ECE 3300 Lab 2:

4x16 Decoder Design

Group J Team Members:
Sean Go - 016224728
Ryan Tran - 016127189

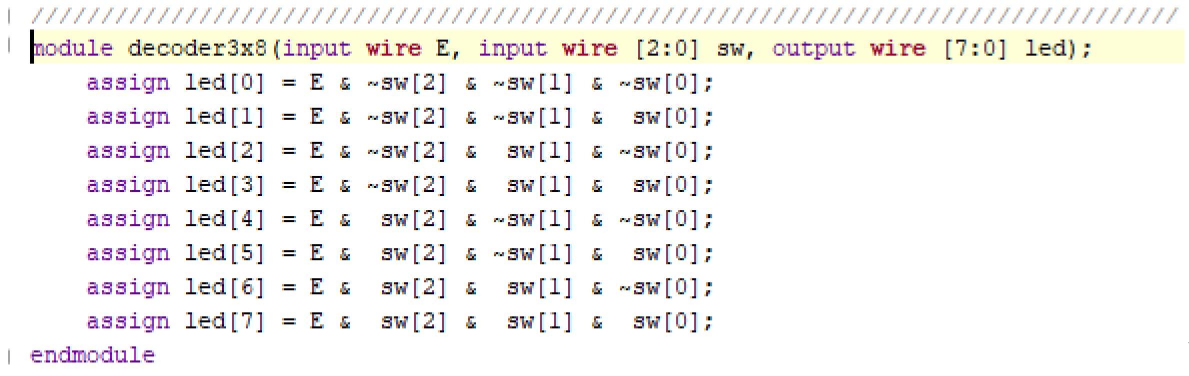
June 25, 2025

Introduction

This lab was to design, simulate, and implement a 4-to-16 line decoder using Verilog HDL on the Nexys A7-100T FPGA board. The decoder takes a 4-bit input along with an enable signal and activates exactly one of sixteen outputs when enabled, while keeping all outputs low when disabled. Through this lab, we practiced both structural (gate-level) and behavioral coding techniques, created a self-checking testbench for functional verification, and explored FPGA resource utilization and timing analysis using Xilinx Vivado. The hands-on implementation provided practical experience in digital design workflows and hardware testing.

Gate-Level Implementation

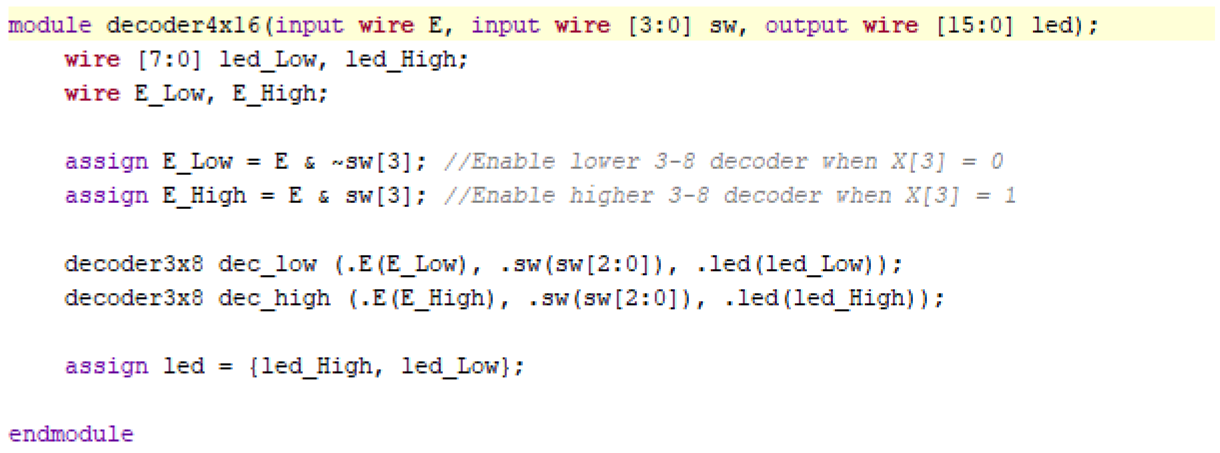
decoder3x8



```
//////////////////////////////////////////////////////////////////////////////////
module decoder3x8(input wire E, input wire [2:0] sw, output wire [7:0] led);
    assign led[0] = E & ~sw[2] & ~sw[1] & ~sw[0];
    assign led[1] = E & ~sw[2] & ~sw[1] &  sw[0];
    assign led[2] = E & ~sw[2] &  sw[1] & ~sw[0];
    assign led[3] = E & ~sw[2] &  sw[1] &  sw[0];
    assign led[4] = E &  sw[2] & ~sw[1] & ~sw[0];
    assign led[5] = E &  sw[2] & ~sw[1] &  sw[0];
    assign led[6] = E &  sw[2] &  sw[1] & ~sw[0];
    assign led[7] = E &  sw[2] &  sw[1] &  sw[0];
endmodule
```

