**Lab 6: Introduction to Logic Simulation and Verilog**

Faizan Bangash

TA : Saurav Kumar Sahu

ECEN 248 – 302

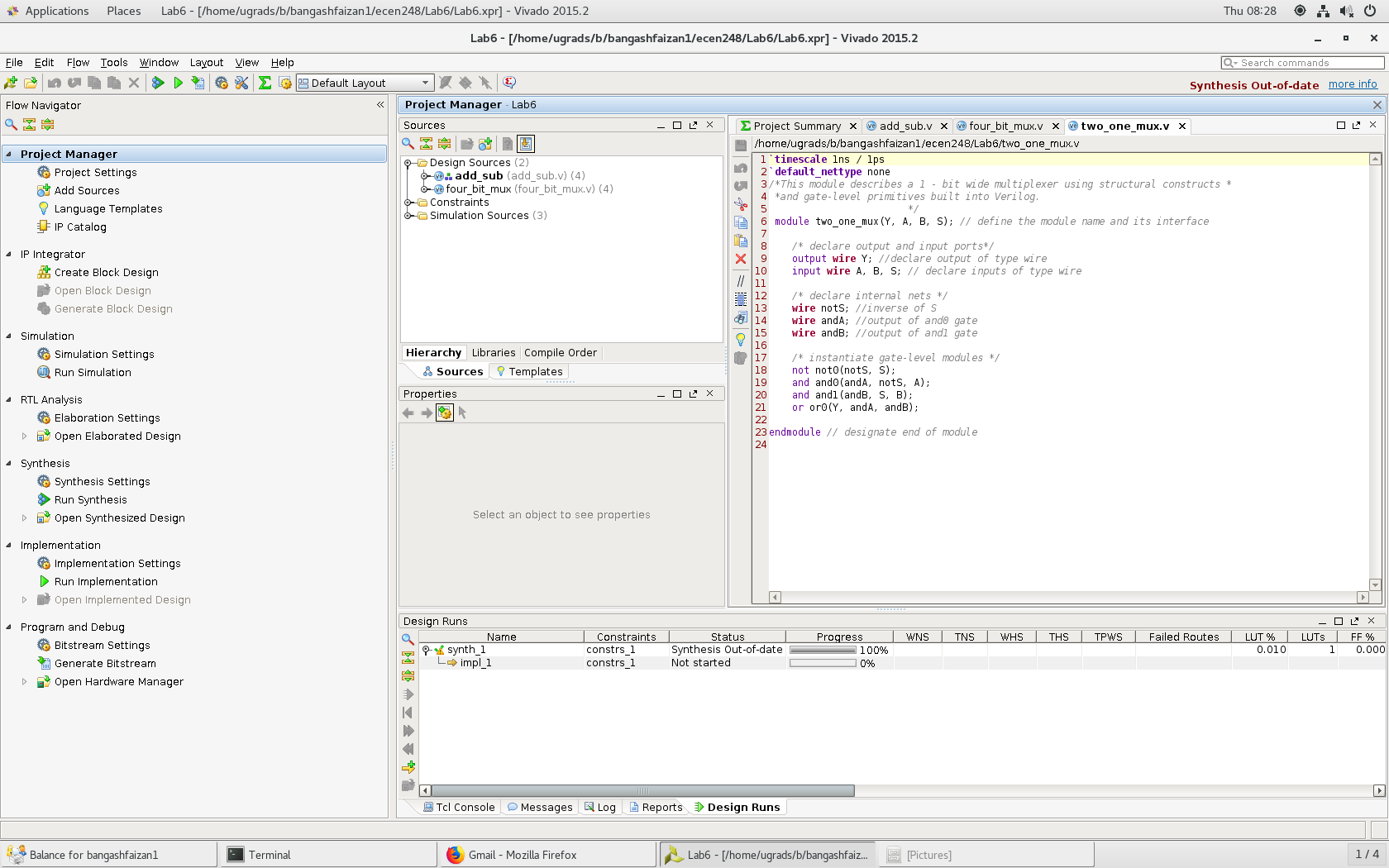
July 12, 2018.

