(Finite) State machines

Imagine a machine that has a finite number of states it could be in at any 1 time to perform a task

⇒ has inputs & outputs

⇒ well defined states

⇒ transitions from 1 state to the other depends on inputs & current state

⇒ outputs are determined by what state it's in

ex: build a digital controller for traffic lights

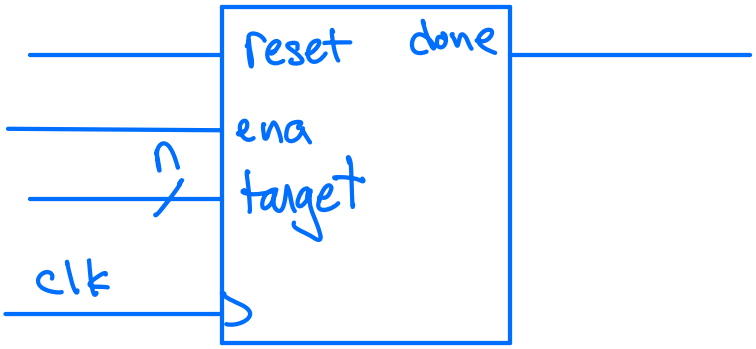
want the transition G→Y to take a time T_g

Y→R " T_y

R→G T_r

want to be able to "program" T_g, T_y, T_r

so need 3 "timers"



ena enables a count-up counter

reset sets the counter to 0

target is a n-bit number

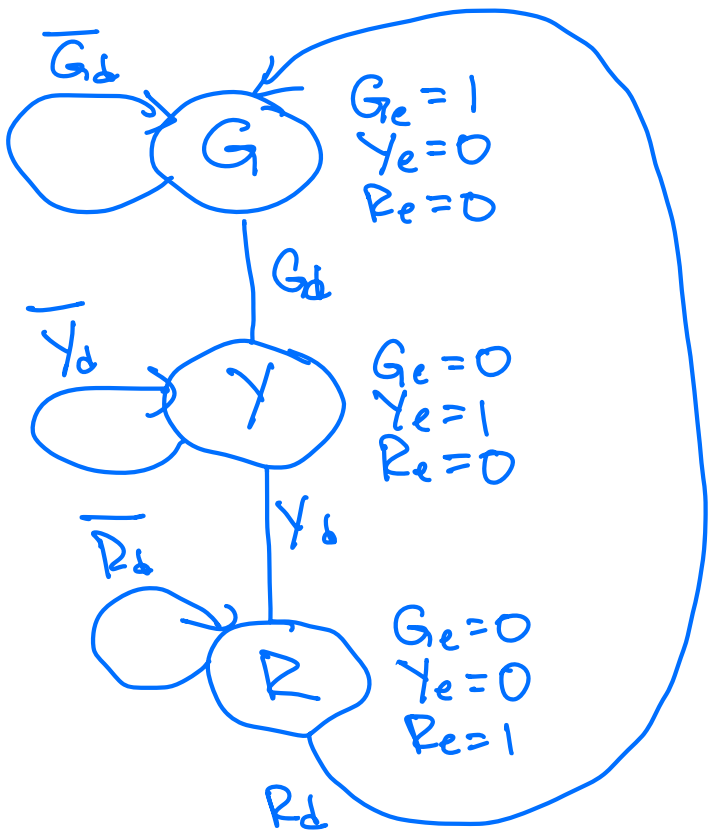
done goes high when the counter = target

note: reset = $\overline{ena}$ (when not enabled, reset to 0)

