**ECEN 248 - Lab Report**

**Lab Number: 7**

**Lab Title: Introduction to Sequential Logic**

**Section Number: 519**

**Student’s Name: Alex Allahar**

**Student’s UIN: 9280098686**

**Date: 11/01/2023**

**TA: Yi Deng**

**Objectives:**

The purpose of this lab is to introduce sequential logic circuits. By starting with latches and flip-flops. Next, the idea of combining flip-flops and combinational logic to simulate synchronous logic. Finally, the use of delay will be observed.

**Design:**

To begin, create a project called lab7. The first module will be a sr\_latch. Using the code provided begins to describe an SR latch at the gate level. Once done, copy the test bench file corresponding to this module from the course directory and simulate. To view the internal signals of the nandSEN and nandREN on the waveform add them from the uut file into the objects panel of the waveform. Chance all the delays from 2 to 4 and resimulate. Next, design a D-latch using structural Verilog. Once done, copy the test bench file corresponding to this module from the course directory and simulate. Using this module create a D flip-flop module that calls on the D latch module. Once done, copy the test bench file corresponding to this module from the course directory and simulate. Add internal signals Qm, and notClk to the waveform and rerun the simulation. Recreate the d latch using behavioral modeling to allow the module to be synthesized. Using the starter code in the manual to create this module, and then create the D flip-flop behavioral design source. With these two sources done, add both the test bench files with corresponding names from the course directory and simulate one at a time.

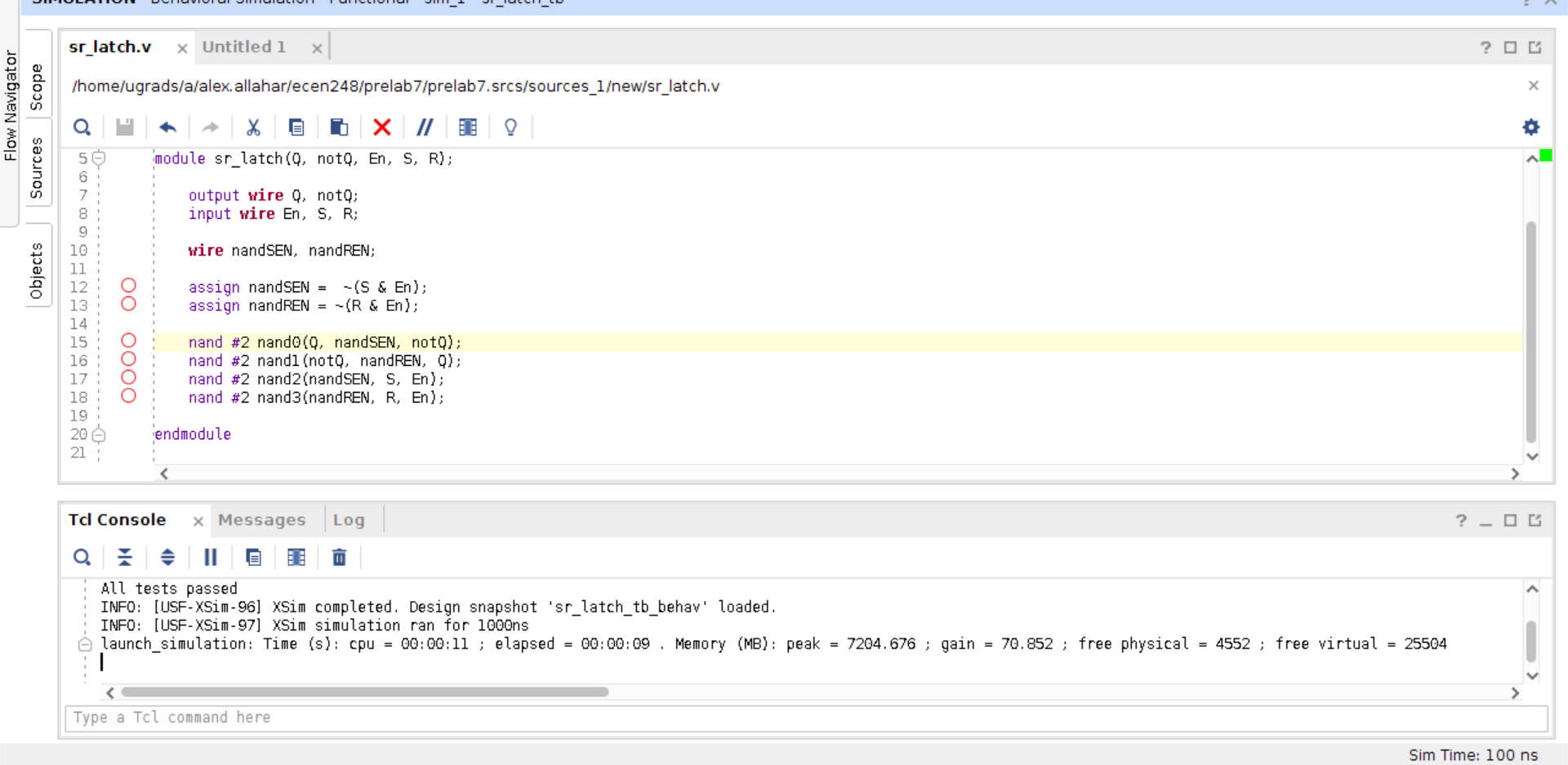
For the next part of the lab, copy the full-adder design in lab5 into the project. Using this design a module adder\_2bit using the code in the lab manual as a starting point. Once done, copy the test bench file corresponding to this module from the course directory and edit it to ensure the test bench file is working correctly. Then simulate. Create a new module called adder\_synchronous using the code in the lab manual as a starting point. Once done, copy the test bench file corresponding to this module from the course directory and simulate. If the module can simulate, then change the CLOCK\_PERIOD in the test bench file from 40 to 18, and resimulate the module. Keep increasing this parameter until the module passes all its tests.

