ECEN 248 - Lab Report

Lab Number: 6

Lab Title:Introduction to Behavioral Verilog and Logic Synthesis

Student's Name:Paola Avila

Student's UIN: 731007033

Date:3/11/2025

Objectives:

In this week's Laboratory, we explore the different methodologies for describing digital circuits using Furlong HDL, including structural, data flow and behavioral modeling. Emphasis is shifted towards behavioral modeling, which provides greater scope for abstractions. Participants will gain practical experience in the design of digital circuits by replicating multiplexers, binary encoders and decoders using behavioral Verilog.In addition, the lab uses logic synthesis to convert Verilog's code into verilog-ready digital logic and instructs participants on how to synthesize parts for implementation. At the end of the course, participants test their designs by programming ZYBO's Z7- 10 board, improving their knowledge of real-world digital circuit implementation.

Design:

All of these deliverables were created with the instructions provided in the lab manual. I created a 1-bit 2:1 MUX in behavioral Verilog and simulated it with the given test bench code. Then I created a 4-bit 2:1 MUX in behavioral Verilog to simulate and test, and to create a 4-bit 4:1 MUX using the indicated code hints in the lab digital manual to create a 4-bit MUX and to simulate it. Next, I used the behavioral Verilog and the parameter values supplied in the lab manual to create a 2:4 binary decoder and a 4:2 binary encoder. Both of them were simulated. Then, I used the statement provided in the lab manual that was in the background section to create the priority encoder. I used behavioral Verilog to simulate the priority encoder, as instructed in the lab manual. The Experiment 3 involved setting up and programming the ZYBO Z7-10 board. Following the instructions in the lab manual, I created an analog, set up boards, simulated it, and programmed ZYBO Z7-10. Then I used the board-specific switches and buttons to conclude the programmed ZYBO Z7-10 to create the truth tables.

Design implementations schematics :

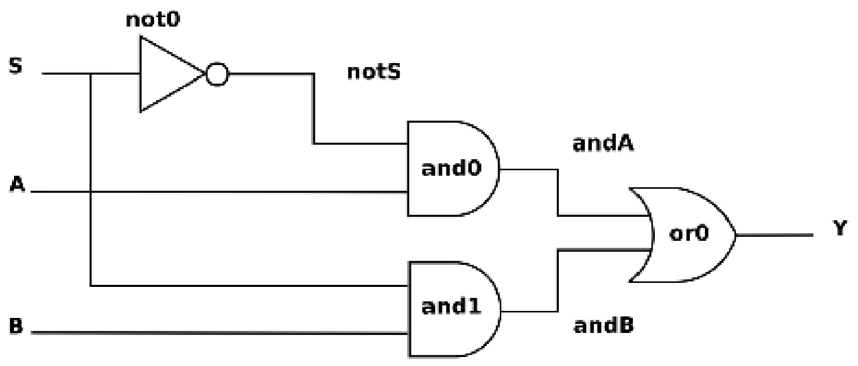


Figure 1: Gate-level Schematic of a 1-bit wide, 2:1 MUX

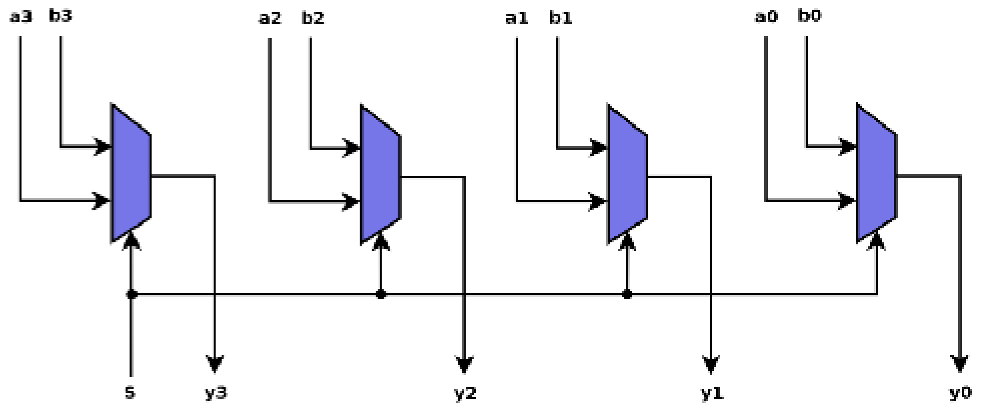


Figure 2: 4-bit wide, 2:1 MUX

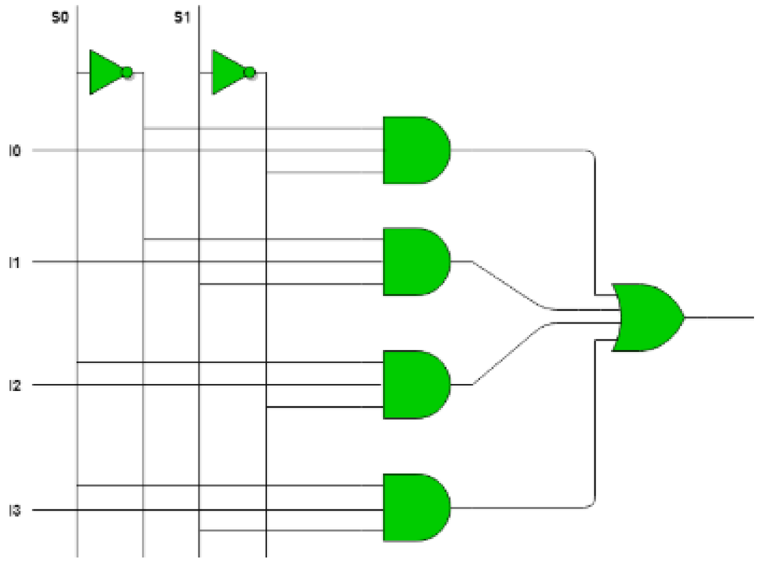


Figure 3 : 4:1 multiplexer

Table 1: 2:4 Binary Decoder Truth Table

En	w_1	w_0	y_3	y_2	y_1	y_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0
0	X	X	0	0	0	0

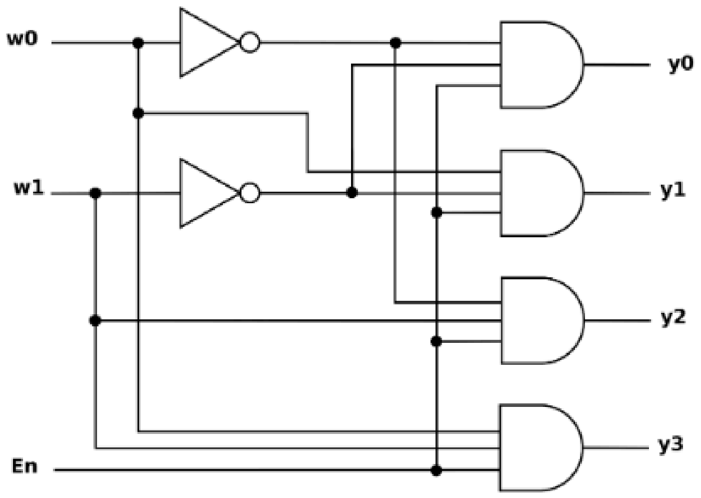


Figure 1: 2:4 Binary Decoder Gate-level Schematic

Table 2: 4:2 Binary Encoder Truth Table

w_3	w_2	w_1	w_0	y_1	y_0	$zero$
0	0	0	0	X	X	1
0	0	0	1	0	0	0
0	0	1	0	0	1	0
0	1	0	0	1	0	0
1	0	0	0	1	1	0

