Compilation Issues For a Simultaneous Multithreading Processor
(Abstract)

Jack L. Lo, Susan J. Eggers, Henry M. Levy, Dean M. Tullsen
{jlo,eggers,levy,tullsen}@cs.washington.edu

Department of Computer Science & Engineering
University of Washington

Simultaneous multithreading (SMT) is a technique that permits multiple independent threads to issue multiple instructions each cycle to a superscalar's functional units [TEL95]. SMT combines the multiple-instruction-issue features of modern superscalar processors with the latency-hiding ability of multithreaded architectures. Unlike conventional multithreaded architectures [LGH94][ALKK90][Smi81][ACC+90], which depend on fast context switching to share processor execution resources, all hardware contexts in an SMT processor are active simultaneously, competing each cycle for all available resources. This dynamic sharing of the functional units allows simultaneous multithreading to substantially increase throughput, attacking the two major impediments to processor utilization --- long latencies and limited per-thread parallelism.

This benefit of SMT can be realized without extensive changes to a conventional wide-issue superscalar, by heavily leveraging off existing superscalar technology and projecting these design trends 3-5 years into the future. We have designed a low-level SMT architecture that achieves three goals: (1) it minimizes the architectural impact on the conventional superscalar design, (2) it has minimal performance impact on a single thread executing alone, and (3) it achieves significant throughput gains when running multiple threads. Simulations have shown that this simultaneous multithreading architecture achieves a throughput of 5.4 instructions per cycle, a 2.5-fold improvement over an unmodified superscalar with the same hardware resources [TEE+96].

Thus far, we have only evaluated simultaneous multithreading with a workload of uniprocessor applications. Each thread was assigned a distinct program, modeling a multiprogrammed workload, as opposed to a parallel processing workload. The multiple hardware contexts in an SMT processor can also be used, however, to exploit parallelism within a single program. For implicitly parallel programs, parallelism can be extracted via parallelization techniques, such as those provided by the Stanford University Intermediate Format Compiler [WFW+94]. We are currently using the SUIF compiler to create threads that exploit loop-level parallelism found in a variety of scientific programs. The SUIF compiler will also serve as part of a framework for investigating other compilation issues critical to simultaneous multithreading.

For our analysis of medium-grained parallellism, we are using the SPEC92 [Dix92] and Perfect Club [BCK+89] benchmark suites. These programs are sequential programs, but exhibit medium-grained loop-level parallelism. By using the SUIF compiler, we can create threads to execute the iterations of a parallel loop to take full advantage of both this parallelism and the capabilities of an SMT processor. These threads rely on the SUIF parallel runtime library, which we have rewritten for our SMT processor.

In addition to handling parallelization issues, the compiler for an SMT processor also must consider the cache effects of multiple simultaneous threads. In contrast to the threads on multiprocessors, threads on an SMT processor share a cache, and therefore, we can ignore cache coherency issues between them. However, because of this sharing, we need to minimize the cache interference caused by the interaction of multiple simultaneous threads from a single program. We plan to look at the applicability of blocking and other mechanisms to address these issues. As with the other medium-grained experiments, we will start with the blocking algorithms developed for the SUIF compiler.

References

[ACC+90] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith. The Tera computer system. In *International Conference on Supercomputing*, pages 1–6, June 1990.

[ALKK90] A. Agarwal, B.H. Lim, D. Kranz, and J. Kubiatowicz. APRIL: a processor architecture for multiprocessing. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pages 104–114, May 1990.

[BCK+89] M. Berry, D. Chen, P. Koss, D. Kuck, S. Lo, Y. Pang, L. Pointer, R. Roloff, A. Sameh, E. Clementi, S. Chin, D. Schneider, G. Fox, P. Messina, D. Walker, C. Hsiung, J. Schwarzmeier, K. Lue, S. Orszag, F. Seidl, O. Johnson, R. Goodrum, and J. Martin. The PERFECT Club Benchmarks: Effective performance evaluation of supercomputers. *International Journal of Supercomputer Applications*, 3(3):9–40, Fall 1989.

[Dix92] K.M. Dixit. New CPU benchmark suites from SPEC. In *COMPCON*, pages 305–310, Spring 1992.

[LGH94] J. Laudon, A. Gupta, and M. Horowitz. Interleaving: A multithreading technique targeting multiprocessors and workstations. In *Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 308–318, October 1994.

[Smi81] B.J. Smith. Architecture and applications of the HEP multiprocessor computer system. In *SPIE Real Time Signal Processing IV*, pages 241–248, 1981.

[SWG92] J.-P. Singh, W.-D. Weber, and A. Gupta. SPLASH: Stanford parallel applications for shared memory. *Computer Architecture News*, 20(1):5–44, March 1992.

[TEE+96] D.M. Tullsen, S.J. Eggers, J.S. Emer, H.M. Levy, J.L. Lo, and R.L. Stamm. Exploiting choice: Instruction fetch and issue on an implementable simultaneous multithreading processor. *Submitted for publication*, 1996.

[TEL95] D.M. Tullsen, S.J. Eggers, and H.M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 392–403, June 1995.

[WFW+94] R.P. Wilson, R.S. French, C.S. Wilson, S.P. Amarasignhe, J.M. Anderson, S.W.K. Tjiang, , S.-W. Liao, C.-W. Tseng, M.W. Hall, M.S. Lam, and J.L. Hennessy. SUIF: An infrastructure for research on parallelizing and optimizing compilers. *ACM SIGPLAN Notices*, 29(12):31–37, December 1994.

[WOT+95] S.C. Woo, M. Ohara, E. Torrie, J.-P. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *22nd Annual International Symposium on Computer Architecture*, pages 24–36, June 1995.