ON-LINE MULTIPLICATIVE NORMALIZATION

A.L. Grnarov
Electrical Engineering Department,
University of Skopje, Yugoslavia

M.D. Ercegovac
UCLA Computer Science Department
University of California, Los Angeles

ABSTRACT

In this article we describe a derivation and an algorithm for on-line multiplicative normalization of fractions. The algorithm is a variation of the continued product normalization algorithm and it is used for on-line evaluation of elementary functions.

I. INTRODUCTION

In several algorithms for evaluation of elementary functions [1-4] the computation requires an iterative transformation of the argument into a constant. If the evaluation of a function is embedded in a sequence of arithmetic operations, an additional improvement in speed can be obtained by applying on-line arithmetic [5-8].

In this paper we consider the problem of on-line multiplicative normalization [9,10,11]. The multiplicative normalization of a given argument $|X_0| \in [1/2, 1)$ is defined as a sequence of transformations such that $X_0 \prod_{i=0}^{m} M_i \rightarrow 1$ for sufficiently large m. The multipliers are of the form $M_k = 1 + S_k\, r^{-k}$, where r is the radix and s_k is a digit in a redundant radix r number system. The sequence $< S_0, S_1, ..., S_m >$ is interpreted as a continued product representation of $1/X_0$ and it is used to compute results such as quotients and logarithms in m steps. The form of the multipliers simplifies the implementation of algorithms for normalization and result evaluation [1], [4].

The on-line arithmetic algorithms have the following properties: (a) the results are obtained in a digit-by-digit manner, beginning with the most significant digit, and (b) in order to obtain the j-th digit of the result it is necessary and sufficient to have (j + δ) most significant digits of the arguments. The on-line delay δ is usually a small positive integer. The on-line algorithms achieve speedup by allowing overlap between successive computations at the digit level [4], [6], [7], [9].

In the following sections we derive an on-line multiplicative normalization algorithm for general radix with on-line delay δ = 1.

II. DERIVATION OF THE ALGORITHM

When all digits of the argument $X_0 = \sum_{i=0}^{m} x_i\, r^{-i}$ are available at the beginning of the operation, the multiplicative normalization is performed recursively as

$$X_j = X_{j-1}(1+S_{j-1}r^{-j+1}) \qquad j = 1,2,...,m \tag{2.1}$$

by choosing the values of S_j digits so that X_m converges to 1 [4]. This procedure is modified as follows for the on-line mode of operation.

Let Z_j be the value represented by the first j+ δ digits of the argument X_0 :

$$Z_j = \sum_{i=0}^{j+\delta} x_i r^{-i} \tag{2.2}$$

Recursively,

$$Z_j = Z_{j-1} + x_{j+\delta}r^{-j-\delta}, \quad j = 1,2,...,m-\delta \tag{2.3}$$

where $x_{j+\delta}=0$ for $j > m-\delta$ and

$$Z_0 = \sum_{i=0}^{\delta} x_i r^{-i}.$$

The digits x_j are assumed to belong to a symmetric redundant digit set, i.e.:

$$x_j \in D_\rho = \{-\rho,...,-1,0,1,...,\rho\}$$

where

$$r/2 \leqslant \rho < r$$

In on-line normalization, instead of the argument X_0 , its on-line form Z_j is used. In order to obtain the same rate of convergence as in the conventional normalization, the following must hold for the partially normalized arguments Y_j:

$$Y_j = Z_j D_j \quad X_m = Y_m \tag{2.4}$$

where

$$D_j = \prod_{i=1}^{j}(1 + S_{i-1}r^{-i+1}) \tag{2.5}$$

is the j-th product of the normalizing multipliers and

$$S_i \in = \{-\sigma,...,-1,0,1,...,\sigma\}$$

The on-line multiplicative normalization recursion follows from (2.3-5):

$$Y_j=(Z_{j-1} + x_{j+\delta}r^{-j-\delta})D_{j-1}(1 + S_{j-1}r^{-j+1}) \tag{2.6}$$

