

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](#) (MMLLogic or MML). In this lab you will practice creating truth tables and implementing logic based on those truth tables.

Tutorial

Before starting the lab assignment, follow this Tutorial (used to be available from the MML Help Menu):

Tutorial

This describes how to build a very simple circuit.

- Start a new circuit by clicking on  from the toolbar.
- Bring up the Tool Palette by clicking on  from the toolbar.
- Select the Switch by clicking on  from the palette.
- Place a switch by clicking the left mouse button on schematic.
- Select an LED by clicking on  from the palette.
- Place an LED by clicking the left mouse button to the right of the switch.
- Select the Wiring device by clicking on  from the palette.
- Click the left mouse button down on the output node (black dot) and hold it.
- Drag the mouse over to the LED input node (black dot) and let mouse go.
- Select the Pointer Tool by clicking on  from the palette.
- 
- Double click on the center of the LED
- Change the color to Yellow and Click OK.
- Disable the Tool Palette by clicking on  from the toolbar.
- Run the simulator by clicking on  from the toolbar.
- 
- Click on the Switch on the schematic and notice what happens.
- Stop the simulator by clicking on  from the toolbar.

This tutorial is designed to teach you the basics of using the simulator. It is not an attempt to teach logic design.

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 or second pages of the template 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.

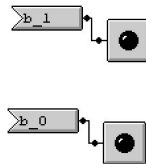
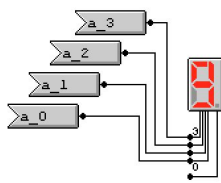
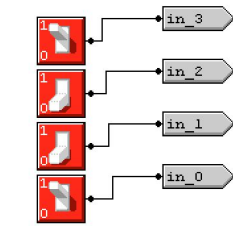
Part B

Consider the switches as a 4-bit binary number.

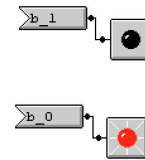
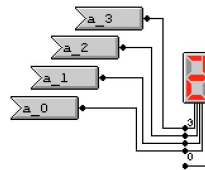
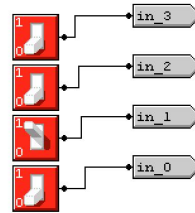
Design a circuit that turns ON the “b₀” LED when the switches indicate an even number.

Design a circuit that turns ON the “b₁” LED when the switches indicate a value of 12 (base 10) or more.

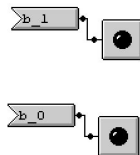
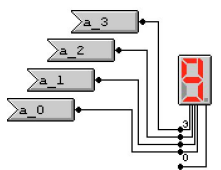
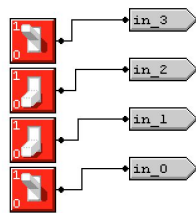
Sample Output



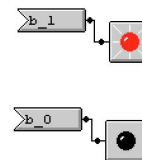
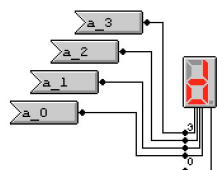
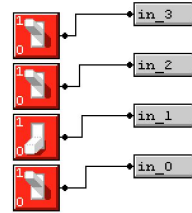
b₀ is OFF when the input is odd number



b₀ is ON when the input is even number



b₁ is OFF when the input is 0 - 11



b₁ is ON when the input is 12 - 15

Part C

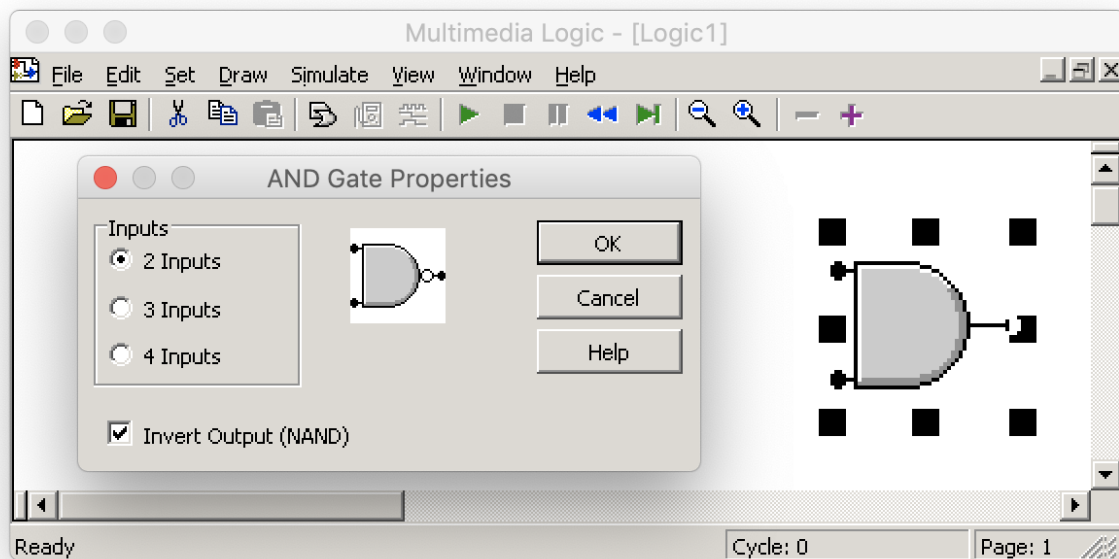
For Part C, you may ignore in₃.

