CMSC 313 Spring 2024
Homework 4
due Monday, March 4, 11:59pm

Introduction: For HW4, you'll be using a Logic Simulator to build cobinational circuits. The simulator we'll use in this class is Logisim-Evolution which you can download from: https://github.com/logisim-evolution/logisim-evolution. There are download instructions on the README.

You are provided a starting circuit file which has a framework which you will edit. This file contains 6 sub-circuits: FA, FourBitAdd, FourBitAddSub, SHL, SHR, & ALU, which you can see on the left hand panel in Logisim.

A "test vector" is a file used to test the boolean logic of a circuit in Logisim. Each circuit will have a provided test-vector which you can use to check your work. If your circuit passes every test within the test vector, you will get full points on that part. Grades will be assessed based on passing number of tests on each circuit.

You shall submit a single file which is labeled "`HW4_<your first name>_<your last name>.circ`".

Testing: Included in this assignment are test vector files. Each of the circuits you will create will be tested with a different test vector. To test your circuit, first make sure you have opened (double clicked) your selected circuit. Then select `Simulate > Test Vector...` after which you will need to upload the vector file to test the specific circuit. Each circuit's test vector provided is named after the corresponding circuit with the trailing suffix "-test.txt". For example, the ALU circuit's test vector is "ALU-test.txt" You will get full marks on that circuit if all tests pass. A grade will be given based on the number of tests that pass.

Exercise 1. Create an 4-bit ripple adder/subtractor that adds or subtracts two binary numbers $a = a_3a_2a_1a_0$ and $b = b_3b_2b_1b_0$, and produces an output $d = d_3d_2d_1d_0$.

a. (5 pts) Create a Full Adder. This circuit is named "FA" and has inputs **{a, b, cin}** and outputs **{s, cout}**.

b. (10 pts) Create a 4-bit ripple adder by chaining together full adders from part a. This circuit is named "FourBitAdd". This 4-bit ripple adder should have inputs exactly **{a[4], b[4], cin}** and outputs **{s[4], cout, overflow}** where **[4]** means that the input is a bus/bundle of width 4. The output **overflow** is defined the same way 2's complement overflow is explained in class. *Hint: if a_3 and b_3 are the same (both numbers are positive or both numbers are negative), and if s_3 is different from a_3 (the answer has a different sign then one of the inputs), then overflow = 1 otherwise, overflow = 0.*

c. (10 pts) Create a 4-bit ripple subtractor/adder by reusing the module from part b. This circuit is named "FourBitAddSub". The subtractor should perform 2's complement addition $a + b$ whenever enable input **en=0** and perform 2's complement subtraction $a - b$ whenever enabled input **en=1**. The subtractor/adder has inputs exactly **{a[4], b[4], en}** and outputs **{s[4], cout, overflow}**.

Exercise 2. Create a Shift Left module and a Shift Right module in their own sub-circuits.

a. (5 pts) The Shift Left circuit shift all bits to the left by one. It performs the operation `s = a<<1` given the input **{a[4]}** and output**{s[4]}**. The most significant bit should be truncated, and the new unassigned bit in the least significant bit place shall be filled with a zero. This circuit is named "SHL".

b. (5 pts) The Shift Right circuit shift all bits to the right by one. It performs the operation `s = a>>1` given the input **{a[4]}** and output**{s[4]}**. The least significant bit should be truncated, and the new unassigned bit in the most significant bit place shall be filled with a zero. This circuit is named "SHR".

Exercise 3. (15 pts) Create a 4 operation 4-bit ALU that supports the following operations: Add, Subtract, Shift Left and Shift Right for the op values 00, 01, 10, and 11 respectively. This circuit is named "ALU". The ALU has inputs **{a[4], b[4], op[2]}** and output **{s[4], cout, overflow}**.

op	operation name	action
00	add	s = a + b
01	subtract	s = a - b
10	Shift Left	s = a <<1
11	Shift Right	s = a >>1

The adder and subtractor should be implemented with the same FourBitAddSub subcircuit, don't use a separate subcircuit for the adder and subtractor logic. Also, the outputs **{cout, overflow}** should only be enabled if the add or subtract operation is being used; otherwise, they should be set to 0.