# On-Line Algorithms for Division and Multiplication

KISHOR S. TRIVEDI AND MILOŠ D. ERCEGOVAC

***Abstract*—In this paper, on-line algorithms for division and multiplication are developed. It is assumed that the operands as well as the result flow through the arithmetic unit in a digit-by-digit, most significant digit first fashion. The use of a redundant digit set, at least for the digits of the result, is required.**

***Index Terms*—Computer arithmetic, division, multiplication, on-line algorithms, pipelining, radix, redundancy.**

## I. INTRODUCTION

IN THIS PAPER we consider problems of division and multiplication in a computational environment in which all basic arithmetic algorithms satisfy the "on-line" property. This implies that to generate the $j$th digit of the result, it is necessary and sufficient to have the operands available up to the $(j + \delta)$th digit, where the index difference $\delta$ is a small positive constant. It is necessary to accumulate $\delta$ initial digits of the operands in order to produce the first digit of the result. Subsequently, one digit of the result is produced upon receiving one digit of each of the operands. Thus $\delta$ is the on-line delay. Such algorithms can be used to speed up an arithmetic multiprocessor structure due to their potential to perform a sequence of operations in an overlapped fashion. Another possible application is in performing variable precision arithmetic. The on-line property implies a left-to-right digit-by-digit type of algorithm and consequently, a redundant representation, at least, of the results. For addition and subtraction, algorithms satisfying the on-line property can be easily specified. Multiplication requires a somewhat more elaborate approach and there are several possible ways of defining an on-line algorithm. However, the existence of an on-line division algorithm is not obvious and its analysis appears interesting.

Note that it is possible to define the on-line property with respect to a subset of the operands and/or the result. For example, conventional division has the on-line property with respect to the quotient. Similarly, conventional multiplication has the on-line property with respect to the multiplier. Several authors have extended the on-line property of multiplication to the product digits as well [1], [2]. It is also possible to define the on-line property with respect to a right-to-left mode of operation. Addition and subtraction operations always satisfy this property. Atrubin [3] has developed a right-to-left on-line algorithm for multiplication. In this paper, we will retain our earlier definition of the on-line property where all of the operands as well as the result digits flow through the arithmetic unit in a left-to-right digit-by-digit fashion.

Consider an $m$-digit radix $r$ number $N = \Sigma_{i=1}^{m} n_i r^{-i}$. In the conventional representation, each digit $n_i$ can take any value from the digit set $\{0,1,\cdots,r-1\}$. Such representations, which allow only $r$ values in the digit set, are nonredundant since there is a unique representation for each (representable) number. By contrast, number systems that allow more than $r$ values in the digit set are redundant. Redundant number representations are often useful in speeding up arithmetic operations [4], [5]. It is not difficult to see that the use of redundant number representation is mandatory for on-line algorithms. If we were to use a nonredundant number system, then even for simple operations like addition and subtraction, there is an on-line delay $\delta = m$ due to carry propagation. If we allow redundancy in the number representation, then it is possible to limit carry propagation to one digital position. A well-known example is the totally parallel addition in the signed digit representation [4]. This gives us an on-line algorithm for addition (and subtraction) with $\delta = 1$. Campeau [6] has developed an on-line algorithm for multiplication with $\delta = 1$. In this paper, we develop a new on-line algorithm for division where the value of $\delta$ depends upon the radix and other properties of the number system employed. In particular, we will present an on-line binary division algorithm with $\delta = 4$ (i.e., a 4-bit on-line delay). We will also show that an on-line decimal division algorithm can be obtained with $\delta = 4$ (i.e., a 4-digit on-line delay). We also develop a compatible on-line multiplication algorithm with $\delta = 1$.

After introductory remarks, an analysis of the on-line division problem is given in Section II. The radix-2 case with nonredundant operands is considered first and a feasible on-line algorithm is defined. Later, a generalization of this on-line algorithm for redundantly represented operands and higher radices is given. In Section III, a compatible on-line multiplication algorithm is considered.

## II. DIVISION

Let the radix $r$ representations of the positive dividend, divisor, and quotient be denoted by $N$, $D$, and $Q$, respectively, such that

$$N = \sum_{i=1}^{m} n_i r^{-i}, \quad D = \sum_{i=1}^{m} d_i r^{-i}, \quad Q = \sum_{i=1}^{m} q_i r^{-i},$$

Manuscript received January 20, 1976; revised January 5, 1977. This work was supported in part by the National Science Foundation under Grant US NSF DCR 73-07998.

K. S. Trivedi was with the University of Illinois at Urbana-Champaign, Urbana, IL 61801. He is now with the Department of Computer Science, Duke University, Durham, NC 27706.

