ECEN 248: Introduction to Digital Design
Department of Electrical and Computer Engineering
Texas A&M University
Lab Report

Laboratory Exercise #5

Introduction to Logic Simulation and Verilog

ECEN-248-509

Joaquin Salas

731000141

10-24-2022

TA: Sri Hari Pada Chandanam Kodi

Objectives:

In the previous lab, we experimented with two ways to describe digital circuits in Verilog HDL: structural and dataflow modeling. In this lab we will be introduced to a higher level of abstraction in Verilog HDL, this is called behavioral modeling. For the first lab experiment, we will recreate the previous lab's multiplexers using behavioral Verilog. For the second part of this experiment, we will use behavioral Verilog to describe binary encoders and decoders as mentioned in the lecture. This lab will also introduce logic synthesis, the process of translating HDL code into implementable digital logic. Experiment part 3 will show us the process of synthesizing decoders and encoders simulated in previous parts. As well as programming the ZYBO Z7-10 board on the workbench to test the components described in Verilog.

Design:

I will be expected to complete procedures introduced in the previous lab for the following lab experiments, like creating a new Vivado project and adding / loading testbench files to run simulations.

Experiment Part 1:

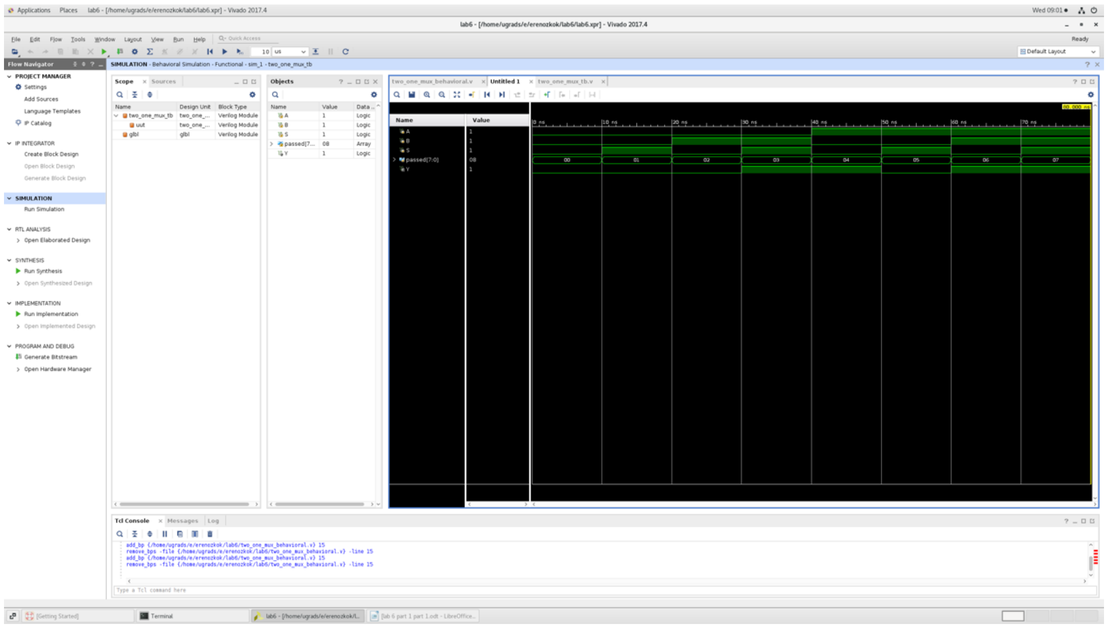
For part 1, we will be describing hardware in behavioral Verilog. To do this we will begin by simulating the multiplexer which was provided in the Lab 6 Manual. We will then extend that multiplexer to a 4-bit wide, 2:1 multiplexer, and to a 4-bit wide, 4:1 multiplexer.