**Conclusion:**

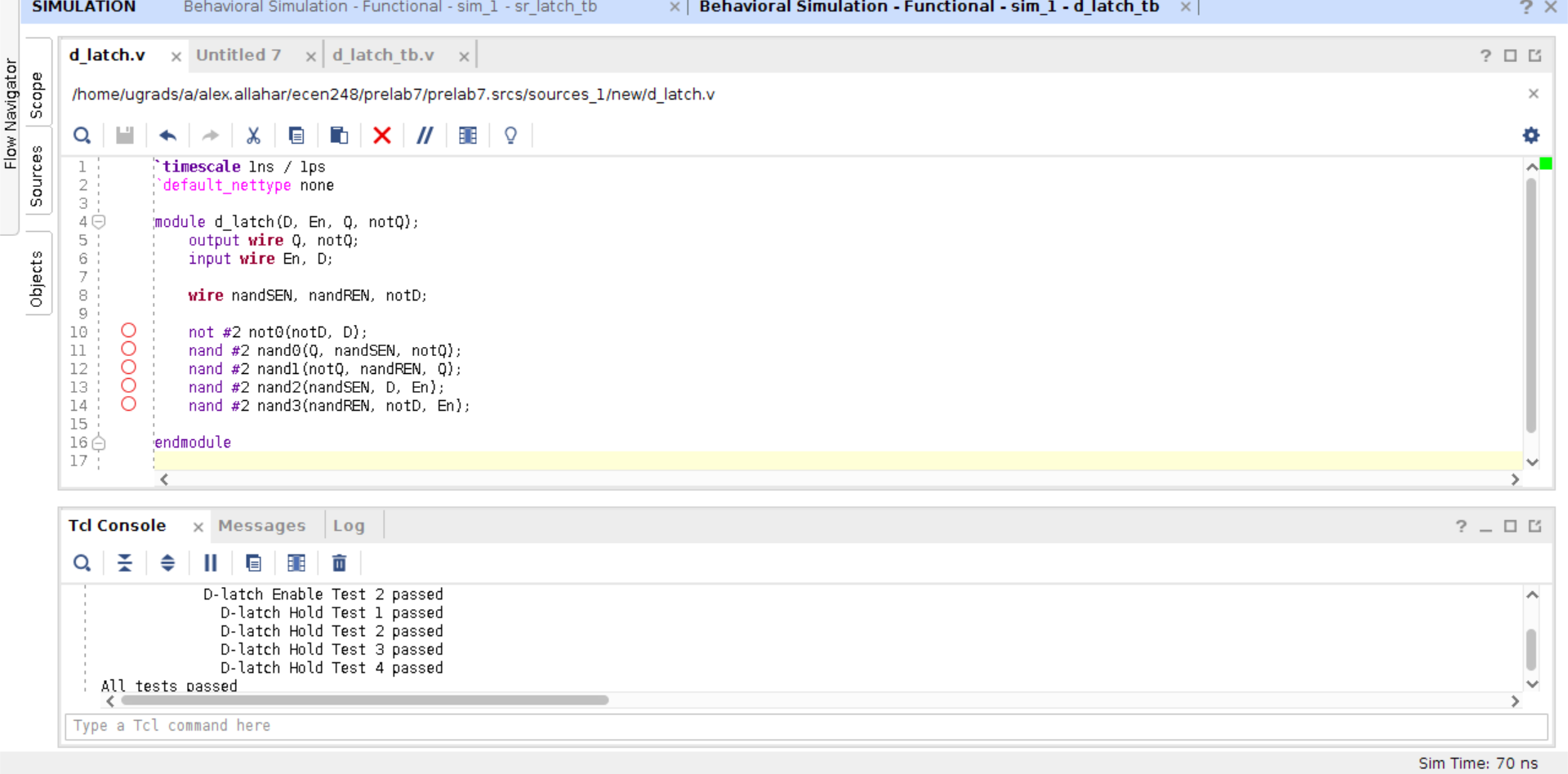
In this lab, I designed various latches and flip-flops. After I designed adders and incorporated synchronicity and delay to simulate synchronous logic.

