Sub-Nanosecond Arithmetic III

M. J. Flynn, B. Wooley, G. De Micheli

Stanford University

Results from Prior NSF Support

In recent years, computer applications have increased in their computational complexity. The popularity of 3D graphics, video compression and decompression and signal processing have forced processor designers to pay careful attention to arithmetic and especially floating point operations. Of course, all of this is done in the context of optimizing the area–time tradeoff required by VLSI chip design.

Our SNAP II work can be separated into two areas: (1) Basic issues in technology and (2) Improving floating-point operations. Most of our SNAP papers and dissertations are available from our web site (`http://umunhum.stanford.edu`).

Technology Issues

1. The completion of a wave pipelined vector processor [T6]. Wave pipelining is a technique we developed earlier, based on latchless pipelines where data is stored in the guaranteed minimum path delay ensured by sophisticated CAD tools. We applied our tools to a vector processor and showed that cycle time improvements of about three times could reasonably be expected even when process and environmental variations were considered.

2. We completed a study of scalable circuit models for CMOS circuits and, using this model, we were able to show that one can automate the process of the design of optimal multi-level cache organizations [T12]. The optimization process uses the physical process parameters such as feature size to modify both circuit design and line lengths to optimize cache cycle time performance. A report on this work was awarded a "best paper" conference award.

3. We completed a study of support for multimedia applications [T11], [P33]. This study showed that software-only technology could be used to improve the performance of MPEG type multimedia applications. The techniques developed use subword parallelism within a standard unmodified floating point unit.

Floating Point Operations

Algorithms tie together technology, CAD, and software into an implementation which suits a particular application. In this section, we review SNAP advances in familiar floating point arithmetic operations.

Floating Point Add

Floating-point add basically consists of five steps: (1) Exponent difference, (2) (Pre) shift for significand (fraction) alignment, (3) Significand add/subtract, (4) (Post) shift for significand realignment, (5) Round and align. These five steps can be reduced to four by observing that a long (pre-)shift and a long (post-)shift are mutually exclusive.

In an early SNAP result, we used compound adders to replace the final addition which is required for rounding [T1]. The compound adder takes x and y as inputs, and produces both $x + y$ as the sum, and $x + y + 1$ as an augmented sum. For most cases of rounding, one of these is sufficient to give the final correct result of a floating point addition. Control circuitry selects which of the possible precomputed outcomes is the correct one.

Recent SNAP work by Oberman [T8], [P18] takes this one step further, using variable latency algorithms. This approach recognizes that there are many common cases where the result could be available early. For example, in certain cases, the two operands have the same exponent and the operation is addition, so only the simple left shift could be required. In another case, if the exponents are quite far apart, the smaller number simply slips out of range of the larger number and, depending on rounding, perhaps no addition may be necessary. In each of these cases possible simplification can be detected early and functional speedup is possible. For example, instead of a three-cycle floating addition, it might be possible to do addition in one or two cycles for at least some of the operands, and three cycles in the worst case. Oberman shows that a speedup of 1.33 is possible.

Floating Point Multiply

In our SNAP work, Al-Twaijry [T13], [P14] has completed an important study on the possible ways to implement floating point multiplication. By doing a complete design and layout for almost 1,000 separate multiply implementations, he examined the effect of various algorithmic and implementation approaches on multiplier performance. Among the most interesting findings is that the algorithmic selection is a function of certain technology parameters such as the feature size. At large feature sizes such as 1 micron, the overall multiplier delay is determined primarily by gate delays through the circuits themselves. As feature sizes shrink, the wires determine the overall delay for the multiplier implementation. Thus, techniques that shorten the wires produce the fastest multiplier. In a typical multiplier implementation, most of the circuitry and the wires are in the partial product reduction step. In order to optimize delay:

1. The designer can use a Booth encoding of the multiplier. By reducing the number of partial products, we shrink the size of the partial product reduction tree, even though we increase the circuitry necessary to generate the partial products. At small feature

sizes, the wires in the partial product reduction tree dominate the delay, and hence Booth encoding is preferable.

2. In implementing the partial product reduction tree, Al-Twaijry shows that an algorithmically laid out tree of counters using CAD tools in a so-called Wallace tree is better than the more regular "4, 2" compressor layout, which constructs a balanced binary tree of counters.

Floating Point Division

Division is a time-consuming operation. Even under the best cases, division using multipliers based upon Newton-Raphson methods requires between 8 and 10 cycles to perform a double-precision floating point division. The question for the floating point division implementor is: how fast does a divider really have to be? Is it possible to arrange the code so that the use of the quotient does not occur until a number of instructions later?

