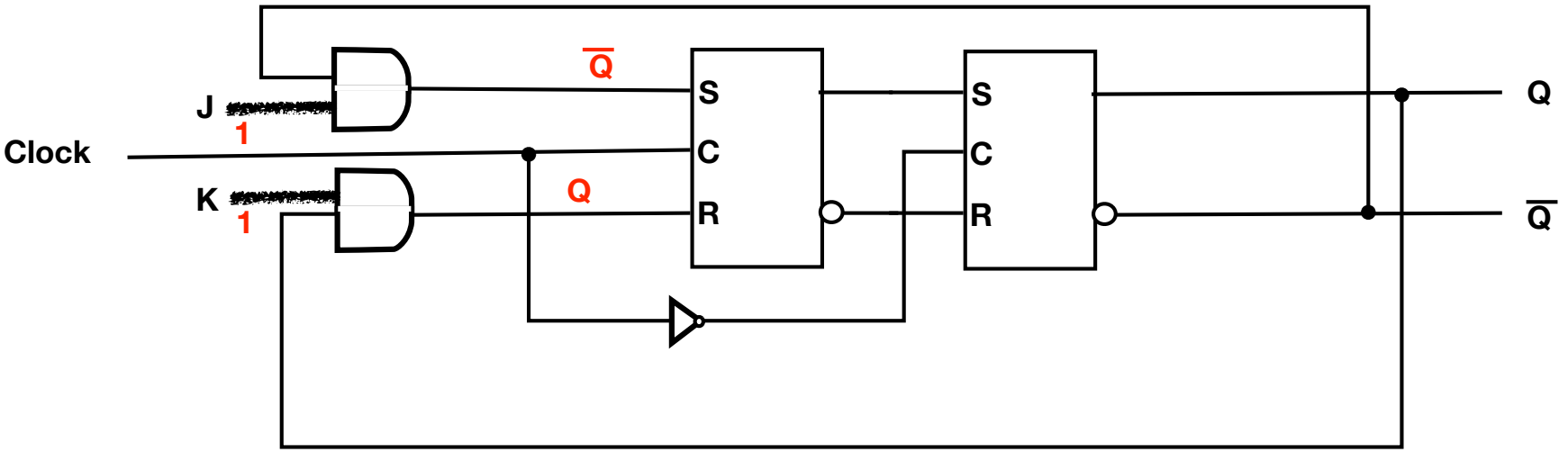
- J=0, K=1
  - Q(t-1)=1, SR F.F. fed S=0, R=1, reset: Q(t)=0
  - Q(t-1)=0, SR F.F. fed S=0, R=0, stay same: Q(t) = 0
  - So regardless of Q(t-1) value, J=0, K=1 **resets** the JK F.F.: Q(t) = 0

JK Flip Flop from SR F.F.



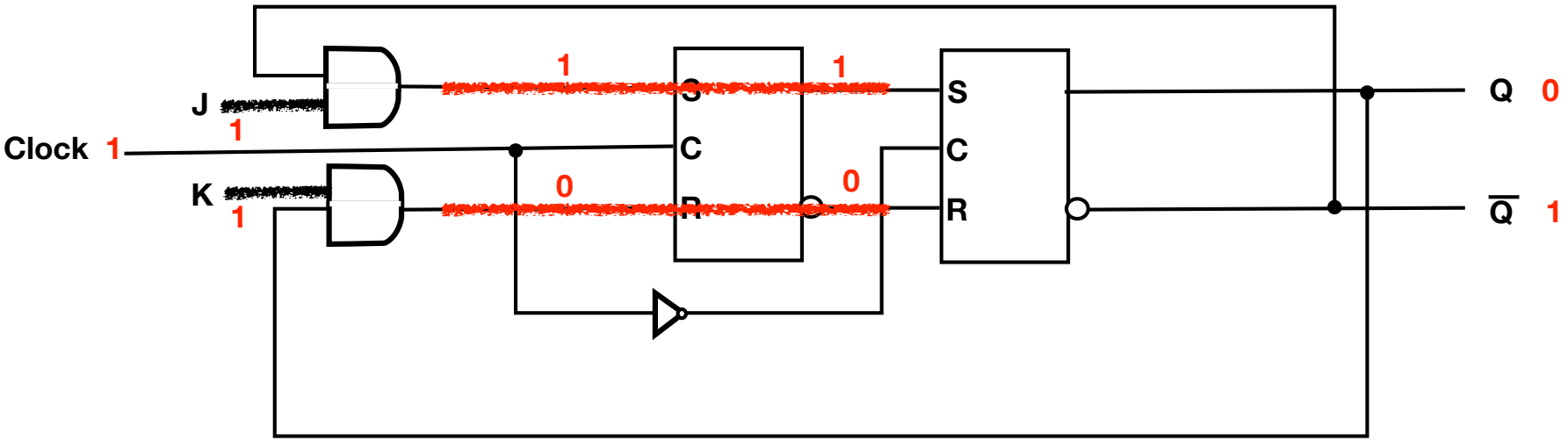
- J=0, K=0
  - S=0, R=0, regardless of J,K values, reset: Q(t)=Q(t-1)
  - J=0, K=0, F.F. **stays same**

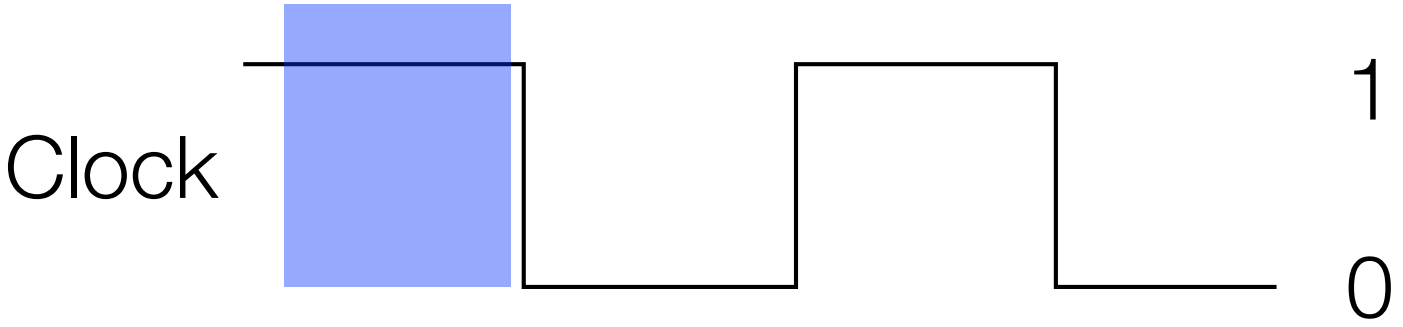
JK Flip Flop from SR F.F.



