ECEN 248 - Lab Report

Lab Number: 6

Lab Title: Introduction to Behavioral Verilog and Logic Synthesis

Section Number: 508

Student's Name: Joseph Nguyen

Student's UIN: 931006629

Date: 10/28/2022

TA: Sri Hari Pada Chandanam Kodi

Note: all sections listed below are mandatory in your post lab reports.

Objectives:

The purpose of this lab is to introduce logic simulation and Verilog HDL as a way to build circuits.

Design:

Describe all the steps (be as specific as possible) you use to complete the lab.

First, I logged onto the computers designated in the lab rooms. I then opened up a new terminal and typed in ”mkdir ecen248”, followed by “source /opt/coe/Xilinx/Vivado/2018.3/settings64.sh” then “vivado” in order to load the Vivado software. I then created a new project named lab6 in my directory. After this I followed the lab 6 manual by creating the necessary files to be tested and testing them with the simulation files that were provided. For each file, I had to write up code that essentially builds the circuit for a 1-bit 2:1 MUX, a 4-bit 2:1 MUX, a 4-bit 4:1 MUX, a 2:4 binary decoder and a 4:2 binary encoder, and a primary encoder. For each file, I had to synthesize and implement the code in order to run the simulation for the waveform. After running the simulation, and obtaining the waveform, I took pictures of each waveform along with its respective code. At the end of the manual for part 3, I used the codes for the 2:4 decoder, 4:2 encoder, and the primary encoder to program the FPGA ZYBO Z7-10 board

Results:

Show your results and discuss the observations you made during the lab.

After testing each file, the simulation would show a waveform that shows the inputs tested. The Tcl console would also read whether or not all of the tests in the simulation file passed. For the FPGA, I tested the cases through the switches on the board. Turning all of the switches to “1” would light up the first light, turning all but the last one would light up the last light, and turning all of the switches on except for one of the middle ones would not light up any of the lights.

Conclusion:

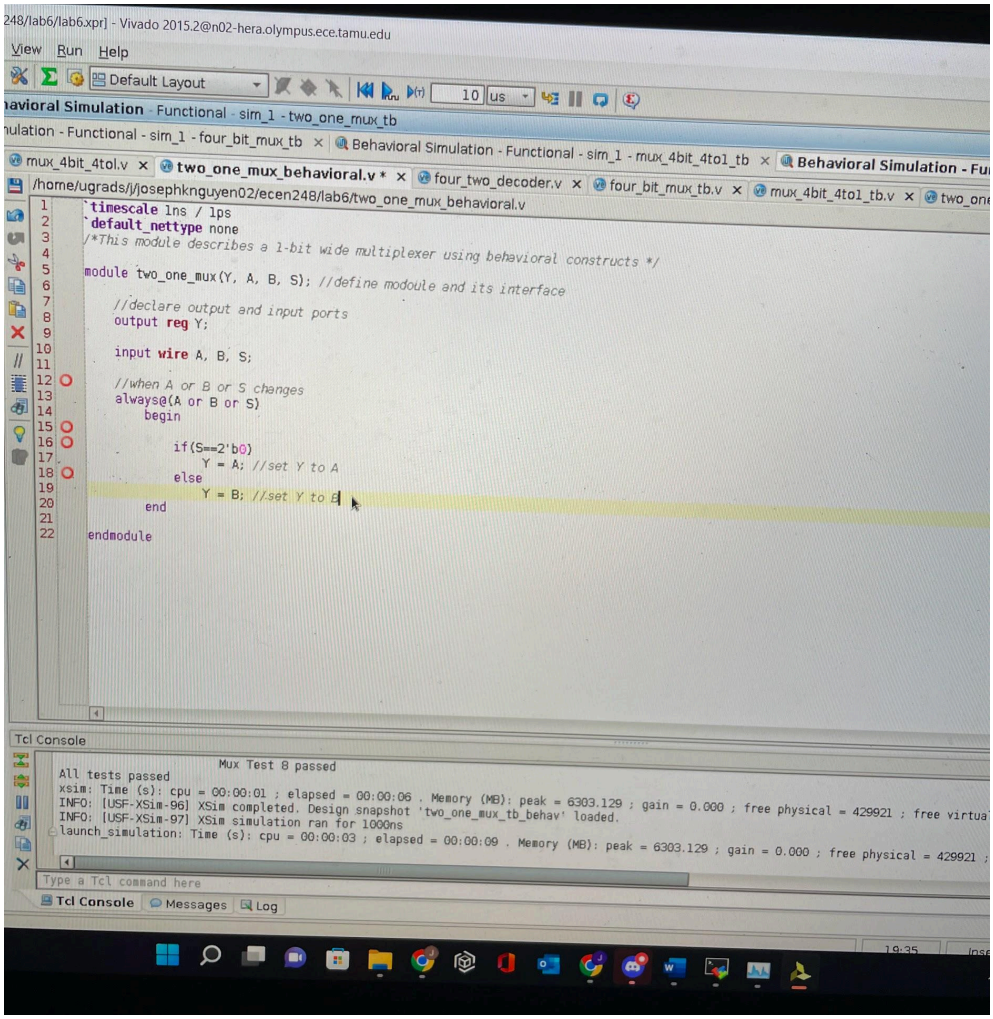
Summarize what you have done and what you have learned from this lab.

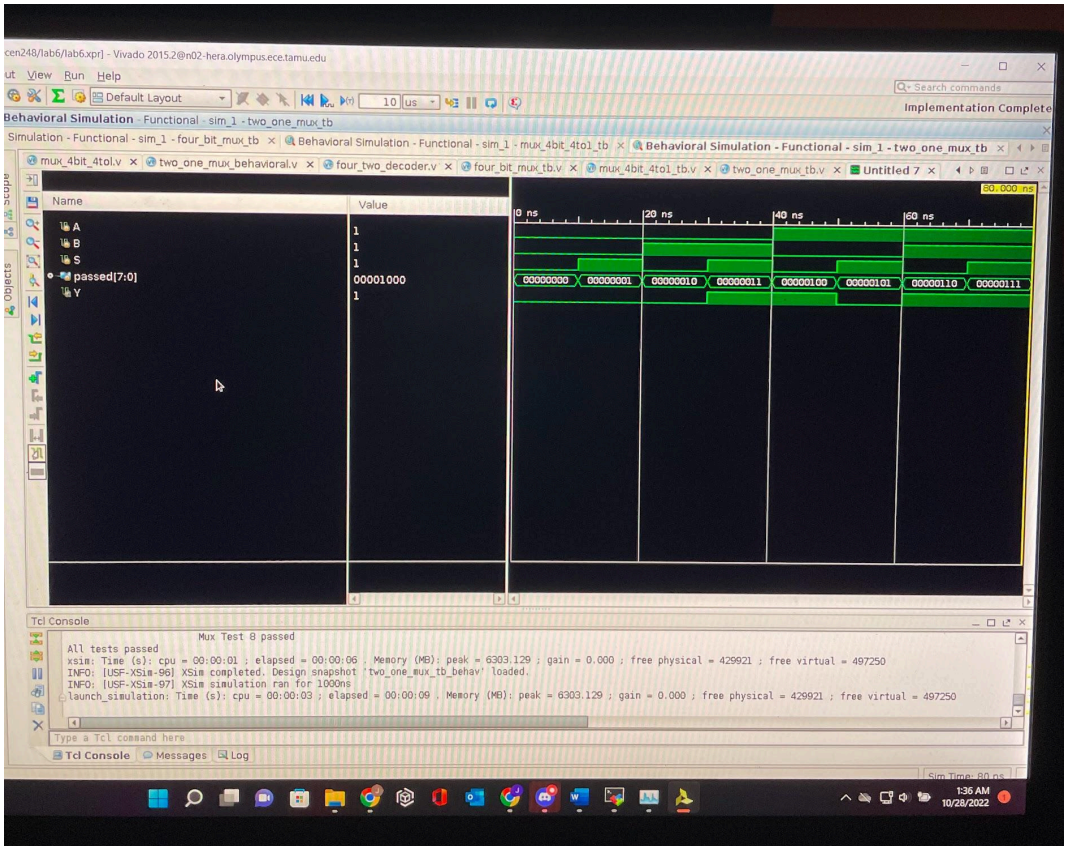
In conclusion, I was able to create and test all of the necessary files in the lab manual and was able to learn how to synthesize different types of digital circuits using behavioral verilog.