3 done lines: G_d, Y_d, R_d input to FSM

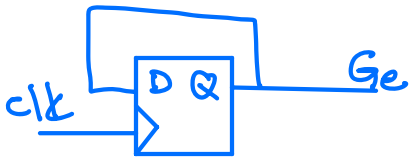
3 ena " : G_e, Y_e, R_e output of FSM

3 signals to turn on the 3 lights, can use timer ena lines



we can build this out of DFFs and logic gates

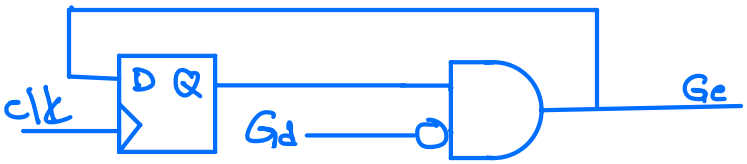
start in G state



with feedback, stays in the state

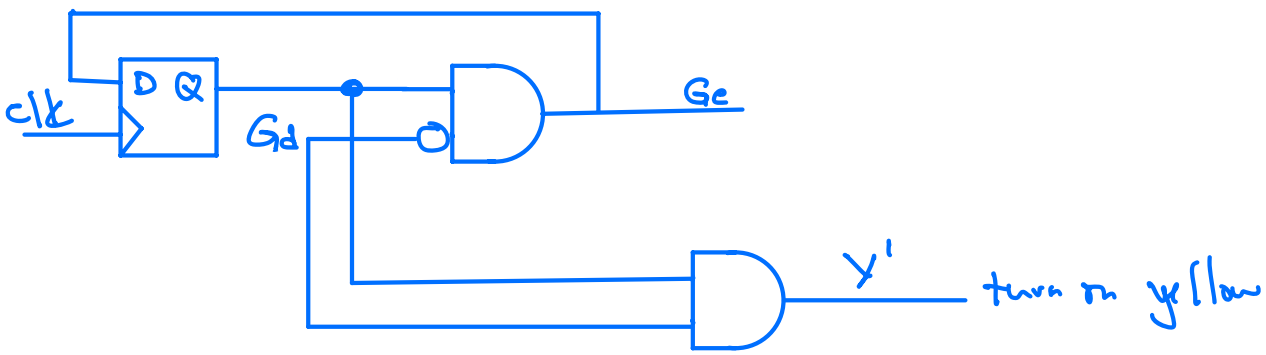
Ge asserted turns on green light and starts green timer

⇒ But want to go to yellow when Gd is asserted



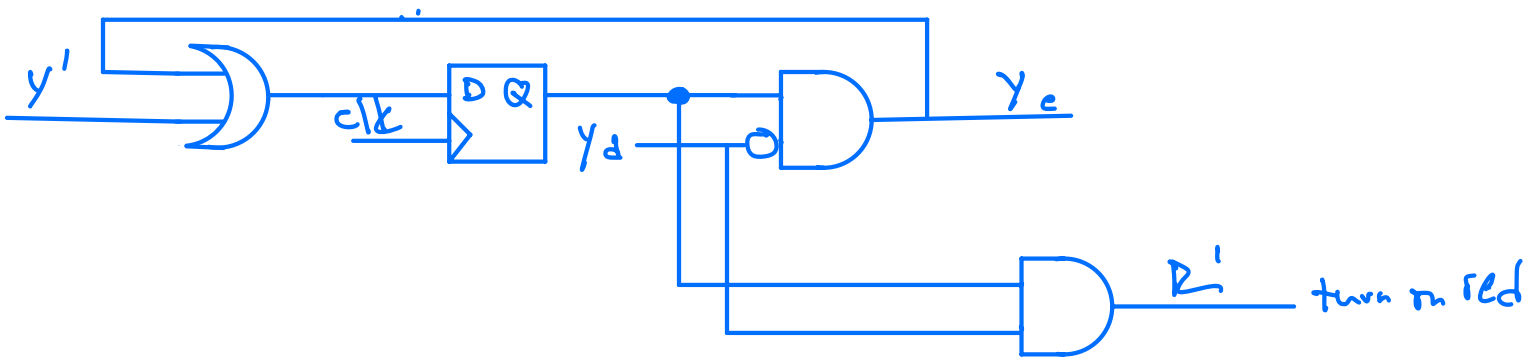
now we transition to Y state on next clock tick