M. D. Ercegovac was with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801. He is now with the Department of Computer Science, School of Engineering and Applied Science, University of California, Los Angeles, CA 90024.

and

$$Q = N/D \text{ to } m \text{ digit precision.}$$

If all the digits of the operands $N$ and $D$ are known in advance then the division can be carried out by the following well-known algorithm $D1$.

*Algorithm D1:*

Step 1 [Initialize]: $P_0 \leftarrow N; j \leftarrow 0;$

Step 2 [Selection]: $q_{j+1} \leftarrow$ Select $(rP_j, D)$;

Step 3 [Basic Recursion]: $P_{j+1} \leftarrow rP_j - q_{j+1}D;$

Step 4 [Test]: If $j < m - 1$ then $j \leftarrow j + 1$, and go to Step 2;

End $D1$;

In this algorithm, $P_j$ is known as the $j$th partial remainder and $rP_j$ is known as the $j$th shifted partial remainder. The selection procedure of Step 2 obtains the new quotient digit $q_{j+1}$ such that $P_{j+1}$ satisfies certain range restrictions [5]. This process requires the comparison of $rP_j$ against some constant multiples of $D$. The use of redundancy in the representation of each digital position of the quotient allows us to select $q_{j+1}$ based on the inspection of a limited number of leading digits of $rP_j$ and $D$ [5]. Furthermore, these methods can also be extended to the case when both the partial remainder and the divisor are in redundant form [7]. It is clear that without the use of redundancy it is not possible to avoid a set of *full* precision comparisons.

The present problem is that the digits of the dividend and the divisor are not known in advance but are available on-line, digit-by-digit, most significant digit first. It should be clear at the outset that in the absence of redundancy the problem cannot be solved.

Let us assume that the first digit $(q_1)$ of the quotient can be obtained after $\delta$ leading digits of the dividend and the divisor are known. Thereafter, one new digit of the quotient can be obtained upon receiving one digit each of the dividend and the divisor. We can then specify the following algorithm $D2$.

*Algorithm D2:*

Step 1 [Initialize]: $\hat{P}_0 \leftarrow \sum_{i=1}^{\delta} n_i r^{-i};$

$$\hat{D}_0 \leftarrow \sum_{i=1}^{\delta} d_i r^{-i}; j \leftarrow 0;$$

Step 2 [Selection]: $q_{j+1} \leftarrow$ Select $(r\hat{P}_j, \hat{D}_j)$

Step 3 [Basic Recursions]:

$$\hat{D}_{j+1} \leftarrow \hat{D}_j + d_{j+\delta+1} r^{-j-\delta-1}$$

$$\hat{P}_{j+1} \leftarrow r\hat{P}_j + n_{j+\delta+1} r^{-\delta} - q_{j+1}\hat{D}_{j+1} - \left(\sum_{i=1}^{j} q_i r^{-i}\right) \cdot d_{j+\delta+1} r^{-\delta}$$

Step 4 [Test]: If $j < m - 1$ then $j \leftarrow j + 1$ and go to Step 2;

End $D2$;

Note that we have assumed that the dividend and the divisor are padded with $\delta$ zero digits to the right. The basic recursion of algorithm $D2$ is slightly more complex than that in algorithm $D1$ due to the corrective action necessitated by the operand digits that arrive at the $j$th step.

The convergence of algorithm $D2$ can be established as follows. From the recursions of algorithm $D2$, the following expression for $\hat{P}_j$ is easily established by induction on $j$.

$$\hat{P}_j = r^j \left[\sum_{i=1}^{j+\delta} n_i r^{-i} - \left(\sum_{i=1}^{j} q_i r^{-i}\right)\left(\sum_{i=1}^{j+\delta} d_i r^{-i}\right)\right], \quad (2.1)$$

which implies that

$$\hat{P}_m = r^m[N - QD]$$

and

$$Q = N/D - \hat{P}_m r^{-m}/D. \quad (2.2)$$

Therefore, if we can establish a range restriction

$$|\hat{P}_j| \leq \alpha D \quad (2.3)$$

with $\frac{1}{2} \leq \alpha \leq 1$, then from (2.2) we deduce that $Q$ is the required quotient to $m$-digit precision [5]. Note that the value of $\alpha$ is determined by the properties of the number system employed [5]. For $j = 0$, (2.3) can be satisfied by appropriately preshifting the dividend. Assume that there is a selection procedure that generates the quotient digit $q_{j+1}$ so as to guarantee $|\hat{P}_{j+1}| \leq \alpha D$ given that $|\hat{P}_j| \leq \alpha D$. Then by induction, the range restriction (2.3) will hold for all values of $j$. Therefore, we will now derive such a selection procedure. Instead of proceeding directly, we will first establish a bound on $|P_j - \hat{P}_j|$ and then a selection procedure will be developed that guarantees $|P_j| \leq \alpha' D$. This in turn gives us a bound on $\hat{P}_j$. We will first discuss a simple case in which the radix is two and the divisor and the partial remainders are in nonredundant forms. Later, we will generalize to an arbitrary radix with all operands in redundant form.

