Electrical Engineering and Computer Science Department

Digital Logic Design Laboratory (ECCE210)

Spring 2023

Project

Section B3 - Group 4

FPGA Controller Design for an Elevator

Shayma Alteneiji	100063072
Hessa Naser	100060385
Fakhera Alhassani	100060400

Table of content

Introduction 3
Design Procces 3
The design method 3
The Finite State Machine 3
The block diagram 4
FPGA Results 6
The working of the elevator 6
Overload and Fire alarm FPGA results 9
Verilog Code 10
Source Code 10
Constraints Code 16
Work Contribution 18

Introduction

An elevator is a machine that helps transport people and items between floors in a building. Historically a machine like this used to be powered by people's strength. Nowadays, we can use motors to assist us instead. The elevator that we were asked to design a controller for has four floors: ground, first, second, and third floor. The elevator either moves up, down, or stays in its position. The door to the elevator stays open when not in use. Switches inside the elevator and at each floor call it to the desired floor. A display screen shows the current position of the elevator.

The elevator has a number of sensors, including proximity and overload sensors. The proximity sensor output is one when passing through or stopping at a floor. The overload sensor detects any overload situation. A fire alarm switch is installed in the building; if the switch is on, then the elevator moves to the ground floor, and the display screen reads (FA). Once the situation is resolved, the fire alarm switch is switched off, and the display screen no longer shows the message (FA).

Design Procces

The design method

We approached the problem by listing all the design components and identifying the inputs and outputs. We then divided the problem description into separate tasks and modules. Designing the FSM by then was easy, as described in the following section. After ensuring the elevator worked as required, the overload and fire alarm features were then easily added.

The Finite State Machine

The most crucial part of the design is the Finite State Machine. The Finite state machine consists of states, transitions, and events triggering the transitions, which include the inputs and the clock. As the building has four floors, including the ground floor, the FSM has four states: Ground_F = 00, First_F = 01, Second_F = 10, and Third_F = 11. The transitions are that the elevator is either moving up (transitioning to the next state), moving down (transitioning back to the previous state), or staying in the same position (staying in the same state). The inputs triggering the transitions of state or movement of the elevator are:

1. The switches on the elevator or the switches on each floor requesting the elevator; As both sets of switches have the same functionality, they are accounted as one factor.
2. The current position or floor the elevator is at.

Instead of controlling the transitions using two inputs, we used another module, the comparator module, which gave out the signal (motor) as 00, 01, or 10 to indicate that motor is not moving, moving up, or moving down, respectively. The module compares the current position of the elevator and the requested floor, which is possible by encoding the requested floor from one hot decoding to binary numbers.

When the encoded requested floor is greater than the current position, the motor output is 01, and the motor is moving up. Else, when the encoded requested floor is less than the current position, the motor output is 10, and the motor is moving down. Else, the motor is not moving.

Finally, the proximity sensor output is the controlling signal or clock of the FSM. At every rising edge of the proximity sensor, which is when a floor is detected, a state transition occurs. The FSM of the design is shown in Figure 1.

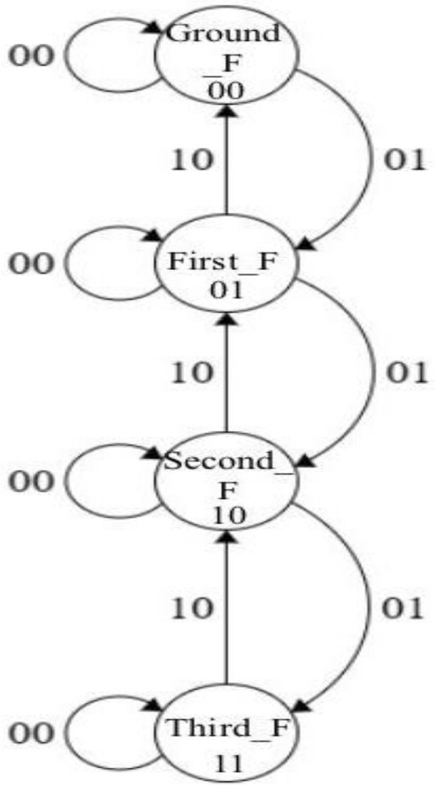

Figure 1: The Finite State Machine diagram. Each floor represents a state. At every raising edge of the proximity sensor output, the two-bit motor input triggers the transition in state.

