ECE 497: Special Project Weekly Report

Week 02

Alexander Lukens Karl Hallsby

Illinois Institute of Technology

February 4, 2021

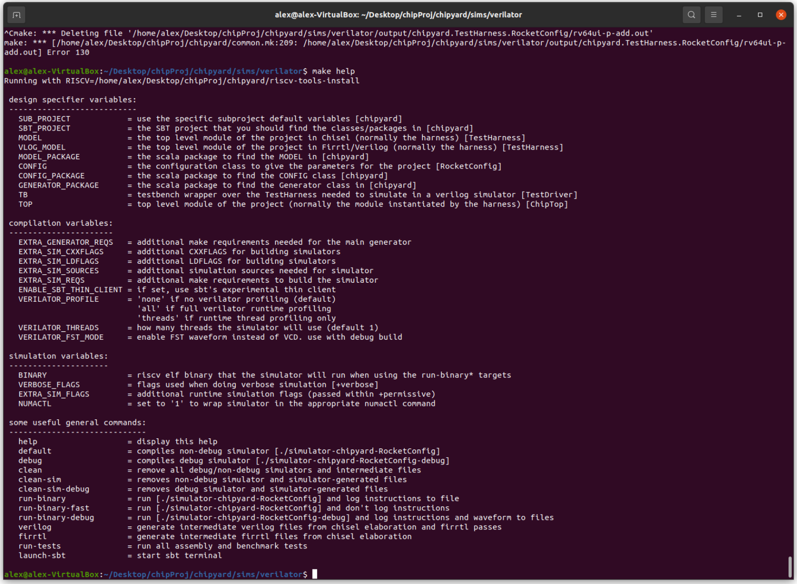
What We Did

- Set up Virtual machines to have similar environments.
- Clone the ChipYard repository.
- Build the toolchains required (Quite time-consuming)
- Followed documentation's example on how to generate a generic RISC-V chip.
- Used Verilator to simulate the default chip design.
  - Ran all tests (`make run-asm-tests`) (Quite time consuming)
  - Ran all benchmarks (`make run-bmark-tests`)
  - As expected, the default chip design passed all tests and benchmarks successfully.

What We Learned

- *Simulated* chip designs operate at O(1 kHz).
    - According to the documentation, these are significantly easier to debug.
- *FPGA-accelerated* chip designs operate at O(100 MHz).
    - These are also significantly more difficult to debug.
    - This speed is reached only when using FireSim (AWS).
    - We believe this implies we can write out the Verilog to Alex's FPGA and test there, albeit more slowly.
- `make run-asm-tests` runs all the instructions in the CPU design, ensuring ISA compliance.
- To get waveform outputs, run `make debug` when generating the simulated chip binary.

- The Chipyard built-in tests for simulations are sufficient to ensure complete RISC-V compliance of a custom SoC.
- Testing the entire ISA takes a significant amount of time, should only be completed when finalizing custom design, before FPGA implementation.
- There are many simulator options available. Significant options include `VERILATOR_THREADS`, `make verilog`, `make run-binary-debug`
- Using `make verilog` to generate the design completely in verilog, the default design should be able to be synthesized on the FPGA board


alex@alex-VirtualBox: ~/Desktop/chipProj/chipyard/sims/verilator

```
^Cmake: *** Deleting file '/home/alex/Desktop/chipProj/chipyard/sims/verilator/output/chipyard.TestHarness.RocketConfig/rv64ui-p-add.out'
make: *** [/home/alex/Desktop/chipProj/chipyard/common.mk:209: /home/alex/Desktop/chipProj/chipyard/sims/verilator/output/chipyard.TestHarness.RocketConfig/rv64ui-p-
add.out] Error 130

alex@alex-VirtualBox:~/Desktop/chipProj/chipyard/sims/verilator$ make help
Running with RISCV=/home/alex/Desktop/chipProj/chipyard/riscv-tools-install

 design specifier variables:
 ---------------------------
   SUB_PROJECT            = use the specific subproject default variables [chipyard]
   SBT_PROJECT            = the SBT project that you should find the classes/packages in [chipyard]
   MODEL                  = the top level module of the project in Chisel (normally the harness) [TestHarness]
   VLOG_MODEL             = the top level module of the project in Firrtl/Verilog (normally the harness) [TestHarness]
   MODEL_PACKAGE          = the scala package to find the MODEL in [chipyard]
   CONFIG                 = the configuration class to give the parameters for the project [RocketConfig]
   CONFIG_PACKAGE         = the scala package to find the CONFIG class [chipyard]
   GENERATOR_PACKAGE      = the scala package to find the Generator class in [chipyard]
   TB                     = testbench wrapper over the TestHarness needed to simulate in a verilog simulator [TestDriver]
   TOP                    = top level module of the project (normally the module instantiated by the harness) [ChipTop]

 compilation variables:
 ---------------------
   EXTRA_GENERATOR_REQS   = additional make requirements needed for the main generator
   EXTRA_SIM_CXXFLAGS     = additional CXXFLAGS for building simulators
   EXTRA_SIM_LDFLAGS      = additional LDFLAGS for building simulators
   EXTRA_SIM_SOURCES      = additional simulation sources needed for simulator
   EXTRA_SIM_REQS         = additional make requirements to build the simulator
   ENABLE_SBT_THIN_CLIENT = if set, use sbt's experimental thin client
   VERILATOR_PROFILE      = 'none' if no verilator profiling (default)
                            'all' if full verilator runtime profiling
                            'threads' if runtime thread profiling only
   VERILATOR_THREADS      = how many threads the simulator will use (default 1)
   VERILATOR_FST_MODE     = enable FST waveform instead of VCD. use with debug build

 simulation variables:
 ---------------------
   BINARY                 = riscv elf binary that the simulator will run when using the run-binary* targets
   VERBOSE_FLAGS          = flags used when doing verbose simulation [+verbose]
   EXTRA_SIM_FLAGS        = additional runtime simulation flags (passed within +permissive)
   NUMACTL                = set to '1' to wrap simulator in the appropriate numactl command

 some useful general commands:
 ----------------------------
   help                   = display this help
   default                = compiles non-debug simulator [./simulator-chipyard-RocketConfig]
   debug                  = compiles debug simulator [./simulator-chipyard-RocketConfig-debug]
   clean                  = remove all debug/non-debug simulators and intermediate files
   clean-sim              = removes non-debug simulator and simulator-generated files
   clean-sim-debug        = removes debug simulator and simulator-generated files
   run-binary             = run [./simulator-chipyard-RocketConfig] and log instructions to file
   run-binary-fast        = run [./simulator-chipyard-RocketConfig] and don't log instructions
   run-binary-debug       = run [./simulator-chipyard-RocketConfig-debug] and log instructions and waveform to files
   verilog                = generate intermediate verilog files from chisel elaboration and firrtl passes
   firrtl                 = generate intermediate firrtl files from chisel elaboration
   run-tests              = run all assembly and benchmark tests
   launch-sbt             = start sbt terminal

alex@alex-VirtualBox:~/Desktop/chipProj/chipyard/sims/verilator$ 
```

Figure: Chipyard Simulator Options and Flags

Next Steps

- Write the default chip out to Alex's FPGA and test.
- Use Scala/Chisel to generate a custom chip.
- Simulate the custom chip in software.
- Write the custom chip out to the FPGA.

References

Alon Amid et al. “Chipyard: Integrated Design, Simulation, and Implementation Framework for Custom SoCs.” In: *IEEE Micro* 40.4 (2020), pp. 10–21. ISSN: 1937-4143. DOI: 10.1109/MM.2020.2996616.