### A. Binary Division with Nonredundant Operands

From the recursions of algorithm $D1$, the following equation can be derived by induction.

$$P_j = r^j \left[\sum_{i=1}^{m} n_i r^{-i} - \left(\sum_{i=1}^{j} q_i r^{-i}\right)\left(\sum_{i=1}^{m} d_i r^{-i}\right)\right]. \quad (2.4)$$

From (2.1) and (2.4) we have

$$P_j - \hat{P}_j = r^j \left[\sum_{i=j+\delta+1}^{m} n_i r^{-i} - \left(\sum_{i=1}^{j} q_i r^{-i}\right)\left(\sum_{i=j+\delta+1}^{m} d_i r^{-i}\right)\right]. \quad (2.5)$$

Recall that a binary ($r = 2$) nonredundant form of the dividend and the divisor implies that $n_i, d_i \in \{0,1\}$. Since the quotient is required to be in a redundant form, we let $q_i \in \{0,1,-1\}$. Therefore, from (2.5) we can get the bounds on $P_j - \hat{P}_j$,

$$P_j - \hat{P}_j \leq 2^j \left[\sum_{i=j+\delta+1}^{m} 2^{-i} + \left(\sum_{i=1}^{j} 2^{-i}\right)\left(\sum_{i=j+\delta+1}^{m} 2^{-i}\right)\right]$$

and

$$P_j - \hat{P}_j \geq -2^j \left[ \left( \sum_{i=1}^{j} 2^{-i} \right) \left( \sum_{i=j+\delta+1}^{m} 2^{-i} \right) \right],$$

from which we have

$$-2^{-\delta} \leq P_j - \hat{P}_j \leq 2 \cdot 2^{-\delta}. \tag{2.6}$$

Assuming that the divisor is the fractional part of a normalized floating-point number, we have $\frac{1}{2} \leq D < 1$. From the digit set $\{0,1,-1\}$ of quotient digits, we obtain the range restriction on $P_j$ [5],

$$-D \leq P_j \leq D. \tag{2.7}$$

From (2.6) and (2.7) we get opaque

$$-D + 2^{-\delta} \leq \hat{P}_j \leq D - 2 \cdot 2^{-\delta}. \tag{2.8}$$

From (2.7), we get

$$-2D \leq 2P_j \leq 2D.$$

For a given value of $q_{j+1}$, there is a range of $2P_j$ values such that the selection of this value of $q_{j+1}$ in step 2 of algorithm $D1$ will force the subsequent value of $P_{j+1}$ to satisfy the range restriction (2.7). These ranges can be specified as follows:

If $0 \leq 2P_j \leq 2D$, then $q_{j+1} = 1$,

if $-D \leq 2P_j \leq D$, then $q_{j+1} = 0$, and

if $-2D \leq 2P_j \leq 0$, then $q_{j+1} = -1$.

Corresponding ranges of $2\hat{P}_j$ values can be obtained by using the inequality (2.6).

If $2 \cdot 2^{-\delta} \leq 2\hat{P}_j \leq 2D - 4 \cdot 2^{-\delta}$, then $q_{j+1} = 1$,

if $-D + 2 \cdot 2^{-\delta} \leq 2\hat{P}_j \leq D - 4 \cdot 2^{-\delta}$, then $q_{j+1} = 0$, and

if $-2D + 2 \cdot 2^{-\delta} \leq 2\hat{P}_j \leq 0 - 4 \cdot 2^{-\delta}$, then $q_{j+1} = -1$. (2.9)

It is not possible to use these rules for quotient digit selection in algorithm $D2$ since the knowledge of all the bits of the divisor $D$ is implied above. However, the redundancy in the representation of the quotient allows us to make the selection, independent of the value of $D$.

Each inequality of (2.9) determines a wedge-shaped area on the $2\hat{P}_j$ versus $D$ plane as shown in Fig. 1. This area together with its associated value of $q_{j+1} = i$ will be called the $i$-selection region. Let the intersection of the $i$-selection region with $(i+1)$-selection region be called $(i,i+1)$-selection overlap region. The existence of nonnull selection overlap regions is due to the redundancy in quotient digits. The $(-1,0)$ and $(0,1)$-selection overlap regions are respectively given by

$$-D + 2 \cdot 2^{-\delta} \leq 2\hat{P}_j \leq -4 \cdot 2^{-\delta}$$

and

$$2 \cdot 2^{-\delta} \leq 2\hat{P}_j \leq D - 4 \cdot 2^{-\delta}.$$

Now we require that there be nonzero-selection overlap for all values of $\frac{1}{2} \leq D < 1$. Since the worst case occurs when $D = \frac{1}{2}$, we get the condition

$$2 \cdot 2^{-\delta} < \tfrac{1}{2} - 4 \cdot 2^{-\delta}.$$

Since $\delta$ is required to be an integer, we get $\delta \geq 4$. Next we select two constants $C_1$ and $C_2$ such that the line $2\hat{P}_j = C_1$ is entirely within the $(0,1)$-selection overlap region. A similar condition holds for $C_2$ with respect to the $(-1,0)$-selection overlap region. These two lines can be used as the selection lines to obtain the following selection procedure.

