Today

Sequential logic

- Latches
- Flip-flops
- Counters

Time

Until now: we have essentially ignored the issue of time

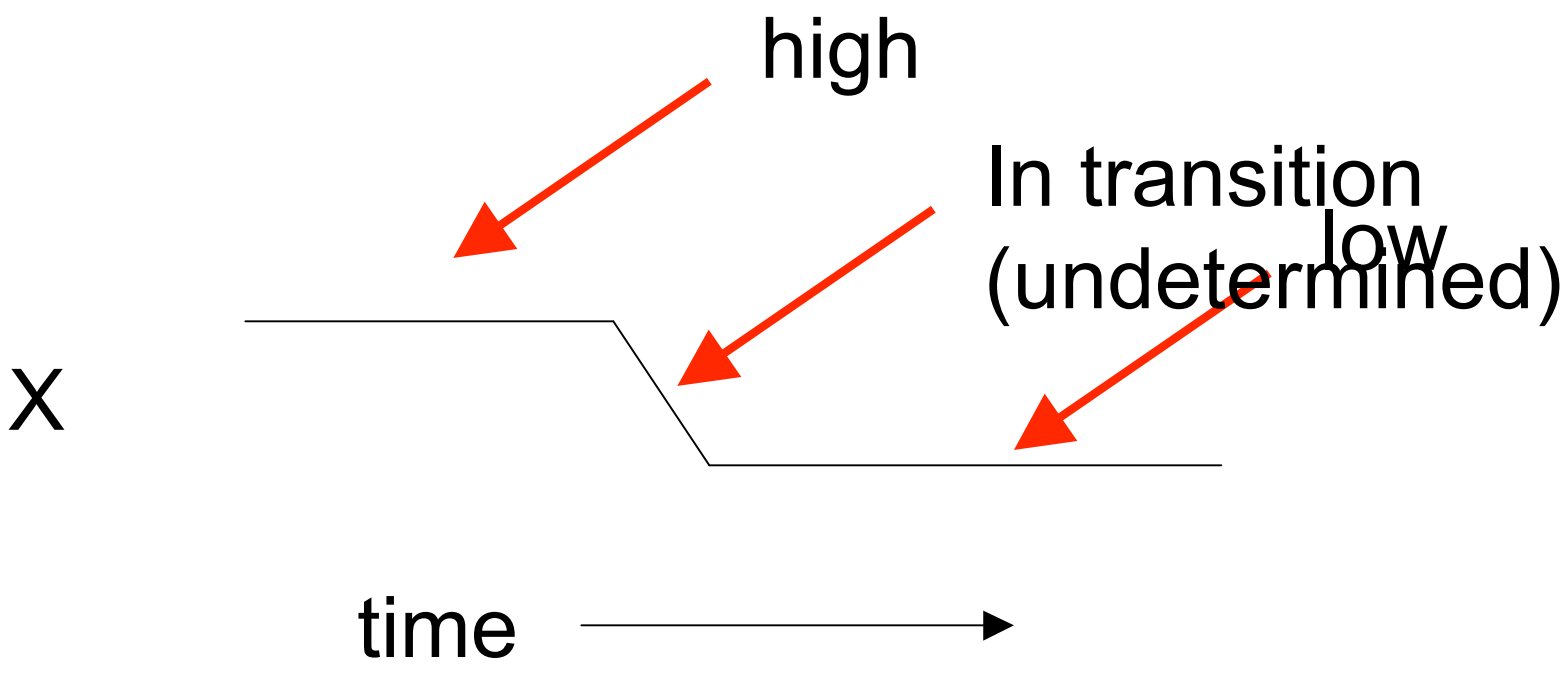
- We have assumed that our digital logic circuits perform their computations instantaneously
- Our digital logic circuits have been "stateless"
  - Once you present a new input, they forget everything about previous inputs

Time

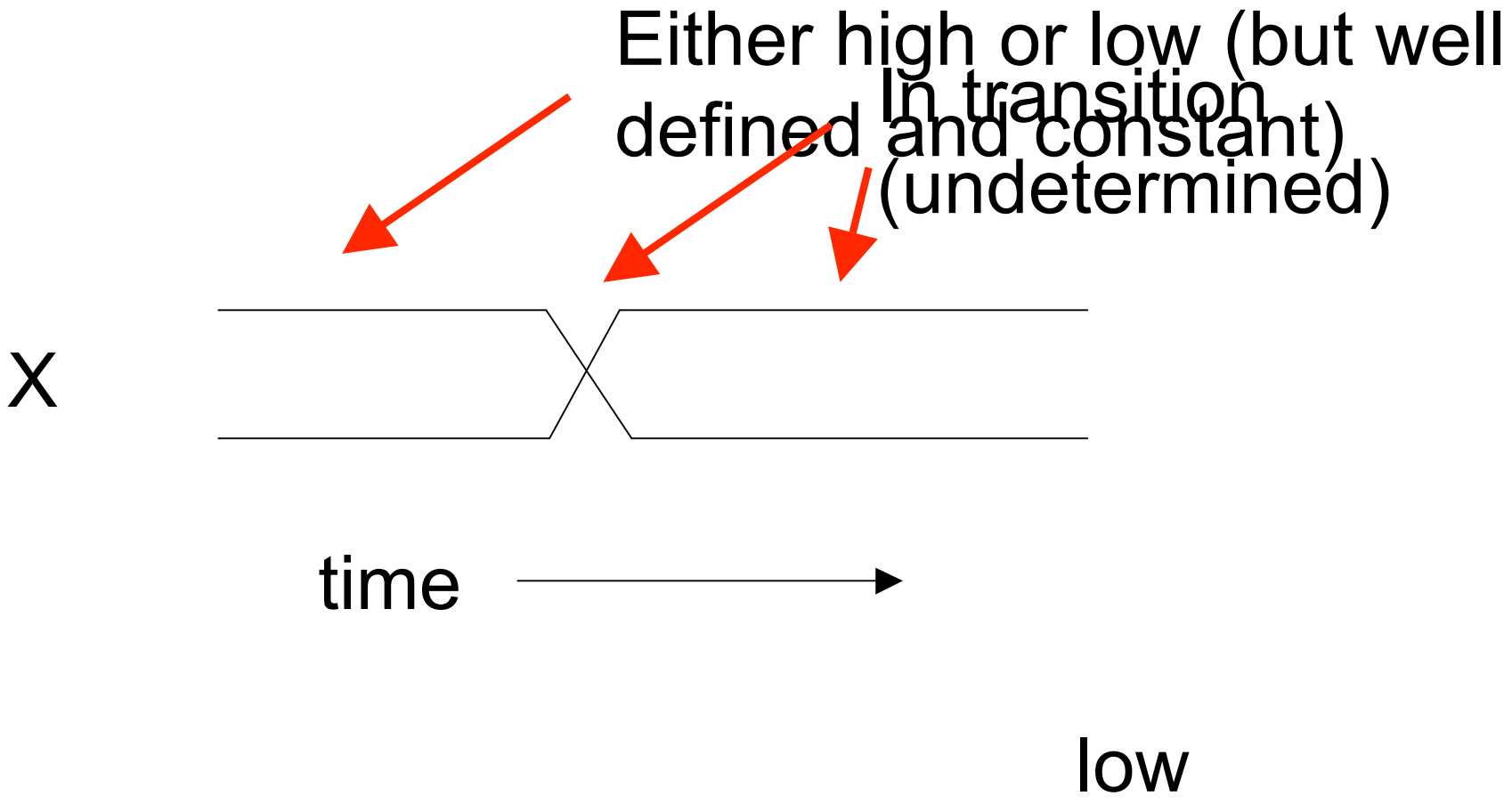
In reality, time is an important issue:

- Even our logic gates induce a small amount of delay (on the order of a few nanoseconds)
- For much of what we do – we actually want our circuits to have some form of memory

Timing Notation

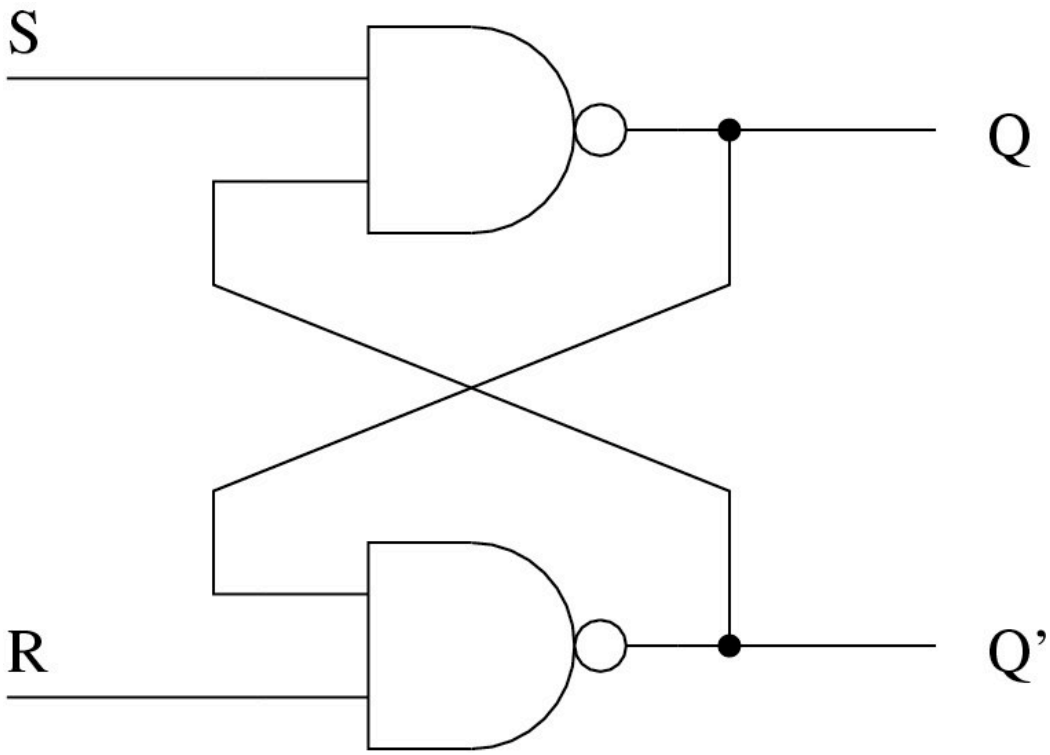


Timing Notation



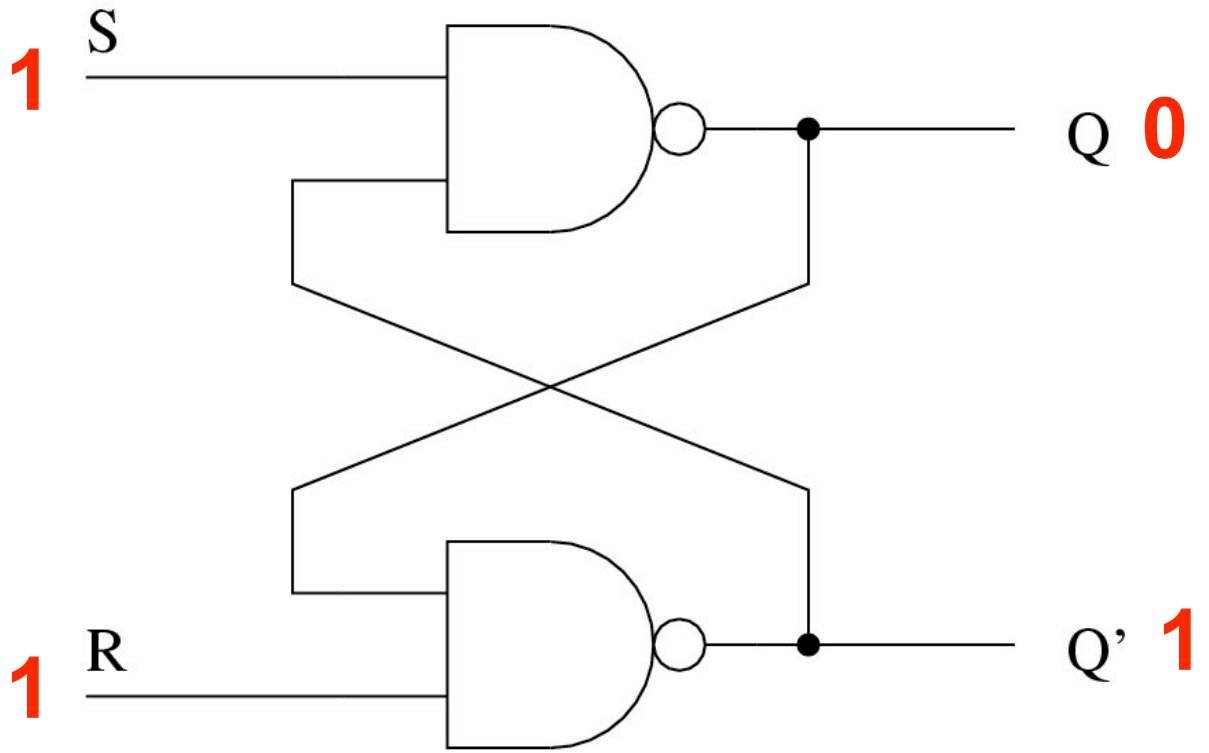
NAND Latch

What does this circuit do?



NAND Latch

Consider this initial state

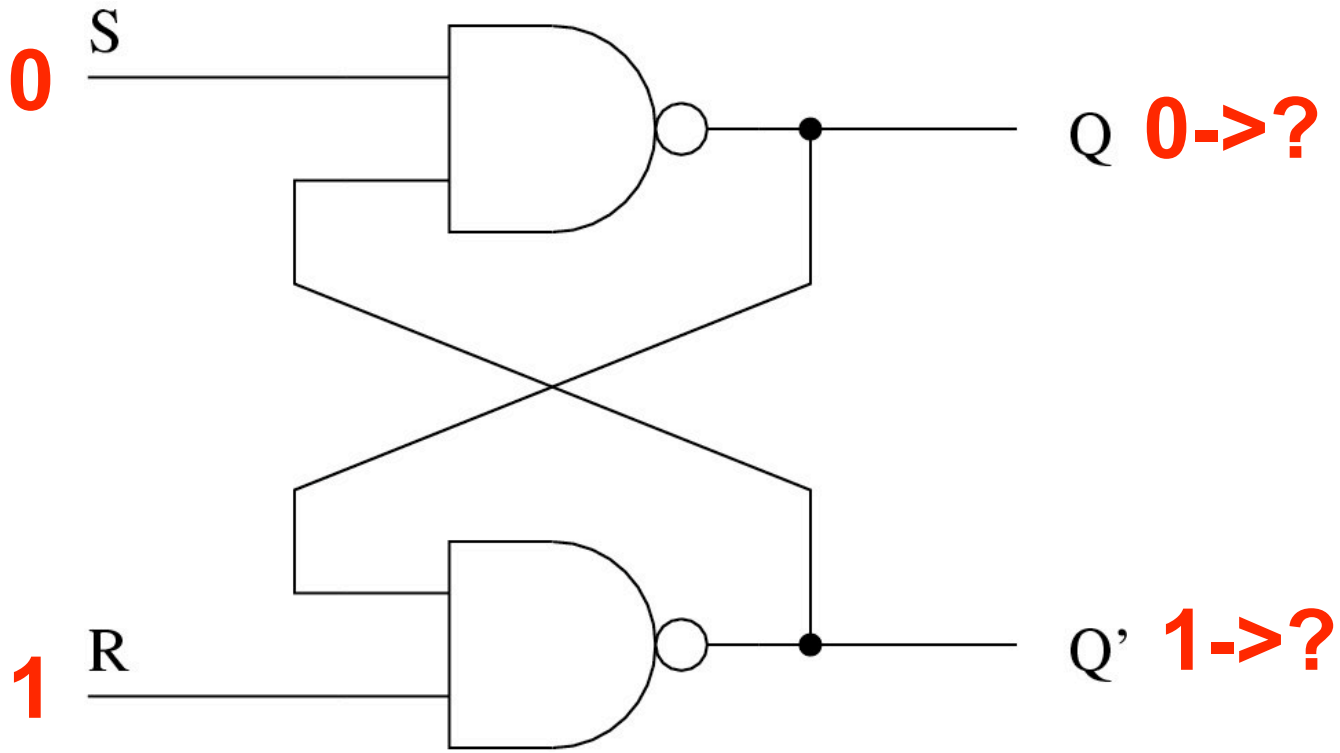


Is this a stable state?

Yes!

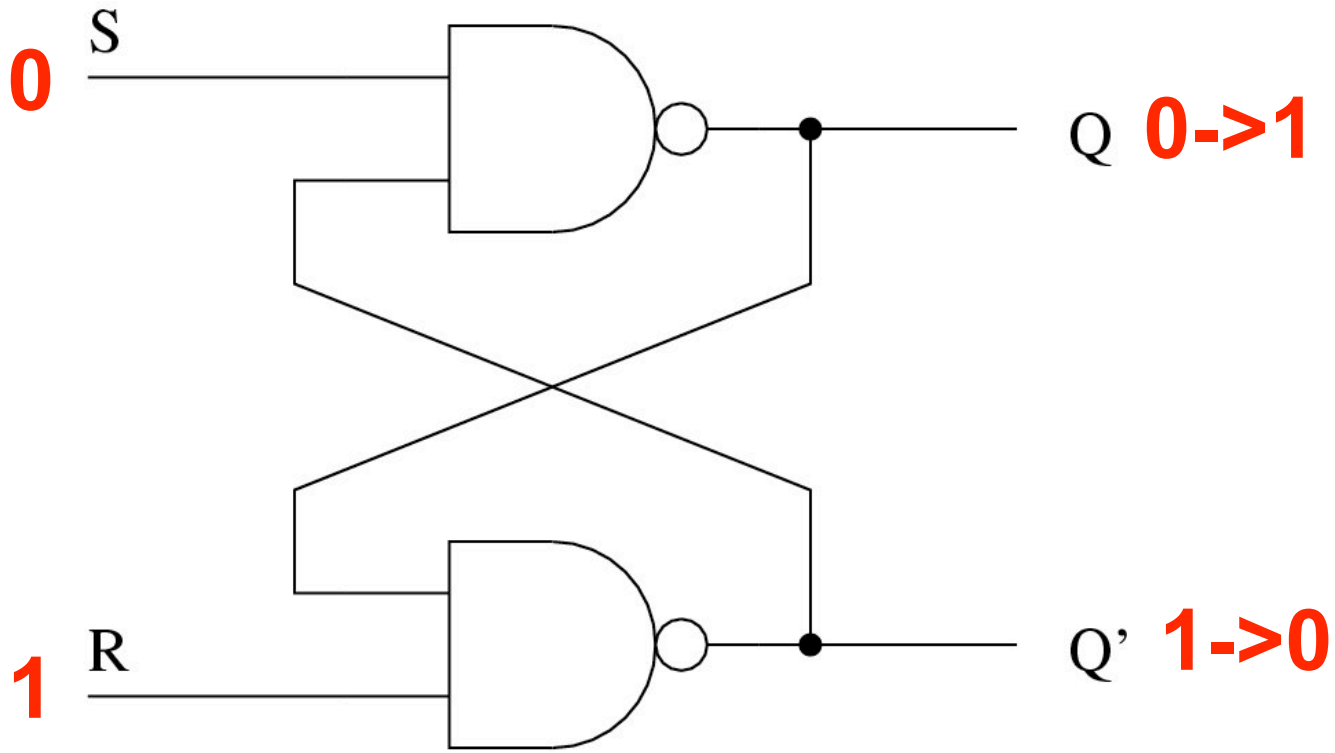
NAND Latch

What happens with S is set to 0?



NAND Latch

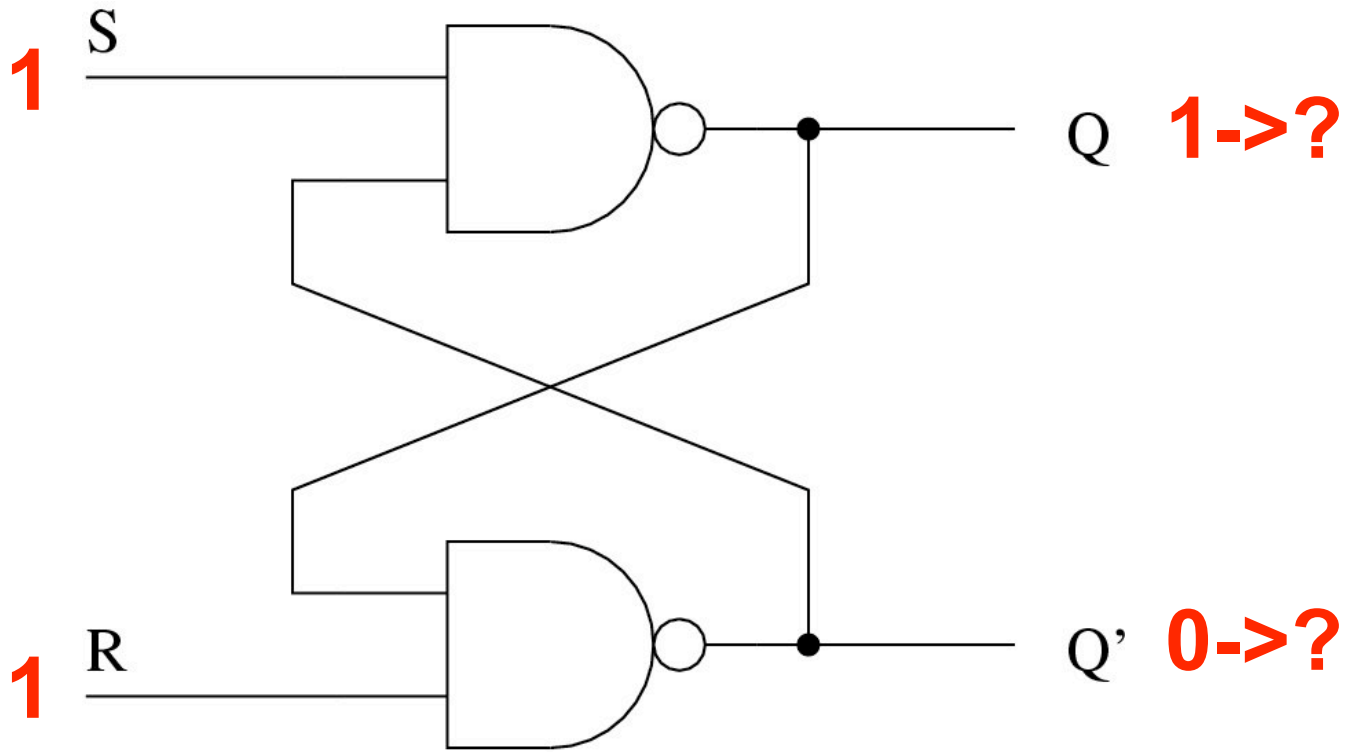
What happens with S is set to 0?



