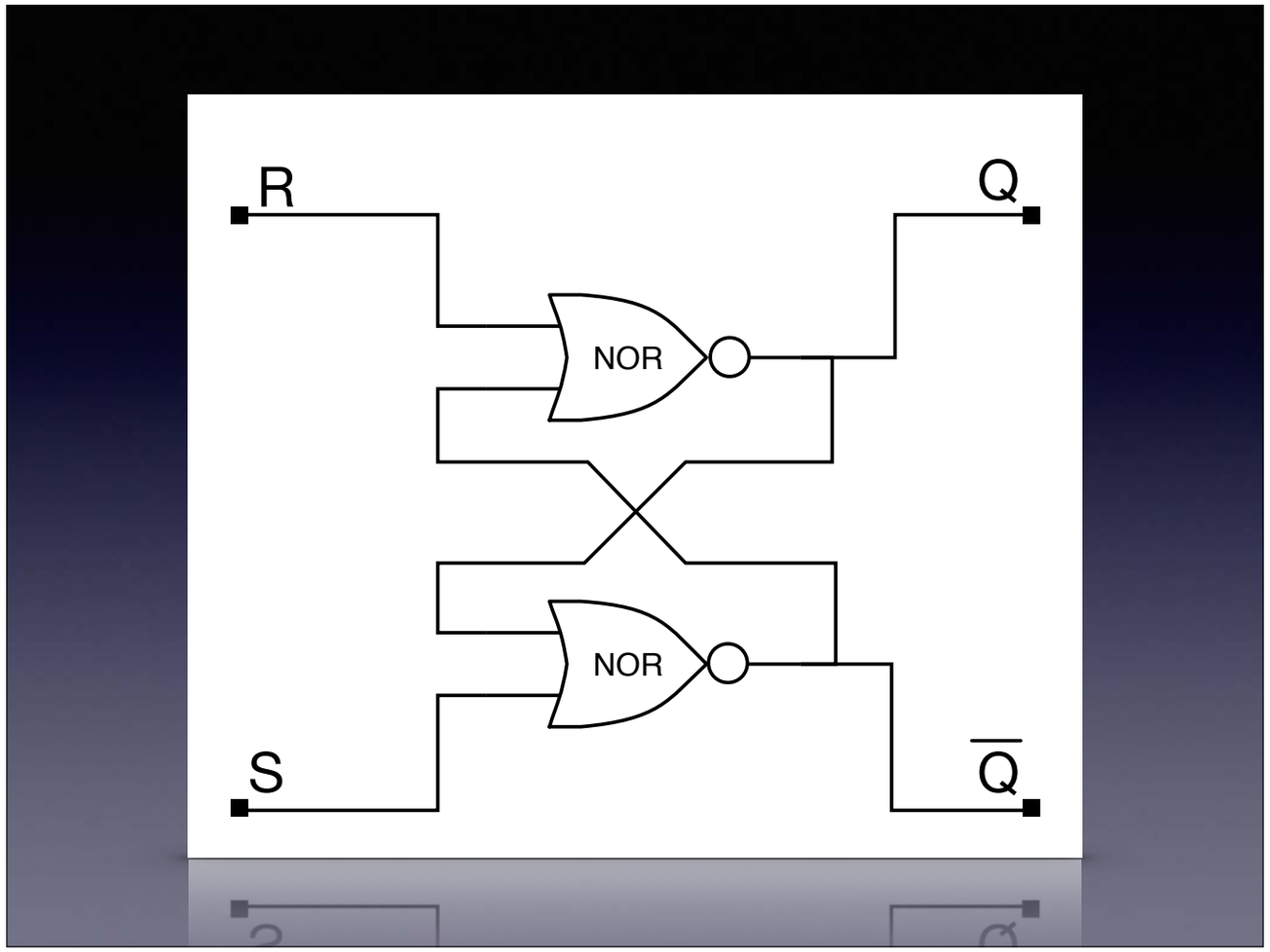
Sequential Logic

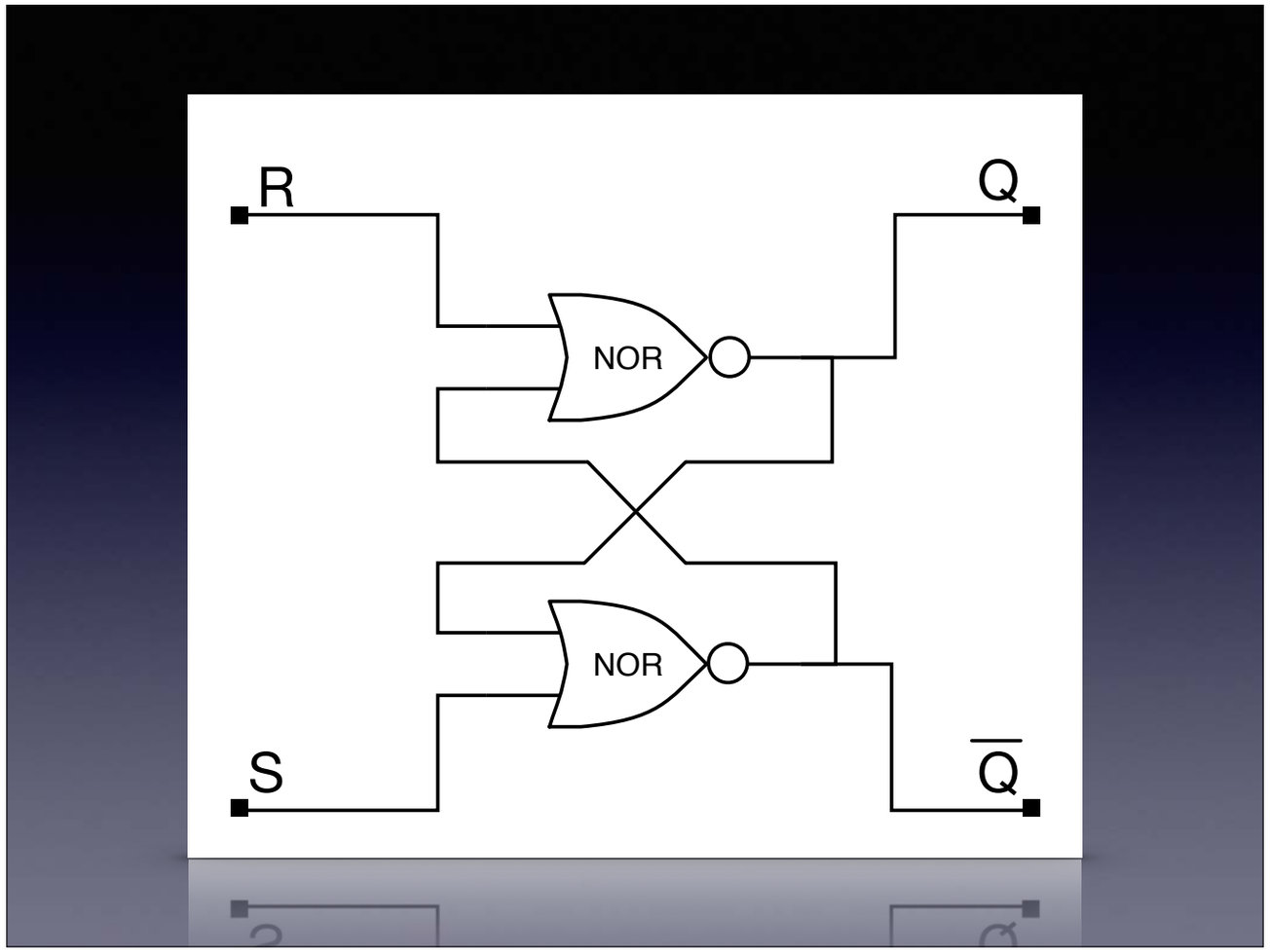
Steven R. Bagley

Introduction

- Seen some simple logic circuits
  - Adders
  - Multiplexers

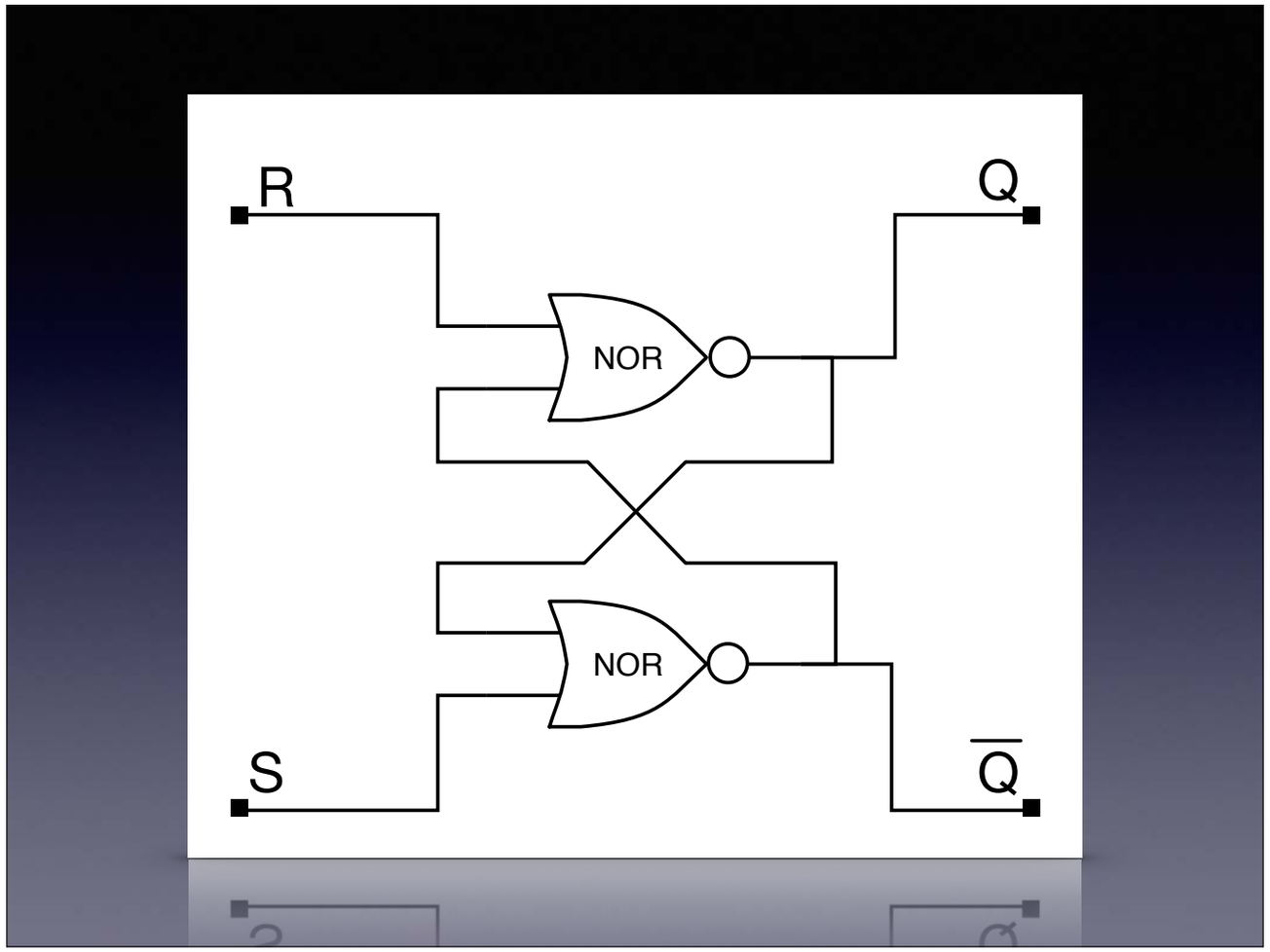


What does this circuit do?



So what about our mystery circuit?

Can apply the same thing — but our inputs are connected to our output

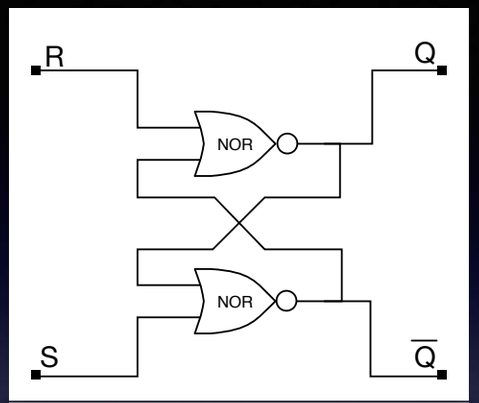


So what about our mystery circuit?