Post-lab Deliverables:

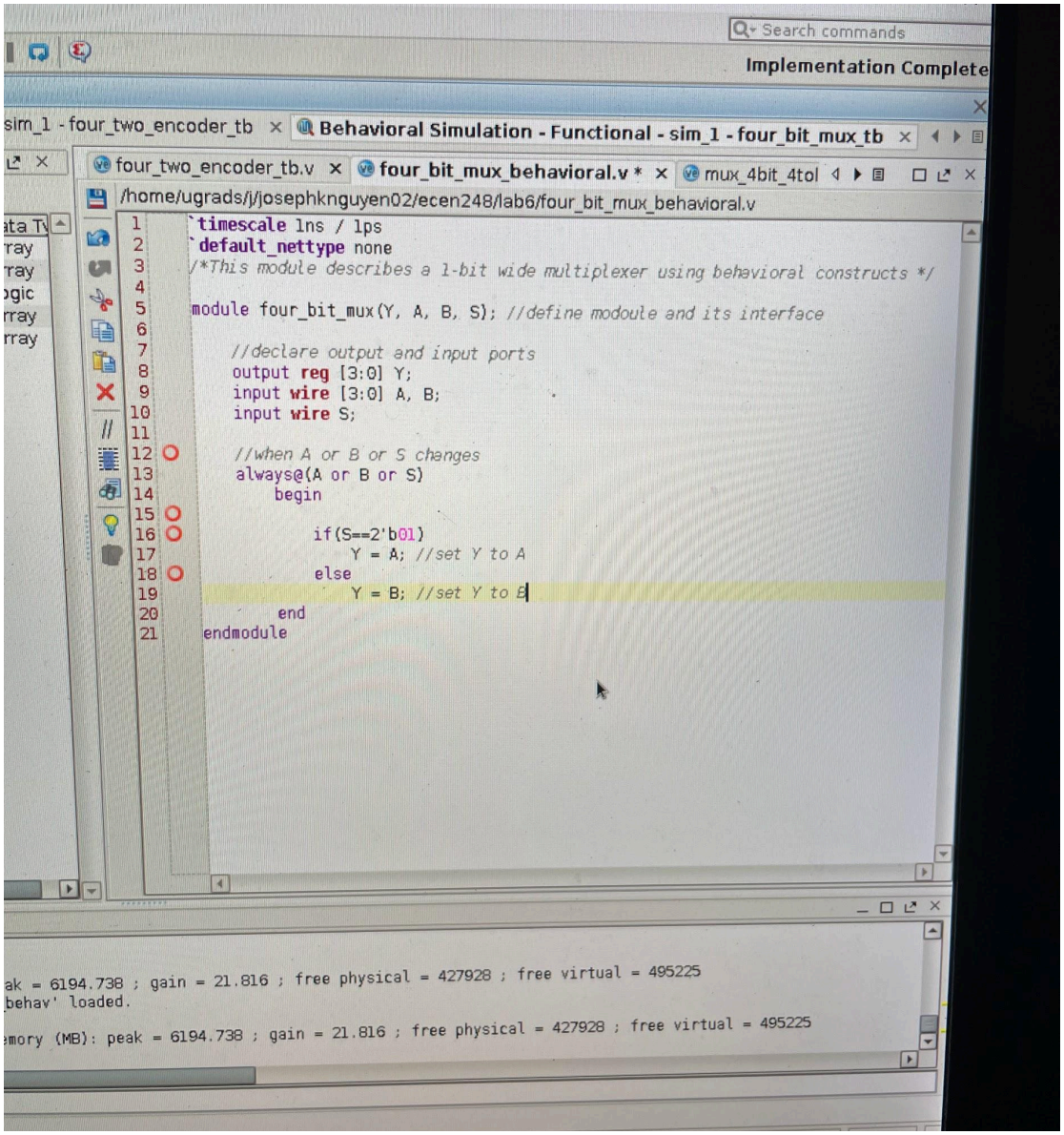
1.) **Verilog code and waveforms for specified circuits**

