06-storage-notes

Storage Elements

Agenda

- 0. Re-Orienting
- 1. Our Logic So Far
- 2. Latches
- 3. Some Simple Applications
- 4. Edginess
- 5. Flip-Flops

---

0. Re-Orienting

---

electricity to gates

---

representing numbers as binary

---

building circuits that calculate functions on binary numbers

---

the physical reality of the circuits can affect the logic design. (The abstraction barrier has holes in it!)

- gates take time to respond
- wires take time to carry signals
- gates cost
- zeros can cost (due to the convention of asymmetric current)
- one output can't drive too many inputs

and (for cultural edification) we talked about how things get built

- discrete transistors, or small ICs, or large ICs (including FPGAs or ASICs)
- on breadboards or wirewrapping or printed circuit boards
- probably via a "program" in an HDL, which gets compiled (and optimized) down to a circuit layout

1. Our Logic So Far

So far, the digital logic circuits we've built have all been DAGs: directed acyclic graphs

"Combinatorial logic": the outputs depends on the inputs right now.

What if we do something weird, like introduce cycles?

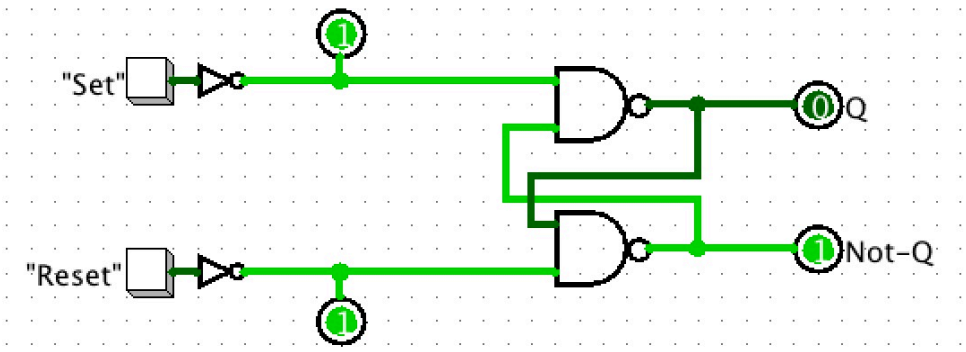
In **rs-two-push.circ (https://ssl.cs.dartmouth.edu/~sws/cs51-s15/06-storage/demo/rs-two-push.zip)** ...

- trace through the values
- press a button, then trace through. (think of the inputs as "active low"...but LogiSim only gives active-high pushbuttons, hence the inverters)
- notice what happens when you release the button!
- (when i was a kid, I ended up building things like this for "block signals" in my neighbor's model railroad layout)

2. Latches

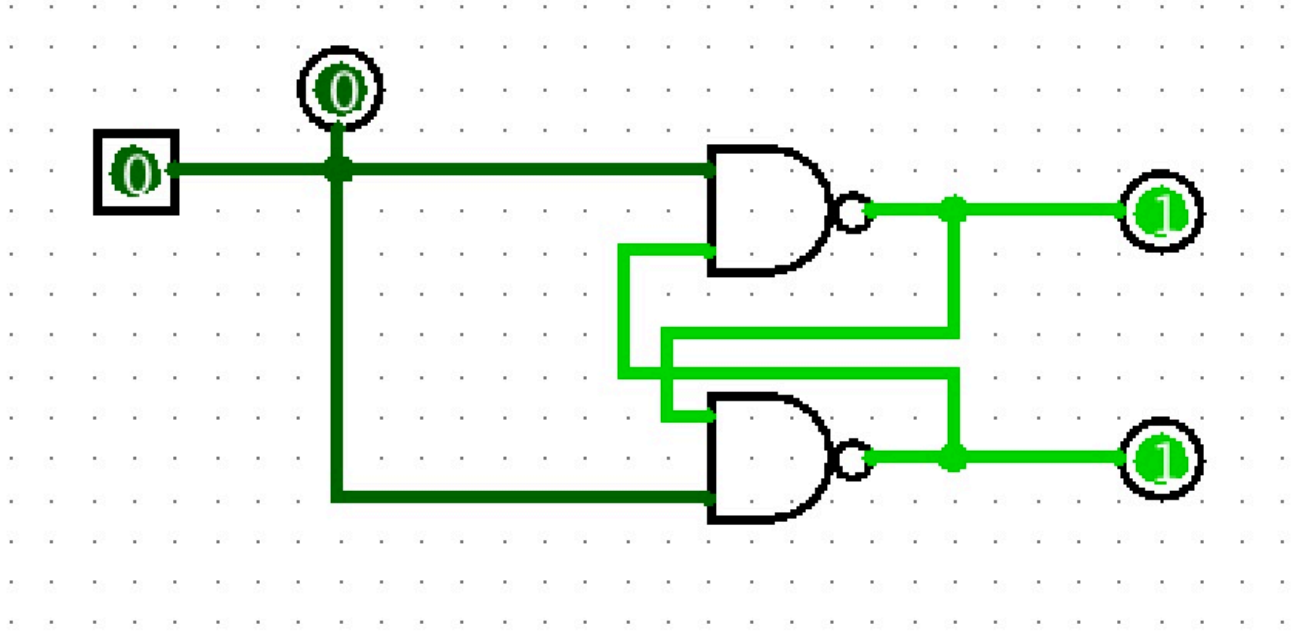
This thing we get by wiring a pair of NANDs this way (with sort of positive feedback) is called an "SR latch"

