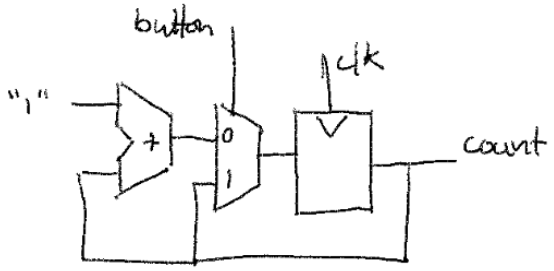
ES 4 Using switches as inputs to synchronous circuits

Counting button presses

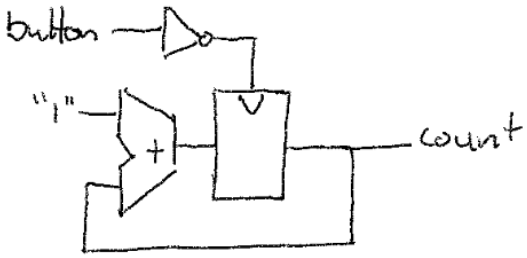
Suppose we have a counter, and we want to increment the count every time the button is pushed. Assume that the switch is connected to ground and the FPGA pullup resistor is enabled, so pushing the switch gives a logic 0.

We could make a simple counting circuit like this, where the button selects whether we keep the old value (i.e., don't increment when button pin is 1) or take the incremented value.



However, the FPGA's clock runs at 24 MHz by default, so it's almost impossible to increment by only 1 on a button press. If you push the button down for even a millisecond, the counter will count up to 24,000 in that time. You could slow the clock down, but that's likely to cause other problems if you have a large design.

Instead, we could try to build the circuit below, which uses the button input as the clock. Every time the switch is pushed (causing the pin to go from high to low), the counter clock has a rising edge and the counter increments.

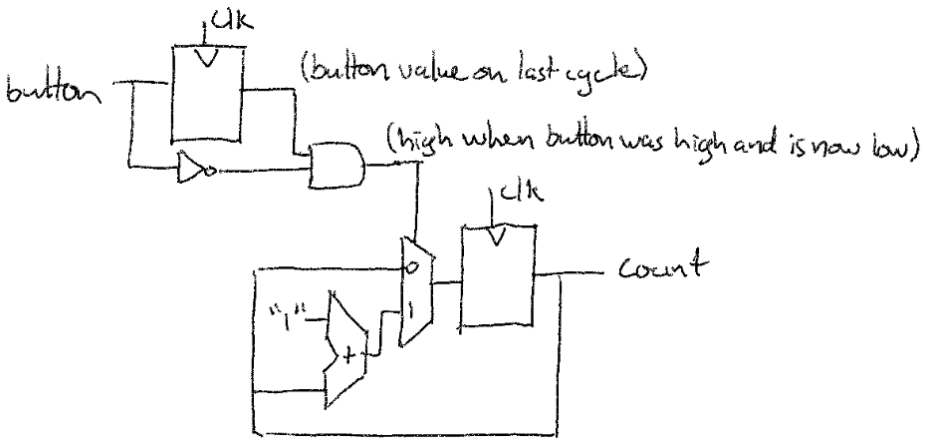


This is a perfectly reasonable thing to do in many contexts, but it will cause problems on an FPGA[1].

Because the clock signal is so common and so important, all FPGAs have internal wires (known as "clock networks") which are optimized for clock signals and are connected directly to the FPGA's flip-flops. While it's possible to inject other signals to the clock networks (i.e., use things that aren't the clock as clock inputs to flip-flops), it can only be done a few times in the whole design, and it introduces a slew of problems with synchronization.

So we need a better way: the counter clock register will still be clocked at the full rate (i.e., it shares the same clock signal as everything else), but it will only count when the counter is enabled, using a circuit like this:

[1] Probably not immediately — you'll get away with it for this lab. But if you use this technique in a more complex design, everything will come crashing down and you'll be tearing out your hair wondering why your design doesn't meet timing.



The top flip-flop holds the value of the button from the previous clock cycle. So the AND gate will be high if the button was high (not pressed) on the previous clock cycle, and now is low (pressed). This essentially detects a falling edge, but synchronizes it with the global `clk` signal.

Debouncing

Since switches operate by mechanically flipping a metal contact, there is a possibility that the contact will rebound briefly when it hits the open or closed position. It setttles and makes continuous contact within milliseconds, but a digital circuit running at tens of megahertz will easily see all of the bounces. If you don't do something to "debounce" the switch, then your circuit will register each bounce as a separate button push.

If you happen to have access to an oscilloscope, it's fun to experiment with switch bounce. Just connect one side of your switch to ground, and put a pullup resistor on the other side (either by using the pullup mode of an FPGA pin or with a discrete resistor to V_{DD}).

You can read more about the problem and solution here: https://www.fpga4fun.com/Debouncer.html The author is using Verilog, but you should be able to translate ideas into VHDL without too much difficulty. Feel free to reach out to any of the teaching staff if you have questions!