VLSI Project: Elevator Control System

Part 1

Name: Prajwal Paul & Khoa Diep

ktd170030

psp170030

CE 6325

Instructor: Prof. Sechen

Date: 9/9/2023

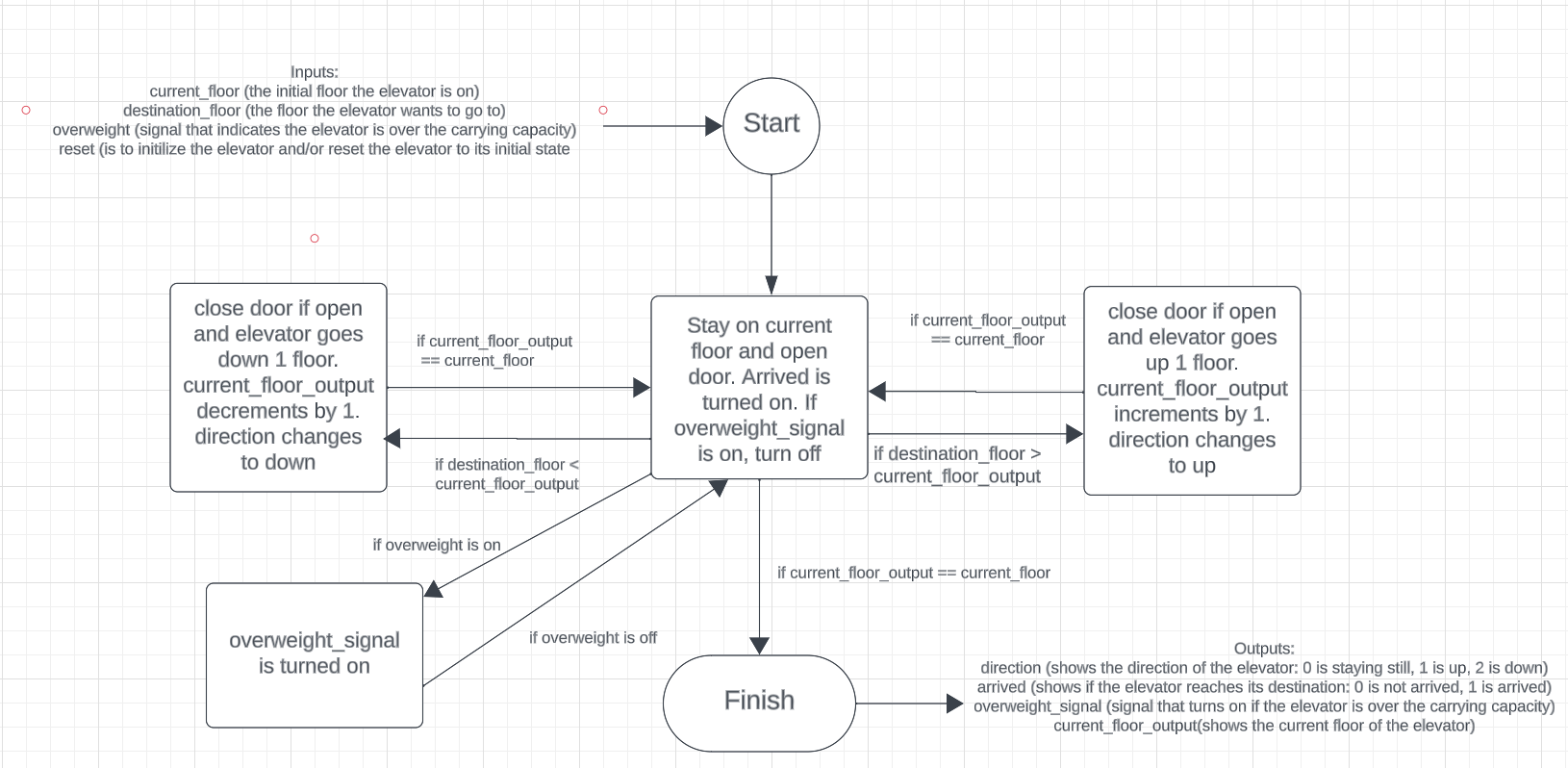
**Abstract**

The general purpose of this project is to create an elevator control system that takes into account the current floor where the elevator is at, the destination floor that the user is trying to get to and whether the elevator is over the weight load limit which prohibits movement. On the positive edge of the clock, we run through various switch cases, set up in descending priority, to determine what the appropriate action is. The first case is a reset function that helps reset the elevator system. This command is used as a “new command is given” method and resets the arrived at destination and overweight signals, while also setting the current floor where the elevator is at to the output. The next case is when there is too heavy of a load on the elevator. Currently this number is arbitrary but will be set to 3000lbs. In this action, the overweight signal is turned on, the arrived at destination signal is forced on, opening the doors and the direction is set to 0, which means no movement. The next case is when the targeted floor is above the current floor and the load on the elevator is not overweight. In this action, the direction is set to 1 meaning going up and the elevator moves up one level per clock cycle, which is done by incrementing the current floor output by one. The arrived at destination signal is kept off during movement. Also, as a precaution, the overweight signal is turned off as the weight check has already