Q becomes 1 (thus S ‘sets’ Q)

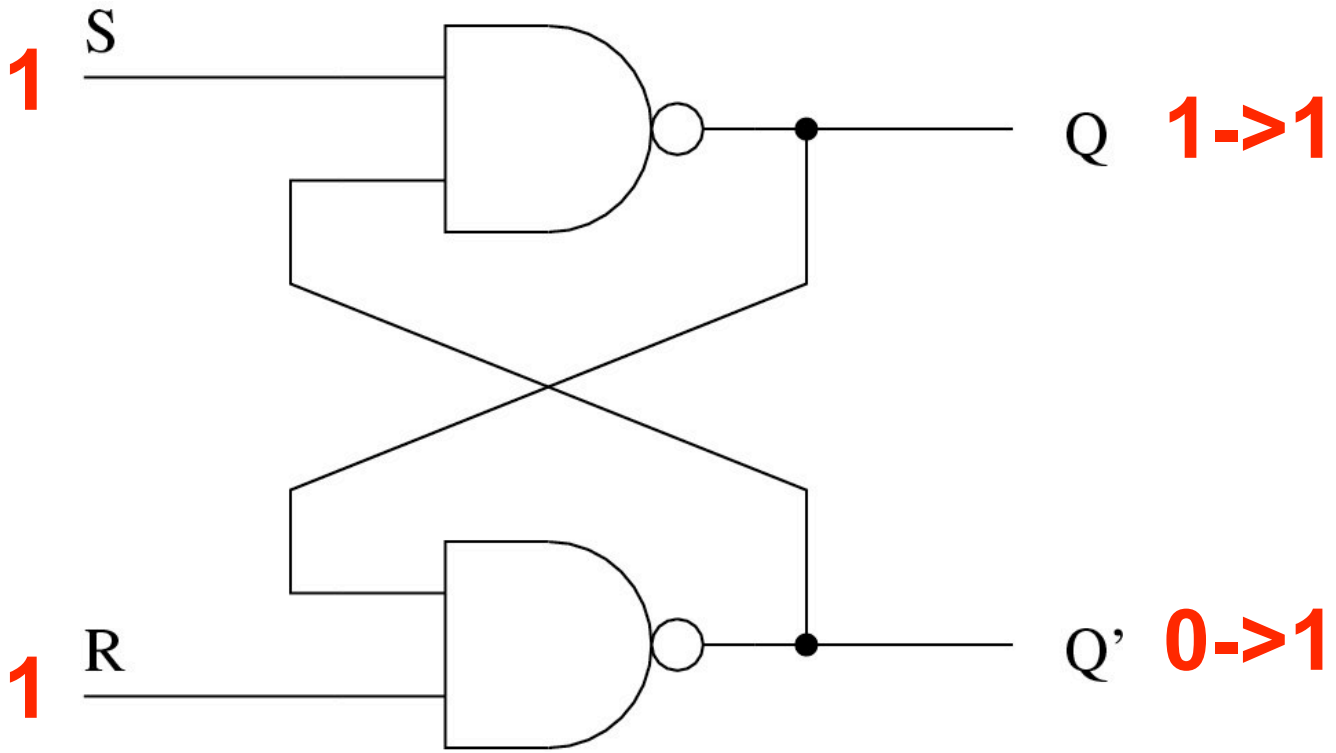
NAND Latch

Now S is set 1 – what happens?



NAND Latch

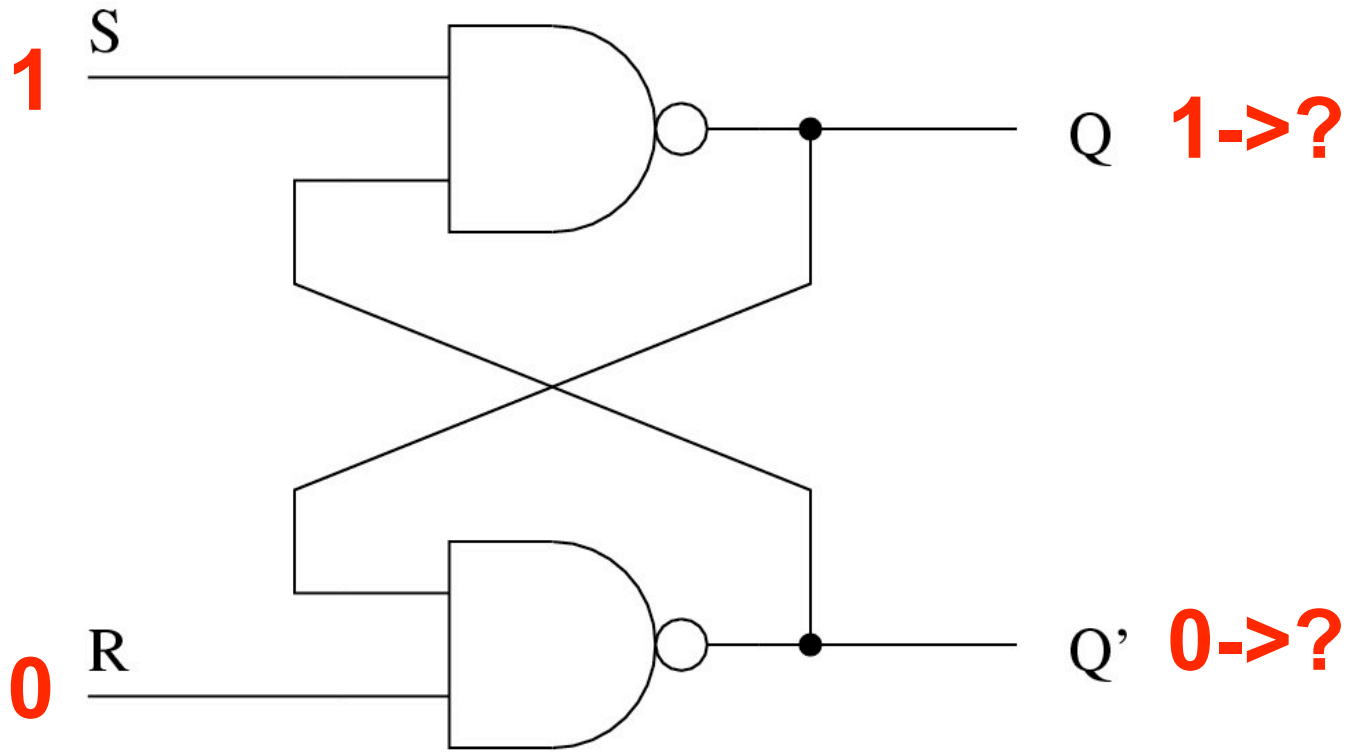
Q and Q' remain the same!



So Q and Q' retain a memory of past state!

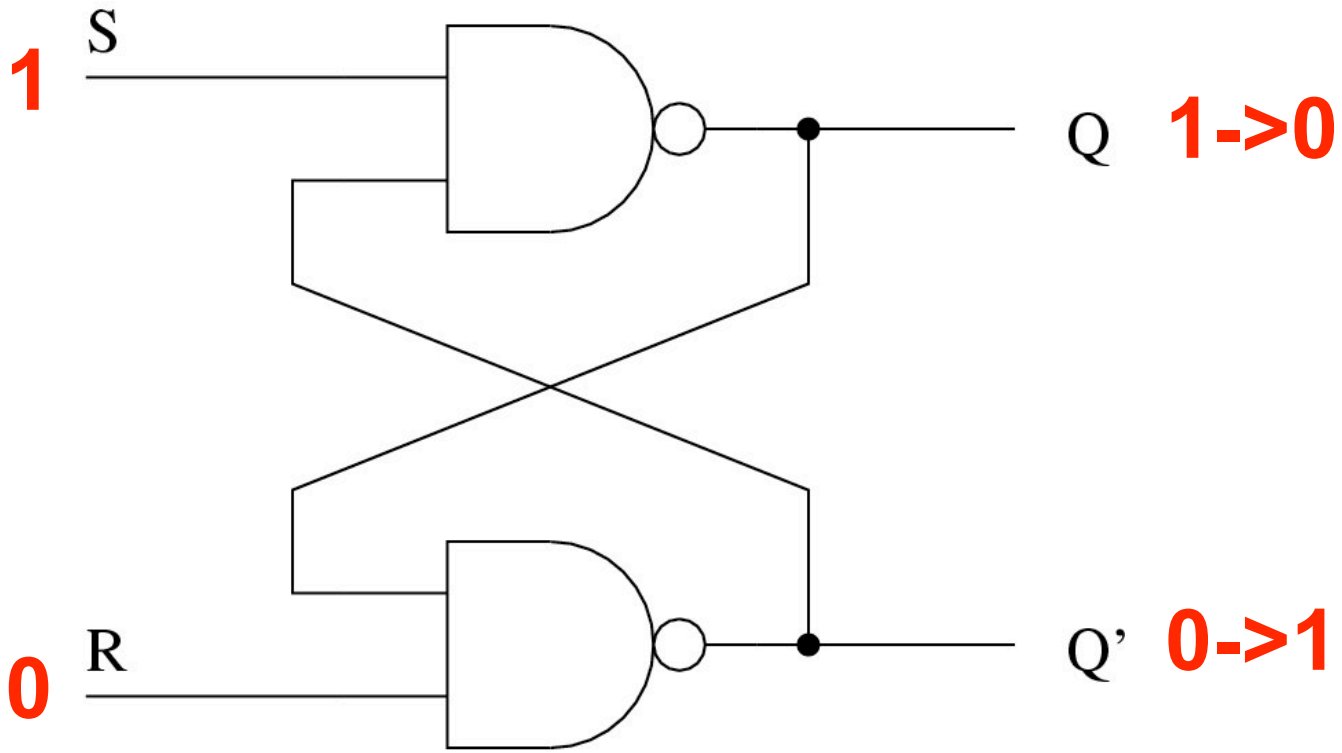
NAND Latch

Now set R to 0 – what happens?



NAND Latch

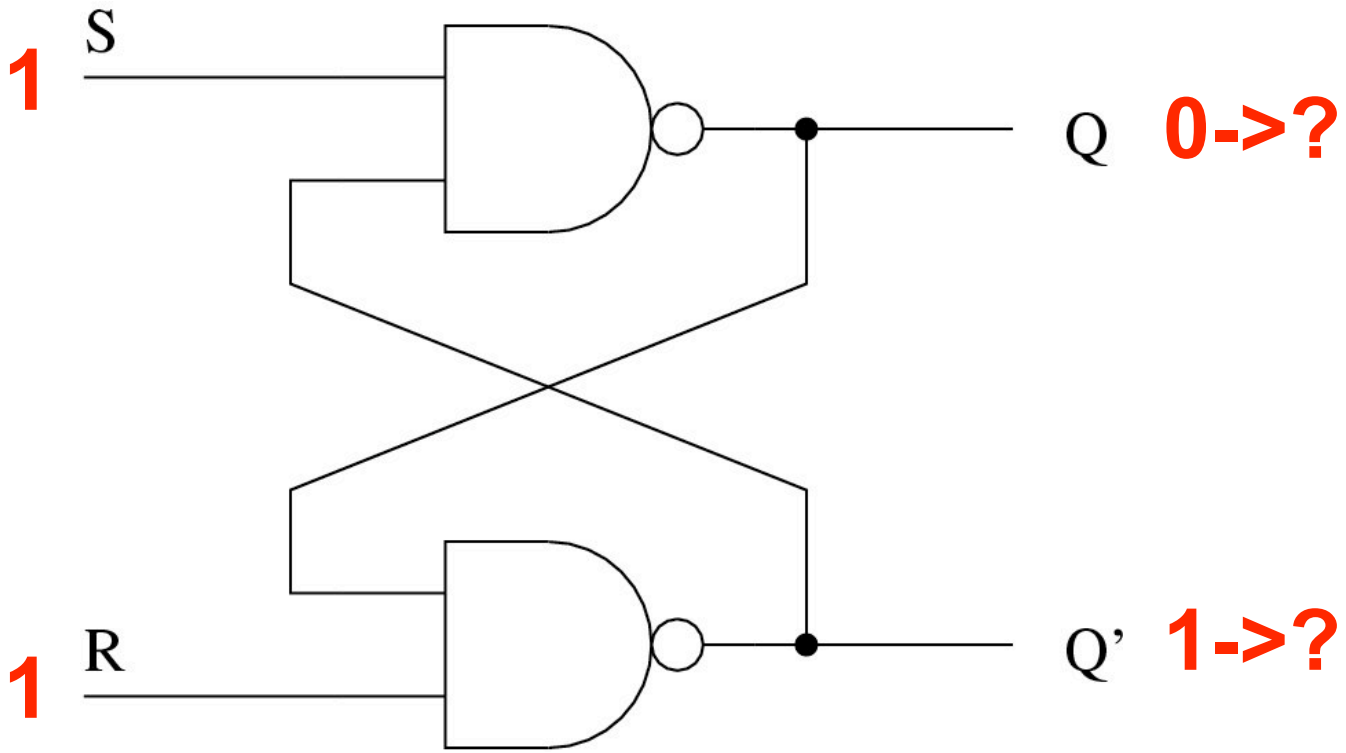
Now set R to 0 – what happens?



The state flips back (Q is ‘reset’)

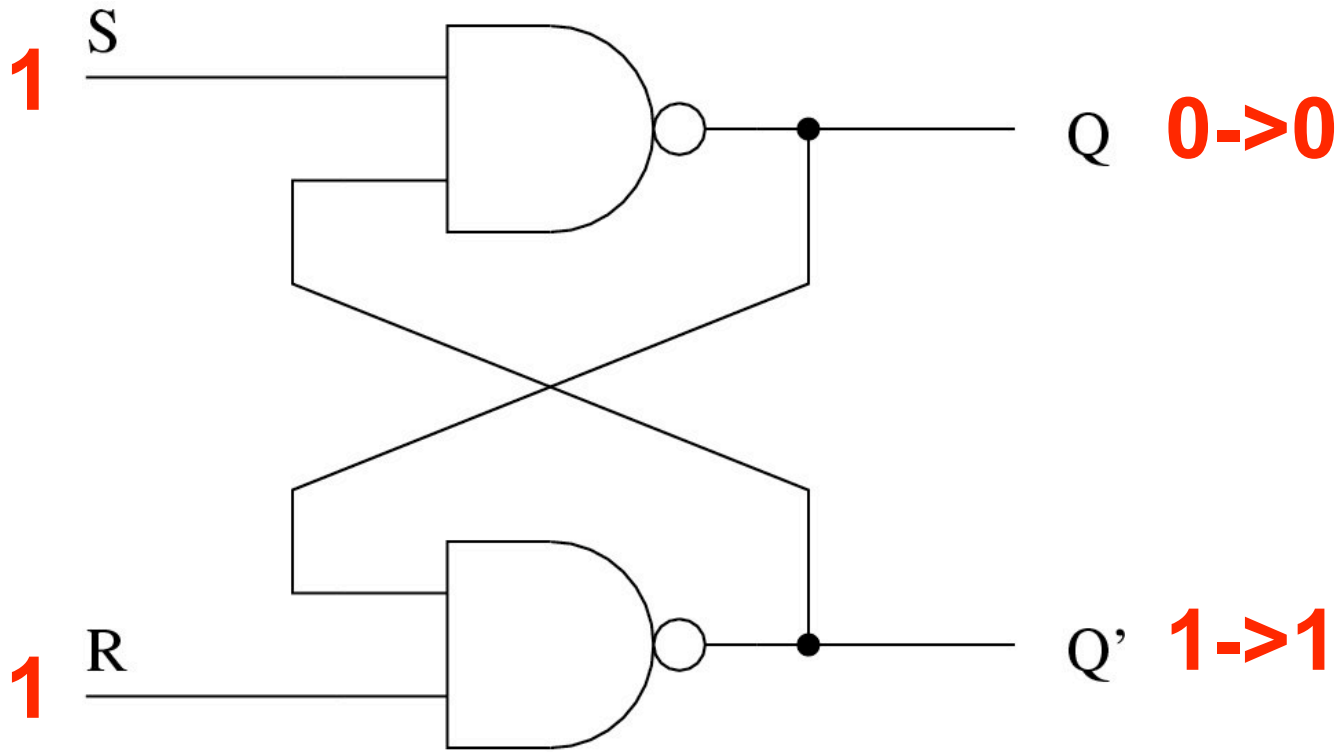
NAND Latch

Finally: set R to 1 – what happens?



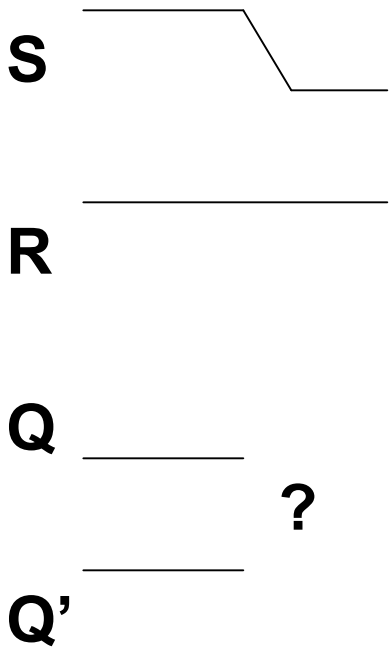
NAND Latch

Finally: set R to 1 – what happens?

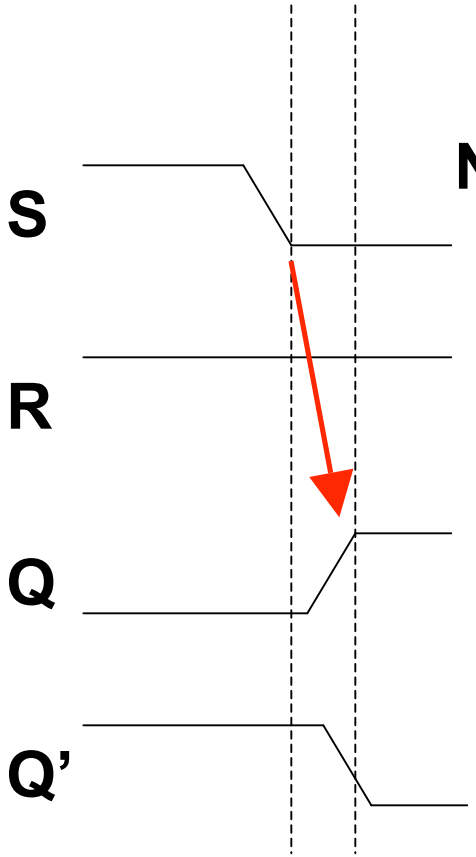


Q and Q' do not change state

Timing Diagram Representation

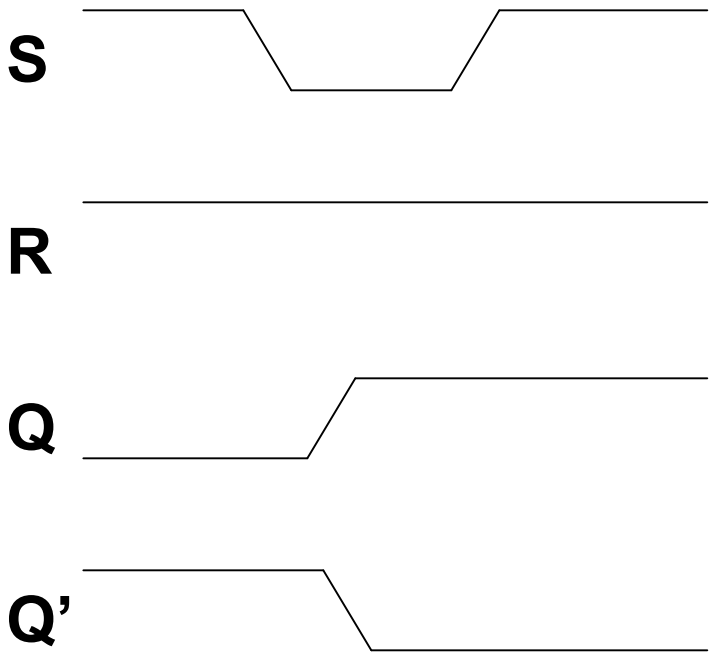


Timing Diagram Representation



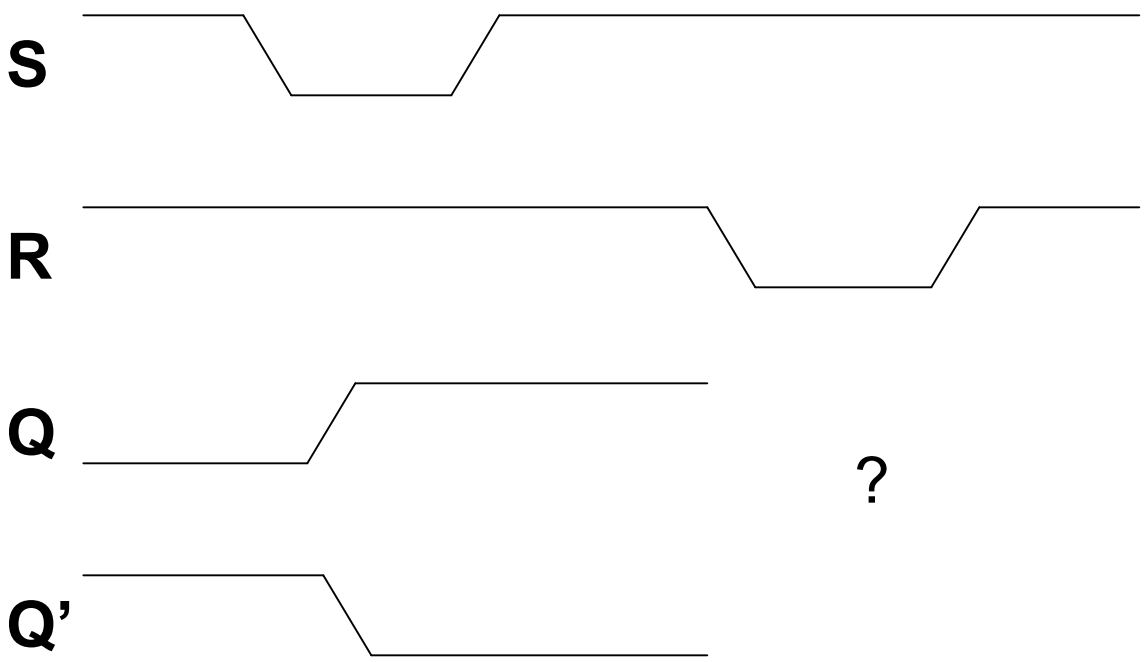
Note small delay in response in Q and Q'

