Dept. Of Electronics and Communication Engineering,

VIT UNIVERSITY,

CHENNAI CAMPUS

"Four Elevator Controller"

A Report submitted for PBL in VLSI System Design (ECE301)

by

1. Akshay Thakur (11BEC1057)
2. Bhagwat Singh (11BEC1070)
3. Lovish Jindal (11BEC1129)

Under the guidance and supervision of

Dr. Vigneswaran T

Dept. of SENSE

VIT UNIVERSITY,

CHENNAI CAMPUS

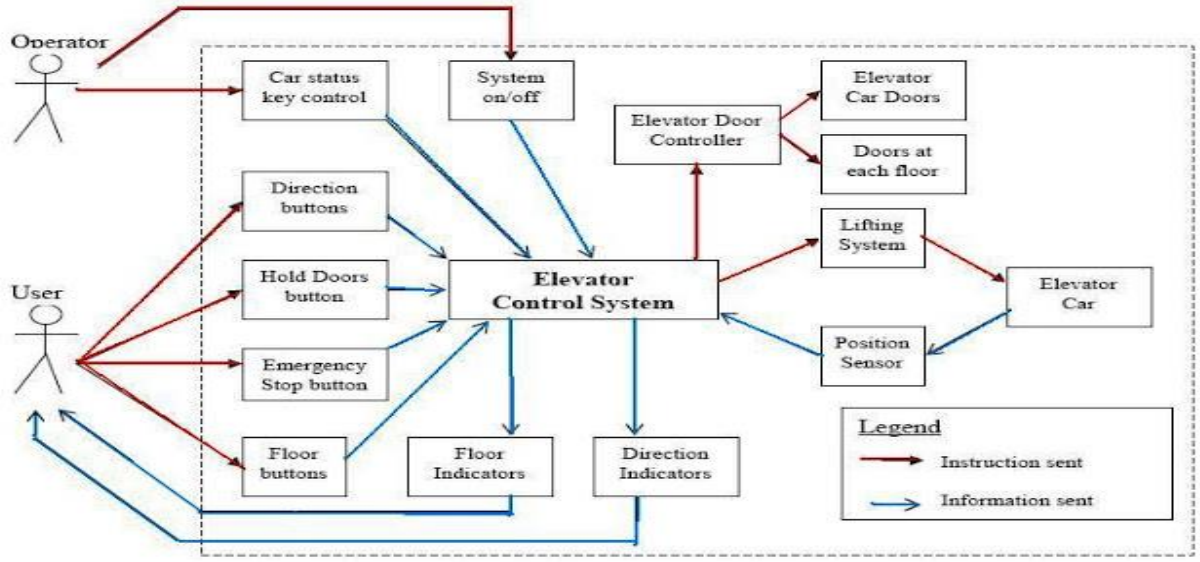
Aim: To design and implementation of embedded based elevator control system on FPGA board using Xilinx tool.

Abstract: The elevator control system is one of the important aspects in electronics control module in automotive application. In this investigation elevator control system is designed with different control strategies. First the elevator control system is implemented for multi-storage building. This implementation is based on FPGA based Fuzzy logic controller for intelligent control of elevator group system. This proposed approach is based on algorithm which is developed to reduce the amount of computation required by focusing only on relevant rules and ignoring those which are irrelevant to the condition for better performance of the group of elevator system. Here we have designed controller for four elevators group.

