Overall design flow:

In this in-class exercise, you will create and simulate a synchronous sequential logic system. You will practice using VHDL to build a provided state diagram. You will also write a testbench for a synchronous system to fully simulate the state machine's operation.

Overview

You've been asked to design an elevator controller that will traverse four floors (numbered 1 to 4). The system is specified as follows:

- It has two external **inputs**, i_*up_down* and i_*stop*.
- When i_*up_down* is '1' and i_*stop* is '0', the elevator will move up until it reaches the top floor (one floor per clock, of course).
- When i_*up_down* is '0' and i_*stop* is '0', the elevator will move down until it reaches the bottom floor (one floor per clock).
- When i_*stop* is '1', the system stops at the current floor.
- When the elevator is at the top floor, it will stay there until i_*up_down* goes inactive while i_*stop* is inactive. Likewise, it will remain at the bottom until told to go up and *i_stop* is inactive.
- The system should **output** the floor it is on (1 – 4) as a four-bit binary number (see section and diagram below). Note: For Lab4 you will need an elevator that goes up to Floor 15. How would you change the transition diagram below to extend to 1-15 floors?

You immediately translate this description into the state transition diagram shown below. You would now like to simulate the controller's behavior to see if you have effectively captured what your user wanted in an elevator controller.

Inputs: i_up_down, i_Stop

i_up_down: '1' Up, '0' Down
i_stop: '1' Stop, '0' Go

Outputs:

Floor 1, 0001
Floor 2, 0010
Floor 3, 0011
Floor 4, 0100

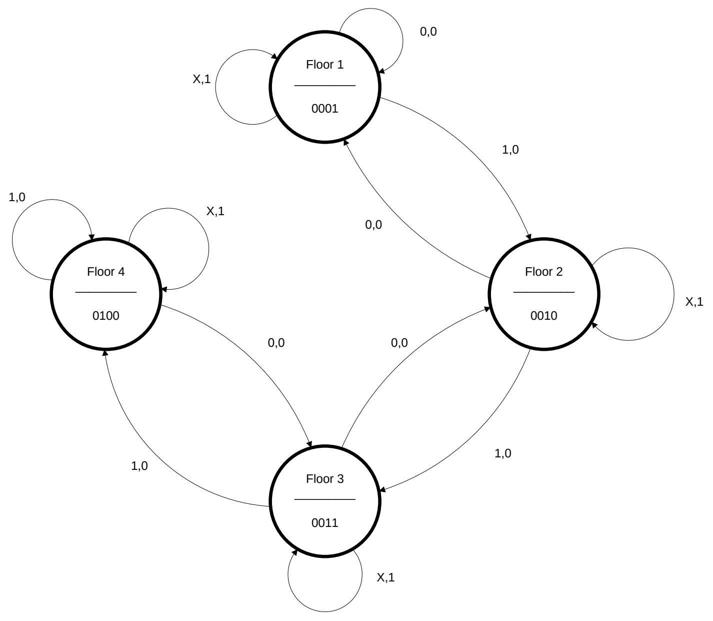

Figure 1. Elevator Controller State Transition Diagram

Tasks

Create an elevator controller module in VHDL

1. In Vivado, create a new project and name it **ICE5**.
2. Add a copy of the provided "MooreElevatorController.vhd" to your design.
    a. This template has already been loaded with enumerated types for your states.
    b. Enumerated types allow you to define your next state and output logic in terms of the state name and not the state bits! This means you don't have to create the equations for a chosen encoding type. Vivado gets to take care of that for you. Your first job in this ICE is to complete the provided elevator controller shell. Read through the provided code and comments, and then build the state machine in VHDL.
    c. Use a synchronous reset for your controller FSM
    d. type sm_floor is (s_floor1, s_floor2, s_floor3, s_floor4);
        1. This defines your type of states and all possible values that type can have
    e. signal f_current_state, f_next_state : sm_floor;
        1. This creates a signal of your new type. This signal can be assigned the values identified in the previous step. So f_current_state and f_next_state can each hold s_floor1, s_floor2, s_floor3, or s_floor4.
3. When you have completed your code and checked the syntax, it's time to make sure your state machine works as you expect. Let's make a testbench!

Create a Synchronous Testbench

4. In Vivado, create a new testbench in your project called "MooreElevatorController_tb.vhd" and associate it with your Moore state machine design. A template has been provided to you with much of the testbench complete.
5. Write a testbench that will start your elevator at floor 1 (hint: i_reset <= '1') and go up to floor 4
6. The stop signal should not have to be asserted on floor 4 (when going Up) for the elevator to stop there.
7. Once stopped on floor 4, have your testbench return to floor 1 without stopping.
8. The stop signal should not have to be asserted on floor 1 (when going Down) for the elevator to stop there.
    a. An example of what a good waveform looks like (without the output) can be seen in Figure 2 below

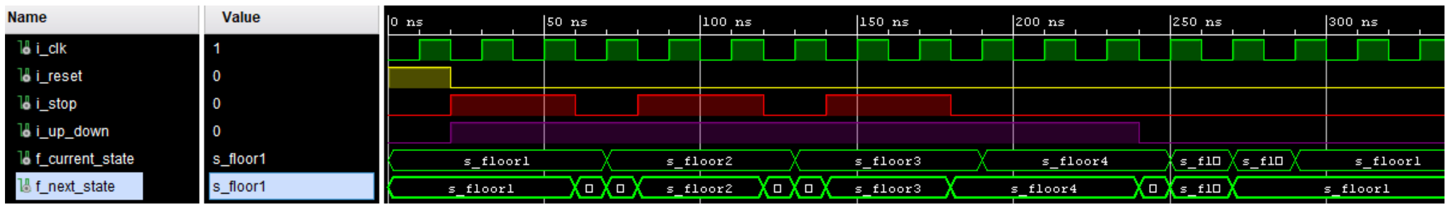

Figure 2. Example simulation waveform with output removed

When you have finished, submit a screenshot of your completed test bench to your instructor by pushing to your git repository! Be sure to display both the current states and the outputs!