Timing Diagram Representation

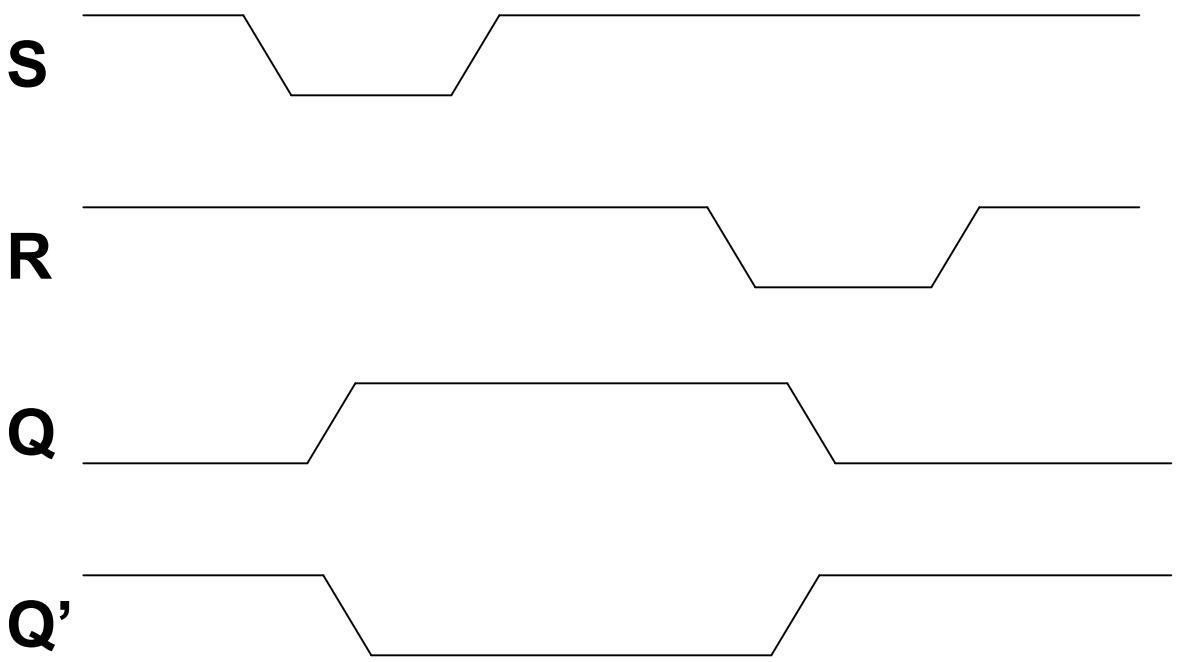


When S returns to high – both Q and Q' remain in the same state

Timing Diagram Representation

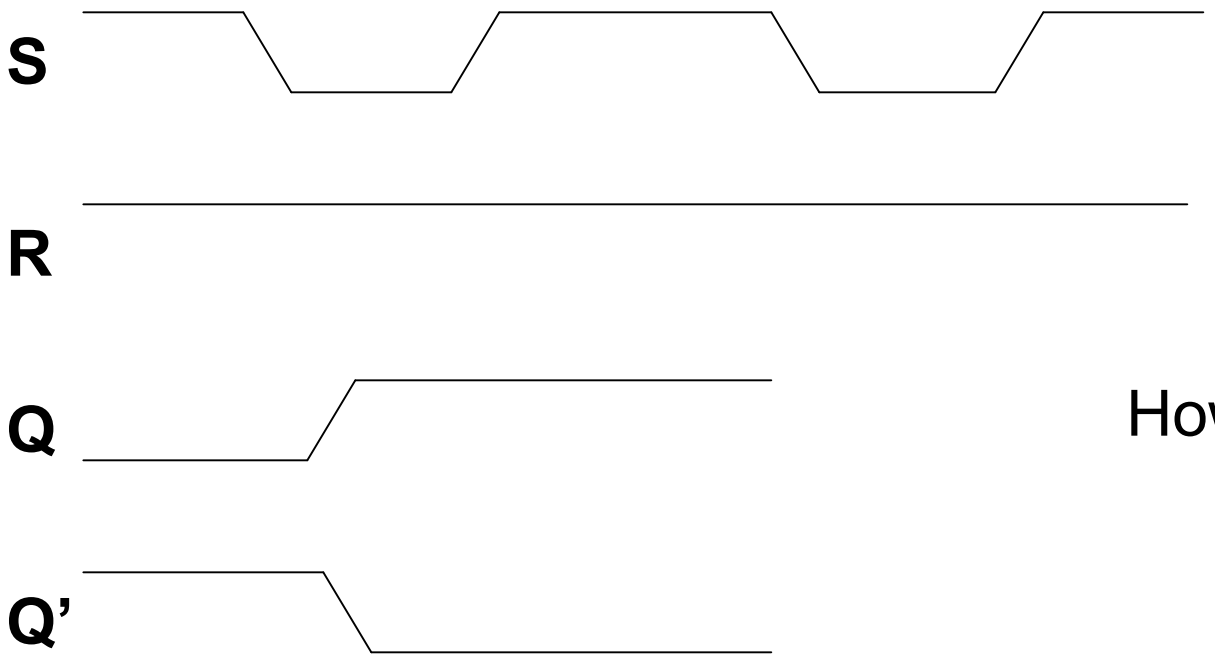


Timing Diagram Representation



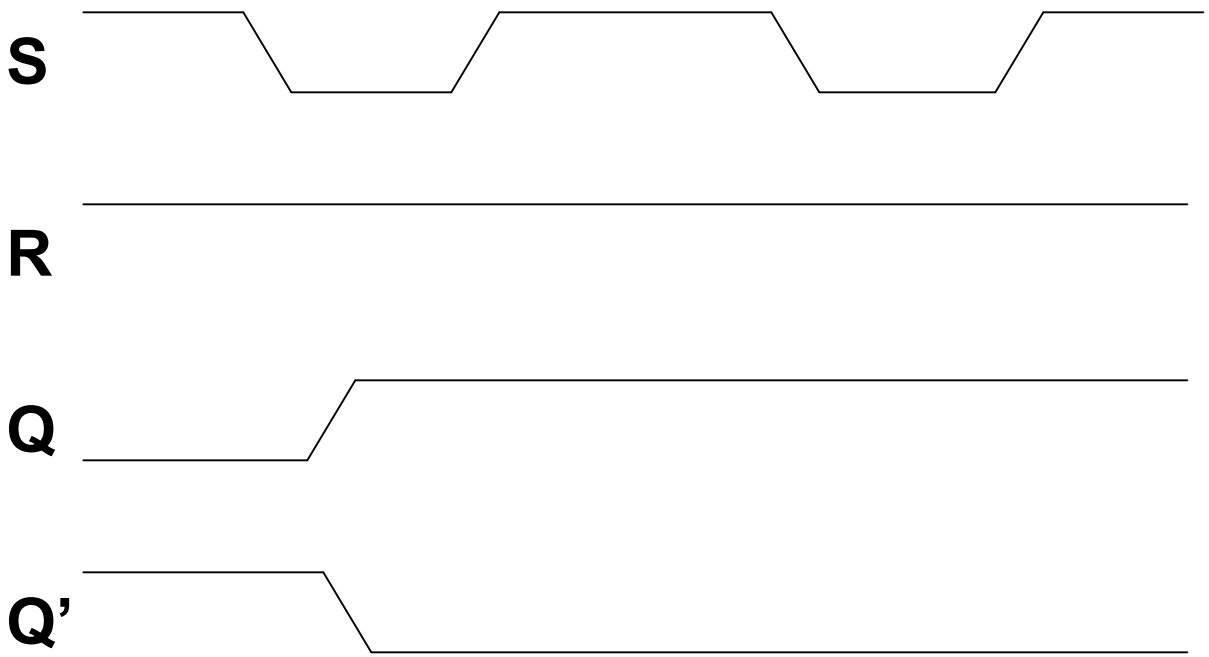
Q and Q' flip state

Timing Diagram Representation



How about this case?

Timing Diagram Representation



No change in Q and Q’

Latches

Provide us with a simple form of memory

- State of the circuit depends not only on the current inputs, but also on the recent history of the inputs

Latches

But: our circuit responds any time the inputs are low

- We want to limit the state change to a very narrow time period
- This will allow us to synchronize the state change of several devices

-> Flip Flops

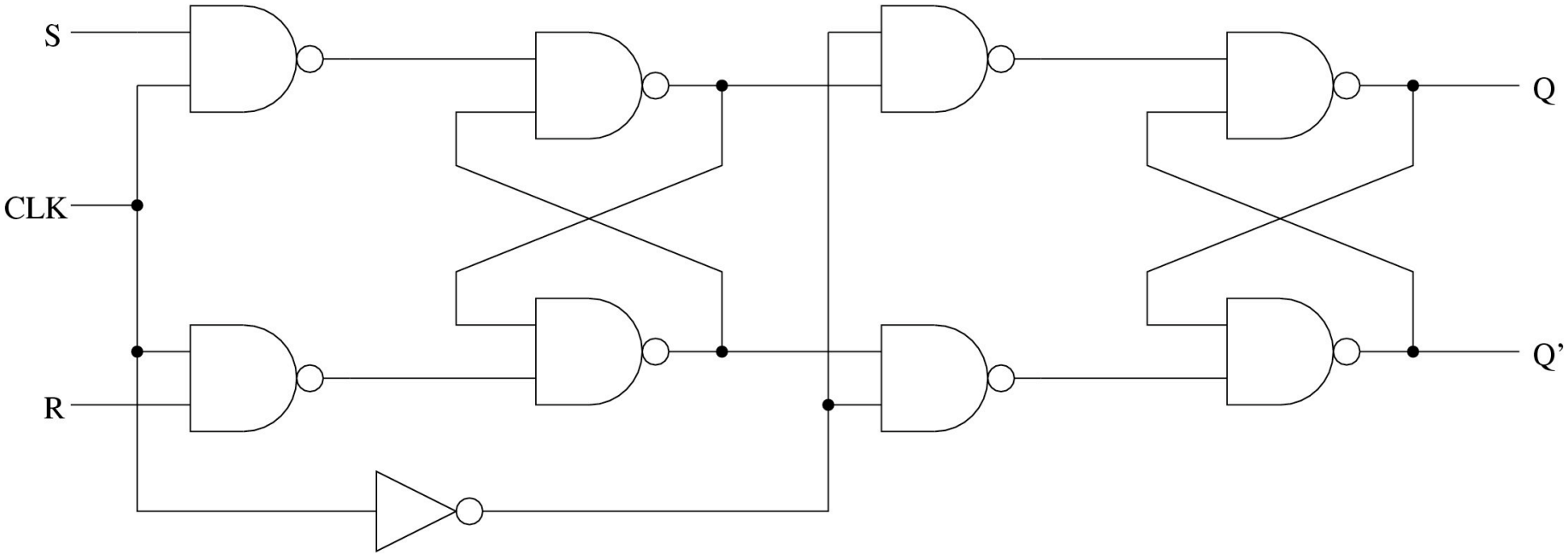
Flip Flops

- Add one more input to the circuit: a “clock” signal
- We will only allow the state of the output to change in response to S & R when the clock transitions from 1 to 0

Flip Flops

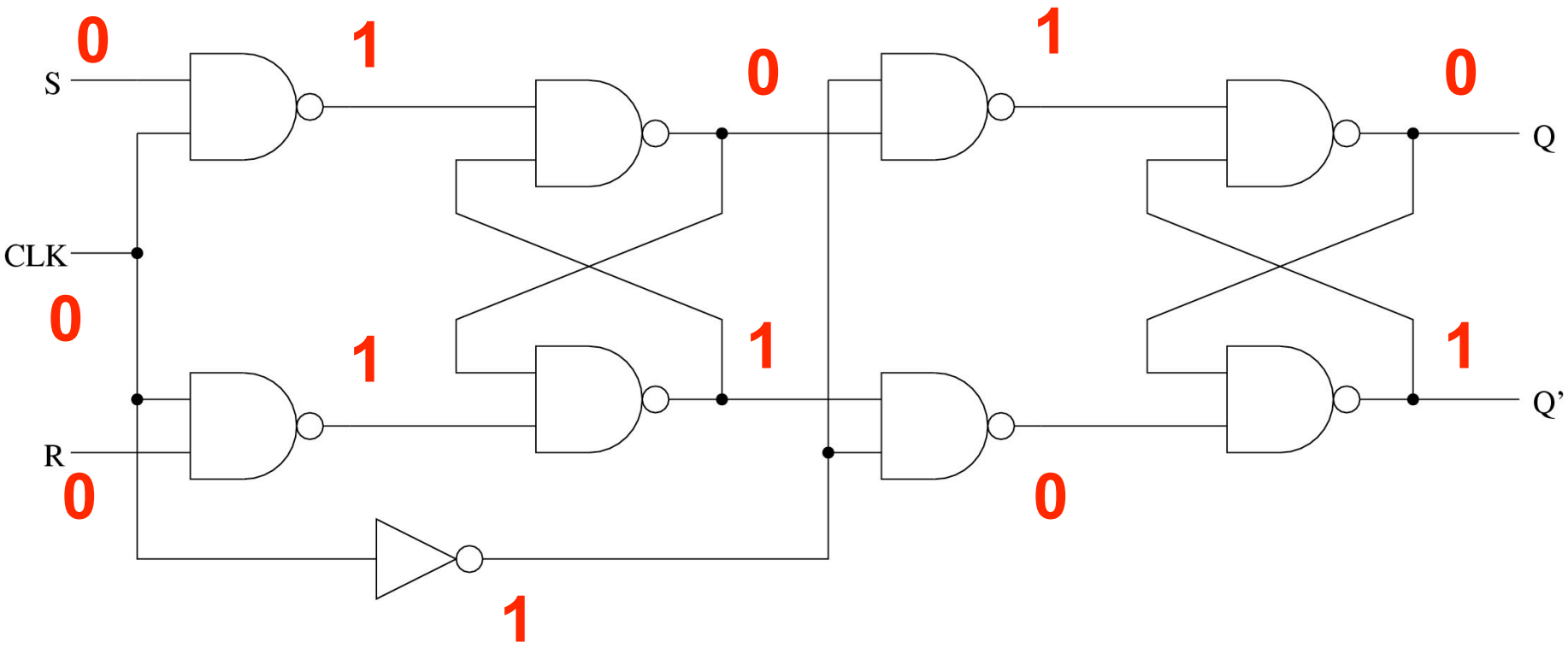
- Add one more input to the circuit: a “clock” signal
- We will only allow the state of the output to change in response to S & R when the clock transitions from 1 to 0

Flip Flops



R-S Flip Flop

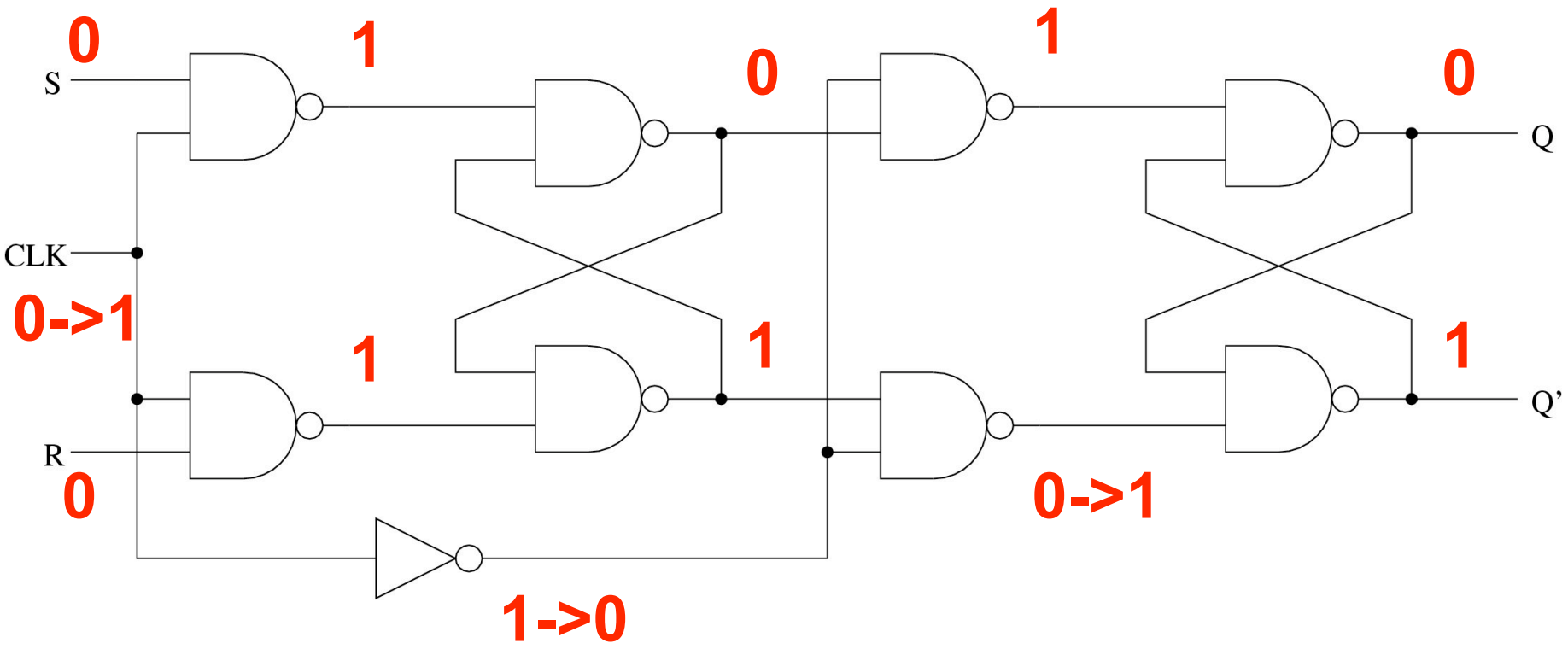
Initial state



Note that the meaning of S & R has been inverted

R-S Flip Flop

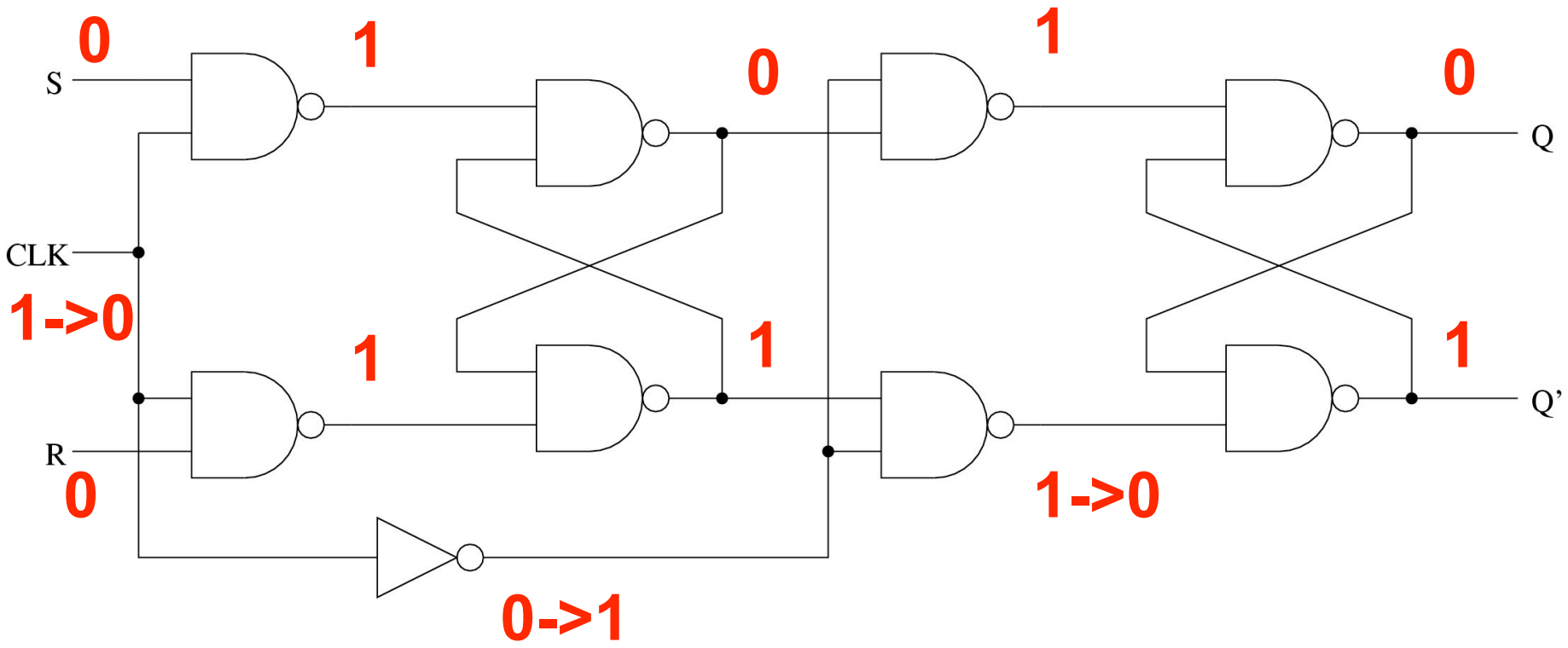
Clock goes high



No change in Q and Q’

R-S Flip Flop

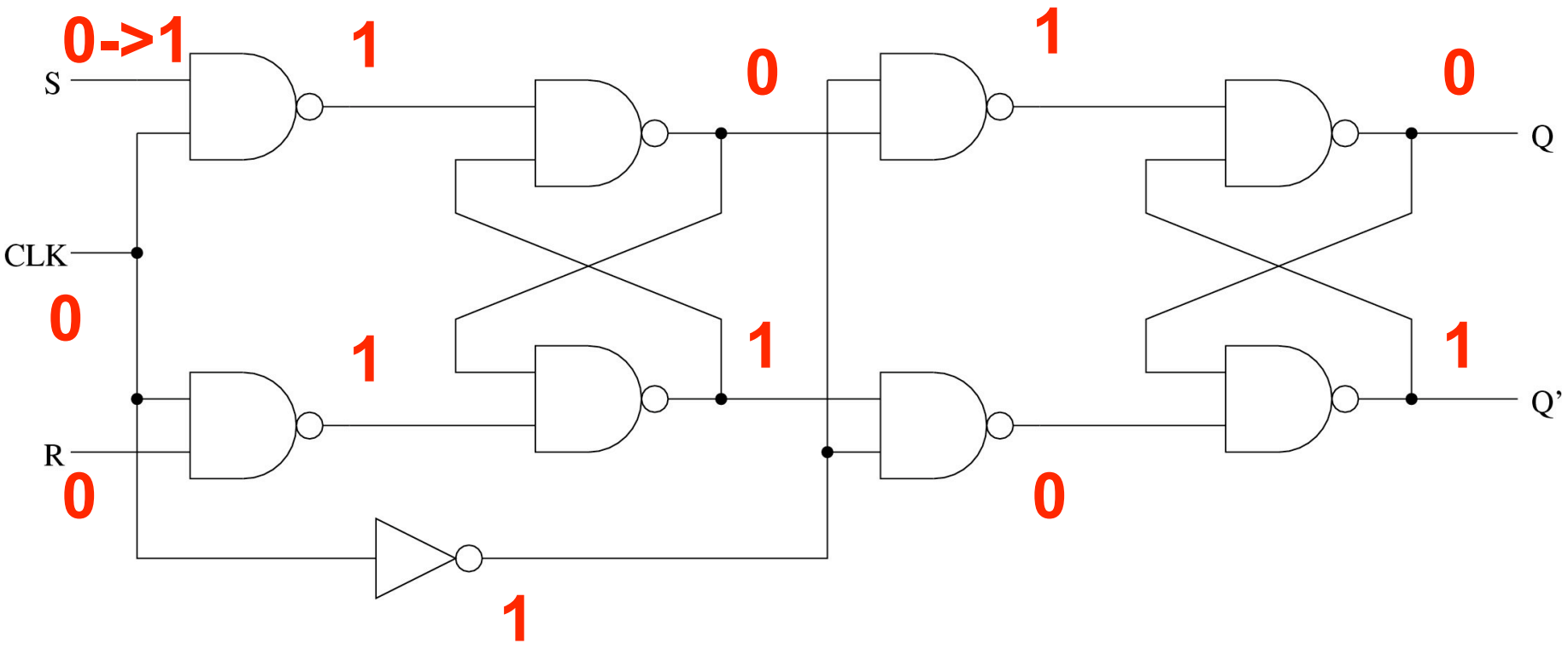
Clock goes low again



Still no change in Q and Q'

