Time

Until now: we have essentially ignored the issue of time

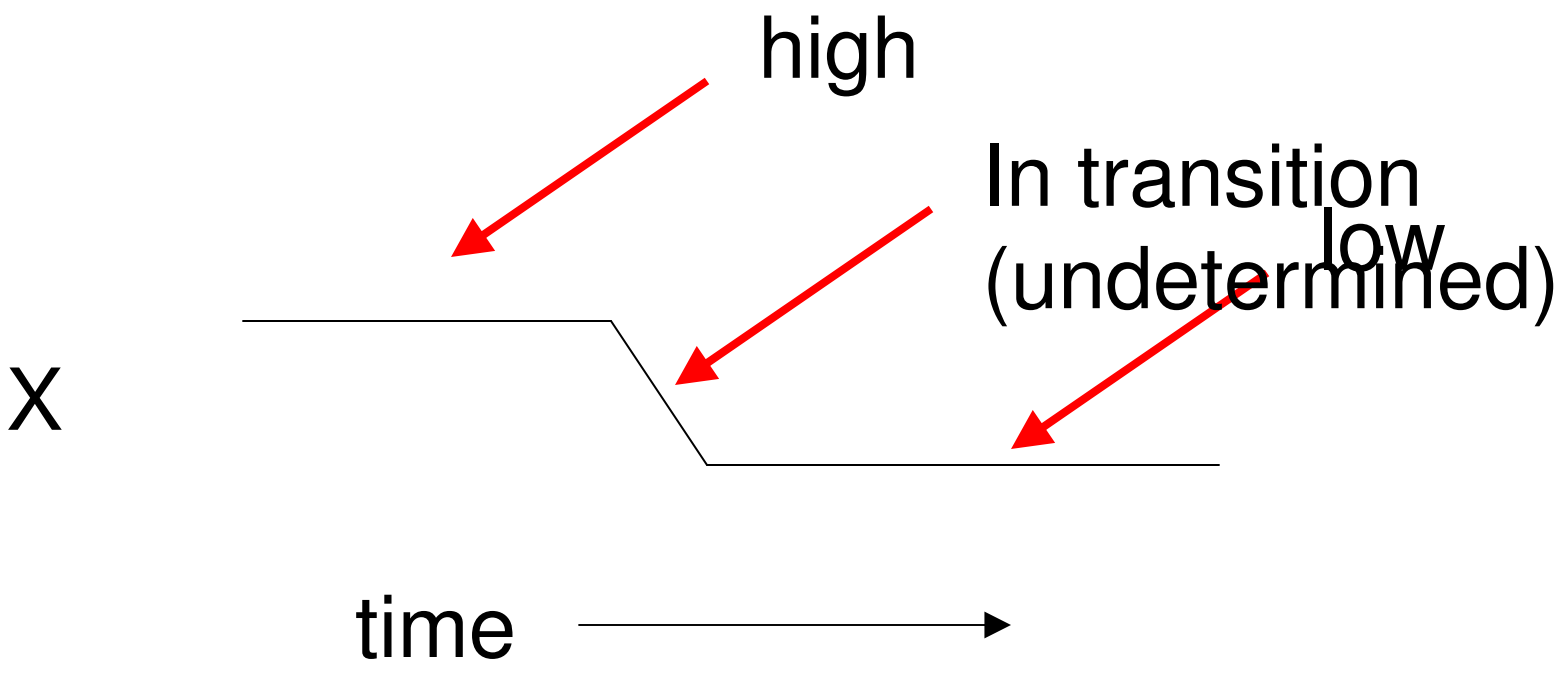
- We have assumed that our digital logic circuits perform their computations instantaneously
- Our digital logic circuits have been "stateless"
  - Once you present a new input, they forget everything about previous inputs

Time

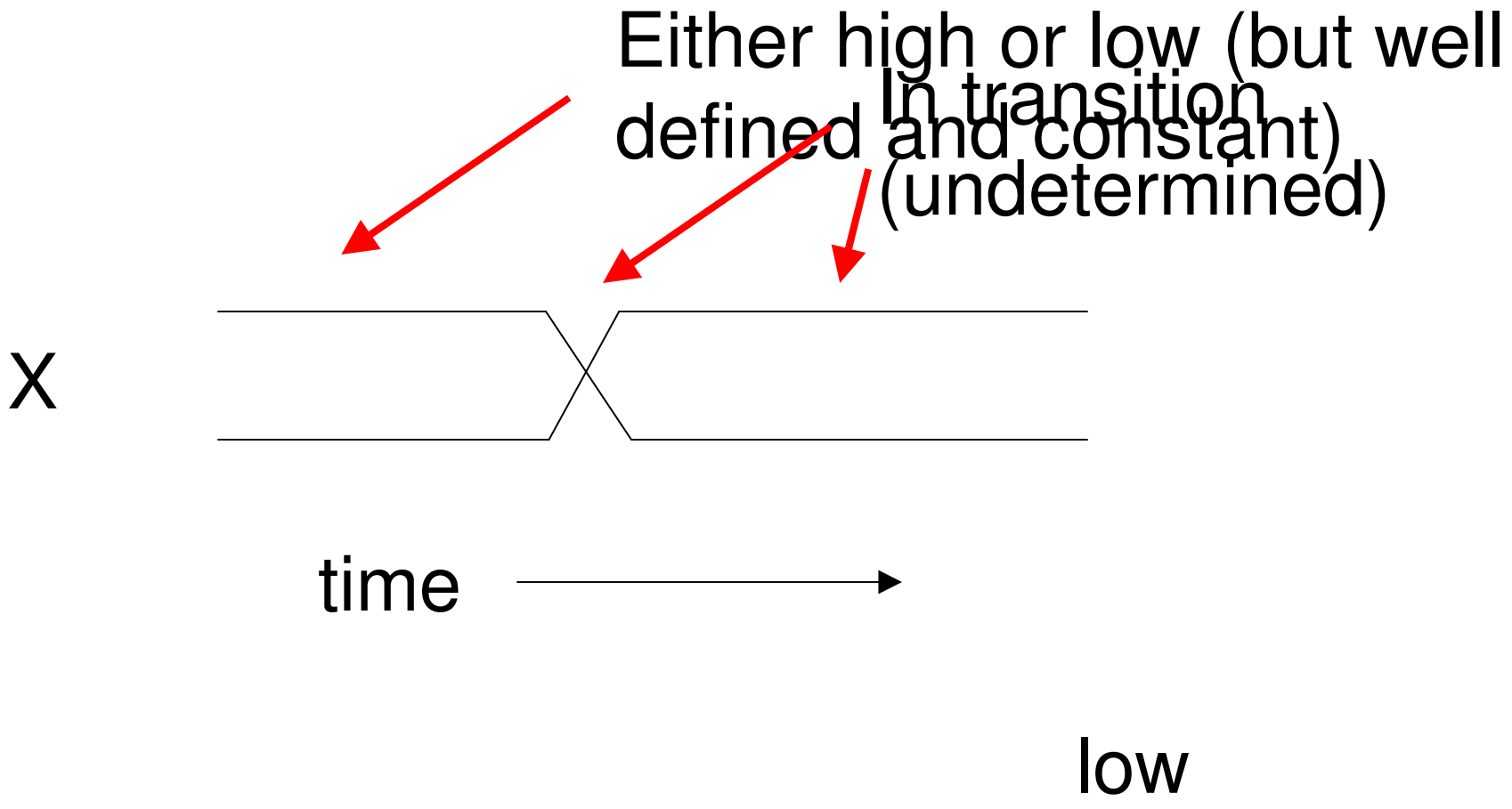
In reality, time is an important issue:

- 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



Today

- Project 1 continued
- Sequential logic

- Homework 3 will be out next Tuesday

Beacon Receiver

The preprocessor translates the IR sensor signal into a 2-bit number

The state of each IR sensor is encoded with its own pair of bits

B1	B0		Semantics
0	0		No signal
0	1		Low signal
1	0		Medium signal
1	1		Strong signal

Robot Control Interface

3 output lines will determine the motion of the robot

C2	C1	C0		Semantics
0	0	0		Stop
0	0	1		Forward
0	1	0		Backward
0	1	1		Left
1	0	0		Right
1	0	1		Forward-Right
1	1	0		Forward-Left
1	1	1		x

Robot Control Interface

2 output lines will determine the turret position

T1	T0		Semantics
0	0		Forward
0	1		Left
1	0		Right
1	1		x

Your Job

Design and build a controller from basic logic gates

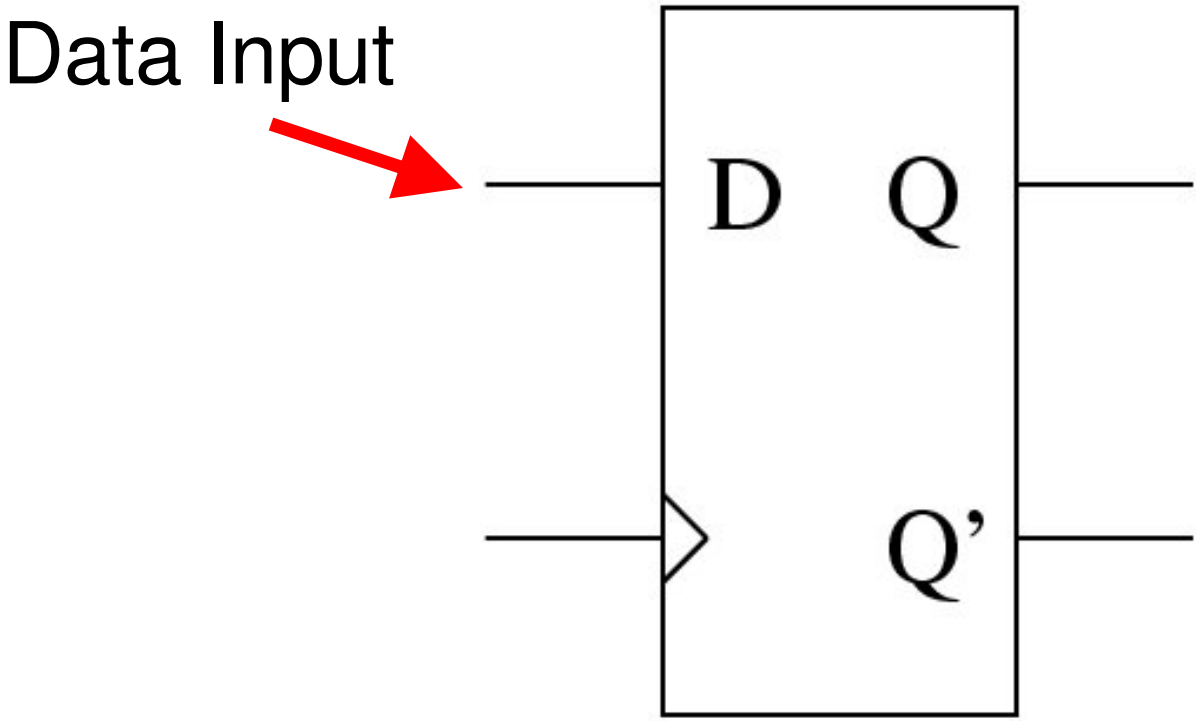
- Design the function: given each possible input from the sensors, what should the robot do?
- For each of these cases what command must you generate (C2, C1, C0, T1, T0)?
- What circuit will generate this command?
- Build the circuit

Group Design (Now)

