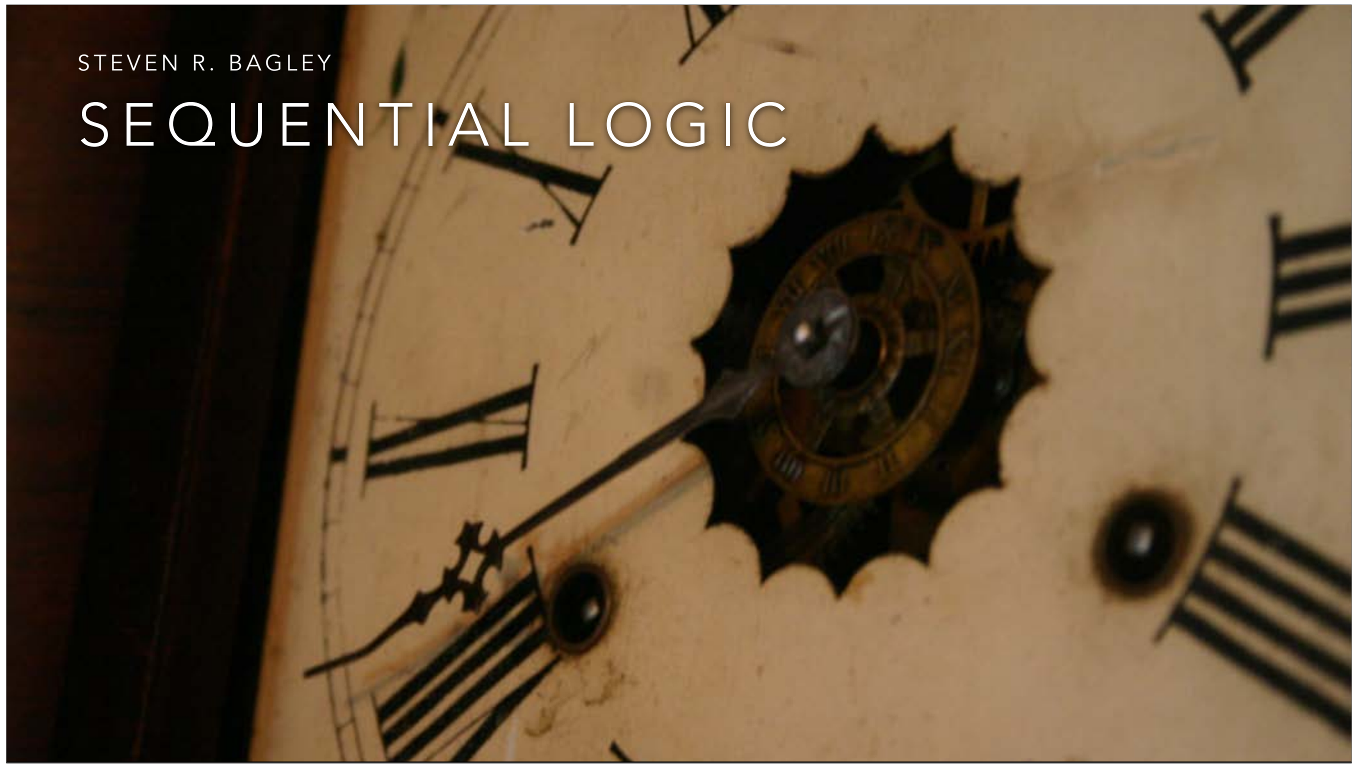


STEVEN R. BAGLEY

SEQUENTIAL LOGIC

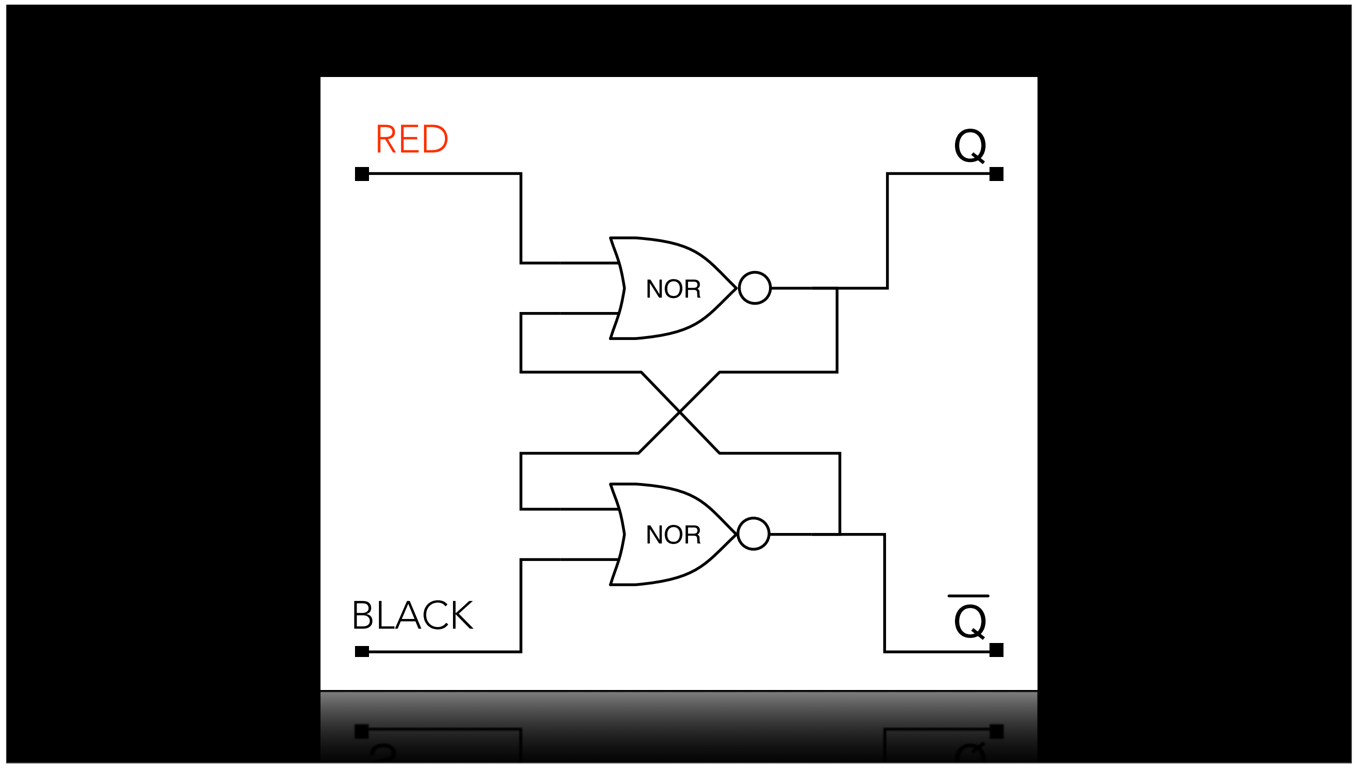
INTRODUCTION

- Looked at how we can arrange logic gates to form various circuits
