LSU EE 4755 Homework 1 Due: 7 September 2023

For instructions visit `https://www.ece.lsu.edu/koppel/v/proc.html`*. For the complete Verilog for this assignment without visiting the lab follow* `https://www.ece.lsu.edu/koppel/v/2023/hw01.v.html`*.*

Collaboration Rules

Each student is expected to complete his or her own assignment. It is okay to work with other students and to ask questions in order to get ideas on how to solve the problems or how to overcome some obstacle (be it a question of Verilog syntax, interpreting error messages, how a part of the problem might be solved, etc.) It is also acceptable to seek out digital design resources for help on Verilog, digital design, etc. It is okay to make use of AI LLM tools such as ChatGPT and Copilot to generate sample Verilog code. (Do not assume LLM output is correct. Treat LLM output the same way one might treat legal advice given by a lawyer character in a movie: it may sound impressive, but it can range from sage advice to utter nonsense.)

After availing oneself to these resources **each student is expected to be able to complete the assignment alone**. Test questions will be based on homework questions and **the assumed time needed to complete the question will be for a student who had solved the homework assignment on which it was based**.

Problem 0: Following instructions at `https://www.ece.lsu.edu/koppel/v/proc.html`, set up your class account, copy the assignment, and run the Verilog simulator on the unmodified homework file, `hw01.v`. Do this early enough so that minor problems (*e.g.*, password doesn't work) are minor problems.

Testbench

To compile your code and run the testbench press $\boxed{\text{F9}}$ in an Emacs buffer in a properly set up account. The testbench will test 3 modules, `minmax2p1` ($n = 2$), `minmax4` ($n = 4$), and `minmax8` ($n = 8$). Each module will be tested on 100 inputs. If a module's output on a particular input is incorrect, a message will be printed showing the incorrect and correct output. This output will only be shown for the first few errors, but a tally will be shown near the end counting all errors.

In an unmodified assignment the testbench will generate output that includes the following near the end:

```
Error n=8  max    z != 8107 (correct)
Error n=8  min    z !=  907 (correct)
Error n=8  max    z != 8156 (correct)
Error n=8  min    z !=  243 (correct)
Error n=8  max    z != 6424 (correct)
Done with n=8, tests, 100 min 100 max errors found.
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
Total number of errors: 600
```

The `z` in the output above means that the `minmax8` outputs (both `min` and `max`) were set to `z`, meaning the output was not connected (z is often used to indicate high impedance). The output of the testbench for a correctly completed assignment is:

```
Done with n=2, tests, 0 min 0 max errors found.
Done with n=4, tests, 0 min 0 max errors found.
Done with n=8, tests, 0 min 0 max errors found.
```

```
xmsim: *W,RNQUIE: Simulation is complete.
xcelium> exit
Total number of errors: 0
```

Helpful Examples

A good past assignment to look at is 2017 Homework 1. A question very similar to Problem 1 was asked in 2018 Homework 1 Problem 1. So, do not look at 2018 Homework 1 until after you have made a very serious attempt at Problem 1.

Problem 1: Module `minmax2`, shown below, sets output `min` to the smaller of its two inputs `a0` and `a1`, and sets `max` to the larger of those two inputs:

```
module minmax2
  #( int w = 10 )
   ( output uwire [w-1:0] min, max,   input uwire [w-1:0] a0, a1 );
   assign { min, max } = a0 <= a1 ? { a0, a1 } : { a1, a0 };
endmodule
```

Notice that `minmax2` uses a continuous assignment statement. Complete module `minmax2p1` so that it does the same thing as `minmax2`, but without a continuous assignment and without procedural code. Instead instantiate `compare_lt` and `mux2` modules (shown below). Follow other guidelines and requirements shown in the checkboxes in the Verilog file.

```
module compare_lt
  #( int w = 31 )
   ( output uwire lt, input uwire [w-1:0] a0, a1 );

   // Set lt to 0 if a1 < a0, set lt to 1 otherwise.
   assign lt = a0 <= a1;
endmodule

module mux2
  #( int w = 3 )
   ( output uwire [w-1:0] x,   input uwire s,  input uwire [w-1:0] a0, a1 );
   assign x = s ? a1 : a0;
endmodule
```

There is another problem on the next page.

Problem 2: Modules `minmax4` and `minmax8` each have outputs `min` and `max`, which are to be set to the smallest and largest values of their input. Input `a` to `minmax4` is a 4-element array of `w`-bit unsigned integers, and input `a` to `minmax8` is an 8-element array. Complete these modules as described below.

For this problem use modules `minmax2`, `min2`, and `max2`. Module `min2`, as one might guess, sets its output to the smaller of its two inputs. Module `max2` is similar. Assume that the combined cost of a `min2` and `max2` module is greater than one `minmax2` (but less than the cost of two `minmax2` modules).

```
module min2 #( int w = 10 )
             ( output uwire [w-1:0] min, input uwire [w-1:0] a0, a1 );
   assign min = a0 < a1 ? a0 : a1;
endmodule

module max2 #( int w = 10 )
             ( output uwire [w-1:0] max, input uwire [w-1:0] a0, a1 );
   assign max = a0 < a1 ? a1 : a0;
endmodule
```

(*a*) Complete module `minmax4` using instantiations of modules `minmax2`, and possibly `min2` and `max2` as needed. Do not use `assign` statements or procedural code. Follow other guidelines shown in the checklist in the code. Pay attention to the relative cost of the `min2`, `max2`, and `minmax2` modules.

(*b*) Complete module `minmax8` using instantiations of modules `minmax4`, and possibly `minmax2`, `min2`, and `max2` as needed. Do not use `assign` statements or procedural code. Follow other guidelines shown in the checklist in the code. Pay attention to the relative cost of the `min2`, `max2`, and `minmax2` modules.

The code must by synthesizable. To synthesize your code issue the command `genus -files syn.tcl`. If there are no errors, running this command will generate output that includes like the following:

```
Module Name                                  Area   Delay   Delay
                                                   Actual  Target
minmax2p1_w8                                15086   2.191  10.000 ns
minmax2_w8                                  15086   2.191  10.000 ns
minmax4_w8                                  49094   4.412  10.000 ns
minmax8_w8                                 117111   6.632  10.000 ns
minmax2p1_w8_1                              19678   1.029   0.100 ns
minmax2_w8_1                                19678   1.029   0.100 ns
minmax4_w8_1                                75084   1.496   0.100 ns
minmax8_w8_1                               214774   2.567   0.100 ns
```