The block diagram

The block diagram is shown in Figure 2. The figure showing the connection between the components of the elevator and the block diagram is shown in Figure 3. The block diagram consists of three modules and four (or) gates. The inputs and outputs of the design are listed as shown. The reset signal is connected to all modules. The switches on each floor are donated as s0, s1, s2, and s3, and the internal switches of the elevator are donated as b0, b1, b2, and b3. The switches corresponding to the same floor are conned to an (or) gate and then to the comparator module. The Motor signal is the output of the comparator module and is connected as an input to the disp_mux and the FSM module. The current_position wire is the output of the FSM module and is connected as an input to the comparator module and the disp_mux.

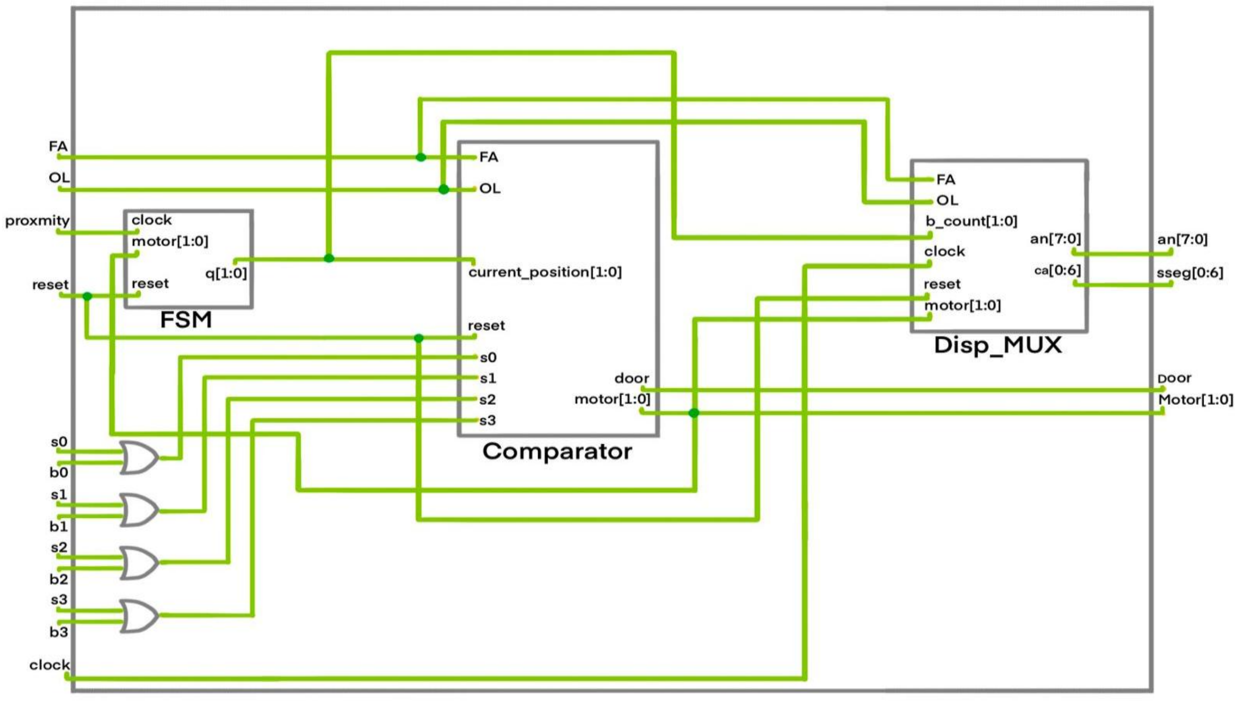

Figure 2: The design block diagram

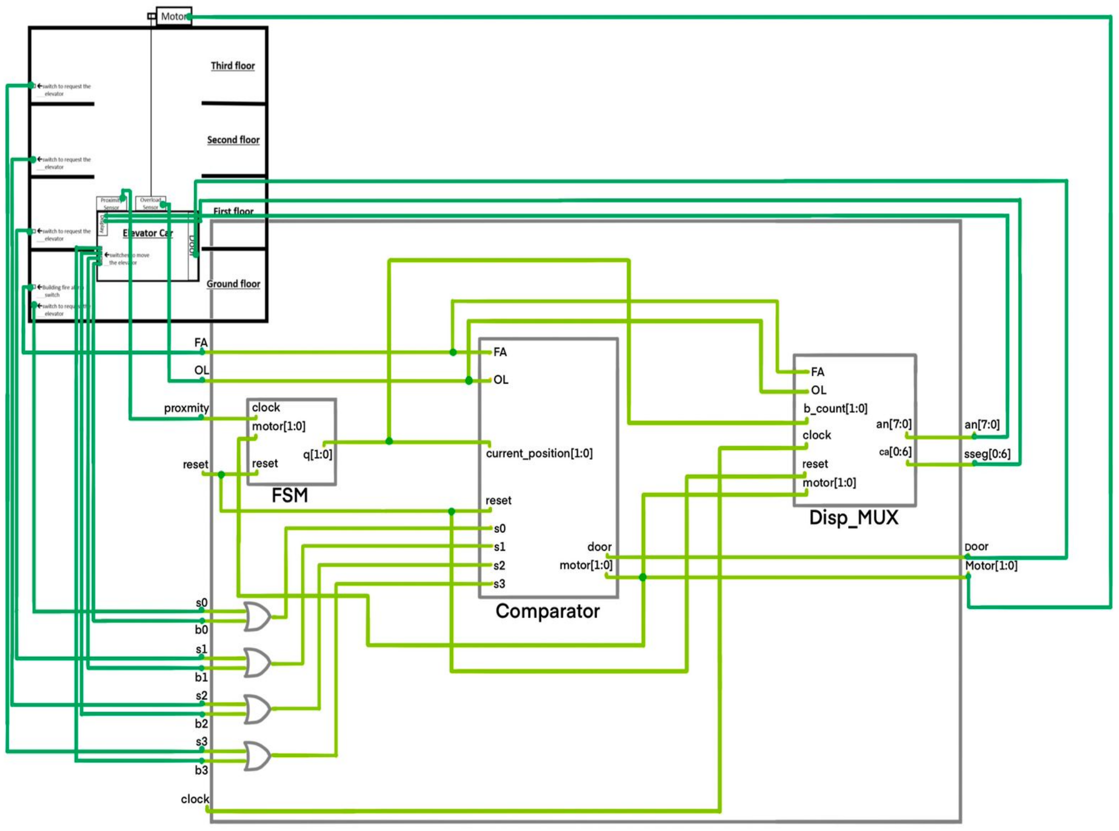

Figure 3: The connections between the components of the elevator and the block diagram

FPGA Results

The working of the elevator

The first figure, Figure 4, shows the elevator on the reset state, Ground_F. The door is open (the Door LED is on), and the motor is not moving (both Motor LEDs are turned off). In Figure 5, a switch on the third floor is pressed (s3 = 1), requesting the elevator. The door is closed (Door LED is off), and the motor is moving up (Motor [0] = 1, and Motor [1] = 0). The display shows the up arrow.

