**ECEN 248 - Lab Report**

**Lab Number: 5**

**Lab Title: Introduction to Logic Simulation and Verilog**

**Section Number: 519**

**Student’s Name: Alex Allahar**

**Student’s UIN: 928009686**

**Date: 10/18/2023**

**TA: Yi Deng**

**Objectives:**

In this lab, we will be exposed to and learn the basics of simulating digital circuits using Verilog. This lab focuses on the concept of design process and digital circuit simulation. Using the digital circuit design in labs on a breadboard, we will create and see their Verilog comparison.

**Design:**

To start the lab, open vivado and create a new project titled “lab5”. When creating the project select “Zybo Z7-10”. In this project create a new file, “two\_one\_mux.v”. The file should open an editor window in which the 2:1 MUX Code from the lab manual can be copied. Add this file to the design sources in the Hierarchy panel. To import the test bench file, download the “VerilogFiles\_Fall2021.zip”, and move the file, “two\_one\_mux\_tb.v” into the lab5 vivado project folder directory. Once the file is in this directory, add this file to the simulation sources using the add sources function in the hierarchy panel. To simulate the 2:1 MUX code, set the test bench file as the top in the hierarchy and run the simulation. After running the simulation a waveform output panel should appear, as well as a test bench console output. After getting these results I repeated the step with different code and test bench files to simulate the 4-bit 2:1 MUX, +/- UNIT, Full-Adder, and 4-bit ALU.

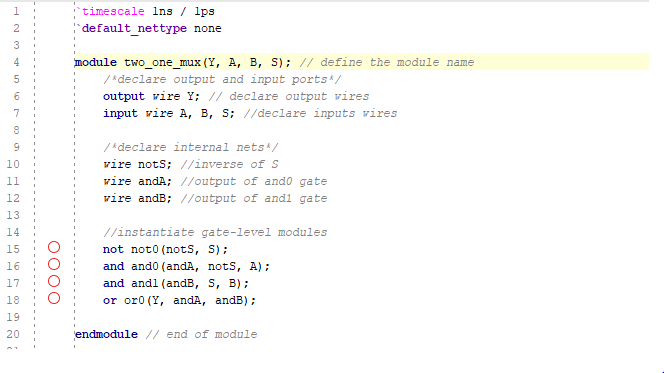
**Results:** All the circuits seemed to follow the logic seen in the previous lab, except the add-sub circuit.

**Conclusion:**

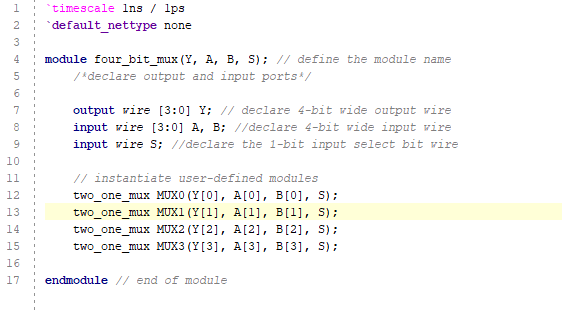
In this lab, I was shown how Vivado can simulate the same digital circuit I have breadboarded. By creating the 2:1 MUX, the 4-bit 2:1 MUX, +/- UNIT, Full-Adder, and 4-bit ALU, I was exposed to all the basic functions of the Verilog code language.

**Post-lab Deliverables:**

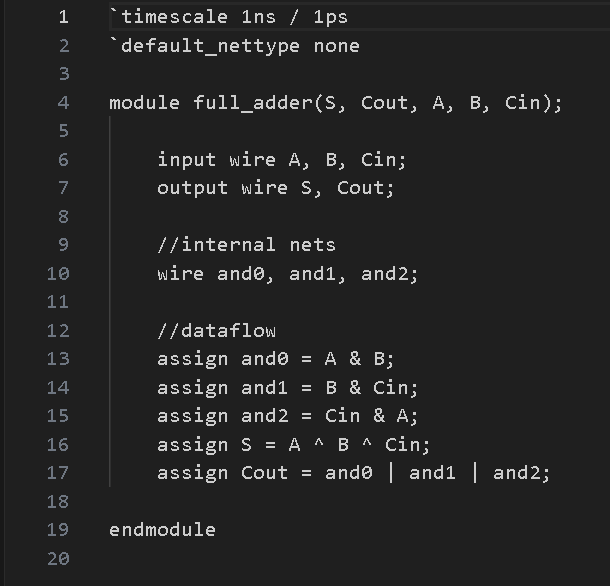
1. Include the source code with comments for all modules you simulated. You do not have to include the test bench code. A code without comments will not be accepted!