- These circuits process the input signals to produce new output signals
- *Combinatorial Logic*
- Today, start to look at *Sequential Logic*

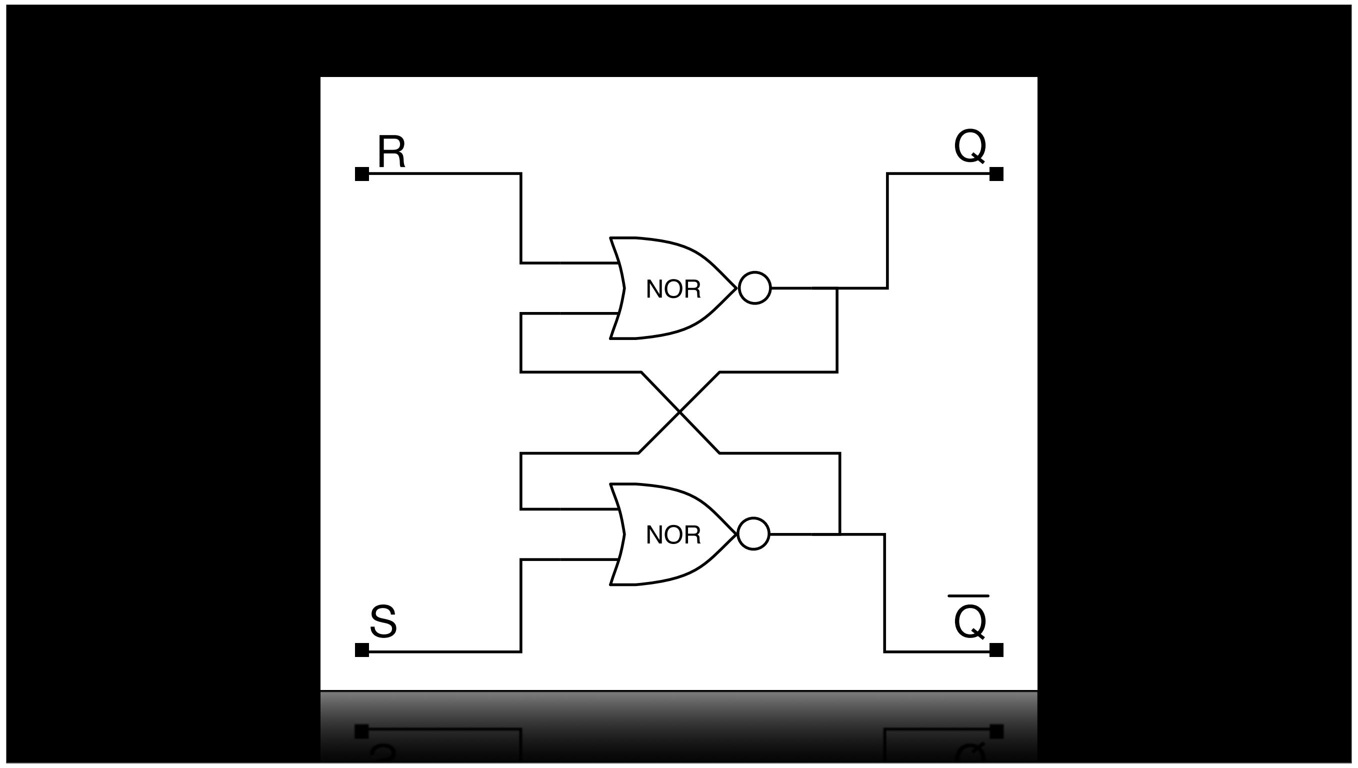
MYSTERY...

