J Sign Process Syst (2010) 58:17–27
DOI 10.1007/s11265-008-0265-8

An Efficient Method for Evaluating Complex Polynomials

Miloš D. Ercegovac · Jean-Michel Muller

Received: 30 October 2007 / Revised: 26 May 2008 / Accepted: 26 August 2008 / Published online: 24 September 2008

Abstract We propose an efficient hardware-oriented method for evaluating complex polynomials. The method is based on solving iteratively a system of linear equations. The solutions are obtained digit-by-digit on simple and highly regular hardware. The operations performed are defined over the reals. We describe a complex-to-real transform, a complex polynomial evaluation algorithm, the convergence conditions, and a corresponding design and implementation. The latency and the area are estimated for the radix-2 case. The main features of the method are: the latency of about m cycles for an m-bit precision; the cycle time independent of the precision; a design consisting of identical modules; and digit-serial connections between the modules. The number of modules, each roughly corresponding to serial-parallel multiplier without a carry-propagate adder, is $2(n+1)$ for evaluating an n-th degree complex polynomial. The method can also be used to compute all successive integer powers of the complex argument with the same latency and a similar implementation cost. The design allows straightforward tradeoffs between latency and cost: a factor k decrease in cost leads to a factor k increase in latency. A similar tradeoff between precision, latency and cost exists. The proposed method is attractive for programmable platforms because of its regular and repetitive structure of simple hardware operators.

Keywords Complex polynomials · Complex powers · Complex-to-real transform · Digit-by-digit algorithms · Left-to-right evaluation

1 Introduction

In this paper we describe a new method for evaluating complex polynomials suitable for hardware implementation. It has a latency of m cycles for m-bit precision and a repetitive implementation which corresponds roughly to $2(n+1)$ serial-parallel multipliers for polynomials of degree n. The coefficients and argument are fixed-point complex numbers. The proposed method is a generalization to the complex domain of a polynomial evaluation method over the reals introduced as the E-method [2, 3], and recently described in [7]. Briefly, the complex E-method (CE-method) achieves the same latency as the real E-method at twice the cost. This paper is based on the report [8], where the complex E-method is introduced and discussed in general terms, and on a shorter version presented at the ASAP 2007 conference [9].

The method uses the following approach: (i) a polynomial is mapped onto a system of linear equations, (ii) a transform is applied to change the complex domain to the real domain, and (iii) the system is solved in a digit-by-digit manner, the most-significant digit first, all elements of the solution vector in parallel. The main characteristics of the method are: (i) the m-digit

This is an expanded version of a paper presented at the ASAP'2007 conference [9].

M. D. Ercegovac (✉)
Computer Science Department, University of California at Los Angeles, Los Angeles, CA 90095, USA
e-mail: milos@cs.ucla.edu

J.-M. Muller
CNRS-Laboratoire LIP, projet Arénaire, Inria, Université de Lyon, Ecole Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France
e-mail: Jean-Michel.Muller@ens-lyon.fr

solution is computed in about m steps, each step consisting of a sum of number-by-digit products, (ii) the cycle time does not depend on the precision m (if redundant additions are used), (iii) for a system of order n, the shortest latency requires n elementary units (modules) for the real part, and n units for the imaginary part, and (iv) the elementary units are interconnected with digit-wide links. As a special case, the proposed method is used to compute consecutive integer powers of a complex argument.

The CE-method is applicable in some other computations. It is particularly efficient when the coefficient matrix of the linear system is sparse as is the case of rational functions and tridiagonal systems [12] which require two off-diagonal elements. Other examples are special expressions such as the complex multiply-add, sum of products and sum of squares [10].

Complex polynomials appear in many areas such as digital signal and image processing, control systems, and applied mathematics. A method of Horner type for evaluating complex polynomials is proposed in [1] at the algorithm level, implicitly assuming a software implementation. The method uses $O(n)$ full-precision multiplications and $O(n)$ additions for a complex polynomial of degree n. If these multiplications and additions are performed in a sequential order, the latency of the method is about $n \times T_{MULT-ADD}$ which is significantly slower than our method. If a parallel algorithm for polynomial evaluation is used, the total time is about $\log n \times T_{MULT-ADD}$ which is still slower than our method at a much higher cost of digit-parallel interconnections. The evaluation of complex polynomials on equispaced arguments [14], error analysis [16], and a complexity analysis [17] are other examples of research involving complex polynomials.

