CS 2110 Lab 7 Assignment: Binary Reduced State Machine

Your brilliant Conte TAs

September 16, 2019

Contents

1 Introduction

- In homework 4, you implement both a **One-Hot State Machine** and a **Reduced State Machine**.
- You will recall that a One-Hot State Machine represents n states in n bits, leaving one state bit high at all times and the rest low. A Binary Reduced State Machine, on the other hand, represents 2^m states in m bits, encoding each state as a binary number. This means a Binary Reduced State Machine will usually require fewer state bits to represent the same number of states!
- In this assignment, you'll implement a Binary Reduced State Machine. **There are some significant differences between the two state machines, so read carefully**.

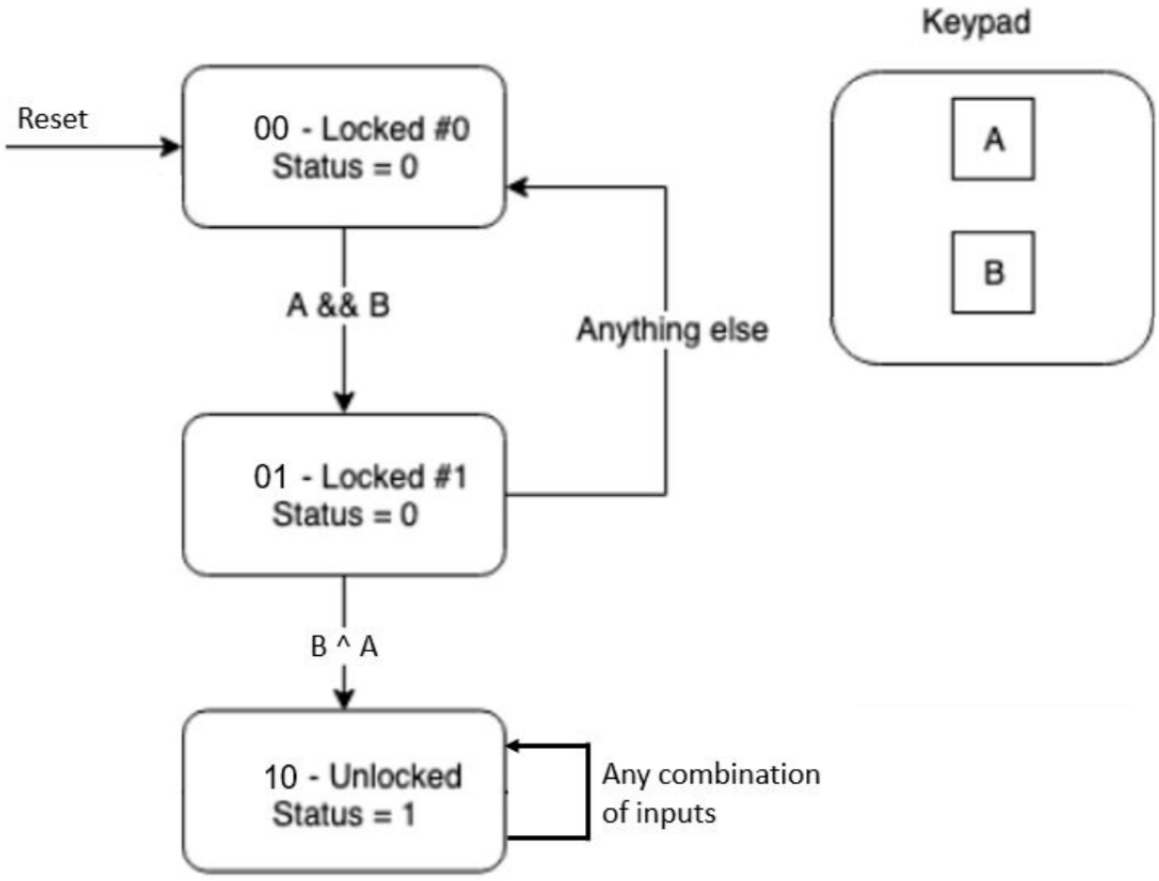
For more detail on both kinds of state machines and their big differences, check out the Lab 07 Guide PDF, available on Canvas → Files → Lab Resources → Lab Guides → Lab 07.pdf

2 Assignment

Given a simple state diagram, you will build a state machine in CircuitSim using the Binary-Reduced style of building state machines. This state machine has some significant differences from the one in the project, so read carefully.

- The circuit will be implemented in the `lab7.sim` file.

The state machine we are going to be building is a safe. The safe has two keys and requires a certain combination to unlock the safe. Once unlocked, the safe remains unlocked indefinitely until reset. The two keys are `A` and `B`. The combination to unlock the safe is a two step process. You must press `A` and `B` at the same time, then press exactly one of `A` and `B` (XOR). If a user enters the wrong combination for the given step they are on, then the safe resets back to the the first step, `Locked 0` state. The outputs of the state machine is `Status` where `0` is locked and `1` is unlocked. See the diagram below for a visual example of this safe translated into a state machine works.



- You will be implementing this state transition diagram as a circuit using the **Binary Reduced** style. Remember for binary reduced you will have a register with the smallest possible of bits to hold all of the binary encoded representations. Each state maps to a binary value (starting at 0b00, 0b000, 0b0000, etc. depending on how many states you need), and you are in that state if the corresponding binary value is stored in the register. There may be **any number** of inputs to the register active at any time, as we can have more than one state bit high at once.
- You must use exactly **one** register. These are found under the Memory tab in CircuitSim.
- The reset input for this circuit will return your state machine to its 0b00 state.
- You must implement this using binary-reduced. Write a truth table, create and complete a Karnaugh map for each output, and then implement the state machine in `lab7.sim`
- As you are reducing with Karnaugh maps, you should use sum-of-products style circuits for each output (e.g. ANDs and ORs, and NOTs if needed). You should not use XORs or XNORs (in order to get better practice with sum-of-products reduction).
- Note that there are four inputs to the subcircuit: `A`, `B`, `Clock`, and `Reset`. The inputs `A`, `B` corresponds to whether or not those keys are pressed, as in the diagram above. `Clock` turning off and on repeatedly will be used to represent clock ticks in your circuit. `Reset` corresponds to an attempt to reset your circuit.

2.1 Autograder

Once complete, run the autograder

```
java -jar lab7-tester.jar
```

inside the Docker container in the command prompt in the same directory as your `lab7.sim` files.

Demonstrate to your TAs that the tests pass and show them your reduced circuit to get checked off early!