DIGITAL ASSIGNMENT

DIGITAL SYSTEM DESIGN

VERILOG CODE:

CODE:

```
module elevator_fsm (
input clk,
input reset,

// Inputs
input [11:0] floor_call_buttons,
input door_open_sensor,
input door_close_sensor,
input [3:0]elevator_position_sensor,
input passenger_weight_sensor,
input fire_alarm_sensor,
input power_outage_sensor,
input disability_sensor,

// Outputs
output elevator_motor_control,
output door_open_control,
output door_close_control,
output [11:0] floor_display,
output alarm_signal,
output emergency_mode_signal
);

reg [3:0] current_state;
reg [3:0] next_state;

// State definitions
parameter IDLE = 4'b0000;
parameter MOVING_UP = 4'b0001;
parameter MOVING_DOWN = 4'b0010;
parameter DOOR_OPENING = 4'b0011;
parameter DOOR_CLOSING = 4'b0100;
parameter PASSENGER_ENTERING = 4'b0101;
parameter PASSENGER_EXITING = 4'b0110;
parameter EMERGENCY_MODE = 4'b0111;
parameter DISABILITY_MODE=4'b1000;

// State transition logic
always @(posedge clk or posedge reset) begin
```

```
if (reset) begin
current_state <= IDLE;
end else begin
current_state <= next_state;
end
end

always @(*) begin
case (current_state)
IDLE: begin
if (floor_call_buttons != 0) begin
next_state <= MOVING_UP;
end else if (door_open_sensor) begin
next_state <= DOOR_OPENING;
end else begin
next_state <= IDLE;
end
end

MOVING_UP: begin
   if (elevator_position_sensor == 11) begin
     next_state <= DOOR_OPENING;
   end else begin
     next_state <= MOVING_UP;
   end
 end

 MOVING_DOWN: begin
   if (elevator_position_sensor == 0) begin
     next_state <= DOOR_OPENING;
   end else begin
     next_state <= MOVING_DOWN;
   end
 end

 DOOR_OPENING: begin
   if (door_open_sensor) begin
     next_state <= PASSENGER_ENTERING;
   end else begin
     next_state <= DOOR_OPENING;
   end
 end

 DOOR_CLOSING: begin
   if (door_close_sensor) begin
     next_state <= IDLE;
   end else begin
     next_state <= DOOR_CLOSING;
```

```
  end
 end

 PASSENGER_ENTERING: begin
  if (passenger_weight_sensor) begin
   next_state <= DOOR_CLOSING;
  end else begin
   next_state <= PASSENGER_ENTERING;
  end
 end

 PASSENGER_EXITING: begin
  if (passenger_weight_sensor == 0) begin
   next_state <= DOOR_CLOSING;
  end else begin
   next_state <= PASSENGER_EXITING;
  end
 end

 EMERGENCY_MODE: begin
  // TODO: Implement emergency mode logic
  next_state <= EMERGENCY_MODE;
 end
 DISABILITY_MODE: begin
   if (disability_sensor)begin
      next_state<=DISABILITY_MODE;
   end
  else if (floor_call_buttons != 0) begin
 next_state <= MOVING_UP;
end
end  default: begin
   next_state <= IDLE;
  end
endcase
end

// Output logic
assign elevator_motor_control = (current_state == MOVING_UP) ? 1'b1 : (current_state ==
MOVING_DOWN) ? 1'b0 : 1'bz;
assign door_open_control = (current_state == DOOR_OPENING) ? 1'b1 : 1'bz;
assign door_close_control = (current_state == DOOR_CLOSING) ? 1'b1 : 1'bz;
assign floor_display = elevator_position_sensor;
assign alarm_signal = (fire_alarm_sensor | power_outage_sensor);
assign emergency_mode_signal = (current_state == EMERGENCY_MODE);
endmodule
```

TESTBENCH:

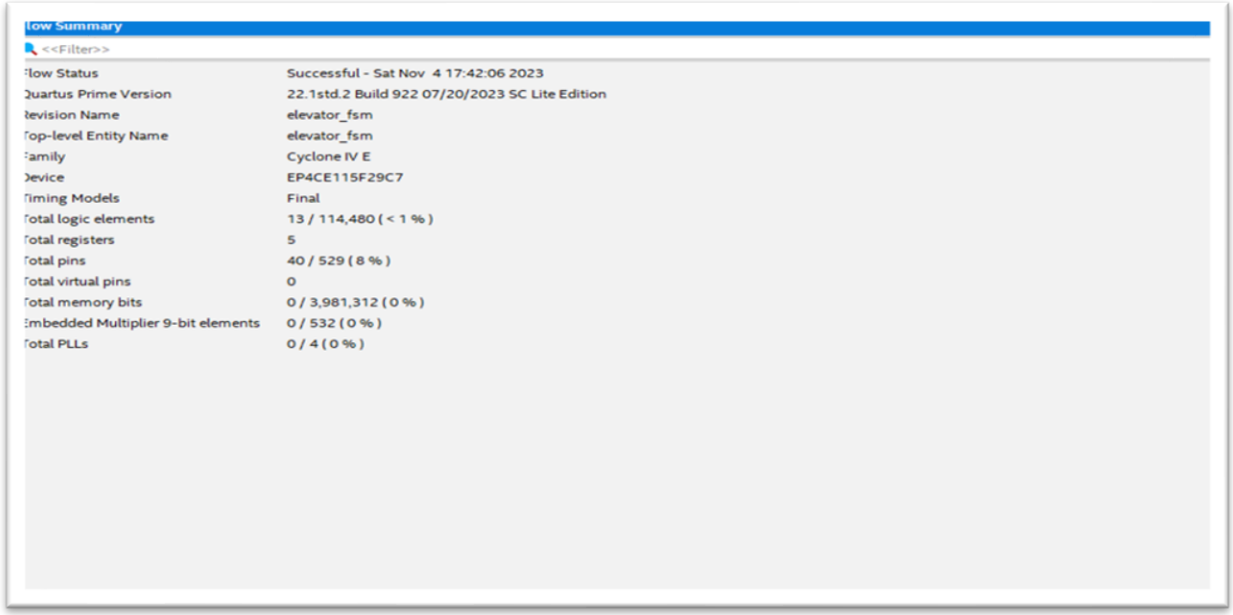
```
module test_lift_fsm;

 // Inputs and outputs of the elevator FSM
 reg clk;
 reg reset;
 reg [11:0] floor_call_buttons;
 reg door_open_sensor;
 reg door_close_sensor;
 reg [3:0]elevator_position_sensor;
 reg passenger_weight_sensor;
 reg fire_alarm_sensor;
 reg power_outage_sensor;
 reg disability_sensor;
 wire elevator_motor_control;
 wire door_open_control;
 wire door_close_control;
 wire [11:0]floor_display;
 wire alarm_signal;
 wire emergency_mode_signal;

 // Elevator FSM instance
 elevator_fsm dut (
  .clk(clk),
  .reset(reset),
  .floor_call_buttons(floor_call_buttons),
  .door_open_sensor(door_open_sensor),
  .door_close_sensor(door_close_sensor),
  .elevator_position_sensor(elevator_position_sensor),
  .passenger_weight_sensor(passenger_weight_sensor),
  .fire_alarm_sensor(fire_alarm_sensor),
  .power_outage_sensor(power_outage_sensor),
  .elevator_motor_control(elevator_motor_control),
  .door_open_control(door_open_control),
  .door_close_control(door_close_control),
  .floor_display(floor_display),
  .alarm_signal(alarm_signal),
  .emergency_mode_signal(emergency_mode_signal),
  .disability_sensor(disability_sensor)
 );

 // Test stimulus
 initial begin
  clk <= 1'b0;
  reset <= 1'b1;
  floor_call_buttons<=12'b000000000000;
```

```
#10 reset <= 1'b0;

// Test case 1: Elevator is idle and a call button is pressed
power_outage_sensor<=1'b0;
fire_alarm_sensor <= 1'b0;
floor_call_buttons[0] <= 12'b000000000000;
elevator_position_sensor<=4'b0000;
door_open_sensor=1'b0;
door_close_sensor=1'b0;
power_outage_sensor<=1'b0;
fire_alarm_sensor<=1'b0;
disability_sensor<=1'b0;
passenger_weight_sensor<=1'b0;
#10;

// Test case 2: Elevator is moving up to 5th floor and reaches the destination floor
 power_outage_sensor<=1'b0;
 fire_alarm_sensor <= 1'b0;
floor_call_buttons[5] <= 12'b000000000101;
door_open_sensor<=1'b1;
door_close_sensor<=1'b0;
elevator_position_sensor <= 4'b0101;
disability_sensor<=1'b1;
#10;

// Test case 3: Elevator is moving down and reaches the destination floor
power_outage_sensor<=1'b0;
floor_call_buttons[10] <= 12'b000000001010;
fire_alarm_sensor <= 1'b0;
elevator_position_sensor <= 4'b1010;
door_open_sensor<=1'b1;
disability_sensor<=1'b0;

#10;

// Test case 4: Elevator doors are opening and all passengers have entered
 power_outage_sensor<=1'b0;
fire_alarm_sensor <= 1'b0;
door_open_sensor <= 1'b1;
door_close_sensor<=1'b0;
passenger_weight_sensor <= 1'b1;
disability_sensor<=1'b1;
#10;

// Test case 5: Elevator doors are closing and all passengers have exited
power_outage_sensor<=1'b0;
fire_alarm_sensor <= 1'b0;
door_close_sensor <= 1'b1;
```