1. We began by typing the behavioral code from the manual's background section, simulated the test bench, and ensured the UUT passed all the tests. The screenshot of the simulation waveform and console output will be shown below:

Code : *2 to 1 Multiplexer*

```
`timescale 1ns / 1ps
`default_nettype none

module two_one_mux(Y, A, B, S);
      output reg Y;  //declare output of type seg since it will be
                               // modified in an always block!
      input wire A, B, S; //declare input of type wire
                          // wires can drive regs in behavioral statements
        always@(A or B or S) // always block which triggers whenever A, B,
or S changes
             begin //block constructs together

                  if( S == 1'b0) //double equals are used for comparisons
                          Y = A; // drive Y with A
                  else
                          Y = B; //instead drive Y with B
             end
endmodule
```

Result: *2 to 1 Multiplexer*



lab6 - [/home/ugrads/e/erenozkok/lab6/lab6.xpr] - Vivado 2017.4

SIMULATION - Behavioral Simulation - Functional - sim_1 - two_one_mux_tb

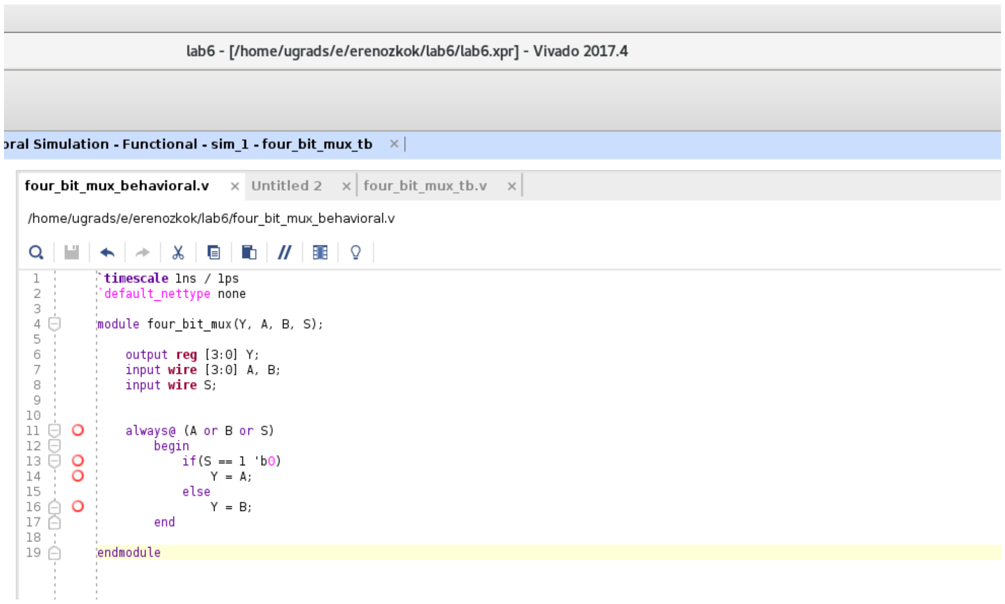
Name	Design Unit	Block Type
two_one_mux_tb	two_one_...	Verilog Module
uut	two_one_...	Verilog Module
glbl	glbl	Verilog Module

Name	Value	Data ...
A	1	Logic
B	1	Logic
S	1	Logic
passed[7...	08	Array
Y	1	Logic

```
add_bp {/home/ugrads/e/erenozkok/lab6/two_one_mux_behavioral.v} 15
remove_bps -file {/home/ugrads/e/erenozkok/lab6/two_one_mux_behavioral.v} -line 15
add_bp {/home/ugrads/e/erenozkok/lab6/two_one_mux_behavioral.v} 15
remove_bps -file {/home/ugrads/e/erenozkok/lab6/two_one_mux_behavioral.v} -line 15
```