- J=1, K=1
  - Q(t-1)=1, $\overline{Q}$(t-1)=0 SR F.F. fed S=0, R=1, reset: Q(t)=0
  - Q(t-1)=0, $\overline{Q}$(t-1)=1, SR F.F. fed S=1, R=0, set: Q(t)=1
  - So J=1, K=1 **compliments** the JK F.F.: Q(t) = $\overline{Q}$(t-1)

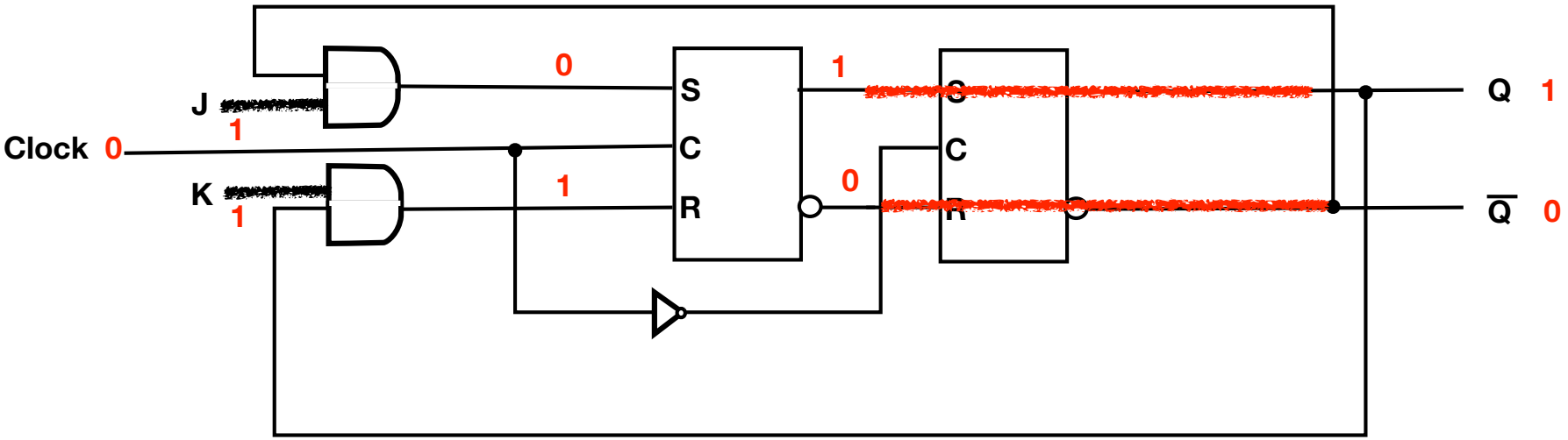
JK Flip Flop with 1-1 inputs (starting with Q=0)

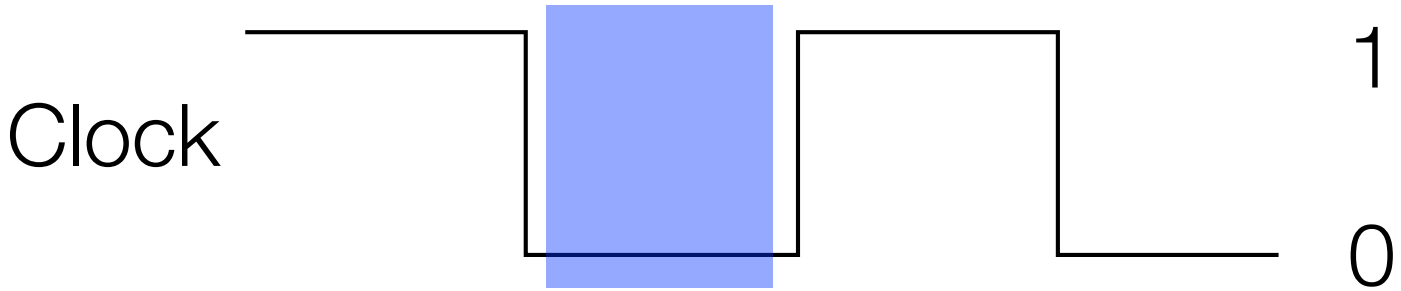




- C=1:
  - Q’ value sent to J, Q value sent to K
  - Leader latch set to compliment values
  - Cannot yet flow through to follower latch

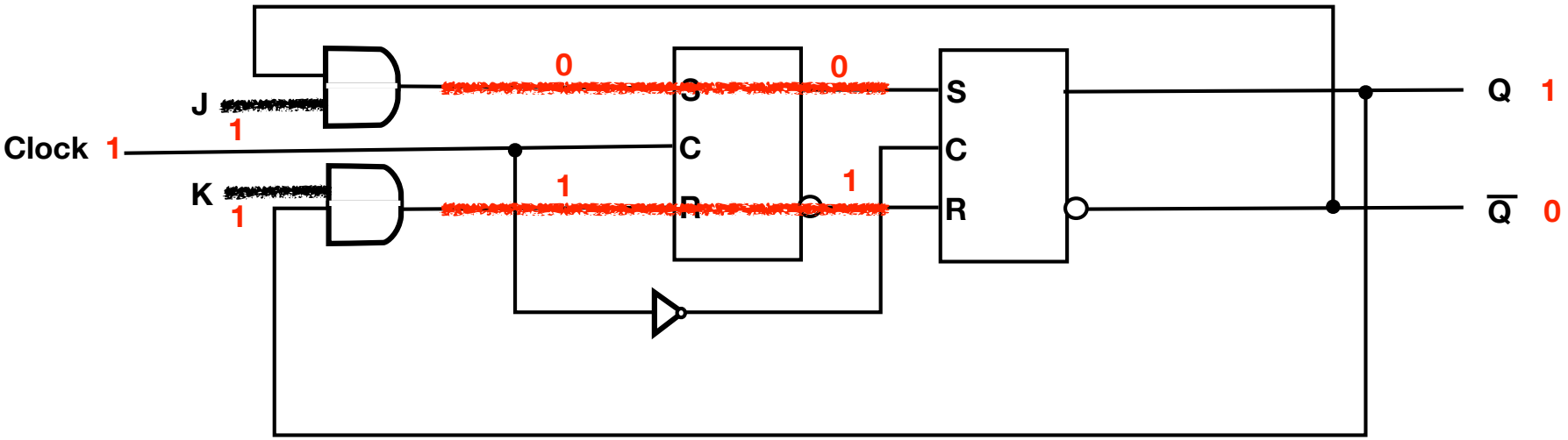
JK Flip Flop with 1-1 inputs (starting with Q=0)

