###### *CSE 260 – Digital Computers: Organization and Logical Design Jon Turner*

Lab 2 Report

##### *Your name 2/18/2013*

***Part A*** (10 points). Draw a state transition diagram for your pattern matcher. Label the states with symbolic names. Label the transitions clearly with the conditions that trigger the transitions and with any actions that should be taken when the transition occurs. You may omit “restart” transitions. Note that this is a fairly complicated diagram. Think carefully about each state and each transition. Organize the diagram to make it easy to follow. Draw it neatly and legibly. You are strongly urged to do this part before you write any VHDL code.

*paste your diagram here*

***Part B***. (20 points) Paste the VHDL for your pattern matcher below. Make sure that your VHDL is consistent with your state diagram. Note that the next paragraph is formatted using the ”code style” which uses a fixed-width font and has appropriately spaced tabs. Please always use this paragraph style for your VHDL code. Also, be sure to format your code so that lines do not wrap-around in the lab report. Points will be deducted for code that is badly formatted or difficult to read.

paste your code here

***Part C.*** (10 points). You will find an incomplete *testPatMatch* testbench in your repository. Add tests to the testbench following the instructions in the comments. Make sure that your tests cover *every transition in your state transition diagram*. Paste the code for your tests below.

paste your code here

***Part D***. (10 points) Run the simulator using the *testPatMatch* testbench and paste a screenshot of the waveform window below. Set the radix to decimal for the *patCount* signal. If your screenshot extends over a longer time period than 400 ns, split it into two screenshots and paste them one after the other. Find the first two places in the simulation where a pattern is matched and paste a screenshot showing the portion of the simulation output where the matches occur. Make sure that all text is clearly visible and legible on the printed copy.

*paste screenshot here*

At what time is the last symbol of the pattern matched? In each case, what pattern is matched and what time is the last symbol in the pattern matched. Is the *patCount* output incremented at the appropriate time?

Find two places in the simulation where a pattern match is started, but fails before it completes. Paste a screenshot showing this portion of the simulation output below.

*paste screenshot here*

At what times are the failed matches detected? What state is the circuit in at the time the failed match is detected? What is the input symbol that causes the match to fail?

***Part E.*** (10 points) In your repository, you will find a *binaryInMod,* an output module called *outMod* and a partial implementation of a *top* module. You are to complete the *top* module so that it connects the *patternMatcher* to an instance of *binaryInMod* and an instance of *outMod*. The *restart* input of *patternMatch* should be connected to output *pulse(1)* of the *binaryInMod* circuit, *valid* should be connected to *pulse(2)*, *inSym* should be connected to *inBits(3..0)* and *repCount* should be connected to *inBits(11..8)*. The *outMod*‘s *valid* and *inSym* inputs should be connected to the corresponding inputs of *patternMatcher* and its *patCount* input should be connected to the *patCount* output of *patternMatcher*. You should also connect the *inSym* and *repCount* inputs of *patternMatcher* to *led(3..0) and led(7..4)* respectively. The *outMod* circuit displays the current *inSym* value as an ASCII character at the right end of the top row of the display. Recently input symbols appear to the left of the current symbol. The value of *patCount* is displayed in hex in the bottom row of the display. Paste a copy of your top circuit below.

paste your code here

***Part F***. (15 points) Simulate your completed circuit using the provided *testTop* testbench. Remove the default signals that are supplied by the simulator and include the following instead, using the indicated radix settingd.

* the *clk*, *btn* (binary) and *knob* (binary) inputs to *top*, along with the *led* (hex) outputs
* the *resetOut*, *pulse* (hex) and *inbits* (hex) outputs from *binaryInMod*
* all signals from *patternMatcher*, including the state signal; use hex for the *repCount* and *inSym* inputs and use the unsigned integer radix for the *patCount* output
* the *top* (ASCII) and *bot* (unsigned integer) signals from *outMod*

Organize the signals in the waveform window in appropriate groups, with dividers labeling the different groups. Make sure that the *operationMode* constant in *commonDefs.vhd* is set to 0 for this part. Paste a screenshot showing the first 20 microseconds of the simulation below.

*paste screenshot here*

How many patterns are matched during this part of the simulation run? How do you know?

Paste a screenshot below showing the portion of the simulation from time 60 microseconds to 80 microseconds. Make sure that the final values of *top* and *bot* are clearly visible.

*paste screenshot here*

Based on the simulation output, what would you expect to see on the LCD display of the prototype board, if the same test data was input to the prototype board? What does the displayed information tell you about the test data?

Paste a screenshot below, showing the simulation output from time 7.5 microseconds to 10.5 microseconds.

*paste screenshot here*

What pattern is matched during this time period? How can you tell? What is the value on the LEDs at time 8.5 microseconds? What is the significance of this value?

Paste a screenshot below, showing the portion of the simulation run from time 53 microseconds to 54 microseconds.

*paste screenshot here*

Observe that the valid input to *patternMatcher* goes high at 53.2 microseconds. Explain how changes to the inputs of the *top* module cause valid to go high. To answer this question, you may need to study the code for *binaryInMod*.

Now, observe the changes to the knob input to *top*. Explain how these changes affect the inputs to *patternMatcher.*

***Part G***. (10 points) Proceed to this part only after you have successfully completed part *F* and have carefully checked the simulation results. Add the file *protoBoard.ucf* to your project (by selecting *Add Source* from the *Project* menu). Generate a bit file for your completed circuit. Make sure that the *operationMode* constant in *commonDefs.vhd* is set to 1. Then download the bit file to a prototype board using the Impact tool. The prototype boards are kept in a locker in Bryan Hall, Room 316 and can be signed out as needed. Stuart Cranor (who can usually be found in 316 or 308) can sign out a board for you to use. If he is not available, any of the TAs can also sign out a board for you. Boards should normally be kept for no more than two hours at a time, but you may check them out overnight. In this case, you should return them first thing the next morning (by 9:00). Note that you are responsible for these boards when they are checked out to you. They cost about $150 each, so please handle them with care.

You may do your testing using any of the PCs in the Bryan 316 lab or in any of the Urbauer labs. They all have the necessary software. However, if there is a class that is using the Bryan 316 lab, please do your testing in Urbauer. Once you have your circuit loaded onto the board and you have convinced yourself that it works correctly, fill in your name below on the printed copy and have one of the TAs check it and sign their name below, after assigning the appropriate number of demo points..

Student name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ has successfully demonstrated the *patternMatcher* circuit on the prototype board.

TA name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TA signature:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Demo points (out of 10):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Comments (if the circuit does not work 100% correctly, make a note of all issues below):