2. Describe a 4-bit, 2:1 multiplexer in behavioral Verilog and simulate the operation.

Code: *four_bit_mux_behavioral.v*



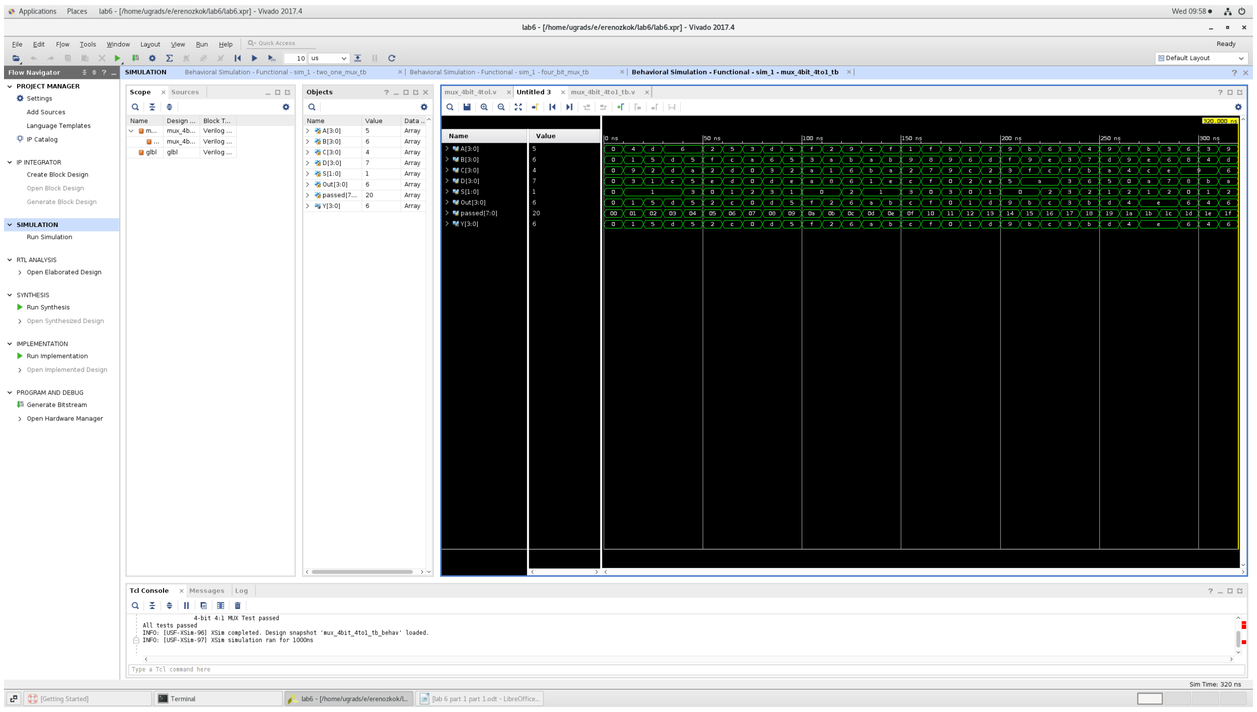
lab6 - [/home/ugrads/e/erenozkok/lab6/lab6.xpr] - Vivado 2017.4

oral Simulation - Functional - sim_1 - four_bit_mux_tb

/home/ugrads/e/erenozkok/lab6/four_bit_mux_behavioral.v

```verilog
`timescale 1ns / 1ps
`default_nettype none

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

    output reg [3:0] Y;
    input wire [3:0] A, B;
    input wire S;


    always@ (A or B or S)
        begin
            if(S == 1 'b0)
                Y = A;
            else
                Y = B;
        end

endmodule
```