$$=(Y_{j-1} + D_{j-1}x_{j+\delta}r^{-j-\delta})(1 + S_{j-1}r^{-j+1})$$

where $D_0=1$, and $j=1,2,...,m$. For implementation reasons, this recursion is replaced by the scaled remainder recursion:

$$R_j = (rR_{j-1} + D_{j-1}x_{j+\delta}r^{-\delta-1})(1+S_{j-1}r^{-j+1}) + S_{j-1} \tag{2.7}$$

where

$$R_j = r^{j-1}(Y_j-1) \tag{2.8}$$

The j-th digit of the continued product representation can be selected in several ways using the scaled remainder recursion.

CH1892-9/83/0000/0151$01.00 © 1983 IEEE

We propose to determine the value of S_j as a function of

$$A_j = rR_j + D_j x_{j+\delta+1} r^{-\delta-1} \tag{2.9}$$

in such a way that the error e_j satisfies

$$|e_j| = |1 - Y_j| \leqslant \frac{\rho}{r-1} r^{-j+1} \tag{2.10}$$

i.e., the partially normalized argument x_j differs by no more than r^{-j+1} from 1. According to (2.8) this implies that

$$|R_j| \leqslant \frac{\rho}{r-1} \tag{2.11}$$

Now we establish the selection rules. From the remainder recursion (2.7), the interval boundaries for every digit value $S_j \epsilon D_r$ are defined by the following expressions

$$\underline{B}(S_j) = \frac{r^j(\underline{R}_{j+1} - S_j)}{r^j + S_j} \tag{2.12}$$

and

$$\overline{B}(S_j) = \frac{r^j(\overline{R}_{j+1} - S_j)}{r^j + S_j}$$

so that the digit value S_j can be selected whenever

$$\underline{B}(S_j) \leqslant A_j \leqslant \overline{B}(S_j) \tag{2.13}$$

In the expressions (2.12) and (2.13), $\underline{R}_{j+1}$ and $\overline{R}_{j+1}$ denote the smallest and largest possible value of R_{j+1}.

A. Determination of the Remainder Bounds

The values of $\underline{R}_j$ and $\overline{R}_j$ must satisfy the error, containment and continuity conditions.

A1. Error condition (2.10) implies that

$$|\underline{R}_j|, |\overline{R}_j| \leqslant \frac{\rho}{r-1}$$

A2. Continuity condition of the remainder range requires that the selection intervals overlap. Moreover, these interval overlaps should be as large as possible in order to allow the use of low-precision comparison constants and an estimate of A_j as the selection argument. The size of the interval for the selection of S_j is

$$\Delta(S_j) = \overline{B}(S_j) - \underline{B}(S_j) \tag{2.15}$$

$$= \frac{r^j}{r^j + S_j}(\overline{R}_{j+1} - \underline{R}_{j+1})$$

According to (2.15)

$$(\Delta_j)_{\max} = \frac{r^j}{r^j - \sigma}(\overline{R}_{j+1} - \underline{R}_{j+1}) \tag{2.16}$$

$$(\Delta_j)_{\min} = \frac{r^j}{r^j + \sigma}(\overline{R}_{j+1} - \underline{R}_{j+1}) \tag{2.17}$$

It can be seen from (2.16) and (2.17) that $(\Delta_j)_{\max}$ is a decreasing and $(\underline{\Delta}_j)_{\min}$ is an increasing function of j. Also, $(\Delta_\infty)_{\max} = (\Delta_\infty)_{\min} = \overline{R}_{j+1} - \underline{R}_{j+1}$. As an example, Figure 1 illustrates $\Delta_j/(\overline{R}_{j+1} - \underline{R}_{j+1})$ for r=16, $\rho = 15$ and $\sigma = 10$.

