On Classes of Positive, Negative, and Imaginary Radix Number Systems

ISRAEL KOREN, MEMBER, IEEE, AND YORAM MALINIAK

***Abstract*—A unified approach to a broad class of finite number representation systems is proposed. This class contains all positive and negative radix systems and other well-known number systems. In addition, it can be extended to include imaginary radix number systems. The proposed approach enables us to develop a single set of algorithms for arithmetic operations.**

***Index Terms*—Arithmetic operations, finite number representation systems, imaginary radix, negative radix, positive radix, radix-complement.**

Manuscript received July 23, 1979; revised May 21, 1980.

I. Koren was with the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA. He is now with the Departments of Electrical Engineering and Computer Science, Technion, Haifa, Israel.

Y. Maliniak is with the Department of Electrical Engineering, University of California, Santa Barbara, CA 93106.

I. Introduction

THE positive-radix and radix-complement number systems are the most commonly used finite number representation systems and numerous algorithms for arithmetic operations in these systems have been developed and implemented in digital computers. Other fixed-radix systems have received a great amount of attention in recent years [1]-[6], [11], [12]. Various algorithms for arithmetic operations in these systems have been developed and new applications facilitated by their use have been presented, e.g., digital filters [7].

A unified approach to these finite number systems is clearly in order [1]-[3]. Such an approach is proposed here and it is shown that the above mentioned systems are members of a broad class of finite number representation systems. We first

0018-9340/81/0500-0312$00.75

define this class and state its basic properties. Next, unified algorithms for arithmetic operations are presented. Finally, a new imaginary radix system based on the 2's complement system is analyzed.

II. Definitions and Basic Properties

Consider a class of n-digit, fixed radix number systems in which every number system is characterized by a radix β, which is a positive integer and by a vector Λ of length n, $\Lambda = (\lambda_{n-1}, \lambda_{n-2}, \cdots, \lambda_0)$, where $\lambda_i \in \{1, -1\}$. Such a number system with the standard digit set $\{0, 1, \cdots, \beta - 1\}$ is characterized by the triple $\langle n, \beta, \Lambda \rangle$. The algebraic value X of an n-tuple $(x_{n-1}, x_{n-2}, \cdots, x_0)$ in the system $\langle n, \beta, \Lambda \rangle$ is defined by

$$X = \sum_{i=0}^{n-1} \lambda_i x_i \beta^i. \tag{1}$$

For a given radix β we have in this class 2^n distinct number systems. Among them is the positive radix number system, i.e., $\lambda_i = +1$ for every i, the negative radix number system, i.e., $\lambda_i = (-1)^i$ for every i, and the radix-complement number system with x_{n-1} as a sign digit (i.e., $x_{n-1} \in \{0, 1\}$) and the characterizing vector $\Lambda = (-1, 1, 1, \cdots, 1)$.

All the number systems in this class are clearly nonredundant and complete [8], i.e., in a given number system $\mathcal{A}$ defined by $\langle n, \beta, \Lambda \rangle$ we have a unique representation for any number X within the range of $\mathcal{A}$. The largest representable integer in $\mathcal{A}$ is the positive number P whose representation is

$$p_i = \begin{cases} \beta - 1 & \text{if } \lambda_i = +1 \\ 0 & \text{otherwise} \end{cases}; \quad i = 0, 1, \cdots, n-1. \tag{2}$$

Its value is given by

$$\begin{aligned} P &= \frac{1}{2} \sum_{i=0}^{n-1} (\lambda_i + 1)(\beta - 1)\beta^i \\ &= \frac{1}{2} \left[\sum_{i=0}^{n-1} \lambda_i (\beta - 1)\beta^i + \sum_{i=0}^{n-1} (\beta - 1)\beta^i \right] \\ &= \frac{1}{2} [R + (\beta^n - 1)] \end{aligned} \tag{3}$$

where R is the algebraic value of the number $(\beta - 1, \beta - 1, \cdots, \beta - 1)$ in $\mathcal{A}$. Similarly, the smallest integer is the negative number N whose representation is

$$n_i = \begin{cases} \beta - 1 & \text{if } \lambda_i = -1 \\ 0 & \text{otherwise;} \end{cases} \quad i = 0, 1, \cdots, n-1$$

and the corresponding value is

$$N = \frac{1}{2} \sum_{i=0}^{n-1} (\lambda_i - 1)(\beta - 1)\beta^i = \frac{1}{2} [R - (\beta^n - 1)]. \tag{4}$$

