AIM:

To design and implement an elevator controller system using Verilog HDL on an FPGA that manages elevator operations efficiently providing real-time service in multi-story buildings.

INTRODUCTION:

This project utilizes Verilog hardware description language (HDL) to design and implement an FPGA-based elevator controller system. With the continuous growth of urban infrastructure, efficient elevator management has become essential to minimizing wait times and optimizing energy consumption in high-rise buildings. The project showcases how an FPGA can be leveraged to develop a system that dynamically handles elevator operations and incorporates a password-protected access mechanism for enhanced security. The system monitors user requests for elevator service, manages floor selection, and updates the elevator's current position in real-time, ensuring unauthorized access is prevented through password verification.

The inherent parallel processing capabilities of FPGAs enable the system to perform multiple functions simultaneously, boosting efficiency and response time.

APPLICATIONS:

Elevator controller systems play a crucial role in managing efficient and secure transportation within multi-story buildings. They are essential in residential complexes, commercial offices, hospitals, and hotels, where they streamline vertical movement and reduce wait times. With password-protected access, they enhance security in government buildings, data centres, and industrial facilities. Integrated into smart building automation, these systems optimize energy consumption and adapt operations based on occupancy. In educational institutions and shopping malls, they effectively manage large foot traffic and minimize congestion, improving user experience. Their ability to provide real-time responsiveness using FPGA technology makes them indispensable in modern infrastructure.

COMPONENTS USED:

FPGA MAX10 10M50DAF484C7G

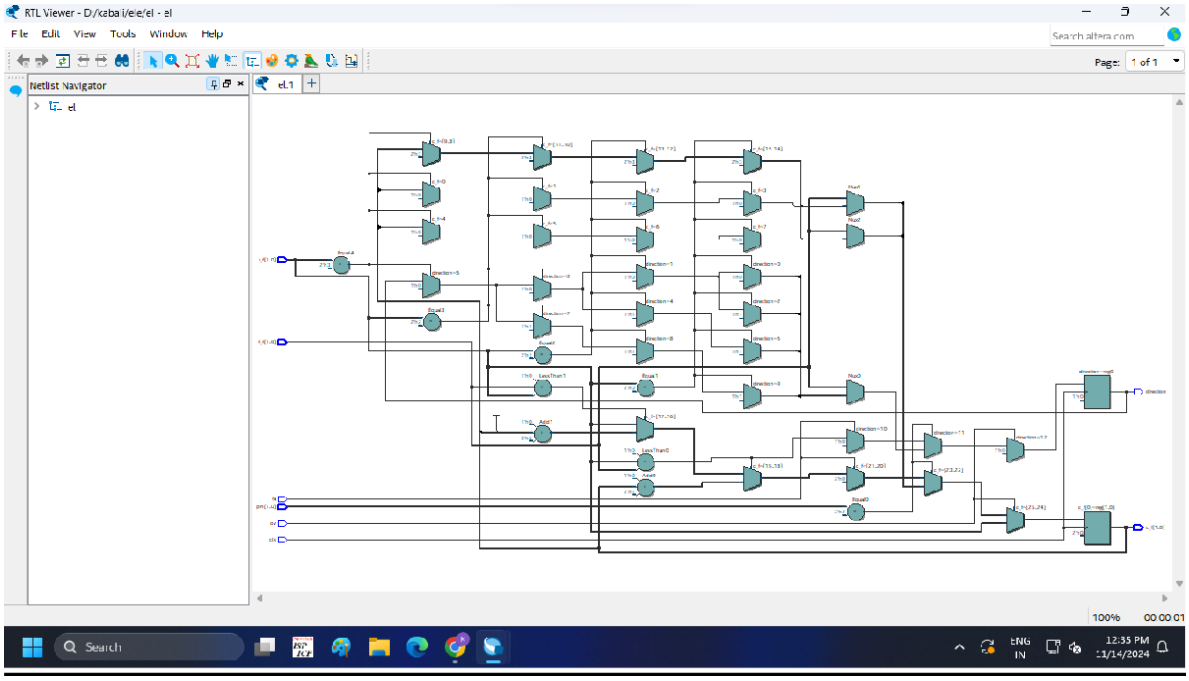
CODE:

```
module el (
   input [1:0] i_f,
```