Since $\overline{B}(S_{j+1})$ and $\underline{B}(S_j)$ are monotonically decreasing functions of S_j, the continuity condition becomes

$$\overline{B}(S_{j+1}) - \underline{B}(S_j) \geqslant 0 \tag{2.18}$$

for $-\sigma \leqslant S_j < \sigma$. From (2.13) and (2.18) it follows that the following condition must be satisfied [10].

$$\overline{R}_{j+1} \geqslant \frac{r^j + \underline{R}_{j+1}}{r^j + S_j} + \underline{R}_{j+1} \tag{2.19}$$

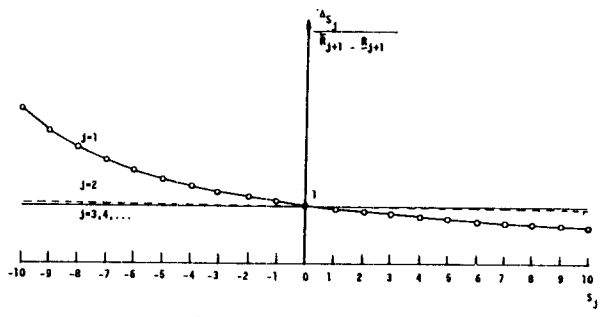

Figure 1: Selection Interval Overlaps

According to (2.19) $\overline{R}_{j+1}$ is a monotonically decreasing function of S_j and j.

The monotonicity of the remainder upper bound is illustrated in Figure 1. For $j \to \infty$, (2.19) becomes

$$\overline{R}_{j+1} - \underline{R}_{j+1} \geqslant 1$$

or, for $\underline{R}_{j+1} = -\overline{R}_{j+1}$

$$\overline{R}_\infty \geqslant \frac{1}{2}$$

A3. Containment Condition

The scaled remainder recursion can be decomposed into two mappings:

$$A_j = rR_j + D_j X_{j+\delta+1} r^{-\delta-1} \tag{2.20}$$

$$R_{j+1} = A_j(1 + S_j r^{-j}) + S_j \tag{2.21}$$

For the convergence of the algorithm, the following must hold

$$(a) \quad \underline{B}_j(\sigma) \leqslant A_j \leqslant \overline{B}_j(\sigma) \tag{2.22}$$

$$(b) \quad \underline{R}_{j+1} \leqslant R_{j+1} \leqslant \overline{R}_{j+1} \tag{2.23}$$

According to (2.8) and (2.12)

$$(R_{j+1})\ \min = \underline{R}_{j+1} \text{ for } A_j = \underline{B}_j(S_j),\ S_j \epsilon \{-\sigma, \ldots, \sigma\}$$

and

$$(R_{j+1})\ \max = \overline{R}_{j+1} \text{ for } A_j = \overline{B}_j(S_j),\ S_j \epsilon \{-\sigma, \ldots, \sigma\}$$

The condition (b) in (2.23) is always satisfied as illustrated in Figure 2.

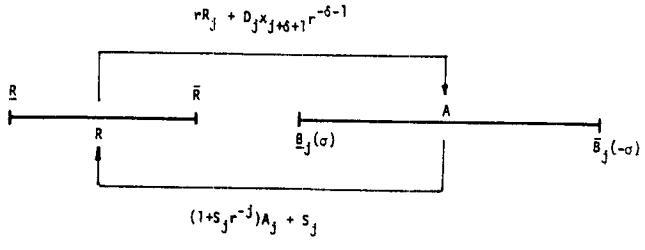

Figure 2: A-R Mappings

The condition (a) is now considered in detail. Since $X_0 \epsilon [\frac{1}{2},1)$, the first digit S_0 is either 0 or 1. In order to simplify the selection in the initial steps of the algorithm, the selection of S_0 is performed in such a way that

$$\max|R_1| = \max|(1+S_0)Z_1 - 1| \tag{2.24}$$

is minimal. This happens if S_0 is defined as

$$S_0 = \begin{cases} 0 \text{ if } Z_1 > \hat{C} \\ 1 \text{ otherwise} \end{cases} \tag{2.25}$$

where $\hat{C}$ is an estimate of C. The selection constant C=2/3 for the selection of S_0 can be determined from the condition

$$1 - C = 2C - 1 \tag{2.26}$$

According to (2.25) and (2.20)

$$(\hat{C}-1)r - \rho r^{-\delta-1} \leqslant A_1 < (2\hat{C}-1)r + 2\rho r^{-\delta-1} \tag{2.27}$$