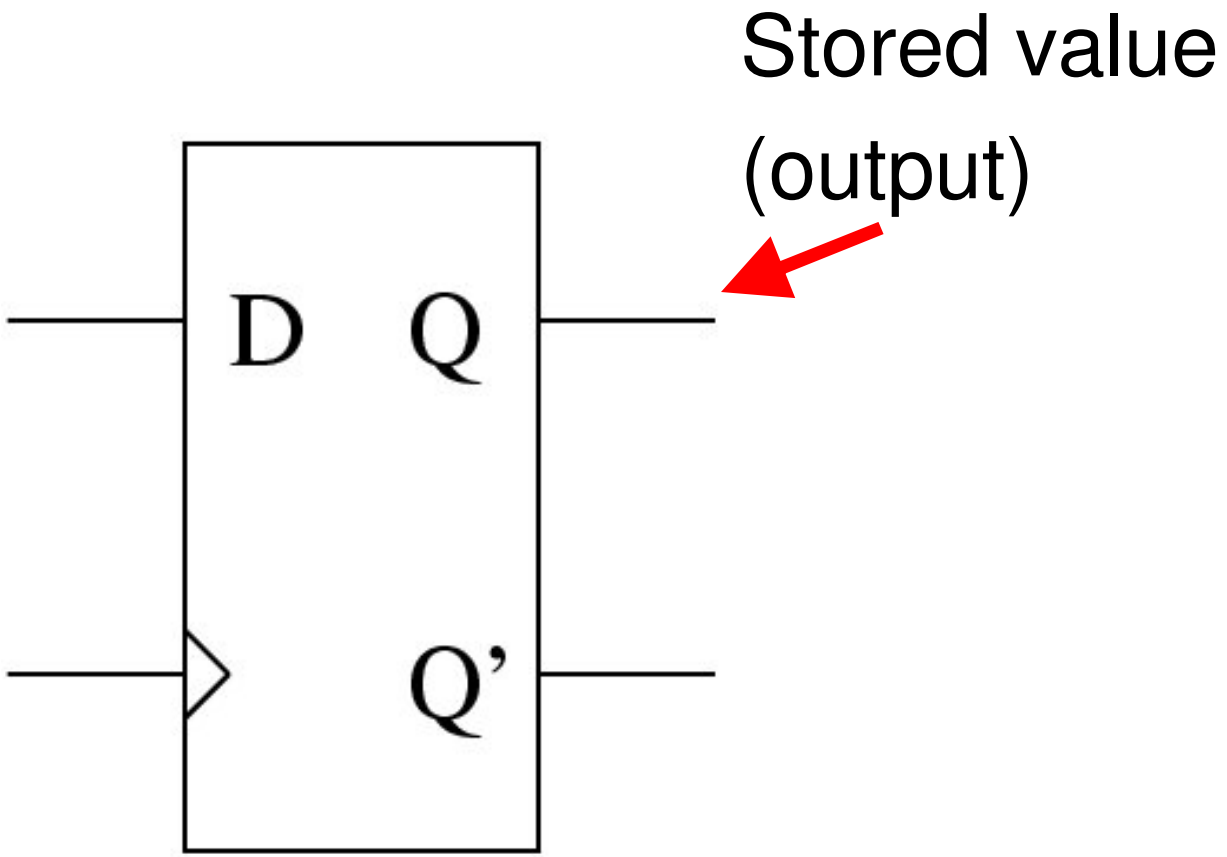
- What are the different possible sensor states?
- Can you simplify this set of states?
- A: For each state, what should the robot do?
- B: What is the truth table?
- C: Generate the K-maps

By end of class: hand in A, B & C

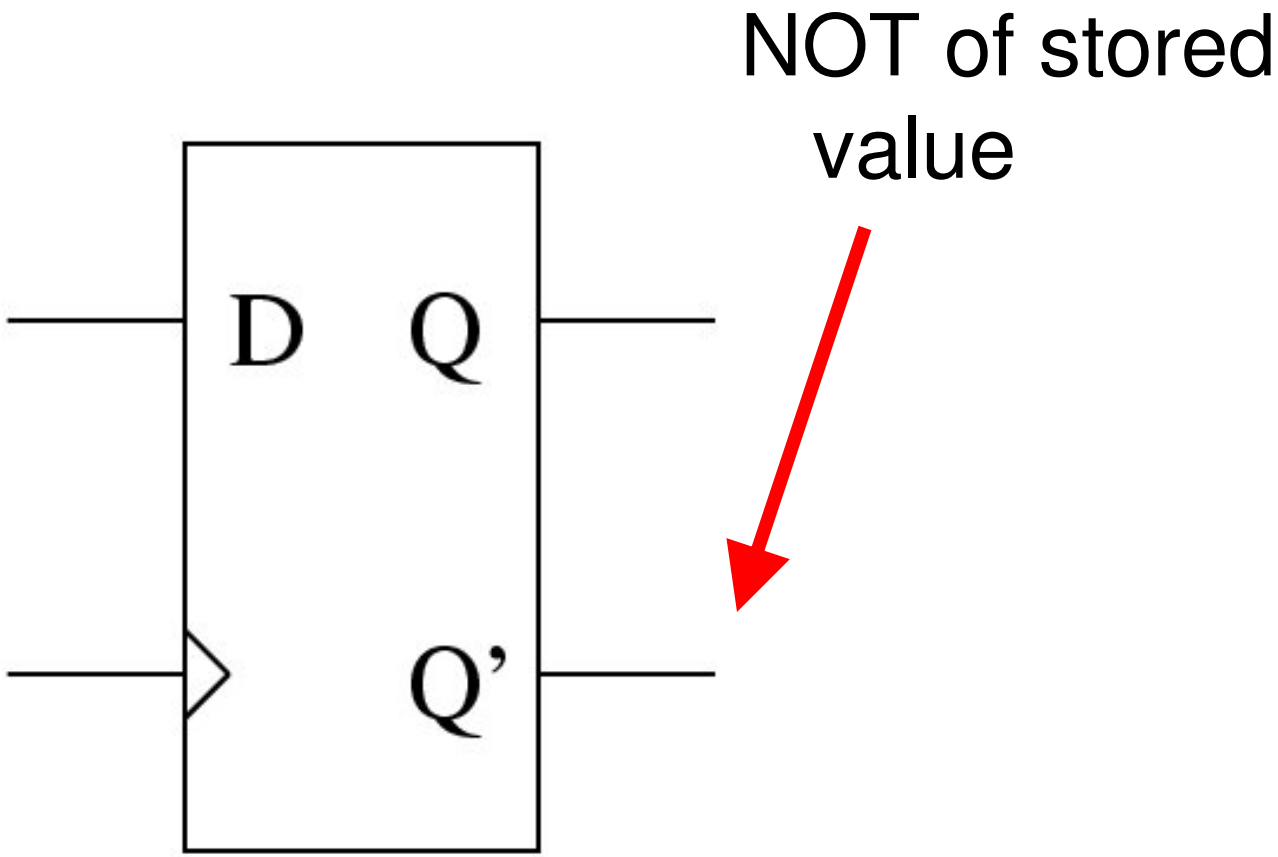
D Flip Flops



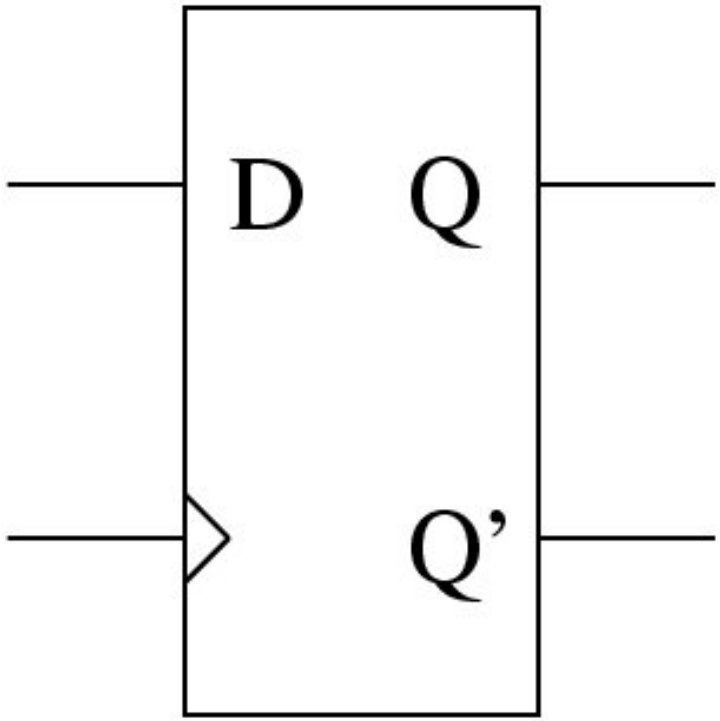
D Flip Flops



D Flip Flops

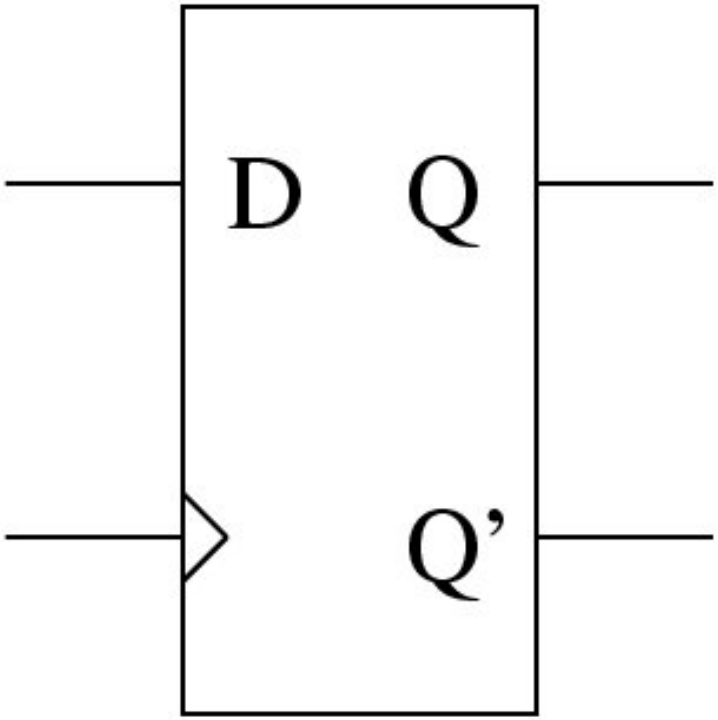


D Flip Flops



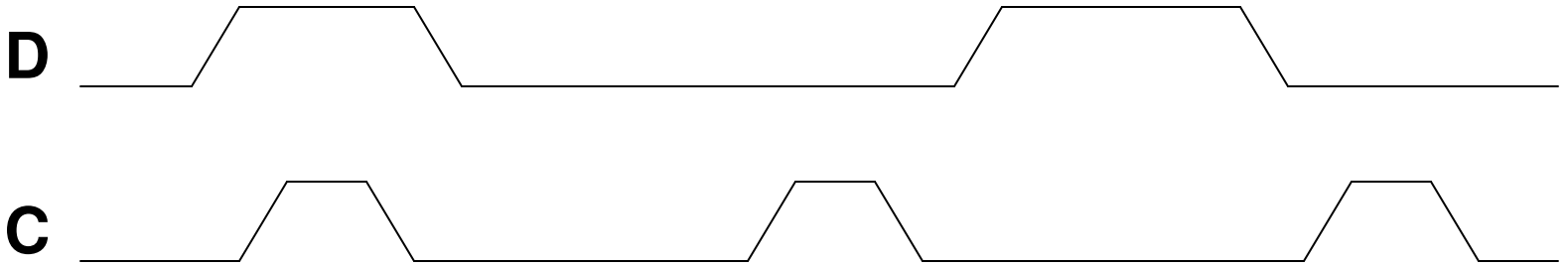
Clock input

D Flip Flops



When the clock transitions from high to low:
the value of D is stored

D Flip Flop

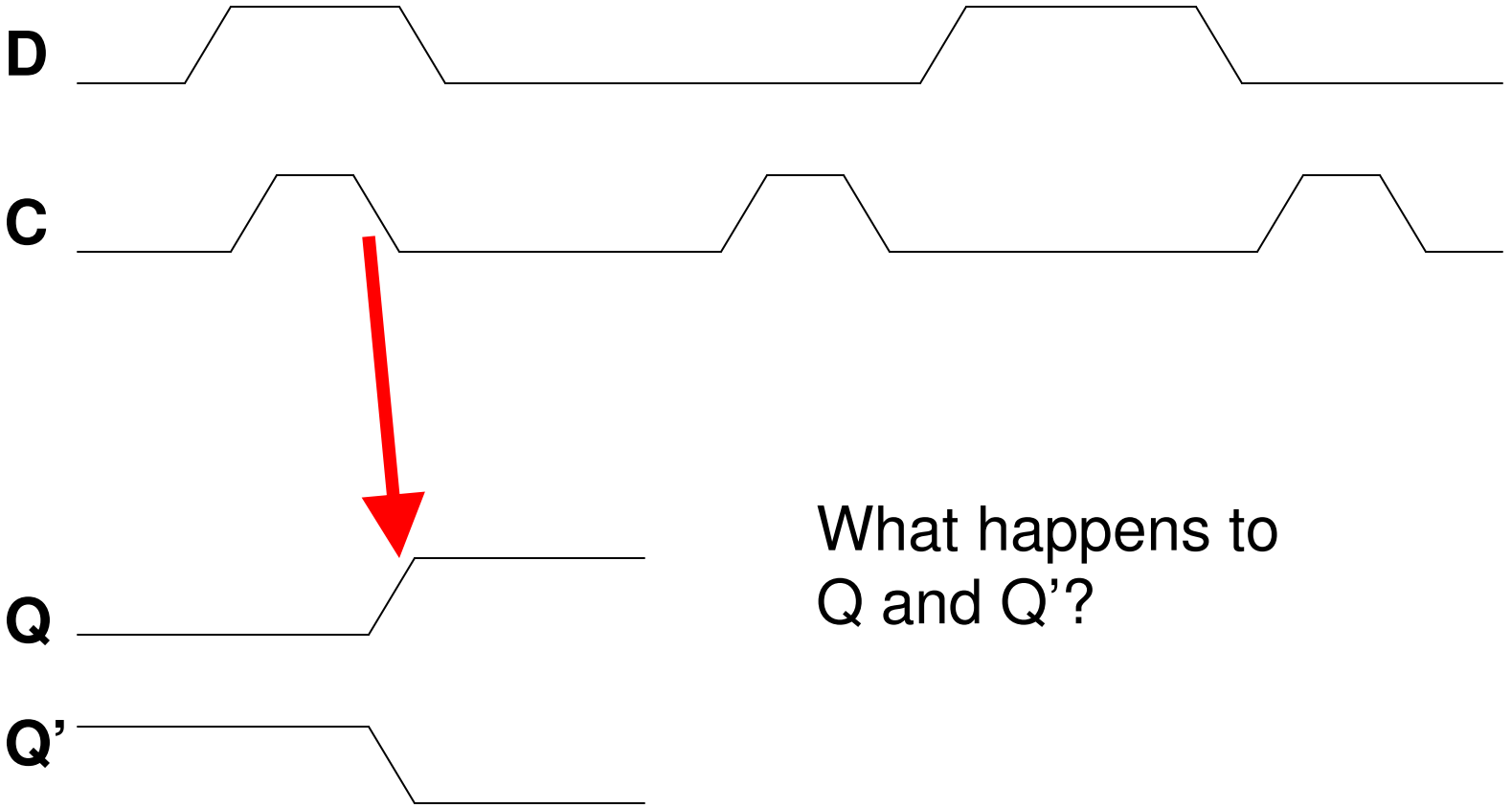


Q

Q'