R-S Flip Flop

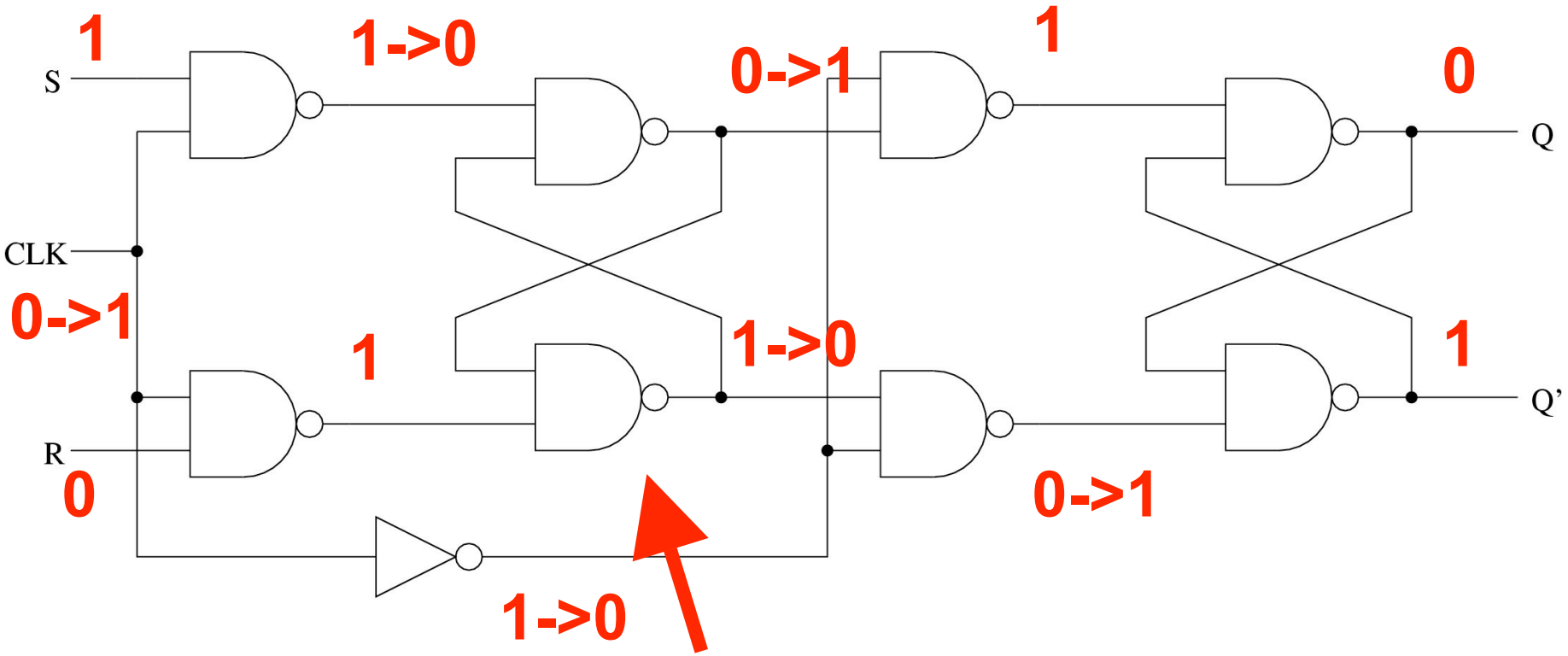
S goes high



Nothing in the circuit changes

R-S Flip Flop

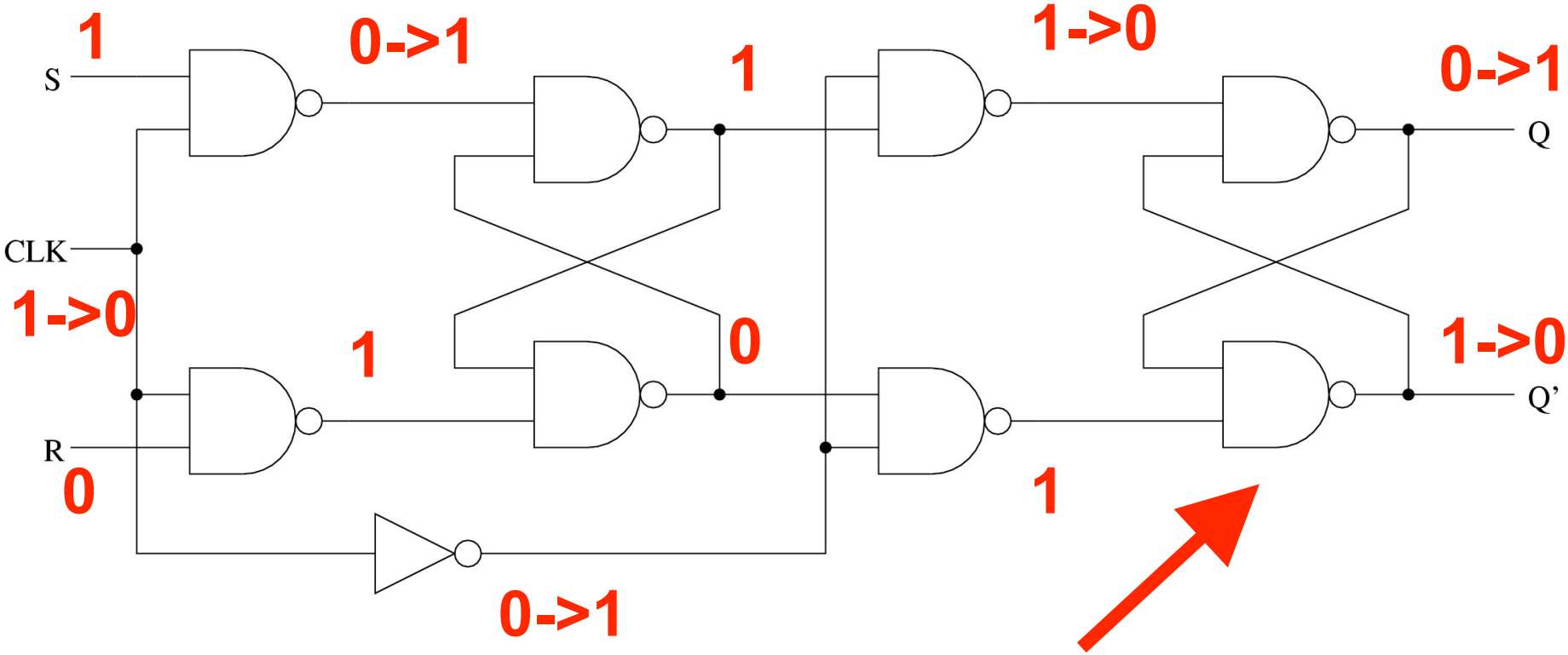
Now: clock goes high



The state of the first latch changes

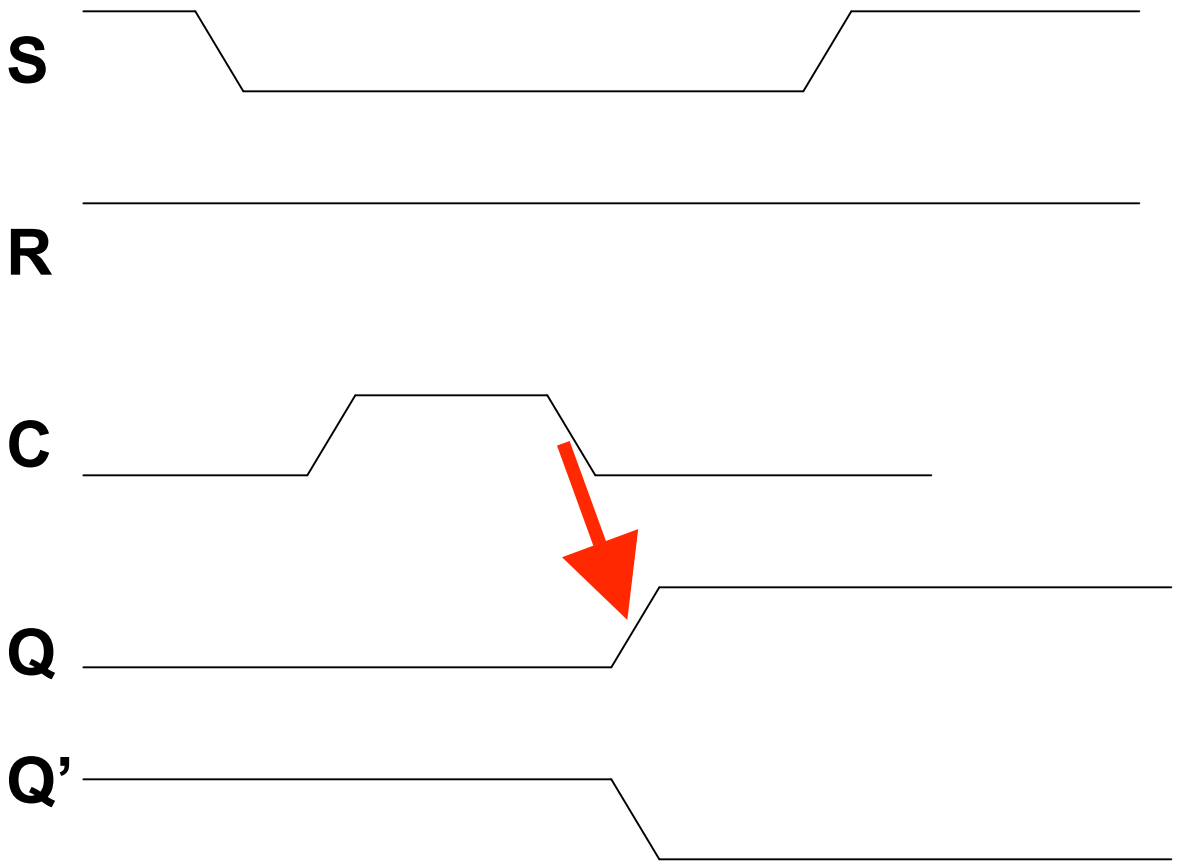
R-S Flip Flop

Now: clock goes low



The state of the second latch changes!

R-S Flip Flop
Timing Diagram Representation



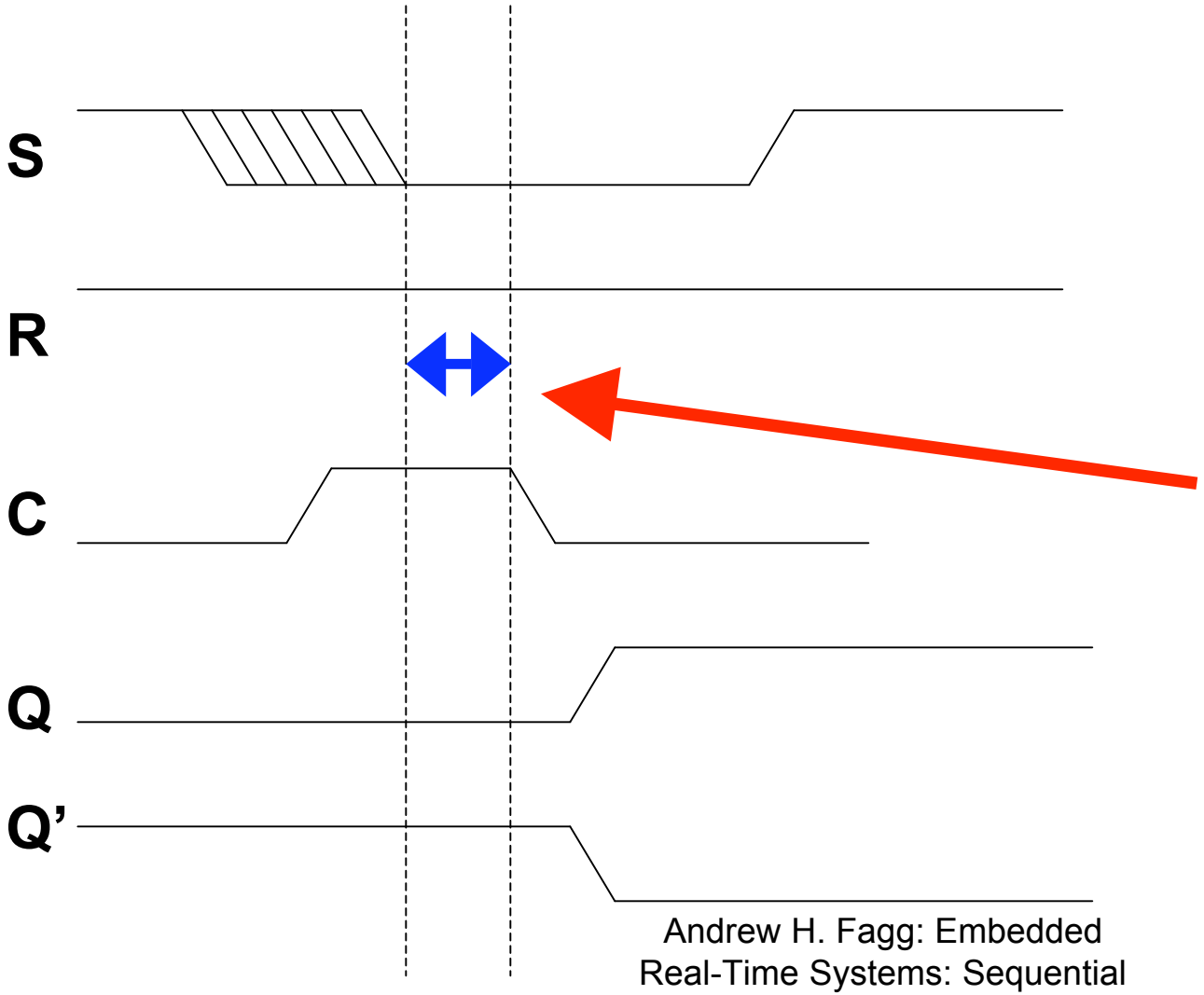
Q and Q' flip state only after the clock goes low

R-S Flip Flop

The timing of the drop of S is not critical

- But it must do so before the clock goes low

R-S Flip Flop
Timing Diagram Representation



The circuit will require a specified amount of “setup time”

R-S Flip Flop Summary

Behaves like an R-S latch – but:

- The flip flop will only “pay attention” to the R-S inputs on the falling edge of the clock

Next Time

- D flip flops
- Binary number encoding
- Shift registers
- Counters

Last Time

- Project 1 specification
- Sequential logic:
  - R-S Latch
  - R-S Flip flop

Today

Sequential circuits continued

- Clocked R-S latch
- D Flip flop
- Binary coding
- Shift registers
- Counters

Administrivia

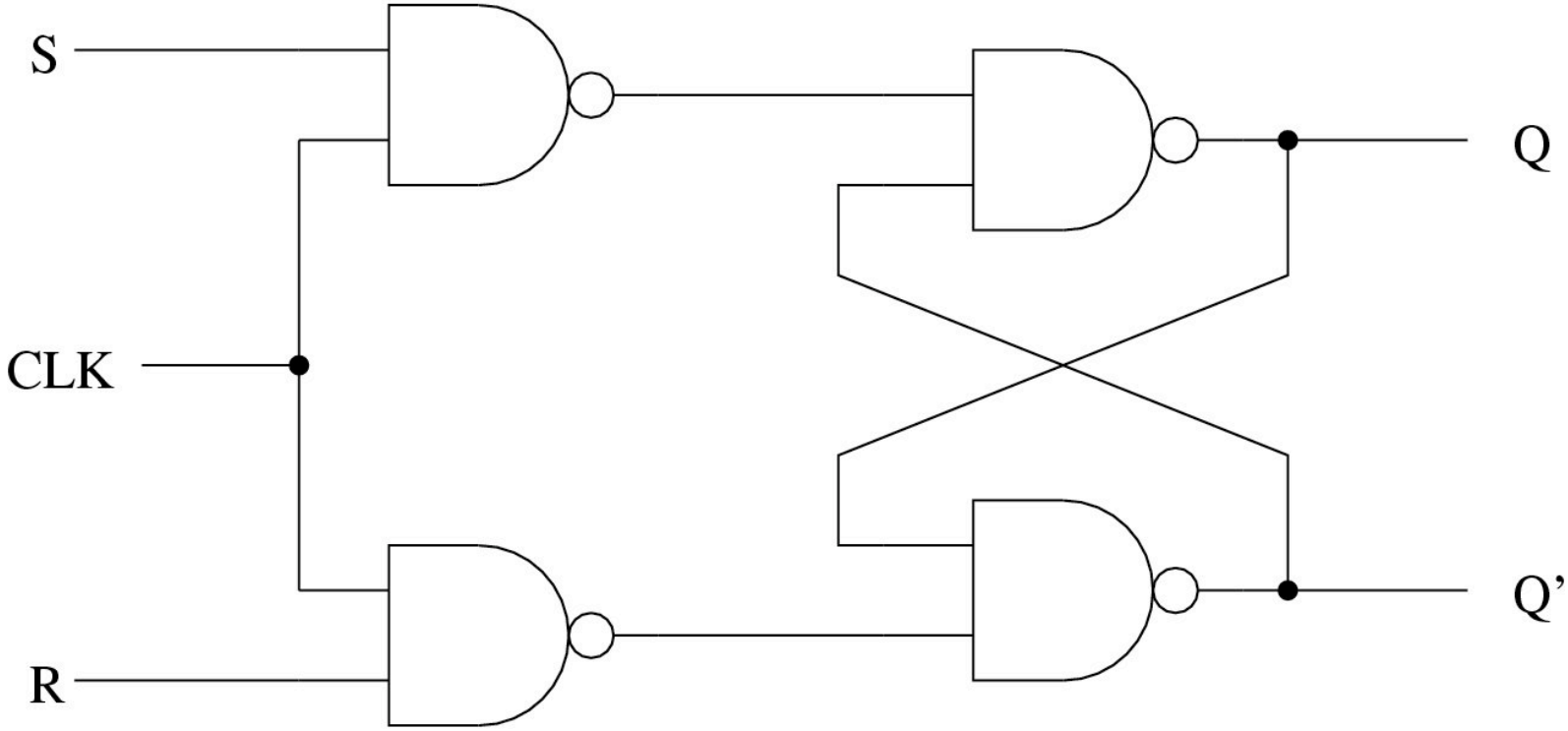
- Mark back?
- Homework 1 is out:
  - Due Feb 17th @ 5:00
- Project 1:
  - Worth 8% of your final grade
  - The group that demonstrates successfully first will receive an extra 0.5% of extra credit

Latch vs Flip flop

- Latch implements a simple form of memory
- A flip flop adds:
  - Precise control over when the state of the memory changes

Clocked R-S Latch

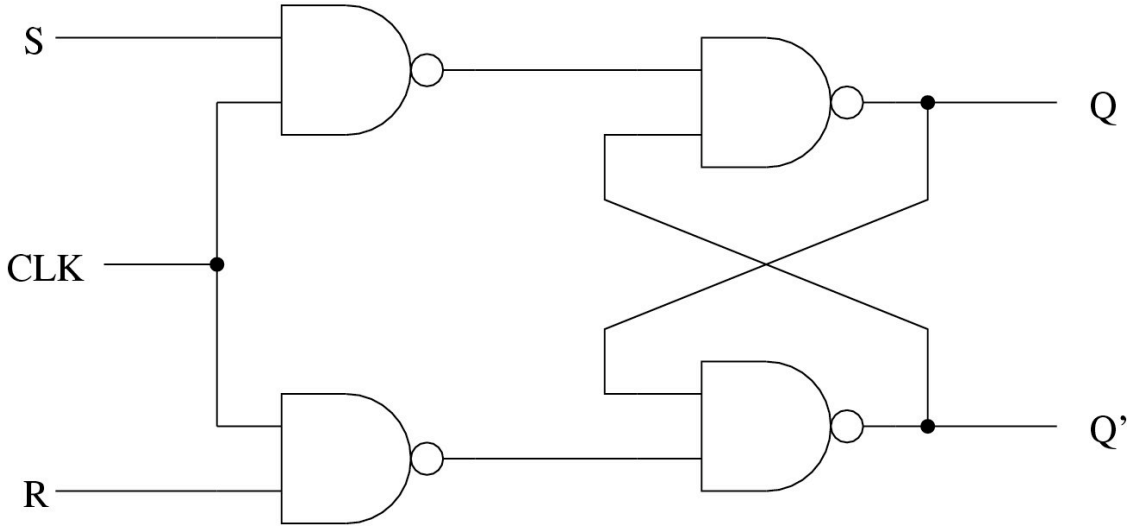
Allows some control over when the latch changes state