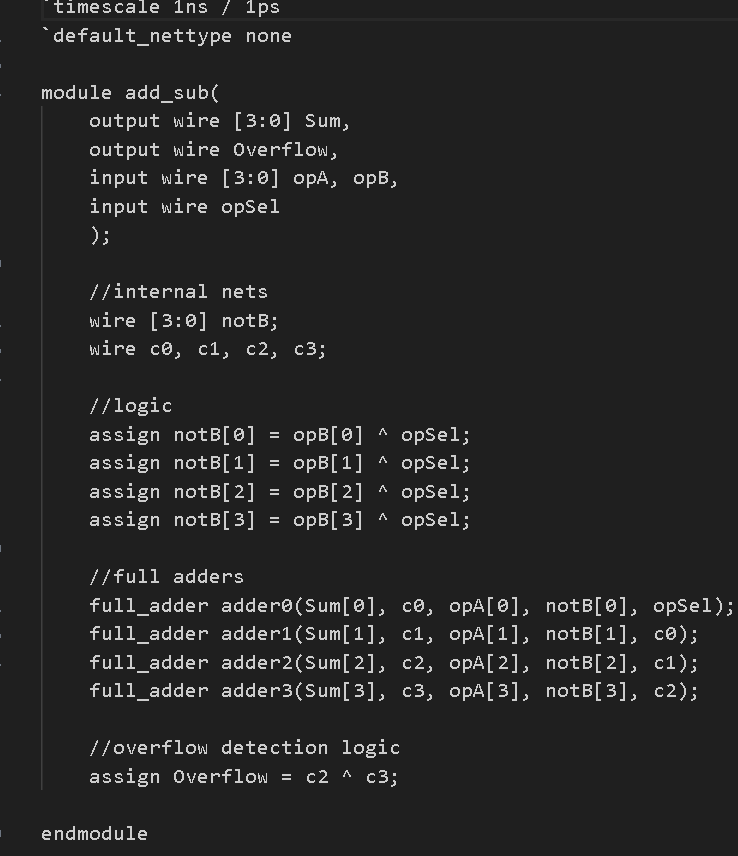
**Figure 1: 2:1 MUX Code**

****

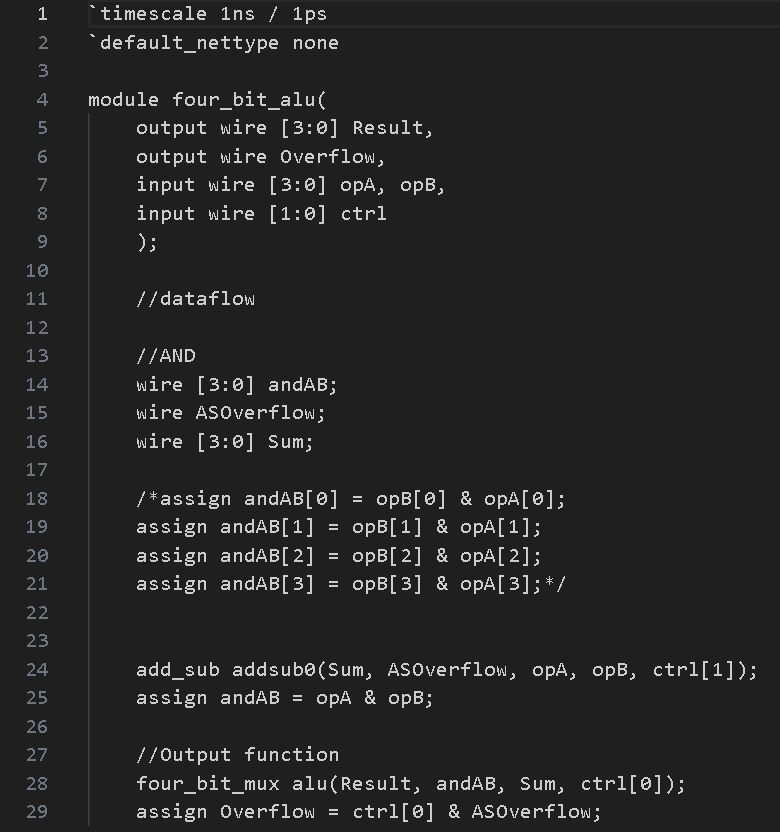
**Figure 2: 4:1 MUX Code**

****

**Figure 3: Full Adder Code**

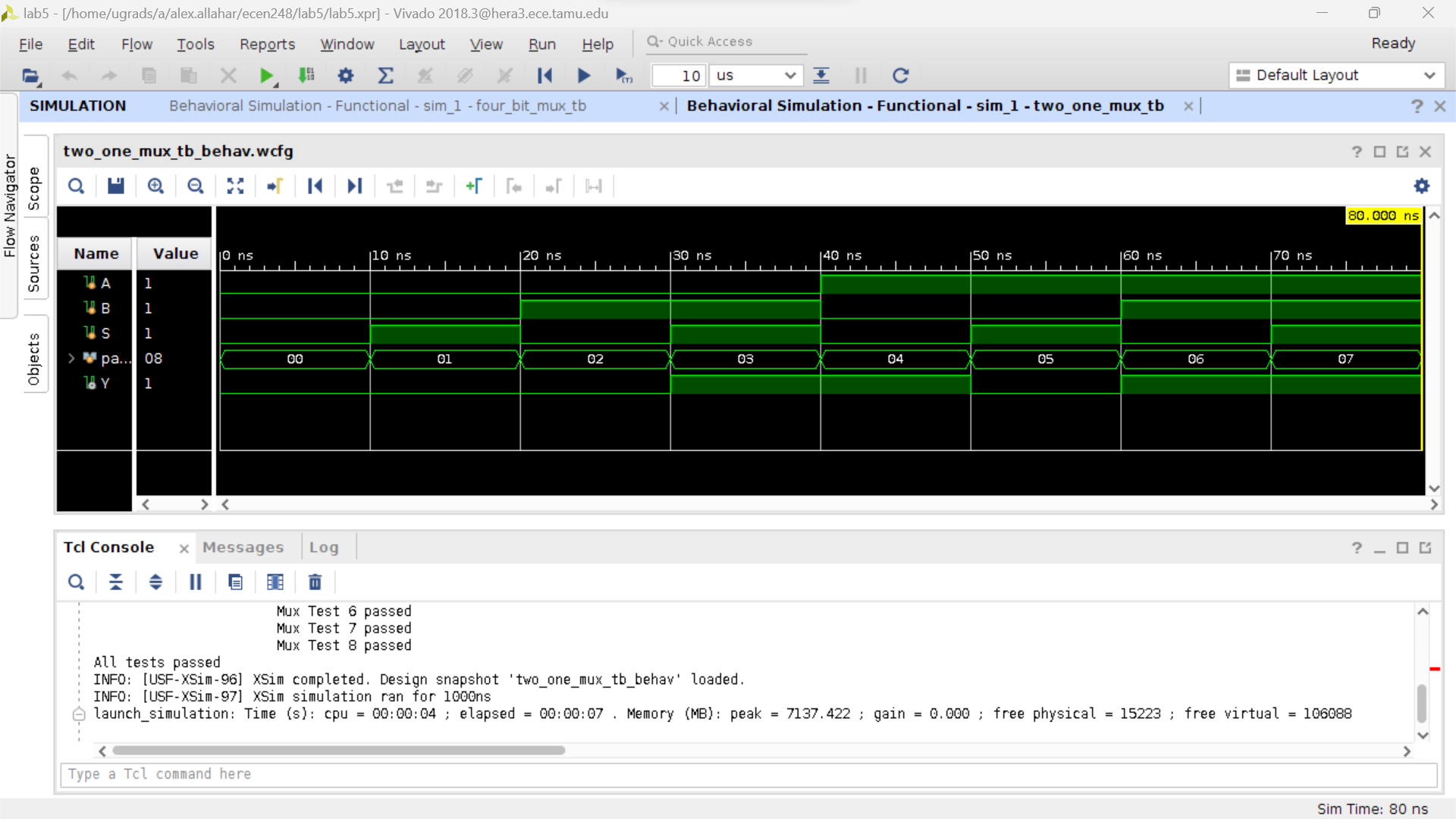