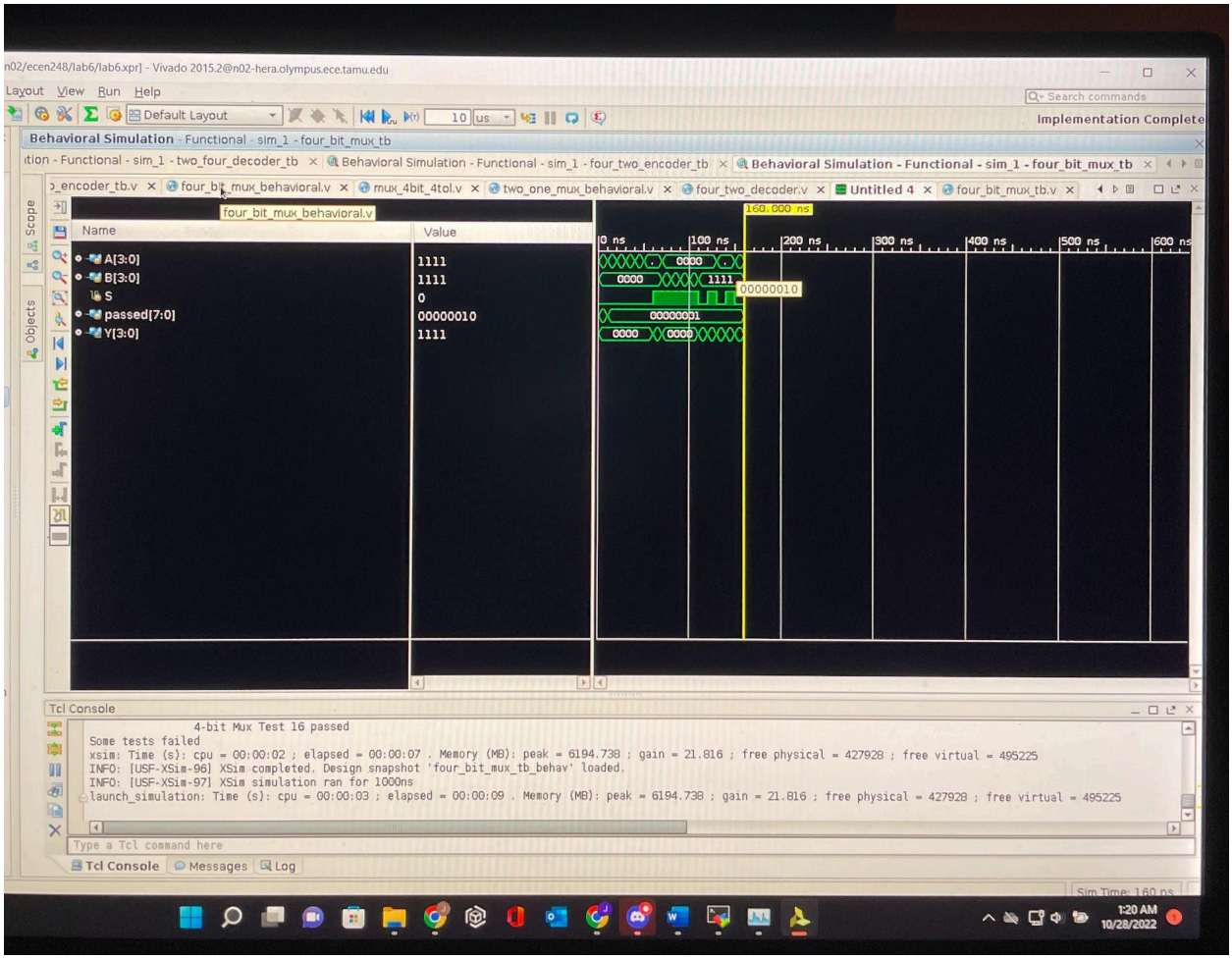
1-bit 2:1 MUX:



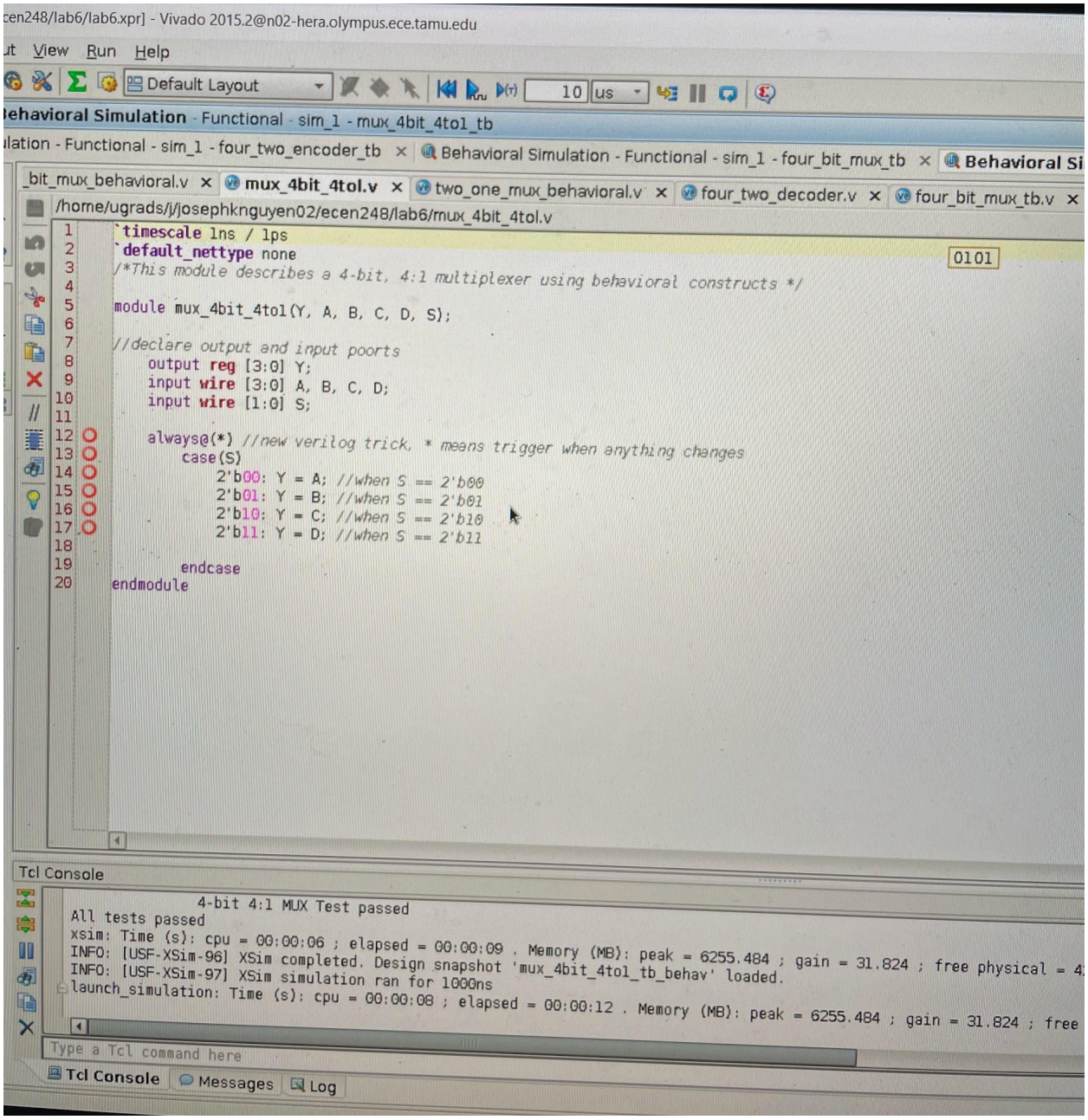

Behavioral Simulation - Functional - sim_1 - two_one_mux_tb
Name
Value
A
B
S
passed[7:0]
Y
1
1
1
00001000
1
0 ns
20 ns
40 ns
60 ns
00000000
00000001
00000010
00000011
00000100
00000101
00000110
00000111
Tcl Console
Mux Test 8 passed
All tests passed
xsim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:06 . Memory (MB): peak = 6303.129 ; gain = 0.000 ; free physical = 429921 ; free virtual = 497250
INFO: [USF-XSim-96] XSim completed. Design snapshot 'two_one_mux_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:09 . Memory (MB): peak = 6303.129 ; gain = 0.000 ; free physical = 429921 ; free virtual = 497250
Type a Tcl command here
Tcl Console
Messages
Log
1:36 AM
10/28/2022

