Keynote Talk

The Revolution in Electronic Design Automation: Implications for Automated Software Engineering

Prof. A Richard Newton
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley

Over the past quarter century, we have increased the number of logic transistors per chip from a few thousand to over ten million. By 2010, we will have more than half a billion logic gates on a single piece of silicon. Today, the tooling alone for a complex chip costs more than $500,000 and is also growing. The cost of an error in a chip and our inability to use a simple edit-recompile-link approach to design debugging has resulted in a significant effort in the area of design validation, verification, and test. Of course, the solution to such complex design and validation problems is a lot more than good tools and depends at least as much on methodology. The challenge is finding the “right” combination of mathematical techniques, tools, and methodology that provide maximum design productivity, minimum expressive constraints for high performance, and yet result in a correct design.

In today’s high-end chip designs, more than half of the engineers working on the “chip” are actually writing software for use as part of the chip itself. It must be as correct as the hardware. For this reason, a large number of electronic design technology researchers are now turning their attention to software and the hardware/software interface as the next big EDA challenge. A significant community has built up around the formal models of hardware and mixed hardware/software systems, with emphasis today on a synchronous reactive model of such systems. New languages are being developed for the design of embedded systems and, for many applications, the notion of automated synthesis of run-time environments is being implemented. Formal verification techniques, automatic generation of “covers”, and algebras for embedded software are all under development. Many of the ideas being explored have their roots in the software community, for example today’s emphasis on “interface based design” or “communication based design” in hardware looks a lot like object-oriented design. Nonetheless, coupling the powerful ideas and emphases of the hardware community with the state-of-the-art in software for embedded systems is bound to result in some new, major developments.

We are about to witness another revolution in the way silicon chips and embedded systems are designed and verified. Software development is, without doubt, the aspect of the process likely to see the most changes.