Cpr E 281 FINAL PROJECT

ELECTRICAL AND COMPUTER ENGINEERING
IOWA STATE UNIVERSITY

FINAL Project

1.0 Objectives

The main objective of this lab is to gain experience in putting it all together and programming the Altera DE2 board to carry out an independent activity. We will design a circuit to realize one digital machine that will accept input from switches and keyboard and output to LEDs and 7-segment displays.

2.0 Setup

Begin by creating a new folder **/home/*username*/CPRE281/Proj** to save your work.

3.0 Selection of Machines

You are free to choose any one of the following machines or a machine of your choices (must include a state machine) to implement for your final project.

In all cases, come prepared with your designs and test plans. Demonstrate your project to your TAs and ask general questions during the last two labs of the semester.

1. Register File: A register file design was described during the lecture. You are going to implement a 4-bit wide 16-register file. Use Sw3-Sw0 (MSB to LSB) for the first register read address (RA1), Sw7-Sw4 (MSB-LSB) for the second register read address (RA2), Sw11-Sw8 (MSB-LSB) for the write register address (WA), Sw15-Sw11 (MSB-LSB) for the write data input value (LD_DATA), and Sw16 as WR (write enable signal). You will use Sw17 for clock input. When switch 17 is OFF, the clock signal is zero, and when it is ON, the clock signal is one. A positive (negative) edge occurs when switch in moved from OFF (ON) to ON (OFF) position. If the switch bouncing is a problem (that is an off-on-off sequence generates more than one clock) then use a de-bouncing circuit as described in Appendix. You will display the following items:

   DATA1 output: on one 7-segment display of your choice
   DATA2 Output: on one 7-segment display of your choice
   Current State (see below): on one 7-segment display of your choice

   You should design a state machine with eight states to carry out the following operation. With each clock signal (off-on-off transition of Sw17), the machine should advance to the next state. Before you generate each clock signal, you should manually set in switches as follows. Initially set RA1=H'0, RA2=H'1, WA=H'02 LD_DATA=H'F, and WR=1. RA1, RA2, and WA should be incremented by 1 after each clock (you change the switch position by hand), and LD_DATA should be decremented by 1 after each state. WR should be a logical 1 for two cycles and 0 for the next cycle and then repeat this sequence.

Design your register file, work out your test plan and expected results, and show to your TA that your circuit is functioning as planned for a few cycles.

2. A Serial Shift Adder: You will design a serial-shift-add circuit. The circuit includes three parallel access shift registers (shift registers that can be loaded with new values when LD signal is 1). You will use two shift registers (Call them A and B) to store two 4-bit numbers that can be specified in switches. One shift register (Call it C) will hold the result. One-bit adder FSM (Moore) will perform the addition. You will use the switches as follows. SW3-SW0 will be used to specify the value to be loaded. SW4, SW5, and SW6 will control loading of register A, B, and C, respectively. You will design a state machine with eight states to manage the operation of the machine. SW7 will control the advancement of the control state machine (if it is 1, then the state machine will advance to the next state, otherwise the circuit will remain in the same state). The following operations are carried out.

   When the machine is in state 0 and SW4=on (or state 1 and SW5=on, or state 2 and SW 6=on), register A (or correspondingly register B or C) is loaded with the new value in SW3-SW0. If State = 3, 4, 5, or 6, then the machine adds one bit and then shifts inputs and output registers by one position. The machine does not do anything in state 7.

   Design your adder, work out your test plan with random but different inputs to each register and demonstrate the functionality to your TA for a few cycles. Include cases when the machine advances or doesn't.

3. Design a vending machine that accepts a nickel, dime, or a quarter (represented by SW1, SW0 = 01, 10, or 11, respectively). Thus you can only input one coin at a time. It also does not accept more than 35 cents. In real operation, the coin will be simply released. It can disburse two products, one costing 15 cents (represented by SW2) and the other costing 20 cents (represented by SW3). If both SW2 and SW3 are ON then the machine releases the change. No coin is accepted if a product request is set. You should display the amount with the machine using two seven segment displays. You should also display the next coin being accepted as a two digit number (00, 05, 10, or 25) on two 7-segment displays using a simpler decoding. The next product being disbursed can be displayed on another 7-segment (none – 0, first – 1, second – 2, and change 3) when state advances to the next state.
4. Design a sorting machine. The idea is to design a two port read two-port write register file with k registers. The data are stored in registers using some input switches (address and data are specified by switches). Then there are two

counters, C1 and C2. A four state machine sorts the numbers as follows.

Load the registers with initial values. Start the machine in state S0.

State S0: C1 is initialized to 0. Go to state S1.
State S1: C2 is initialized to C1 +1. (Need add 1 circuit). Go to State S2.
State S2: Read two registers from two addresses specified by C1 and C2.
Call them D1 and D2.
Feed D1 and D2 into a maximizer/minimizer circuit.
It yields MAX and MIN on two ports.
At clock edge
MAX is written into register C1 and MIN is written into register C2.
(This swaps max and min).
If C1 = k-2 then Go to state S3
Else if C2 = k-1 then increment C1 and Go to State S1
Else increment C2 and Go to State S2
State S3: Done. At this point a register selected by an address in switches is displayed in a 7-segment

4.0 Complete

You are done with this lab. Ensure that all lab files are closed, exit Quartus II, log off the computer, and ensure that your TA has signed the demo sheet. **Don't forget to write down your name, student ID, and your lab section number**.