Project 2

Due August 23, 2024 at 9:00 PM

This project description is subject to change at any time for clarification. For this project you will be working with a partner.

Desired Outcomes

- Exposure to using circuit simulator (Logisim Evolution)
- Exposure to processing button input
- An understanding of how to develop sequential logic circuits

Project Description

You will be using Logisim Evolution for this project. The overall objective is to design a digital lock. The project will be broken up into three subcircuits. You may use any of the built-in components of Logisim Evolution, except for those in the Arithmetic group. All class projects will be run through MOSS like software to determine if students have excessively collaborated. Excessive collaboration, or failure to list external sources will result in the matter being referred to Student Judicial Affairs.

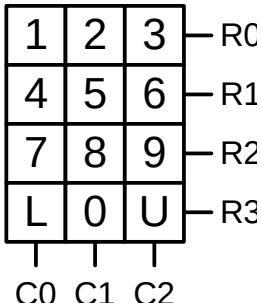

Figure 1. Control Panel

1. The control for the digital lock is a 12-button panel with 7 outputs R0..R3 and C0..C2. Figure 1 shows the layout of the control panel. When a button is pressed the column line C0, C1, or C2 will output a 1, and the row line R0, R1, R2, or R3 will output a 1. In order to simplify subsequent subcircuits you will be converting the outputs of the control panel into an output button code and a valid bit. If only a single button is pressed the valid bit V will be 1, and the button code bits B3..B0 will be the value of the button 0 = 0000, 1 = 0001, …, L=1010, and U=1011. If multiple buttons are pressed or no button is pressed V will be 0.

```
Inputs:     R0 R1 R2 R3 C0 C1 C2
Outputs:    B3 B2 B1 B0 V
Circuit:    Panel
```

2. The input from the control panel must be buffered so that the subsequent logic can distinguish between the multiple button presses of the same button and holding a single button. When a valid button press is detected (V=1), the buffer will output the button code for one clock cycle and then return to the no code 1111. The input must go back to invalid button press (V=0) for one clock cycle in between button presses. The second button press can be ignored of a two-button press sequence if the valid bit V does not return to zero first. In practice this will never happen since the clock CLK is much faster than a human can press.

```
Inputs:     B3 B2 B1 B0 V CLK
Output:     C3 C2 C1 C0
Circuit:    InputBuffer
```

3. Once the input from the control panel has been buffered, the last step is to design the lock circuit. The lock circuit controls a small motor that will spin to lock or unlock the lock. In order to lock the lock the outputs to the motor must be set so L=1 and U=0 for 4 clock cycles. In order to unlock the lock the outputs to the motor must be set so L=0 and U=1 for 4 clock cycles. After the lock or unlock sequence, the motor must be put into idle with the output L=0 and U=0. During the locking or unlocking of the lock, codes input from the input buffer can be ignored. The lock initializes in the unlocked state with the code 000_{10}.

   The lock can be locked (from the unlocked state) by receiving the L code (1010) twice in a row or by the L code followed by a three digit unlock combination. The double L code will lock the lock keeping the previous three digit unlock combination, the L code followed by the three-digit combination will update the unlock combination. Input of the L code during a three-digit lock combination will start the sequence over again ignoring the one or two digits that were input. Input of the U code (1011) during a three-digit lock combination will return the lock to the unlocked state cancelling the lock process.

   The lock can be unlocked (from the locked state) by receiving the U code followed by a three digit unlock combination that matches the previously stored unlock code. If combination does not match, the lock returns to the locked state. Input of the L code during the three-digit unlock combination will return the lock to the locked state with the previously input digits being ignored. Input of the U code during the three-digit unlock combination will start the process over, where the lock will be waiting for the three-digit unlock combination.

```
Inputs:     C3 C2 C1 C0 CLK
Output:     L U
Circuit:    Lock
```

You can unzip the given tgz file with utilities on your local machine, or if you upload the file to the CSIF, you can unzip it with the command:

```
tar -xzvf proj2given.tgz
```

You **must** submit the circuit file, README.md file, and `.git` directory in a tgz archive. You can tar gzip a directory with the command:

```
tar -zcvf archive-name.tgz directory-name
```

You should avoid using existing circuits as a primer that are currently available on the Internet. You **MUST** specify in your README.md file any sources of circuits that you have viewed to help you complete this project. Your README file **MUST** have both partner's name and SID number, a brief description of what circuits work/don't work, and a list of sources you used for designing of your circuit (you do not need to list the book or lecture notes it is assumed these have been used). You **MUST** properly document **ALL** uses of Generative AI following the guidelines outlined in the Generative AI Restrictions. All class projects will be submitted to MOSS like software to determine if students have excessively collaborated. Excessive collaboration, or failure to list external code sources will result in the matter being referred to Student Judicial Affairs.

Grading

The point breakdown can be seen in the table below. Make sure your circuit executes correctly in Logisim Evolution 3.8.0 as that is where it is expected to execute. You will make an interactive grading appointment to have your assignment graded. You must have a working webcam for the interactive grading appointment. Project submissions received 24hr prior to the due date/time will received 10% extra credit. The extra credit bonus will drop off at a rate of 0.5% per hour after that, with no additional credit being received for submissions within 4hr of the due date/time.

Points	Description
10	Has git repository with appropriate number of commits
10	Has README.md with proper documentation
15	Panel correctly outputs on inputs
15	InputBuffer correctly buffers input
10	Lock locks on at least one of two valid sequences (holds L for 4 cyc)
10	Lock unlocks with valid code (holds U for 4 cyc)
10	Lock does not unlock with invalid code
10	Lock functions fully as specified
10*	Student understands all circuits they have provided

* Students who are unable to demonstrate understanding of their circuit could receive negative points and resulting in score as low as zero overall regardless of functioning of circuit submitted.

Helpful Hints

- For testing you will want to select Auto-Tick Enabled on the Simulate menu.
- Drawing a state diagram will clarify when changes should occur.
- Do not be afraid of creating internal signals to simplify the state changes. For example, you might want a signal for lock command or unlock command that will be true if the code C3..C0 is the lock or unlock code respectively.
- For part 1 you may want to create valid column and valid row signals for use in the Panel circuit that are true when only one column (or row) is true, this can be used to easily create the V signal. For each B bit, you may just want to OR all the combinations of buttons that will make the value true.
- For part 2 you need more than two states, most likely three states will be necessary. For output you may want to use a mux to select between the buffered code and all 1s.

- For part 3 break up the states into top level state and sub state. The top-level states should be Unlocked, Locking, Locked, and Unlocking. The Unlocking and Locking states will transition to the Unlocked or Locked states after 4 clock cycles (a counter will be helpful for this). The Unlocked and Locked states can both be broken down into the separate sub states based upon the button presses.