A New Model for On-Line Arithmetic with an Application to the Reciprocal Calculation *

HENK J. SIPS

Delft University of Technology, Delft, The Netherlands

AND

HAI XIANG LIN

IBBC-TNO Netherlands Organization for Applied Scientific Research, Rijswijk, The Netherlands

A new model for on-line arithmetic functions is presented. It differs from the current models that in each iteration step the exact function value is used to generate the on-line digits. From this model the on-line properties of a large class of arithmetic functions can be determined. In general, the on-line properties of an arithmetic function derived from the model cannot be improved by any iterative approximation algorithm for that arithmetic function. The on-line properties of a number of standard arithmetic functions are given. It is shown that the above model of on-line computation can be easily implemented by means of a table look-up system. Furthermore, a table implementation can be used to start the on-line computation of an iterative approximation method. This is shown by an example, the reciprocal calculation, where the combination of a seed table and an adapted Newton-Raphson iteration method leads to a system with a low on-line delay and fast cycle times. The algorithm works for normalized, quasi-normalized, and pseudo-normalized numbers and can therefore be applied to chained on-line computations. © 1990 Academic Press, Inc.

1. INTRODUCTION

In on-line computations, the input operands, as well as the results, flow through arithmetic units in a digit by digit manner, starting with the most significant digit. An on-line algorithm is said to have an on-line delay of δ , if for the generation of the j th digit of the result, $(j + \delta)$ digits of the input operands are required [1].

In the conventional digit-serial arithmetic (i.e., least significant digit first), in general, all digits must be known in advance before a result (or a part of a result) can be generated. In conventional digit-parallel arithmetic units, high-

speed multioperand processing requires full precision bandwidth (as many bits as the number format) between the arithmetic units in a parallel or/and pipelined computational structure. This is not desirable or feasible as the complexity of the computation increases. In contrast to the conventional digit-parallel approach, on-line arithmetic [2] reduces the interconnection complexity between the processing units to a minimum of one digit per operand while still providing a good speedup ratio by overlapping or pipelining successive computations. Figure 1 illustrates the principle of on-line computation. It shows that digit on-line arithmetic units can be chained to obtain a fast throughput of digit-serial operands.

In recent years many algorithms for the on-line generation of arithmetic functions have been developed. Addition and multiplication are relatively simple [1]. Division and square root are more complicated [1, 3, 4, 5, 13]. In essence, two algorithmic models have been used to find suitable algorithms for the on-line calculation of an arithmetic function. The first model is based on the use of recurrence equations (R-model). In this model a partial result (or an equivalent function) is calculated each iteration step by using the partial result from the previous cycle [1, 2, 13]. The second model is based on function minimization (M-model). The function to be calculated is transformed to an equivalent function which is minimized each iteration step. Most of the algorithms in this model are based on the use of continued sum/product (CSP) algorithms [11, 3, 4, 5].

This paper deals with the question: what is the minimum on-line delay of an arbitrary arithmetic function? The general answer to this question is very difficult. Moreover, there are two ways of approaching the problem. One way is to construct an algorithm for the evaluation of a certain arithmetic function which, when executed by some machine, gives the lowest possible on-line delay for that function. We

* Parts of this work have been presented at the 8th Symposium on Computer Arithmetic, Como, Italy, 1987.

FIG. 1. Chained on-line computation.

call this upper bound the *attainable* on-line delay. The other way is to establish on-line delays which at least must hold for any on-line algorithm on any machine. We call this lower bound the *conditional* on-line delay. This does not necessarily imply that an algorithm satisfying this low bound can be found. Only when the conditional on-line delay and the attainable on-line delay of a function are proved to be equal has the *minimal* on-line delay of that function been found. Owens [9] has established conditional on-line delays for a few arithmetic operations through proving by example. However, Owens did not prove that his bounds are strict, nor does he give a general method for deriving such bounds for arbitrary functions.

In this paper the problem is approached from the algorithmic side. A new model for on-line function generation is presented: the E-model (Section 2). From this model on-line delays for continuous and differentiable functions can be derived.

Second, it is shown in Section 3 that the model can be implemented by a table look-up system, which need only generate two output digits per iteration cycle. In Section 4 it is shown that the E-method, besides of its theoretical value, can also be used (by a seed table) to start an M-method. The example is the use of a multiplicative M-method (Newton-Raphson) for reciprocal evaluation. It is shown that a low on-line delay can be obtained, while keeping a simple and fast on-line digit selection mechanism. The proposed algorithm is valid for all radices and accepts normalized, quasi-normalized, and pseudo-normalized input operands.

2. THE E-MODEL

2.1. Components of the Model

The E-model consists of three parts (Fig. 2): *function generation*, *on-line digit selection*, and *format alignment*. In the model $Z_n = F(X_n)$ denotes the exact result of an arithmetic function F on a set of N input operands X_n and $Z_i = F(X_i)$ denotes the i th approximation to Z_n ($1 \leq i \leq n$).

Figure 2 shows the operations within the i th iteration (for nomenclature, see the Appendix).

In the function generation part the required arithmetic function $Z_i = F(X_i)$ is calculated. In the on-line digit selection part the function value Z_i is rounded by some rounding function R^i and the result is truncated at the i th digit (denoted by T^i), yielding $\bar{Z}_i$ (i.e., $\bar{Z}_i = T^i\{R^i(Z_i)\}$). The rounding function R^i is defined as $R^i(Z_i) = Z_i + \text{Round} \cdot r^{-(i-\phi-1)}$, and $|\text{Round}| < 1$, where r^ϕ is the weight of the most significant digit of the final (redundant) result. From $\bar{Z}_i$ and $\bar{Z}_{i-1}$ the new on-line digit $\dot{z}_i$ is generated. The format alignment part takes care of the proper output format of the result (for example, the suppression of leading zeros).

The function arguments X_i are a set of N input numbers represented by a symmetric, radix- r redundant digit set $(-\eta, \dots, -1, 0, 1, \dots, \eta)$, where η is the redundancy factor ($r/2 \leq \eta \leq r-1$). The numbers X_i are recursively defined as

$$X_i = X_{i-1} + x_{i+\delta_j} \cdot r^{-i-\delta_j} \quad \text{with} \quad X_0 = \sum_{j=1}^{\delta_j} x_j \cdot r^{-i}, \quad (1)$$

where $j = \text{index}(x)$, $1 \leq j \leq N$.

The elements δ_j , $1 \leq j \leq N$, of δ represent the number of digits of the respective input operands which must be known in order to generate the first digit $\dot{z}_1$ of the result. From the definition of δ , it follows that the δ_j 's might have different values. Therefore, we define $\delta_g = \max(\delta)$ as the *generation on-line delay* of $F(X_i)$.

If the input numbers are floating-point numbers the definition given in [6] is used:

DEFINITION 1. A nonzero redundant floating-point number with an n digit mantissa,

$$P = \sum_{j=1}^n p_j \cdot r^{-j},$$

FIG. 2. The E-model for on-line arithmetic.

is said to be:

- (1) normalized if $r^{-1} \leq |P| < 1$;
- (2) quasi-normalized if $r^{-2} \leq |P| < 1$;
- (3) pseudo-normalized if $r^{-q} \leq |P| < 1$

with $3 \leq q \leq n$. ■

In the format alignment phase, the suppression of leading zeros introduces a delay δ_a , called the *alignment* delay. Besides delay, a simple digit conversion may also be necessary (e.g., to generate quasi-normalized results). The total on-line delay δ_F of $F(\mathbf{X}_i)$ is $\delta_F = \delta_g + \delta_a$. The on-line digits of the result are not produced in zero-time. The time it takes for the first digit of the result to become available, after the acceptance of the input operands, is called the *latency time*. In a synchronous system with a single-phase clock, an extra delay of one must be added to δ_F to obtain the latency delay: $\delta_L = \delta_F + 1$. Without losing generality, we restrict ourselves in the following discussion to the function generation and on-line digit selection phase. The format alignment phase can be treated independently and will be dealt with later on. First, an appropriate definition of on-line function representations within the E-model has to be given:

DEFINITION 2. Let $Z_n = F(\mathbf{X}_n)$ be the exact result of an arithmetic function F on a set of input numbers $\mathbf{X}_n$, and $\dot{z}_i \in \{-\eta, \dots, -1, 0, 1, \dots, \eta\}$ be the on-line digit generated in the i th iteration, then

$$\dot{Z}_i = \sum_{j=1}^i \dot{z}_j \cdot r^{-(j-\varphi-1)} \quad (i \geq 1)$$

is an on-line representation of Z_n if

$$\lim_{i \rightarrow \infty} |\dot{Z}_i - Z_n| \rightarrow 0, \quad (2)$$

where r^φ is the weight of the most significant digit of $\dot{Z}_i$. ■

This definition ensures that the on-line representation $\dot{Z}_i$ converges for $i \rightarrow \infty$ to the function value Z_n . Note that the definition is valid for all values of n ($n > 0$). From Definition 2, it follows that the representational error of $\dot{Z}_i$ is bounded by $|E(\dot{Z}_i)| = |\dot{Z}_i - Z_n| < (\eta/(r-1)) \cdot r^{-(i-\varphi-1)}$, which is less than 1 in the last digit position of $\dot{Z}_i$. This can be seen by observing that the difference between $\dot{Z}_i$ and $\dot{Z}_{i \rightarrow \infty}$ can be at most

$$|\dot{Z}_{i \rightarrow \infty} - \dot{Z}_i| = \sum_{j=i+1}^{\infty} \eta \cdot r^{-(j-\varphi-1)} = \left(\frac{\eta}{r-1} \right) \cdot r^{-(i-\varphi-1)}. \quad (3)$$

In other words $\dot{Z}_i$ can "correct" itself by maximally this amount in order to satisfy Eq. (2). It also follows that a sufficient convergence condition is

$$|\dot{Z}_i - Z_n| \leq \hat{K} \cdot r^{-i}, \quad (4)$$

where $\hat{K} \leq (\eta/(r-1)) \cdot r^{\varphi+1}$ is a bounded constant.