The containment condition is now defined as

$$r(\bar{R}_2+S_1^*)\frac{}{r-S_1^*} \geqslant (2\hat{C}-1)r + 2\rho r^{-\delta-1} \tag{2.28}$$

and

$$\frac{r(\sigma - \underline{R}_2)}{r+\sigma} \geqslant (1-\hat{C})r + \rho r^{-}\delta - 1 \tag{2.29}$$

where S_1^* is the magnitude of the largest negative S_1 digit. It can be determined from (2.28) and the fact that $(\bar{R}_2)_{\min} > 1/2$ and $(\bar{R}_2)_{\max} < \rho/(r-1)$.

B. Determination of $\bar{R}_j$, $\underline{R}_j$ and σ

The containment condition for $j \geqslant 2$ becomes:

$$\frac{r^j(\bar{R}_{j+1}+\sigma)}{r^j-\sigma} \geqslant r\bar{R}_j + D_j\rho r^{-\rho-1} \tag{2.31}$$

and

$$\frac{r^j(\underline{R}_{j+1}-\sigma)}{r^j+\sigma} \leqslant r\underline{R}_j - D_{,}\rho\, r^{-\delta-1} \tag{2.32}$$

From (2.31) and (2.32), it follows that

$$\bar{R}_{j+1} \geqslant (1-\frac{\sigma}{r^j})(r\bar{R}_j + D_j\rho r^{-\delta-1}) - \sigma \tag{2.33}$$

and

$$\underline{R}_{j+1} \leqslant (1+\frac{\sigma}{r^j})(r\underline{R}j - D_j\rho r^{-\delta-1}) + \sigma \tag{2.34}$$

For $j \to \infty$

$$\bar{R}_\infty \geqslant \frac{1}{r-1}(\sigma - D\rho r^{-\delta-1}) \tag{2.35}$$

and

$$\underline{R}_\infty \geqslant -\frac{1}{r-1}(\sigma - D\rho r^{-\delta-1}) \tag{2.36}$$

From the expressions (2.33) to (2.36) it follows that both $\bar{R}_j$ and $\underline{R}_j$ are decreasing functions of j. Since

$$\bar{R}_j - \bar{R}_{j+1} > 0 \tag{2.37}$$

and

$$\underline{R}j - \underline{R}_{j+1} > 0 \tag{2.38}$$

we obtain that $\bar{R}j$ and $\underline{R}_j$ must satisfy

$$\bar{R}_j < \frac{\sigma - z + \frac{z\sigma}{r^j}}{r-1-\frac{\sigma}{r^{j-1}}} \tag{2.39}$$

and

$$\underline{R}^j > -\frac{\sigma - z - \frac{z\sigma}{r^j}}{r-1+\frac{\sigma}{r^{j-1}}} \tag{2.40}$$

where $z = \dfrac{D_j\rho}{r^{\delta+1}}$

According to (2.19) the continuity condition becomes

$$\underline{R}_2 \leqslant \frac{(r-S_1^*)\,\bar{R}_2 - r}{r-S_1^*+1} \tag{2.41}$$

where S_1^* is a function of r and σ only.

From (2.31) and (2.32)

$$\bar{R}_j \leqslant \frac{1}{r}\left[\frac{r^j(\bar{R}_{j+1}+\sigma)}{r^j-\sigma} - D_j\rho r^{-\delta-1}\right] \tag{2.42}$$

and

$$\underline{R}_j \geqslant \frac{1}{r}\left[\frac{r^j(\underline{R}_{j+1}-\sigma)}{r^j+\sigma} + D_j\rho r^{-}\delta-1\right] \tag{2.43}$$

Assuming that $\bar{R}_j'+1=\bar{R}_\infty$, $\underline{R}_j'+1=\underline{R}_\infty$ and $D_j = 2$, the boundary values of $R_j, j = 2,3,\ldots,j'$ can be obtained applying recursions (2.42) and (2.43). Determination of $\bar{R}_j$ and $\underline{R}_j$ in such a way has the following advantages: (i) The containment condition is always satisfied, and (ii) The continuity of selection intervals is maximal.