been completed. The next case is when the targeted floor is below the current elevator floor. Here, the direction is set to 2 meaning going down and the elevator goes down a floor per clock cycle by decrementing the floor output by one. The arrived at destination signal is kept off during movement here as well. Also, as a precaution, the overweight signal is turned off as the weight check has already been completed. The final case is when the elevator has arrived at the targeted floor, where the arrived signal is turned on to open the doors and the direction is set to 0 which means no movement.

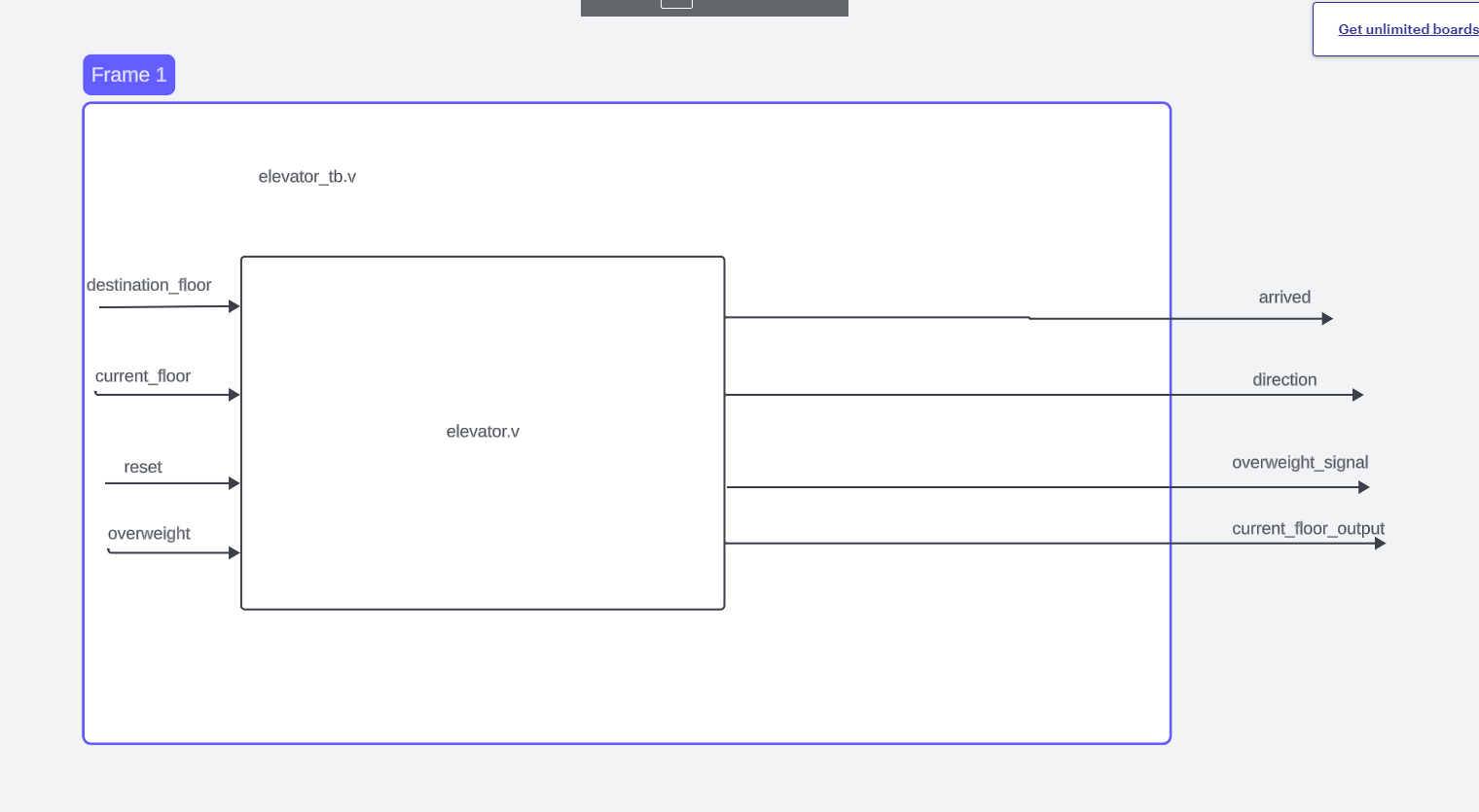
In the testbench for this project, we run through the different cases and manipulate the different inputs to ensure proper function. We start with a reset of all signals, then give a new command but with no targeted floor so the elevator doesn't move. Next, we request the elevator to move up to the top floor from the bottom floor and press the new command or reset button. After giving enough clock cycles, the elevator arrives at the destination and we check the outputs of direction and arrival during this time. After this, we make the elevator go back to the ground floor while checking all the outputs during function. As the final check, we turn on the overweight, which outputs the overweight signal and request a floor change, which does not happen and the doors stay open. Only after the overweight signal is changed and the command button is pressed does the elevator begin moving to the directed floor. With this, we conduct a check of all possible inputs and outputs for the elevator control system.