****

**Figure 4: Add-Sub Unit Code**

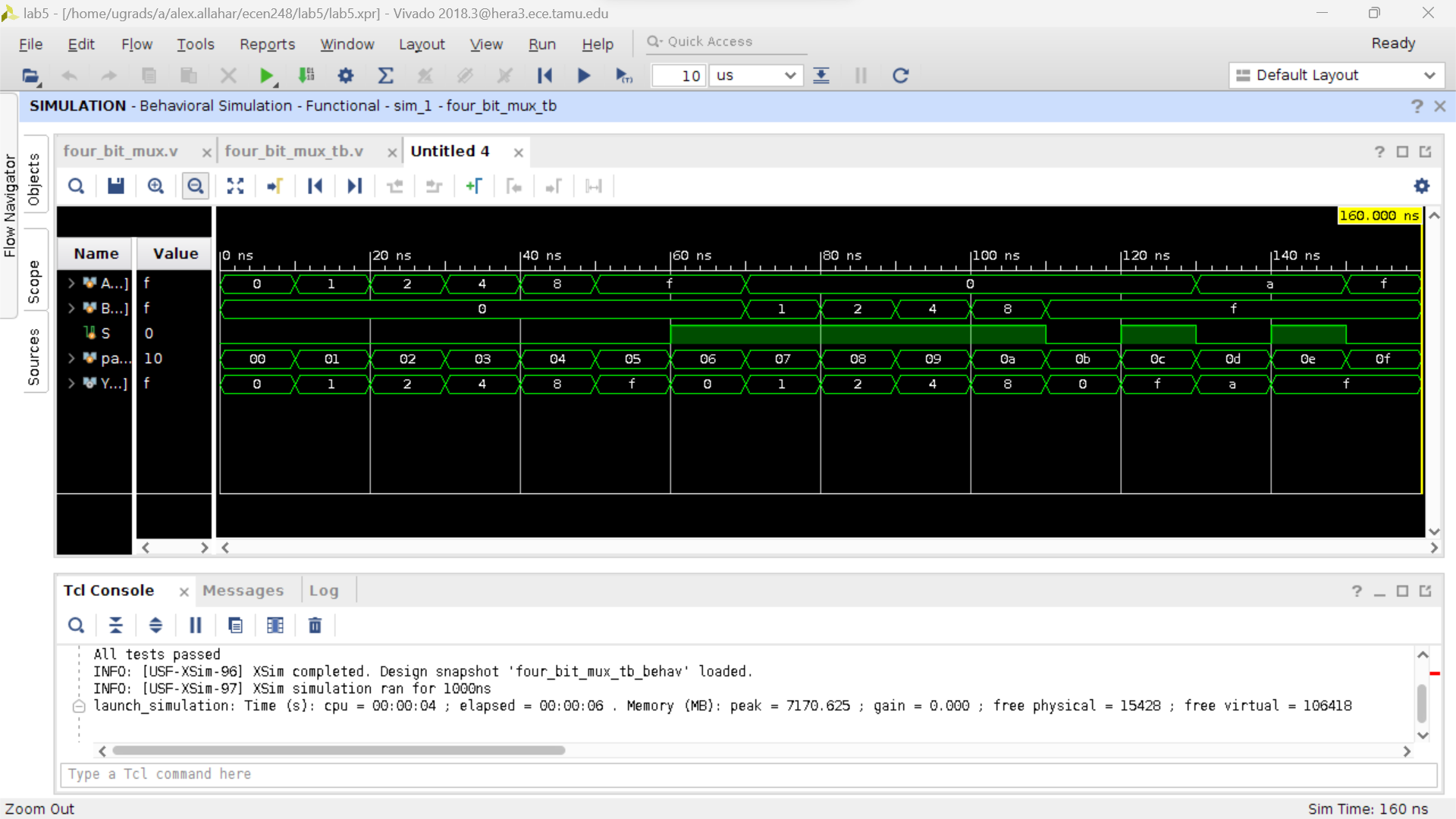
****

**Figure 5: 4-bit ALU Code**

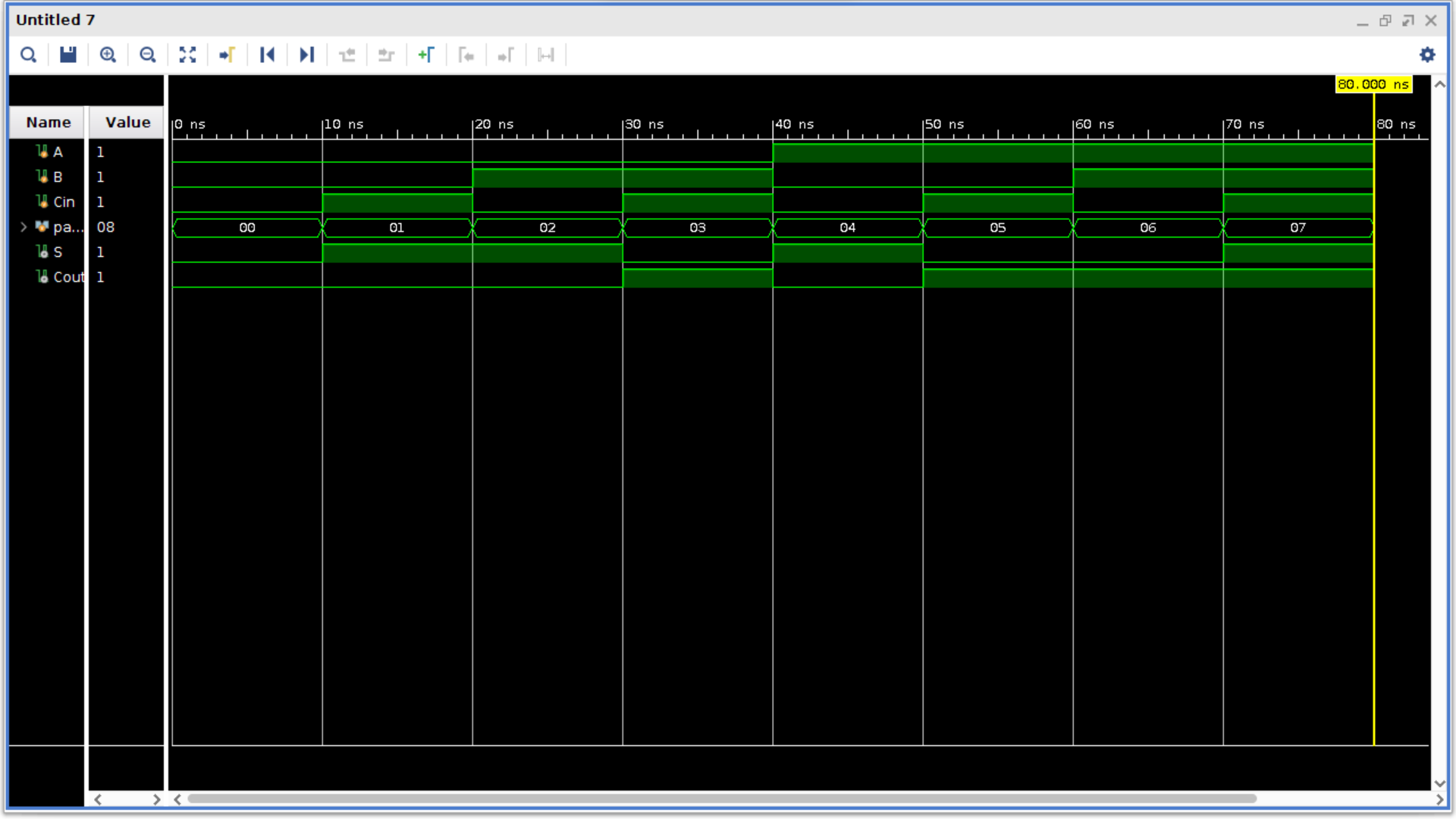
1. Include screenshots of all waveforms captured during simulation in addition to the test bench console output for each test bench simulation. Please ensure these are legible in your report. Waveforms need to be properly fit.