If $2\hat{P}_j > C_1$, then $q_{j+1} = 1$,

if $2\hat{P}_j < C_2$, then $q_{j+1} = -1$

otherwise $q_{j+1} = 0$. (2.10)

The development of the selection rules (2.10) was based on range restriction (2.8). Therefore, using (2.2), we see that algorithm $D2$ will converge. We choose the following values of the constants $C_1$ and $C_2$ from Fig. 1: $C_1 = \frac{1}{4}$ and $C_2 = -\frac{1}{4}$.

If we substitute the selection rules (2.10) in algorithm $D2$ together with $\delta = 4$, $r = 2$, $C_1 = \frac{1}{4}$, and $C_2 = -\frac{1}{4}$, then we have a binary on-line division algorithm. In a similar fashion, we can also obtain on-line division algorithms for higher radices.

An example of binary on-line division now follows.

Let $m = 24$,

$N = 0.10100011011010111010010101$, and

$D = 0.11110110010110001111001 1$.

| $\hat{P}_j$ | $q_{j+1}$ |
|---|---|
| 0.10100000000000000000000 | 1 |
| 0.01010000000000000000000 | 1 |
| 1.10100100000000000000000 | −1 |
| 0.01000010000000000000000 | 1 |
| 1.10011110000000000000000 | −1 |
| 0.00110010000000000000000 | 1 |
| 1.01110011010000000000000 | −1 |
| 1.11101100110000000000000 | 0 |
| 1.11001110111000000000000 | −1 |
| 0.10011001011110000000000 | 1 |
| 0.00111100100110000000000 | 1 |
| 1.10010010110110000000000 | −1 |
| 0.00101100000010000000000 | 1 |
| 1.01100111000110101000000 | −1 |
| 1.10111001111100000100000 | −1 |
| 0.01011111100111000010000 | 1 |
| 1.11001110010000100011000 | −1 |
| 0.10010010110111010101000 | 1 |
| 0.00111111011000011011000 | 1 |
| 1.01111101110011010100001 | −1 |
| 1.11110111010101100100100 | 0 |
| 1.11101110101011001001001 | 0 |
| 1.11011101010110010010010 | −1 |
| 0.10110001000010110011101 | 1 |

We note that negative values of $\hat{P}_j$ are represented in two's complement notation.

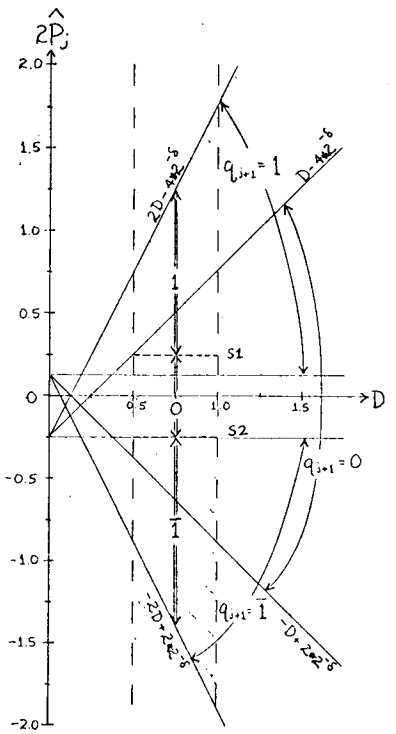


Fig. 1. The selection regions for binary on-line division ($\delta = 4$).

### B. Division with Redundant Operands

In Section A, we required that the operands and the partial remainder be in nonredundant form. As a result, we cannot use carry-free addition (or subtraction) in Step 3 of algorithm $D2$. In this section we generalize to the case where the dividend, the divisor, and the partial remainder are in a redundant form. As a result, carry-free addition in Step 3 of algorithm $D2$ can be used. For a discussion of carry-free addition see [4].

