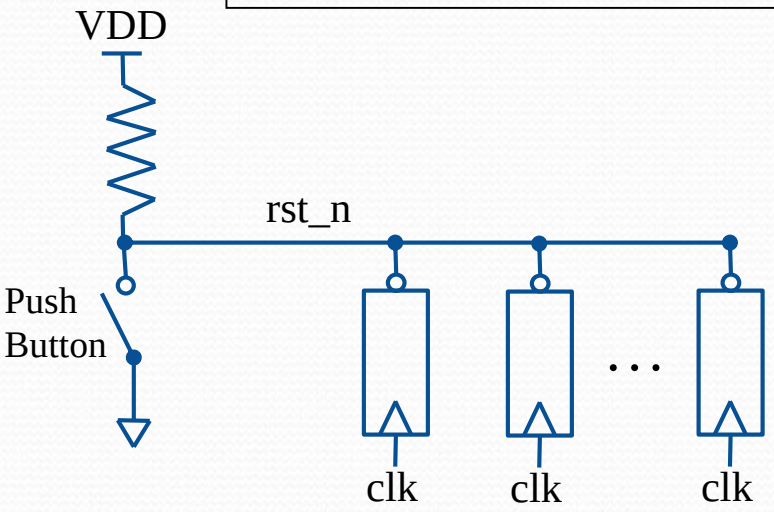
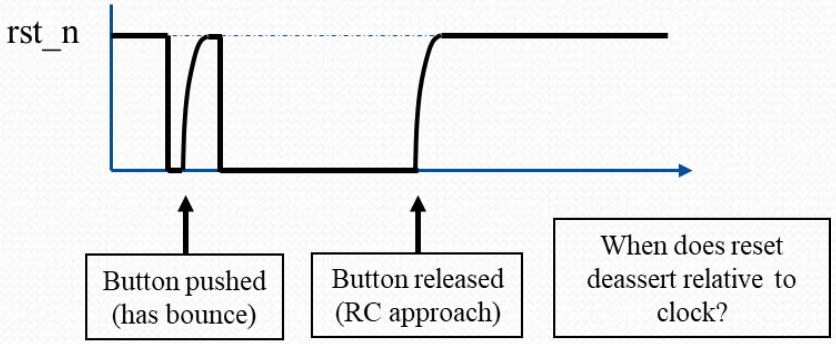
Exercise 9 (Synchronizing a PB reset):

- On the FPGA board we have a couple of push button switches. We will use one as the source for our asynchronous reset.
- It is simply a momentary push button switch to ground with a pull-up resistor.

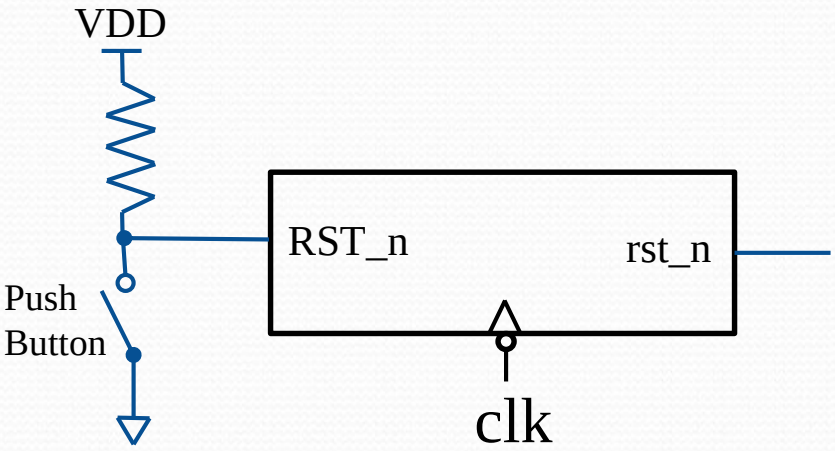
Imagine what a disaster this implementation of a global reset would be!!





