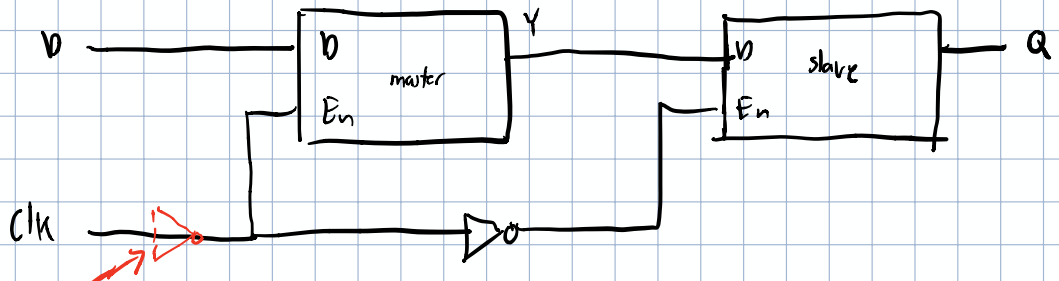
FLIP-FLOPS

- D latch with pulse in En is essentially flip-flop.
- as long as pulse remains at 1, changes in data input will change output
  ↳ this is a problem! how to get rid of noise?

  ⓐ use the edge of a clock pulse instead of the active high
- Flip-flops solve this problem by either
  1) Configuring master/slave latch to prevent changes
  2) Produce flip-flop that triggers only during signal transition

EDGE-TRIGGERED D FLIP-FLOP

→ Recall that latching saves state & turning "S" on/off has no effect.



On positive edge, D is sampled and the state is "saved" in Y.

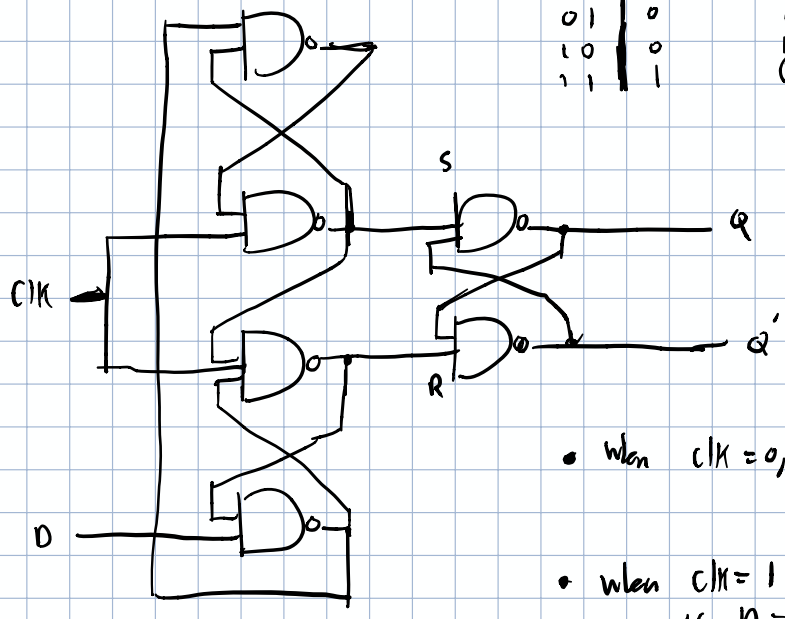
On negative edge, Y propagates to Q.

★ What if D changes multiple times through logic-1 state?
↳ last value before negative edge is preserved.

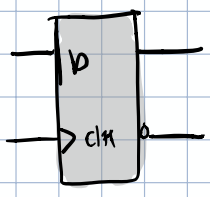
? how do we make it trigger on positive edge instead?
↳ add another inverter

Alternatively,

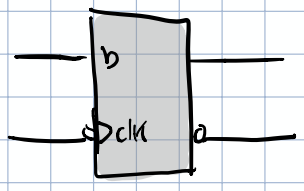
x y	AND	NAND
0 0	0	1
0 1	0	1
1 0	0	1
1 1	1	0



- when clk = 0, S & R are active and state is stored
- when clk = 1;
  - if D = 0, R = 0
    ↳ changing output has no effect while clk = 1
  - if D = 1, S = 0
    ↳ changing output has no effect while clk = 1



positive edge



negative edge

Similar to latch, but ">" indicates dynamic input, triggered at edge of clk.