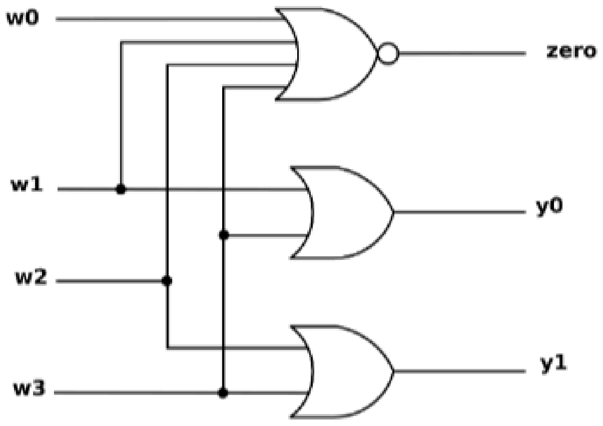


Figure 2: 4:2 Binary Encoder Gate-level Schematic

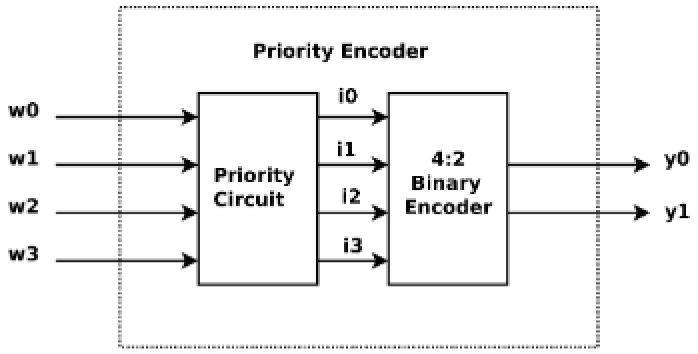
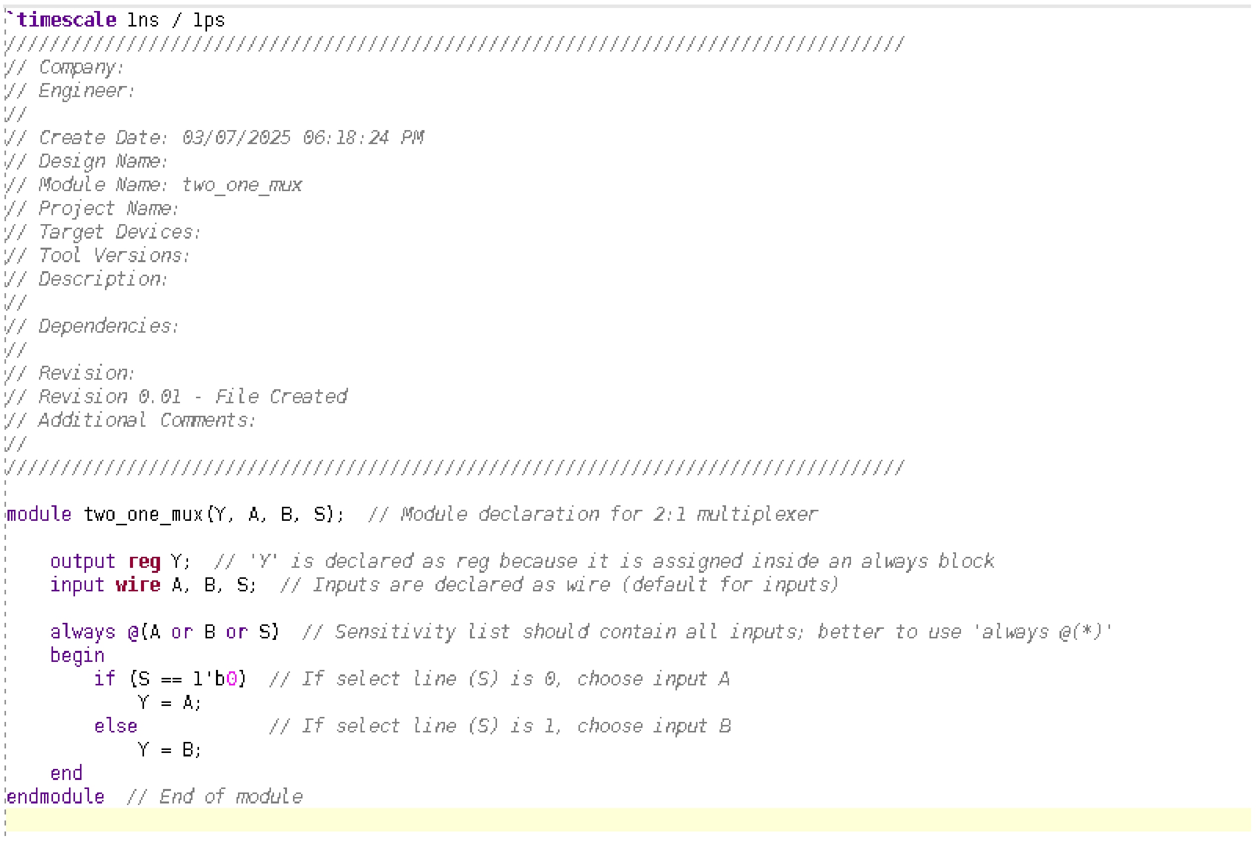


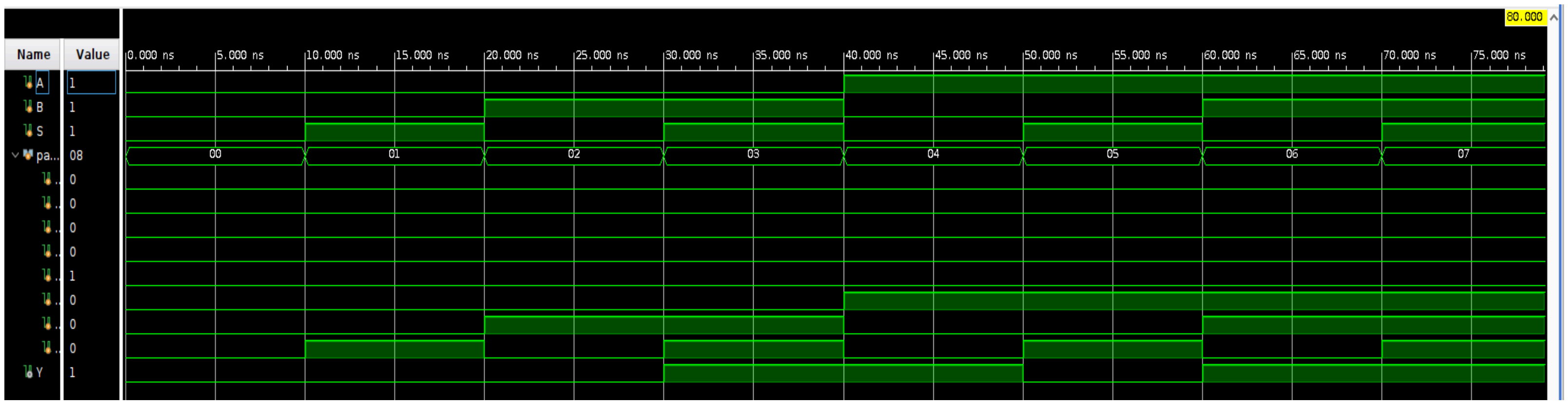
Figure 5: Priority Encoder

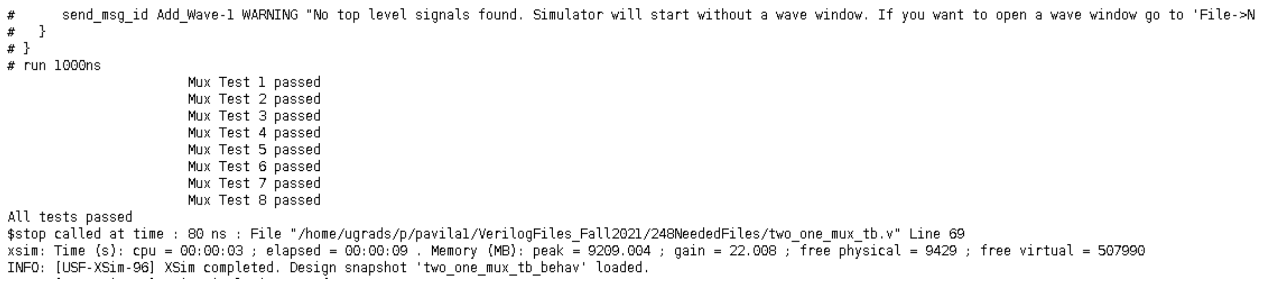
Results

Two_One_Mux :