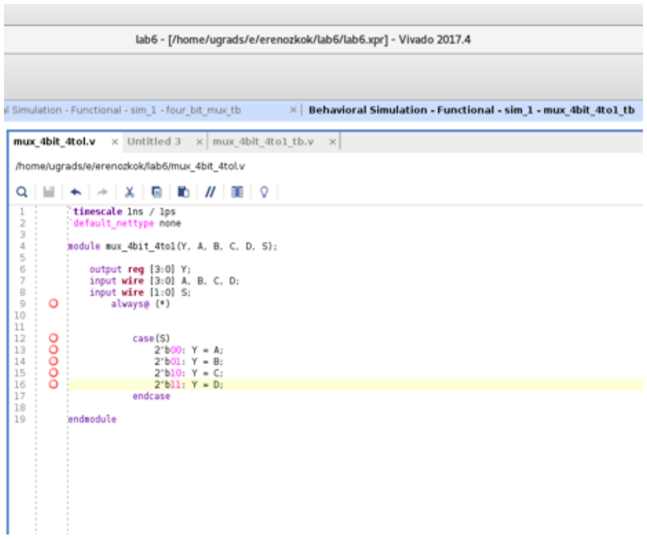
Result: *four_bit_mux_behavioral.v*



```
            4-bit 4:1 MUX Test passed
All tests passed
INFO: [USF-XSim-96] XSim completed. Design snapshot 'mux_4bit_4tol_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

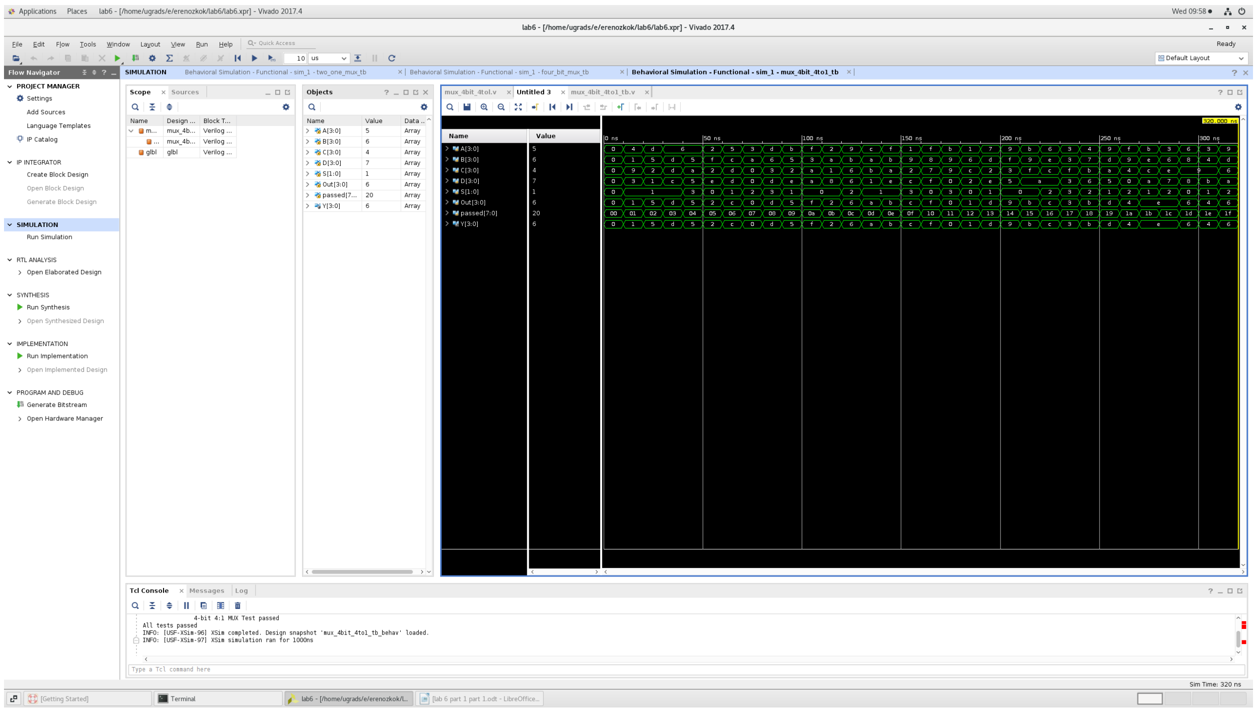
3. Describe a 4-bit, 4:1 multiplexer in behavioral Verilog and simulate its operation

Code : *4-bit, 4:1 multiplexer*



```
`timescale 1ns / 1ps
`default_nettype none

module mux_4bit_4to1(Y, A, B, C, D, S);

    output reg [3:0] Y;
    input wire [3:0] A, B, C, D;
    input wire [1:0] S;
        always@ (*)


            case(S)
                2'b00: Y = A;
                2'b01: Y = B;
                2'b10: Y = C;
                2'b11: Y = D;
            endcase

endmodule
```

Result: *4-bit, 4:1 multiplexer*



```
        4-bit 4:1 MUX Test passed
All tests passed
INFO: [USF-XSim-96] XSim completed. Design snapshot 'mux_4bit_4to1_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Experiment Part 2:

For experiment part 2, we will be exposed to binary encoders and decoders and reinforce the behavioral Verilog concept.

