[6] L. E. Fogarty and R. M. Howe, "Trajectory optimization by a direct descent process," *Simulation*, September 1968.
[7] G. A. Korn, *Random-Process Simulation and Measurements*. New York: McGraw-Hill, 1966.
[8] L. A. Rastrigin, "The convergence of the random-search method," *Automation and Remote Control*, vol. 24, p. 1337, 1963.
[9] G. A. Korn, "Digital-computer interface systems," *Simulation*, December 1968.
[10] R. S. Gonzalez, "An optimization study on a hybrid computer," M.S. thesis, Department of Electrical Engineering, University of Arizona, Tucson, 1969.
[11] M. A. Schumer and K. Steiglitz, "Adaptive-step-size random search," *IEEE Trans. Automatic Control*, vol. AC-13, pp. 270–276, June 1968.
[12] H. Kosako, ACL Memorandum 185, Department of Electrical Engineering, University of Arizona, Tucson, 1969.

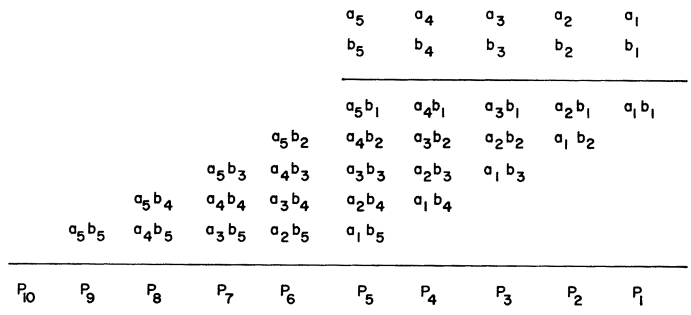

Fig. 1. The process of multiplying two 5-bit binary numbers.

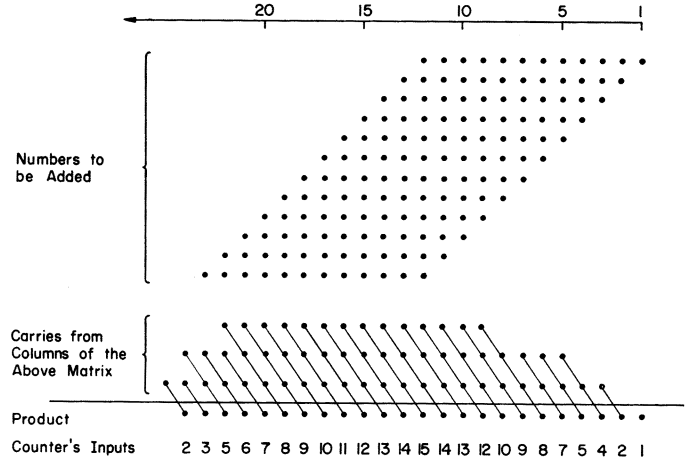

Fig. 2. Multiplication (12×12 bit) through addition, in a single stage, using a parallel counter for each column. Carries are propagated through the counters.

Fast Multipliers

A. HABIBI AND P. A. WINTZ

***Abstract*—A number of schemes for implementing a fast multiplier are presented and compared on the basis of speed, complexity, and cost. A parallel multiplier designed using the carry-save scheme and constructed from 74 series integrated circuits is described. This multiplier multiplies 10-bit by 12-bit binary numbers with a worst-case multiplication time of 520 ns. The cost of the integrated circuits was less than $500.**

***Index Terms*—Dadda's multiplier, digital multipliers, fast multipliers, parallel multipliers, simultaneous multipliers, Wallace's multipliers.**

I. Introduction

The operations required to multiply two 5-bit binary numbers are illustrated in Fig. 1. The 25 summands a_ib_j $i, j = 1, 2, 3, 4, 5$ must be formed and then added by columns (with carries) to form the product $P_{10} \cdots P_2P_1$. Note, for example, that if $a_1 = a_2 = a_3 = b_1 = b_2 = b_3 = 1$ then two carries are produced in the generation of P_3 so that a total of six entries must be added to generate P_4.

For an n-bit multiplicand $a_n \cdots a_2a_1$ and m bit multiplier $b_m \cdots b_2b_1$ the nm summands a_ib_j $i = 1, 2, \cdots, n; j = 1, 2, \cdots, m$ can be generated in parallel by using nm NAND gates and complementing their outputs. Hence, the basic problem in designing a high speed multiplier is to reduce the time required to add the summands. The summand matrix for a 12-bit multiplier is shown in Fig. 2.

