University of California, Davis
Department of Electrical and Computer Engineering

EEC180 **DIGITAL SYSTEMS II** **Winter Quarter 2024**

LAB 3: Combinational Logic Design using Verilog II

Objective:
Leading zero detector (LZD) circuit is useful in many applications such as floating-point arithmetic. As the name suggests, an LZD-k circuit outputs the number of leading zeros in a k-bit binary input. For example, in an LZD-4 circuit the input is a 4-bit binary number and the output is 3-bit number denoting the number of leading zeros:

If input is 1010 output is 000, If input is 0010 output is 010.
If input is 0000 output is 100, If input is 0101 output is 001.
If input is 1111 output is 000, and so on ...

Prelab – LZD-4 circuit design (15 pts)

Design a combinational circuit for LZD-4 circuit starting from the truth table and using K-maps to minimize the logic and derive the output. Create the truth-table, K-map and show the schematic of the circuit.

Part 1 – LZD-4 (35 points)

Assume the inputs to your circuit are the switches SW [0:3] and the outputs are the LEDR [0:2].

Perform the following steps:

1. Implement your design in Verilog and verify that the design works correctly by simulation using a self-checking testbench. **Demonstrate your testbench to your TA.**

2. Synthesize and download the design to the DE-10 Lite board and demonstrate the LDZ-4 design to the TA. **Demonstrate your circuit to your TA.**

Part 2 – LZD-8 (50 points)

Design a LDZ-8 circuit that detects the number of leading zeros in a 8 bit number. **Note that you must use LZD-4 modules to build your LZD-8 circuits hierarchically.**

Since enumerating all $2^{n\ inputs}$ possible inputs is not scalable, use the **$random** task to generate random numbers in the testbench to test your design. An example of this task in a simple testbench is provided in the Appendix. You may test this code using https://www.tutorialspoint.com/compile_verilog_online.php to see how it works. This is a useful website to quickly test out bits of Verilog code. **Demonstrate your testbench to your TA.**

Use SW [0:7] as inputs and LEDR [0:3] as the output to test your design on the DE-10 board. **Demonstrate your circuit to your TA.**

Submission (100 pts total)

Submit the following items for verification and grading:

- Report containing
  - First page: Lab cover sheet with TA verifications for
    1) Prelab
    2) Part I – HW demonstration & testbench
    3) Part II – HW demonstration & testbench
  - Prelab solution
  - Outputs and/or waveforms from self-checking testbenches for Pt I-II
- Complete Verilog source code for all lab exercises including
  - partI.v, tb_partI.v
  - partII.v, tb_partII.v
  - Any additional design files required to compile your ModelSim/Quartus projects

Appendix – Using $random Verilog task in testbenches

The $random Verilog task is a convenient built-in function provided to quickly generate random bit vectors. Random stimulus is a key principle in HDL verification. Below is an example usage that prints 10 random numbers using a Verilog for loop.

```
module main;
integer x, i;
        initial begin
                for  (i=0; i<10; i=i+1) begin
                        x = $random % 256;
                        $display("Random Number is %h", x);
                end
                $finish;
        end
endmodule
```