Last time, we formed an R/S-latch (2 binary inputs, $Q$ and $\neg Q$ as output) and a D-latch (added clock input).

![Diagram of an R/S-latch and D-latch]

The clock input was an alternating signal that was flipping between 0 and 1 on some time scale. To figure out the output of a D-latch, we have to think about 1) whether the latch is open or closed (ie. the clock is 0 or 1); and 2) what value $Q$ takes on with that input.

How do we think about this? We have a stored-value $Q$. Take $D$ in and record the value of $D$ to $Q$. Or the value of $D$ comes in but $Q$ does not change, it holds. Open the latch, write the value of $D$ to $Q$. Close the latch, hold the value of $Q$ until the next time the latch is opened.

We can think about an R/S-latch process with state diagrams.

We can also think about the subset of a D-latch that comes after the AND gates as an R/S-latch with inputs $C \land \neg D$ and $C \land D$.

- If $C = 0 \implies S = 0, R = 0 \implies Q$ holds.
- If $C = 1, D = 1 \implies Q \to 1$. $Q$ gets the value of $D$.
- If $C = 1, D = 0 \implies Q \to 0$. $Q$ gets the value of $D$.
One limitation of this model is that things can change the entire time that the clock is open. We’d like to restrict/know better when things can change. If you know the instant that things are read/written, that’s more powerful than just knowing the timeframe during which the change can happen.

How do we do this? We’re going to box off the D-latch, and use two of them to create a Flip-Flop circuit. (Note: we don’t use the first $\neg Q$ in the bigger circuit).

Q switches to match D at the moment that C flips from $1 \rightarrow 0$ – this is called a “falling edge” circuit.

Q switches to match D at the moment that C flips from $0 \rightarrow 1$ – this is called a “rising edge” circuit.

The value of D can be fed in at any point up until Q switches. This gives us efficiency in how we use our circuits.