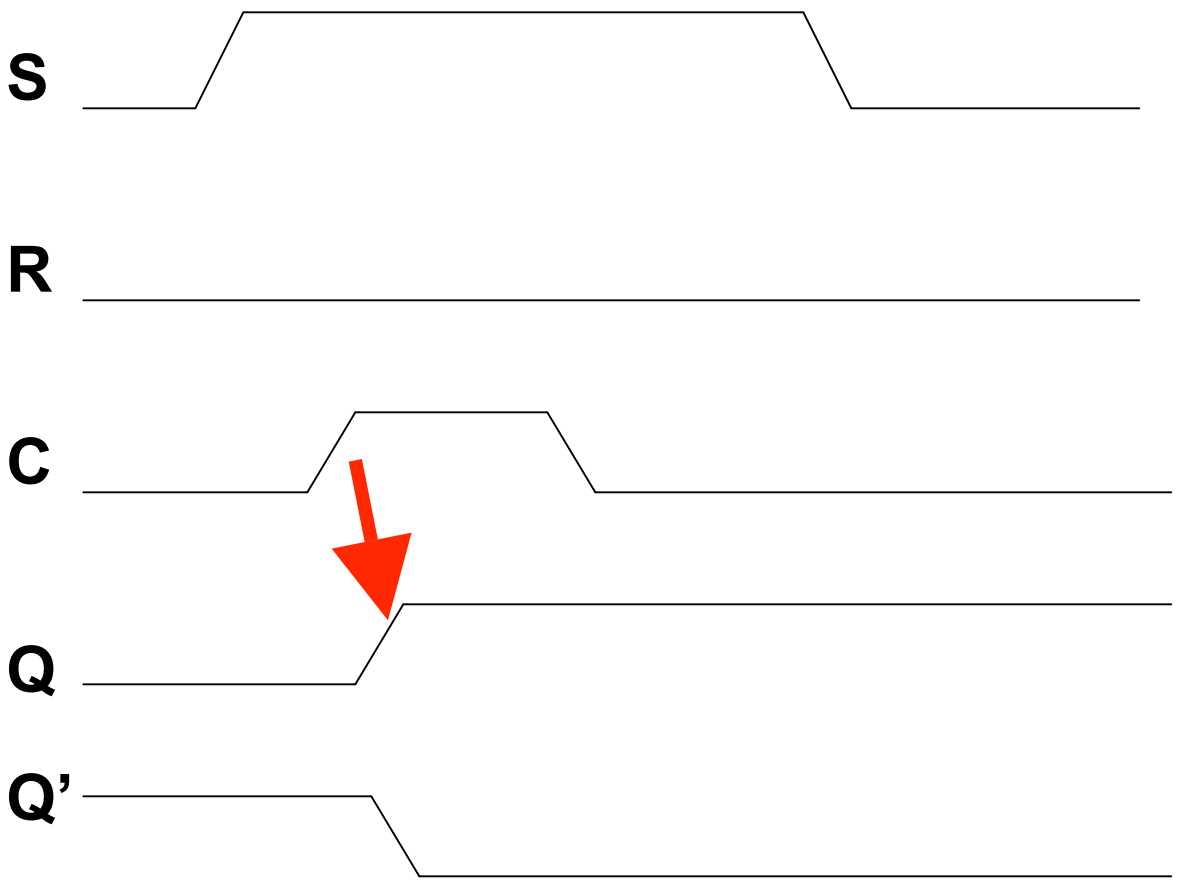
Clocked R-S Latch

State can only change when the clock is high

Note that R or S must be high to cause a reset or a set



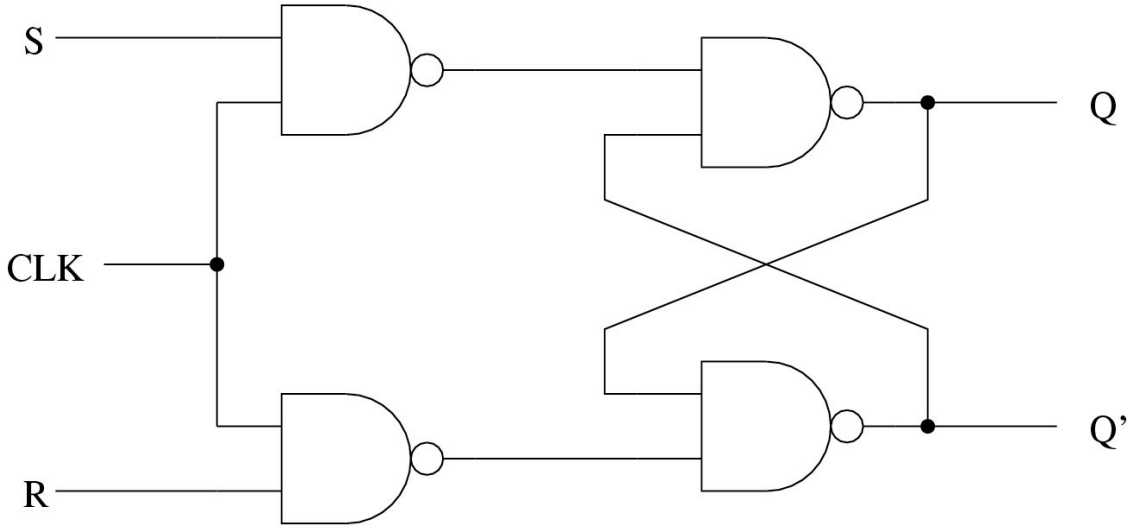
Clocked R-S Latch



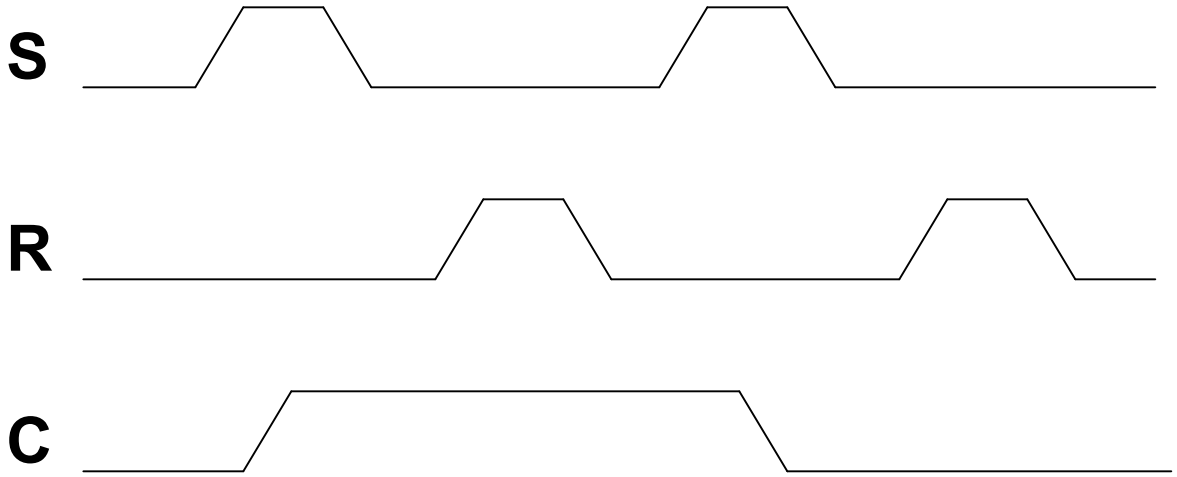
Q and Q' flip state when the clock is high

Clocked R-S Latch

How is this different than our R-S flip flop?

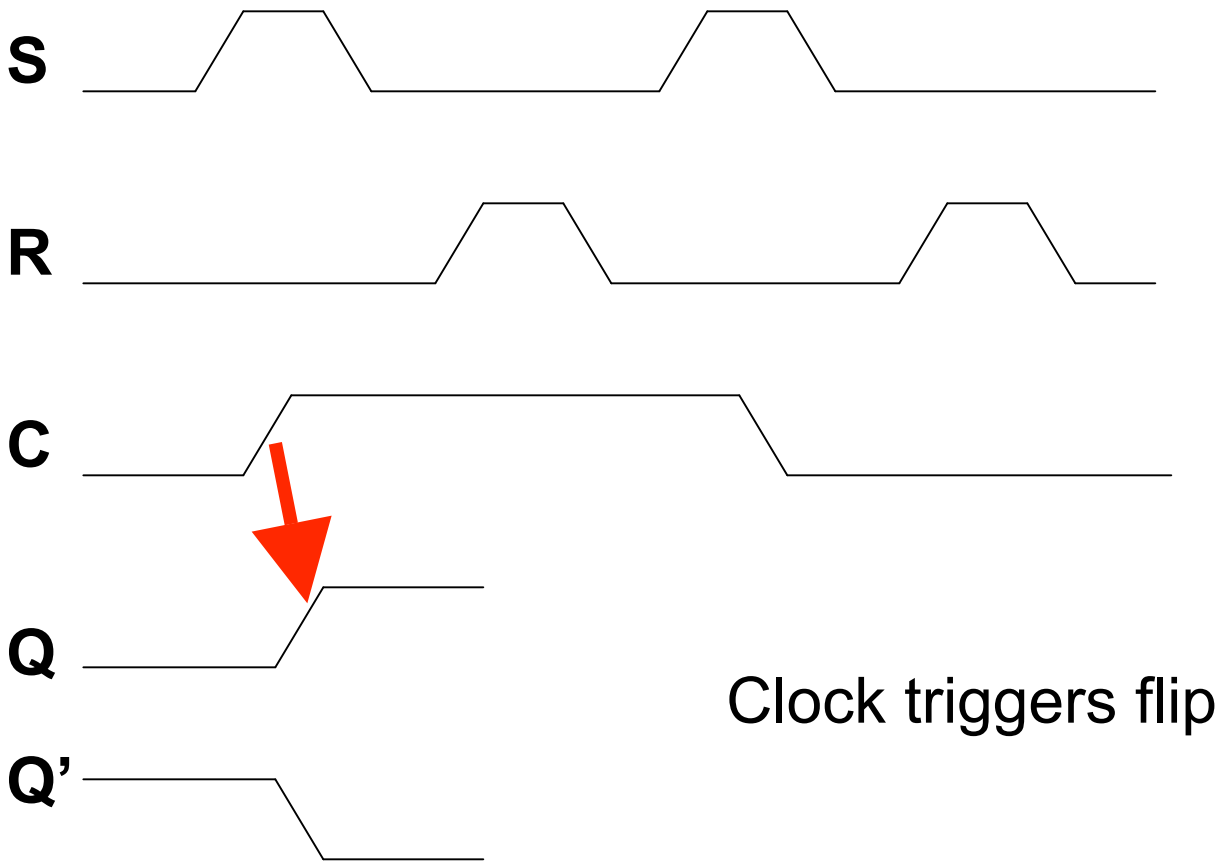


Clocked R-S Latch



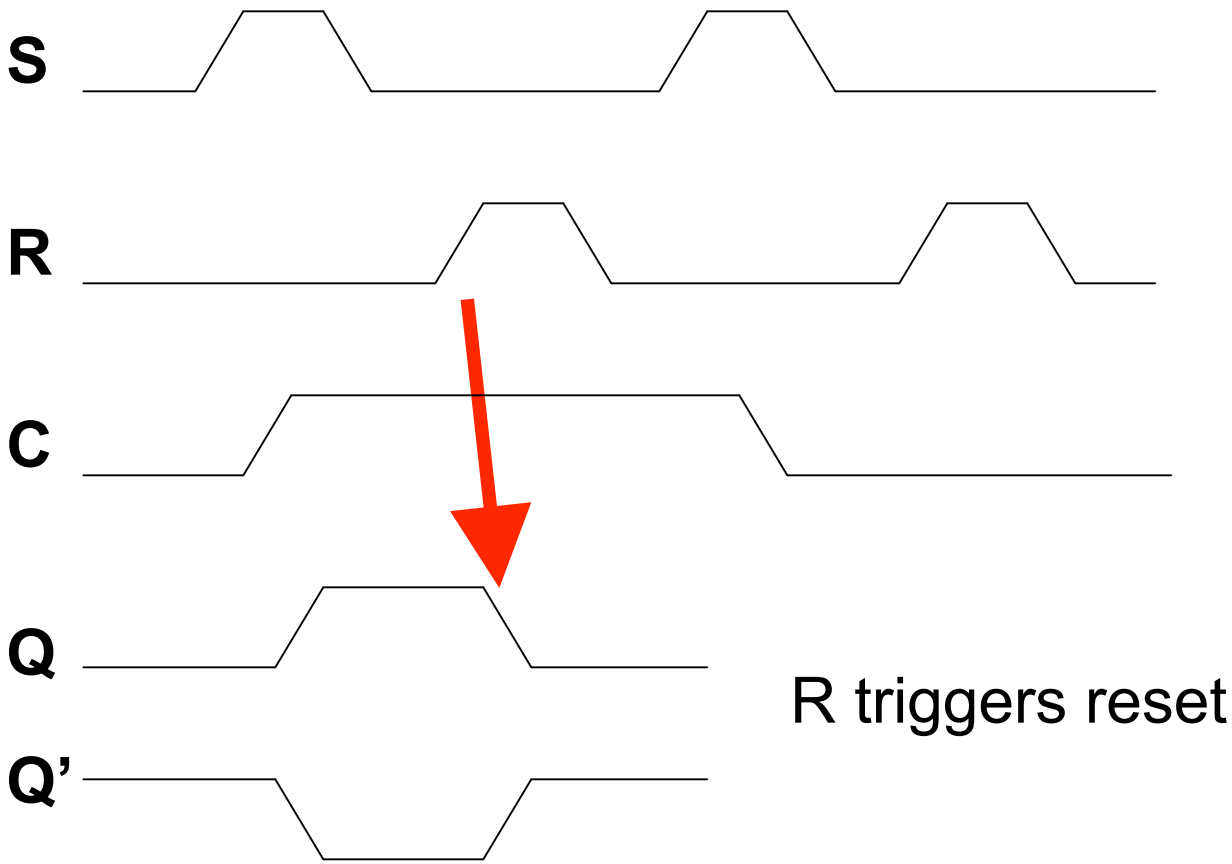
What do Q and Q' do?

Clocked R-S Latch

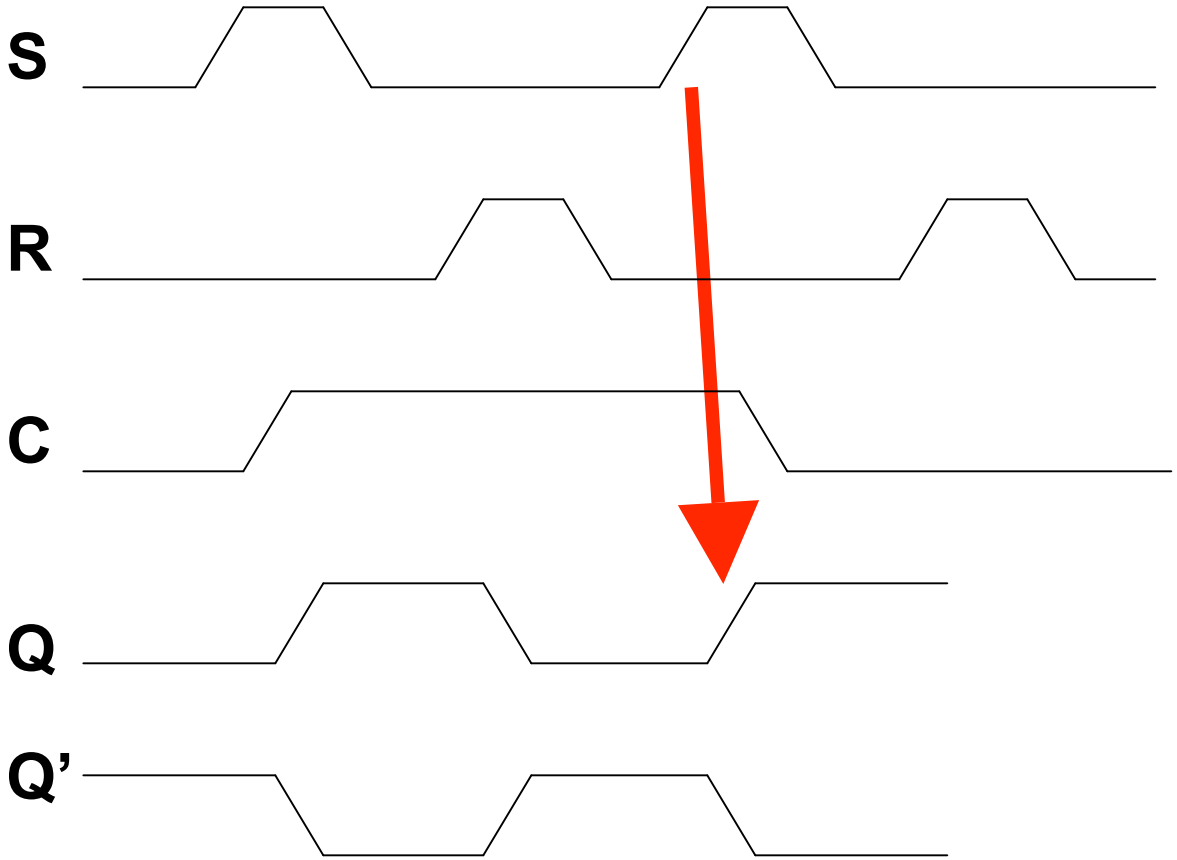


Clock triggers flip

Clocked R-S Latch

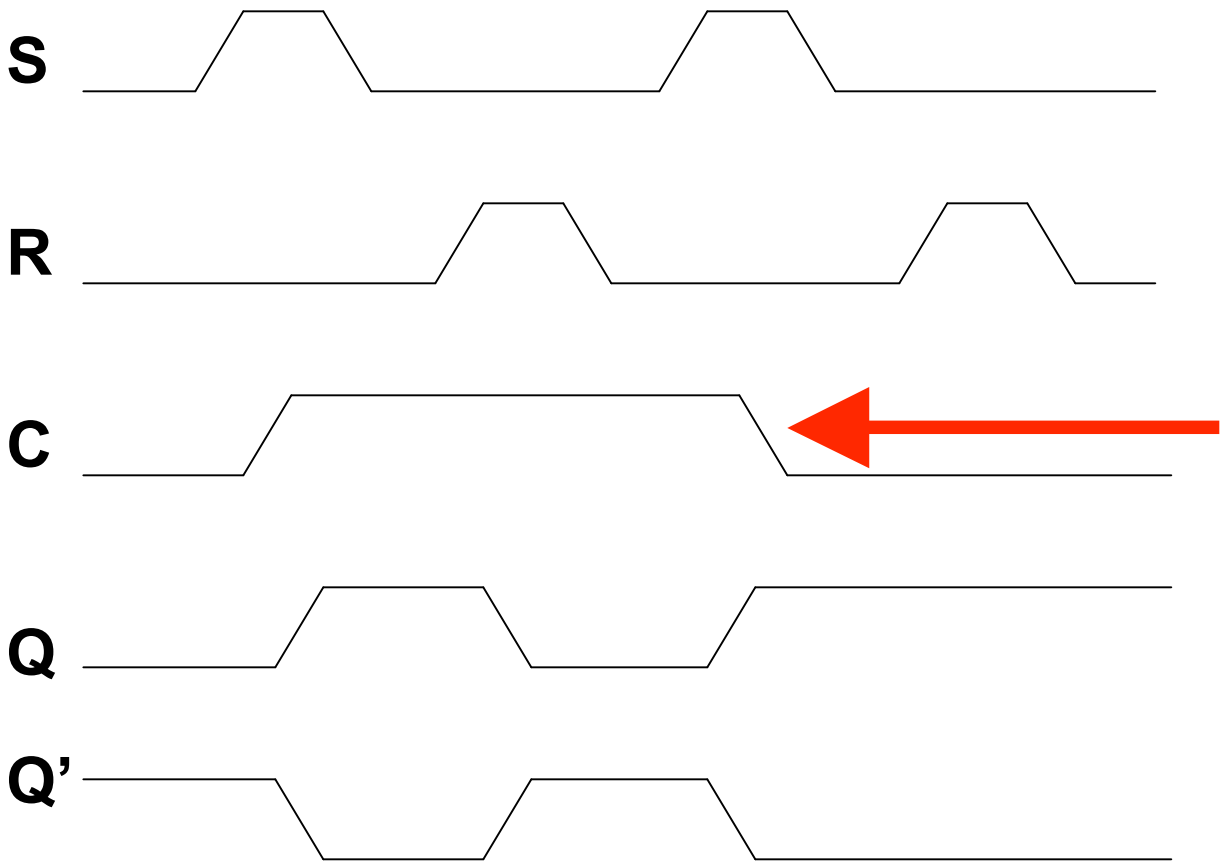


Clocked R-S Latch



S triggers set

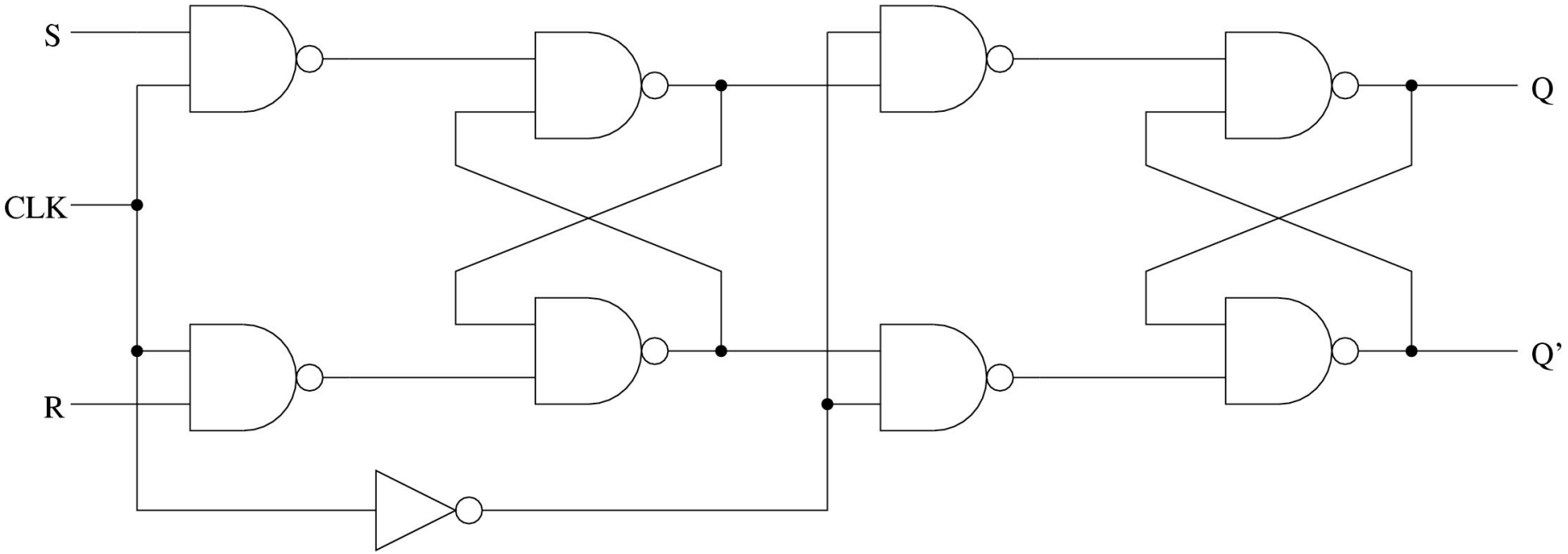
Clocked R-S Latch



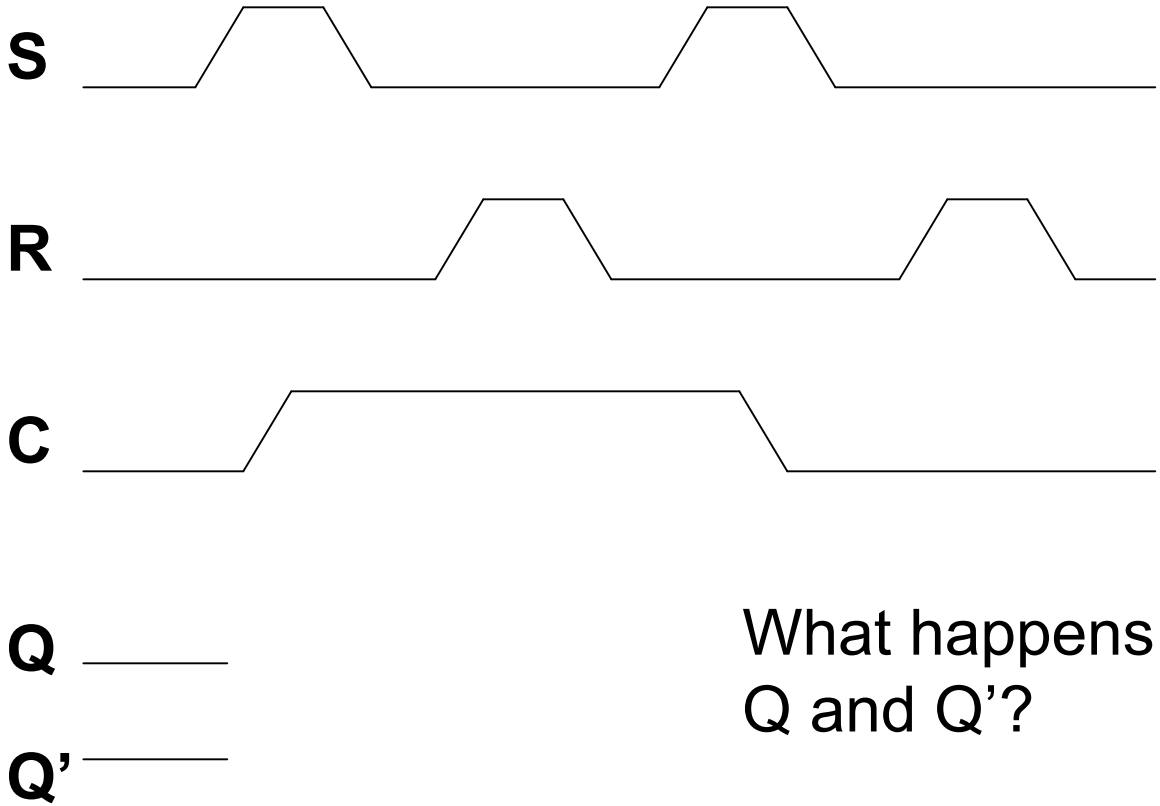
Clock goes low:
No further
changes in state

R-S Latch vs Flip Flop

What would the R-S flip flop do?