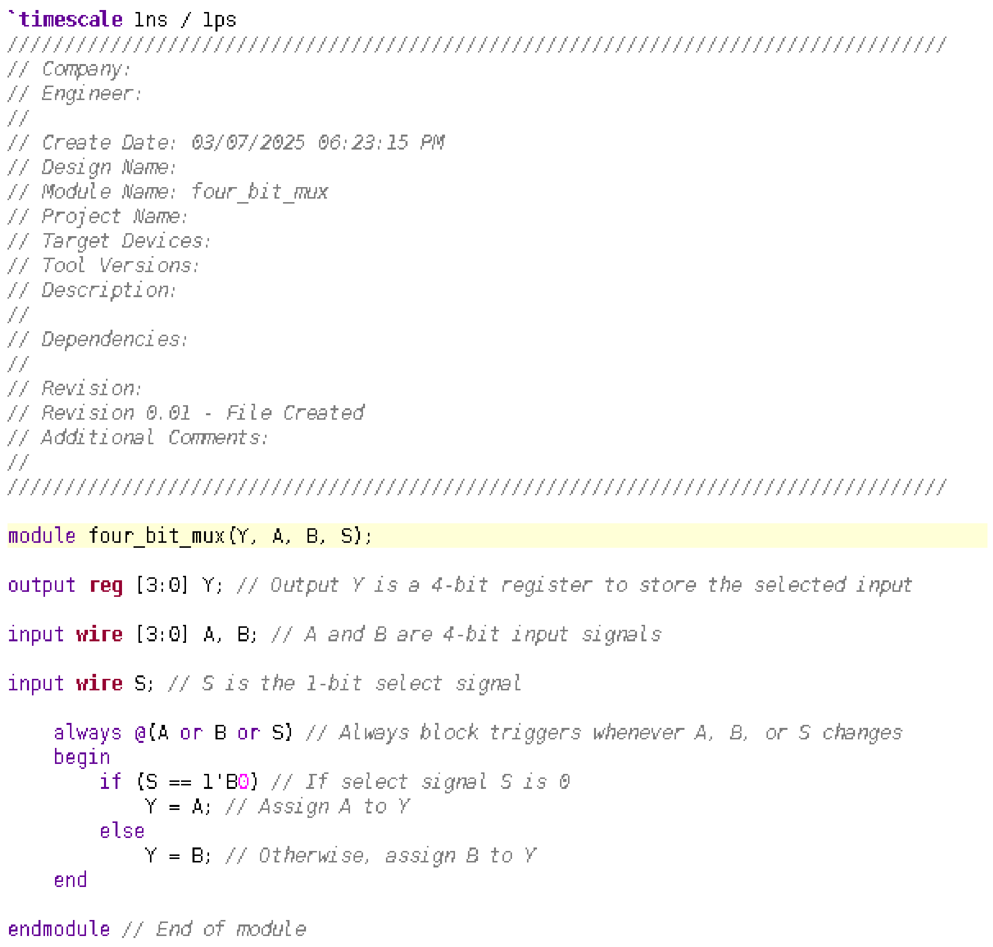
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/07/2025 06:18:24 PM
// Design Name:
// Module Name: two_one_mux
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module two_one_mux(Y, A, B, S);  // Module declaration for 2:1 multiplexer

    output reg Y;  // 'Y' is declared as reg because it is assigned inside an always block
    input wire A, B, S;  // Inputs are declared as wire (default for inputs)

    always @(A or B or S)  // Sensitivity list should contain all inputs; better to use 'always @(*)'
    begin
        if (S == 1'b0)  // If select line (S) is 0, choose input A
            Y = A;
        else            // If select line (S) is 1, choose input B
            Y = B;
    end
endmodule  // End of module
```



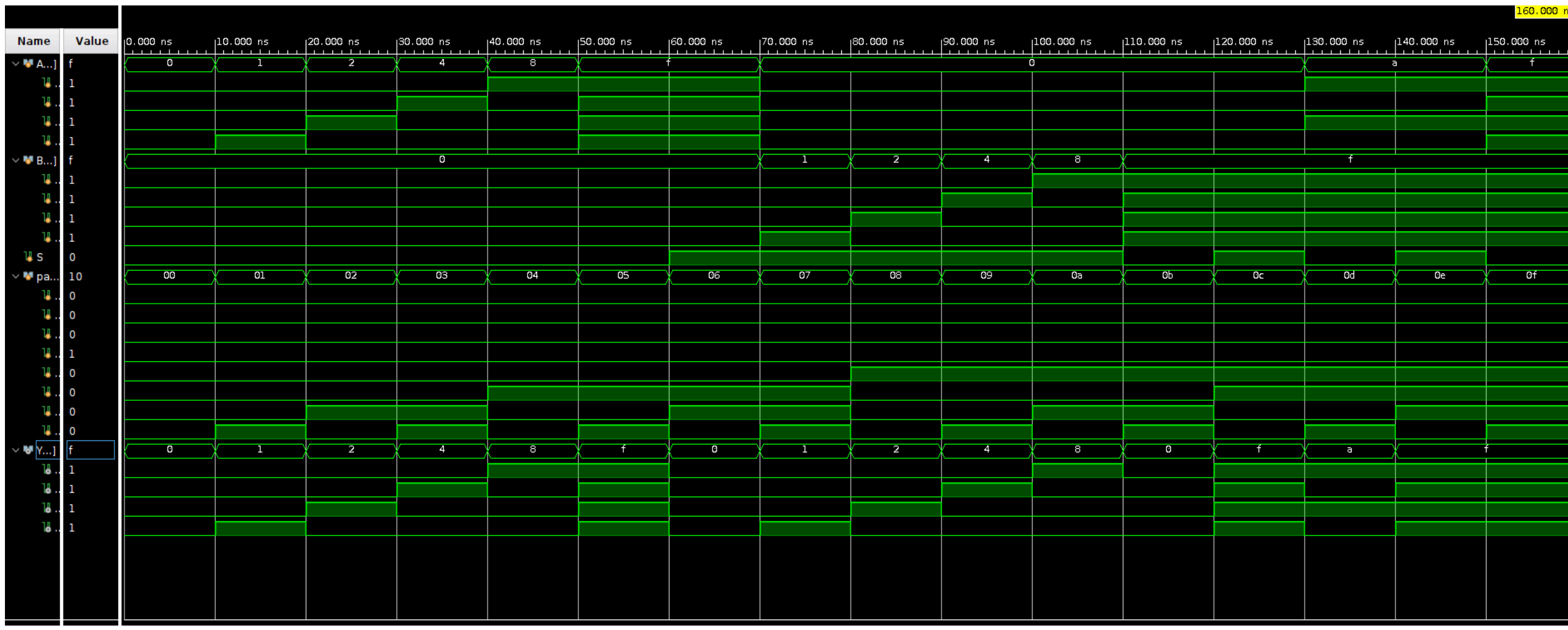