The number of integers in the range $N \leqq X \leqq P$ is $P - N + 1 = \beta^n$, and the range of the system $\mathcal{A}$ is, in general, asymmetric. A measure of the asymmetry can be the difference $P - |N|$ which equals $P + N = R$.

Example: The negative radix system $\langle n, \beta, \Lambda = (\cdots, -1, +1, -1, +1) \rangle$ is asymmetric and for n even there are β times as many negative numbers as positive ones. If we prefer to have more positive numbers, we can use instead the system $\langle n, \beta, \Lambda = (+1, -1, \cdots, +1, -1) \rangle$. Two binary systems are nearly symmetric. One is the 2's complement system $\langle n, \beta = 2, \Lambda = (-1, 1, \cdots, 1) \rangle$ for which we have $P - |N| = R = -1$. The other is the system $\langle n, \beta = 2, \Lambda = (+1, -1, \cdots, -1) \rangle$ for which we have $P - |N| = R = +1$.

The complement of a number in a system $\mathcal{A}$ can be defined as follows. Let $\bar{x}_i \triangleq (\beta - 1) - x_i$; the complement $\bar{X}$ of a number X is

$$\bar{X} \triangleq \sum_{i=0}^{n-1} \bar{x}_i \lambda_i \beta^i = R - X. \tag{5}$$

Hence,

$$-X = \bar{X} - R = \bar{X} + (-R) \tag{6}$$

i.e., the additive inverse of X may be formed by adding the additive inverse of R to the complement of X. (Note that no carry propagation is involved while generating $\bar{X}$.) For example, in the 2's complement system $R = -1$ and $-X = \bar{X} + 1$. In the negabinary system $-R = (\cdots, 0, 1, 0, 1)$ and the following algorithm for sign-flipping (or polarization [4], [5]) results in $(\cdots \bar{x}_3 \bar{x}_2 \bar{x}_1 \bar{x}_0) + (\cdots 0\ 1\ 0\ 1)$, e.g., the additive inverse of $a = 110011001011$ [1] is

$$(001100110100) + (010101010101) = (010001011001).$$

In the negative radix system with $\beta > 2$ the additive inverse of R is $-R = (\cdots, 2, 2, 2, 2, 1)$, e.g., the additive inverse of $a = 08019$ in the nega-decimal system [4] is $91980 + 22221 = 12001$.

The additive inverse may be employed in subtraction either by using the equation

$$Y - X = Y + \bar{X} + (-R) \tag{7}$$

or by using the following easily verified equation

$$Y - X = \overline{(\bar{Y} + X)}. \tag{8}$$

Here, only one addition with carry propagation is needed. However, a more efficient algorithm for subtraction is presented in the next section.

III. Addition and Subtraction

Different algorithms for addition and subtraction in positive radix and negative radix systems have been developed [4]–[6]. We show that a unified treatment of all number systems in the previously defined class is possible and a single addition–subtraction algorithm can be developed.

Let X and Y be two numbers to be added or subtracted. Since a mixture of positive and negative digit weights is used in our system, we have to use borrows in addition and carries in subtraction. Hence, there is no need to distinguish between addition and subtraction and a single set of rules is developed for $S = X \pm Y$. These rules specify the values of the sum digit s_i, the carry c_{i+1}, and the borrow b_{i+1} from the values of x_i, y_i, the carry in c_i, and the borrow in b_i. The basic equation that should be satisfied is

$$\lambda_i (x_i \pm y_i) + c_i - b_i = \lambda_i s_i + \beta c_{i+1} - \beta b_{i+1}; \quad i = 0, 1, \cdots, n-1 \tag{9}$$