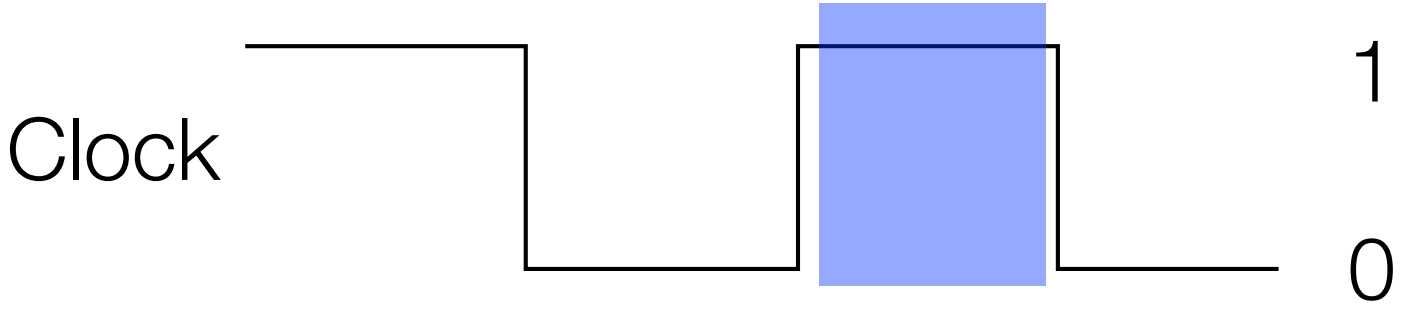




- C=0:
  - Compliment values flow from Leader to Follower latches
  - Output values have been complimented (start of new cycle)
  - Leader inputs complimented, but latch is not enabled yet

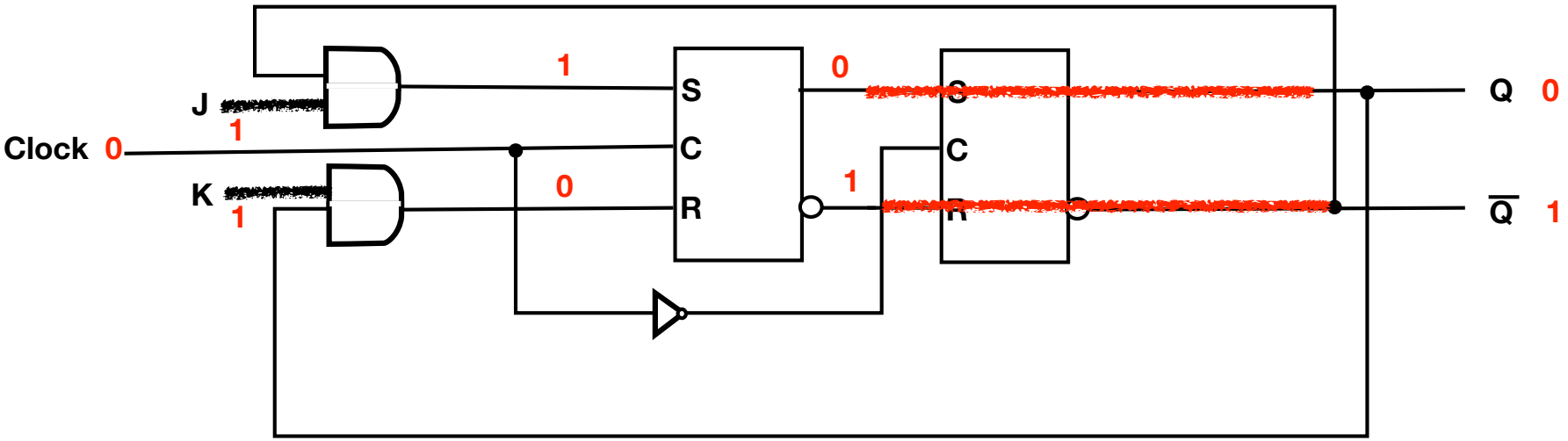
JK Flip Flop with 1-1 inputs (starting with Q=0)

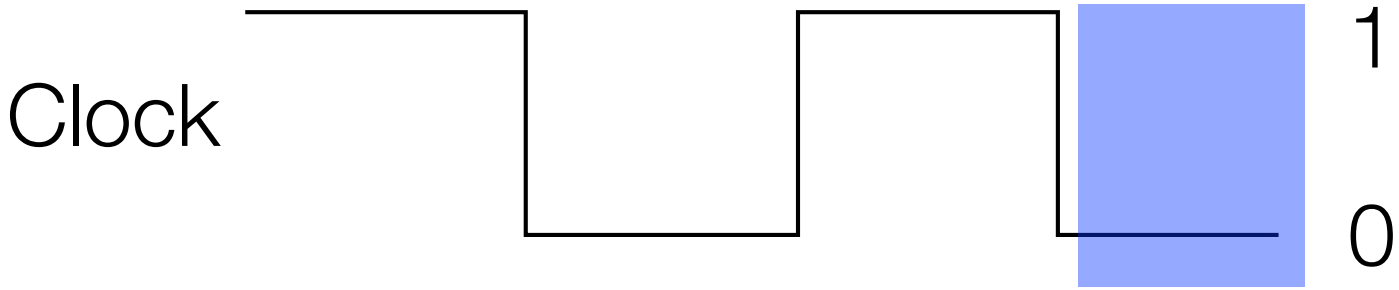




- C=1:
  - Q’ value sent to J, Q value sent to K
  - Leader latch set to compliment values
  - Cannot yet flow through to follower latch

