**University of California, Santa Cruz**
**Board of Studies in Computer Engineering**

**CMPE012/L: Computer Systems and Assembly Programming**

# Lab 2: Introduction to Logic with MultiMedia Logic

**Minimum Submission Requirements:**

- Lab2_tutorial.lgi in the lab2 folder
- Lab2.lgi in the lab2 folder
- README.txt in the lab2 folder
- Commit and Push your repo
- A Google form submitted with the Commit ID taken from your GITLAB web interface verifying that the above files are correctly named in their correct folders
- All of the above must be completed by the lab due date

**Lab Objective:**

This lab introduces you to a logic simulation program, Multimedia Logic, for Windows to do schematic entry and simulation. This is an excellent tool for getting an intuitive handle of the logic and gate level of computer circuits, and get a sense of timing and execution. You will also gain some experience designing combinatorial logic circuits, and using combinatorial logic as part of more complex circuits.

You will be using Multimedia Logic (MML) to simulate a number of circuits and logic gates of increasing complexity (very simple at this stage). Note that MML is a free program and you can download it from the softronix website at: http://www.softronix.com/logic.html.

You will be graded on the lab report (in your README.txt) and the functionality of the files submitted through the usual submission process.

**Lab Preparation:**

Read this document in its entirety carefully. Review basic logic gates (Chapter 3 of the Patt and Patel Reader, Appendix B of Patterson and Hennessy, up to B-4) and make sure you understand them. If you are having difficulty, supplement your reading with online tutorials on logic gates.

**Lab Specifications:**

You will be turning in MML files (.lgi extension) with the results of your explorations in this lab. Each and every file should have the following block comment at the top of the first page:

- Your name and email@ucsc.edu
- Lab number and title
- Due date
- Your section (e.g.: 01A), your section TA/tutor

Subsequent pages should have a header indicating the part of the lab that the page contains (e.g.: Part C).

Make sure that you comment your schematics (pages) to help the graders and yourselves to understand what is going on; include a brief description of the circuit (e.g.: "Implementation of a truth table"). At the very minimum label the page and the various inputs and outputs.

For this lab, you will write up your lab report within your README.txt file. This does not need to be very long, but will be a significant part of the grade; we don't break down the points for each but assess the entire write-up as a whole. Ensure that it is readable, and that it has some sensible structure. Describe what you learned, what was surprising, what worked well and what did not. Lab reports shouldn't need to be long. One page is sufficient.

Your lab report (README.txt) should include:

- Your name, email, ID, lab assignment, and section number (see headers above)
- How would you make your own 7-segment display from Part B if you didn't have one in MML?
- How do you think the random number generator works?
- How can things be really random in a computer when it is made of logic gates, which are supposed to be deterministic?

To alleviate file format issues we want lab reports in plain text. Feel free to use a word processor to type it up but please submit a plain text file *and CHECK IT ON GITLAB to make sure it is readable*.

## PART A: TUTORIAL

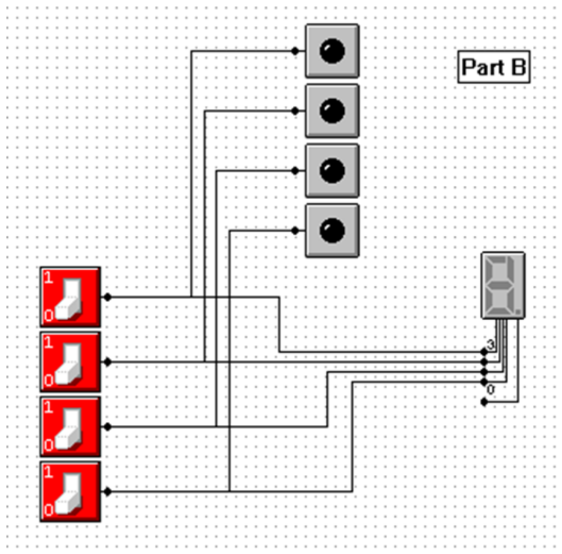
Do the tutorial for Multimedia Logic (Help->Tutorial) or look at the MML_Tutorial.pdf provided on CANVAS. This simple tutorial will walk you through building and simulating simple circuit, save the resulting file as Lab2_tutorial.lgi. Use the "Text" tool to put the required comments on your schematic.

