

tient. If m bits are used from the reciprocal the table requires 2^m entries and establishes m bits of quotient.

Since the table doubles in size for each additional bit of accuracy required in the initial quotient guess, small changes in the convergence rate per iteration may reflect substantial changes in the size of the starting table. Notice in almost all schemes the error is biased, hence it (or part of it) can be subtracted from the quotient, slightly reducing the average error. Referring back to Fig. 1, since we are approaching the root from the left side uniformly, we may predict ahead part of the distance for the next iteration. While this is attractive, the error bias may serve a useful function when left in the iterant. In certain cases it will serve to protect the integrity of integers (i.e., integer quotients will be preserved in their usual representation).

CONCLUSION

The problem of finding complexity or efficiency bounds for division is much more difficult than for add or multiply because of the multiplicity of approaches. The best known techniques require two basic arithmetic operations (add or multiply) to double the precision of the quotient. Even relatively small improvements in the convergence rate of a scheme can result in considerable hardware savings in the area of a starting table. The development of these techniques remains an open problem as does the application of non-Newtonian higher order iterations.

REFERENCES

- [1] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electronic Computers*, vol. EC-13, pp. 14-17, February 1964.
- [2] S. F. Anderson, J. G. Earle, R. E. Goldschmidt, and D. M. Powers, "IBM System/360 Model 91: floating-point execution unit," *IBM J. Res. Develop.*, vol. 11, pp. 34-53, January 1967.
- [3] R. E. Goldschmidt, "Applications of division by convergence," M.S. thesis, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Mass., June 1964.
- [4] M. J. Flynn, "Very high-speed computing systems," *Proc. IEEE*, vol. 54, pp. 1901-1909, December 1966.
- [5] H. H. Laughlin, "Large-number division by calculating machine," *Am. Math. Monthly*, vol. 37, pp. 287-293, 1930.
- [6] D. E. Knuth, "Seminumerical algorithms," in *The Art of Computer Programming*, vol. 2. Reading, Mass.: Addison-Wesley, 1969, p. 215.
- [7] M. Lehman, D. Senzig, and J. Lee, "Serial arithmetic techniques," *1965 Fall Joint Computer Conf., AFIPS Proc.*, vol. 27. Washington, D. C.: Spartan, 1965, pp. 715-725.
- [8] E. V. Krishnamurthy, "On optimal iterative schemes for high-speed division," *IEEE Trans. Computers*, vol. C-19, p. 227-231, March 1970.
- [9] J. F. Traub, *Iterative Methods for the Solution of Equations*. Englewood Cliffs, N. J.: Prentice-Hall, 1964.
- [10] R. K. Richards, *Arithmetic Operations in Digital Computers*. New York: Van Nostrand Reinhold, 1955.
- [11] M. V. Wilkes, D. J. Wheeler, and S. Gill, *The Preparation of Programs for an Electronic Digital Computer*. Cambridge, Mass.: Addison-Wesley, 1951.
- [12] R. J. Fieg, "Analysis of a computer divide algorithm," unpublished communication.
- [13] P. Rabinowitz, "Multiple precision division," *Commun. ACM*, vol. 4, p. 98, February 1961.
- [14] K. Kunz, *Numerical Analysis*. New York: McGraw-Hill, 1957.
- [15] D. Ferrari, "A division method using a parallel multiplier," *IEEE Trans. Electronic Computers*, vol. EC-16, pp. 224-226, April 1967.

High-Speed Computer Multiplication Using a Multiple-Bit Decoding Algorithm

H. LING, MEMBER, IEEE

Abstract—This paper presents a method of performing the binary multiplication beyond the scheme of multiple ADD and SHIFT. The binary multiplication algorithm will be discussed first, followed by block decoding method, logic implementation, hardware consideration, and two examples which are at the end of the discussion.

Index Terms—Block decoding technique, fast multiplication, high-speed computer logic, high-speed multiplication, parallel multiplication.

Manuscript received May 20, 1969; revised December 12, 1969, and February 22, 1970.

The author is with the Information Sciences Department, IBM Research Laboratory, San Jose, Calif.

INTRODUCTION

ONE problem which the computer field has been concerned with for many years is how to improve the process of binary multiplication beyond the technique of repetitive ADD and SHIFT.