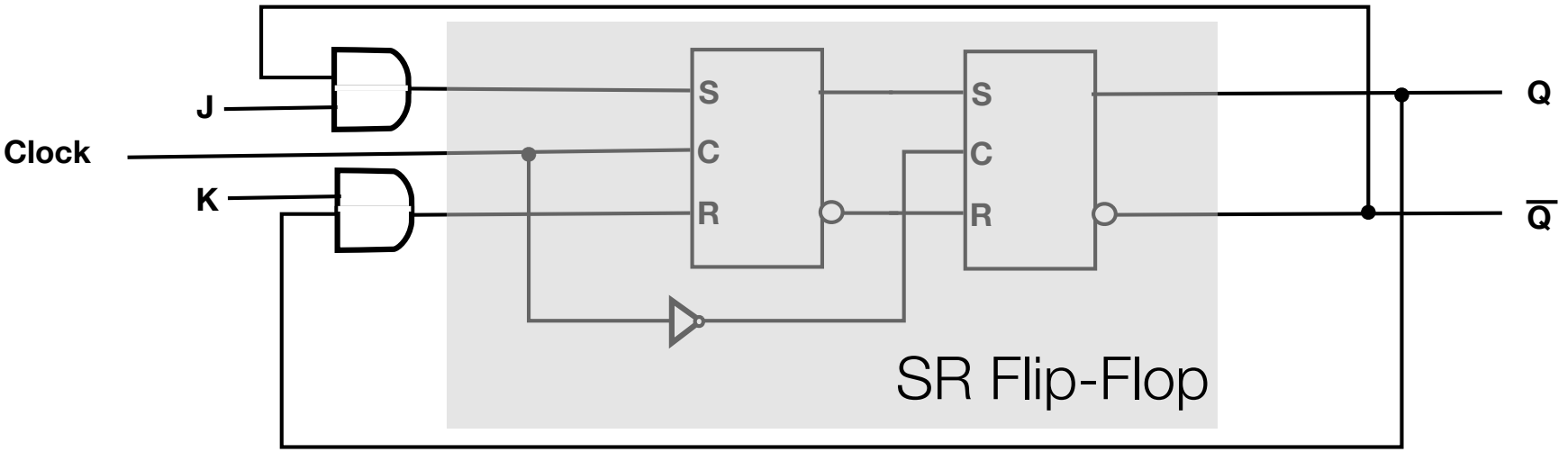
JK Flip Flop with 1-1 inputs (starting with Q=0)





- C=0:
  - Compliment values flow from Leader to Follower
  - Output values have been complimented (start of new cycle)
  - Leader inputs complimented, but latch is not enabled yet

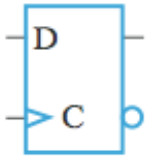
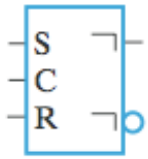
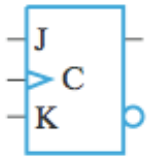
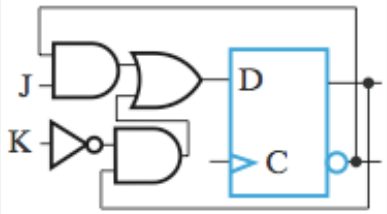
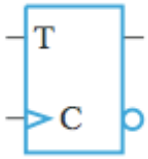
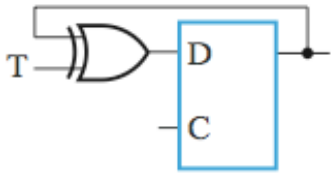
JK Flip Flop from SR F.F.



JK Flip-Flop Characteristic Table

J(t)	K(t)	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q}$(t)

Two more flip-flops: JK and T flip-flops

Type	Symbol	Logic Diagrams	Characteristic Table	Characteristic Equation	Excitation Table
D		See Figure 5-12	**D** | **Q(t+1)** | **Operation** 0 | 0 | Reset 1 | 1 | Set	$Q(t+1) = D(t)$	**Q(t+1)** | **D** | **Operation** 0 | 0 | Reset 1 | 1 | Set
SR		See Figure 5-9	**S** | **R** | **Q(t+1)** | **Operation** 0 | 0 | $Q(t)$ | No change 0 | 1 | 0 | Reset 1 | 0 | 1 | Set 1 | 1 | ? | Undefined	$Q(t+1) = S(t) + \overline{R}(t)\,Q(t)$	**Q(t)** | **Q(t+1)** | **S** | **R** | **Operation** 0 | 0 | 0 | X | No change 0 | 1 | 1 | 0 | Set 1 | 0 | 0 | 1 | Reset 1 | 1 | X | 0 | No change
JK			**J** | **K** | **Q(t+1)** | **Operation** 0 | 0 | $Q(t)$ | No change 0 | 1 | 0 | Reset 1 | 0 | 1 | Set 1 | 1 | $\overline{Q}(t)$ | Complement	$Q(t+1) = J(t)\,\overline{Q}(t) + \overline{K}(t)\,Q(t)$	**Q(t)** | **Q(t+1)** | **J** | **K** | **Operation** 0 | 0 | 0 | X | No change 0 | 1 | 1 | X | Set 1 | 0 | X | 1 | Reset 1 | 1 | X | 0 | No Change
T			**T** | **Q(t+1)** | **Operation** 0 | $Q(t)$ | No change 1 | $\overline{Q}(t)$ | Complement	$Q(t+1) = T(t) \oplus Q(t)$	**Q(t+1)** | **T** | **Operation** $Q(t)$ | 0 | No change $\overline{Q}(t)$ | 1 | Complement

Flip-Flop Trigger Types

Edge v. Pulse triggered FF's

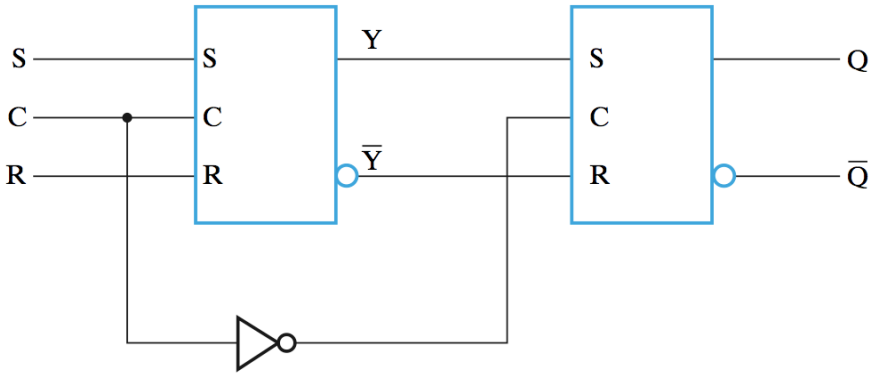
- **Edge triggered**: the output value of the FF depends only on the inputs at the instant in time when the clock transitions in value
- **Pulse triggered**: the output value of the FF can depend on the sequence of input values during the interim of the pulse
- Positive or Negative:
  - **Positive Edge**: output value depends on the input during the 0-to-1 transition
  - **Negative Edge**: output value depends on the input during the 1-to-0 transition
  - **Positive Pulse**: Pulse Triggered and Leader latch active when C=1
  - **Negative Pulse**: Pulse Triggered and Leader latch active when C=0

