Computer Arithmetic: An Introduction and Overview

DANIEL E. ATKINS, III, AND HARVEY L. GARNER

MOST of the papers contained in this issue's special section on computer arithmetic had their origin in the Second Symposium on Computer Arithmetic held May 15-16, 1972 at the University of Maryland. The Symposium was sponsored by the Technical Committee on Computer Architecture (TCCA) of the IEEE Computer Society. The Guest Editors for this issue served as Chairman (H. L. Garner) and Program Chairman (D. E. Atkins, III) for the Symposium. There were 45 participants registered for the symposium and 30 papers were presented. The program was divided into sessions broadly titled: 1) theory of numbers and arithmetic, arithmetic codes, and residue arithmetic; 2) arithmetic algorithms and their implementation; and 3) control of precision and error in computation.

Unfortunately, due to space limitations, it was not possible to include all of the symposium papers in this issue's special section on computer arithmetic. The Guest Editors eliminated their own papers. Several other papers have been published elsewhere. Other papers were not included, either because time did not permit the necessary revision or because the research was in a state too preliminary for publication. As a result the papers in this special section of this issue do not reflect the breadth of research in digital computer arithmetic.

Recent activities in digital computer arithmetic can be categorized as theory, design (logic, algorithmic, or system), control of errors, and the control of precision in the computational process. These activities are for the most part motivated by the demands and requirements of the computer users. Diverse applications of computers have created real requirements for increased flexibility, increased speed, fault-tolerant computing, accurate estimation of precision, and economy. These requirements are not necessarily mutually compatible. In particular, economy is not necessarily compatible with either speed or fault-tolerant computing. The designer, the user, and the sponsor must determine the appropriate relative weight to be given to various requirements. The success of solid-state technology in providing economical and reliable components has had a profound effect on the design of digital computers and on digital computer arithmetic systems and logic. This technology has mediated, if not negated, the need to minimize the number of components and has permitted practical designs involving redundancy and pipelining. Even the so-called "brute force" design has in several instances been physically realized.

Spira's paper, "Computation Times of Arithmetic and Boolean Functions in (d, r) Circuits," is a theoretical paper concerned with the lower bounds on the time required to compute logical functions, including the standard arithmetic operations. The paper surveys the known research in this area. Included are previously unpublished results that have generally been known to specialists in this field.

The paper by Robertson and Trivedi, "The Status of Investigations of Computer Hardware Design Based on the Use of Continued Fractions," describes current research in the use of continued fractions. Practicality has not as yet been demonstrated, but the theoretical results are promising. The authors enumerate three fundamental requirements that a proposed representation in numbers must satisfy for hardware implementation. This paper is an interesting case study of the difficulties arising when a new data representation format is considered.

The paper, "Radix-16 Evalutation of Certain Elementary Functions," by Ercegovac describes algorithms for certain elementary functions based on the continued product form of representation. This paper is primarily concerned with higher radix implementations and is an extension of the work of DeLugish [1]. One of the earliest hardware implementations of the continued product representation is the CORDIC technique report by Volder [2]. This concept was further generalized by Walther [3]. An excellent paper on the subject of hardware generation of electrical functions is given by Chen [4]. This paper presents an interesting technique as well as an excellent summary of the field.

Solid-state electronics has allowed the designer to use many more components. One consequence of this is pipelining which is discussed by Hallin and Flynn [5]. The same trends have led to serious consideration of strictly combinational approaches of arithmetic. The correspondence by Jacobsohn in the special section of this issue describes a combinatoric division algorithm for fixed integers.

The paper by Avižienis, "Arithmetic Algorithms for Error-Coded Operands," describes in detail the arithmetic error-correcting algorithms used on the STAR computer developed by the Jet Propulsion Laboratory. The paper gives a good picture of the additional complexity required to implement error correction in an arithmetic unit.

The last five papers are concerned with the representation errors inherent in floating-point representation, bounds on and the control of roundoff error, and the effect of the number base on the errors. An excellent introduction to the advan-

Manuscript received February 2, 1973.
D. E. Atkins, III is with the Department of Electrical Engineering and Computer Engineering, University of Michigan, Ann Arbor, Mich.
H. L. Garner is with the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pa.

tages and pitfalls of floating-point arithmetic is given by Knuth [6]. In this volume he comments, "Unfortunately the design of such floating-point hardware usually includes some anomolies which result in dismally poor behavior in certain circumstances, and we hope that future computer designers will pay more attention to providing the proper behavior than they have in the past." These papers should be of great interest to the designers of floating-point arithmetic systems.

