Lab #6 Combinational lock using FSM

Leonardo Fusser (1946995)

Objective:

- Design an FSM.
- Test the design using a test bench.

Hardware: see previous lab.

To hand in:

Listing of the following for each of your designs on LEA:

1. This sheet with answers and all screenshots.
2. Commented Verilog source files - module and top module.
3. Verilog testbench source file (with proper prolog headers).
4. A block diagram (Visio or other).
5. A state diagram using Visio or other Apps. Failure to submit a state diagram will result in **15 points** lost.

Lab work:

In your code, you must separate the system in two OR three parts:

1. Next state logic code and Output logic code together.
2. State register code.

OR

1. Next state logic code.
2. State register code.
3. Output logic code.

Design and test bench a combinational lock using an FSM

You must design a combinational lock using an FSM.

- To open the lock, the keys must be pressed one at a time in a specific sequence.

  Example of sequence 2814:

First code is 2	Second code is 8	Third code is 1	Fourth code is 4
00**0**0	**0**000	000**0**	0**0**00

- If the keys are pressed in a different order (e.g., 8214) the lock would not open, so the combination is not only the numbers but their orders.
- There can be **no consecutive** same code. (e.g., 2**88**4). The key can be press/maintained by the user.
- A bar graph LED must display the different states of the machine:

	Bar graph LED				
	LD7	LD3	LD2	LD1	LD0
No code entered or reset or wrong code	**0**				
First code successfully entered					**0**
Second code successfully entered				**0**	**0**
Third code successfully entered			**0**	**0**	**0**
Fourth code successfully entered		**0**	**0**	**0**	**0**

- If no keys are being pressed or if the FSM is reset, only LD7 turns on.
- When the first code is entered successfully, LD0 turns on. When the second code is entered successfully, LD1 turns on etc.…
- When the four codes are entered in the right sequence then LD0 through LD3 is on. Also, once the combination is entered successfully, the user can reset the FSM by using a reset switch.
- Whenever the user enters a wrong code, LD0 through LD3 turn off and LD7 turns on and the sequence must restart from the beginning.

You must use `localparam` to improve readability

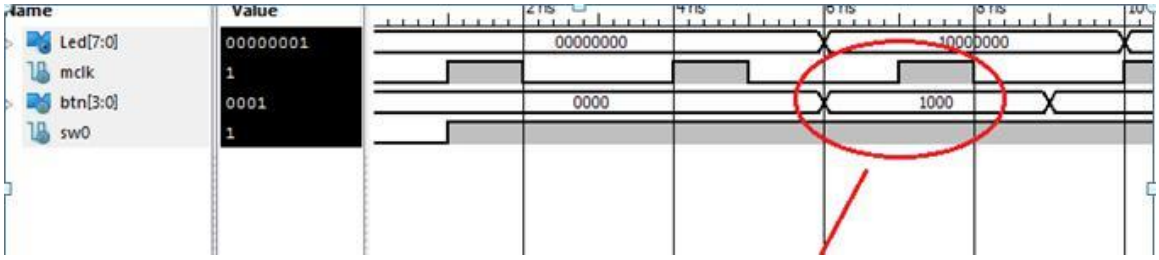
1. Draw the system's block diagram.
2. Draw the system's state diagram.
3. Write a module for your FSM.

```
module lock(
    input clk,
    input rst,
    input [3:0] key,
    output reg [7:0] bargraph
);
```

Test Bench:

4. Test bench you FSM. In your test bench, make sure the clock and inputs do not change at the same time by inserting delays:

```
btn=4'b1000;
#1;
mclk = ~ mclk;
#1;
mclk = ~ mclk;
#1;
btn=4'b0010;
#1;
```



There must always be a clock positive edge during a new key code.

5. You must improve your design by generating good and bad combination sequencing. Provide screenshots.

   ➢ Refer to "Figure 1" and "Figure 2" on next page.

Test on the target:

6. In order to keep your code portable and modular you must take a top module approach. Create a top module to instantiate and connect all modules to the board's I/Os.

```
module top( input
    mclk, input sw0,
    input [3:0] btn,
    output [7:0] Led

);
```

7. Download the code on the target and test it.

Final result to be approved!

After lab questions:

How many DFFs were required in the State Machine? Explain.