In terms of scaling up this design, there are a wide variety of options. The first would obviously be to increase the number of floors that the elevator can cover. Various skyscrapers have 40+ floors and this would increase the bit count and bus width when passing data. The next is to have a dual to quad elevator system that controls all the elevators together. An application would be when one elevator is requested at the 3rd floor and another on the 5th floor, the system would send one to each floor and not have to send both to each floor. Another would be to have an emergency stop button and freezes the elevator at the current location it's at, be it in between floors or on a floor. This is a very important function that is necessary in times of danger such as an earthquake or fire. Another is the button for maintenance that stops that specific elevator from moving. Ideally this would be used in a multiple elevator system, so that the remaining elevators are still able to be used. The next would be an alarm button that would sound the alarm for the control system overall and also place a call with the fire department who are normally contacted during an elevator emergency. The final and most basic function is the door close and open function. Normally, after a floor is pressed, the system waits a bit then closes the door, but with this we can speed this up or even slow it down to allow for more time for passengers to get on the elevator.

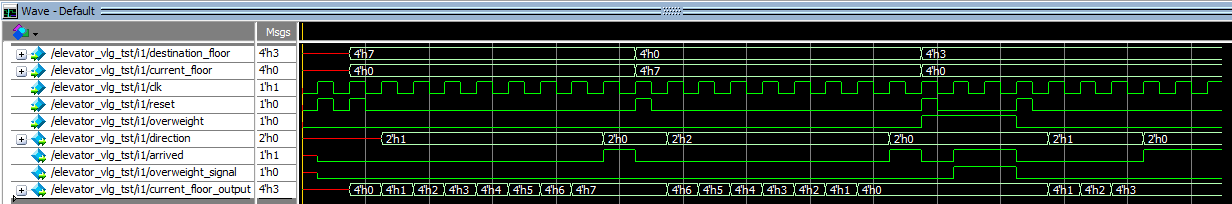
**Design: Block Diagram**



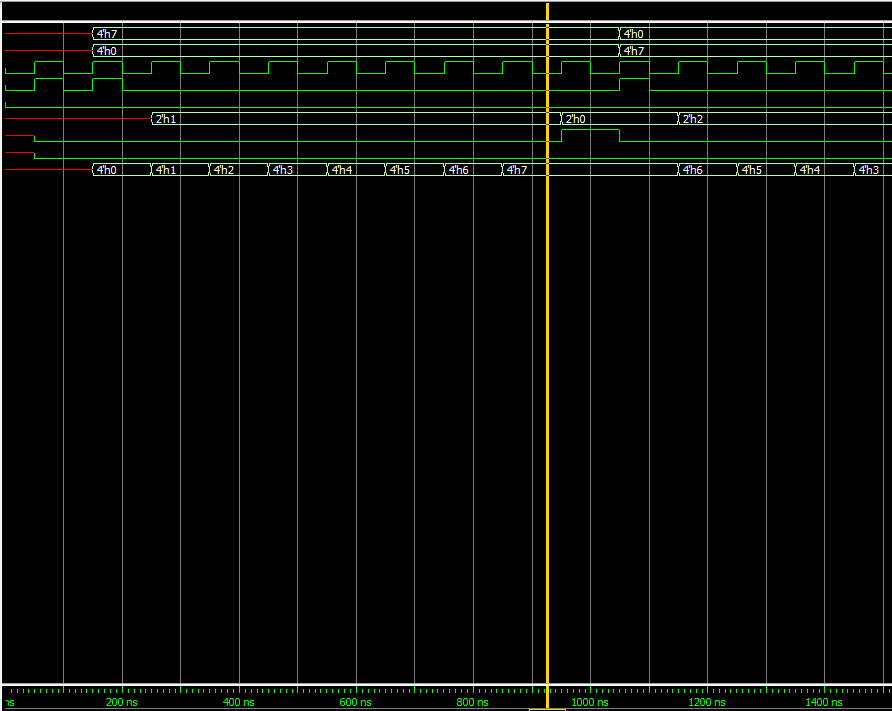
**Modules & Test Bench: Block Diagram**

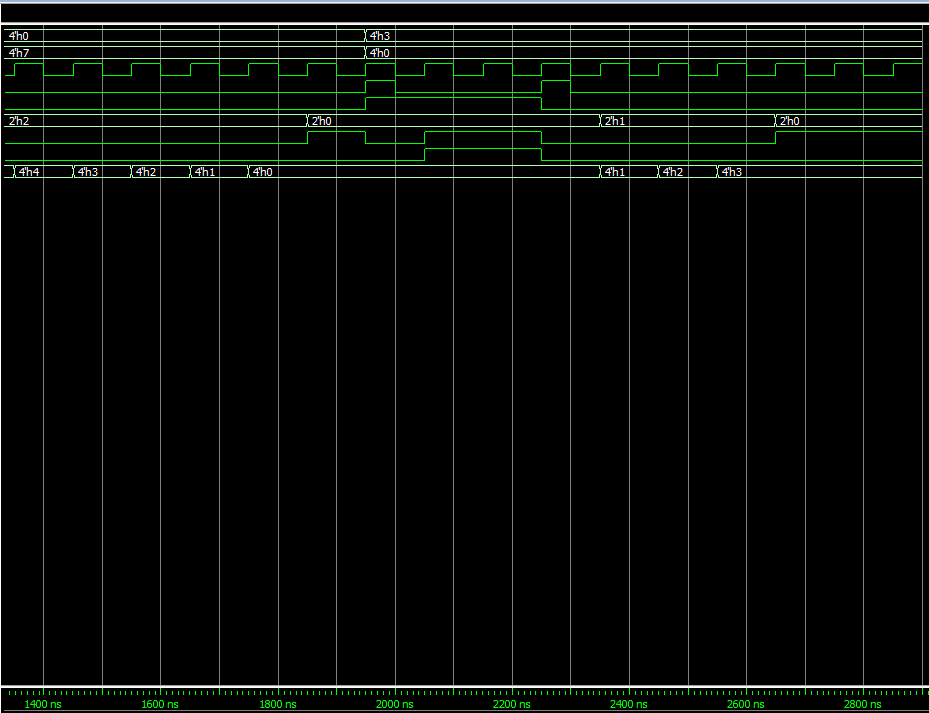


**Waveform Results**



When destination\_floor = 7 and current\_floor = 0, since the elevator is not overweight (overweight = 0), it will start moving up from 