Experimentalists have traditionally assigned probable error to results. Determination of probable error in computed results is difficult because: 1) the standard floating-point format does not provide for errors; 2) the rounding process may not be ideal; and 3) input data may be correlated thus complicating analysis.

Significant digit arithmetic (SDA) [7] was developed for the representation and processing of inexact data inputs. SDA provides a data format identifying significant digits and is a solution for difficulty 1) mentioned above. If part of the computational process is concerned solely with exact inputs then SDA leads to improper representation. This difficulty is overcome using the algorithms presented by Metropolis in his paper, "Analyzed Binary Computing," given in this issue's special section. These algorithms have been implemented in the software of Maniac II and a hardware implementation is being considered.

Yohe in his paper, "Roundings in Floating-Point Arithmetic," maintains that no manufactured computer performs ideal rounding. Five different roundings are defined. These are: the standard rounding; truncation; augmentation; and two directed roundings. Yohe considers data formats and the effect of guard digits. Yohe suggests that an optimum floating-point arithmetic system should provide for true rounding and both directed roundings. If this is done then: 1) upper and lower bounds for propagated roundoff error for a sequence of machine operations are more easily and accurately computed and 2) optimum rounding results in greater accuracy. For example if optimum rounding is used then the machine calculated square root is either a machine representable number or one of two consecutive machine representable numbers bounding the square root. 3) Hardware implemented directed rounding permits interval arithmetic operations to be performed in $\frac{1}{5}$ to $\frac{1}{10}$ of the time required for a software implementation.

Marasa and Matula in their paper, "A Simulative Study of Correlated Error Propagation in Various Finite-Precision Arithmetics," consider errors associated with four arithmetic modes using both rounding and truncation with floating and logarithmic data formats. The computation model involves a random sequence of arithmetic operations applied to operands selected from a randomly generated set of input data. The cardinality of the data set remains constant since the result of each operation replaces one of the operands. This replacement process introduces correlated error in the data set. The result of the study shows that the error growth rate is highly dependent on the operator mix and only modestly dependent on the arithmetic mode.

Cody, in "Static and Dynamic Numerical Characteristics of Floating-Point Arithmetic," summarizes what is known about the dynamic and static characteristics of floating-point number systems. The static characteristics are the average and maximum relative representation error and the range of the machine representable numbers. Bases 2, 4, and 16 are studied. The general conclusion is that there is little difference between the static characteristics of binary and hexadecimal representation, though based solely on maximum relative representation error, binary representation is superior to hexadecimal representation. The average relative representation error is the least for base 4. The dynamic behavior is characterized by the accumulated roundoff error. Different rounding techniques plus different number of guard digits are considered.

Arithmetic systems differing only in binary and hexadecimal representation appear statistically to give the same computational accuracy but a base 4 system appears to give better accuracy than either binary or hexadecimal. Unbiased rounding is superior to truncation.

Almost maximum accuracy can be obtained with true unbiased rounding and two guard digits or with truncation and one guard digit for larger values of the base. With respect to the choice of base, Cody concludes, "There may be valid reasons for prefering binary over hexadecimal, but the inferior performance of current hexadecimal machines appears to be due to other design considerations. It appears that the best machine design would use unbiased rounding with two guard characters and possibly a base 4 representation."

The final paper by Brent, "On the Precision Attainable with Various Floating-Point Number Systems," contains results obtained from both analytic analysis and simulation. Parts of this paper appear in summary form in the previous paper by Cody. Brent shows that the error due to truncation is twice as much as that due to rounding. He also shows that bases higher than 8 are inferior to base 4 on an rms relative error criterion. The paper contains an interesting discussion in choice of number base and the number of guard digits. The list of references is quite complete. It should be very useful for anyone wishing to study the field of floating-point arithmetic error.

Despite the fact that arithmetic is something learned by children and now taken for granted by most computer users, it is still a lively subject undergoing innovative development. It is hoped that this issue's special section on computer arithmetic will help substantiate this viewpoint and also make a contribution to this topic that has played such an important part in the history of the world.

Acknowledgment

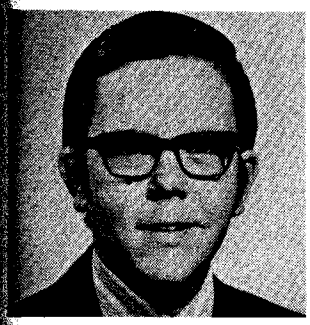
The Guest Editors wish to take this opportunity to thank the reviewers whose efforts made this issue possible. We also wish to extend appreciation to Prof. M. J. Flynn for his advice, patience, and help.