In the next section we describe the transformation which maps computation from the complex to the real domain. In Section 3 we show the CE-method for complex polynomials. In Section 4 iterations and convergence conditions are considered. Implementation aspects are discussed in Section 5. The special case of complex integer powers is described in Section 6. Some extensions and applications to complex division are given in Section 7.

2 Complex-Real (CR) Transforms

As commonly known, complex numbers can be represented by 2×2 skew-symmetric matrices

$$x + iy \leftrightarrow \begin{pmatrix} x & -y \\ y & x \end{pmatrix} \tag{1}$$

This is an isomorphism which holds for complex addition and multiplication - the operations used in the proposed method :

$$\begin{aligned}(a+ib)+(c+id) &\leftrightarrow \begin{pmatrix} a & -b \\ b & a \end{pmatrix} + \begin{pmatrix} c & -d \\ d & c \end{pmatrix} \\ &= \begin{pmatrix} a+c & -b-d \\ b+d & a+c \end{pmatrix} \\ &\leftrightarrow (a+c)+i(b+d)\end{aligned} \tag{2}$$

$$\begin{aligned}(a+ib)\times(c+id) &\leftrightarrow \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \times \begin{pmatrix} c & -d \\ d & c \end{pmatrix} \\ &= \begin{pmatrix} ac-bd & -(ad+bc) \\ ad+bc & ac-bd \end{pmatrix} \\ &\leftrightarrow (ac-bd)+i(bc+ad)\end{aligned} \tag{3}$$

Consequently, an $m \times n$ matrix of complex numbers can be represented as a $2m \times 2n$ matrix of real numbers. For $n \times n$ complex matrices, considered in this paper, the transform from the complex domain to the real domain is shown next.

Definition 1 The *CR-transform* of the n-dimensional *complex* linear system

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & \cdots & a_{3,n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \times \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_n \end{pmatrix} \tag{4}$$

is the $2n$-dimensional *real* linear system

$$\begin{pmatrix} a^r_{1,1} & -a^i_{1,1} & a^r_{1,2} & -a^i_{1,2} & \cdots & a^r_{1,n} & -a^i_{1,n} \\ a^i_{1,1} & a^r_{1,1} & a^i_{1,2} & a^r_{1,2} & \cdots & a^i_{1,n} & a^r_{1,n} \\ a^r_{2,1} & -a^i_{2,1} & a^r_{2,2} & -a^i_{2,2} & \cdots & a^r_{2,n} & -a^i_{2,n} \\ a^i_{2,1} & a^r_{2,1} & a^i_{2,2} & a^r_{2,2} & \cdots & a^i_{2,n} & a^r_{2,n} \\ a^r_{3,1} & -a^i_{3,1} & a^r_{3,2} & -a^i_{3,2} & \cdots & a^r_{3,n} & -a^i_{3,n} \\ a^i_{3,1} & a^r_{3,1} & a^i_{3,2} & a^r_{3,2} & \cdots & a^i_{3,n} & a^r_{3,n} \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \\ a^r_{n,1} & -a^i_{n,1} & a^r_{n,2} & -a^i_{n,2} & \cdots & a^r_{n,n} & -a^i_{n,n} \\ a^i_{n,1} & a^r_{n,1} & a^i_{n,2} & a^r_{n,2} & \cdots & a^i_{n,n} & a^r_{n,n} \end{pmatrix} \times \begin{pmatrix} s^r_1 \\ s^i_1 \\ s^r_2 \\ s^i_2 \\ s^r_3 \\ s^i_3 \\ \vdots \\ s^r_n \\ s^i_n \end{pmatrix}$$

$$= \left(t^r_1, t^i_1, t^r_2, t^i_2, t^r_3, t^i_3, \ldots, t^r_n, t^i_n\right)^T \tag{5}$$

where $a_{j,k} = a^r_{j,k} + ia^i_{j,k}$, $s_j = s^r_j + is^i_j$ and $t_j = t^r_j + it^i_j$. These two linear systems are *equivalent* which can be

shown either by a straightforward calculation, or by using the fact that the transformation

$$x + iy \rightarrow \begin{pmatrix} x & -y \\ y & x \end{pmatrix}$$

is an isomorphism.

