

UNIVERSITY OF CALIFORNIA AT BERKELEY
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ASSIGNED:	Week of 2/11
DUE:	Week of 2/18, 10 minutes after start (xx:20) of <i>your assigned</i> lab section.

Lab 3

Verilog Synthesis & FSMs

1.0 Motivation

In Lab1, you became acquainted with Synplify as well as the Xilinx Place and Route tools. In Lab3, you worked with ModelSim and Verilog to produce a working design. This lab is designed to give you a chance to bring those skills together to design, specify and then implement a fully working Finite State Machine in Verilog, using a combination lock as an example.

For this lab we have given you all of the support modules and design necessary, your job is to take our high level design and translate it into working Verilog. In order to do this you will need to force yourself to use good Verilog coding style, acquaint yourself with the Verilog language and become at least somewhat proficient with the CAD tools.

2.0 Introduction

In this lab you will be making a **2bit, 2 digit combination lock** such as those sometimes found on secure doors. The inputs to the lock consist of **two code switches** and **three buttons**. The **code switches** (SW9[2:1]) are used to enter the digits in the combination. The three buttons are **Reset** (SW1) which is used to reset the lock, **Enter** (SW2) which is used to enter a digit of the combination and **ResetCombo** (SW6) which would not be accessible normally and which will reset the combination that will open the lock to a default value.

To operate the lock, a user would:

1. **Set the code** on SW9[2:1] to the **first digit** and **press enter** (SW2).
2. **Set the code** on SW9[2:1] to the **second digit** and **press enter** (SW2).
3. The lock will **Open**.
4. The user would then **press enter** (SW2).
5. **Set the code** on SW9[2:1] to the **new first digit** and **press enter** (SW2).
6. **Set the code** on SW9[2:1] to the **new second digit** and **press enter** (SW2).
7. Cycle back to **step 3** above...

When someone gets the combo wrong it would go like this:

1. **Set the code** on SW9[2:1] to a **wrong digit** and **press enter** (SW2).
2. **Set the code** on SW9[2:1] to **any digit** (right or wrong) and **press enter** (SW2).
3. The lock will show **Error**
4. The lock will stay in this state until the user **presses Reset** (SW1).

3.0 Prelab

Please make sure to complete the prelab before you attend your lab section. **You will not be able to finish this lab in 3hrs otherwise!**

1. **Read this handout thoroughly.** Pay particular attention to section 4.0 Lab Procedure as it describes in detail the circuits you must create.
2. **Examine the Verilog** provided for this weeks lab.
 - a. Most of the modules you have seen before.
 - b. Make sure you **understand FPGA_TOP2** as it instantiates your module, and **handles the I/O.**
3. **Write your Verilog ahead of time.**
 - a. **Lab3Lock.v**
 - i. Your lock FSM, which instantiates the comparator
 - b. **Lab3Testbench.v**
 - i. A testbench/Code Breaker for the lock FSM
 - ii. The testbench will function as a code breaker that goes through all of the lock inputs until it finds the correct one and displays that.
 - iii. Make sure that this tests **as much of the lock and comparator as possible.**
 - iv. Refer to **past testbenches** as a starting point.
4. You will need the **entire 3hr lab to test and debug your Verilog!**
 - a. Remember it has to pass **simulation and synthesis**

4.0 Lab Procedure

Since we expect you to write your verilog ahead of time, and Verilog is nothing more than a bunch of standard text in a file with a *.v extension you **can do this part of the lab entirely from home** in your favorite text editor or by connecting to Kramnik for remote access to Xilinx tools:

<http://inst.eecs.berkeley.edu/~cs150/fa06/Kramnik/tutorial.php>.

Or you can come into the lab and use the tools there. For those of you who like maintaining a single Xilinx Project Navigator project for each lab, you can even **create the project ahead of time and write your Verilog from within Project Navigator.**

Remember to **manage your Verilog, projects and folders well.** Doing a poor job of managing your files can cost you **hours of rewriting code**, if you accidentally delete your files.

4.1 Lab3Top

The Lab3Top module is very simple and does not need to contain any behavioral Verilog. The Lab3Top module **simply instantiates the Lab3Lock FSM and Lab3Compare modules, and ties them together** with the appropriate signals.