Can apply the same thing — but our inputs are connected to our output

Q often used to represent output

Qbar signal is always the opposite of Q



$$Q = \overline{R + \overline{Q}}$$

$$\overline{Q} = \overline{S + Q}$$

So what about our mystery circuit?

Can apply the same thing — but our inputs are connected to our output

Q often used to represent output

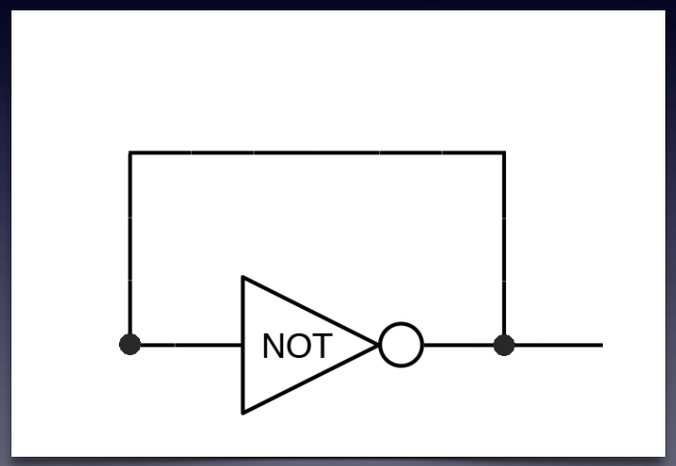
Qbar signal is always the opposite of Q

Propagation

- Our circuit feeds back on itself