- "set", "reset".
  - note that "set" usually means "make true"
  - and "reset" usually means "make false"
  - (so if you say "set to 0," people may look at you oddly)
- "latch" (Folks like to use "Q" for "the bit of data we care about.")



What else can we do with it?

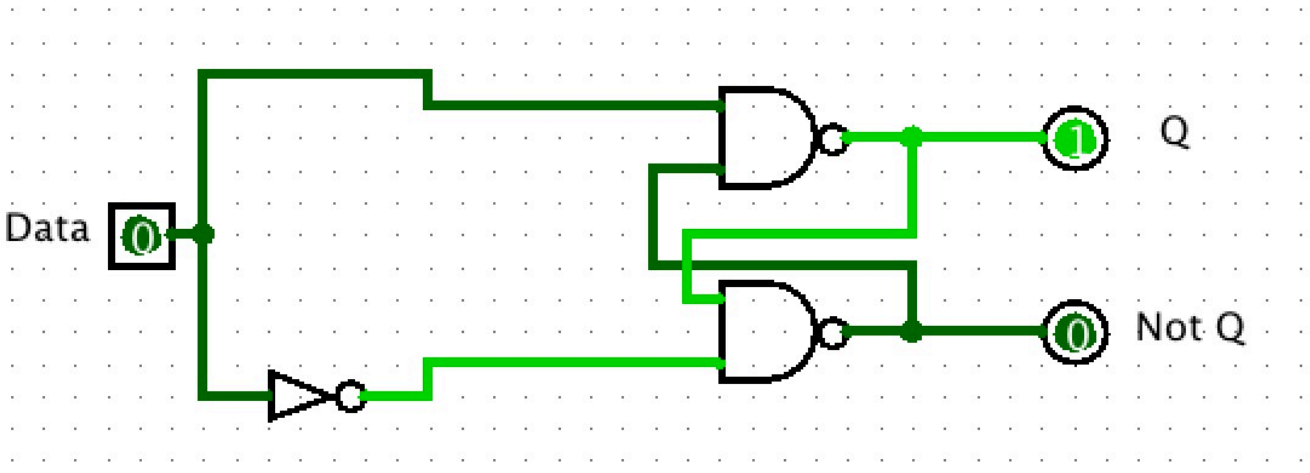
Can we control it with one pushbutton switch?



- What if try it in LogiSim? **rs-one-push.circ (https://ssl.cs.dartmouth.edu/~sws/cs51-s15/06-storage/demo/rs-one-push.zip)**
- What the heck is going on here?

