Pranav Konduri
Mitchell Stacha
Cora English

CSCE 312-201

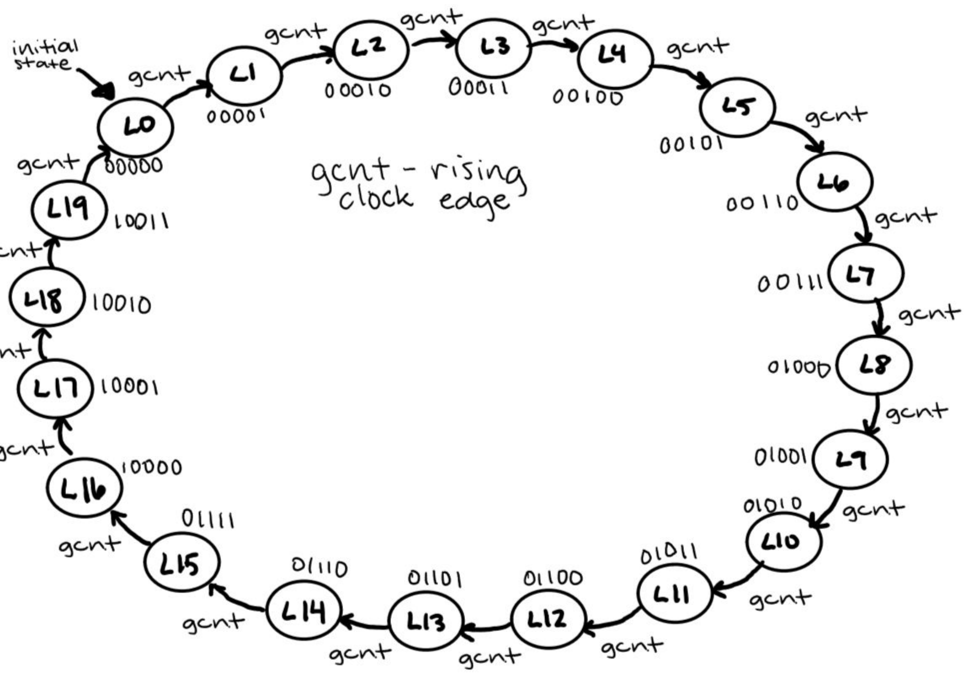
