CMPE 650 - Homework #1

Start assignment as soon as possible. Work individually, but you can ask your classmates for help when you get stuck, with consideration to the **course collaboration policy (please read it in the course website)**. Please send me email if something isn't clear and I will update the assignment. Changes are logged at the bottom of this page.

Before getting started, you should go through the verilog notes located under Course Readings on the course home page.

A paper copy of everything and electronic copies of all your code and testing files (all in one zipped file) **are due at the beginning of class on the due date**.

Notes:

- [15% of points] Clearly state whether your design is fully functional, and state the failing sections if any exist.
- Make sure your design and code are easily readable and understandable (clear and well commented).
  - Up to 5% extra credit will be given for especially thorough, well-documented, or insightful solutions.
- *** Where three '*'s appear in the homework, perform the required test(s) and turn in a printout of either:
  1. a table printed by your verilog testbench module listing all inputs and corresponding outputs,
  2. an Isim waveform plot which clearly shows (labeled and highlighted) corresponding inputs and outputs, or
  3. a section of testing code which **clearly** compares the designed circuit and a simple reference circuit, and two short cut & paste sections of text from your simulation (one for pass, and one for fail where you purposely make a slight change to your reference code to make it fail) that look something like this:
     ```
     Error: input=0101, out_module=11110000, out_ref=11110001
     ```

In all three options, each test case must be marked whether the output is correct or not.

Keep "hardware" modules separate from testing code. Instantiate a copy of your processing module(s) in your testing module (the highest level module) and drive the inputs and check the outputs from there.

1. [35 pts] Design and write the verilog for a block that adds three 5-bit numbers into a 2's complement output that is sufficiently large to represent all inputs but with no extra bits. Use one stage of 3:2 carry-save adders and one carry-propagate (ripple) adder using a "+" in verilog. The three inputs are as follows:
    - `a` is in 2's complement 1.4 format
    - `b` is in 2's complement 2.3 format
    - `c` is in unsigned 5.0 format

    a) [2 pts] How many bits does the output have and where is its decimal point?
    b) [4 pts] Show the adder's block diagram.
    c) [3 pts] What is the output's minimum attainable negative value (most negative)?
    d) [3 pts] What is the output's minimum attainable positive value?
    e) [3 pts] What is the output's maximum attainable positive value?
    f) [20 pts] Test the circuit over at least 15 input values (including all extreme cases). Turn in ***, option 1

```
        5     +-------+
a ------/-----|       |
        5     |       |     ?
b ------/-----|   +   |-----/------ out
        5     |       |
c ------/-----|       |
              +-------+
```

2. [30 pts] Design a block which adds 27 single-bit numbers, but where each input is not a standard binary number--instead each input represents -1 or +1. A zero input is -1 and a one input is +1. So the sum ranges from [-27, +27] and the output of the adder is to be represented by a 2's complement number sufficiently wide to represent all input combinations. You may use 3:2 carry save, carry ripple (propagate) adder, and half adders (implemented as submodules however your wish; e.g., wire, reg, table) to add the inputs efficiently. Use Minimum number of carry ripple (propagate) adder, which you can implement with a "+" or "-" in verilog.

```
              +-------+
        27    |       |     ?
in ------/-----|   +   |-----/------ out
              |       |
              +-------+
```

    a) [20 pts] Draw a block diagram for an efficient adder.
    b) [5 pts] Total up and state how much hardware (in area) your design requires in units of 3:2 adders assuming a half adder costs 0.5 3:2 adders.
    c) [10 pts] Write your design in verilog, test it, and turn in ***, Option1.

3. [35 pts] Design and write the verilog for a block that performs floating point to fixed point number conversion. The floating point input has a 7-bit signed, 2's complement mantissa in "2.5" format and a 3-bit 2's complement integer exponent. The fixed point number (output) has enough bits to fully represent the converted floating point number, but no more.

   a) [6 pts] How many bits does the fixed-point output have and where is its decimal point?
   b) [3 pts] What is the output's minimum attainable negative value?
   c) [3 pts] What is the output's minimum attainable positive value?
   d) [3 pts] What is the output's maximum attainable positive value?
   e) [20 pts] Test the circuit over at least 15 input values (including extreme cases). Turn in ***, option 1

```
               7     +--------+
mantissa ------/-----| float  |     ?
                     |  to    |-----/------ out
     exp ------/-----| fixed  |
               3     +--------+
```