- To implement this state machine, only six DFFs are required. This is because in the "CombinationLock" module, three DFFs are required for the current FSM state variable and another three DFFs are required for the next FSM state variable. Therefore, the two together add up to six DFFs.

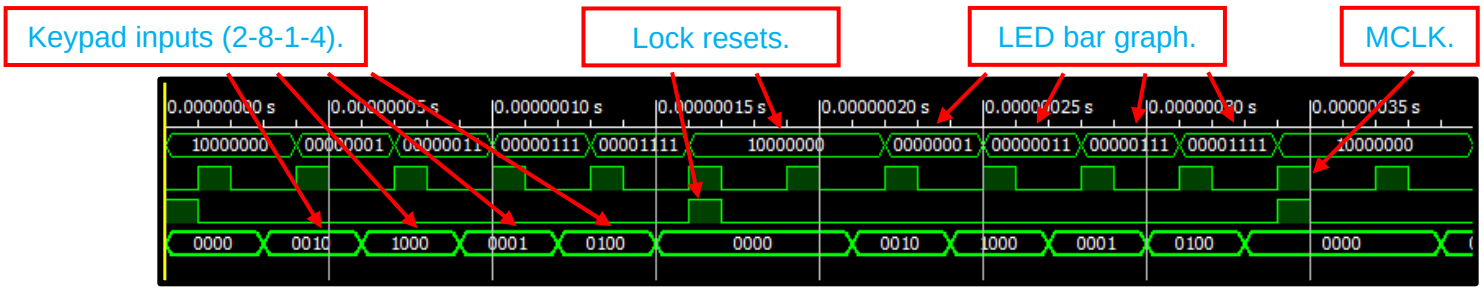


Figure 1. Timing diagram shown for the Verilog "CombinationLock" module. Result above shows what happens when the correct keypad input sequence is entered.

Figure 2. Another timing diagram shown for the Verilog "CombinationLock" module. Result above shows what happens when the incorrect keypad input sequence is entered.