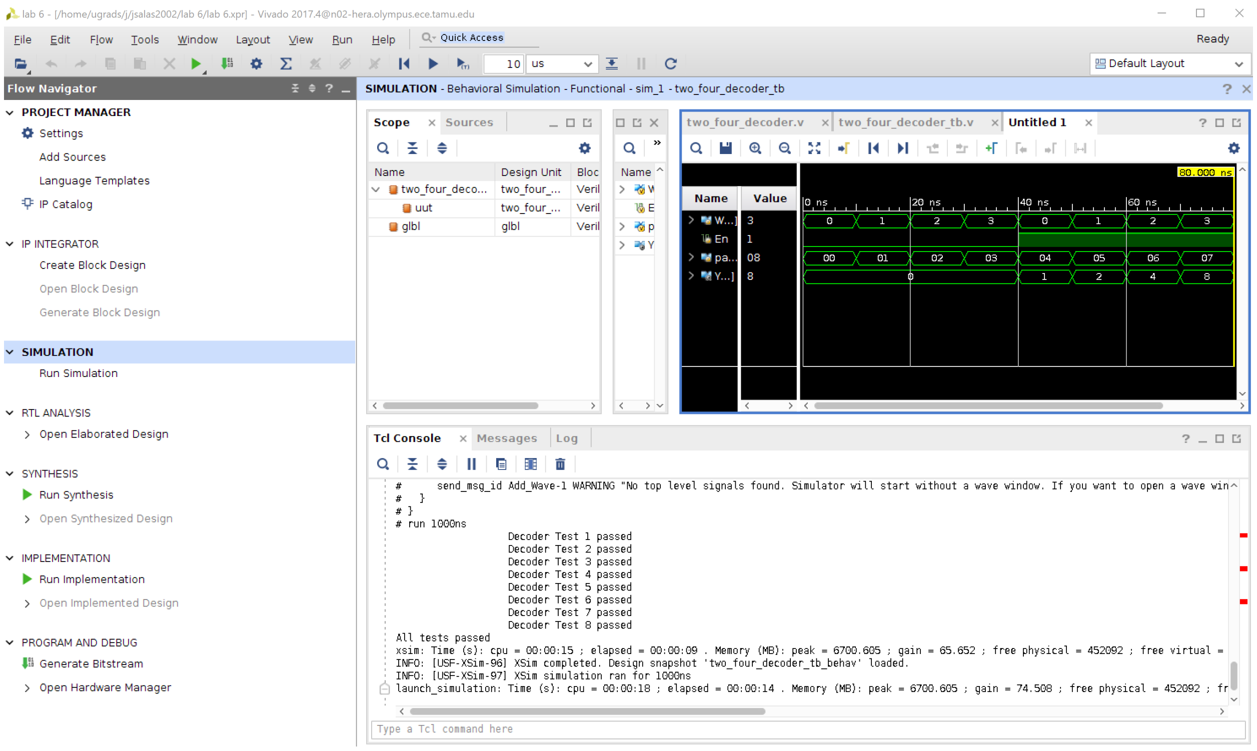
1. Use behavioral Verilog to describe 2:4 binary decoder and 4:2 binary encoder:

Code: *two_four_decoder.v*

```
`timescale 1ns / 1ps
`default_nettype none
/*This module describes a 2:4 decoder using behavioral constructs in
Verilog HDL.   */

/*module interface for the 2:4 decoder*/
module two_four_decoder(
    input wire [1:0] W,
    input wire En,
    output reg [3:0] Y
    //Y should be a 4-bit output of type reg
    );

    always@(En or W) //something is missing here... trigger when En or W
changes
       begin //not necessary because if is single clause but looks better
         if(En == 1'b1) //can put case within if clause
```

```
            case(W) //selection based on W
                2'b00: Y = 4'b0001;  //4'b signifies a 4-bit binary value
                2'b01: Y = 4'b0010;
                2'b10: Y = 4'b0100;
                   //fill in code here...
                2'b11: Y = 4'b1000; //light up y[3]
            endcase //designates the end of a case statement
        else //if not Enable
             Y = 4'b0000; //disable all outputs
      end

endmodule
```

Results: two_four_decoder



lab 6 - [/home/ugrads/j/jsalas2002/lab 6/lab 6.xpr] - Vivado 2017.4@n02-hera.olympus.ece.tamu.edu
File Edit Flow Tools Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
SIMULATION - Behavioral Simulation - Functional - sim_1 - two_four_decoder_tb
Scope
Sources
two_four_decoder.v
two_four_decoder_tb.v
Untitled 1
Tcl Console
Messages
Log