MYSTERY…

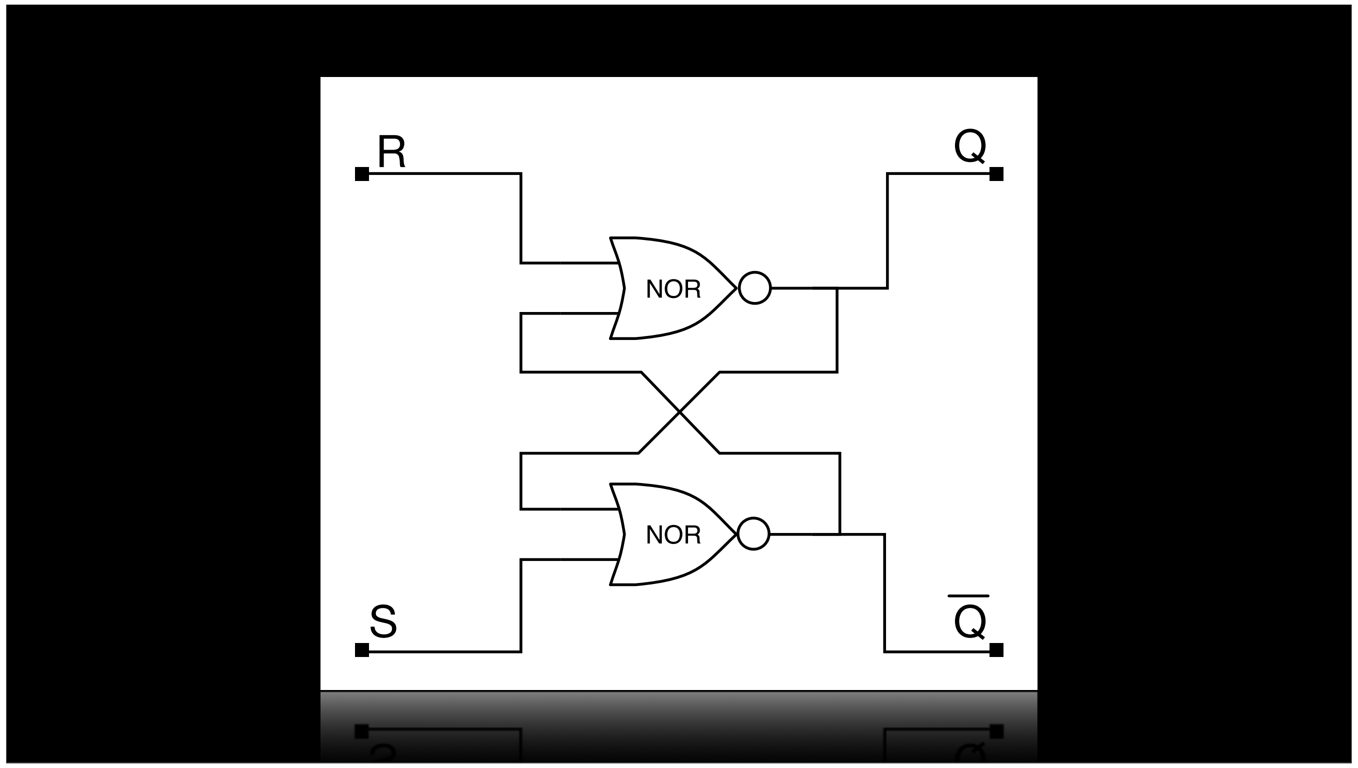
A	B	RESULT
0	0	
0	1	
1	0	
1	1	



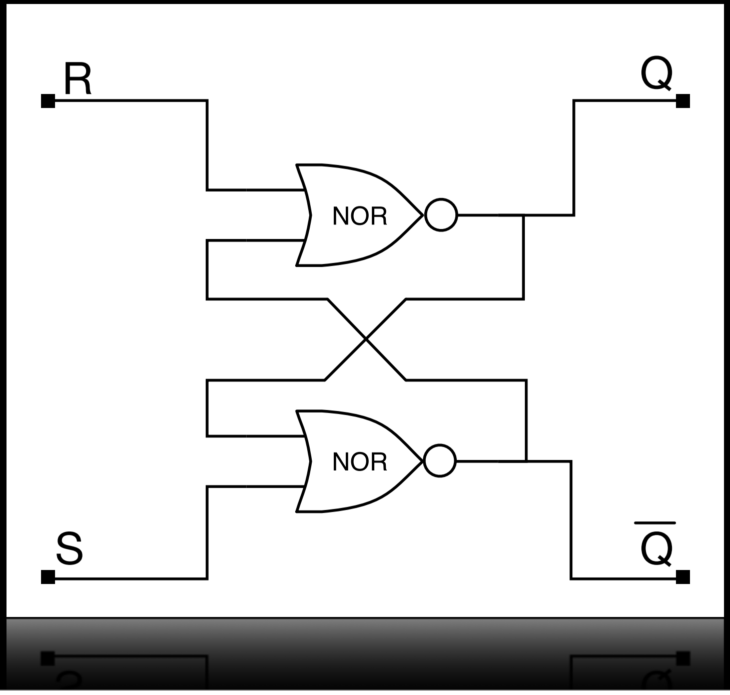
Could start to try and work out the logic equations to this…
Show Computerphile extract (How computer memory works)

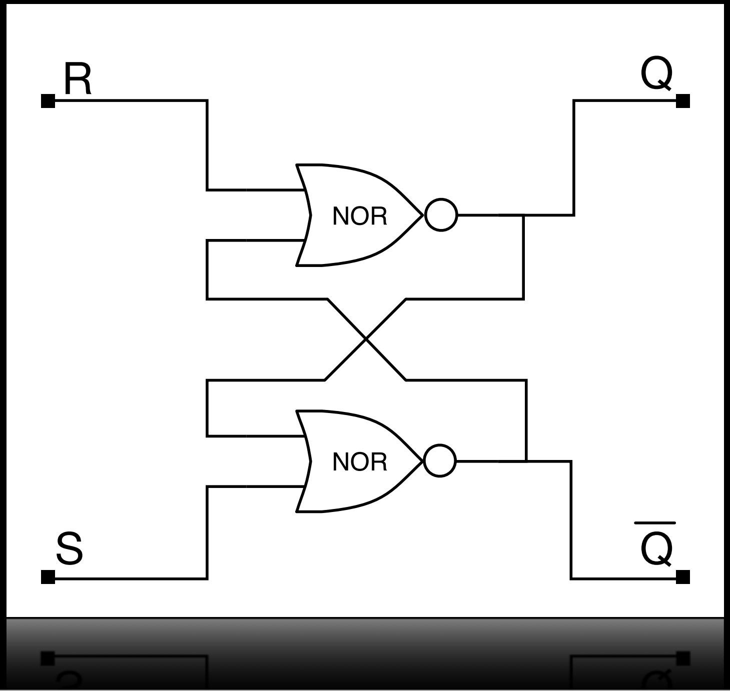


Could start to try and work out the logic equations to this…
Show Computerphile extract (How computer memory works)



