

Circuit	LPC-set	hybrid BACKSIM	BACKSIM
C432	2.4	2.2	2.5
C499	5.4	5.6	6.0
C880	5.7	4.9	X
C1355	8.3	7.8	9.0
C1908	7.7	7.4	7.9

Figure 11: Comparing the LPC-set, BACKSIM, and hybrid BACKSIM strategies. The compiled simulators were expressed in C code, timings are seconds per 5000 input vectors. The X indicates where the C compiler blew up.

about because the input simulator is written in a high-level language, yielding ease of construction, modification, and debugging. Partial evaluation provides the missing link between the two kinds of simulator implementations: the versatile interpreted method and the efficient compiled method.

Using the partial evaluator, we were able to implement and compare many different strategies for compiled simulations: PC-set, LPC-set, BACKSIM, and hybrid BACKSIM. We showed that LPC-set was superior to PC-set, that BACKSIM wasn't worth its overhead, and that the hybrid BACKSIM was best of all. Which one wins out is a function of compile time for a tuned compiler, the longer compile time required, the less useful the method. Because we are using a general program for generating the compiled simulations, we cannot estimate what compilation time would be when using a special purpose program for creating compiled simulators. (The partial evaluator we are using for our experiments runs very slowly, it's not fast enough to routinely generate compiled simulators.)

Future work along this line is to try to generate compiled simulators for other algorithms such as event-driven simulations. This involves some complication because we are then specializing a *data-dependent* program. The partial evaluator will have to deal with more side-effects, such as those on the state vector, at partial evaluation time. Although FUSE is capable of dealing with side-effects, and we did produce correct code for a data-dependent simulator, the code does not seem promising enough to deliver the high performance we are after.

Another point is the flexibility of the input structure that we allow in the partial evaluation. Only the input-values are allowed to vary in the residual code. The input structure must be completely static data. This arrangement of course produces the high performance we achieved in our experiments. But once again, we are trading efficiency for flexibility. To achieve this flexibility, in addition to producing less efficient code, the partial evaluator must handle more relaxed promises about the input data.

Finally, we could produce even higher performance compiled simulators by supplying actual input values at compilation time. For example, one could generate a compiled simulator that had the clocking strategy built into it. This would allow many more simplifications to occur at compile time and dramatically boost performance.

## References

- [1] M. Chiang and R. Palkovic, "LCC simulators speed development of synchronous hardware," *Computer Design*, Mar. 1, 1986, pp. 87-91.
- [2] D. M. Lewis, "Hierarchical Compiled Event-Driven Logic Simulation," *Proceedings of ICCAD-89*, pp. 498-500.

- [3] S. P. Smith, M. R. Mercer, B. Brock, "Demand Driven Simulation: BACKSIM," *Proceedings of the 24th Design Automation Conference*, 1987, pp. 181-87.
- [4] P. M. Maurer, Z. Wang, "Techniques for Unit-delay Compiled Simulation," *Proceedings of the 27th Design Automation Conference*, 1990, pp. 480-84.
- [5] R. E. Bryant, D. Beatty, K. Brace, K. Cho, T. Sheffler, "COSMOS: A Compiled Simulator for MOS Circuits," *Proceedings of the 24th Design Automation Conference*, 1987, pp. 9-16.
- [6] Z. Barzilai, J. L. Carter, B. K. Rosen, J. D. Rutledge, "HSS-A High-Speed Simulator," *IEEE Trans. on Computer-Aided Design*, 6(4), July, 1987, pp. 601-16.
- [7] Z. Wang, P. M. Maurer, "LECSIM: A Levelized Event Driven Compiled Logic Simulator," *Proceedings of the 27th Design Automation Conference*, 1990, pp. 491-96.
- [8] H. Abelson, G. J. Sussman, *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, MA, 1985.
- [9] D. Weise, *Hierarchical Formal Multilevel Verification of Digital MOS/VLSI Circuits*, PhD. Thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory Technical Report 978, Cambridge, MA, 1984.
- [10] A. Berlin, D. Weise, "Compiling Scientific Code Using Partial Evaluation," *Computer*, IEEE, 23(12), pp. 25-37, 1990.
- [11] D. Weise, "Graphs as an Intermediate Representation for Partial Evaluation," Stanford University, Computer System Laboratory Technical Report CSL-TR-90-421, Stanford, CA, 1990.
- [12] Brglez, Franc, P. Pownall, R. Hum, "Accelerated ATPG and Fault Grading Via Testability Analysis," *Proceedings of the International Symposium on Circuit and Systems*, 1985.