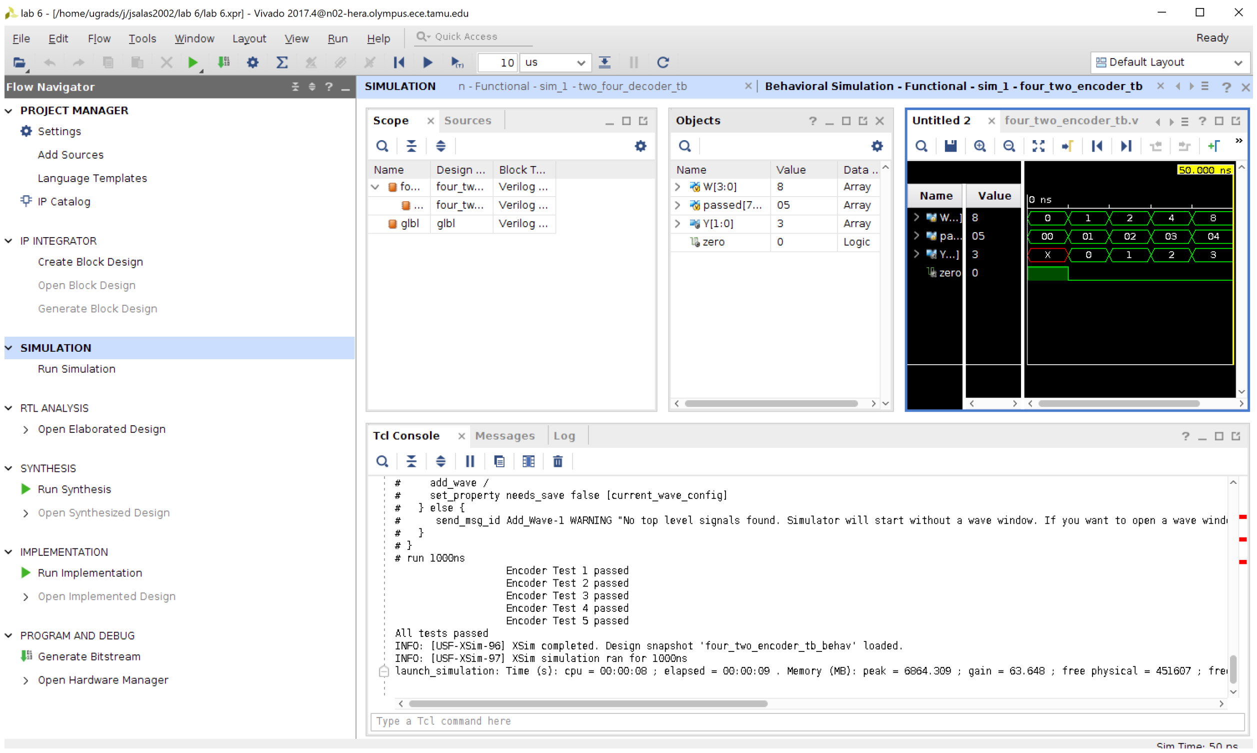
```
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave win
#   }
# }
# run 1000ns
                 Decoder Test 1 passed
                 Decoder Test 2 passed
                 Decoder Test 3 passed
                 Decoder Test 4 passed
                 Decoder Test 5 passed
                 Decoder Test 6 passed
                 Decoder Test 7 passed
                 Decoder Test 8 passed
All tests passed
xsim: Time (s): cpu = 00:00:15 ; elapsed = 00:00:09 . Memory (MB): peak = 6700.605 ; gain = 65.652 ; free physical = 452092 ; free virtual =
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_four_decoder_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:18 ; elapsed = 00:00:14 . Memory (MB): peak = 6700.605 ; gain = 74.508 ; free physical = 452092 ; fr
```

Code: *four_two_encoder.v*

```
`timescale 1ns / 1ps
`default_nettype none

module four_two_encoder(
    input wire [3:0] W,
    output wire zero,
    output reg [1:0] Y
    );
    assign zero = (W == 4'b0000);

    always@(W)
       begin
```

```
            case(W)
                4'b0001 : Y = 2'b00;
                4'b0010 : Y = 2'b01;
                4'b0100 : Y = 2'b10;
                4'b1000 : Y = 2'b11;
                default : Y = 2'bXX;
            endcase
        end
endmodule
```