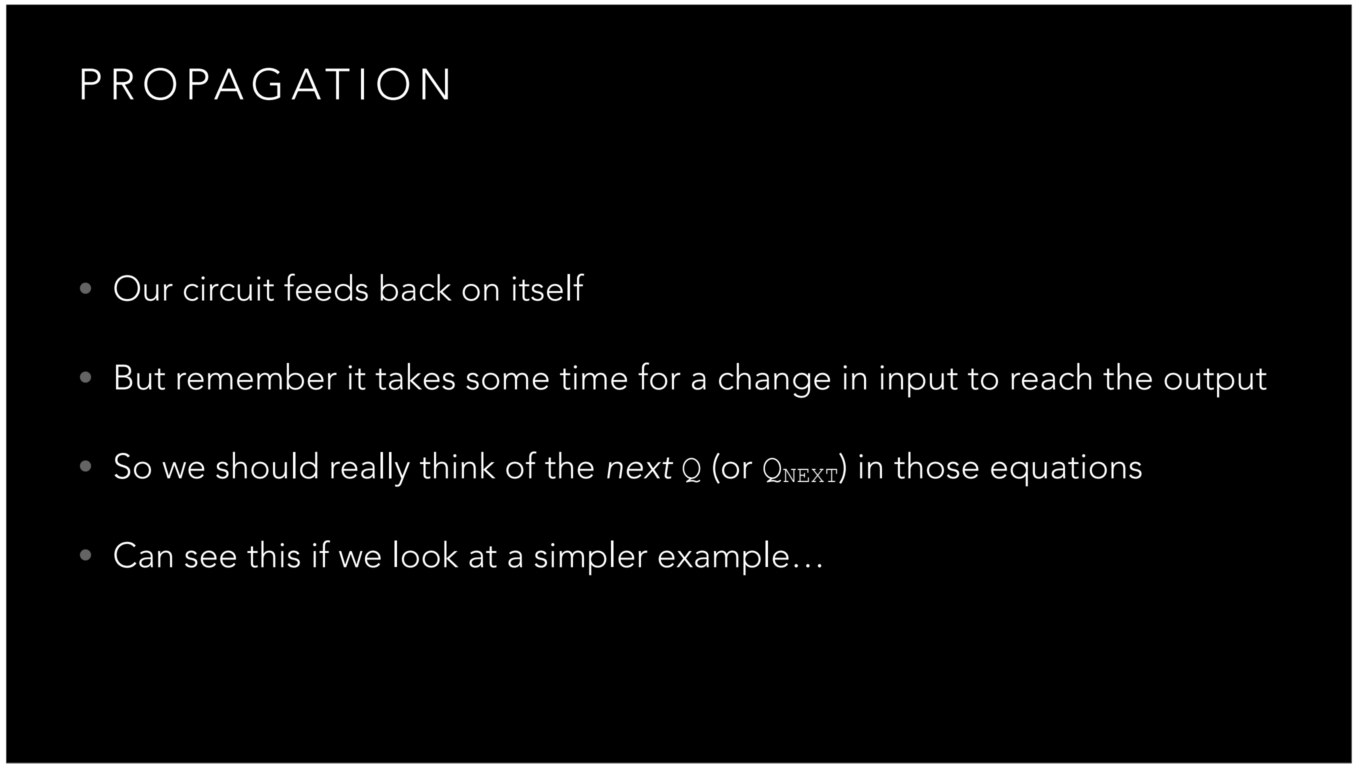
So what about our mystery circuit?
Can apply the same thing — but our inputs are connected to our output
Q often used to represent output
Qbar signal is always the opposite of Q


R
Q
NOR
NOR
S
$\overline{Q}$



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

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



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}) in those equations
- Can see this if we look at a simpler example…

Propogation delay of 74hct02 is ~7ns
And build up a series of truth tables.

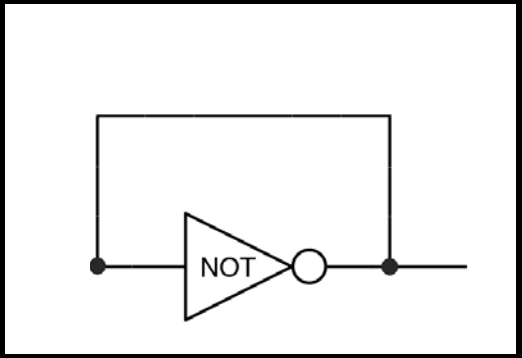
PROPAGATION

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 $\mathtt{Q}_{\mathtt{NEXT}}$) in those equations
- Can see this if we look at a simpler example…