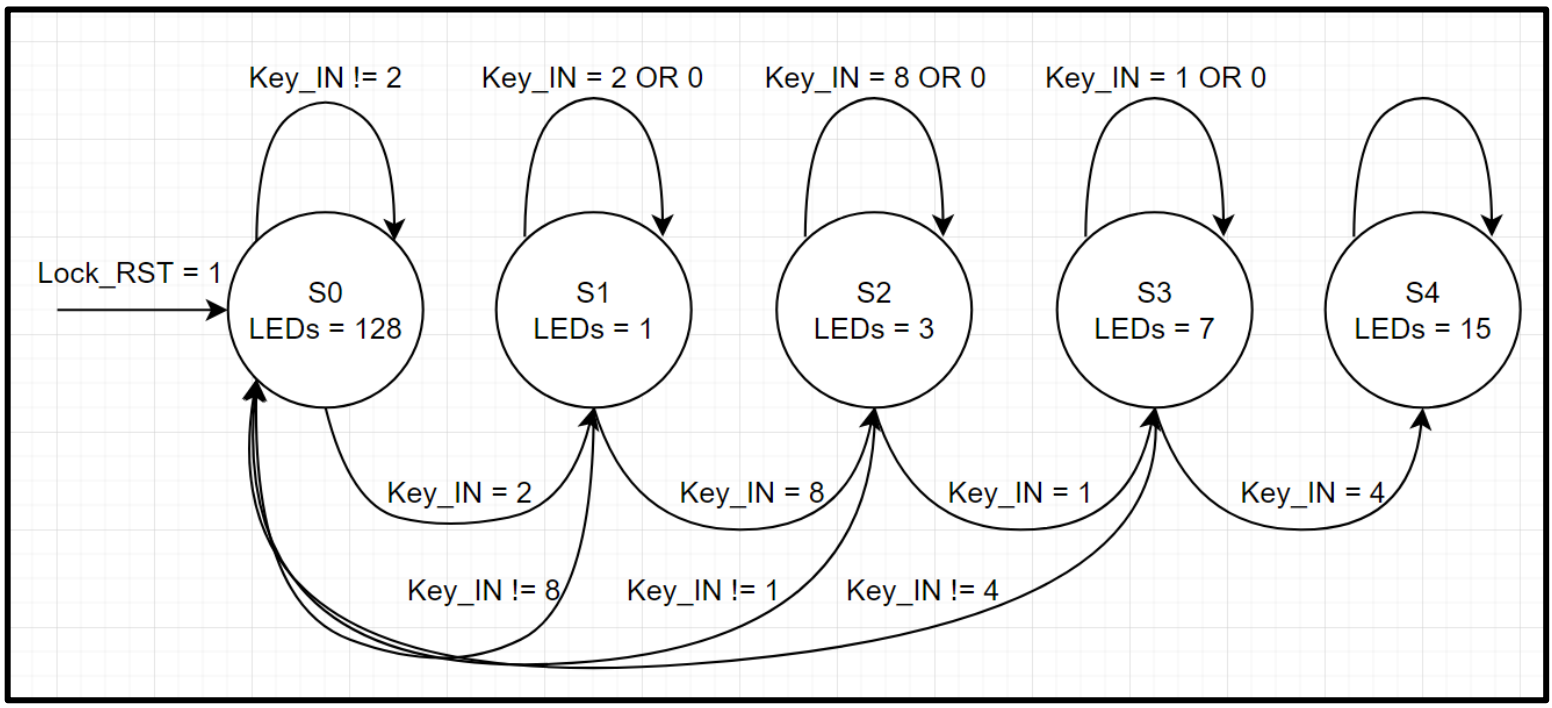


Figure 3. State diagram for the Verilog "CombinationLock" module shown above.

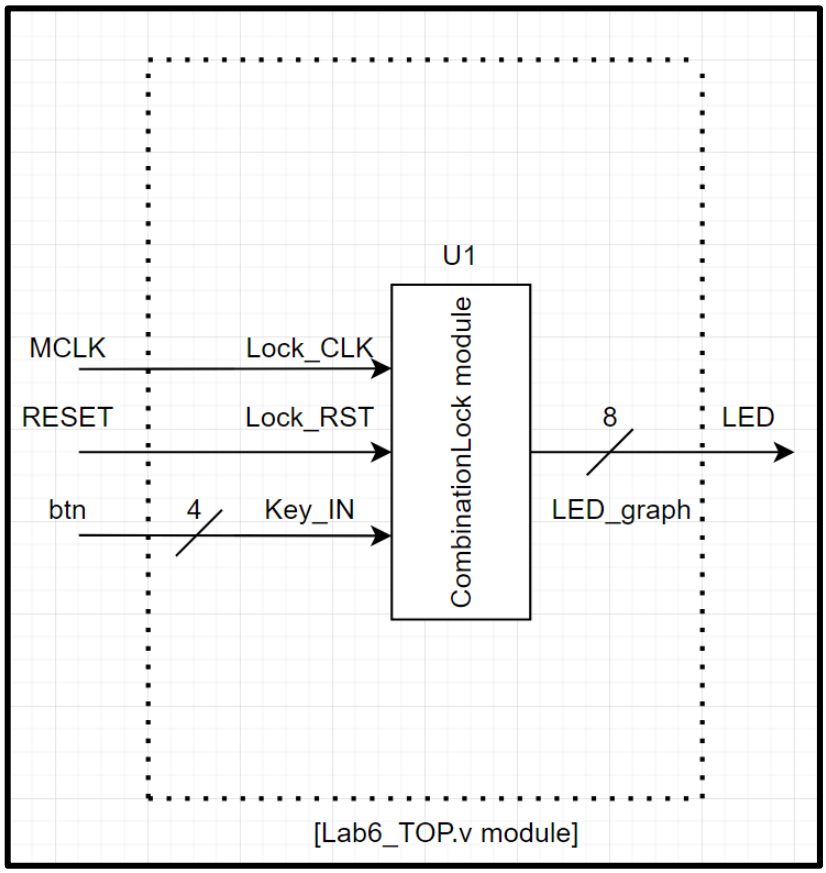


Figure 4. Block diagram shown for Lab 6.

[Verilog code for Lab 6]

- Refer to Verilog “CombinationLock” module code in “CombinationLock.txt” attached to this report submission.
- Refer to Verilog “Lock_TB” module code in “Lock_TB.txt” attached to this report submission.
- Refer to Verilog top module “Lab6_TOP” code in “Lab6_TOP.txt” attached to this report submission.

[Verilog constraints file for Lab 6]

- Refer to Verilog constraints file “Lab6_constraints” code in “Lab6_constraints.txt” attached to this report submission.