Leveraging Modular Simulation for Systematic Design Space Exploration

Veerle Desmet
Ghent University, Belgium
veerle.desmet@elis.UGent.be

Grigori Fursin, Sylvain Girbal, Olivier Temam
INRIA Futurs, France
{grigori.fursin,sylvain.girbal,olivier.temam}@inria.fr

Due to the growing complexity of processors, architecture design space exploration has always been more an art than a systematic process. The architect usually applies a trial-and-error process where intuition and experience often drive the creation and selection of appropriate designs. The widespread use of multi-cores only further increases the design space by adding the number of cores, their communications means, memory coherence issues, and a wealth of associated software issues. Therefore, a more systematic design approach could help streamline the process and thus keep design cycles reasonable, or at the very least, it could help the designer focus on the design areas with the best potential.

Both in companies and academia, more systematic design space exploration techniques are emerging, such as the exhaustive or statistical exploration of the target design parameters ranges [5]. However, such techniques can only assist in the final stages of the design process by maximizing the benefit of the target design and simplifying the task of dimensioning the design components. As a result, though the design space is still fairly large at that stage, it is only a fraction of the actual design space that architects must explore.

The purpose of our research work is to extend systematic design space exploration beyond simple components parameterization by allowing the automatic and systematic selection of architectural components themselves. For instance, instead of simply configuring all the different possible cache parameters (cache size, line size, access time, etc) for a given cache architecture, the goal is to automatically try out different cache architectures and policies (victim caching, prefetching techniques, etc). Consequently, the design space targeted by systematic exploration would be far larger than what is currently possible which allows to select the most promising designs from a wider space.

We make such broader design space exploration possible by tightly integrating design space exploration with modular simulation, within a framework called UNISIM [1]. More precisely, four features are combined for that purpose: (1) decomposition of complex architectures into a set of independent hardware modules, (2) explicit and standardized modules communications, (3) an indexed repository of modules, (4) a stored representation of modules compatibility.

After several years of monolithic and large simulators [2], architects have progressively recognized the benefits of modular simulators where architectures are broken down into individual hardware components, much like in the classic hardware block diagrams [1, 3, 4].
Beyond modularity, one of the key benefits of UNISIM is to provide explicit and standardized communications interfaces between modules. For instance, any cache module respecting a given processor-to-memory interface, can be easily plugged into that interface, without internal modifications in any processor or memory module. Moreover, various cache modules can be tried out provided they respect that interface. In UNISIM, the modules connection principles are sufficiently standardized to perform such plugging automatically.
The UNISIM environment has another specific feature: it is tightly coupled with an open but formatted repository, where modules can be uploaded and stored with their key characteristics. The format includes the nature of the modules, their communications interfaces, their possible hardware derivatives (e.g., different variations of the same prefetching strategy), and the history of known compatible modules. This history can either be collected when a full simulator is uploaded, i.e. the archiving system records which modules are connected to which other modules, or because it respects one of the standardized and typical interfaces (processor to cache, bus or memory interfaces, etc).

Based on this infrastructure, it is then possible to try out any configuration of compatible modules. Since modules also embed the possible ranges of their parameters (including complex functional relations between parameters), modules exploration can be coupled with the more

classic parameters exploration. Because the parameters of some modules depend on parameters of other modules (e.g., L2 cache lines must be broken down into sets which size is equal to the bus width), modules are also fitted with an introspection capability which can be probed by the modules they are connected to, in order to expose some of their internal parameters.

Beyond targeted design space exploration tasks, it is also possible to implement continuous design space exploration in order to find the best possible architecture. Any time a new module is uploaded to the open repository, design space exploration is restarted to account for possible better designs. Moreover, UNISIM offers a number of services, such as power and area modeling, which can be used to find the best possible architectures under different constraints and performance targets.

References

[1] David I. August, Jonathan chang, Sylvain Girbal, Daniel Gracia Pérez, Gilles Mouchard, David Penry, Olivier Temam, and Neil Vachharajani. UNISIM: An open simulation environment and library for complex architecture design and collaborative. *Computer Architecture Letters*, September 2007.

[2] Doug Burger, Todd M. Austin, and Steve Bennett. Evaluating future microprocessors: The SimpleScalar Tool Set. Technical report, Computer Sciences Department, University of Wisconsin-Madison, July 1996.

[3] Daniel Gracia Pérez, Gilles Mouchard, and Olivier Temam. MicroLib: A case for the quantitative comparison of micro-architecture mechanisms. In *Proceedings of the 37th Annual International Symposium on Microarchitecture*, pages 43–54, December 2004.

[4] Manish Vachharajani, Neil Vachharajani, David A. Penry, Jason A. Blome, and David I. August. Microarchitectural exploration with Liberty. In *Proceedings of the 35th Annual International Symposium on Microarchitecture*, pages 271–282, November 2002.

[5] Joshua J. Yi, David J. Lilja, and Douglas M. Hawkins. Improving computer architecture simulation methodology by adding statistical rigor. *IEEE Transactions on Computers*, 54(11):1360–1372, November 2005.