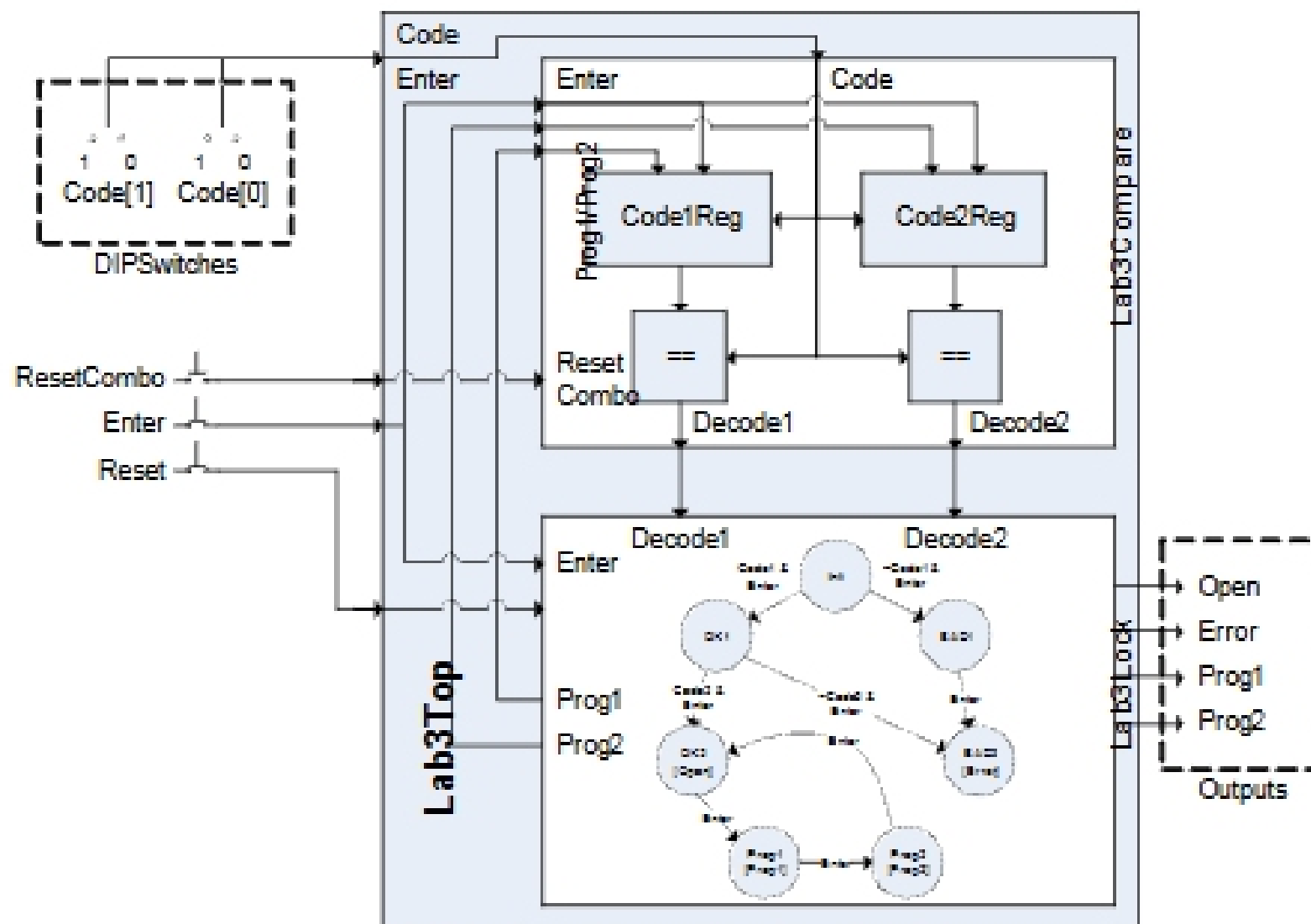


Figure 1: Lab3Top Block Diagram

In order to **simplify the Lab3Lock FSM** and make the whole lock design more flexible we have separated the FSM and the comparator, which **compares the value on the switches to the digits of the combination**. Essentially the code input from the switches is fed into the Lab3Compare module which determines the **Decode1** and **Decode2** signals needed by the Lab3Lock FSM. Based on these and the **Enter** and **Reset** buttons the Lab3Lock tracks its state and generates the appropriate outputs.

Table 1 below shows the inputs and outputs from the Lab3Top module.

Signal	Width	Dir	Description
Code	2	I	The code value from dipswitch SW9[2:1]
Enter	1	I	The Enter button (SW2)
ResetCombo	1	I	Reset the combination to the default (SW6)
Clock	1	I	The Clock signal
Reset	1	I	Reset the FSM to the Init state (SW1)
Open	1	O	Indicates that the lock is open (OK2 state)
Error	1	O	Indicates that the wrong combo was entered (BAD2 state)
Prog1	1	O	Indicates that the lock will accept a new 1 st digit
Prog2	1	O	Indicates that the lock will accept a new 2 nd digit
LED	8	O	Debug outputs, for your use on the board

Table 1: Port Specification for Lab3Top