We assume that the digits of the dividend, the divisor, and the quotient are all chosen from the symmetric redundant digit set

$$D_\rho = \{-\rho, \cdots -1,0,1, \cdots \rho\} \text{ and } \frac{r}{2} \le \rho \le r - 1.$$

$K = \rho/(r-1)$ is known as the degree of redundancy. It can be shown that the appropriate range restriction on $P_j$ is given by [5], [7]

$$-KD \le P_j \le KD. \tag{2.11}$$

From (2.5) and the fact that $|n_i|$, $|d_i|$,$|q_i| \le \rho$, we have

$$|P_j - \hat{P}_j| \le r^j \left[ \rho \cdot \frac{r^{-j-\delta-1} - r^{-m-1}}{1 - r^{-1}} + \rho \cdot \left( \frac{r^{-1} - r^{-j-1}}{1 - r^{-1}} \right) \cdot \rho \cdot \left( \frac{r^{-j-\delta-1} - r^{-m-1}}{1 - r^{-1}} \right) \right]$$

$$= (K + K^2)r^{-\delta} - (K + K^2)r^{-m+j} - K^2 r^{-j-\delta} + K^2 r^{-m},$$

which implies that

$$-(K + K^2)r^{-\delta} \le P_j - \hat{P}_j \le (K + K^2)r^{-\delta}. \tag{2.12}$$

From (2.11) and (2.12), we get the range restriction on $\hat{P}_j$:

$$-KD + (K + K^2)r^{-\delta} \le \hat{P}_j \le KD - (K + K^2)r^{-\delta}. \tag{2.13}$$

For a given value of $-\rho \le i \le \rho$, there exists a range of values of $rP_j$ such that the selection $q_{j+1} = i$ guarantees the range restriction (2.11) on $P_{j+1}$. The range of $rP_j$ values for $q_{j+1} = i$ is given by

$$(-K + i)D \le rP_j \le (K + i)D. \tag{2.14}$$

The corresponding $i$-selection region for $r\hat{P}_j$ is obtained using (2.12) and (2.14) as

$$(-K + i)D + (K + K^2)r^{-\delta+1} \le r\hat{P}_j \le (K + i)D - (K + K^2)r^{-\delta+1}. \tag{2.15}$$

In Fig. 2, we have shown these selection regions on a graph of $r\hat{P}_j$ versus $D$. Such a graph is known as a $P$-$D$ plot in the literature [7]. But for obvious reasons, we call it a $\hat{P}$-$D$ plot. The use of inequality (2.15) in a selection procedure will imply a set of full precision comparisons between $r\hat{P}_j$ with several multiples of $D$. However, only $\delta$ digits of $D$ are known initially. The existence of ranges of $r\hat{P}_j$ values, where more than one value of $q_{j+1}$ is possible, allows us to make a selection with the limited information on $D$ [6], [7]. Recall that the redundancy in number representation is the cause of such selection overlap regions. Quotient digit selection can now be made from the estimate $R_j$ of $\hat{P}_j$ and the estimate $\hat{D}$ of $D$. Assume that $\delta$ most significant digits of $\hat{P}_j$ are used as its estimate $R_j$, and $\beta$ most significant digits of the divisor $D$ are used as its estimate $\hat{D}$. Since only $\delta$ digits of the divisor are available initially, $\beta \le \delta$ must hold. Let $\Delta p$ denote the upper bound on the error made in estimating $r\hat{P}_j$ by $rR_j$, i.e., $|r\hat{P}_j - rR_j| \le \Delta p$. Similarly, let $|D - \hat{D}| \le \Delta d$. The rest of the analysis for determining the appropriate selection rules and determining the values of $\delta$ and $\beta$ is very similar to the analysis in Atkins' Ph.D. dissertation [7]. The difference is that his $P$-$D$ plot is that of $rP_j$ versus $D$, whereas ours is that of $r\hat{P}_j$ versus $D$. The main point is that the errors of estimation ($\Delta p$ and $\Delta d$) should be small compared to the latitude of choice allowed by the selection overlap regions. It may be deduced from (2.15) that the smallest selection overlap region occurs near the minimal value, $D_{\min}$, of the divisor [7]. Therefore, we require that the rectangle of height $2\Delta p$, width $2\Delta d$, and with its center at $\hat{D}_{\min}$, should be completely contained in the $(i-1,i)$-selection overlap region (see Fig. 2). The above condition is equivalent to the condition [7] opaque

$$\Delta p + \tfrac{1}{2}(2i - 1)\Delta d + (K + K^2)r^{-\delta+1} \le \tfrac{1}{2}(2K - 1)\hat{D}_{\min}.$$

Note that the worst case occurs when $i = \rho$, therefore, we get the condition

$$\Delta p + (\rho - \tfrac{1}{2})\Delta d + (K + K^2)r^{-\delta+1} \le (K - \tfrac{1}{2})\hat{D}_{\min}. \quad (2.16)$$

To obtain a bound on $\Delta d$, we note that it consists of the trailing digits of $D$ starting from the $(\beta + 1)$st digit. The largest value of each of these digits is $\rho$.