```
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->N
#   }
# }
# run 1000ns
                       Mux Test 1 passed
                       Mux Test 2 passed
                       Mux Test 3 passed
                       Mux Test 4 passed
                       Mux Test 5 passed
                       Mux Test 6 passed
                       Mux Test 7 passed
                       Mux Test 8 passed
All tests passed
$stop called at time : 80 ns : File "/home/ugrads/p/pavila1/VerilogFiles_Fall2021/248NeededFiles/two_one_mux_tb.v" Line 69
xsim: Time (s): cpu = 00:00:03 ; elapsed = 00:00:09 . Memory (MB): peak = 9209.004 ; gain = 22.008 ; free physical = 9429 ; free virtual = 507990
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_one_mux_tb_behav' loaded.
```

Four_Bit_Mux:

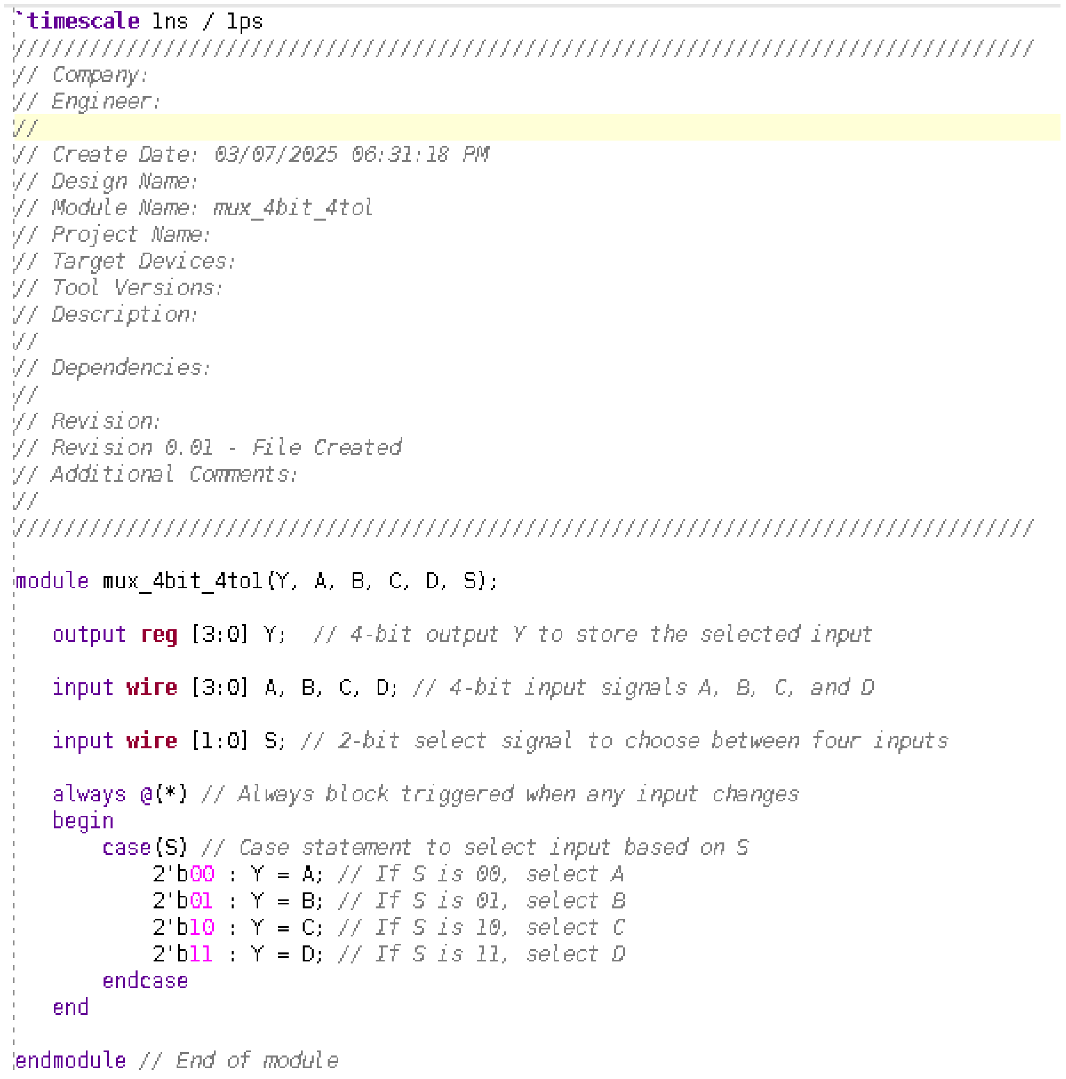


```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/07/2025 06:23:15 PM
// Design Name:
// Module Name: four_bit_mux
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module four_bit_mux(Y, A, B, S);

output reg [3:0] Y; // Output Y is a 4-bit register to store the selected input

input wire [3:0] A, B; // A and B are 4-bit input signals

input wire S; // S is the 1-bit select signal

    always @(A or B or S) // Always block triggers whenever A, B, or S changes
    begin
        if (S == 1'B0) // If select signal S is 0
            Y = A; // Assign A to Y
        else
            Y = B; // Otherwise, assign B to Y
    end