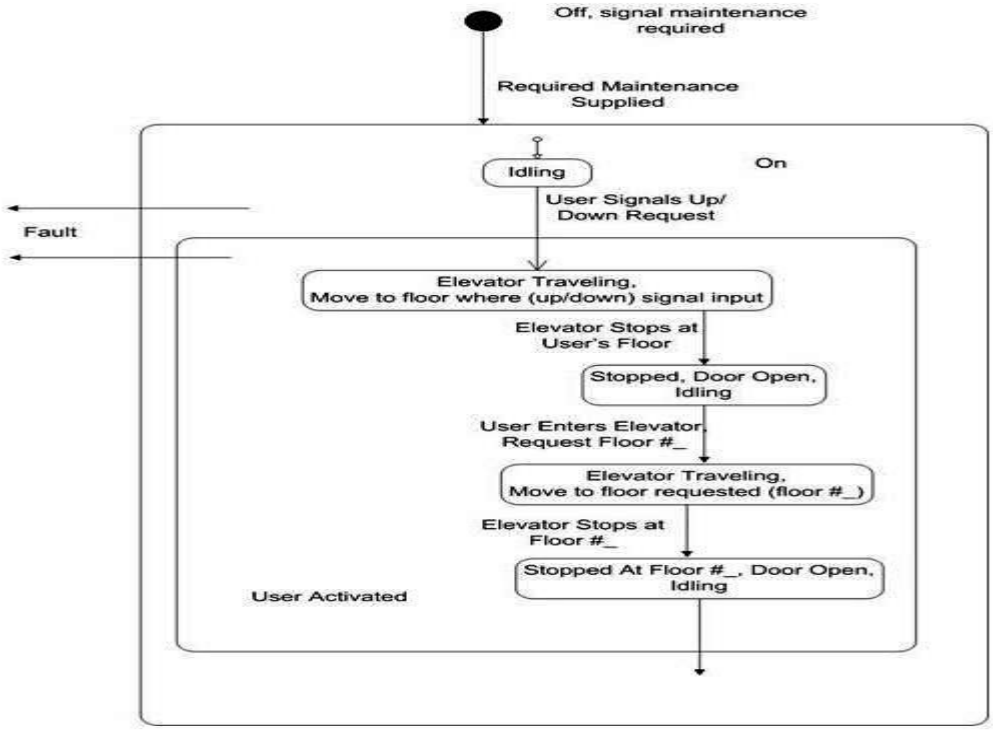
Algorithm: Elevator controller controls the entire operation of the Dual elevator system. The proximity sensors located to sense the positions of the elevators, provide the current state storing it in register. The obstruction sensors provide the status of obstruction. The elevator controller also reads the requests through the flip-flops. If the door of any elevator is open, the timer signals from the elevator keep the controller informed of being busy. The control state machine receives all these signals. It is programmed to the algorithm by which it should control the system. The CSM, then generates control signals for the next position and movement of the elevators. Elevators on receiving the signal address to the request, as been asked by the controller.

Block diagrams:

Simple elevator control system inputs and outputs:

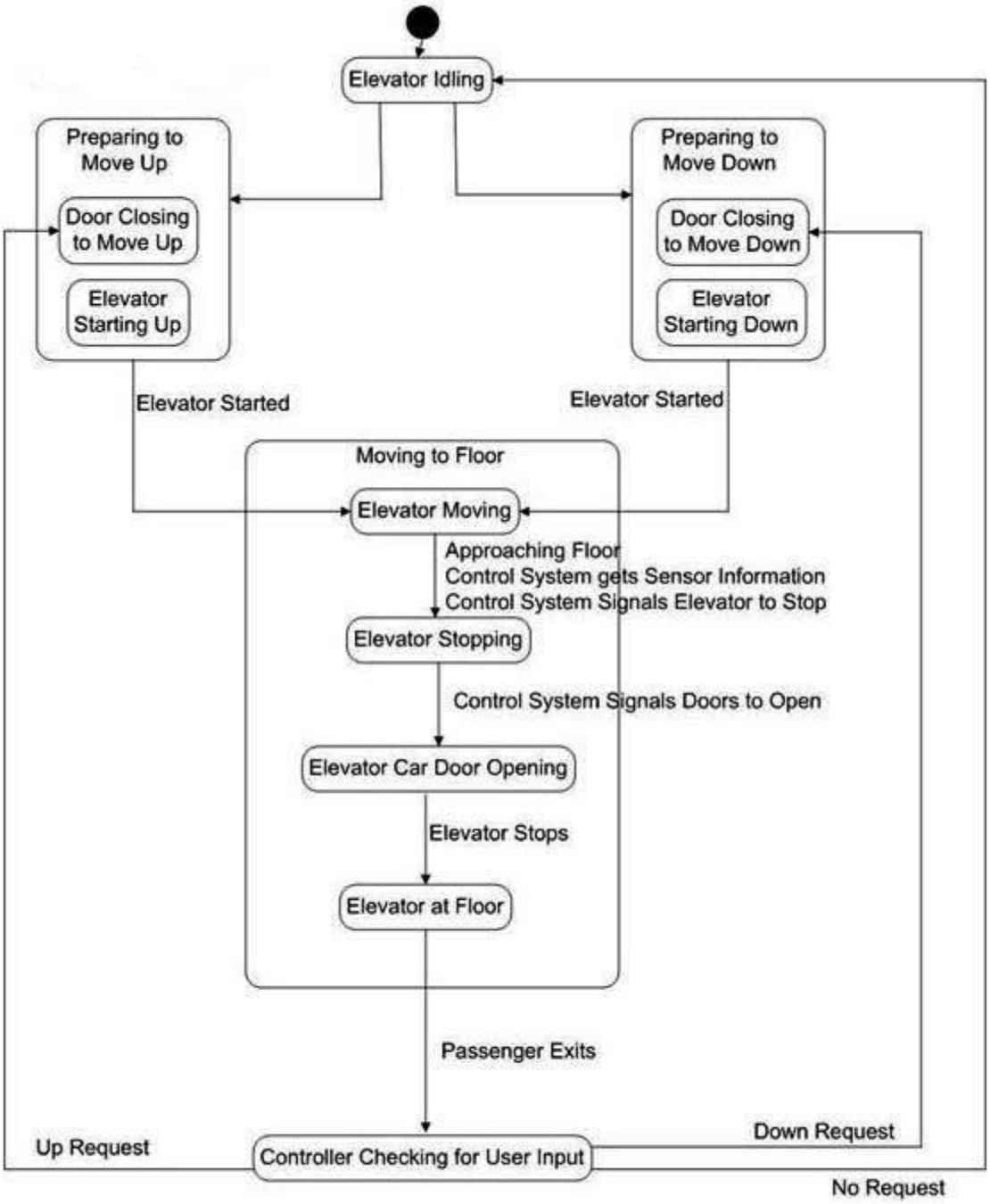


From the point of view of an individual user:

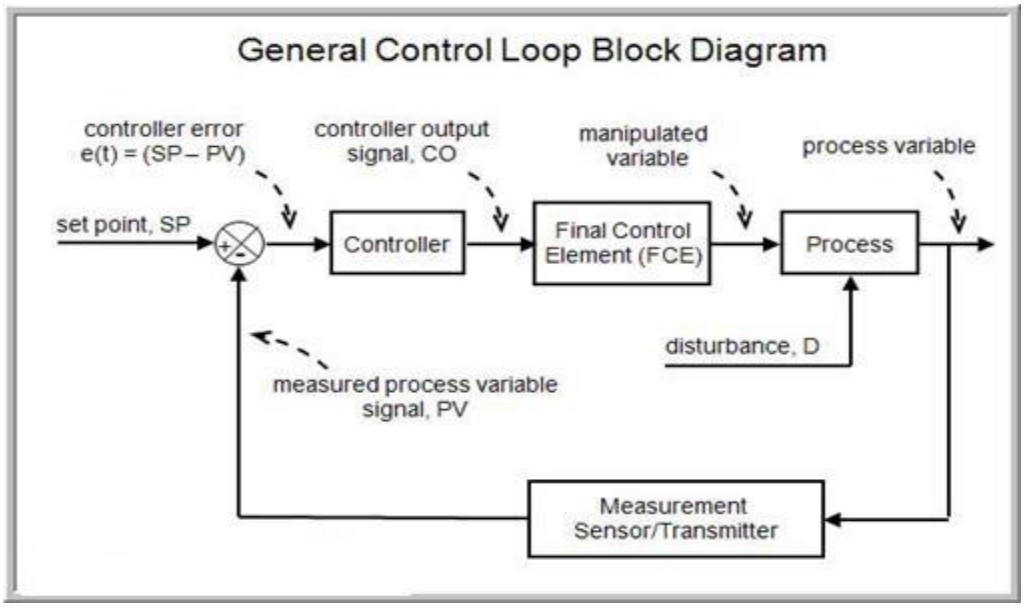


Statechart for User Activity

From the point of view as a system being acted on by many users:



General Control Loop block diagram:



Assumptions:

Initially sensors and timers input are taken by default (no timer or sensor obstacles)

Elevators Priority:

Every lift will give higher priority to inside inputs than outside inputs

If input from 1st floor outside than E1 will complete the task.

If input from 2nd floor outside than E2 will complete the task.

If input from 3rd floor outside than E3 will complete the task.

If input from 4th floor outside than E4 will complete the task.

