Homework #3 – Digital Logic Design

Due Tuesday, March 1, at 5:00pm

Directions:

- All work must be **submitted via Gradescope**.
  - For short-answer questions, submit your answers in **PDF format** as a file called **<NetID>-hw3.pdf**. ***Word documents will not be accepted.***
  - For programming questions, submit your source file using the filename specified in the question.
- **You must do all work individually.**
  - All submitted circuits will be tested for suspicious similarities to other circuits, and the test will uncover cheating, even if it is "hidden" (by reordering components, by renaming labels, etc.).
- You will be using Logisim for this assignment to implement circuits. Logisim is pre-installed on the ECE/CS 250 containers.
- For Logisim questions:
  - **Circuits will be tested using an automated system, so you must name the input/output pins exactly as described and submit using the specified filename!**
  - **You may only use the basic gates (NOT, AND, OR, NAND, NOR, XOR, XNOR), input/output pins, tunnels, probes, power, ground, D flip-flops, multiplexers, splitters, constants, and clocks. You must construct any other component from these basic components. Some *questions have additional restrictions on the gates you can use.***
- Late penalties are applied to the **entire** assignment regardless of whether portions of it where submitted before the deadline.
- **The latest submission to Gradescope will be graded** regardless of whether it's the highest scoring submission.

Access the Assignment Files

There is an **automated self-test tool** provided in the usual place on Sakai->Resources. Note, **the automated tests provided are not exhaustive**, and additional tests will be used by the instructors for grading.

Debugging

To get started with debugging, consult the "A Guide to Debugging Logisim" document available under Sakai > Resources. In addition, the "Making Custom Test Cases for the Test Kit" document under Sakai > Resources outlines the steps to add your own automated tests .

Back Up Your Work

Logisim is known to have a bug where **files get corrupted**. To make sure you do not lose progress on the off chance this happens to you, **regularly backup your work (to Box, Git, wherever).** Corrupted files cannot be recovered!!

Q1. Boolean Algebra

(a) [5 points] Write a truth table for the following function:

$$out = \overline{AB} + ((C) + (B\bar{C}))$$

Your truth table should be in the standard order (i.e. inputs counting up in binary) and your columns should be A, B, C, output from left to right. **Not following this will results in a -1 point penalty.** See question 1(c) for an example of a truth table in the standard order.

(b) [10] Use Logisim to implement and test the circuit from (a). Name this file circuit1a.circ. Your circuit must have the following pins:

Label	Type	Bit(s)
A	input	1
B	input	1
C	input	1
out	output	1

Refer to the first page of the assignment regarding which Logisim components you may use. **Answers which use disallowed components will be penalized 50%.**

You will submit `circuit1a.circ` to Gradescope.

The following are the tests provided with the test kit program for this problem:

Test Number	**Parameters Passed**
0	A=0, B=0, C=0
1	A=0, B=0, C=1
2	A=1, B=0, C=1
3	A=1, B=1, C=1

(c) [5 points] Write a sum-of-products Boolean function for both outputs in the following truth table. You should use only AND, OR, and NOT gates. **Answers which use gates other than these will receive a 0.**

A	B	C	out1	out2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

(d) [10] Use Logisim to implement and test the circuit from (c). Name this file circuit1c.circ. Your circuit must have the following pins:

Label	Type	Bit(s)
A	input	1
B	input	1
C	input	1
out1	output	1
out2	output	1

Refer to the first page of the assignment regarding which Logisim components you may use. **Answers which use disallowed components will be penalized 50%.**

You will submit `circuit1c.circ` to Gradescope.
The following are the tests provided with the test kit program for this problem:

Test Number	**Parameters Passed**
0	A=0, B=0, C=0
1	A=0, B=0, C=1
2	A=0, B=1, C=0
3	A=1, B=0, C=0

Q2. Adder/Subtractor Design

[30] Use Logisim to build and test a 16-bit ripple-carry adder/subtractor. You must first create a 1-bit full adder that you then use it as a module in the 16-bit adder. The circuit should perform `A+B` if the `sub` input is zero, or `A−B` if the `sub` input is 1. (It must be A-B, instead of B-A.) The circuit should also output an overflow signal (`ovf`) indicating if there was a signed overflow.

Name the file adder.circ. Your circuit must have the following pins:

Label	Type	Bit(s)
A	input	16
B	input	16
sub	input	1
result	output	16
ovf	output	1

Note: You can use splitters to split the 16-bit inputs and to combine the individual outputs of the one-bit adders together.

Refer to the first page of the assignment regarding which Logisim components you may use. **Answers which use disallowed components will be penalized 50%. Further, answers which utilize the built-in adder or subtractor will receive a 0.**

You will submit `adder.circ` to Gradescope.

The following are the tests provided with the test kit program for this problem:

Test Number	**Parameter Passed**
0	A=0xABCD, B=0x5678, sub=0
1	A=0x7531, B=0x6420, sub=1
2	A=0x9BDF, B=0x8ACE, sub=0
3	A=0xABCD, B=0x5678, sub=1

Q3. Finite State Machines

Design the following finite state machine (FSM). It has two 1-bit inputs (`in1` and `in2`) and one 1-bit output (`out`). The output (`out`) bit should be equal to one if, on both of the last two cycles, `in1` and `in2` were not equal to each other; otherwise, `out` should equal zero. When the FSM starts, assume for the sake of determining the output value that the "previous cycles" (which didn't exist) do not satisfy the requirements for the output being equal to 1.

(a) [10] Draw a state transition diagram, where each state has a unique "name" that is a string of bits (e.g., states 00, 01, and 11) as well as the associated value for `out`. Label all of the arcs between transitions with the inputs that cause those transitions. You MUST implement a Moore machine. **Not doing so will result in a 50% penalty.**

(b) [10] Draw a truth table for the state transition diagram. The inputs are `in1`, `in2`, and the current state bits. The outputs are `out` and the next state bits.

(c) [30] Use Logisim to implement and test this circuit. Name this file fsm.circ. Your circuit must have the following pins:

Label	Type	Bit(s)
in1	input	1
in2	input	1
out	output	1

Additionally, to keep this problem from becoming either trivial or troublesome, please adhere to the following restrictions:

- Refer to the first page of the assignment regarding which Logisim components you may use. **Answers which use disallowed components will be penalized 50%.**
- Implement your FSM as a "Moore" machine, meaning that the output should depend *exclusively* on the current state. In other words, your output should be written on the state nodes in the

state transition diagram rather than on the edges. When writing the truth table for this, the `out` column should just be based on the current state columns.

- Connect a "Clock" component to all the clock inputs in the DFFs. You may only have one clock in the circuit, and it is distributed to all of the DFFs.

You will submit `fsm.circ` to Gradescope.

The following are the tests provided with the test kit program for this problem:

Test Number	**Parameter Passed**
0	constant input; in1=0, in2=0; 5 cycles
1	constant input; in1=1, in2=0; 5cycles
2	5 cycles variable input