the multiplier (quotient) is recoded into a minimal signed-digit form [10] in order to minimize the number of add/subtract operations. The canonical recoding procedure introduced by Reitwiesner [10] is restricted to binary numbers in the positive radix system. In the following we extend this procedure to numbers in any $\langle n, \beta = 2, \Lambda \rangle$ number system. We first show the existing relationship between the nonredundant system $\langle n, \beta = 2, \Lambda \rangle$ and the redundant signed-digit system. A number $(x_{n-1}, \cdots, x_0)$ in the $\langle n, \beta = 2, \Lambda \rangle$ system can be readily converted to a signed-digit (S-D) number $(y_{n-1}, \cdots, y_0)$ using the equation

$$y_i = x_i \cdot \lambda_i; \qquad i = 0, 1, \cdots, n-1. \tag{18}$$

The digit set for x_i is $\{0, 1\}$, while the digit set for y_i is $\{\bar{1}, 0, 1\}$ ($\bar{1}$ represents -1). An S-D binary number $(y_{n-1}, \cdots, y_0)$ may be converted into a number $(x_{n-1}, \cdots, x_0)$ in a $\langle n, \beta = 2, \Lambda \rangle$ system using the equations

$$x_i = |y_i|$$

$$\lambda_i = \begin{cases} -1 & \text{if } y_i = \bar{1} \\ 1 & \text{otherwise} \end{cases}; \quad i = 0, 1, \cdots, n-1. \tag{19}$$

Let $(x_{n-1}, \cdots, x_0)$ be the representation of a given binary number X in the $\langle n, \beta = 2, \Lambda \rangle$ system. Let $(y_{n-1}, \cdots, y_0)$ be its equivalent S-D representation defined by (18), and let $(d_{n-1}, \cdots, d_0)$ be a minimal S-D representation of X (i.e., $\sum_{i=0}^{n-1} |d_i|$ is minimal). The canonical recoding procedure generates the digits d_i serially, beginning with the least significant digit d_0. At the ith step we inspect the digits y_i and y_{i+1} and a carry digit c_i, and we generate the digit d_i and a carry digit c_{i+1}. The rules of this procedure are given in Table I. Note that the digit set for the carry is $\{\bar{1}, 0, 1\}$, while in the restricted algorithm [10] it is $\{0, 1\}$.

Clearly, this algorithm can also be used to generate the canonical form of any S-D binary number, e.g., the number $(1, 1, \bar{1}, \bar{1}, \bar{1}, \bar{1}, 1)$ is converted using Table I to $(1, 0, 0, 0, 1, 0, \bar{1})$, which is a minimal S-D representation of 67.

The minimal recoding algorithm can be generalized to radix β number systems. The extension is straightforward and is therefore omitted.

We conclude this section with an example of employing the S.R.T. method to divide binary fractions in the number system with $\Lambda = (1, 1, -1, -1, 1, 1, -1, -1)$. The dividend $X = 30/256$ is represented by 0.01100010 and the divisor $D = 6/16$ is represented by 0.1010. The division steps are

$R_0 = X = 0.01100010$

$R_1 = 2R_0 = 0.01011100$ normalize, $q_1 = 0$

remainder 0.10011000

-0.1010

$R_2 = 2R_1 - D = 0.01111000$ $q_2 = 1$

$R_3 = 2R_2 = 0.01010000$ normalize, $q_3 = 0$

remainder 0.10100000

-0.1010

$R_4 = 2R_3 - D = 0.00000000$ $q_4 = 1$.

The resulting quotient in S-D notation is 0.0101 which represents $Q = 5/16$. To obtain the representation 0.1011 of Q in the above number system a conversion algorithm is needed. Fortunately, the known conversion methods between positive and negative radix systems can easily be extended to any two number systems with the same radix β, but different characterizing vectors. Such an extension of the method introduced by Yuen [6] is presented in [1].