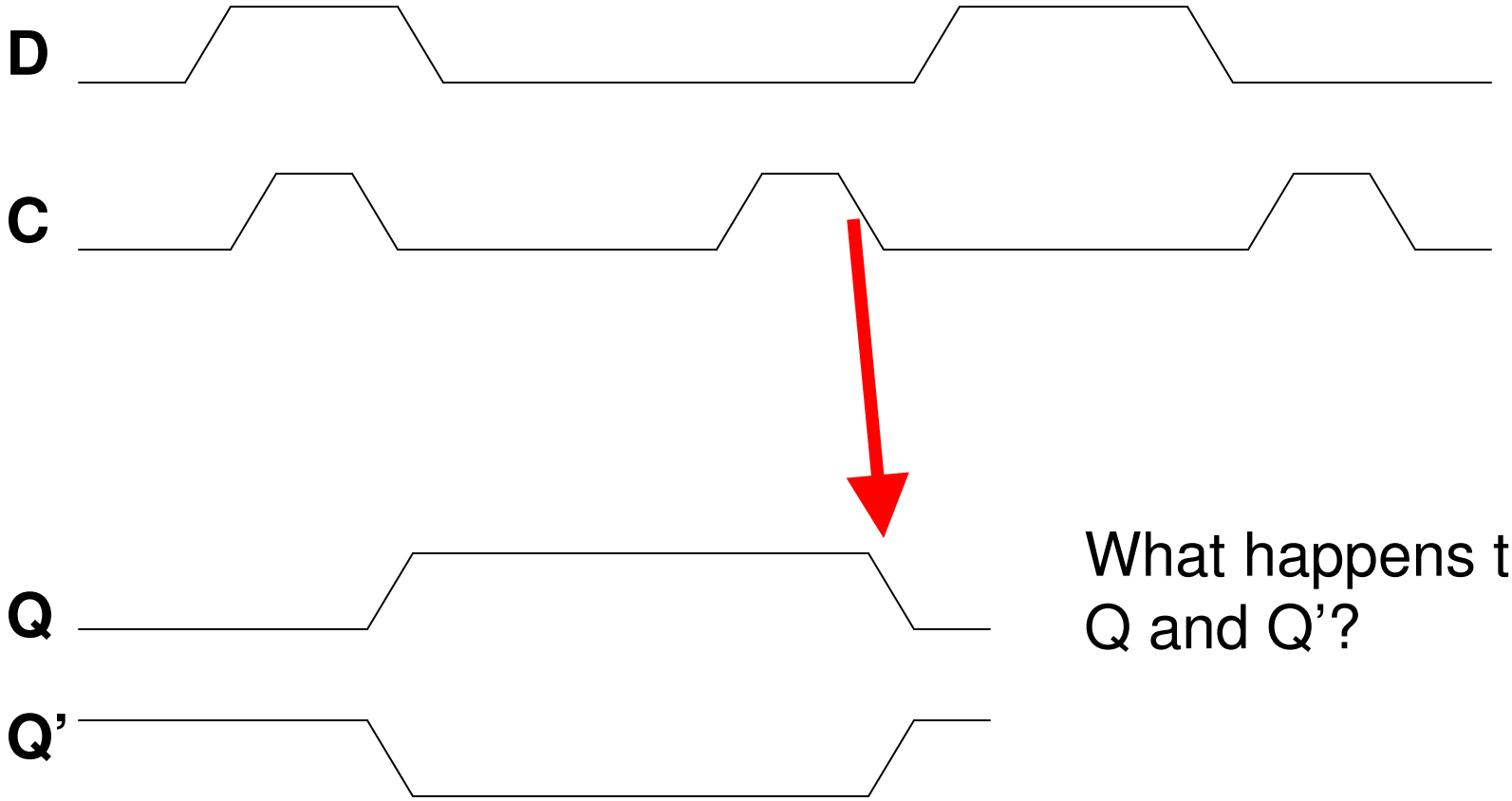
What happens to
Q and Q'?

D Flip Flop



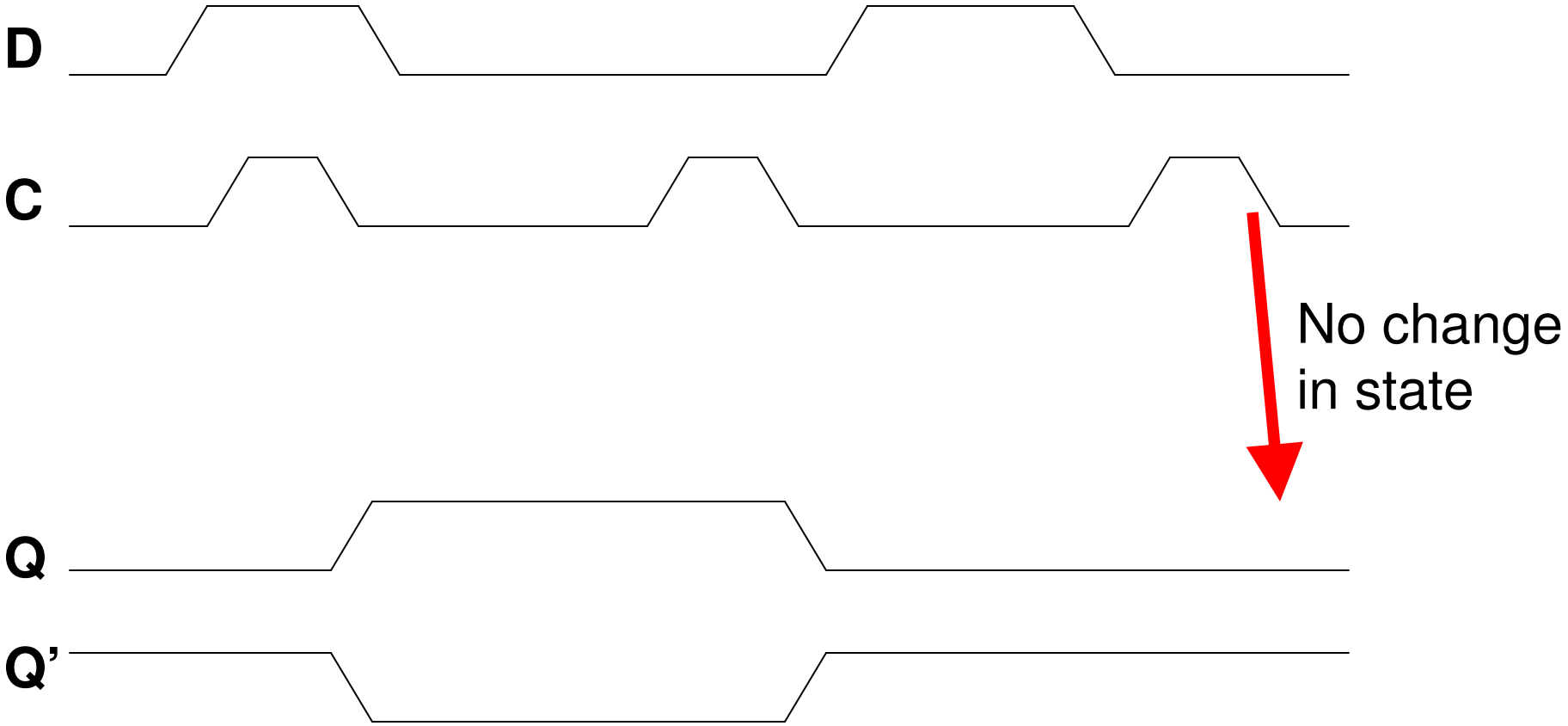
What happens to
Q and Q'?

D Flip Flop



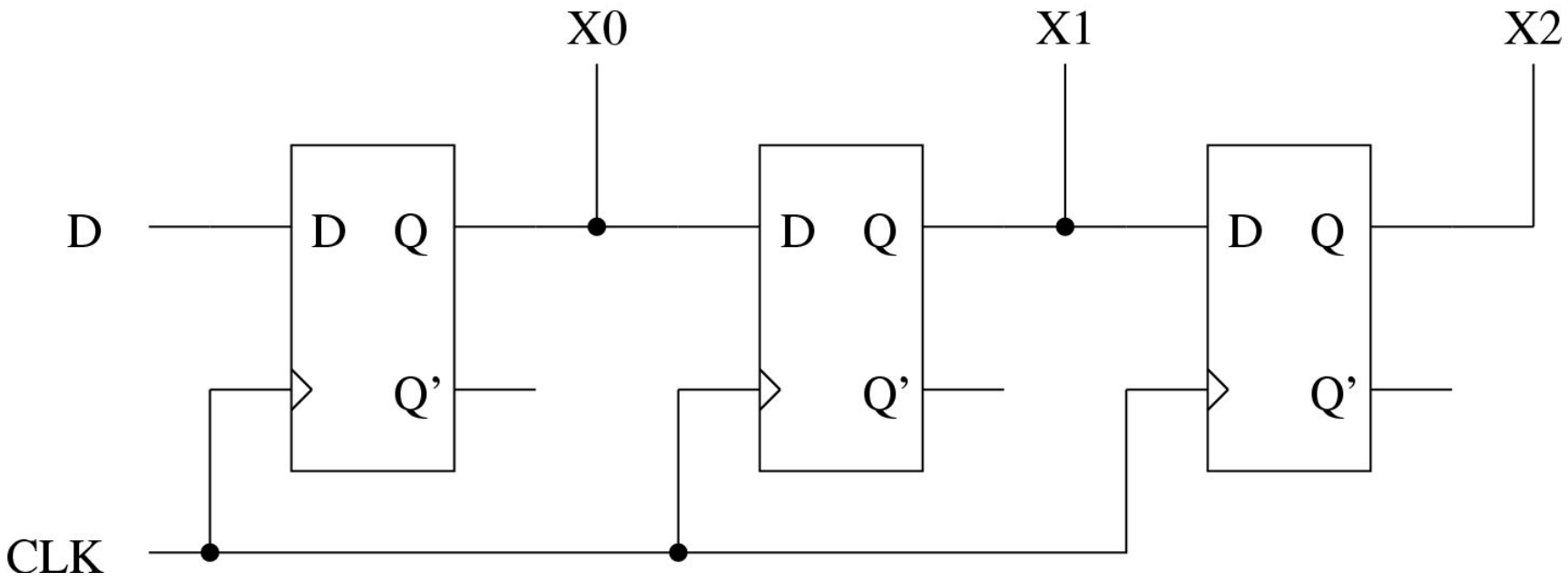
What happens to Q and Q'?