- edge-triggered D flip-flops are most efficient (least # of gates)

D	Q(t+1)
0	0
1	1

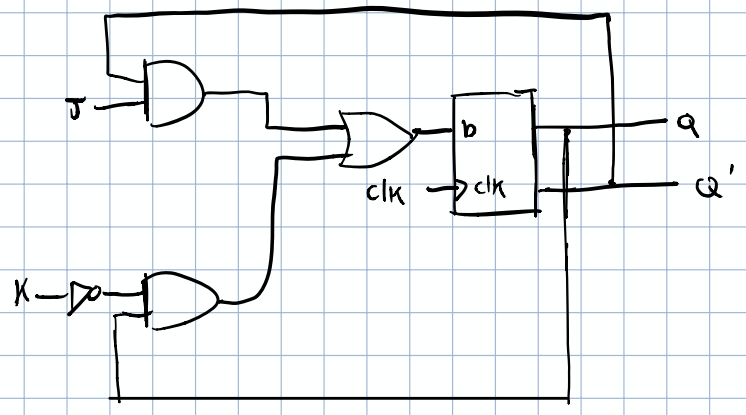
Characteristic table for DFF

JK-FLIP FLOPS

3 operations can be performed with flip-flops:

1) set to 1
2) reset to 0
3) complement output

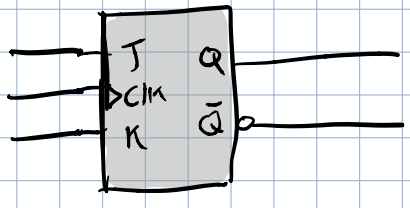
the JK flip-flop can perform all of these three operations



- flip's feedback as an extra layer of safety
  - ↳ set to 1 if previously zero & value is 1 & clk
  - ↳ also logic for inverting / hold.

$$D = JQ' + K'Q$$

- J input sets Q to 1
- K resets Q to 0.
  - ↳ when K=1, D=0 so Q=0 when clk.
- when J=K=1, D = Q', so Q is inverted.
- when J=K=0, D = Q, so Q is held.

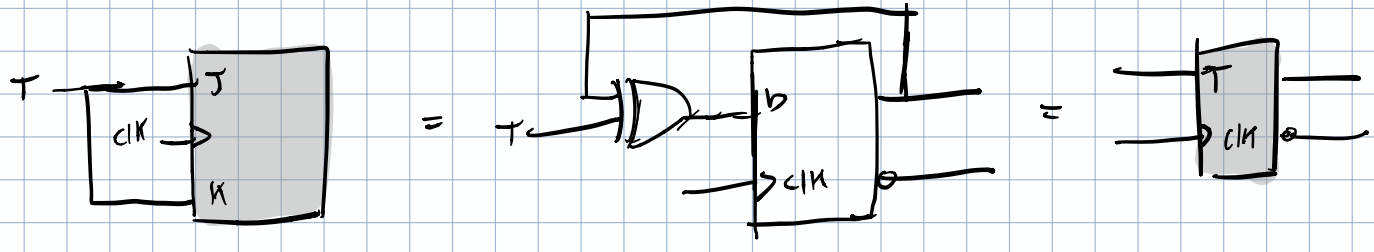


J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

T FLIP-FLOP

Toggle Flip-Flop

We can obtain a T flip flop by tying J & K together...



- J & K are always the same
  - → if J = K = 1, state toggle
  - → if J = K = 0, hold
- $D = JQ' + K'Q = TQ' + T'Q = T \oplus Q$

  $J = K = T$
- useful for designing binary counters

T	Q(t+1)
0	Q(t)
1	Q'(t)

1, 1, 2, 3, 5, 8, 13