In other words, the real linear system Eq. 5 is obtained from the complex linear system Eq. 4 by replacing each element $x + iy$ by the 2×2 matrix defined in Eq. 1. In the next section we consider a hardware-oriented method for solving such a system.

3 CE-Method: An Overview

The E-method [2, 3] provides an iterative approach to solving diagonally dominant linear systems in the real domain. The method has characteristics desirable for efficient hardware implementation: the basic operators are digit-vector multiplexers, and redundant adders of $[q : 2]$ type $(q \in \{3, 4, 6\})$, and registers. The overall structure consists of n elementary units, interconnected digit-serially. The method computes one digit of each element of the solution vector per iteration in the MSDF (Most Significant Digit First) manner which allows digit-serial communication. The modules operate concurrently. The time to obtain the solution to m digits of precision is about m cycles (iterations), and cycle time can be made independent of the precision by using redundancy in the representation of intermediate results (residuals and output digits). The amount of hardware required is roughly related to the number of nonzero terms of the matrix of the system, which makes the proposed method very efficient in hardware resources when the matrix of the system is sparse. Typical applications of the method are evaluation of polynomial and rational functions, since these correspond to such sparse linear systems. Concerning polynomials, the solution of the linear system

$$\begin{pmatrix} 1 & -x & 0 & 0 & 0 \cdots & 0 \\ 0 & 1 & -x & 0 & 0 \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \cdots & \vdots \\ 0 & 0 \cdots & 0 & 0 & 1 & -x \\ 0 & 0 & 0 \cdots & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix}$$

$$= (p_0, p_1, \ldots, p_{n-1}, p_n)^T$$

is

$$\mathbf{y} = \begin{pmatrix} p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n \\ p_1 + p_2 x + \cdots + p_n x^{n-1} \\ \vdots \\ p_{n-1} + p_n x \\ p_n \end{pmatrix}$$

that is, the first element of the solution vector y is

$$y_0 = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n = p(x)$$

Now consider the evaluation of polynomials with complex coefficients and complex argument:

$$p(z) = p_0 + p_1 z + p_2 z^2 + \ldots + p_n z^n$$

where the p_j's and z are complex numbers. As in the real case, the desired value $p(z)$ is clearly equal to the first component of the solution of the linear system

$$\begin{pmatrix} 1 & -z & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & -z & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & -z & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \ldots & 0 & 1 & \end{pmatrix} \times \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} \tag{6}$$

The method cannot directly solve the linear system Eq. 6. If we define real numbers x and y such that $x + iy = z$, and p_j^r and p_j^i such that $p_j = p_j^r + ip_j^i$, then we can apply the CR-transform to Eq. 6, and get the linear system with the coefficient matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & -x & y & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -y & -x & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & -x & y & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & -y & -x & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 & 0 & -x & y \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 1 & -y & -x \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The first two components of the solution **s** of the linear system

$$\mathbf{A} \times \left(s_0^r, s_0^i, s_1^r, s_1^i, \ldots, s_{n-1}^r, s_{n-1}^i, s_n^r, s_n^i\right)^T = \left(p_0^r, p_0^i, p_1^r, p_1^i, \ldots, p_{n-1}^r, p_{n-1}^i, p_n^r, p_n^i\right)^T \quad (7)$$

are equal to the real and imaginary parts of

$$p_0 + p_1 z + p_2 z^2 + \cdots + p_n z^n.$$

For instance, in the case $n = 3$, we get the solutions shown in Fig. 1.