**Objectives** –

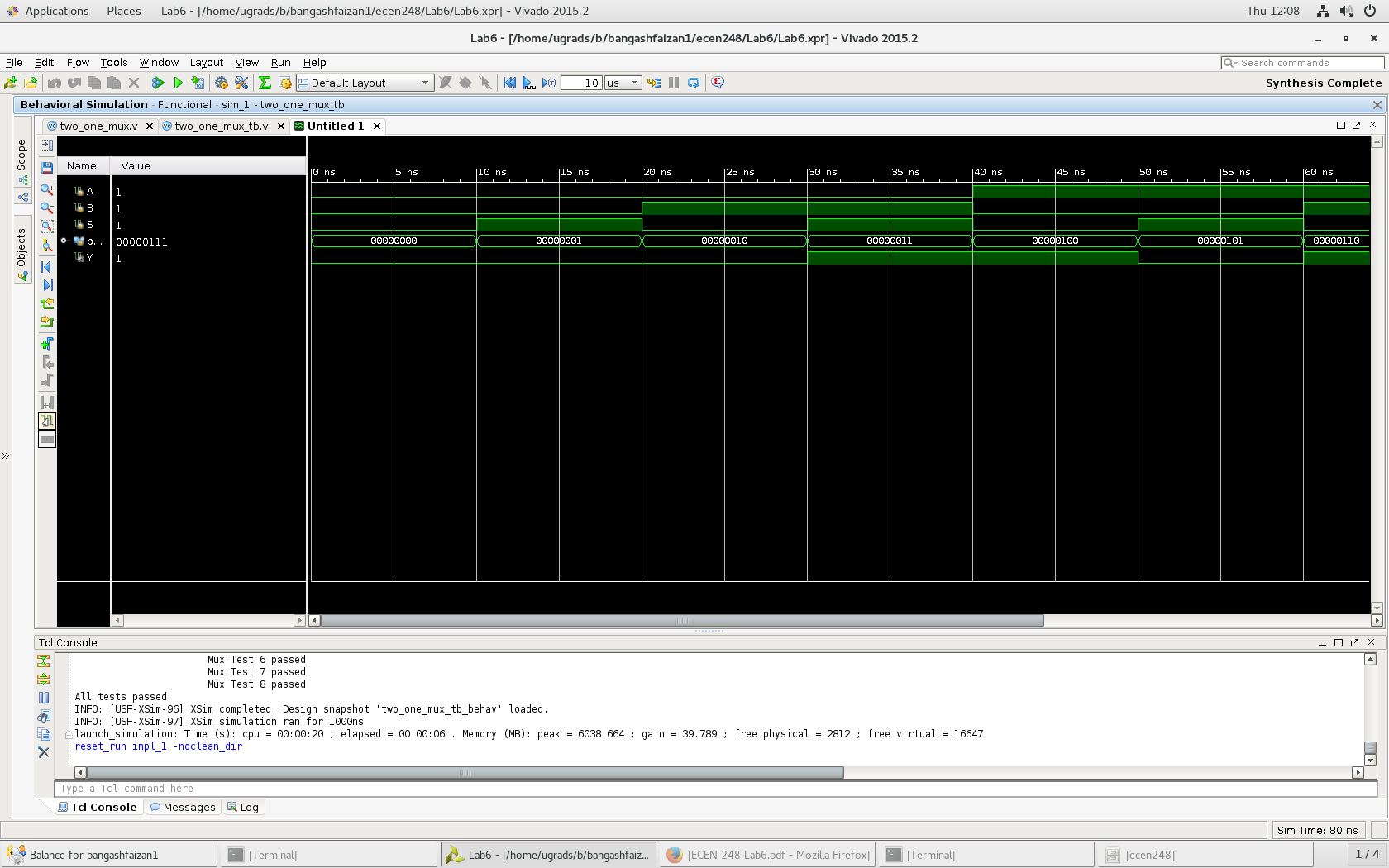
In this lab, I learned how to start Verilog as well as how to make basic programs and implement the programs by simulating other programs.

**Design** –

In this lab, we started by making a 2:1 MUX program and simulating it to make sure everything worked ok. The code for the 2:1 MUX can be seen.

