Project report on

ELEVATOR CONTROLLER

Submitted by:
Team Elevators

Group Members:

1. Aadarsh Chouragade BT21ECE070
2. Kushagra Rai BT21ECE068
3. Ritik Kumar BT21ECE067

A report submitted for the partial fulfilment of the requirements of the course
ECL-303 Hardware Description Languages

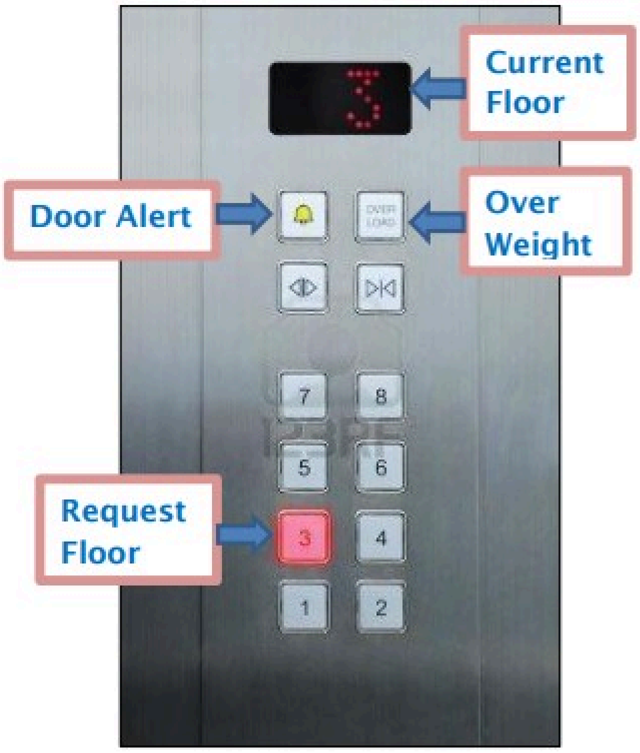
Submission Date: 20/11/2023

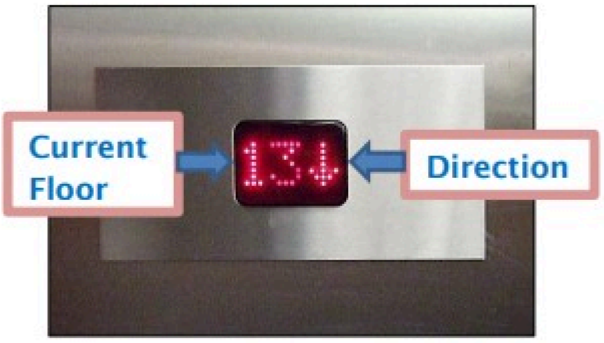
Under the guidance of:
Dr. Mayank Thacker
Department of Electronics and Communication Engineering

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर
Indian Institute of Information Technology, Nagpur

Chapter 1: Introduction

This project is designed for an nine floor elevator controller. of an integrated circuit that can be used as part of elevator controller. The elevator decides moving direction by comparing request floor with current floor. In a condition that the weight has to be less than 4500lb and door has to be closed . If the weight is larger than it, the elevator will alert automatically. There is a sensor at each floor to sense whether the elevator has passed the current floor. This sensor provides the signal that encodes the floor that has been passed. The core parts of the design are , three cases of elevator ,implemented using if -else statements when we receive requested floor.





HARDWARE MODEL

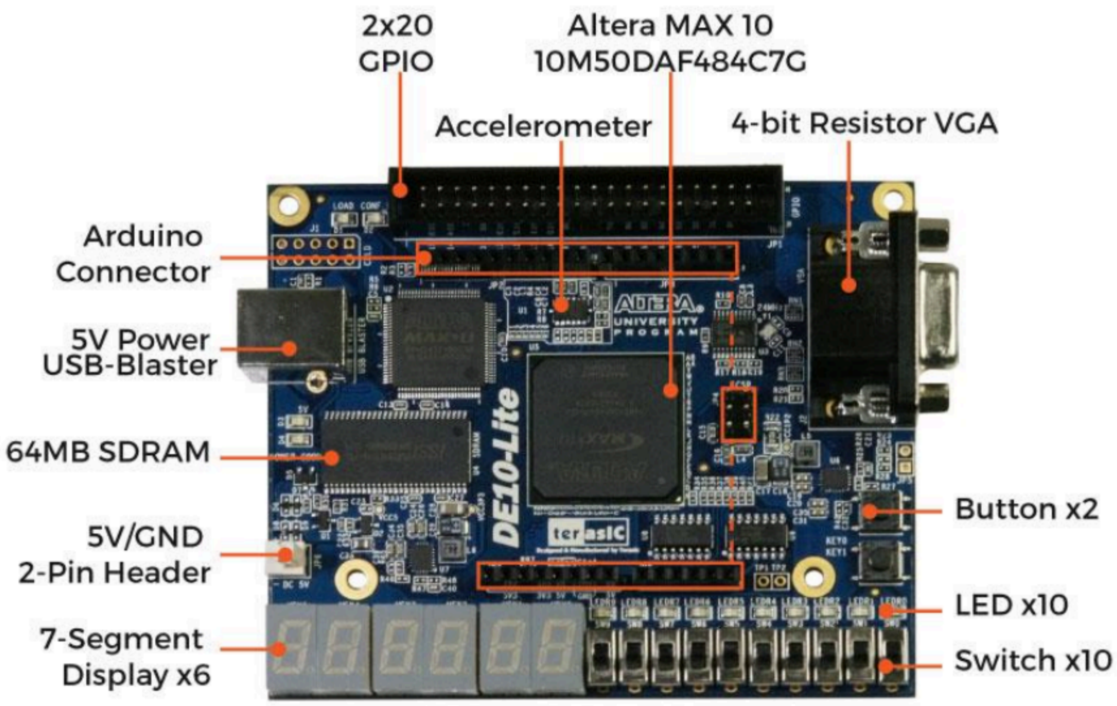


Figure 1-2 Development Board (top view)

