ECEN 2350: Digital Logic

Assignment #5

1. [10 points.] On canvas is lab5-q1.v, which is a **buggy** 8-bit priority encoder written in verilog. That is, it is a priority encoder that does not quite work: at least one of its inputs will lead to an incorrect.

   However, given that it is an 8-bit priority encoder, there are 256 possible inputs - too many to manually write a truth table for. Instead, we'll use a **testbench** to find the bugs.

   Write a **testbench** to test our buggy priority encoder. Your testbench should instantiate the priority encoder (named `priorityEncoder8`), enumerate all potential 256 inputs, and correctly identify the case(s) where the priority encoder fails to produce the correct result, and print out:

   ```
   Err:  input iiiiiiii produced ooo
   ```

   where `iiiiiiii` is the 8-bit input, and `ooo` is the 3-bit output. For instance, your code might print out:

   ```
   Err:  input 00000100 produced 011
   ```

   (Note that input should have produced the output 010) You can ignore the `valid` bit output for this part, and the case where the input is all 0s (`00000000`).

   Submit your testbench as `priority-tb.v`.

2. [5 points.] Fix the incorrect 8-bit priority encoder above, to create `priorityEncoder8(input [7:0] in, output [2:0] out, output valid);` but with correct functionality. Test it against your testbench, and verify that it produces the correct outputs.

   For this part, you should make your `valid` bit be 1 iff the output is "valid"; that is, it is encoding a valid input (at least one bit in the input is set).

   Submit your priority encoder as `priority.v`

3. [5 points.] Extend your answer in Part 2 to create a 32-input priority encoder. Your module should have the prototype:

`priorityEncoder32(input [31:0] in, output[4:0] out, output valid);`

You should use modular design: you may want to create a `priorityEncoder16`. See the notes from class on how to create a modular decoder, and extend this idea to make a modular encoder.

Submit your priorityEncoder32 as `priority32.v`

4. [5 points.] Write a testbench for the `priorityEncoder32` in Part 3. Note that this is probably too large for you to enumerate all test cases, so you will want to create *random* cases.

   For this part, you do not need to write code to explicitly check if the output is as expected, but you should instead print out your random test cases. Each line should appear as follows:

   `Input iii...iii produced ooooo, v`

   Where `iii...iii` is the 32-bit input, `ooooo` is the 5-bit output, and `v` is the 1-bit valid.

   Your code should output 10 rows like this; spot check your answers manually to make sure your priorityEncoder32 is working properly!

   Submit your testbench as `priority32-tb.v`