Lab 1: Intro to Logic Simulation

Due Monday, 18 January 2021, 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
- Complete the Google Form with the correct commit ID of your final submission

Note

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

Objective

This lab will introduce you to a schematic entry logic simulation program, Multimedia Logic (MMLogic 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 on 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 the 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 the 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.

- 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.

Resources

Browse the Index in the help menu in MML:
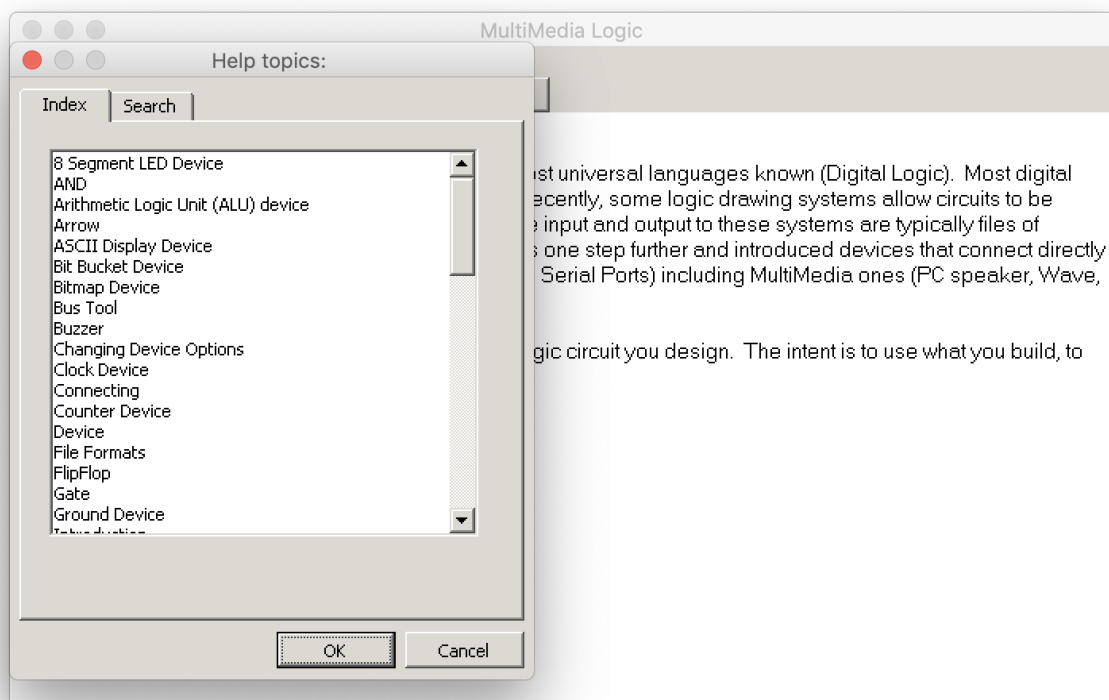
These YouTube videos might be helpful.

Digital Logic

Multimedia Logic - Tutorial 1 - Simple Circuits

Winter 2021

Template

Use 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 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.

Additional wires and logic circuits shall be drawn on subsequent pages. **Remember to rename the template file** to Lab1.lgi before committing to your repository.

On the third page, there are placeholder signal senders and receivers that you can use. You may remove these senders and receivers as you use them in your design.

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**). The first letter of Lab1 should be capitalized.

Specification

For all parts of the design, assume an ON LED represents "1" and an OFF LED represents "0." Consider the input switches as a 4-bit binary number with input switch in_3 as the most significant (left-most) bit and input switch in_0 as the least significant (right-most) bit.

Part A

Connect the wires from the user input switches to the 7 segment display component.

Part B

Design a circuit that outputs the three bits shown in the truth table given the four bits as input. The three output bits have b_2 as the left-most bit and b_0 as the right-most bit.