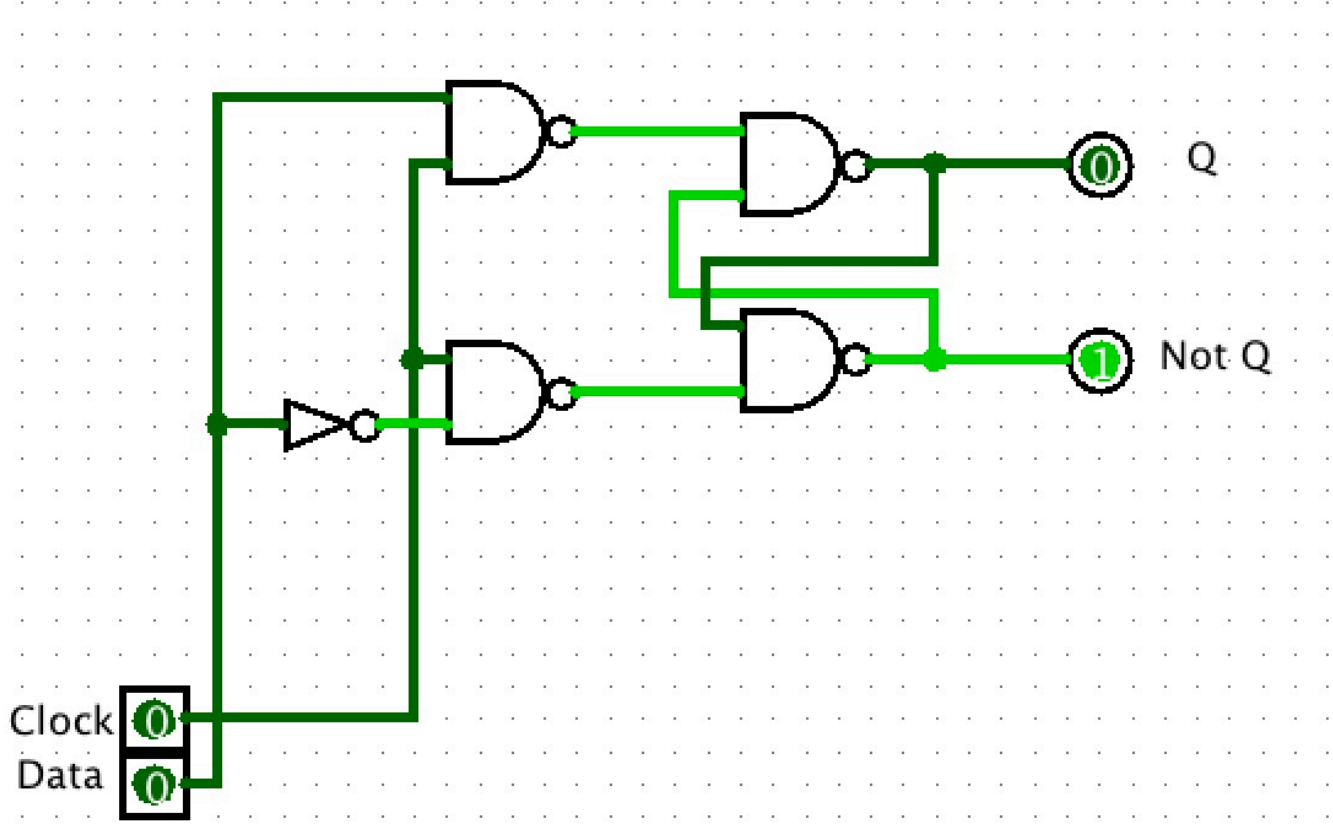
Better ensure that we don't pull set and reset low at the same time.

See "attempt 1" in **latch.circ (https://ssl.cs.dartmouth.edu/~sws/cs51-s15/06-storage/demo/latch.zip)**



That's not too interesting. Let's add another line, to enable the latch to see the set/reset. (See "attempt 2" in latch.circ.)

"write enable" is also called "clock" (but be warned... terminology fun is looming)