\

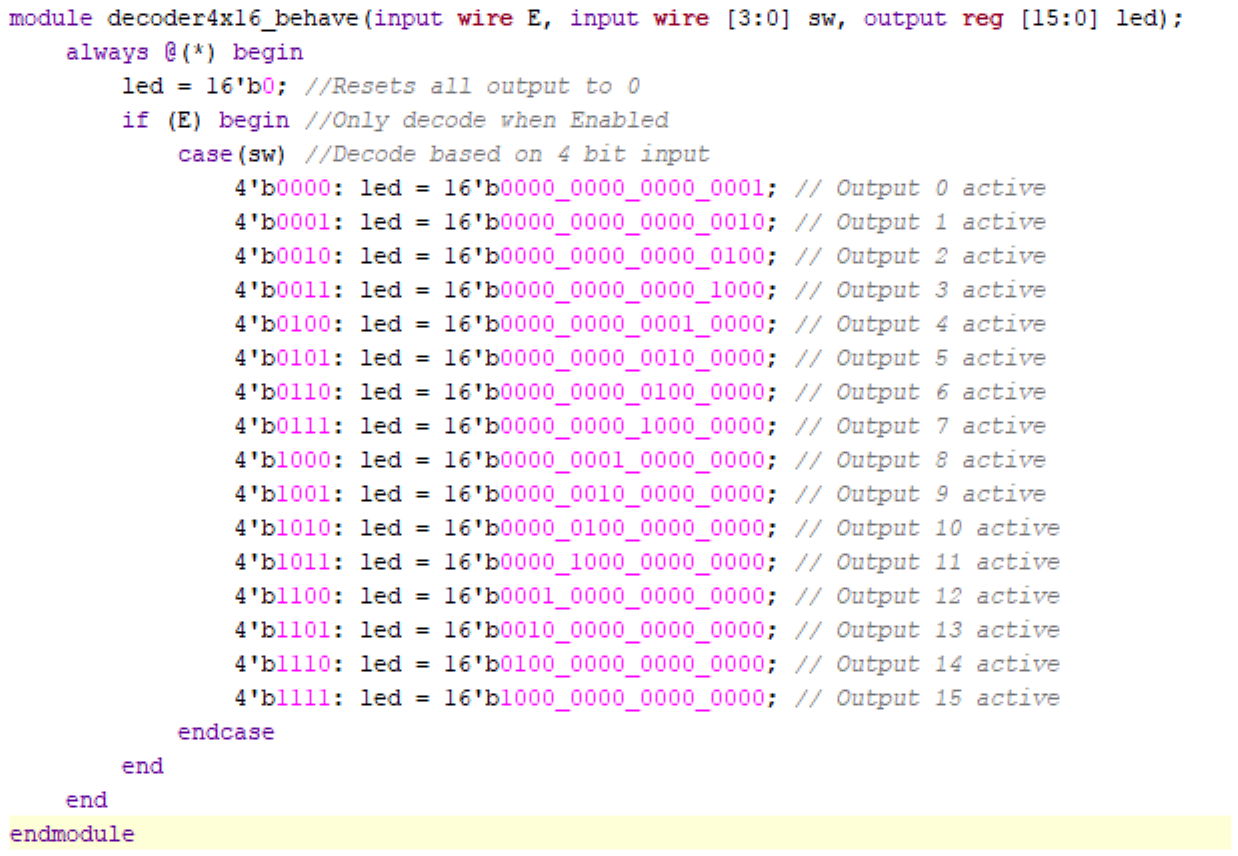
decoder4x16



```
module decoder4x16(input wire E, input wire [3:0] sw, output wire [15:0] led);
    wire [7:0] led_Low, led_High;
    wire E_Low, E_High;

    assign E_Low = E & ~sw[3]; //Enable lower 3-8 decoder when X[3] = 0
    assign E_High = E & sw[3]; //Enable higher 3-8 decoder when X[3] = 1

    decoder3x8 dec_low (.E(E_Low), .sw(sw[2:0]), .led(led_Low));
    decoder3x8 dec_high (.E(E_High), .sw(sw[2:0]), .led(led_High));

    assign led = {led_High, led_Low};

endmodule
```