D Flip Flop



An Application of D Flip Flops

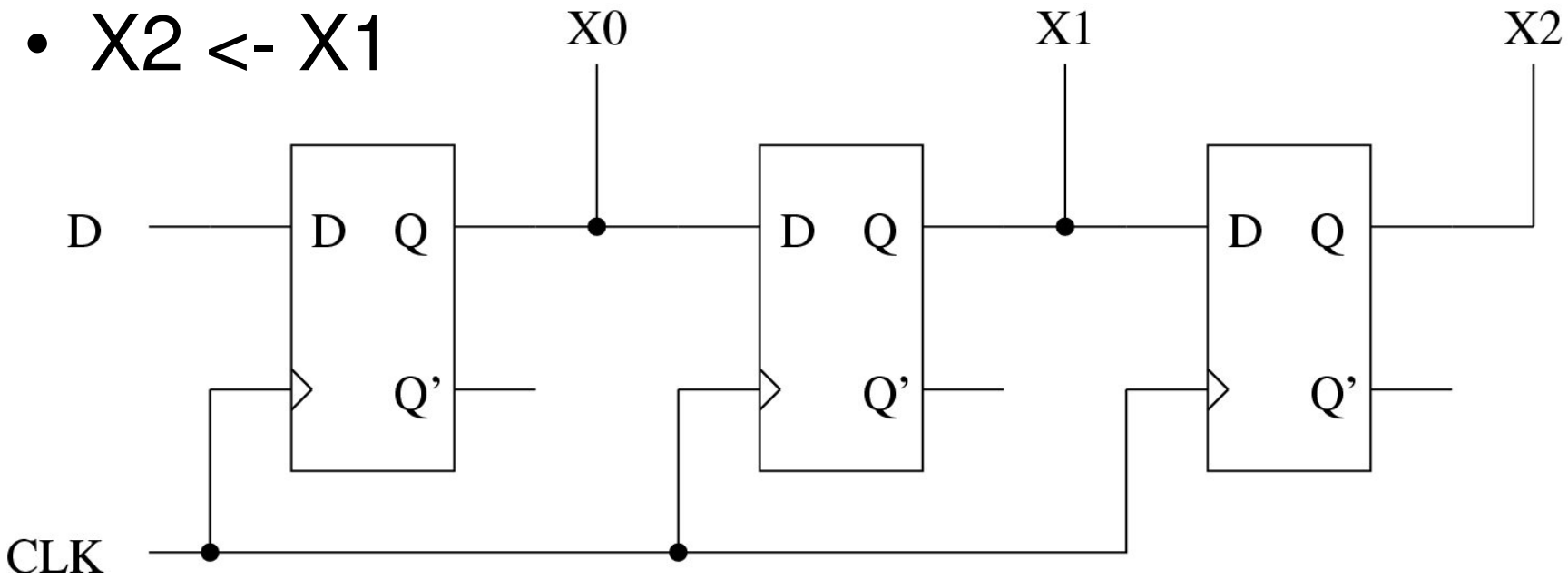
What does this circuit do?



Shift Register

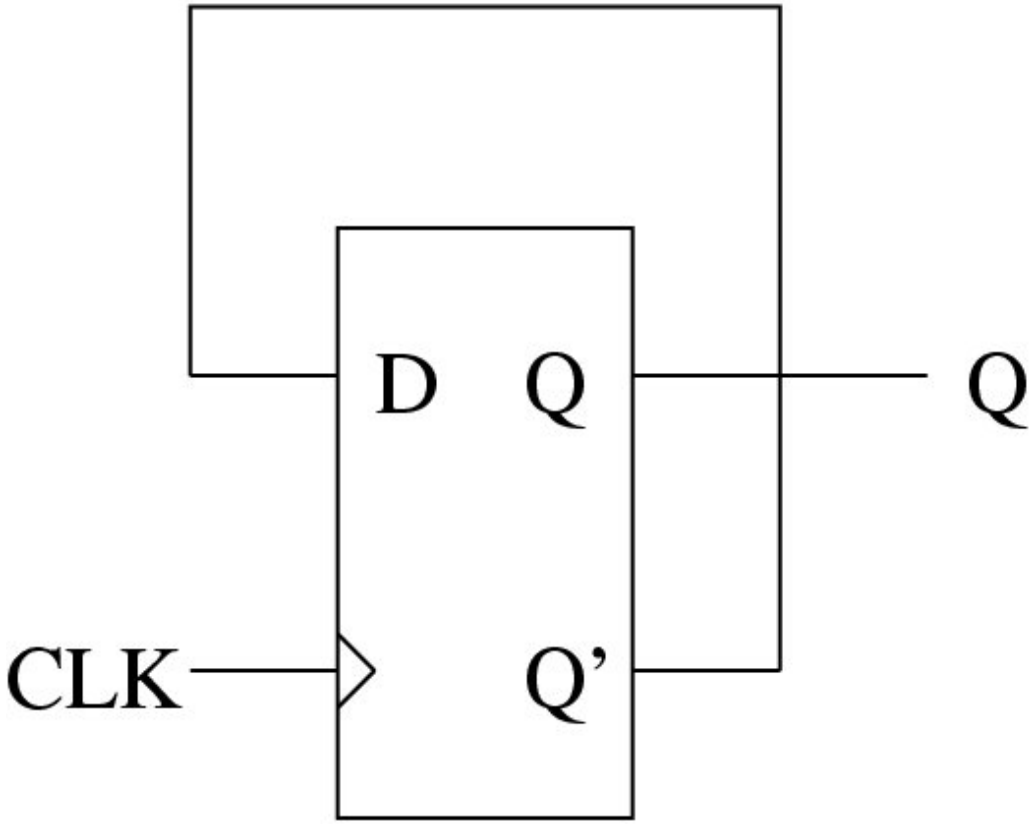
On each clock transition from high to low:

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



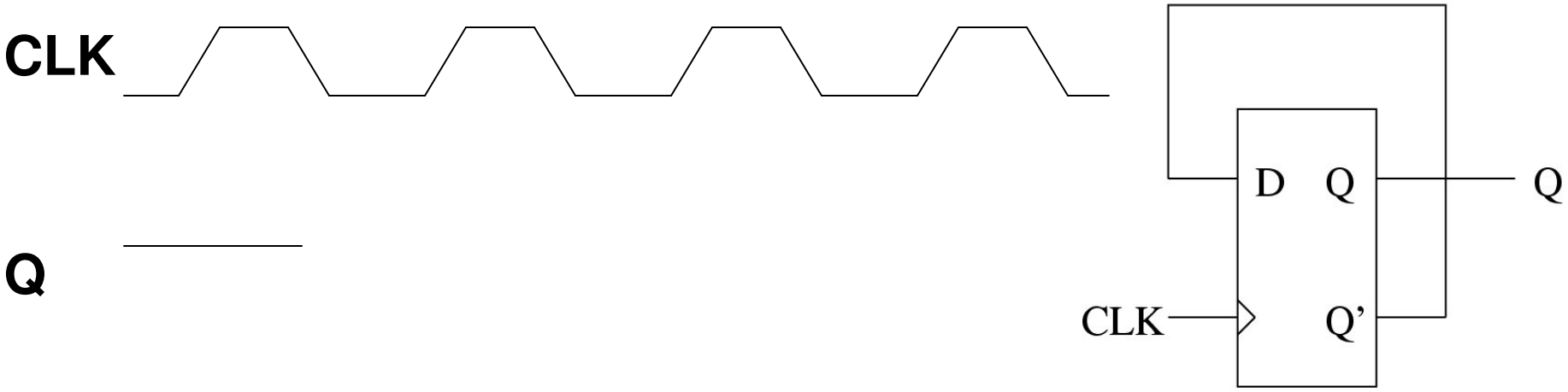
Another D Flip Flop Circuit

How does this circuit behave?



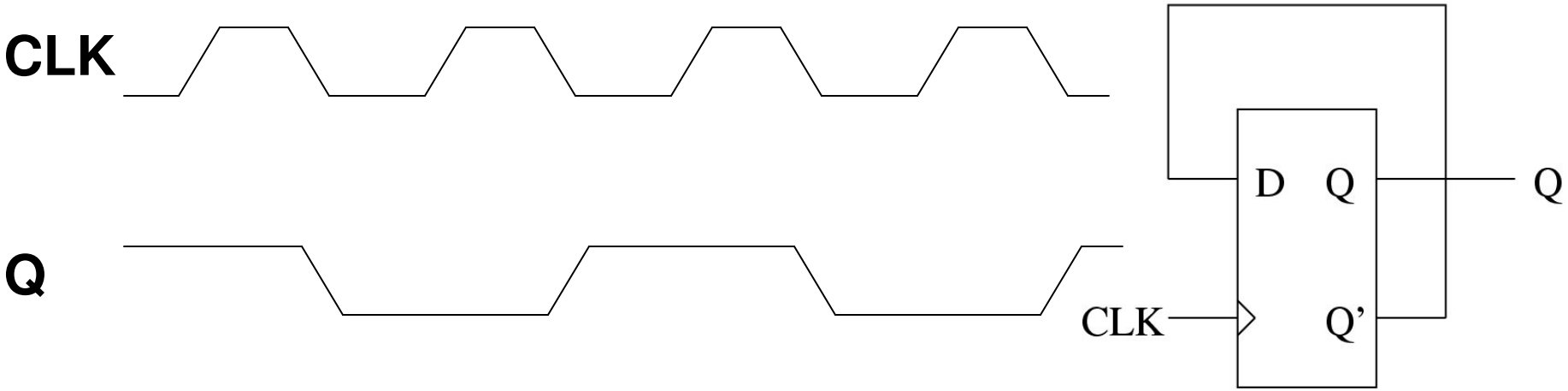
Frequency Divider

How does this circuit behave?



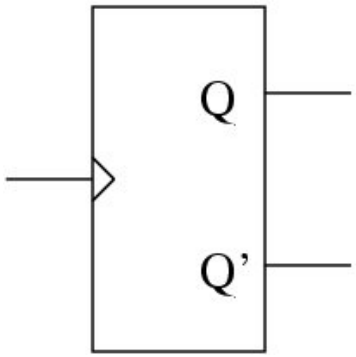
Frequency Divider

Q flips state on every downward edge of the clock

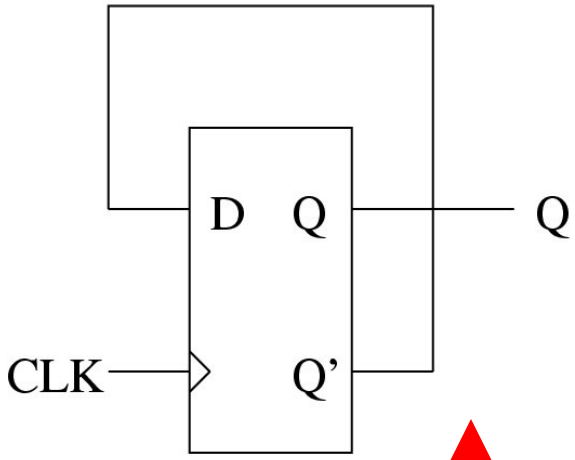


T Flip-Flops

T flip-flops change state every time the clock transitions from high to low



CLK

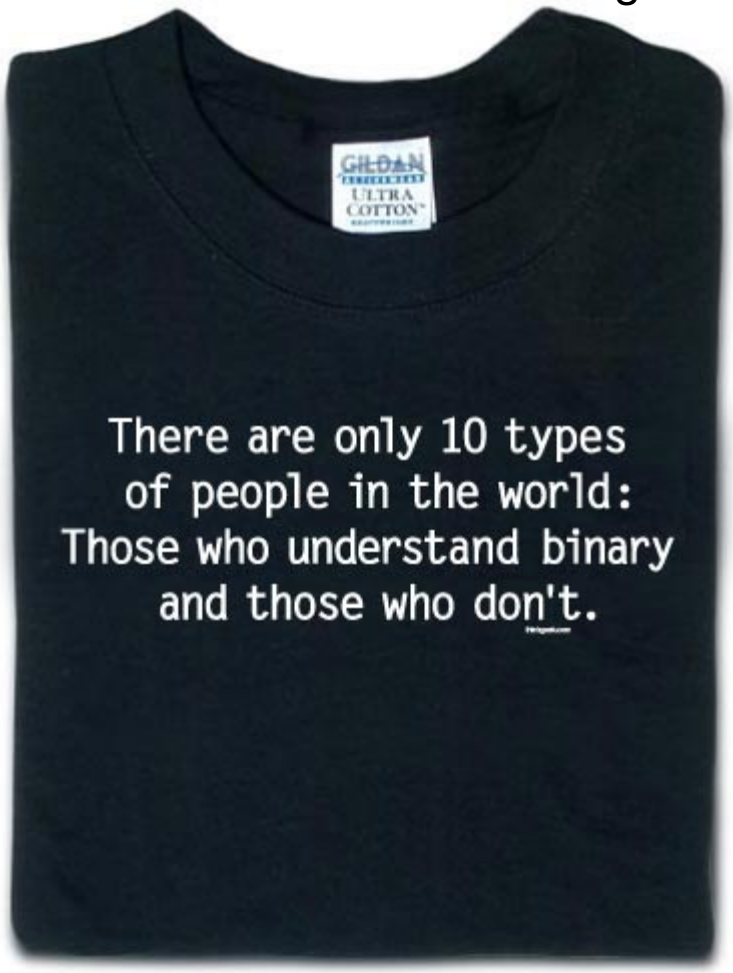


This is the same behavior as our D flip-flop circuit

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
- Binary number system

Today

- A little more on number systems
- Use of flip-flops
- Homework 2
- NAND latches

Administrivia

- Homework 3 available tonight
- Project 1:
  - Four robots are now up and stable
  - Due in 9 days

Binary to Decimal Conversion

$$value = B_0 + B_1 * 2^1 + B_2 * 2^2 + B_3 * 2^3 + \ldots$$

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

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?