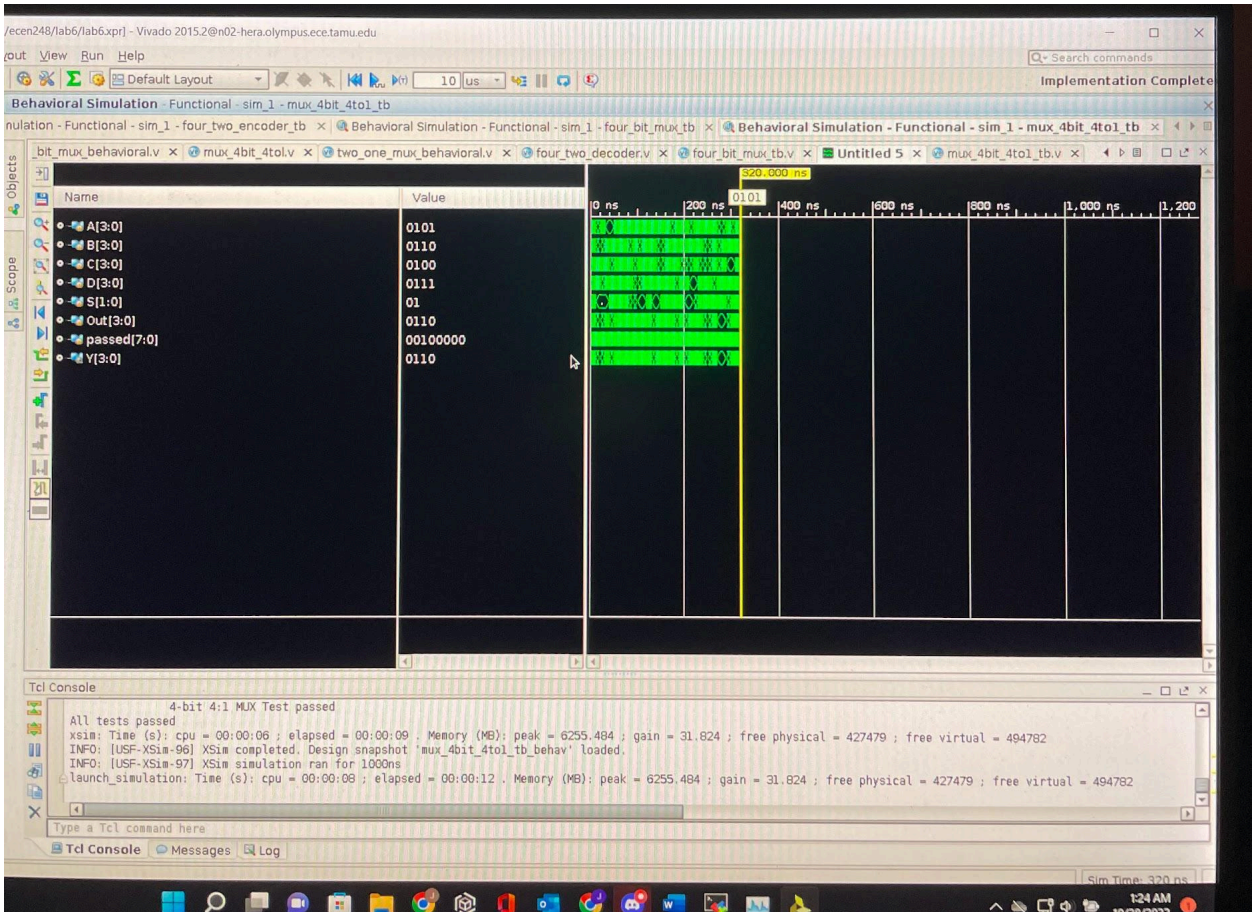
4-bit 2:1 MUX:


Search commands
Implementation Complete
sim_1 - four_two_encoder_tb
Behavioral Simulation - Functional - sim_1 - four_bit_mux_tb
four_two_encoder_tb.v
four_bit_mux_behavioral.v *
mux_4bit_4to1
/home/ugrads/j/josephknguyen02/ecen248/lab6/four_bit_mux_behavioral.v
1 `timescale 1ns / 1ps
2 `default_nettype none
3 /*This module describes a 1-bit wide multiplexer using behavioral constructs */
4
5 module four_bit_mux(Y, A, B, S); //define modoule and its interface
6
7 //declare output and input ports
8 output reg [3:0] Y;
9 input wire [3:0] A, B;
10 input wire S;
11
12 //when A or B or S changes
13 always@(A or B or S)
14 begin
15
16 if(S==2'b01)
17 Y = A; //set Y to A
18 else
19 Y = B; //set Y to B
20 end
21 endmodule
ak = 6194.738 ; gain = 21.816 ; free physical = 427928 ; free virtual = 495225
behav' loaded.
emory (MB): peak = 6194.738 ; gain = 21.816 ; free physical = 427928 ; free virtual = 495225