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 c₀.

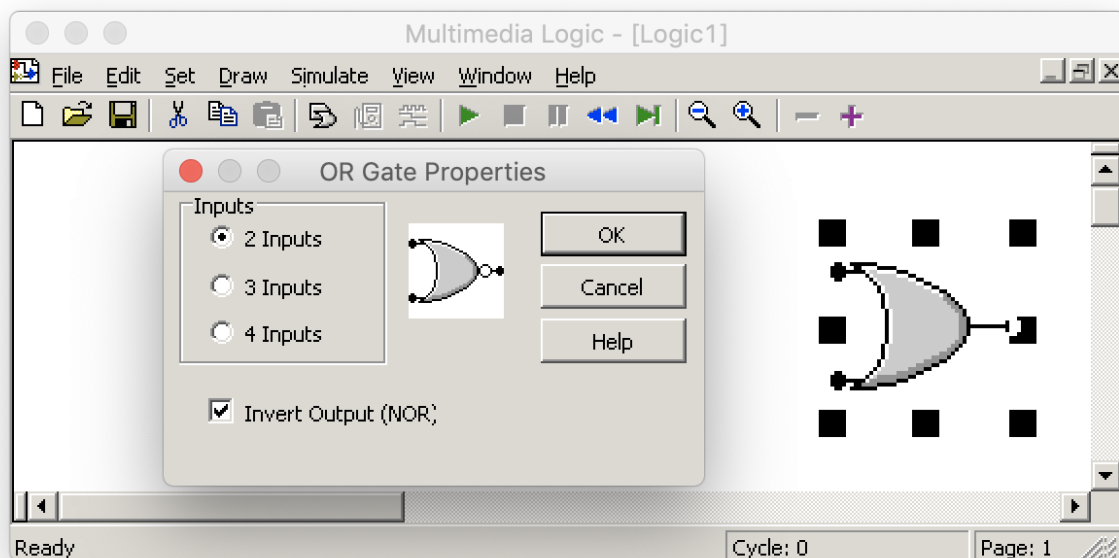
Then, implement the same truth table using only NAND gates. Output this result to the LED connected to receiver `c_1`. Assume an ON LED represents “1” and an OFF LED represents “0.”

Lastly, build the same circuit from using only NOR gates. Output the result to `c_2`.

You can create a NAND gate by double clicking on an AND gate which brings up the options:



You can create a NOR gate by double clicking on an OR gate which brings up the options:

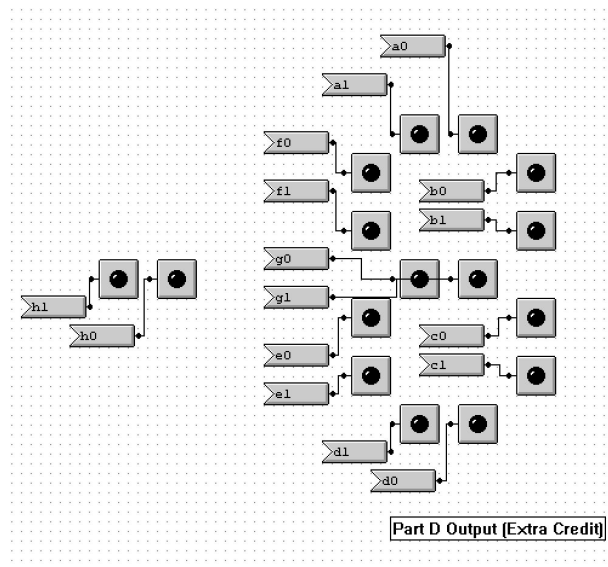


in_2	in_1	in_0	c_0
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Extra Credit

On the second page of the template, you are given a 7-segment display made of individual LEDs. Assume the inputs (in_3 through in_0) represent a 4-bit two's complement number. Implement the logic to display this two's complement number as a human readable, signed decimal number.