Therefore,

$$\Delta d \le \sum_{i=\beta+1}^{m} \rho r^{-i} \le \frac{\rho}{r-1} r^{-\beta} = Kr^{-\beta}.$$

Similarly,

$$\Delta p \le \frac{\rho}{r-1} r^{-\delta+1} = Kr^{-\delta+1}.$$

Therefore, the inequality (2.16) reduces to

$$Kr^{-\delta+1} + (\rho - \tfrac{1}{2})Kr^{-\beta} + (K + K^2)r^{-\delta+1} \le (K - \tfrac{1}{2})\hat{D}_{\min}$$

$$(2K + K^2)r^{-\delta+1} + K(\rho - \tfrac{1}{2})r^{-\beta} \le (K - \tfrac{1}{2})\hat{D}_{\min}. \quad (2.17)$$

If we assume that the divisor $D$ is standardized [7], i.e., $d_1 \neq 0$, then since $|d_i| \le \rho$, we have

$$\hat{D}_{\min} = \frac{1}{r} - \sum_{i=2}^{\beta} \rho r^{-i}$$

or

$$\hat{D}_{\min} = \frac{1}{r} - \frac{\rho}{r^2}\frac{1 - r^{-\beta+1}}{1 - r^{-1}} = \frac{1}{r} - \frac{\rho}{r}\frac{1 - r^{-\beta+1}}{r - 1} = \frac{1}{r}(1 - K(1 - r^{-\beta+1})).$$

Therefore, the condition (2.17) reduces to

$$(2K + K^2)r^{-\delta+1} + K(\rho - \tfrac{1}{2})r^{-\beta} \le \frac{1}{r}(K - \tfrac{1}{2})[1 - K(1 - r^{-\beta+1})]$$

or

$$(2K + K^2)r^{-\delta+2} + K(\rho - K)r^{-\beta+1} \le (1 - K)(K - \tfrac{1}{2}). \quad (2.18)$$

Once $r$ and $K$ are decided, we can determine $\beta$ and $\delta$ from this condition. From condition (2.18) we infer that number systems with $K = \frac{1}{2}$ (i.e., a nonredundant system) and $K = 1$ (i.e., a maximally redundant system) are not permissible. As an example, let $r = 10$, $\rho = 6$ and $K = \frac{2}{3}$. Then from (2.18), we have

$$(4/3 + 4/9)10^{-\delta+2} + 2/3(16/3)10^{-\beta+1} \le \frac{1}{3}\cdot\frac{1}{6},$$

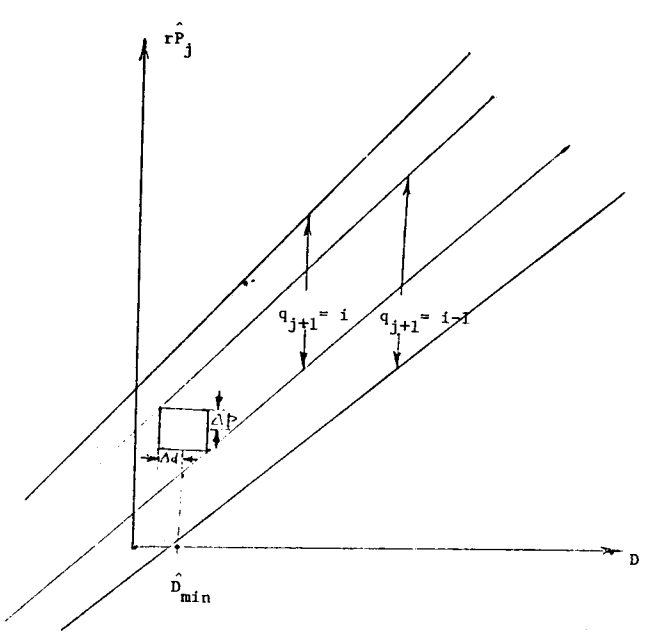


Fig. 2. $\hat{P}$-$D$ plot for higher radix on-line division.

or

$$(3200)10^{-\delta} + (640)10^{-\beta} \le 1.$$

If we let $\delta = \beta + 1$, then

$$(320 + 640)10^{-\beta} \le 1$$

or

$$\beta \ge 3, \delta \ge 4.$$

Thus, we can define a decimal on-line division algorithm with a four digit on-line delay. The selection rules may be obtained from the $\hat{P}$-$D$ plot following the procedure in [7], but the details are omitted here.

## III. Multiplication

An on-line algorithm for multiplication, compatible with the previously considered on-line division algorithm, can be conveniently derived following the well-known technique of incremental multiplication, as used in the digital differential analyzers [6], [8], combined with the use of redundant number of systems.