- But remember it takes some time for a change in input to reach the output
- So we should really think of the *next* Q (or Q_{NEXT}) rather in those equations
- Can see this if we look at a simpler example…

And build up a series of truth tables.

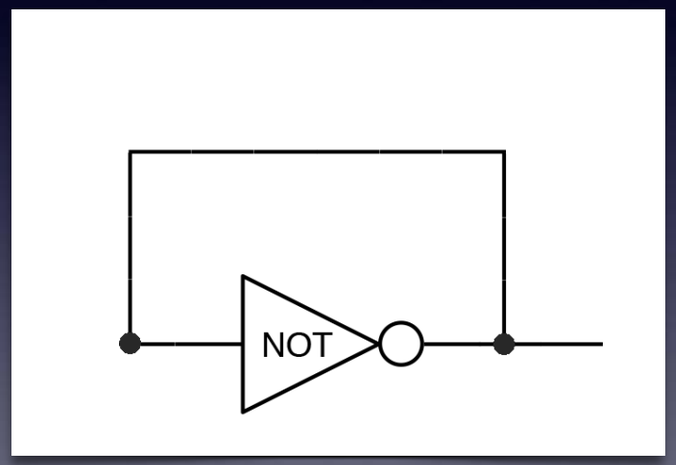
NOT Oscillator



Input	Output

Connecting a NOT gate’s output to its input will create an output that oscillates between 0 and 1 (rate determined by the propagation delay)

