Jianyu Tao

CSE 143

Final Project Report

Jun 14, 2020

Elevator Controller Project Report

Introduction

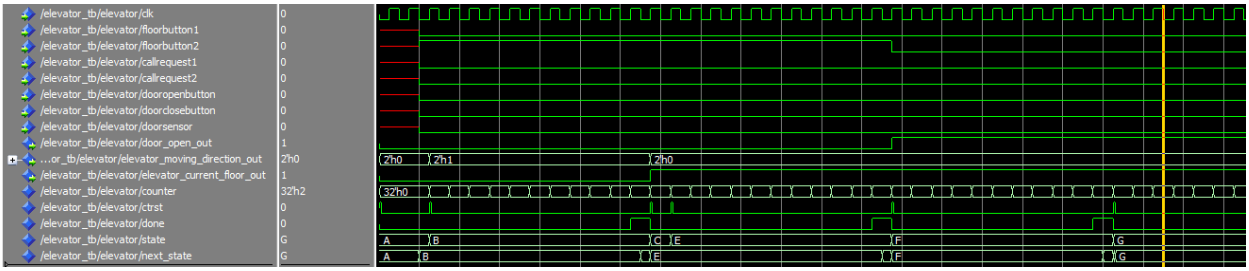
The goal of this project is to design an elevator controller that is similar to a real-world elevator. Therefore, this elevator must contain some most basic features. For example, the elevator can respond to call request outside the elevator and floor button inside the elevator, and it can also respond to open/close door button conditionally. It will automatically close door if idling for a specific of time, and it will not continue closing the door if the door gets stuck by something. I begin with designing a two-level elevator and after that I expand my design to a four-level elevator.

Description of the 2-level elevator controller

Implement the controller with finite state machine. There are 7 states in total: A: Staying at first floor. B: Going up from first floor to second floor. C: Staying at second floor. D: Going down from second floor to first floor. E: Opening the door. F: Door staying open. G: Closing the door. There will be a timer to simulate the physical movement. To be convenient for testing, all kind of physical movement will have the same time in demo. The input of this design includes the following: clock input, reset input, floorbutton1, floorbutton2, callrequest1, callrequest2, dooropenbutton, doorclosebutton, doorsensor. The output of this design includes the following: door status, elevator moving direction, elevator current floor. I also use some local variables and signals in the design. There is an integer signal counter to count for the clock cycle, and a constant value timer which indicates the time cost of physical movement. There is a done flag which indicates when counter is bigger than timer. There is also state and next state signal to indicates the current state and next state. I use 5 concurrent processes in the architecture. First, a sequential state machine process to update state to next_state at every clock cycle. Second, a combinational next state process to update next_state based on current state, current input, current local variables/signals. Third, a combinational output process to update the output signals using current state. Forth, a sequential counter process to increment the counter every clock, if counter reset signal received, reset the counter to 0, if counter is bigger than timer, then output done flag. Fifth, a sequential counter reset process to set counter reset signal at every beginning of each new state.

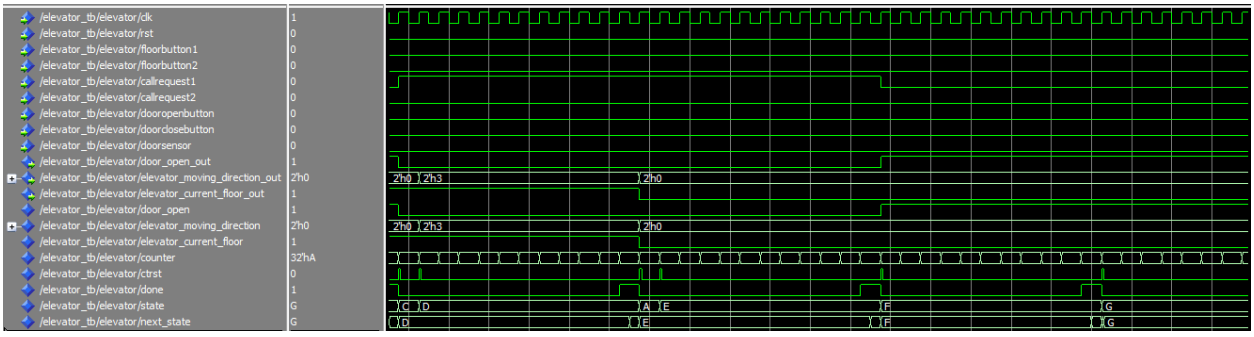
Simulation Results:

