CS M152A Lab 3

Stopwatch

In this lab, you will be designing a stopwatch on the FPGA.

Introduction

In this assignment, you will go through the complete FPGA design flow by designing a stopwatch circuit and implementing it on the Nexys™3 Spartan-6 FPGA Board. The inputs of the stopwatch are buttons and slider switches. By implementing it you will review the design techniques you learned from Lab 1. The digits of the stopwatch are displayed through the on-board seven segment display. To configure it you will need to study and explore the reference manuals, which is also an essential step in more realistic and complex digital design tasks.

Pre Lab

Sketch your initial high level design for the Stopwatch. A hand drawing is perfectly fine, but you are welcome to use a program like Logism to facilitate this process.

Functionality

The stopwatch will start as a basic clock which counts minutes and seconds. The left two digits of the seven-segment displays will count minutes, and the two on the right will count seconds. For example, without touching anything, after 1 minute and 43 seconds, the stopwatch should display: "0143". The time stored in the stopwatch should be adjustable by the use of two inputs: SEL and ADJ. Both of these inputs are switches on the FPGA.

- **SEL** is a select switch which will choose minutes or seconds in the adjusting mode.

SEL	Selected
0	Minutes
1	Seconds

- When the **ADJ** input is set to logic high, the clock is in adjustment mode. During this mode, normal increments of the clock is halted. Instead, the selected portion of the clock increments at a rate of 2 ticks per second (2Hz) to allow the user to set a new clock time. The portion of the clock that's incremented is selected by the SEL input. The unselected portion of the clock is effective frozen while in adjustment mode.

ADJ	Action
0	Stopwatch behaves normally
1	Stopwatch stops and *'Selected'* increases at 2Hz

In addition to the two switches, your circuit will use two push buttons (on the FPGA) which will control the timer behavior.

- **RESET** will force all of your counters to the initial state 00:00
- **PAUSE** will pause the counter when the button is pressed, and continue the counter if it is pressed again

Implementation

Counter

The basic building block of the stop watch is the decade (modulo 10) counter. The seconds counter will increment every second, and every 60 seconds, it will reach the maximum value of 59 and enable the minutes counter, which will increment on the next rising edge of the clock.

Clock

For this assignment, you will be using four different clocks – a 2 Hz clock, a 1 Hz clock, a much faster clock (50 – 700 Hz), and a clock for blinking in the adjust mode (>1Hz). We recommend that you create a Clock Module that takes the 100 MHz master clock (internally connected to pin V10 of the FPGA board) as input and outputs 4 different clock signals.

The main purpose of the third, faster clock is to assist in the multiplexing of the seven-segment display. The display is designed in such a way that all four seven-segment displays will show the same digit if no additional circuitry it used. For this reason, you will need to cycle through the four digits using a much faster clock so the human eye sees four digits. The cycling frequency should be between 50-700 Hz; we'll leave it to you to find a reasonable value.

In addition to the function mentioned above, the faster clock can also serve as the under sampling clock that filter noises in the debouncer circuit.

The clock for blinking in the adjust mode needs to blink at least 1Hz. It is up to you to choose a blinking rate that looks good. Note that this rate cannot be 2Hz because we will not see the minutes or seconds increment correctly while adjusting.

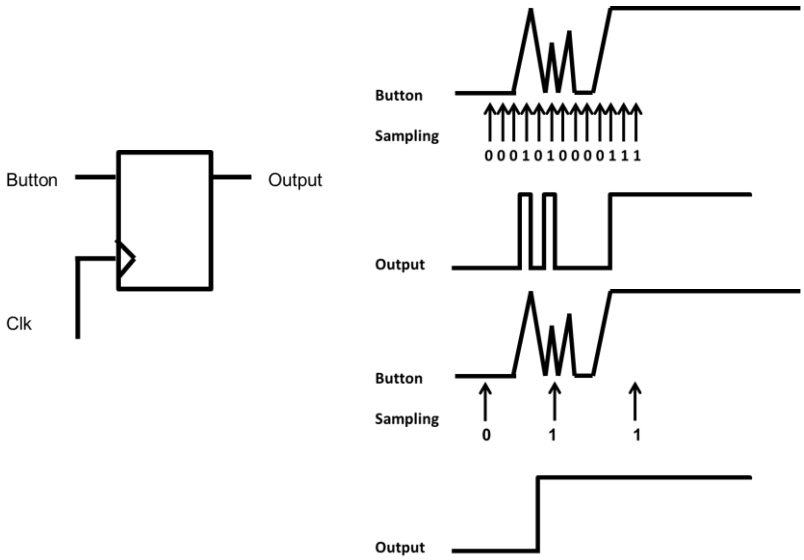
Seven-Segment Display

Introduction to the seven-segment display on the Nexys3 board can be found in the Nexys3 Reference Manual on the course website. Alternatively, the reference manual can be found on the