Results: four_two_encoder



```
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      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 windo
#   }
# }
# run 1000ns
                   Encoder Test 1 passed
                   Encoder Test 2 passed
                   Encoder Test 3 passed
                   Encoder Test 4 passed
                   Encoder Test 5 passed
All tests passed
INFO: [USF-XSim-96] XSim completed. Design snapshot 'four_two_encoder_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:08 ; elapsed = 00:00:09 . Memory (MB): peak = 6864.309 ; gain = 63.648 ; free physical = 451607 ; fre
```

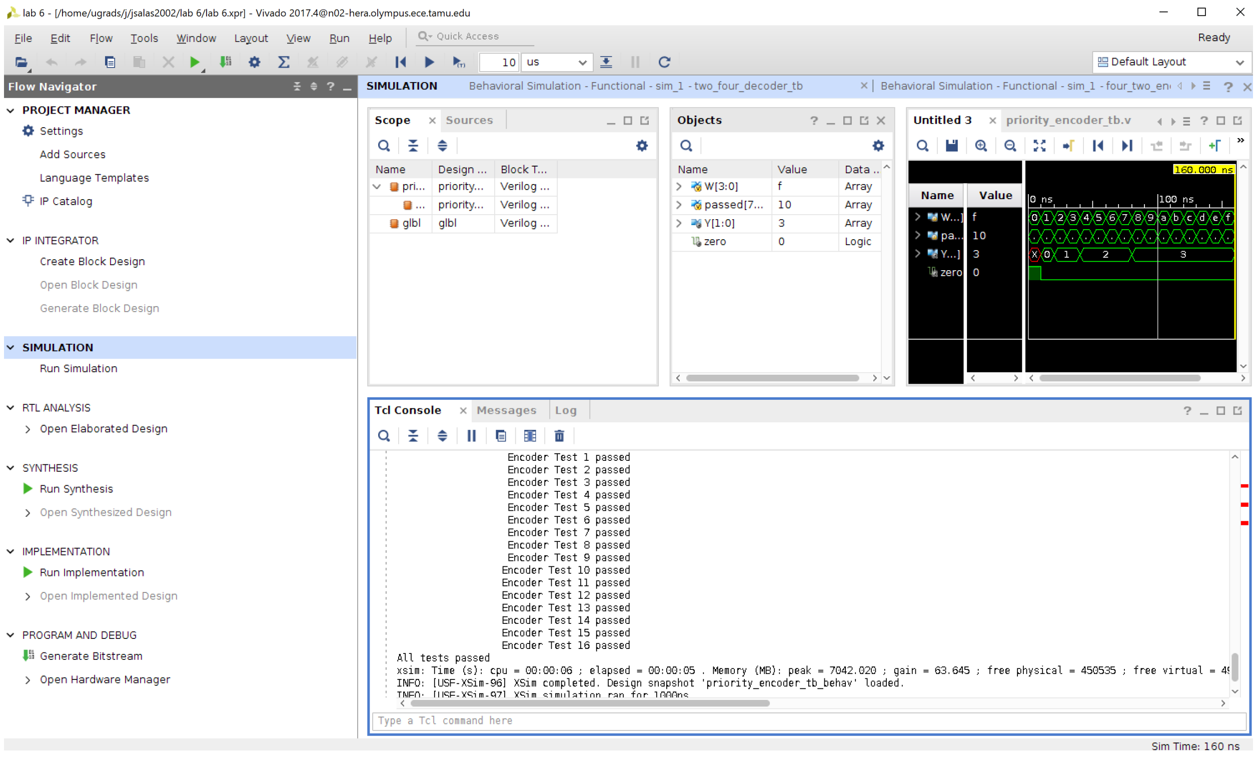
2. Describe a Verilog model for the priority encoder discussed in the background section.