As the elevator passes through the first floor, the proximity sensor output is 1, the switch is turned on, as shown in Figure 6, and the display shows the number 1, First_F state. The proximity sensor output is zero; hence the switch is turned off until the elevator passes through the second floor, and the display shows the number 2, Second_F state, as shown in Figures 7 and 8.

In Figure 9, as the elevator stops at the third floor, the proximity sensor is turned on, Third_F state. The display shows a dash, meaning that the elevator is not moving, the door is open (Door LED is on), and both Motor LEDs are turned off.

In Figure 10, the internal switch of the elevator requesting the second floor is pressed (b2 = 1). The door is closed (Door LED is off), and the motor is moving down (Motor [0] = 0, and Motor [1] = 1). The display shows the down arrow. The proximity sensor is turned off until the elevator reaches the second floor, as shown in Figure 11. The door is then opened (Door LED is on), and the motor is not moving (both Motor LEDs are turned off), the display shows a dash.

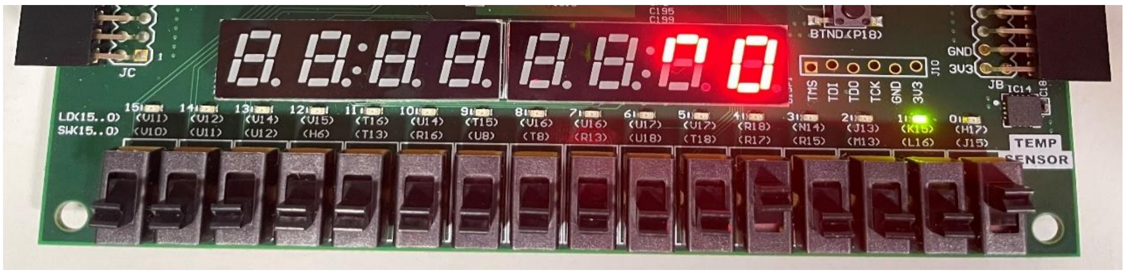

Figure 4: The elevator is on the reset, Ground_F state (00). The proximity sensor output is 1.

