LETTERS TO THE EDITOR

Historical perspective offered on “Fast Computer Algebra”

Editor:

Regarding the article, “Fast Computer Algebra,” by Edmund A. Lamagna (Computer, September 1982), I would like to point out some excellent work in fast computational algorithms. It obviously started in the “Golden Age which was Greece” from which we derive our present square root algorithms. It was generalized in pseudo-division, pseudomultiplication form by Meggitt, IBM Journal, April 1962; D. S. Cochran, Hewlett-Packard Journal, September 1968; and specifically embodied by J. Yang et al. from Lockheed at the recent Custom Design Conference in Rochester, New York.

The ninth-century Arab mathematician, Al-Khwarizmi, from whose name the word “algorithm” is derived, and Henry Briggs in his Arithmetica Logarithmica, published in 1624, added insight along the way. These algorithms allow you to perform transcendental functions in essentially two multiply times: for example, the root extractor by J. Yang et al. performs 16-bit square root in 200 ns in nonclocked ripple, carry-forward, single-chip hardware. Exponentiation for noninteger values can be accomplished in a similar manner.

As integrated current density and performance increase, we find that many of the very esoteric problems can be solved with relatively inexpensive hardware with great precision and increased performance.

David S. Cochran
Vice-president
Integrated Circuit Engineering
Sunnyvale, California

Author’s reply:

One objective of my article was to convey a feeling for the nature of work in algorithm analysis and computational complexity. To do so, I considered several basic problems of computer algebra, presenting some new algorithms along with the more familiar, classical ones.

Although the methods discussed in my paper are perhaps best suited for implementation in software, it is also possible to study the efficiency of hardware algorithms from a similar perspective. D. S. Cochran’s letter points to some interesting work on two related numerical problems—root finding and the evaluation of transcendental functions. I wish to take this opportunity to pass along some additional references for readers desiring further information along these lines.

Books by D. J. Kuck and J. E. Savage present complexity analyses of algorithms for computer arithmetic from the standpoint of circuit size and depth. More recent work has focused on the area and time of VLSI implementations. The paper by M. J. Foster and H. T. Kung and Chapter 8 of the book by C. Mead and L. Conway provide a good introduction to the issues involved in the design of efficient algorithms for VLSI.

Volume 2 of D. E. Knuth’s encyclopedic series on The Art of Computer Programming is undoubtedly the best starting point for those interested in the analysis of algebraic and numeric algorithms. Another interesting book by L. I. Kronjsjö discusses algorithms for numerical problems (e.g., iterative methods and solving linear systems) from their convergence and stability properties, as well as efficiency.

Edmund A. Lamagna
Associate Professor
of Computer Science
University of Rhode Island,
Kingston


Ed. Note: Dr. Lamagna’s article contained the following production errors: (a) Figures 13 and 14 (pp. 54 and 55) were inadvertently placed with the wrong captions; (b) on p. 48, the second displayed equation should not contain “m(x)”; (c) on p. 49, the first displayed equation should have “i=0” under the second sigma, not “i=0”; and (d) p. 56, line 16 should read “O(n^{2.49})” not “O(m^{2.49}).” We apologize for these oversights and any misinterpretation they might have caused.
SEPM: “an excellent start”

Editor:

“Validating Solutions to Major Problems in Software Engineering Project Management” by Richard H. Thayer, Arthur Pyster, and Roger C. Wood (Computer, August 1982) was an excellent start at determining what constitutes a successful software development effort. Don’t stop!

There is, however, one bone I would like to pick. The following statement was made: “Projects that used software quality standards were not appreciably more successful than those that did use them.” Yet Table 7, which lists the software quality assurance standards, indicated much better than a 2:1 ratio of success to nonsuccess.

Being a proponent of software quality assurance and highly interested in the evaluation of the effectiveness (cost/benefit ratio) of applying software quality assurance standards and practices, I strongly question the “validity” of such a conclusive statement. Was it a typographical error and should it have read “Successful projects that used software quality standards were not appreciably more successful than those that did not use them”? 

Al Freund
Burroughs Corporation
Paoli, California

Author’s reply:

We thank Mr. Freund for reading our paper. The confusion he has with our classification is centered around a rather obscure footnote at the bottom of Table 6 on page 69.

Note that this table, as well as Tables 7 through 13 and 15 through 26, has a mean success rate of 67 percent based on all 60 projects reported. This means that 67 percent of the projects were successful regardless of reason. For us to judge that a factor contributed to a project’s success, it had to appear in more than 67 percent of the projects. We tried to be more precise in the statistical analysis of this data, but the small number of projects in the vast number of variables made a more precise analysis impossible. Your dilemma is understandable. We too would like to be able to conclude absolutely that software quality assurance standards contributed measurably to the project’s success. However, our data just did not bear up. Let’s keep trying.

Arthur Pyster
TRW
Redondo Beach, California