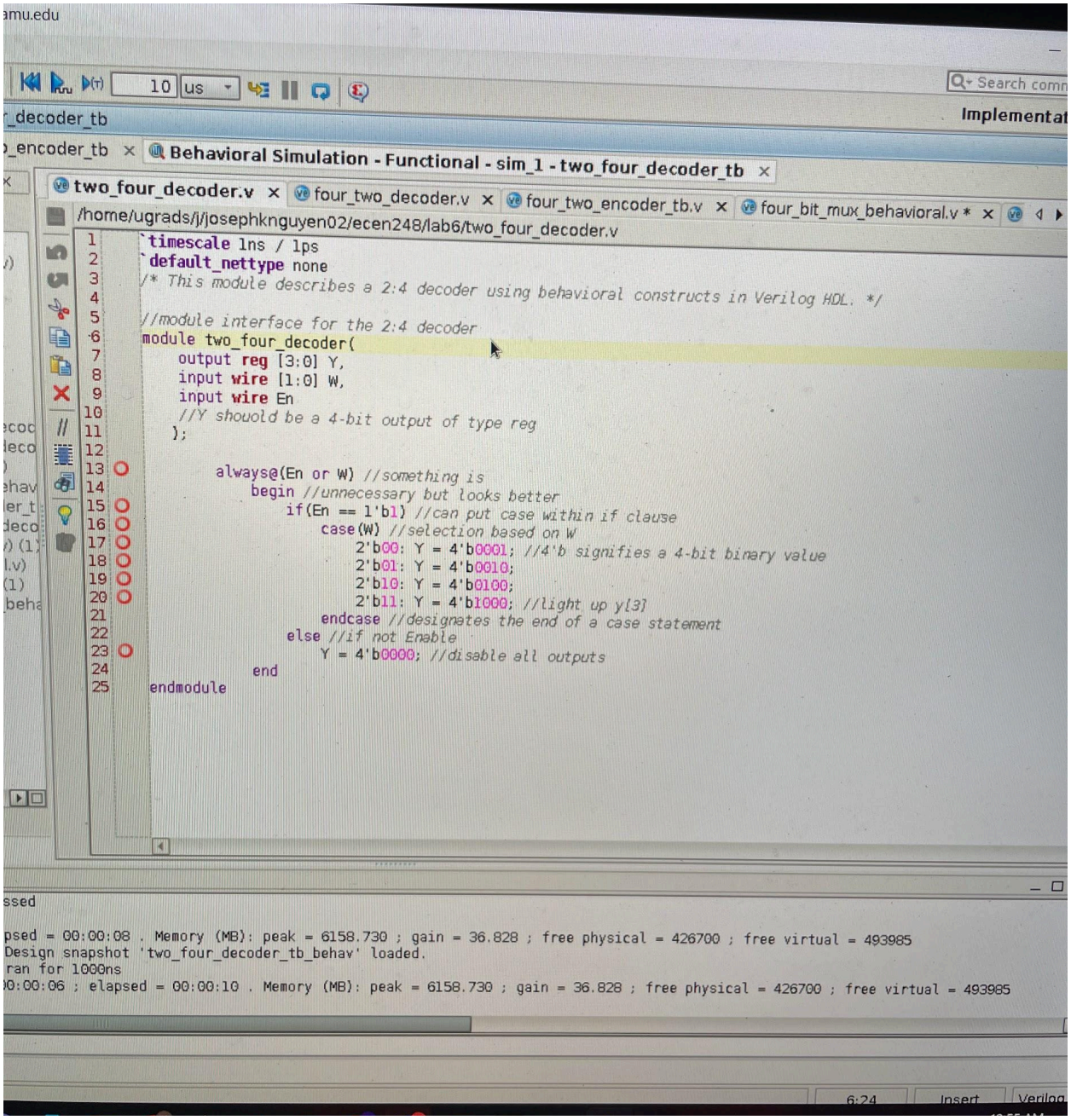

Behavioral Simulation - Functional - sim_1 - four_bit_mux_tb
four_bit_mux_behavioral.v
Name
Value
A[3:0]
1111
B[3:0]
1111
S
0
passed[7:0]
00000010
Y[3:0]
1111
160.000 ns
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
Tcl Console
4-bit Mux Test 16 passed
Some tests failed
xsim: Time (s): cpu = 00:00:02 ; elapsed = 00:00:07 . Memory (MB): peak = 6194.738 ; gain = 21.816 ; free physical = 427928 ; free virtual = 495225
INFO: [USF-XSim-96] XSim completed. Design snapshot 'four_bit_mux_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:09 . Memory (MB): peak = 6194.738 ; gain = 21.816 ; free physical = 427928 ; free virtual = 495225
Type a Tcl command here
Tcl Console
Messages
Log
1:20 AM
10/28/2022

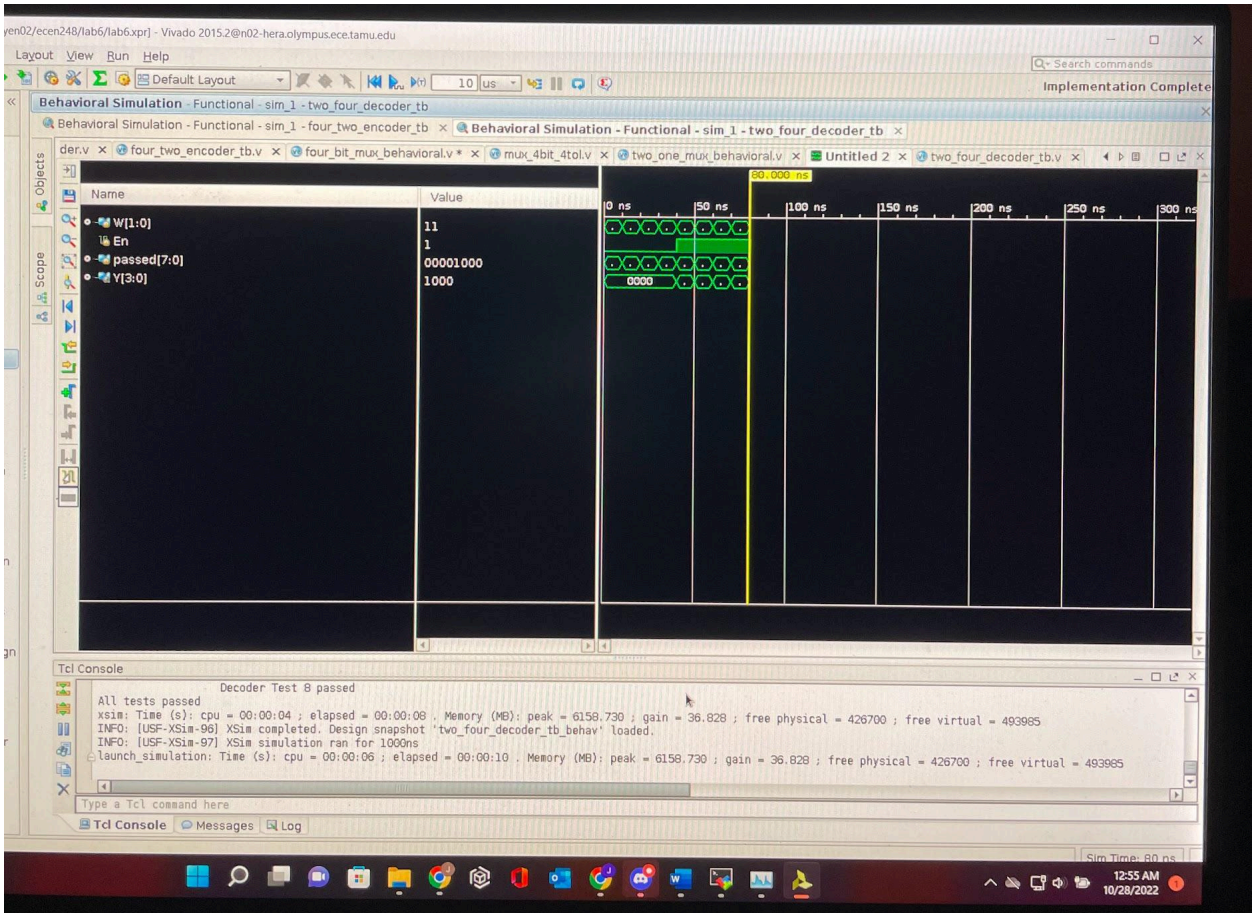
4-bit 4:1 MUX:


cen248/lab6/lab6.xpr] - Vivado 2015.2@n02-hera.olympus.ece.tamu.edu
ut View Run Help
Default Layout
10 us
ehavioral Simulation - Functional - sim_1 - mux_4bit_4to1_tb
ilation - Functional - sim_1 - four_two_encoder_tb
Behavioral Simulation - Functional - sim_1 - four_bit_mux_tb
Behavioral Si
_bit_mux_behavioral.v
mux_4bit_4to1.v
two_one_mux_behavioral.v
four_two_decoder.v
four_bit_mux_tb.v
/home/ugrads/j/josephknguyen02/ecen248/lab6/mux_4bit_4to1.v
0101
1 `timescale 1ns / 1ps
2 `default_nettype none
3 /*This module describes a 4-bit, 4:1 multiplexer using behavioral constructs */
4
5 module mux_4bit_4to1(Y, A, B, C, D, S);
6
7 //declare output and input poorts
8 output reg [3:0] Y;
9 input wire [3:0] A, B, C, D;
10 input wire [1:0] S;
11
12 always@(*) //new verilog trick, * means trigger when anything changes
13 case(S)
14 2'b00: Y = A; //when S == 2'b00
15 2'b01: Y = B; //when S == 2'b01
16 2'b10: Y = C; //when S == 2'b10
17 2'b11: Y = D; //when S == 2'b11
18
19 endcase
20 endmodule
Tcl Console
4-bit 4:1 MUX Test passed
All tests passed
xsim: Time (s): cpu = 00:00:06 ; elapsed = 00:00:09 . Memory (MB): peak = 6255.484 ; gain = 31.824 ; free physical = 4
INFO: [USF-XSim-96] XSim completed. Design snapshot 'mux_4bit_4to1_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:08 ; elapsed = 00:00:12 . Memory (MB): peak = 6255.484 ; gain = 31.824 ; free
Type a Tcl command here
Tcl Console
Messages
Log