													Motor[1] = 0	Motor[0] = 0	Door = 1
proxmity = 1	OL = 0	FA = 0					b3 = 0	b2 = 0	b1 = 0	b0 = 0	s3 = 0	s2 = 0	s1 = 0	s0 = 0	reset = 1

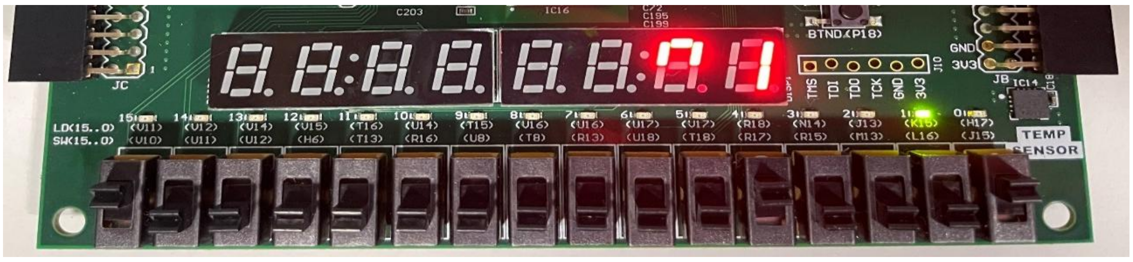

Figure 5: The third floor is requested, the elevator is moving up, door is closed. The proximity sensor output is zero.

													Motor[1] = 0	Motor[0] = 1	Door = 0
proxmity = 0	OL = 0	FA = 0					b3 = 0	b2 = 0	b1 = 0	b0 = 0	s3 = 1	s2 = 0	s1 = 0	s0 = 0	reset = 1

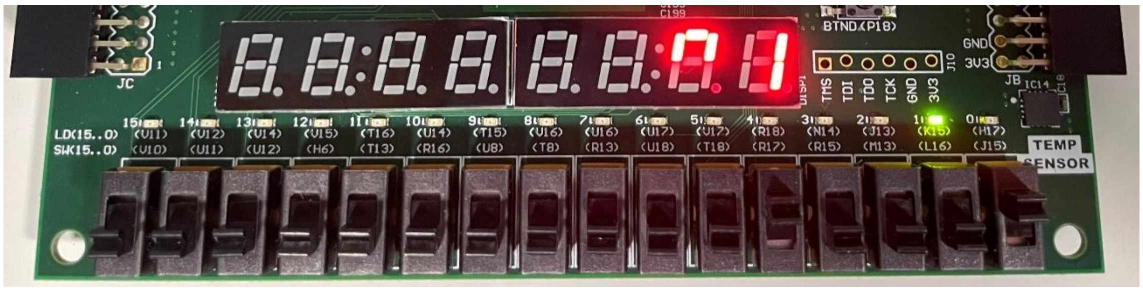
Figure 6: First_ F state. The elevator is moving up, door is closed. The proximity sensor output is one, detecting a floor.

													Motor[1] = 0	Motor[0] = 1	Door = 0
proximity = 1	OL = 0	FA = 0					b3 = 0	b2 = 0	b1 = 0	b0 = 0	s3 = 1	s2 = 0	s1 = 0	s0 = 0	reset = 1

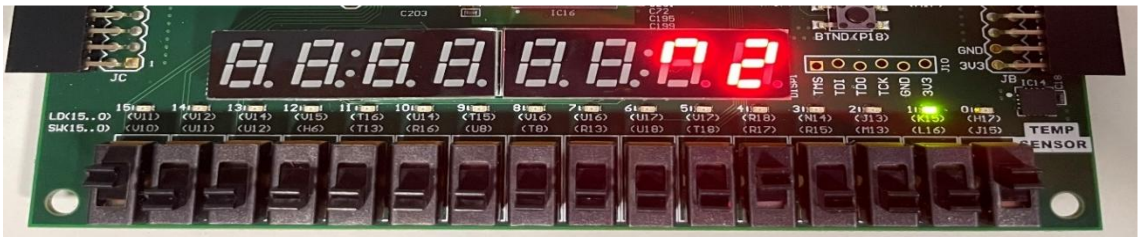
Figure 7: First_F state. The elevator is moving up, door is closed. The proximity sensor output is zero, detecting no floor.

													Motor[1] = 0	Motor[0] = 1	Door = 0
proxmity = 0	OL = 0	FA = 0					b3 = 0	b2 = 0	b1 = 0	b0 = 0	s3 = 1	s2 = 0	s1 = 0	s0 = 0	reset = 1

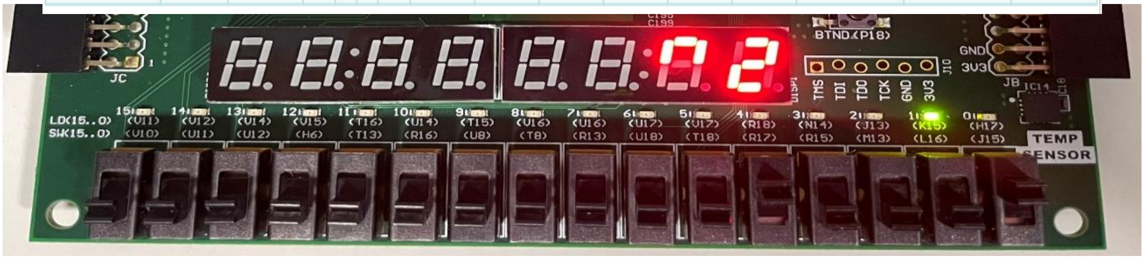
Figure 8: Second_F stat. The elevator is moving up, door is closed. The proximity sensor output is one, detecting a floor.

