EGRE 427 Advanced Digital Design
Verilog Homework

This homework must be pledged as your own (individual) work – write it out and sign it.

1. In this exercise, you will develop a Verilog description of a binary to 7-segment display driver. This display driver is different from the usual ones in that it accepts an unsigned 7 bit binary number and drives two seven segment displays to display values from 0 to 99. Values over 99 result in a display of “EE”. A block diagram of the display driver is shown below:

To help you test your design, two special components have been created. The first is a component called `binary_input` that during simulation, prompts you to enter an integer value into the simulator control window and then drives this value on a 7-bit `std_logic` signal at its output. The second is a component called `led_driver`, that takes the 14 inputs, `A1` through `G0`, and creates a two digit, 7-segment display in a new Unix window. The values driven on the `A1` through `G0` inputs will cause segments to “light up” in the display.

In addition to these components, a “template” for the `display_driver` component you will design has been created as well as a test bench, called `led_testbench`, that hooks all of the components together for functional simulation. The test bench connects the output of the `binary_input` component to the input of the `display_driver` component and the outputs of the display driver component to the `led_driver` component as shown on the next page. Note that all of the components except for the `display_driver` are provided for you and are written in VHDL. A template for the Verilog description of the `display_driver` component is also provided for you to use as a starting point for your description.

To use these files, first make a directory for the exercise and copy the various source files for the display driver:
To compile the complete simulation package and run a test, type:

```
>> cd egre427
>> mkdir hw3
>> cd hw3
>> cp -r /mentor/examples/verilog/verilog_labs/led led
>> cd led
```

To compile the complete simulation package and run a test, type:

```
>> make
>> vsim led_testbench
```

When the simulation window appears, type “run –all” at the VSIM prompt. Note that when you do this, you will see the message “7 Segment LED” displayed in a new Unix like the one shown below and the message “Input decimal number:” will appear in the simulation control window. Note that the 7 Segment LED display is comprised of “B”s. This indicates the A1-G0 inputs to the LED display are not known (i.e., they are either ‘U’ or ‘X’) and the display is “blinking.”
You can start up a Wave window if you like and wave the signals in the top-level region. At that point, you can type in different numbers and the display will alternate between displaying an ‘8’ in digit 0 and digit 1.

You are now ready to edit the display_driver.v file and write your Verilog description of the led display driver. To compile the Verilog description, you can simply type “make” again at the Unix prompt in your terminal window.

2. In this exercise, you will develop a Verilog description of a traffic light controller. The controller is a finite state machine with a reset and clock input and it is assumed that the clock runs on a 1 Hz frequency (although for simulation purposes, we will actually simulate in on a 1 \( \mu \)s clock) so that you may use it as a timer.

The traffic light sits at the intersection between two, two lane roads, which run North-South (NS), and East-West (EW). The light cycles between green in the NS and EW directions on a two-minute period. There is an eight second yellow light period between green and red. The controller has outputs to turn on and off the red, yellow, and green lights in either direction (NS_red, NS_yellow, etc.). To make things a little more interesting, the controller has two inputs, car_in_NS and car_in_EW, that are connected to sensors in the road that tell it if a car is approaching the intersection and is within 200 feet of the intersection in the NS (either North or South) and EW directions.

These inputs are used in the following manner:

- If the light is about to turn red and there is a car approaching in the green direction and there is no car in the red direction, the start of the cycle to red is delayed for 20 seconds, one time only, to allow that car additional time to pass through.
- If the light has been red for more than one minute and a car approaches the light in the red direction and no car is approaching in the green direction, then the cycle to green will start immediately.
- Finally, at any point when the above decisions to modify the start of the light transition cycle are made, they are carried out regardless if another car approaches the intersection in any direction after they are made.

As with the previous problem, a simple display component, a template for the Verilog controller component, and a test bench have been created. The composition of the supplied test bench is shown on the next page. This time, however, there are two external inputs to the test bench, car_in_NS and car_in_EW, which signal the presence of a car in either direction as described above. You will have to either force values onto these signals during your simulation, or write another component (or process) in the test bench to drive them to values at the proper time.

Note that the signals that drive the car_in_NS and the car_in_EW inputs to the controller also go to the display device so that the presence of a car can be seen on the display panel.
To use the file that are provided for you, first make a directory for the exercise and copy the VHDL source files for the traffic light controller:

```bash
>> cd egre427
>> cd hw3
>> cp -r /mentor/examples/verilog/verilog_labs/tlc tlc
>> cd tlc
```

To compile and run the simulation type:

```bash
>> make
>> vsim tlc_testbench
```

When the simulation window appears, you can wave the signals in the design and run the simulation by successively typing “run 5000” in the VSIM window.

When you run the simulation, the traffic light display panel will appear in the Unix window where the simulation was run. You can now wave and force signals in the design to test the traffic light controller. An example of what is seen on the display panel when the simulation is running is shown on the next page.

You are now ready to edit the traffic_light_controller.v file and write your Verilog description of the traffic light controller. As before, you can compile the Verilog description by simply typing “make” at the Unix prompt.
Traffic Light Status

North-South

East-West

Last Change: 10 sec  Current Time: 14 sec  Last Change: 10 sec

Last Car at: 5 sec  This Car at: 5 sec