Lab 06: Simple Elevator

(Nov 20, 2018)

Submission deadlines:

Source Code:	18:30, Nov 20, 2018
Report:	23:59, Nov 24, 2018

Objective

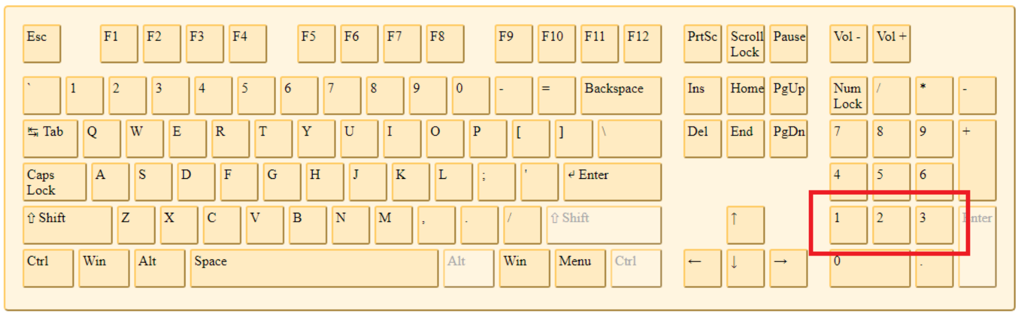
- ➢ To be familiar with modeling finite state machines with Verilog.
- ➢ To be familiar with the FPGA design flow, the demo-board I/Os
- ➢ To be familiar with the keyboard control on FPGA.

Description

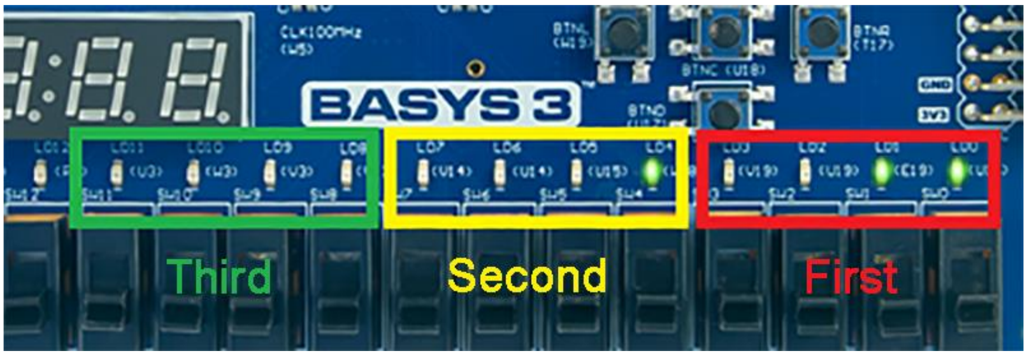
The elevator is a common facility in life. Therefore, we want you to design a Simple Elevator for us and implement it on the FPGA board with I/Os (keyboard, buttons, LEDs and seven-segment display). The only feature of this simple elevator is moving to the floors where people are waiting and carrying them away. You don't need to consider which floor they want to go.

- The elevator can only reach three floors: the first floor, the second floor and the third floor. After reset, the elevator stays on the first floor and no one is waiting on any floor.

- Use the keyboard to increase the number of people on a particular floor. For example, press 2 once to increase the number of people on the second floor by 1.



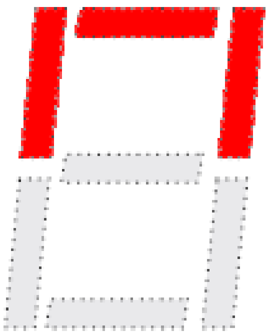
- The number of people waiting for the elevator on each floor is represented by 12 LEDs. From right to left, 1~4 means the number of people waiting on the first floor, 5~8 on the second floor and 9~12 on the third floor. The number of people waiting on each floor can be 0~4, represented from right to left (an LED represents a person).