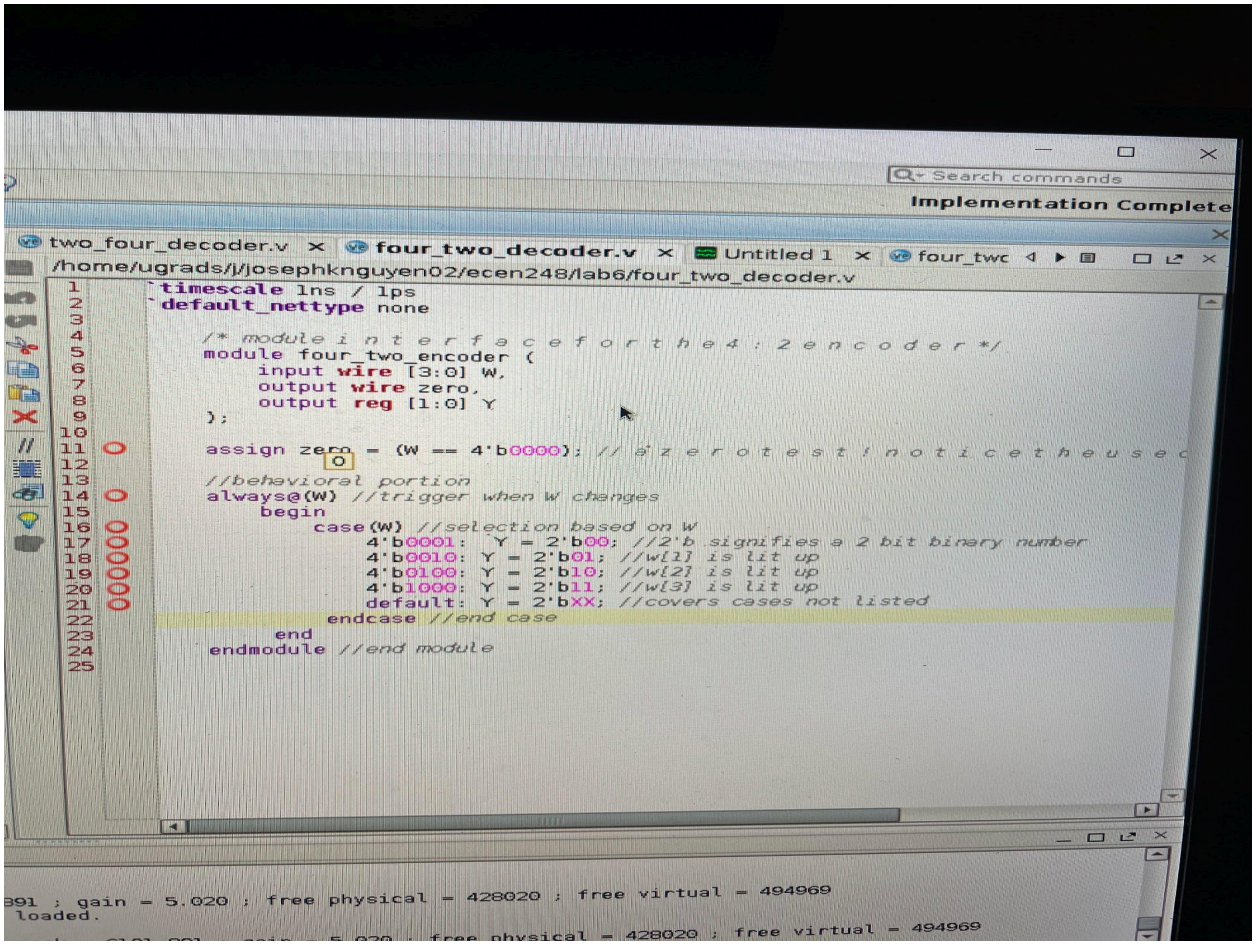


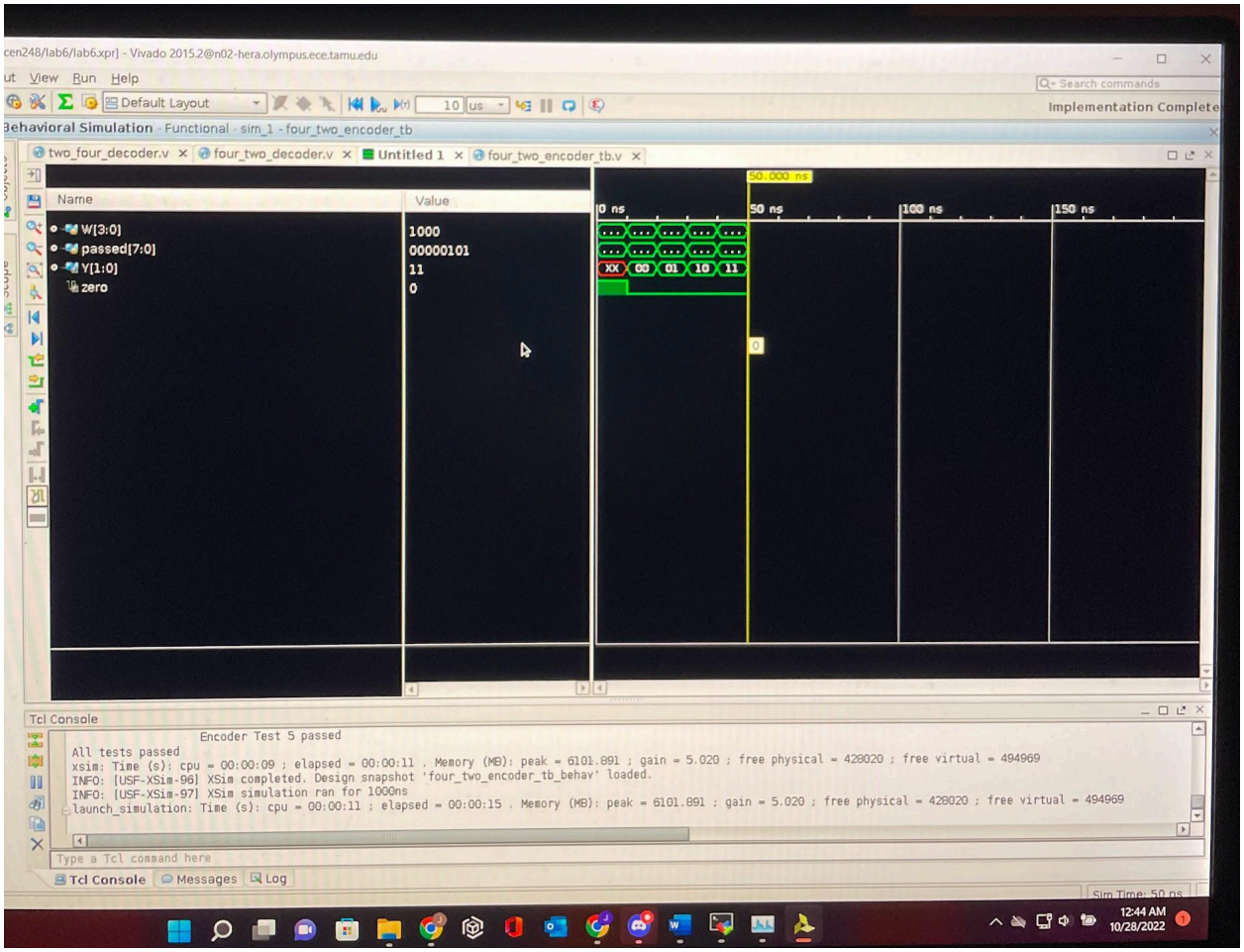
2:4 binary decoder:


