

Prototyping Architectural Support for Program Rollback: An Application to Software Debugging

Radu Teodorescu and Josep Torrellas
University of Illinois at Urbana-Champaign

Several recently-proposed architectural techniques require speculation over long program sections. Examples of such techniques are Thread-Level Speculation [2, 4, 8, 9], speculation on synchronization [7, 5], speculation on the values of invalidated cache lines [3], speculation on conforming to a memory consistency model [1], and speculation on the lack of software bugs [6, 10].

In all these cases, when speculation fails, the architecture has to provide a means to quickly and cleanly rollback the side effects of the speculative code. More specifically, as a thread executes speculatively, the architecture buffers the register and memory state that it generates. If and when the speculation is shown to be correct, the architecture quickly commits the speculative state. If, instead, the speculation is incorrect, the state is discarded and the program is rolled back to before the speculative execution.

This paper reports on a processor and memory-hierarchy prototype based on FPGAs that models hardware for program rollback. The prototype implements register checkpointing and restoration, speculative state buffering in the L1 cache for later commit or discarding, and instructions for transitioning between speculative and non-speculative execution modes.

We use the prototype to demonstrate how to use application rollback to help debug *production code*. The compiler inserts hints into the application to indicate regions of code that are “at risk”. These suspicious regions are then executed in speculative mode. If an external source detects a bug, the suspicious region is rolled-back and re-executed. Upon re-execution, the compiler can choose to enable more instrumentation that will help characterize the buggy code region thoroughly.

For our prototype, we modified a synthesizable VHDL implementation of a 32-bit processor compliant with the SPARC V8 architecture. We map the modified processor to a Xilinx Virtex-II FPGA chip on a dedicated development board. We ran several applications on top of the Linux kernel.

We show that with relatively simple hardware and minimum impact on performance we can enable lightweight, on-the-fly debugging of production code. Our measurements show that the hardware extensions increase the resource requirements of the processor, when targeting the FPGA technology, by less than 2.5%.

We envision this hardware as part of a larger debugging infrastructure that includes compiler and OS assistance to provide bug detection and characterization in production runs.

References

- [1] Chris Gniady, Babak Falsafi, and T. N. Vijaykumar. Is $sc + ilp = rc$? In *Proceedings of the 26th annual international symposium on Computer architecture*, pages 162–171. IEEE Computer Society, 1999.
- [2] Lance Hammond, Mark Willey, and Kunle Olukotun. Data Speculation Support for a Chip Multiprocessor. In *Proceedings of the 8th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 1998.
- [3] Jaehyuk Huh, Jichuan Chang, Doug Burger, and Gurindar S. Sohi. Coherence decoupling: making use of incoherence. In *Proceedings of the 11th international conference on Architectural support for programming languages and operating systems*, pages 97–106. ACM Press, 2004.
- [4] V. Krishnan and J. Torrellas. A Chip-Multiprocessor Architecture with Speculative Multithreading. *IEEE Trans. on Computers*, pages 866–880, September 1999.
- [5] Jose F. Martínez and Josep Torrellas. Speculative synchronization: applying thread-level speculation to explicitly parallel applications. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 18–29. ACM Press, 2002.
- [6] Jeffrey Oplinger and Monica S. Lam. Enhancing software reliability with speculative threads. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 184–196, October 2002.
- [7] R. Rajwar and J. R. Goodman. Speculative Lock Elision: Enabling highly concurrent multithreaded execution. pages 294–305, Austin, TX, December 2001.
- [8] G.S. Sohi, S.E. Breach, and T.N. Vijayakumar. Multiscalar Processors. In *22nd International Symposium on Computer Architecture*, pages 414–425, June 1995.
- [9] J. G. Steffan and T. C. Mowry. The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization. In *Proceedings of the 4th International Symposium on High-Performance Computer Architecture*, February 1998.
- [10] Pin Zhou, Feng Qin, Wei Liu, Yuanyuan Zhou, and Josep Torrellas. iWatcher: Efficient Architecture Support for Software Debugging. In *Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA)*, pages 224–237, June 2004.