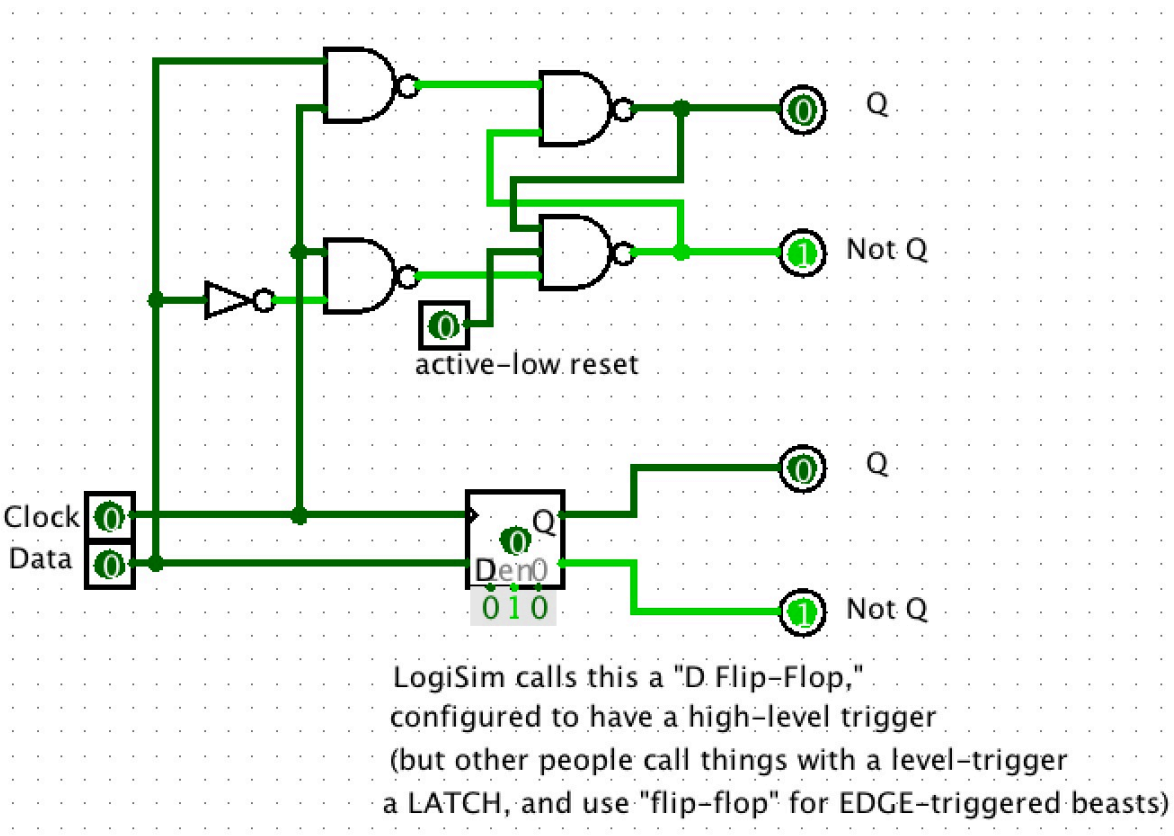
- LogiSim can get confused when you open up the file, because it's not sure what to do with the state. A workaround is to use a three-input NAND, with the third input as an active-low reset. See "attempt 3".

This is useful enough that there's a baked-in LogiSim component.

- what LogiSim calls the "D flip-flop"
- but configured with a **level-triggered clock**

See "attempt 4".

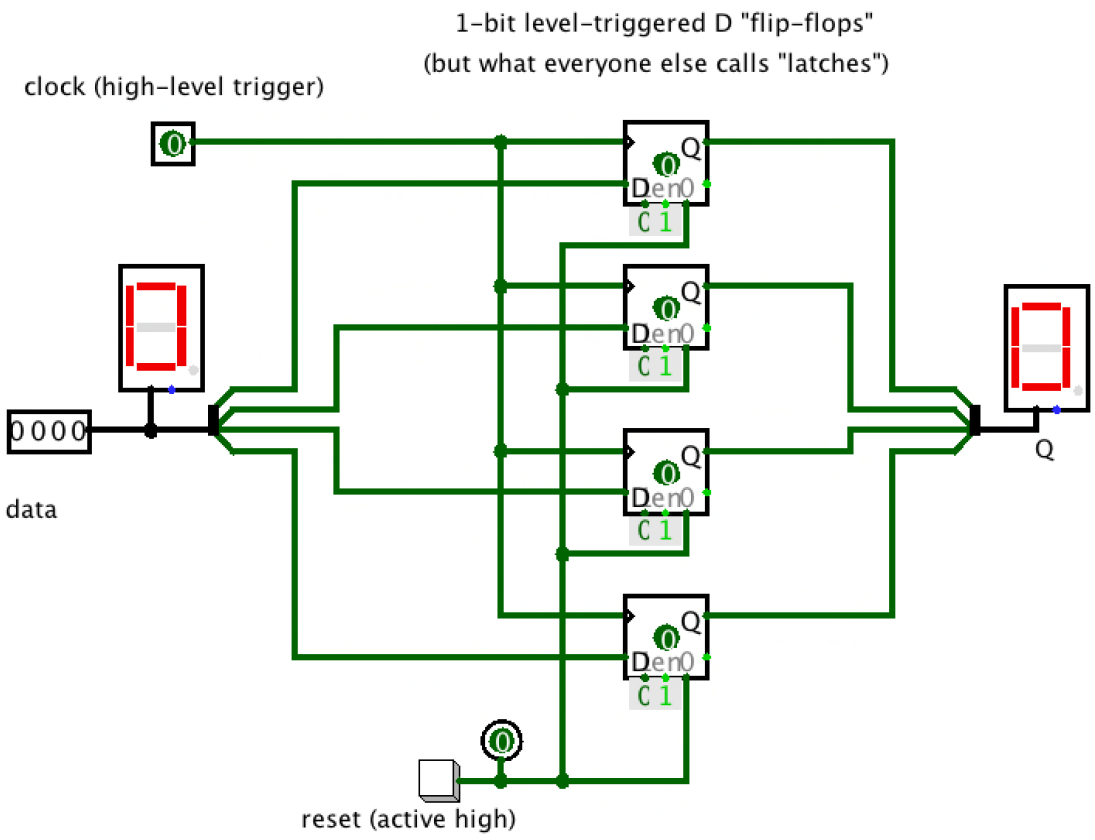
- I added constants for the control lines on the bottom of the baked-in LogiSim "flip-flop."