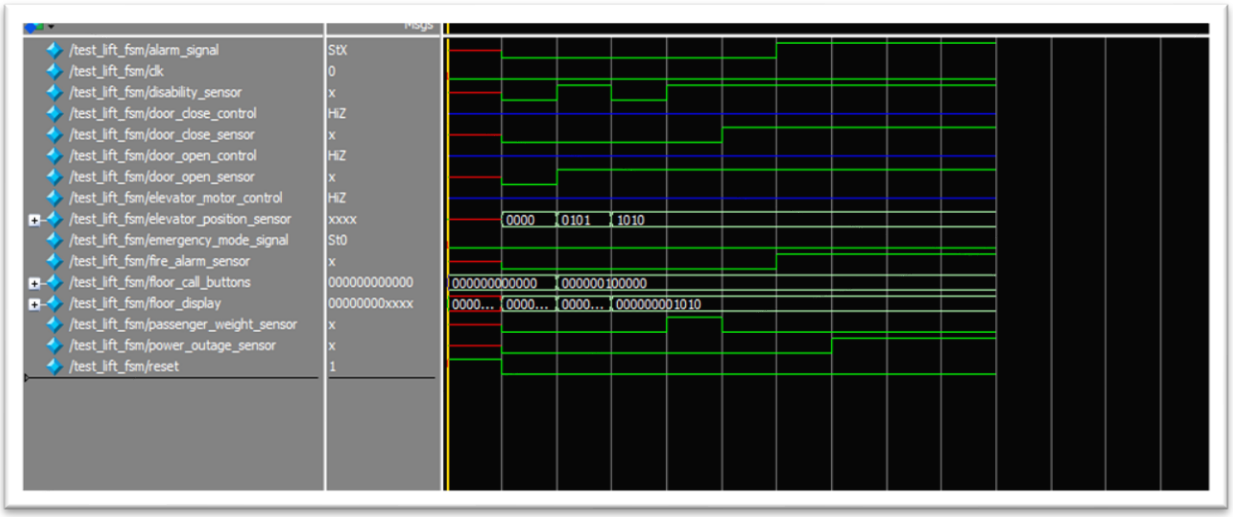
```
    passenger_weight_sensor <= 1'b0;
    disability_sensor<=1'b1;
    #10;

    // Test case 6: Fire alarm is detected
    fire_alarm_sensor <= 1'b1;
    #10;

    // Test case 7: Power outage occurs
    power_outage_sensor <= 1'b1;
    #10;

    //Test case 8: Disability sensor activates
    disability_sensor<=1'b1;
  end

  // Monitor the outputs of the elevator FSM
  always @(posedge clk) begin
    $display("Motor control signal: %b", elevator_motor_control);
    $display("Door open control signal: %b", door_open_control);
    $display("Floor display: %b", floor_display);
    $display("Alarm signal: %b", alarm_signal);
    $display("Emergency mode signal: %b", emergency_mode_signal);
end

endmodule
```

OUTPUT:

FLOW SUMMARY:

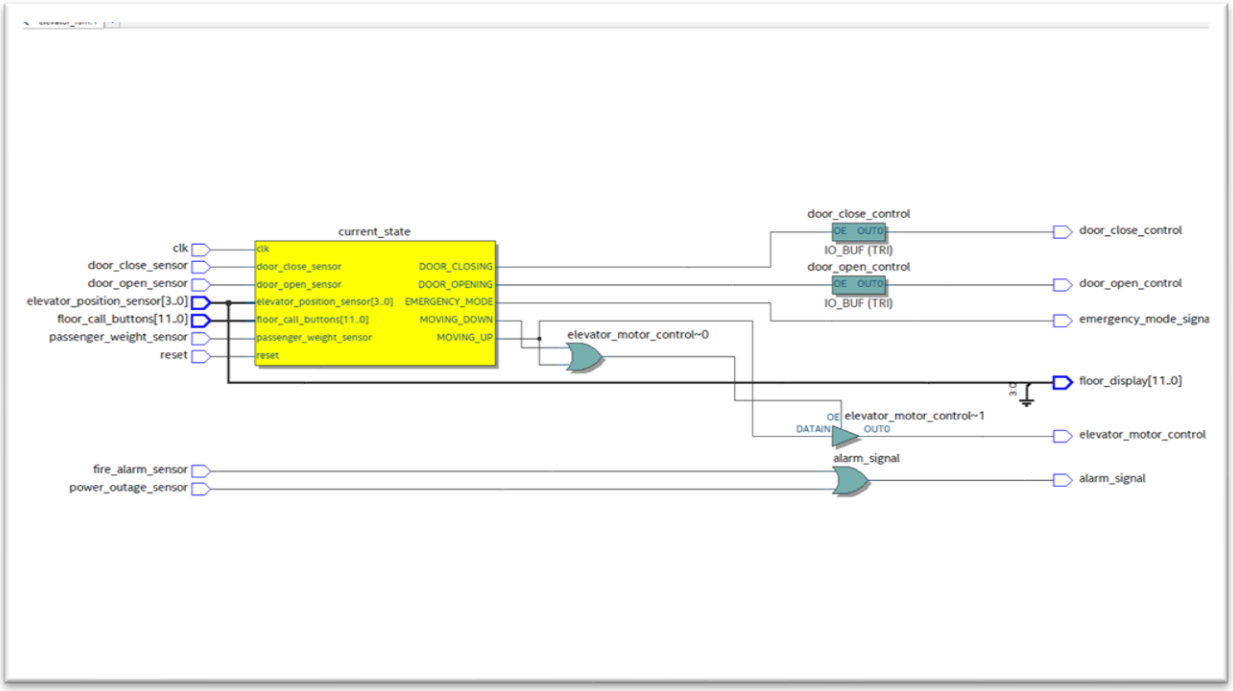


WAVEFORM:

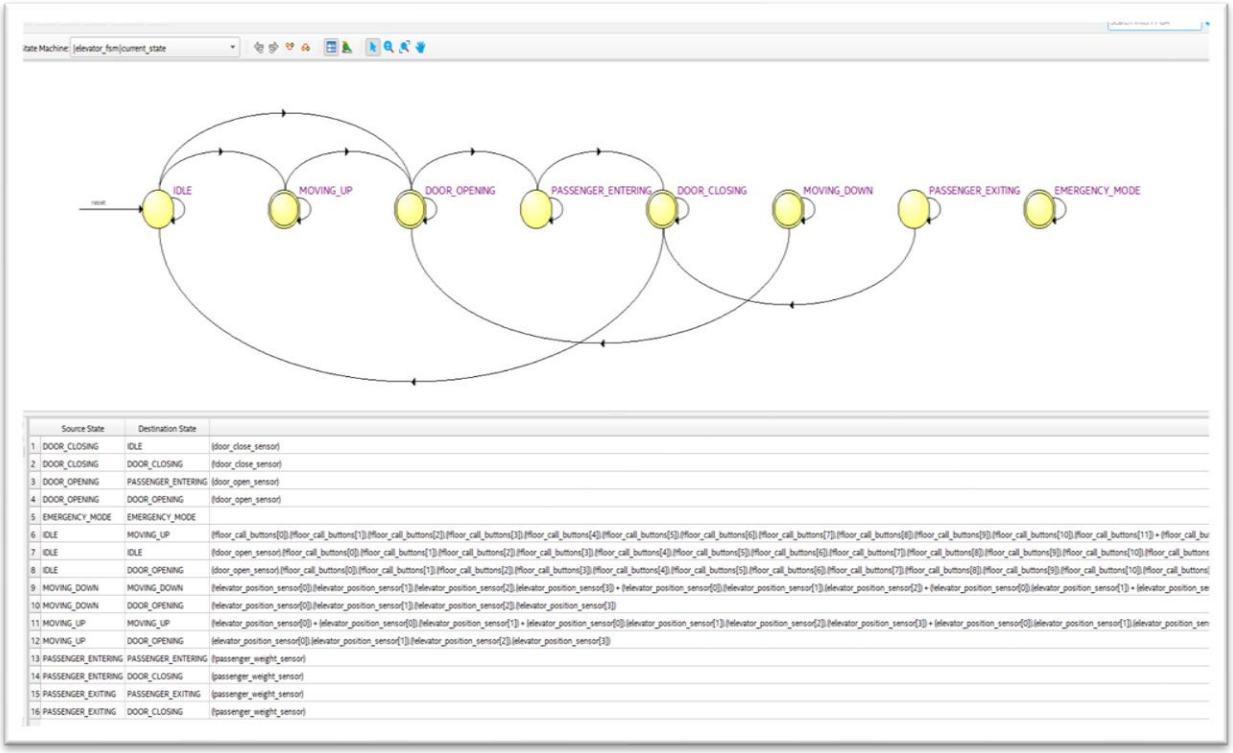


RTL VIEW:

\



STATE DIAGRAM:



	Source State	Destination State	
1	DOOR_CLOSING	IDLE	(door_close_sensor)
2	DOOR_CLOSING	DOOR_CLOSING	(!door_close_sensor)
3	DOOR_OPENING	PASSENGER_ENTERING	(door_open_sensor)
4	DOOR_OPENING	DOOR_OPENING	(!door_open_sensor)
5	EMERGENCY_MODE	EMERGENCY_MODE	
6	IDLE	MOVING_UP	(!floor_call_buttons[0].|!floor_call_buttons[1].|!floor_call_buttons[2].|!floor_call_buttons[3].|!floor_call_buttons[4].|!floor_call_buttons[5].|!floor_call_buttons[6].|!floor_call_buttons[7].|!floor_call_buttons[8].|!floor_call_buttons[9].|!floor_call_buttons[10].|!floor_call_buttons[11]) + (!floor_call_bu
7	IDLE	IDLE	(!door_open_sensor).(!floor_call_buttons[0].|!floor_call_buttons[1].|!floor_call_buttons[2].|!floor_call_buttons[3].|!floor_call_buttons[4].|!floor_call_buttons[5].|!floor_call_buttons[6].|!floor_call_buttons[7].|!floor_call_buttons[8].|!floor_call_buttons[9].|!floor_call_buttons[10].|!floor_call_buttons
8	IDLE	DOOR_OPENING	(door_open_sensor).(!floor_call_buttons[0].|!floor_call_buttons[1].|!floor_call_buttons[2].|!floor_call_buttons[3].|!floor_call_buttons[4].|!floor_call_buttons[5].|!floor_call_buttons[6].|!floor_call_buttons[7].|!floor_call_buttons[8].|!floor_call_buttons[9].|!floor_call_buttons[10].|!floor_call_buttons[
9	MOVING_DOWN	MOVING_DOWN	(elevator_position_sensor[0].|elevator_position_sensor[1].|elevator_position_sensor[2].|elevator_position_sensor[3]) + (elevator_position_sensor[0].|elevator_position_sensor[1].|elevator_position_sensor[2]) + (elevator_position_sensor[0].|elevator_position_sensor[1]) + (elevator_position_se
10	MOVING_DOWN	DOOR_OPENING	(elevator_position_sensor[0].|elevator_position_sensor[1].|elevator_position_sensor[2].|elevator_position_sensor[3])
11	MOVING_UP	MOVING_UP	(elevator_position_sensor[0]) + (elevator_position_sensor[0].|elevator_position_sensor[1]) + (elevator_position_sensor[0].|elevator_position_sensor[1].|elevator_position_sensor[2].|elevator_position_sensor[3]) + (elevator_position_sensor[0].|elevator_position_sensor[1].|elevator_position_sen
12	MOVING_UP	DOOR_OPENING	(elevator_position_sensor[0].|elevator_position_sensor[1].|elevator_position_sensor[2].|elevator_position_sensor[3])
13	PASSENGER_ENTERING	PASSENGER_ENTERING	(!passenger_weight_sensor)
14	PASSENGER_ENTERING	DOOR_CLOSING	(passenger_weight_sensor)
15	PASSENGER_EXITING	PASSENGER_EXITING	(!passenger_weight_sensor)
16	PASSENGER_EXITING	DOOR_CLOSING	(passenger_weight_sensor)