**ECE 4250/ 7250: VHDL and Programmable Logic Devices  
Laboratory**

**Lab #7  
Lab Title: BCD Converter**

**Group #2  
Group Names: Chris Smith, Benjarit Hotrabhavananda**

**Teaching Assistant Use Only:**

**Points Earned Reasons for Deduction**

**Pre-lab:**

**Post Lab report:**

**Demonstration:**

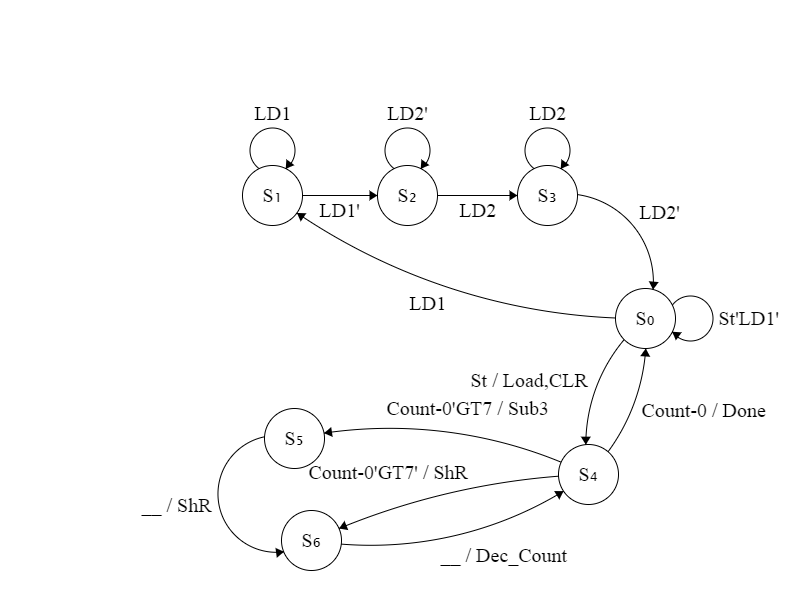
**Final Lab Grade:**

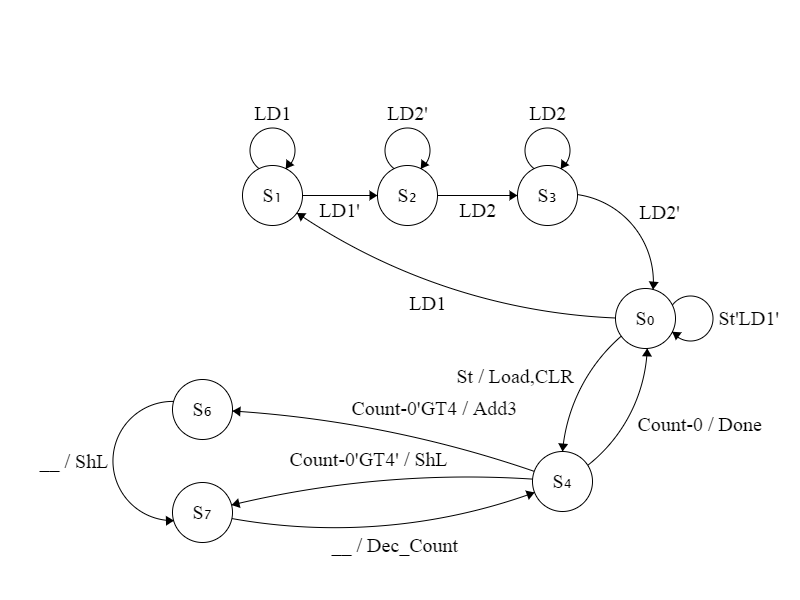
**Comments to students:**

**I. Objective**

The objective for this lab is to use the Data Path Technique in designing and implementing BCD Converters. We have to convert BCD to binary and also from binary to BCD. Also this lab makes us familiar with the process of creating a project, synthesizing, implementing and downloading a simple design to the *Xilinx Spartan-3 FPGA* board by using the *Xilinx Project* *Navigator*.

**II. Implementation:**



**Figure1: BCD to Binary state diagram**

**Figure2: Binary to BCD state diagram**

They consist of 6 states(0 to 5), the first three states will load the first 8 bits into input register as low bits which is a signal, and the other 8 bits as high and check if the start button is push to go on to the state 4. State 4 will check if the input is greater than 7 or not and counter if it is 16, when it is, it goes back to state 0. When input is more than 7, we subtract 3 to it, and go to state 5. State 5 is where shifting happens. State goes back and forth between 4 and 5 until counter is 16. The binary to BCD conversion proceeded in a very similar manner, however the initial loaded values are shifted to the left instead of to the right (bit by bit goes from the binary input into the BCD output. When a particular “block” of the BCD output which is each 4 bits of the output is greater than four, 3 is added. We were able to implement very similar code, with the only differences being the shift method, and adding three instead of subtracting.

As the TA taught us; there are three steps to the operation of converting BCD to binary

* **Shift**
* **Check if greater than or equal to 8**
* **Subtract if greater (from each particular “block”)**

The binary to BCD conversion involves similar steps

* **Shift**
* **Check if greater than or equal to four**
* **Add if greater (to each particluar “block”)**

**III. Conclusion:**

We have implemented a BCD converter which takes in either two binary numbers representing a BCD or Hex number. We have also used subtract and shift method using a state machine by the modelsim program; then after being sure from its correctness; we loaded it to the FPGA board, getting the files required to input the numbers and display the output. At first, we faced some problems in both converting binary to bcd and bcd to binary. The problem was it did not give the correct result. One of the problems that we solve was we instead of using component subtractor we just add the 2’s complement of the number 3 directly to the particular column, or adding three in the case of binary to BCD. The other problem was when we synthesize it, we could not load the input and output. We were finally able to implement our design to the board by assigning our output once the done condition has been set. Assigning the signal as a concurrent process caused the outputs to never be set on the FPGA board.

**IV. Code:**

**BCDtoBinary.vhd**

**BinarytoBCD.vhd**

**LEDDisplay.vhd**

**dec\_7seg.vhd**

**AnodeControl.vhd**

**lab7bench.ucf**

**lab7bench \_bcd2bin.vhd**

**lab7bench\_bin2bcd.vhd**