The above definitions do not define how to generate $\dot{Z}_i$ from Z_i . The most accurate basis for determining the increment to the previous result is to subtract the previously generated on-line result $\dot{Z}_{i-1}$ from the exact result Z_i , i.e., $Z_i - \dot{Z}_{i-1}$. To obtain a single digit as increment at the i th position $Z_i - \dot{Z}_{i-1}$ must be rounded to and truncated at the i th digit. Therefore,

$$\dot{z}_i = T^i\{R^i(Z_i - \dot{Z}_{i-1})\} \quad (5)$$

which is equal to

$$\dot{z}_i = T^i\{R^i(Z_i)\} - \dot{Z}_{i-1}. \quad (6)$$

By definition $\bar{Z}_i = T^i\{R^i(Z_i)\}$; therefore

$$\dot{z}_i = \bar{Z}_i - \dot{Z}_{i-1}. \quad (7)$$

Since $\dot{Z}_0 = 0$, it follows that $\dot{z}_1 = \bar{Z}_1$, $\dot{z}_2 = \bar{Z}_2 - \bar{Z}_1$, etc., resulting in

$$\dot{Z}_i = \bar{Z}_1 + (\bar{Z}_2 - \bar{Z}_1) + (\bar{Z}_3 - \bar{Z}_2) + \dots = \bar{Z}_i. \quad (8)$$

Thus, the above choice on generating $\dot{z}_i$ leads to the equality $\dot{Z}_i = \bar{Z}_i$. Therefore,

$$\dot{z}_i = \bar{Z}_i - \bar{Z}_{i-1}, \quad (9)$$

which gives a rationale for the choice in the E-model (Fig. 2) on the selection of $\dot{z}_i$.

Although Definition 2 completely defines the on-line conditions, more refined conditions are needed to establish a lower bound on δ_g . The following lemma gives a local (i.e., single step) condition for the result $\dot{Z}_i$ being on-line.

LEMMA 1. Given $Z_i = F(\mathbf{X}_i)$, then the on-line representation $\dot{Z}_i$ of Z_i can be generated if

$$|\dot{z}_i| = |\bar{Z}_i - \bar{Z}_{i-1}| < ((\eta+1)/r) \cdot r^{-(i-\varphi-2)}, \quad (10)$$

where $\bar{Z}_i = T^i\{R^i(Z_i)\}$.

Proof. From Eq. (8) it follows that $\dot{Z}_i - Z_n = \bar{Z}_i - Z_n = T^i\{R^i(Z_i)\} - Z_n$. By definition it holds that $Z_i = Z_n$ for $i \geq n$. Therefore, $T^i\{R^i(Z_i)\} = Z_n$ for $i \rightarrow \infty$, resulting in the on-line condition of Eq. (2) being satisfied. Finally, $\dot{z}_i$ must consist of a single digit, with weight $r^{-(i-\varphi-1)}$, for which $|\dot{z}_i| = |\bar{Z}_i - \bar{Z}_{i-1}| < ((\eta+1)/r) \cdot r^{-(i-\varphi-2)}$ is a sufficient condition. Q.E.D.

The applied rounding function $R^i(Z_i)$ directly influences the actual value of $|\bar{Z}_i - \bar{Z}_{i-1}|$. Of course, this difference

should be kept as small as possible. The next lemma states the best choice on R^i :

LEMMA 2. *If Z_i is represented in signed-digit form and $\text{sign}(Z_i) = 1$, if $Z_i \geq 0$, and $\text{sign}(Z_i) = -1$, otherwise, then*

$$R^i(Z_i) = Z_i + \text{sign}(Z_i) \cdot (r/2) \cdot r^{-(i-\varphi)} \quad (11)$$

is the best choice on all possible rounding functions R^i , in the sense that $|\bar{Z}_i - Z_i|$ is minimized.

Proof. The rounding function of Eq. (11) is the symmetrical rounding of Z_i to the i th digit. From Eq. (8) it follows that

$$|\dot{Z}_i - Z_i| = |\bar{Z}_i - Z_i| = |T^i\{R^i(Z_i)\} - Z_i|. \quad (12)$$

Let $Z_i = Z_i^A + Z_i^B$, where $Z_i^A = T^i(Z_i)$ and $Z_i^B = Z_i - T^i(Z_i)$, then Eq. (13) becomes

$$\begin{aligned} |\dot{Z}_i - Z_i| &= |T^i\{R^i(Z_i^A + Z_i^B)\} - Z_i| \\ &= |T^i\{R^i(Z_i^B)\} - Z_i^B|. \end{aligned} \quad (13)$$

If we normalize Eq. (13) such that the radix point is placed just to the right of the i th digit, then it holds that $|Z_i^B| < 1$. The nearest values of $T^i\{R^i(Z_i^B)\}$ are $(-1, 0, 1)$. From this it can easily be seen that by taking $R^i(Z_i) = Z_i + \text{sign}(Z_i) \cdot (r/2) \cdot r^{-(i-\varphi)}$ the minimal possible value of Eq. (13) is selected. Q.E.D.

From Lemma 1 and Lemma 2 the on-line conditions for $F(\mathbf{X}_i)$ can be derived. The following theorem states these conditions. Let $|\mathbf{x}|$ denote a vector for which the elements have absolute values, i.e. $|\mathbf{x}| = (|x^1|, |x^2|, \dots, |x^N|)$, and $\mathbf{x} \cdot \mathbf{y}$ denote the inner product of the vectors $\mathbf{x}$ and $\mathbf{y}$, then,

THEOREM 1. *Let $F(\mathbf{X})$ be continuous on $[\mathbf{X}_A, \mathbf{X}_B]$ and differentiable on $(\mathbf{X}_A, \mathbf{X}_B)$, then $F(\mathbf{X})$ can be generated with an on-line delay of $\delta_g \geq \max(\delta)$ if*

$$\Gamma(\mathbf{X}_K) \leq r^{\varphi+1} \left\{ 1 - \frac{(r-1)}{2\eta} \right\}, \quad (14)$$

where

$$\Gamma(\mathbf{X}_K) = \max_{\mathbf{X}_K} \{ |\nabla F(\mathbf{X}_K)| \cdot \mathbf{d} \} \quad \text{with } \mathbf{X}_K \in [\mathbf{X}_A, \mathbf{X}_B]$$

and

$$\mathbf{d} = (r^{-\delta_1}, r^{-\delta_2}, \dots, r^{-\delta_N}).$$

Proof. The error caused by rounding Z_i symmetrically to the i th digit is given by

$$|\bar{Z}_i - Z_i| \leq (r/2) \cdot r^{-(i-\varphi)}. \quad (15)$$

Since we need an expression for $|\bar{Z}_i - \bar{Z}_{i-1}|$, we rewrite $|\bar{Z}_i - \bar{Z}_{i-1}|$ as

$$\begin{aligned} |\bar{Z}_i - \bar{Z}_{i-1}| &= |(\bar{Z}_i - Z_i) - (\bar{Z}_{i-1} - Z_{i-1}) \\ &\quad + (Z_i - Z_{i-1})|. \end{aligned} \quad (16)$$

From Eq. (15) and Eq. (16) it follows that

$$\begin{aligned} |\bar{Z}_i - \bar{Z}_{i-1}| &\leq |\bar{Z}_i - Z_i| + |\bar{Z}_{i-1} - Z_{i-1}| \\ &\quad + |Z_i - Z_{i-1}| \leq (r/2) \cdot r^{-(i-\varphi)} \\ &\quad + (r/2) \cdot r^{-(i-\varphi-1)} + |Z_i - Z_{i-1}|. \end{aligned} \quad (17)$$

The error term $(r/2) \cdot r^{-(i-\varphi)} + (r/2) \cdot r^{-(i-\varphi-1)}$ of Eq. (17) can only be reached if the maximal rounding error occurs when rounding both Z_i and Z_{i-1} and it holds that $\bar{Z}_i - Z_i$ and $\bar{Z}_{i-1} - Z_{i-1}$ have opposite signs. A maximum rounding error occurs when a "1" ($Z_i > 0$) or a "-1" ($Z_i < 0$) is to be added to the i th digit position (weight $r^{-(i-\varphi-1)}$). Therefore, it holds $|\bar{Z}_i| \geq r^{-(i-\varphi-1)}$ and by the same reasoning $|\bar{Z}_{i-1}| \geq r^{-(i-\varphi-2)}$. Since $\bar{Z}_i - Z_i$ and $\bar{Z}_{i-1} - Z_{i-1}$ have opposite signs, also $\bar{Z}_i$ and $\bar{Z}_{i-1}$ have opposite signs, if maximum rounding takes place. It follows that $|\bar{Z}_i - \bar{Z}_{i-1}| = |\bar{Z}_i| + |\bar{Z}_{i-1}| \geq r^{-(i-\varphi-1)} + r^{-(i-\varphi-2)}$. However, this implies that the on-line condition of Eq. (10) cannot be satisfied in this case. Thus, if the on-line condition is satisfied the maximum rounding error of Eq. (17) does not occur, meaning that we can replace the $\leq$ sign by the $<$ sign, yielding

$$\begin{aligned} |\bar{Z}_i - \bar{Z}_{i-1}| &< (r/2) \cdot r^{-(i-\varphi)} + (r/2) \cdot r^{-(i-\varphi-1)} \\ &\quad + |Z_i - Z_{i-1}|. \end{aligned} \quad (18)$$