The linear system Eq. 7 is easily solved by the proposed method, provided that it is diagonally dominant. The iterations and convergence conditions are discussed in the next section. Note that the method does not evaluate directly the expressions given for the solution s_0. These would require at least 16+16 full-precision multiplications which, assuming enough multipliers are available, would take at least 3 consecutive multiply times. Moreover, the reduction of product terms would require a [10:2] reduction. Of course, all the interconnections are of full precision. Instead, as explained later, the proposed method computes s_0 on 12 serial-parallel (left-to-right) multipliers, including the additions, in about *one serial-parallel multiplication time*. Moreover, the interconnections are digit-serial which simplifies routing and reduces power dissipation.

4 Iteration and Convergence Conditions

For simplicity, we discuss here radix-2 iterations. Adaptation to higher radices is rather straightforward. The radix-2 method solves the n-dimensional real linear system

$$\mathbf{As} = \mathbf{p}$$

by using the following basic recursion on residuals:

$$\mathbf{w}^{(j)} = 2 \times \left[\mathbf{w}^{(j-1)} - \mathbf{A}\mathbf{d}^{(j-1)}\right] \quad (8)$$

with $\mathbf{w}^{(0)} = [p_0, p_1, \ldots, p_n]^T$, and $\mathbf{d}^{(j)} = [d_0, d_1, \ldots, d_n]^T$ where the digits $d_k^{(j)}$ are in $\{-1, 0, 1\}$. Define the number $D_k^{(j)} = d_k^{(0)}.d_k^{(1)}d_k^{(2)}\ldots d_k^{(j)}$ (the $d_k^{(j)}$ are the digits of a radix-2 signed-digit representation of $D_k^{(j)}$). By induction, we show that

$$\mathbf{w}^{(j)} = 2^j\left[\mathbf{w}^{(0)} - AD^{(j-1)}\right] \quad (9)$$

and, for a precision of m bits, after performing $m+1$ iterations

$$\mathbf{AD}^{(m)} - \mathbf{p} = -2^{-(m+1)}\mathbf{w}^{(m+1)} \quad (10)$$

where $\mathbf{D}^{(m)}$ is the solution **s** with error less than 2^{-m}. Using Eq. 9, one can show that if the residuals $|w_k^{(j)}|$ are bounded, then for all k, $D_k^{(j)}$ goes to s_k as j goes to infinity. The problem at step j is to find a *selection function* that produces a value of the digits $d_k^{(j)}$ from the residuals $w_k^{(j)}$ such that the values $w_k^{(j+1)}$ remain bounded. In [3], the following selection function as a form of rounding is proposed

$$SEL(x) = \begin{cases} \text{sign } x \times \lfloor |x| + 1/2 \rfloor, & \text{if } |x| \le 1 \\ \text{sign } x \times \lfloor |x| \rfloor, & \textit{otherwise}, \end{cases} \quad (11)$$

We apply the selection function to an estimate of the residual to avoid carry-propagate addition: $d_k^{(j)} = SEL(\hat{w}_k^{(j)})$, where $\hat{w}_k^{(j)}$ is a low-precision approximation to $w_k^{(j)}$.

Consider next in more detail evaluation of an n-degree complex polynomial.

$$p_n z^n + p_{n-1} z^{n-1} + \cdots + p_0$$

Figure 1 Solutions to system Eq. 7.

$$\mathbf{s} = \begin{pmatrix} -3\,xy^2p_3^r + x^3p_3^r - 3\,yx^2p_3^i + x^2p_2^r - 2\,xyp_2^i + xp_1^r + y^3p_3^i - y^2p_2^r - yp_1^i + p_0^r \\ -y^3p_3^r + 3\,yx^2p_3^r - 3\,y^2xp_3^i + 2\,yxp_2^r - y^2p_2^i + yp_1^r + x^3p_3^i + x^2p_2^i + xp_1^i + p_0^i \\ -y^2p_3^r + x^2p_3^r - 2\,yxp_3^i + xp_2^r - yp_2^i + p_1^r \\ x^2p_3^i + 2\,xyp_3^r - y^2p_3^i + yp_2^r + xp_2^i + p_1^i \\ xp_3^r - yp_3^i + p_2^r \\ yp_3^r + xp_3^i + p_2^i \\ p_3^r \\ p_3^i \end{pmatrix}$$

at the complex point $z = x + iy$, with $p_k = p_k^r + ip_k^i$. The matrix of the CR-transform, introduced in Section 3, is