Let

$$X = \sum_{i=1}^{m} x_i \cdot r^{-i}$$

$$Y = \sum_{i=1}^{m} y_i \cdot r^{-i} \quad (3.1)$$

be the radix $r$ representations of the positive multiplicand and the multiplier, respectively. Define

$$X_j = \sum_{i=1}^{j} x_i \cdot r^{-i} = X_{j-1} + x_j \cdot r^{-j}$$

$$Y_j = \sum_{i=1}^{j} y_i \cdot r^{-i} = Y_{j-1} + y_j \cdot r^{-j} \quad (3.2)$$

to be the $j$ digit representations of the operands $X$ and $Y$, available at the $j$th step by definition of an on-line algorithm. The corresponding partial product is, then,

$$X_j \cdot Y_j = X_{j-1} \cdot Y_{j-1} + (X_j \cdot y_j + Y_{j-1} \cdot x_j) r^{-j}. \tag{3.3}$$

Let $P_j$ be the scaled partial product, i.e.,

$$P_j = X_j \cdot Y_j \cdot r^j \tag{3.4}$$

so that the recursion of the multiplication algorithm can be expressed as follows:

$$P_j = rP_{j-1} + X_j \cdot y_j + Y_{j-1} \cdot x_j. \tag{3.5}$$

Defining $P_0 = 0$, the scaled product $P_m = X \cdot Y \cdot r^m$ can be generated in $m$ steps (3.5). If a nonredundant number system is used in representing the partial products, the digits of the desired product appear in a right-to-left fashion, as determined by the conventional carry propagation requirements. If, however, a redundant number representation is adopted, the desired left-to-right generation of the product digits, as required by the on-line property, can be easily provided. Moreover, the redundancy in number representation can make the time required to perform the recursive step independent of the operand precision (i.e., a carry-free addition is possible).

We will use a symmetric redundant digit set

$$D_\rho = \{-\rho, -(\rho - 1), \cdots, -1, 0, 1, \cdots, \rho - 1, \rho\} \tag{3.6}$$

where

$$\frac{r}{2} \leq \rho \leq r - 1.$$

Following the general computational method described in [9], the basic recursion (3.5) of the multiplication algorithm is redefined in the following way:

$$w_j = r(w_{j-1} - d_{j-1}) + X_j \cdot y_j + Y_{j-1} \cdot x_j \tag{3.7}$$

where the digits $d_j \in D_\rho$, can be determined by the following selection function:

$$d_j = S(w_j) = \text{sign } w_j \cdot \lfloor |w_j| + 1/2 \rfloor \tag{3.8}$$

which clearly corresponds to a rounding procedure.

Then, from (3.5) and (3.7), the following relation can be obtained by induction

$$w_j = P_j - \sum_{i=1}^{j-1} d_i \cdot r^{(j-i)}. \tag{3.9}$$

Substituting $j = m$ in (3.9) and rearranging, we have

$$P_m = X \cdot Y \cdot r^m = r^m \sum_{i=1}^{m-1} d_i r^{-i} + w_m \tag{3.10}$$

or

$$X \cdot Y = \sum_{i=1}^{m} d_i r^{-i} + (w_m - d_m) r^{-m}.$$

By definition of the selection function $S(w_j)$, $|w_m - d_m| \leq \frac{1}{2}$, so that $\Sigma_{i=1}^{m} d_i r^{-i}$ is indeed the redundant representation of the most significant half of the product $X \cdot Y$. Thus, the algorithm will converge.

We have assumed above that the selection function (3.8) will produce the digit $d_i$ such that $|d_i| \leq \rho$. This condition will be satisfied provided

$$|w_j| < \rho + \tfrac{1}{2} \tag{3.11}$$

holds for $j = 1,2,\cdots,m$. We will now obtain an upper bound $M$, on the values of the operands $X$ and $Y$ such that (3.11) holds. Let $|X|, |Y| \leq M$. Noting that $|w_{j-1} - d_{j-1}| \leq \frac{1}{2}$ and $|y_j|, |x_j| \leq \rho$, from (3.7) we get

$$|w_j| \leq \frac{r}{2} + 2M\rho.$$

Now using (3.11) and rearranging we get

$$M < \frac{1}{2} - \frac{r-1}{4\rho}.$$

Thus, for a minimally redundant system, defined by $\rho = r/2$, the required operand bound is

$$|X|, |Y| < \frac{1}{2r}$$

and for a maximally redundant system, defined by $\rho = r - 1$, the required operand bound is

$$|X|, |Y| < \frac{1}{4}.$$

On closer inspection, we are often able to improve upon these bounds. For example, $|X|, |Y| \leq \frac{1}{2}$ suffices for $r = 2$.