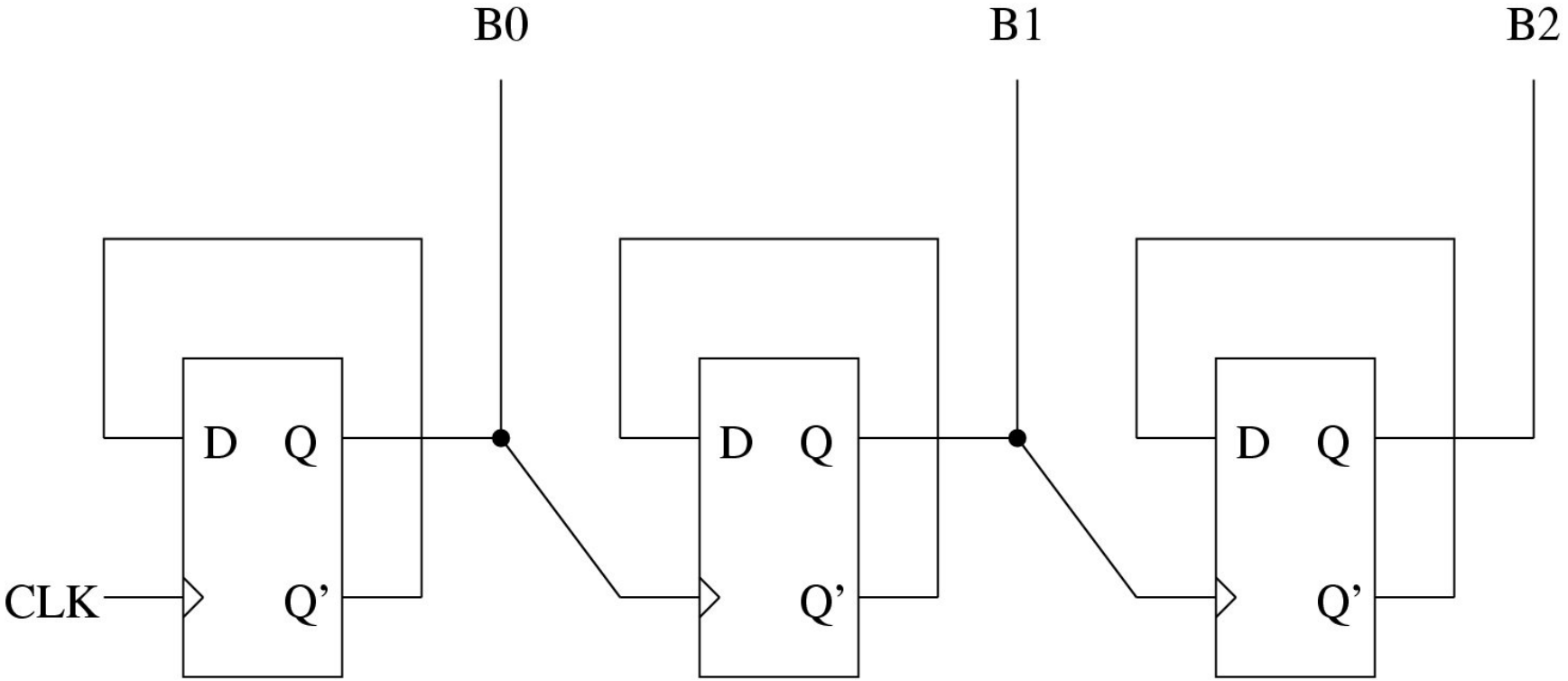
Insight:

- 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 …



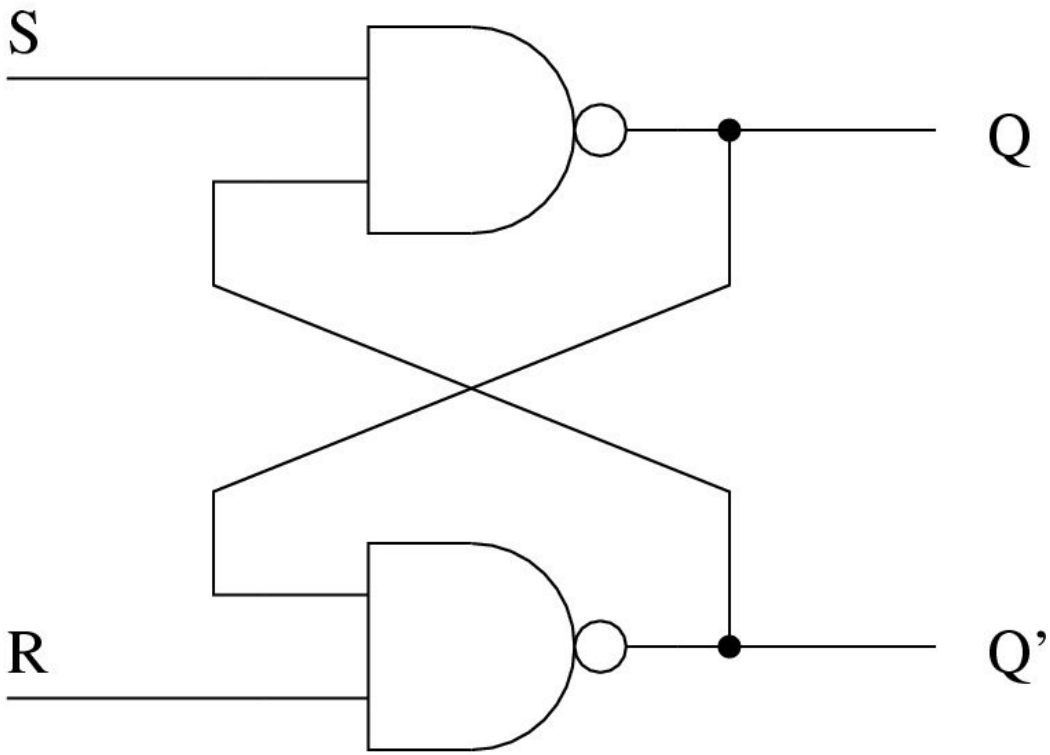
Ripple Counter

- Problem: the bits do not change state at the same time
- How would we go about designing a circuit to fix this?

Sequential Counter

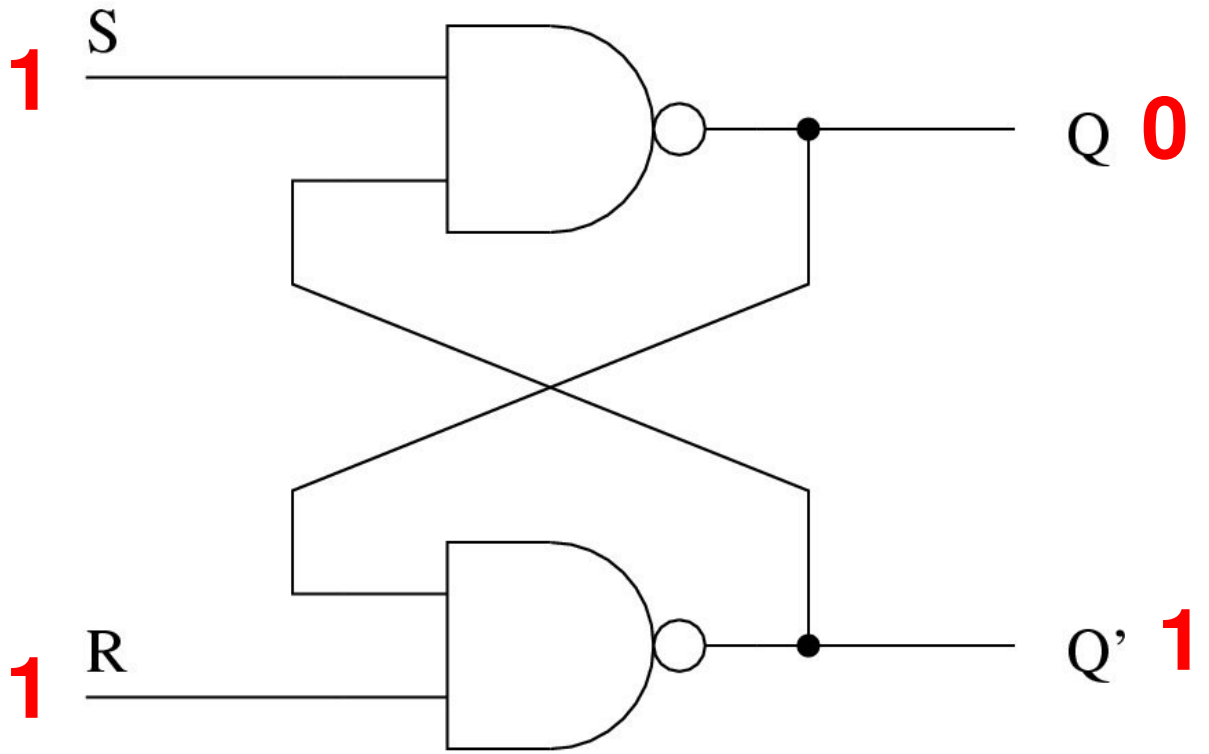
NAND Latch

What does this circuit do?



NAND Latch

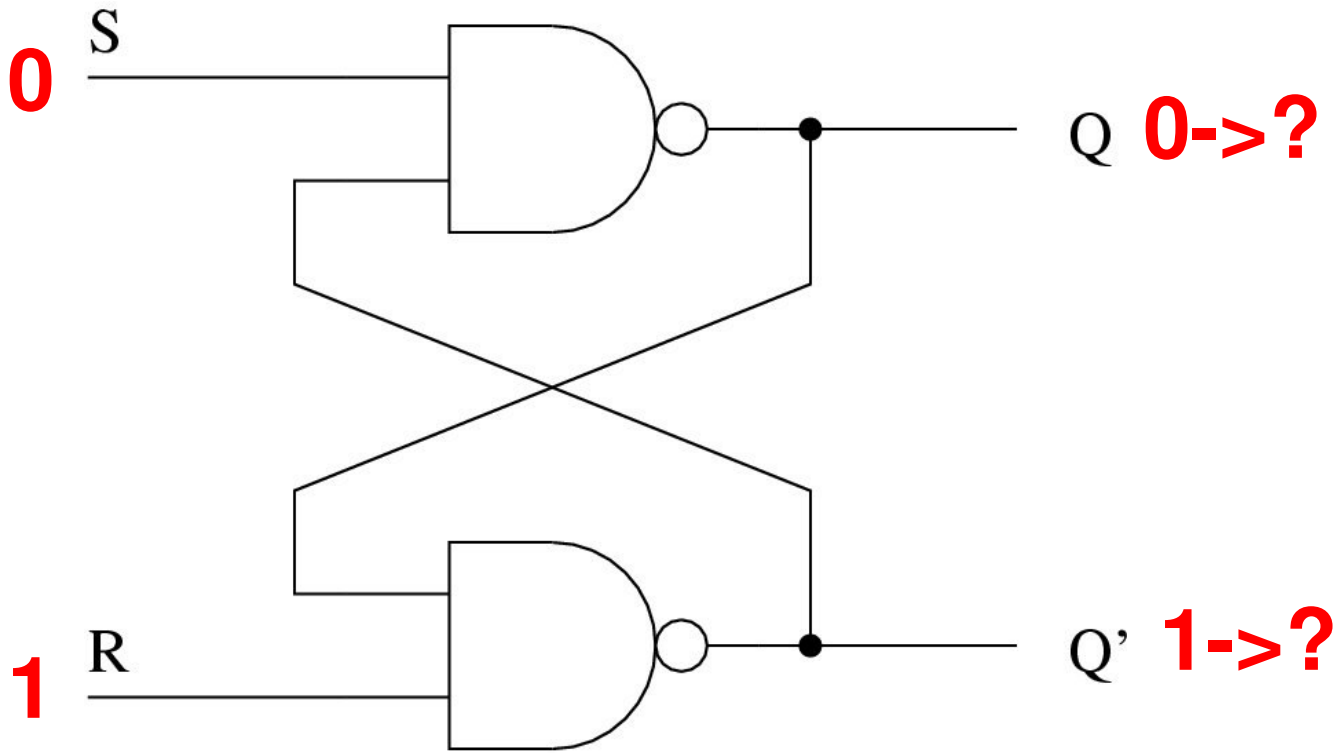
Consider this initial state



Is this a stable state? Yes!