condition is that Gd is asserted AND we are in G state

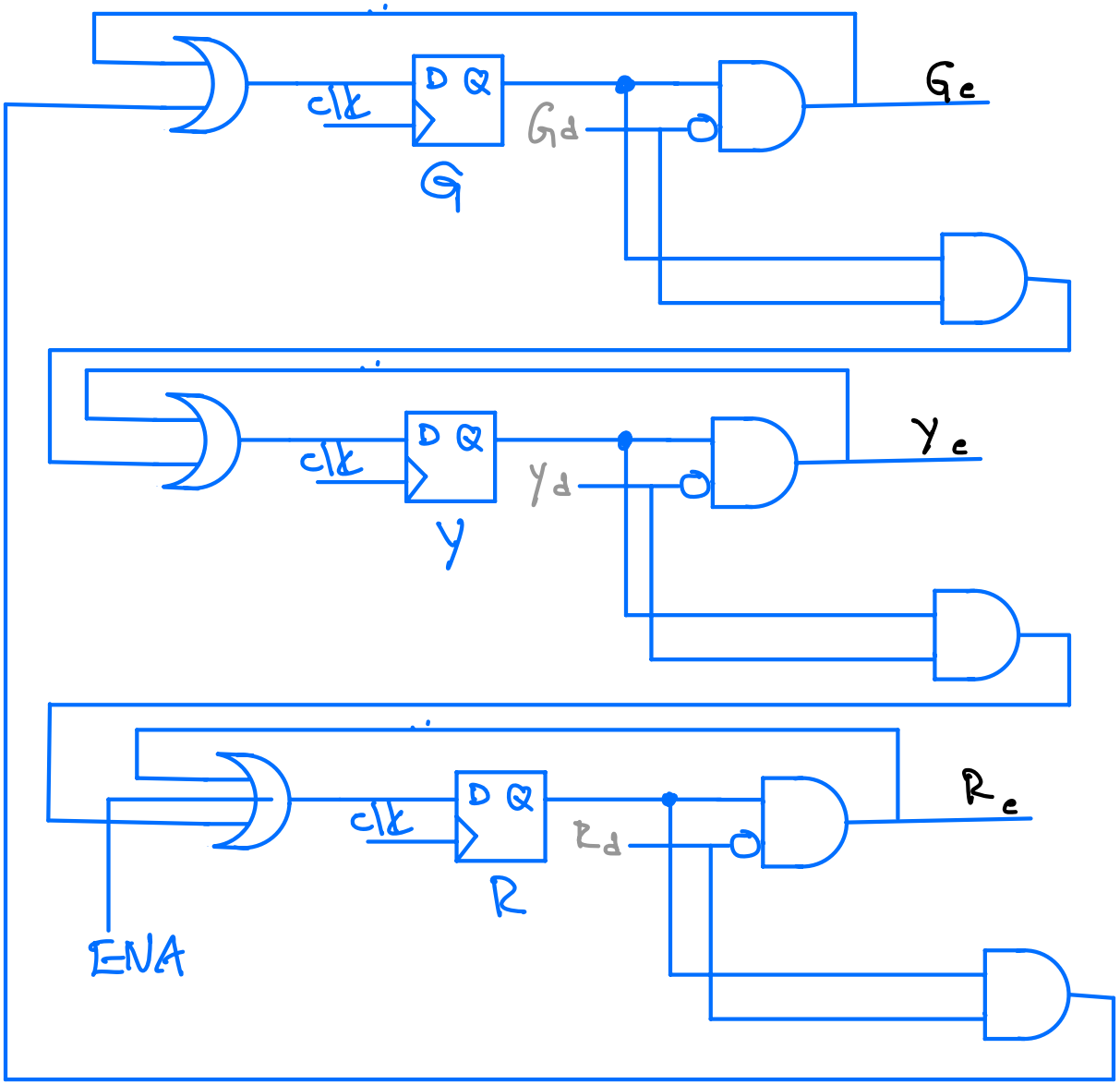


Y state has same feedback circuit as G state

⇒ but it is turned on by Y' or feedback circuit



now put it all together:



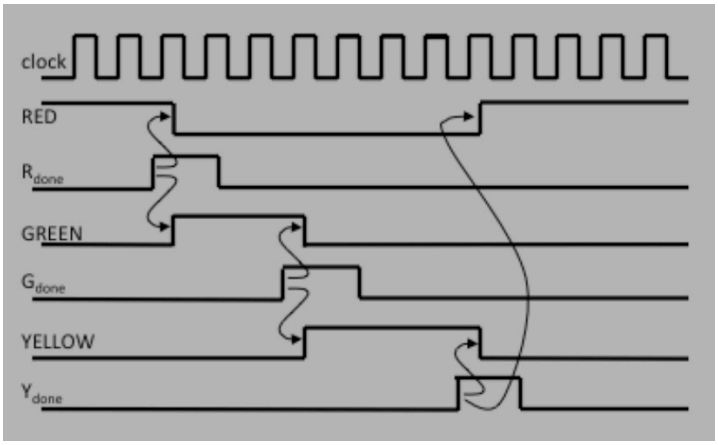
Note: timer enables also turn on the lights