1. Press the floorbutton2 when elevator is staying at the first floor:



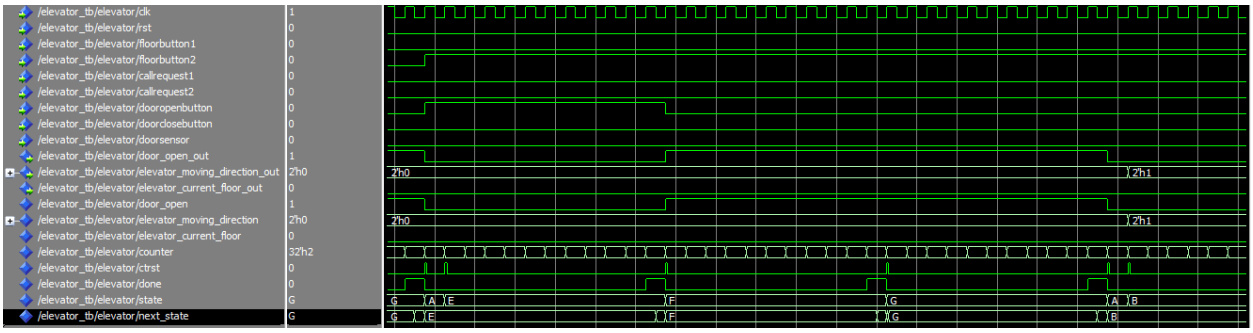
As expected, the elevator will go up from the first floor to the second floor. Then open the door, waiting for the counter, and close the door.

2. Press the callrequest1 to call the elevator back to the first floor:

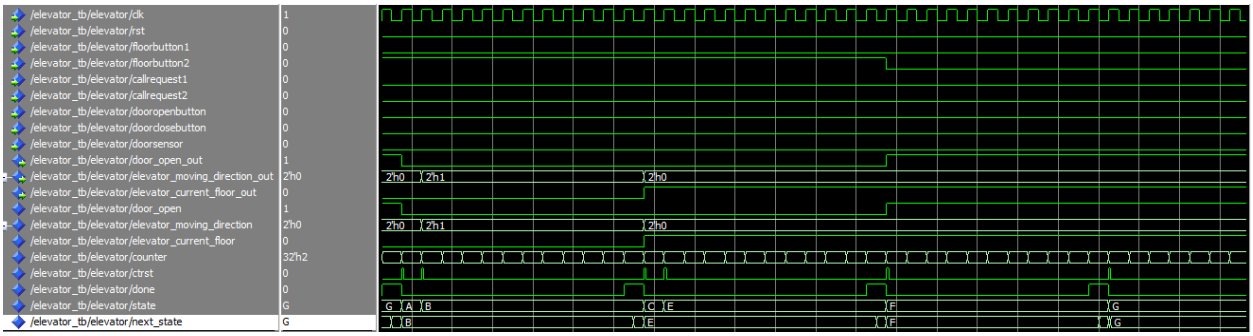


As expected, the elevator goes down from the second floor to the first floor. Then open the door, waiting for the counter, and close the door.

3. Press the floorbutton2 and open-door button.

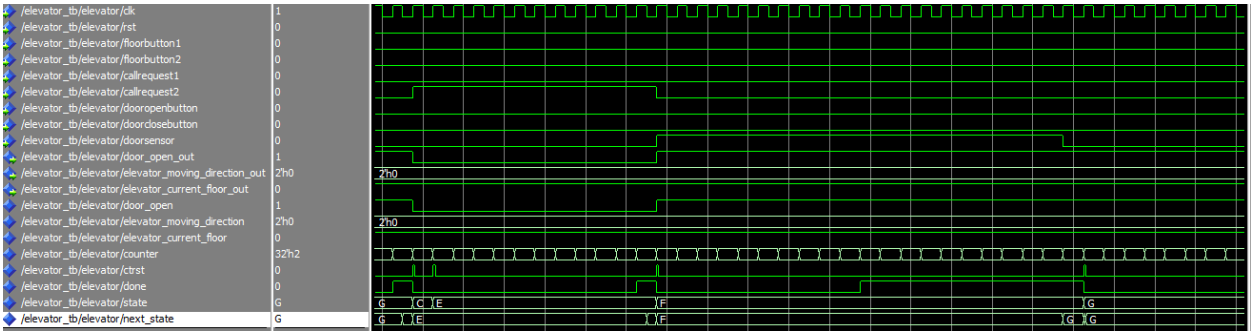


In this way, the door will first open and close.



Then the elevator will go up to the second floor.

4. When the door is jammed by something



The door will keep open even if the done flag is set until door sensor becomes 0.

Critically Analysis:

This elevator controller makes elevator to stop for one clock cycle at every floor even if the elevator will not actually stop at this floor. This seems to be inconsistent with real world elevator, but I think it can be easily addressed by letting elevator stop when the staying state last for more than 1 clock cycle.

