EECE 444 Microprocessor System Design

LAB 3: Parking Meter

Introduction:

The objective of this lab is to design (code, simulate and implement) a parking meter like the ones found around Chico State. The meter should be able to simulate coins being added and show the appropriate time remaining. Also, it should flash slowly when less than 200 seconds are remaining and flash quickly when time has expired.

Description:

You will design a finite state machine that will simulate the operation of a parking meter. The buttons on the board will represent different coin denominations and the seven segment LED display will output the total amount of seconds remaining before the meter expires.

Configuration:

Button 0 add 50 seconds

Button 1 add 150 seconds

Button 2 add 250 seconds

Button 3 add 500 seconds

As soon as a button is pushed, the time should be added immediately. When less than 10 seconds remains, 2 tri-color LEDs should flash RED with period 2 seconds and duty cycle 50% (ON for 1 sec and OFF for 1 sec) to indicate time is about to expire. When time has expired, the tri-color LEDs should flash RED with period 1 second and duty cycle 50% (ON for 0.5 sec and OFF for 0.5 sec). For example when the board starts, it should be in the 0 time remaining state and the LEDs are flashing at a 0.5 second rate. If button 2 is pushed, the display should read 250 seconds and begin counting down and the tri-color LEDs should flash GREEN with period 2 second and duty cycle 50%. When the time is at 150, and button 1 is pressed, the display should read 300 (150+150). As long as the remaining time is >10, the tri-color LEDs should continue to flash GREEN.

The max value of time will be 9999 and any attempt to increment beyond 9999, should result in the counter defaulting to 9999 and counting down from there.

Your circuit will consist of three parts:

1. Input module, which takes the input from the buttons on the board and outputs a de-bounced single pulse signal. You should implement a de-bouncing circuit to make this module work. Input to this module should be the 4 push buttons input from your board; output should be the de-bounced signal. You will also need to implement a single pulse state machine, to make sure every pulse resulted on one and only one incrementing of the counter (refer to the lecture notes for description of the de-bouncing and single pulse circuits).
2. Output module, which displays the output on the 7-segment displays and the tri-color LEDS. Input to this module should be the 16- bit BCD time to display and any other control signals. Read the board manual and understand how to correctly drive the multi-digit 7 –segment display and the tri-color LEDs.
3. The controller and data path: this module will consist of the control unit and the data path. Inputs to the control unit should be the de-bounced input signal, clk and any reset signals. Output should be all the control signals that will control the data path. In the controller module, you will need to add the time count whenever a button is pushed and subtract the time count every second afterwards. You can design in a way such that you use all BCD operations (BCD addition and subtraction), see figure 1. You can also design to keep all your counts in binary and then convert to BCD using the shift&add-3 algorithm. However, please not that you cannot divide by 10 using the Artix7 FPGA hardware, so DONOT use any binary to BCD conversion that rely on dividing by 10.

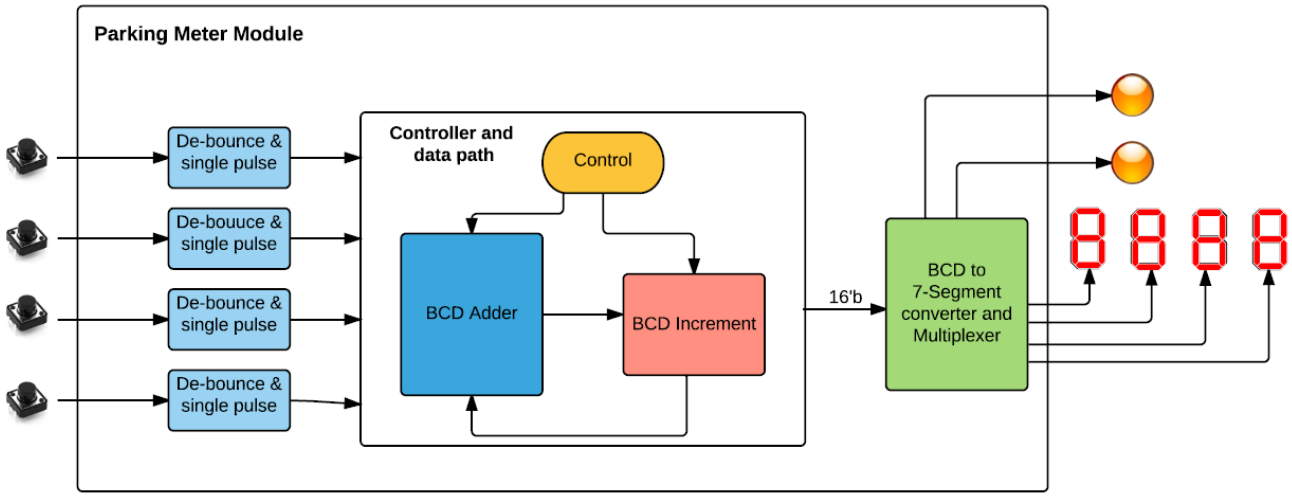

Figure 1

Important Notes:

1. Please design, test and simulate each module separately before you create the top module. Start by designing the BCD adder and Decrementer and test each module separately using different inputs. This procedure will help you debug any problems you may face while implementing the top module.
2. Make sure to go over the Nexys 4DDR manual to understand how to multiplex the 7-segment displays.
3. Check for the overflow condition (when the counter reaches 9999) in you code and makes sure it works.
4. Your design may be simpler if you make the BCD adder and Decrementer as a single always block, however it is up to you to make them as two separate always blocks.
5. Make sure to simulate your design before you implement it.
6. If your design doesn't work on the board, submit the simulation results and the output schematics.
7. Make sure your design doesn't result in any extra latches.

What to Turn In?

1. Please indicate how many hours you spent working on this lab.
2. The controller FSM with all inputs and outputs labeled.
3. All Verilog code
4. Your UCF file.
5. A snapshot of you working meter or your simulation results.
6. Please indicate how many hours you spend to complete this lab.