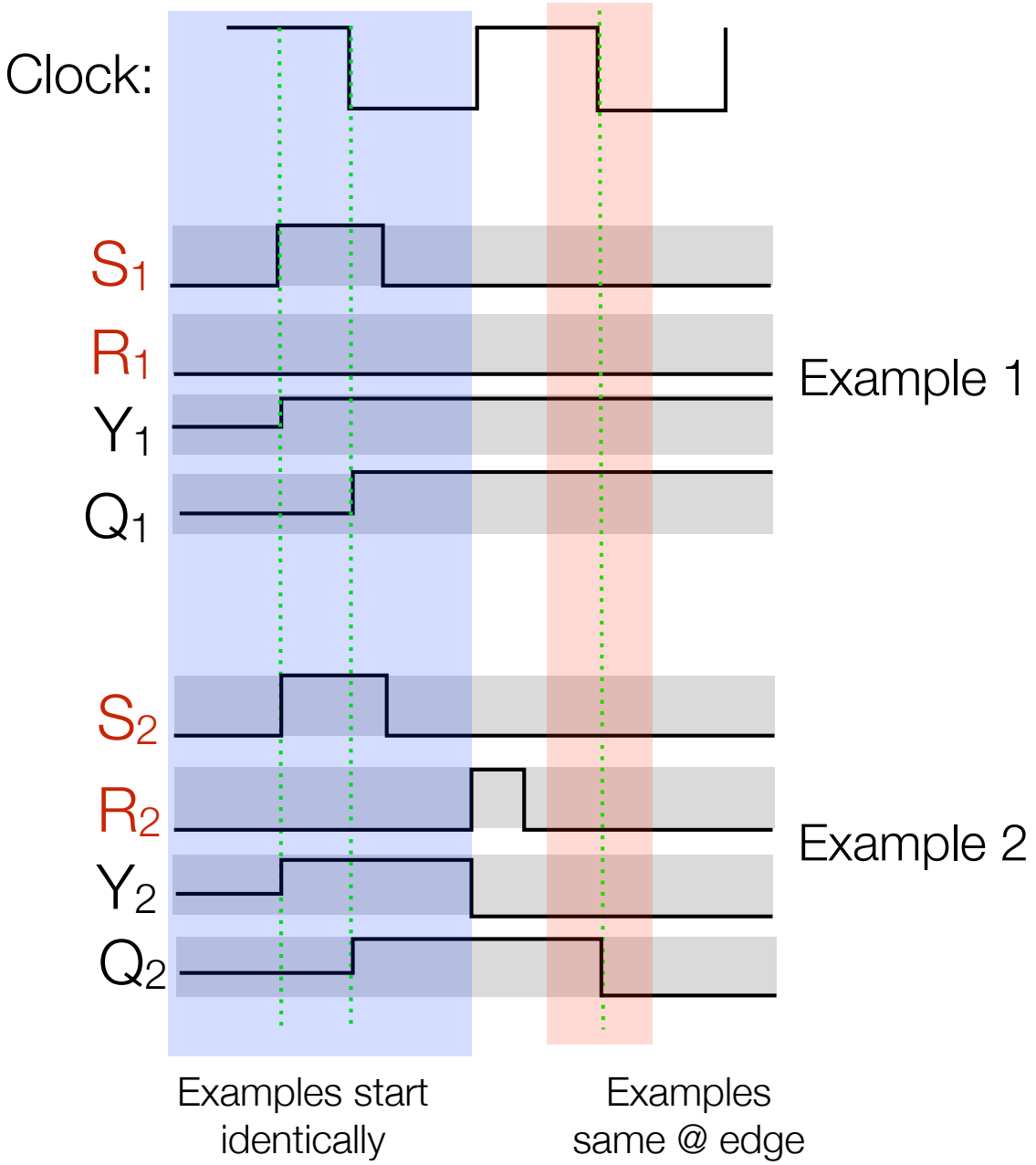
- D FF's are negative edge triggered (take on whatever value D is set to when clock "flops" from 1 to 0

- SR FF's are positive pulse triggered (e.g., S=1, R=0 at start of pulse, then switch to S=0, R=0 before end).

Pulse triggered example

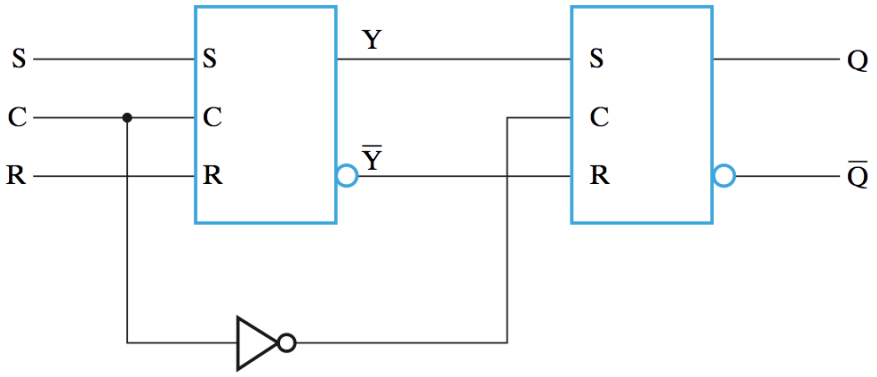
- Pulse triggered: cannot simply look @ inputs at pulse edge to determine value of output
  - Example:
    - S_1, R_1 are inputs for 1st example to pulse-triggered FF
    - S_2,R_2 are inputs to 2nd example