References

[1] B. G. DeLugish, "A class of algorithms for automatic evaluation of certain elementary functions in a binary computer," Dep. Comput. Sci., Univ. Illinois, Urbana, Rep. 399, June 1970.

[2] J. E. Volder, "The CORDIC trigonometric computing technique," *IEEE Trans. Electron. Comput.*, vol. EC-8, pp. 330-334, Sept. 1959.

[3] J. S. Walther, "A unified algorithm for elementary functions," in *1971 Spring Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 38. Montvale, N.J.: AFIPS Press, 1971, pp. 379–385.
[4] T. C. Chen, "Automatic computation of exponentials, logarithms, ratios, and square roots," *IBM J. Res. Develop.*, vol. 16, pp. 380–388, July 1972.
[5] T. G. Hallin and M. J. Flynn, "Pipelining of arithmetic functions," *IEEE Trans. Comput.* (Short Notes), vol. C-21, pp. 880–886, Aug. 1972.
[6] D. E. Knuth, *The Art of Computer Programming: Seminumerical Algorithms*, vol. 2. Reading, Mass.: Addison-Wesley, 1969, ch. 4.
[7] N. Metropolis and R. L. Ashenhurst, "Significant digit computer arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-7, pp. 265–267, Dec. 1958.



Daniel E. Atkins, III (S'68-M'70) was born in Jacksonville, Fla., on April 12, 1943. He received the B.S. degree in electrical engineering from Bucknell University, Lewisburg, Pa., in 1965, the M.S. degree in electrical engineering, and the Ph.D. degree in computer science from the University of Illinois, Urbana, in 1967 and 1970, respectively.

Between 1963 and 1967 he held summer positions with the Freas-Rooke Computing Center, Bucknell University, and the U.S. Naval Ordnance Laboratory, Silver Spring, Md. While attending the University of Illinois he participated in the design of the Illinois Pattern Recognition computer (Illiac III) and conducted research in the area of computer arithmetic. He has published several papers evolving from this work. From January 1970 through May 1972 he was an Assistant Professor of Computer Science, Bucknell University. He is currently an Assistant Professor of Computer Engineering at the University of Michigan, Ann Arbor, where he is teaching in the area of digital system design and serving as Co-Chairman of the Computer Engineering Curriculum Committee. He is a member of the Systems Engineering Laboratory and a faculty participant in the graduate program in Computer, Information, and Control Engineering (CICE). He is active in the Technical Committee on Computer Architecture and recently served as Program Chairman for the Symposium on Computer Arithmetic held at the University of Maryland. His current research activities are in the area of design of high-level language machines and novel arithmetic processors.

Prof. Atkins is a member of Sigma Xi, Tau Beta Pi, and the Association for Computing Machinery. Within the ACM he is a member of SIGARCH and SIGCSE.

Harvey L. Garner (S'51-A'51-M'57-SM'71) was born in Lake, Colo., on December 23, 1926. He received the B.S. and M.S. degrees in physics from the University of Denver, Denver, Colo., in 1949 and 1951, respectively, and the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, in 1958.

From 1958 to 1970 he was a member of the faculty of the Department of Electrical Engineering and the Department of Computer and Communications Sciences, University of Michigan, serving as Acting Chairman of the Department of Computer and Communications Sciences from 1965 to 1967. In 1961 he was awarded a Ford Faculty Development Fellowship and was an ACM National Lecturer in 1965. At the University of Michigan he was involved with the special summer computer programs from 1956 to 1970. He served as Chairman of the Second Symposium on Computer Arithmetic in 1972 and General Chairman of the Islands of Applications Conference in Tokyo in 1972 and was the General Chairman of the First National Computer Conference held in June 1972. Since 1970 he has been the Director and Professor of the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia. He has been involved with digital computers since 1951, including the design of the MIDAC and MIDSAC computers at the University of Michigan. He has designed several special-purpose machines and has published papers in the area of computer architecture and digital computer arithmetic.

Dr. Garner is listed in *Who's Who in America* and *American Men of Science*. He is a member of the Association for Computing Machinery, Sigma Xi, Sigma Pi Sigma, Eta Kappa Nu, the American Association for the Advancement of Science, and the American Society for Engineering Education.