The number of summands can be made less than nm by using some simple multiples of the multiplicand on the basis of two or more multiplier digits. This reduction will not be considered here, as the schemes studied will work also for a reduced number of summands.

Before proceeding we point out that two binary numbers of arbitrary sign can be multiplied by a positive multiplier by taking the proper complement of the negative inputs and taking the proper complement of the output if the inputs had different signs.

In Section II we present a brief discussion of three schemes that have been proposed for adding the summands. In Section III we make a detailed comparison of these three schemes on the basis of speed, complexity, and cost. A fast multiplier that was designed and constructed at Purdue University is described in Section IV.

Manuscript received March 7, 1969; revised September 23, 1969. This research was supported by NASA Grant NGR 15-005-087.
The authors are with the Department of Electrical Engineering, Purdue University, Lafayette, Ind.

II. Three Schemes for Fast Multipliers

In this section we present a brief description of three schemes for implementing a parallel multiplier.

A. Dadda's Scheme

The most elementary scheme for adding the summands was proposed by Dadda [1]. In this scheme parallel (p inputs, q outputs) counters are used to obtain the sum of 1's in each column of the summand matrix (see Fig. 2). The input of the ith counter (for the ith column) consists of the elements in the ith column of the summand matrix and also the outputs (carriers) from the lower order counters. The diagonal line segments at the bottom of Fig. 2 indicate how the carries produced by each counter are fed to the higher order counters. The time required for the addition of summands using this scheme is rather large due to the complexity of a counter with a large number of terminals. In order to minimize the delay, the process can be divided into two steps. In the first step, from the original set of summands a set of two numbers is obtained whose sum equals the product. In the second step, the product is formed by adding these two numbers. The first step can be accomplished without carry propagation in the following way. First the original matrix is

transformed by means of parallel counters into a second matrix with a smaller number of rows composed of the output of the counters. This matrix in turn is transformed to a matrix with a smaller number of rows in the same manner. The process is continued until a two-row matrix results.

An essential feature of this procedure is that during the phase of reduction of summands some carry propagation can already be taking place at the right-hand end of the double product. As a result the effective time lag for the addition of the final two numbers is shorter than its apparent value.

Dadda [1] also introduced a procedure for reducing the summands to two numbers using full adders, i.e., (3, 2) counters. His procedure is optimum in the sense that it uses a minimum number of full adders. He also achieves the minimum time lag required to complete the reduction. The time lag increases as the logarithm of the number of the bits in the multiplier.

B. Wallace's Scheme

A different procedure for reducing the summands was proposed by Wallace [2]. He uses strings of (3, 2) counters that always take groups of *three* rows reducing them to *two*. This increases the number of stages that is necessary for the reduction of the summands. It could be avoided by simply using some of the counters as half adders to shift some particular summands to higher order positions. It is this form of Wallace's technique that we consider here. The difference between this technique and Dadda's technique is in their ways of connecting the full adders.

C. The Carry–Save Scheme

Another scheme for adding the summands uses carry–save adders. This scheme is discussed in Braun [3]. A faster version of this scheme can be realized by using the two step procedure already discussed. In this scheme the first state for the reduction of n^2 summands uses $n-1$ half adders to generate the second least significant bit of the product. (The least significant bit is available without any transformation.) Each one of the following states uses $n-1$ full adders. In each stage one bit of the product is obtained and the number of summands is reduced by one. After $n-1$ stages the n least significant bits of the product are available and the number of summands is reduced to two $(n-1)$-bit numbers. In this scheme going from n to $n+1$ bit numbers requires one additional half adder and $2n-1$ additional full adders. Thus a multiplier requires a total of n half adders and n^2-2n full adders to obtain the final product from the summands. This is the same number of full adders required by Dadda's optimum scheme, but it is slower because in general it requires more stages for the reduction of summands. However, it has the advantage of producing two $(n-1)$-bit numbers in the final stage rather than the two $(2n-2-\ln n/\ln 1.5)$-bit numbers produced in the final stage by Dadda's system.

III. System Comparisons

In this section we compare the three schemes discussed in Section II on the basis of speed, complexity, and cost. We also present a block diagram of each system to indicate the different constructions.

A detailed comparison of two different variations of each of the three multipliers described in Section II is presented in Table I. These numbers are for a 12-bit plus sign multiplicand and a 12-bit plus sign multiplier.

