2022-2023 FALL CS 303 TERM PROJECT

Elevator Control System

Yusuf Kılıç – 29431

Mustafa Adnan Arasan - 29413

DATE 22.01.2022

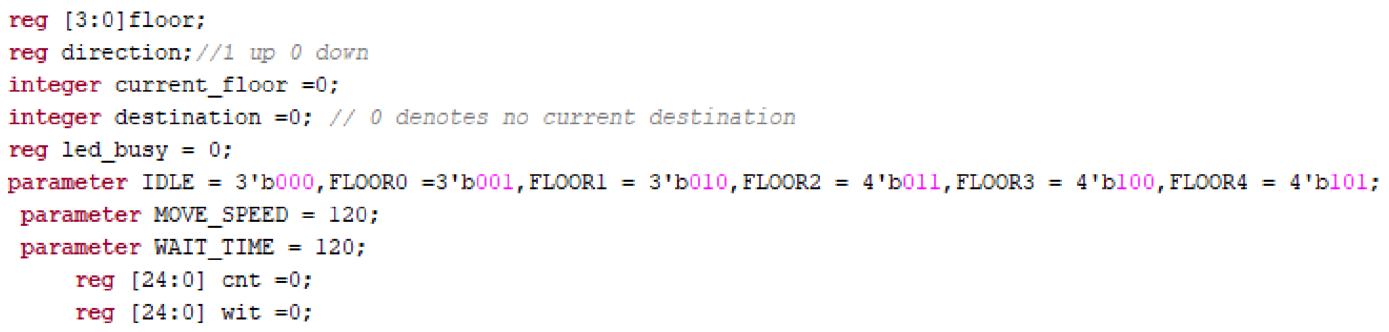
Introduction and Purpose

The aim of the project is to design and simulate an elevator control system obtained by creating a state machine via Verilog HDL over Vivado and then implement it on the Artix-7 FPGA Board.

Design and Simulation Phase

Design:

Variables and Their Usage:



```
reg [3:0]floor;
reg direction;//1 up 0 down
integer current_floor =0;
integer destination =0; // 0 denotes no current destination
reg led_busy = 0;
parameter IDLE = 3'b000,FLOOR0 =3'b001,FLOOR1 = 3'b010,FLOOR2 = 4'b011,FLOOR3 = 4'b100,FLOOR4 = 4'b101;
 parameter MOVE_SPEED = 120;
 parameter WAIT_TIME = 120;
     reg [24:0] cnt =0;
     reg [24:0] wit =0;
```

Floor is used store Parameters and when it changes state of the algorithm changes according to it.

Direction is for up and down.

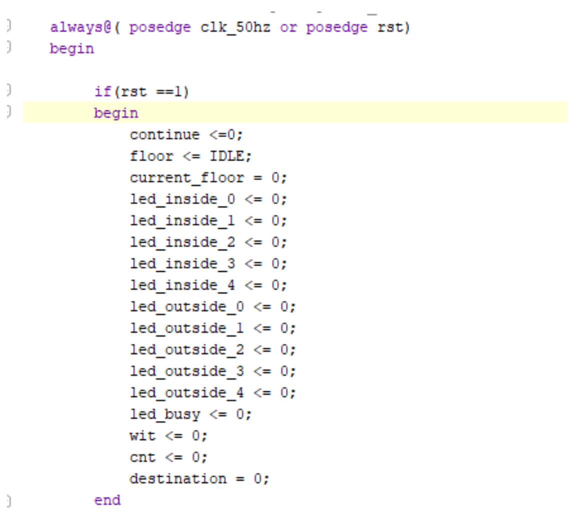
Current_floor and destination is used to check conditions for lighting up leds outside and inside the elevator.

Move_SPEED and WAIT_TIME is for the registers cnt and wit, together they decide how fast the elevator will be and how long it should wait.

Please note that that led_busy here is not the input for led_busy in the simulation. It is replaced by continue in the inputs.

State Machine:

If rst button is asserted as, one reset every output and variables.

