ECE3300L Lab 2

Group B

By Faris Khan (ID #: 012621102)

25 June 2025

Design

Gate-Level:

```
module decoder4x16_gate (
    input  wire [3:0] SW,    // Slide switches SW3..SW0
    input  wire       EN,    // Enable switch SW4
    output wire [15:0] LED   // LEDs LED0..LED15
);

    // Generate each LED output
    assign LED[0]  = EN & ~SW[3] & ~SW[2] & ~SW[1] & ~SW[0];
    assign LED[1]  = EN & ~SW[3] & ~SW[2] & ~SW[1] &  SW[0];
    assign LED[2]  = EN & ~SW[3] & ~SW[2] &  SW[1] & ~SW[0];
    assign LED[3]  = EN & ~SW[3] & ~SW[2] &  SW[1] &  SW[0];
    assign LED[4]  = EN & ~SW[3] &  SW[2] & ~SW[1] & ~SW[0];
    assign LED[5]  = EN & ~SW[3] &  SW[2] & ~SW[1] &  SW[0];
    assign LED[6]  = EN & ~SW[3] &  SW[2] &  SW[1] & ~SW[0];
    assign LED[7]  = EN & ~SW[3] &  SW[2] &  SW[1] &  SW[0];
    assign LED[8]  = EN &  SW[3] & ~SW[2] & ~SW[1] & ~SW[0];
    assign LED[9]  = EN &  SW[3] & ~SW[2] & ~SW[1] &  SW[0];
    assign LED[10] = EN &  SW[3] & ~SW[2] &  SW[1] & ~SW[0];
    assign LED[11] = EN &  SW[3] & ~SW[2] &  SW[1] &  SW[0];
    assign LED[12] = EN &  SW[3] &  SW[2] & ~SW[1] & ~SW[0];
    assign LED[13] = EN &  SW[3] &  SW[2] & ~SW[1] &  SW[0];
    assign LED[14] = EN &  SW[3] &  SW[2] &  SW[1] & ~SW[0];
    assign LED[15] = EN &  SW[3] &  SW[2] &  SW[1] &  SW[0];

endmodule
```

In the gate-level implementation, we manually construct the logic equations for each output line. Each output is generated by AND'ing the enable signal with a specific combination of the input bits, using both the true and inverted forms This mirrors how digital circuits would be constructed using basic gates such as in a logic diagram.

Behavioral level:

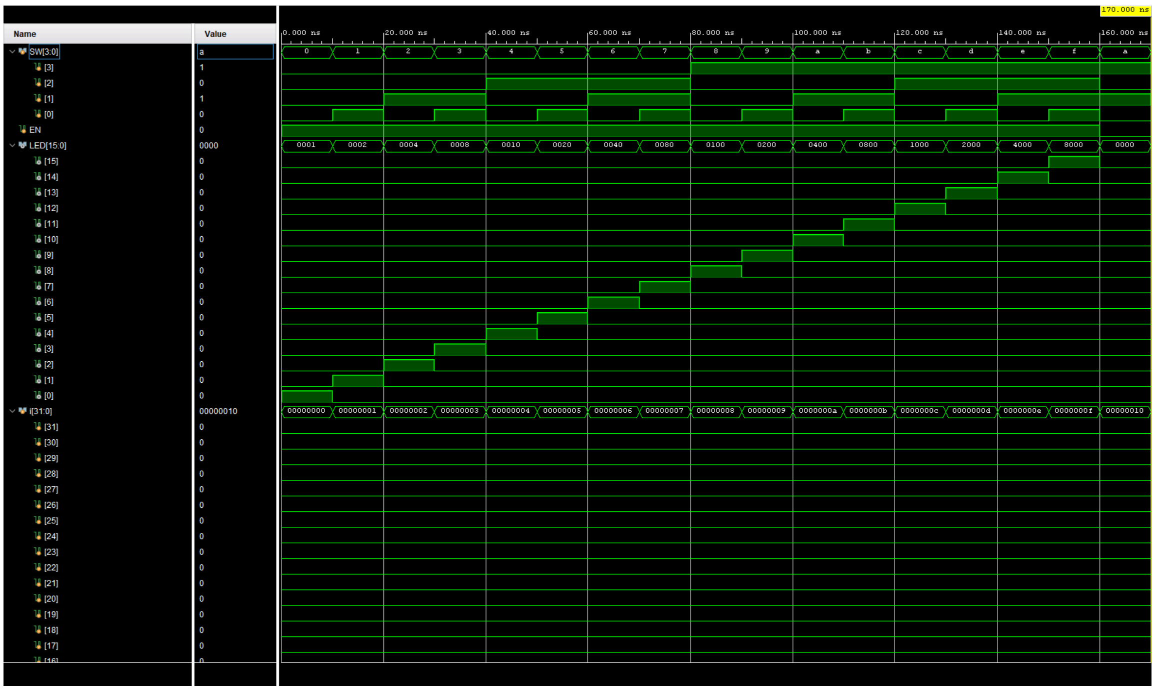
```
module decoder4x16_behav (
    input  wire [3:0] SW,    // Slide switches SW3..SW0
    input  wire       EN,    // Enable switch SW4
    output reg  [15:0] LED   // LEDs LED0..LED15
);

    always @(*) begin
        LED = 16'b0;            // reset all LEDs to 0
        if (EN) begin          // only decode when enabled
            case (SW)
                4'b0000: LED = 16'b0000_0000_0000_0001; // LED0 on
                4'b0001: LED = 16'b0000_0000_0000_0010; // LED1 on
                4'b0010: LED = 16'b0000_0000_0000_0100; // LED2 on
                4'b0011: LED = 16'b0000_0000_0000_1000; // LED3 on
                4'b0100: LED = 16'b0000_0000_0001_0000; // LED4 on
                4'b0101: LED = 16'b0000_0000_0010_0000; // LED5 on
                4'b0110: LED = 16'b0000_0000_0100_0000; // LED6 on
                4'b0111: LED = 16'b0000_0000_1000_0000; // LED7 on
                4'b1000: LED = 16'b0000_0001_0000_0000; // LED8 on
                4'b1001: LED = 16'b0000_0010_0000_0000; // LED9 on
                4'b1010: LED = 16'b0000_0100_0000_0000; // LED10 on
                4'b1011: LED = 16'b0000_1000_0000_0000; // LED11 on
                4'b1100: LED = 16'b0001_0000_0000_0000; // LED12 on
                4'b1101: LED = 16'b0010_0000_0000_0000; // LED13 on
                4'b1110: LED = 16'b0100_0000_0000_0000; // LED14 on
                4'b1111: LED = 16'b1000_0000_0000_0000; // LED15 on
                default: LED = 16'b0;
            endcase
        end
    end
endmodule
```

In this behavioral implementation, the function of the code is being described: As long as the enable switch is on, output the corresponding LED based on which switches are turned on and off. By default, LED 0 will be on if the enable switch is on.

Simulation

Waveform:



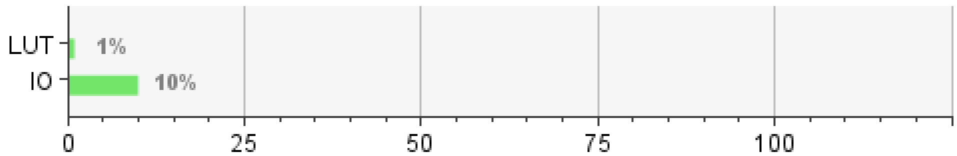
Testbench:

```
PASS: SW=0000, LED=0000000000000001
PASS: SW=0001, LED=0000000000000010
PASS: SW=0010, LED=0000000000000100
PASS: SW=0011, LED=0000000000001000
PASS: SW=0100, LED=0000000000010000
PASS: SW=0101, LED=0000000000100000
PASS: SW=0110, LED=0000000001000000
PASS: SW=0111, LED=0000000010000000
PASS: SW=1000, LED=0000000100000000
PASS: SW=1001, LED=0000001000000000
PASS: SW=1010, LED=0000010000000000
PASS: SW=1011, LED=0000100000000000
PASS: SW=1100, LED=0001000000000000
PASS: SW=1101, LED=0010000000000000
PASS: SW=1110, LED=0100000000000000
PASS: SW=1111, LED=1000000000000000
PASS: EN=0, all LEDs off
```

Implementation

Utilization Table:

Resource	Utilization	Available	Utilization %
LUT	8	63400	0.01
IO	21	210	10.00



Design Timing Summary:

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	16	Total Number of Endpoints:	16	Total Number of Endpoints:	NA

There are no user specified timing constraints.

Contribution

Faris: Code, Report, Video - 100%

Video Link: https://www.youtube.com/watch?v=tyky2iLUO2c