Oberman [T8], [P19] has performed an extensive study of the latency tolerance of floating point division. He shows that for typical programs without any particular software support, the quotient is required about three instructions after the divide instruction. Thus, a division that takes, say, 10 cycles will cause a 7-cycle disruption on a machine that executes an instruction each cycle. He also shows that it is possible, by using a compiler to reorder code, to create more latency tolerance on behalf of the floating point divide operation. This improves the separation between the divide operation and the user using instruction to over 10 instructions.

In other SNAP division work, we have developed a novel approach to very high radix division [P23] and provided insight to efficient implementations of SRT division algorithms [P21].

Publications from SNAP II

[P1] H. A. Al-Twaijry, S. F. Oberman, S. T. Fu and M. J. Flynn. "The SNAP Project: Building Validated Floating Point Units." In *Proceedings of SCAN'97* Lyons, September 1997.

[P2] A. Bogliolo, L. Benini and G. De Micheli. "Adaptive Least Mean Square Behavioral Power Modeling," EDTC, *Proceedings of the European Design and Test Conference* pp. 404–410, Paris, 1997.

[P3] G. T. Byrd and M. J. Flynn. "Evaluation of Communication Mechanisms in Invalidate-Based Shared Memory Multiprocessors." Parallel Computer Routing and Communication Workshop, June 1997.

[P4] M. J. Flynn. "What's Ahead in Computer Design?" Invited keynote. In *Euromicro'97 Proceedings*, pages 4–9, Budapest, September 1997.

[P5] M. J. Flynn. "Time and Area Optimization in Processor Architecture." Invited keynote. In *Proceedings of ARCS'97*, pp. 1–9, Rostock, Germany, September 1997.

[P6] Steve T. Fu, Nhon Quach and Michael J. Flynn,. FUPA: Floating Point Unit Cost Performance Metric and its Application to Microprocessors. *IEEE Transactions on VLSI Systems*, 1995.

[P7] Steve T. Fu and Michael J. Flynn. CacheOpt–A High Level Synthesis Tool for On-Chip Cache Hierarchy Synthesis. *33rd Design Automation Conference*, 1996.

[P8] Steve T. Fu and Michael J. Flynn. Optimal On-Chip Cache Hierarchy Synthesis with Scaling of Technology. In *Proceedings of the 15th Annual IEEE International Phoenix Conference on Computers and Communications*, March 1996.

[P9] S. T. Fu, D. F. Zucker, and M. J. Flynn. "Memory Hierarchy Synthesis of a Multimedia Embedded Processor," in *Proceedings of International Conference of Computer Design*, Austin, October, 1996.

[P10] D. L. Harris, S. F. Oberman, and M. A. Horowitz. "SRT Division Architectures and Implementations," in *Proceedings of the 13th IEEE Symposium on Computer Arithmetic*, pages 18–25, July 1997.

[P11] F. Klass and A. van de Goor. "Fast multiplication in VLSI using wave pipelining techniques." *Journal of VLSI Signal Processing* 7:233–248, 1994.

[P12] Justin Leung, Masoud Zargari, Bruce A. Wooley, and S. Simon Wong. "Active Substrate Membrane Probe Card," *IEEE International Electron Device Meeting* (IEDM), pp. 709–712, 1995.

[P13] S. F. Oberman, H. Al-Twaijry, and M. J. Flynn. "Advances in High Performance Floating Point Unit Design," in *Proceedings of the 15th IMACS World Congress on Scientifc Computation, Modelling, and Applied Mathematics*, August 1997.

[P14] S. F. Oberman, H. Al-Twaijry, and M. J. Flynn. "The SNAP Project: Design of Floating Point Arithmetic Units," in *Proceedings of the 13th IEEE Symposium on Computer Arithmetic*, pages 156–165, July 1997.

[P15] Stuart F. Oberman and Michael J. Flynn. "Implementing Division and Other Floating-Point Operations: A System Perspective." In Proceedings of *SCAN-95* (International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics), Wuppertal, Germany, September 1995.

[P16] Stuart F. Oberman and Michael J. Flynn. "Reducing Division Latency with Reciprocal Caches." In *Proceedings of SCAN-95*, Wuppertal, Germany, September 1995.

[P17] S. F. Oberman and M. J. Flynn, "Reducing division latency with reciprocal caches," *Reliable Computing*, vol. 2, no. 2, pp. 147–153, April 1996.

[P18] S. F. Oberman and M. J. Flynn. "A Variable Latency Pipelined Floating-Point Adder," in *Proceedings of Euro-Par'96*, Springer LNCS vol. 1124, pages 183–192, August 1996.

[P19] S. F. Oberman and M. J. Flynn. "Design Issues in Division and Other Floating-Point Operations," *IEEE Transactions on Computers* 46(2):154–161, February 1997.

[P20] S. F. Oberman and M. J. Flynn. "Division Algorithms and Implementations," *IEEE Transactions on Computers* 46(8):833–854, August 1997.

