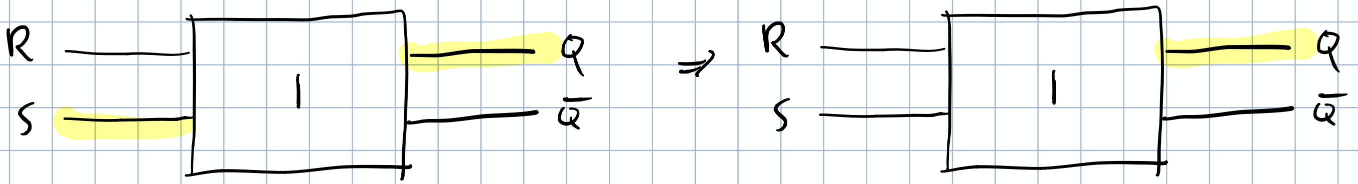
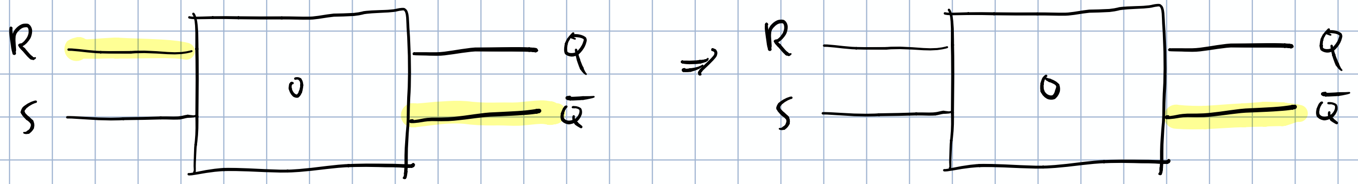
LATCHES

- storage elements operating on signal levels, rather than transitions are latches
  - ↳ level-sensitive
- " " " " clock transitions are flip-flops
  - ↳ edge-sensitive

- all flip-flops are made of latches
- latches useful for storing binary info. & async. circuits, but not practical for sync.

SR Latch

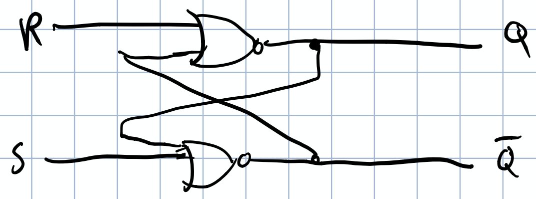




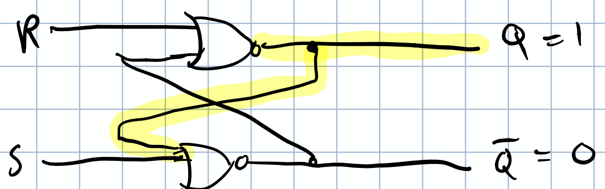
↪ Q memorizes previous input (R or S)
→ Q = 1 if S = 1 → 0
→ Q = 0 if R = 1 → 0

active high implementation

→ can be implemented using XOR or XAND

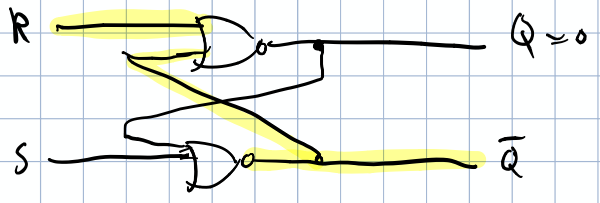


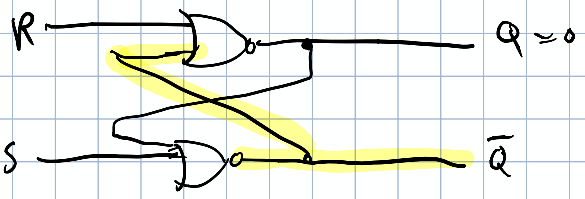
starting state



X	Y	NOR
0	0	1
0	1	0
1	0	0
1	1	0

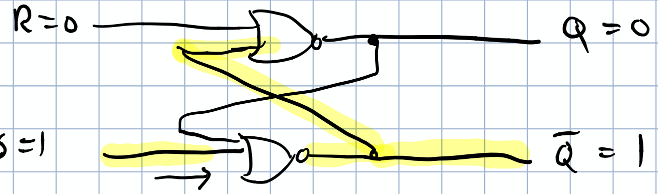
resetting state (R=1):