Some methods have been proposed, all of which have some disadvantages. It was pointed out by Lamdan and Aspinall [1], for example, that the realization of simultaneous multipliers necessitates a large number of components. Recently, carry save adders have generally been used to increase the speed of multiplication. However, due to the re-

quirement of a large amount of hardware support [2], it is applicable only to a larger machine such as the 360/91.

The author [3] proposed a decomposition scheme to perform multiplication in 1966. This paper presents the detailed binary multiplication algorithm based upon a multiple-bit decoding technique. The sum (I) and difference (J) of the factors are assumed to have been computed in advance. A combinational logic module S is defined which has the transfer function

$$S(x) = x - \frac{x^2}{2},$$

if the set of n inputs is interpreted as a binary fraction x . If n is large enough to accommodate the entire length of I (or J), the product is obtained in three additions, as demonstrated by (20). If the number of bits exceeds n , then copies of S may be applied to segments of I and J in parallel and the results summed to form the complete product. In particular, if the total length of the product is $n \cdot 2^k$ bits, then a total of $1 + 2k$ additions are required.

THEORY

Let A and B be two fractions whose product is being sought (we note that with proper scaling A, B can be integers or even general floating-point numbers).

Lemma 1: For $\delta = 0$ or 1 , $f(\delta) = \delta$. (2)

Lemma 2: $f(x + y) = f(x) + f(y) + xy$. (3)

Lemma 3: $f(ny) = n^2 f(y) - yf(n - 1)$. (4)

From (4) $f(y)$ can be rewritten as

$$f(y) = (1/4)y + (1/4)f(2y). \quad (5)$$

Taking a binary fraction

$$x = \sum_{n=1}^k x_n 2^{-n},$$

where $x_n = 0$ or 1 , applying (2), (3), and (5) systematically, we have

$$f(x) = f(0.x_1x_2x_3 \cdots x_n) = \sum_{k=1}^n (1/4^k)(1 + 2x_k) \sum_{l=1}^{n-k+1} x_l + k - 1^{2^{-l}} \quad (6)$$

where $1 > x \geq (1/2)$, $x_1 = 1$.

Equation (6) can be rewritten as follows:

$$f(x) = D(x) + 2L(x) \quad (7)$$

$$= 3D(x) - 2O(x). \quad (8)$$

where¹

$$D(x) = (1/2)x - B(x) \quad (9)$$

$$L(x) = \begin{matrix} 0.0 & 0 & (x_1x_1) & (x_1x_2) & (x_1x_3) & (x_1x_4) & \cdots & (x_1x_n) \\ +0.0 & 0 & 0 & 0 & (x_2x_2) & (x_2x_3) & \cdots & (x_2x_n) \\ +0.0 & 0 & 0 & 0 & & \cdots & & (x_nx_n). \end{matrix} \quad (10)$$

$$O(x) = \begin{matrix} 0.0 & 0 & (x'_1x_1) & (x'_1x_2) & (x'_1x_3) & (x'_1x_4) & \cdots & (x'_1x_n) \\ +0.0 & 0 & 0 & 0 & (x'_2x_2) & (x'_2x_3) & \cdots & (x'_2x_n) \\ + \cdots + \\ +0.0 & 0 & 0 & 0 & & \cdots + & & (x'_{n-1}x_n). \end{matrix} \quad (11)$$

$$B(x) = 0.0 \ 0x_10x_20x_30x_4 \cdots 0x_{n-1}0x_n. \quad (12)$$

Then

$$AB = 2[f(I) - f(J) - (I - J)/2] \quad (1)$$

with

$$\begin{aligned} I &= (A + B)/2 \\ J &= (A - B)/2 \\ f(x) &= (1/2)x(x + 1). \end{aligned}$$

It should be noted that the time required to square a number will be equal to half of that required in the general multiplication case due to $J=0$. The purpose of this section is to decompose $f(I), f(J)$ into efficiently manageable form. The following lemmas are presented:

If both I and J are binary integers,

$$I = 2^n(0 \cdot i_1i_2i_3 \cdots i_n) = 2^ni \quad (13)$$

$$J = 2^m(0 \cdot j_1j_2j_3 \cdots j) = 2^mj. \quad (14)$$

the product AB can be rewritten as

$$AB = 2\{f[2^ni] - f[2^mj] - (I - J)/2\}. \quad (15)$$

By substituting (4) into (15), the product AB becomes

¹ Parenthesized terms represent bits expressed as logical AND functions, with x' representing logical complement of x .