Code: *priority_encoder.v*

```
`timescale 1ns / 1ps
`default_nettype none

module priority_encoder(
    input wire [3:0] W,
    output wire zero,
    output reg [1:0] Y
    );
    assign zero = (W == 4'b0000);

    always@(W)
```

```
        begin
          casex(W)
            4'b0001: Y = 2'b00;
            4'b001X: Y = 2'b01;
            4'b01XX: Y = 2'b10;
            4'b1XXX: Y = 2'b11;
            default: Y = 2'bXX;
          endcase
        end
endmodule
```

Results: four_two_encoder



lab 6 - [/home/ugrads/j/jsalas2002/lab 6/lab 6.xpr] - Vivado 2017.4@n02-hera.olympus.ece.tamu.edu

```
                 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
xsim: Time (s): cpu = 00:00:06 ; elapsed = 00:00:05 . Memory (MB): peak = 7042.020 ; gain = 63.645 ; free physical = 450535 ; free virtual = 49
INFO: [USF-XSim-96] XSim completed. Design snapshot 'priority_encoder_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
```

Sim Time: 160 ns

Experiment Part 3:

For the final part of this experiment, we will be putting the designs we created in the previous experiment onto the Zybo Z7-10 board. We will use the switches, push buttons, and LEDs built into the board to verify the functionality of the components. To do this we will follow the steps found in the lab manual which include loading the .xcd file to the “Add or Create Design Constraints” and generating then clicking on “Generate Bitstream File.” After this, I will connect the board and show my TA Sri that it is in fact working as planned. After we will conduct the same experiment for part 2, we will see if the 2:4 binary decoder works. After these are done we will test that the 4:2 binary encoder that we also simulated works and compare the priority encoder to the binary decoder and demonstrate to our TA once we find the design to work. This concludes experiment #6.

Results:

All my code resulted in passed tests for all types of circuits, as shown with all the experiment parts, especially in parts 1 and 2, the code is shown and properly labeled as well as the waveforms that were correct with the tests passed.

Conclusion:

In this lab, we were introduced to behavioral modeling in Verilog which is used to describe the algorithmic components of a circuit, although it is the most confusing type of modeling to do compared to structural and dataflow, I am starting to understand it and get a better grasp on this subject.

Post-lab Deliverables:

1. Include the source code with comments for all modules you simulated. You do not have to include test bench code. Code without comments will not be accepted!

The code is provided throughout the lab report and clearly labeled.

2. Include screenshots of all waveforms captured during simulation in addition to the test bench console output for each test bench simulation.

All screenshots and waveforms captured from the lab are shown throughout the lab report and labeled properly.

3. Provide a comparison between the behavioral Verilog used in this week's lab and the structural and dataflow Verilog used in last week's lab. What might be the advantages and disadvantages of each?

Structural code is used to distinguish hardware in the circuit more than the overall process of the circuit. The next type, which is dataflow, is used to help someone understand what is happening to the input throughout a circuit. Behavioral modeling is used when the person already understands the algorithmic components of the circuit.

4. Compare the process of using a breadboard to implementing a digital circuit on an FPGA. State some advantages and disadvantages of each. Which process do you prefer?

I do prefer the breadboarding compared to implementing digital circuits using an HDL, however, I do understand that to simulate more complex and rigorous circuits an HDL is necessary and I do appreciate what it does it is hard to learn but I believe that I am starting to understand it much better than I did the first lab with Verilog.

Important Student Feedback

The last part of the lab requests your feedback. We are continually trying to improve the laboratory exercises to enhance your learning experience, and we are unable to do so without your feedback. Please include the following post-lab deliverables in your lab report.

Note: If you have any other comments regarding the lab that you wish to bring to your instructor's attention, please feel free to include them as well.

1. What did you like most about the lab assignment and why? What did you like least about it and why?

I enjoyed learning the difference between the three types of ways that code is written in Verilog, structural, dataflow, and behavioral, the pre-lab did take me a fair amount of time but I believe it was much worth the work. I did not like how the Zybo-10 board did not work with our computer at the end of the lab.

2. Were there any sections of the lab manual that were unclear? If so, what was unclear? Do you have any suggestions for improving the clarity?

All parts of the lab manual were very clear, except for the last part on experiment 3 where there was a crucial step that was left out to get the Zybo board to work correctly.

3. What suggestions do you have to improve the overall lab assignment?

I have no suggestions for improving the overall lab assignment.