NOT OSCILLATOR

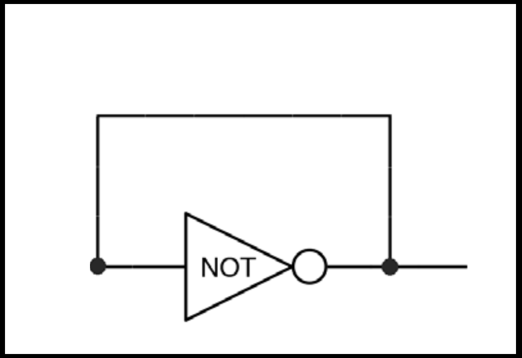
INPUT OUTPUT



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)

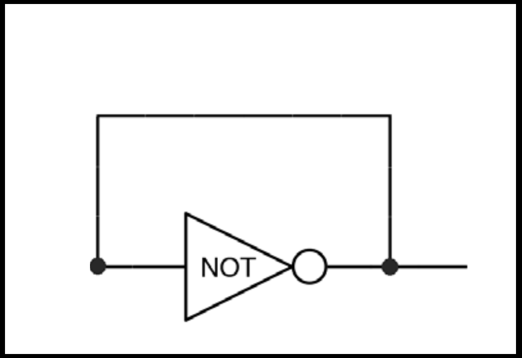
NOT OSCILLATOR

INPUT	OUTPUT
0	1



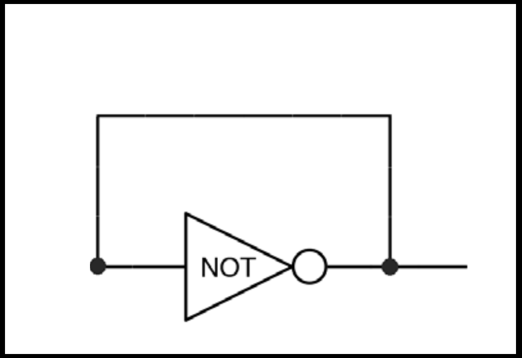
NOT OSCILLATOR

INPUT	OUTPUT
0	1
1	0



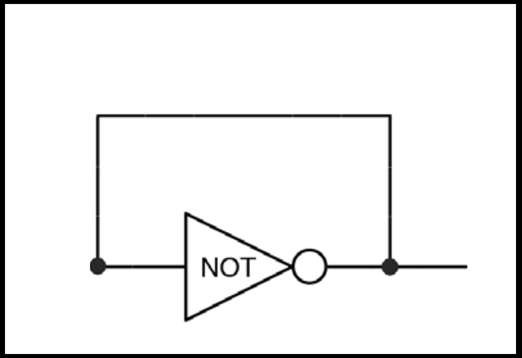
NOT OSCILLATOR

INPUT	OUTPUT
0	1
1	0
0	1

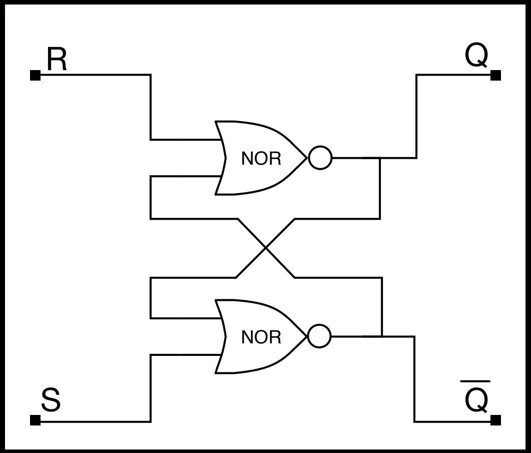


NOT OSCILLATOR

INPUT	OUTPUT
0	1
1	0
0	1
1	0



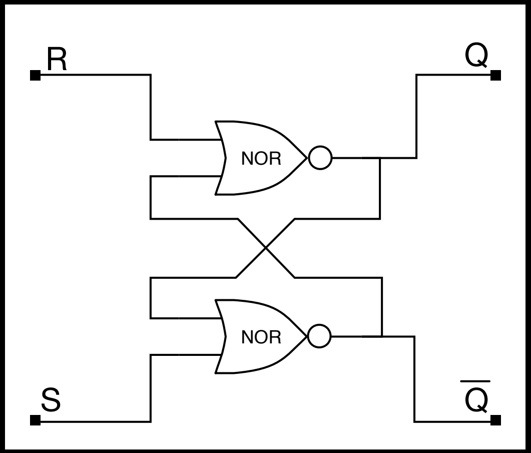
SR LATCH



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

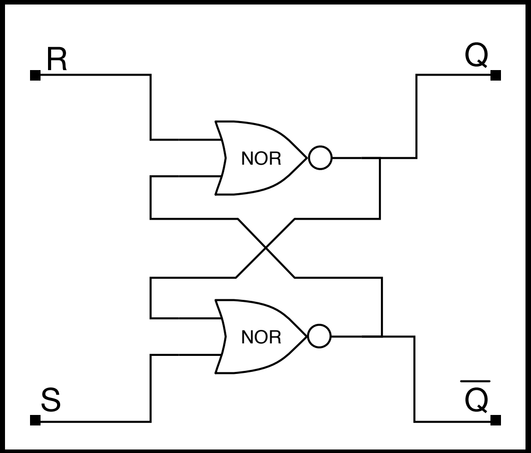
When S goes high, the outputs wobble and then Q also goes high
When S then goes low, Q still stays high
Causes Q to remember it is set…

