Lab 1: Intro to Logic Simulation

Due Friday, 19 April 2019, 11:59 PM

Minimum Submission Requirements

- Your Lab1 folder must contain the following files:
  - Lab1.lgi (note the capitalization for both the file name and extension)
  - README.txt
- Commit and push your repository

Note

You should have created the Lab1 directory and a blank README file in the last lab assignment.

Lab Objective

This lab will introduce you to a schematic entry logic simulation program, Multimedia Logic. In this lab you will practice creating truth tables and implementing logic based on those truth tables.

Tutorial

Before starting the lab assignment, follow the tutorial listed in the Help menu.

Resources

These YouTube videos might be helpful.

https://www.youtube.com/playlist?list=PL4CFA1D985CE6B2F7

https://www.youtube.com/watch?v=hJq2gECXYWc

For the Extra Credit

https://www.youtube.com/watch?v=rnQwucEfT6A&list=PLA8F3EE4391DF4D0A&index=10

Template

Start working from the template provided on canvas. You must use senders and receivers such that the output is shown on the first page of the template. **Do not change the first page of the schematic file**, except for the text fields - your name, CruzID, and descriptions of the outputs. Additional wires and logic circuits shall be drawn on subsequent pages of your Multimedia Logic schematic. **Remember to rename the template file** to Lab1.lgi.

Note

Some operating systems capitalize the extension and will save the file as Lab1.LGI. You must **rename this file** to have the extension .lgi (**in all lowercase**).

Specification

Part A

The given template has input switches *in_3, in_2, in_1,* and *in_0*. Considering the input switch *in_3* as the most significant bit, connect the wires from the user input switches to the 7 segment display component.

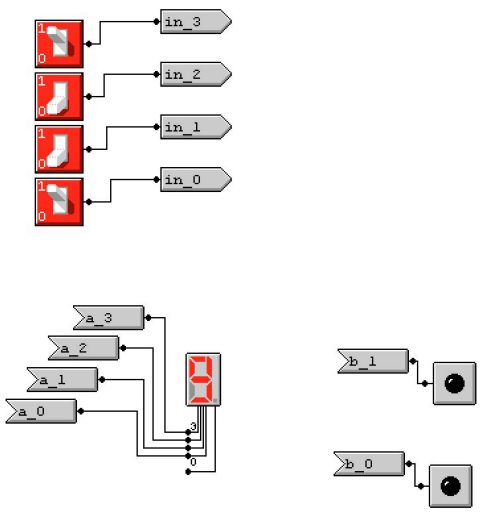
Part B

Consider the switches as a 4-bit binary number.

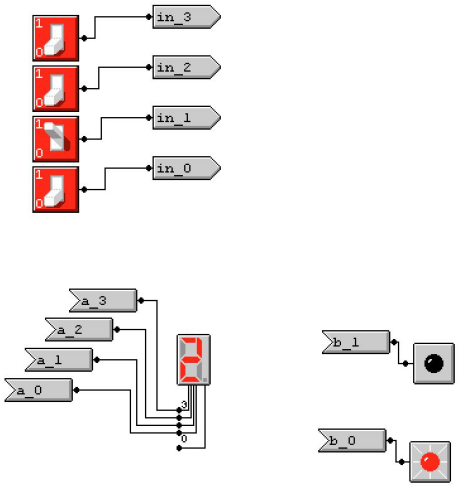
Design a circuit that turns ON the “b_0” LED when the switches indicate an even number.

Design a circuit that turns ON the “b_1” LED when the switches indicate a value of 12 (base 10) or more.