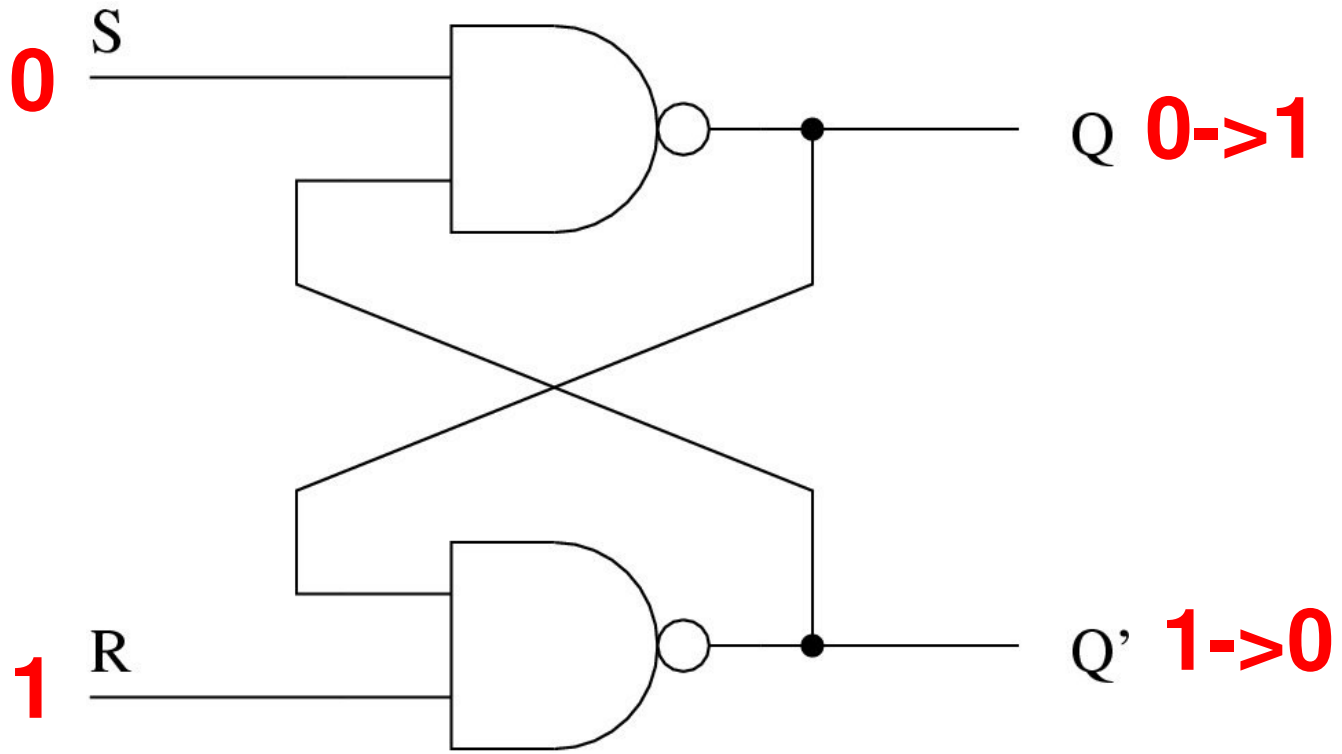
NAND Latch

What happens with S is set to 0?



NAND Latch

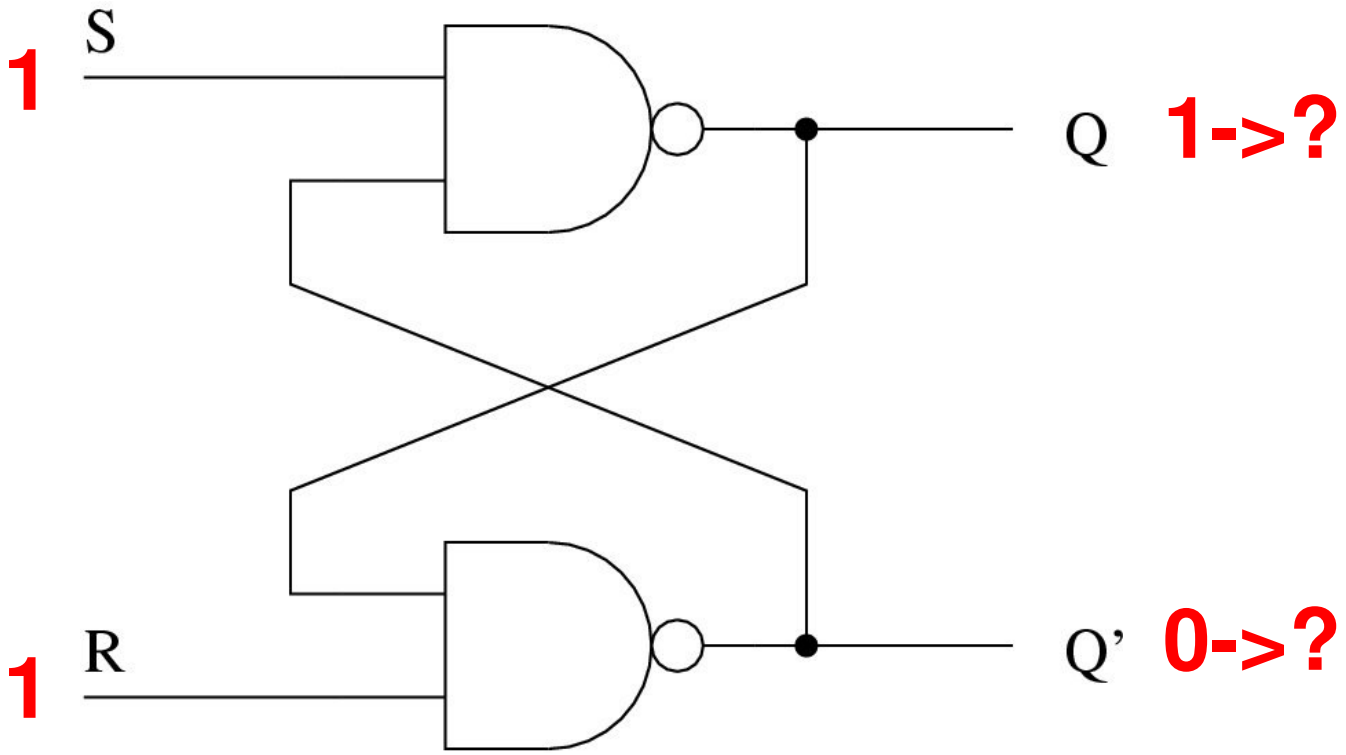
What happens with S is set to 0?



Q becomes 1 (thus S 'sets' Q)

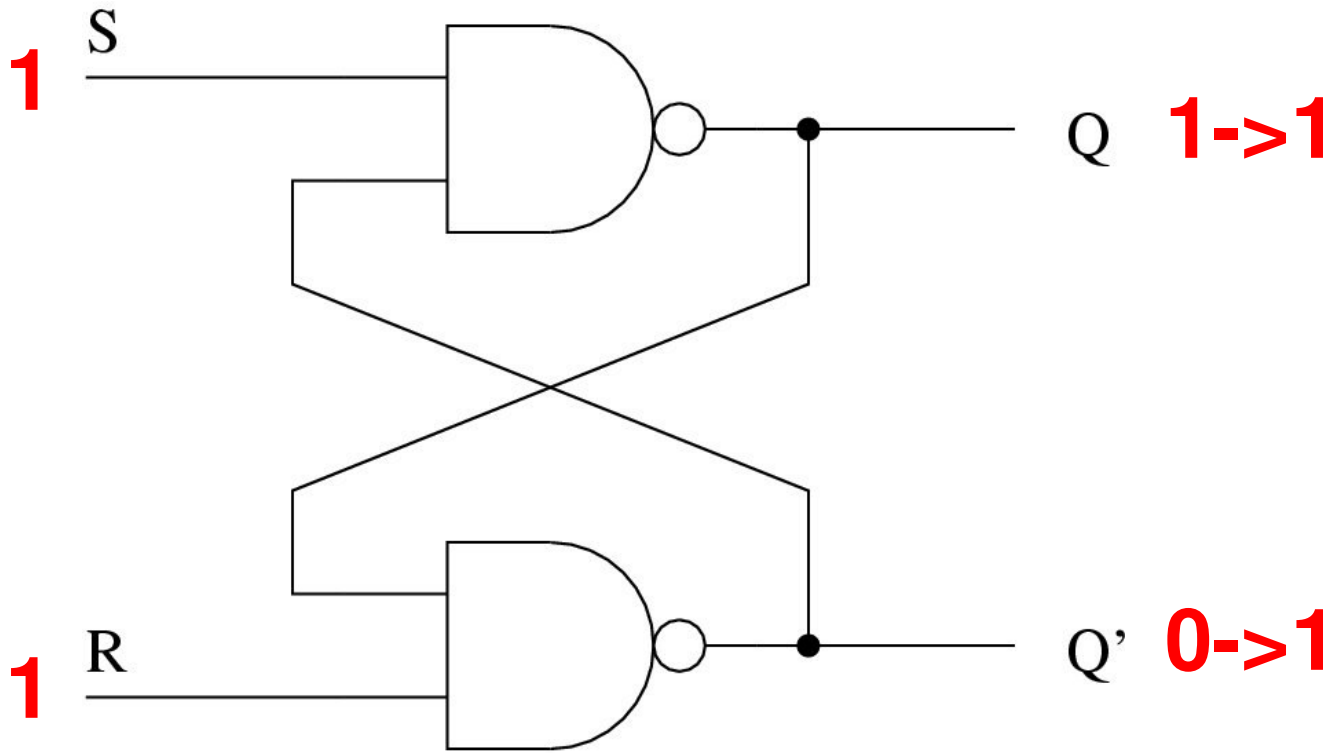
NAND Latch

Now S is set 1 – what happens?



NAND Latch

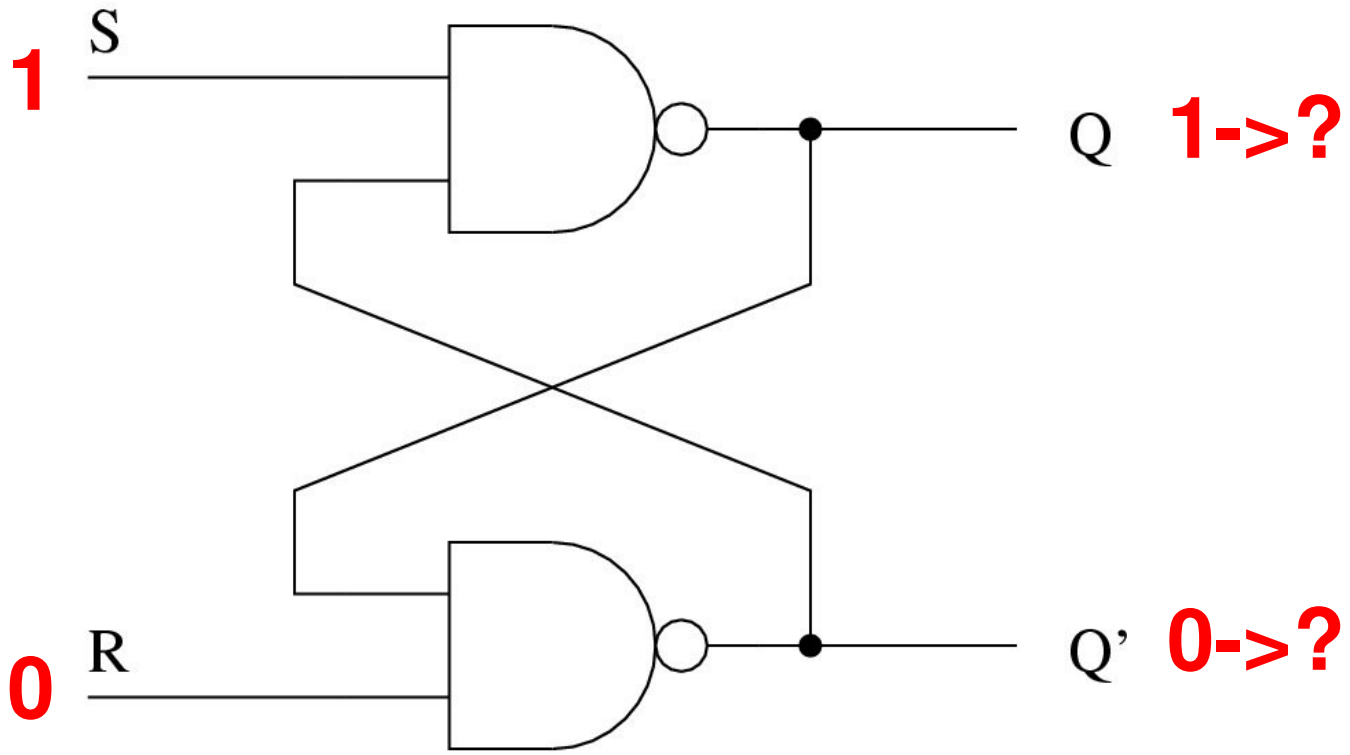
Q and Q' remain the same!