C. Selection Precision

The selection interval overlap Δ_{OVj} is given by

$$\Delta_{OVj} = \frac{r^j}{(r^j+S_{j+1})(r^j+S_j)} \tag{2.44}$$

$$\left[(\bar{R}_{j+1}-\underline{R}_{j+1})\,(r^j+S_j) - \underline{R}_{j+1} - r^j\right]$$

It can be seen that Δ_{OVj} is an increasing function of j and that

$$\Delta OV_\infty = \bar{R}_\infty - \underline{R}\infty - 1 = \frac{2(\sigma-z)-r+1}{r-1} \tag{2.45}$$

For selection of S_0, $\delta + 1$ digits of the argument should be known Acording to [10]

$$A_1 = rR_1 + \frac{D\rho}{r^{\delta+1}}$$

i.e., for selection of S_1 at most $\delta + 2$ digits are necessary. Since Δ_{OVj} is an increasing function of j, the smallest overlap for the remaining steps occurs for j=1.

Assuming that A_i is in a redundant form, S_i can be determined from an estimate $\hat{A}_i$ of A_i such that the following holds

$$A_i = \hat{A}_i + \sum_{i=k_j+1}^{m} d_i r^{-i} \tag{2.46}$$

and

$$\sum_{i=k_j+1}^{m} \rho r^{-i} \leqslant 2^{-t_j}$$

Consequently, k_j should satisfy $r^{-k_j} - r^{-m} \leqslant 2^{-t_j}$.

Since $|A_j|_{max} < \sigma+1$, the total number of bits necessary for determination of S_j is given by

$$b_j = \left(\left\lceil \log_2 \delta \right\rceil + 1\right) + k_j\left(\left\lceil \log_2 r + 1 \right\rceil\right) \tag{2.47}$$

D. Selection of S_j by Rounding

In the case when

$$-S_j - \underline{B}(S_j) \geq 1/2 \tag{2.48}$$

and

$$\bar{B}(S_j) + S_j \geq 1/2 \tag{2.49}$$

determination of S_j can be performed by rounding. The condition (2.48) is satisfied if j $\geq$j'

$$j' \geq \left\lceil \frac{\log\left\lceil \frac{\sigma(\sigma-1/2)}{(-R_{j+1}-1/2)} \right\rceil}{\log r} \right\rceil \tag{2.50}$$

while condition (2.49) is always satisfied.

The number of necessary bits for determination of S_j, for j $\geq$j', can be obtained as $b_j = (\left\lceil \log_2 \sigma \right\rceil + 1) + 2$

E. Simplification of the Recursion

The exponential term $A_j S_j r^{-j}$ is of a diminishing influence on R_{j+1} as j increases. Since $A_j$is in the redundant form, at the j-th step the digits of $R_{j'}$ (j'=j+1,j+2,...,2j-3) necessary for selecting of $S_{j'}$are all known. For a precision of m digits, all remaining S-digits are actually available when

$$j \geq j_1 \quad \text{and} \quad j_1 = \lceil (m+3)/2 \rceil \tag{2.51}$$

After this step the recursion can be simplified

$$R_j = rR_{j-1} + D_{j-1}x_{j+\delta}r^{-\delta-1} + S_{j-1} \tag{2.52}$$

The effect of the term D_j on the remainder computation is determined in a similar manner. By using $D_j = D_{j_2}$ for j $\geq j_2$ an error is introduced:

$$|\Delta| \leq 2\sigma\rho r^{-\delta-1}(r^{-j_2} - r^{-m-1}) \tag{2.53}$$

Assuming that the impact of this error on the result should be less than ρr^{-m-1} we find

$$j_2 = \left\lceil \frac{m+1}{2} - \delta + \frac{\log 2\sigma}{\log r} \right\rceil \tag{2.54}$$

For different radices, the values of δ_{min}, corresponding $\sigma(\sigma_{min})$ and values of $\sigma(\sigma_1)$ for δ=1 are given in Table 1.