By applying the mean value theorem to the last term of Eq. (18) it follows that

$$\begin{aligned} Z_i - Z_{i-1} &= F(\mathbf{X}_i) - F(\mathbf{X}_{i-1}) \\ &= (\mathbf{X}_i - \mathbf{X}_{i-1}) \cdot \nabla F(\mathbf{X}_K) \end{aligned} \quad (19)$$

with $\mathbf{X}_K \in [\mathbf{X}_{i-1}, \mathbf{X}_i]$. Furthermore, with $|\mathbf{x}_i| \leq \eta \cdot \mathbf{I}$, $\mathbf{I}$ being the unit vector, it holds

$$|\mathbf{X}_i - \mathbf{X}_{i-1}| = |\mathbf{x}_i \odot \mathbf{d} \cdot \mathbf{r}^{-i}| \leq \eta \cdot r^{-i} |\mathbf{d}| \quad (20)$$

with

$$\mathbf{d} = (r^{-\delta_1}, r^{-\delta_2}, \dots, r^{-\delta_N}),$$

where $\mathbf{y} = \mathbf{x} \odot \mathbf{z}$ denotes the pairwise product of the vector elements; i.e., $y_j = x_j \cdot z_j$, $1 \leq j \leq N$. Substitution of Eqs. (19) and (20) in Eq. (18) results in

$$|\bar{Z}_i - \bar{Z}_{i-1}| < (r/2) \cdot r^{-(i-\varphi)} + (r/2) \cdot r^{-(i-\varphi-1)} + \eta \cdot r^{-i} \cdot \Gamma(\mathbf{X}_K) \quad (21)$$

with

$$\Gamma(\mathbf{X}_K) = \max_{\mathbf{X}_K} \{ |\nabla F(\mathbf{X}_K)| \cdot \mathbf{d} \}$$

or

$$|\bar{Z}_i - \bar{Z}_{i-1}| < r^{-(i-\varphi-2)} \{ (1/2)r + (1/2) + \eta \cdot r^{-\varphi-2} \cdot \Gamma(\mathbf{X}_K) \}. \quad (22)$$

Therefore, from Eq. (10) and Eq. (22) it follows that

$$\Gamma(\mathbf{X}_K) \leq r^{\varphi+1} \left[1 - \frac{(r-1)}{2\eta} \right]. \quad (23)$$

Since $\mathbf{X}_K \in [\mathbf{X}_{i-1}, \mathbf{X}_i]$ must be valid for every iteration from 1 to n , $\mathbf{X}_K$ must cover the complete range of the input numbers, denoted by $\mathbf{X}_K \in [\mathbf{X}_A, \mathbf{X}_B]$. Q.E.D.

For fully redundant number representations Eq. (14) reduces to

$$\Gamma(\mathbf{X}_K) \leq (1/2) \cdot r^{\varphi+1}. \quad (24)$$

The last question is the tightness of the bound. From Eq. (21) it follows that the errors produced by the symmetrical rounding procedure determine the tightness of the bounds. Since for most arithmetic functions rounding errors near the maximum rounding error will occur, the bound is likely to be very good.

2.2. Comparison with Other Models

An important difference between the E-model and the R- and M-models is that the latter use function approximations instead of the exact function value in each iteration step. The use of the function approximation $\hat{Z}_i = Z_i + \epsilon_i$, where ϵ_i denotes the error of the approximation in the i th iteration, leads to an extra error term in Eq. (18) of $|\epsilon_i - \epsilon_{i-1}|$. An extra error term implies an extra restriction on the on-line properties of that function. Also, in actual R- or M-algorithms additional restrictions can influence the on-line behavior, such as the number of digits used to determine the selection of the next on-line digit. Therefore, in general, the E-model produces stronger bounds on the attainable on-line delays than any algorithm in the R- or M-model class.

2.3. Generation Delays of Some Basic Arithmetic Functions

The results from Theorem 1 will be illustrated by some standard arithmetic functions. These functions will be con-

sidered for fixed- and floating-point numbers. For fixed-point numbers the input numbers are defined on the interval $(-1, 1)$. For floating-point numbers the interval $\pm[r^{-q}, 1]$ is considered for the mantissa of the input operands. The weight of the most significant digit of the result (r^q) will depend on q , i.e., $\varphi = f(q)$, where f is determined by the arithmetic function under consideration. In the following examples a fully redundant digital set is assumed.

2.3.1. Addition: $X + Y$. For addition it holds $|\nabla F(X + Y)| = (1, 1)$. Therefore, Eq. (24) becomes for addition

$$\Gamma(\mathbf{X}_K) = 2 \cdot r^{-\delta} \leq (1/2) \cdot r^{\varphi+1}. \quad (25)$$

It can easily be verified that $\varphi = 1$ for $r = 2$ (due to the rounding) and $\varphi = 0$ for $r > 2$. Thus, $\delta_K = 0, \forall r$. For $r = 3$ this result does not directly follow from Eq. (25). However, the rounding function is in this case not a single digit, but a fraction; i.e., $r/2 = \frac{3}{2} = (1.11\ldots)_3$. Therefore, rounding up (or down) only occurs if the last digit of $X + Y$ is 2 (or -2). This implies that the rounding error is $r^{-(i-\varphi)}$ instead of $(r/2) \cdot r^{-(i-\varphi)}$. It then follows with Eq. (27) that $\delta_K = 0$ for $r = 3$. If no overflow in the result of fixed-point addition is allowed (i.e., $|X + Y| < 1$), it holds $\varphi = 0$ for $r = 2$ (see Section 2.4.1).

2.3.2. Multiplication: $X \cdot Y$. For multiplication it holds $|\nabla F(X \cdot Y)| = (|X|, |Y|)$. Since $|X| = |Y| < 1$ and $\varphi = 0$ (due to the rounding), we get from Eq. (24)

$$\Gamma(X_{\max}, Y_{\max}) = \{2 \cdot r^{-\delta} - 2 \cdot r^{-n}\} \leq (1/2) \cdot r \quad (26)$$

with $X_{\max} = Y_{\max} = 1 - r^{-n}$. Thus, $\delta_g = 1$ for $r = 2, 3$ and $\delta_g = 0$ for $r \geq 4$. In Fig. 3 an example of the multiplication operation is shown for $r = 2$ and $r = 4$.

2.3.3. Reciprocal: $1/X$. For the reciprocal function we get $|\nabla F(1/X)| = |X^{-2}|$. The weight of the m.s.d. is r^q (i.e., $\varphi = q$). The maximum value of this function is reached by taking $X = r^{-q}$, giving with Eq. (24)

$$\Gamma(r^{-q}) = r^{-\delta+2q} \leq (1/2) \cdot r^{q+1} \quad \text{or} \quad r^{-\delta+q-1} \leq 1/2. \quad (27)$$

From Eq. (27) it follows that $\delta = q$ for $r \geq 2$. The result $\delta = q$ for $r \geq 2$ implies that we now have an even better bound for $r = 2$ and $r = 3$ than derived in [15].

2.3.4. Division: X/Y . The division operation gives $|\nabla F(X/Y)| = (|1/Y|, |X \cdot Y^{-2}|)$. The weight of the m.s.d. is r^q ($\varphi = q$, because of the rounding). The value of $\Gamma(\mathbf{X}_K)$ can be found by taking $\mathbf{X}_K = (X, Y) = (1 - r^{-n}, r^{-q})$, yielding for Eq. (24)

$$\Gamma(1 - r^{-n}, r^{-q}) = r^{-\delta_x} + (1 - r^{-n}) \times r^q \cdot r^{-\delta_y} \leq (1/2) \cdot r. \quad (28)$$

From Eq. (28) the following minimum delays are derived:

iteration i	X_i, Y_i	X_i, Y_i	Round	$\bar{Z}_i$	$\bar{z}_i, r^{(i-1)}$
1	$X_1 = 0.11$ $Y_1 = 0.11$	0.1001 (0.01)	+ 0.1	$\bar{Z}_1 = 1.$ (0.)	$\bar{Z}_1 - \bar{Z}_0 = 1.r^0$ (0)
2	$X_2 = 0.11\bar{1}$ $Y_2 = 0.111$	0.100011 (0.1001)	+ 0.01	$\bar{Z}_2 = 0.1$ (0.1)	$\bar{Z}_2 - \bar{Z}_1 = 1.r^1$ (1)
3	$X_3 = 0.11\bar{1}0$ $Y_3 = 0.1110$	0.10001100 (0.100011)	+ 0.001	$\bar{Z}_3 = 0.10$ (0.10)	$\bar{Z}_3 - \bar{Z}_2 = 0.r^2$ (0)
4	$X_4 = 0.11\bar{1}01$ $Y_4 = 0.11101$	0.1001100001 (0.10001100)	+ 0.0001	$\bar{Z}_4 = 0.101$ (0.100)	$\bar{Z}_4 - \bar{Z}_3 = 1.r^3$ (0)
5	$X_5 = 0.11\bar{1}01$ $Y_5 = 0.11101$	0.1001100001 (0.1001100001)	+ 0.00001	$\bar{Z}_5 = 0.1010$ (0.1010)	$\bar{Z}_5 - \bar{Z}_4 = 0.r^4$ (10.)
a. on-line multiplication with $r=2, \delta_x=1, \varphi=0, X=0.11\bar{1}01, Y=0.11101$ (the case $\delta_x=0$ is shown between brackets)					
1	$X_1 = 0.3$ $Y_1 = 0.3$	0.21	+ 0.2	$\bar{Z}_1 = 1.$	$\bar{Z}_1 - \bar{Z}_0 = 1.r^0$
2	$X_2 = 0.33$ $Y_2 = 0.33$	0.3201	+ 0.02	$\bar{Z}_2 = 1.0$	$\bar{Z}_2 - \bar{Z}_1 = 0.r^1$
3	$X_3 = 0.33\bar{3}$ $Y_3 = 0.333$	0.320013	+ 0.002	$\bar{Z}_3 = 0.32$	$\bar{Z}_3 - \bar{Z}_2 = 2.r^2$
4	$X_4 = 0.3330$ $Y_4 = 0.3330$	0.32001300	+ 0.0002	$\bar{Z}_4 = 0.320$	$\bar{Z}_4 - \bar{Z}_3 = 0.r^3$
5	$X_5 = 0.33303$ $Y_5 = 0.33303$	0.3201310201	+ 0.00002	$\bar{Z}_5 = 0.3202$	$\bar{Z}_5 - \bar{Z}_4 = 2.r^4$
b. on-line multiplication with $r=4, \delta_x=0, \varphi=0, X=33303, Y=33303$					

FIG. 3. Example of on-line multiplication $X \cdot Y$. In case (a) it is shown that $r = 2$ will go wrong with $\delta_x = 0$ and $\varphi = 0$ in the fifth iteration.