													Motor[1] = 0	Motor[0] = 1	Door = 0
proxmity = 1	OL = 0	FA = 0					b3 = 0	b2 = 0	b1 = 0	b0 = 0	s3 = 1	s2 = 0	s1 = 0	s0 = 0	reset = 1

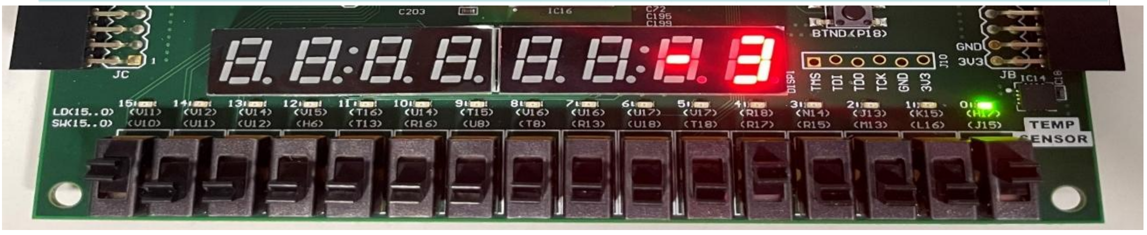
Figure 9: Second_F state. The elevator is moving up, door is closed. The proximity sensor output is zero, detecting no floor.

													Motor[1] = 0	Motor[0] = 1	Door = 0
proxmity = 0	OL = 0	FA = 0					b3 = 0	b2 = 0	b1 = 0	b0 = 0	s3 = 1	s2 = 0	s1 = 0	s0 = 0	reset = 1

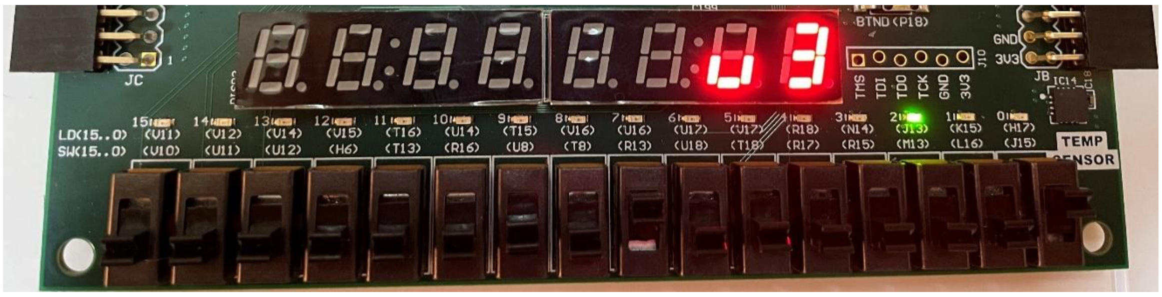

Figure 10: Third_F state. The second floor is requested, the elevator is moving down, and the door is closed. The proximity sensor output is zero, detection no floor.

												Motor[1] = 1	Motor[0] = 0	Door = 0
proxmity = 0	OL = 0	FA = 0					b3 = 0	b2 = 1	b1 = 0	b0 = 0	s3 = 0	s2 = 0	s1 = 0	s0 = 0	reset = 1

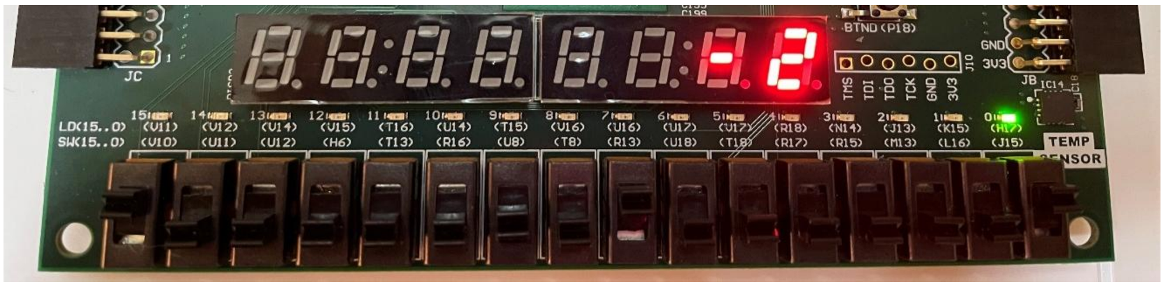

Figure 11: Second_F state. Second floor is reached, elevator is not moving, door is open. The proximity sensor output is one, detection a floor.