endmodule // End of module
```

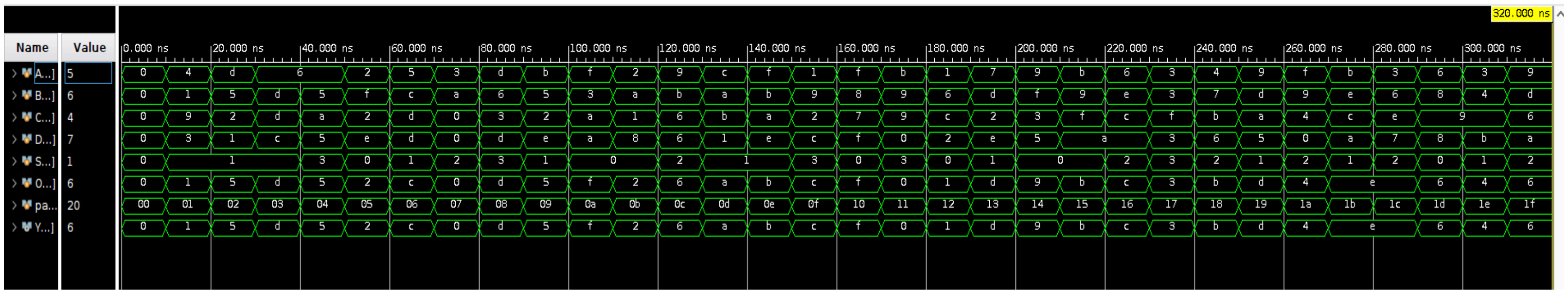


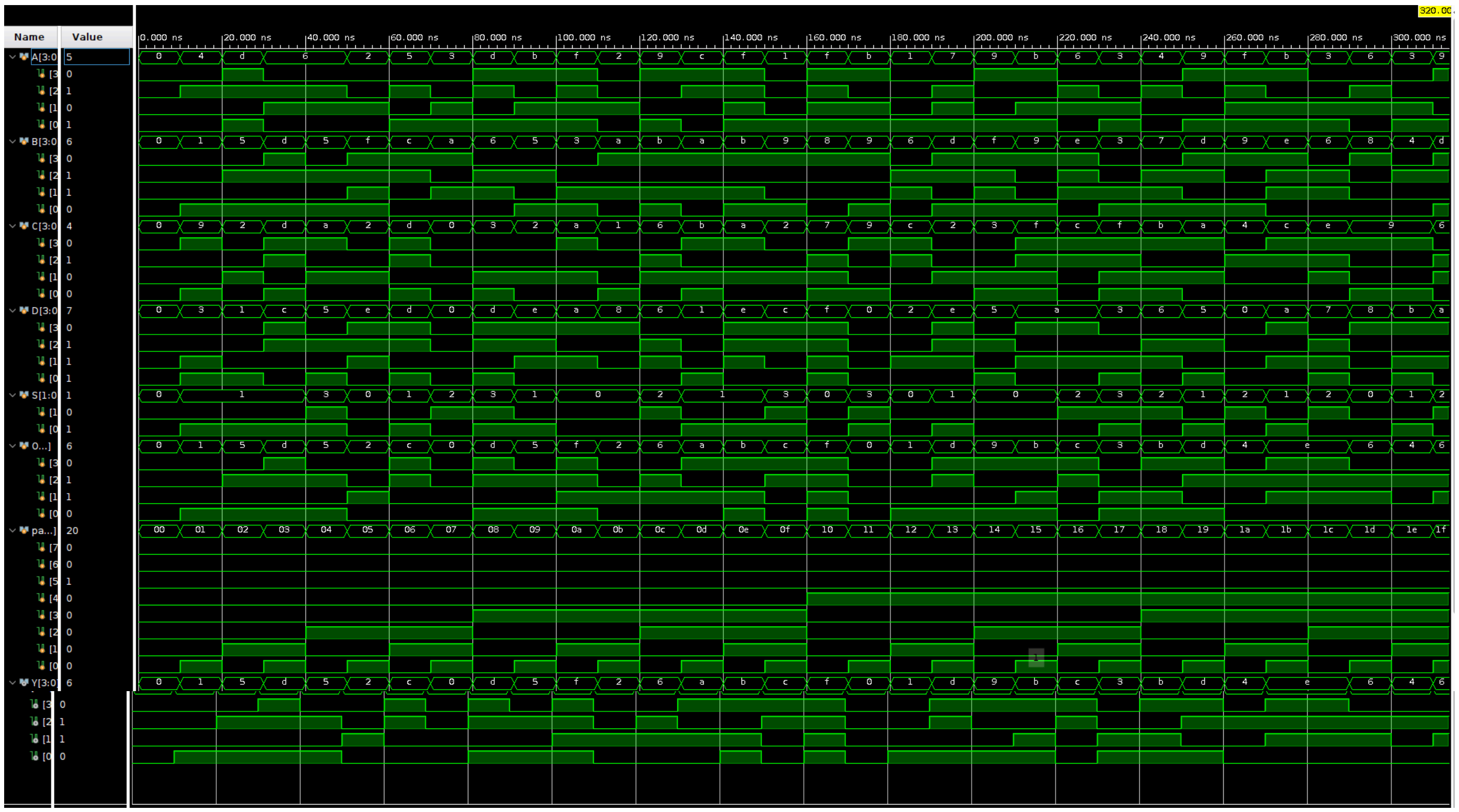
```
t }
t run 1000ns
                 4-bit Mux Test 1 passed
                 4-bit Mux Test 2 passed
                 4-bit Mux Test 3 passed
                 4-bit Mux Test 4 passed
                 4-bit Mux Test 5 passed
                 4-bit Mux Test 6 passed
                 4-bit Mux Test 7 passed
                 4-bit Mux Test 8 passed
                 4-bit Mux Test 9 passed
                4-bit Mux Test 10 passed
                4-bit Mux Test 11 passed
                4-bit Mux Test 12 passed
                4-bit Mux Test 13 passed
                4-bit Mux Test 14 passed
                4-bit Mux Test 15 passed
                4-bit Mux Test 16 passed
All tests passed
$stop called at time : 160 ns : File "/home/ugrads/p/pavila1/VerilogFiles_Fall2021/248NeededFiles/four_bit_mux_tb.v" Line 76
xsim: Time (s): cpu = 00:00:10 ; elapsed = 00:00:13 . Memory (MB): peak = 9096.633 ; gain = 100.758 ; free physical = 10783 ; free virtual = 509338
```

Mux_4bit_4tol :



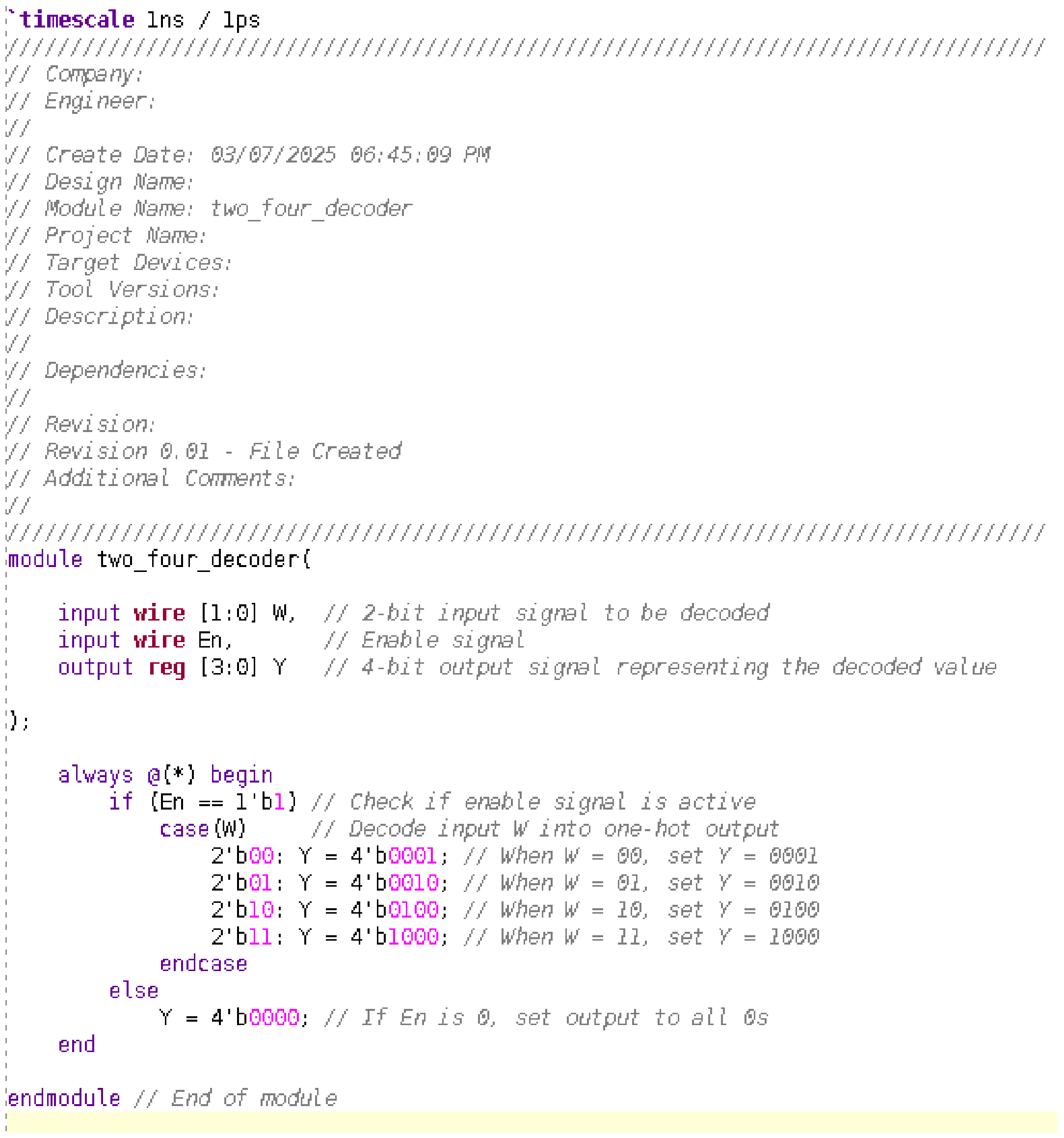
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/07/2025 06:31:18 PM
// Design Name:
// Module Name: mux_4bit_4tol
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module mux_4bit_4tol(Y, A, B, C, D, S);

   output reg [3:0] Y;  // 4-bit output Y to store the selected input

   input wire [3:0] A, B, C, D; // 4-bit input signals A, B, C, and D

   input wire [1:0] S; // 2-bit select signal to choose between four inputs

   always @(*) // Always block triggered when any input changes
   begin
       case(S) // Case statement to select input based on S
           2'b00 : Y = A; // If S is 00, select A
           2'b01 : Y = B; // If S is 01, select B
           2'b10 : Y = C; // If S is 10, select C
           2'b11 : Y = D; // If S is 11, select D
       endcase
   end

endmodule // End of module
```