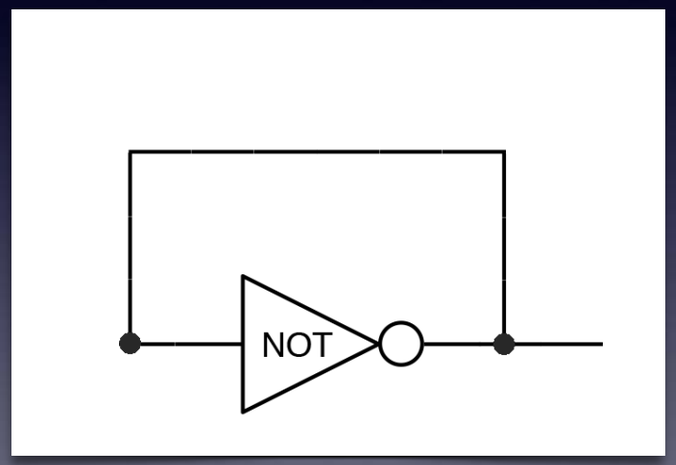
NOT Oscillator



Input	Output
0	1

Connecting a NOT gate’s output to its input will create an output that oscillates between 0 and 1 (rate determined by the propagation delay)

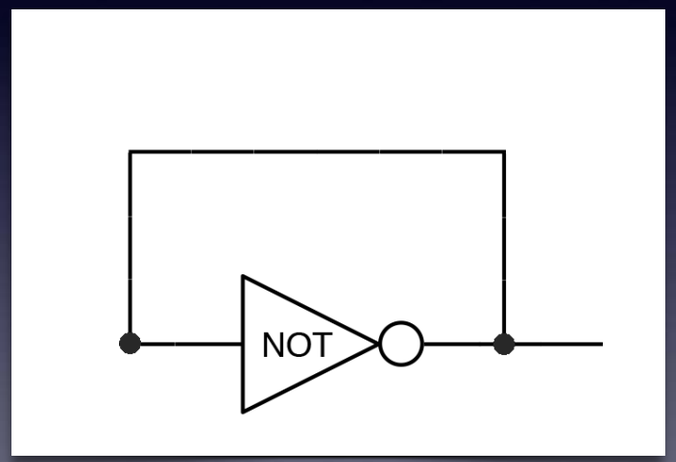
NOT Oscillator



Input	Output
0	1
1	0

Connecting a NOT gate’s output to its input will create an output that oscillates between 0 and 1 (rate determined by the propagation delay)

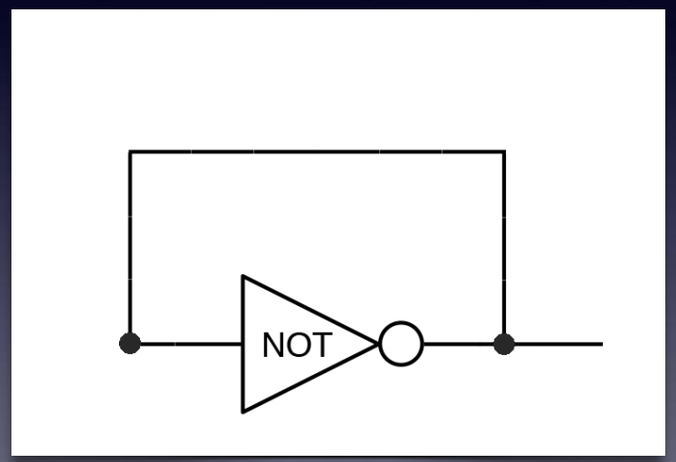
NOT Oscillator



Input	Output
0	1
1	0
0	1

Connecting a NOT gate’s output to its input will create an output that oscillates between 0 and 1 (rate determined by the propagation delay)

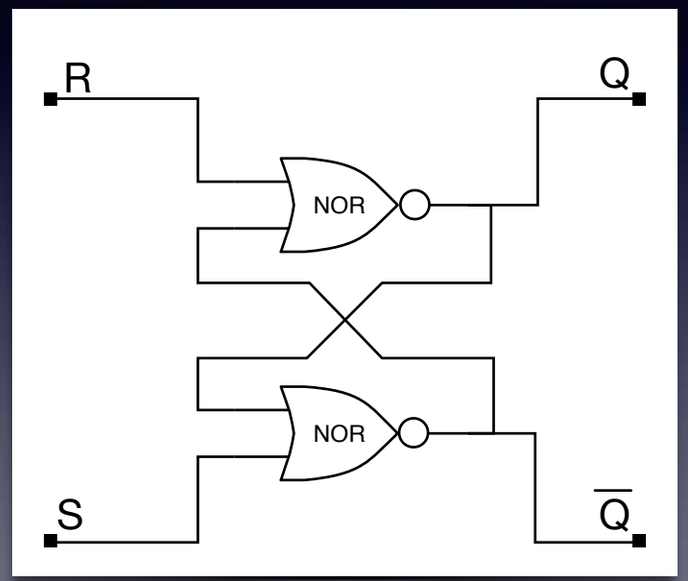
NOT Oscillator



Input	Output
0	1
1	0
0	1
1	0

Connecting a NOT gate’s output to its input will create an output that oscillates between 0 and 1 (rate determined by the propagation delay)