0 to 7. This is shown in our waveform when current\_floor\_output is incrementing near the beginning of the waveform, as well as direction is equal to 1. When current\_floor\_output reaches 7, direction changed to 0 (indicating not moving) and arrived changed from 0 to 1 (showing that the elevator reached its destination). Similarly, when destination\_floor = 0 and current\_floor = 7, it will start moving from down from 7 to 0. This is shown in our waveform when current\_floor\_output is decreasing as well as direction is changing to 2. When current\_floor\_output reaches 7, direction changed to 0 (indicating not moving) and arrived changed from 0 to 1 (showing that the elevator reached its destination).

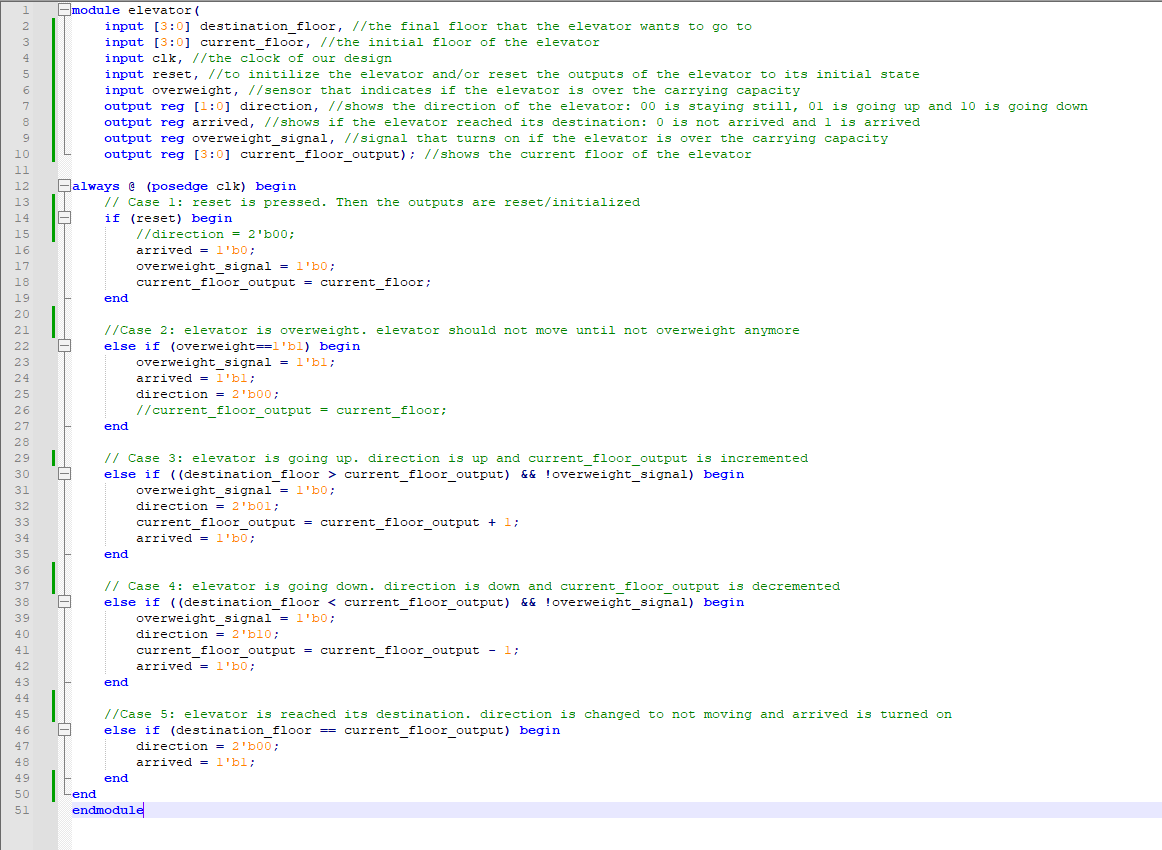




When the elevator is overweight, it will not move until it is not overweight anymore. It will start moving a clock cycle after it is not overweight.