When the elevator is closing the door, it will open the door immediately when the doorsensor is on or dooropenbutton is pressed. However, the time used to open the door will not decrease even though the door is not fully closed. This is also inconsistent to the real world elevator, but I think in real word, the elevator will have a sensor which indicates whether the door is fully closed or open, instead of using timer to simulate the physical movement.

Expand 2-level Elevator to 4-level Elevator

Introduction

The change is that at second floor and third floor, there are two call request buttons for each floor. One is up button and the other is down button, indicating whether the users want to go up or down. In this situation, if the users press the up button, and the moving direction of the elevator is down, even though the elevator reaches the same floor, it will not stop and open the door since the elevator is going down. In conclusion, for 4-level elevator, elevator moving direction is not just an output, it can also determine whether or not the elevator will stop in respond to request button.

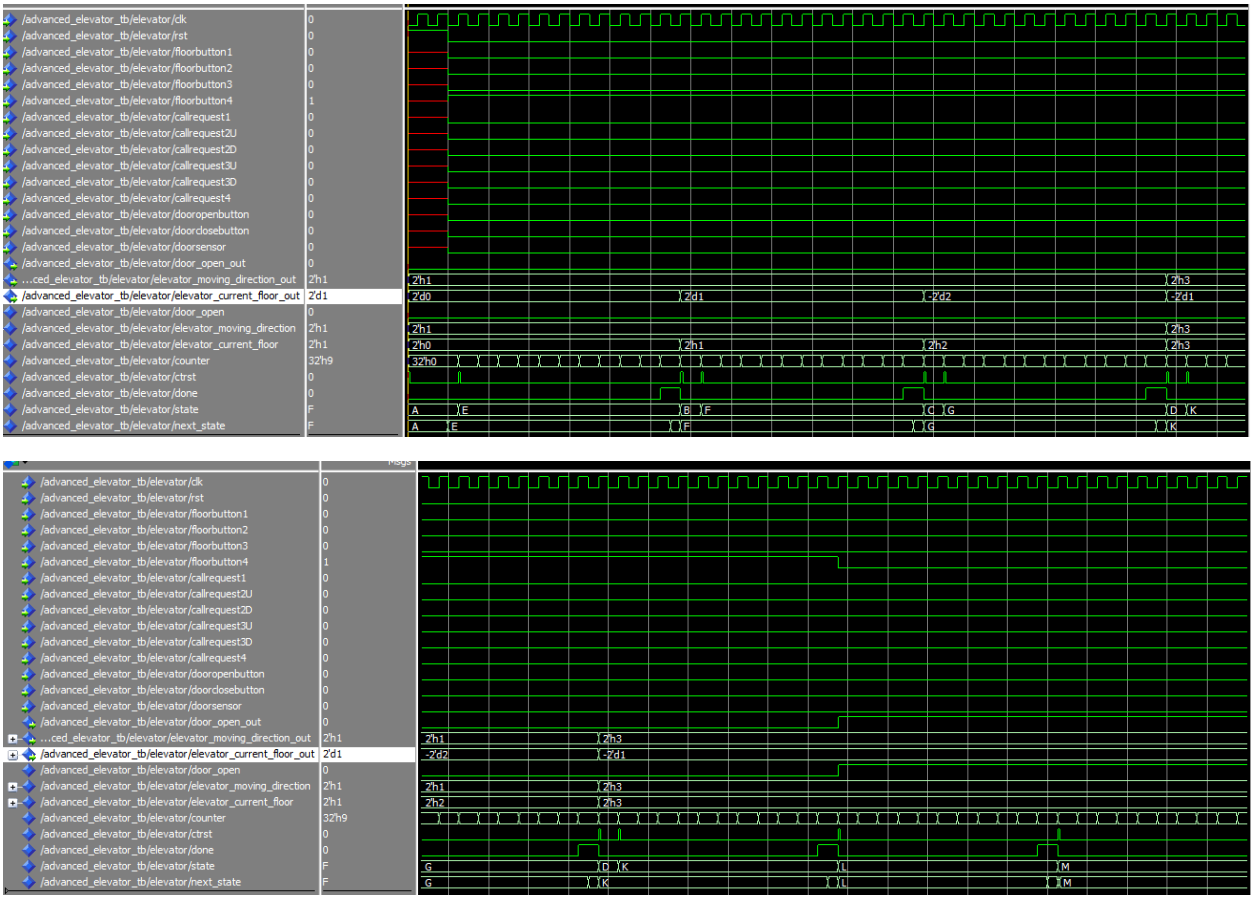
Description of 4-level elevator controller:

Similar to 2-level elevator controller, I use finite state machine to implement. There are 13 states in total: A: staying at floor 1, B: staying at floor 2, C: staying at floor 3, D: staying at floor 4, E: going up from 1 to 2, F: going up from 2 to 3, G: going up from 3 to 4, H: going down from 4 to 3, I: going down from 3 to 2, J: going down from 2 to 1, K: opening the door, L: door staying open, M: closing the door. There will be a timer to simulate the physical movement. To be

convenient for testing, all kind of physical movement will have the same time in demo. The input of this design includes the following: clock input, reset input, floorbutton1, floorbutton2, floorbutton3, floorbutton4, callrequest1, callrequest2U, callrequest2D, callrequest3U, callrequest3D, callrequest4, dooropenbutton, doorclosebutton, doorsensor. ‘U’ denotes ‘up’, and ‘D’ denotes ‘down’. The output of this design includes the following: door status, elevator moving direction, elevator current floor. I use 6 concurrent processes in the architecture. First, a sequential state machine process to update state to next_state at every clock cycle. Second, a combinational next state process to update next_state based on current state, current input, current local variables/signals. Third, a combinational output process to update the output signals using current state. Forth, a sequential counter process to increment the counter every clock, if counter reset signal received, reset the counter to 0, if counter is bigger than timer, then output done flag. Fifth, a sequential counter reset process to set counter reset signal at every beginning of each new state. Sixth, combinational moving direction process to determine the moving direction of the elevator based on all the input button.

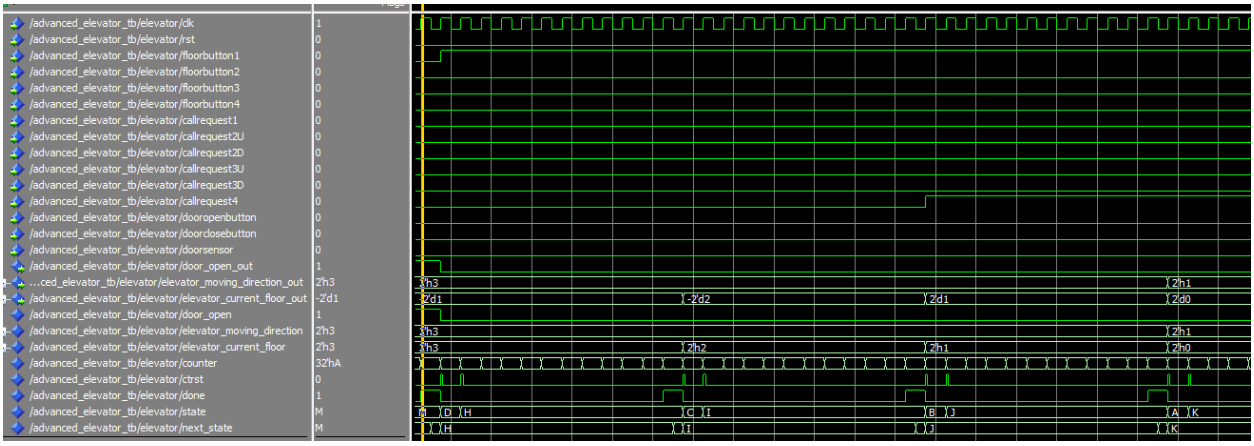
Simulation Results:

1. Press the floorbutton4:

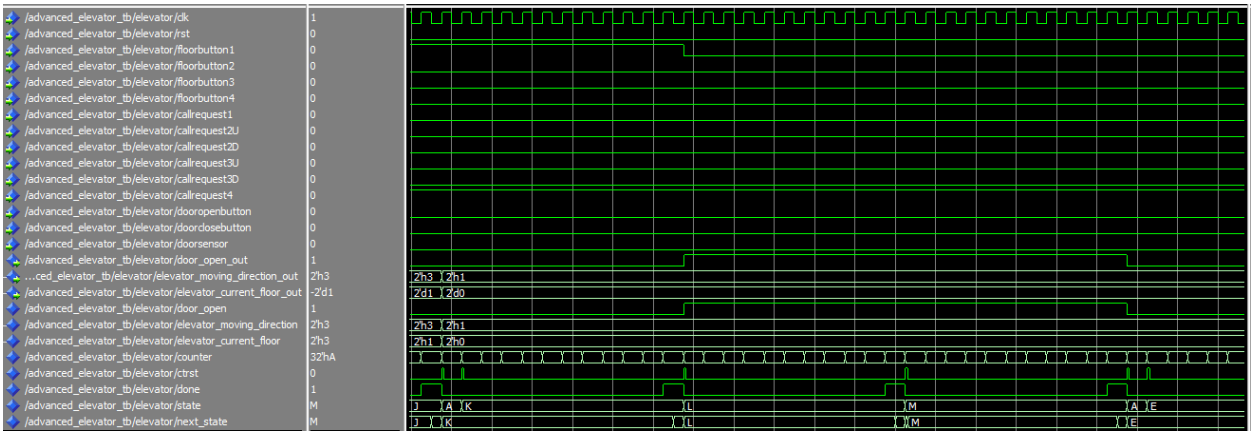


As expected, the elevator will go up to forth floor, passing through second and third floor. Then open the door, waiting for the counter, and close the door.

