timescale 1ns / 1ps  
//////////////////////////////////////////////////////////////////////////////////  
// Company:  
// Engineer:  
//  
// Create Date:    14:22:50 04/24/2024  
// Design Name:  
// Module Name:    Lift\_controller  
// Project Name:  
// Target Devices:  
// Tool versions:  
// Description:  
//  
// Dependencies:  
//  
// Revision:  
// Revision 0.01 - File Created  
// Additional Comments:  
//  
//////////////////////////////////////////////////////////////////////////////////  
 module lift\_controller(  
 input clk,           // Clock input  
 input reset,         // Reset input  
 input call0,         // Call button for floor 0  
 input call1,         // Call button for floor 1  
 input call2,         // Call button for floor 2  
 input bottom\_sensor, // Sensor detecting bottom position  
 input middle\_minus\_sensor, // Sensor detecting middle-minus position  
 input middle\_plus\_sensor,  // Sensor detecting middle-plus position  
 input top\_sensor,    // Sensor detecting top position  
 output reg motor\_up,    // Direction control for motor  
 output reg motor\_down,       // Enable control for motor  
 output reg indicator0,   // Indicator light for floor 0  
 output reg indicator1,   // Indicator light for floor 1  
 output reg indicator2    // Indicator light for floor 2  
);  
  
// Define states  
parameter FLOOR0 = 3'b000;  
parameter FLOOR1 = 3'b001;  
parameter FLOOR2 = 3'b010;  
parameter  MOVING\_UP\_1 =3'b011;  
parameter  MOVING\_UP\_2 = 3'b100;  
parameter MOVING\_DOWN\_1 = 3'b101;  
parameter MOVING\_DOWN\_2 = 3'b110;  
  
  
// Internal signals  
reg [2:0] next\_state, current\_state;  
  
// State register  
always @(posedge clk or posedge reset) begin  
    if (reset) begin  
        current\_state <= MOVING\_DOWN\_2;  
    end else begin  
        current\_state <= next\_state;  
    end  
end  
  
// Combinational logic for next state and motor control  
always @(\*) begin  
    case (current\_state)  
        FLOOR0: begin // Floor 0  
indicator0 =1;  
indicator1 =0;  
indicator2 =0;  
            if (call0) begin // Button for floor 0 pressed  
                next\_state = FLOOR0;  
                motor\_up = 0; // Stop motor  
motor\_down = 0;  
            end else if (call1) begin // Button for floor 1 pressed  
                next\_state = MOVING\_UP\_1; // Move up to Floor 1  
            end else if (call2) begin // Button for floor 2 pressed  
                next\_state = MOVING\_UP\_2; // Move up to Floor 2  
            end else begin  
next\_state = FLOOR0;  
                motor\_up = 0; // Stop motor  
motor\_down = 0;  
  end  
        end  
        FLOOR1: begin // Floor 1  
indicator0 =0;  
indicator1 =1;  
indicator2 =0;  
            if (call1) begin // Button for floor 1 pressed  
                 next\_state = FLOOR1;  
                 motor\_up = 0; // Stop motor  
 motor\_down = 0;  
            end else if (call0) begin // Button for floor 0 pressed  
                next\_state = MOVING\_DOWN\_2; // Move down to Floor 0  
            end else if (call2) begin // Button for floor 2 pressed  
                next\_state = MOVING\_UP\_2; // Move up to Floor 2  
            end else begin  
next\_state = FLOOR1;  
                motor\_up = 0; // Stop motor  
motor\_down = 0;  
end  
        end  
        FLOOR2: begin // Floor 2  
indicator0 =0;  
indicator1 =0;  
indicator2 =1;  
            if (call2) begin // Button for floor 2 pressed  
                 next\_state = FLOOR2;  
 motor\_up = 0; // Stop motor  
 motor\_down = 0;  
            end else if (call0) begin // Button for floor 0 pressed  
                next\_state = MOVING\_DOWN\_2; // Move down to Floor 0  
            end else if (call1) begin // Button for floor 1 pressed  
                next\_state = MOVING\_DOWN\_1; // Move down to Floor 1  
            end else begin  
next\_state = FLOOR2;  
motor\_up = 0; // Stop motor  
motor\_down = 0;  
end  
        end  
        MOVING\_UP\_1: begin // Moving Up 1  
motor\_up = 1;  
motor\_down = 0;  
indicator0=0;  
indicator1=0;  
indicator2=0;  
            if (middle\_minus\_sensor && middle\_plus\_sensor) begin  
                next\_state = FLOOR1; // Reached Floor 1  
                motor\_up = 0; // Stop motor  
motor\_down = 0;  
indicator1=1;  
            end else begin  
                next\_state = MOVING\_UP\_1;  
            end  
        end  
        MOVING\_UP\_2: begin // Moving Up 2  
motor\_up = 1;  
motor\_down = 0;  
indicator0=0;  
indicator1=0;  
indicator2=0;  
            if (top\_sensor) begin  
                next\_state = FLOOR2; // Reached Floor 2  
                motor\_up = 0; // Stop motor  
motor\_down = 0;  
indicator2 =1;  
            end else begin  
                next\_state = MOVING\_UP\_2;  
            end  
        end  
        MOVING\_DOWN\_1: begin // Moving Down 1  
motor\_up = 0;  
motor\_down = 1;  
indicator0=0;  
indicator1=0;  
indicator2=0;  
            if (middle\_minus\_sensor && middle\_plus\_sensor) begin  
                next\_state = FLOOR1; // Reached Floor 1  
motor\_up = 0; // Stop motor  
motor\_down = 0;  
indicator1 =1;  
            end else begin  
                next\_state = MOVING\_DOWN\_1;  
            end  
        end  
        MOVING\_DOWN\_2: begin // Moving Down 2  
motor\_up = 0;  
motor\_down = 1;  
indicator0=0;  
indicator1=0;  
indicator2=0;  
            if (bottom\_sensor) begin  
                next\_state = FLOOR0; // Reached Floor 0  
                motor\_up = 0; // Stop motor  
motor\_down = 0;  
indicator0 =1;  
            end else begin  
                next\_state = MOVING\_DOWN\_2;  
            end  
        end  
        default: begin // Default state  
             next\_state = current\_state;  
motor\_up = 0;  
motor\_down = 0;  
        end  
    endcase  
end  
  
