Homework #3 – Digital Logic Design

Due Wednesday, March 2 at 5:00pm

<u>You must do all work individually</u>, and you must submit your circuits electronically via Sakai.

All submitted circuits will be tested for suspicious similarities to other circuits, and the test will uncover cheating, even if it is "hidden."

Boolean Algebra

1) [5 points] Write a truth table for the following function: $Output = \overline{(A + \overline{B})} \cdot C + \overline{((A \cdot B) + (C \cdot B))}$

2) [5 points] Write a sum-of-products Boolean function for both outputs in the following truth table and then minimize them (individually) using Boolean logic, de Morgan's laws, etc. Use only AND, OR, and NOT gates. You do NOT have to have a perfectly optimal circuit, but you must show at least one Boolean optimization (in which your logic function has fewer terms than originally) for Output1 and at least one for Output2. You will build and test this circuit in Question 5.

A	B	C	Output1	Output2
0	0	0	0	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Finite State Machines

3) [10] Design the following finite state machine (FSM). It has two 1-bit inputs (in1 and in2) plus a clock input (clk). It has one 1-bit output (out). The output bit should be equal to one if, on all of the last three cycles, in1 and in2 weren't equal to each other; otherwise, the output bit should equal zero. For full credit, you must use the systematic design methodology we covered in class:

a) 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). Label all of the arcs between transitions with the inputs that cause those transitions. Do not worry about labeling the clock input.

b) 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. Do not include the clock input.

You will build and test this circuit in Question 6.

Building Logic with Logisim

The next several questions involve you building and testing logic circuits with Logisim.

You may use all basic gates (NOT, AND, OR, NAND, NOR, XOR) and D flip-flops. Everything else you must construct from these basic gates and flip-flops.

You may not use any pre-existing Logisim circuits (i.e., that you could possibly find by searching the internet). Plagiarism of Logisim code will be treated as academic misconduct.

For each question, upload your circuit (named questionX.circ for Question #X) to Sakai.

4) [10] Use Logisim to build and test the circuit from Question #1.
5) [10] Use Logisim to build and test the circuit from Question #2.
6) [30] Use Logisim to build and test the circuit from Question #3.
7) [30] Use Logisim to build and test a 16-bit (ripple-carry) adder. You **must** first create a 1-bit full adder that you then use as a module in the 16-bit adder. I want you to become comfortable with modules and hierarchical design.