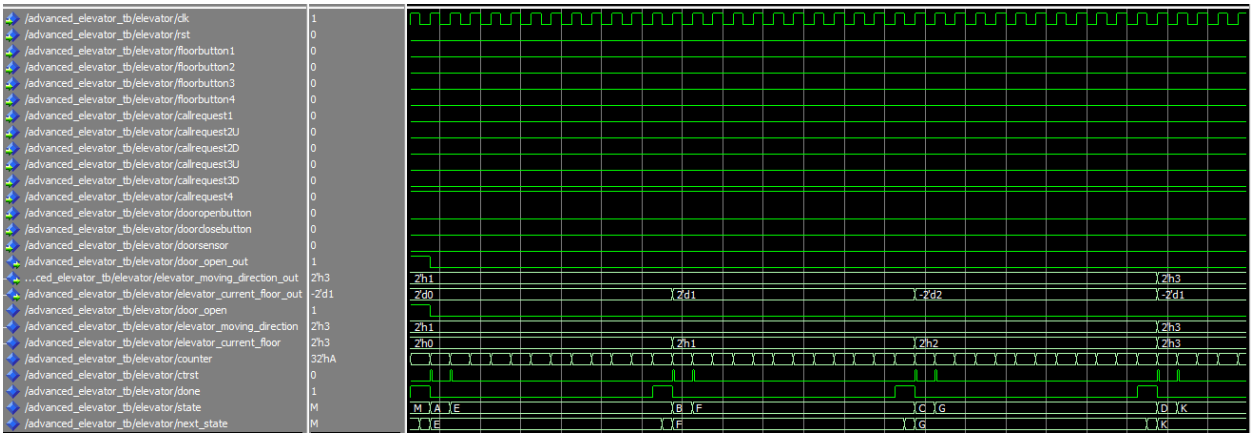
2. Press the floorbutton1, when reaching second floor, press callrequest4:



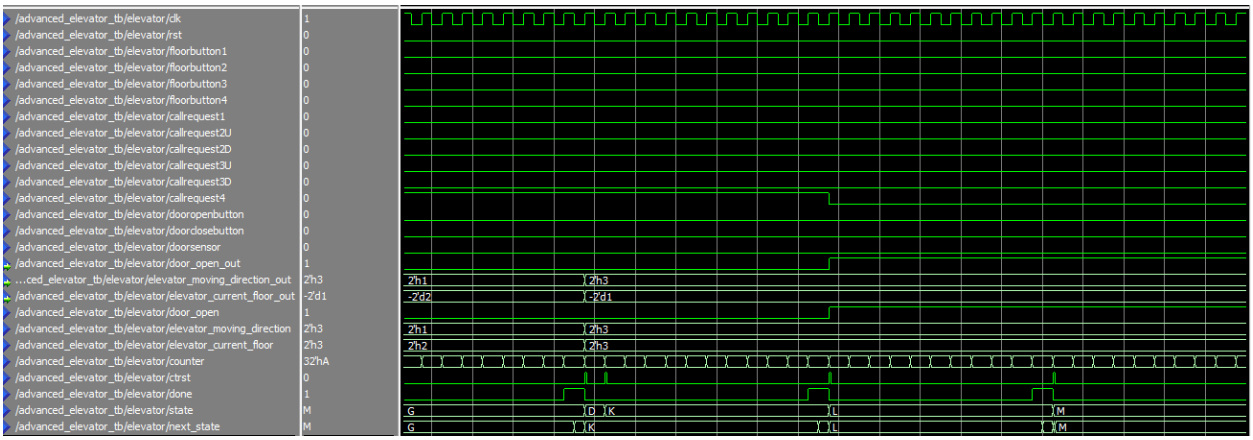
(the elevator going down to the first floor)



(the elevator opens the door and closes the door at the first floor)