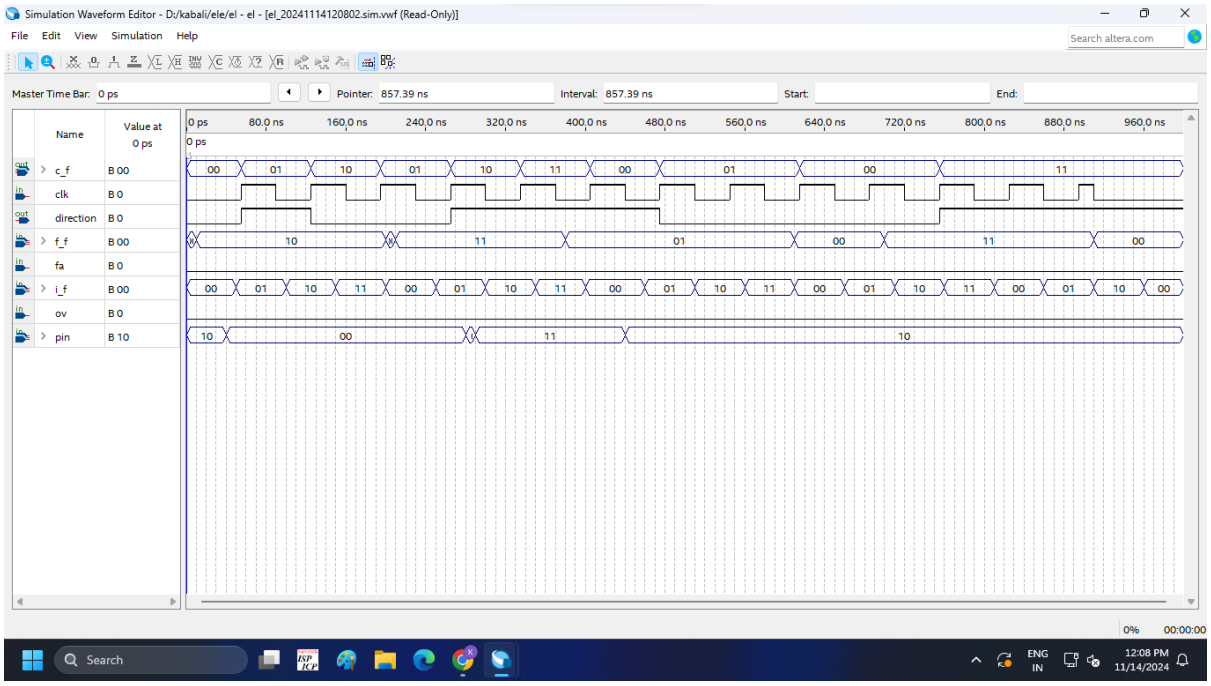
```
    input [1:0] f_f,
    input clk,
    input ov, fa,
    input [1:0] pin,
    output reg direction,
    output reg [1:0] c_f
);

always @(posedge clk) begin
    if (ov == 1) begin
        c_f = i_f;
        direction = 0;
    end
    else if (pin == 2'b10) begin
        case (f_f)
            2'b00: begin
                if (i_f == 2'b00) begin direction = 1'b0; c_f = 2'b00; end
                else if (i_f == 2'b01) begin direction = 1'b0; c_f = 2'b00; end
                else if (i_f == 2'b10) begin direction = 1'b0; c_f = 2'b00; end
                else if (i_f == 2'b11) begin direction = 1'b0; c_f = 2'b00; end
            end
            2'b01: begin
                if (i_f == 2'b00) begin direction = 1'b1; c_f = 2'b01; end
                else if (i_f == 2'b01) begin direction = 1'b0; c_f = 2'b01; end
                else if (i_f == 2'b10) begin direction = 1'b0; c_f = 2'b01; end
                else if (i_f == 2'b11) begin direction = 1'b0; c_f = 2'b01; end
            end
            2'b10: begin
                if (i_f == 2'b00) begin direction = 1'b1; c_f = 2'b10; end
                else if (i_f == 2'b01) begin direction = 1'b1; c_f = 2'b10; end
                else if (i_f == 2'b10) begin direction = 1'b0; c_f = 2'b10; end
                else if (i_f == 2'b11) begin direction = 1'b0; c_f = 2'b10; end
            end
            2'b11: begin
                if (i_f == 2'b00) begin direction = 1'b1; c_f = 2'b11; end
                else if (i_f == 2'b01) begin direction = 1'b1; c_f = 2'b11; end
                else if (i_f == 2'b10) begin direction = 1'b1; c_f = 2'b11; end
                else if (i_f == 2'b11) begin direction = 1'b0; c_f = 2'b11; end
            end
        endcase
```