**Post-lab Deliverables:**

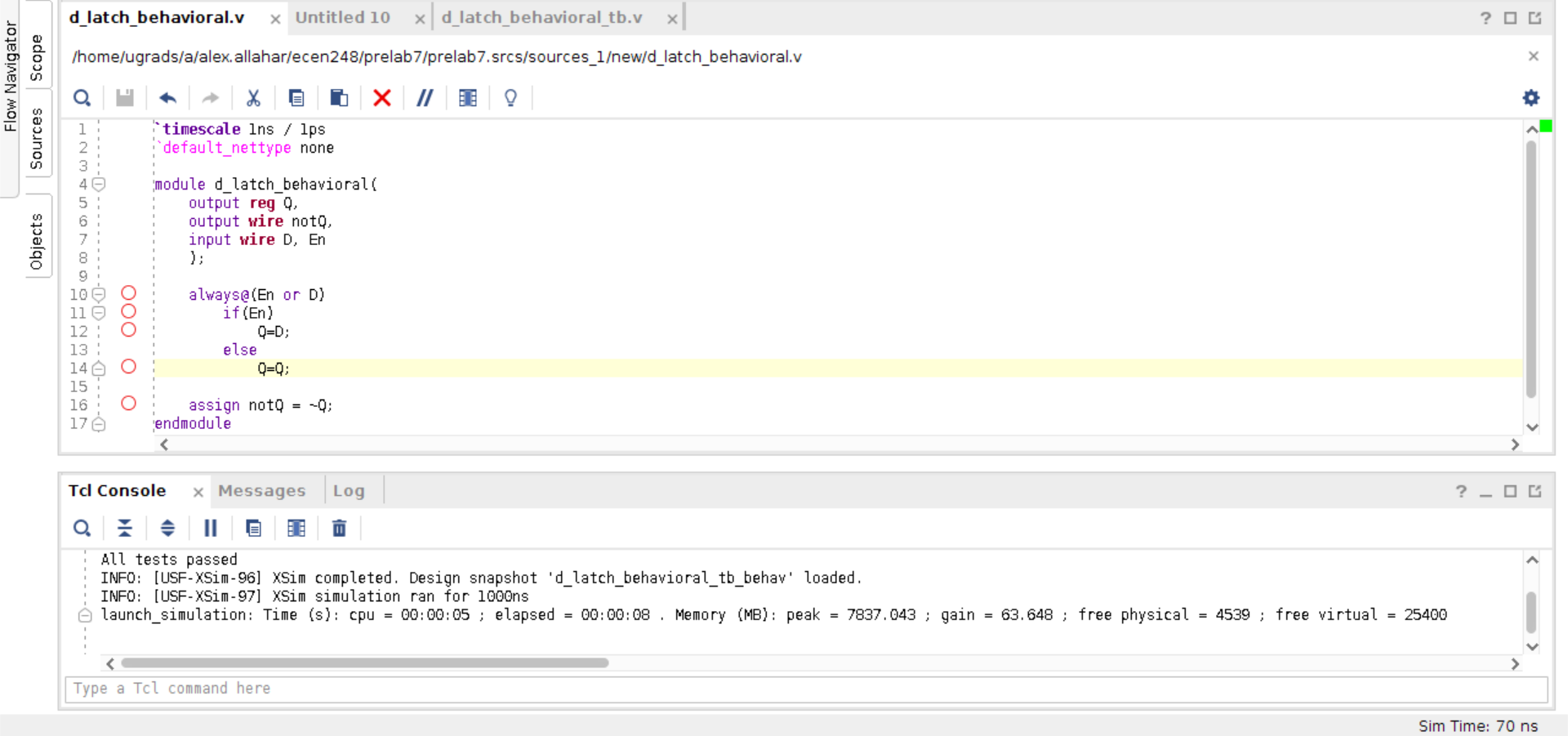
**1. Source Code**

****

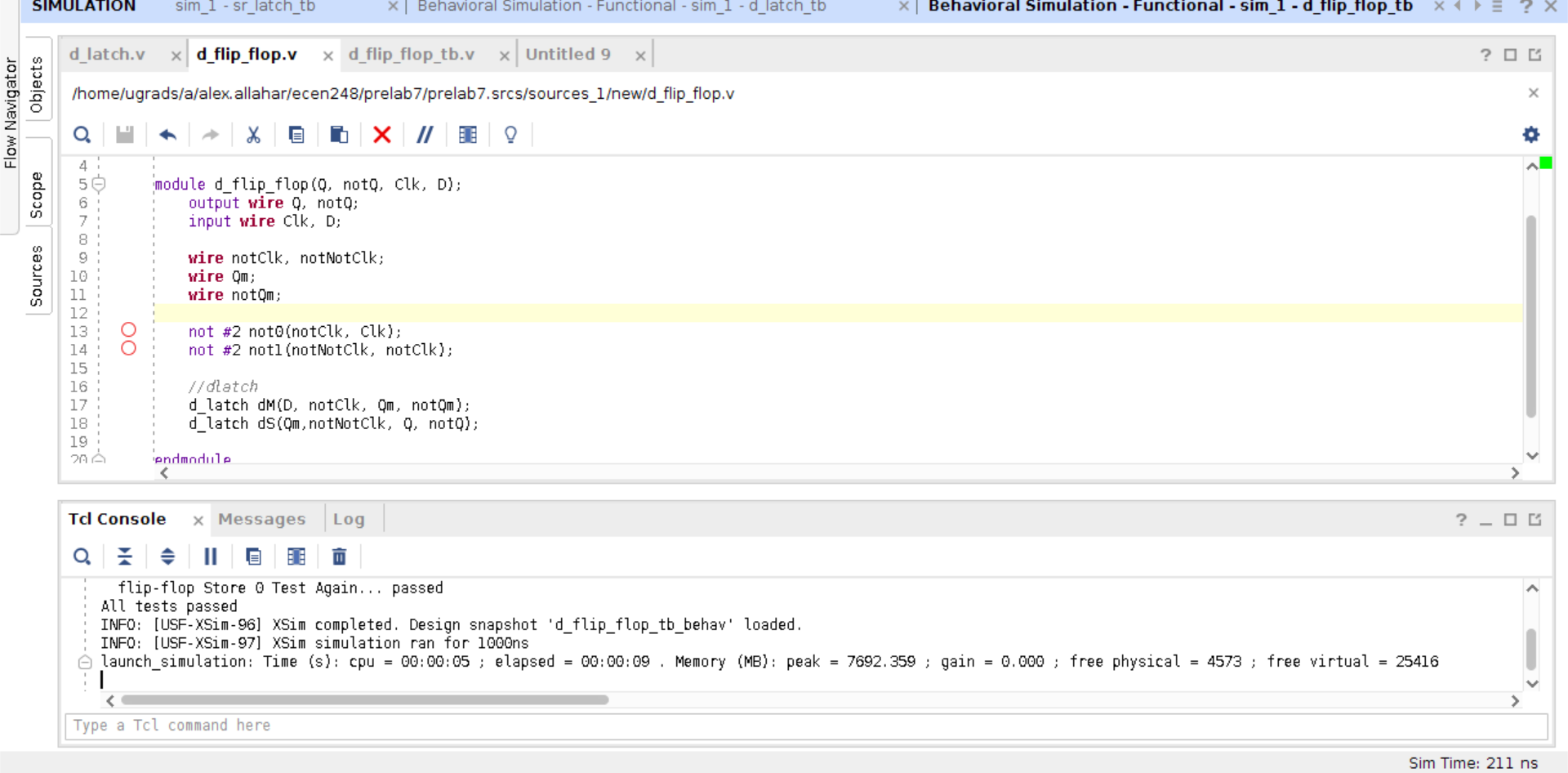
**Figure 1: SR Latch Code**

****

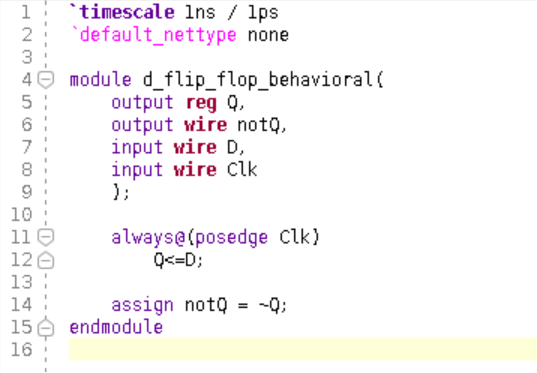