Lab 4 Report

March 5, 2020

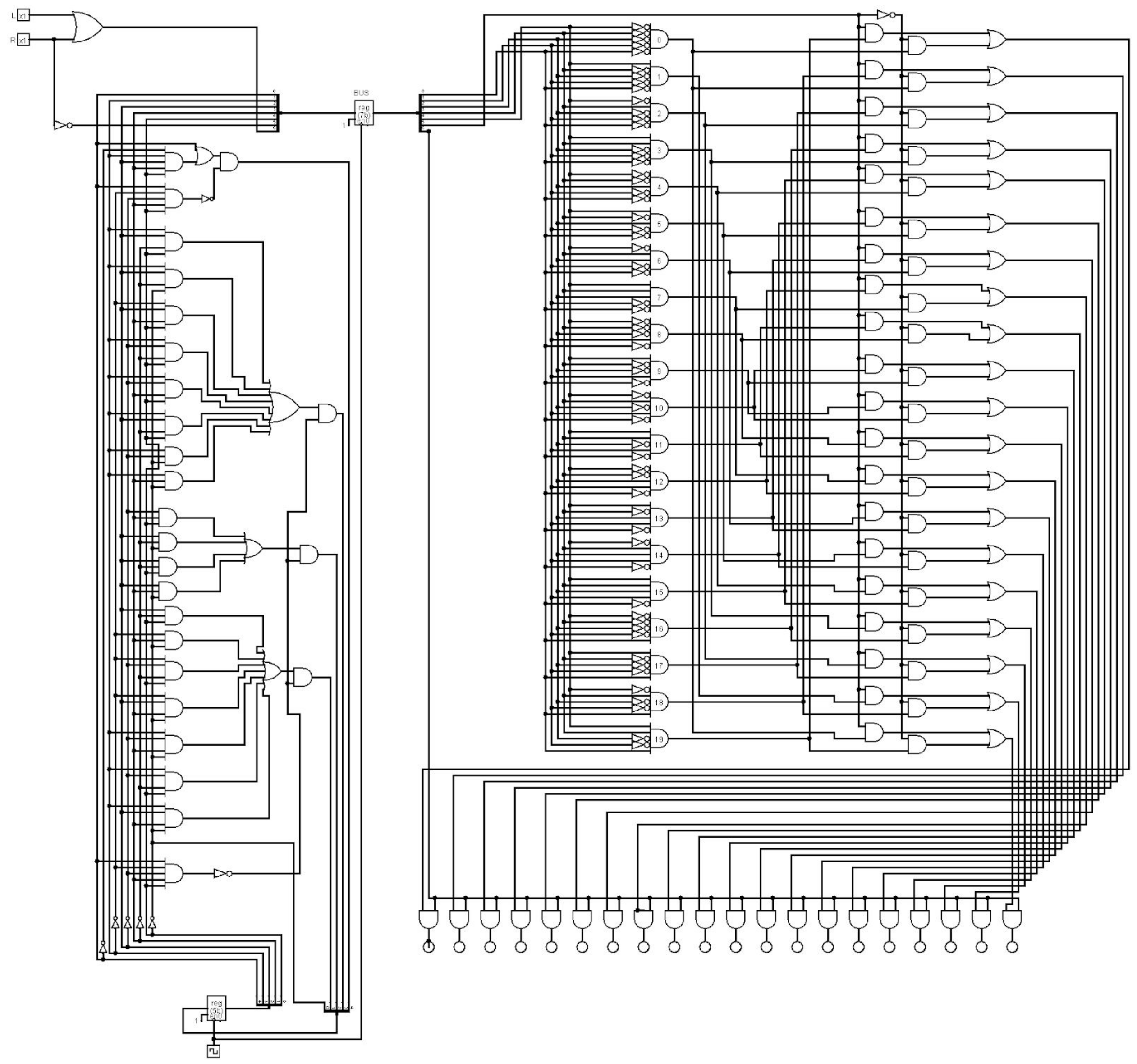
Problem 1:

1. Draw the finite state machine representation of the required sequential circuit.

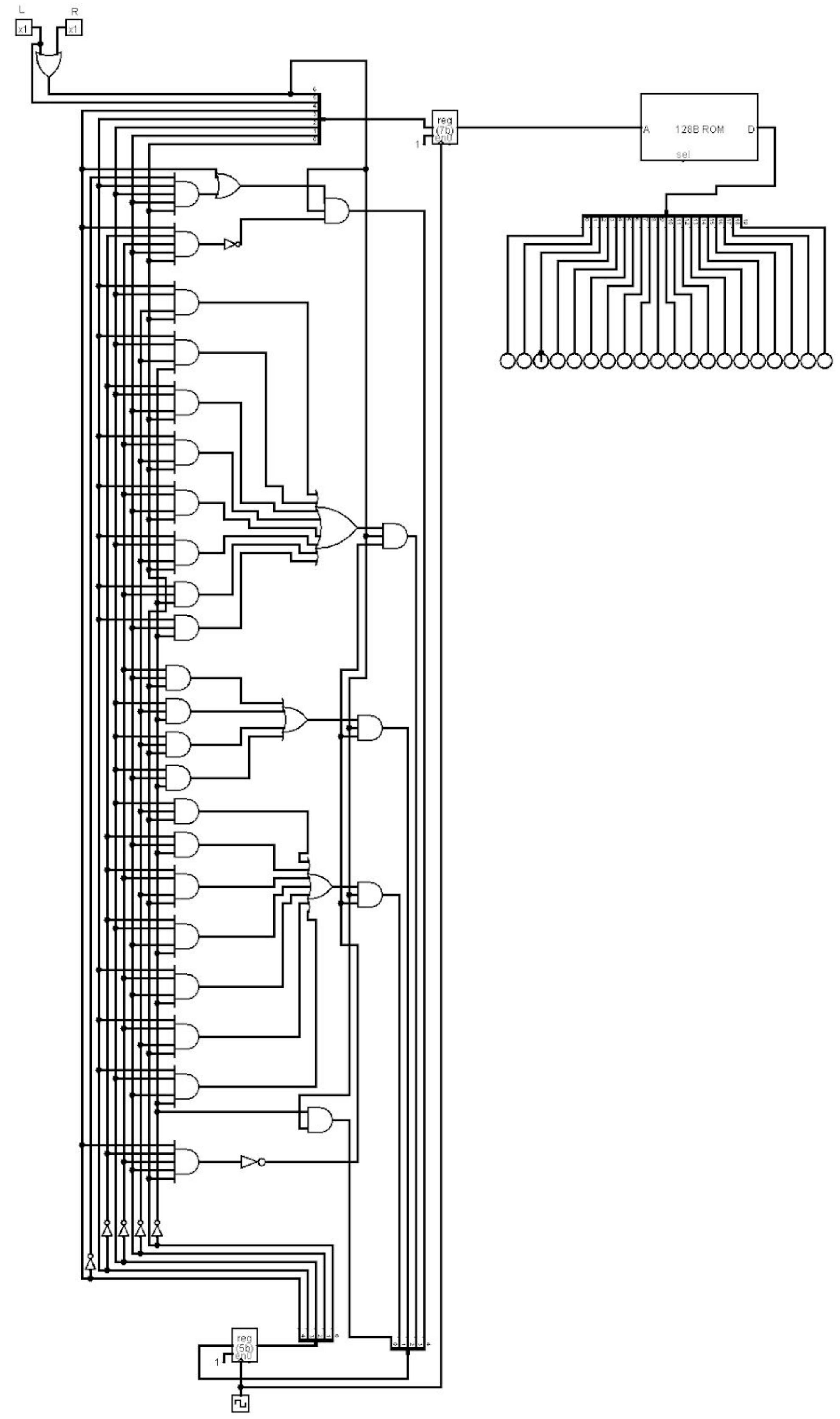
- Because of the way we implemented the FSM and the decoder to operate the lights, our FSM will only count up from 0 to 19 before repeating. The decoder will determine which direction the lights turn on (right or left), then AND gates will check whether an enable signal coming from the bus (which is the OR of right and left inputs) is true in order to turn on the corresponding turn signal light.



2. Use flip-flops and logic gates to implement the required circuit. The design should have a clearly identifiable data bus.



3. Instead of a decoder, use a ROM and re-implement the same circuit.



Problem 2:

1. Implement a unidirectional 8 bit address and control bus, and bidirectional 8 bit data bus, which are suitable for the 8 bit microprocessor (like Intel 8088).
2. Interface these buses with a ROM and RAM as found in Logisim, and verify the combined operation of these busses, ROM and RAM.
3. Design the IO system circuitry (an addressable I/O system) for the required microprocessor based system (see the tips below).
4. Integrate these busses to the designed IO system.
5. Provide brief text description, truth table and signal timing diagram to illustrate the operation of each sub-system or modules in your design.
6. The microprocessor will orchestrate the reading, writing from the digital and analog I/O system and the memory. These reading and writing operations are the tasks that have to be executed in the system. Write down the sequence of operations/tasks in- (see tip #13)

