Carleton University

Department of Systems and Computer Engineering

SYSC 2310 Fall 2017

Introduction to Digital Systems

Lab #3

Lab Report Due: At **6:00 am** on Monday, Oct. 30.

Submit the report as a SINGLE PDF FILE. Place the report in a folder named Lab3, along with your lab work as described below. Zip the folder (Lab3.zip) and submit it to the appropriate location in the cuLearn webpage for your lab section (not the main course webpage!).

Only submit your work in the Windows-compatible ZIP format. Do not submit your work in RAR, 7z or other format. (If the TAs can't open your work using Windows' built-in zip utility then they will not mark it.)

Note that the system will automatically stop accepting submissions at the due date/time for your lab section.

LATE submissions will NOT be accepted.

Read this document completely and carefully before deciding what to do.

Please bring questions about the lab to class for discussion at the beginning of class.

In this lab you will:

- Design a logic circuit that compares two integer values under both Unsigned and Signed interpretations.
- Enter circuit designs into Logisim.
- Test and debug circuits using Logisim's simulation capability.

"[Report]" indicates content that should be addressed in the lab report.

This is Version 3 of the lab instructions. Fixed typo in Bonus section: replaced "-127" with -128"

Version 2: Changed due date. Fixed typo: replaced several occurrences of "flag" with "output".

Step 1: Read this entire document before proceeding to Step 2.

While processing information, Digital Systems must often compare two values before deciding what action should be taken. The Integer values X and Y are often compared by calculating X – Y and examining the result. In this lab you will design, build and test a circuit that takes two 8-bit numbers X and Y as inputs, calculates their difference (i.e. X – Y) and generates the six interpretation-specific outputs shown in the table below:

Output	Condition for Output = 1
U=	X = Y under Unsigned interpretation
U<	X < Y under Unsigned interpretation
U>	X > Y under Unsigned interpretation
S=	X = Y under Signed interpretation
S<	X < Y under Signed interpretation
S>	X > Y under Signed interpretation

Many of the steps below require some analysis. These steps all begin with "Do some analysis of X, Y and X – Y to … ". In every step, you should also consider the analysis you have done in earlier steps. (Hint: It may turn out that the analysis and/or circuit from an earlier step may be a valuable part of a later step.)

Step 2: Download the Lab3.zip file to a persistent work area, and unzip the folder.

The file Lab3Start.circ and any libraries it needs have been provided in Lab3.zip. The Lab3TestHelper provided in Lab 3 is a bit different than the Lab1TestHelper used in previous labs. The Lab3TestHelper has been pre-programmed with specific pairs of 8-bit test values to be used when testing your circuit. A pair of test values is always present at the X and Y outputs. Pressing the R button Resets the TestHelper to output the first pair of test values. Pressing the C button advances the TestHelper to the next pair of test values. Once all of the data pairs have been output, the Done LED turns ON, and the TestHelper must be Reset before it can be used to cycle through the test pairs again.

Step 3): Extend Lab3Start to include an 8-bit Subtracter that uses negation to compute X + (– Y) (build the circuit that was described in class and is shown in the course slides). Instead of using individual 1-bit Full (or Half) Adders to build a carry-ripple Adder as described in the slides, use one (and only one) instance of Logisim's built-in 8-bit Adder (available in the Arithmetic library in the Explorer Pane). Connect the Subtracter inputs to the Lab3TestHelper outputs, and use the TestHelper's test pairs to confirm that your circuit is working correctly.

Save your circuit in the unzipped Lab3 folder as Lab3Solution (saving work often is always a good idea).

Step 4): Do some analysis of X, Y and X – Y to determine how a circuit could decide that X = Y under Unsigned interpretation. (Paraphrasing: By considering the Unsigned Integers X, Y and X – Y (and possibly their binary representations), what condition is always true when X = Y under Unsigned interpretation and is always false when X ≠ Y under Unsigned interpretation? How could a circuit detect that condition?) Extend your circuit to implement the U= output. Add an LED (with the label "U=") to the U= output.

Step 5): Do some analysis of X, Y and X – Y to determine how a circuit could decide that X = Y under Signed interpretation. Extend your circuit to implement the S= output. Add an LED (with the label "S=") to the S= output.

Step 6): Do some analysis of X, Y and X – Y to determine how a circuit could decide that X < Y under Unsigned interpretation. Extend your circuit to implement the U< output. Add an LED (with the label "U<") to the U< output.