Chapter 2: Code

```
 module elevator(seconds,clk,on,over_weight,
requested_floor,up,weight_alert,down,open_door,rescue,responce_rescue,
out1,reset,current_floor_reg);

 //input ports
  input  clk;
  input   on;
  input  rescue;
  input  over_weight;
  input  [3:0] requested_floor;
  input reset;

//output ports
  output  up;
  output  weight_alert;
  output  down;
  output  open_door;
  output  responce_rescue;
 output [3:0] req_floor;
  output seconds;
```

```
//registers
  output reg [3:0] current_floor_reg = 4'b0000;  // using
  reg up_pressed;
  reg down_pressed;
  reg door_open;
  reg weight_over;
  reg rescue_alert;

delay name(clk,seconds);
always @(posedge seconds)
begin



else if(rescue == 1’b1) begin
                    up_pressed <= 1'b0;
                   down_pressed <= 1'b0;
                    rescue_alert <= 1'b1;
                    end
else    if(over_weight == 1’b1) begin
                    up_pressed <= 1'b0;
                    down_pressed <= 1'b0;
                    weight_over <= 1'b1;
door_open <= 1'b1;
        end
else
 begin  // here starts the normal working mode of elevator -->>
if (current_floor_reg < requested_floor)
begin   //If requested floor is greater than current floor,the lift moves Up
```

```
        current_floor_reg <= current_floor_reg + 1;
        up_pressed <= 1'b1; //up_pressed is made high
        down_pressed <= 1'b0;
end
else if ( current_floor_reg > requested_floor)
begin
        current_floor_reg <= current_floor_reg- 1;
        up_pressed <= 1'b0;
        down_pressed <= 1'b1;
      end

 if (current_floor_reg == requested_floor && door_open == 1'b0) //Here
we have reached the current floor
begin
     door_open <= 1'b1;
 end
else
begin
                    door_open <= 1'b0;
      end
    end
  end

assign up = up_pressed;
  assign down = down_pressed;
  assign open_door = door_open;
  assign weight_alert = weight_over;
assign responce_rescue = rescue_alert;
assign req_floor = current_floor_reg;



  output [13:0] out1;
  wire [6:0] o1,o2;
  sev_Seg inst1(requested_floor,o1);
  sev_Seg inst2(current_floor_reg,o2);
  assign out1 = {o2,o1};

endmodule




module sev_Seg(bcd,s);
     input wire [3:0] bcd;
     output wire [6:0] s;
     reg [6:0] seg;
            assign s = seg;

    always @(bcd)
    begin
        case (bcd) //case statement
         4'b0000 : seg = 7'b1000000;
                              4'b0001 : seg = 7'b1111001;
                              4'b0010 : seg = 7'b0100100;
                              4'b0011 : seg = 7'b0110000;
                              4'b0100 : seg = 7'b0011001;
```

```
                        4'b0101 : seg = 7'b0010010;
                        4'b0110 : seg = 7'b0000010;
                        4'b0111 : seg = 7'b1111000;
                        4'b1000 : seg = 7'b0000000;
                        4'b1001 : seg = 7'b0011000;
                        default: seg = 7'b1111111;
        endcase
    end
endmodule
module delay (clk,seconds);
output reg seconds;
input clk;
reg [26:0] count;

always @(posedge clk)
begin
            if (count == 27'd25_000_000) begin
        count   = 0;
                    seconds=~seconds;
    end else begin
        count   <= count + 1'b1;
    end
end
endmodule
module sev_Seg(bcd,s);

     input wire [3:0] bcd;
    output wire [6:0] s;
```

```
     reg [6:0] seg;
            assign s = seg;
    always @(bcd)
    begin
        case (bcd) //case statement
         4'b0000 : seg = 7'b1000000;
                              4'b0001 : seg = 7'b1111001;
                              4'b0010 : seg = 7'b0100100;
                              4'b0011 : seg = 7'b0110000;
                              4'b0100 : seg = 7'b0011001;
                              4'b0101 : seg = 7'b0010010;
                              4'b0110 : seg = 7'b0000010;
                              4'b0111 : seg = 7'b1111000;
                              4'b1000 : seg = 7'b0000000;
                              4'b1001 : seg = 7'b0011000;
                              default: seg = 7'b1111111;
        endcase
    end
endmodule
module delay (clk,seconds);
output reg seconds;
input clk;
reg [26:0] count;
always @(posedge clk)
begin
          if (count == 27'd25_000_000)
       begin
        count   = 0;
```

```
      seconds=~seconds;
   end
else
begin
       count   <= count + 1'b1;
   end
end
endmodule
```

REFERENCES:

1)https://www.javatpoint.com/verilog

2)Verilog HDL: A Guide to Digital Design and Synthesis