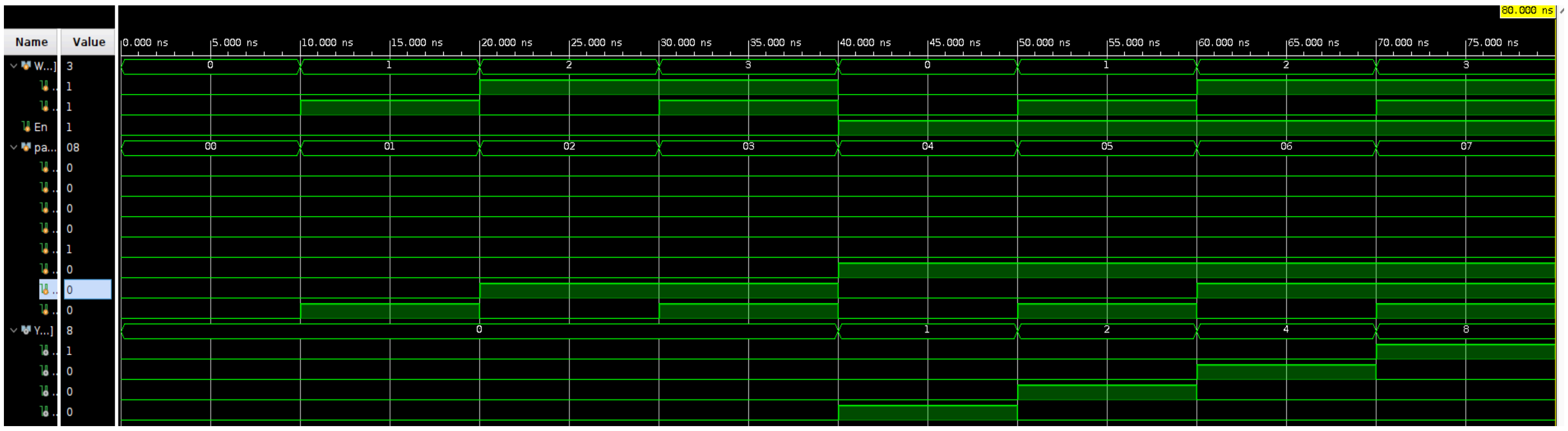


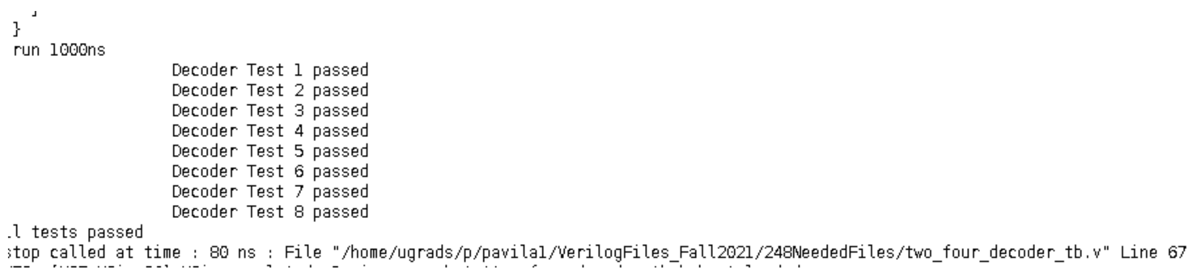
```
# ;
# }
# run 1000ns
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
                4-bit 4:1 MUX Test passed
All tests passed
$stop called at time : 320 ns : File "/home/ugrads/p/pavilal/VerilogFiles_Fall2021/248NeededFiles/mux_4bit_4to1_tb.v" Line 82
xsim: Time (s): cpu = 00:00:04 ; elapsed = 00:00:06 . Memory (MB): peak = 9259.828 ; gain = 103.855 ; free physical = 12375 ; free virtual = 51093
INFO: [USF-XSim-96] XSim completed. Design snapshot 'mux_4bit_4to1_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Two_Four_Decoder :



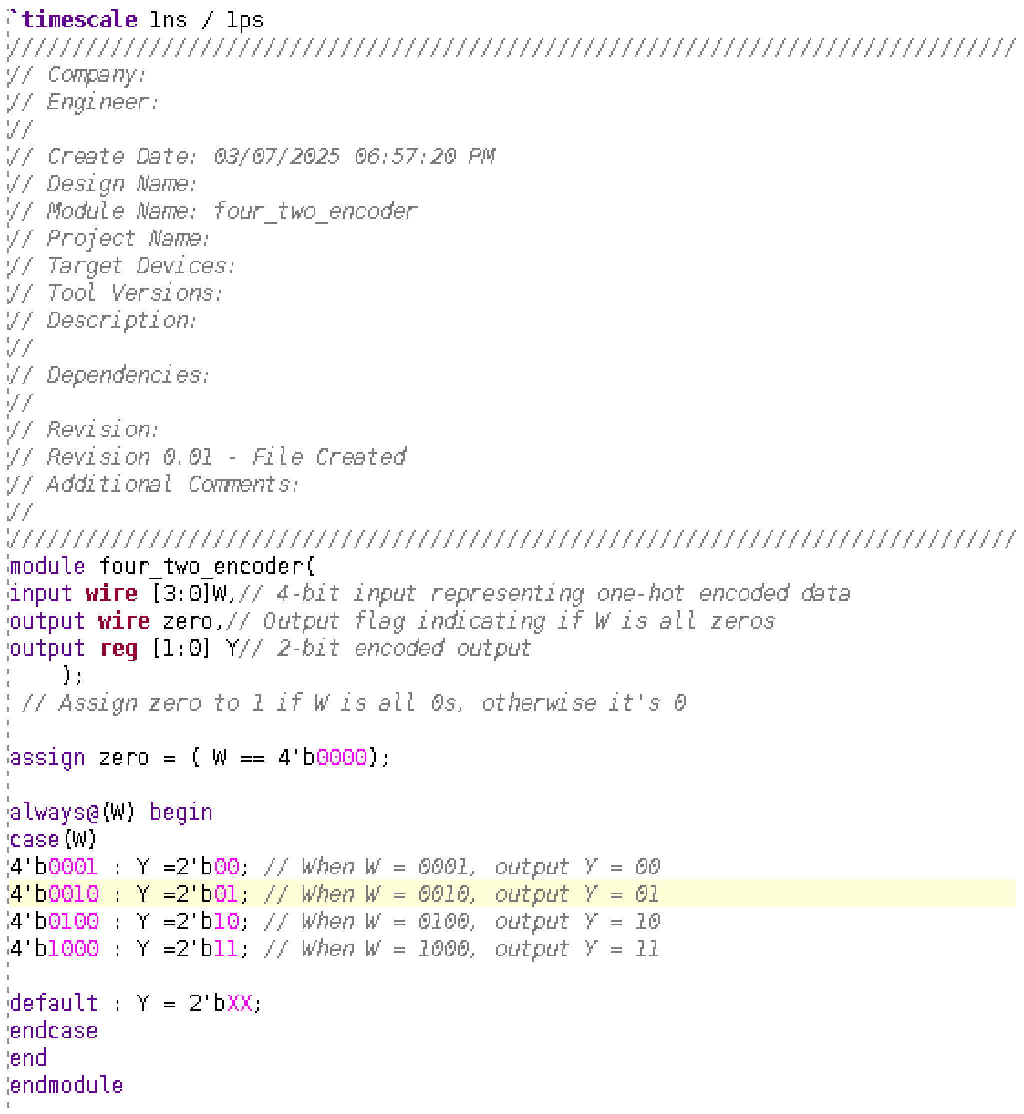
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/07/2025 06:45:09 PM
// Design Name:
// Module Name: two_four_decoder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module two_four_decoder(

    input wire [1:0] W,  // 2-bit input signal to be decoded
    input wire En,       // Enable signal
    output reg [3:0] Y   // 4-bit output signal representing the decoded value

);

    always @(*) begin
        if (En == 1'b1) // Check if enable signal is active
            case(W)     // Decode input W into one-hot output
                2'b00: Y = 4'b0001; // When W = 00, set Y = 0001
                2'b01: Y = 4'b0010; // When W = 01, set Y = 0010
                2'b10: Y = 4'b0100; // When W = 10, set Y = 0100
                2'b11: Y = 4'b1000; // When W = 11, set Y = 1000
            endcase
        else
            Y = 4'b0000; // If En is 0, set output to all 0s
    end

endmodule // End of module
```