The procedure for generating the summands in all three schemes is the same. The required time T_G for this process is the propagation delay through the RING SUMS (EXCLUSIVE OR circuits) and the NAND gates. Denoting the propagation delay of a logic gate by Δ, the delay time T_G is 3Δ. For multiplying two n-bit numbers this process requires n^2 NOR gates and $2n$ ring sums that can be constructed from $8n$ NAND gates.

The procedure for the reduction of the summands is the basic difference between the three schemes. The time required for this process T_{RD} is proportional to the number of stages needed to reduce the summand matrix to two numbers. Each stage corresponds to one unit of propagation delay through a full adder Δ_F. Both Dadda's and Wallace's schemes require the same number of stages. Table II gives a list of the number of stages versus n the number of bits in the multiplier for a multiplier using Dadda's scheme. It is evident from Table II that in Dadda's and Wallace's schemes the number of stages increase almost as the logarithm of n whereas in the carry–save scheme the number of stages is $n-1$. In general for Wallace's and Dadda's schemes the number of stages is roughly equal to $\ln n/\ln 1.5$. Thus Dadda's and Wallace's schemes are much faster than the carry–save scheme for large values of n.

Both Dadda's and the carry–save schemes are optimum in the sense of using a minimum number of full adders. Wallace's technique in general uses more full adders.

The simplest form of an adder is a ripple adder—a string of full adders connected together. The addition time of a ripple adder for adding two n bit numbers is $(n-1)\Delta_c$ seconds where Δ_c is the carry delay of a full adder. Therefore the time lag T_{AD} for adding the final two numbers in Dadda's and Wallace's schemes is about $(2n-2-\ln n/\ln 1.5)\Delta_c$ seconds. A faster adder, the more complicated carry–look-ahead adder, is considered in detail in [3] and [4]. In a carry–look-ahead adder the carries at different levels are generated simultaneously. This eliminates the carry propagation delay and the only delay will be a fixed delay of 6Δ plus a delay of T_c which is the propagation delay through multi-input AND gates used to generate the carries. Using AND gates with a maximum of eight inputs the delay T_c for adding two n bit numbers is $2k\Delta$ where k is the smallest integer larger than or equal to $(n-1)/8$. The addition time in a carry–look-ahead adder increases only slightly as the number of bits to be added increases; hence it is ideal for addition of large numbers of bits. The speed im-

TABLE I

COMPARISON OF DIFFERENT SCHEMES FOR PARALLEL MULTIPLIERS FOR MULTIPLYING TWO SIGN + 12-BIT NUMBERS*

Type of Multiplier	Type of Adders	$T_G+2\Delta$	T_{RD}	T_{AD}	Total Multiplication Time (ns)	Number of Logic Gates for Summands	Number of Full Adders	Number of Multi-input Gates for CLA Adder	Cost of Integrated Circuits
Wallace	CLA Adder	5Δ	$5\Delta_F$	12Δ	420	336	136	315	\$1125
	RA	5Δ	$5\Delta_F$	$4\Delta_F$	415	336	159	0	\$ 797
Dadda	CLA Adder	5Δ	$5\Delta_F$	12Δ	420	336	110	315	\$1060
	RA	5Δ	$5\Delta_F$	$4\Delta_F$	415	336	132	0	\$ 708
Carry–Save	CLA Adder	5Δ	$11\Delta_F$	10Δ	635	336	120	153	\$ 712
	RA	5Δ	$11\Delta_F$	$3\Delta_F$	625	336	132	0	\$ 596

* The cost of the integrated circuits is obtained from the Texas Instruments Price List published in September, 1967.

TABLE II

NUMBER OF STAGES REQUIRED FOR A PARALLEL MULTIPLIER VERSUS NUMBER OF BITS OF THE MULTIPLIER n

Number of bits in the multiplier	Number of stages
3	1
4	2
$4<n\leq 6$	3
$6<n\leq 9$	4
$9<n\leq 13$	5
$13<n\leq 19$	6
$19<n\leq 28$	7
$28<n\leq 42$	8
$42<n\leq 63$	9

