Erich Viebrock

Gilbert Rivera

ECPE 174

**Lab Report #3**

Problem Summary:

Our third assignment is to build a finite state machine that resembles a combinational lock. The lock has three inputs and one output, labeled as “A,” “B,” “C,” and “Z,” respectively.

The lock looks for the sequence “BACB” in order to unlock, otherwise will remain locked. Only one input can be entered into the lock at a time. If at any time an incorrect input is sent to the input stream, the lock will revert back to the initial state. Once the lock is unlocked, the lock will remain unlocked until the input stream sees another input.

The purpose of this assignment is to familiarize ourselves with writing VHDL for asynchronous finite state machines.

Assumptions:

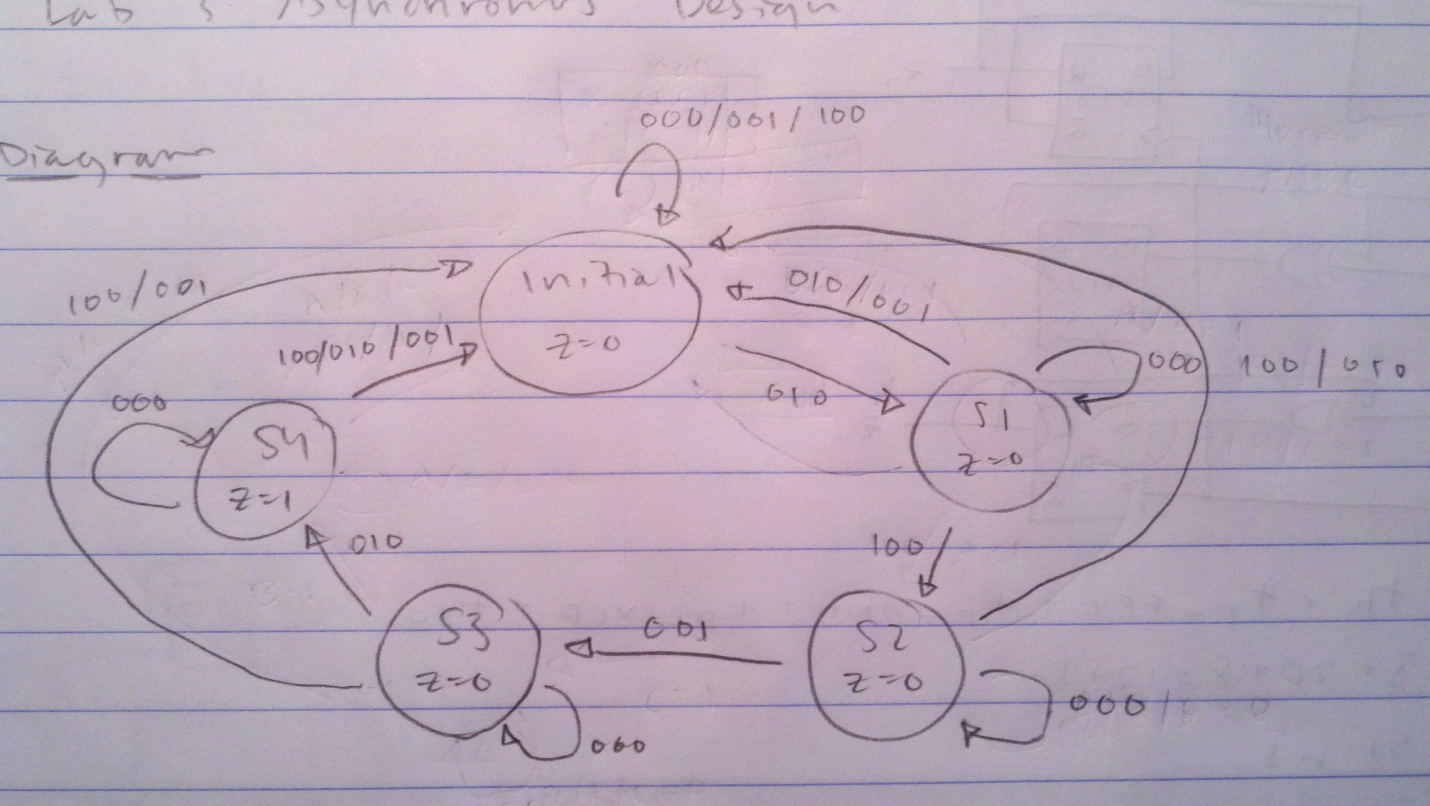
We assumed if more than one input was entered to the lock at a time, the lock would return to initial state.

Design Approach:

In our design, we decided to format the VHDL to implement a Moore asynchronous finite state machine. We chose a Moore finite state machine because the syntax was easier to write.

1. **State Machine Diagram:**

**(Assume any input not mentioned in each state returns to IDLE state)**



1. **State Assignment Table:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Present State | Next State  (A, B, C) | | | | | | | | Output |
| y3y2y1 | (000) | (001) | (010) | (011) | (100) | (101) | (110) | (111) | Z |
| Y3Y2Y1 | Y3Y2Y1 | Y3Y2Y1 | Y3Y2Y1 | Y3Y2Y1 | Y3Y2Y1 | Y3Y2Y1 | Y3Y2Y1 |
| 000 | 000 | 000 | 001 | 000 | 000 | 000 | 000 | 000 | 0 |
| 001 | 001 | 010 | 000 | 000 | 000 | 000 | 000 | 000 | 0 |
| 010 | 010 | 000 | 000 | 000 | 010 | 000 | 000 | 000 | 0 |
| 011 | 011 | 000 | 100 | 000 | 000 | 000 | 000 | 000 | 0 |
| 100 | 100 | 000 | 000 | 000 | 000 | 000 | 000 | 000 | 1 |
| 101 | dd | dd | dd | dd | dd | dd | dd | dd | dd |
| 110 | dd | dd | dd | dd | dd | dd | dd | dd | dd |
| 111 | dd | dd | dd | dd | dd | dd | dd | dd | dd |