SR LATCH



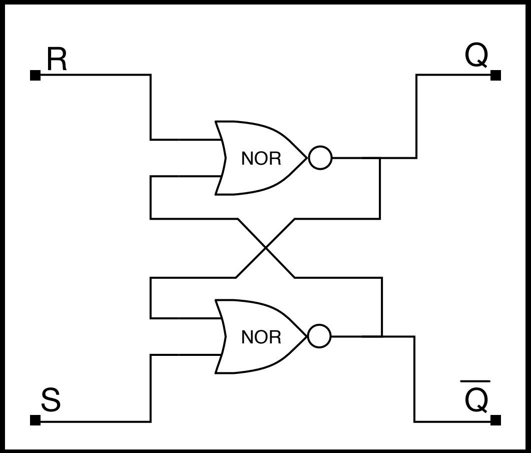
R	S	Q	$\bar{Q}$	Q_{next}	$\bar{Q}_{next}$
0	0	—	—	$\bar{\bar{Q}}$	$\bar{Q}$

SR LATCH



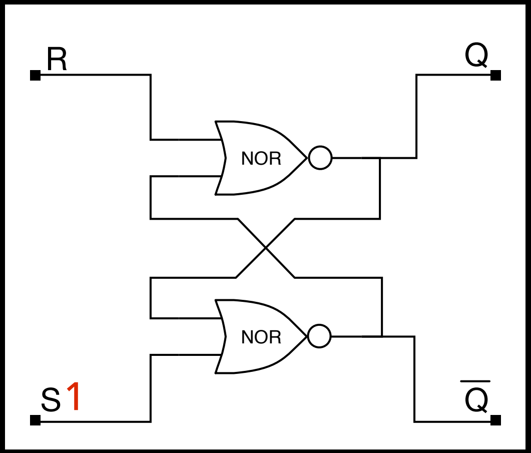
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	Q	$\overline{Q}$

SR LATCH



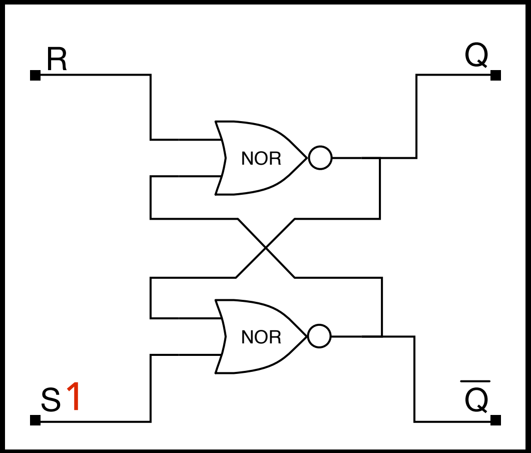
R	S	Q	$\bar{Q}$	Q_{next}	$\bar{Q}_{next}$
0	0	—	—	$\bar{\bar{Q}}$	$\bar{Q}$

SR LATCH



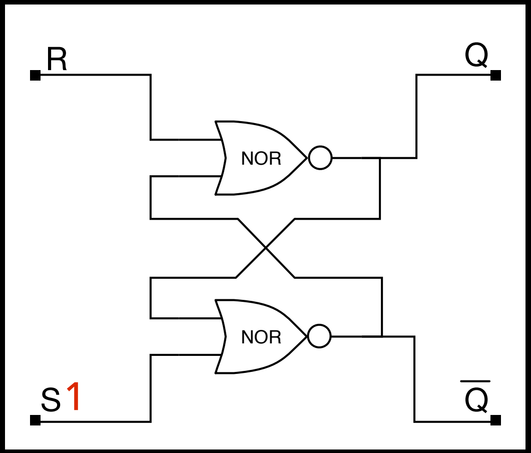
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$

SR LATCH



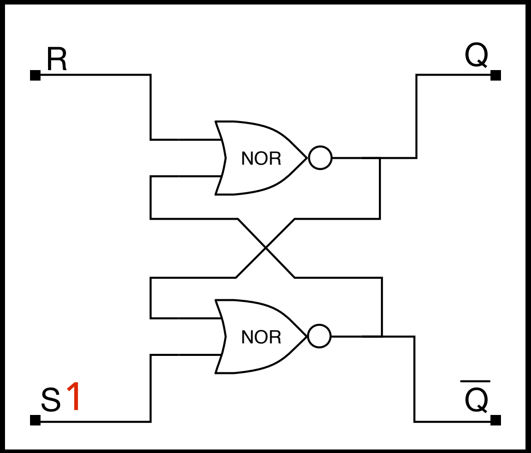
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0

SR LATCH



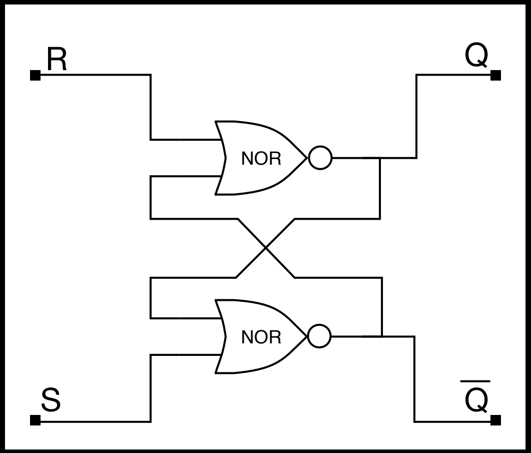
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0
0	1	Q	0	1	0

SR LATCH



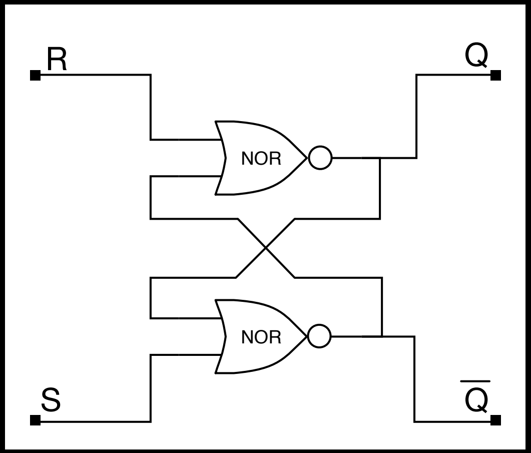
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0
0	1	Q	0	1	0
0	1	1	0	1	0