TABLE I
CANONICAL RECODING OF S-D NUMBERS

y_{i+1}	y_i	c_i	d_i	c_{i+1}	y_{i+1}	y_i	c_i	d_i	c_{i+1}
0	0	0	0	0	0	0	1	1	0
1	0	0	0	0	1	0	1	$\bar{1}$	1
$\bar{1}$	0	0	0	0	$\bar{1}$	0	1	$\bar{1}$	1
0	1	$\bar{1}$	0	0	0	1	0	1	0
1	1	$\bar{1}$	0	0	1	1	0	$\bar{1}$	1
$\bar{1}$	1	$\bar{1}$	0	0	$\bar{1}$	1	0	$\bar{1}$	1
0	$\bar{1}$	1	0	0	0	0	$\bar{1}$	$\bar{1}$	0
1	$\bar{1}$	1	0	0	1	0	$\bar{1}$	1	$\bar{1}$
$\bar{1}$	$\bar{1}$	1	0	0	$\bar{1}$	0	$\bar{1}$	1	$\bar{1}$
0	1	1	0	1	0	$\bar{1}$	0	$\bar{1}$	0
1	1	1	0	1	1	$\bar{1}$	0	1	$\bar{1}$
$\bar{1}$	1	1	0	1	$\bar{1}$	$\bar{1}$	0	1	$\bar{1}$
0	$\bar{1}$	$\bar{1}$	0	$\bar{1}$					
1	$\bar{1}$	$\bar{1}$	0	$\bar{1}$					
1	$\bar{1}$	$\bar{1}$	0	$\bar{1}$					

choose c_{i+1} such that $d_{i+1} = 0$ will result

y_{i+1} doesn't matter for these cases

V. IMAGINARY-RADIX NUMBER SYSTEMS

An imaginary-radix number system for representing complex numbers has been introduced by Knuth [11] and recently modified by Slekys and Avizienis [12]. This imaginary number system is closely related to the negative-radix system and the algorithms for arithmetic operations in complex numbers are based on similar algorithms in the negative-radix system. In this section we show that a broad class of imaginary-radix systems may be defined and each of the previously considered systems can serve as a basis for an imaginary system. In particular, we are interested in the imaginary binary system based upon the 2's complement system since the latter is still the most frequently used one with numerous arithmetic algorithms and hardware implementations available.

Let $\mathcal{A}$ be a $2n$-digit system with an imaginary radix $i\sqrt{\beta}$ (where $i = \sqrt{-1}$ and β is a positive integer) and $\Lambda = \{\lambda_j\}$. The algebraic value of a $2n$-tuple $(x_{2n-1}, x_{2n-2}, \cdots, x_0)$ is

$$X = \sum_{j=0}^{2n-1} x_j \lambda_j (i\sqrt{\beta})^j. \tag{20}$$

The weight w_j of the jth digit x_j is

$$w_j = \lambda_j (i\sqrt{\beta})^j = \begin{cases} \lambda_j \beta^{j/2} & \text{if } j = 0 \bmod 4 \\ i\lambda_j \sqrt{\beta}\beta^{(j-1)/2} & \text{if } j = 1 \bmod 4 \\ -\lambda_j \beta^{j/2} & \text{if } j = 2 \bmod 4 \\ -i\lambda_j \sqrt{\beta}\beta^{(j-1)/2} & \text{if } j = 3 \bmod 4. \end{cases} \tag{21}$$

Consequently, the weights of the odd-positioned (even-positioned) digits are imaginary (real) and we may partition the complex number X into imaginary and real parts

$$X = \sum_{j \text{ even}} x_j \lambda_j (-1)^{j/2} \beta^{j/2} + i\sqrt{\beta} \sum_{j \text{ odd}} x_j \lambda_j (-1)^{(j-1)/2} \beta^{(j-1)/2}. \quad (22)$$

Replacing j by $2k$ for the real part, and by $2k + 1$ for the imaginary part yields

$$X = \sum_{k=0}^{n-1} x_{2k} \lambda_{2k} (-1)^k \beta^k + i\sqrt{\beta} \sum_{k=0}^{n-1} x_{2k+1} \lambda_{2k+1} (-1)^k \beta^k \triangleq \text{Re}\,[X] + i\,\text{Im}\,[X]. \quad (23)$$

Hence, the real part is a number in the system $\langle n, \beta, M_{\text{Re}} = \{\mu_k = (-1)^k \lambda_{2k}\}\rangle$ and similarly, the imaginary part when divided by $\sqrt{\beta}$ is a number in the system $\langle n, \beta, M_{\text{Im}} = \{\mu_k = (-1)^k \lambda_{2k+1}\}\rangle$. If $\Lambda = \{\lambda_j = 1\}$ both real and imaginary parts are represented in the negative-radix system [11], [12]. A different choice for Λ results in representation of the real (and imaginary) part in one of the systems considered in the previous sections.