													Motor[1] = 0	Motor[0] = 0	Door = 1
proxmity = 1	OL = 0	FA = 0					b3 = 0	b2 = 1	b1 = 0	b0 = 0	s3 = 0	s2 = 0	s1 = 0	s0 = 0	reset = 1

Overload and Fire alarm FPGA results

Figure 12 shows the case when the fire alarm switch is turned on. The door is closed (Door LED is off), and the motor is moving down (Motor [0] = 1, and Motor [1] = 0) regardless of the requested floor. Figure 13 shows the case when the Overload sensor output is one (OL switch is turned on). The door is open (Door LED is on), and the motor is not moving (both Motor LEDs are turned off) regardless of the requested floor.

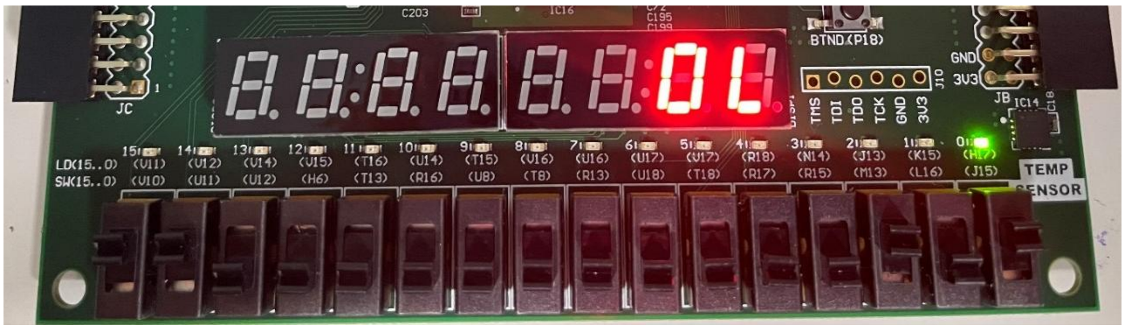

Figure 12: The overload output is one, OL switch is turned on. The elevator is not moving, and the door is open.

													Motor[1] = 0	Motor[0] = 0	Door = 1
proxmity = 1	OL = 1	FA = 0					b3 = 0	b2 = 0	b1 = 0	b0 = 0	s3 = 0	s2 = 0	s1 = 1	s0 = 0	reset = 1

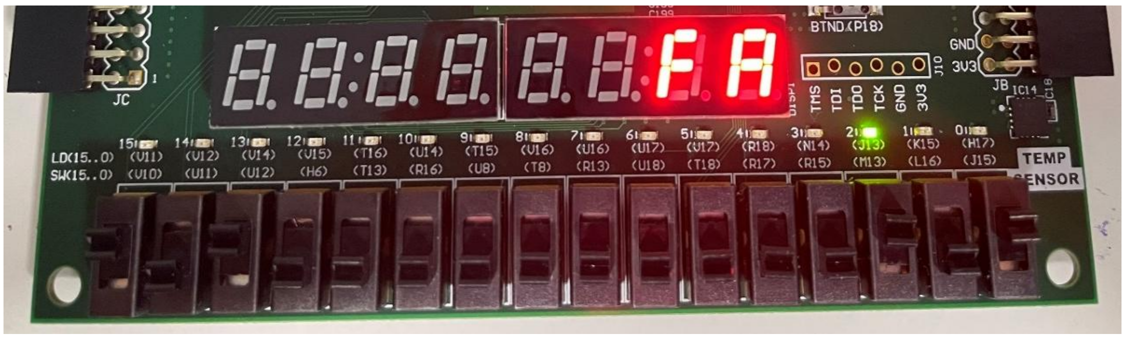

Figure 13: The fire alarm is turned on. The elevator is moving down, and the door is closed.