Default State:

E1 –first floor

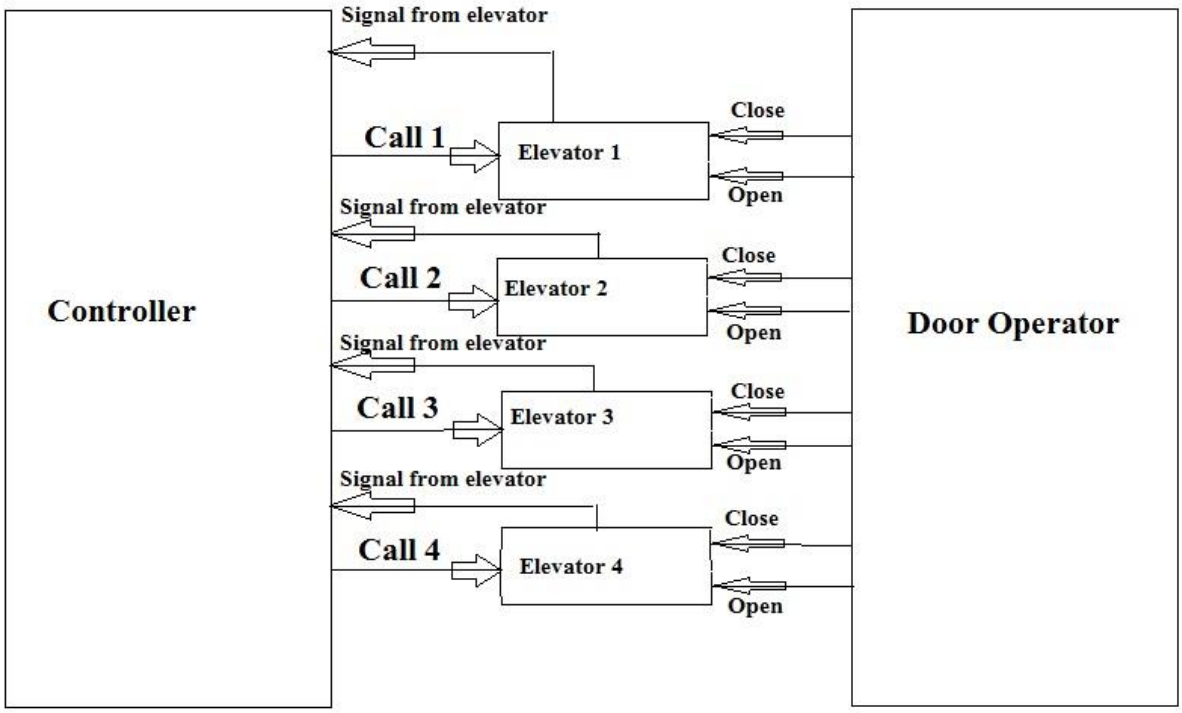
E2 –second floor

E3 –third floor

E4 –fourth floor

Working:

Every elevator has inside buttons for 1 to 4 floors and outside call button at every floor. The sensor signal will sense the obstruction and timer, so that the signal from controller will be send to door operator and hence the door operator will control the elevator according to the request.



Block diagram for controller working

Verilog Code:

`timescale 1ps/1ps // time scale of 1ns and resolution 0.01 ns

module fe_demo(input clk, reset, req_1F, req_2F, req_3F, req_4F,

req_E1_1, req_E1_2, req_E1_3, req_E1_4,

req_E2_1, req_E2_2, req_E2_3, req_E2_4,

req_E3_1, req_E3_2, req_E3_3, req_E3_4,

req_E4_1, req_E4_2, req_E4_3, req_E4_4,

```
output reg [1:0]E1_pos, E2_pos, E3_pos, E4_pos,

output reg [7:0]o1, o2, o3, o4);

reg [1:0]E1_next_pos, E2_next_pos, E3_next_pos, E4_next_pos;

reg dg;

parameter firstfloor= 2'b00,

secondfloor = 2'b01,

thirdfloor = 2'b10,

fourthfloor = 2'b11;


always@(posedge clk, reset) begin

if (reset)

begin

E1_pos <= firstfloor;

E2_pos <= secondfloor;

E3_pos <= thirdfloor;

E4_pos <= fourthfloor;

end

else

begin

E1_pos <= E1_next_pos;

E2_pos <= E2_next_pos;

E3_pos <= E3_next_pos;

E4_pos <= E4_next_pos;
```

```
end

if (E1_pos==firstfloor)

begin

o1=8'b10011111;

dg=4'b1110;

end

if (E1_pos==secondfloor)

begin

o1=8'b00100101;

dg=4'b1110;

end

if (E1_pos==thirdfloor)

begin

o1=8'b00001101;

dg=4'b1110;

end

if (E1_pos==fourthfloor)

begin

o1=8'b10011001;

dg=4'b1110;

end

if (E2_pos==firstfloor)

begin

o2=8'b10011111;
```

```
dg=4'b1101;
end
if (E2_pos==secondfloor)
begin
o2=8'b00100101;
dg=4'b1101;
end
if (E2_pos==thirdfloor)
begin
o2=8'b00001101;
dg=4'b1101;
end
if (E2_pos==fourthfloor)
begin
o2=8'b10011001;
dg=4'b1101;
end
if (E3_pos==firstfloor)
begin
o3=8'b10011111;
dg=4'b1011;
end
if (E3_pos==secondfloor)
begin
```

```
o3=8'b00100101;

dg=4'b1011;

end

if (E3_pos==thirdfloor)

begin

o3=8'b00001101;

dg=4'b1011;

end

if (E3_pos==fourthfloor)

begin

o3=8'b10011001;

dg=4'b1011;

end

if (E4_pos==firstfloor)

begin

o4=8'b10011111;

dg=4'b0111;

end

if (E4_pos==secondfloor)

begin

o4=8'b00100101;

dg=4'b0111;

end

if (E4_pos==thirdfloor)
```

begin

o4=8'b00001101;

dg=4'b0111;

end

if (E4_pos==fourthfloor)

begin

o4=8'b10011001;

dg=4'b0111;

end

end

always@(posedge clk, reset, E1_pos, E2_pos, E3_pos, E4_pos, req_1F, req_2F, req_3F, req_4F, req_E1_1, req_E1_2, req_E1_3, req_E1_4,

req_E2_1, req_E2_2, req_E2_3, req_E2_4, req_E3_1, req_E3_2, req_E3_3, req_E3_4, req_E4_1, req_E4_2, req_E4_3, req_E4_4)

begin

begin

if (req_1F==1)

begin

E1_next_pos<=firstfloor;

end

if (req_2F==1)

begin

E2_next_pos<=secondfloor;

end

if (req_3F==1)

```
begin
E3_next_pos<=thirdfloor;
end
if (req_4F==1)
begin
E4_next_pos<=fourthfloor;
end
end
begin
if (req_E1_1==1)
E1_next_pos<=firstfloor;
if (req_E1_2==1)
E1_next_pos<=secondfloor;
if (req_E1_3==1)
E1_next_pos<=thirdfloor;
if (req_E1_4==1)
E1_next_pos<=fourthfloor;
end
begin
if (req_E2_1==1)
E2_next_pos<=firstfloor;
if (req_E2_2==1)
E2_next_pos<=secondfloor;
if (req_E2_3==1)
```

```
E2_next_pos<=thirdfloor;

if (req_E2_4==1)

E2_next_pos<=fourthfloor;

end

begin

if (req_E3_1==1)

E3_next_pos<=firstfloor;

if (req_E3_2==1)

E3_next_pos<=secondfloor;

if (req_E3_3==1)

E3_next_pos<=thirdfloor;

if (req_E3_4==1)

E3_next_pos<=fourthfloor;

end

begin

if (req_E4_1==1)

E4_next_pos<=firstfloor;

if (req_E4_2==1)

E4_next_pos<=secondfloor;

if (req_E4_3==1)

E4_next_pos<=thirdfloor;

if (req_E4_4==1)

E4_next_pos<=fourthfloor;

end
```

end

endmodule

verilog testbench code:

`timescale 1ns / 1ps

//

// Company:

// Engineer:

//

// Create Date: 12:13:31 11/11/2013

// Design Name: fe_demo

// Module Name: C:/Users/BHAGWAT/Documents/Xilinx/final_review/test.v

// Project Name: final_review

// Target Device:

// Tool versions:

// Description:

//

// Verilog Test Fixture created by ISE for module: fe_demo

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

```
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module test;

	// Inputs
	reg clk;
	reg reset;
	reg req_1F;
	reg req_2F;
	reg req_3F;
	reg req_4F;
	reg req_E1_1;
	reg req_E1_2;
	reg req_E1_3;
	reg req_E1_4;
	reg req_E2_1;
	reg req_E2_2;
	reg req_E2_3;
	reg req_E2_4;
	reg req_E3_1;
```

```
reg req_E3_2;

reg req_E3_3;

reg req_E3_4;

reg req_E4_1;

reg req_E4_2;

reg req_E4_3;

reg req_E4_4;


// Outputs

wire [1:0] E1_pos;

wire [1:0] E2_pos;

wire [1:0] E3_pos;

wire [1:0] E4_pos;

wire [7:0] o1;

wire [7:0] o2;

wire [7:0] o3;

wire [7:0] o4;


// Instantiate the Unit Under Test (UUT)

fe_demo uut (

        .clk(clk),

        .reset(reset),
```