SR Latch



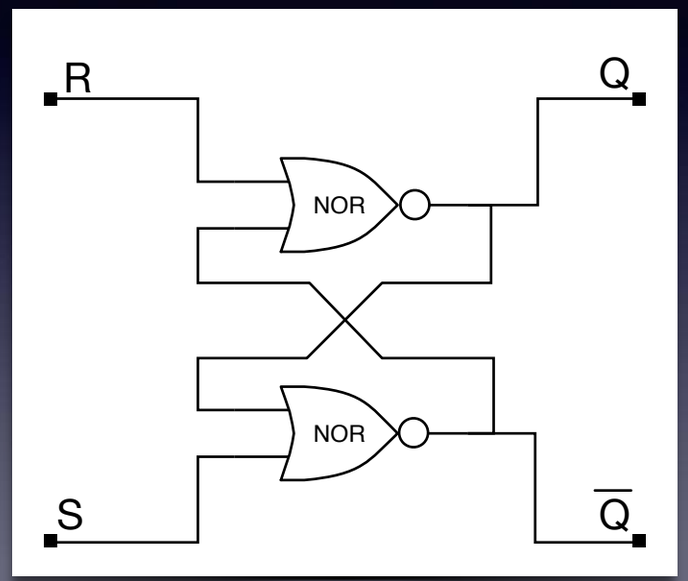
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$

When S goes high, the outputs wobble and then Q also goes high

When S then goes low, Q still stays high

Causes Q to remember it is set…

SR Latch



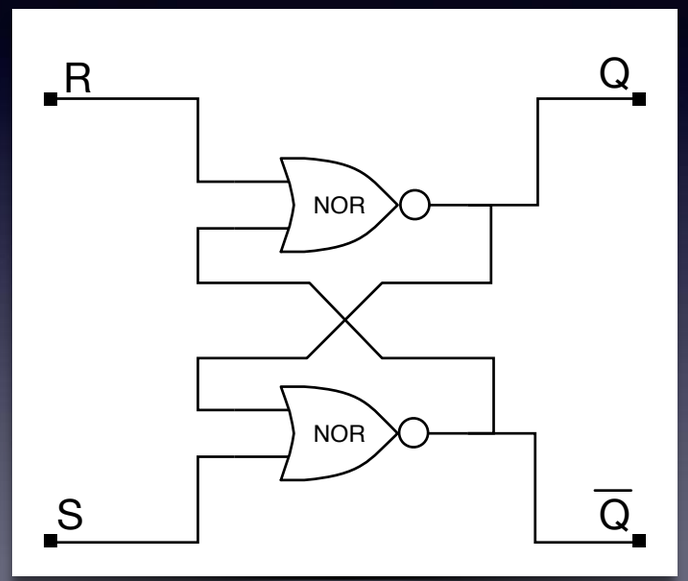
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{Q}$	$\overline{Q}$

When S goes high, the outputs wobble and then Q also goes high

When S then goes low, Q still stays high

Causes Q to remember it is set…

SR Latch



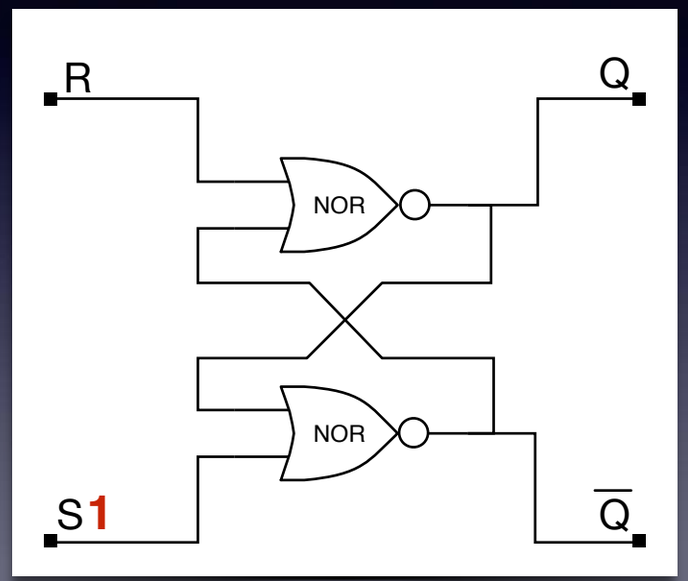
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0

When S goes high, the outputs wobble and then Q also goes high

When S then goes low, Q still stays high

Causes Q to remember it is set…

SR Latch



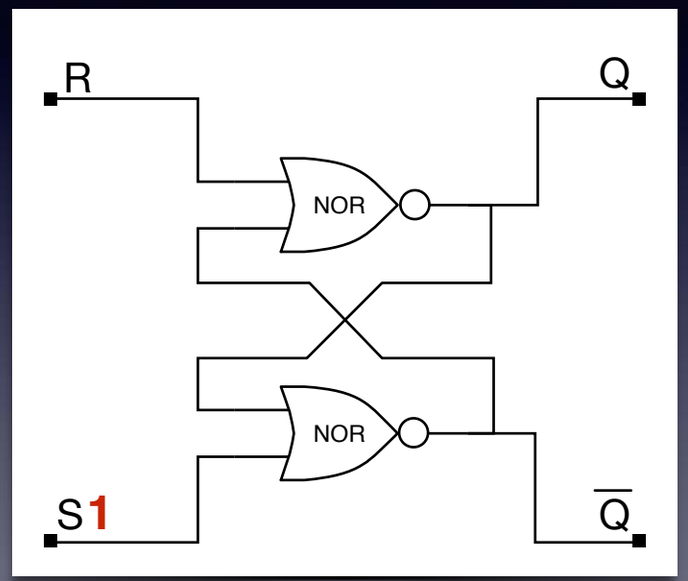
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0