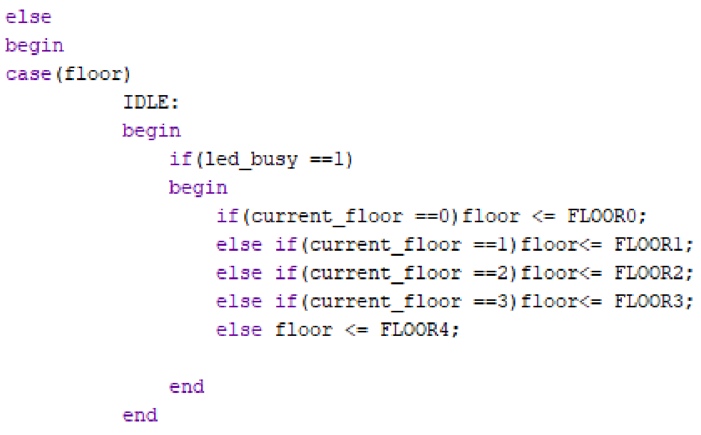


```
always@( posedge clk_50hz or posedge rst)
begin

     if(rst ==1)
     begin
         continue <=0;
         floor <= IDLE;
         current_floor = 0;
         led_inside_0 <= 0;
         led_inside_1 <= 0;
         led_inside_2 <= 0;
         led_inside_3 <= 0;
         led_inside_4 <= 0;
         led_outside_0 <= 0;
         led_outside_1 <= 0;
         led_outside_2 <= 0;
         led_outside_3 <= 0;
         led_outside_4 <= 0;
         led_busy <= 0;
         wit <= 0;
         cnt <= 0;
         destination = 0;
     end
```

Idle state:

Elevator starts in Idle state and goes to other states when led_busy is one (not the led for output).



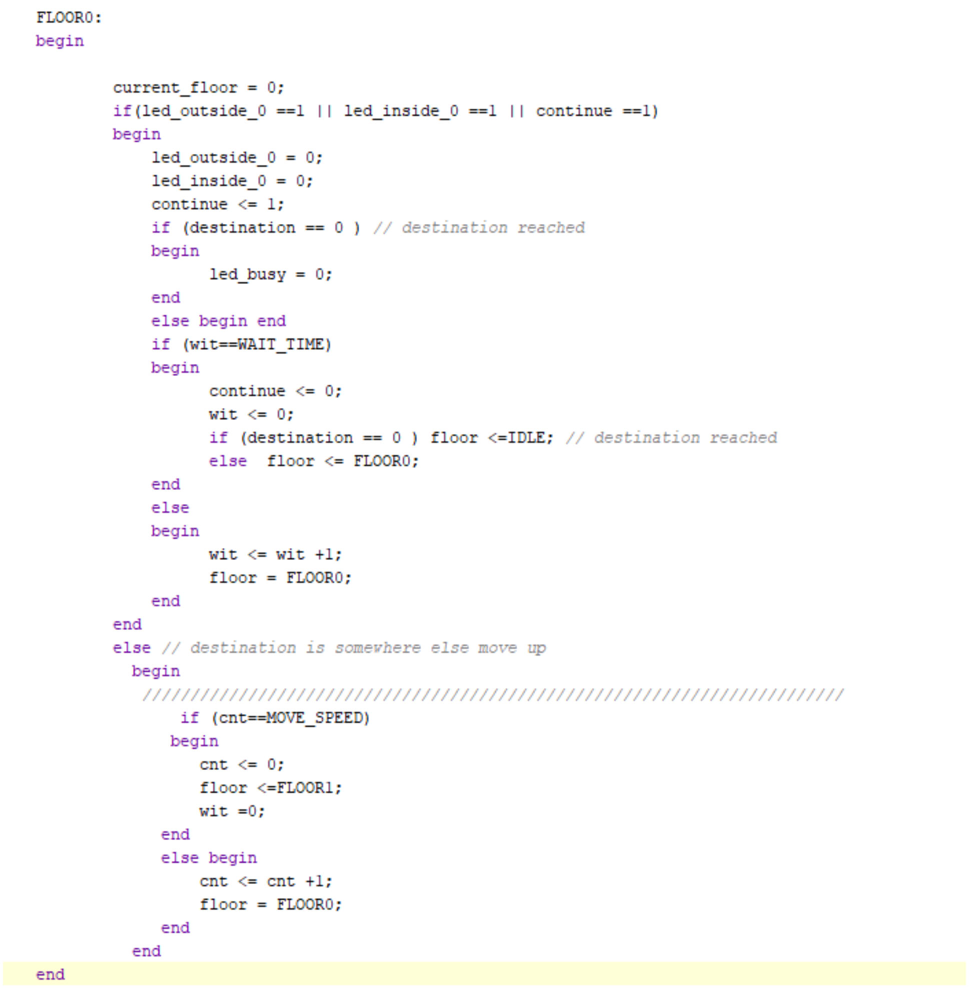
```
else
begin
case(floor)
         IDLE:
         begin
             if(led_busy ==1)
             begin
                 if(current_floor ==0)floor <= FLOOR0;
                 else if(current_floor ==1)floor<= FLOOR1;
                 else if(current_floor ==2)floor<= FLOOR2;
                 else if(current_floor ==3)floor<= FLOOR3;
                 else floor <= FLOOR4;

             end
         end
```