**Figure 2: D Latch Code**

****

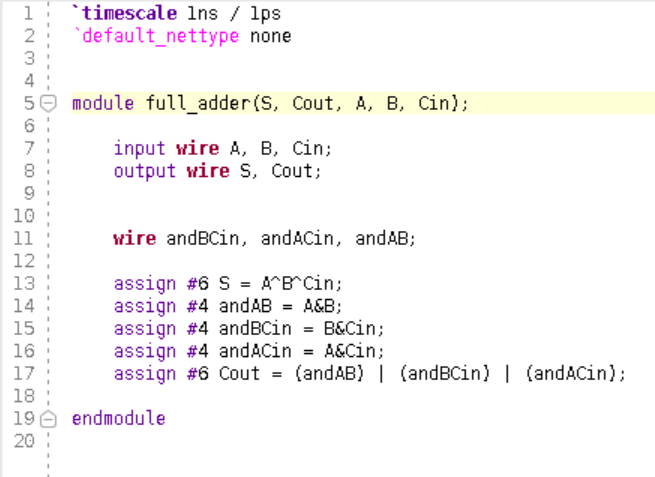
**Figure 3: D Latch Behavioral Code**

****

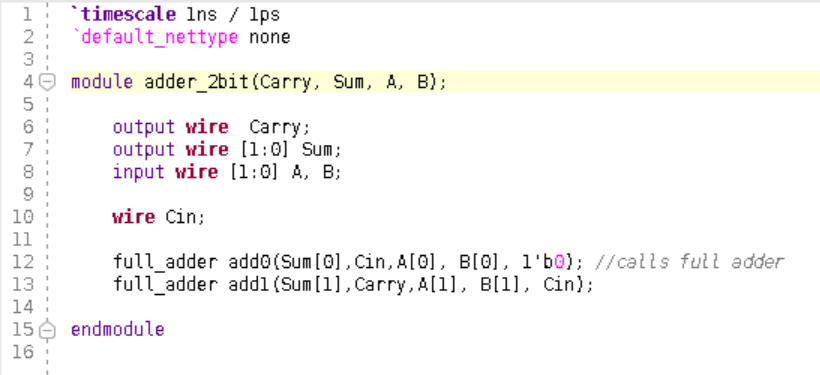
**Figure 4: D Flip Flop Code**

****

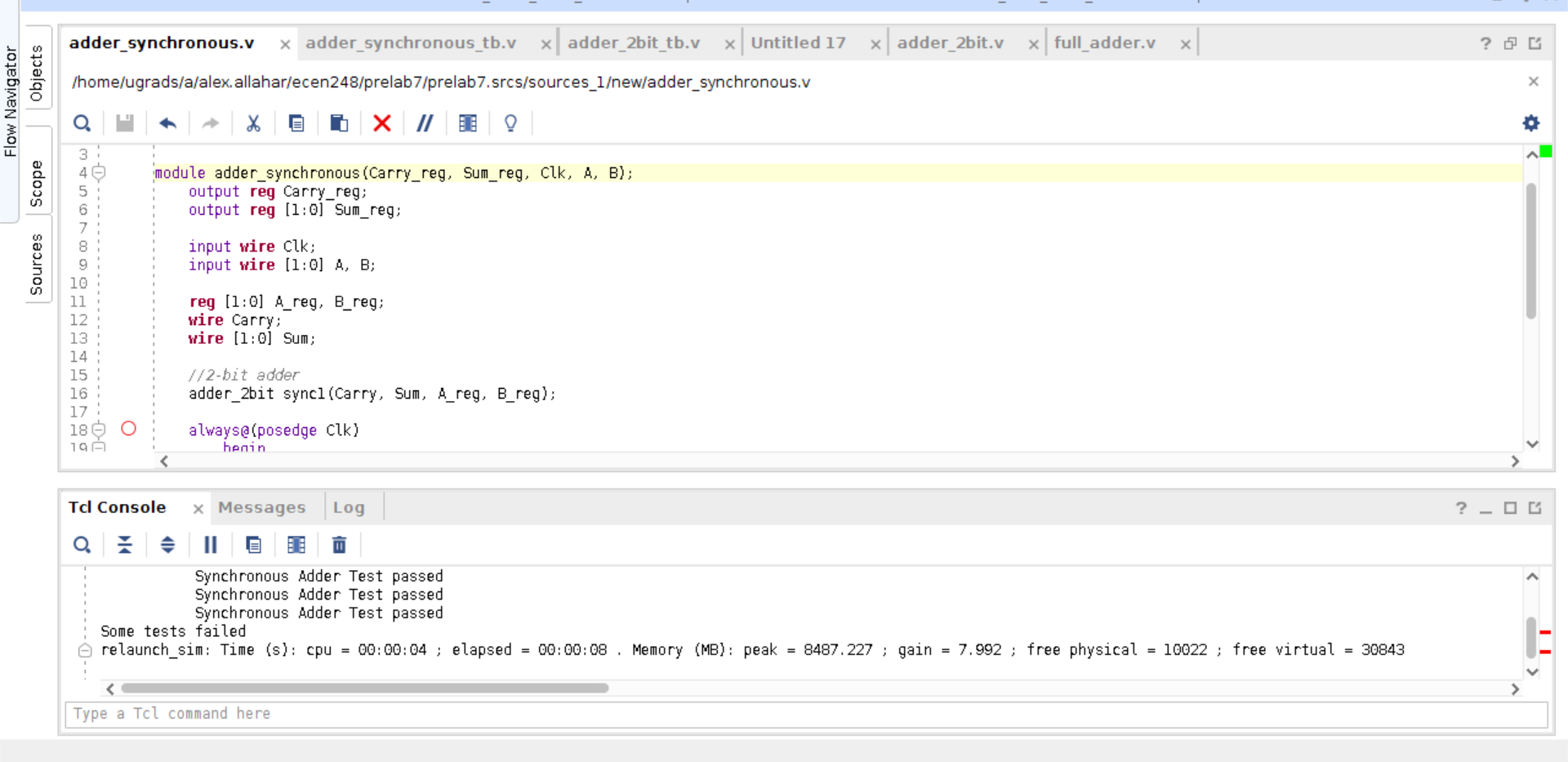
**Figure 5: D Flip Flop Behavioral Code**