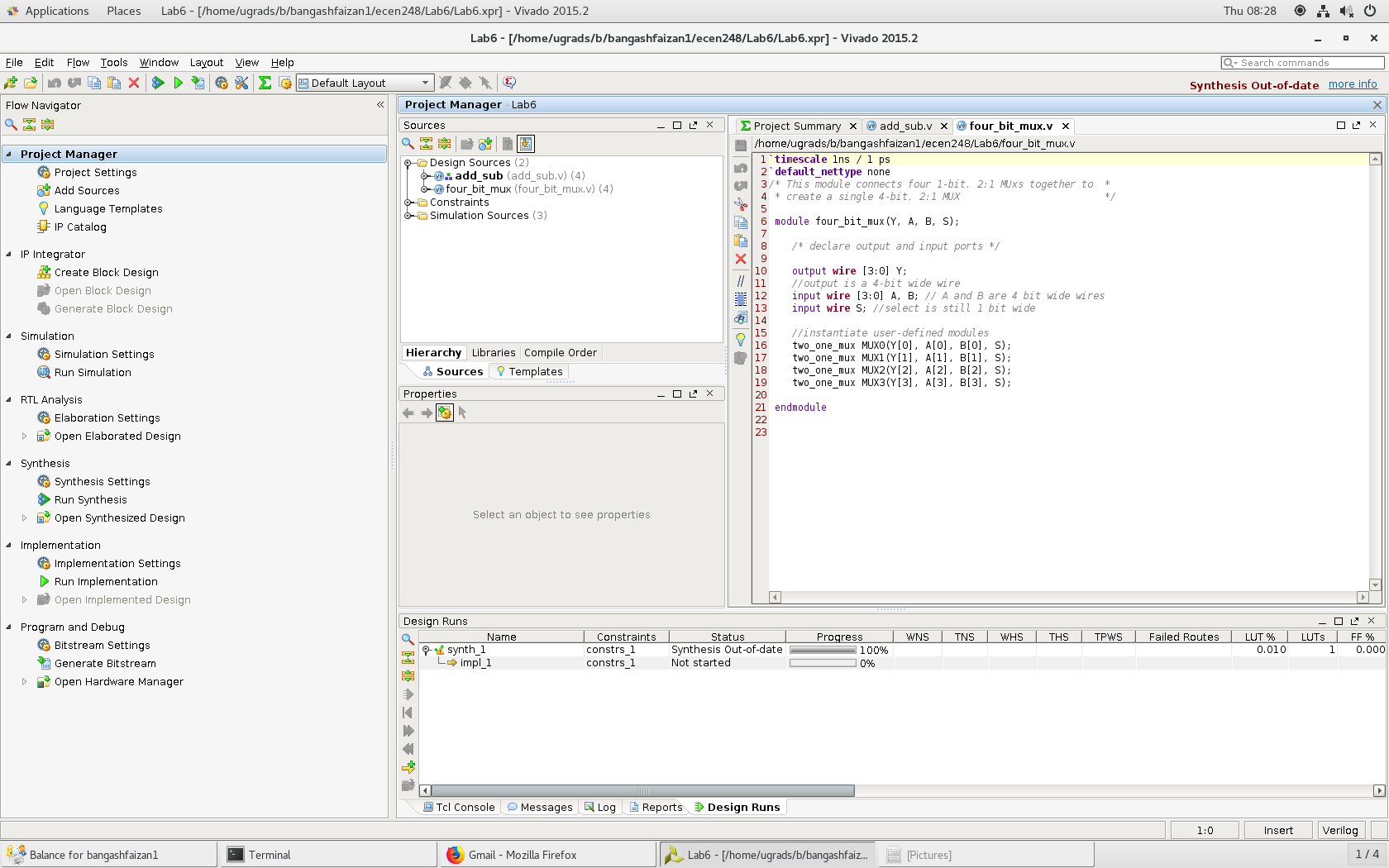


Code for the 2:1 MUX

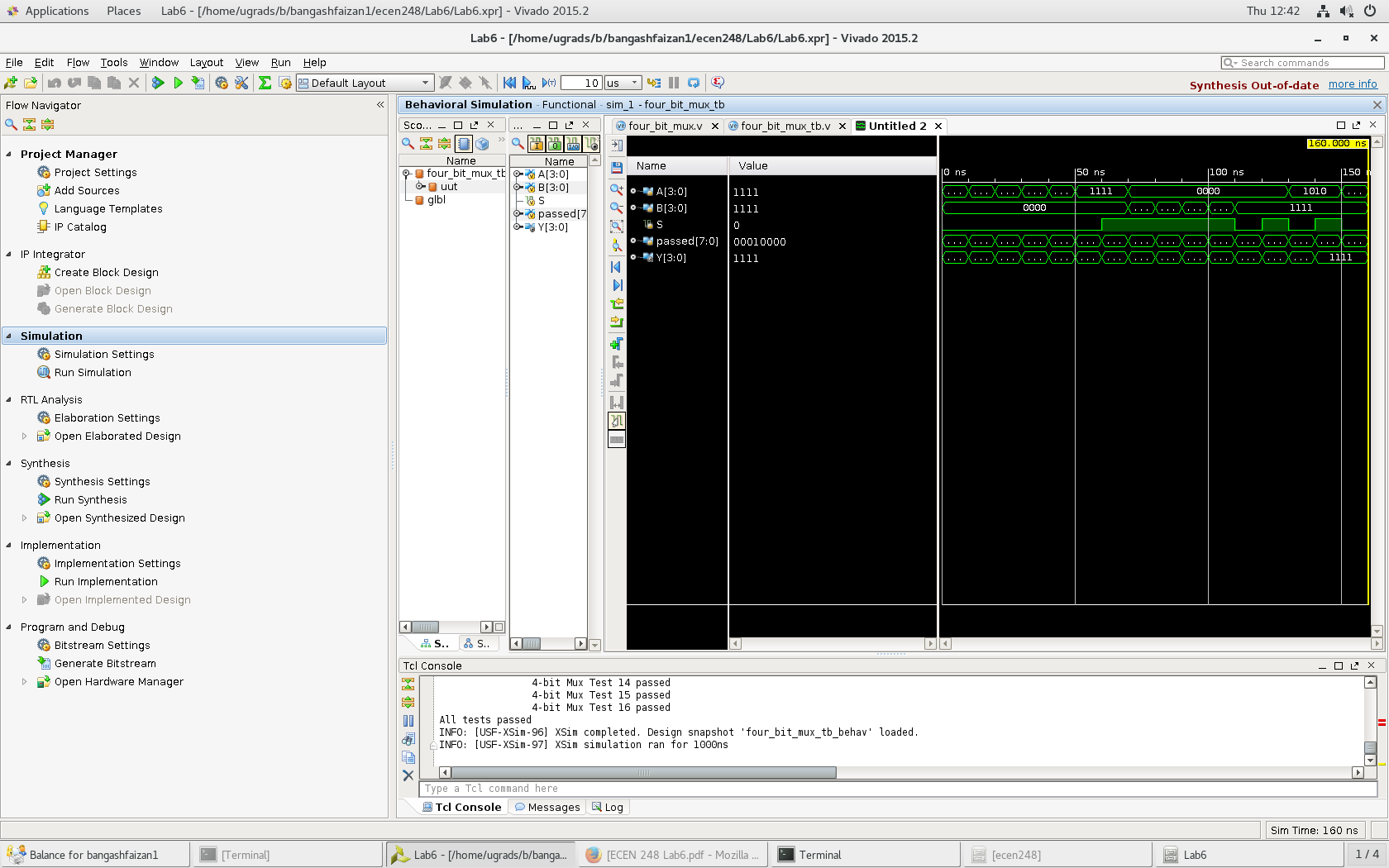


Results for the 2:1 MUX

After verifying the 2:1 MUX worked, we then created a 4-bit mux and verified that it worked as well. The code and results can be seen.