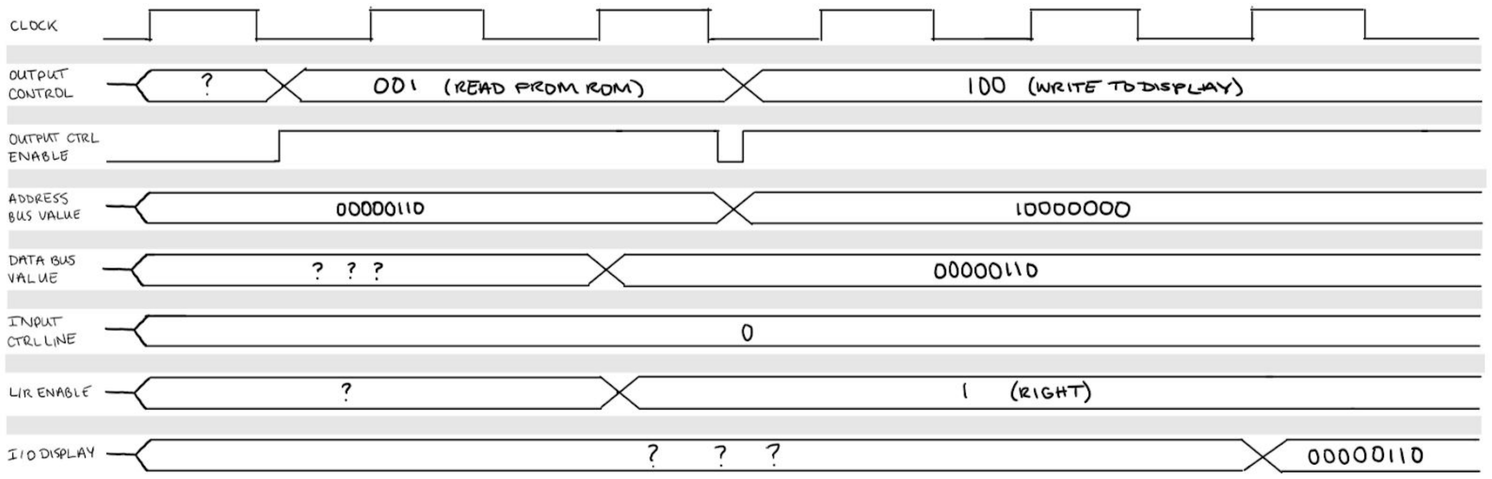
(I) Pseudo-code. (II) Task timing diagram for the following tasks (only do 1 detailed task):

a) Reading from ROM
   i) Read from in port with address
   ii) Pull the value from the ROM w/ given address into bidirectional data bus
   iii) Swap controls to writing to the output stream
   iv) I/O Writes the data out into the output pin.
b) Reading from RAM
   i) Read from in port with address
   ii) Pull the value from the RAM w/ given address into bidirectional data bus
   iii) Swap controls to writing to the output stream
   iv) I/O Writes the data out into the output pin.
c) Writing into RAM
   i) Swap control to read from user input/ROM
   ii) Store user input / ROM value into the bidirectional data bus
   iii) Swap to write into the RAM, given an address location
   iv) The RAM updates the data at the given address with the value stored in the bi-direcitonal data bus
d) Reading from digital input line
   i) Read in the signal from the enable which can be determined via the pin on the line
   ii) If this is 1, then the data can be read and stored/outputted
   iii) If this is 0, then do nothing.
e) Writing to a digital output line
   i) Read in the signal from the 3 bit pin on the output control line

ii) Based on the data from the pin, the corresponding action will be performed

iii) The actions are as follows:

Signal	Action
000	Disable all
001	Read ROM
010	Read RAM
011	Write RAM
100	Write I/O
101	Read I/O



7. Transform the pseudo code as designed above to a C code. The code should be setup to run in an infinite loop to repeatedly execute these tasks. (See tip # 13). Note: While you write appropriate C language statements, since the hardware does not exist in real life, this code will not be executed on the CS machines. You can model your template similar to the templates we provided for Lab-1.

Problem 3:

Design and verify the following components. Each component should be able to handle 8-bit inputs. (Use basic logic gates for the design. You may use the built in modules for registers, and mux/demux)

i. Adder.

ii. Subtractor (2's complement)

iii. Magnitude comparator.

iv. Left and right - barrel shifter. (A barrel shifter shifts multiple bits at a time)

All of these can be found in the logisim files that are in the zipped folder.