****

**Figure 6: Full Adder Code**

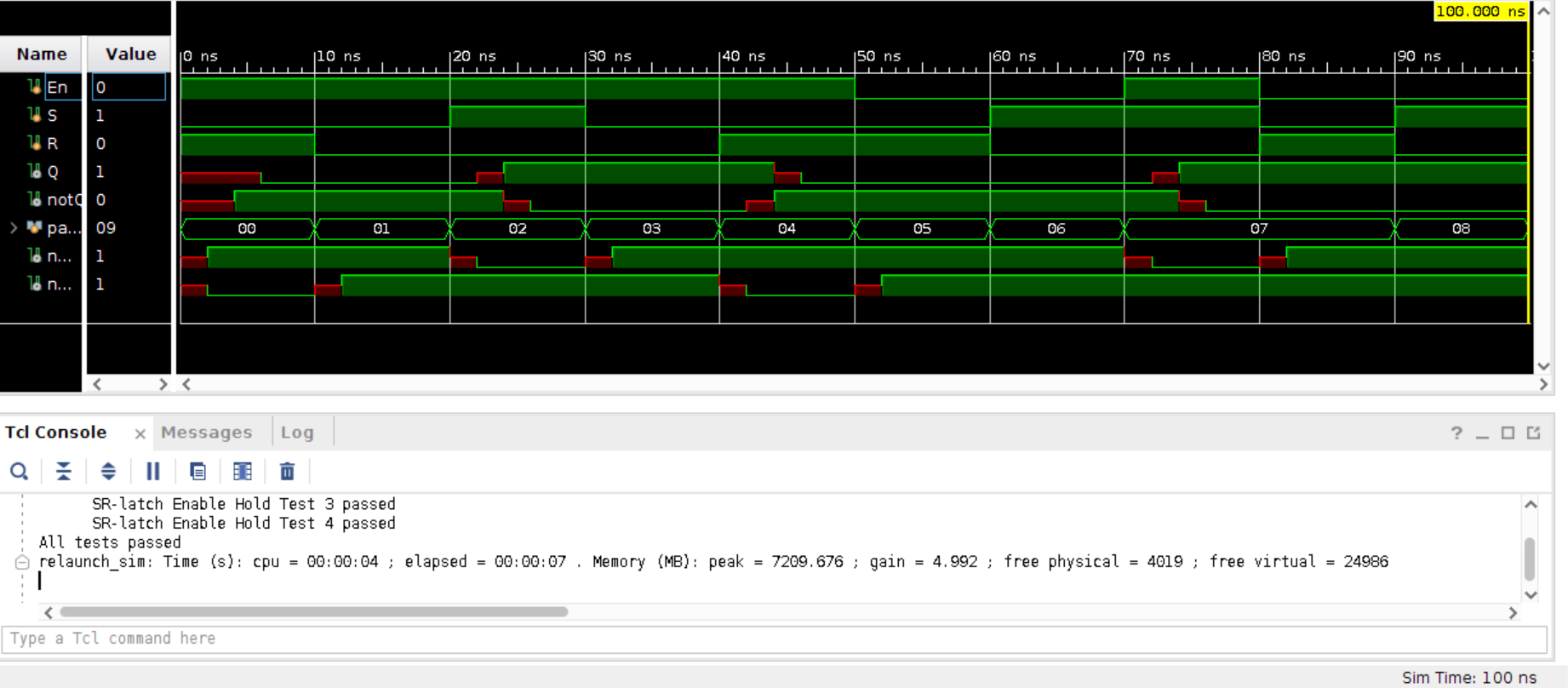
****

**Figure 7: 2-bit Adder Code**

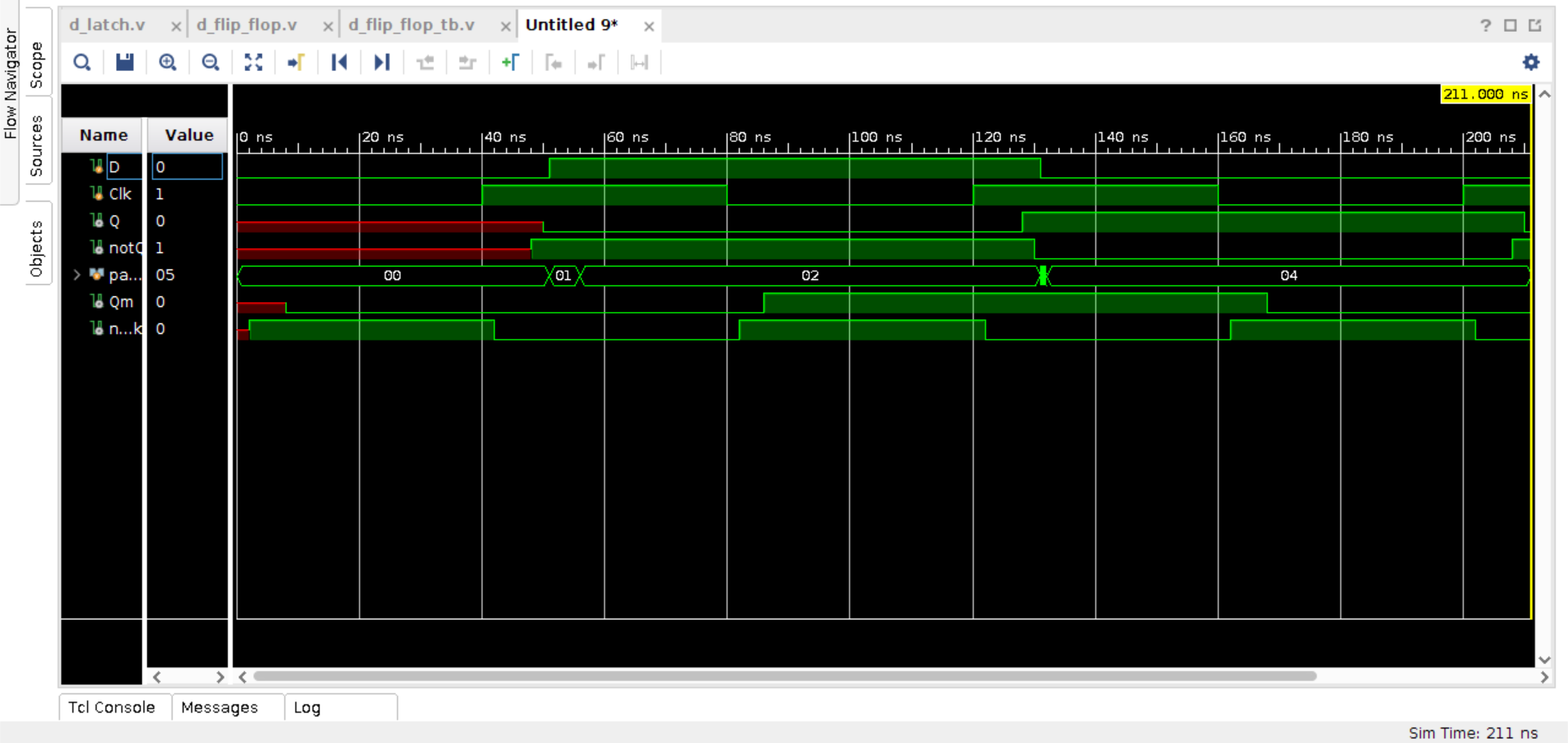
****

**Figure 8: Synch Adder Code**

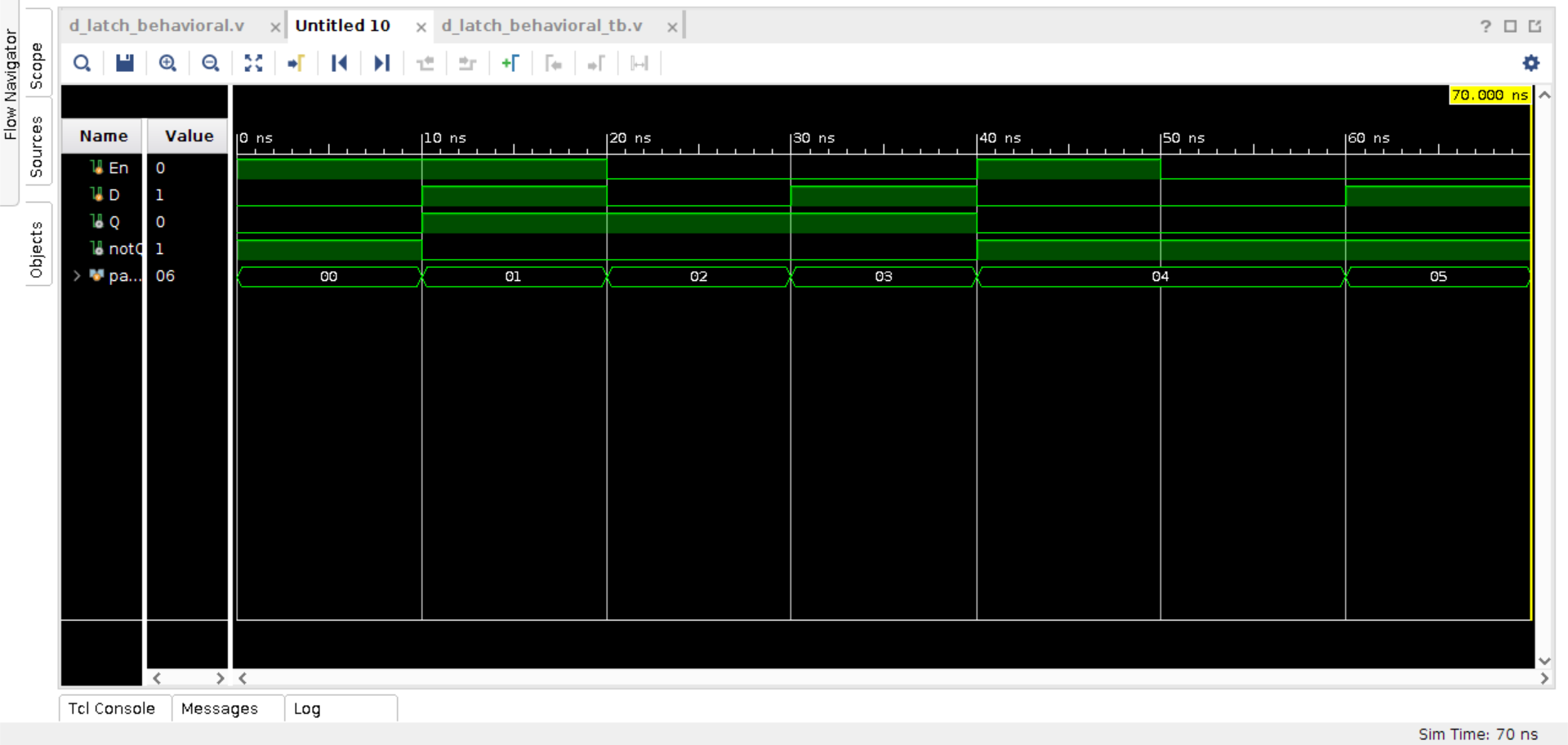
**2. Waveforms**

****

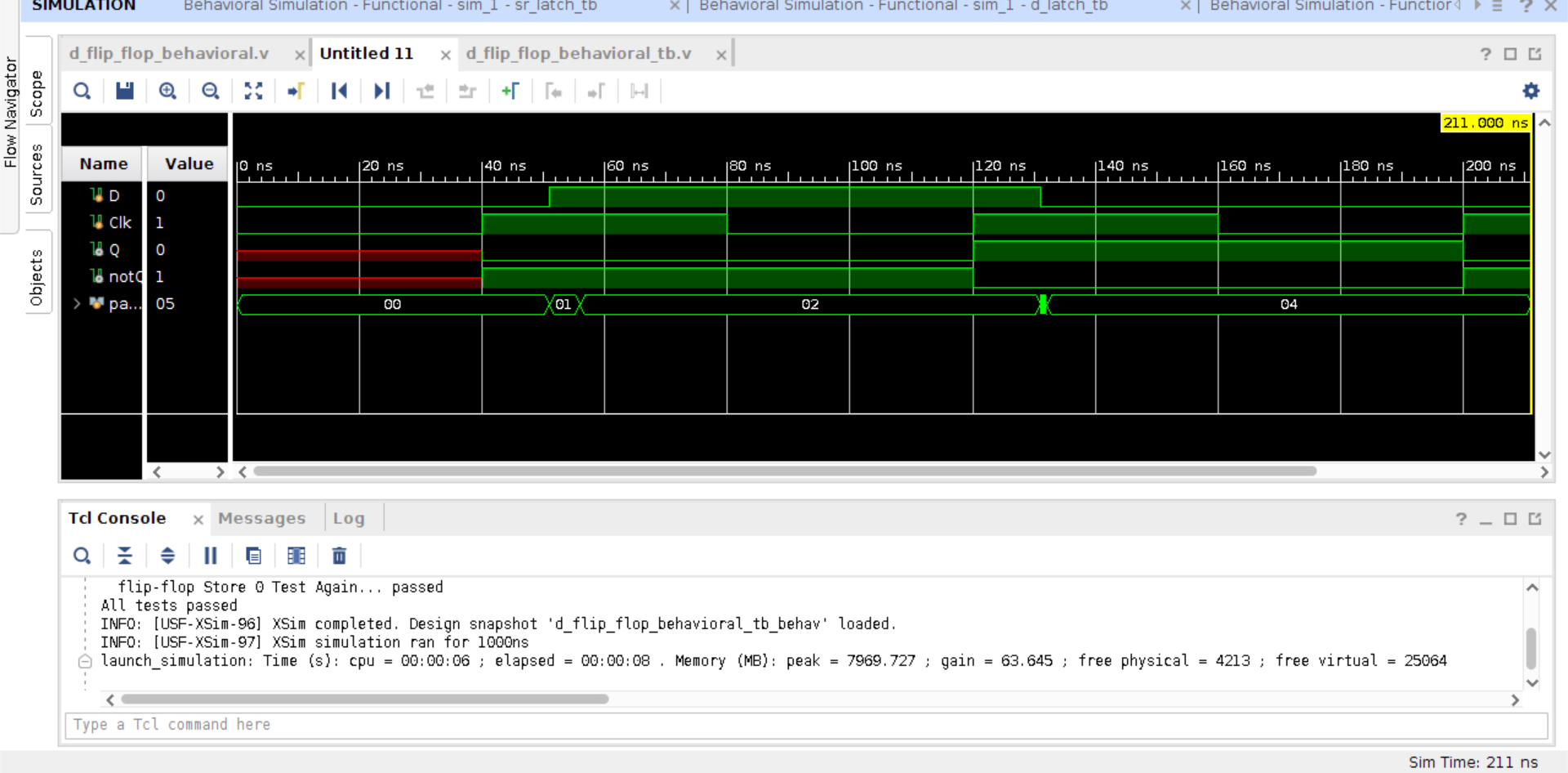
**Figure 9: SR Latch waveform**

****

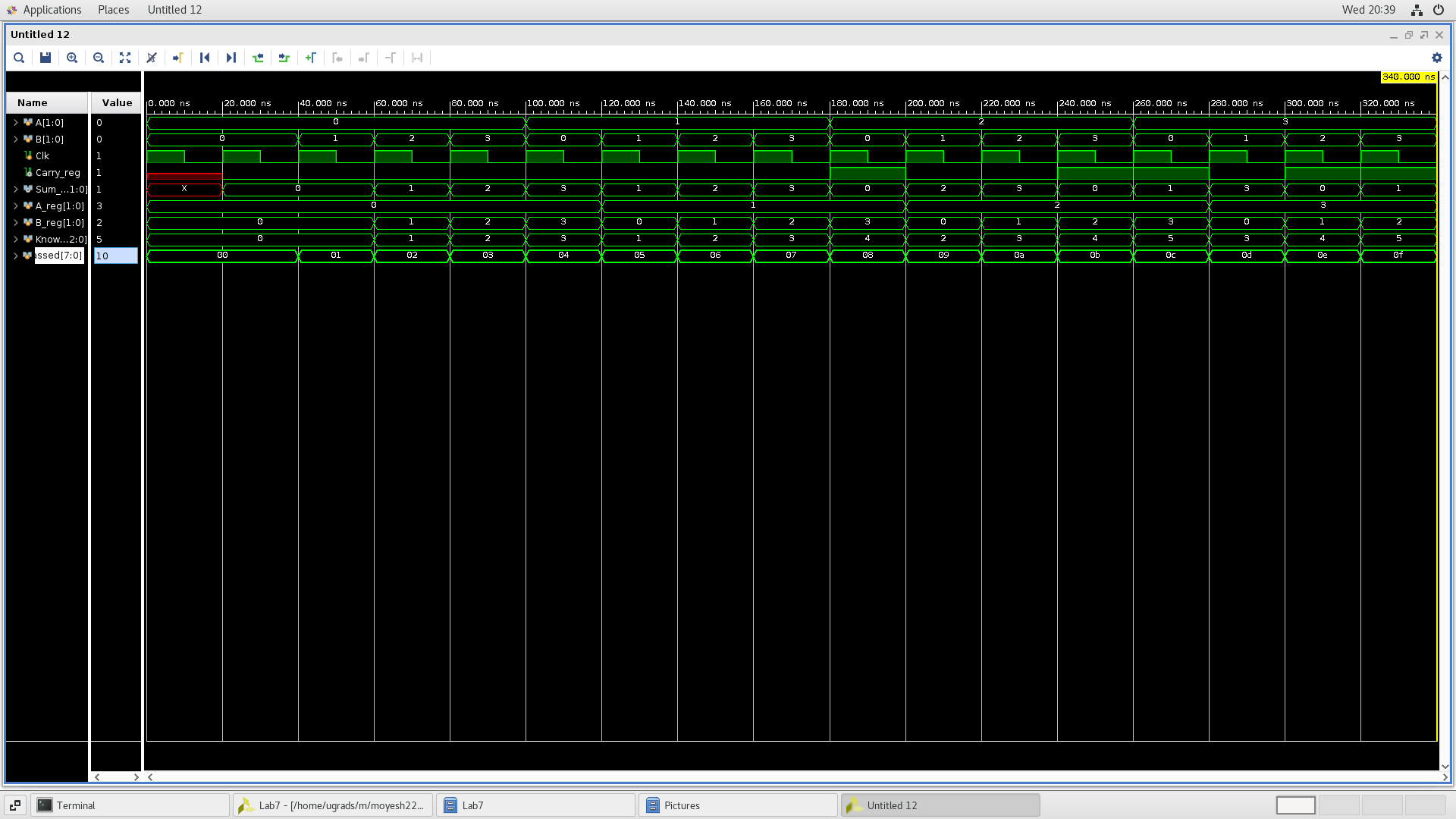
**Figure 10: D Flip Flop waveform**

****

**Figure 11: D Latch Behavioral waveform**

****

**Figure 12: D Flip Flop Behavioral waveform**

****