Floor states:

All of them are similar if not the same so only one of them will be described. When continue (actual led_busy) is one or the leds in that floor is one continue(led_busy) will turn on and elevator will wait for 5 seconds. (wit==WAIT_TIME) After waiting for 5 seconds if destination is reached elevator will go to Idle mode and if destination is not that floor elevator will start moving.

After 5 seconds it will reach the next floor (cnt == MOVE_SPEED).



```
FLOOR0:
begin

        current_floor = 0;
        if(led_outside_0 ==1 || led_inside_0 ==1 || continue ==1)
        begin
            led_outside_0 = 0;
            led_inside_0 = 0;
            continue <= 1;
            if (destination == 0 ) // destination reached
            begin
                  led_busy = 0;
            end
            else begin end
            if (wit==WAIT_TIME)
            begin
                  continue <= 0;
                  wit <= 0;
                  if (destination == 0 ) floor <=IDLE; // destination reached
                  else  floor <= FLOOR0;
            end
            else
            begin
                  wit <= wit +1;
                  floor = FLOOR0;
            end
        end
        else // destination is somewhere else move up
          begin
           ////////////////////////////////////////////////////////////////////////
              if (cnt==MOVE_SPEED)
             begin
                cnt <= 0;
                floor <=FLOOR1;
                wit =0;
             end
             else begin
                cnt <= cnt +1;
                floor = FLOOR0;
             end
          end
end
```

Input Conditions:

Program starts the check input whenever they are equal to one.

```
always@( floor_4_d ==1, floor_3_d==1, floor_2_d==1, floor_1_d==1, floor_0_d==1, floor_0_p==1,
        floor_1_p==1, floor_2_p==1, floor_3_p==1, floor_4_p==1)
```

Conditions: since most of the conditions are same two different ones will be described if floor 0 button is inserted and elevator is not on the move led 0 should be lit (same for floor_0_d)

```
if (  floor_0_p == 1 && led_busy == 0)
begin
    led_outside_0 =1;
    direction = 0;
    if(led_busy == 0) begin destination = 0;led_busy = 1;end
    else led_busy = 1;
```

Destination is 0 and, on the move, condition is activated, Direction is 0.

Here the first condition is for if the elevator is currently above floor1 and destination is below floor1, second and the third one is same with floor_0_p button except for directions. Fourth condition is for if the elevator is currently below floor one and destination is above floor 1. If these conditions are satisfied led outside should turn on and destination should be updated only if there is no current destination already.

```
if ((floor_1_p ==1 && direction == direction_1 && led_busy == 1&& current_floor < 1&& destination >  1 )
    ||(floor_1_p ==1 && led_busy ==0 && current_floor > 1 && direction_1 == 0 )
    ||(floor_1_p ==1 && led_busy ==0 && current_floor < 1 && direction_1 == 1 )
    ||(floor_1_p ==1 && direction == direction_1 && led_busy == 1&& current_floor > 1&& destination <  1 ))
begin
    led_outside_1 =1;
     if(current_floor>1) direction =0;
              else begin direction =1; end
    if(led_busy ==0)
    begin destination = 1; led_busy =1;end
     else led_busy =1;

end
else begin end
```