$\delta_x = 1, \delta_y = q + 1$, for $r = 2, \delta_x = 1, \delta_y = q$, for $r = 3$, and $\delta_x = 0, \delta_y = q$, for $r \geq 4$.

2.3.5. *Square root: $\sqrt{X}$* . The square root function gives $|\nabla F(\sqrt{X})| = (1/2) \cdot (\sqrt{X})^{-1}$. The weight of the m.s.d. is r^0 ($\varphi = 0$). Taking $X_K = r^{-q}$ yields

$$\Gamma(r^{-q}) = (1/2) \cdot r^{-\delta} \cdot r^{(1/2)q} \leq (1/2) \cdot r. \quad (29)$$

Therefore, $\delta = \lceil q/2 \rceil - 1, \forall r$.

2.4. Alignment Delay

The generation on-line delay only specifies the minimum number of digits of the input operand required to produce the first significant digit of the result. However, the output digits produced may not be in a form required by the next arithmetic unit. If compound functions are calculated, such as the inverse square root of a sum of squares as described in [14], alignment to a standard format is not necessary and there is no alignment delay between the successive calculations. If the arithmetic units are to be coupled independently, a standard output format is required.

For a fixed-point result, we require the weight of the m.s.d. to be r^{-1} . Due to rounding, for some functions (e.g., multiplication and square root) the weight of the first digit of the generated result can be r^0 . The result must then be converted. This can be done by applying the following pro-

cedure: let $\bar{z}_i$ be a digit of the generated result and $\bar{z}_i^*$ be a digit of the aligned result. Then

procedure: fix_align

end := 'true';

if $\bar{z}_1 \neq 0$ then end := 'false' fi;

do $i = 1, n$

if end := 'false' then

if $\bar{z}_{i+1} = 0$ then $\bar{z}_i^* := \text{sign}(\bar{z}_i) \cdot (r - 1)$

else $\bar{z}_i^* := \text{sign}(\bar{z}_i) \cdot (r - |\bar{z}_{i+1}|)$;

end := 'true' fi;

else $\bar{z}_i^* := \bar{z}_{i+1}$ fi; od;

For floating-point arithmetic we require the result of the operation to be quasi-normalized. The following procedure produces quasi-normalized results.

procedure: float_align

exp := e; quasi := 'false';

do $i = 1, n$

if quasi = 'false' then

then

if $\bar{z}_i = 0$

then exp := exp - 1

else if $(|\bar{z}_i| = 1 \text{ and } \text{sign}(\bar{z}_i) \neq \text{sign}(\bar{z}_{i+1}))$

then $\bar{z}_{i+1} :=$

$\text{sign}(\bar{z}_i) * (r - |\bar{z}_{i+1}|)$;

exp := exp - 1;

else da := i - 1;

quasi := 'true';

$\bar{z}_1^* := \bar{z}_i$ fi; fi;

else $\bar{z}_{i+da}^* := \bar{z}_i$ fi; od;

Because the quasi-normalization procedure always inspects two digits, the minimal value of $\delta_a = 1$. The following values of δ_a can be derived:

2.4.1. *Addition/Subtraction*. For fixed-point numbers the weight of the m.s.d. is r^{-1} , if no overflow is allowed. Because $\varphi = 0, \forall r$ (also for $r = 2$, see Section 2.3.1), it holds for the alignment delay $\delta_a = 1, \forall r \geq 2$.

For floating-point numbers we must discriminate between addition and subtraction. First addition is considered. If the input operands are in the range $\pm[r^{-q}, 1)$, the result is in the range $\pm[2r^{-q}, 2)$. Since for $r = 2$ the m.s.d. of the result has a weight of r^{-1} ($\varphi = 1$), the exponent must be increased by 2, transforming the result into the range $\pm[2^{-(q+1)}, 2^{-1})$. This implies that 0 to $q - 1$ leading zeros must be suppressed by applying the quasi-normalization procedure, i.e., $0 \leq \delta_a \leq q - 1$.

For $r \geq 3$, the result is transformed into the range $\pm[2r^{-(q+1)}, 2r^{-1})$ through increasing the exponent by 1. If $|\bar{z}_1| = 2$ (it holds $|\bar{z}_1| \leq 2$), the result is already quasi-normalized, yielding $\delta_a = 0$. If $|\bar{z}_1| \leq 1$, the quasi-normalization procedure is applied, giving $1 \leq \delta_a \leq q - 1$. Therefore, it holds $\delta_a = 0$ for $q = 1, r = 2, 1 \leq \delta_a \leq q - 1$ for $q \geq 2, r = 2$, and $0 \leq \delta_a \leq q - 1$, for $q \geq 1, r \geq 3$.

In the case of subtraction an arbitrary number of leading zeros may be generated. Therefore, it holds for subtraction $1 \leq \delta_a \leq n-1$.

2.4.2. Multiplication. For multiplication the weight of the m.s.d. is r^0 . For fixed-point operands the first digit must have a weight of r^{-1} . Therefore, the procedure *fix_align* must be applied. If $|\dot{z}_1| = 1$, then the first nonzero digit $\dot{z}_i$ will have an opposite sign to $\dot{z}_1$, because the absolute value of the result is < 1 , resulting in $\delta_a = 1$.

For floating-point operands the result is in the range of $[r^{-2q}, 1)$, if the input operands are in the range $[r^{-q}, 1)$. Therefore, $\delta_a = 0$, for $q = 1$, and $1 \leq \delta_a \leq 2(q-1)$, for $q \geq 2$.

2.4.3. Reciprocal and Division. For the reciprocal function, only floating-point input operands are considered. If the input operand is in the range $[r^{-q}, 1)$, the result is in the range $(1, r^q]$. Through scaling by r^{-q-1} , this output range becomes $(r^{-q-1}, r^{-1}]$. Thus, $\delta_a = 0$, for $q = 1$, and $1 \leq \delta_a \leq q-1$, for $q \geq 2$.

For division, only floating-point operations are considered. If the input operands are in the range $[r^{-q}, 1)$, the result is in the range (r^{-q}, r^q) . Scaling by r^{-q} yields $(r^{-2q}, 1)$. Therefore, $\delta_a = 0$, for $q = 1$, and $1 \leq \delta_a \leq 2(q-1)$, for $q \geq 2$.

2.4.4. Square Root. For a fixed-point result, we require the weight of the m.s.d. to be r^{-1} . Since the weight of the generated results is r^0 , the procedure *fix_align* must here also be used, giving $\delta_a = 1$.

For a floating-point input operand in the range $[r^{-q}, 1)$, the result of the square root operation has a range of $[r^{-(1/2)q}, 1)$. Therefore, the alignment delays are $\delta_a = 0$, for $1 \leq q \leq 4$, and $1 \leq \delta_a \leq \lceil q/2 \rceil - 2$, for $q > 4$.

2.5. Summarized Results

The total on-line delay of an arithmetic function is given by $\delta_F = \delta_g + \delta_a$. In essence, three cases can be considered: compound functions, where no alignment is necessary ($\delta_a = 0$); fixed-point output format; and floating-point output format. The results in these three cases for the five arithmetic functions treated in the previous sections are summarized in Table I for $r = 2, 4$, and 8 .

Also a comparison can be made between some of the results in Section 2.2 and the results presented by Owens [9] for multiplication, division, and square root. A difficulty is that all results are dependent on the range of the input operands. The generation delays given in this paper can be calculated for any range, but this does not hold for the results in [9]. Therefore, we have to adapt our results to the range restrictions used in [9]. For multiplication Owens gives a conditional delay $\delta \geq 0$ if $X, Y \leq 1$. The results in Section 2.3.2 also hold for $X, Y \leq 1$. This shows that the conditional and attainable on-line delays are equal, except for $r = 2$ and $r = 3$. For division the conditional delay is $\delta \geq q$, if $X \leq 1$

TABLE I
The Total On-Line Delay $\delta_F = \delta_g + \delta_a$ for a
Number of Standard Operations

		Compound ($\delta_a = 0$)	Fixed point	Floating point	
				$q = 1$	$q = 2$
Addition	$r = 2$	0	1	0	1
	$r = 4, 8$	0	1	0	(0, 1)
Multiplication	$r = 2$	1	2	1	(2, 3)
	$r = 4, 8$	0	1	0	(1, 2)
Reciprocal	$r = 2, 4, 8$	q	—	1	3
Division	$r = 2$	$q + 1$	—	2	(4, 5)
	$r = 4, 8$	q	—	1	(3, 4)
Square root	$r = 2, 4, 8$	$\lceil q/2 \rceil - 1$	$\lceil q/2 \rceil$	0	0

and $Y \geq r^{-q}$, which is equal to the attainable delay, except for $r = 2$. Square root yields $\delta \geq q/2$ if $X \geq r^{-q}$ with the restriction that q must be even. The attainable delay from Section 2.3.5 states $\delta = \lceil q/2 \rceil - 1$, so for q is even the bound of the conditional on-line delay is not tight enough. More thorough comparisons between conditional and attainable on-line delays for arbitrary arithmetic functions can only be made if a general method or theory is available for deriving tight bounds on the conditional on-line delays.

3. TABLE IMPLEMENTATION OF THE ON-LINE MODEL

To implement the E-model the exact function value must be known in each iteration step. However, in reality we need only a restricted number of digits of Z_i for the rounding and truncation to $\bar{Z}_i$. Most conventional arithmetic units could provide the precision needed. However, the result must be exact up to the digits needed and may not be rounded in advance.