For example, for input = 1111, the output should be 010 which means

b_2 = 0 = OFF

b_1 = 1 = ON

b_0 = 0 = OFF

Do you notice any patterns? You can use these patterns and Boolean algebra or intuition to simplify your circuit.

in_3	in_2	in_1	in_0	b_2	b_1	b_0
0	0	0	0	0	0	0
0	0	0	1	1	1	0
0	0	1	0	1	0	0
0	0	1	1	0	1	0
0	1	0	0	0	0	0
0	1	0	1	1	1	0
0	1	1	0	1	0	0
0	1	1	1	0	1	0
1	0	0	0	0	0	0
1	0	0	1	1	1	0
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	1	1	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

Part C

For Part C, you may ignore input in_3. Implement the truth table below using either Sum of Products (SOP) or Product of Sums (POS). Output the result on the LED connected to receiver c_0.

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

Next, implement the SOP or POS using only NAND gates. Output this result to the LED connected to receiver c_1.

Lastly, build the same SOP or POS 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:
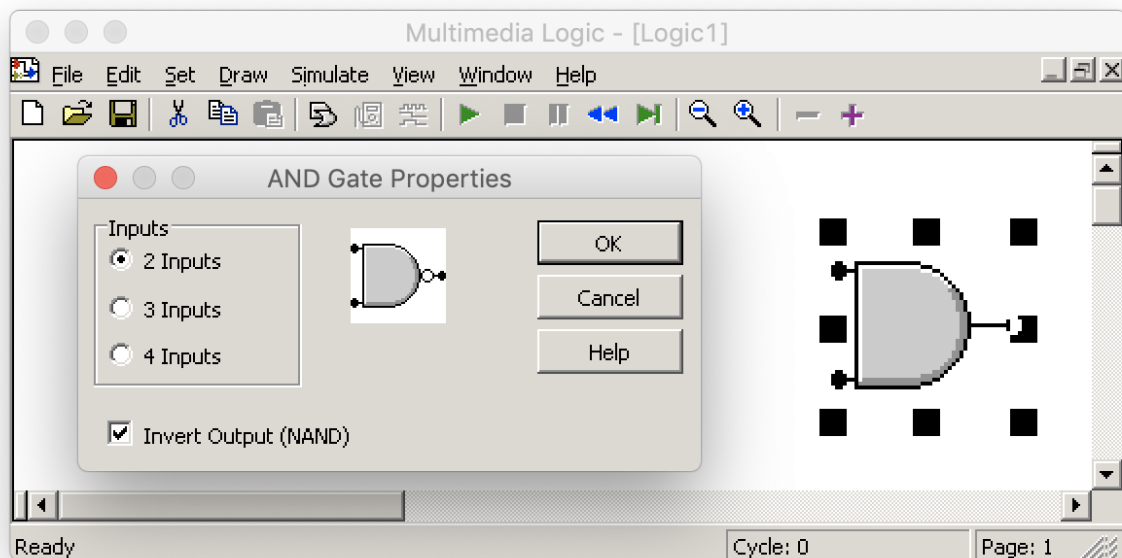
You can create a NOR gate by double clicking on an OR gate which brings up the options:
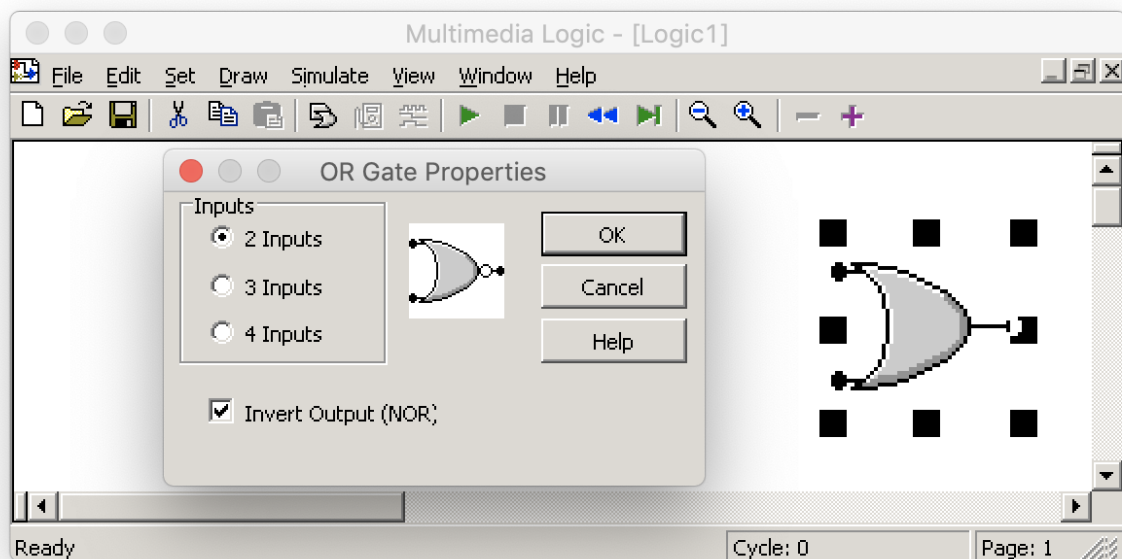
Documentation Standards
Follow the documentation guidelines found here. Refer to the sections on the README and schematic visual structure. Diagram.pdf is not required for this lab.