TABLE 1

RADIX	$\hat{C}$	δ_{min}	σ_{min}	S^*_1	$\delta = 1$	
					σ_1	S^*_1
2	3/4	2	1	-1		
4	3/4	1	3	-2		
8	5/8	0	7	-3	5	-2
16	10/16	0	12	-4	10	-3
32	21/32	0	24	-9	21	-8
64	42/66	0	44	-16	41	-15
128	85/128	0	88	-33	85	-32
256	170/256	0	172	-64	169	-63

III. Radix-16 On-Line Multiplicative Normalization

As an illustration, we discuss the case of radix-16 algorithm with $\rho = 15$, $\delta = 1$, $\hat{C} = \frac{10}{16}$, $(S_1)_{min} = -3$, and $\sigma = 10$

The selection intervals for $S_1 = f_1(16A_1)$ and $S_2 = f_2(16A_2)$ are computed using (2.12) and shown in Tables 2 and 3. It can be seen that the intervals for selecting $S_1 \in \{-4,...,10\}$ overlap.

According to (2.50) for $j \geq j' = 3$, the selection can be performed by rounding.

S1	a	b
10	-104.70	-91.71
9	-98.65	-85.13
8	-92.09	-78.02
7	-84.97	-70.28
6	-77.19	-61.84
5	-68.68	-52.59
4	-59.31	-42.42
3	-48.96	-31.18
2	-37.46	-18.69
1	-24.60	-4.73
0	-10.14	10.98
-1	6.25	28.78
-2	24.98	49.12
-3	46.60	72.59
-4	71.81	99.97
-5	101.62	132.33

TABLE 2: S_1 Selection Intervals

S2	a	b
10	-164.11	-143.81
9	-149.27	-128.90
8	-134.32	-113.87
7	-119.25	-98.73
6	-104.08	-83.48
5	-88.78	-68.10
4	-73.37	-52.61
3	-57.84	-37.00
2	-42.19	-21.27
1	-26.41	-5.41
0	-10.52	10.57
-1	5.51	26.67
-2	21.65	42.90
-3	37.93	59.26
-4	54.33	75.75
-5	70.87	92.37
-6	87.54	109.13
-7	104.34	126.01
-8	121.27	143.04
-9	138.35	160.20
-10	155.56	177.50

TABLE 3: S_2 Selection Intervals

Algorithm NORM-16

STEP 1 {Initialization}

$$Z_1=\sum_{i=0}^{2} X_i 16^{-i}$$

$$S_0 \leftarrow 1 \text{ if } \frac{1}{2} \leqslant Z_1 \leqslant \frac{10}{16}$$

$$S_0 \leftarrow 0 \text{ if } \frac{10}{16} < Z_1 \leqslant 1$$

$$D_1 \leftarrow (1+S_0)$$

$$R_1 \leftarrow ((1+S_0)Z_1-1)$$

STEP 2 {Recursion}

for j= 1,2,..., m-1 **do**:

2.1: $A_j \leftarrow 16R_j+D_jX_{j+2}16^{-2}$

2.2: $S_j \leftarrow f(\hat{A}_j)$

2.3: $R_{j+1} \leftarrow (1+S_j16^{-j})\ A_j+S_j \quad \text{if } j < j_1$

$R_{j+1} \leftarrow A_j + S_j \quad \text{if } j \geqslant j_1$

$D_{j+1} \leftarrow D_j(1+S_j16^{-j}) \quad \text{if } j < j_2$

$D_{j+1} \leftarrow D_j \quad \text{if } j \geqslant j_2$

end

where $f(\hat{A}_j)$ is the selection function, implemented as a set of switching functions, $j' = 3, j_1 = \left\lfloor \frac{m+3}{2} \right\rfloor$ and $j_2 = \left\lfloor \frac{m+1}{2} \right\rfloor$.

