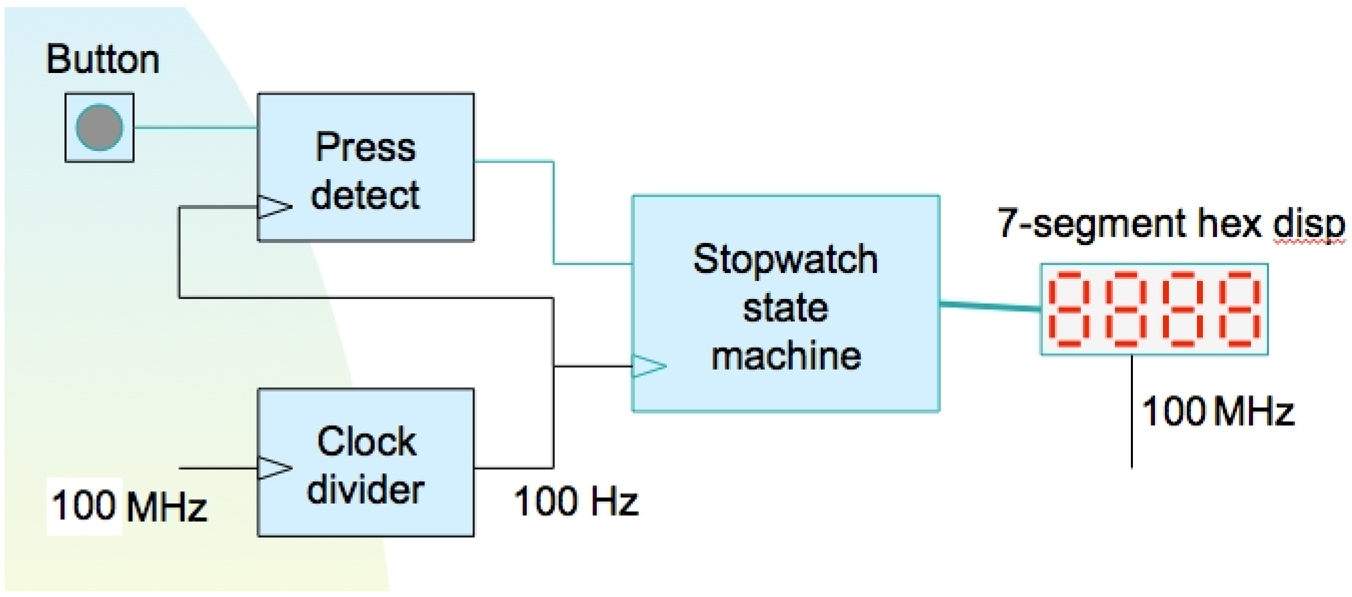
FYSIKUM

Lab 4: Digital Stopwatch



Introduction

In this lab you will design a one-button stopwatch with start, stop and reset functions.

The background for this project may be found in the Lecture 6 slides on this website.

Assignment

This is a more "advanced" assignment in this course, in which you should use your understanding of VHDL and FPGA design to produce a design that is both well-structured and works correctly.

You are free to choose your own approach. That said, a typical stopwatch design will include the following elements:

- A counter-based clock divider driven by the FPGA input clock (100 MHz) with a symmetrical 100 Hz clock output.
- A "Press Detect" state machine (typically driven by the 100 Hz clock) that tells the Stopwatch state machine that the button has been pressed.

- A Stopwatch state machine containing a 16-bit counter (ms steps) and (at least) the following states:
    1. RESET: Disable counting, counter value reset to zero.
    2. START: Enable counting, counter value increased by one every 100 Hz clock cycle (10 ms).
    3. STOP: Disable counting, counter value stays at the last (highest) count.
- A four digit, 7-segment LED display receives and displays the parallel counter value. You can download the VHDL code at this link: disp4.vhd

More about "Press Detect"

The press buttons on the Basys2 developer boards have capacitors in parallel with them, so they do not "bounce" when pressed. So no debounce logic (covered in Lecture 6) is needed. But it is important to start the stopwatch state machine at the *start* of the button press (even if the button is held for a long time).

There are multiple ways to do this, and you are encouraged to try out your own ideas. One possible approach is a state machine that reads the button value at each 100 Hz clock cycle. Such machine would need at least two states:

- WAIT_FOR_PRESS: As long as the button is not pressed ('0'), stay in this state and assert output '0'. When a button press is detected ('1') assert output '1' and change the state to WAIT_FOR_RELEASE
- WAIT_FOR_RELEASE: Assert output '0', and wait for the button to be released. When the button value is '0' again, transition back to WAIT_FOR_PRESS

Notice that this approach produces a button_press signal that lasts for exactly one 100 Hz clock cycle (10 ms).

Testing your design

After simulating the design, implement it and download it to the Basys3 developer board. Print out the code for your final design and hand it in to the instructor.