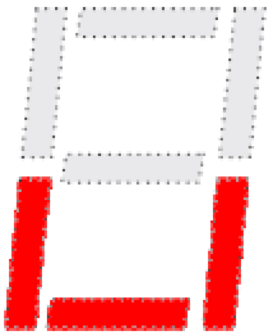
This example shows that there are two people on the first floor, one on the second floor and nobody on the third floor.

- The elevator has three states: STAY, UP and DOWN:
  - In the STAY state, the elevator will stay on the same floor. If someone is on the same floor as the elevator. The elevator will carry them one by one (with frequency of clk/(2^26)).
  - In the UP/DOWN state, the elevator will move up/down. The elevator will take four clock (clk/(2^26)) cycle time to go to the next floor. All the passengers have to wait when the elevator is moving.
  - If there is no passenger waiting, the elevator stops on the last floor.

- Use two 7-segment displays to indicate the status of the elevator. The right one indicates the location of the elevator while the left one indicates the movement of the elevator. In the STAY state, the left one will use symbols to indicate that whether the elevator will go up or down after carrying all the people on this floor. And in the UP/DOWN state, the left one will use animations to represent the state.

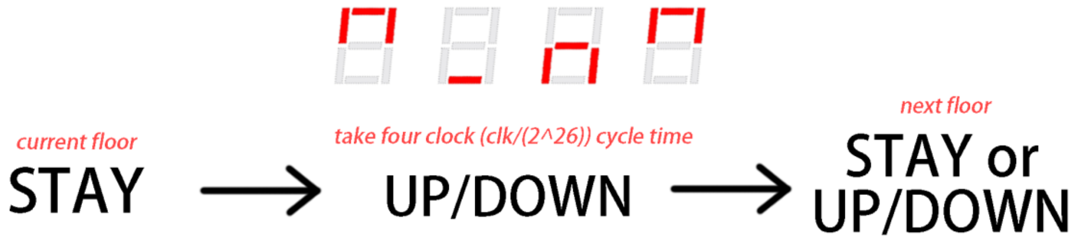


“UP” symbol



“DOWN” symbol

Animations demo : https://i.imgur.com/aqvPMTd.gif
https://i.imgur.com/WVl6lsz.gif



Animation of the “up” sign

- The elevator needs to keep its trend. For example, if the elevator goes to the second floor from the first floor, then it should go to the third floor first if there are people on both the third floor and the first floor, and vice versa.

Demo video: https://youtu.be/YVGCXC7MroI

I/O signal specification:

- **rst**: **asynchronous** active-high reset (connected to **BTNC**). The location of the elevator should be reset to the first floor and all LEDs are off when the **reset** signal is 1
- **clk**: clock signal with the frequency of 100MHz (connected to pin **W5**)

- **DISPLAY[6:0]**: signals to show the digits on the 7-segment displays.
- **DIGIT[3:0]**: signals to enable one of the 7-segment displays.
- **LED[11:0]**: indicating how many people on each floor.

Note:

1. The clock frequency of the 7-segment display controller is clk/(2^13). And the clock frequency of the elevator controller is clk/(2^26).
2. For keyboard control, you may refer to KeyboardController and KeyboardDecoder mentioned in class for reference.
3. The number of people on each floor should be able to increase and decrease at the same time.(as the demo video shows)

Hint:

1. You must design at least one finite state machine (FSM). You may have at least three states in your FSM design.

You can use the following template for your design.

```
module lab06(clk, rst, LED, PS2_CLK, PS2_DATA, DISPLAY,
DIGIT);
  input clk, rst;
  inout PS2_CLK, PS2_DATA;
  output [3:0] DIGIT;
  output [6:0] DISPLAY;
  output [11:0] LED;

  //add your design here
  //…

endmodule
```

Attention:

1. You should hand in only lab06.v. If you have several modules for your design, put them in lab06.v together. (Except the KeyboardDecoder. You don't need to put it into lab06.v or hand in KeyboardDecoder.v. If you modified the KeyboardDecoder, please mention it in the report.)

2. You should also hand in your report as lab06_report_StudentID.pdf (i.e., lab06_report_105080001.pdf).

3. You should be able to answer questions of this lab from TA during the demo.

4. You need to generate bitstream before demo.