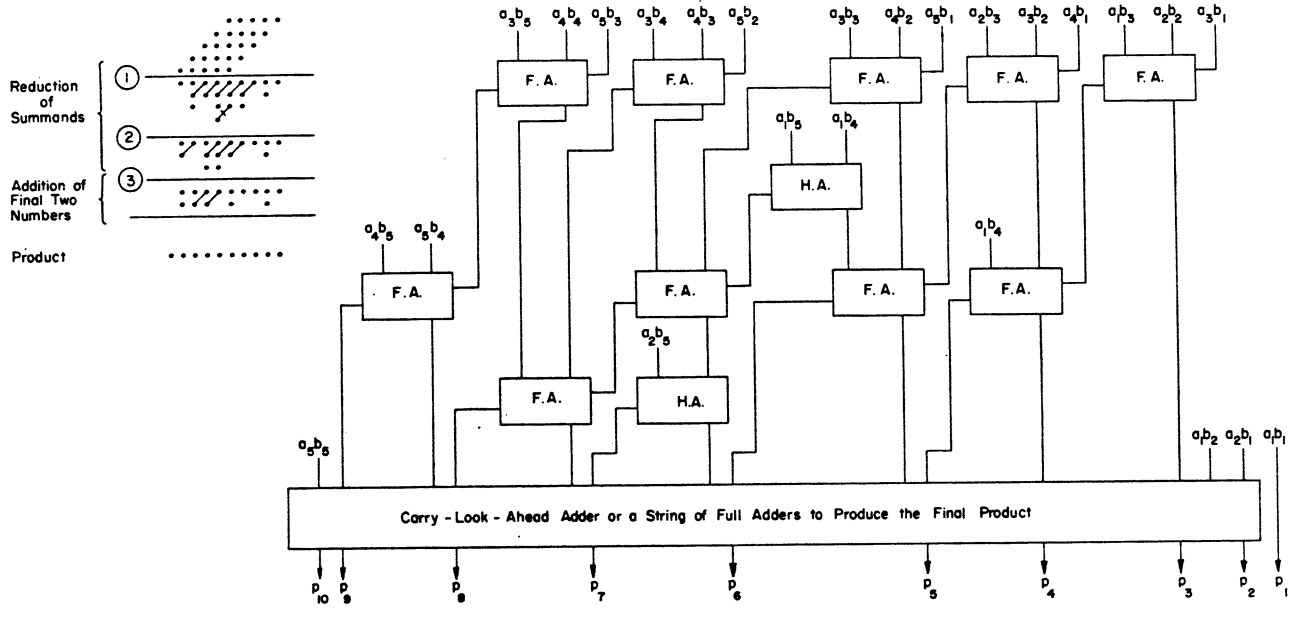

Fig. 3. Block diagram of a 5×5 multiplier using Wallace's scheme.

provement over a ripple adder is not significant for small numbers of bits.

Taking the complement of the output of the positive multiplier adds 2Δ to the multiplication time and requires $8n$ additional NAND gates.

Table I was prepared using the propagation delay and cost of integrated circuits as published by Texas Instruments in September of 1967 for 74 TTL series. For this series the delay constants introduced above are $\Delta=13$ ns, $\Delta_F=40$ ns, and $\Delta_c=10$ ns.

Block diagrams of each of the three multipliers are presented in Figs. 3, 4, and 5. In the block diagram of Wallace's scheme the carry inputs are used as a terminal for adding three bits. In this scheme the carry

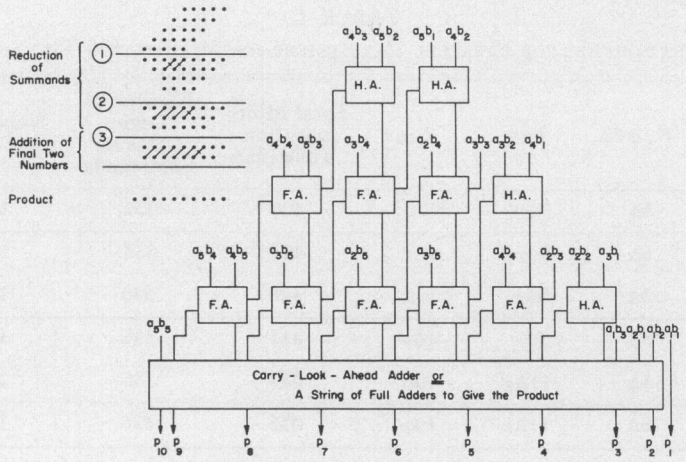

Fig. 4. Block diagram of a 5×5 multiplier using Dadda's scheme.