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 lose points.

Google Form
You are required to answer questions about the lab in this <u>Google Form</u>

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 senders and receivers liberally**.

Grading Rubric (40 pt total)
```
 4 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.

```
26 pt  output matches the specification
        6  pt  part A
       10 pt  part B
       10 pt  part C

 2 pt  complete header comments on every page of schematic
 2 pt  useful & sufficient comments
 2 pt  clean visual structure / use of white space
 2 pt  README.txt file complete
 2 pt  Google form complete with at least 150 words

-2 pt  for using AND + NOT gates instead of NAND gates
-2 pt  for using OR + NOT gates instead of NOR gates
```

```
-10 pt  if first or second page of template is modified incorrectly, or
        if the template isn't used properly
-10 pt  incorrect naming convention (e.g .LGI or lab1.lgi)
```

Appendix

Here is some documentation from MML's help menu:

MultiMedia Logic

Ground Device

The Ground Device is fixed output device that always generate a 0. You can use one ground to ground many inputs.

Behavior:

   This device always outputs a 0.

See Also:

Plus Device
Switch Device

See Examples:

MEMORY.LGI

Options:

None:

MultiMedia Logic

8 Segment LED Device

The 8 segment LED is a device that reports to you, visibly, what its input signals are doing. It comes in two flavors 5 input or 8 input. The 5 input mode will translate the inputs such that the appropriate hexadecimal digit is displayed, in 8 input mode you control which segments are on or off.

Behavior (as 5 input):

When the input is 0 - 15 the LED device will show the digit 0 - F (hex).
If any input is Unknown a U will be displayed.

Behavior (as 8 input):

Each input is routed to the segment that will be On when its input is 1.

See Also:

LED Device

See Examples:

LED8.LGI

Winter 2021

Signal Sender Device

The Signal Sender Device is used to send your signal to other locations (or pages) without using a wire. It will be received by a Signal Receiver Device, of the same name, with no simulator delay; it will act as a solid wire. A Signal Sender Device may only drive one Signal Receiver. All Signal Sender/Receiver pairs must be uniquely named by you (checks will be performed when you start the simulator). To send a signal, without delay, to multiple pages, a driving device may drive as many Signal Senders as you wish. Signal names are case sensitive (.e.g. "AA" will NOT match with "aa").

Behavior:

When the driving wire fires a state into the input of the Signal Sender, the Output wire of the corresponding Signal receiver will drive that same state, all within the same cycle.

See Also:

Signal Receiver Device

See Examples:

SIGNAL.LGI

Options:

Signal Name (Default "None"): Any String

Signal Receiver Device

The Signal Receiver Device is used to receive your signal from other locations (or pages) without using a wire. It will be sent by a Signal Sender Device, of the same name, with no simulator delay; it will act as a solid wire. A Signal Receiver Device may only be driven by one Signal Sender. All Signal Sender/Receiver pairs must be uniquely named by you (checks will be performed when you start the simulator. Signal names are case sensitive (e.g. "AA" will NOT match with "aa").

Behavior:

The Output of the Signal Receiver will fire with the state driven onto the input wire of the corresponding Signal Sender device, all within the same cycle.

See Also:

Signal Sender Device

See Examples:

SIGNAL.LGI
PAGES.LGI

Options:

Signal Name (Default "None"): Any String