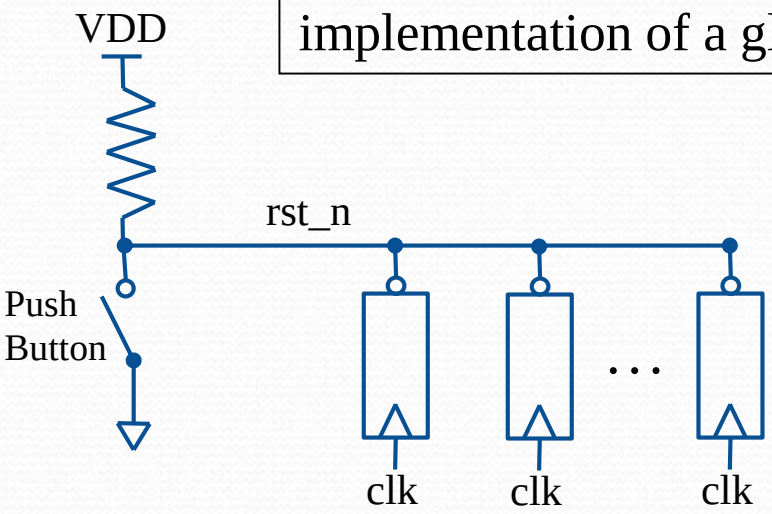
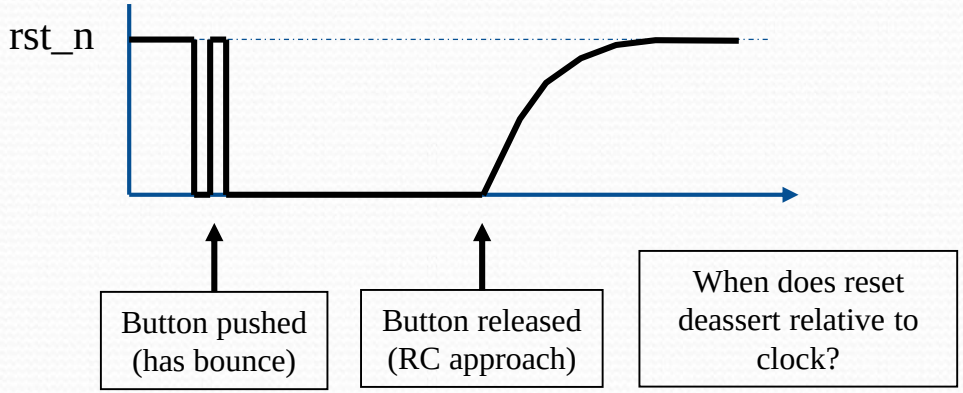
Exercise 9 (Testing ESC_interface through DE0-Nano)

- On the FPGA board we have a couple of push buttons. 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.

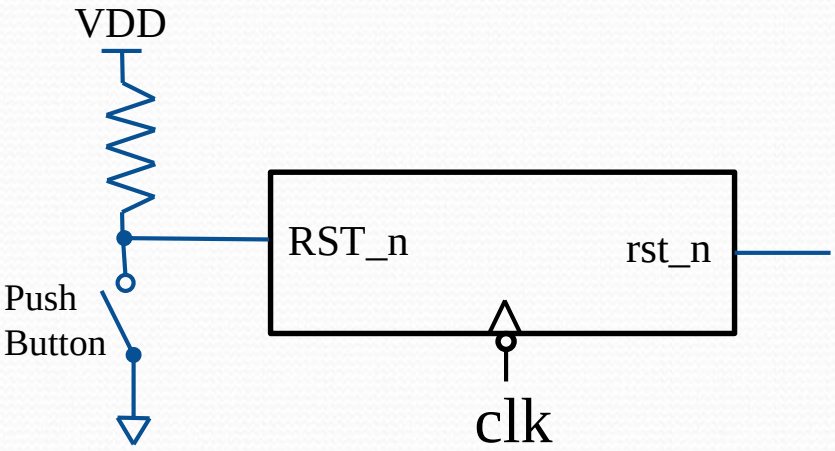
Imagine what a disaster this implementation of a global reset would be!!