[P21] S. F. Oberman and M. J. Flynn. "Minimizing the Complexity of SRT Tables," to appear in *IEEE Transactions on VLSI Systems*.

[P22] S. F. Oberman and M. J. Flynn. "Reducing the Mean Latency of Floating Point Addition," to appear in *Theoretical Computer Science*.

[P23] D. Wong and M. Flynn. Fast division using accurate quotient approximations to reduce the number of iterations. In *Proceedings of the 10th Symposium on Computer Arithmetic*, Grenoble, France, June 1991.

[P24] B. Wooley, M. Zargari, J. Leung and S. S. Wong. "A BiCMOS Active Substrate Probe Card Technology for Digital Testing," *ISSCC Dig. Tech. Papers* pp. 308–309, Feb. 1996.

[P25] Alice Yu, R. B. Lee, and M. J. Flynn. "An Evaluation of Fidelity Metrics," in *Compcon '97 Digest of Papers*, pp. 49–55, February 1997.

[P26] Alice Yu, R. B. Lee, and M. J. Flynn. "Early Detection of All-Zero Coefficients in H.263," in *Picture Coding Symposium Proceedings* pp. 159–164, Berlin, September 1997.

[P27] Alice Yu, R. B. Lee, and M. J. Flynn. "Performance Enhancement of H.263 Encoder Based on Zero Coefficient Prediction," in *ACM Multimedia '97 Proceedings*, pp. 21–29, Seattle, November 1997.

[P28] C. P. Yue, C. Ryu, J. Lau, T. H. Lee, and S. S. Wong. "A physical model for planar spiral inductors on silicon." *International Electron Devices Meeting Technical Digest* pp. 155–158, December 1996.

[P29] Masoud Zargari and Bruce A. Wooley, "A BiCMOS Active Pull-Down ECL Output Driver for Low Power Applications," *IEEE Symposium on Low-Power Electronics* Digest of Technical Papers, pp. 50–51, 1995.

[P30] Masoud Zargari, Justin Leung, S. Simon Wong, and Bruce A. Wooley. "A BiCMOS Active Substrate Probe Card Technology for Digital Testing," *IEEE International Solid-State Circuit Conference* (ISSCC) Digest of Technical Papers, pp. 308–309, 1996.

[P31] D. F. Zucker, M. J. Flynn, and R. B. Lee. "Improving Performance for Software MPEG Players," in *Proceedings of COMPCON 96*, February, 1996.

[P32] D. F. Zucker, M. J. Flynn, and R. B. Lee. "A Comparison of Hardware Prefetching Techniques for Multimedia Benchmarks," in *Proceedings of International Conference on Multimedia Computing and Systems*, Hiroshima, June 1996.

[P33] D. F. Zucker, R. B. Lee, and M. J. Flynn. "Achieving Subword Parallelism by Software Reuse of the Floating Point Data Path," in *Multimedia Hardware Architectures 1997*, San Jose, CA, February, 1997.

PhD dissertations under SNAP II

[T1] Quach, Nhon T. *Reducing the Latency of Floating-Point Arithmetic Operation*, 01/94.

[T2] Bewick, Gary W. *Fast Multiplication: Algorithms and Implementation*, 03/94.

[T3] Klass, E. Fabian. *Wave Pipelining: Theoretical and Practical Issues in CMOS*, 09/94 (Delft University of Technology).

[T4] Chao, Chin-Chieh. *Multiport Memory Design for an MCM Coprocessor*, 12/94.

[T5] Biellak, Stephen A. *Reactive Ion Etched Unstable and Stable Semiconductor Lasers*, 08/95.

[T6] Nowka, Kevin J. *High-Performance CMOS System Design using Wave Pipelining*, 09/95.

[T7] Johnson, John D. *Expansion Caches for Superscalar Machines*, 04/96.

[T8] Oberman, Stuart F. *Design Issues in High Performance Floating Point Arithmetic Units*, 01/97.

[T9] Zargari, Masoud. *A BiCMOS Active Substrate Probe Card Technology for Digital Testing*, 03/97.

[T10] Leung, Justin. *Active Substrate Membrane Probe Card*, 04/97.

[T11] Zucker, Daniel F. *Architecture and Arithmetic for Multimedia Enhanced Processors*, 06/97.

[T12] McFarland, Grant W. *CMOS Technology Scaling and Its Impact on Cache Delay*, 06/97.

[T13] Al-Twaijry, Hesham. *Area and Performance Optimized CMOS Multipliers*, 08/97.

Expected to complete by June 1998:

Steve T. Fu, *Cost Performance Optimization of Microprocessors.*
Alice C. Yu, *Improving Video Performance for Multimedia Applications.*