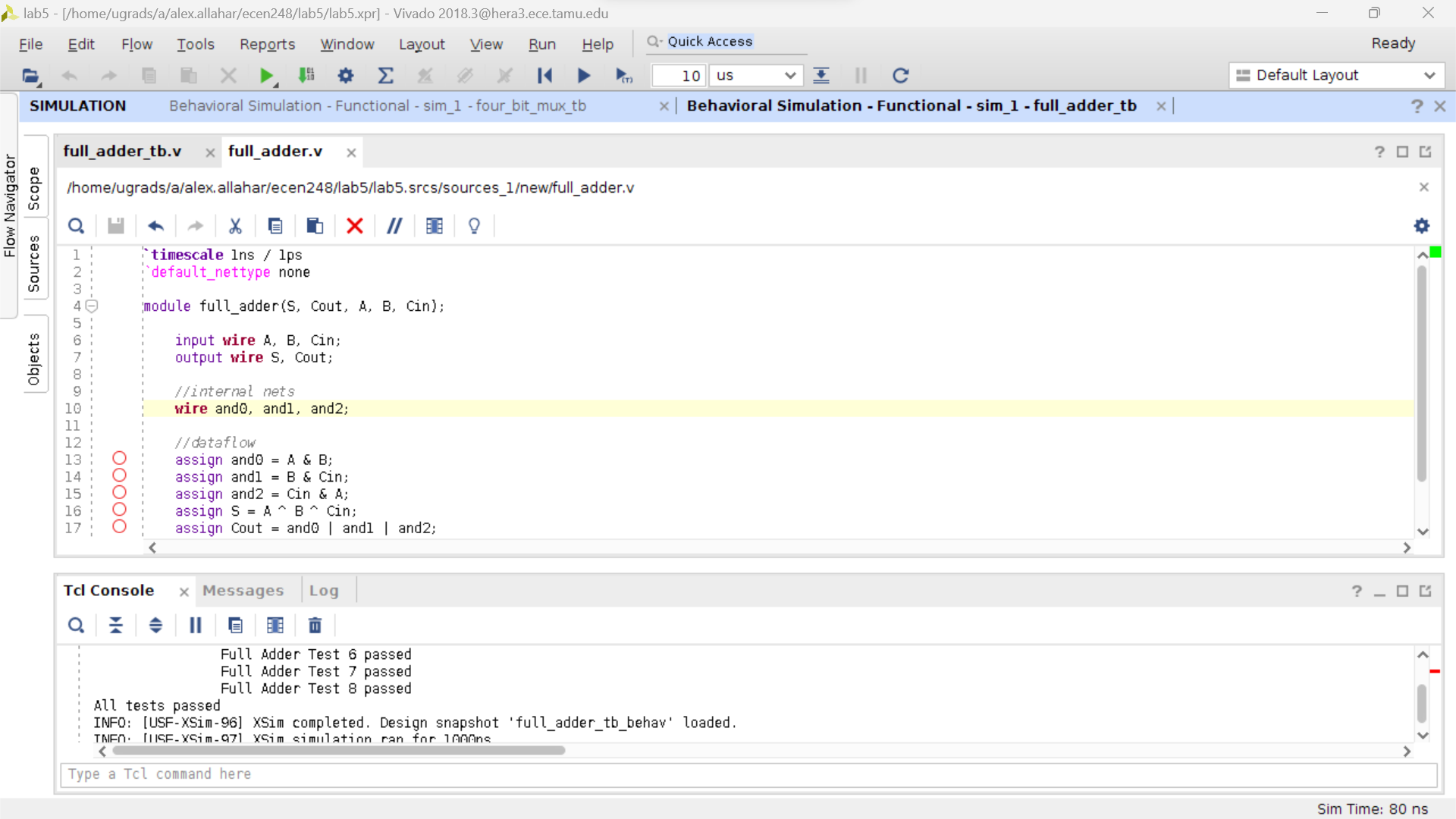


**Figure 6: 2:1 MUX Waveform and Test Console**

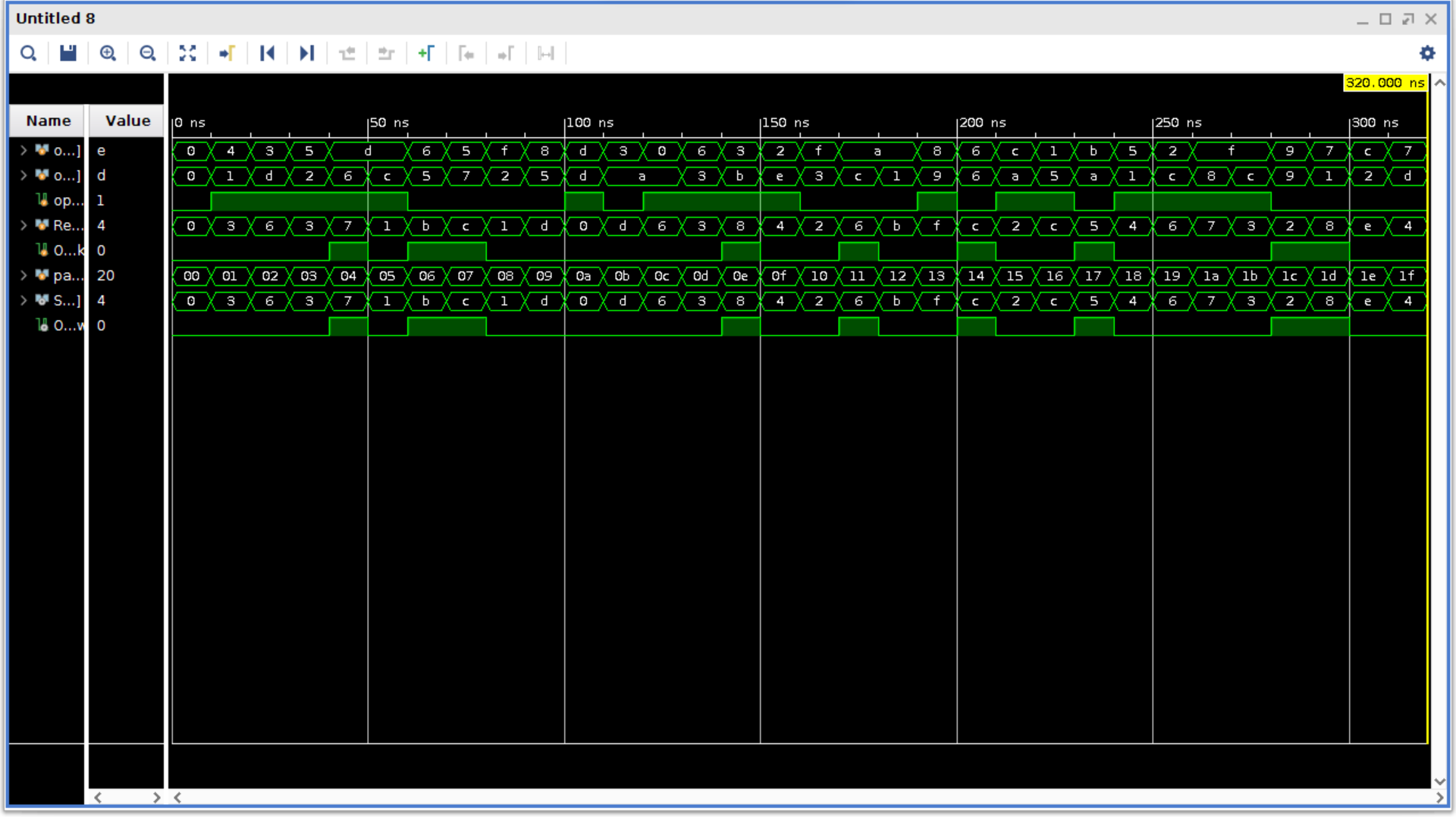
