EECS 151/251A Homework 2

Due Monday, Feb 11th, 2019

Suggestions for this HW Assignment

You will be asked to write several Verilog modules as part of this HW assignment. You are highly encouraged to test your modules by running them through a simulator. Several simulator options were discussed in Discussion 2 including `https://www.edaplayground.com` which is a free, online, Verilog simulator.

Problem 1: Interpreting Verilog

(a) Sketch a gate-level schematic of the circuit described by the Verilog code below.

(b) Try to simplify the schematic by sharing gates when possible.

```
module circuit1(a, b, c, y z)
  input a, b, c;
  output y,z;

  assign y = a & b & c | a & b & ~c | a & ~b & c;
  assign z = a & b | ~a & ~b;
endmodule
```

Problem 2: Interpreting and Re-Implementing Verilog

(a) Sketch a gate-level schematic of the circuit described by the Verilog code below (using only simple logic gates)

(b) Try to simplify the schematic by sharing gates and eliminating unnecessary gates.

(c) Write an equivalent implementation of circuit2 that uses the Verilog "case" construct.

```
module circuit2(a, y)
  input [3:0] a;
  output [1:0] y;

  always @(*)
    if (a[0]) y = 2'b11;
    else if (a[1]) y=2'b10;
    else if (a[2]) y=2'b01;
    else if (a[3]) y=2'b00;
    else           y=a[1:0];
endmodule
```

Problem 3: Translating to Verilog

Consider the decoder examples presented in lecture 3, page 5 of the notes. These examples used a fake HDL. Rewrite these examples in proper Verilog.

Problem 4: Testing a Grey-code to Binary Converter

(a) Write a Verilog testbench that would instantiate and exhaustively test a 4-bit Gray-code to binary-code converter. (Hint: you may use the generator presented in class.)

(b) Draw the gate-level circuit diagram of the 4-bit Grey-code to binary converter.

Problem 5: Implementing an Adder/Subtractor

Subtraction, for instance A-B, can be implemented as A+(-B). Therefore, any adder circuit can be converted to a subtractor by converting B to its 2's complement, defined as ∼B+1.

Write a *structural* Verilog module as follows: `module add_sub(A, B, R, add)` which produces A+B if "add" is equal to 1 and produces A-B otherwise.

(hint, for an adder, the carry-in to the first stage provides an easy way to add 1 to the final result).

Problem 6: 251A only — *Optional* Challenge Question for 151

(a) Write the Verilog description for a LUT3 (3 input LUT with 1 output) module. It will have 4 inputs. The first three are the single bit LUT inputs: a, b, c. The last input, F, specifies the LUT function as an 8 bit number. For instance if F=8'h01, the LUT implements an AND gate. If F = 8'h7F, it implements an OR gate.

(b) Using only the LUT3 module you created, write a LUT4 module using *structural* Verilog.