In this section a different way of generating the on-line digits of the E-model is shown. The method is based on the use of tables and it is shown that the digit generation process is very simple. In Section 4 an example of this procedure is shown in the use of starting of an M-algorithm.

The table look-up method works as follows: The argument X_i is used as the address to a read-only memory, where the appropriate function values are stored. It is shown that the memory need only output the digits $\bar{z}_i$ and $\bar{z}_{i-1}$ of $\bar{Z}_i$ in order to generate the on-line digit $\dot{z}_i$.

It is to note that $\bar{Z}_i$ need not be in redundant form, since it is to be calculated in advance and be used in the table. For $\bar{Z}_i$ a conventional sign-magnitude format representation is chosen. From Eq. (10) it follows that $|\bar{Z}_i - \bar{Z}_{i-1}| < r^{-(i-q-2)}$, if the on-line condition is to be satisfied. In the following lemma a bound on the difference between $\bar{Z}_i$ and $\bar{Z}_{i-1}$ is established:

LEMMA 3. Given $\bar{Z}_i$ and $\bar{Z}_{i-1}$ be in nonredundant sign-magnitude format. If $|\bar{Z}_i - \bar{Z}_{i-1}| < r^{-(i-\varphi-2)}$, then $(\bar{Z}_{i-1} - \bar{Z}_{i-1})$ equals either $\bar{z}_i \cdot r^{-(i-\varphi-1)}$ or $[-\text{sign}(\bar{Z}_i) \cdot r + \bar{z}_i] \cdot r^{-(i-\varphi-1)}$ and $\bar{z}_i \neq 0$ in the latter case.

Proof. Given $|\bar{Z}_i - \bar{Z}_{i-1}| < r^{-(i-\varphi-2)}$, it follows that

$$-\bar{z}_i \cdot r^{-(i-\varphi-1)} - r^{-(i-\varphi-2)} < \bar{Z}_i - \bar{Z}_{i-1} - \bar{z}_i \cdot r^{-(i-\varphi-1)} < r^{-(i-\varphi-2)} - \bar{z}_i \cdot r^{-(i-\varphi-1)} \quad (30)$$

The nonredundant sign-magnitude format of $\bar{Z}_i$ can be interpreted as each digit having the same sign, i.e., $\text{sign}(\bar{z}_i) = \text{sign}(\bar{Z}_i)$. With $|\bar{z}_i| \leq (r-1)$, the above formula can be transformed to

$$\begin{aligned} -2r^{-(i-\varphi-2)} + r^{-(i-\varphi-1)} &< \bar{Z}_i - \bar{Z}_{i-1} - \bar{z}_i \cdot r^{-(i-\varphi-1)} \\ &< r^{-(i-\varphi-2)} \quad \text{if } \bar{Z}_i \geq 0; \\ -r^{-(i-\varphi-2)} &< \bar{Z}_i - \bar{Z}_{i-1} - \bar{z}_i \cdot r^{-(i-\varphi-1)} \\ &< 2r^{-(i-\varphi-2)} - r^{-(i-\varphi-1)} \quad \text{if } \bar{Z}_i < 0. \end{aligned} \quad (31)$$

Obviously, $\bar{Z}_i - \bar{Z}_{i-1} - \bar{z}_i \cdot r^{-(i-\varphi-1)}$ must be an integral multiple of $r^{-(i-\varphi-2)}$; thus Eq. (31) is equivalent to

$$\begin{aligned} \bar{Z}_i - \bar{Z}_{i-1} - \bar{z}_i \cdot r^{-(i-\varphi-1)} &= \begin{cases} -r^{-(i-\varphi-2)} & \text{if } \bar{Z}_i \geq 0 \\ 0 & \end{cases} \\ \bar{Z}_i - \bar{Z}_{i-1} - \bar{z}_i \cdot r^{-(i-\varphi-1)} &= \begin{cases} r^{-(i-\varphi-2)} & \text{if } \bar{Z}_i < 0 \\ 0 & \end{cases} \end{aligned} \quad (32)$$

Therefore, from Eq. (32) it follows with $|\bar{Z}_i - \bar{Z}_{i-1}| < r^{-(i-\varphi-2)}$,

$$\bar{Z}_i - \bar{Z}_{i-1} = \begin{cases} \bar{z}_i \cdot r^{-(i-\varphi-1)} \\ [-\text{sign}(\bar{Z}_i) \cdot r + \bar{z}_i] \cdot r^{-(i-\varphi-1)} \end{cases} \quad \text{and } \bar{z}_i \neq 0. \quad (33)$$

Q.E.D.

Because $\text{sign}(\bar{z}_i) = \text{sign}(\bar{Z}_i)$ and $\bar{z}_i \neq 0$, it holds $[-\text{sign}(\bar{Z}_i) \cdot r + \bar{z}_i] \leq (r-1)$. Therefore, Lemma 3 shows that $\bar{Z}_i - \bar{Z}_{i-1}$ can be represented by one single digit $\dot{z}_i$ with a weight of $r^{-(i-\varphi-1)}$. This means that the function can be approximated by using $\dot{z}_i = (\bar{Z}_i - \bar{Z}_{i-1}) \cdot r^{-(i-\varphi-1)}$ as its i th digit.

Each iteration step a digit $\dot{z}_i$ can be generated from z_i , according to Eq. (33). The digit $\bar{z}_i$ can be generated from a table with X_i as argument. However, the knowledge of $\bar{z}_i$ is insufficient to generate $\dot{z}_i$, because it is not known which of the two equations of Eq. (33) must be chosen. The following theorem states how this choice can be made:

THEOREM 2. Let $|\bar{Z}_i - \bar{Z}_{i-1}| < r^{-(i-\varphi-2)}$, $\dot{z}_1 = \bar{Z}_1(1)$, and $\dot{z}_i = (\bar{Z}_i - \bar{Z}_{i-1}) \cdot r^{-(i-\varphi-1)}$ for $i \geq 2$ be the (i) th digit of the on-line result, then it holds

$$\dot{z}_i = \begin{cases} \bar{z}_i & \text{if } \bar{z}_{i-1} = \bar{z}'_{i-1} \\ \text{com}(\bar{z}_i) & \text{if } \bar{z}_{i-1} \neq \bar{z}'_{i-1} \end{cases} \quad \text{for } i \geq 2, \quad (34)$$

where $\bar{z}_i$ and $\bar{z}_{i-1}$ are the (i) th and $(i-1)$ th digit of $\bar{Z}_i$, respectively. $\bar{z}'_{i-1}$ is the $(i-1)$ th digit of $\bar{Z}_{i-1}$, and $\text{com}(\bar{z}_i)$ is the radix- r complement of $\bar{z}_i$; i.e., $\text{com}(\bar{z}_i) = -\text{sign}(\bar{Z}_i) \cdot r + \bar{z}_i$.

Proof. Lemma 3 states that $\bar{Z}_i - \bar{Z}_{i-1}$ is equal to either $\bar{z}_i \cdot r^{-(i-\varphi-1)}$ or $[-\text{sign}(\bar{Z}_i) \cdot r + \bar{z}_i] \cdot r^{-(i-\varphi-1)}$. Obviously, $\bar{Z}_i - \bar{Z}_{i-1} = \bar{z}_i \cdot r^{-(i-\varphi-1)}$ means that $\bar{Z}_i - \bar{Z}_{i-1} - \bar{z}_i \cdot r^{-(i-\varphi-1)} = 0$. The following cases can be considered:

1. $\text{sign}(\bar{Z}_i) = \text{sign}(\bar{Z}_{i-1})$
2. $\text{sign}(\bar{Z}_i) \neq \text{sign}(\bar{Z}_{i-1})$ and $|\bar{Z}_{i-1}| > 0$
3. $\text{sign}(\bar{Z}_i) \neq \text{sign}(\bar{Z}_{i-1})$ and $\bar{Z}_{i-1} = 0$

(sub 1) $\text{sign}(\bar{Z}_i) = \text{sign}(\bar{Z}_{i-1})$ implies that $\bar{z}_j = \bar{z}'_j$ for all j ($1 \leq j \leq i-1$), where $\bar{z}_j$ denotes the j th digit of $\bar{Z}_i$ and $\bar{z}'_j$ denotes the j th digit of $\bar{Z}_{i-1}$.

(sub 2) $\text{sign}(\bar{Z}_i) \neq \text{sign}(\bar{Z}_{i-1})$ and $|\bar{Z}_{i-1}| > 0$ implies that $|\bar{Z}_i - \bar{Z}_{i-1}| \geq r^{-(i-\varphi-2)}$, which cannot occur because of the condition $|\bar{Z}_i - \bar{Z}_{i-1}| < r^{-(i-\varphi-2)}$.

(sub 3) $\text{sign}(\bar{Z}_i) \neq \text{sign}(\bar{Z}_{i-1})$ and $\bar{Z}_{i-1} = 0$ implies $\bar{z}_j = \bar{z}'_j = 0$ for all j ($1 \leq j \leq i-1$).

Therefore, in this case all digits are equal for $1 \leq j \leq i-1$. If the second case of Eq. (34) can be proved, a test on the $(i-1)$ th digit is sufficient.