endmodule

NET "clk" LOC ="E3" | IOSTANDARD = "LVCMOS33";  
NET "call0" LOC ="P18" | IOSTANDARD = "LVCMOS33";  
NET "call1" LOC ="P17" | IOSTANDARD = "LVCMOS33";  
NET "call2" LOC ="M18" | IOSTANDARD = "LVCMOS33";  
NET "top\_sensor" LOC ="C17" | IOSTANDARD = "LVCMOS33";  
NET "middle\_plus\_sensor" LOC ="D18" | IOSTANDARD = "LVCMOS33";  
NET "middle\_minus\_sensor" LOC ="D14" | IOSTANDARD = "LVCMOS33";  
NET "bottom\_sensor" LOC ="F16" | IOSTANDARD = "LVCMOS33";  
//NET "stop" LOC ="J15" | IOSTANDARD = "LVCMOS33";  
NET "reset" LOC ="M17" | IOSTANDARD = "LVCMOS33";  
//NET "emergency" LOC ="N17" | IOSTANDARD = "LVCMOS33";  
  
  
//NET "out\_ground" LOC ="K1" | IOSTANDARD = "LVCMOS33";  
//NET "out\_first" LOC ="H1" | IOSTANDARD = "LVCMOS33";  
//NET "out\_second" LOC ="H4" | IOSTANDARD = "LVCMOS33";  
NET "indicator0" LOC ="H17" | IOSTANDARD = "LVCMOS33";  
NET "indicator1" LOC ="K15" | IOSTANDARD = "LVCMOS33";  
NET "indicator2" LOC ="J13" | IOSTANDARD = "LVCMOS33";  
  
//NET "curr\_ground" LOC ="G2" | IOSTANDARD = "LVCMOS33";  
//NET "curr\_first" LOC ="G4" | IOSTANDARD = "LVCMOS33";  
//NET "curr\_second" LOC ="H2" | IOSTANDARD = "LVCMOS33";  
  