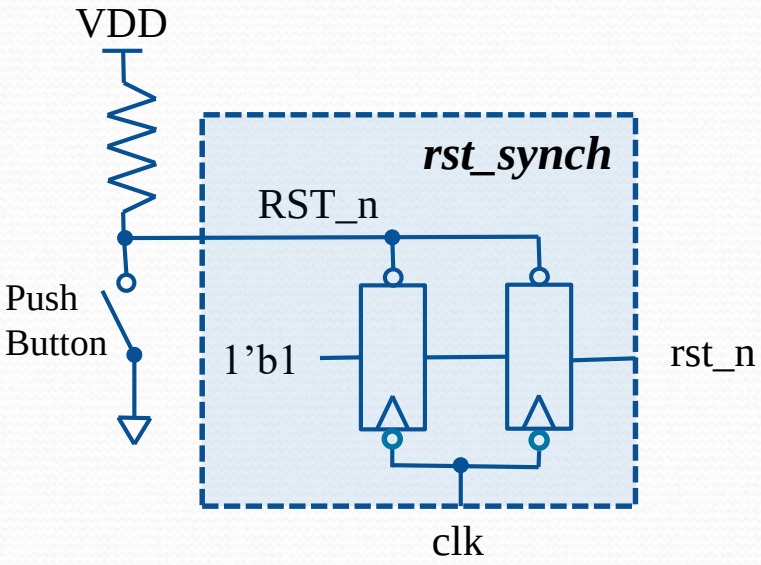
Remember…you want your reset de-asserted on the opposite edge of clock that your other flops are active on. This means we want our reset to de-assert (rise) on negative edge of clock.

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.

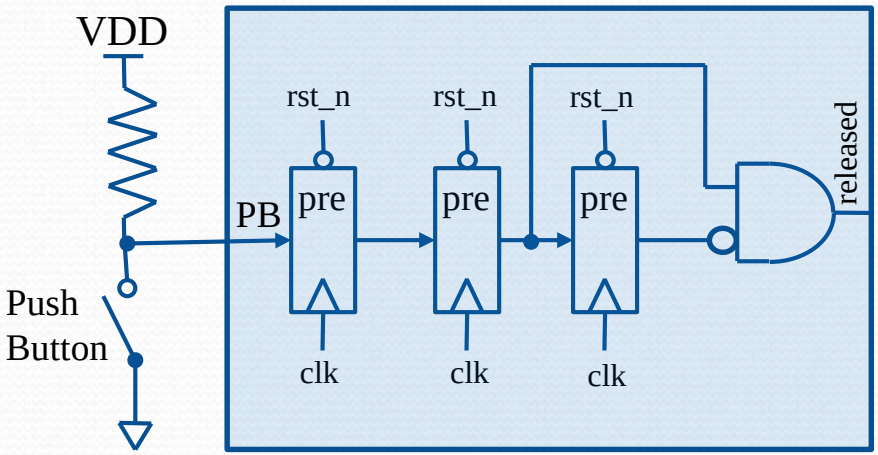
It will have an interface of:
RST_n = raw input from push button
clk = clock, and we use negative edge
rst_n = our syncrhronized output which will form the global reset to the rest of our chip.



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.sv**)

Exercise 9 (Synching & Edge Detecting a PB input):



PB_release.sv

NOTE: these flops are asynch preset not reset

Study this…does it make sense?

- The first two flops are just double flopping for meta-stability purposes. PB is asych input right?
- Third flop is used to implement a rising edge detector. If the input to 3rd flop is high, but it output is low then a rising edge must be coming through (i.e. the release of the button.)

Implement this in system Verilog. Call it: **PB_release.sv**

Don't worry about testing it yet

Exercise 9 (Testing of ESC_interface):

- In Exercise08 you coded **ESC_interface.sv** which has the interface shown below.

Signal:	Dir:	Description:
clk	in	50MHz clock
rst_n	in	Active low asynch reset
SPEED[10:0]	in	Result from flight controller telling how fast to run each motor.
OFF[9:0]	in	**Unsigned** offset added to SPEED to compensate for variation in ESC/motor pairs. Give all 4 motors the same pulse width and they don't run close to the same speed. This offset is used to compensate that to first order.
PWM	Out	Output to the ESC to control motor speed. It is effectively a PWM signal.

- Now you are going to build a test wrapper (**ESC_test.sv**) for this that will be mapped to your DE0-Nano board and used to test it (Exercise10).
- The test wrapper (**ESC_test.sv**) consists of a couple of 4-bit counters that count up when enabled.
- One 4-bit counter is connected to bits [10:7] of SPEED. Bits [6:0] being 0.
- The other 4-bit counter is connected to bits [8:5] of OFF. Bits [9] & [4:0] being 0.
- Which counter is incremented upon button push is determined by a slide switch on the DE0-Nano.

Exercise 9 (Testing of ESC_interface):