Now, consider the case $\bar{Z}_i - \bar{Z}_{i-1} = [-\text{sign}(\bar{Z}_i) \cdot r + \bar{z}_i] \cdot r^{-(i-\varphi-1)}$. This means that $\bar{Z}_i - \bar{Z}_{i-1} - \bar{z}_i \cdot r^{-(i-\varphi-1)} = -\text{sign}(\bar{Z}_i) \cdot r^{-(i-\varphi-2)}$ (see Eq. (33)). It is shown that this always implies $\bar{z}_{i-1} \neq \bar{z}'_{i-1}$. Let us assume that $\bar{z}_{i-1} = \bar{z}'_{i-1}$, then it follows that $\bar{Z}_i - \bar{Z}_{i-1} - \bar{z}_i \cdot r^{-(i-\varphi-1)}$ is an integral multiple of $r^{-(i-\varphi-3)}$. Since $-\text{sign}(\bar{Z}_i) \cdot r^{-(i-\varphi-2)}$ is not an integral multiple of $r^{-(i-\varphi-3)}$, the assumption leads to a contradiction with the given condition, meaning that $\bar{z}_{i-1} \neq \bar{z}'_{i-1}$. So, we conclude that the two cases of the value $\bar{Z}_i - \bar{Z}_{i-1}$ can be distinguished by observing the digits $\bar{z}_{i-1}$ and $\bar{z}'_{i-1}$, i.e.,

$$\bar{Z}_i - \bar{Z}_{i-1} = \begin{cases} \bar{z}_i \cdot r^{-(i-\varphi-1)} & \text{if } \bar{z}_{i-1} = \bar{z}'_{i-1} \\ [-\text{sign}(\bar{Z}_i) \cdot r + \bar{z}_i] \cdot r^{-(i-\varphi-1)} & \text{if } \bar{z}_{i-1} \neq \bar{z}'_{i-1} \end{cases}$$

By definition, it holds $\dot{z}_i = (\bar{Z}_i - \bar{Z}_{i-1}) \cdot r^{-(i-\varphi-1)}$ and $\text{com}(\bar{z}_i) = -\text{sign}(\bar{Z}_i) \cdot r + \bar{z}_i$. Therefore, the i th digit $\dot{z}_i$ equals

$$\dot{z}_i = \begin{cases} \bar{z}_i & \text{if } \bar{z}_{i-1} = \bar{z}'_{i-1} \\ \text{com}(\bar{z}_i) & \text{if } \bar{z}_{i-1} \neq \bar{z}'_{i-1} \end{cases} \text{ for } i \geq 2.$$

Q.E.D.

Theorem 2 shows that the implementation of the on-line table look-up method is simple. To generate an on-line function approximation of m digits with an error less than $(r/2) \cdot r^{-(m-\varphi)}$ requires a FUNCTION table having $(m \cdot N + \delta_1 + \delta_2 + \dots + \delta_N)$ digits as its input address. If one of the input operands has a significant leading digit, this digit can be used to indicate the first position of the group of digits. If no operand has a significant leading digit in a fixed position, one extra bit in the address is required to indicate the first position of the group of digits. The FUNCTION table returns the value of two digits $\bar{z}_{i-1}$ and $\bar{z}_i$. The i th online digit $\dot{z}_i$ is determined by these two digits and the digit $\bar{z}'_{i-1}$ (kept in a register) of the previous interaction. The size of the FUNCTION table can be reduced if the redundant numbers X_i are being transformed into a nonredundant sign-magnitude number representation X_i^* , and using this number as the input address. The transformation can be done by using a radix- r digit count-down counter. The counter can be implemented as a radix- r counter which can perform count-down/append operations at different digit positions. Another elegant method for performing the transformation is on-the-fly conversion, proposed in [12]. Here, no borrow propagation is necessary. The realization of an on-line table look-up unit for a single input operand is shown in Fig. 4.

The table method of implementing an on-line arithmetic function can only be used if the table size can be kept reasonably small. Besides any practical use, the table method shows that the on-line delays derived in the previous sections can in fact be realized.

4. ON-LINE RECIPROCAL APPROXIMATION WITH AN ADAPTED NEWTON-RAPHSON METHOD

In this section the E-method is combined with an M-method to provide an algorithm for the reciprocal approxi-

FIG. 4. Block diagram of an on-line table look-up system.

mation. A number of on-line algorithms for division have been developed. The division methods can in essence be divided into two classes: the use of nonrestoring division methods [1, 13] or the use of continued sum/product methods [3-5, 9, 11]. Both approaches have their advantages and limitations. Trivedi and Ercegovac's algorithm [1] has an on-line delay of $\delta = 4$ (for fractions), but accepts only non-fully redundant digit sets, thus excluding radix-2. In [13] an algorithm for radix-4 division (floating-point) has been described with an on-line delay of $\delta = 6$ to 8, but with a simple digit selection mechanism. A radix-2 division algorithm with $\delta = 3$ and a simple selection mechanism has been described by Ercegovac and Lang [14]. This algorithm is part of a compound function and a relative low on-line delay can be achieved because the result range is known in advance. The division algorithms of Owens [4, 5] are based on the CSP approach and have on-line delays of $\delta = 3$ to $\delta = 5$.

In the previous sections, the on-line delay for reciprocal approximation has been derived. If implemented by a function table, the size of the table grows exponentially with the word length, e.g., to obtain an accuracy of 2^{-24} , a memory of size $O(16M \lceil \log r \rceil)$ bit is required for the reciprocal table. This is impractical. In this section, an adapted Newton-Raphson method is used in addition to the E-method. In the initialization phase, the on-line table look-up method is applied to generate the first m digits; then the adapted Newton-Raphson method is applied to produce the remaining $n - m$ digits (n = number of digits of the mantissa of a floating-point number). The Newton-Raphson equation for the reciprocal approximation can be written as

$$\begin{aligned} \dot{Z}_i &= \dot{Z}_{i-1}(2 - \dot{Z}_{i-1} \cdot X) \\ \text{or } R_i &= \dot{Z}_{i-1}(1 - \dot{Z}_{i-1} \cdot X), \end{aligned} \quad (35)$$

where $\dot{Z}_i$ is the i th approximation to the reciprocal ($1/X$). It will be shown that using a converted (quadratic) method reduces the on-line delay and simplifies the digit selection unit, as compared with the usual linear iteration methods.

Now $\dot{Z}_{i-1}$ is usually the result of the previous iteration according to Eq. (35). This choice has the disadvantage that at each iteration step a complete evaluation of the right-hand expression must be made. Instead of that we substitute $\dot{Z}_{i-1}$ for $\dot{Z}_{i-1}$. Then incremental multiplication techniques can be applied. This substitution transforms Eq. (35) to

$$\begin{aligned} \dot{Z}_i &= \dot{Z}_{i-1}(2 - \dot{Z}_{i-1} \cdot X) \quad \text{or} \\ R_i &= \dot{Z}_{i-1}(1 - \dot{Z}_{i-1} \cdot X). \end{aligned} \quad (36)$$

The next step is the extraction of the new on-line digit $\dot{z}_i$ from R_i , denoted as $\dot{z}_i = \text{SEL}(R_i)$. This new digit is then appended to the on-line result $\dot{Z}_{i-1}$ generated in the previ-

ous iteration step; i.e., $\dot{Z}_i = \dot{Z}_{i-1} + \dot{z}_i \cdot r^{-(i-q-1)}$ ($\varphi = q$ for the reciprocal function).

For convenience, the following notations are used: $\Delta z_i = \dot{z}_i \cdot r^{-(i-q-1)}$ and $\Delta x_i = x_{i+\delta} \cdot r^{-(i+\delta)}$, which are the weighted i th on-line result digit and the weighted $(i+\delta)$ th input digit, respectively. The value of δ must satisfy the convergence and on-line requirements of both the on-line table look-up method and the adapted Newton-Raphson method. The following algorithmic description of both procedures can be given:

1. *Initialization:*

$\dot{Z}_1 = 0$; $\Delta z_0 = 0$; $X_0 = X_{init}$; /* X_{init} accumulates the first δ digits */

do $i = 1, m$

$\dot{Z}_{i-1} = \dot{Z}_{i-2} + \Delta z_{i-1}$;

$R_i = (\dot{Z}_{i-2} + \Delta z_{i-1}) * \{1 - (\dot{Z}_{i-2} + \Delta z_{i-1}) * (X_{i-1} + \Delta x_i)\}$;

generate the i th digit $\dot{z}_i$ with the on-line table look-up method, **od**;

2. *Adapted Newton-Raphson iterations:*

do $i = m+1, n$

$\dot{Z}_{i-1} = \dot{Z}_{i-2} + \Delta z_{i-1}$;

$R_i = (\dot{Z}_{i-2} + \Delta z_{i-1}) * \{1 - (\dot{Z}_{i-2} + \Delta z_{i-1}) * (X_{i-1} + \Delta x_i)\}$;

select a new on-line digit; $\dot{z}_i = SEL(R_i)$; **od**;

$SEL(R_i)$ is the selection function: $SEL(R_i) = Round(R'_i)$, where R'_i represents the L most significant digits of R_i , which are fully propagated from the $(i+L-1)$ th to the (i) th digit; i.e., the value of R_i is first truncated after digit $i+L-1$ to avoid full carry propagation. The *Round* function symmetrically rounds R'_i to the (i) th digit (the digit with weight $r^{-(i-q-1)}$).