													Motor[1] = 1	Motor[0] = 0	Door = 0
proxmity = 1	OL = 0	FA = 1					b3 = 0	b2 = 0	b1 = 0	b0 = 0	s3 = 0	s2 = 0	s1 = 1	s0 = 0	reset = 1

Verilog Code

Source Code

```
// Shayma Alteneiji   | 100063072, Hessa Naser Al Ali | 100060385, Fakhera Alhassani | 100060400

module FSM(

input reset,

output reg [1:0] q,

input clock,

input [1:0] motor

);

reg [1:0] q_next;

always @(posedge clock or negedge reset)

if(reset == 1'b0)

q <= 1'b0;

 else

q <=q_next;

always @ (q, motor)

case(q)

2'b00: if(motor==2'b01) q_next = 2'b01;

       else q_next = q;

2'b01: if(motor==2'b01) q_next = 2'b10;

       else if(motor==2'b10) q_next = 2'b00;

       else q_next = q;

2'b10: if(motor==2'b01) q_next = 2'b11;

       else if(motor==2'b10) q_next = 2'b01;

       else q_next = q;

2'b11: if(motor==2'b10) q_next = 2'b10;

             else q_next = q;

endcase

endmodule
```

```verilog
// Shayma Alteneiji   | 100063072, Hessa Naser Al Ali | 100060385, Fakhera Alhassani | 100060400
module comparator (
input s1,
input s0,
input s2,
input s3,
input [1:0] current_position,
input reset,
output reg [1:0]  motor,
output reg Door,
input OL,
input FA
);
reg [1:0] s_decode;
always @( s3, s2, s1, s0, FA)
case ({s3, s2, s1, s0, FA})
5'b00010: s_decode = 2'b00;
5'b00000: s_decode = 2'b00;
5'b00100: s_decode = 2'b10;
5'b01000: s_decode = 2'b10;
5'b10000: s_decode = 2'b11;
default : s_decode = 2'b00;
endcase

always @ ( current_position,  s_decode, OL)
if (OL)
begin
motor = 2'b00;
Door = 1'b1;
end
```

```
else if ( s_decode ==  current_position)
begin
motor = 2'b00;
Door = 1'b1;
end
else if ( s_decode >  current_position)
begin
motor = 2'b01;
Door = 1'b0;
end
else
begin
motor = 2'b10;
Door = 1'b0;
end
endmodule
module disp_mux(
   input [1:0] b_count,
   input reset,
   input clock,
   output reg [0:6] ca,
   output reg [7:0] an,
   input [1:0] motor,
   input FA,
   input OL
   );
   localparam N = 17;
   reg [N-1:0] q;
   wire [N-1:0] q_next;
```

```
// Shayma Alteneiji   | 100063072, Hessa Naser Al Ali | 100060385, Fakhera Alhassani | 100060400
 always @(posedge clock or negedge reset)
            if(reset == 1'b0)
                q <= 1'b0;
            else
            q <=q_next;

assign q_next=q+1;

  reg [3:0] muxout;
  always @ (q, b_count, motor, OL, FA)
  case(q[N-1])
  0:
  if (OL == 1'b1)
  begin
    muxout =4'b0111 ;
    an=8'b11111110;
    end
    else if ( FA == 1'b1)
    begin
      muxout = 4'b1000;
      an=8'b11111110;
      end
     else
    begin
    muxout = b_count[1:0] ;
    an=8'b11111110;
    end
    1:
     if (OL == 1'b1)
    begin
    muxout = 4'b0000;
    an=8'b11111101;
        end
    else if ( FA == 1'b1)
        begin
        muxout = 4'b1001;
        an=8'b11111101;
        end
        else
    begin
    muxout = {0,1,motor};
    an=8'b11111101;
    end
    endcase
      always@ (muxout)
       case(muxout)
       4'b0000:ca=7'b0000001;
       4'b0001:ca=7'b1001111;
       4'b0010:ca=7'b0010010;
       4'b0011:ca=7'b0000110;
       4'b0100:ca=7'b1111110;
       4'b0101:ca=7'b0011101;
       4'b0110:ca=7'b1100011;
       4'b0111:ca=7'b1110001;
       4'b1000:ca=7'b0001000;
       4'b1001:ca=7'b0111000;
        endcase
endmodule
```

```
// Shayma Alteneiji   | 100063072, Hessa Naser Al Ali | 100060385, Fakhera Alhassani | 100060400

module par(

output [1:0] Motor,

output Door,

input proxmity,

input s1,

input s0,

input s2,

input s3,

input b1,

input b0,

input b2,

input b3,

input reset,

input clock,

output [0:6] sseg,

output [7:0] an,

input FA,

input OL);

wire [1:0] q1;

wire z0;

wire z1;

wire z2;

wire z3;

assign z0 = s0 | b0;

assign z1 = s1 | b1;

assign z2 = s2 | b2;

assign z3 = s3 | b3;

comparator comparator_unit ( .OL(OL), .FA(FA), .s1(z1), .s0(z0), .s2(z2), .s3(z3), .current_position(q1), .reset(reset),
.motor(Motor), .Door(Door));

FSM FSM_unit(.reset(reset), .q(q1), .clock(proxmity) ,.motor(Motor));

disp_mux disp_mux_unit (.b_count(q1), .reset(reset), .clock(clock),  .ca(sseg),  .an(an), .motor(Motor), .FA(FA), .OL(OL));

endmodule
```