The representations of the real part and imaginary part are interleaved in the $2n$-tuple and consequently, each part can be handled separately. Thus, all the properties and the arithmetic algorithms presented in previous sections apply here. For example, to generate the complex conjugate of a given number X, the additive inverse of the imaginary part has to be formed using any of the methods introduced in Sections II and III.

In the following we consider the bi-imaginary system $\langle 2n, i\sqrt{2}, \Lambda\rangle$ based upon the 2's complement system, i.e., the real and imaginary parts are represented in the $\langle n, 2, M = \{-1, 1, 1, \cdots, 1\}\rangle$ system. This complex number system has the following advantages over the system considered in [11] and [12]. The real and imaginary parts have a nearly symmetric range and the detection of their signs is simple; only the most significant bits (the sign bits) have to be checked.

To obtain 2's complement representations for the real and imaginary parts the characterizing vector Λ is selected as follows. The weights w_{2n-1} and w_{2n-2} must be negative; hence, (21) yields

$$\lambda_{2n-1} = \lambda_{2n-2} = \begin{cases} +1 & \text{if } n \text{ is even} \\ -1 & \text{if } n \text{ is odd.} \end{cases} \quad (24)$$

All other weights have to be positive; thus

$$\lambda_j = \begin{cases} +1 & \text{if } j = 0, 1 \bmod 4 \\ -1 & \text{if } j = 2, 3, \bmod 4 \end{cases}; \quad j = 0, 1, \cdots, 2n - 3. \quad (25)$$

The resulting weights are

$$w_j = \begin{cases} \beta^{j/2} & \text{if } j \text{ is even} \\ i\sqrt{\beta}\beta^{(j-1)/2} & \text{if } j \text{ is odd} \end{cases}; \quad j = 0, 1, \cdots, 2n - 3. \quad (26)$$

Example: The 10-bit complex number system based on the 2's complement system has the characterizing vector $\Lambda = (-1, -1, -1, -1, 1, 1, -1, -1, 1, 1)$. The representation of the number $Z_1 = 2 + i3\sqrt{2}$ is obtained by interleaving the 5-bit 2's complement representations of 2 and 3, i.e., 00010 and 00011, respectively. Hence, Z_1 is represented by 0000001110. The number $6 - i2\sqrt{2}$ is similarly represented by 1000110100.

Note that due to the factor $\sqrt{\beta}$ in (23) the quantity Im $[Z]/\sqrt{2}$ has to be determined when representing the number Z in our complex system. Since $\sqrt{2}$ is irrational, a rational imaginary part is represented in a nonterminating way, producing truncation and rounding errors [11], [12], We can overcome this problem by using the modification suggested in [12], i.e., a number Z is represented by $Z^* = \text{Re}\,[Z] + i(\sqrt{2}\,\text{Im}\,[Z])$.

When adding (subtracting) complex numbers we add (subtract) the real and imaginary parts separately and there is no advantage in representing the numbers in an imaginary-radix system. The possible usefulness of imaginary-radix systems is their facility of multiplication [11]. When multiplying two complex numbers we can multiply $2n$-bit complex numbers instead of performing four multiplications of n-bit real numbers and two add/subtract operations. If a shift and add type multiplication algorithm is employed, an ordinary shift can be used for even-indexed multiplier bits. For an odd-indexed multiplier bit a multiplication by $i\sqrt{2}$ is needed. We may use a unit similar to the one presented in Section IV or the following simpler method. When a $2n$-tuple is shifted to the left the even-positioned bits (the real part) are shifted to odd positions having the same λ, while odd-positioned bits (the imaginary part) are shifted to even positions having the opposite value of λ, i.e., these bits are multiplied by $-i\sqrt{2}$ instead of $i\sqrt{2}$. This can be corrected by forming the 2's complement of the imaginary part before shifting. Similarly, when dividing by $i\sqrt{2}$ the 2's complement of the real part is formed before shifting to the right.