---

3. Some Simple Applications

a 4-bit latch: [4bitR.circ (https://ssl.cs.dartmouth.edu/~sws/cs51-s15/06-storage/demo/4bitR.zip)](https://ssl.cs.dartmouth.edu/~sws/cs51-s15/06-storage/demo/4bitR.zip)



(Built from the 1-bit D flip-flops, with high-level trigger, from LogicSim)

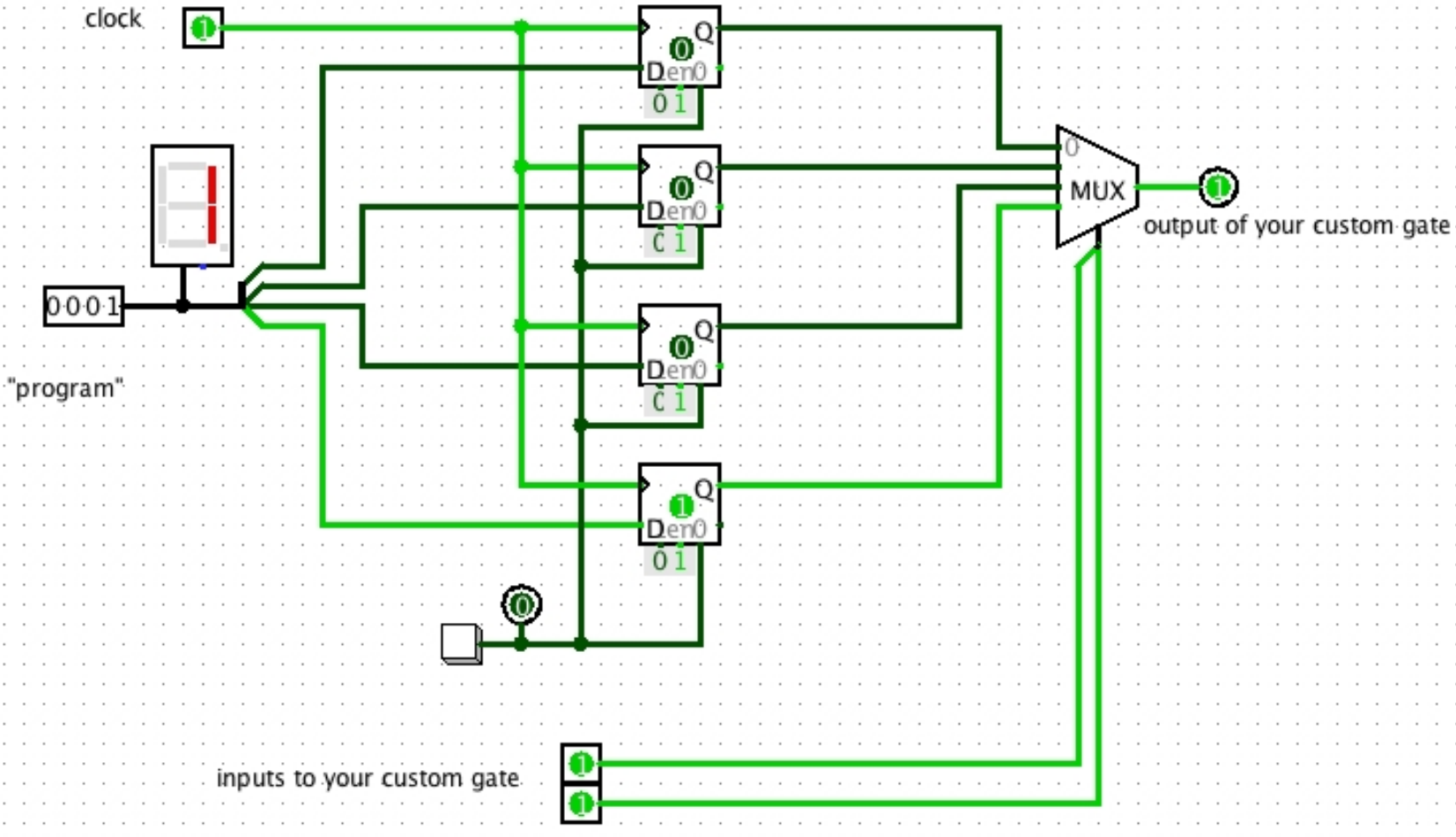
And recall....