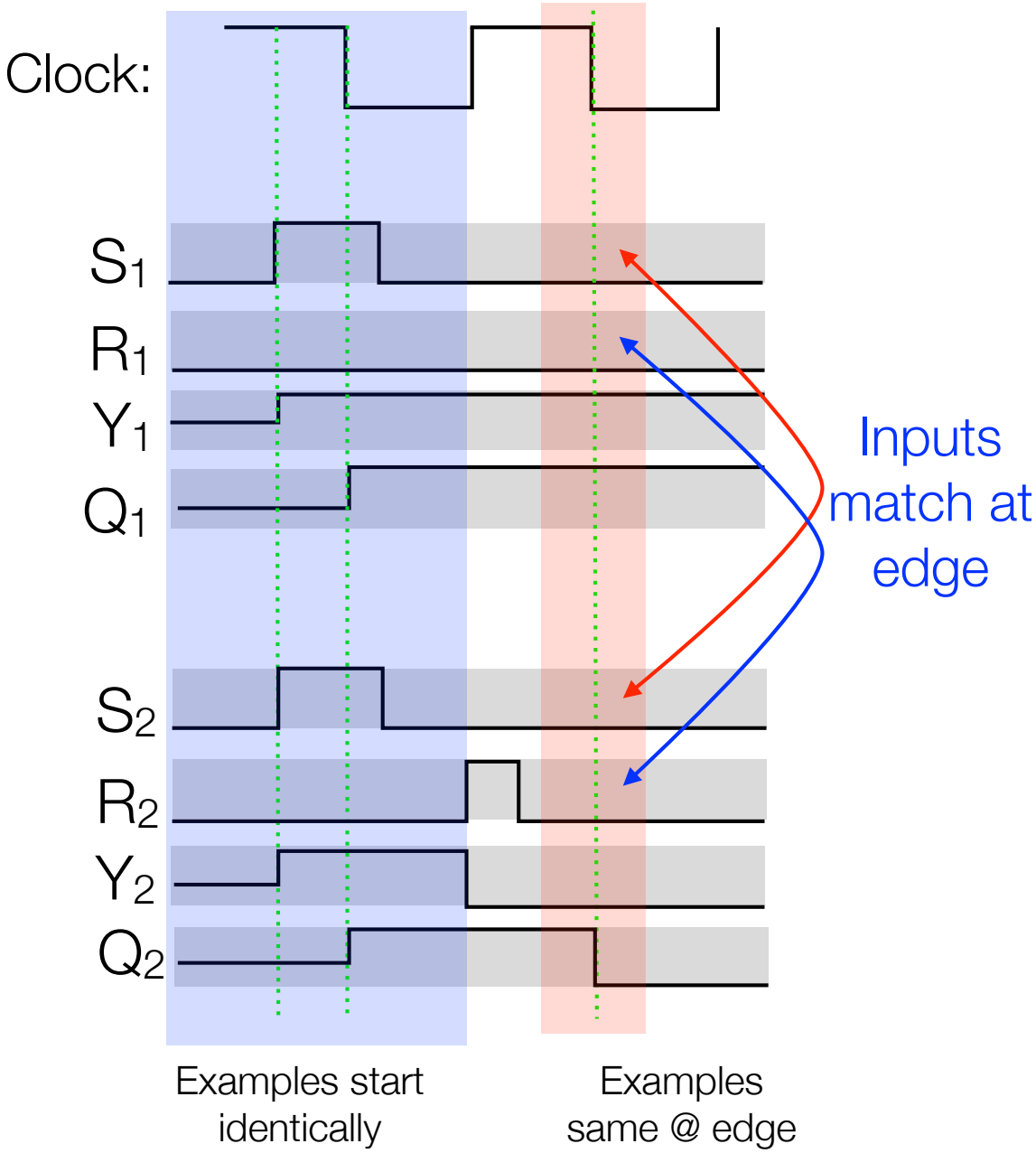




Pulse triggered example

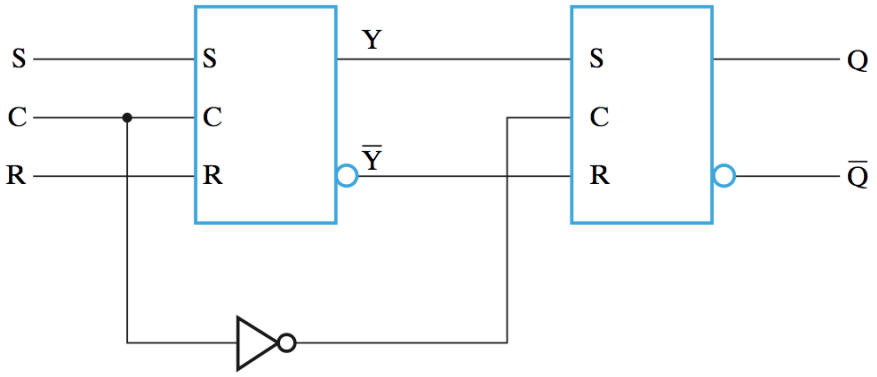
- Pulse triggered: cannot simply look @ inputs at pulse edge to determine value of output
  - Example:
    - S_1, R_1 are inputs for 1st example to pulse-triggered FF
    - S_2,R_2 are inputs to 2nd example