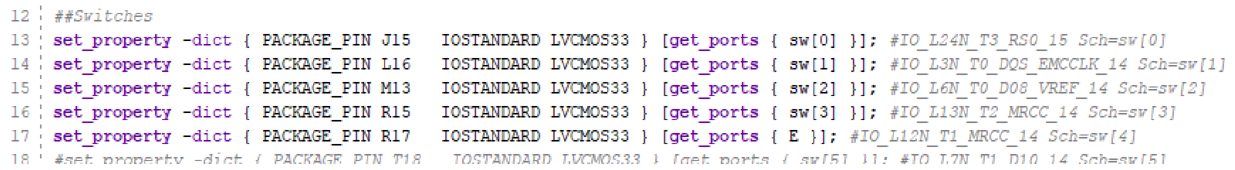
Behavioral Implementation:

decoder4x16_behave

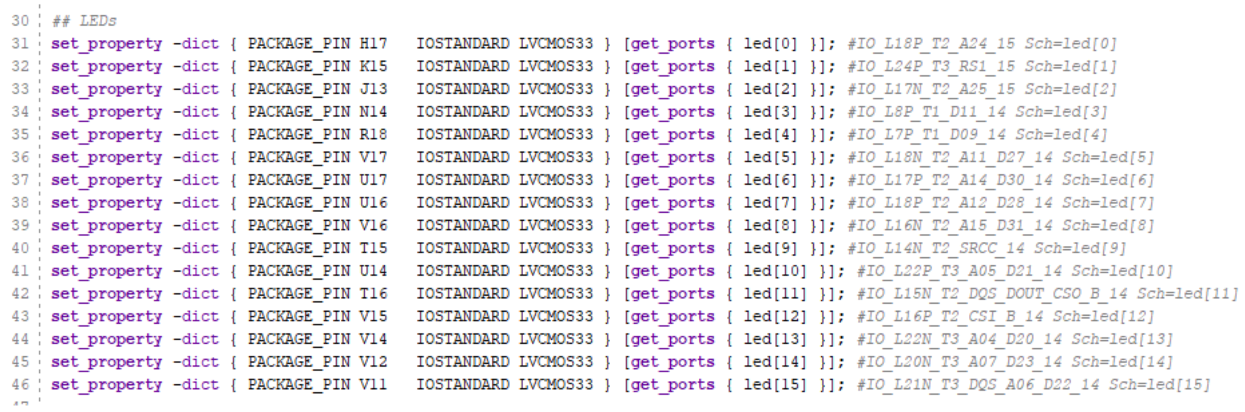


```verilog
module decoder4x16_behave(input wire E, input wire [3:0] sw, output reg [15:0] led);
    always @(*) begin
        led = 16'b0; //Resets all output to 0
        if (E) begin //Only decode when Enabled
            case(sw) //Decode based on 4 bit input
                4'b0000: led = 16'b0000_0000_0000_0001; // Output 0 active
                4'b0001: led = 16'b0000_0000_0000_0010; // Output 1 active
                4'b0010: led = 16'b0000_0000_0000_0100; // Output 2 active
                4'b0011: led = 16'b0000_0000_0000_1000; // Output 3 active
                4'b0100: led = 16'b0000_0000_0001_0000; // Output 4 active
                4'b0101: led = 16'b0000_0000_0010_0000; // Output 5 active
                4'b0110: led = 16'b0000_0000_0100_0000; // Output 6 active
                4'b0111: led = 16'b0000_0000_1000_0000; // Output 7 active
                4'b1000: led = 16'b0000_0001_0000_0000; // Output 8 active
                4'b1001: led = 16'b0000_0010_0000_0000; // Output 9 active
                4'b1010: led = 16'b0000_0100_0000_0000; // Output 10 active
                4'b1011: led = 16'b0000_1000_0000_0000; // Output 11 active
                4'b1100: led = 16'b0001_0000_0000_0000; // Output 12 active
                4'b1101: led = 16'b0010_0000_0000_0000; // Output 13 active
                4'b1110: led = 16'b0100_0000_0000_0000; // Output 14 active
                4'b1111: led = 16'b1000_0000_0000_0000; // Output 15 active
            endcase
        end
    end
endmodule
```