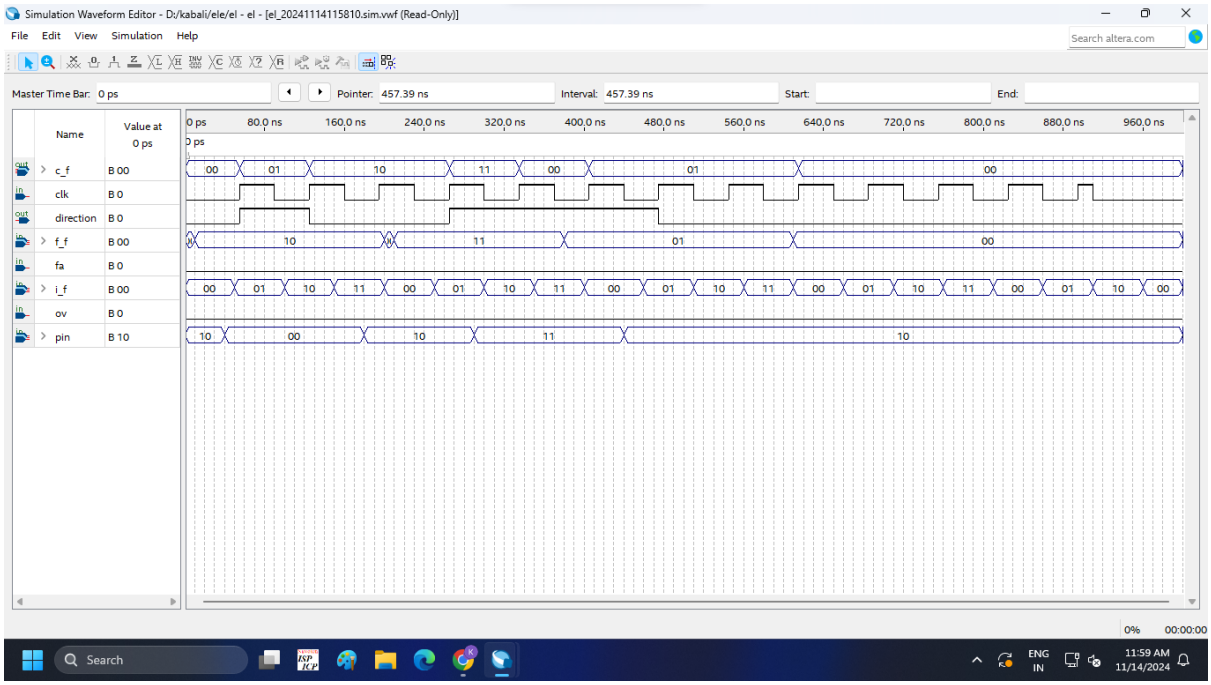
```
    end
    else if (fa == 1) begin
        c_f = 2'b00;
        direction = 0;
    end
    else begin
        if (i_f < f_f) begin
            c_f <= c_f + 1;
            direction <= 1'b1;
        end
        else if (i_f > f_f) begin
            c_f <= c_f - 1;
            direction <= 1'b0;
        end
        else begin
            c_f <= i_f;
            direction <= 1'b0;
        end
    end
end

endmodule
```

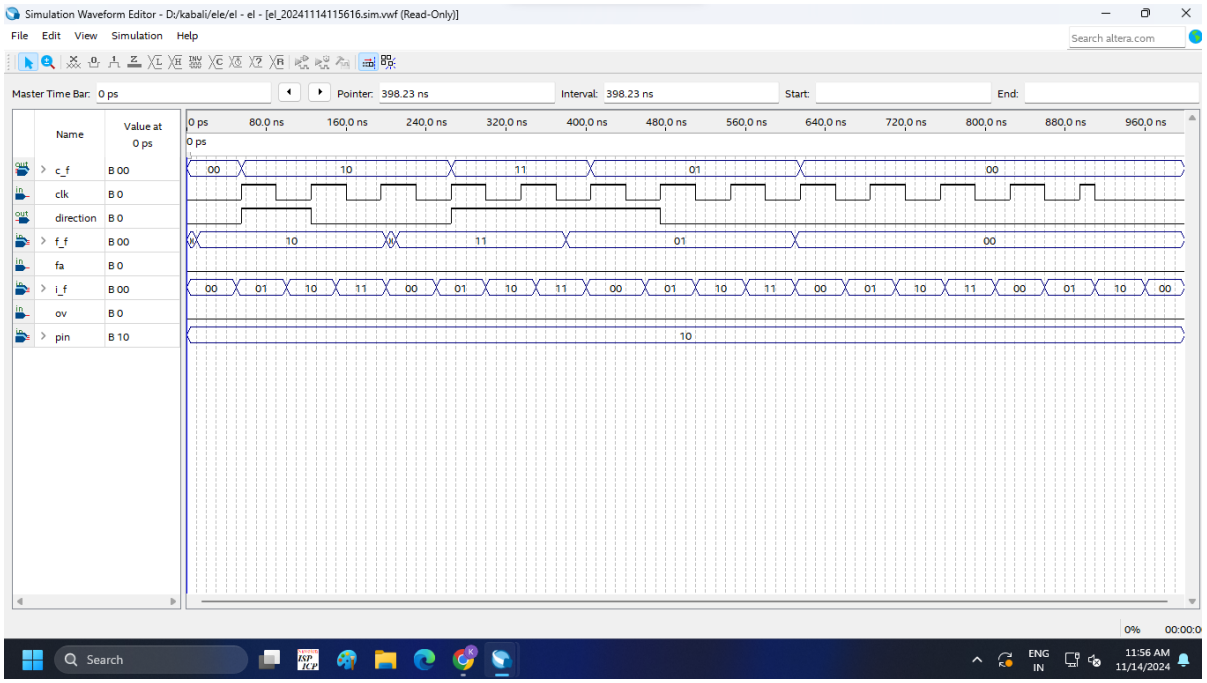
RTL VIEW:



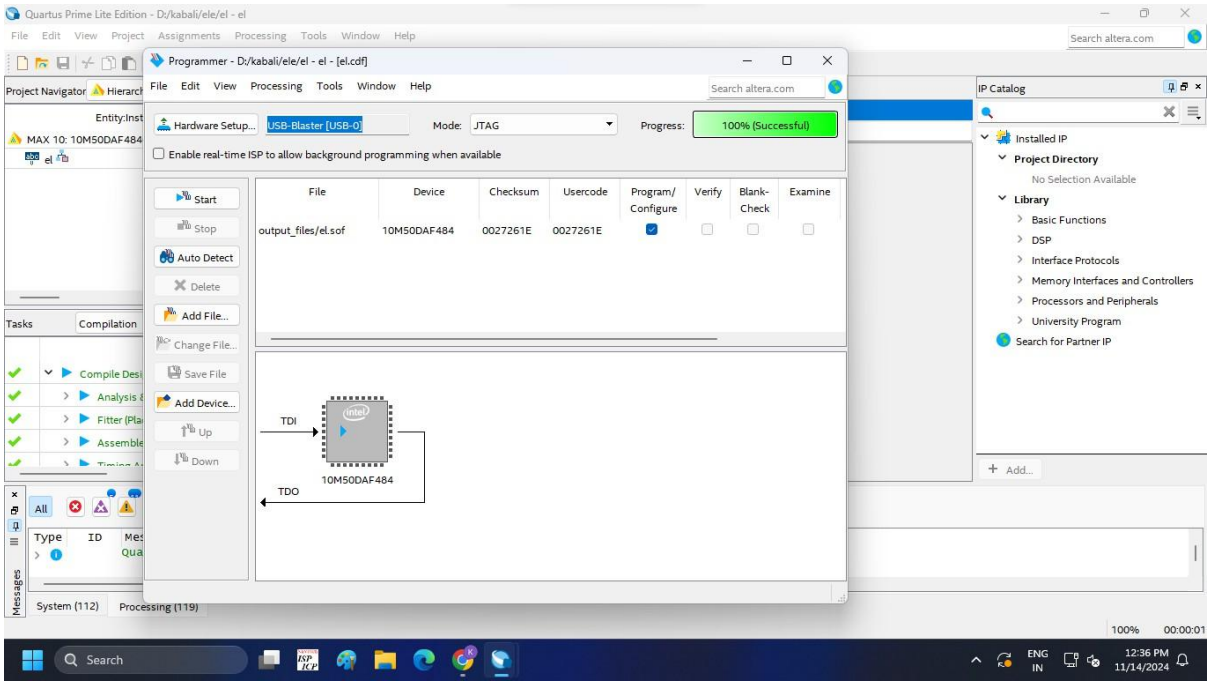
WAVEFORM:

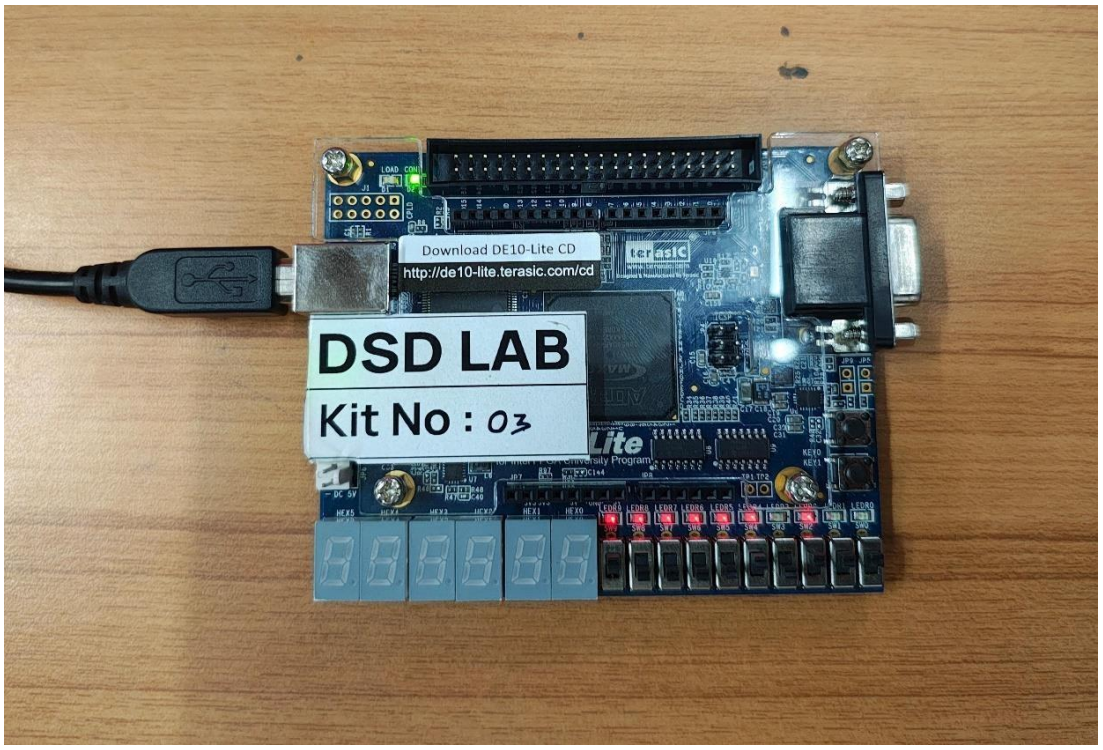

Simulation Waveform Editor - D:/kabali/ele/el - el - [el_20241114120802.sim.vwf (Read-Only)]
File Edit View Simulation Help
Search altera.com
Master Time Bar: 0 ps
Pointer: 857.39 ns
Interval: 857.39 ns
Start:
End:
Name
Value at 0 ps
c_f B 00
clk B 0
direction B 0
f_f B 00
fa B 0
i_f B 00
ov B 0
pin B 10
0 ps 80.0 ns 160.0 ns 240.0 ns 320.0 ns 400.0 ns 480.0 ns 560.0 ns 640.0 ns 720.0 ns 800.0 ns 880.0 ns 960.0 ns
0%
00:00:00
Search
ENG IN
12:08 PM 11/14/2024