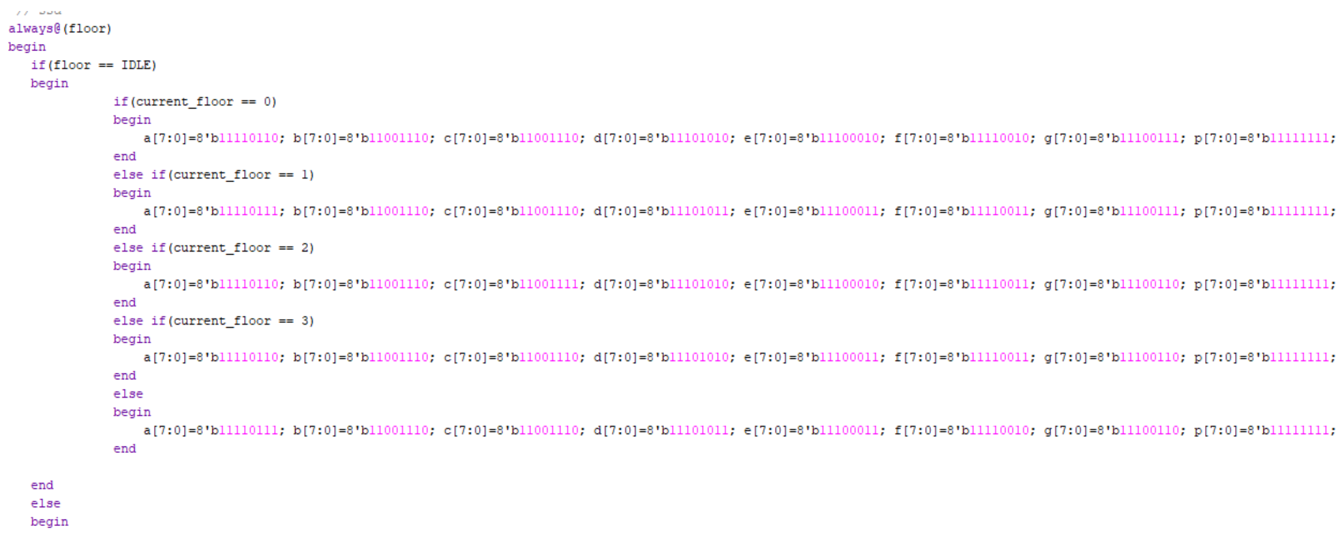
This structure is same for led inside conditions except for direction testing.

```
if(floor_1_d ==1 && led_busy ==0 || floor_1_d ==1 && led_busy == 1 && destination<1 && current_floor >1 ||
       floor_1_d ==1 && led_busy == 1 && destination > 1 && current_floor < 1 )
begin
       if(led_busy ==0) destination = 1;
       else begin end
         led_busy =1;
        led_inside_1 = 1;
if(current_floor>1) direction =0;
else begin direction =1; end
```

SSD:

There are 3 main conditions for SSD module to decide which are when elevator is moving up, moving down and it is in idle state.

All of them have the same structure so only one of them is shown below.



```
// ssd
always@(floor)
begin
   if(floor == IDLE)
   begin
             if(current_floor == 0)
             begin
                 a[7:0]=8'b11110110; b[7:0]=8'b11001110; c[7:0]=8'b11001110; d[7:0]=8'b11101010; e[7:0]=8'b11100010; f[7:0]=8'b11110010; g[7:0]=8'b11100111; p[7:0]=8'b11111111;
             end
             else if(current_floor == 1)
             begin
                 a[7:0]=8'b11110111; b[7:0]=8'b11001110; c[7:0]=8'b11001110; d[7:0]=8'b11101011; e[7:0]=8'b11100011; f[7:0]=8'b11110011; g[7:0]=8'b11100111; p[7:0]=8'b11111111;
             end
             else if(current_floor == 2)
             begin
                 a[7:0]=8'b11110110; b[7:0]=8'b11001110; c[7:0]=8'b11001111; d[7:0]=8'b11101010; e[7:0]=8'b11100010; f[7:0]=8'b11110011; g[7:0]=8'b11100110; p[7:0]=8'b11111111;
             end
             else if(current_floor == 3)
             begin
                 a[7:0]=8'b11110110; b[7:0]=8'b11001110; c[7:0]=8'b11001110; d[7:0]=8'b11101010; e[7:0]=8'b11100011; f[7:0]=8'b11110011; g[7:0]=8'b11100110; p[7:0]=8'b11111111;
             end
             else
             begin
                 a[7:0]=8'b11110111; b[7:0]=8'b11001110; c[7:0]=8'b11001110; d[7:0]=8'b11101011; e[7:0]=8'b11100011; f[7:0]=8'b11110010; g[7:0]=8'b11100110; p[7:0]=8'b11111111;
             end

   end
   else
   begin
```

SIMULATION:

Main test case file

First part of the simulation

Rst and clk is initialized.