Verification Procedure:

When we attempted to initially run our VHDL to the FPGA board, the desired output was not formed. No matter what inputs were used, the lock would never open. This issue resided within our VHDL code. After speculation, we determined we needed the statement:

z<= '1' when y=O else '0';

in order to produce desired output. The following code means output will be HIGH when in “O” state, or else would have LOW output. After implementing this line of code, our lock behaved as the desired combinational lock. When the input stream “BACB” was entered, the lock would open, and close once another input was entered. For all other inputs, the lock would remain locked, and return to initial state when a single wrong input was entered.

Post-Lab Questions:

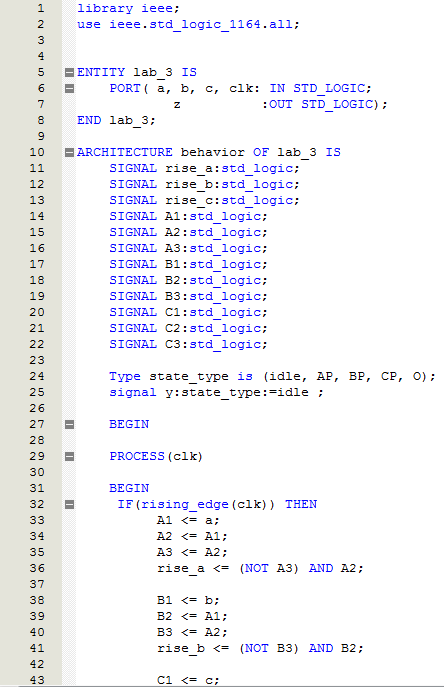
1. What does the Quartus state machine viewer generate? How does it compare to your state machine? Explain any discrepancies.
   1. The Quartus state machine viewer generates a Moore finite state machine, synchronized from the rising clock edge. The state machine viewer generates 5 states, with different situations of proceeding to the next state, staying at the current state, and reverting back to initial state.
   2. The Quartus state machine viewer compares exactly to our presumed state machine. In our state machine, we had 5 states, with different situations of proceeding to the next state, staying at the current state, and reverting back to initial state. Never is there a situation one state reverts to a previous state other than initial state.
2. What complexities arise in designing an asynchronous FSM compared to a synchronous FSM?
   1. The only complexities that arise in designing an asynchronous FSM compared to a synchronous FSM is synchronizing inputs to the rising edge or falling edge of a clock. In this lab, we chose to synchronize our inputs to the rising edge of the clock. Otherwise, designing an asynchronous FSM was very similar to designing a synchronous FSM.

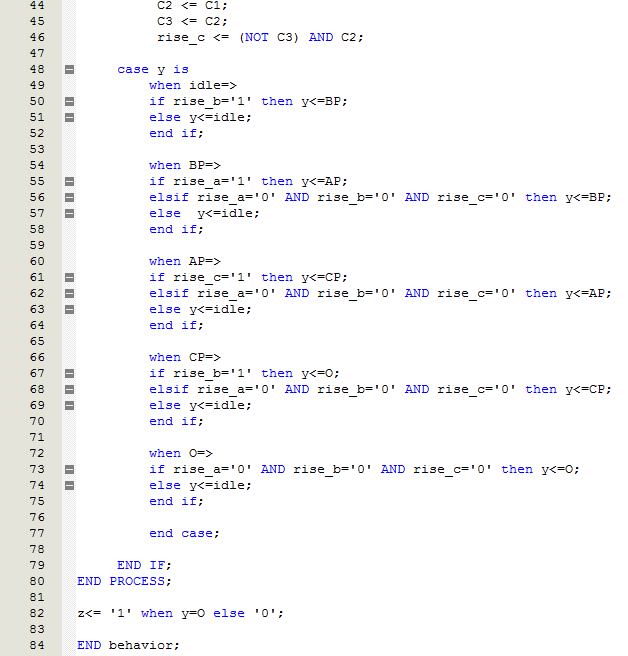
Good. Work on using precise language – it is important in technical writing that you pick the correct words so that someone else could follow your procedure.

Rubric:  
Organization = 2  
Quality = 1.5  
Amount = 2  
Technical Content = 1.5

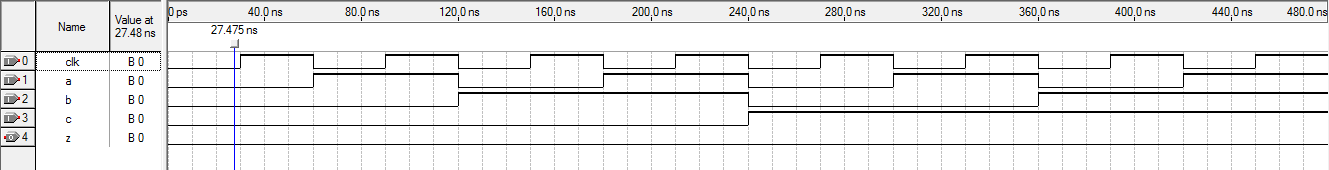
Appendix:

**VHDL**





**Vector Waveform**



**State Machine Viewer:**