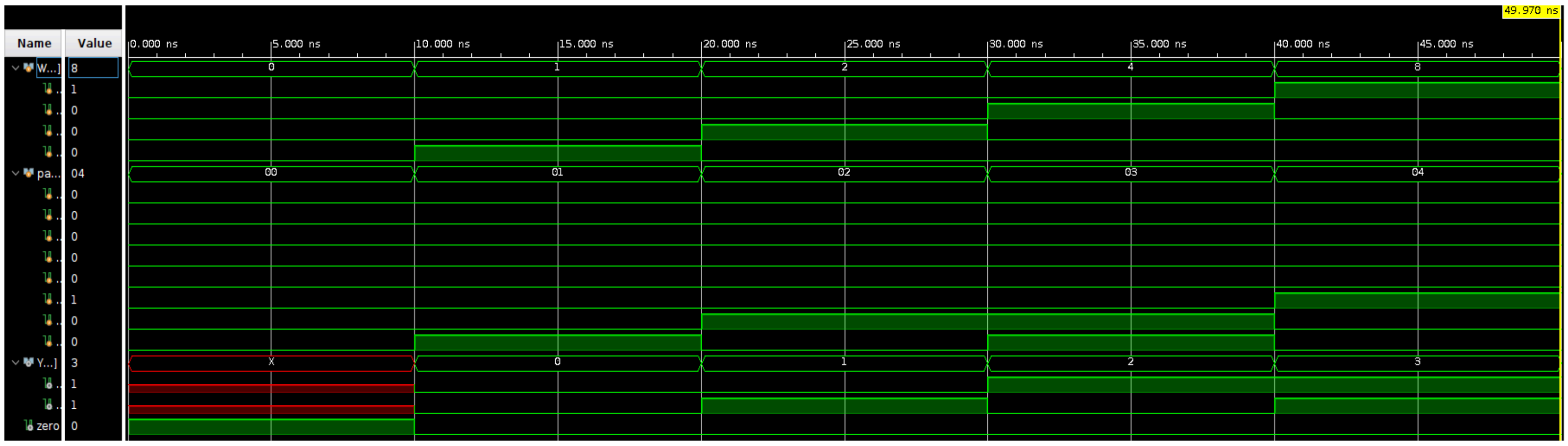


```
}
run 1000ns
                Decoder Test 1 passed
                Decoder Test 2 passed
                Decoder Test 3 passed
                Decoder Test 4 passed
                Decoder Test 5 passed
                Decoder Test 6 passed
                Decoder Test 7 passed
                Decoder Test 8 passed
.l tests passed
;top called at time : 80 ns : File "/home/ugrads/p/pavila1/VerilogFiles_Fall2021/248NeededFiles/two_four_decoder_tb.v" Line 67
```

Four_Two_Encoder :

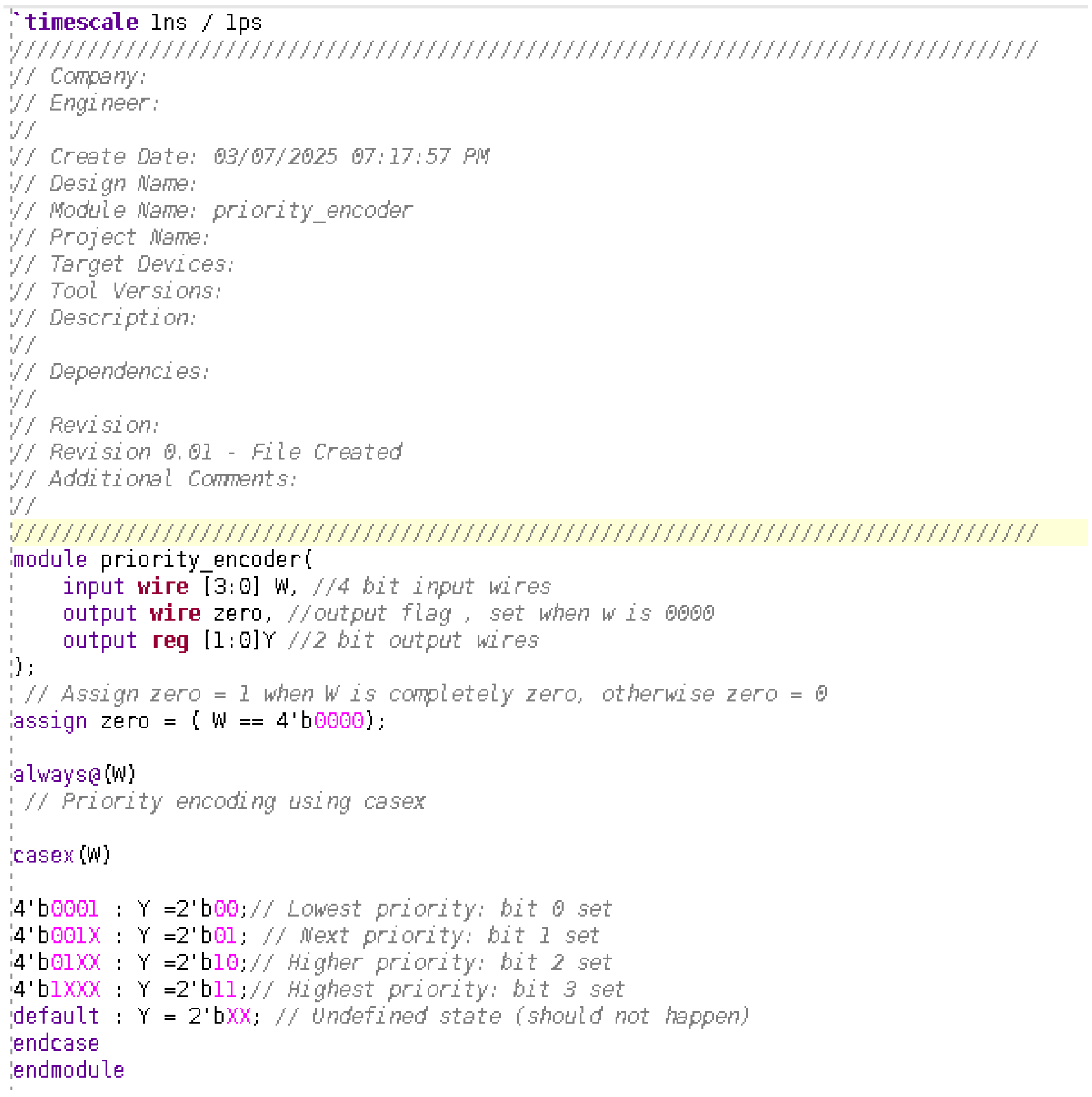


```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/07/2025 06:57:20 PM
// Design Name:
// Module Name: four_two_encoder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module four_two_encoder(
input wire [3:0]W,// 4-bit input representing one-hot encoded data
output wire zero,// Output flag indicating if W is all zeros
output reg [1:0] Y// 2-bit encoded output
    );
 // Assign zero to 1 if W is all 0s, otherwise it's 0

assign zero = ( W == 4'b0000);

always@(W) begin
case(W)
4'b0001 : Y =2'b00; // When W = 0001, output Y = 00
4'b0010 : Y =2'b01; // When W = 0010, output Y = 01
4'b0100 : Y =2'b10; // When W = 0100, output Y = 10
4'b1000 : Y =2'b11; // When W = 1000, output Y = 11

default : Y = 2'bXX;
endcase
end
endmodule
```