amu.edu
10 us
Search comm
Implementat
_decoder_tb
_encoder_tb
Behavioral Simulation - Functional - sim_1 - two_four_decoder_tb
two_four_decoder.v
four_two_decoder.v
four_two_encoder_tb.v
four_bit_mux_behavioral.v *
/home/ugrads/j/josephknguyen02/ecen248/lab6/two_four_decoder.v
1 `timescale 1ns / 1ps
2 `default_nettype none
3 /* This module describes a 2:4 decoder using behavioral constructs in Verilog HDL. */
4
5 //module interface for the 2:4 decoder
6 module two_four_decoder(
7 output reg [3:0] Y,
8 input wire [1:0] W,
9 input wire En
10 //Y shouold be a 4-bit output of type reg
11);
12
13 always@(En or W) //something is
14 begin //unnecessary but looks better
15 if(En == 1'b1) //can put case within if clause
16 case(W) //selection based on W
17 2'b00: Y = 4'b0001; //4'b signifies a 4-bit binary value
18 2'b01: Y = 4'b0010;
19 2'b10: Y = 4'b0100;
20 2'b11: Y = 4'b1000; //light up y[3]
21 endcase //designates the end of a case statement
22 else //if not Enable
23 Y = 4'b0000; //disable all outputs
24 end
25 endmodule
ssed
psed = 00:00:08 . Memory (MB): peak = 6158.730 ; gain = 36.828 ; free physical = 426700 ; free virtual = 493985
Design snapshot 'two_four_decoder_tb_behav' loaded.
ran for 1000ns
0:00:06 ; elapsed = 00:00:10 . Memory (MB): peak = 6158.730 ; gain = 36.828 ; free physical = 426700 ; free virtual = 493985
6:24
Insert
Verilog