## PART B: PLAYING WITH NUMBERS

Create a new MML file as Lab2.lgi and add the appropriate header comments.

Now, build the schematic below. See if you can make it cleaner than the one given. Simulate the circuit and play around with the switches, getting a feel for inputting a binary number and seeing the result on the display.

*The schematic below is not complete until you label your inputs and outputs*



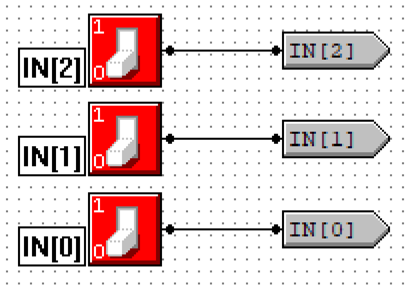
## Part C: Truth Table to Gates

Use the "+" button to add a new page to your MML file, and add the appropriate header comments.

Design logic that implements the following truth table.

| IN[2] | IN[1] | IN[0] | Output |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

Connect your circuit's output to an LED to verify your circuit works correctly. Be sure to test your logic afterwards. Make sure that for all inputs the correct output is on the LEDs. Use sender/receiver pairs to direct the input to the rest of your circuit. Note that each sender must have at least one receiver, and that the receiver must have the exact same name as the sender. The left side of your circuit should like this:



You may use any circuit topology you like: sum-of-products, product-of-sums, or a gate-reduced topology of your own. Whatever topology you choose, be sure you can reproduce your design process.

## Part D: Guessing Game

Use the "+" button to add a new page to your MML file, and add the appropriate header comments.

Now you will create a fun guessing game in logic! Create a design on a new page labeled PART D that allows the user to play "guess the number." In this game, the user presses a momentary pushbutton which generates a random 2-bit number. Then, the user can switch two toggle switches to attempt to guess that number. When the user is correct, an LED lights up.

Your circuit will need the following components:

- The random number generator circuit element. You only need to connect two of the outputs (any two will do), the outputs are the pins on the right side of the box.
- A push-button switch to drive the random number "generation". Connect the switch to "C+."
- Two toggle switches for user input, this will be the "user guess."
- Combinational logic to test for equality.
- An LED to indicate if the user's guess was correct or incorrect.

Basically, you are checking if the output of the random number generator (2 bits of it) is equal to the number on the toggle switches put in by the user. If you are having trouble figuring this out, write out the truth table. As a hint, think logical AND.

You might want to (optionally) use two LEDs to indicate the output of the random number generator. This makes for a boring game, but might make it easier to debug your circuit.

## NOTES:

This lab is using a new program, so it can take some time. Start early and there should be no issues with finishing it well before the due date. We will be using this software for subsequent labs, so get to know it.

Ensure that you have met the minimum submission requirements. Failure to meet this requirement will result in failing the class regardless of any other scores. You can submit a new commit ID as many times as you like, we will always grade the last one you submit on the google form. Also note that until you push, the files don't exist in a place where we can get to them. This is how the program knows which files to pull from your repo. This is why we need the commit ID to match what you want to turn in.

One reminder about committing and pushing: You should get in the habit of committing early and often. This is very easy to do, and guarantees that you have something turned in. Every time you make a change, add a file, modify a file, go ahead and commit and push. You will always be able to return to that place and never lose your work.

Lastly, when submitting on the google form, ensure that YOU are logged into google using your @ucsc.edu account. The form captures the email that is there and if it is someone else's, then your entry into the google form will not be valid.