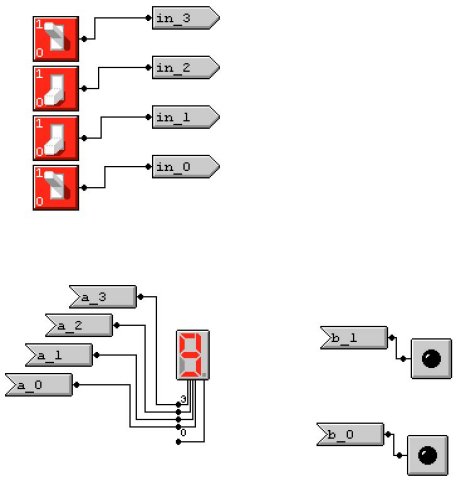
Sample Output



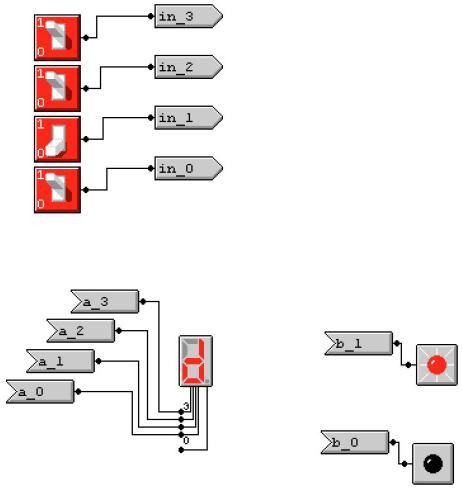
b_0 is OFF when the input is odd number



b_0 is ON when the input is even number



b_1 is OFF when the input is 0 - 11



b_1 is ON when the input is 12 - 15

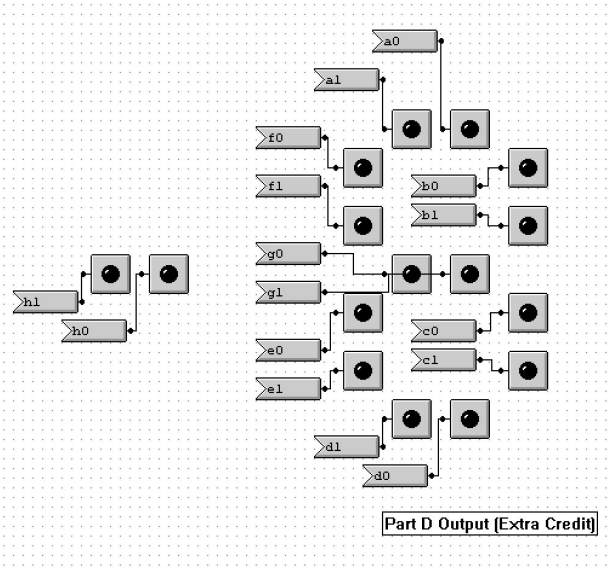
Part C

Implement the truth table below using either Sum of Products (SOP), Product of Sums (POS), or another method. Output the result on the LED connected to receiver *c_0*. Then, implement the same truth table using only NAND gates. Output this result to the LED connected to receiver *c_1*. Assume an ON LED represents "1" and an OFF LED represents "0." Lastly, build the same circuit from using only NOR gates. Output the result to *c_2*.

in_2	in_1	in_0	c_0
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Extra Credit

In the template, you are given a 7 segment display made of individual LEDS on the second page. Implement the logic required to have this display represent the two's complement number based on user input.



Simulation

To ensure the circuit simulates without error, make sure there is **at least one receiver for every sender** and that **each receiver has exactly one sender**. In addition, **do not modify the canvas size**.

If you do not have at least one receiver for every sender or if you have more than one sender with the same name, your circuit will not simulate and you will miss all points for the output meeting the specification.

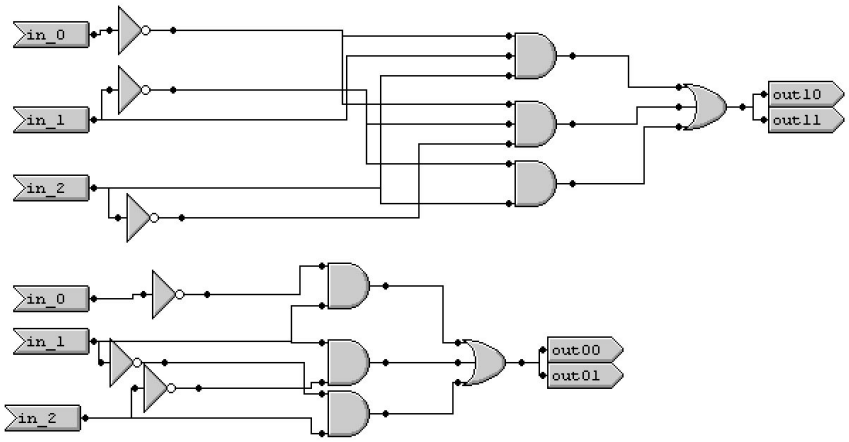
Comments

Each page of your Multimedia Logic schematic should be labeled with your last name, first name, and CruzID (the name used in your UCSC email address). Label each circuit with a description of the functionality and the part of the lab that it is for.

