

ECE 206, Fall 2001: Lab 1

Digital Logic

Learning Objectives

This lab will give you practice in building and analyzing digital logic circuits. You will use a logic simulator to implement circuits and see how they behave. The circuits include both basic gates (AND, OR, NOT) and higher-level functional blocks (decoder, multiplexer, adder).

The Logic Simulator

A logic simulator allows you to build a virtual logic circuit on a computer, so you can see how it works before building an actual hardware circuit.

The simulator considers only the *logical behavior* of a circuit. It ignores physical characteristics, such as the time it takes a signal to get from one end of a wire to the other. Therefore, a logic simulator is a good tool to know if your circuit is correct, but it won't tell you how fast your circuit might be.

For this lab, we will use a public-domain logic simulator called MultiMedia Logic (MML). It is installed on the lab computers, but you can also download it from the lab web page if you want to play at home. It runs on Windows 95/98/2K/NT machines.

Before coming into the lab, read the short introduction to MML posted on the lab web site. You'll need this to answer the questions in Prelab Exercise 1. This material is extracted from MML's on-line help files, so you don't need to print this out to bring to lab unless you want to.

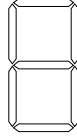
For your circuits, you will use the following MML components:

- Toggle switch – allows you to change inputs to 1 or 0.
- LED – simple output device that turns on when its input is 1.
- Wiring Tool – used to connect components to each other.
- AND, OR, NOT – basic logic gates. AND and OR can take up to 4 inputs, and you can invert the output to create NAND and NOR gates. (Double-click or right-click on the component to set its options.)
- Text Tool – used to add text labels to your circuit diagram.

You might also want to use the Node Tool, which helps you control how wires are drawn. The Wiring Tool just connects the points and draws the wire however it wants. Instead, you can connect an output to a node, then connect that node to another component or node, etc.

Background: Seven-Segment Display

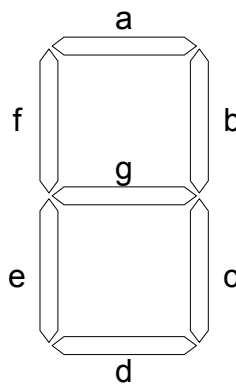
In the second laboratory exercise, you will be building a circuit to drive a “seven-segment display.” This device consists of seven LED bars that are illuminated to form digits and letters. You should be familiar with the pattern from clock radios, calculators, etc.



The seven LED's are arranged like this: We can make numbers by filling in (or turning on) one or more of the segments. Here are the 7-segment representations of the digits 0 through 7:



The segments are named ‘a’ through ‘g’, where ‘a’ is the top segment, ‘b’ is the top right, and so on, as shown in the following diagram.



To display a certain digit, we need to specify which segments should be turned on. For example, to display the digit “7”, we must turn on **a**, **b**, and **c**. To display the digit “4”, we must turn on **b**, **c**, **f**, and **g**.

Suppose we have a three-bit unsigned binary number $n[2:0]$, representing the values from 0 to 7. For each segment in the display, we can build a logic circuit that tells whether that segment should be turned on to display $n[2:0]$. In Prelab Exercises 2 and 3, you will create a truth table and derive the logic equations for each segment.

Instead of the “raw” three-bit number, suppose we use a 3-to-8 decoder¹ to generate signals d_0, d_1, \dots, d_7 . In Prelab Exercise 4, you will derive the logic equations for each segment using d_0 - d_7 , rather than $n[2:0]$. These are the equations that you will actually use in Lab Exercise 2

¹ See Section 3.3.1 in the text.

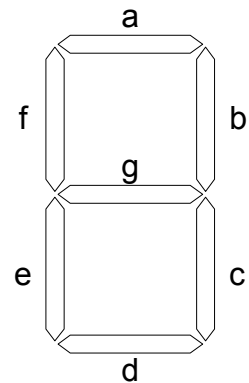
Prelab Exercises (30 points)

*You must turn in the prelab exercises at the beginning of the lab period.
Make a copy of the equations in (4) for yourself before the lab begins.*

1. (6 pts) Read the introduction to MultiMedia Logic posted on the Lab web site.
Answer the following questions:
 - (a) Name three details of a circuit implementation that are ignored by this logic simulator.
 - (b) Why are three states (True, False, and Unknown) used by the logic simulator, rather than two?
 - (c) List the steps needed to connect two components together in the MultiMedia Logic simulator.

2. (8 pts) Fill in the truth table below for the seven-segment display.
The first row is done for you.

| n | n[2] | n[1] | n[0] | a | b | c | d | e | f | g |
|---|------|------|------|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | | | | | | | |
| 2 | 0 | 1 | 0 | | | | | | | |
| 3 | 0 | 1 | 1 | | | | | | | |
| 4 | 1 | 0 | 0 | | | | | | | |
| 5 | 1 | 0 | 1 | | | | | | | |
| 6 | 1 | 1 | 0 | | | | | | | |
| 7 | 1 | 1 | 1 | | | | | | | |



3. (8 pts) Based on the truth table above, write the logic equation (using $n[2]$, $n[1]$, and $n[0]$ as inputs) for each of the seven segments.

a =

b =

c =

d =

e =

f =

g =

4. (8 pts) Now assume we have a 3-to-8 decoder, as described in the text, with outputs $d0$ - $d7$. Output $d0$ is 1 if $n[2:0] = 000$; $d1$ is 1 if $n[2:0] = 001$, etc.