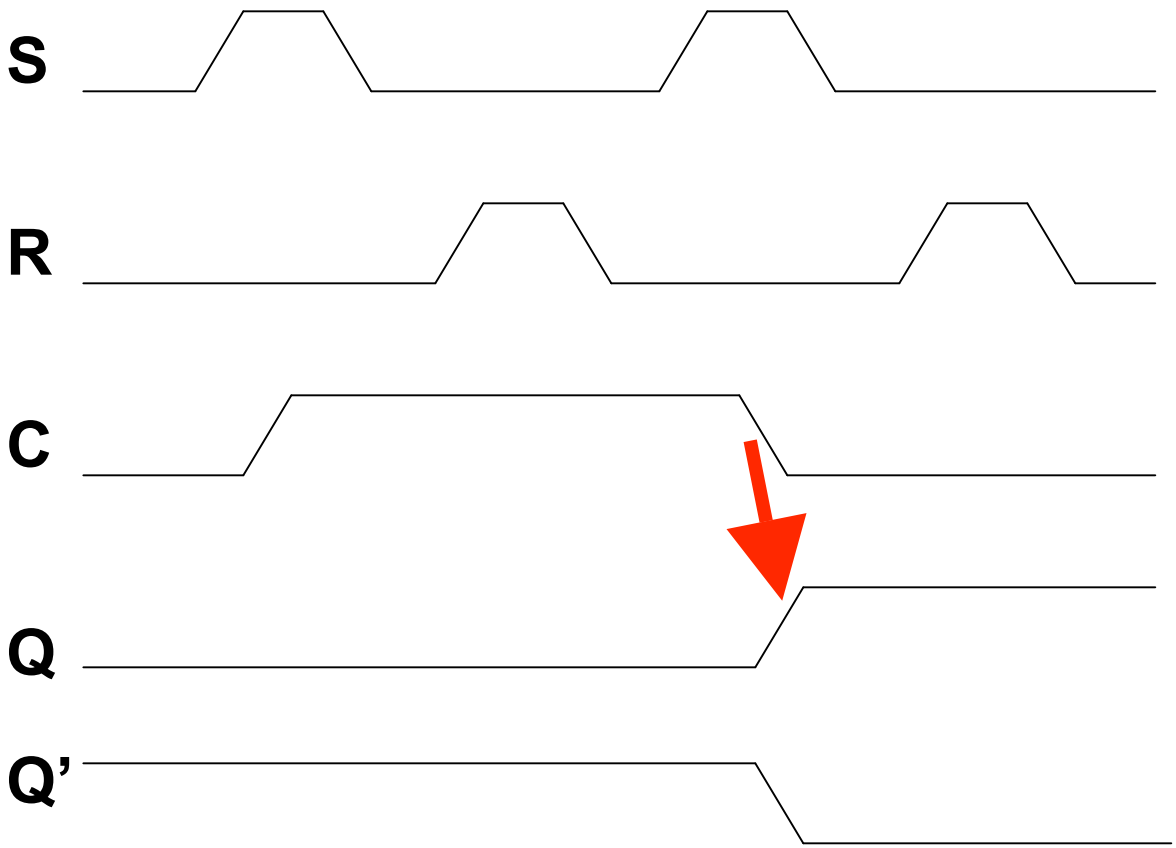


R-S Flip Flop



What happens to Q and Q'?

R-S Flip Flop



State change only on downward edge of the clock

R-S Flip Flop

State change happens at a very precise time

R-S Flip Flop

State change happens at a very precise time

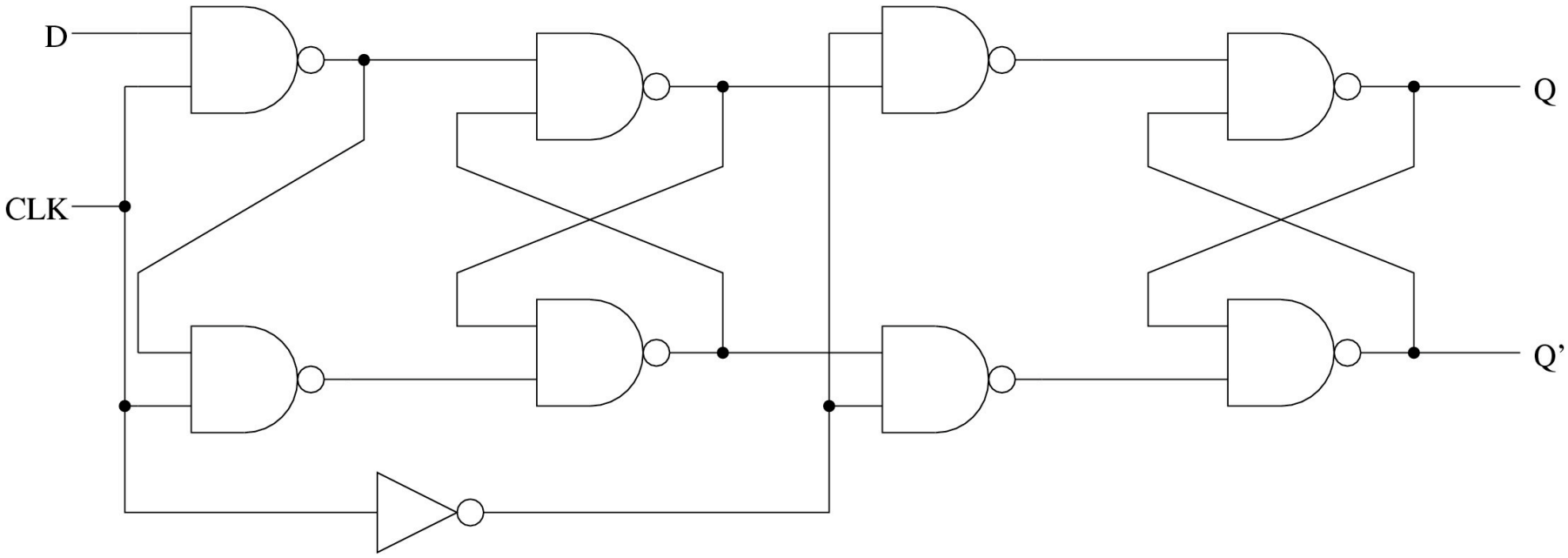
But:

- We must guarantee that R and S are never high at the same time
- We would like to be able to store the high/low state of a single line

D-Type Flip Flop

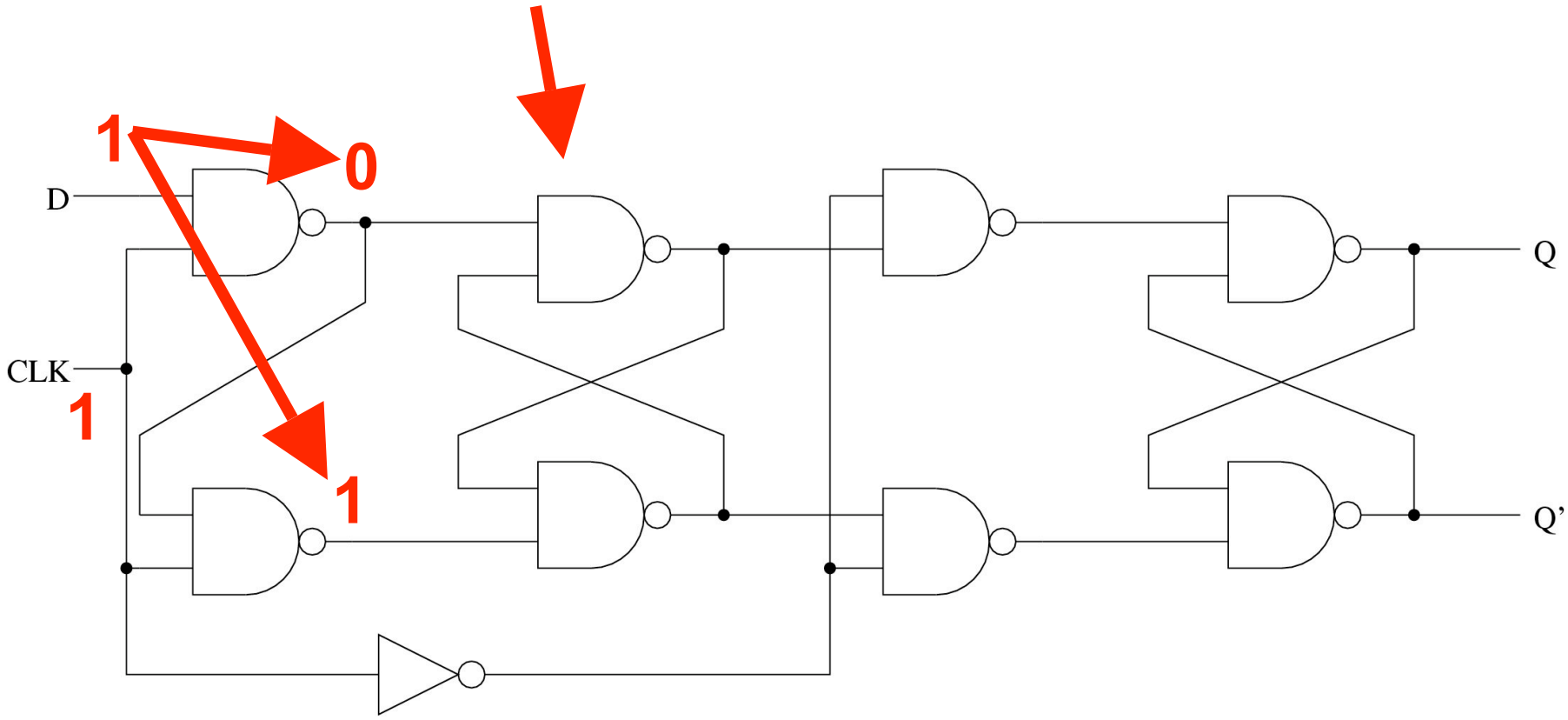
Replace R/S with D

- In essence, R is replaced with D’