When S goes high, the outputs wobble and then Q also goes high

When S then goes low, Q still stays high

Causes Q to remember it is set…

SR Latch



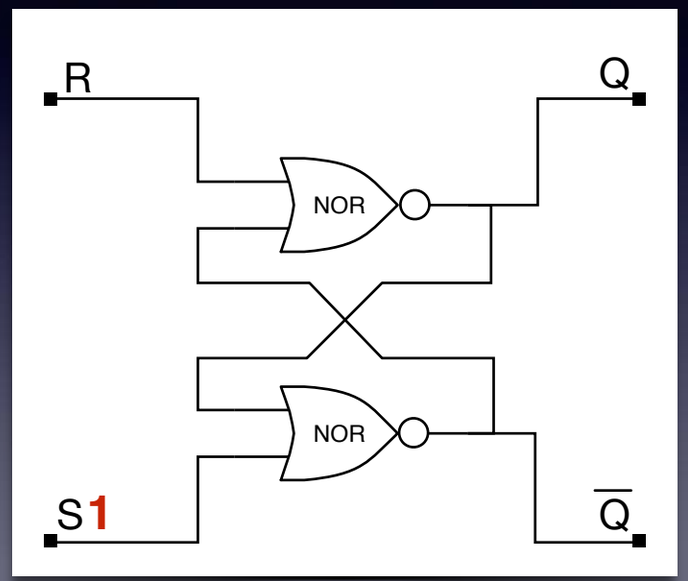
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0
0	1	Q	0	1	0

When S goes high, the outputs wobble and then Q also goes high

When S then goes low, Q still stays high

Causes Q to remember it is set…

SR Latch



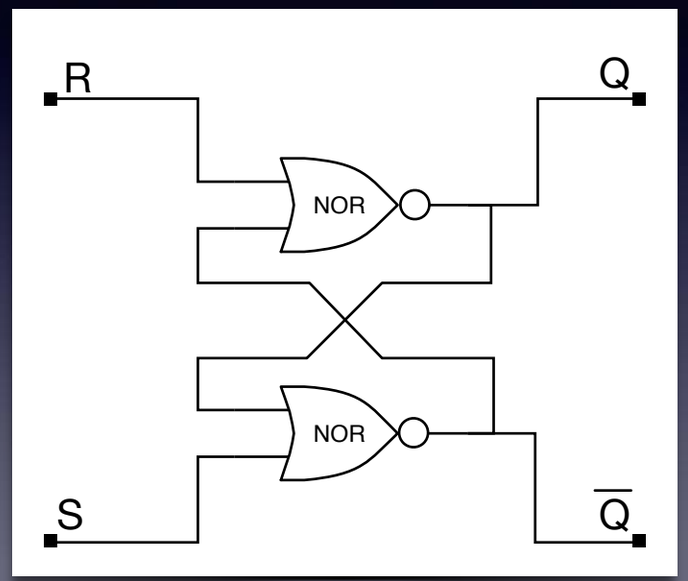
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0
0	1	Q	0	1	0
0	1	1	0	1	0

When S goes high, the outputs wobble and then Q also goes high

When S then goes low, Q still stays high

Causes Q to remember it is set…

SR Latch



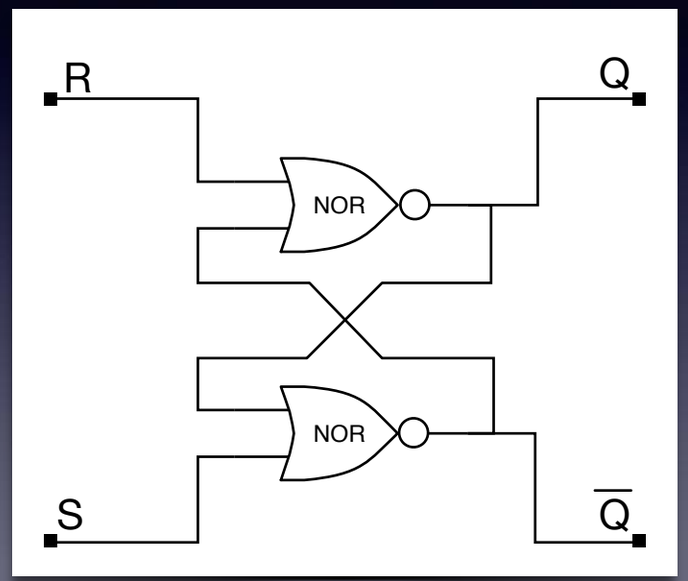
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0
0	1	Q	0	1	0
0	1	1	0	1	0

