We will primarily be using a more sophisticated toolflow with Synopsys VCS for our Verilog simulator, as described in the sequence of tutorials on the ECE 5745 website. This short tutorials illustrates how to try working with Verilog using the Icarus Verilog simulator (either on your own machine or on the brg servers). These instructions assume you are working on the brg servers.

We first grab the source from the git repos:

```
% source setup-ece5745.sh
% ece5745-lab-admin -begin ece5745-S01-verilog-review
% cd ${HOME}/ece5745/ece5745-S01-verilog-review
% tree

--- build
   |--- Makefile
   `--- template.sav
--- gcd
   |--- gcd-GcdUnit.t.v
   |--- gcd-GcdUnit.v
   |--- gcd.mk
   `--- gcd-sim.v
--- vc
   |--- vc- Arbiters.t.v
   |--- vc- Arbiters.v
   |--- vc- Arith.t.v
   |--- vc- Arith.v
   |--- vc- Assert.v
   |--- vc- Memories.t.v
   |--- vc- Memories.v
   |--- vc- Misc.t.v
   |--- vc- Misc.v
   |--- vc.mk
   |--- vc- Muxes.t.v
   |--- vc- Muxes.v
   |--- vc- Queues.t.v
   |--- vc- Queues.v
   |--- vc- RAMs.t.v
   |--- vc- RAMs.v
   |--- vc- Regfiles.t.v
   |--- vc- Regfiles.v
   |--- vc- StateElements.t.v
   |--- vc- StateElements.v
   |--- vc- TestSink.v
   |--- vc- TestSource.v
   |--- vc- Test.v
   `--- vc- Trace.v
```

The vc subdirectory contains various Verilog components such as muxes, state elements, arbiters, etc. Files that end in .t.v are unit tests and are written using the simple unit testing framework provided in vc-Test.v. The gcd subdirectory contains the actual GCD unit (gcd-GcdUnit.v) along with the unit tests (gcd-GcdUnit.t.v) and the simulator (gcd-sim.v). The gcd.mk makefile fragment tells the build system which files are part of the gcd subproject.
You can build a single unit test and run it in isolation using the following:

```
% make vc-Muxes-utst && ./vc-Muxes-utst
iverilog -g2005 -Wall -Wno-sensitivity-entire-vector
   -Wno-sensitivity-entire-array -o vc-Muxes-utst \
   -I ../vc -I ../vc  ../vc/vc-Muxes.t.v
VCD info: dumpfile vc-Muxes.vcd opened for output.
```

Entering Test Suite: vc-Muxes
   + Running Test Case: vc_Mux2_w32
   + Running Test Case: vc_Mux4_w32

You can see more debug information by using the `+verbose` command like option.

```
% make vc-Muxes-utst && ./vc-Muxes-utst +verbose=2
make: `vc-Muxes-utst' is up to date.
VCD info: dumpfile vc-Muxes.vcd opened for output.
```

Entering Test Suite: vc-Muxes
   + Running Test Case: vc_Mux2_w32
   [ passed ] Test ( sel == 0 ) succeeded, [ 0a0a0a0a == 0a0a0a0a ]
   [ passed ] Test ( sel == 1 ) succeeded, [ b0b0b0b0 == b0b0b0b0 ]

```
+ Running Test Case: vc_Mux4_w32
   [ passed ] Test ( sel == 0 ) succeeded, [ 0a0a0a0a == 0a0a0a0a ]
   [ passed ] Test ( sel == 1 ) succeeded, [ b0b0b0b0 == b0b0b0b0 ]
   [ passed ] Test ( sel == 2 ) succeeded, [ 0c0c0c0c == 0c0c0c0c ]
   [ passed ] Test ( sel == 3 ) succeeded, [ d0d0d0d0 == d0d0d0d0 ]
```

You can also use gtkwave to view the waveforms.

```
% gtkwave vc-Muxes.vcd &
```

You can run all of the unit tests using the following:

```
% make check
```

You can build the GCD simulator and run it using the following:

```
% make
% make check
% ./gcd-sim +a=15 +b=5
```

Again, you can view the waveforms with gtkwave:

```
% gtkwave dump.vcd &
```