FAST: FPGA-based Acceleration of Simulator Timing models

Derek Chiou, University of Texas at Austin. derek@ece.utexas.edu

Computer architectures have long been sufficiently complex to require simulation to model performance with any precision. Building accurate simulators, however, is a time-consuming and error-prone task that can consume a large amount of engineering resources. In fact, the difficulty of building a new simulator whose results can be relied on is a significant barrier to entry for novel architectures that is probably influencing what architecture exploration is actually done; it is much easier to start with a working, verified simulator than brewing a new one from scratch. In addition, cycle-accurate simulators are invariably slow since they model parallel hardware structures that are inefficient to simulate in software. Current cycle-accurate simulators run at about 10K instructions per second, much too slow to run realistic applications and thus relegating such simulators to either run toy benchmarks or requiring sampling techniques to reduce the number of instructions that need to be run.

Many cycle-accurate simulators[1, 2] are implemented as two intertwined components: a timing model that implements structures that affect timing and resource contention such as caches, branch predictors and a limited number of ALUs and a functional model that accurately emulates the ISA. Current simulators spent the vast majority of their time in the timing model, since it models hardware structures that tend to be very parallel in nature and thus are difficult or impossible to accurately simulate in software quickly. Hardware, on the other hand, can very easily exploit the parallelism expressed in the timing model.

The FAST project is attempting to dramatically speed up cycle-accurate (and functional) simulation by implementing most of the timing model, as well as key parallelizable parts of the functional model, in FPGAs. Efficient communication mechanisms between the hardware-based timing model and the software-based functional model are key to the success of this concept. Our approach will take the form of a simulator compiler that will allow the user to assume he/she has access to all of the state in all of the models. The compiler will generate communication structures that only provide access to accessed state. Preliminary estimates indicate that timing models for very aggressive processors could easily fit in a single FPGA. In fact, it's very likely that multiple timing models could fit in a single FPGA.

The FAST project relies heavily on modular, reusable components to construct the timing models, making it easy for a user to mix and match components to quickly build new, accurate simulators. The components will provide timing model functionality and are thus much less complex and much more reusable than components designed to actually implement processor functionality. For example, an ALU module only models delay, rather than having to implement every operation that ALU can perform. Modularization of the hardware modules implementing the timing model will enable considerable reuse of components leading to higher productivity and more correctness confidence. The goal is to provide a set of modules that will allow a user to very quickly (in a matter of a few hours to a few days) a timing model from scratch. Swapping out individual components will be almost instantaneous and writing new modules should be possible in a matter of days.

Such a simulation environment promises at least two to three orders of magnitude performance boost in the timing model. The goal is to make the functional model the bottleneck, and then applying acceleration techniques to functional model performance bottlenecks such as rollback support for speculation. The hope is that such a simulator will run at least at 1M instructions per second and perhaps as fast as 100M instructions per second. Such performance will open the door to much more extensive simulation and the elimination of the need for sampling techniques (though sampling techniques will still be handy to run even larger examples.) In addition, the modular structure of the simulator dramatically improves the ability for architects to explore new ideas quickly and accurately.

[1] Doug Berger and Todd M. Austin. "The SimpleScalar Tool Set, Version 2.0". *University of Wisconsin-Madison Computer Science Department Technical Report #1342*, June 1997.
[2] Joel Emer, et. al. Asim: "A Performance Model Framework". *Computer*. 35(2):68-76, 2002.