Alex Oswald
EECE 2323
Due: 22 Sep 2020

Pre-Lab Assignment 1

1. Complete the table below for the expected input and output values for the 8 bit adder circuit you are describing in lab 1. Note that the syntax used for constant values is from Verilog; a,b and f are two's complement number.

a	b	f	ovf
8'd0	8'd0	**8'd0**	**1'b0**
8'd12	8'd34	**8'd46**	**1'b0**
-8'd12	-8'd34	**8'd22**	**1'b0**
8'd100	-8'd50	**8'd50**	**1'b0**
-8'd100	8'd50	**-8'd50**	**1'b0**
8'd100	8'd100	**-8'd56**	**1'b1**
-8'd100	-8'd100	**8'd56**	**1'b1**

2. Write a Boolean or verilog equation for the overflow output ovf based on input values a and b. Your equation can also incorporate output f.

```
assign ovf = (a[7] && b[7] && ~f[7]) || (~a[7] && ~b[7] && f[7]);
```

3. A good simulation testbench tests that every input and output bit can take the values zero and one. For the values in question 1 answer the following questions:
   a. The adder has 16 bits of input and 9 bits of output. Do the values in question 1 taken together set every input bit to both one and zero and every output bit to both one and zero? Explain your answer. Identify which bits are not tested in this manner.
   b. Add additional input (a and b) values that test every bit in both the inputs and outputs so that every bit takes either a zero or a one value in your test cases.

3a: No. They do not set every bit to both one and zero and every output bit to both 1 and 0. What's most significant is that there are 65,536 combinations of 1's and 0's of input not that at some point each bit position (e.g. b[2] or f[7]) has among any of the tests been a 0 or a 1.

3b: see code below…

```
`timescale 1ns / 10ps

module adder8_tb_full ();
    reg signed [7:0] a;
    reg signed [7:0] b;
    wire signed [7:0] f;
    wire ovf;

    eightbit_adder uut
        (.a(a),
         .b(b),
         .f(f),
         .ovf(ovf));

    // Test Every Possible Option
    initial
        begin
            #0
            a = 8'b00000000;
            repeat (255)
            begin
                b = 8'b00000000;
                repeat(255)
                begin
                    #5  b = b + 8'b00000001;
                end
                #5  a = a + 8'b00000001;
            end
        #10 $stop;
        end


    // Display Log
    initial
        begin
            $display ("Test of 8-bit adder.");
            $monitor("[a b f ovf]: a=%d  b=%d  f=%d  time=%d", a, b, f, ovf, $time);
        end

endmodule
```