.req_1F(req_1F),

.req_2F(req_2F),

.req_3F(req_3F),

.req_4F(req_4F),

.req_E1_1(req_E1_1),

.req_E1_2(req_E1_2),

.req_E1_3(req_E1_3),

.req_E1_4(req_E1_4),

.req_E2_1(req_E2_1),

.req_E2_2(req_E2_2),

.req_E2_3(req_E2_3),

.req_E2_4(req_E2_4),

.req_E3_1(req_E3_1),

.req_E3_2(req_E3_2),

.req_E3_3(req_E3_3),

.req_E3_4(req_E3_4),

.req_E4_1(req_E4_1),

.req_E4_2(req_E4_2),

.req_E4_3(req_E4_3),

.req_E4_4(req_E4_4),

.E1_pos(E1_pos),

.E2_pos(E2_pos),

```
        .E3_pos(E3_pos),
        .E4_pos(E4_pos),
        .o1(o1),
        .o2(o2),
        .o3(o3),
        .o4(o4)
);

initial begin
        // Initialize Inputs
        clk = 0;
        end
        always
        clk=#10 ~clk;
        initial begin

        reset = 1;
        req_1F = 0;
        req_2F = 0;
        req_3F = 0;
        req_4F = 0;
        req_E1_1 = 0;
```

```
req_E1_2 = 0;

req_E1_3 = 0;

req_E1_4 = 0;

req_E2_1 = 0;

req_E2_2 = 0;

req_E2_3 = 0;

req_E2_4 = 0;

req_E3_1 = 0;

req_E3_2 = 0;

req_E3_3 = 0;

req_E3_4 = 0;

req_E4_1 = 0;

req_E4_2 = 0;

req_E4_3 = 0;

req_E4_4 = 0;


// Wait 100 ns for global reset to finish

#100;

reset = 0;

req_1F = 0;

req_2F = 0;

req_3F = 0;
```

```
        req_4F = 0;
        req_E1_1 = 0;
        req_E1_2 = 1;
        req_E1_3 = 0;
        req_E1_4 = 0;
        req_E2_1 = 0;
        req_E2_2 = 0;
        req_E2_3 = 0;
        req_E2_4 = 1;
        req_E3_1 = 1;
        req_E3_2 = 0;
        req_E3_3 = 0;
        req_E3_4 = 0;
        req_E4_1 = 0;
        req_E4_2 = 0;
        req_E4_3 = 1;
        req_E4_4 = 0;

        // Wait 100 ns for global reset to finish
        #100;
reset = 0;
        req_1F = 1;
```

```
req_2F = 1;

req_3F = 1;

req_4F = 1;

req_E1_1 = 0;

req_E1_2 = 0;

req_E1_3 = 0;

req_E1_4 = 0;

req_E2_1 = 0;

req_E2_2 = 0;

req_E2_3 = 0;

req_E2_4 = 0;

req_E3_1 = 0;

req_E3_2 = 0;

req_E3_3 = 0;

req_E3_4 = 0;

req_E4_1 = 0;

req_E4_2 = 0;

req_E4_3 = 0;

req_E4_4 = 0;



// Add stimulus here
```

end

endmodule

Simulation result:

For t=1ns to 100 ns

Inputs:

Reset=1

Output:

E1-first floor

E2-second floor

E3-third floor

E4-fourth floor

For t=101ns to 200ns

Inputs:

Inside 2nd floor request from E1 elevator

Inside 4th floor request from E2 elevator

Inside 1st floor request from E3 elevator

Inside 3rd floor request from E4 elevator

Outputs:

E1-second floor

E2-fourth floor

E3-first floor

E4-third floor

For t=201ns to 300ns

Inputs:

Request from 1st floor outside

Request from 2nd floor outside

Request from 3rd floor outside

Request from 4th floor outside

Outputs:

E1-first floor

E2-second floor

E3-third floor

E4-fourth floor

Conclusion:

Four elevator controller is implemented successfully on FPGA board using Xilinx tool. It is also simulated on Xilinx tool.