NET "motor\_up" LOC ="F6" | IOSTANDARD = "LVCMOS33";  
NET "motor\_down" LOC ="J2" | IOSTANDARD = "LVCMOS33";  
//NET "buzz" LOC ="E6" | IOSTANDARD = "LVCMOS33";

module lift\_controller(  
 input clk,           // Clock input  
 input reset,         // Reset input  
 input call0,         // Call button for floor 0  
 input call1,         // Call button for floor 1  
 input call2,         // Call button for floor 2  
 input bottom\_sensor, // Sensor detecting bottom position  
 input middle\_minus\_sensor, // Sensor detecting middle-minus position  
 input middle\_plus\_sensor,  // Sensor detecting middle-plus position  
 input top\_sensor,    // Sensor detecting top position  
 output reg motor\_up,    // Direction control for motor  
 output reg motor\_down,       // Enable control for motor  
 output reg indicator0,   // Indicator light for floor 0  
 output reg indicator1,   // Indicator light for floor 1  
 output reg indicator2    // Indicator light for floor 2  
);  
  
    // Define states  
    parameter FLOOR0 = 2'b00;  
    parameter FLOOR1 = 2'b01;  
    parameter FLOOR2 = 2'b10;  
  
  
    // Define state register and next state logic  
    reg [1:0] state, next\_state;  
reg clk\_20KHz;  
integer counter = 1;  
always @(posedge clk) begin  
    if(counter == 5000) begin  
          counter = 1;  
          clk\_20KHz = ~clk\_20KHz;  
      end else begin  
          counter = counter + 1;  
      end  
end  
  
    always @(posedge clk\_20KHz or posedge reset) begin  
        if (reset) begin  
            state <= FLOOR0;  
            while(~(bottom\_sensor)) begin  
                motor\_up <=0;  
  motor\_down <=1;  
            end  
            motor\_up <=0;  
 motor\_down <=0;  
            indicator0 <= 1;  
        end else begin  
            state <= next\_state;  
        end  
    end  
  
    // State transition and control logic  
    always @(\*) begin  
        case(state)  
            FLOOR0: begin  
                if (call1) begin  
                    indicator0 = 0;  
                    while(~(middle\_minus\_sensor) && ~(middle\_plus\_sensor)) begin  
                        motor\_up = 1;  
              motor\_down = 0;  
                    end  
                    motor\_up = 0;  
         motor\_down = 0;  
next\_state = FLOOR1;  
                    indicator1 = 1;  
                end else if (call2) begin  
                    indicator0 = 0;  
                    while(~(top\_sensor)) begin  
                        motor\_up = 1;  
              motor\_down = 0;  
                    end  
                   motor\_up = 0;  
         motor\_down = 0;  
next\_state = FLOOR2;  
                    indicator2 = 1;  
                end else begin  
                    next\_state = FLOOR0;  
                end  
            end  
            FLOOR1: begin  
                if (call0) begin  
                    indicator1 = 0;  
                    while(~(bottom\_sensor)) begin  
                         motor\_up = 0;  
              motor\_down = 1;  
                    end  
                   motor\_up = 0;  
         motor\_down = 0;  
                    next\_state = FLOOR0;  
                    indicator0 = 1;  
                end else if (call2) begin  
                    indicator1 = 0;  
                    direction = UP;  
                    while(~(top\_sensor)) begin  
                        motor\_up = 1;  
              motor\_down = 0;  
                    end  
                   motor\_up = 0;  
         motor\_down = 0;  
                    next\_state = FLOOR2;  
                    indicator2 = 1;  
                end else begin  
                    next\_state = FLOOR1;  
                end  
            end  
            FLOOR2: begin  
                if (call0) begin  
                    indicator2 = 0;  
                    while(~(bottom\_sensor)) begin  
                         motor\_up = 0;  
              motor\_down = 1;  
                    end  
                    motor\_up = 0;  
         motor\_down = 0;  
                    next\_state = FLOOR0;  
                    indicator0 = 1;  
                end else if (call1) begin  
                    indicator2 = 0;  
                    while(~(middle\_minus\_sensor && middle\_plus\_sensor)) begin  
                             motor\_up = 0;  
                  motor\_down = 1;  
                    end  
                    motor\_up = 0;  
         motor\_down = 0;  
                    next\_state = FLOOR1;  
                    indicator1 = 1;  
                end else begin  
                    next\_state = FLOOR2;  
                end  
            end  
            default: next\_state = state;  
                   motor\_up = 0;  
         motor\_down = 0;  
        endcase  
    end  
  
endmodule