An example of the on-line normalization algorithm is given in Figure 3.

$$x = 0.5079589839_{10} = 0.820999785_{16}$$

j	Aj	Sj	Rj	Dj	Yj
0	0.0000000000	1	0.0156250000	2.0000000000	1.0156250000
1	0.2500000000	0	0.2500000000	2.0000000000	1.0156250000
2	4.0703125000	-4	0.0067138672	1.9687500000	1.0000262260
3	0.1766357422	0	0.1766357422	1.9687500000	1.0000431240
4	2.8953857422	-3	-0.1047467981	1.9686598778	0.9999984017
5	-1.6067380700	2	0.3932588654	1.9686636327	1.0000003750
6	6.3459724920	-6	0.3459702225	1.9686629286	1.0000000206
7	5.5970442771	-6	-0.4029558480	1.9686628846	0.9999999985
8	-6.4088431205	6	-0.4088431294	1.9686628874	0.9999999999
9	-6.5107297128	7	0.4892702865	1.9686628876	1.0000000000

Figure 3: On-line Normalization Example

IV. SUMMARY

We have presented an on-line algorithm for multiplicative normalization of fractions $|x| \epsilon [1/2,1)$ to one. The algorithm has the on-line delay of one. A radix 16 version has been illustrated. A modular implementation suitable for VLSI technology is discussed in [9,10]. As expected, the implementation is relatively complex with respect to the internal module structure but it has simple overall organization. The normalization algorithms are of potential use in techniques for evaluation of functions. The on-line property allows additional speed-up by overlapping successive operations.

Acknowledgements: This work has been supported in part by a research grant from the Batelle Memorial Institute and by the ONR contract N00014-79-C-0866 (Research in Distributed Processing).

REFERENCES

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

[2] W. H. Specker, "A class of algorithms for ln x, exp x, sin x, cos x, $\tan^{-1}$ x, and cot^{-1} x", IEEE Trans. Electron. Comput., vol. EC-14, pp.85-86, Feb. 1965.

[3] T. C. Chen, "Automatic computation of exponentials, logarithms, ratios and square roots," IBM J. Res. Develop., vol. 16, pp.380-388, July 1972.

[4] M. D. Ercegovac, "Radix-16 evaluation of certain elementary functions", IEEE Trans. Comput. vol. C-22, pp. 561-566, June 1973.

[5] M. D. Ercegovac, "A general hardware-oriented method for evaluation of functions and computations in a digital computer", IEEE Trans. Comput. vol. C-26, pp. 667-680, July 1977.

[6] K. S. Trivedi and M. D. Ercegovac, "On-line algorithms for division and multiplication", IEEE Trans. Comput., Vol. C-26, pp. 681-687, July 1977.

[7] M. J. Irwin, "An arithmetic unit for on-line computation", Ph.D. dissertation, Report No. 873, Dept. Comput. Science, Univ. Illinois, Urbana-Champaign, May 1977.

[8] M. D. Ercegovac, "An on-line square rooting algorithm", Proc. 4th IEEE Symp. Comput. Arithmetic, pp.183-189, October 1978.

[9] A. L. Grnarov and M. D. Ercegovac, "An Algorithm for On-Line Normalization", UCLA Computer Science Department Quarterly, Vol.7, No.3, July 1979, pp.81-94.

[10] A.L. Grnarov and M.D. Ercegovac, "On-Line Multiplicative Normalization", UCLA Computer Science Department, Technical Report,1983.

[11] R.M. Owens, "Compound Algorithms for Digit Online Arithmetic", Department of Computer Science, The Pennsylvania State University, Technical Report, CS-81-1, January 1981.