XDC Code

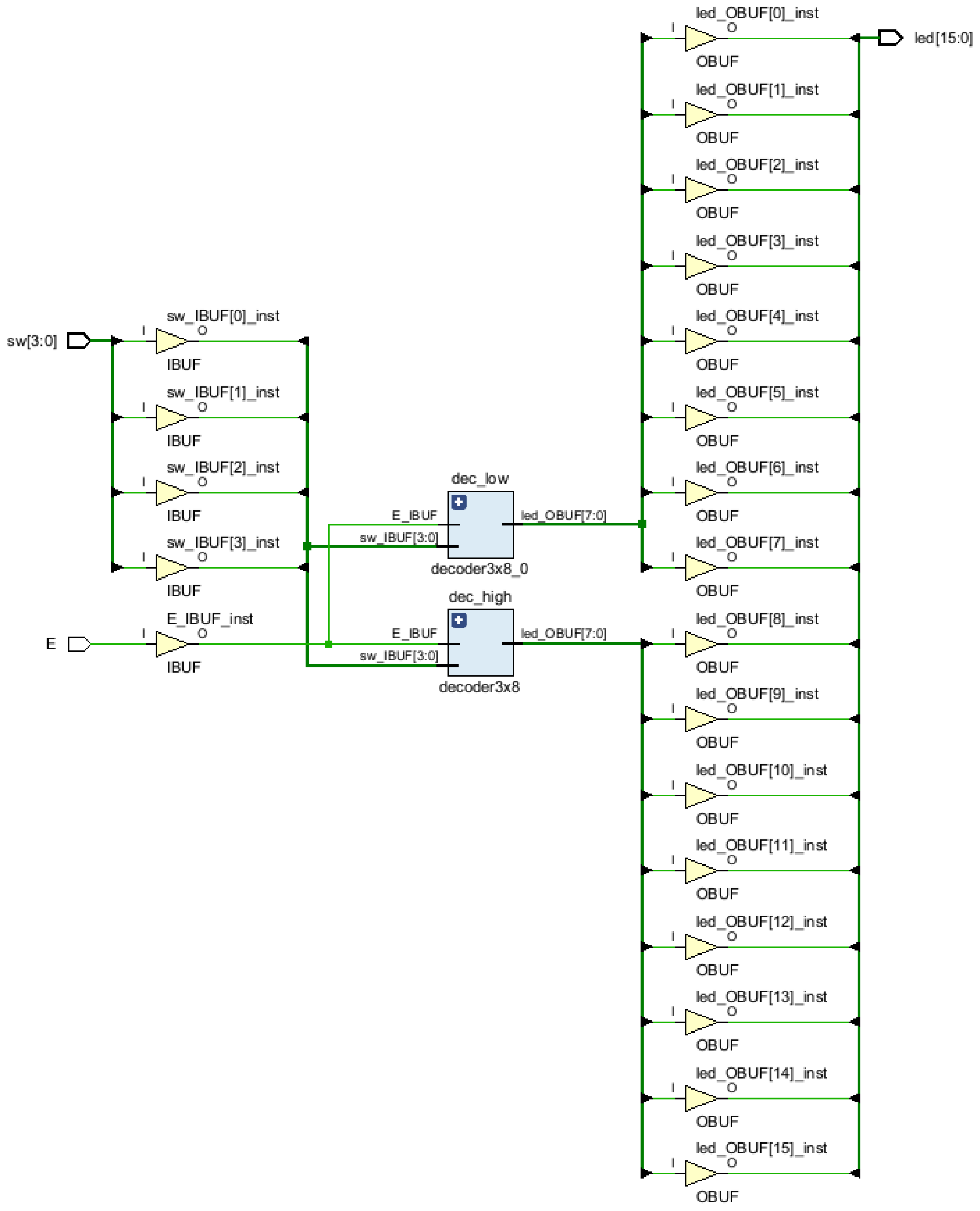


```
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { sw[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { sw[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { sw[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { E }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { sw[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
```