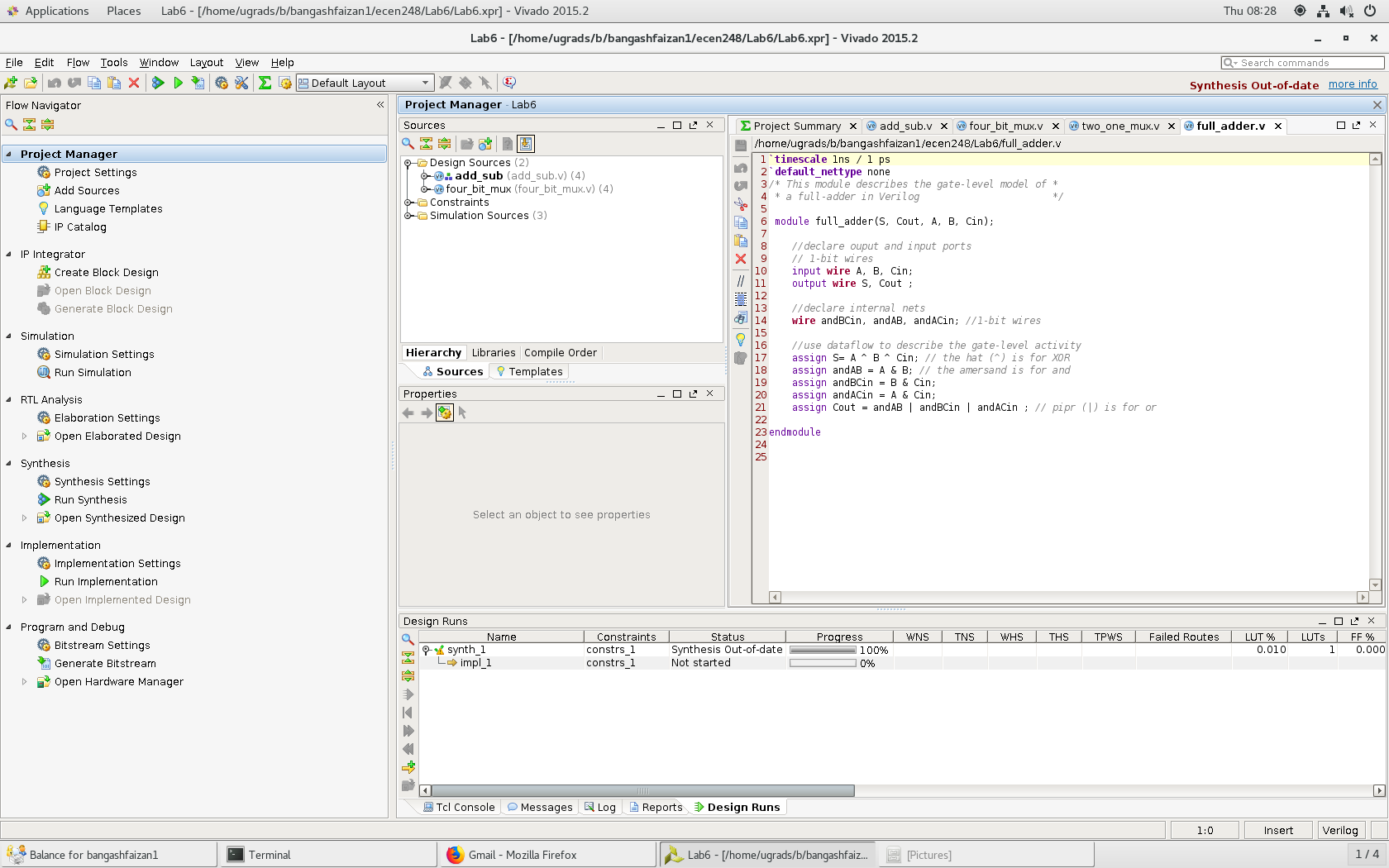


Code for the 4:1 MUX

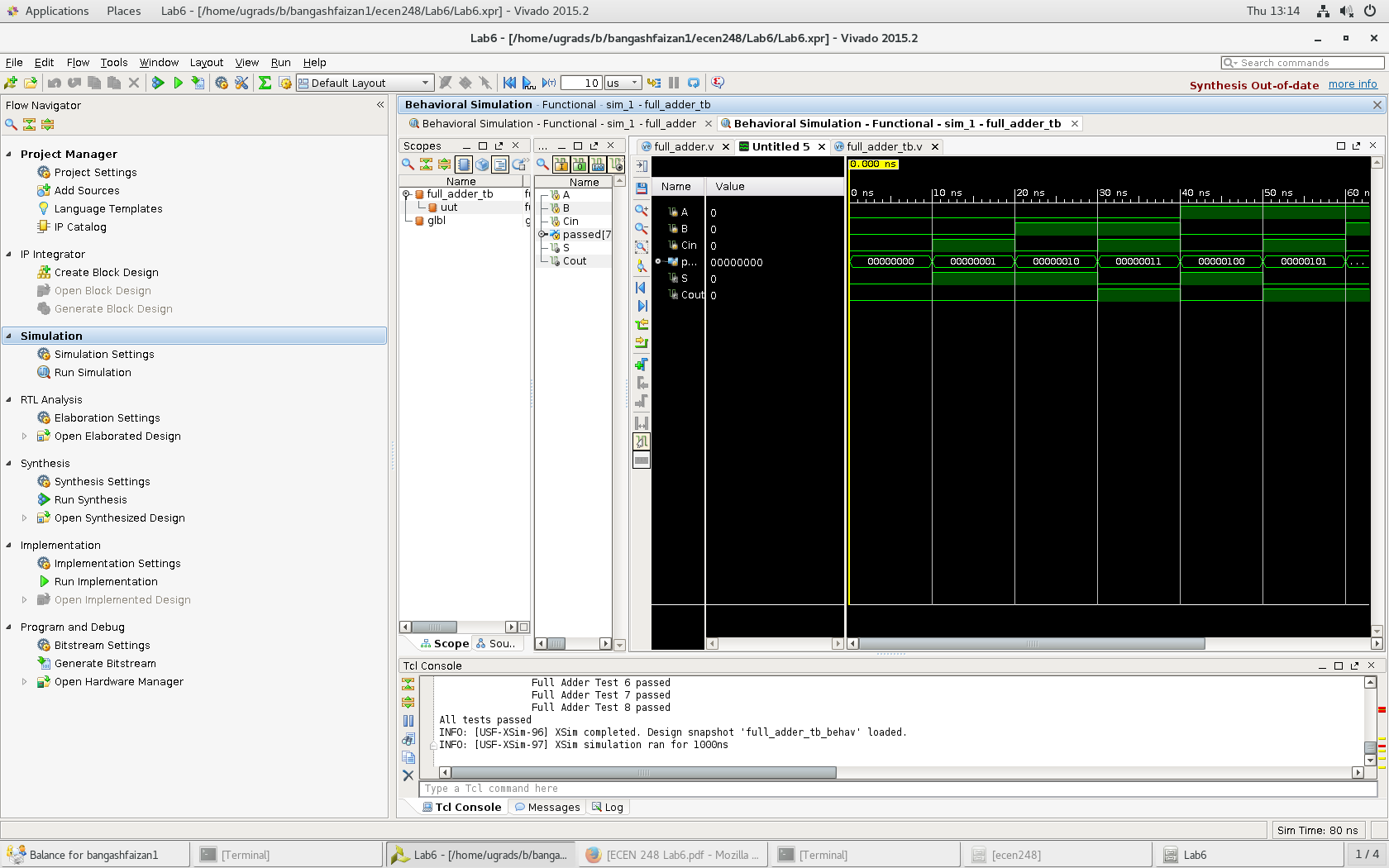


Results for the 4:1 MUX

After verifying the 4-bit MUX worked, we then created a simple full adder used in previous labs to add two numbers. The full adder code can be seen.

