###### *CSE 260 – Introduction to Digital Logic and Computer 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) Run the simulator using the provided *testPatMatch* testbench and paste a screenshot of the waveform window below. Set the radix to decimal for the *patCount* signal. Be sure to include your state signal in the waveform display, as well as any other registers that are defined by your *patMatch* code. Paste a screenshot showing the portion of the simulation from 100 ns to 800 ns. Make sure that all text is clearly visible and legible on the printed copy.

*paste your screenshot here*

How many patterns are matched during this period? Exactly what patterns are matched?

Paste a screenshot below showing the portion of the simulation from 1000 ns to 1800 ns.

*paste your screenshot here*

How many patterns are matched during this period? How many times is a pattern match started but then the match fails part way through? At what times are failed matches detected? What are the partial patterns, including the first symbol that doesn’t match?

***Part D.*** (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 E.*** (10 points). You will find an incomplete *testTop* 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*. Include a 50 microsecond delay in your testbench, after the last test input. Paste the code for your tests below.

paste your tests here

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

* 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 a portion of the simulation where at least three patterns are matched. Make sure that the *patternMatcher* signals are all clearly readable. If you need to, split the screenshot into two parts to improve legibility.

*paste your screenshot here*

What are the first three patterns that are matched during this part of the simulation? At what times does the pattern matcher recognize that a match has occurred?

Paste a screenshot below showing the final portion of your simulation, including the final values of the *top* and *bot* signals from *outMod*.

*paste your 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 a close-up view of a single pattern being matched.

*paste your screenshot here*

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

Paste a screenshot below, with an even closer-up view of the simulation showing a place where the valid signal goes high for a clock tick and also where the *inBits* signal from *binaryInMod* changes. Make sure that the button and knob signals are clearly readable.

*paste your screenshot here*

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):