**Figure 13: Synchronous Adder waveform**

**3. Questions**

**1. For 1(f) in part 1, explain the results of the simulation.**

Changing the delays from 2 to 4, caused some tests to fail.

**2. For 3 in part 1, check the waveform with the internal signals. Are the latches behaving as expected? Why or why not?**

No, because the delay of Q and not Q don't receive the Clk at the same time.

**3. For 1.4 in part 1, compare the waveforms you captured from the behavioral Verilog to those from the structural Verilog. Are they different? If so, how?**

The waveforms are different. Behavioral Verilog waveforms do not show delays, unlike the structural Verilog.

**4. For 1(e) in part 2, what is the worst-case propagation delay through the adder?**

The progression delay of the slowest path is 175,000ns.

**5. Based on the clock period you measured for your synchronous adder, what would be the theoretical maximum clock rate (frequency)? What would be the effect of increasing the width of the adder on the clock rate? How might you improve the clock rate of the design?**

The maximum frequency is the same as the CLOCK\_PERIOD in the test bench. Increasing the width of the adder slows the clock rate. To make the clock rate faster increase the clock rate of the adder instantiated in the module. When modifying the test bench, a CLOCK\_PERIOD of 20 and 40 worked.

**Feedback:**

**The final adder synchronous module worked for every case except one. I have looked at the code again and compared it with my lab partner, and was unable to identify the root of the problem for the one test case.**