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & -x & y & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -y & -x & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & -x & y & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & -y & -x & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & 0 & 1 & 0 & -x & y \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 1 & -y & -x \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

We adjust the notation for **w**, **d** and the residual recurrence Eq. 8 to adapt them to the complex case. The residual vector $\mathbf{w}^{(j)}$ is

$$\mathbf{w}^{(j)} = \left[w_{0,r}^{(j)}, w_{0,i}^{(j)}, w_{1,r}^{(j)}, w_{1,i}^{(j)}, \cdots, w_{n,r}^{(j)}, w_{n,i}^{(j)}\right],$$

and its initial value are given by

$$w_{k,r}^{(0)} = p_k^r, \quad w_{k,i}^{(0)} = p_k^i$$

The digit-vector $d^{(j)}$ will be denoted

$$\mathbf{d}^{(j)} = \left[d_{0,r}^{(j)}, d_{0,i}^{(j)}, d_{1,r}^{(j)}, d_{1,i}^{(j)}, \cdots, d_{n,r}^{(j)}, d_{n,i}^{(j)}\right].$$

Therefore, iteration Eq. 8 becomes

for $k = 0, \ldots, n-1$

$$\begin{aligned} w_{k,r}^{(j)} &= 2\left[w_{k,r}^{(j-1)} - d_{k,r}^{(j-1)} + x d_{k+1,r}^{(j-1)} - y d_{k+1,i}^{(j-1)}\right] \\ w_{k,i}^{(j)} &= 2\left[w_{k,i}^{(j-1)} - d_{k,i}^{(j-1)} + y d_{k+1,r}^{(j-1)} + x d_{k+1,i}^{(j-1)}\right] \end{aligned} \tag{12}$$

for $k = n$

$$\begin{cases} w_{n,r}^{(j)} = 2\left[w_{n,r}^{(j-1)} - d_{n,r}^{(j-1)}\right] \\ w_{n,i}^{(j)} = 2\left[w_{n,i}^{(j-1)} - d_{n,i}^{(j-1)}\right] \end{cases}$$

We now examine the convergence conditions. The iterations converge to the desired result if the residual vector $w^{(j)}$ is bounded. Define constants ξ, α and Δ (with $0 \le \Delta < 1$) such that

1. $|x| + |y| \le \alpha$;
2. for any k between 0 and n, $|p_k^r| \le \xi$, $|p_k^i| \le \xi$, $|w_{k,r}^{(j)} - \hat{w}_{k,r}^{(j)}| \le \frac{\Delta}{2}$, and $|w_{k,i}^{(j)} - \hat{w}_{k,i}^{(j)}| \le \frac{\Delta}{2}$

Since $|d_{k,r}^{(j-1)} - \hat{w}_{k,r}^{(j-1)}| \le 1/2$ and $|d_{k,i}^{(j-1)} - \hat{w}_{k,i}^{(j-1)}| \le 1/2$, from Eq. 19 we find

$$|w_{k,r}^{(j)}| \le 2\left(\frac{1}{2} + \frac{\Delta}{2} + \alpha\right) = 1 + \Delta + 2\alpha. \tag{13}$$

The same bound holds for $|w_{k,i}^{(j)}|$. For this bound to be feasible, we must assure that a suitable choice of $d_{k,r}^{(j)}$ and $d_{k,r}^{(j)}$ in $\{-1, 0, 1\}$ is possible. This requires that $|w_{k,r}^{(j)}|$ and $|w_{k,i}^{(j)}|$ should be less than 3/2. This immediately gives the following condition

$$\Delta + 2\alpha \le \frac{1}{2} \tag{14}$$

Now, let us turn to the initial values. Since $|w_{k,r}^{(0)}|$ and $|w_{k,i}^{(0)}|$ must also be less than 3/2, we get

$$\xi \le \frac{3}{2}. \tag{15}$$

Consider the following example: we wish to evaluate

$$p(z) = (1+i)\, z^3 - (0.5 + 1.25\, i)\, z^2 + z + 1.$$

at point

$$z = \frac{1}{100} + \frac{i}{10}.$$

We assume that $\Delta = 0$ (that is, we use non-redundant residuals). We get:

- Initialization:

  $$\begin{aligned} \mathbf{w}^{(0)} &= \left[p_0^r, p_0^i, p_1^r, p_1^i, p_2^r, p_2^i, p_3^r, p_3^i\right]^t \\ &= \left[1, 0, 1, 0, -0.5, -1.25, 1, 1\right]^t \end{aligned}$$

- Step 1: from $w^{(0)}$ and the selection function, we get

  $$\mathbf{s}^{(0)} = [1, 0, 1, 0, 0, -1, 1, 1]^t,$$

  which gives

  $$\mathbf{w}^{(1)} = [0.02, 0.2, 0.2, -0.02, -1.18, -0.28, 0, 0]^t.$$

- Step 2: from $w^{(1)}$ and the selection function, we get

  $$\mathbf{s}^{(1)} = [0, 0, 0, 0, -1, 0, 0, 0]^t,$$

  which gives

  $$\mathbf{w}^{(2)} = [0.04, 0.4, 0.38, -0.24, -0.36, -0.56, 0, 0]^t.$$

- After 20 iterations, the number

  $$d_{0,r}^{(0)}.d_{0,r}^{(1)}d_{0,r}^{(2)} \cdots d_{0,r}^{(20)} + i \times d_{0,i}^{(0)}.d_{0,i}^{(1)}d_{0,i}^{(2)} \cdots d_{0,i}^{(20)}$$

  is equal to

  $$\frac{533789}{524288} + \frac{57727}{524288}\, i \approx 1.018121719 + 0.110105514\, i$$

  whereas the exact value of $p(z)$ is

  $$p(z) = 1.018121 + 0.110106\, i$$

Table 1 shows first 17 iterations of the computation of $p(z)$ using the CE-method recurrences. For brevity, only the real and imaginary signed bits of all four

Table 1 Computing $p(z)$ using CE-method.

j	$d_{0,r}^{(j)}$	$d_{0,i}^{(j)}$	$d_{1,r}^{(j)}$	$d_{1,i}^{(j)}$	$d_{2,r}^{(j)}$	$d_{2,i}^{(j)}$	$d_{3,r}^{(j)}$	$d_{3,i}^{(j)}$	$s_{0,r}^{(j)}$	$s_{0,r}^{(j)}$
0	1	0	1	0	0	−1	1	1	1.0	0.0
1	0	0	0	0	−1	0	0	0	1.0	0.0
2	0	0	0	0	0	−1	0	0	1.0	0.0
3	0	1	1	0	−1	1	0	0	1.0	0.001
4	0	0	0	−1	1	0	0	0	1.0	0.001
5	1	0	−1	0	−1	0	0	0	1.00001	0.001
6	−1	−1	1	−1	0	−1	0	0	1.000001	0.000111
7	0	0	0	1	0	0	0	0	1.000001	0.000111
8	1	0	0	0	1	0	0	0	1.00000101	0.000111
9	−1	0	−1	0	0	0	0	0	1.000001001	0.000111
10	1	1	0	0	0	1	0	0	1.0000010011	0.0001110001
11	−1	0	−1	0	0	−1	0	0	1.00000100101	0.0001110001
12	0	−1	1	0	−1	1	0	0	1.00000100101	0.000111000011
13	0	0	−1	−1	1	−1	0	0	1.00000100101	0.000111000011
14	1	0	1	1	−1	0	0	0	1.00000100101001	0.000111000011
15	0	0	0	−1	1	0	0	0	1.00000100101001	0.000111000011
16	0	0	1	0	0	1	0	0	1.00000100101001	0.000111000011
17	−1	0	−1	1	0	1	0	0	1.00000100101000111	0.000111000011

solution elements are shown. We also show the accumulated real and imaginary part of $p(z)$ in the binary form.