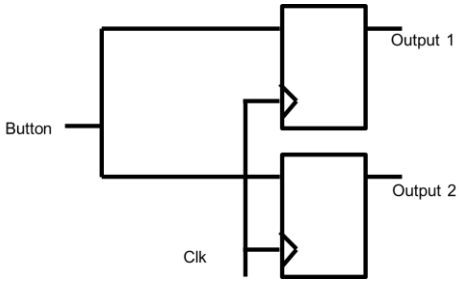
official Digilent website. You are required to read the manual to understand how it operates and how you implement the design.

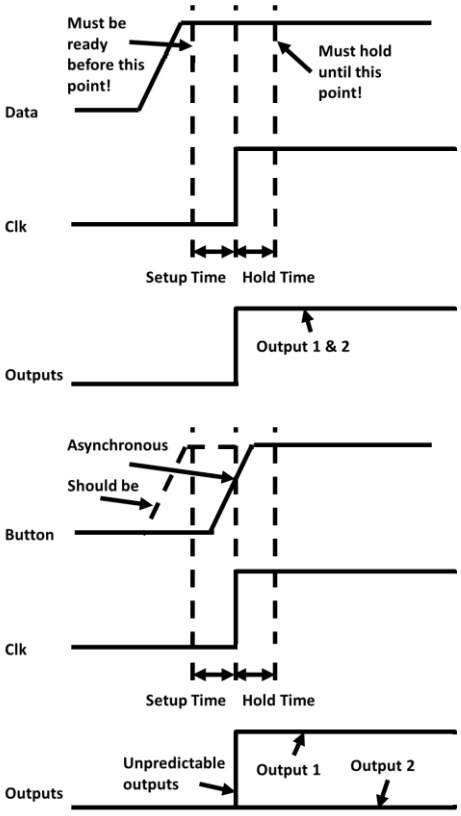
Debouncers

In this lab, you must make both buttons and switches on the board useful. However, due to physical contact instability, the buttons and switches are very bouncy: it means that when you press the button once, it generates an oscillation of signals due to poor metal contact, and thus causing considerable **noise** in the input. A simple solution to filter out the noise is sample at a frequency lower than that of the noise to. In that case, the glitches, which may otherwise be considered by the system as a valid button push, will be filtered out (see the figure below).



Another problem, **metastabilty**, exists because of the asynchronous nature of the button and slider switch input. To ensure reliable operation in digital circuits, the input to a register must be stable for a minimum time before the clock edge (register setup time or t_{SU}) and for a minimum time after the clock edge (register hold time or t_H), which is the signal timing requirements in its simplest form. In synchronous systems, the input signals must always meet the register timing requirements. Yet in the case of the buttons and slider switches, the signal may come at any time, causing unpredictable (and possibly different) outputs to the other modules connected to the signal. A simple solution to the problem is to use a flip-flop after the asynchronous input, and regard the output of the flip-flop as the input. With that, the outputs to all other modules will be consistent.





You can find example implementation from the project in Lab 1. What is the frequency used for sampling button input in the example project?

User Constraint Files (UCF)

Implementation constraints are instructions given to the FPGA implementation tools to direct the mapping, placement, timing or other guidelines for the implementation tools to follow while processing an FPGA design. Implementation constraints are generally placed in the UCF file, but may exist in the HDL code, or in a synthesis constraints file. Examples of implementation constraints are LOC (placement) constraints and PERIOD (timing) constraints.

With the help of example UCF file in Lab 1 (nexys3.ucf), you should be able to create a UCF file for this project. You can refer to the nexys3 reference manual for further information on the "LOC" constraints.

Project Submission and Demo

When you finish, you should demo your design to the TA, and explain your design to him/her. The following should also be submitted for this lab:

1. Project Code: the Xilinx ISE project folder should be cleaned up (*Project > Cleanup Project Files*), zipped and uploaded in the corresponding assignment page on the course website

2. Lab Report (Electronic Version): the lab report should be uploaded in the corresponding assignment page
3. Lab Report (Paper Version): the paper version of the lab report should be printed out on both sides and handed in on assigned date