**Code**

Screenshot of elevator.v (design)



Code of elevator\_tb.v

module elevator(

input [3:0] destination\_floor, //the final floor that the elevator wants to go to

input [3:0] current\_floor, //the initial floor of the elevator

input clk, //the clock of our design

input reset, //to initilize the elevator and/or reset the outputs of the elevator to its initial state

input overweight, //sensor that indicates if the elevator is over the carrying capacity

output reg [1:0] direction, //shows the direction of the elevator: 00 is staying still, 01 is going up and 10 is going down

output reg arrived, //shows if the elevator reached its destination: 0 is not arrived and 1 is arrived

output reg overweight\_signal, //signal that turns on if the elevator is over the carrying capacity

output reg [3:0] current\_floor\_output); //shows the current floor of the elevator

always @ (posedge clk) begin

// Case 1: reset is pressed. Then the outputs are reset/initialized

if (reset) begin

//direction = 2'b00;

arrived = 1'b0;

overweight\_signal = 1'b0;

current\_floor\_output = current\_floor;

end

//Case 2: elevator is overweight. elevator should not move until not overweight anymore

else if (overweight==1'b1) begin

overweight\_signal = 1'b1;

arrived = 1'b1;

direction = 2'b00;

//current\_floor\_output = current\_floor;

end

// Case 3: elevator is going up. direction is up and current\_floor\_output is incremented

else if ((destination\_floor > current\_floor\_output) && !overweight\_signal) begin

overweight\_signal = 1'b0;

direction = 2'b01;

current\_floor\_output = current\_floor\_output + 1;

arrived = 1'b0;

end

// Case 4: elevator is going down. direction is down and current\_floor\_output is decremented

else if ((destination\_floor < current\_floor\_output) && !overweight\_signal) begin

overweight\_signal = 1'b0;

direction = 2'b10;

current\_floor\_output = current\_floor\_output - 1;

arrived = 1'b0;

end

//Case 5: elevator is reached its destination. direction is changed to not moving and arrived is turned on

else if (destination\_floor == current\_floor\_output) begin

direction = 2'b00;