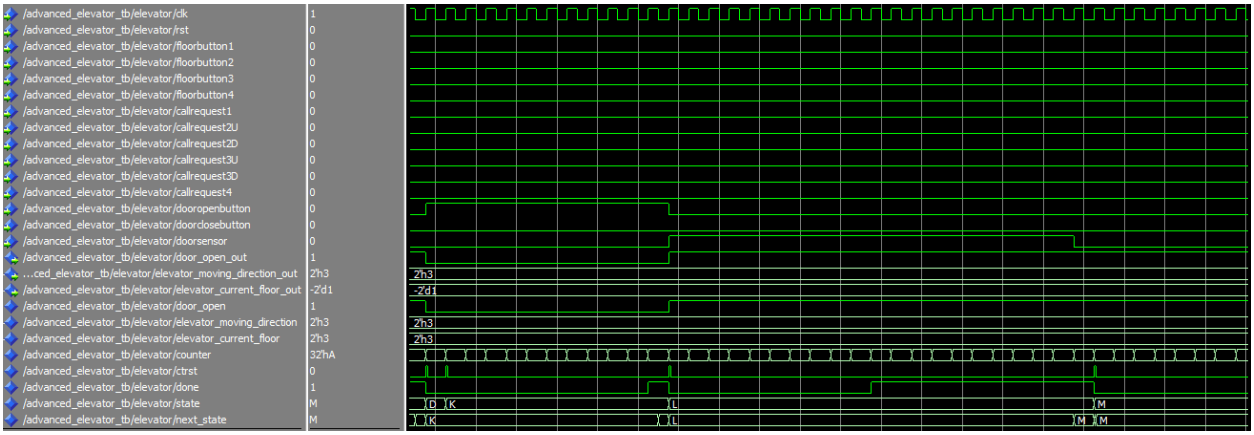
(the elevator goes up to the fourth floor)



(the elevator opens and closes the door at fourth floor)

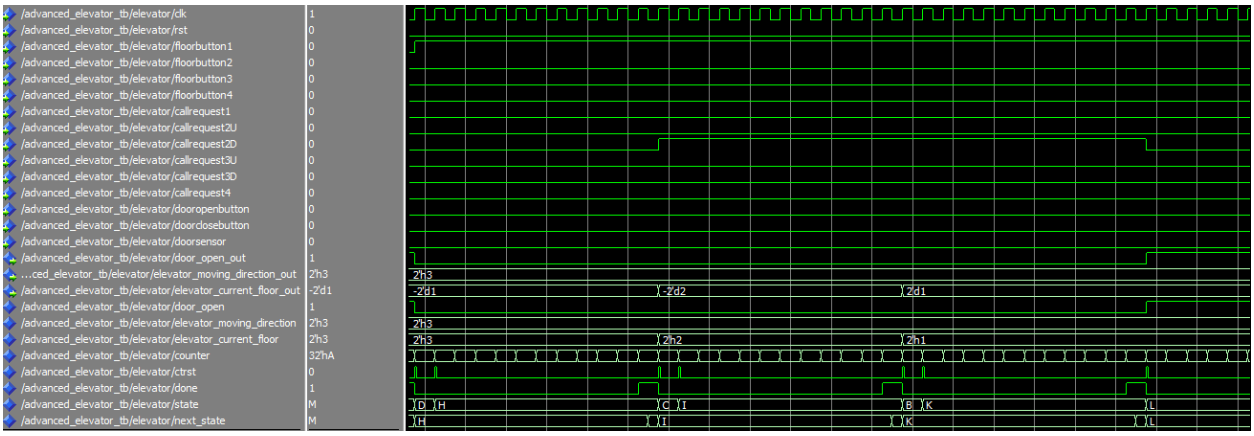
As expected, even though the callrequest4 is pressed when elevator reaches second floor, the elevator will not go up immediately since the current moving direction is down. Therefore, the elevator will first reach the first floor, then the fourth floor.

3. When the door is jammed by something

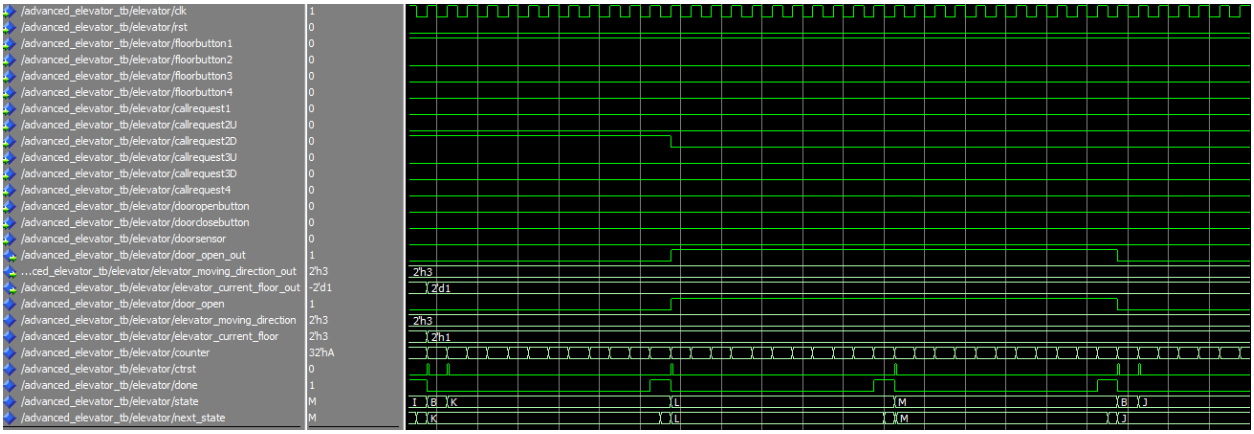


Same as the 2-level elevator, the door will not close until the door sensor is off.