When S goes high, the outputs wobble and then Q also goes high

When S then goes low, Q still stays high

Causes Q to remember it is set…

SR Latch



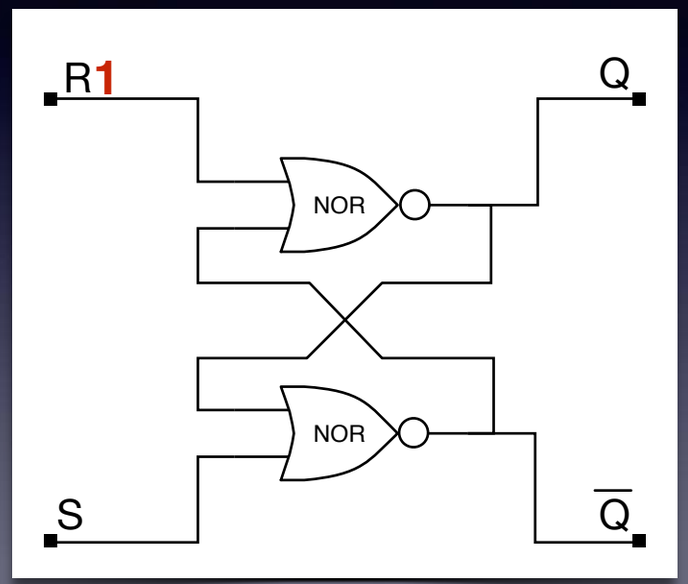
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0
0	1	Q	0	1	0
0	1	1	0	1	0
0	0	1	0	1	0

When S goes high, the outputs wobble and then Q also goes high

When S then goes low, Q still stays high

Causes Q to remember it is set…

SR Latch



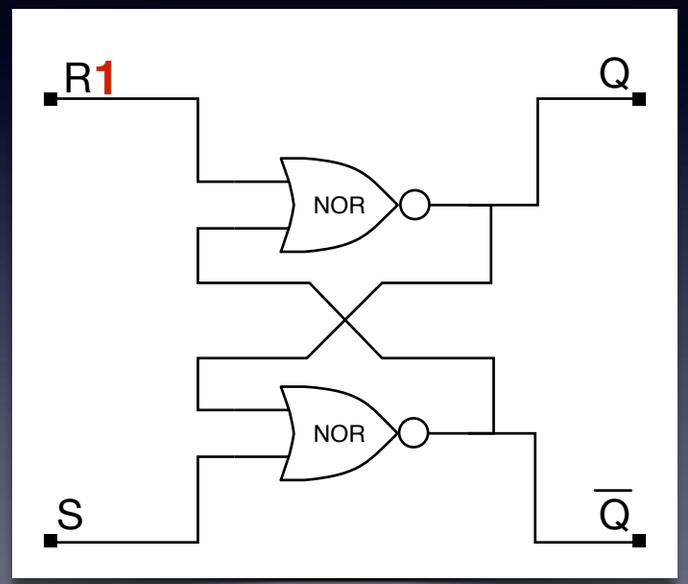
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$

When R goes high, the outputs wobble and then Q lies

When R then goes low, Q still stays low

Causes Q to reset to 0

SR Latch



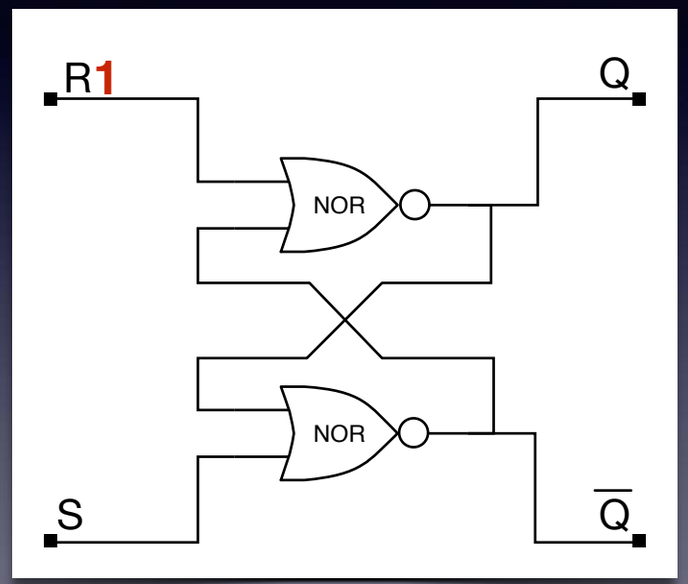
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0

When R goes high, the outputs wobble and then Q lies

When R then goes low, Q still stays low

Causes Q to reset to 0

SR Latch



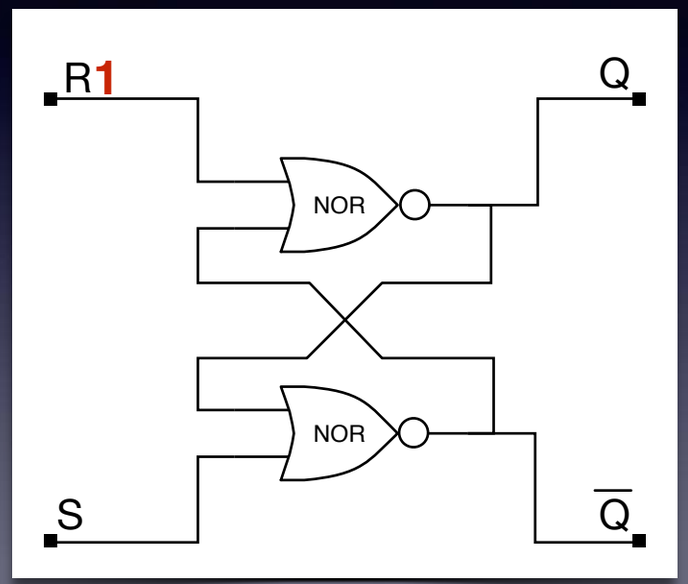
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0
1	0	1	0	0	0