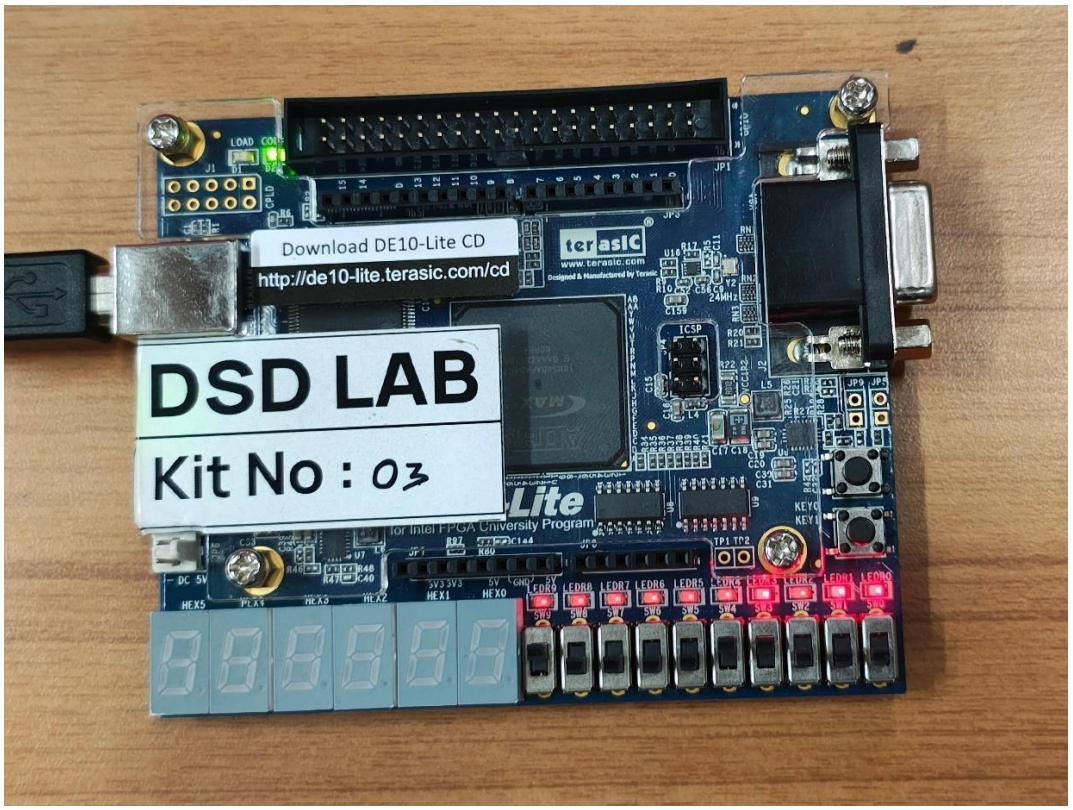

Simulation Waveform Editor - D:/kabali/ele/el - el - [el_20241114115810.sim.vwf (Read-Only)]
File Edit View Simulation Help
Search altera.com
Master Time Bar: 0 ps
Pointer: 457.39 ns
Interval: 457.39 ns
Start:
End:
Name
Value at 0 ps
c_f B 00
clk B 0
direction B 0
f_f B 00
fa B 0
i_f B 00
ov B 0
pin B 10
0 ps 80.0 ns 160.0 ns 240.0 ns 320.0 ns 400.0 ns 480.0 ns 560.0 ns 640.0 ns 720.0 ns 800.0 ns 880.0 ns 960.0 ns
0%
00:00:00
Search
ENG IN
11:59 AM 11/14/2024