Inputs are neutralized to 0 after each floor input.

Correct input for floor 3 outside is given.

Correct input for floor 1 is given.

Correct input for floor 2 is given.

Wrong input for floor 1 is given.

Wrong input for floor 4 is given.

So, the expected result is.

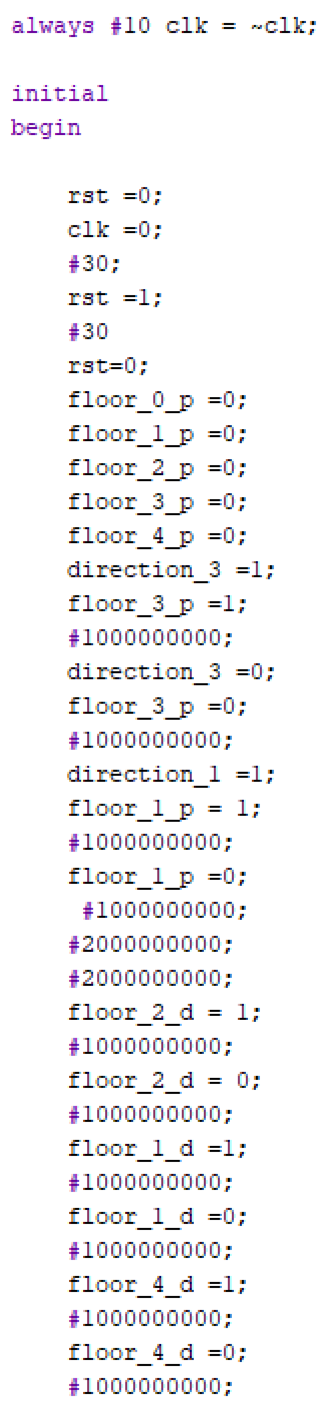
Led_outside_3 should turn on.

Then led_outside_1 should turn on.

Then led_inside_2 should turn on.

Wrong inputs should be ignored.

Elevator should stop at floors 1 2 3 (led_busy should turn on)



```
always #10 clk = ~clk;

initial
begin

    rst =0;
    clk =0;
    #30;
    rst =1;
    #30
    rst=0;
    floor_0_p =0;
    floor_1_p =0;
    floor_2_p =0;
    floor_3_p =0;
    floor_4_p =0;
    direction_3 =1;
    floor_3_p =1;
    #1000000000;
    direction_3 =0;
    floor_3_p =0;
    #1000000000;
    direction_1 =1;
    floor_1_p = 1;
    #1000000000;
    floor_1_p =0;
     #1000000000;
    #2000000000;
    #2000000000;
    floor_2_d = 1;
    #1000000000;
    floor_2_d = 0;
    #1000000000;
    floor_1_d =1;
    #1000000000;
    floor_1_d =0;
    #1000000000;
    floor_4_d =1;
    #1000000000;
    floor_4_d =0;
    #1000000000;
```

Simulation Result:

Expected results,

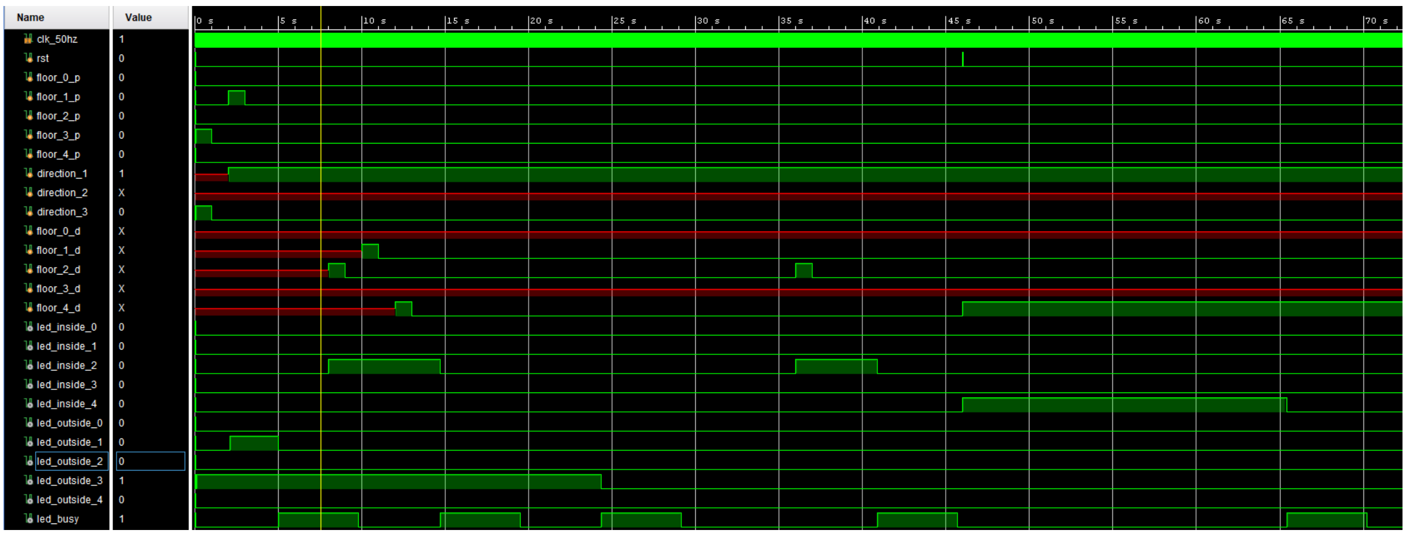
Led_outside_3 should turn on.

Then led_outside_1 should turn on.

Then led_inside_2 should turn on.

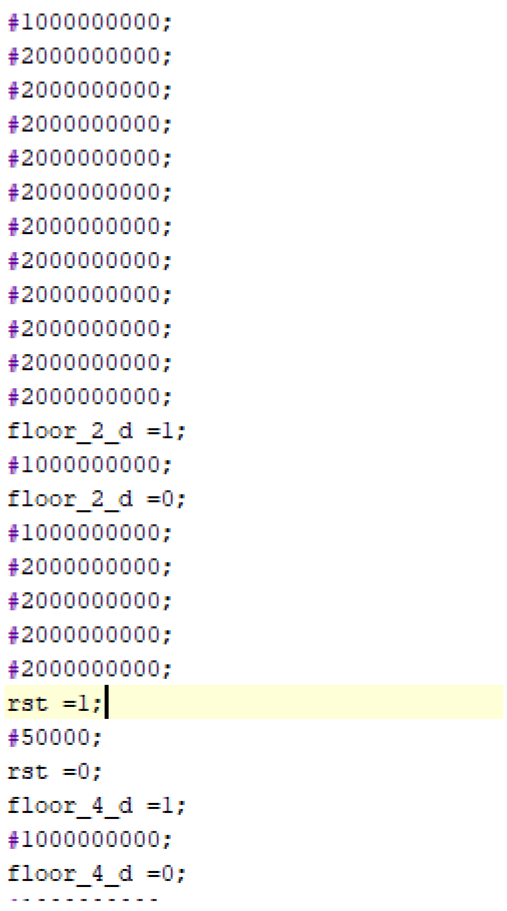
Wrong inputs should be ignored.

Elevator should stop at floors 1 2 3 (led_busy should turn on)



After 30 seconds first part of the simulation is finished and the next one starts. This part is to show elevator can move down so there are not many tests. Rst is also shown in this part.

First elevator starts moving down from floor 3 to floor 2 then rst is activated and elevator is back to the beginning and starts going to floor 4 and stops there which takes 25 seconds as it should be.



```
#1000000000;
#2000000000;
#2000000000;
#2000000000;
#2000000000;
#2000000000;
#2000000000;
#2000000000;
#2000000000;
#2000000000;
#2000000000;
#2000000000;
floor_2_d =1;
#1000000000;
floor_2_d =0;
#1000000000;
#2000000000;
#2000000000;
#2000000000;
#2000000000;
rst =1;
#50000;
rst =0;
floor_4_d =1;
#1000000000;
floor_4_d =0;
```