Exactly as in the real case, even if polynomial p and point z do not satisfy the convergence constraints, one can easily "transform" them using mere shifts, so that $p(z)$ can be computed using the proposed method. Once Δ is chosen, and α is defined as $\frac{1}{4} - \Delta/2$, this is done as follows:

1. Find the smallest integer k such that $|\Re(z/2^k)| + |\Im(z/2^k)|$ should be less than α.
2. Now, $p(z) = \pi(t)$, where the degree-m coefficient of polynomial π is $2^{mk} p_m$, and $t = z/2^k$. If at least one of the coefficients of π has the absolute value of its real or imaginary part greater than $\xi = 3/2$, then divide π by 2^ℓ, where ℓ is the smallest integer such that $\rho = \pi/2^\ell$ has the absolute value of the real and imaginary parts of its coefficients less than ξ.
3. What we actually compute using the method is $\rho(z/2^k)$. This result will then be multiplied by 2^ℓ (a simple left-shift) to get $p(z)$. Consequently $\ell + k$ extra iterations are performed.

Consider the following example:

$$p(z) = 4iz^2 + (1 - i)z + 5,$$

in the domain defined by $-1 < \Re(z) < +1$ and $-1 < \Im(z) < +1$, with $\alpha = 1/8$. First, the smallest k such that $t = z/2^k$ satisfies $|\Re(t)| + |\Im(t)| < \alpha$ is $k = 4$. We therefore get a new polynomial

$$\pi(t) = 256 \times 4it^2 + 16 \times (1 - i)t + 5.$$

Second, the smallest ℓ such that the real and imaginary parts of the coefficients of $\pi(t)/2^\ell$ have absolute value less than $3/2$ is $\ell = 10$. By dividing $\pi(t)$ by 2^ℓ we get the polynomial

$$\rho(t) = it^2 + \left(\frac{1}{64} - \frac{i}{64}\right)t + \frac{5}{1024}.$$

We evaluate ρ using the CE-method. It is straightforward to check that ρ satisfies the convergence conditions and that

$$p(z) = 2^{10}\rho\left(\frac{z}{2^4}\right).$$

5 Implementation

In this section we discuss implementation of the proposed method. The main difference from implementation of a real domain evaluation E-method is that the number of non-zero off-diagonal elements doubles: for the polynomial case from one to two. This has two consequences. First, the bounds on the elements are smaller by a factor of two, and second, the cycle time is increased as explained later in this section. The corresponding implementations considered for the real domain method are discussed in [3, 7].

A scheme for evaluation of complex polynomials is shown in Fig. 2 for $n = 3$ and the corresponding elementary unit (PEU) is illustrated in Fig. 3. A bit-parallel bus transmits x and y values in a broadcast mode, while the real and imaginary coefficients p^r and p^i can be preloaded into local storage in each PEU in separate cycles prior to the evaluation iterations so that