The convergence of the adapted Newton-Raphson method is shown if the used selection function $SEL(R_i)$ guarantees that $\dot{Z}_i$ is one digit more accurate than $\dot{Z}_{i-1}$. Suppose $\dot{Z}_{i-1} = 1/X_{i-1} + \epsilon_{i-1}$ and $X_i = X_{i-1} + \xi_i$, where ϵ_{i-1} and ξ_i are the errors to $1/X_{i-1}$ and X_{i-1} , respectively, with $|\epsilon_{i-1}| \leq |c_{i-1}| \cdot r^{-(i-q-2)}$, $|c_{i-1}| < 1$, $|\xi_i| \leq (r-1) \cdot r^{-(i+\delta)}$, and δ is the on-line delay of the adapted Newton-Raphson method. Substitution of $\dot{Z}_{i-1}$ and X_i into Eq. (36) gives

$$\begin{aligned} \dot{Z}_i &= \{(1/X_{i-1}) + \epsilon_{i-1}\} \cdot \{2 - \{(1/X_{i-1}) + \epsilon_{i-1}\} \\ &\quad \times \{X_{i-1} + \xi_i\}\} = (1/X_i) + E_i \end{aligned} \quad (37)$$

with

$$E_i = \frac{\xi_i}{X_i X_{i-1}} - \frac{\xi_i}{X_{i-1}^2} - \frac{2 \cdot \epsilon_{i-1} \cdot \xi_i}{X_{i-1}} - \epsilon_{i-1}^2 \cdot X_i.$$

Because $|\xi_i| \leq (r-1) \cdot r^{-(i+\delta)}$ and $r^{-q} \leq |X| < 1$, it holds

$$\begin{aligned} |E_i| &< (r-1)^2 \cdot r^{-(i+2\delta-2q+1)} + 2(r-1) |c_{i-1}| \\ &\quad \times r^{-(i+\delta-q-1)} + (c_{i-1})^2 \cdot r^{-(i-q-3)}. \end{aligned} \quad (38)$$

From Eq. (36) it can easily be seen that $\dot{Z}_i = \dot{Z}_{i-1} + R_i$. Combining this equation with Eq. (37) gives $\dot{Z}_{i-1} + R_i = (1/X_i) + E_i$, which can be written as $\dot{Z}_{i-1} + \Delta z_i + (R_i - \Delta z_i) = (1/X_i) + E_i$, leading to

$$|\dot{Z}_{i-1} + \Delta z_i - (1/X_i)| \leq |E_i| + |R_i - \Delta z_i|. \quad (39)$$

Let $\dot{z}_i = SEL(R_i)$. From the definition of the selection function SEL , it follows that

$$|R_i - \Delta z_i| \leq (r/2) \cdot r^{-(i-q)} + r^{-(i-q+L-2)}, \quad (40)$$

where $(r/2) \cdot r^{-(i-q)}$ is the rounding error, and $r^{-(i-q+L-2)}$ is the truncation error as a result of taking only the digits with a weight larger than $r^{-(i-q+L-2)}$ into consideration by the selection at the i th iteration. Substitution of Eq. (40) in Eq. (39) yields

$$\begin{aligned} |\dot{Z}_i - (1/X_i)| \\ \leq \{|E_i| r^{-(i-q-1)} + 1/2 + r^{-(L-1)}\} \cdot r^{-(i-q-1)}. \end{aligned} \quad (41)$$

From Eq. (41) it follows that

$$|c_i| \leq |E_i| r^{-(i-q-1)} + 1/2 + r^{-(L-1)}. \quad (42)$$

The value of E_i decreases as δ and i increase; therefore, the (i) th approximation $\dot{Z}_i$ will have an error less than $r^{-(i-q-1)}$, if the values of the parameters m ($i \geq m+1$), δ , and L are chosen such that $\{|E_i| r^{-(i-q-1)} + 1/2 + r^{-(L-1)}\} < 1$. Thus, the convergence of the adapted Newton-Raphson method, with selecting $\dot{z}_i$ by means of the selection function $SEL(R_i)$, can be obtained by a proper choice of δ , m , and L .

The next question is, can δ , m , and L be chosen such that the adapted Newton-Raphson method is on-line. The output $\dot{Z}_i$ is on-line if $\dot{z}_i$ can be represented by one single digit with a weight of $r^{-(i-q-1)}$. If we require that $|R_i| < r^{-(i-q-2)} - (r/2) \cdot r^{-(i-q)} - r^{-(i-q+L-2)} = (r-1/2 - r^{-(L-1)}) \cdot r^{-(i-q-1)}$, then it holds that $\Delta z_i < r \cdot r^{-(i-q-1)}$. This means that $\dot{z}_i$ can be represented by one single digit with a weight of $r^{-(i-q-1)}$. It holds that $\dot{Z}_{i-1} = 1/X_{i-1} + \epsilon_{i-1}$, and $\dot{Z}_i = 1/X_i + E_i$ (see Eq. (37)); therefore,

$$\begin{aligned} R_i &= \dot{Z}_i - (1/X_{i-1} + \epsilon_{i-1}) = \dot{Z}_i - \dot{Z}_{i-1} \\ &= 1/X_i - 1/X_{i-1} + E_i - \epsilon_{i-1} \\ &\leq |1/X_i - 1/X_{i-1} + E_i| + |\epsilon_{i-1}|. \end{aligned} \quad (43)$$

So, it can be concluded that the output $\dot{Z}_i$ is on-line if

$$\tilde{E}_i \cdot r^{(i-q-1)} < r - 1/2 - r^{-(L-1)} \quad (44)$$

with $\tilde{E}_i = |1/X_i - 1/X_{i-1} + E_i| + |\epsilon_{i-1}|$. Since $|\xi_i| \leq (r-1) \cdot r^{-(i+\delta)}$ and $r^{-q} \leq |X| < 1$, from Eq. (37) it follows that

$$\begin{aligned} \tilde{E}_i \cdot r^{(i-q-1)} &< (r-1) \cdot r^{-(\delta-q+1)} + 2(r-1)|c_{i-1}| \\ &\times r^{-(i+\delta-q-1)} + (c_{i-1})^2 \cdot r^{-(i-q-3)} + |c_{i-1}| \cdot r. \end{aligned}$$

Furthermore, it holds $i \geq (m+1)$, such that the on-line condition of Eq. (44) becomes

$$\begin{aligned} &[(r-1) \cdot r^{-(\delta-q+1)} + 2(r-1)|c_{i-1}| \cdot r^{-(m+\delta-q)} \\ &+ (c_{i-1})^2 \cdot r^{-(m-q-2)} + |c_{i-1}| \cdot r] \\ &\leq r - 1/2 - r^{-(L-1)}. \end{aligned} \quad (45)$$

Now we have to minimize δ under the constraint of Eq. (45), with Eq. (38) and Eq. (42). The above problem is not easy to solve analytically. However, a low bound can be derived by letting m and $L \rightarrow \infty$. Then Eq. (42) becomes $|c_{i-1}| \leq \frac{1}{2}$, $i \geq m$. Substitution in Eq. (45) leads to $(r-1) \cdot r^{-(\delta-q+1)} + (r/2) \leq r - \frac{1}{2}$. From this condition, it follows that $\delta = q$, $\forall r$. This implies that in the limit case a perfect match with the table look-up system can be obtained. The appropriate values of m and L can be found by through-substitution by using Eq. (42) and Eq. (38), while checking the condition of Eq. (45). The start value, c_m , is obtained from the error in the table look-up system. It holds that $\dot{Z}_i = 1/X_i + c_m \cdot r^{-(i-q-1)}$ and $\dot{Z}_i = \dot{Z}_i$ for the table look-up system, giving according to Eq. (15) $|\dot{Z}_i - 1/X_i| \leq (r/2) \cdot r^{-(i-q)}$, yielding $|c_m| \leq \frac{1}{2}$. By starting with $c_m = \frac{1}{2}$ the calculated parameter values of δ , m , and L are given in Table II. The table shows that $\delta = q+1$ for $r=2$ and $\delta = q$ for $r \geq 3$. An example of the adapted Newton-Raphson method with the table look-up method for the initialization is shown in Fig. 5.

TABLE II

Several Values of the Parameters δ , m , and L for Which the On-Line Condition Are Satisfied

r	δ	m		L	
		$q-1$	$q-2$	$q-1$	$q=2$
$r=2$	$q+1$	5	6	8	7
$r=3$	q	5	5	4	7
$r=4$	q	4	5	3	3
$r=8$	q	$q+2$		3	

Round				$\dot{z}_{i-1}$	$\dot{z}_i$
$X_1 = 0.101$	$1/X_1 = 01.0 \dots + 0.1$	$\dot{Z}_1 = 1.$			1
$X_2 = 0.101\bar{1}$	$1/X_2 = 01.10 \dots + 0.01$	$\dot{Z}_2 = 10.$		1	0
$X_3 = 0.101\bar{1}\bar{1}$	$1/X_3 = 01.110 \dots + 0.001$	$\dot{Z}_3 = 10.0$		0	0
$X_4 = 0.101\bar{1}\bar{1}\bar{1}$	$1/X_4 = 01.1101 \dots + 0.0001$	$\dot{Z}_4 = 10.01$		1	1
$X_5 = 0.101\bar{1}\bar{1}\bar{1}0$	$1/X_5 = 01.11010 \dots + 0.00001$	$\dot{Z}_5 = 10.011$		1	1
table initialization part					
$X_6 = 0.101\bar{1}\bar{1}\bar{1}01$	$R_6 = 0.00001001011\bar{1}01$	$\dot{Z}_6 = 10.0111$		$\dot{z}_6 = \text{SEL}(R_6) = \bar{1}$	
$X_7 = 0.101\bar{1}\bar{1}\bar{1}010$	$R_7 = 0.000001001010\bar{1}01$	$\dot{Z}_7 = 10.01111$		$\dot{z}_7 = \text{SEL}(R_7) = 1$	
$X_8 = 0.101\bar{1}\bar{1}\bar{1}010$	$R_8 = 0.000000001010101\bar{1}101$	$\dot{Z}_8 = 10.011110$		$\dot{z}_8 = \text{SEL}(R_8) = 0$	
newton raphson iterations					

FIG. 5. Illustration of the adapted Newton-Raphson method with the table look-up method for the initialization. $X = 0.101\bar{1}\bar{1}010\bar{1}$ is a radix-2 number with $q=1$, $\delta=2$, and $m=5$.