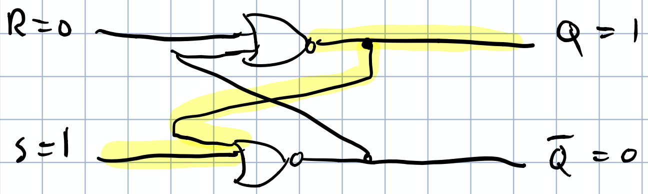


removing R keeps state

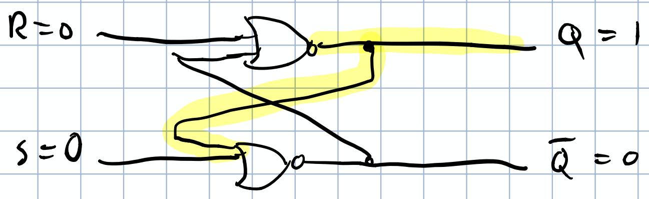
Case 3:
→ stores state of S



set state

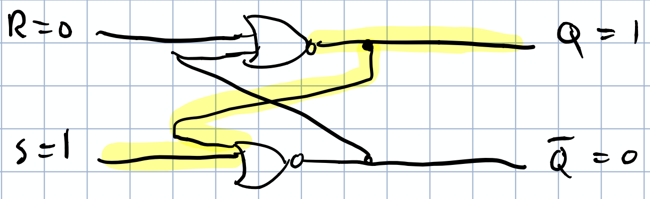


removing S:

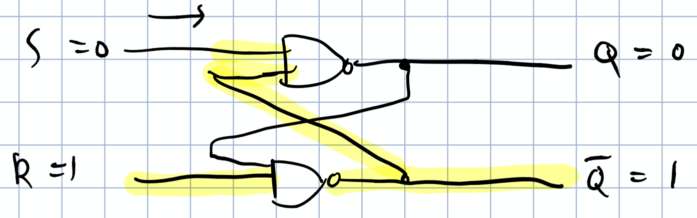


hold state

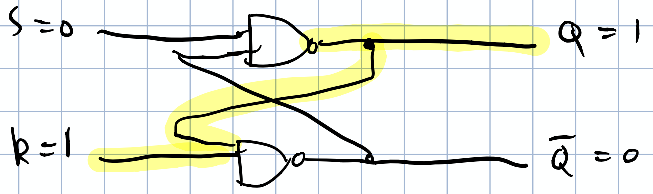
applying S again has no effect!



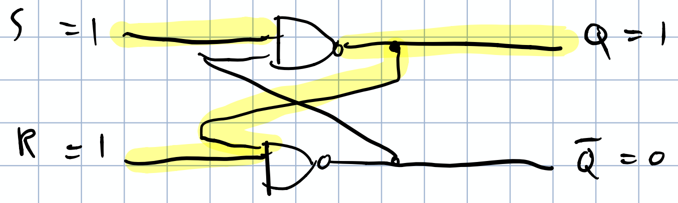
Case 2:
→ stores state of S



set state

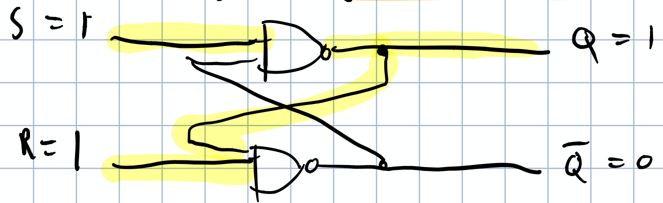


removing S:

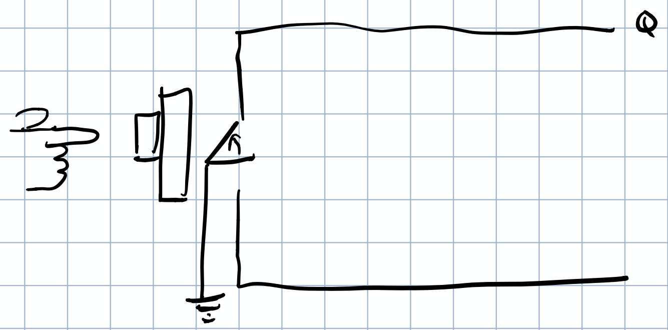


hold state

applying S again has no effect!