Example: The product of $Z_1 = 2 + i3\sqrt{2}$ and $Z_2 = 3 + i2\sqrt{2}$ is given by

$$\begin{array}{r} 001110 \\ \times\, 001101 \\ \hline 001110 \\ 001110 \\ 100110 \\ \hline 0101100110. \end{array}$$

The algebraic value of the result is $Z_1 \times Z_2 = -6 + i13\sqrt{2}$.

In addition, the generation of the conjugate of a given complex number is greatly simplified, only the 2's complement of the imaginary part has to be formed.

In summary, existing hardware for 2's complement arithmetic can be used for complex number manipulations. Such hardware includes high-speed adders, multioperand adders, and fast multipliers.

VI. Conclusions

A unified approach to a class of number systems that contains the well-known and widely-used number systems has been presented. Such an approach allows a unified treatment of arithmetic operations in various number systems and thus enables the design of a single arithmetic unit capable of performing operations in several number systems. It has been shown by Zohar [7] that the use of negabinary arithmetic

where

$$c_0 = b_0 = 0.$$

Let $z_i \triangleq \lambda_i(x_i \pm y_i) + c_i - b_i$, then the rule for s_i is

$$s_i = (\lambda_i z_i) \bmod \beta. \tag{10}$$

For c_{i+1} and b_{i+1} we have two cases as follows.

Case 1: $z_i \geq \lambda_i s_i$, then $b_{i+1} = 0$ and $c_{i+1} = (z_i - \lambda_i s_i)/\beta$.

Case 2: $z_i < \lambda_i s_i$, then $c_{i+1} = 0$ and $b_{i+1} = (\lambda_i s_i - z_i)/\beta$.

Clearly, the carry and borrow outputs of stage $n-1$ indicate overflow and underflow, respectively.

For binary systems $\langle n, \beta = 2, \Lambda \rangle$ a truth table can be constructed and logical expressions derived [1]. For convenience, we replace the vector Λ by a binary vector whose elements satisfy $\gamma_i = \frac{1}{2}(1 + \lambda_i)$; $i = 0, 1, \cdots, n-1$. We define another binary variable α, which satisfies $\alpha = 0$ for addition and $\alpha = 1$ for subtraction. Using this notation, the resulting logical expressions are

$$s_i = x_i \oplus y_i \oplus (c_i + b_i), \tag{11}$$

$$c_{i+1} = \delta_i(u_i + y_i)c_i + \bar{\delta}_i \bar{u}_i \bar{y}_i c_i + \delta_i u_i y_i \bar{b}_i, \tag{12}$$

$$b_{i+1} = \bar{\delta}_i(u_i + y_i)b_i + \delta_i \bar{u}_i \bar{y}_i b_i + \bar{\delta}_i u_i y_i \bar{c}_i \tag{13}$$

where $u_i = x_i \oplus \alpha$ and $\delta_i = \gamma_i \oplus \alpha$.

These general equations can be reduced to any given binary system $\langle n, \beta = 2, \Lambda \rangle$. For example, in the case of the conventional binary system, the appropriate substitutions yield the well-known equations for a full adder/subtractor [1].

Equations (12) and (13) have the same functional form and we may write

$$c_{i+1} = f(\delta_i, u_i, y_i, c_i, b_i) = f(\gamma_i \oplus \alpha, u_i, y_i, c_i, b_i), \tag{14}$$

$$b_{i+1} = f(\bar{\delta}_i, u_i, y_i, b_i, c_i) = f(\bar{\gamma}_i \oplus \alpha, u_i, y_i, b_i, c_i). \tag{15}$$

Thus, in a stage of the parallel adder the internal carry logic is identical to the internal borrow logic. Moreover, if a positive stage (i.e., $\gamma_i = \lambda_i = 1$) has been implemented using

$$A_i: \begin{cases} s_i = x_i \oplus y_i \oplus (c_i + b_i) \\ c_{i+1} = f(\bar{\alpha}, u_i, y_i, c_i, b_i) \\ b_{i+1} = f(\alpha, u_i, y_i, b_i, c_i \end{cases} \tag{16}$$

the same circuit can be used for a negative stage (i.e., $\gamma_i = 0$ or $\lambda_i = -1$) if the carry and borrow lines are connected crosswise. For example, a negabinary adder is shown in Fig. 1.