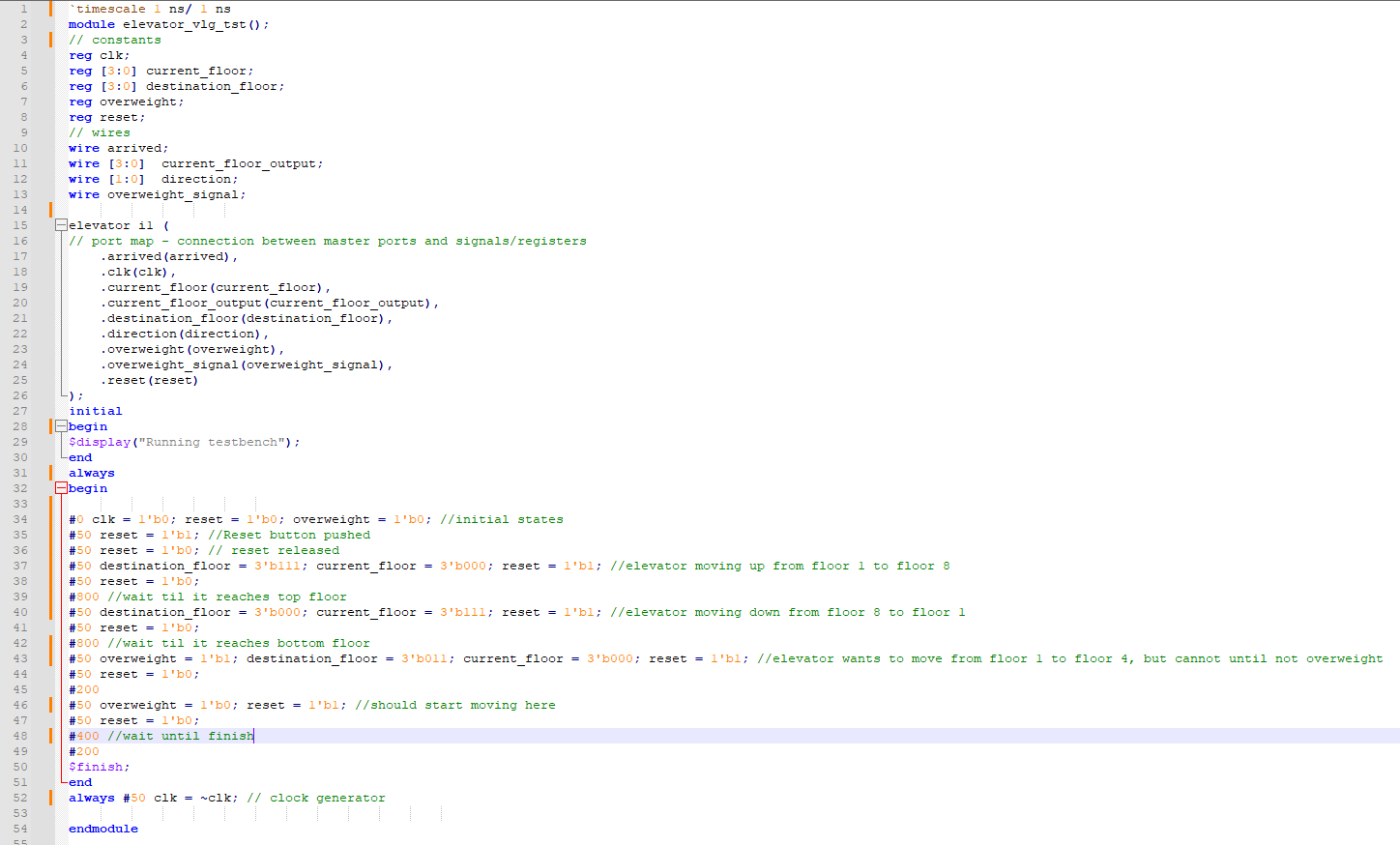
arrived = 1'b1;

end

end

endmodule

Screenshot of elevator\_tb.v (testbench)



Code of elevator\_tb.v

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

`timescale 1 ns/ 1 ns

module elevator\_vlg\_tst();

// constants

reg clk;

reg [3:0] current\_floor;

reg [3:0] destination\_floor;

reg overweight;

reg reset;

// wires

wire arrived;

wire [3:0] current\_floor\_output;

wire [1:0] direction;

wire overweight\_signal;

elevator i1 (

// port map - connection between master ports and signals/registers

.arrived(arrived),

.clk(clk),

.current\_floor(current\_floor),

.current\_floor\_output(current\_floor\_output),

.destination\_floor(destination\_floor),

.direction(direction),

.overweight(overweight),

.overweight\_signal(overweight\_signal),

.reset(reset)

);

initial

begin

$display("Running testbench");

end

always

begin

#0 clk = 1'b0; reset = 1'b0; overweight = 1'b0; //initial states

#50 reset = 1'b1; //Reset button pushed

#50 reset = 1'b0; // reset released

#50 destination\_floor = 3'b111; current\_floor = 3'b000; reset = 1'b1; //elevator moving up from floor 1 to floor 8

#50 reset = 1'b0;

#800 //wait til it reaches top floor

#50 destination\_floor = 3'b000; current\_floor = 3'b111; reset = 1'b1; //elevator moving down from floor 8 to floor 1

#50 reset = 1'b0;

#800 //wait til it reaches bottom floor

#50 overweight = 1'b1; destination\_floor = 3'b011; current\_floor = 3'b000; reset = 1'b1; //elevator wants to move from floor 1 to floor 4, but cannot until not overweight

#50 reset = 1'b0;

#200

#50 overweight = 1'b0; reset = 1'b1; //should start moving here

#50 reset = 1'b0;

#400 //wait until finish

#200

$finish;

end

always #50 clk = ~clk; // clock generator

endmodule

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_