****

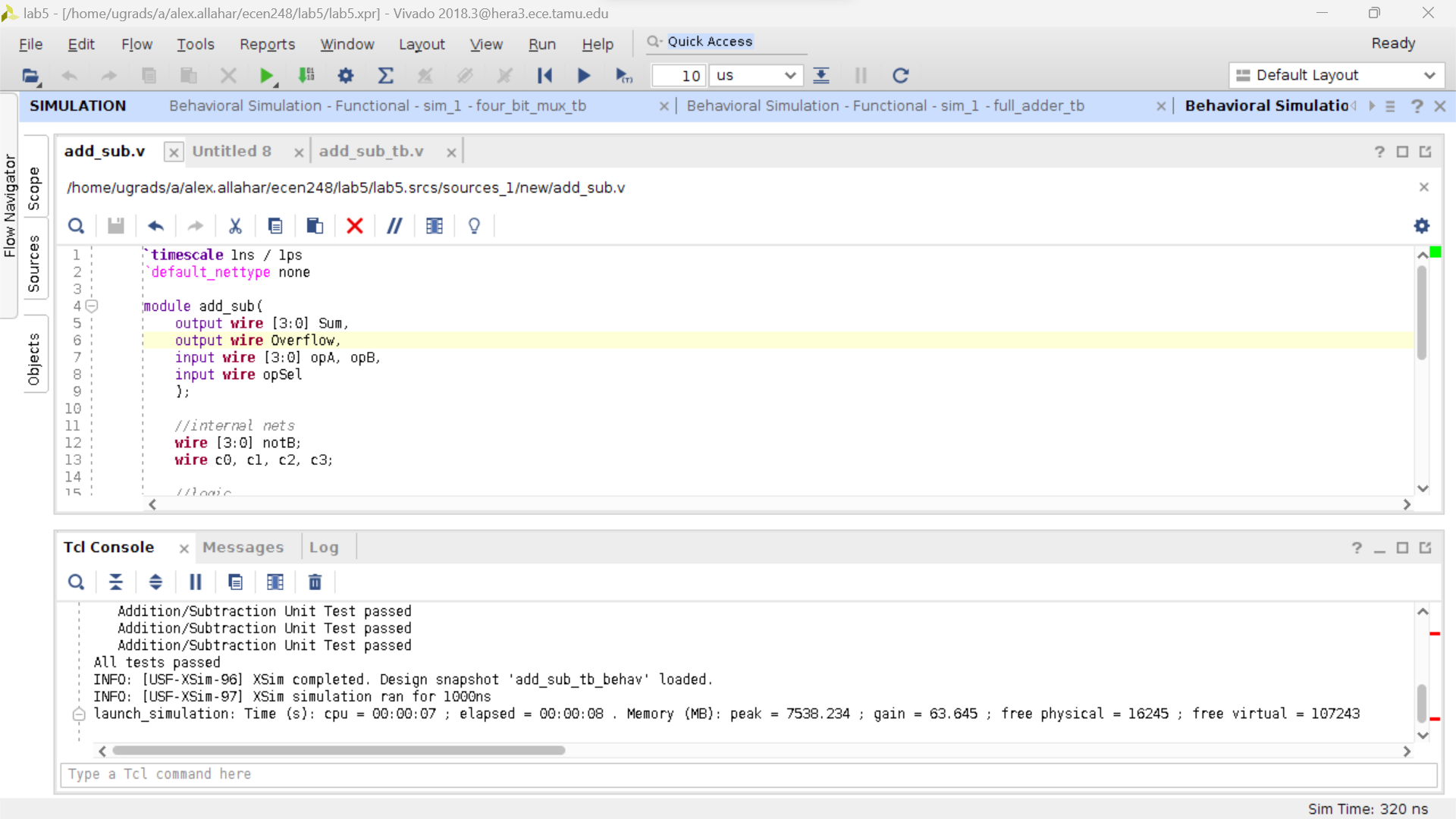
**Figure 7: 4:1 MUX Waveform and Test Console**

