Chalk, a language and tool for architecture design and analysis

Wouter Swierstra, Koen Claessen, Carl Seger, Mary Sheeran and Emily Shriver
Chalmers University of Technology and Intel

1 Introduction

Computer hardware is becoming increasingly complex. Nowadays, most personal computers are furnished with several cores sharing gigabytes of memory behind multi-level caches. Current hardware and architecture description languages are struggling to keep up with this burgeoning complexity of today's architectures. We see two possible ways forward: either bite the bullet and start trying to reason about SystemC, as recently argued for by Vardi [6], or start again and try to provide exactly the right abstractions in a much simpler setting. We have chosen the latter approach, in a joint project that has been running since Feb. 2009. Our proposed talk presents progress so far and plans for future work.

2 The problem

Computer architects must strike a balance between performance, power consumption, cost, and many other non-functional design constraints. The exact repercussions of early design decisions are not always clear. How will an additional cache improve performance? How much additional power will the cache require? How much will it cost? It is crucial to provide architects with tool support to make such decisions. Architectural analysis tools have traditionally concentrated on performance estimation, but are less well suited for power estimation. Mishra and Dutt argue that it is difficult to design an Architecture Description Language for a wide variety of architectures to perform different tasks using the same specification [5]. We see, once again, a window of opportunity for advanced programming language technology, and particularly functional programming languages.

3 Chalk

We have been inspired by earlier work on Lava [1] and Hawk [2, 3]. Both Lava and Hawk are embedded domain specific languages in Haskell developed about 10 years ago. They differ both in their aims and in how the embedding is constructed. Lava is a deeper embedding of a purely structural description language, while Hawk permits attractive high level executable architecture descriptions, but doesn't allow the user to do much else. We have aimed to find a sweet spot between these two extremes, exploiting modern functional programming techniques (such as applicative functors and symbolic simulation) to make abstractions and analysis methods available to the user, while still allowing the description of high level behaviours. The resulting language and tool is Chalk. Chalk descriptions can be *inspected*, in order, for example, to generate netlists or perform analyses; but at the same time, the value types of signals are *general* and can be user-defined; any Hawk circuit description can be turned into a Chalk description by a simple translation process.

The proposed talk will present Chalk via example architecture descriptions, and will describe the associated analyses, including a simple activity analysis. Next, we will show how Chalk is implemented, including the effect of the choice to make the interface *applicative* [4]. We will also discuss the current limitations of Chalk, present our plans for future work, and request feedback from the audience. Given that we are not alone in investigating the use of embedded high level hardware description languages in Haskell, we would be keen to start a discussion on possible collaboration in this field.

References

[1] Per Bjesse, Koen Claessen, Mary Sheeran, and Satnam Singh. Lava: Hardware design in Haskell. In *ICFP '98: Proceedings of the Third ACM SIGPLAN International Conference on Functional Programming*, 1998.

[2] B. Cook, J. Launchbury, and J. Matthews. Specifying superscalar microprocessors in Hawk. *Formal Techniques for Hardware and Hardware-like Systems. Marstrand, Sweden*, 1998.

[3] J. Launchbury, J.R. Lewis, and B. Cook. On embedding a microarchitectural design language within Haskell. In *Proceedings of the fourth ACM SIGPLAN international conference on Functional programming*, pages 60–69. ACM New York, NY, USA, 1999.

[4] Conor McBride and Ross Paterson. Applicative programming with effects. *Journal of Functional Programming*, 18(01):1–13, 2007.

[5] P. Mishra and N. Dutt. Architecture description languages for programmable embedded systems. *IEE Proc.-Comput. Digit. Tech.*, 152(3), May 2005.

[6] Moshe Vardi. Formal Techniques for System-Level Verification, invited talk. In *FMCAD*, 2009. http://fmv.jku.at/fmcad09/slides/vardi.pdf.