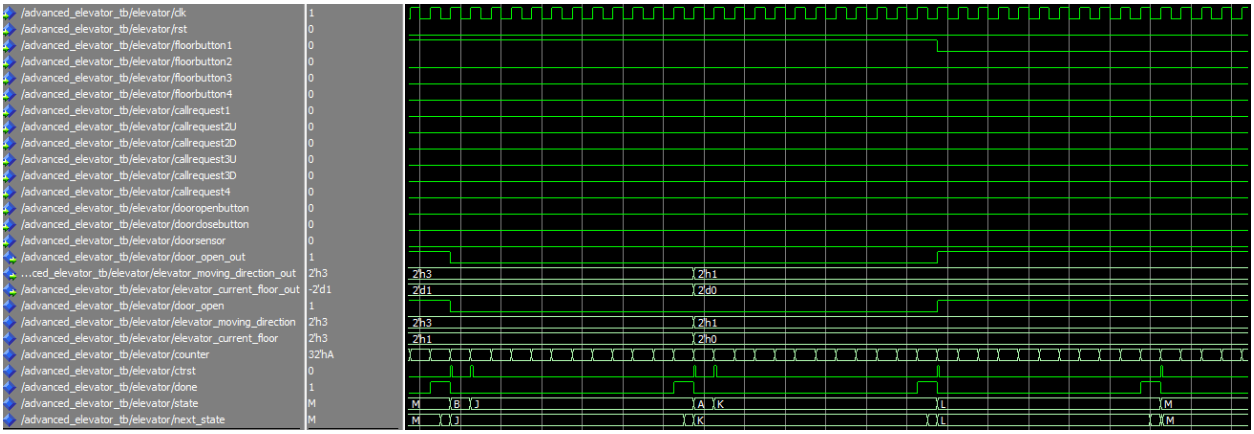
4. Press the floorbutton1, when the elevator reaches third floor, press callrequest2D.



(the elevator goes down to second floor)



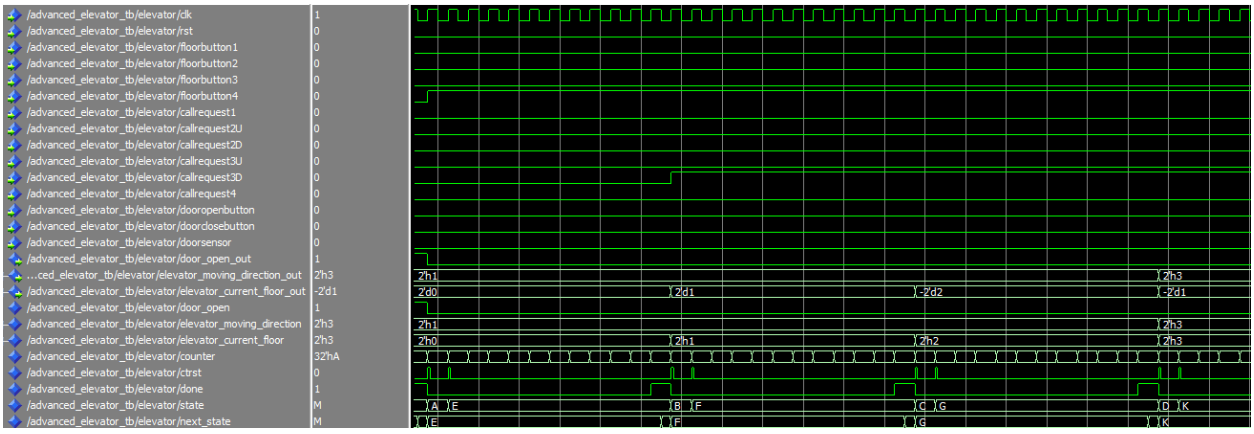
(the elevator opens and closes the door at second floor)