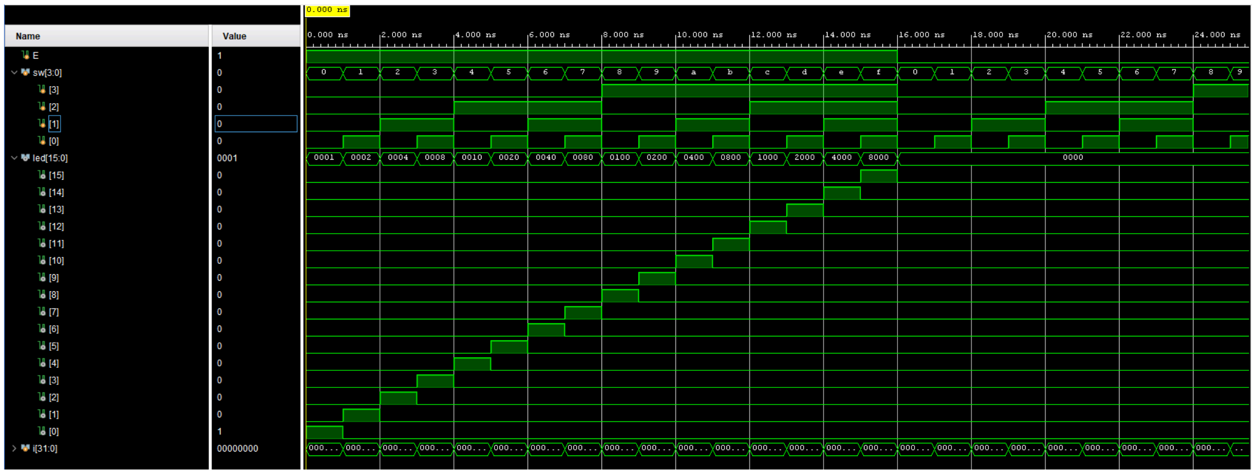


```
## LEDs
set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { led[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { led[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { led[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { led[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { led[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { led[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { led[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { led[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { led[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { led[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports { led[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { led[11] }]; #IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch=led[11]
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { led[12] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports { led[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports { led[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 } [get_ports { led[15] }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]
```

Schematic:


sw[3:0]
E
sw_IBUF[0]_inst
sw_IBUF[1]_inst
sw_IBUF[2]_inst
sw_IBUF[3]_inst
E_IBUF_inst
IBUF
I
O
dec_low
decoder3x8_0
dec_high
decoder3x8
E_IBUF
sw_IBUF[3:0]
led_OBUF[7:0]
led_OBUF[0]_inst
led_OBUF[1]_inst
led_OBUF[2]_inst
led_OBUF[3]_inst
led_OBUF[4]_inst
led_OBUF[5]_inst
led_OBUF[6]_inst
led_OBUF[7]_inst
led_OBUF[8]_inst
led_OBUF[9]_inst
led_OBUF[10]_inst
led_OBUF[11]_inst
led_OBUF[12]_inst
led_OBUF[13]_inst
led_OBUF[14]_inst
led_OBUF[15]_inst
OBUF
led[15:0]

Test bench:



Group video link
https://youtu.be/bMu9dKWkCvg

Contributions
Sean Go (50%) - Implementation and board demo
Ryan Tran (50%) - Verilog programming and test benching

Reflection

Completing this lab provided valuable hands-on experience in digital design and FPGA development. I gained a deeper understanding of how gate-level and behavioral coding approaches differ in terms of readability, flexibility, and hardware efficiency. Writing the self-checking testbench taught me how important automated verification is when ensuring design correctness, especially for larger projects. Additionally, using Vivado's synthesis and implementation tools gave me practical insights into resource utilization and timing analysis. Programming the Nexys A7-100T board and seeing the physical LEDs respond as expected was a very satisfying demonstration of how HDL code translates to working hardware. Overall, this lab strengthened both my coding and debugging skills and reinforced key concepts in digital systems design.

Conclusion
This lab demonstrated how to design, simulate, and implement a 4x16 decoder with an enable control on an FPGA. Both structural and behavioral coding approaches were explored, with functional verification achieved through a self-checking testbench and board testing. The final design met all functional and timing requirements with very low resource utilization.