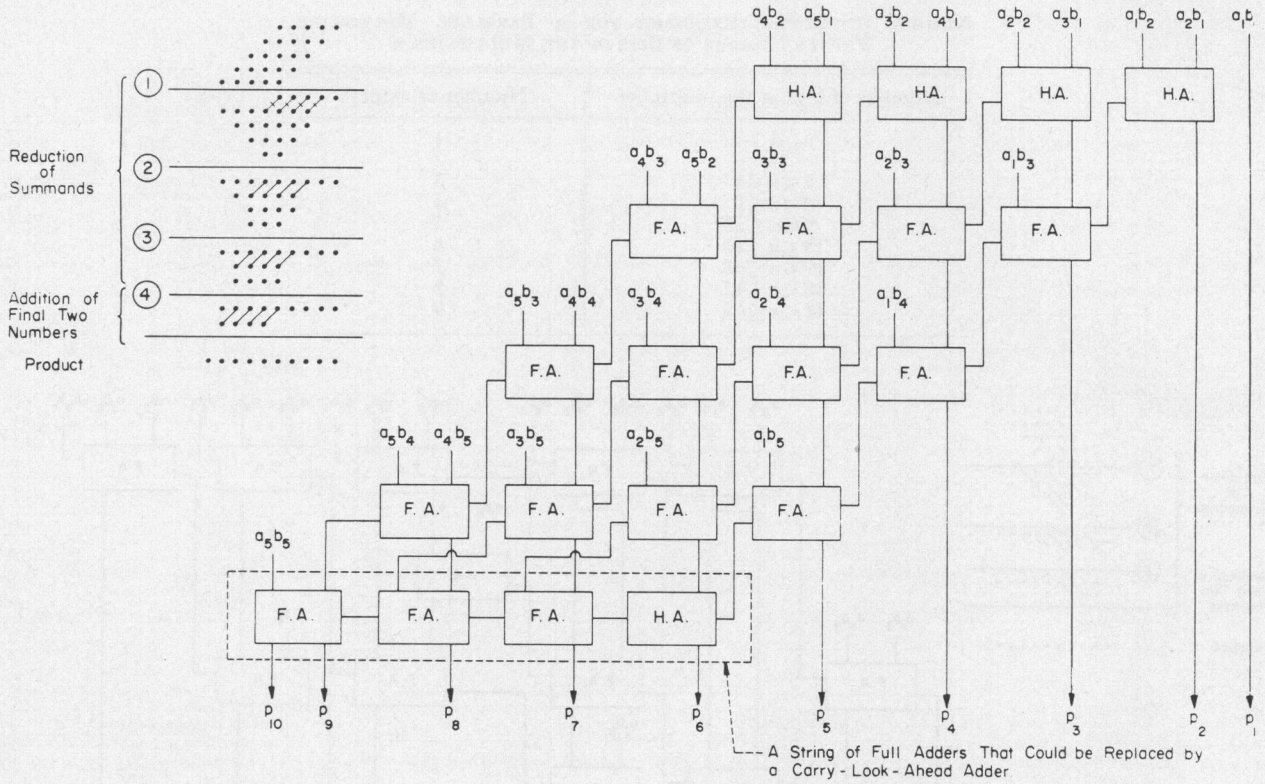

Fig. 5. Block diagram of a 5×5 multiplier using carry-save technique.

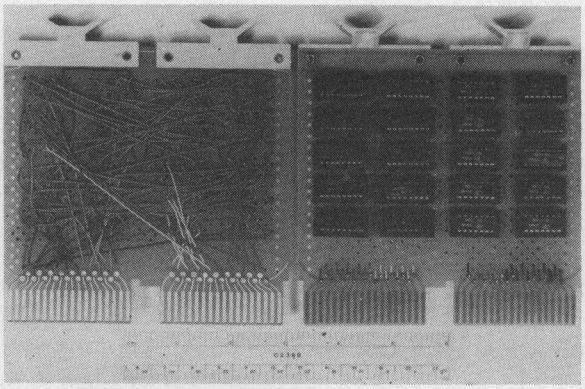

Fig. 6. The front and back side of one card of the Purdue fast multiplier. Three cards are required for the 10-bit by 12-bit multiplier.

output of one adder is not fed into the carry input of the following adder so that most of the adders have to be implemented using a single adder or two-bit adders rather than the more economical 4-bit adders. Dadda's scheme is more systematic and more 4-bit adders are used in full capacity. As shown in Fig. 5, the structure of the carry–save multiplier is very systematic and well suited for using 4-bit adders.