SR LATCH



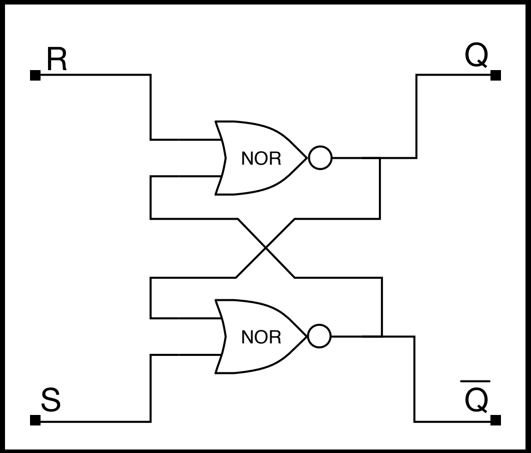
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	—	—	$\overline{\overline{Q}}$	$\overline{Q}$
0	1	Q	$\overline{Q}$	$\overline{\overline{Q}}$	0
0	1	Q	0	1	0
0	1	1	0	1	0

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

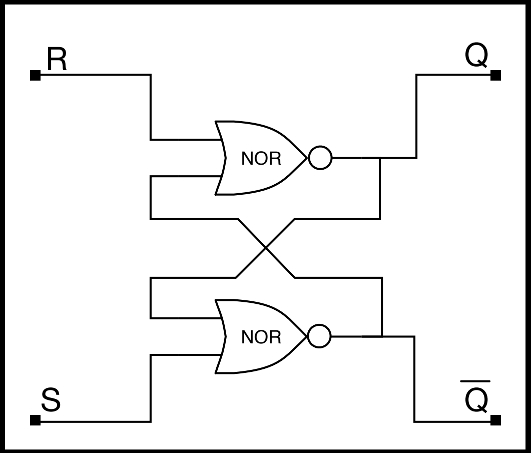
SR LATCH



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

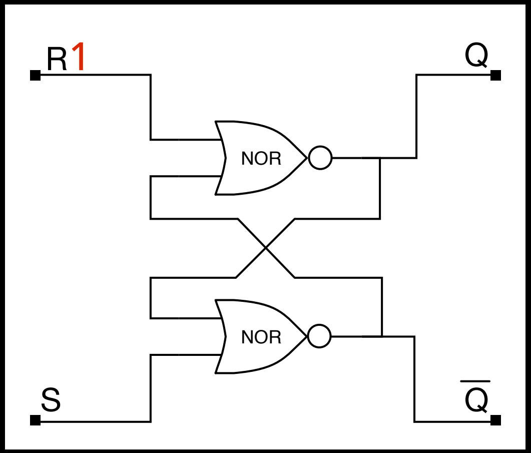
When R goes high, the outputs wobble and then Q goes 0
When R then goes low, Q still stays low
Causes Q to reset to 0

SR LATCH



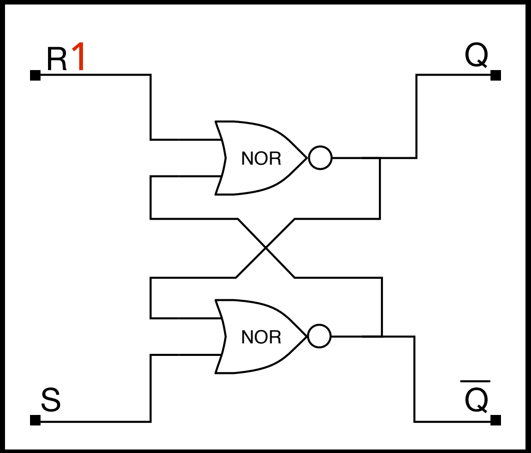
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0

SR LATCH



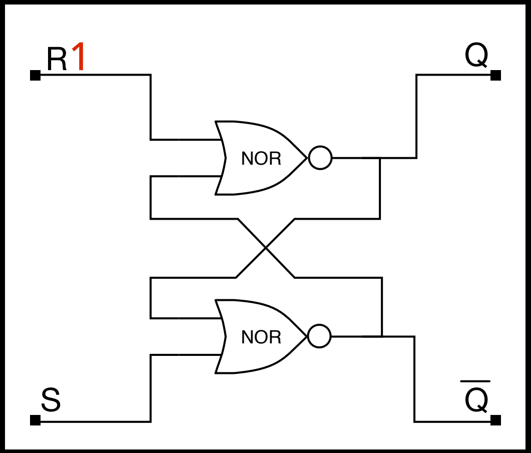
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0

SR LATCH



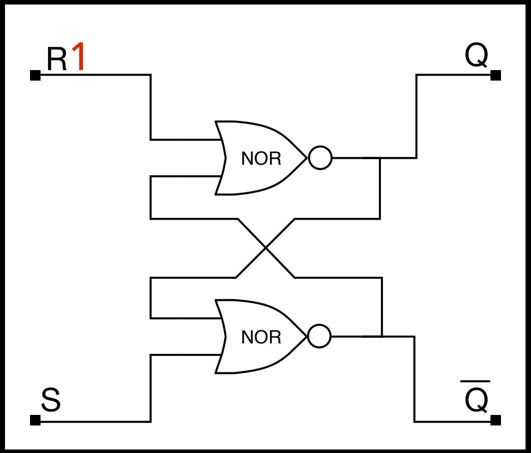
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0
1	0	1	0	0	0