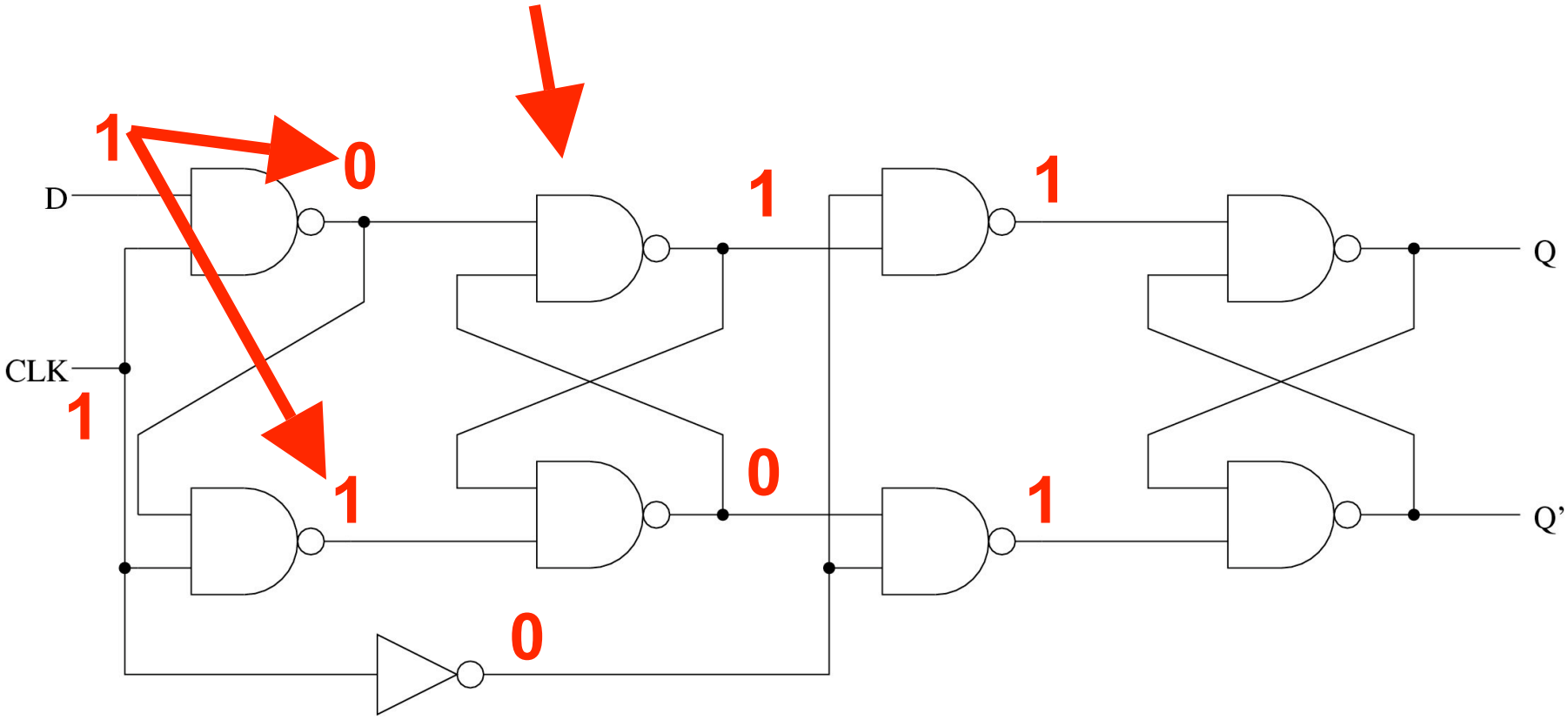
D-Type Flip Flop

D=1 results in a ‘set’ of the latch



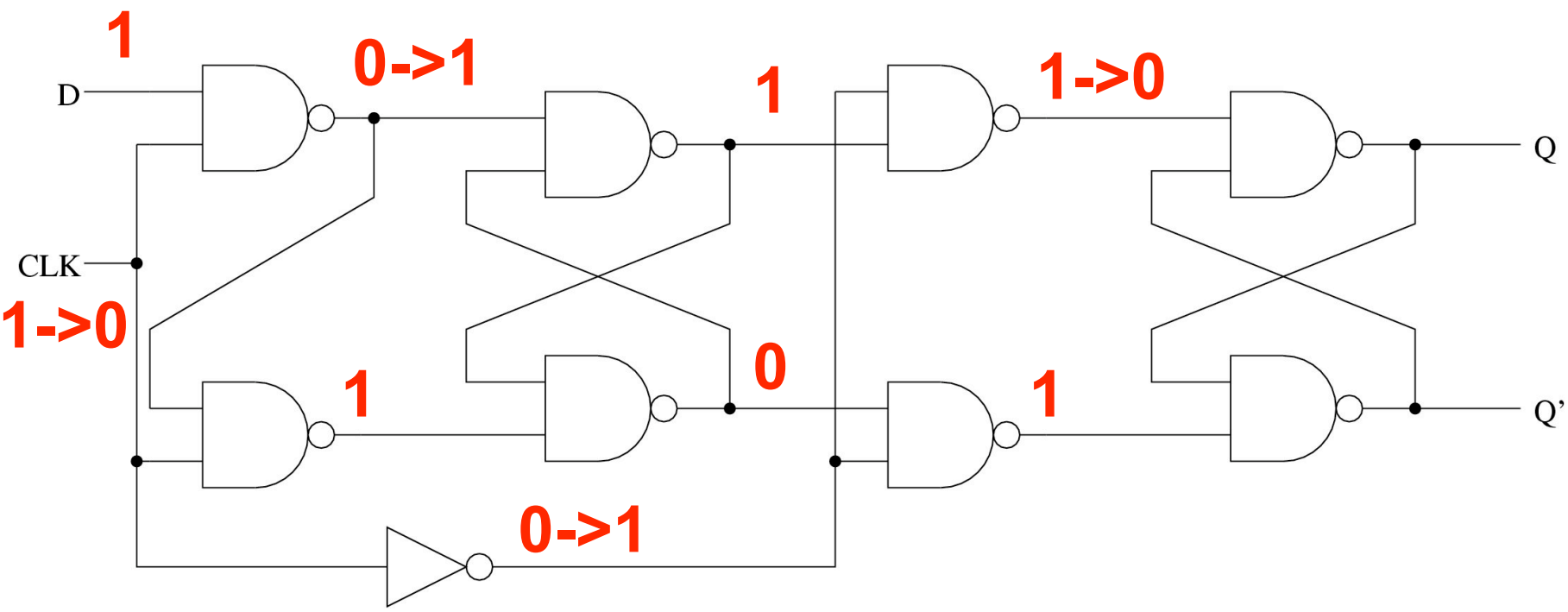
D-Type Flip Flop

D=1 results in a ‘set’ of the latch



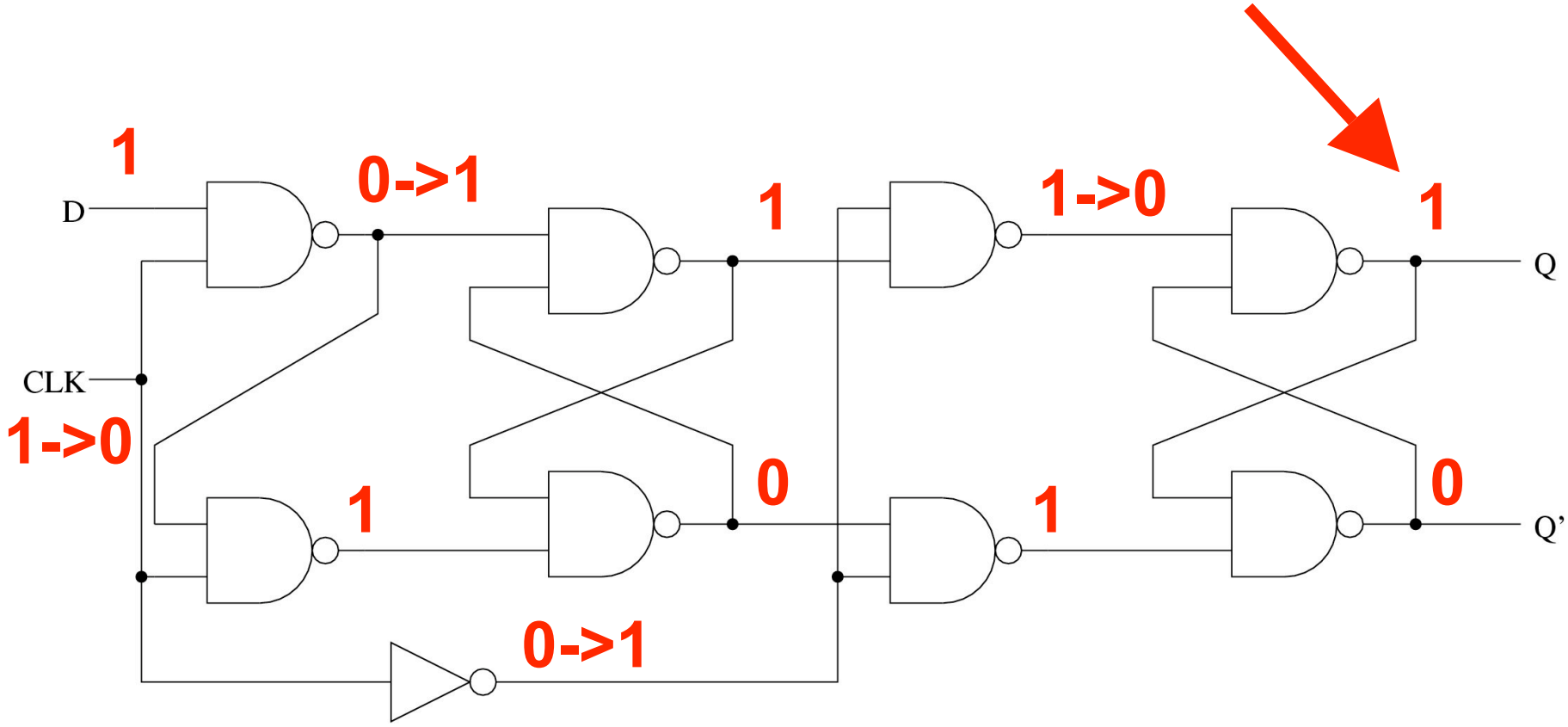
D-Type Flip Flop

Clock transitions from high to low



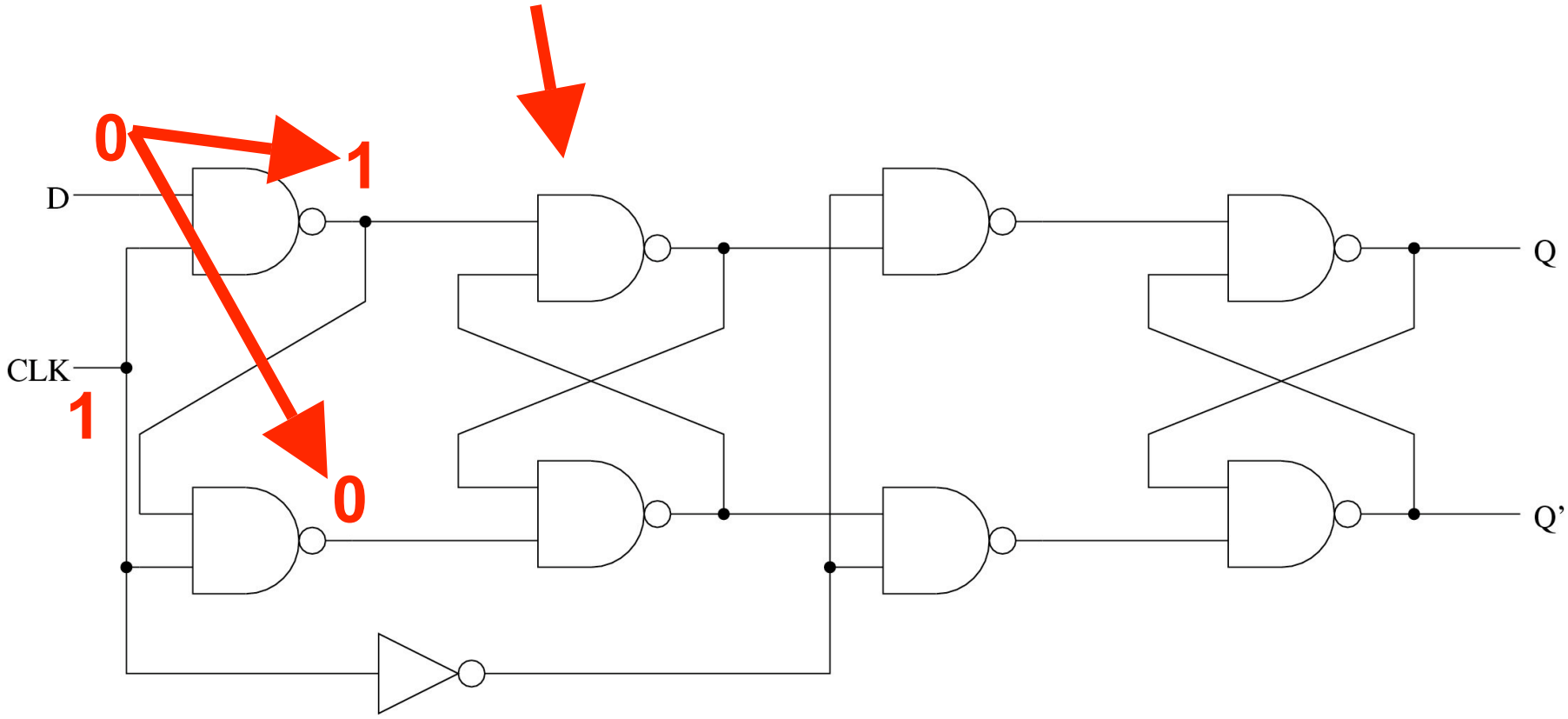
D-Type Flip Flop

Clock transition -> ‘set’ of the slave latch



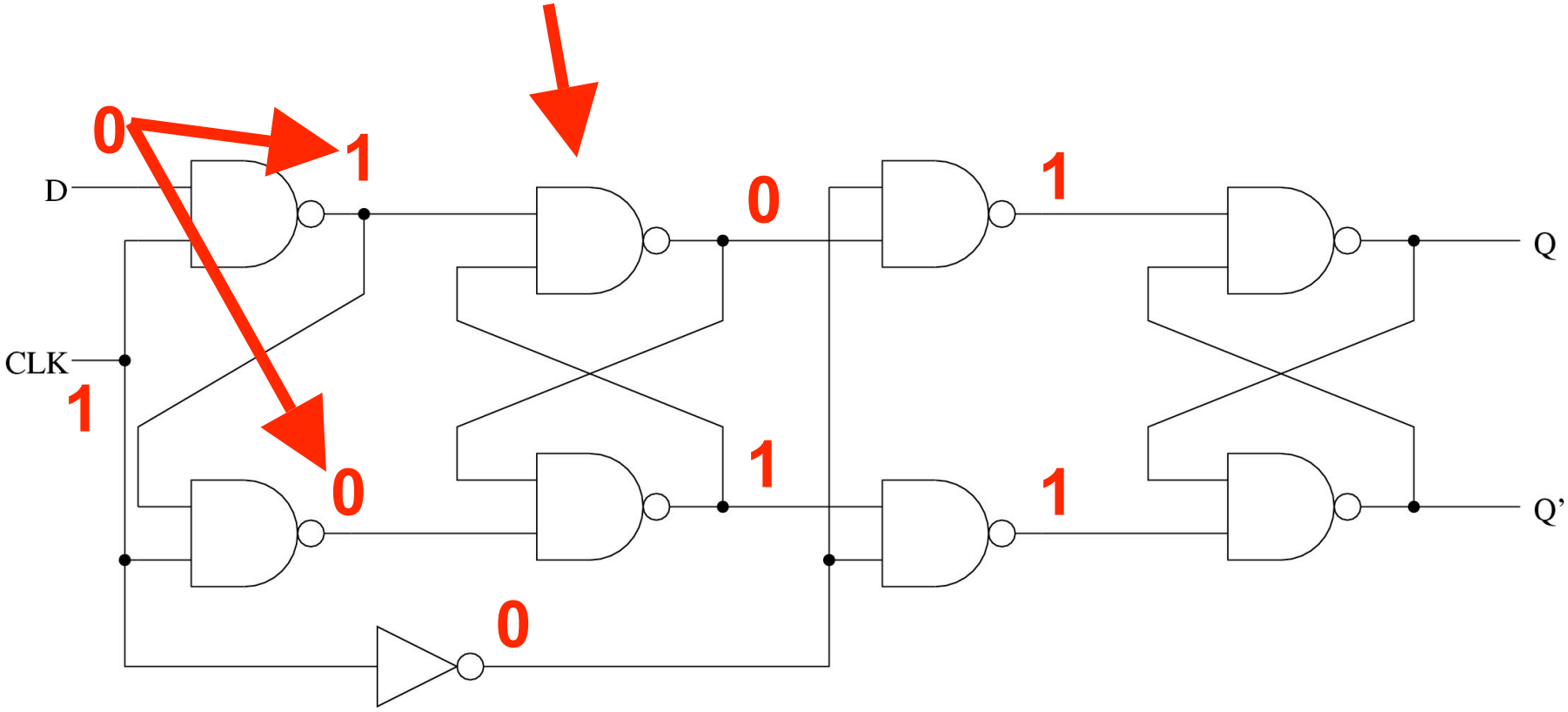
D-Type Flip Flop

D=0 results in a ‘reset’ of the latch



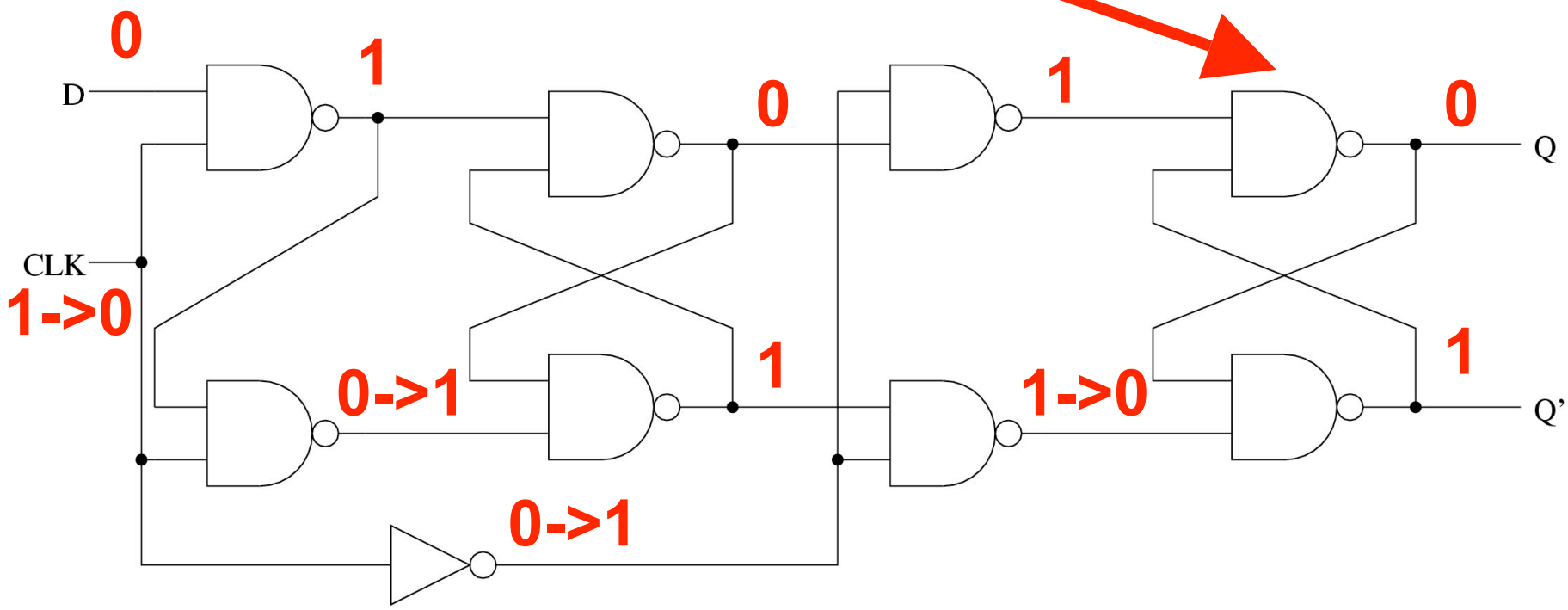
D-Type Flip Flop

D=0 results in a ‘reset’ of the latch

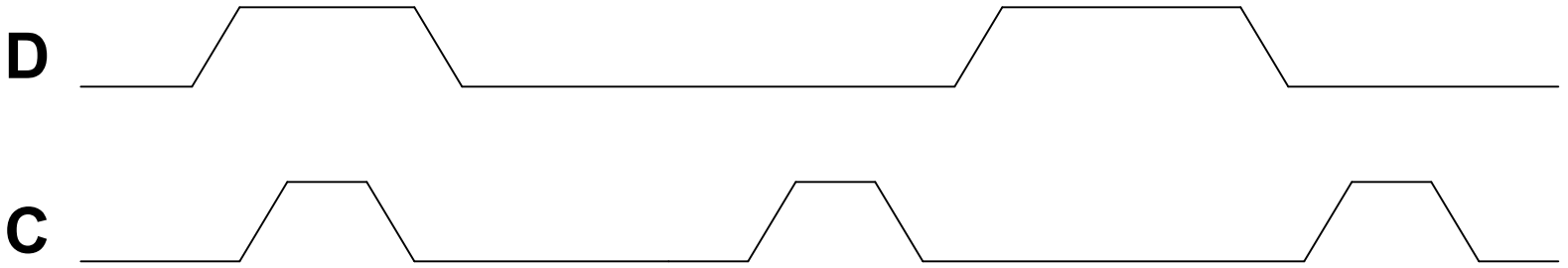


D-Type Flip Flop

Clock transitions from high to low results in a ‘reset’ of the slave latch



D Flip Flop

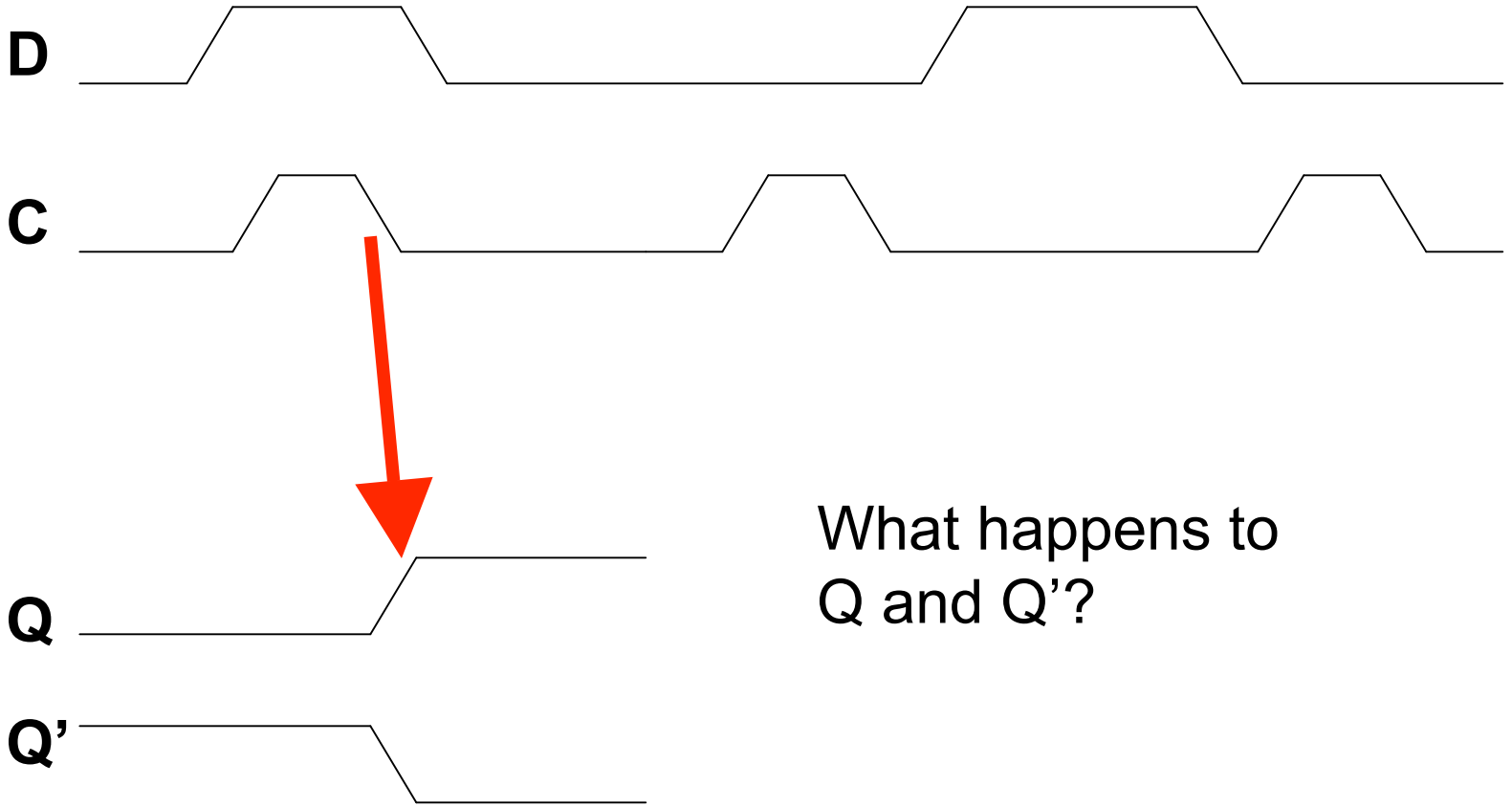


Q

Q'

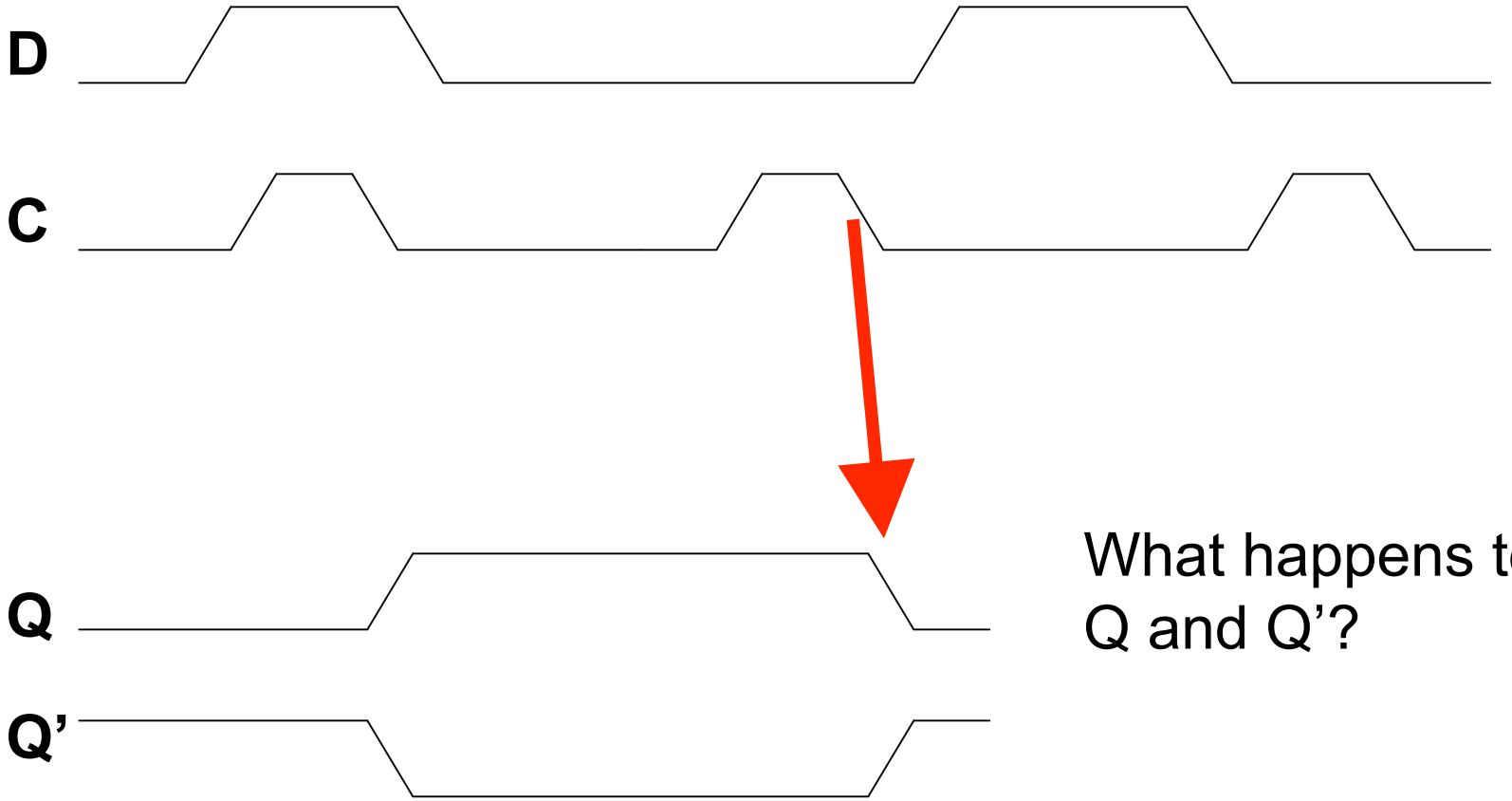
What happens to
Q and Q'?

D Flip Flop



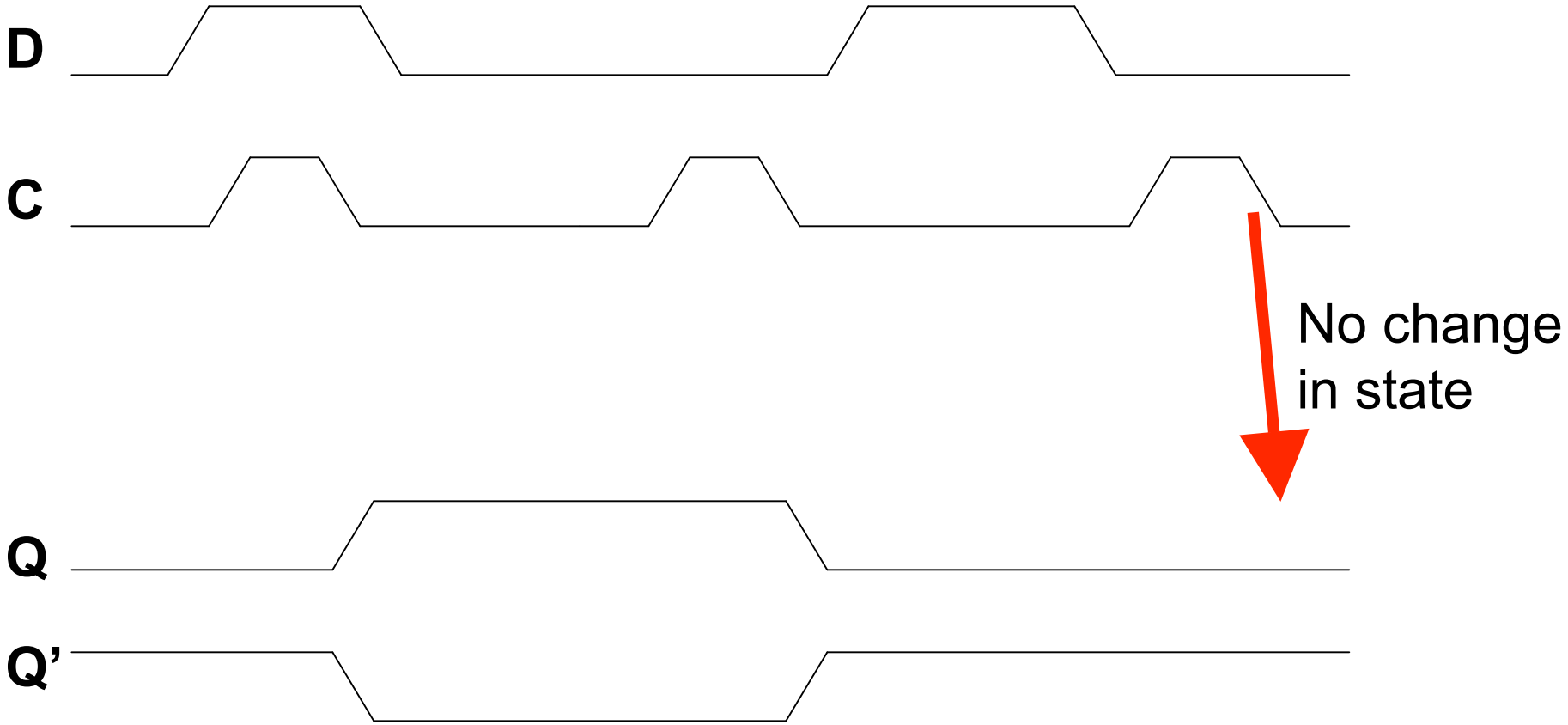
What happens to
Q and Q'?

