Cache Coherence Verification with TLA+

Homayoon Akhiani, Damien Doligez, Paul Harter, Leslie Lamport, Joshua Scheid, Mark Tuttle*, and Yuan Yu

Compaq Computer Corporation

We used the specification language TLA+ to analyze the correctness of two cache-coherence protocols for shared-memory multiprocessors based on two generations (EV6 and EV7) of the Alpha processor. A *memory model* defines the relationship between the values written by one processor and the values read by another, and a *cache-coherence protocol* manipulates the caches to preserve this relationship. The cache-coherence protocol is a fundamental component of any shared-memory multiprocessor design. Proving that the coherence protocol implements the memory model is a high-leverage application of formal methods. The analysis of the first protocol was largely a research project, but the analysis of the second protocol was a part of the engineers' own verification process.

The EV6-based multiprocessor uses a highly-optimized, very complicated cache-coherence protocol. The protocol uses about sixty different types of messages, and the documentation for the protocol consists of a stack of twenty documents about four inches tall, none of it complete or precise enough to be the basis of a proof. After more than two man-years of effort, four of us were able to write a 1900-line specification of the algorithm, a 200-line specification of the Alpha memory model, and about 3000 lines of proof that the algorithm implements the memory model. This was far from a complete proof, but enough of a proof to subject the algorithm to a rigorous analysis, and to discover one bug in the protocol and one bug in the memory model.

The cache-coherence protocol for EV7-based multiprocessors is dramatically simpler, bringing a complete correctness proof within the realm of possibility. A new tool, a model checker for TLA+ called TLC, increased the odds of success. TLC enumerates the reachable states in a finite-state model of a specification written in an expressive subset of TLA+, and it checks that an invariant written in TLA+ holds in each of these states. When TLC discovers an error, a minimal-length sequence of states leading from an initial state to a bad state is reported. One of us wrote an 1800-line specification of the algorithm. Using TLC to check multiple invariants uncovered about 66 errors of various kinds. The engineers were also able to use state sequences output by TLC as input to their own RTL-verification tools, an interesting case of formal methods helping engineers use their own tools more efficiently.

We were pleased to see that the basic verification methodology, refined through years of research, works pretty much as expected, although the proofs were hard. The engineers had little difficulty learning to read and write TLA+ specifications. We hope TLA+ will play a role in other projects in the near future.

* Mark Tuttle, Cambridge Research Lab, Compaq Computer Corporation, One Kendall Square, Building 700, Cambridge, MA 02139, mark.tuttle@compaq.com.