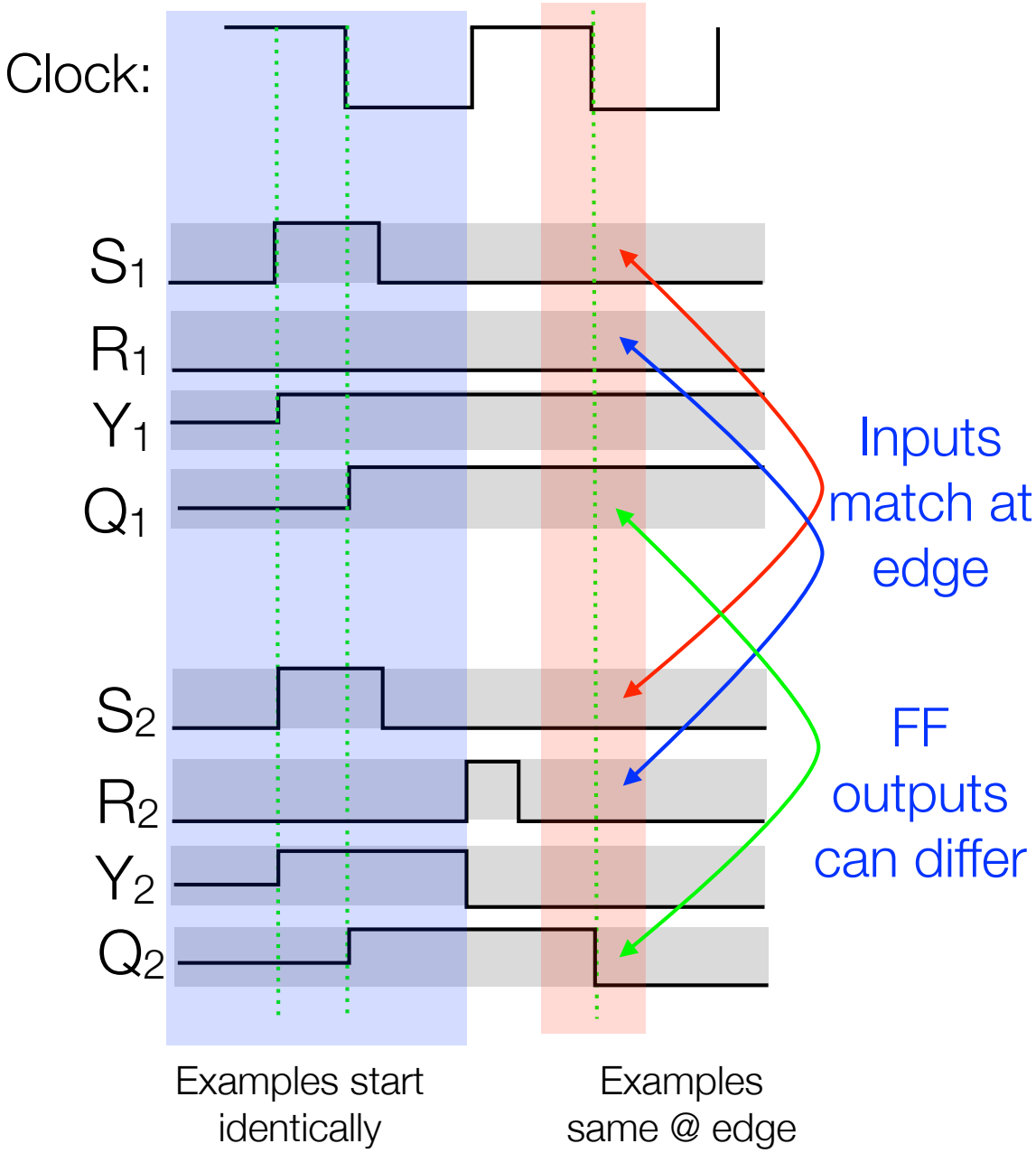




Pulse triggered example

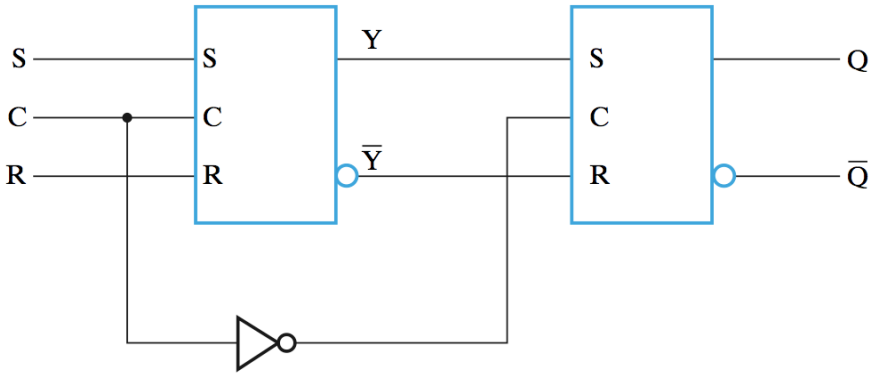
- Pulse triggered: cannot simply look @ inputs at pulse edge to determine value of output
  - Example:
    - S_1, R_1 are inputs for 1st example to pulse-triggered FF
    - S_2,R_2 are inputs to 2nd example