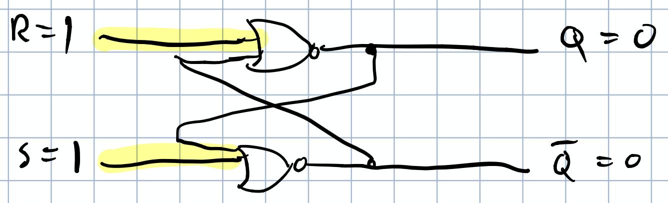
application:



switch might generate multiple on/off signals when pressed, causing issues
→ aka switch bounce
→ what if we could hold state after first on?

s	r	Q	$\overline{Q}$	
0	0	1	0	if prev state (1,0)
		0	1	else
0	1	0	1	
1	0	1	0	
1	1	0	0	invalid

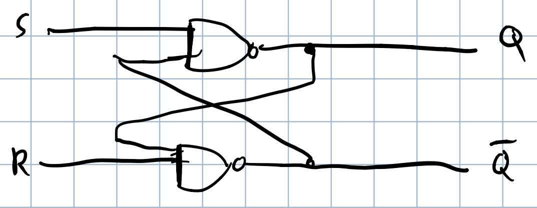
if R and S are both high



* if both fall to zero at same time, we have race condition
  - → whichever one falls last will determine the output
  - → impossible to tell next state of circuit w/o enough info

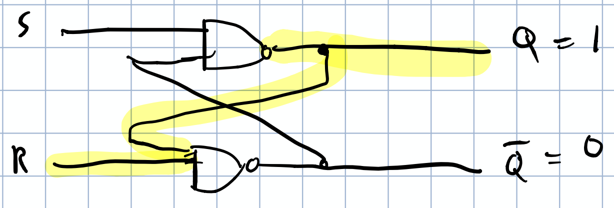
active low implementation

→ can be implemented with NAND



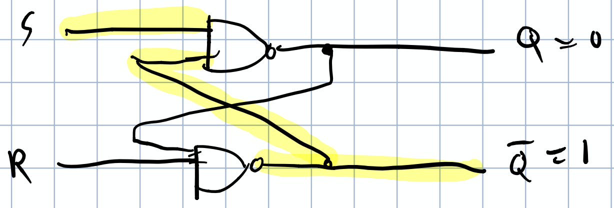
s	r	Q	$\overline{Q}$	
0	0	1	1	invalid
0	1	1	0	
1	0	0	1	
1	1	1	0	if prev state Q=1, $\overline{Q}$=0
		0	1	otherwise

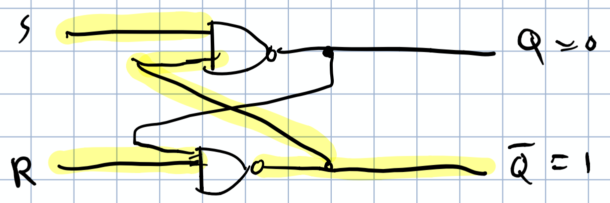
starting state



X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

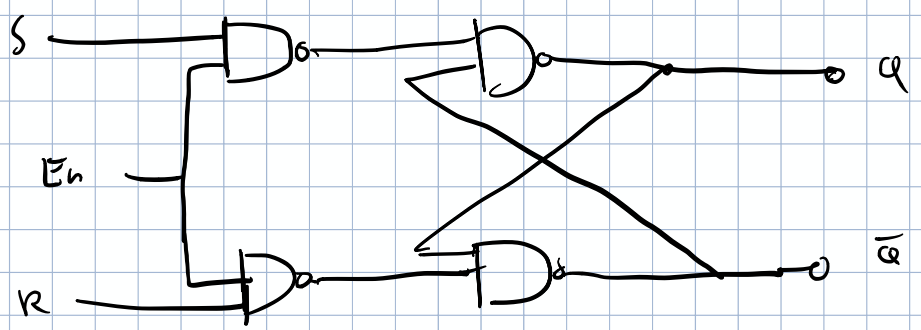
resetting state (R=1):





removing R keeps state

We can add an enable to the SR Latch



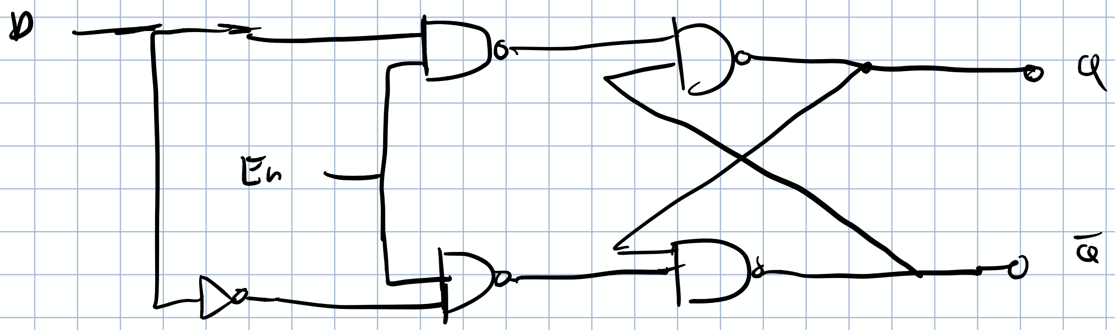
to reset, S = 0, R = 1, En = 1,

when S = 0, R = 0, En = 1, state is held
when En = 0, state is held.

if S = 1, R = 1, En = 1, indeterminate (race cond.)