Write the equation for each of the seven segments, using $d0$ - $d7$ as inputs.
The first one is done for you.

Try to use the minimum number of terms in your equations. Note that it may be easier to use a NOR gate to combine cases in which the segment is *not* turned on.

$$a = d0 + d2 + d3 + d5 + d6 + d7 = \overline{d1 + d4}$$

b =

c =

d =

e =

f =

g =

Lab Exercises (70 points)

NOTE: This lab is due at the end of the lab period. You **must** turn in your truth table and save your circuit files on your lab instructor's floppy to get credit for this lab.

If you have a lab partner, you can turn in your work together. Make sure both of your names are on the truth table and on each circuit that you submit.

1. Logic Analysis (10 points)

First start the MultiMedia Logic program – your lab instructor will tell you how.

Open the circuit file called “lab1-1.lgi” in the “Lab1” directory.

Immediately, save the circuit to a different file on the Desktop. (Ask your lab instructor if you don't know how.) This is because the file is read-only, but the simulator wants to modify the file when it runs.

This circuit uses a memory element to implement an arbitrary logic function with one output. Your task is to figure out the logic function.

Simulate the circuit (by pushing the “play” button) and toggle the switches controlling the A, B, and C inputs. Notice how the output LED changes for each combination of inputs.

(5 pts) *On a separate sheet of paper*, write the truth table for this function.

(5 pts) Underneath the truth table, write a logic equation that corresponds to the truth table.

Stop the simulation (using the “stop” button) and close the circuit without saving it.

Turn in your truth table and equation to the lab instructor before leaving.

2. DeMorgan's Law (10 points)

Create a new circuit (using the “New Project” button).

Build a circuit that demonstrates DeMorgan's Law, using two inputs and two outputs.

One output LED shows $A + B$; the other shows $\overline{\overline{A} \cdot \overline{B}}$.

Demonstrate to the lab instructor that both outputs are always equal, for all input values. Have the instructor make a note on your truth table (Exercise 1) that this circuit was completed.

When the lab instructor has signed off, delete the circuit (project) without saving it.

3. Creating a Logic Circuit (30 points)

Open the circuit file called “lab1-2.lgi” in the “Lab1” directory.

Immediately save the circuit to a different file on the Desktop.

This is the file that you will use to implement the 7-segment display described above. On the left are the three input bits, $n[2:0]$. Note that the most significant bit ($n[2]$) is on the bottom and the least significant bit ($n[0]$) is on the top. These inputs are fed into a 3-to-8 decoder, resulting in eight signals, D0-D7.

On the right of the circuit is the seven-segment display. We have used a pair of LED's to approximate each LED bar of the display. A node is provided for each output signal (a-g), and these nodes are already connected to the appropriate LED's.

Using the equations you developed in Prelab Exercise 4, create the logic circuit to display the input number ($n[2:0]$) on the seven-segment display. Simulate your circuit to make sure it works correctly. (You don't have to wait until the entire circuit is complete before simulating.)

Add your name (or both names, if you have a lab partner) to the bottom right corner of the circuit, using the Text Tool.

Save the circuit (to your newly-created file) and close it. When you're done with the next exercise, copy both circuit files onto the lab instructor's floppy disk.

4. Using a Multiplexer (15 points)

Open the circuit file called "lab1-3.lgi" in the "Lab1" directory.

Immediately, save the circuit to a different file on the Desktop.

This circuit will give you practice in using a multiplexer² to choose between multiple inputs. To the left is a hexadecimal input pad. Clicking on one of the buttons will cause the four output signals to take on the corresponding digit. Note that the least significant bit (0) is on the bottom, and the most significant bit (3) is on the top.

On the right is a collection of 2:1 multiplexers, one for each bit. The outputs of the multiplexers are connected to LED's and to a seven-segment display.

Add logic gates to the circuit to implement the following function:

```
if the input number is even,  
    perform a one-bit logical left shift;  
else,  
    take the one's complement (flip all bits).
```

You'll need to provide the logic to compute each function, and also the logic to make the decision about which result to pass through the mux. Simulate your circuit to make sure it works properly.

Add your name (or both names, if you have a lab partner) to the bottom right corner of the circuit, using the Text Tool.

² See Section 3.3.2 of the text.

Save the circuit (to your newly-created file) and close it. Copy both circuit files (from 2 and 3) onto the lab instructor's floppy disk.

5. Cleanup (5 pts)

Delete your files from the desktop. If you accidentally overwrote or deleted any of the circuit files used in the lab, get the lab instructor to restore them.

If the instructor finds any of your files on the desktop, or finds that lab circuit files have been deleted or altered (without notifying the instructor), you will not get credit for these 5 points.