**Lab 0 Report**

*Matthew Dragotto, Luke Moore, David Chen*

1. **Implementation:**

Lab 0 was fairly straightforward in that most of the steps for completing the lab were actually detailed in the specification. However, when finishing the steps listed by the project and reading over the provided code, it was apparent that the LEDs that were counting our 4-bit binary number were changing at the same speed as our FPGA clock. Therefore, in order to observe counting in action with our own eyes, we needed to implement a clock divider.

In our implementation of the clock divider, we added a 32-bit register that would hold a decimal value. When this value reached 50,000,000, we would add one to the register that held the 4-bit binary number.

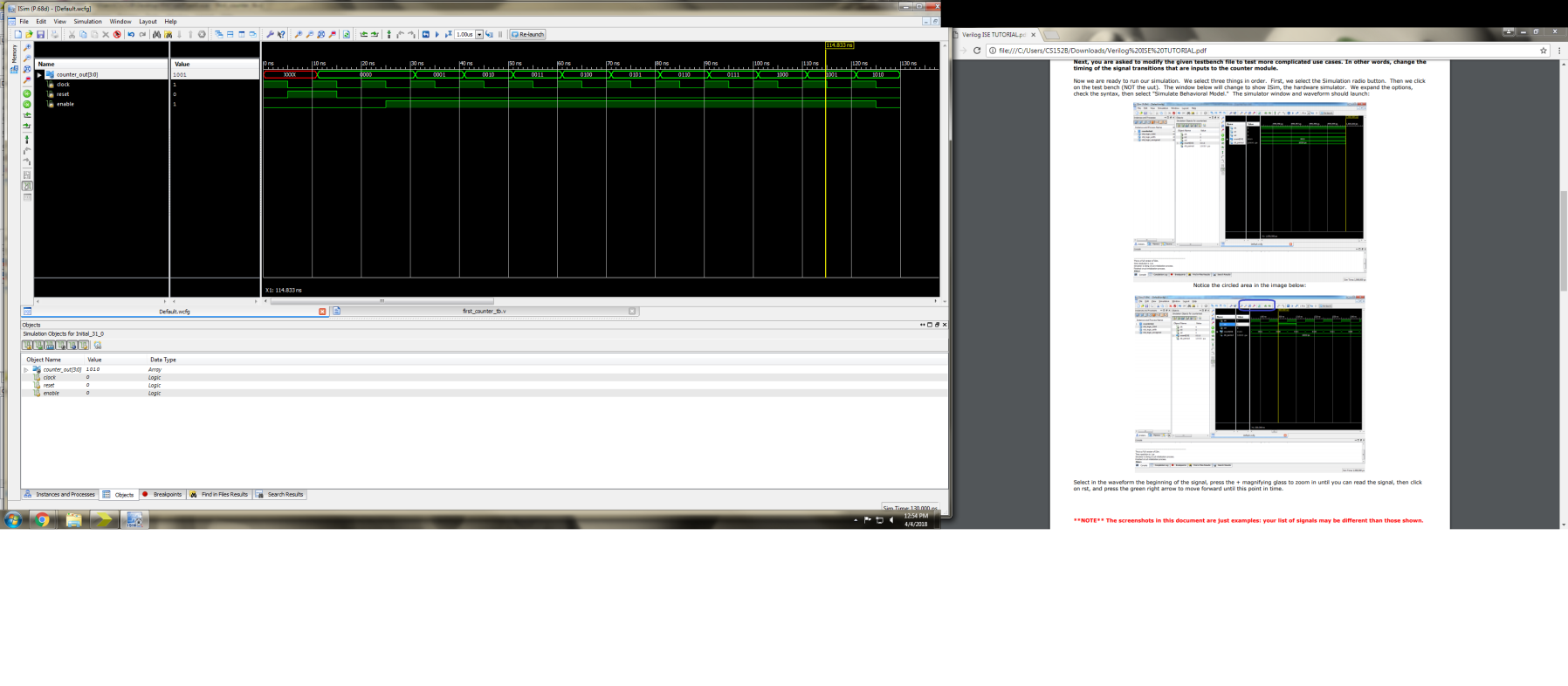


Figure : Waveform produced by the provided code

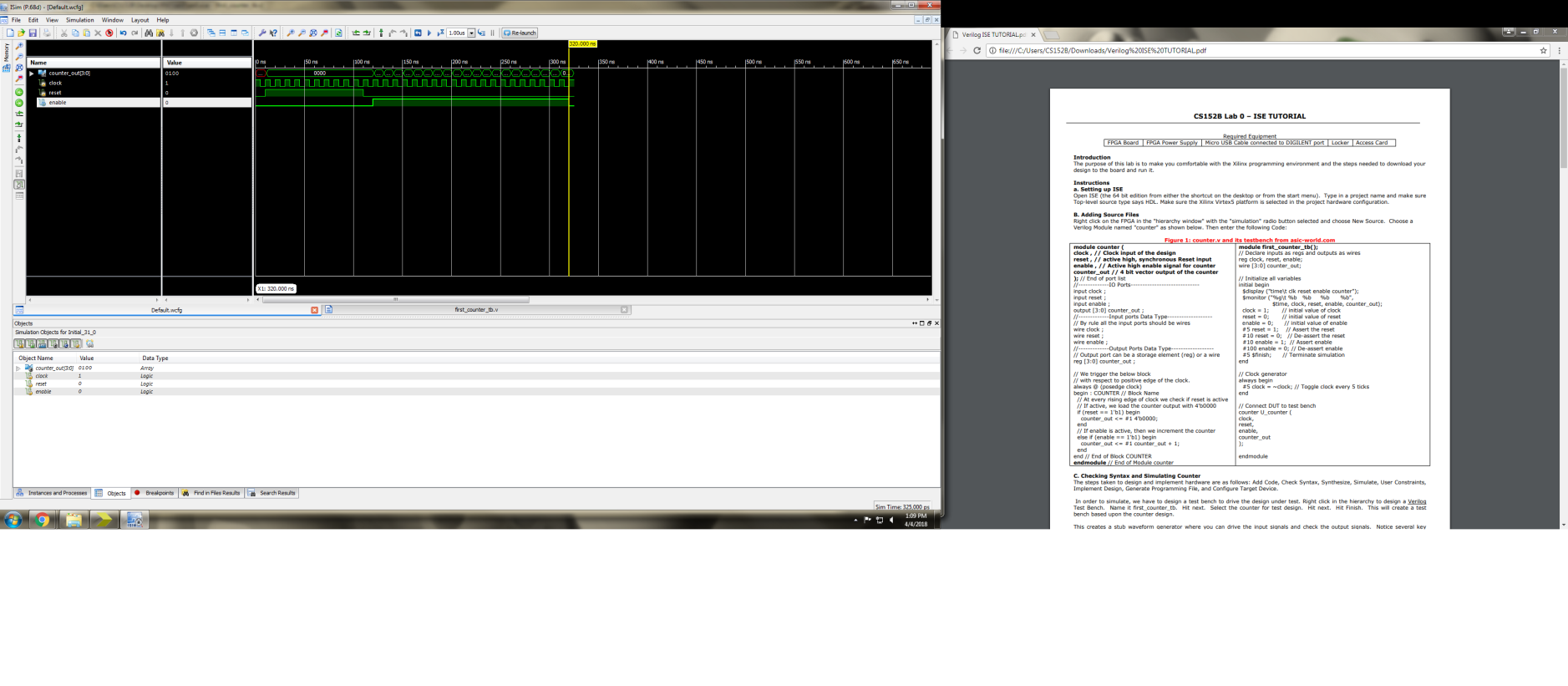


Figure : Waveform produced after modifying code and adding a clock divider

1. **Problems Encountered**

One problem that we encountered when implementing the clock divider was getting our reset to behave as intended. At first we tried a modular approach to the clock divider, where we made a separate module for it and hooked up the inputs and outputs from our counter to it, but having to still refresh ourselves on Verilog, this caused all sorts of bugs. Basically we just ended up complicating it more than it needed to be, since we created a new 1Hz clock frequency in our clock divider module, which then passed that back to our counter module, etc. This caused our reset not to work, and other weird behavior as a result of two separate pos-edge clock triggers that were both reliant on reset.

Our solution was to just combine the two loops and implement the clock divider inside of our counter module. This ended up simplifying everything a bit, and we were able to get a working implementation.