=> when done is asserted, the light goes out after a single AND gate

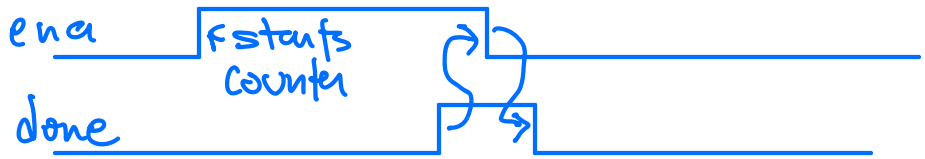
=> transition to next state happens at posedge of next clock via DFF, so can't have 2 lights on at same time!

note: need a global start ENA

Timing diagram



note that done signal is a pulse → comes from timer, which is reset when enable signal goes away



this is called a "handshake":

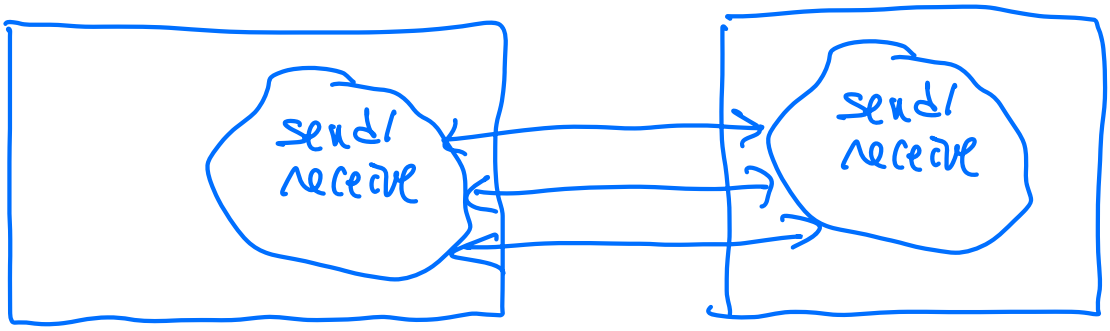
1. ena asserted starts counter
2. done asserted when counter reaches target
3. ena deasserted when done asserted
4. done reset (deasserted) when ena deasserted (waits!)

done doesn't have a required width since it is just used by the FSM to trigger a transition

Handshake

Often, 2 state machines "talk" to each other

ex:

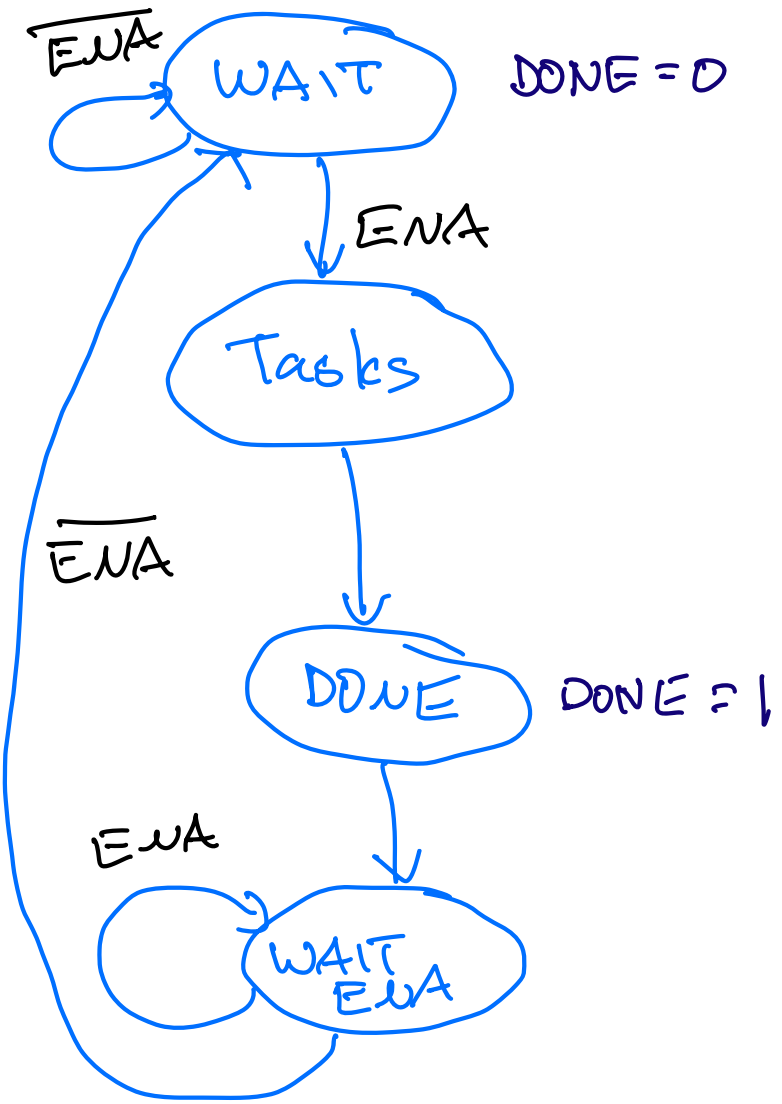


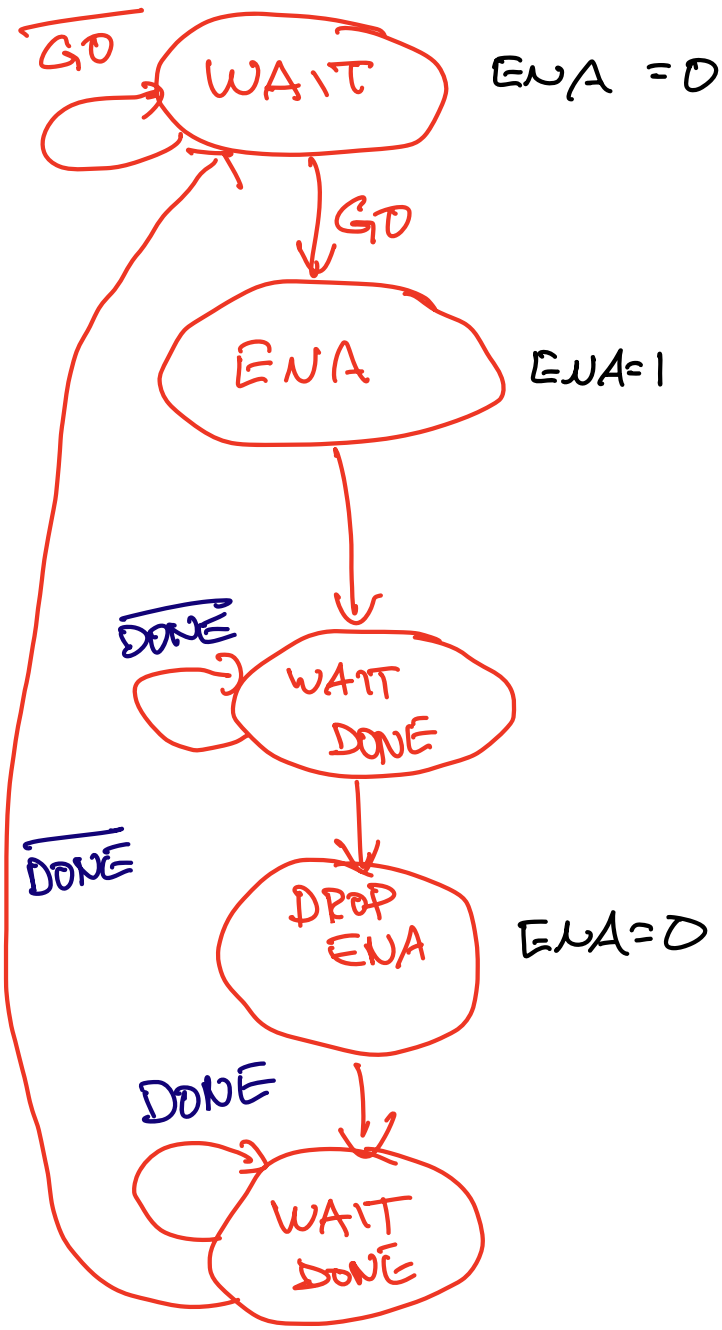
communication

each FSM sends signals to the other
⇒ need to be clear on the protocol

ex:
- FSM1 receives ENA from FSM2
- FSM1 then performs a task & sends DONE
- FSM2 waits for DONE