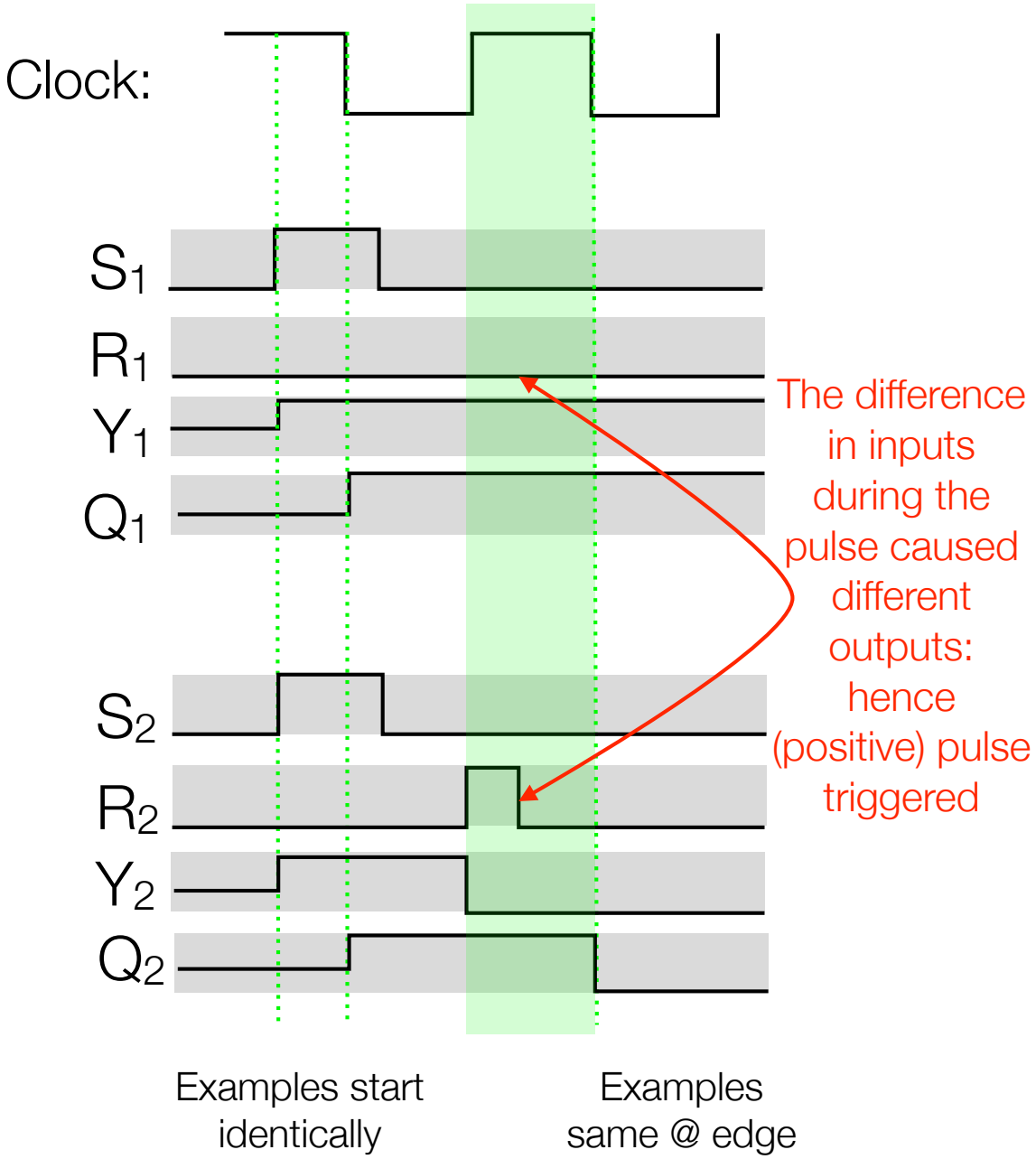




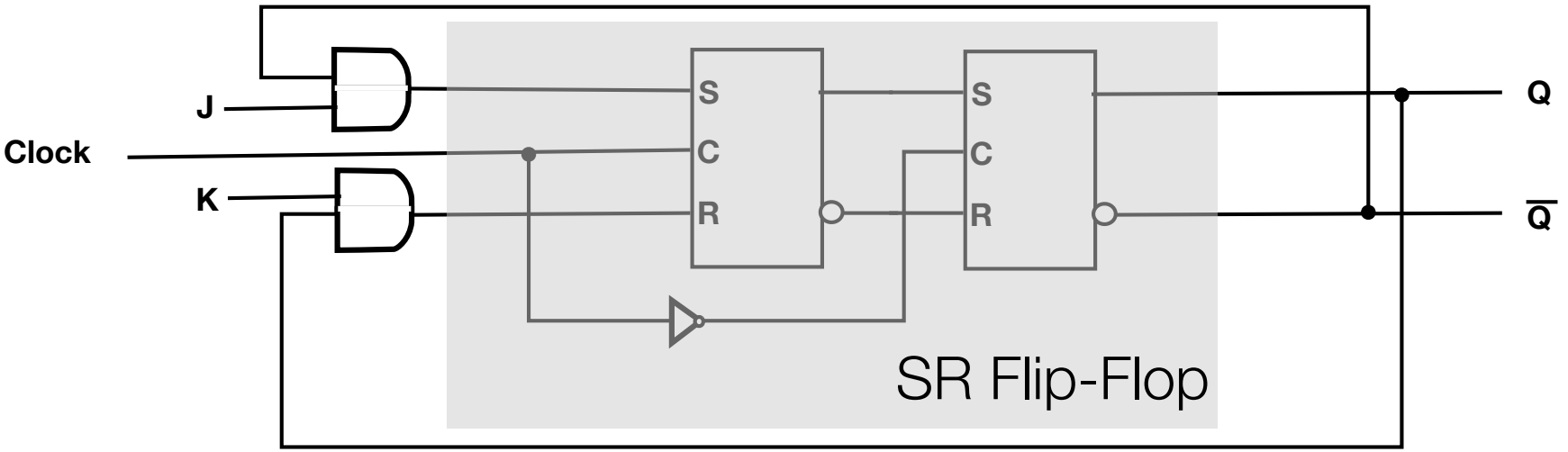
Pulse triggered example

- Pulse triggered: cannot simply look @ inputs at pulse edge to determine value of output
  - Example:
    - S_1, R_1 are inputs for 1st example to pulse-triggered FF
    - S_2,R_2 are inputs to 2nd example





JK Flip Flop from SR F.F.

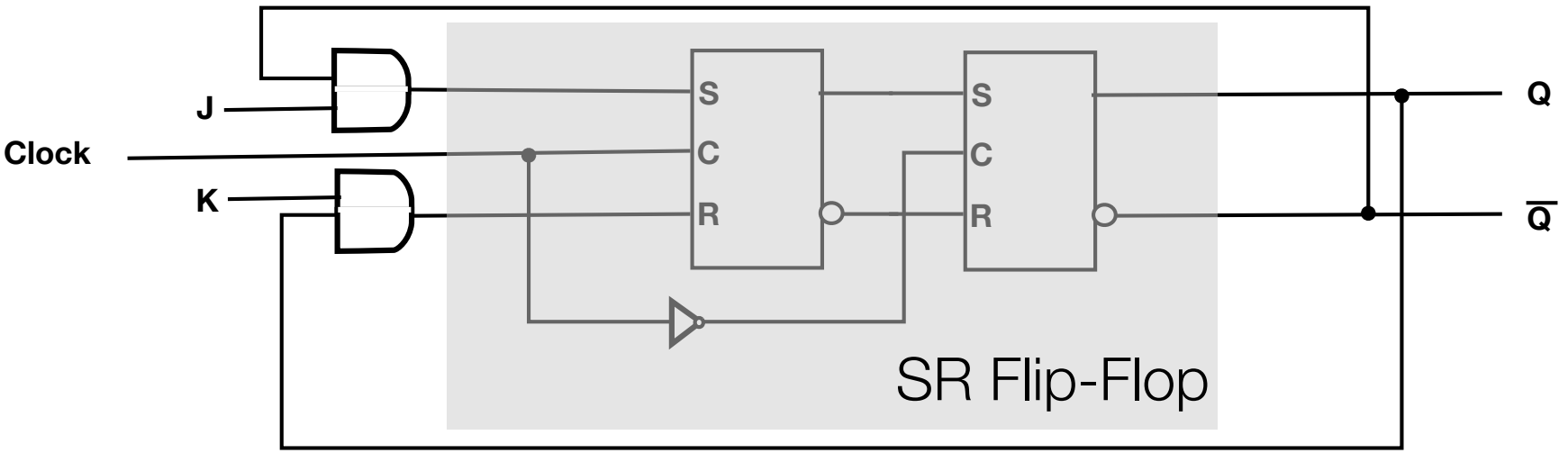


JK Flip-Flop Characteristic Table

J(t)	K(t)	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q}$(t)

Q: Edge or pulse triggered?

JK Flip Flop from SR F.F.



JK Flip-Flop Characteristic Table

J(t)	K(t)	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q}$(t)

Q: Edge or pulse triggered?

A: Pulse: during positive pulse, set JK=11 (leader is opposite follower), then change to JK=00 to hold this opposite value