Constraints Code

```
// Shayma Alteneiji   | 100063072, Hessa Naser Al Ali | 100060385, Fakhera Alhassani | 100060400

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets proxmity_IBUF]


set_property PACKAGE_PIN T10 [get_ports {sseg[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sseg[0]}]

set_property PACKAGE_PIN R10 [get_ports {sseg[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sseg[1]}]

set_property PACKAGE_PIN K16 [get_ports {sseg[2]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sseg[2]}]

set_property PACKAGE_PIN K13 [get_ports {sseg[3]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sseg[3]}]

set_property PACKAGE_PIN P15 [get_ports {sseg[4]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sseg[4]}]

set_property PACKAGE_PIN T11 [get_ports {sseg[5]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sseg[5]}]

set_property PACKAGE_PIN L18 [get_ports {sseg[6]}]

set_property IOSTANDARD LVCMOS33 [get_ports {sseg[6]}]

set_property PACKAGE_PIN J17 [get_ports {an[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]

set_property PACKAGE_PIN J18 [get_ports {an[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]

set_property PACKAGE_PIN T9 [get_ports {an[2]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]

set_property PACKAGE_PIN J14 [get_ports {an[3]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]

set_property PACKAGE_PIN P14 [get_ports {an[4]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[4]}]

set_property PACKAGE_PIN T14 [get_ports {an[5]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[5]}]

set_property PACKAGE_PIN K2 [get_ports {an[6]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[6]}]

set_property PACKAGE_PIN U13 [get_ports {an[7]}]

set_property IOSTANDARD LVCMOS33 [get_ports {an[7]}]

set_property PACKAGE_PIN V10 [get_ports {proxmity}]
set_property IOSTANDARD LVCMOS33 [get_ports {proxmity}]

set_property PACKAGE_PIN J15 [get_ports {reset}]

set_property IOSTANDARD LVCMOS33 [get_ports {reset}]

set_property PACKAGE_PIN E3 [get_ports {clock}]

set_property IOSTANDARD LVCMOS33 [get_ports {clock}]

set_property PACKAGE_PIN H17 [get_ports {Door}]

set_property IOSTANDARD LVCMOS33 [get_ports {Door}]

set_property PACKAGE_PIN K15 [get_ports {Motor[0]}]

set_property IOSTANDARD LVCMOS33 [get_ports {Motor[0]}]

set_property PACKAGE_PIN J13 [get_ports {Motor[1]}]

set_property IOSTANDARD LVCMOS33 [get_ports {Motor[1]}]

set_property PACKAGE_PIN L16 [get_ports {s0}]

set_property IOSTANDARD LVCMOS33 [get_ports {s0}]

set_property PACKAGE_PIN M13 [get_ports {s1}]

set_property IOSTANDARD LVCMOS33 [get_ports {s1}]

set_property PACKAGE_PIN R15 [get_ports {s2}]

set_property IOSTANDARD LVCMOS33 [get_ports {s2}]

set_property PACKAGE_PIN R17 [get_ports {s3}]

set_property IOSTANDARD LVCMOS33 [get_ports {s3}]

set_property PACKAGE_PIN U12 [get_ports {FA}]

set_property IOSTANDARD LVCMOS33 [get_ports {FA}]

set_property PACKAGE_PIN U11 [get_ports {OL}]

set_property IOSTANDARD LVCMOS33 [get_ports {OL}]

set_property PACKAGE_PIN T18 [get_ports {b0}]

set_property IOSTANDARD LVCMOS33 [get_ports {b0}]

set_property PACKAGE_PIN U18 [get_ports {b1}]

set_property IOSTANDARD LVCMOS33 [get_ports {b1}]

set_property PACKAGE_PIN R13 [get_ports {b2}]

set_property IOSTANDARD LVCMOS33 [get_ports {b2}]

set_property PACKAGE_PIN T8 [get_ports {b3}]

set_property IOSTANDARD LVCMOS33 [get_ports {b3}]
```

Work Contribution

The team demonstrates their contribution to the paper as follows: Introduction: Hessa Nasser, Fakhera Alhassani; Design Methods: Shayma Alteniji; FSM Diagram: Hessa Nasser; Block Diagram: Hessa Nasser; FPGA Results: Shayma Alteniji, Fakhera Alhassani; Verilog Code: Shayma Alteniji, Hessa Nasser, and Fakhera Alhassani; Design and Data Collection: Fakhera Alhassani. All authors reviewed the results and approved the final version of the report.