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

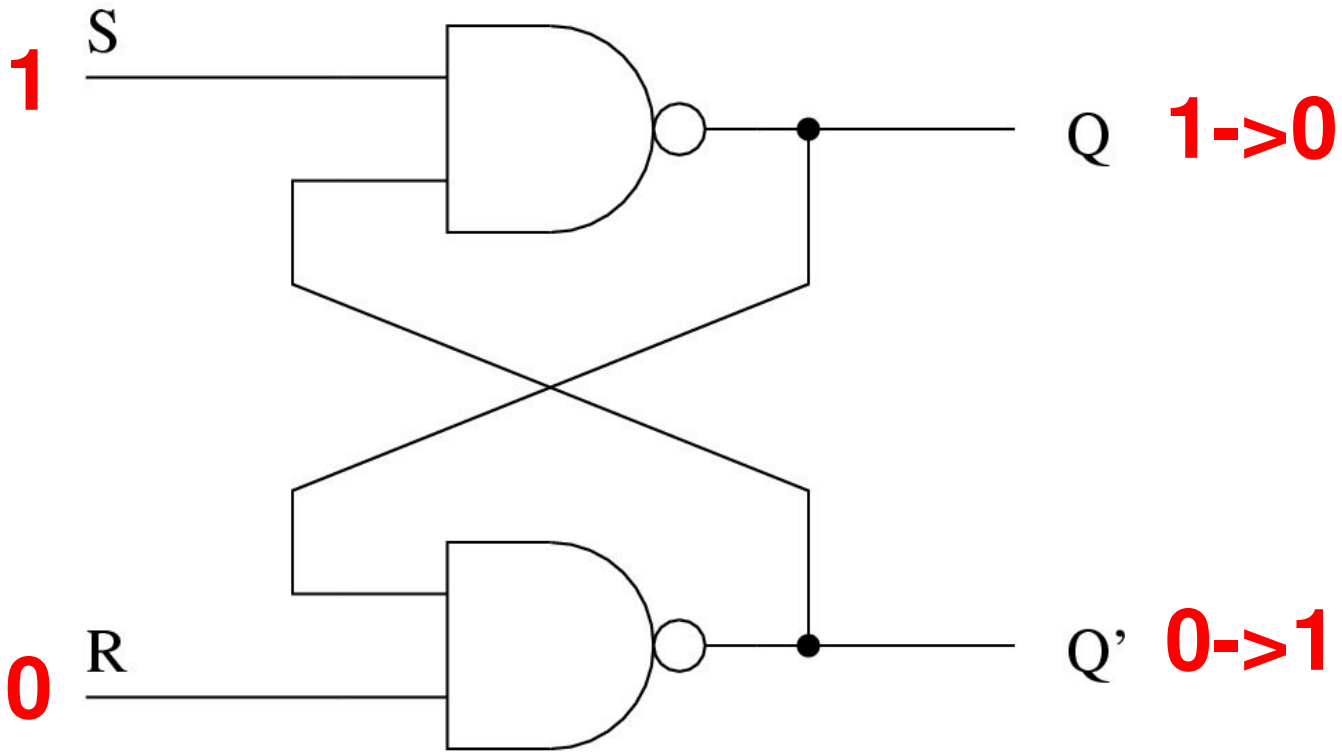
NAND Latch

Now set R to 0 – what happens?



NAND Latch

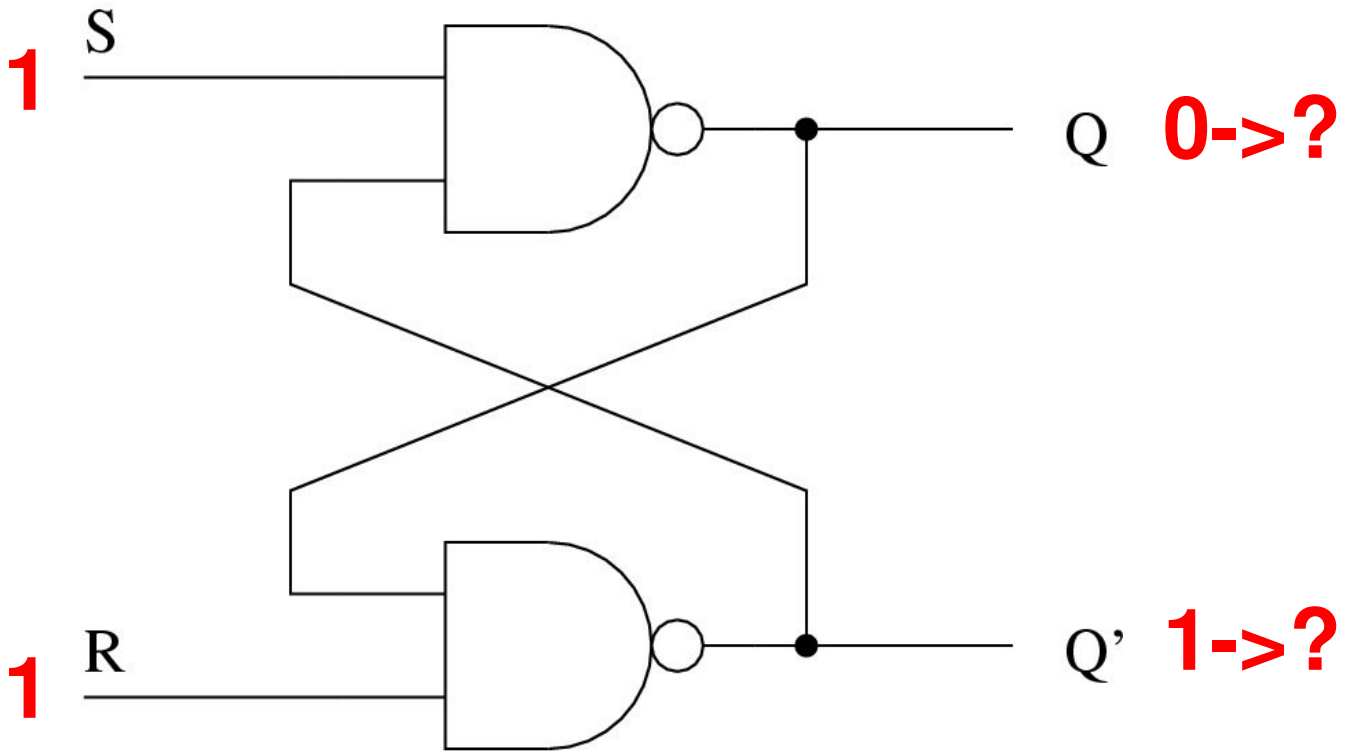
Now set R to 0 – what happens?



The state flips back (Q is ‘reset’)

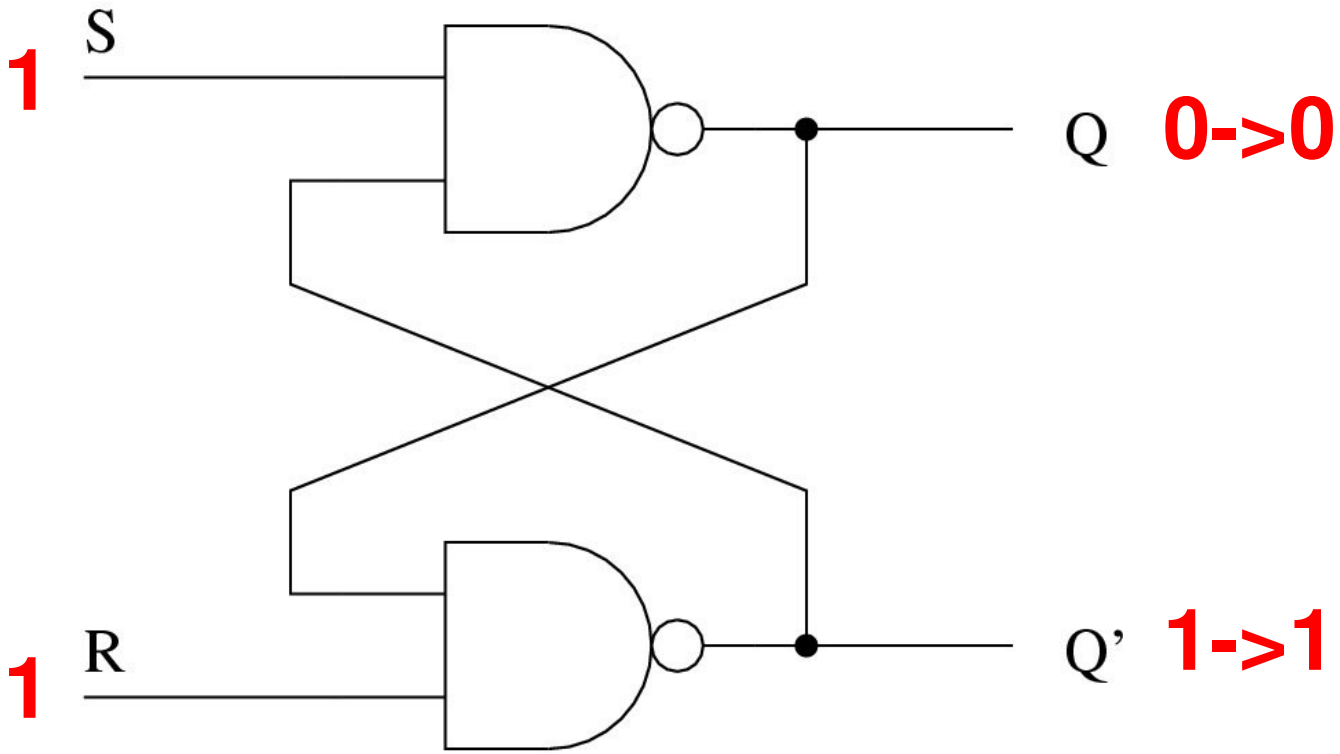
NAND Latch

Finally: set R to 1 – what happens?



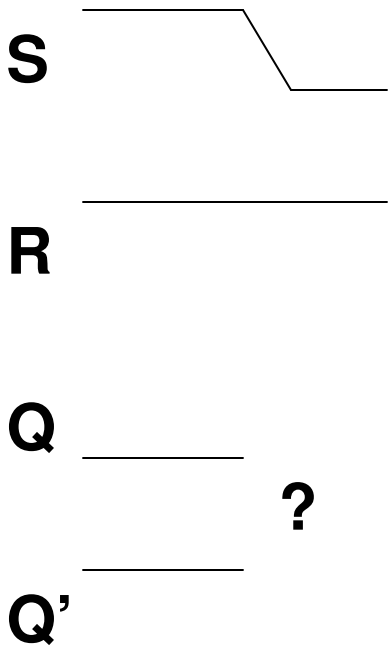
NAND Latch

Finally: set R to 1 – what happens?