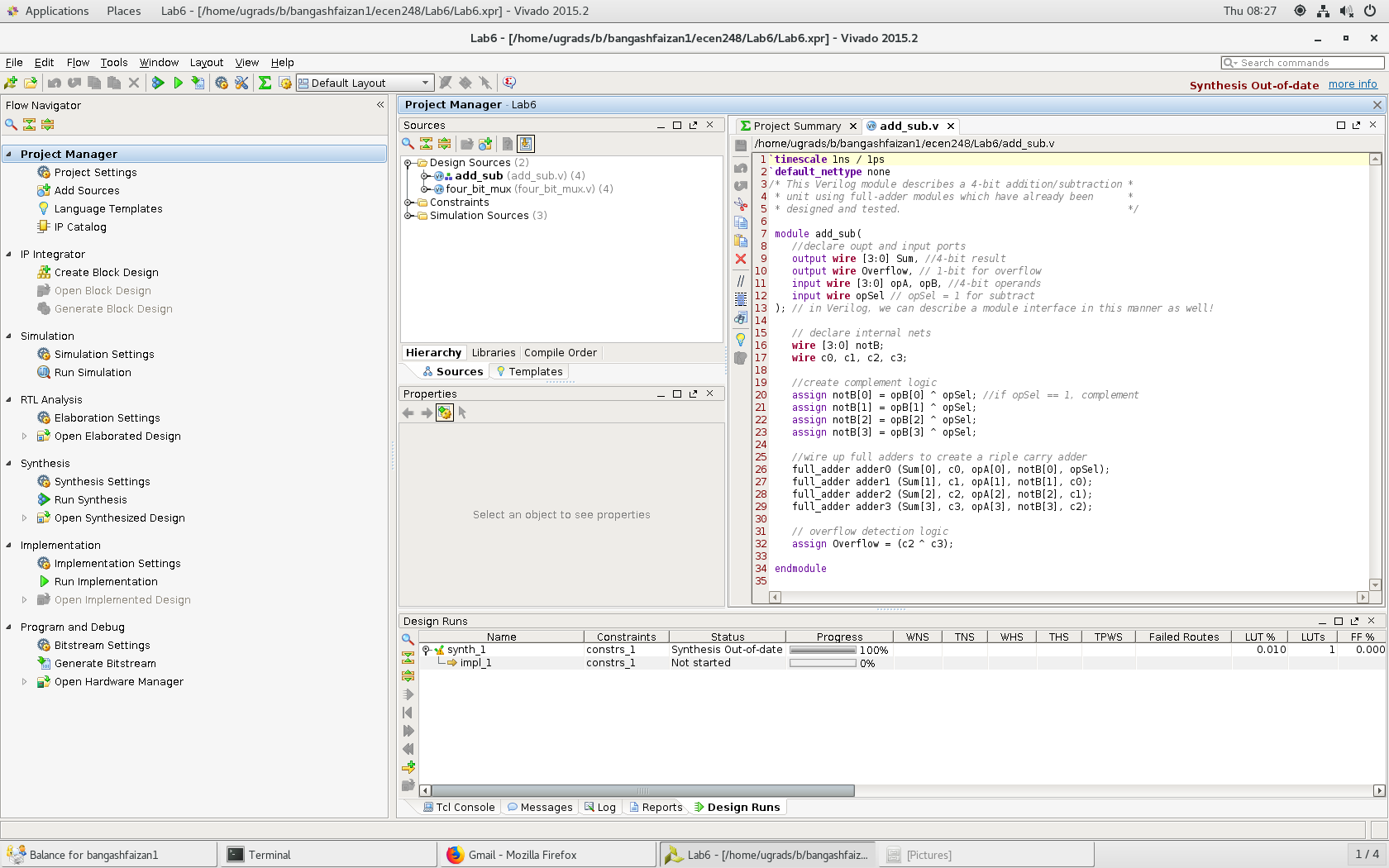


Code for the full adder

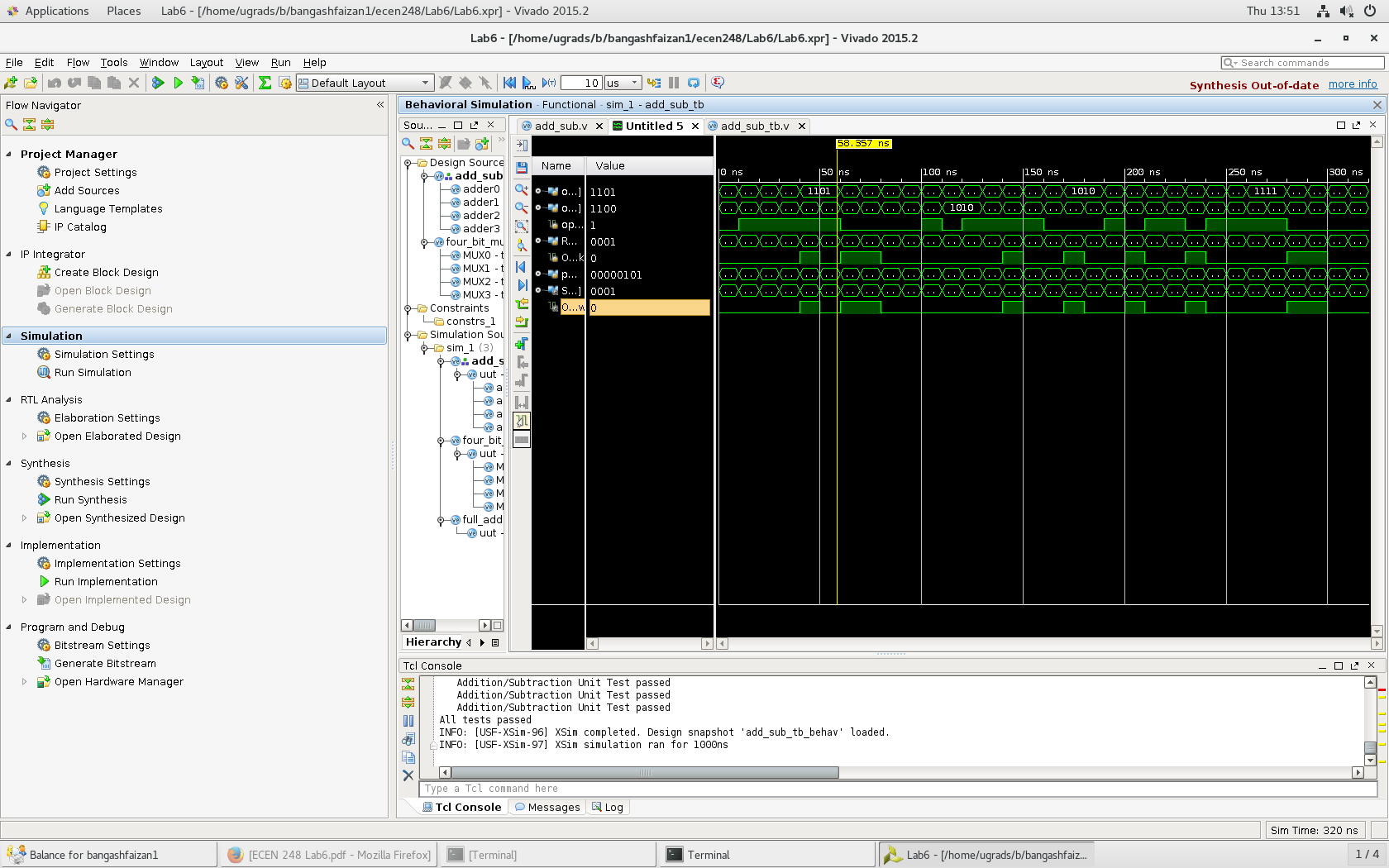


Results for the full adder

After making sure the full adder worked, we then created an add/subtract program that would add or subtract two numbers based on an operation input and implemented the full adder code. The code for this program as well as its results can be seen.

