ECE 697Ce: Foundations of Computer Engineering

Project 1: Elevator Operation

This assignment will give you experience in Verilog by implementing a Finite State Machine using of a simplified elevator operation. It will also expose you to buffering and serialization when multiple elevator buttons are pressed simultaneously.

1. Problem Statement

This assignment requires an elevator operation to be simulated in Verilog. There are 4 floors between which the elevator can move and there are buttons on each floor, all external to the elevator. A button press implies certain action to be taken, which are described in detail in the following sections. The elevator system can be modeled as a Finite State Machine that undergo state transitions and produces outputs depending on the inputs provided. Design the Finite state machine and describe its behavior either by using a state diagram or a table. Implement the system according to the specifications mentioned below, in Verilog and submit the code as well as the waveforms along with a detailed write up describing the same.

1.1 Input description

There are 6 inputs to the system in addition to a clock signal. The clock signal is used for synchronization purposes. The button 1U is in the first floor, 2U & 2D in the second floor, 3U & 3D in the third floor, and 4D in the 4^{th} floor. A button press and its implication is tabulated below.

Note that the switches with the same name at different floors are electrically connected together. So, if, for example, 1U is pressed at any floor just one signal will be active and be fed to the elevator controller.

INPUT (Button Press)	IMPLICATION
1U	Person in the first floor wants to go to the second floor
2U	Person in the second floor wants to go to the third floor
3U	Person in the third floor wants to go to the fourth floor
4D	Person in the fourth floor wants to go to the third floor
3D	Person in the third floor wants to go to the second floor
2D	Person in the second floor wants to go to the first floor

1.2 Output description

There are 3 outputs indicating the action that is to be taken and is tabulated below.

OUTPUT (elevator action)	IMPLICATION
Up	elevator moves in the upward direction
Down	elevator moves in the downward direction
Stay	elevator stays in the same floor

1.3 System behavior

The elevator can be in any of the 4 floors; Floor 1, Floor 2, Floor 3, or Floor 4. At each clock cycle, depending on whether a button is pressed or not, certain action has to be taken which is tabulated below:

Current Position of elevator	Input	Action sequence to be taken
First floor	1U	Move the elevator to the Second floor
	2U	Move the elevator to the Second floor and then to the Third
	2D	Move the elevator to the Second floor and then to the First
	3U	Move the elevator to the 3rd floor and then to the 4th
	3D	Move the elevator to the Third floor and then to the Second
	4D	Move the elevator to the 4th floor and then to the 3rd
Second floor	1U	Move the elevator to the First floor and then to the Second
	2U	Move the elevator to the Third floor
	2D	Move the elevator to the First floor
	3U	Move the elevator to the 3rd floor and then to the 4th
	3D	Move the elevator to the Third floor and then to the Second
	4D	Move the elevator to the 4th floor and then to the 3rd
Third Floor	1U	Move the elevator to the First floor and then to the Second
	2U	Move the elevator to the Second floor and then to the Third
	2D	Move the elevator to the Second floor and then to the First
	3U	Move the elevator to the 4th floor
	3D	Move the elevator to the 2nd floor
	4D	Move the elevator to the 4th floor and then to the 3rd
Fourth Floor	1U	Move the elevator to the First floor and then to the Second
	2U	Move the elevator to the Second floor and then to the Third
	2D	Move the elevator to the Second floor and then to the First
	3U	Move the elevator to the 3rd floor and then to the 4th
	3D	Move the elevator to the 3rd floor and then to Second floor
	4D	Move the elevator to the the 3rd floor

1.4 Processing multiple inputs: Buffer

In order to account for the cases when multiple buttons are pressed at the same time or one after the other, we need to have a mechanism to keep track of the input requests made (a buffer) and means to prioritize them. In order to simplify the code, consider the following to be the order of priority: 1U, 2U, 3U, 2D, 3D, 4D.

For instance, consider the following requests to be made one after the other: 2U, 3D and 1U. The order of processing them would be as follows: 1U, 2U and 3D irrespective of the current position of the elevator.

While designing the mechanism to keep track of the input requests made, note that the order of arrival of requests and the current position of the elevator is irrelevant in deciding the order of execution. This is because our system is supposed to follow the order of priority mentioned above. Also ignore an input request if the same request is already present in the buffer.

2. Finite state machine implementation

The system can be modeled as a finite state machine with inputs 1U, 2U, 3U, 2D, 3D, and 4D, and producing outputs Up, Down or Stay. The FSM should completely model the behavior described in the specifications and hence decide the number of distinct states accordingly. Note that if there are no inputs to be processed then the system has to produce the output Stay and remain in the same state.

3. Way to approach (Suggestion)

Start by designing the state machine that models the behavior described in the specifications. Implement the state machine in Verilog and verify its behavior. Proceed to implement the buffering system and verify its behavior as before. Once that is done, integrate the elevator state machine and the buffering system. Decide on the number of control signals that would be needed to coordinate both and verify the behavior of the entire system by applying the necessary stimuli.

4. Deliverables (what you need to submit as your solution to this project)

1. A short formal report including, problem statement, your approach for solving it, the steps you took to find the result, the problems you faced and your solutions to them, the procedure that you used to test your solution to show it works properly, and a conclusion section.

2. The Verilog code for each sections of the system (with plenty of comments for clarifications).

3. The Testbench Verilog code (with comments).

4. The screenshots of the Testbench output waveforms highlighting the points of interests for various input sequences (at least three cases).