ECE 3544: PROJECT 2

MODELING THE TIMING OF A DEVICE

DESIGN PROJECT

THIS ASSIGNMENT IS TO BE DONE INDIVIDUALLY

The purpose of this project is to use Verilog to model an available component, the CD74HC85 4-bit magnitude comparator, and to use the model to analyze the timing of a system that utilizes the component.

Step 1. Develop and verify an untimed Verilog model for the CD74HC85 4-bit magnitude comparator described in the file cd74hc85.pdf included with this specification.

You should implement the function table on page 2 using a Verilog behavioral model. Use the following module declaration for your circuit:

```
module hc85(a_in, b_in, ia_lt_b, ia_eq_b, ia_gt_b, oa_lt_b,
   oa_eq_b, oa_gt_b);
   input [3:0] a_in, b_in;
   input ia_lt_b, ia_eq_b, ia_gt_b;

   output oa_lt_b, oa_eq_b, oa_gt_b;
```

You must use exactly this module declaration, or we will be unable to verify the operation of your module when you submit it and you will receive a 0 for this portion of your project grade.

Use the following module declaration for your testbench:

```
module hc85_tb();
```

Note the testbench has no ports. The testbench should exhaustively test the `hc85` module by driving it with an 11-bit counter module created by copying and modifying the 4-bit counter supplied with this project description. The new counter should have a module name of `counter_11bit`. There is also a `clk` module supplied that can be used to drive `counter_11bit`. The clk and 4-bit counter modules are demonstrated in the accompanying test benches. You should study the test benches to be sure you understand the operation of the modules before you use them or modify them. Your 11-bit counter should have the same delay parameters as the 4-bit counter.

To simulate your `hc85` module with the `counter` module, your `hc85` module must have a `` `timescale `` directive before the module declaration. Use the same directive as the 4-bit counter example, `` `timescale 1 ns/100 ps ``. The timescale directive tells the simulator that the value of one unit of time (#1) is 1 ns, and the precision (smallest fraction of a time unit) is 100 ps.

Your report should include waveforms of inputs to and outputs from the `hc85` untimed module that confirms consistency with the data sheet's function table on page 2 of the datasheet. You

do not have to include all combinations of inputs on your waveforms, but you must show that your module behaves correctly for each line of the function table.

Do not proceed to step 2 until you have verified the correct operation of the untimed model.

Step 2. Use a specify block to add propagation delays to your `hc85` model. For the timing, please use the typical propagation delays for the HC types at 25 degrees C and 5V from the CD74HC85 datasheet, bottom of page 4 and top of page 5. You do not have to worry about the output transition time, only the propagation delays.

Simulate the timed model using your testbench. Your report should include waveforms showing the correct operation of the model with delays. You should also include waveforms with sufficient resolution to show that the delays are correct.

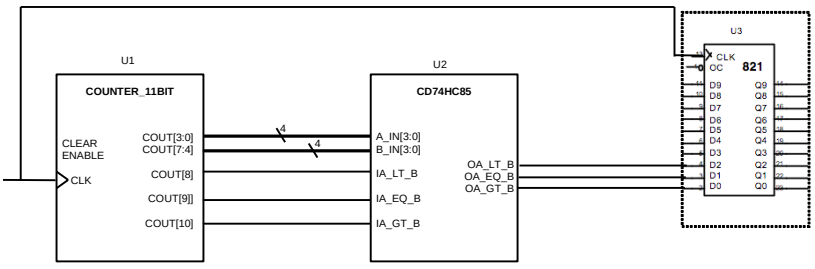
Step 3. Create a behavioral model of a 10-bit register. Your register must have the following module declaration:

```
module register_10bit(clk, d_in, q_out);
      input clk;
      input [9:0] d_in;
      output[9:0] q_out;
```

This register should load the values of d_in in parallel on every rising edge of clock. The functionality is essentially that listed for the 74FCT821 component described in this data sheet, http://www.ti.com/lit/ds/symlink/cy74fct821t.pdf, but without the ability to tri-state the outputs. Your model of the 10-bit register should not include any delays.

Create a testbench for your 10-bit register and verify that it operates correctly. Your report should include a waveform showing its correct operation.

Step 4. Use your models for the counter, the CD74HC85, and the 10-bit register (74FCT821) to create the model of a digital system shown below:



The system uses the counter (module instance U1) to provide the inputs to the CD74HC85 (module instance U2). The outputs of the CD74HC85 are then stored in the low order bits of the register (module instance U3). The counter and the register both use the same clock signal. The `counter_11bit` module should use the default propagation delay values that are in `counter_4bit`. The overall schematic should be created in a testbench module named `system_tbX()`, which should instantiate U1, U2, and U3. The `clk` module supplied with this project can be used to source all clocks.

You should create two versions of `system_tb`: The first version, `system_tb1()`, should define a clk PERIOD parameter that is sufficiently large to respect the propagation delays such that the system shows the correct behavior at all times, i.e., the correct values for OA_LT_B, OA_EQ_B, and OA_GT_B given the present values of A and B are always stored in the register. The second version, `system _tb2()`, should define a clk PERIOD parameter that is small enough that the system shows incorrect behavior sometimes for each of the three signals. For both versions of the testbench, the delays in your 11-bit counter should be the default delays in the 4-bit counter.

Your report should include waveforms showing that each of the modules you have created functions correctly, and showing both behaviors (valid data and invalid data). Your report should a brief description of any design decisions you made for your modules, as well as a brief analysis of how you determined the clock PERIOD parameter to demonstrate valid and invalid behavior. In particular, you should provide an analysis of the shortest clock period that will always allow correct (valid) behavior of the system.

Your project submission should include the following items:

1. Project report in PDF

2. Source files for:

- hc85 with timing
- hc85 testbench
- 11-bit counter
- 10-bit register
- 10-bit register testbench
- system_tb1
- system_tb2

Some of your modules will be tested with our own secret testbenches, so it is important that you use the module declarations given above. Failure to do so will result in a grade of 0 for those portions of the project. Some modules may be submitted to the MOSS service (theory.stanford.edu/~aiken/moss/) to check for plagiarism. Any copying flagged by MOSS will be treated as Honor Code violations.

Your report and source files should be submitted as a single zip file on the project assignment page.

Grading for your submission will be as described on the cover sheet included with this description. The cover sheet must be included as the first page of your report.