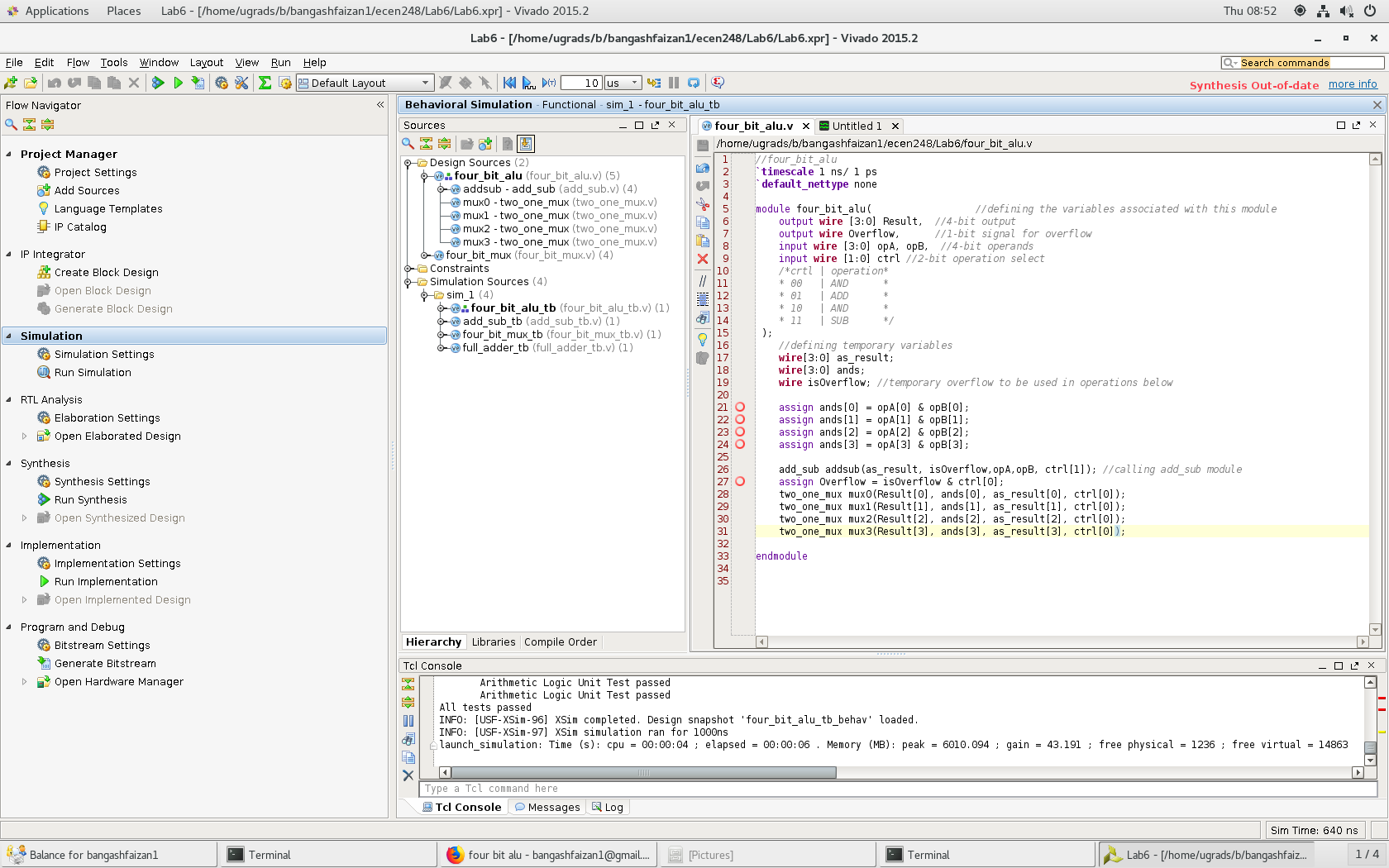


Code for the Add/Subtractor

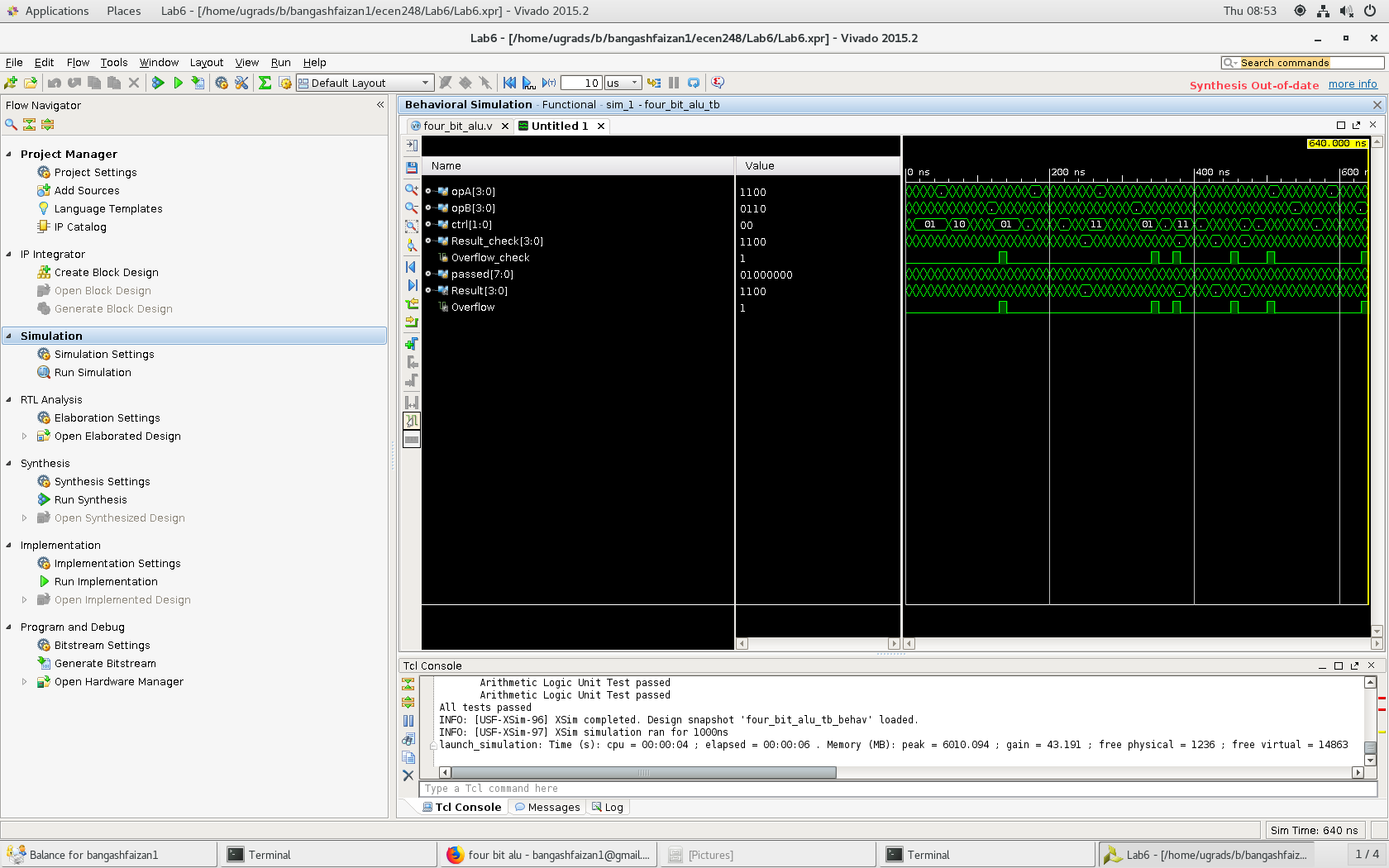


Results for the Add/Subtractor

Finally, we put everything together and we get the four bit ALU working. The ALU will read in a control value as well as two 4-bit numbers and either output the ANDs or the addition/subtraction. The code for the four bit ALU as well as its results can be seen below.



Code for the four-bit ALU



four-bit ALU results

**Results –**

During the testing of my programs, all of the programs passed every test the first time except for the final four-bit ALU. The only issue I ran into on the four-bit ALU was calculating the overflow, however, this was easily fixed. Overall, everything worked quite well and the tests were easy to pass. The only change I would make is using the 4:1 MUX in figure 9 instead of two 2:1 MUX.

**Conclusion –**

Overall, this lab was a very good intro into Verilog. It was able to teach me the basics as well as give me enough time to figure out how to log in and set up the software. The basic coding knowledge I learned in this lab will be extremely helpful in future labs where the circuits will become more advanced.

**Questions –**

*3. Examine the 1-bit, 2:1 MUX test bench code. Attempt to understand what is going on in the code. The test bench is written using behavior Verilog, which will read much like a programming language. Explain briefly what it is the test bench is doing.*

The test bench manages the outputs that show up in the simulation screen and keeps track of which tests passed and which tests failed.

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

No because for a 4 bit mux there are 16 tests and for a 1 bit mux there are 8 tests, therefore not all possible tests are being run.

*5. In this lab, we approached circuit design in a different way compared to previous labs. Compare and contrast breadboarding techniques with circuit simulation. Discuss the advantages and disadvantages of both. Which do you prefer? Similarly, provide 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.*

The use of Verilog over breadboarding has made the lab much more engaging and much less tedious. It involves more thought and computer skills than the breadboard however the pain of hand wiring hundreds of connections does not incline me to say that the breadboard is the simpler approach.

*6. 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 is the use of individual gates to create a Verilog code. Dataflow allows the user to implement large circuits by emphasizing data flow and not requiring the code for every individual gate in a large logic tool. Thus dataflow is the best choice for large logical tools and structural is a good choice for small logic tools