EE290C HW 1

Vighnesh Iyer

August 29, 2018

1 Chisel Simulations

1. Which backend would you use to run unit tests for a small block? Which backend would you use to run integration tests for a whole RISC-V core? Explain your answer.

   Use the FIRRTL interpreter to test small blocks for small amounts of time, since it becomes much slower than Verilator for more than 1k cycles or even a moderately complex block. Use Verilator for testing large blocks (such as a whole RISC-V core) since its simulation speedup is much greater than its startup penalty.

2. You can compute a rough estimate of the "frequency" of a simulation by dividing the number of simulation cycles by simulation time. You shouldn't include compilation and other start-up costs in this compilation time, so don't forget to subtract that out from the total simulation time. Use the 10 cycle simulations as a rough estimate of compilation time and other startup costs. Roughly, what frequency does the interpreter and Verilator achieve for each design complexity?

   For a 10 cycle simulation with a complexity of 51 taps, the startup time for the interpreter is 275 ms and for Verilator is 1426 ms.

   For the 10000 cycle simulation of same complexity:

$$\text{Interpreter Speed} = \frac{10000}{(19510 - 275) \cdot 1\text{e}{-3}} = 520 \text{ Hz}$$
$$\text{Verilator Speed} = \frac{10000}{(1625 - 1426) \cdot 1\text{e}{-3}} = 50 \text{ kHz}$$

3. FPGAs can be a useful way to accelerate simulation. Let's say that building and deploying a design for FPGA takes at least 15 minutes and will run at 10 MHz. Approximately how many cycles do you need to simulate for FPGA emulation to be worthwhile for this design?

   Assuming the Verilator compile time is insignificant compared to the simulation time; let $V(c)$ be the time it takes to simulate c cycles using Verilator, and $F(c)$ for FPGA.

$$V(c) = \frac{c}{50\text{e}3}$$
$$F(c) = 15 \cdot 60 + \frac{c}{10\text{e}6}$$
$$V(c) = F(c) \rightarrow c = 45\text{M cycles}$$

2 Chisel Generator Bootcamp

The IPython notebooks and HTML are attached for bootcamp sections 1, 2.1, 2.2

3 Reading Notes

Summary with my commentary in blue.

3.a Bachrach - DAC 2012 - Chisel: Constructing Hardware in a Scala Embedded Language

1. Identifies issues with using Verilog/VHDL as an HDL to write RTL description of digital circuits
   (a) Originally designed as simulation languages so a strict synthesizable subset needs to be understood by the designer and tools. This is an issue with any HDL as it evolves; there are useful non-syntheizable constructs that must exist along with the design to simplify simulation (magic memory loading/backdoors), formal verif (assert/assume), design assertions, print-based debugging (as used in DESSERT), register initialization (for GLS and maybe FPGAs), linting waivers, etc.
   (b) Poor abstraction facilities compared to modern languages Modern SystemVerilog provides facilities for OOP, structs, interfaces, and many other niceties, but I agree Scala is more powerful even still.
   (c)