Course: CEG 7360 Embedded Systems
Semster: Fall 2025

**Verilog Programming Assignment 1**
**HDL Hello World**
**Due: Sunday 11-20-2025 11:55 PM EST**

WRIGHT STATE UNIVERSITY

---

## Description

The goal of this assignment is for you to set up your HDL simulation environment, and try out new tools. I want you to try out at least 2 different simulation tools for your HDL, in the first part of the assignment. Examples of this can be: Vivado, Icarus Verilog, Questa (Altera FPGA Starter Edition), EDAPlayground, and Verilator. EDAPlayground is a good backup option, especially for Mac/Linux users. If you are a windows user, you should use some of the more industry grade options (Vivado, Questa).

You don't need to only use fully open-source simulators, but if you would like to stick to open-source, I would start with Icarus Verilog, and then look into Verilator. Verilator is an industry relevant tool (not the simplest to start with), but may be helpful for you in your career. There are many other tools to help you run Verilator, with some popular wrappers written in other languages (Verime - Python). If this is your first time ever running HDL code, I would install Questa or Vivado (Windows), and then try Icarus Verilog (Linux/Mac/Windows). You all should be able to easily use EDA Playground for some simple designs, but isn't an appropriate solution for all your HDL work. **You do not need to purchase any licenses for this class/assignment.** All of these tools are free to use, when we are just simulating. They require a license to generate bitstreams for FPGAs, which we won't be doing.

Other software you can use, on Linux/Mac systems, are Cocotb and Surfer. Cocotb allows you to write testbenches in Python, but still uses Icarus Verilog under the hood. It will also come with GTKWave, by default, to load waveforms. Surfer is another popular open-source tool to load waveforms. They should also both work well with Verilator, since the FST and VCD formats are all natively supported.

For our Mac friends, I would suggest using Icarus Verilog, and then use EDAPlayground. Verilator should work in an UNIX environment, but again, may not be the most simple.

## Steps - Simulation Hello World

1. Run the given 4-1 MUX HDL file and test bench, and verify that the simulation waveform matches the expected output in **at least 2 different**. Its just a basic MUX, which we looked at in class. Take a screenshot of the resultant simulation, both a waveform and the debug output, in both tools.

   (a) This is the setup portion of the assignment, so your effort and time should be spent on understanding the software you are installing, and configuring it to work for development. Add some more tests to the testbench file, or maybe try and convert the code to be an 8-1 MUX, increase/decrease the bit width, add an enable signal, etc. None of this is required, but would be a good place to spend some time, if this is all new to you.

2. Implement two basic modules in their own individual files, with their own test benches. You need to create:

   (a) A clocked d-flip flop with a reset and enable signal. I would use behavioral logic, but gate level is okay.

      i. This should use a parameter to control the data width of the Flip Flop.
      ii. Enable is active low, and is fully asynchronous.
      iii. The reset is active low, and can trigger the reset asynchronously, but only release on the positive edge of the clock.
      iv. Create a test bench that showcases the enable and reset functionality, as well as some normal tests. Use your best judgment for this, and ask questions if you need help. Have your test bench output to a debug terminal, as well as view the simulation.

   (b) A sequence detector, detecting whatever 5-digit sequence you desire (just list what you are detecting in the comments. It might help to sketch a state diagram, but you aren't required to create a gate level design. Behavioral logic should be more than enough. You can follow the logic of a Mealy or Moore machine, but again, you don't need to do a gate-level design.

      i. The sequence should also have a parameter to control the data width of each data point in the sequence.
      ii. Make sure you comment on what form of state encoding you use, and describe why you chose it.
      iii. Create your own test bench that fully tests your design, contains a reasonable enough of test patterns, and monitors changes of important regs and wires. Take a screenshot of the output, and the simulation waveform.

## Rubric

This assignment is out of 60 points, with each deliverable being graded to the following rubric. Please submit your assignment as a collection of code, and a word document/PDF that shows the results of the simulations (screenshots). If you want to use a online git repo in place of code submission, let me know.

| | | | |
|---|---|---|---|
| (60 pts.) | 10 pts. Two simulators are used, to verify the original designs file for the MUX. Screenshots of the waveforms are available for each simulation tool. | 20 pts. The clocked d flip flip design matches the given spec. A proper testbench is created, and the resultant simulation is presented. | 30 pts. The sequence detector is built in behavioral logic and matches the given spec. A proper testbench is created, and the resultant simulation is presented. |