****

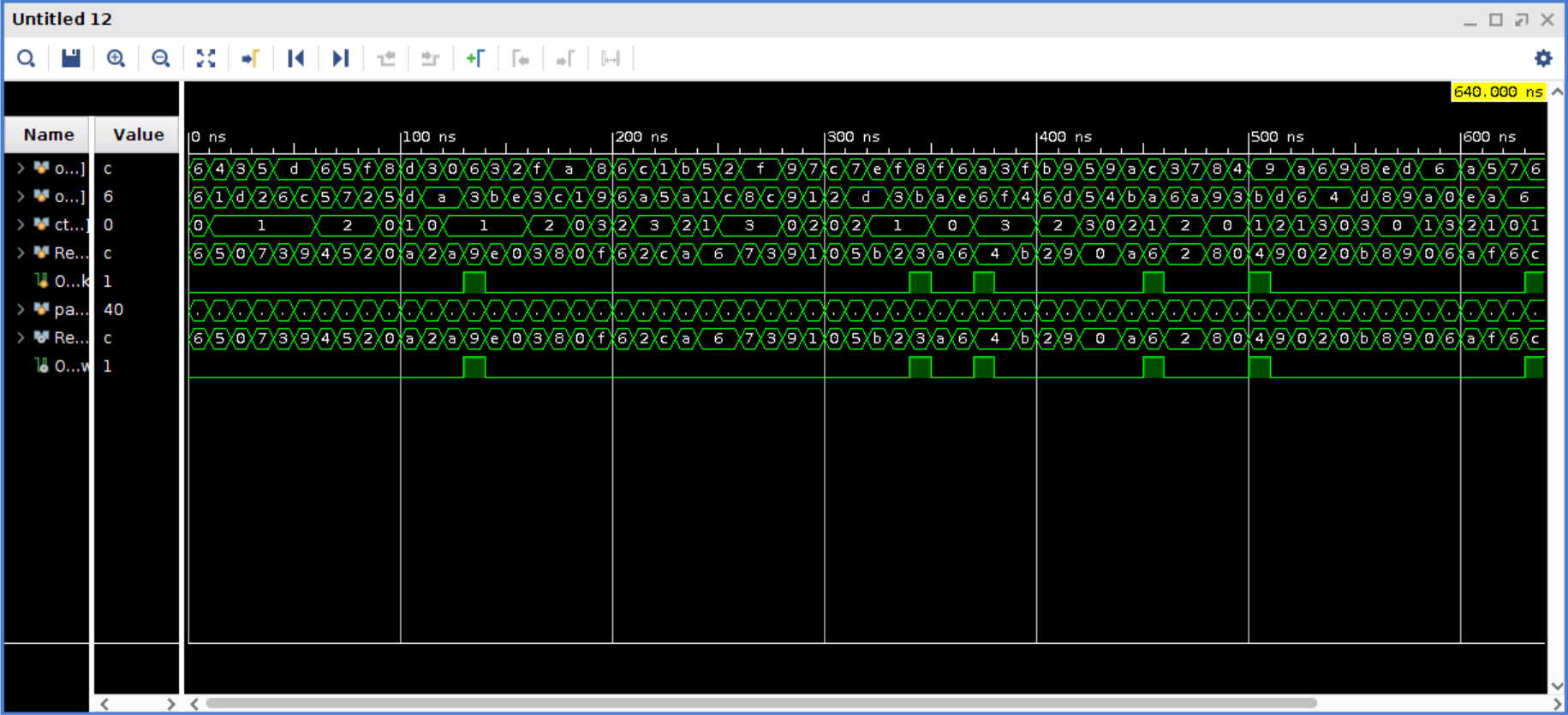
****

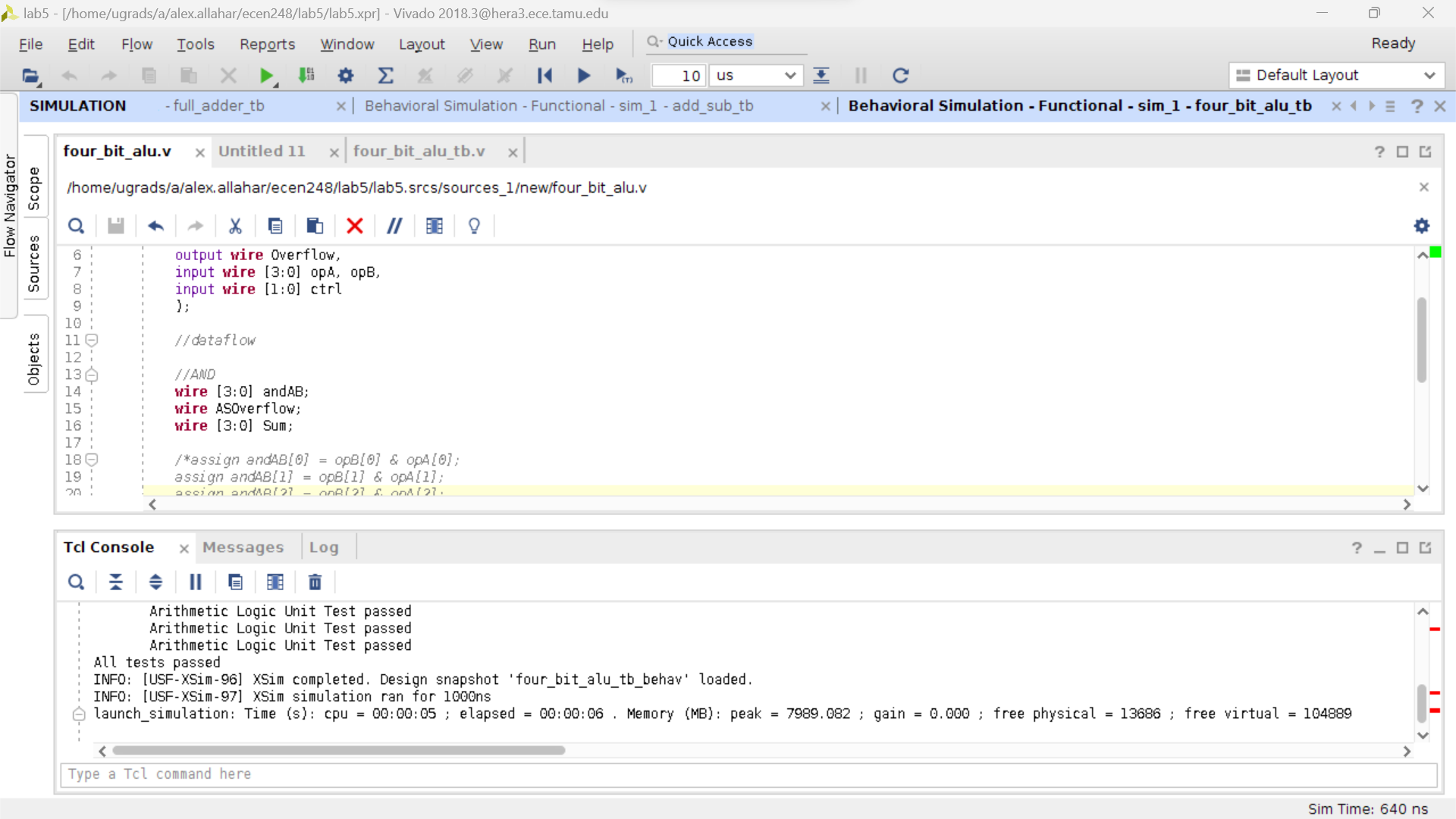
**Figure 8: Full-Adder Waveform and Test Console**

****

****

**Figure 9: Add-Sub Unit Waveform and Test Console**

****

****

**Figure 10: 4-Bit ALU WaveForm and Test Console**

1. Examine the 1-bit, 2:1 MUX test bench code. Attempt to understand what is going on in the code. The test bench uses behavioral Verilog, which reads much more like a programming language. Explain briefly what it is the test bench is doing.

The test bench checks the logic and speed of the Verilog code. The file has a section in which 8 tests are taken to test the logic of each possible A, B, and S combination. This process is the same as us connecting an LED to the output of a circuit breadboarded in to check if the logic is correct.

1. Examine the 4-bit, 2:1 MUX test bench code. Are all of the possible input cases being tested? Why or why not?

All possible input cases are being tested because the console shows that the test bench has tested all possible input cases and showed that they passed.

1. In this lab, we approached circuit design differently compared to previous labs. Compare and contrast breadboarding techniques with circuit simulation. Discuss the advantages and disadvantages of both. Which do you prefer? Similarly, it provides some insight as to why HDLs might be preferred over schematics for circuit representation. Are there any disadvantages to describing a circuit using an HDL compared to a schematic? Again, which would you prefer?

Breadboarding creates more chances for human error and part error. Due to the inconsistency in the different logic gates to the different approaches to wiring a breadboard, it can increase or decrease propagation delay. However, due to the perfect conditions of Verilog simulation, the outcome of the digital circuit could be a more perfectly realistic representation of a real digital circuit. Still, to design bigger circuits that are still being tested and tweaked the use of Veriolg is much more efficient and reasonable. However, when the digital circuit is nearly complete in the test a switch to a physical schematic is important to see how the Verilog translates into the real world.

1. Two different levels of abstraction were introduced in this lab, namely structural and dataflow. Provide a comparison of these approaches. When might you use one over the other?

Structural abstraction focuses on the gates and the function in the circuits; whereas, the dataflow uses the assignment to show where the data is next in the circuit. For circuits with many different operations taking place at multiple levels in a circuit the dataflow approach allows for easy reading and describing the circuit. The structure could be better for a simple circuit with few gates and not many gate levels.