D Flip Flop

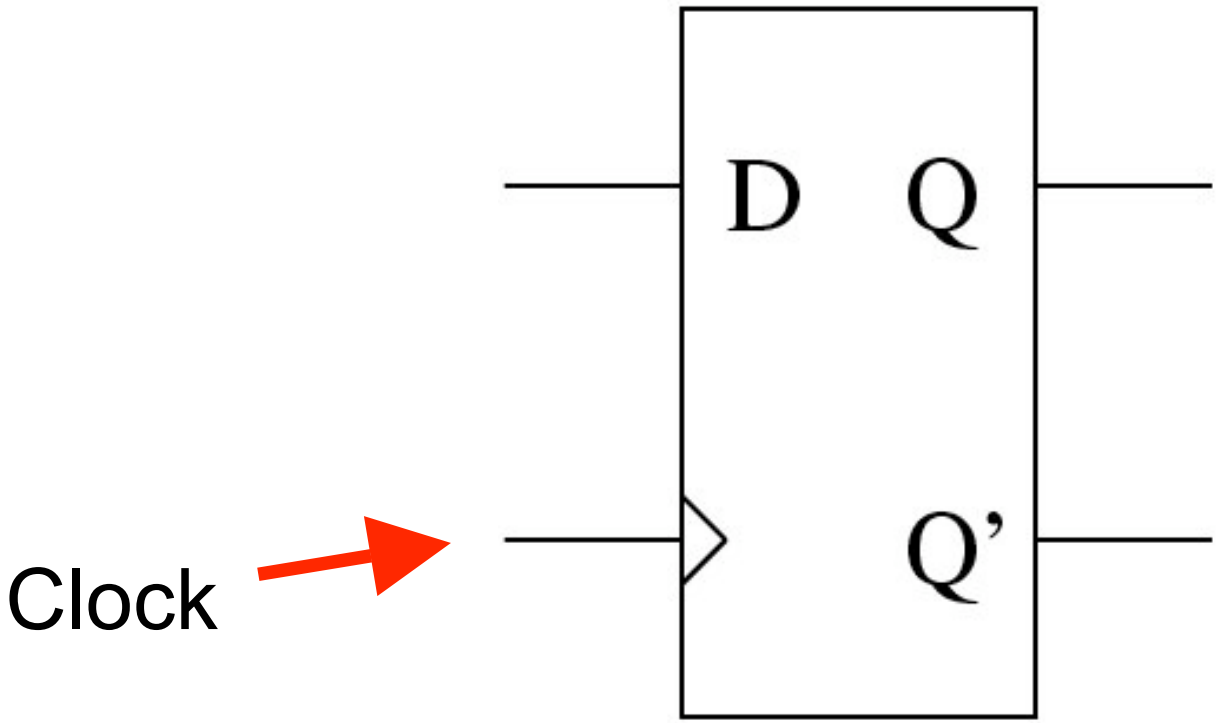


What happens to Q and Q'?

D Flip Flop

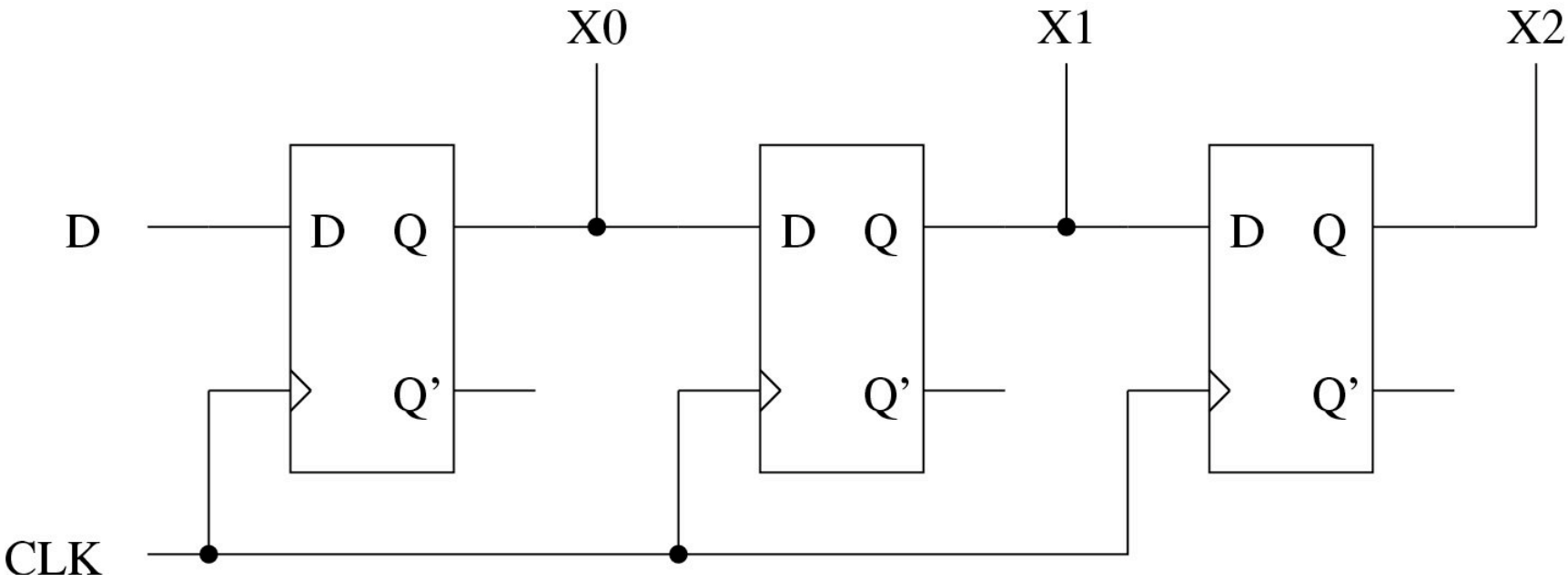


D Flip Flops



An Application of D Flip Flops

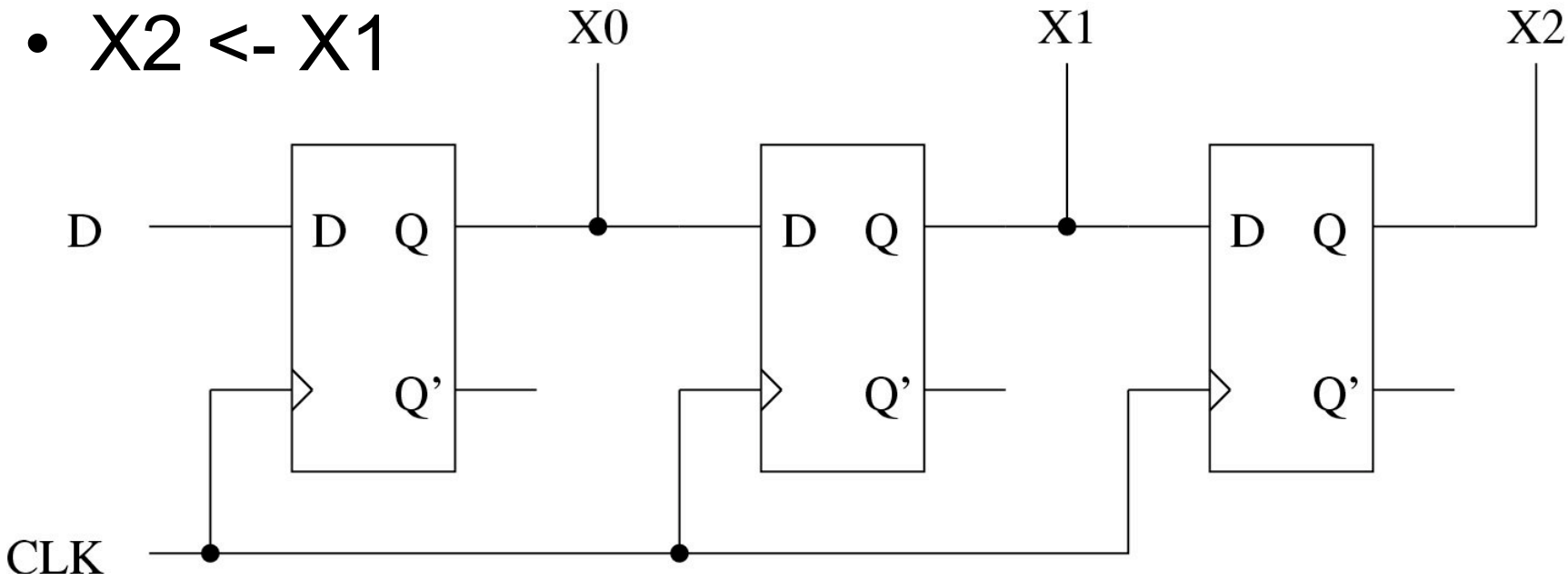
What does this circuit do?



Shift Register

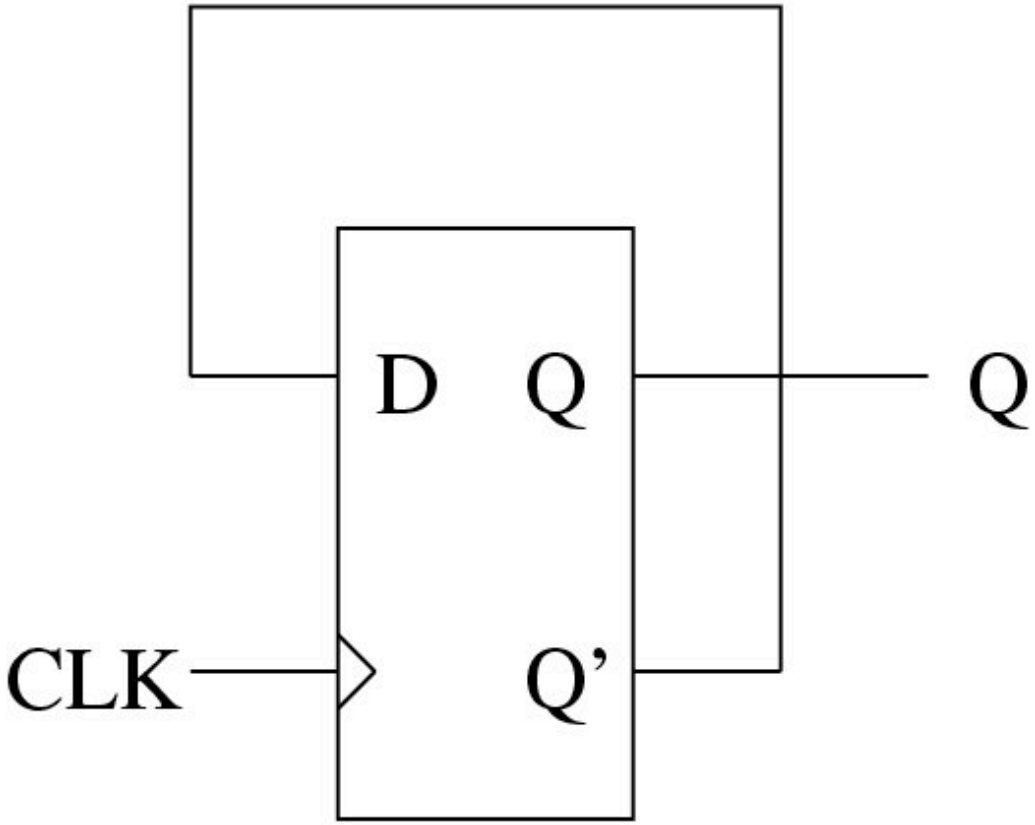
On each clock transition from high to low:

- X0 takes on the current value of D
- X1 <- X0
- X2 <- X1



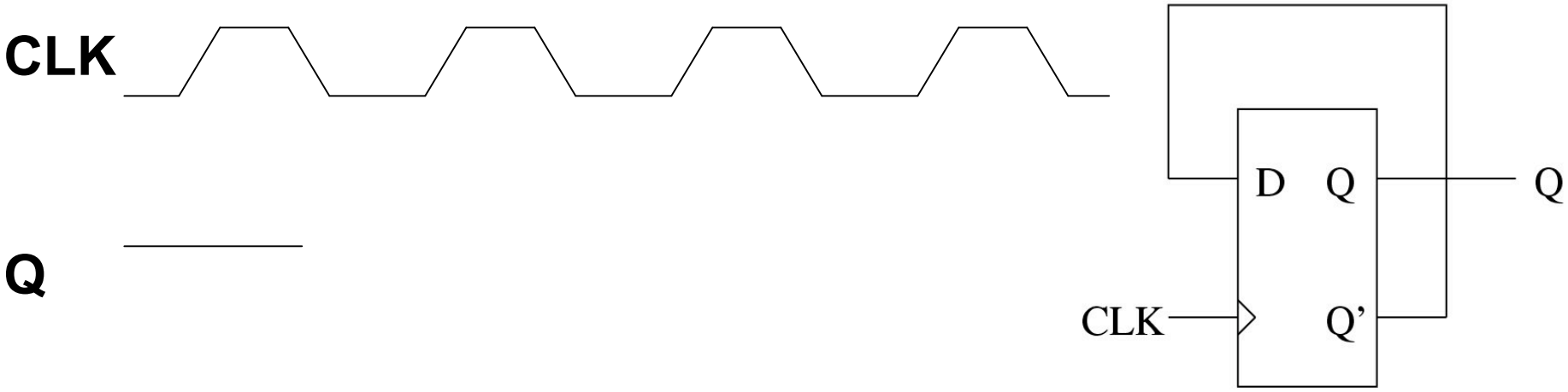
Another D Flip Flop Circuit

How does this circuit behave?



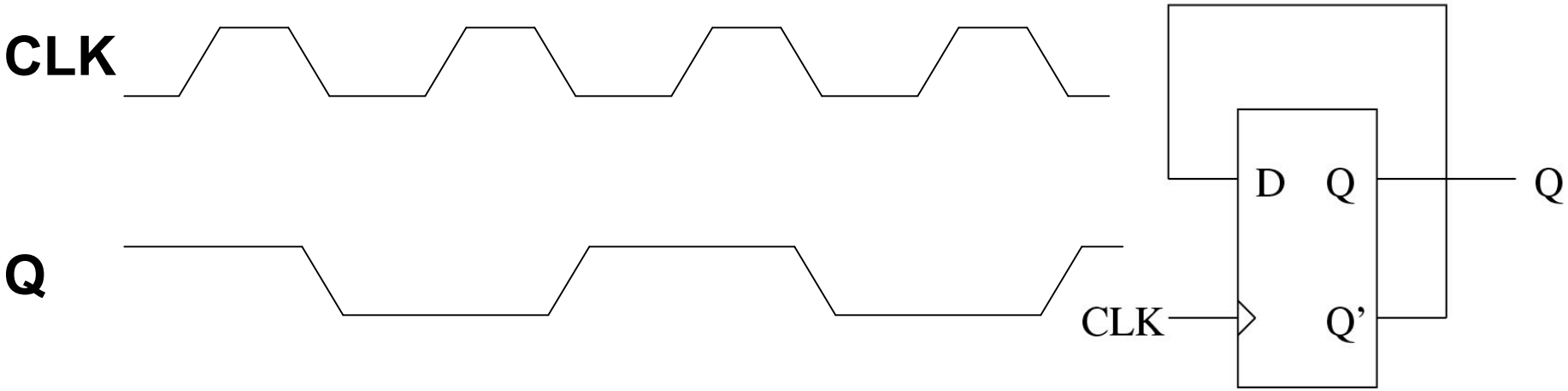
Frequency Divider

How does this circuit behave?



Frequency Divider

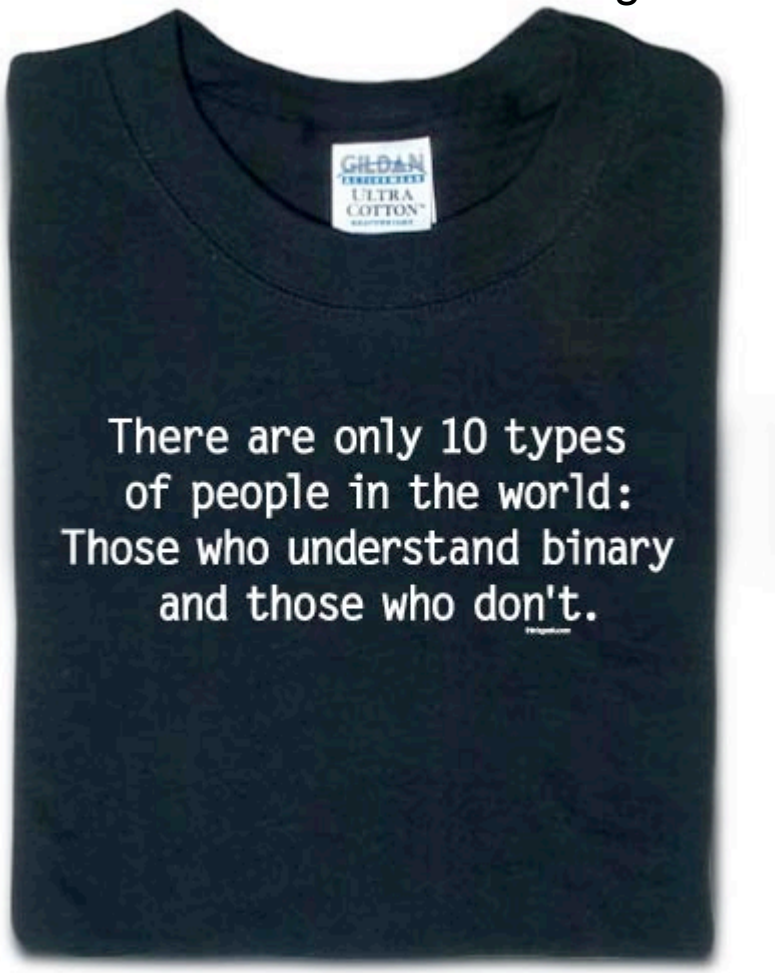
Q flips state on every downward edge of the clock



A Bit About Binary Encoding

www.thinkgeek.com

If a boolean variable can only encode two different values, how do we represent a larger number of values?



Binary Encoding

How do we represent a larger number of values?

- As with our decimal number system: we concatenate binary digits (or “bits”) into strings

Binary Encoding

- The first (rightmost) bit is the 1's digit
- The second bit is the 2's digit
- The ith bit is the 2^{i-1} 's digit

Binary Encoding

How do we convert from binary to decimal in general?

B2	B1	B0		decimal
0	0	0		0
0	0	1		1
0	1	0		2
0	1	1		3
1	0	0		4
1	0	1		5
1	1	0		6
1	1	1		7

Last Time

Sequential Logic

- D Flip Flops
- Shift registers
- Ripple Counters

Binary number system

Today

- A little more on number systems
- Arithmetic operators
- Representing negative numbers
- Multiplication with shift registers
- Arithmetic logic units

Administrivia

- Homework 1 due in 1 week
- Project 1:
  - One robot is now up and stable
  - A complete set of power supplies will be available today

Binary to Decimal Conversion

$$value = B_0 + B_1 * 2^1 + B_2 * 2^2 + B_3 * 2^3 + \text{K}$$

$$value = \sum_{i=0}^{N-1} B_i * 2^i$$

How do we convert from decimal to binary?

Decimal to Binary Conversion

$$\forall i : B_i \leftarrow 0$$

$$while(value \neq 0)$$

$$\{$$

$$Find\ i\ suchthat\ 2^{i+1} > value \geq 2^i$$

$$B_i \leftarrow 1$$

$$value \leftarrow value - 2^i$$

$$\}$$

Binary Counter

How would we build a circuit that counts the number of clock ticks that have gone by?

B2	B1	B0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Binary Counter

How would we build a circuit that counts the number of clock ticks that have gone by?

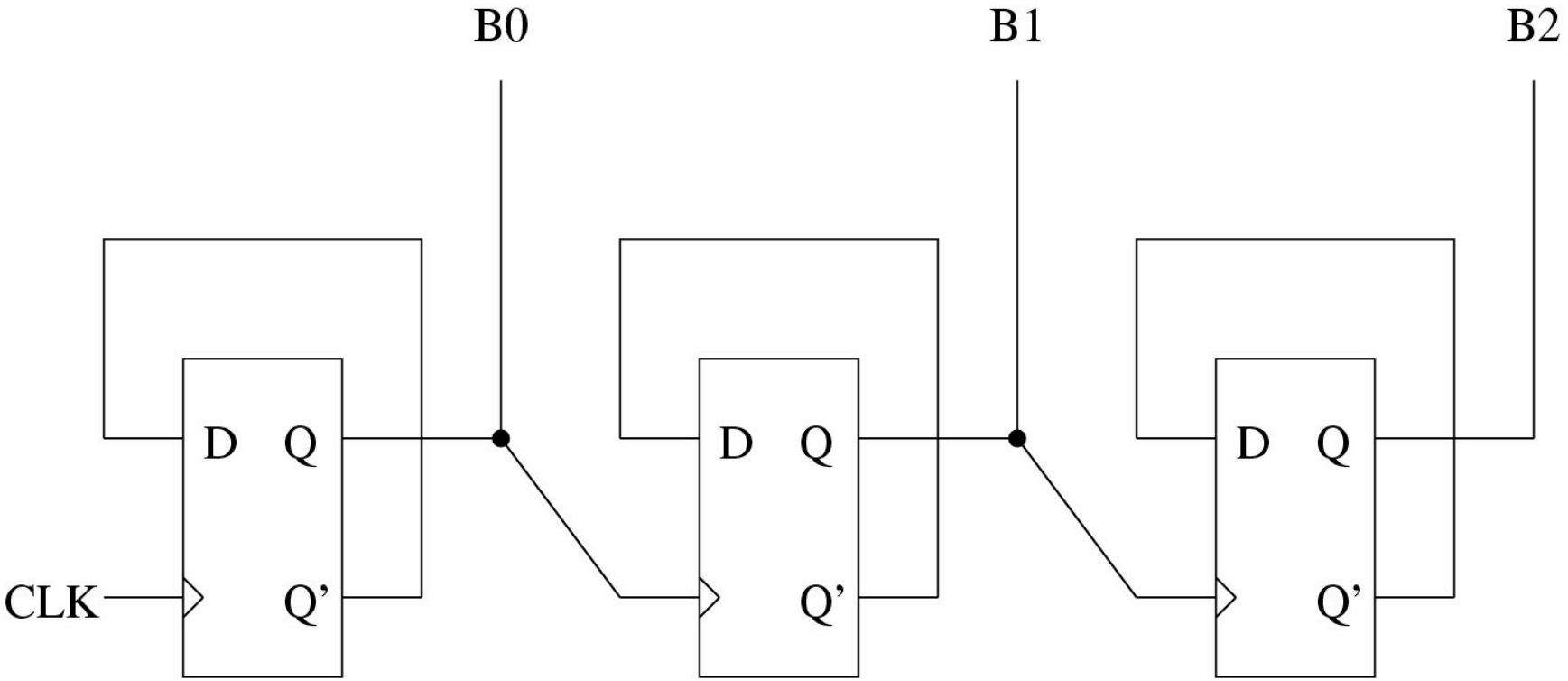
Insight:

- B1 changes state at half the frequency that B0 does
- B2 changes state at half the frequency of B1

B2	B1	B0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Ripple Counter

The carry “ripples” down the chain …



J-K Flip Flops

Behave similarly to R-S flip flops, but:

- Deal properly with the case where both R and S inputs are 1
  - The R-S flip flop will arbitrarily choose one of the possible output states
- The master latch (on the input side) can only change state once while the clock is high

T Flip Flops

- J-K flip flop with R and S tied high
- Every downward clock edge causes the flip flop to change state

- This is just like our D flip flop with D connected to Q’

Next Time

Binary Arithmetic:

- Addition
- Representing negative numbers & subtraction
- A little bit on multiplication

Other number systems

- Octal
- Hexadecimal