As discussed in detail elsewhere [9], it is a simple matter to make the time required for the computation of $w_j$ independent of the precision of the corresponding operands. Thus, by allowing

$$\tfrac{1}{2} < |w_j - d_j| < 1$$

a carry propagation free addition can be utilized in (3.7).

The following example illustrates the on-line multiplication algorithm for $r = 2$.

$X = 0.01101001$
$Y = 0.01110011$

| $j$ | $X_j y_j + Y_{j-1} x_j$ | $w_j$ | $d_j$ | $2(w_j - d_j)$ |
|---|---|---|---|---|
| 1 | 0.0 | 0.0 | 0 | 0.0 |
| 2 | 0.01 | 0.01 | 0 | 0.1 |
| 3 | 0.101 | 1.001 | 1 | 0.01 |
| 4 | 0.0110 | 0.1010 | 1 | −0.11 |
| 5 | 0.0111 | −0.0101 | 0 | −0.101 |
| 6 | 0.0 | −0.101 | −1 | 0.11 |
| 7 | 0.0110100 | 1.0010100 | 1 | 0.010100 |
| 8 | 0.11011011 | 1.00101011 | 1 | 0.0101011 |

$$\sum_{j=1}^{8} d_j 2^{-j} + 2^{-8}(w_8 - d_8) = 0.0010111100101011$$

$$= X \cdot Y. \quad \square$$

## IV. Concluding Remarks

Two compatible algorithms for on-line division and multiplication, based on the redundant number systems, have been presented. These on-line algorithms provide an effective way of speeding up the execution of sequences of the basic arithmetic operations by minimizing the delay between successive operations in an overlapped mode of operation. In real-time applications, where the inputs are serially generated by an analog-to-digital conversion process beginning with the most significant digits, the on-line algorithms can be used to increase the overall speed by overlapping the computation with the conversion. The described algorithms can be seen to have rather simple implementation requirements and properties which are compatible with the desirable modularity in implementation and variable precision operations.

At the present time, computer systems employing a pipelined arithmetic unit for floating-point computation consider fraction arithmetic as a single stage of the pipeline [10], [11]. The on-line algorithms presented in this paper will allow further decomposition of this stage so that a digitally pipelined arithmetic unit will be possible.

## References


[1] A. Avizienis, "On a flexible implementation of digital computer arithmetic," in *Proc. IFIP*, pp. 664–668, 1962.

[2] M. J. Pisterzi, "A limited connection arithmetic unit," Ph.D. dissertation, Dep. Comput. Sci., Univ. of Illinois, Urbana-Champaign, IL, June 1970.

[3] A. J. Atrubin, "A one-dimensional real-time iterative multiplier," *IEEE Trans. Comput.*, vol. C-14, pp. 394–399, 1965.

[4] A. Avizienis, "Signed digit number representation for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 389–400, 1961.

[5] J. E. Robertson, "A new class of digital division methods," *IRE Trans. Electron. Comput.*, vol. EC-7, pp. 218–222, Sept. 1958.

[6] J. O. Campeau, "Communication and sequential problems in the parallel processor," in *Parallel Processor Systems, Technologies and Applications*. New York: Spartan, 1970.

[7] D. E. Atkins, "A study of methods for selection of quotient digits during digital division," Ph.D. dissertation, Dep. of Comput. Sci., Univ. of Illinois, Urbana-Champaign, IL, June 1970.

[8] E. L. Braum, *Digital Computer Design—Logic, Circuitry, and Synthesis*. New York: Academic Press, 1963.

[9] M. D. Ercegovac, "A general method for evaluation of functions and computations in a digital computer," Ph.D. dissertation, Dep. Comput. Sci., University of Illinois, Urbana-Champaign, IL, July 1975.

[10] C. Stephenson, "Case study of the pipelined arithmetic unit for the TI advanced scientific computer," in *IEEE Proc. 3rd Symp. on Computer Arithmetic*, Dallas, TX, Nov. 1975.

[11] S. F. Anderson *et al.*, "The system/360 model 91 floating-point execution unit," *IBM Syst. J.*, vol. 11, p. 34, 1967.



**Kishor S. Trivedi** was born in Bhavnagar, India, on August 20, 1946. He received the Ph.D. degree in computer science from the University of Illinois, Urbana-Champaign, IL, in 1974.

From 1974–1975 he was a Visiting Research Associate at the University of Illinois. In the summer of 1976, he was a Visiting Scientist at the Institute for Computer Applications to Science and Engineering, NASA Langley Research Center, Hampton, VA. He is currently an Assistant Professor of Computer Science at Duke University, Durham, NC. His current interests include computer architecture and operating systems.

**Miloš D. Ercegovac,** for a photograph and biography please see this issue, p. 680.