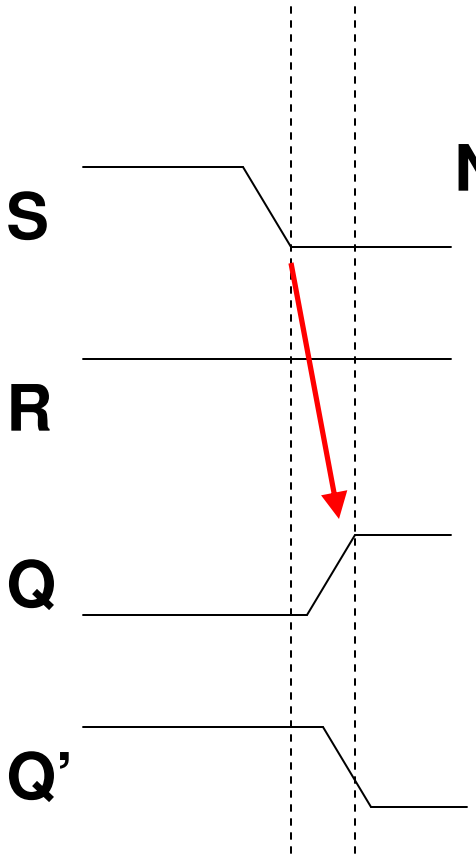


Q and Q' do not change state

Timing Diagram Representation

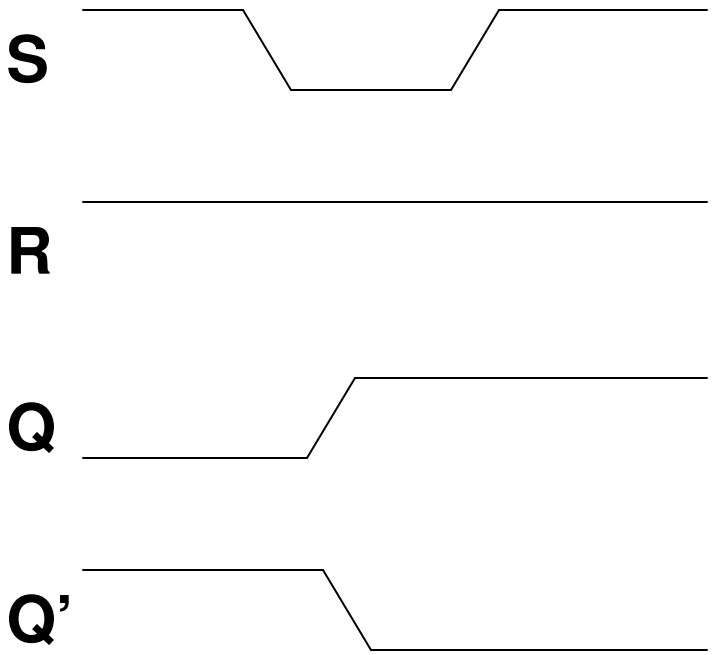


Timing Diagram Representation



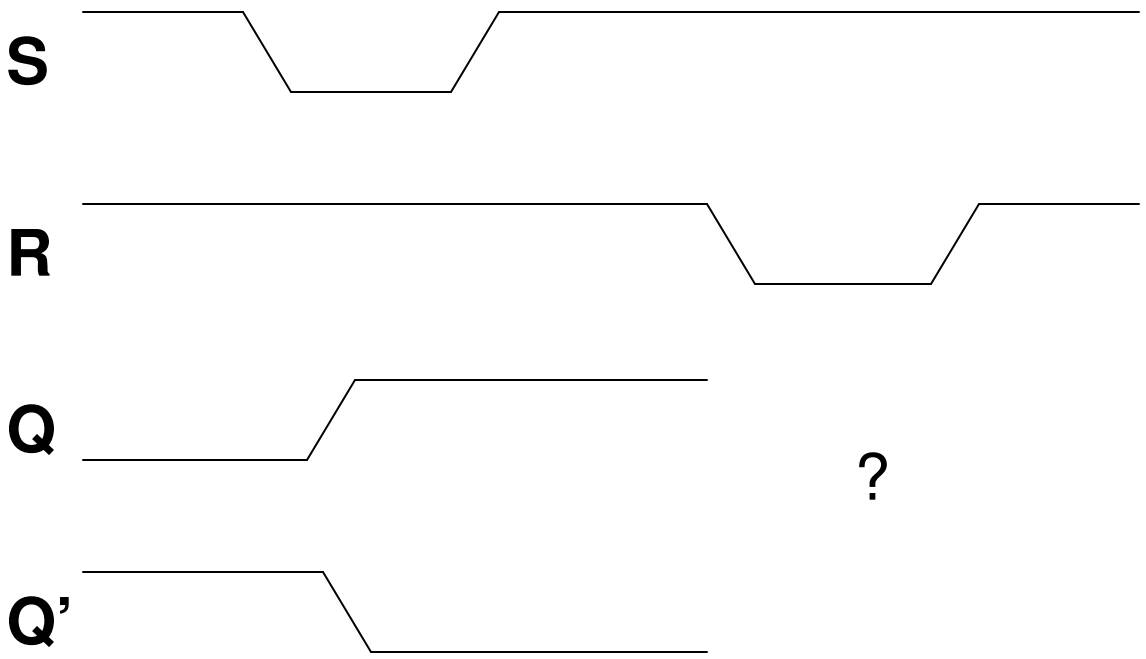
Note small delay in response in Q and Q'

Timing Diagram Representation

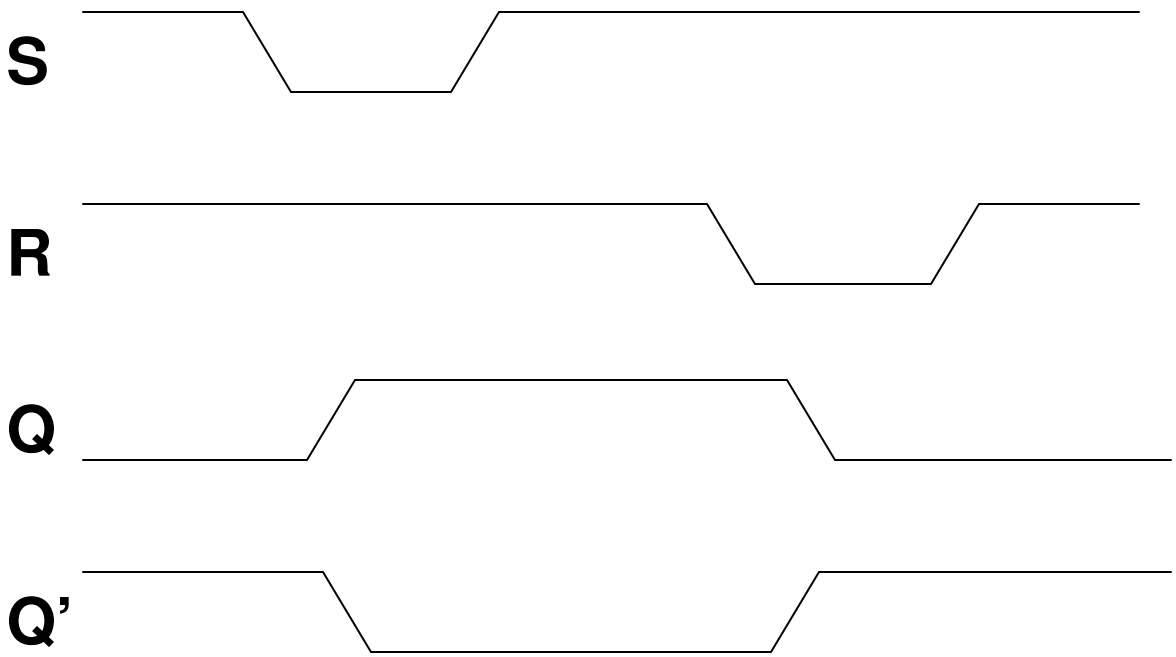


When S returns to high – both Q and Q' remain in the same state

Timing Diagram Representation

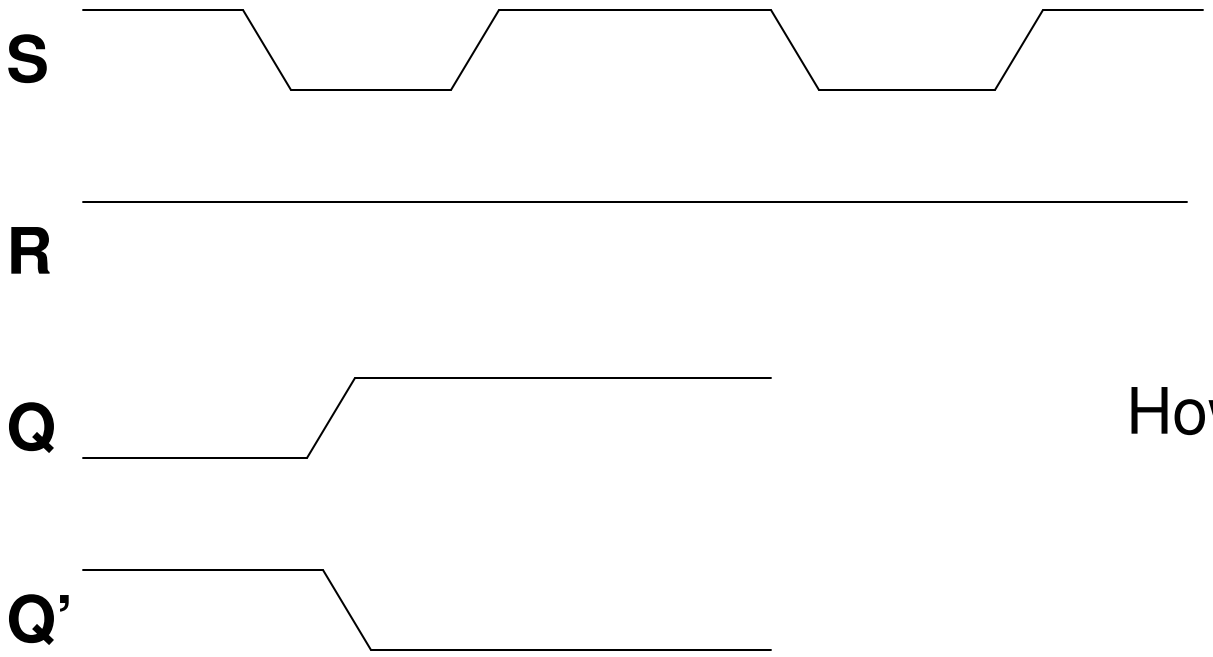


Timing Diagram Representation



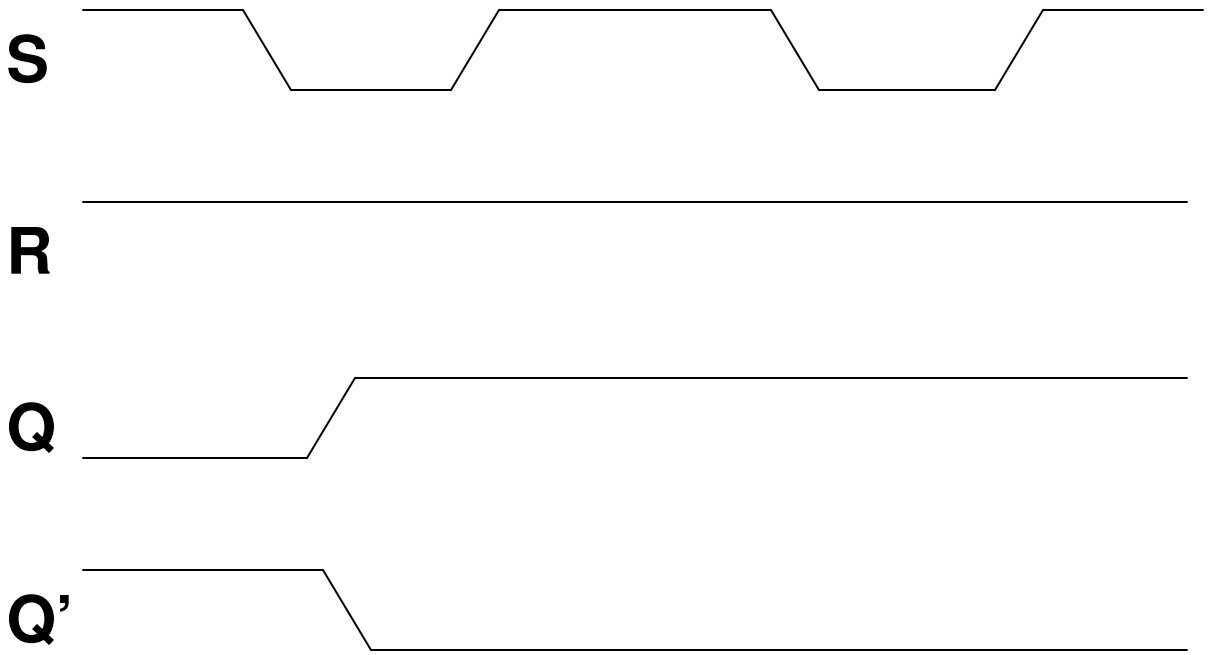
Q and Q' flip state

Timing Diagram Representation



How about this case?

Timing Diagram Representation



No change in Q and Q'

Next Time

- Microprocessors
  - Watch the schedule page for Thursday reading