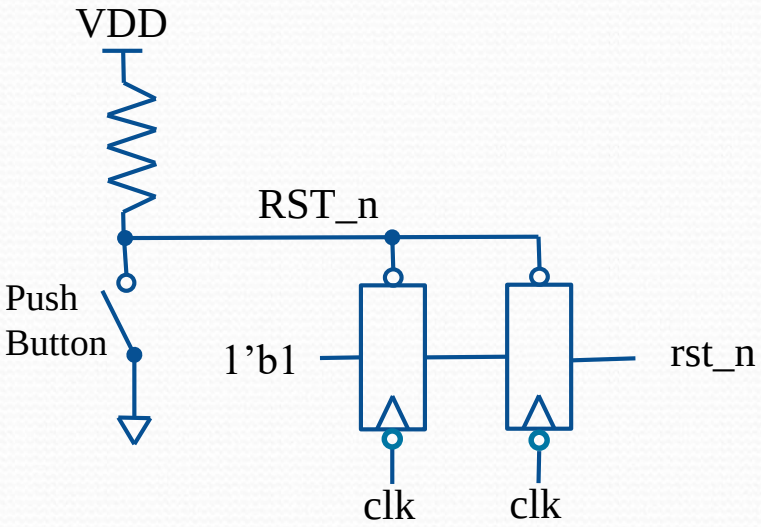
Remember…you want your reset de-asserted on the opposite edge of clock that your other flops are active on. This mean we want our reset to de-assert on negative edge of clock.

Exercise 9 (Synchronizing a PB reset):



We want to build a reset synchronizer that takes in the raw push button signal and creates a signal that is deasserted at the negative edge of clock.

It will have an interface of:
RST_n = raw input from push button
clk = clock, and we use negative edge
rst_n = our syncrhronized output which will form the global reset to the rest of our chip.

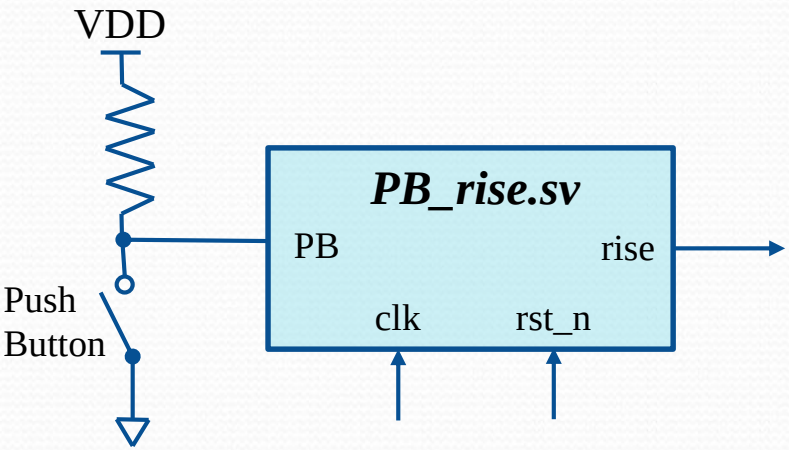


Push of the button will asynch reset the two flops. When button is released we have a double flopping (metastability reasons) to produce our global rst_n. The flops are negative edge triggered so our global reset will deassert on the opposite edge of all our other flops.

Code this reset synch unit (**rst_synch.v**)

Exercise 9 (Synching and Edge Detecting):

- We will also make use of the other push button on the board as a user interface. So we want to make a simple block to synch it (double flop it for meta-stability reasons) and detect a rising edge (release of the button).



Create a block called PB_rise.sv.

It should double flop the push button, Then flop it one more time to create a rising edge detector. Outputs a high for 1 clock when a rising edge of PB occurs.

Since the normal state of the PB input is logic 1, should the flops be reset or preset?

Submit: rst_synch.sv & PB_rise.sv to the dropbox.