For example, if all switches are on (corresponding to the 4-bit binary number 1111), then the LEDs will read "-1" i.e. LEDs will be on for h1, h0, b1, b0, c1, c0.

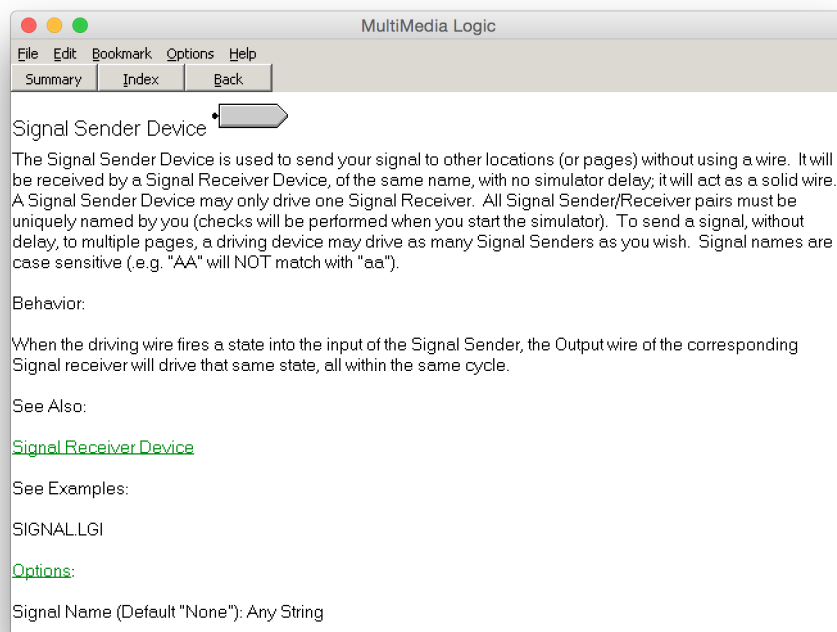
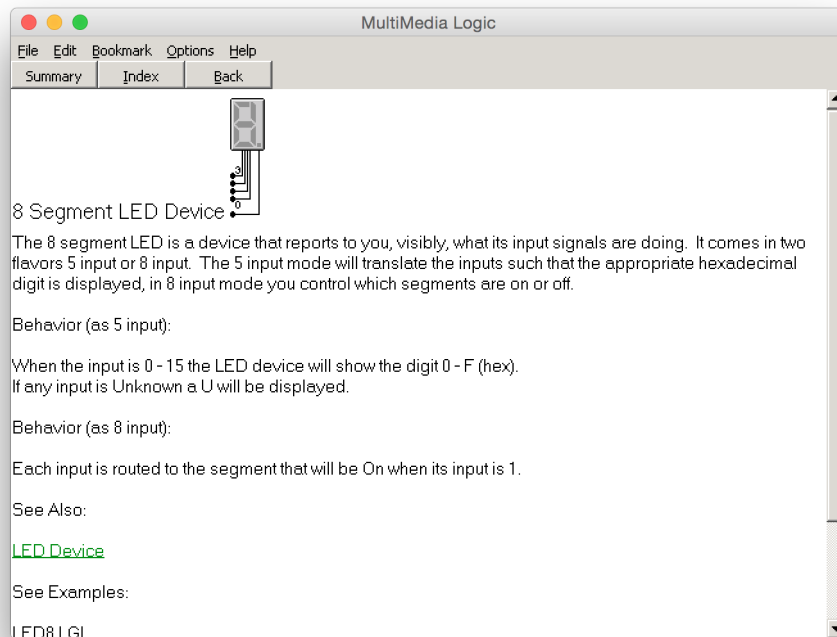