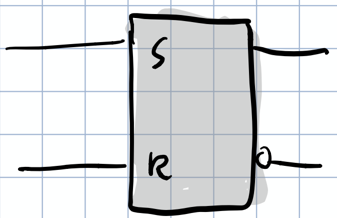
D Latch

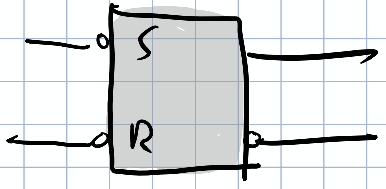
we can fix indeterminate by ensuring S & R not both 1.

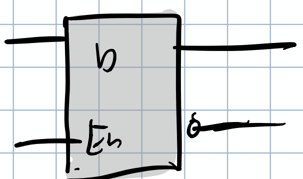


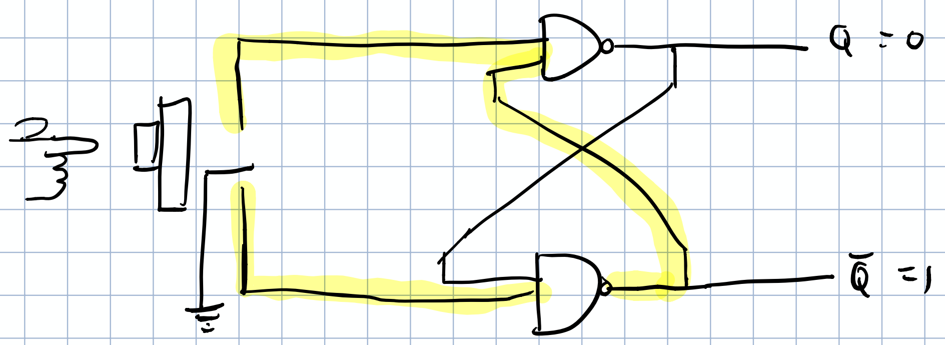
D is only sampled when En = 1.
↳ Set state ; Q always matches D

when En = 0, data is stored.

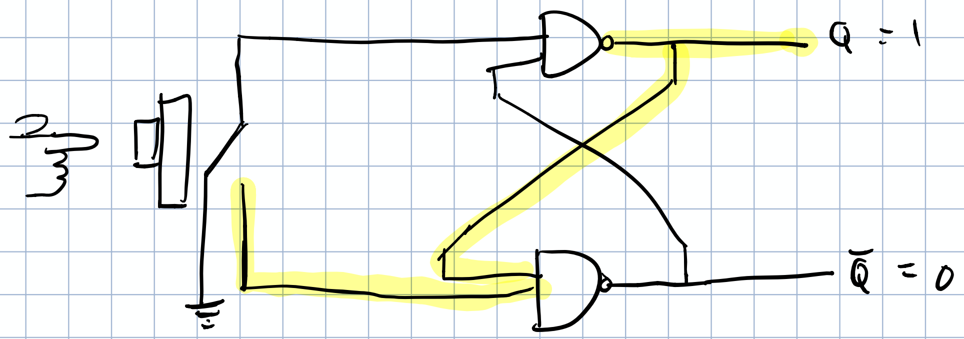

SR Latch

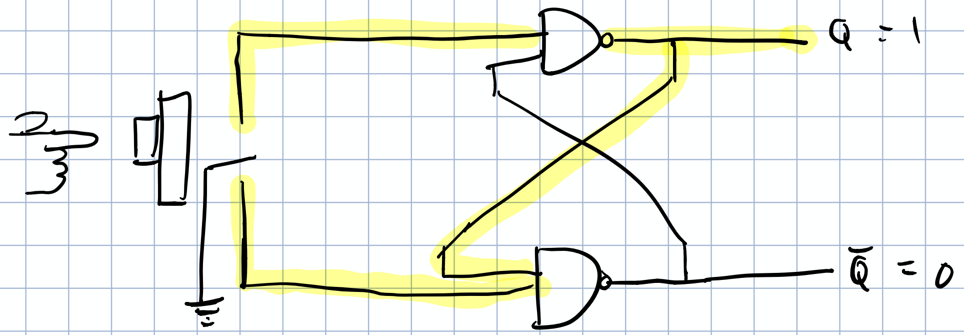

$\bar{S}\bar{R}$ Latch


D Latch



switch grounded, will set S to 0 when pressed





releasing switch/ pressing multiple times does nothing!