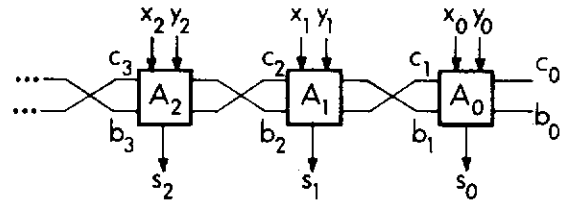


Fig. 1. A negabinary parallel adder.

IV. Multiplication and Division

Multiplication and division in fixed-radix systems are usually done as a series of additions (or subtractions) and simple multiplications or divisions by the radix β (or some integral power of β). In a positive-radix system such a multiplication or division is performed by a proper shift of the intermediate result. In a mixed-radix system not all shift operations correspond to some multiply or divide operations. In the following we investigate the conditions under which a logical shift operation has some arithmetic meaning.

When a number is shifted k positions to the left, the weight of the digit shifted from position i to position $j = i + k$ is either multiplied by β^k if $\lambda_i = \lambda_j$ or by $-\beta^k$ if $\lambda_i \neq \lambda_j$. The complete shift left operation is equivalent to some multiply operation if and only if all shifted digits are multiplied by the same factor, which is either β^k or $-\beta^k$. Hence, a k-position shift left operation has an equivalent multiply operation in the following two cases:

$$1)\ \lambda_i = \lambda_{i+k}; \qquad i = 0, 1, \cdots, n-k-1;$$

$$2)\ \lambda_i \neq \lambda_{i+k}; \qquad i = 0, 1, \cdots, n-k-1.$$

In case 1) there is in Λ a cycle of length k, e.g., in a negative-radix system there is a cycle of length two and as a result double shift left operations multiply the number by β^2. Similarly, in case 2) there is a cycle of length $2k$ since the relation $\lambda_i \neq \lambda_{i+k}$ implies $\lambda_i = \lambda_{i+2k}$; $i = 0, 1, \cdots, n-2k-1$. Identical observations can be done regarding shift right operations. We conclude that unless there is in Λ a cycle of length two or less multiplication (or division) by β cannot be done by shifting. However, we may design a shifting unit with some modifications that are necessary to obtain the multiply/divide by β operation.

Let $(x_{n-1}, \cdots, x_1, x_0)$ represent a number X in a given system $\mathcal{A}$ and let $(s_{n-1}, \cdots, s_1, s_0)$ be the representation in $\mathcal{A}$ of $S = \beta^a X$, where a is 1 (-1) for a multiply (divide) by β operation. The digit s_i is obtained from the shifted digit x_{i-a} (i.e., left shift for multiply and right shift for divide), a carry digit c_i, and a borrow b_i. The last two are needed to correct the result of the shift operation whenever $\lambda_i \neq \lambda_{i-a}$.

The equation for the multiply/divide by β operation is derived from (9) by replacing the $\lambda_i(x_i \pm y_i)$ term with $\lambda_{i-a}x_{i-a}$, yielding

$$\lambda_{i-a}x_{i-a} + c_i - b_i = \lambda_i s_i + \beta c_{i+1} - \beta b_{i+1};$$
$$i = 0, 1, \cdots, n-1 \tag{17}$$

where

$$c_0 = b_0 = x_{-1} = x_n = 0.$$

Thus, the adder-subtractor (with some added connections) can be used for multiplication-division by β or a separate simpler circuit may be used.

