**//DESIGN CODE**

`timescale 1ns / 1ps

module elevator\_controller (

input clk,

input rst,

input [3:0] req, // Requests for floors 0 to 3

input overweight, // 1 = too heavy, do not close doors

output reg [1:0] current\_floor,

output reg door\_open,

output reg moving

);

reg [1:0] state;

reg [1:0] next\_floor;

// State encoding

parameter IDLE = 2'd0;

parameter CLOSE\_DOOR = 2'd1;

parameter MOVE = 2'd2;

// Priority encoder to get requested floor

function [1:0] get\_requested\_floor;

input [3:0] req;

begin

if (req[3]) get\_requested\_floor = 2'd3;

else if (req[2]) get\_requested\_floor = 2'd2;

else if (req[1]) get\_requested\_floor = 2'd1;

else get\_requested\_floor = 2'd0;

end

endfunction

always @(posedge clk or posedge rst) begin

if (rst) begin

current\_floor <= 2'd0;

door\_open <= 1;

moving <= 0;

state <= IDLE;

next\_floor <= 2'd0;

end else begin

case (state)

IDLE: begin

moving <= 0;

door\_open <= 1;

if (!overweight && req != 0) begin

next\_floor <= get\_requested\_floor(req);

if (next\_floor != current\_floor) begin

door\_open <= 0;

state <= CLOSE\_DOOR;

end

end

end

CLOSE\_DOOR: begin

door\_open <= 0;

moving <= 1;

state <= MOVE;

end

MOVE: begin

if (current\_floor == next\_floor) begin

moving <= 0;

door\_open <= 1;

state <= IDLE;

end else begin

if (current\_floor < next\_floor)

current\_floor <= current\_floor + 1;

else if (current\_floor > next\_floor)

current\_floor <= current\_floor - 1;

end

end

endcase

end

end

endmodule

**//TEST BENCH**

`timescale 1ns / 1ps

module tb\_elevator\_controller;

reg clk;

reg rst;

reg [3:0] req;

reg overweight;

wire [1:0] current\_floor;

wire door\_open;

wire moving;

elevator\_controller uut (

.clk(clk),

.rst(rst),

.req(req),

.overweight(overweight),

.current\_floor(current\_floor),

.door\_open(door\_open),

.moving(moving)

);

// Clock generator: 10ns period (5ns high, 5ns low)

always #5 clk = ~clk;

initial begin

$dumpfile("elevator.vcd");

$dumpvars(0, tb\_elevator\_controller);

clk = 0;

rst = 1;

req = 4'b0000;

overweight = 1; // Start in overweight state

#10 rst = 0;

// First person at ground requests 3rd floor (overweight is high)

#10 req = 4'b1000; // At 20ns

// At 40ns, overweight becomes 0

#20 overweight = 0; // At 40ns

// Keep request valid long enough for FSM to detect it

#20 req = 4'b0000; // Clear request at 60ns

// Wait for elevator to finish trip to 3rd floor

#100;

// Second person at 3rd floor requests to go to 1st floor

req = 4'b0010; // At 160ns

#20 req = 4'b0000; // Clear at 180ns

// Let elevator process the second trip

#100;

$finish;

end

initial begin

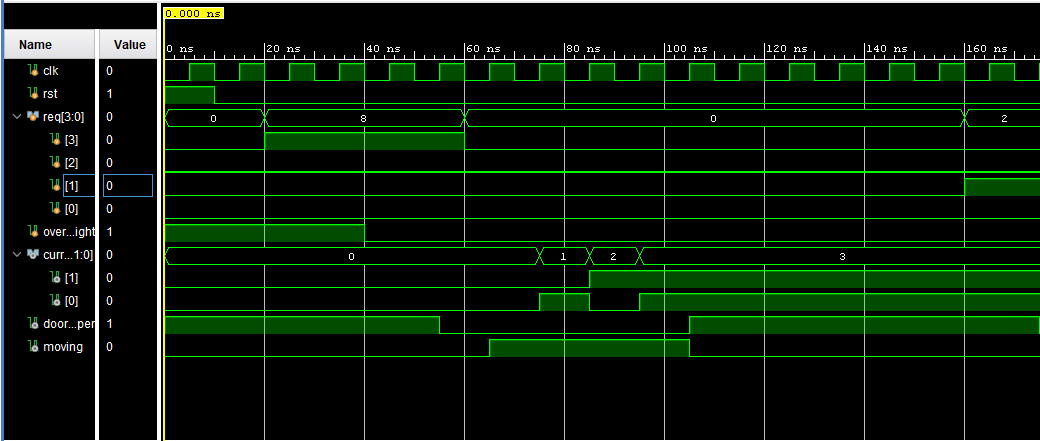
$monitor("Time = %0t | Floor = %d | Door = %b | Moving = %b | Overweight = %b | Req = %b",

$time, current\_floor, door\_open, moving, overweight, req);

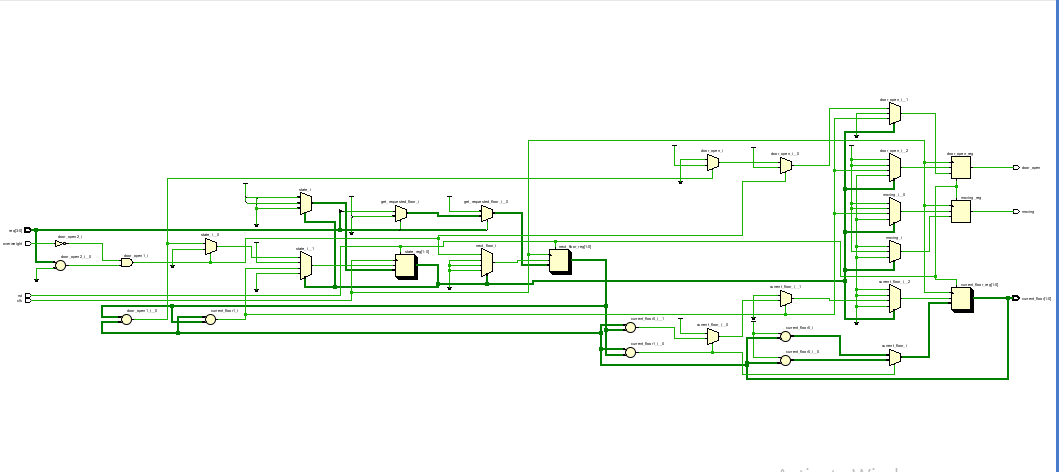
end

endmodule

**//WAVEFORM**

****

**//SCHEMATIC**

****