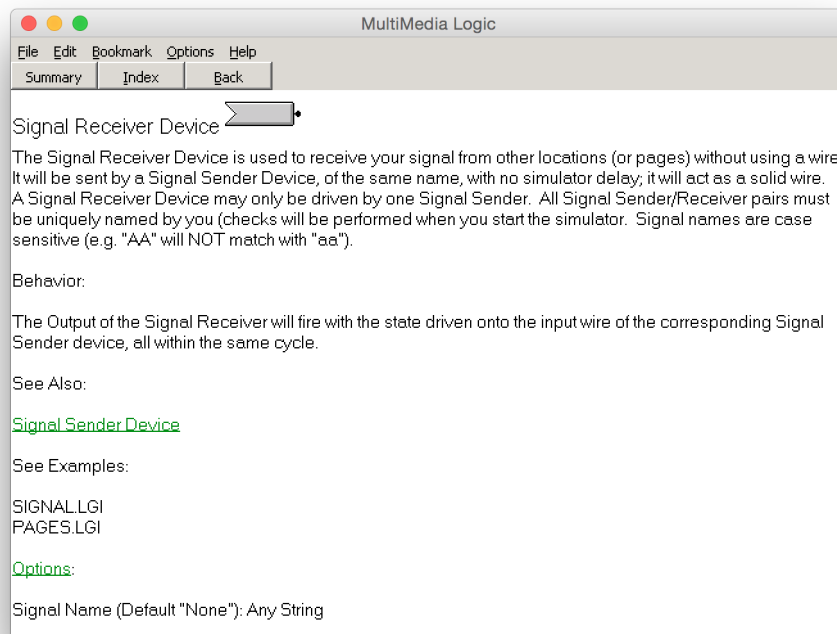
Simulation

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.

Here is some documentation from MML's help menu:





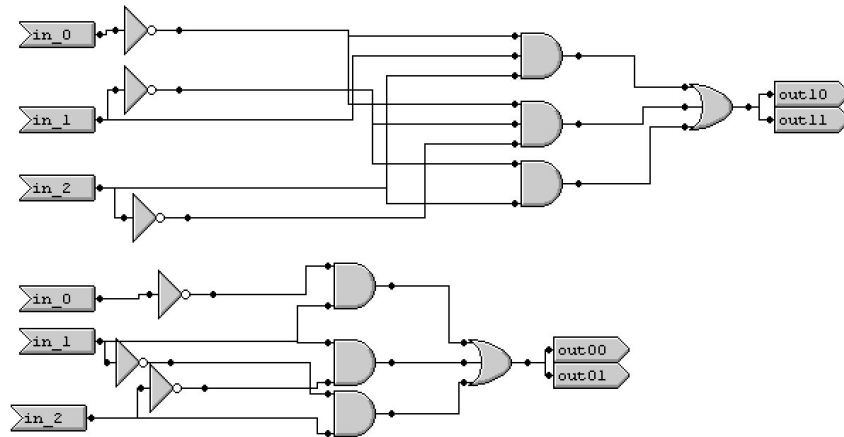
Comments

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.

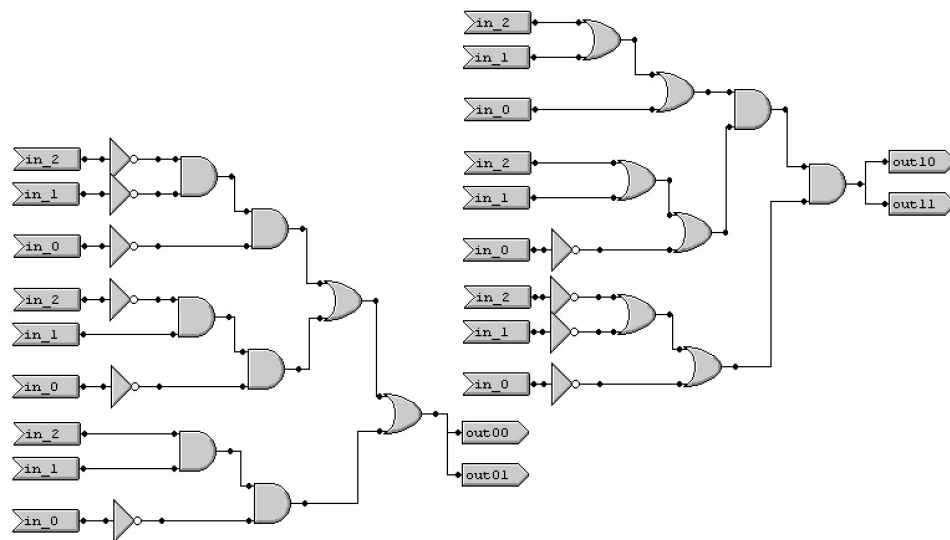
Visual Structure

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



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:

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

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.**



Grading Rubric

1 pt file name matches specification exactly

3 pt simulates without errors

NOTE: All senders must have at least one receiver and every receiver must have exactly one sender.

Do not resize the canvas.

11 pt output matches the specification

1 pt part A

4 pt part B

6 pt part C

NOTE: Credit for output only if circuit simulates without error.

1 pt complete header comments on every page of schematic and README

1 pt useful & sufficient comments

1 pt clean visual structure / use of white space

2 pt README file with at least 150 words

4 pt extra credit

2 pt input values 0000 through 0111

2 pt input values 1000 through 1111

-5 pt if first or second page of template is modified