- An 8-bit value is called a **byte**
- A 4-bit value is sometimes called a **nibble** (really)

What if we combined 2^n D flip-flops with an n-bit multiplexor?

- [lut.circ (https://ssl.cs.dartmouth.edu/~sws/cs51-s15/06-storage/demo/lut.zip)](https://ssl.cs.dartmouth.edu/~sws/cs51-s15/06-storage/demo/lut.zip)

We end up with a programmable look-up table! What n-bit function would you like to calculate? (This is how some FPGAs work! Imagine, instead of the hex keypad, you have your "program" streaming into your chip)



4. Edginess

---

in a latch, what happens if the data changes when the clock is high?

Try sketching out a timing diagram (D, WE, Q, over time...) to show what happens...

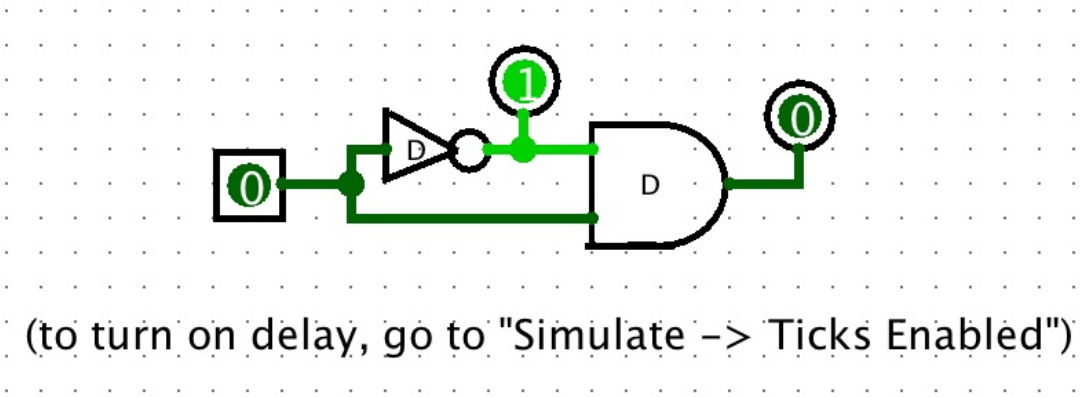
This may strike you as surprising. Wouldn't it be more intuitive (or at least interesting) for the storage circuit to sample the data at an "instant"?

- (point out the "rising edge" of the WE line)

But how?

One approach: use physical properties of the circuit to turn a rising edge into a brief 1...and as we demonstrated earlier.

Here's a revised version (with LogiSim told to explicitly simulate delays in both the NOT and AND gates).



See "riser" in [edgy.circ (https://ssl.cs.dartmouth.edu/~sws/cs51-s15/06-storage/demo/edgy.zip)](https://ssl.cs.dartmouth.edu/~sws/cs51-s15/06-storage/demo/edgy.zip)

- Walk through what happens when you turn the switch on
- logically, what SHOULD happen is that the AND never fires, because its inputs can never both be one simultaneously
- But because in the real world, the inverter takes nonzero duration to respond, there will be a brief moment, at the rising edge of the switch signal, when both inputs are one
- Thus, the rising edge turns into a brief one (the output of the AND)

---

5. Flip-Flops

As you will see in HW3, we can plug this back into the latch, and then have a cell that samples the data at the "single instant" of the rising edge of the clock.