When R goes high, the outputs wobble and then Q lies

When R then goes low, Q still stays low

Causes Q to reset to 0

SR Latch



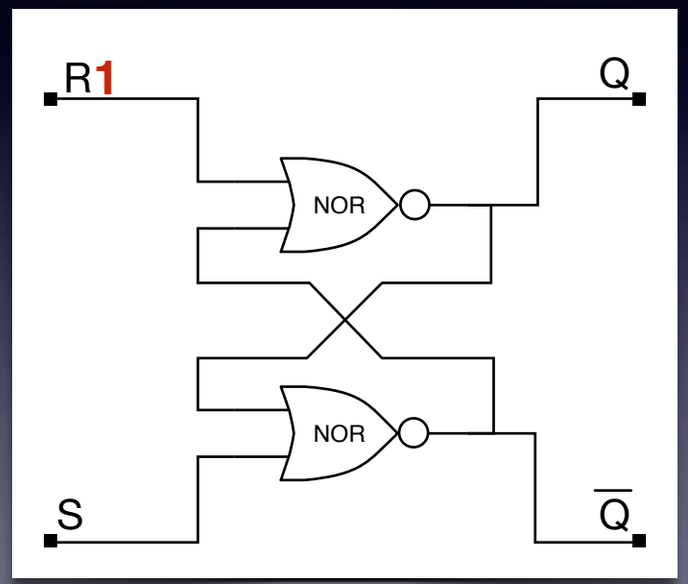
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0
1	0	1	0	0	0
1	0	0	0	0	1

When R goes high, the outputs wobble and then Q lies

When R then goes low, Q still stays low

Causes Q to reset to 0

SR Latch



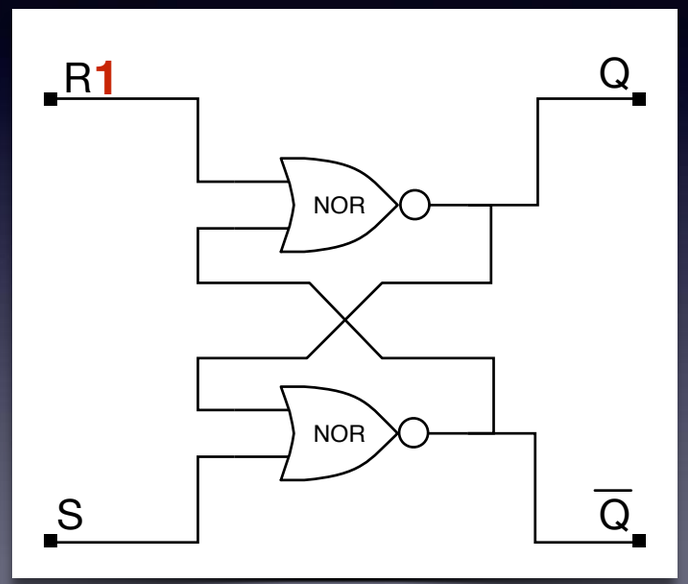
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0
1	0	1	0	0	0
1	0	0	0	0	1
1	0	0	1	0	1

When R goes high, the outputs wobble and then Q lies

When R then goes low, Q still stays low

Causes Q to reset to 0

SR Latch



R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0
1	0	1	0	0	0
1	0	0	0	0	1
1	0	0	1	0	1
0	0	0	1	0	1

When R goes high, the outputs wobble and then Q lies

When R then goes low, Q still stays low

Causes Q to reset to 0

Storing State

- This circuit remembers things!
- It remembers if its been set, or reset
- Called an SR NOR latch
- But things go wrong if both **`R`** and **`S`** set…
- Often have additional circuitry to avoid this…

Work through the table to see what happens if R and S both 1, usually delivers

D Latch

- The D latch is a variation on this that is slightly more useful
- Has a data input (D) and a clock input
- When CLK is high, D goes through to the output
- When CLK is low, output latches last output
- Easy to see how to build this from an SR latch

Show how

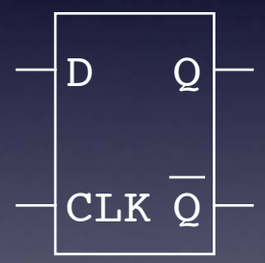
R = ~D and E

S = D and E

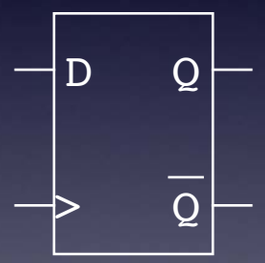
Flip-flops

- Latches pass the input to output whenever the CLK input is high
- Flip-flop is a similar circuit but stores the input on the CLK's transition from low to high
- Also sometimes have a clear input to reset the latch/flip-flop

D-latch D-flip-flop symbol



D latch



D flip-flop

OFten drawn in circuits like this

Sequential Logic

- Can combine flip-flops with standard combination logic to produce circuits
- These circuits are capable of remembering things
- Also forms the basis of the computer's memory

Draw some examples on paper and talk through them.