Language-based Hardware Verification with ReWire: Just Say No! to Semantic Archaeology

Submitted by Katie Dey on Thu, 04/23/2015 - 6:42pm. Contributors: William Harrison, Adam Proctor

Abstract:
There is no such thing as high assurance without high assurance hardware. High assurance hardware is essential, because any and all high assurance systems ultimately depend on hardware that conforms to, and does not undermine, critical system properties and invariants. And yet, high assurance hardware development is stymied by the conceptual gap between formal methods and hardware description languages used by engineers.

This talk presents ReWire, a Haskell-like functional programming language providing a suitable foundation for formal verification of hardware designs, and a compiler for that language that translates high-level, semantics-driven designs directly into working hardware. ReWire's design and implementation are presented, along with a case study in the design of a secure multicore processor based on the Xilinx PicoBlaze, demonstrating both ReWire's expressiveness as a programming language and its power as a framework for formal, high-level reasoning about hardware systems.

The philosophy that drives the development of ReWire may be summed up as the conviction that semantic archaeology is the bane of high assurance computing. By "semantic archaeology", we mean the process of developing a formal specification for an existing computing artifact. Semantic archaeology is time-consuming and expensive, because such artifacts are rarely written with formal semantics in mind, and, consequently, the formal methods scientist must attempt a painstaking reconstruction of the system semantics from informal and often incomplete natural language documents (if, indeed, such a reconstruction is even possible).

In keeping with the proof engineering theme of HCSS 2015, this work aims to engineer maintainable, extensible and reusable proofs via a language design approach. Semantic archaeology is avoided by starting from a semantically-defined language. ReWire is both a computational l-calculus suitable for writing formal specifications and an expressive functional language and compiler for generating efficient hardware artifacts. With ReWire, the text of a design is verified (rather than a reconstructed model of the design) and the compiler transforms that same design into hardware, thereby unifying the languages of specification, design and implementation. The hypothesis of this work is that this duality will position ReWire to avoid the pitfalls of semantic archaeology without sacrificing performance. To date, our case studies have generated artifacts with acceptable performance.

The research presented here is joint work with Professor Michela Becchi and Ian Graves of the University of Missouri and Dr. Gerard Allwein of the US Naval Research Laboratory.

References

Biography:

Dr William Harrison received his BA in Mathematics from Berkeley in 1986 and his doctorate from the University of Illinois at Urbana-Champaign in 2001 in Computer Science. From 2000-2003, he was a post-doctoral research associate at the Oregon Graduate Institute in Portland, Oregon where he was a member of the Programatica project. Dr Harrison is an associate professor in the Computer Science department at the University of Missouri, where he has been since 2003. In December 2007, he received the CAREER award from the National Science Foundation. His interests include all aspects of programming languages research (e.g., language-based computer security, semantics, design and implementation), formal methods and malware analysis.

License: Creative Commons 2.5

Other available formats:

Language-based Hardware Verification with ReWire: Just Say No! to Semantic Archaeology
Switch to normal viewer Switch to experimental viewer