It is known that most shift and add type algorithms for multiplication and division in a positive-radix system can easily be extended to negative-radix systems [4]–[6] and to signed digit systems [9]. Similarly, the extension of these algorithms to the general number system $\langle n, \beta, \Lambda \rangle$ is straightforward if the shift operations are replaced, when necessary, by multiplications or divisions by β. Consequently, we do not present here the details of these extensions since no new insight is gained. In some fast binary multiplication (division) algorithms

provides a simpler and more attractive solution to the hardware design problem for a special application. Other number systems considered in this paper may prove to be useful for other applications.

In the last section imaginary-radix number systems are considered. A novel complex number system, which might be more attractive than previously introduced number systems, is presented.

REFERENCES

[1] I. Koren and Y. Maliniak, "A unified approach to a class of number systems," in *Proc. 4th Symp. Comput. Arithmetic*, Oct. 1978, pp. 25–28.

[2] H. L. Garner, "The classification of finite number systems," in *Proc. IFIPS Conf.*, 1968, pp. 256–259.

[3] M. Davio and J. P. Deschamps, "Addition in signed digit number systems," in *Proc. 8th Symp. Multiple-Valued Logic*, May 1978.

[4] P. V. Sankar, S. Chakrabarti, and E. V. Krishnamurthy, "Arithmetic algorithms in a negative base," *IEEE Trans. Comput.*, vol. C-22, pp. 120–125, Feb. 1973.

[5] D. P. Agrawal, "Arithmetic algorithms in a negative base," *IEEE Trans. Comput.*, vol. C-24, pp. 998–1000, Oct. 1975.

[6] C. K. Yuen, "A note on base-2 arithmetic logic," *IEEE Trans. Comput.*, vol. C-24, pp. 325–329, Mar. 1975.

[7] S. Zohar, "New hardware realizations of nonrecursive digital filters," *IEEE Trans. Comput.*, vol. C-22, pp. 328–338, Apr. 1973.

[8] H. L. Garner, "Number systems and arithmetic," in *Advances in Computers*, vol. 6. New York: Academic, 1965, pp. 131–194.

[9] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, pp. 389–400, Sept. 1961.

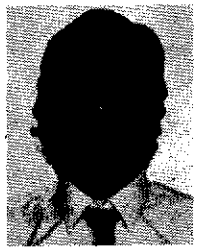
[10] G. W. Reitwiesner, "Binary arithmetic," in *Advances in Computers*, vol. 1. New York: Academic, 1960, pp. 244–265.

[11] D. E. Knuth, "An imaginary number system," *Commun. Ass. Comput. Mach.*, vol. 3, pp. 245–247, Apr. 1960.

[12] A. G. Slekys and A. Avizienis, "A modified bi-imaginary number system," in *Proc. 4th Symp. Comput. Arithmetic*, Oct. 1978, pp. 48–55.

Israel Koren (S'72–M'76) was born on June 23, 1945. He received the B.Sc. (cum laude), M.Sc., and D.Sc. degrees from the Technion–Israel Institute of Technology, Haifa, in 1967, 1970, and 1975, respectively, all in electrical engineering.

From 1968 to 1971 he was with the Computer Center, Israel Ministry of Defense. In 1972 he joined the Department of Electrical Engineering at the Technion, where he became a Lecturer in 1975. From 1976 to 1978 he was an Assistant Professor in the Department of Electrical Engineering and Computer Science at the University of California, Santa Barbara. In 1979 he was an Assistant Professor in the Department of Electrical Engineering-Systems at the University of Southern California, Los Angeles. Presently, he is with the Departments of Electrical Engineering and Computer Science at the Technion–Israel Institute of Technology. His current research interests are computer architecture, digital computer arithmetic, and reliability of digital systems.



Yoram Maliniak received the B.Sc. degree in electrical engineering from Tel-Aviv University, Tel-Aviv, Israel and the M.S. degree in electrical engineering from the University of California, Santa Barbara, in 1978 and 1979, respectively. He is currently working toward the Ph.D. degree in electrical and computer engineering at the University of California, Santa Barbara.

Mr. Maliniak is a member of the Association for Computing Machinery, Tau Sigma Beta, and Eta Kappa Nu.