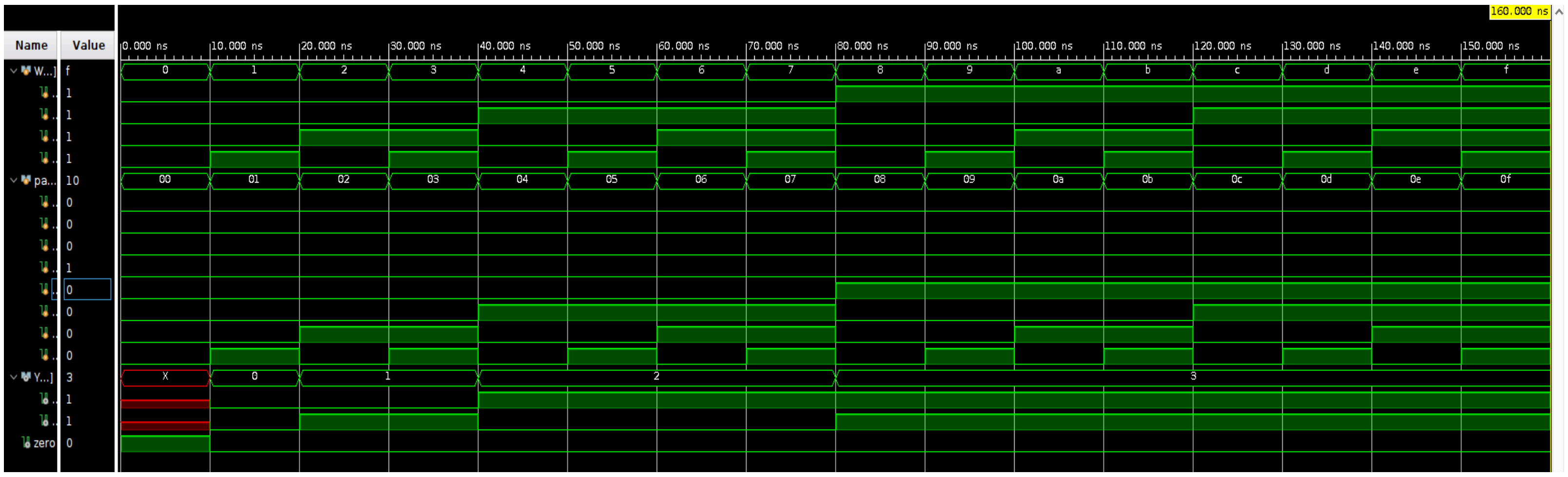


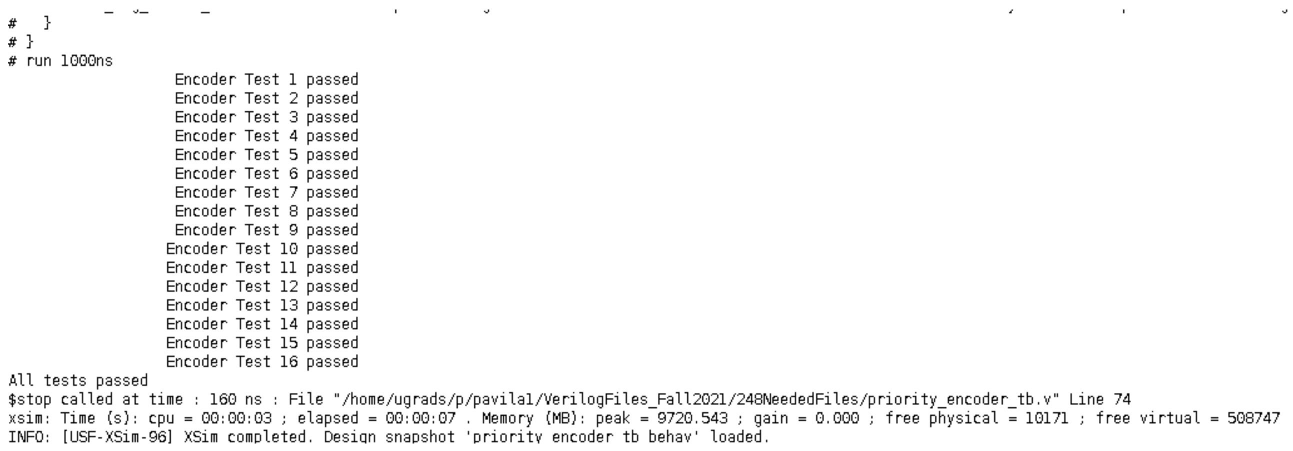
```
 }
 run 1000ns
                   Encoder Test 1 passed
                   Encoder Test 2 passed
                   Encoder Test 3 passed
                   Encoder Test 4 passed
                   Encoder Test 5 passed
ll tests passed
stop called at time : 50 ns : File "/home/ugrads/p/pavila1/VerilogFiles_Fall2021/248NeededFiles/four_two_encoder_tb.v" Line 63
sim: Time (s): cpu = 00:00:03 ; elapsed = 00:00:05 . Memory (MB): peak = 9592.844 ; gain = 93.652 ; free physical = 10445 ; free virtual = 509012
```

Priority_encoder:



```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: 
// Engineer: 
// 
// Create Date: 03/07/2025 07:17:57 PM
// Design Name: 
// Module Name: priority_encoder
// Project Name: 
// Target Devices: 
// Tool Versions: 
// Description: 
// 
// Dependencies: 
// 
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
// 
//////////////////////////////////////////////////////////////////////////////////
module priority_encoder(
    input wire [3:0] W, //4 bit input wires
    output wire zero, //output flag , set when w is 0000
    output reg [1:0]Y //2 bit output wires
);
 // Assign zero = 1 when W is completely zero, otherwise zero = 0
assign zero = ( W == 4'b0000);

always@(W)
 // Priority encoding using casex

casex(W)

4'b0001 : Y =2'b00;// Lowest priority: bit 0 set
4'b001X : Y =2'b01; // Next priority: bit 1 set
4'b01XX : Y =2'b10;// Higher priority: bit 2 set
4'b1XXX : Y =2'b11;// Highest priority: bit 3 set
default : Y = 2'bXX; // Undefined state (should not happen)
endcase
endmodule
```





```
#   }
# }
# run 1000ns
                   Encoder Test 1 passed
                   Encoder Test 2 passed
                   Encoder Test 3 passed
                   Encoder Test 4 passed
                   Encoder Test 5 passed
                   Encoder Test 6 passed
                   Encoder Test 7 passed
                   Encoder Test 8 passed
                   Encoder Test 9 passed
                  Encoder Test 10 passed
                  Encoder Test 11 passed
                  Encoder Test 12 passed
                  Encoder Test 13 passed
                  Encoder Test 14 passed
                  Encoder Test 15 passed
                  Encoder Test 16 passed
All tests passed
$stop called at time : 160 ns : File "/home/ugrads/p/pavilal/VerilogFiles_Fall2021/248NeededFiles/priority_encoder_tb.v" Line 74
xsim: Time (s): cpu = 00:00:03 ; elapsed = 00:00:07 . Memory (MB): peak = 9720.543 ; gain = 0.000 ; free physical = 10171 ; free virtual = 508747
INFO: [USF-XSim-96] XSim completed. Design snapshot 'priority_encoder_tb_behav' loaded.
```

Conclusion:

The goal of this lab is to give students their first exposure to behavioral modeling in Verilog HDL which, in contrast to structural and dataflow modeling, provides a level of HDL abstraction where the digital circuit's behavior can be expressed at a higher level of abstraction using algorithmic constructs. Along the way, students are introduced to logic synthesis and to the concept of design equivalence where an HDL description of a digital circuit can be used to generate an implementation of that circuit in physical digital logic. Additionally, students have their first exposure to hardware testing that helps them to validate designs implemented on real hardware.

Post-lab Deliverables:

● Verilog provides three different coding styles for designing digital circuits: Behavioral, Structural, and Dataflow. Behavioral Verilog handles complex designs well, making it easy to simulate and test, and easy to read and change. It may not be as efficient when dealing with lower level designs. Structural Verilog can be used to optimize timing and power in low level designs, but it requires more work to read and change because it has a lot of control over your design. Dataflow specializes in timing and power optimization, but it might not be the best choice for both low-level and complex designs. Ultimately, the coding style you choose depends on the complexity of your design and the requirements and constraints you have to meet. If you're looking for the best coding style for your design, you'll want to go with the behavioral one, structural one, dataflow one, or dataflow one.

● A breadboard is a simple, inexpensive tool for rapid prototyping of small circuits with little technical know-how. It facilitates the placement of components and wire connections for rapid prototyping, but is prone to signal interference and noise, which limits its usefulness for large-scale designs. An FPGA, on the other hand, is a very versatile platform for the implementation of digital circuits using hardware description languages (HLCs). It is more sophisticated and demanding in design and programming skills, but excels in dealing with complex designs and hardware acceleration. The decision between breadboard and FPGA depends on the requirements and constraints of the project. A breadboard is sufficient for small-scale prototypes or basic circuits, while a FPGA is better for larger, more complex, performance-oriented designs, but requires more expertise and capital investment.

Student Feedback :

My overall feedback is that the lab is quite enjoyable yet I still feel that Verilog should be explored more in these labs but overall everything made sense and I enjoyed the process.