(saving work often is always a good idea)

Step 7): Do some analysis of X, Y and X – Y to determine how a circuit could decide that X > Y under Unsigned interpretation. Extend your circuit to implement the U> output. Add an LED (with the label "U>") to the U> output.

Step 8): Do some analysis of X, Y and X – Y to determine how a circuit could decide that X < Y under Signed interpretation. (Hint: The analysis in this step must consider a few cases, so this circuit will be slightly more complex than the others. Prof. Pearce's solution uses 4 gates.) Extend your circuit to implement the S< output. Add an LED (with the label "S<") to the S< output.

[Report] Describe your analysis and how it relates to your S< circuit in the report.

Step 9): Do some analysis of X, Y and X – Y to determine how a circuit could decide that X > Y under Signed interpretation. (Hint: There is a very simple solution that involves an earlier hint.) Extend your circuit to implement the S> output. Add an LED (with the label "S>") to the S< output.

[Report] Describe your analysis and how it relates to your S> circuit in the report.

Step 10): Log the behaviour of your final circuit. In the log, include the following inputs and outputs in the following order: X Uns Y Uns U< U= U> X Sig Y Sig S< S= S>

Record X Uns and Y Uns as Unsigned values. Record X Sig and Y Sig as Signed values. Recall from Lab 2 that you will have to enter the Signed values manually.

Log the behaviour for all test pairs provided by the TestHelper, starting with the first test pair (i.e. the pair provided after Resetting the TestHelper).

Save your log in the unzipped Lab3 folder as Lab3SolutionLog.txt.

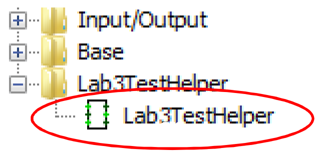
Save your final circuit.

[Report] Include the log in your report.

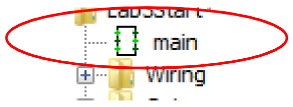
Bonus: Before doing the Bonus part, Save your circuit in the unzipped Lab3 folder as Lab3Bonus, and continue to modify the Lab3Bonus circuit.

As discussed in class, Negation doesn't work properly for the minimum negative value (in this case: the minimum value is –128). Modify the contents of the TestHelper to test this case. **Warning**: using a different procedure to modify the test values could complicate your time spent on this part. To modify the TestHelper test pairs, follow these steps:

1) Reset the TestHelper.
2) Do **NOT** open the Lab3TestHelper.circ file in a separate instance of Logisim (this could get complicated). Instead, in the instance of Logisim that is currently editing your Lab3Bonus circuit, open (expand) the Lab3TestHelper folder in the Explorer Pane and double-click on the Lab3TestHelper icon.



3) The Lab3TestHelper circuit should now be shown on the canvas. The large block in the middle of the circuit holds the test pairs. Each pair is stored as a 16-bit hex value. Look closely at the pairs and you will recognize the hex values from your earlier testing.
4) **Using the Poke tool**, poke the pair that has the black background… the pair should now also be enclosed in a red box. Type in a 16-bit hex value that represents one of your test pairs.
5) Double-click on the "main" icon in the Explorer Pane.



6) Your circuit should now re-appear on the canvas. Reset the TestHelper. The test pair you entered in 4) should now appear as X and Y.
7) Repeat from 2) to modify as many test pairs as needed. (You can modify as many pairs as you like before returning to main circuit. Poke the pair you would like to modify and enter the value. A pair does not need to have the black background in order to be modified. Note that the TestHelper is configured for 8 test pairs … you shouldn't need that many.)

Log the behaviour for your test cases. The contents of the log must have the same columns (in the same order) as in Step 10.

Save your log in the unzipped Lab3 folder as Lab3BonusLog.txt.

Save your final bonus circuit.

[Report] Include the log for your test cases. Briefly describe why your test pairs all generate a Negation error. Based on your test cases, is the Negation error an issue for any of the flags?

The lab is done! ☺ If you could not complete the lab during the scheduled lab time … that's OK … finish up the lab on your own time.

Before you leave the lab (even if you could not complete the lab), show the TA what you have completed. [Attendance marks are associated with this.]

Before the due date for your lab section (i.e. does not have to be done during the scheduled lab): Prepare your report and place the single pdf in the unzipped Lab3 folder containing your lab work. Name the report Lab3Report.PDF (the folder should retain the name Lab3). Zip and submit the Lab3.ZIP folder to the appropriate place in your lab section's cuLearn page.