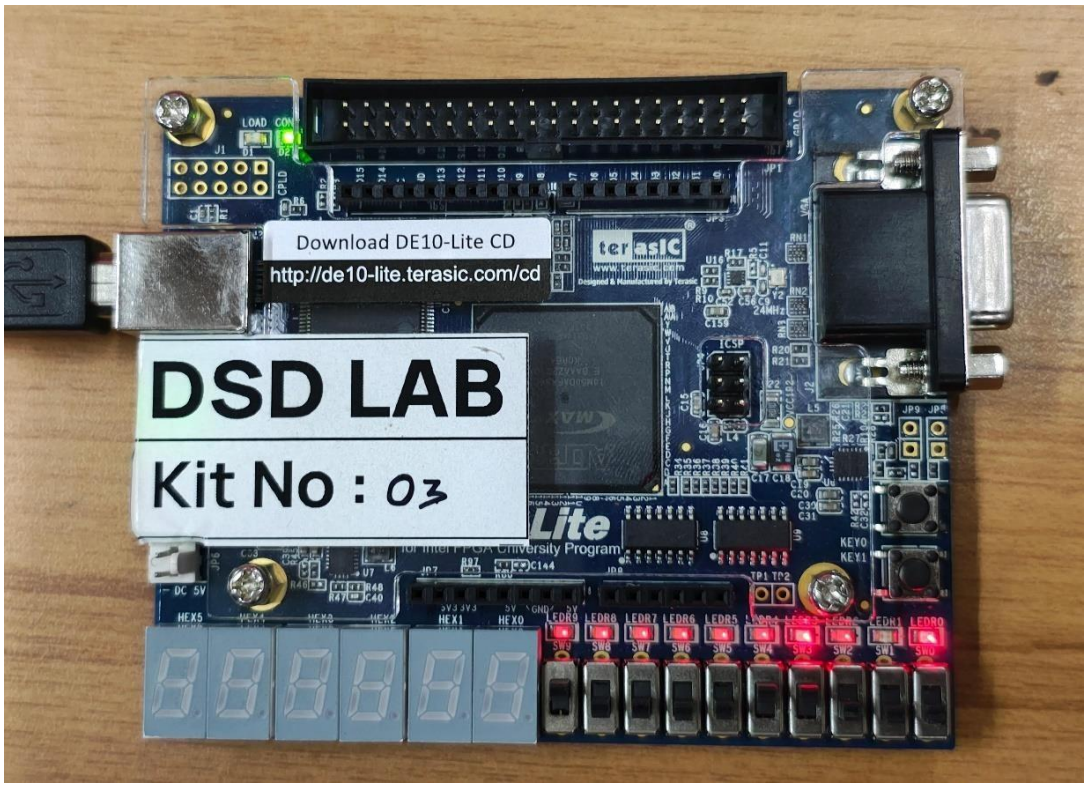


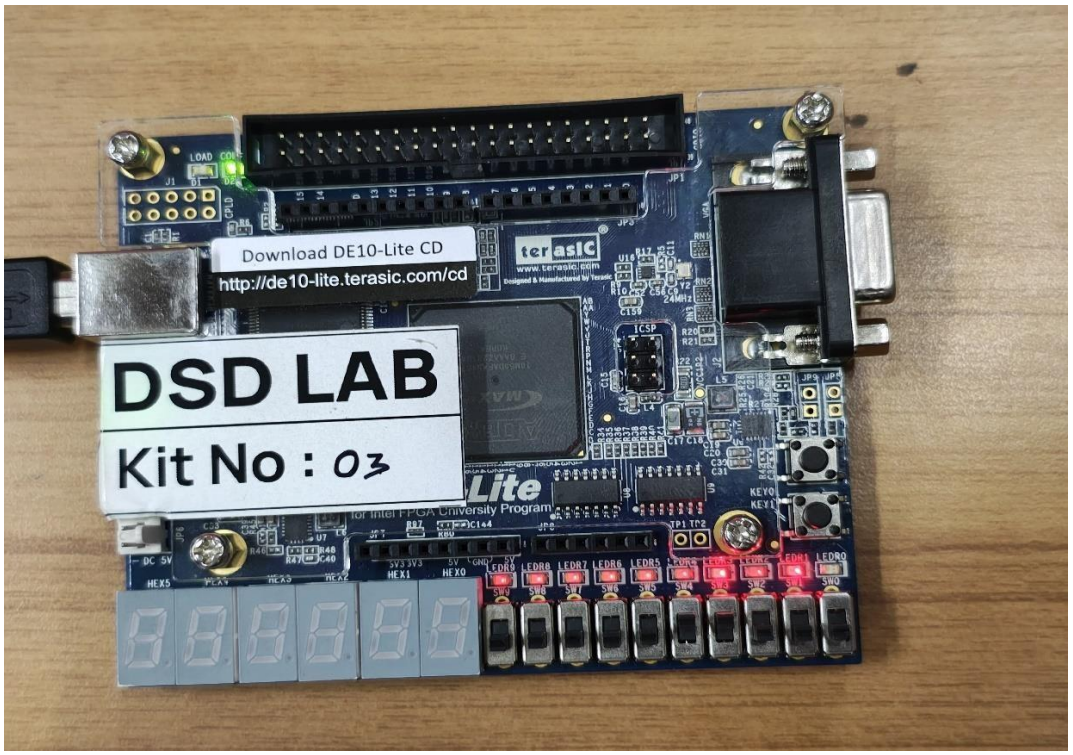
IMPLEMENTATION ON FPGA:

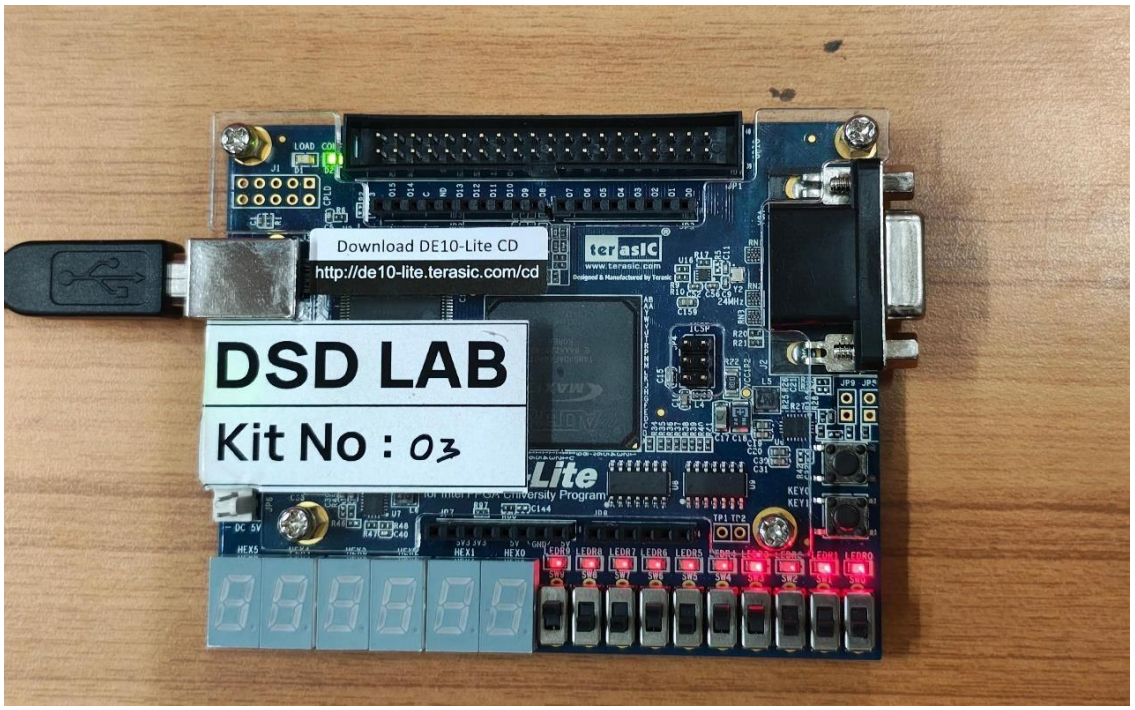


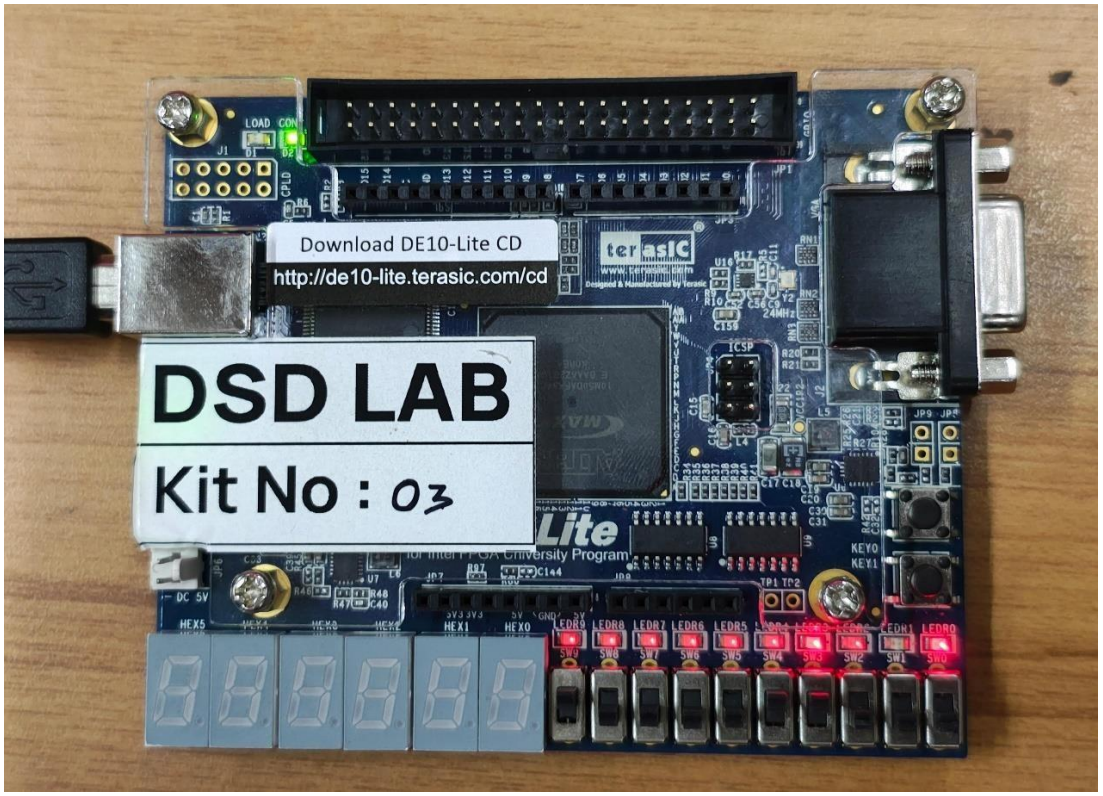

Download DE10-Lite CD
http://de10-lite.terasic.com/cd
DSD LAB
Kit No : 03


Download DE10-Lite CD
http://de10-lite.terasic.com/cd
DSD LAB
Kit No : 03


Download DE10-Lite CD
http://de10-lite.terasic.com/cd
DSD LAB
Kit No : 03


Download DE10-Lite CD
http://de10-lite.terasic.com/cd
DSD LAB
Kit No : 03


Download DE10-Lite CD
http://de10-lite.terasic.com/cd
DSD LAB
Kit No : 03


Download DE10-Lite CD
http://de10-lite.terasic.com/cd
DSD LAB
Kit No : 03