SR LATCH



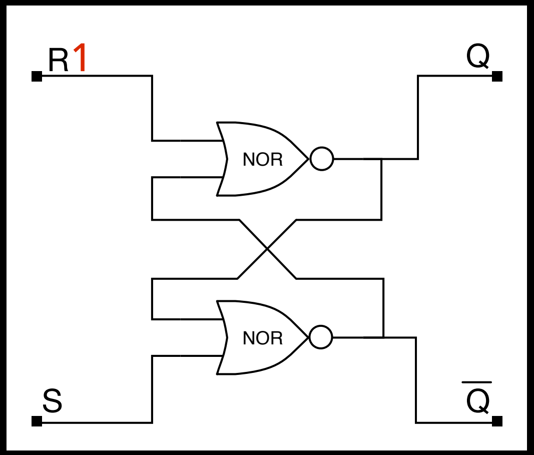
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0
1	0	1	0	0	0
1	0	0	0	0	1

SR LATCH



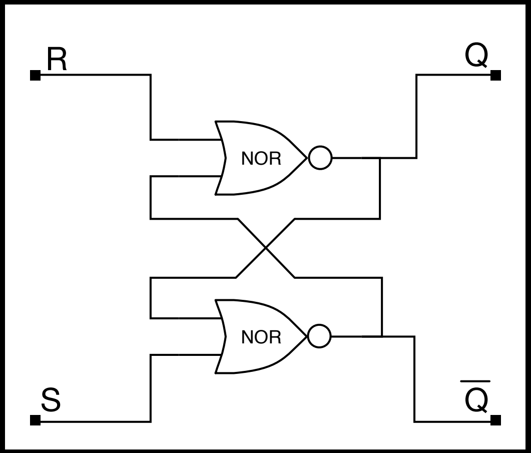
R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0
1	0	1	0	0	0
1	0	0	0	0	1
1	0	0	1	0	1

SR LATCH

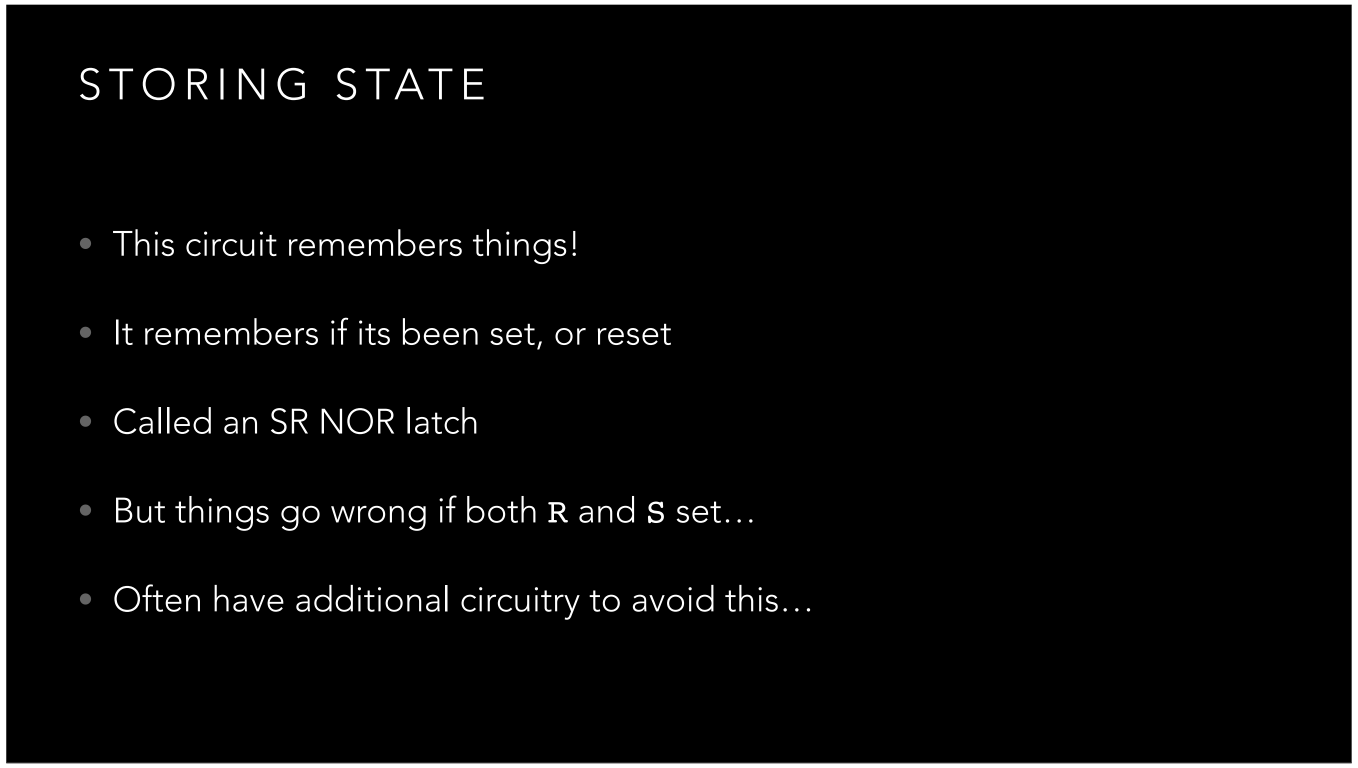


R	S	Q	$\overline{Q}$	Q_{next}	$\overline{Q}_{next}$
0	0	1	0	1	0
1	0	1	0	0	0
1	0	0	0	0	1
1	0	0	1	0	1
0	0	0	1	0	1

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



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