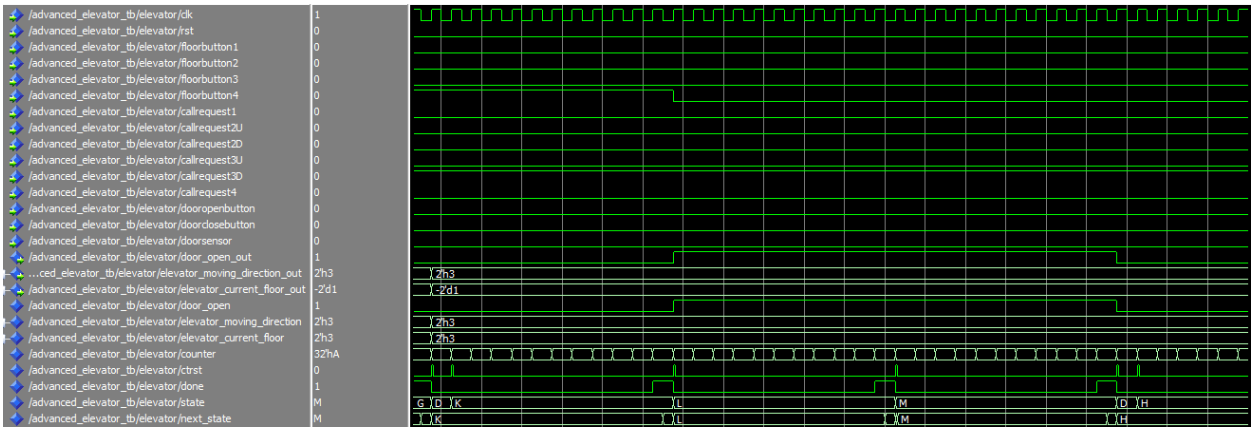
(the elevator reaches first floor and open the door)

As expected, since the callrequest2D is at the same direction of elevator current direction, the elevator will stop at second floor first instead of first stopping at first floor then second floor.

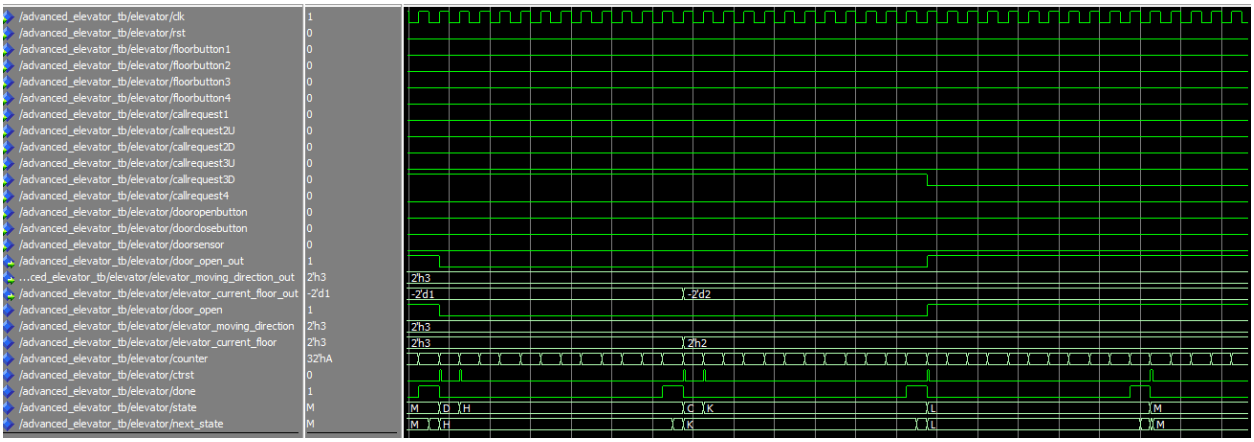
5. Press the floorbutton4, when the elevator reaches second floor, press callrequest3D.



(the elevator going up directly to fourth floor)



(open the door at fourth floor)



(go down to third floor, then open the door)

As expected, since the call request button is at opposite direction of the elevator moving direction, the elevator will ignore the request button at third floor. After opening the door at fourth floor, the elevator will go back to third floor.

Critical Analysis:

This elevator controller makes elevator to stop for one clock cycle at every floor even if the elevator will not actually stop at this floor. This seems to be inconsistent with real world elevator, but I think it can be easily addressed by letting elevator stop when the staying state last for more than 1 clock cycle.

When the elevator is closing the door, it will open the door immediately when the doorsensor is on or dooropenbutton is pressed. However, the time used to open the door will not decrease even though the door is not fully closed. This is also inconsistent to the real world elevator, but I think in real word, the elevator will have a sensor which indicates whether the door is fully closed or open, instead of using timer to simulate the physical movement.

I test this by trying a lot of edge conditions in the testbench, so the basic function of this elevator controller should be robust. However, this elevator controller doesn't modify the input signals, which means when the floorbutton1 is pressed and the elevator reaches the first floor, the floorbutton1 will not become 0 automatically like in real world. The floorbutton1 is set to be 0 manually in the testbench. If not doing this, the door will repeat to open and close forever.