5. IMPLEMENTATION OF AN ON-LINE RECIPROCAL UNIT

In this section, the implementation of the adapted Newton-Raphson part of the reciprocal unit is considered. The formula for the residue R_i can be written as a recursive equation:

$$\begin{aligned} R_i &= R_{i-1} - 2 \cdot \dot{Z}_{i-2} \cdot X_{i-1} \cdot \Delta z_{i-1} + \Delta z_{i-1} - \dot{Z}_{i-2} \cdot \dot{Z}_{i-2} \\ &\times \Delta x_i - 2 \cdot \dot{Z}_{i-2} \cdot \Delta z_{i-1} \cdot \Delta x_i - X_i \cdot \Delta z_{i-1} \cdot \Delta z_{i-1}. \end{aligned}$$

Since $|R_i| < (r-1/2 - r^{-(L-1)}) \cdot r^{-(i-q-1)}$, R_i can be scaled by a factor r during each iteration, resulting that the most significant digit of R_i remains in the same position. This simplifies the selection unit. Substitution of $R_i^* = R_i \cdot r^{-(i-q-2)}$, $\Delta z_{i-1} = \dot{z}_{i-1} \cdot r^{-(i-q-2)}$, and $\Delta x_i = x_{i-\delta} \cdot r^{-(i+\delta)}$ into Eq. (46) gives

$$\begin{aligned} R_i^* &= r \cdot R_{i-1}^* - 2 \cdot \dot{Z}_{i-2} \cdot X_{i-1} \cdot \dot{z}_{i-1} + \dot{z}_{i-1} \\ &- \dot{Z}_{i-2} \cdot \dot{Z}_{i-2} \cdot x_{i+\delta} \cdot r^{-(\delta+q+2)} \\ &- 2 \cdot \dot{Z}_{i-2} \cdot \dot{z}_{i-1} \cdot x_{i+\delta} \cdot r^{-(i+\delta)} \\ &- X_i \cdot (\dot{z}_{i-1})^2 \cdot r^{-(i-q-2)}. \end{aligned} \quad (47)$$

Equation (47) can be partitioned into three independent suboperations, which can be implemented by using propagation-free carry-save adders (i.e., only neighboring carry-propagation). The three suboperations are (a) the multiplication of $\dot{Z}_{i-2} \cdot X_{i-1}$; (b) the multiplication of $\dot{Z}_{i-2} \cdot \dot{Z}_{i-2}$; (c) the calculation of R_i^* . The block diagrams of these three components are shown in Fig. 6 and Fig. 7. Because the iteration index of $\dot{Z}$ in the calculations (a) and (b) is $(i-2)$, and in (c) is $i-1$, the three calculations can be done completely independent in each iteration cycle.

If the terms $2 \cdot \dot{Z}_{i-2} \cdot \dot{z}_{i-1} \cdot x_{i+\delta}$ and $X_i \cdot (\dot{z}_{i-1})^2$ are computed first and then shifted, a complex circuitry will result in addition to a large delay in the circuit. To avoid this, the terms $\dot{Z}_{i-2}$ and X_i are shifted each iteration one digit posi-

FIG. 6. The incremental multiplier for $X_i Z_{i-1}$ or $Z_i Z_{i-1}$. D.M. is a digit-word multiplier.

tion to the right (equivalent to the multiplication with r^{-1}). The constant factors $r^{-\delta}$ and $r^{(q+1)}$ can be implemented by a fixed offset of wires.

The most simple implementation is obtained for $r = 2$. The digit multipliers can then be reduced to sign converters of the input terms. The two's complement multipliers described in [16] can be used with a slight modification to incorporate redundant numbers.

The full precision word length of $X_{i-1}Z_{i-2}$, $Z_{i-2}Z_{i-2}$, and R_i^* is $O(2n)$ digits. However, only n digits are required as output and only L digits of R_i^* are used to select an on-line digit. Thus, to generate a result of n digits, a sufficient condition is that the absolute error in R_i^* is less than $r^{-(n+L)}$ for $1 \leq i \leq n$. This condition can be satisfied if we chose the length of the adders and multiplier outputs to be $O(n + L + \log_r(5n/2))$ and $O(n + L + 2 \log_r(5n/2))$ digits, respectively.

The calculation of R_i^* requires the longest time among the three components, so the maximal delay per iteration is equal to the time of calculating R_i^* . For $r = 2$, the delay per iteration is approximately equal to the sum of the delays of three carry-propagation-free adders and one selection operation. Since all additions are done by carry-save adders, the

FIG. 7. Calculating the residue R_i^* .

circuit delay is small and high computation speed can be obtained.

6. CONCLUSION

A new model for on-line computation has been presented. The model allows easy determination of the on-line properties of arithmetic functions. The derived on-line properties of a certain arithmetic function are lower bounds and can, in general, not be improved by any of the iterative approximation algorithms developed for that function. Usually, the on-line delays of the iterative approximation algorithms are larger than those derived from the model.

It has also been shown that an efficient table look-up implementation of the model exists. Although the table system is impractical for the complete generation of on-line functions, it can sometimes be used as a seed-table for an iterative approximation algorithm. An application of this technique has been shown through the on-line reciprocal evaluation, by using such a seed-table, followed by an adapted Newton-Raphson algorithm. It is to be expected that seed-tables can also improve the on-line performance of other on-line algorithms, since the actual on-line delay is in most algorithms determined by the first iterations.

APPENDIX: NOMENCLATURE

δ	on-line delay
r	radix of number system
η	redundancy factor ($r/2 \leq \eta \leq r-1$)
x_i	i th digit of the input operand ($i \geq 1$)
X_i	input operand up to digit $x_{i+\delta}$
Z_i	infinite precision result of applying a function F to X_i ; i.e., $Z_i = F(X_i)$
$\bar{Z}_i$	Z_i symmetrically rounded and truncated at the i th digit
$\hat{Z}_i$	approximation to Z_i by an iterative algorithm
$\hat{Z}_i$	on-line result up to the i th digit
z_i	i th digit of Z_i
$\bar{z}_i$	i th digit of $\bar{Z}_i$
$\bar{z}_j$	j th digit of $\bar{Z}_{i-1}$
$\hat{z}_i$	i th digit of the on-line result
q	normalization factor of floating-point operand, indicates allowable number of leading zeros; ($q-1$)
φ	weight of the most significant digit of the on-line result, i.e., r^φ
∇	gradient
$\text{sign}(x)$	sign of a digit ($\text{sign}(x)$) or a number ($\text{sign}(X)$)
$\lceil \cdot \rceil$	ceiling function
$\text{com}(x)$	radix- r complement of digit x ; i.e., $\text{com}(x) = -\text{sign}(x) \cdot r + x$

ACKNOWLEDGMENTS

The authors thank the referees for their valuable comments, which have improved the paper considerably.

REFERENCES

1. Trivedi, K. D., and Ercegovac, M. D. On-line algorithms for division and multiplication. *IEEE Trans. Comput.* C-27, (July 1977).
2. Ercegovac, M. D. An online arithmetic: An overview. *SPIE*, Vol. 495, *Real Time Signal Processing VII*, 1984.
3. Irwin, M. J. A pipelined processing unit for on-line division. *Proc. 5th Symposium on Computer Arithmetic*, 1978.
4. Owens, R. M. Compound algorithms for digit online arithmetic. *Proc. 5th Symposium on Computer Arithmetic*, 1981.
5. Owens, R. M. Digit online algorithms for pipelined architectures. Ph.D. thesis, Department of Computer Science, The Pennsylvania State University, 1980.
6. Watanuki, O., and Ercegovac, M. D. Floating-point online arithmetic algorithms. *Proc. 5th Symposium on Computer Arithmetic*, 1981.
7. Watanuki, O., and Ercegovac, M. D. Error analysis of certain floating-point on-line algorithms. *IEEE Trans. Comput.* C-32, 4 (Apr. 1983).
8. Trivedi, K. D., and Rusnak, J. G. High radix on-line division. *Proc. 4th Symposium on Computer Arithmetic*, 1978.
9. Owens, R. M. Techniques to reduce the inherent limitations of fully digit on-line arithmetic. *IEEE Trans. Comput.* C-32, (Apr. 1983).
10. Owens, R. M., and Irwin, M. J. On-line algorithms for the design of pipelined architectures. *Proc. Sixth Annual Symposium of Computer Architecture*, Philadelphia, PA, Apr. 1979.
11. Delugish, B. G. A class of algorithms for automatic evaluation of certain elementary functions in a binary computer. Ph.D. thesis, Department of Computer Science, University of Illinois, 1975.
12. Ercegovac, M. D., and Lang, T. On-the-fly conversion of redundant into conventional representation. *IEEE Trans. Comput.* C-36, 7 (July 1987).
13. Tu, P. K.-G., and Ercegovac, M. D. A radix-4 on-line division algorithm. *Proc. 8th Symposium on Computer Arithmetic*, Como, Italy, 1987.
14. Ercegovac, M. D. and Lang, T. On-line scheme for computing rotation factors. *J. Parallel Distrib. Comput.* 5, 3 (June 1988).
15. Lin, H. X., and Sips, H. J. A novel floating-point on-line division algorithm. *Proc. 8th Symposium on Computer Arithmetic*, Como, Italy, 1987.
16. Sips, H. J. Bit-sequential arithmetic for parallel processors. *IEEE Trans. Comput.* C-33, 1 (Jan. 1984).

HENK J. SIPS was born in Amsterdam, The Netherlands on October 14, 1950. He holds a M.Sc. (1976) in electrical engineering and a Ph.D. (1984) of the Delft University of Technology. Currently, he is an associate professor in applied computer science at Delft University of Technology and a professor in computer science at Brabant University. Dr. Sips is also consultant at the Institute of Applied Computer Science, Delft, The Netherlands. His research interests include computer architecture, parallel programming systems, parallel algorithms, and computer arithmetic.

HAI XIANG LIN was born in Guangxi, China, on September 23, 1961. He received a M.Sc. degree from the Faculty Mathematics and Computer Science at Delft University of Technology in 1986. For this work he received the "Delftse Hogeschool Fonds" award. Since then he works as a graduate student at IBBC-TNO Netherlands Organization for Applied Scientific Research toward his Ph.D., where he is currently involved in a project on parallelizing of finite element analysis software. His research interests include parallel computer system architectures, parallel algorithms, and computer arithmetic.

Received July 30, 1987; revised September 30, 1988