Lab Assignment #4

Guideline

This lab can be done with a partner.

Objective

Your objective in this lab is to design (code, simulate and implement) a parking meter much like the ones around Austin. It should be able to simulate coins being added and show the appropriate time remaining. Also, it should flash slowly when less than 185 or 200 seconds (depending on configuration) are remaining and flash quickly when time has expired.

Description

You will design a finite state machine that will simulate the operation of a traffic 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 U	Add 50 seconds
Button L	Add 150 seconds
Button R	Add 200 seconds
Button D	Add 500 seconds
Switch 0	Reset time to 10 seconds
Switch 1	Reset time to 205 seconds

As soon as a button is pushed, the time should be added immediately. When less than 200 seconds remain, the display should flash with period 2 seconds and duty cycle 50% (on for 1 sec and off for 1 sec; so you will see alternate counts on the display like 200,blank,198,blank,196,…). Make sure you blink such that even values show up and odd values are blanked out. When time has expired, the display should flash with period 1 sec 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 be flashing 0000 at 1 Hz. If button D is then pushed, the display should read 500 seconds and begin counting down. When the time counts down to 180 seconds and button R is pushed, the display should then read 380 seconds (200 + 180). If switch 0 goes high, then the time should change to 10 seconds and stay at 10 seconds until switch 0 goes low again.

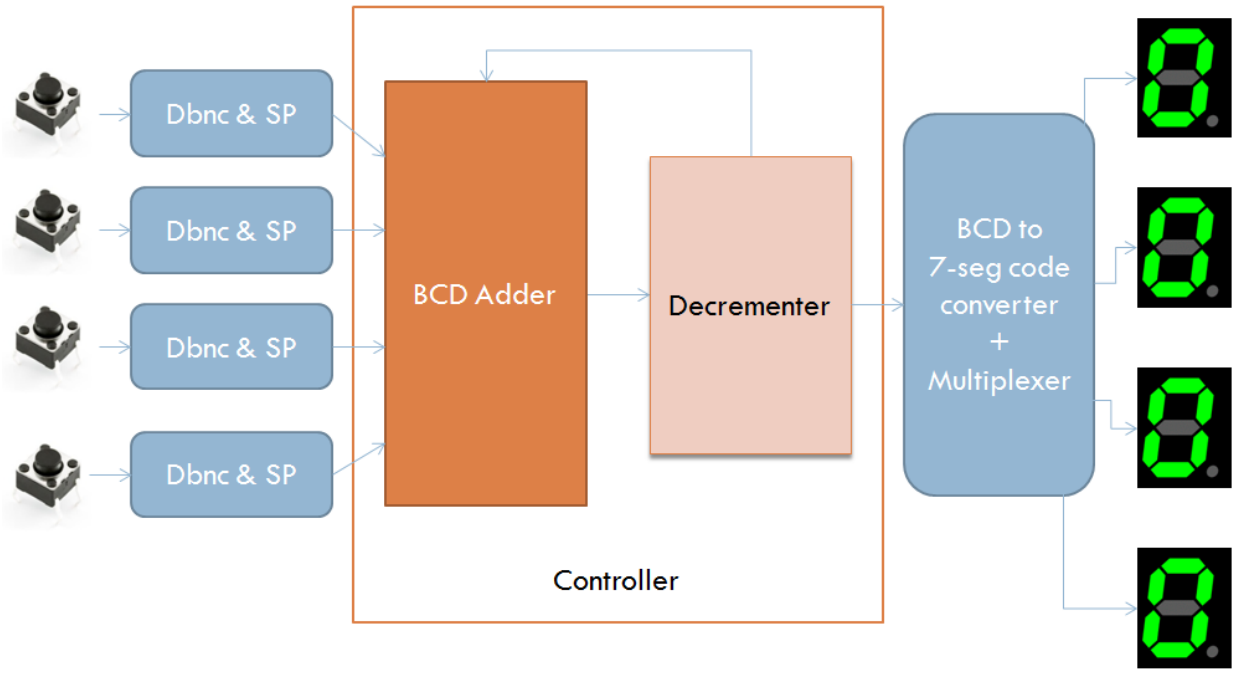
The max value of time will be 9999 and any attempt to increment beyond 9999, should result in the counter defaulting to 9999 and counting down from there.

Your circuit will consist of three parts:

- The input module, which takes the input from the buttons and switches on the board,
- The output module, which displays the output on the 7-segment display, and
- The controller

Although it is not mandatory to follow this structural hierarchy, it is recommended that you implement the input and output parts in separate modules and make sure they are working correctly before putting the whole design together. You need to implement a de-bouncing circuit to make the input module work. The best way to this is to read, understand, and then implement the de-bouncing circuitry described in

the textbook. For the output module you need to read the board manual and understand how to correctly drive the multi-digit 7-segment display. In the controller module, you will need to add the time count whenever a button is pushed and subtract the time count every second. You can design in a way such that you use all BCD operations (by having BCD addition and subtraction like the one shown in the following figure). However, you can also keep your counts in binary and then convert binary numbers to multiple digit BCD numbers before you send them to the output module.



Useful Information

1. Debouncer and Single Pulser circuitry is explained in section 4.7 of the text.
2. The button noise on the boards last somewhere between 1 to 50 milliseconds.
3. BCD Adder is described in section 4.2 of the text. If you don't want to use a BCD adder, you can use an approach similar to problem 4.13 (in the text) for binary to bcd conversion
4. BCD to 7-segment decoder is described in section 4.1 of the text. However, note that the polarities of signals (anodes and cathodes) are not the same as the ones in the text. Please refer to the board's manual for proper polarities.
5. Make sure you go through the Basys3 board manual to understand how to multiplex the 7-segment displays.
6. Check for the overflow condition (saturation at 9999) in your code and make sure it works.
7. It is recommended that you simulate the design using either a test bench or by using the force & run commands from the transcript window.
8. In case your design does not work on the board, submit the testbench Verilog file and/or the .do file, and show the simulation during checkout for partial credit. Make sure that the simulation is extensive, i.e. covers the expiration state, the saturation state, tests debouncer/single pulse, tests decrementer, etc...
9. Ensure that there are no inferred latches in your design when you implement it.

Submission Details

All parts of this lab are to be submitted on Canvas. No hard-copy submission is needed.

1. All Verilog code
2. Bit file and XDC file

Checkout Details

During checkout you will demonstrate the parking meter working on the board. Also, you will be judged on how well you understand your code and other concepts like de-bouncing, multiplexing 7-segments and BCD addition.