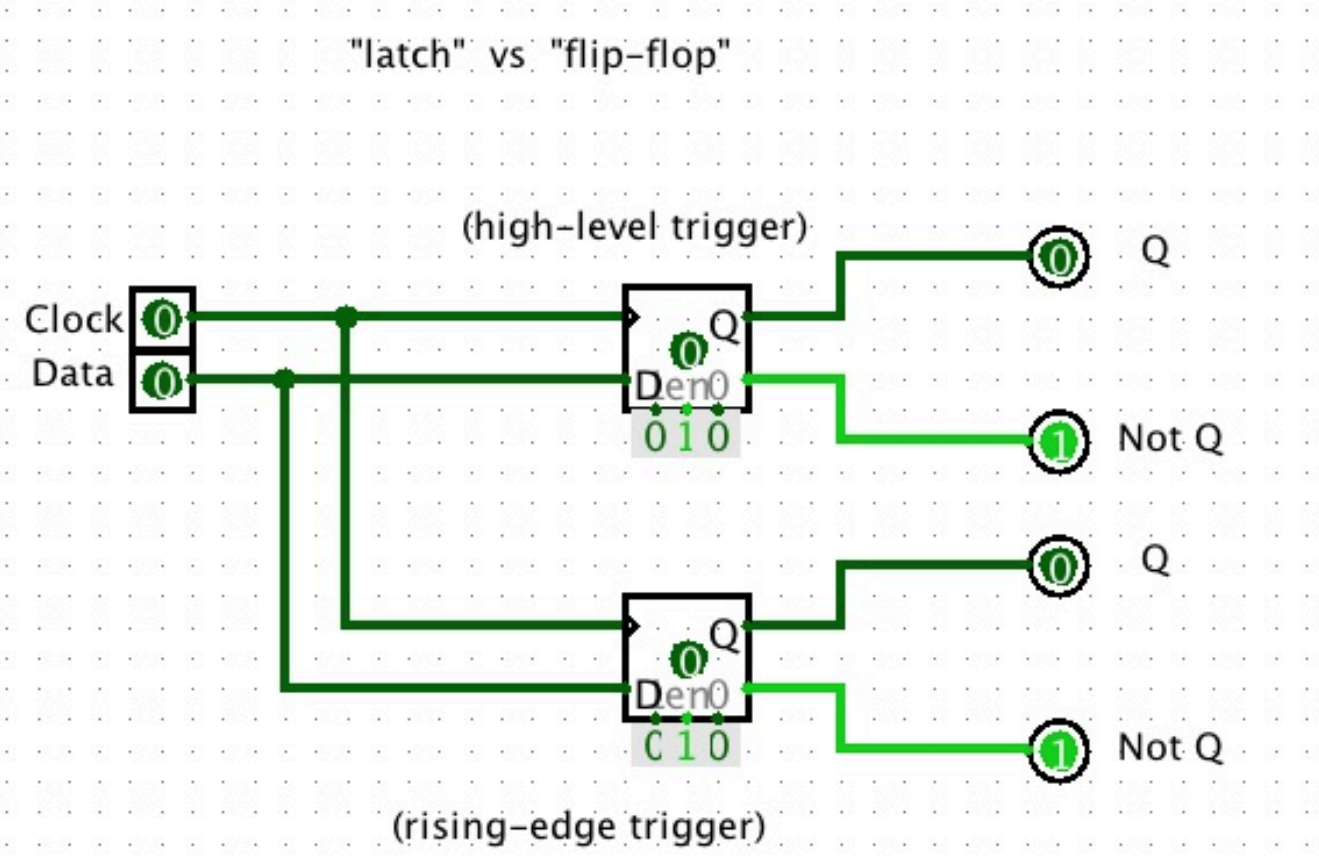
Let's go through this carefully.

- the RS-latch
- turning a data bit into R and S signals
- using physics to turn the clock edge into a short pulse

Pedantic folks call this edge-sensitive version a FLIP-FLOP. The other is a LATCH. But as with all pedantic terminology, you will find it used inconsistently in the real world.

See "latchflop" in edgy.circ.

- compare and contrast happens when you change the data while WE is asserted


"latch" vs "flip-flop"
(high-level trigger)
Clock
Data
Q
D
en
0 1 0
Q
Not Q
Q
D
C
C 1 0
Q
Not Q
(rising-edge trigger)