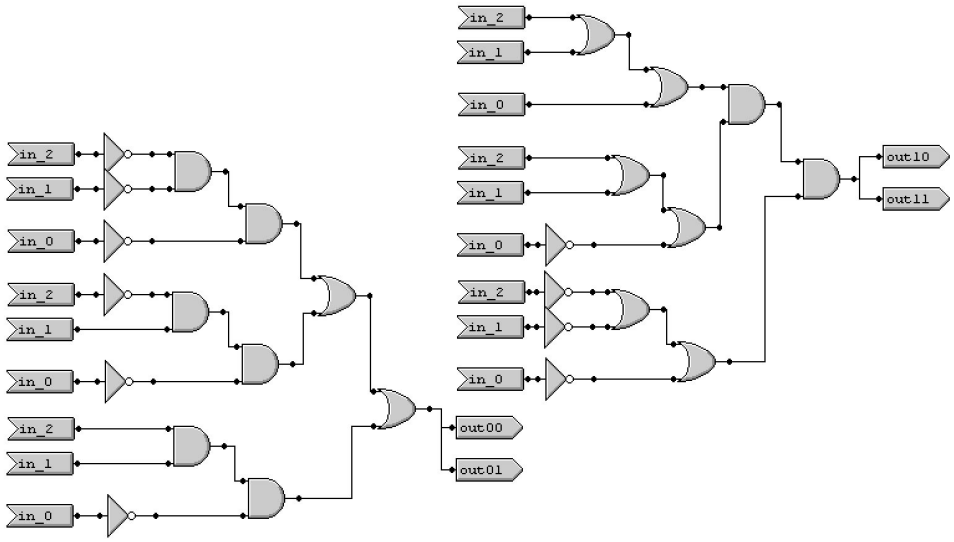
Visual Structure

Presentation of information is an important part of deliverables. **Clean documentation** is easy to comprehend and looks professional. Your circuits should be structured in an organized method that is easy to read and interpret. Using the "Snap to Grid" setting under the View menu makes it easy to line up components. A clean circuit uses many senders and receivers with meaningful names, and has no wires crossing over each other. Note that there may be multiple receivers for one sender. See below for examples of messy and clean circuits.

Messy Circuit Example



Clean Circuit Example



README.txt

This file must be a plain text (.txt) file. For full credit, it should contain your first and last name (as it appears on Canvas) and your CruzID. Your answers to the questions should total at least 150 words. Your README should adhere to the following template:

```
------------------------
LAB 1: Intro to Logic Simulation
CMPE 012 Fall 2018

Last Name, First Name
CruzID
-------------------------

What did you learn in this lab?
Write the answer here.
```

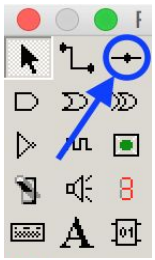
What worked well? Did you encounter any issues?
Write the answer here.

How would you redesign this lab to make it better?
Write the answer here.

What external resources did you use to complete this lab?
(Not including course materials)

Did you work with anyone on the labs? Describe the level of collaboration.

Missing Wire Best Practices



MML has a known bug which causes some wires to disappear during the save process. To reduce the likelihood of this occurring, **DO NOT use the "Node" tool** (it's a black dot located at the top-right of the tool palette). This tool is particularly vulnerable to the bug.

If this bug occurs, the grader will attempt to repair the missing wire in your file. This is only possible if your circuit is very readable. Make sure that wires do not cross whenever possible. Wire paths should be short and direct. **Use receivers liberally**.