Homework #3 – Digital Logic Design

Due Wednesday, March 1, at 5:00pm

Directions:

- For short-answer questions, submit your answers in PDF format as a file called <NetID>-hw3.pdf. (Don't include the <> just have your Duke NetID...) ***Word documents will not be accepted.***
    - Please type your solutions. If hand-written material must be included, ensure it is photographed or scanned at high quality and oriented properly so it appears right-side-up on screen.
    - Please include your name and Duke NetID on all submitted work.
- **You must do all work individually, and you must submit your work 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." Plagiarism of Logisim code will be treated as academic misconduct.
- **Logisim implementations must use only the primitive gates (NOT, AND, OR, NAND, NOR, XOR) and D flip-flops, except for Q2, where you are allowed to use 2-to-1 Multiplexer.**
- **You must construct any other functionality from these primitive gates and flip-flops.**
- **You may not use any pre-existing Logisim circuits (i.e., that you could possibly find by searching the internet).**
- **Since we use the auto-grading system, you must label your pins as stated in the specifications!! Otherwise your circuit will not be considered as correct.**
    - For each question, upload your circuit with the specified filename to Sakai. Any other filename will not be graded.
    - As previous, you will have a self-testing tool (homework3-kit.tar.gz).

Q1. Algebra & Combinational Logic

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

$$out = \bar{B} \cdot C + (A + B) \cdot \bar{C}$$

b. [10] Use Logisim to implement and test the circuit in (a); name the file circuit1.circ. Your circuit MUST have the following pins:

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

c. [5] Write a sum-of-products Boolean function for **both** outputs in the following truth table. (You should use only AND, OR, and NOT gates.) You do NOT have to have to optimize the results.

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

d. [10] Use Logisim to implement and test the circuit in (c); name the file circuit2.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

Q2. Adder/Subtractor Unit Design

[30 points] You are going to write a ripple-carry adder/subtractor that performs (signed) 32-bit addition and subtraction. **You are allowed to use 2-to-1 Multiplexer in this questions.** Your circuit should have the following pins:

Label	Type	Bit(s)
A	input	32
B	input	32
sub	input	1
result	output	32
ovf	output	1

If **sub** = **1**, the adder performs a subtract operation. If **sub** = **0,** it performs an addition.

Important Note: you should properly output the overflow (ovf), based on what you have learned about the signed addition/subtraction in class. Name the file adder.circ.

Q3. Finite State Machines

Design the following finite state machine (FSM). It has three 1-bit inputs (`in1` and `in2` and `in3`) plus a clock input (`clk`). It has two 1-bit outputs (`out1` and `out2`). The output bit `out1` should be equal to one if, on the previous cycle, at least two of the inputs were equal to one; otherwise, `out1` should equal zero. The output bit `out2` should be equal to one if, two cycles ago, exactly one of the inputs was equal to one.

For full credit, you must use the systematic design methodology we covered in class:

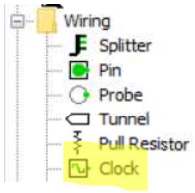
(a) [10 points] Draw a state transition diagram, where each state has a unique "name" that is a string of bits (e.g., states 00, 01, and 11) as well as the associated values for `out1` and `out2`. Label all of the arcs between transitions with the inputs that cause those transitions. Do not label the clock input.

(b) [10] Draw a truth table for the state transition diagram. The inputs are in1, in2, in3, and the current state bits. The outputs are out1, out2, and the next state bits. Do not include the clock input.

(c) [30] Use Logisim to implement and test the circuit for your Finite State Machine; name the file fsm.circ. Your circuit should contain the following pins:

Label	Type	Bit(s)
in1	input	1
in2	input	1
in3	input	1
out1	output	1
out2	output	1

Important Note: for this homework, you don't need to create a "clk" pin. However, you should have a clock source component connecting to the clock input of DFFs.



Additionally, to keep this problem from becoming too troublesome, 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.