"Careful" handshake



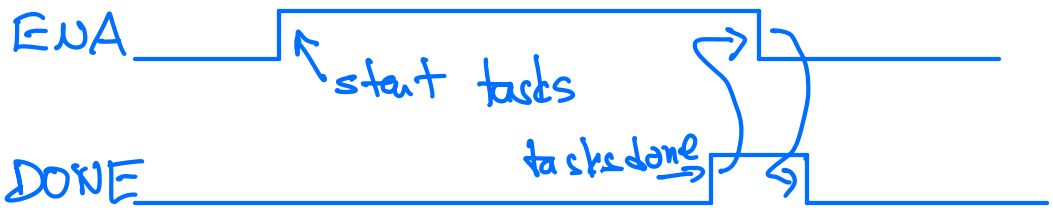


1. GO starts red FSM
2. Red → ENA state, asserts ENA signal and goes to WAITDONE state to wait for DONE signal
3. Blue FSM sees ENA & execute tasks. When finished, go to DONE state and assert DONE signal and goes to WAITENA to wait for ENA to go away

4. Red sees DONE asserted, goes to DROPENA state and deasserts ENA (set to 0) and then goes to WAITDONE state

5. Blue sees ENA deasserted & goes back to WAIT state where DONE is deasserted

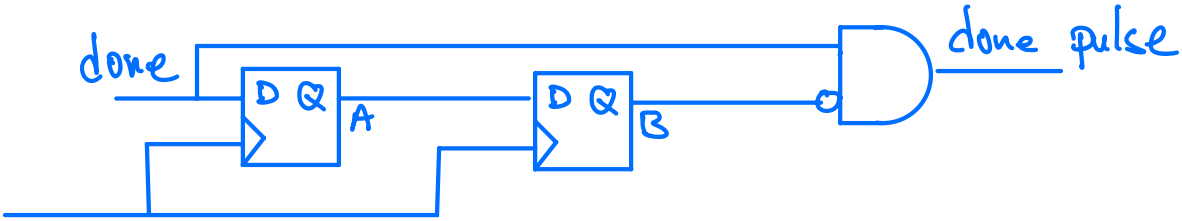
6. Red sees DONE deasserted & goes back to WAIT



HANDS SHAKE!

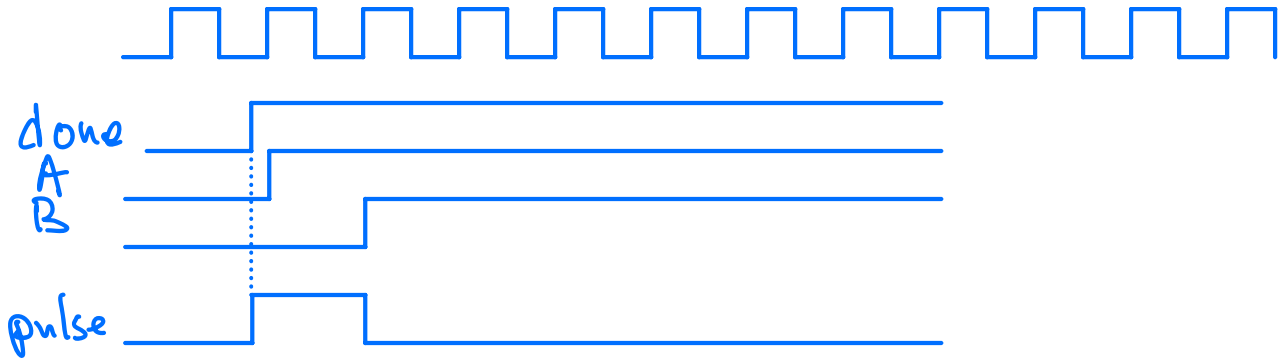
⇒ often, a signal like DONE can stay asserted but have no meaning → somewhat dangerous, can lead to faulty transitions

⇒ here's how you can assure a signal like done has a finite width: "one-shot"



this assumes "done pulse" will have width 1 - 2 ticks

ex: done asserted close to posedge



ex: done asserted far from posedge