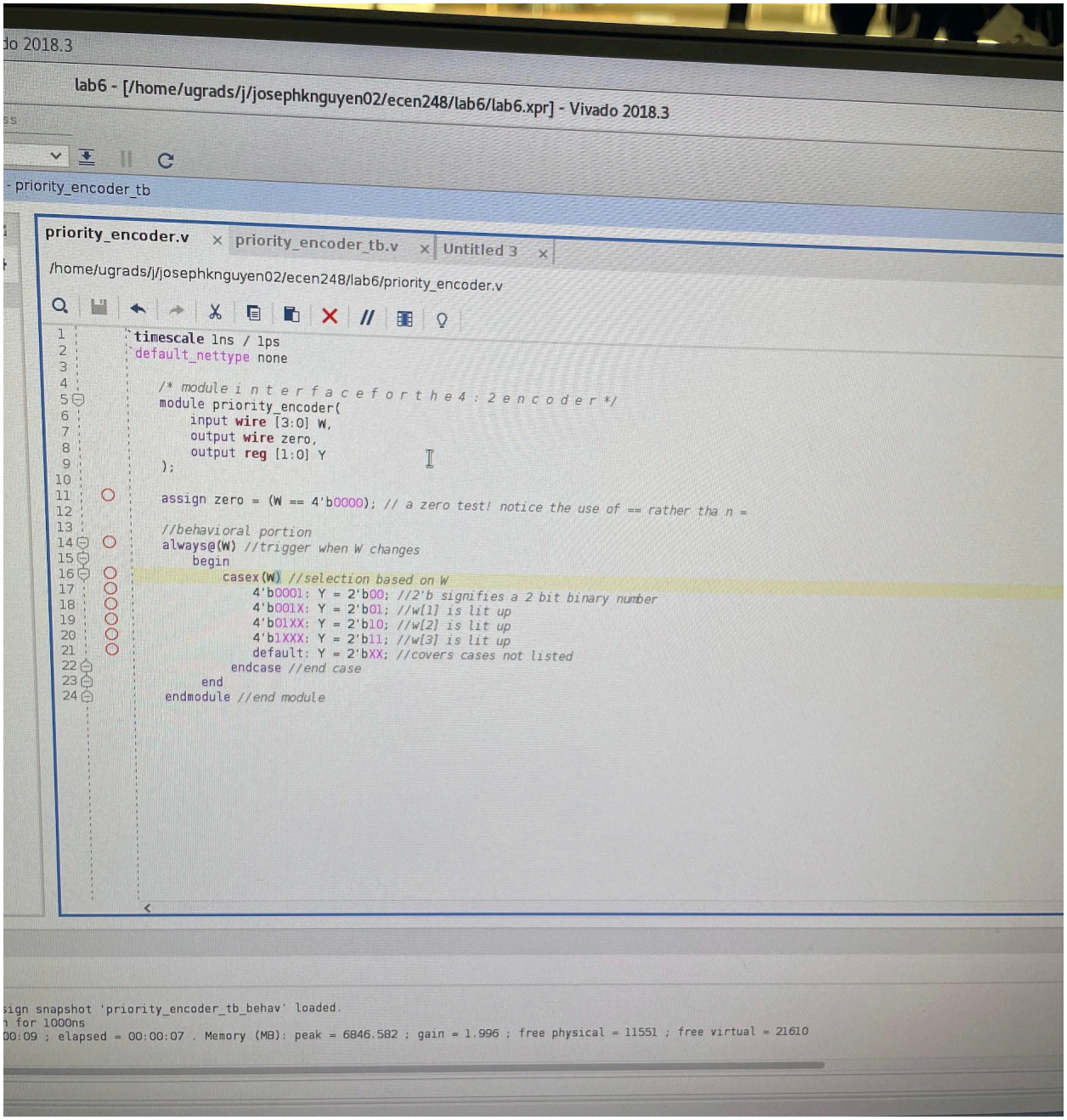


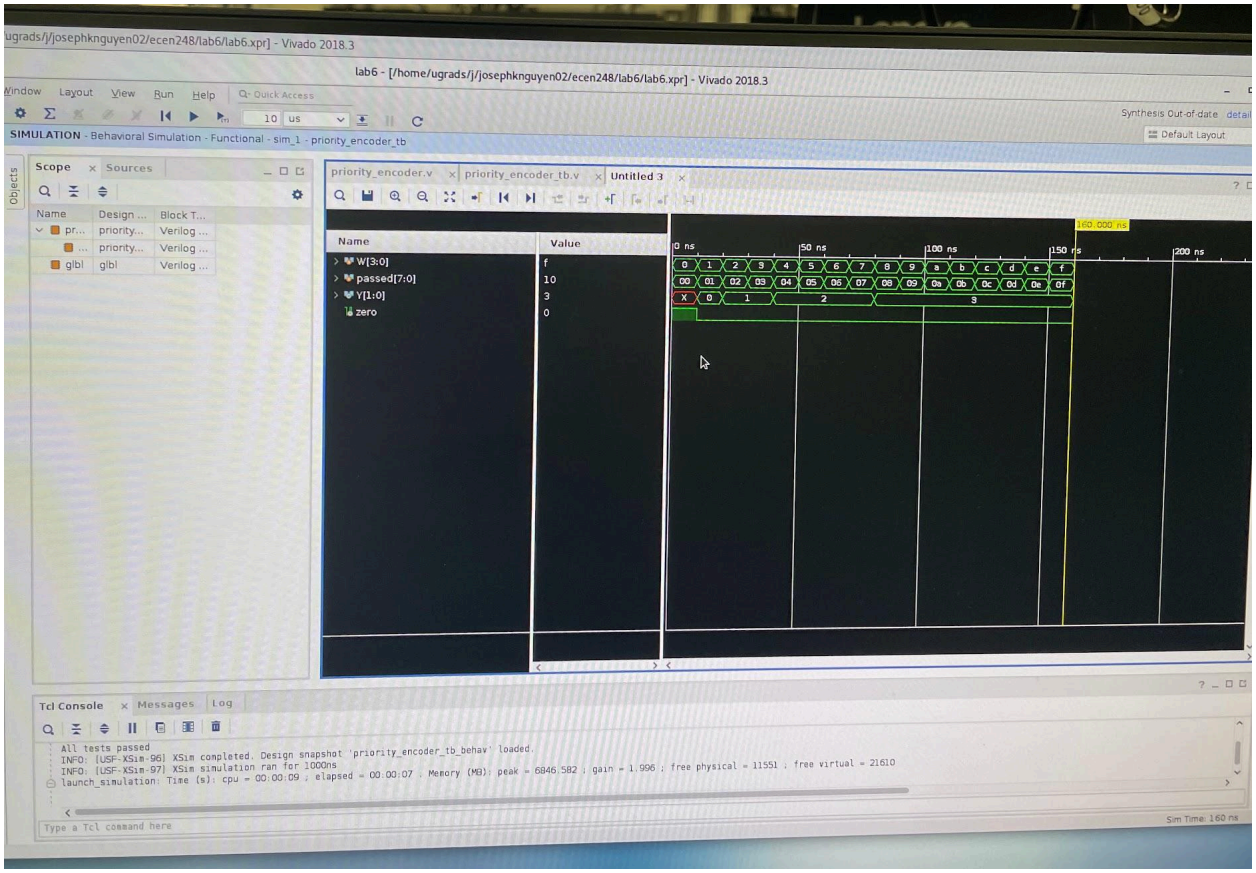
4:2 binary encoder:


Search commands
Implementation Complete
two_four_decoder.v
four_two_decoder.v
Untitled 1
four_twc
/home/ugrads/j/josephknguyen02/ecen248/lab6/four_two_decoder.v
`timescale 1ns / 1ps
`default_nettype none
/* module interface for the 4:2 encoder */
module four_two_encoder (
input wire [3:0] W,
output wire zero,
output reg [1:0] Y
);
assign zero = (W == 4'b0000); // a zero test! notice the used
//behavioral portion
always@(W) //trigger when W changes
begin
case(W) //selection based on W
4'b0001: Y = 2'b00; //2'b signifies a 2 bit binary number
4'b0010: Y = 2'b01; //w[1] is lit up
4'b0100: Y = 2'b10; //w[2] is lit up
4'b1000: Y = 2'b11; //w[3] is lit up
default: Y = 2'bXX; //covers cases not listed
endcase //end case
end
endmodule //end module
391 ; gain = 5.020 ; free physical = 428020 ; free virtual = 494969
loaded.
free physical = 428020 ; free virtual = 494969



priority encoder:


lab6 - [/home/ugrads/j/josephknguyen02/ecen248/lab6/lab6.xpr] - Vivado 2018.3
priority_encoder_tb
priority_encoder.v
priority_encoder_tb.v
Untitled 3
/home/ugrads/j/josephknguyen02/ecen248/lab6/priority_encoder.v
timescale 1ns / 1ps
default_nettype none
/* module interface for the 4:2 encoder */
module priority_encoder(
input wire [3:0] W,
output wire zero,
output reg [1:0] Y
);
assign zero = (W == 4'b0000); // a zero test! notice the use of == rather than =
//behavioral portion
always@(W) //trigger when W changes
begin
casex(W) //selection based on W
4'b0001: Y = 2'b00; //2'b signifies a 2 bit binary number
4'b001X: Y = 2'b01; //w[1] is lit up
4'b01XX: Y = 2'b10; //w[2] is lit up
4'b1XXX: Y = 2'b11; //w[3] is lit up
default: Y = 2'bXX; //covers cases not listed
endcase //end case
end
endmodule //end module
sign snapshot 'priority_encoder_tb_behav' loaded.
for 1000ns
00:09 ; elapsed = 00:00:07 . Memory (MB): peak = 6846.582 ; gain = 1.996 ; free physical = 11551 ; free virtual = 21610



After simulating and implementing the verilog code for the 4:2 binary encoder and the priority encoder, I programmed the two files into the FPGA and tested their behavior.

Provide a comparison between 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?

For behavioral verilog, the code itself does not have to be synthesizable and is structured by procedural statements that describe the function of the design which controls the simulation. For Dataflow, the system is designed in terms of its function and uses boolean equations and operators to relate the inputs and outputs.

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?

For the process of implementing a digital circuit of an FPGA on a breadboard, you would have to choose your input and output wires, and connect your inputs to the different chip gates, and then define your output wires according to the combinations of your

inputs and the gates. For implementing it in Verilog, the process is much cleaner since you can just type out the necessary circuits in a code and then implement the FPGA. Using a breadboard would allow you to clearly see where your inputs and outputs are going due to the wires, however it is much harder to debug since there are a lot of wires that could potentially create confusion.