We conclude that if the number of bits in the multiplicand or multiplier is small, the carry–save scheme using a ripple adder is favored over the other two schemes. It is more economical, less complicated, and the difference in speed is not significant. However, as n increases, the Dadda's multiplier gets increasingly faster than the carry–save multiplier. It uses fewer components than Wallace's multiplier and operates at the same speed; hence it is always better than Wallace's multiplier.

IV. The Purdue Fast Multiplier

In this section we describe the design, construction, and performance of two fast multipliers implemented at the PCM Telemetry Laboratory, Department of Electrical Engineering, Purdue University. These identical multipliers were designed as parts of two digital filters that, in turn, are part of an experimental PCM system [5]. The experimental PCM system is an implementation of the optimum PCM system suggested by Wintz and Kurtenbach [6].

The multiplier is designed to multiply a 12-bit multiplicand by a 10-bit multiplier. It uses the carry–save scheme of reducing the summands and a ripple adder to add the final two numbers. The unit was constructed from Texas Instruments 74 series integrated circuits. The wiring was done on perforated double-size boards manufactured by Digital Equipment Corporation and dual-in-line integrated circuit sockets are used for ease in replacement of modules (see Fig. 6). The whole multiplier occupies three double-size boards and because of limitations in the number of input and output connections it is convenient that parts of the elements of the summand matrix are generated along with the circuitry producing the final product on each board.

The worst case multiplication time is about 520 ns; this is in agreement with the 495 ns predicted. The multiplication time could be decreased to 460 ns by replacing the 74 series logic gates with 74 H series since the propagation delay of the 74 H series is only half that of the 74 series. This would increase the cost of the multiplier from \$456 to \$550. The multiplication time could be decreased to 410 ns by using an 11-bit carry–look-ahead adder constructed from 74 H series for adding the final two numbers. This will boost the cost to \$710. Using Dadda's scheme this unit would have cost about \$770 and the multiplication time would have been 360 ns. Replacing the 74 series with the 74 H series would also reduce the multiplication time of Dadda's system to 270 ns while increasing its cost to \$1010.

Acknowledgment

The authors are grateful to the referees for their valuable suggestions, and in particular, to one of them for his very constructive criticisms.

References

[1] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, March 1965.
[2] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electronic Computers*, vol. EC-13, pp. 14–17, February 1964.
[3] E. L. Braun, *Digital Computer Design*. New York: Academic Press, 1963.
[4] F. Mowle, lecture notes on a course in digital computer design techniques, Purdue University, Lafayette, Ind., 1967.
[5] G. G. Apple and P. A. Wintz, "Experimental PCM system employing Karhunen–Loeve sampling," presented at the 1969 Internatl. Symp. on Information Theory, Ellenville, N. Y., January 1969.
[6] P. A. Wintz and A. J. Kurtenbach, "Waveform error control in PCM telemetry," *IEEE Trans. Information Theory*, vol. IT-14, pp. 650–661, September 1968.

On Range-Transformation Techniques for Division

E. V. KRISHNAMURTHY

***Abstract*—This note points out the close relationship between some of the recently described division techniques, in which the divisor is transformed to a range close to unity. A brief theoretical analysis is presented which examines the choice of quotient digit when this type of division technique is used for conventional and signed-digit number systems.**

***Index Terms*—Conventional and signed-digit number systems, deterministic generation of quotient digit, divide and correct methods, Harvard iterative technique, nonrestoring division, precision of multiplication, range transformation of divisor, Svoboda's method, Tung's algorithm.**

I. Introduction

Recently a number of techniques for division have been described which consist in initially transforming the divisor to a suitable range by premultiplication, so that the choice of the quotient digit is deterministic without any need for a trial and error process [1], [6]. Although all these techniques are closely related, since each one of them has been discovered independently and reported in different journals at about the same time, it is natural that little or no attention has been paid in bringing out the close relationship that exists between them. It is the object of this note to bring out this relationship and place all these techniques on a common basis with the hope that it would be useful for workers in this field.

II. General Development

Let us denote the dividend A and divisor B in floating-point form with integral mantissa in radix β thus:

$$A = \beta^{e_a} \cdot a = \beta^{e_a} \cdot \sum_{j=0}^{n} a_j \beta^j \qquad (1a)$$

Manuscript received February 20, 1969; revised June 27, 1969.
The author is with the Weizmann Institute of Science, Department of Applied Mathematics, Rehovot, Israel, on leave of absence from the Indian Statistical Institute, Calcutta, India.