ECE3270 Digital System Design
Lab 3: State Machines

Lab Overview: The purpose of this lab is to implement a state machine that acts as a lock. The requirements of this lab consist of completing the VHDL design, submitting all modified source files to the assign server, and completing the findings report using the LaTeX template previously provided. You should include discussion of the type of state machine generated and include a copy of the state machine diagram generated by Quartus.

Part I

You are to design a Moore state machine that keeps track of whether a lock is unlocked. Your input will be a single bit, and 5 correct bits are needed to send the unlock signal. However, there are 3 False Codes that will allow the Lock sequence to complete and send the machine to a **FALSE UNLOCKED** state. The correct code will send the machine to an **UNLOCKED** state.

The string to unlock the machine is:	“**10110**”		
The strings to move into the false unlocked states are:	“**10111**”	“**10100**”	“**10101**”

Only paths that follow one of the above 4 strings will continue, a false move will send the machine back to the initial locked state. After reaching a false or real unlocked state, move to the first unlocked state in the next clock cycle.

You will need a total of **9** states: **Reset**, **Unlocked**, **False**, and the rest are the **Locked** states. Encode them as follows:

RESET:	**“0000”**
FALSE:	**“0100”**
UNLOCKED:	**“1100”**
LOCKED:	**Use the remaining values for your other 6 states**

(note the states are carefully chosen for Unlocked and False to allow for simpler encoding)

Write this state machine in its own file, using descriptive names for your I/O pins, internal signals, and your state encoding. Refer to the lecture notes for how to set state encoding. Make sure you have an external **clock** pin that will update your machine on a **rising edge**.

Write a testbench to test this file and verify functionality

Part II

Wrap your above component into a file that will interface with the DE1-SoC board. You will map your input bit to **SW$_0$**, your state outputs to **LEDR$_{3->0}$**, and your clock to **KEY$_0$**. Compile and program your board with this design to verify real-world functionality. Note that the hardware on the board may cause your state machine to move ahead multiple states when attempting to press the key one time. You may use one of the unused switches if the **KEY** hardware appears to bounce too much.

Rubric	
Report - Proper format - All sections included - Valid images where applicable - Proper grammar, punctuation, and spelling	**50%** - 10% - 30% - 5% - 5%
Demo - Live Demonstration o Includes working code and answering questions from the TA - Proper modular design o Thoughtful I/O and Signal names included - Comments o Thoughtful comments, not English translations of code	**40%** - 30% - 5% - 5%
Proper Assign Server Code Submission	**10%**