Lab 4.1

Lab 4.1

- In this lab, you will:
  - Learn the difference between blocking and non-blocking assignments
  - Learn how non-blocking assignments affect sampling
  - Learn when assignments are physically performed

Lab 4.1

- In directory ~/lab4.1, you will find a file named "wave.v". It contains an *initial* block that generates a waveform on a register named "R".
- Run the simulation. You may want to use a waveform viewer to look at the resulting waveform, although a text trace is also provided.
- Modify the blocking assignments to non-blocking assignments (except the first one), then run the simulation again. How are the results different? Why?

Lab 4.1

- In the same directory, you will find a file named "sample.v". It is similar to the file of the same name used in lab 3.1, but it contains different sampling routines.
- Run the simulation and look at the sample waveforms. Why are they different?

Lab 4.1: Optional

- In the same directory, you will find a file named "assign.v". It contains an *initial* block that performs a non-blocking assignment. Then, it waits for the time delay used in the assignment statement, and expects the assignment to have taken place.
- However, the assignment is still pending, as demonstrated by the report displayed by the if (R !== 1'b1) block. Why?

Lab 4.2

Lab 4.2

- In this lab, you will:
  - Learn how wires are used to model busses
  - Learn the difference between assignments and continuous assignments
  - Learn the effect of delays and strength in continuous assignments
  - Learn the effect of the inertial delay model
  - Learn how delays in continuous assignment affect sampling

Lab 4.2

- In directory ~/lab4.2, you will find a file named "bug.v".
- Try to compile the file: you should have errors on the *initial* statement and the continuous assignment.
- These errors often occur when learning to code in Verilog:
  - Wires cannot be directly assigned to using an assignment statement.
  - Registers cannot be assigned to using a continuous assignment.
- Fix the problems, then compile to prove that your changes are correct.

Lab 4.2

- In the same directory, you will find a file named "bus.v". It contains a model of four read-only registers multiplexed onto a shared bus.
- A register is selected when the corresponding bit in the SELECT register is set to '1'. The read-only value of the registers are '0001', '0010', '0100', and '1000', respectively.
- Run the simulation and examine the value on the bus as a function of the selected register(s).
- The value on the bus seems to be random. Why?

Lab 4.2

- The problem resides with the model for the registers: Each assignment simply replaces the previous value on the bus rather than adding a "driver" on that bus.
- To properly model this shared bus, make the following modifications:
  - Change the declaration of "BUS" from reg to wire.
  - Replace the *always* blocks that models the read-only registers with continuous assignments.
- Run the simulation and examine the result on BUS. Explain the waveform obtained.

Lab 4.2

- Repeat the simulation with BUS alternatively declared as *wor*, *wand*, then *trireg*. Explain the difference in simulation results.
- Go back to BUS being declared as a wire. Add delays of #(3), #(2,4), and #(2,4,6), respectively, to individual continuous assignments, leaving one assignment without delay. Simulate again. Explain the results.

Lab 4.2

- Modify the continuous assignments to drive a strong-strength '1', but a pull-strength '0'. You may want to remove the delays to make the simulation output easier to read. Simulate, then explain the results.
- Repeat with different strength combinations for different continuous assignments.

Lab 4.2

- In the same directory, you will find a file named "sample.v". It is similar to the file of the same name used in Labs 3.1 & 4.1, but it contains different sampling routines.
- Run the simulation and look at the sample waveforms. Why are they different?