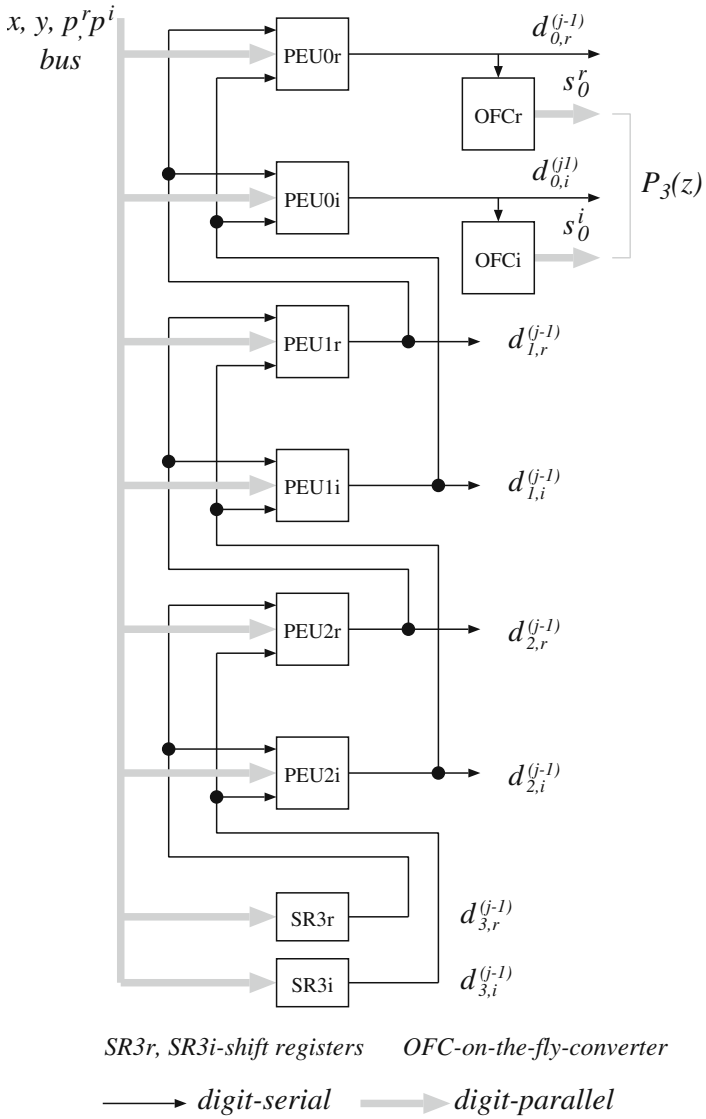

Figure 2 Scheme for evaluating complex polynomial ($n = 3$).

the latency is not affected. Note that the initialization cycles could be shorter than the iteration cycles. These design aspects are not discussed here.

A block diagram of an elementary unit PEU0*r* (real part only) for polynomial evaluation is shown in Fig. 3. The modules are:

- Registers (4)
- Multiple generators MG (2), producing $\{-1, 0, 1\} \times x$ and $\{-1, 0, 1\} \times y$, with buffers
- Multiplexer MUX for initializing the residual
- A [4:2] adder
- Output digit selection SEL (a table or a gate network)

The digit-serial outputs of $PEU0$ can be converted into digit-parallel form using on-the-fly converters $OFCr$ and $OFCi$. The cycle time, in terms of a full adder (complex gate) delay t, is estimated as

$$T_{PEU} = t_{BUFF} + t_{MG} + t_{SEL} + t_{[4:2]} + t_{REG} \approx (0.4 + 0.3 + 1 + 1.3 + 0.9)t = 3.9t \quad (16)$$

The cost, again in terms of area of a full adder A_{FA}, is estimated as

$$\begin{aligned} A_{PEU}(m) &= A_{SEL} + 2A_{BUFF} + (m+2)[2A_{MG} \\ &\quad + A_{MUX} + A_{[4:2]} + 4A_{REG} + 2A_{OFC}] \\ &\approx [5 + 2 \times 0.4 + (m+2)(3 \times 0.45 \\ &\quad + 2.3 + 4 \times 0.6 + 2 \times 2.1)]A_{FA} \\ &\approx [26 + 10m]A_{FA} \end{aligned} \quad (17)$$

The cost is estimated as area occupied by modules using the area of a full-adder A_{FA} as the unit. The areas

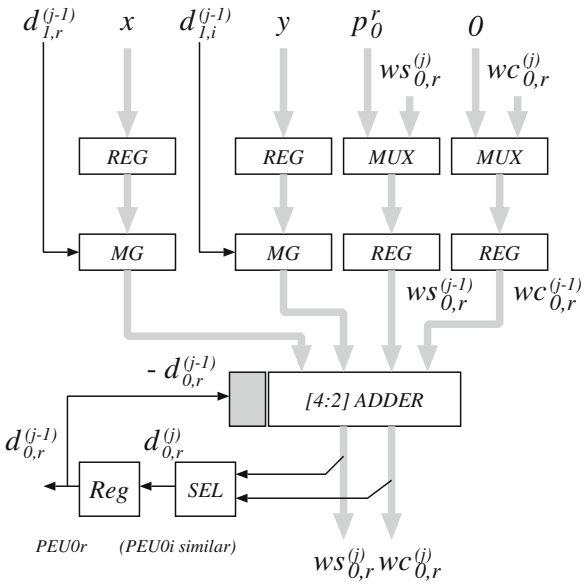

Figure 3 Block diagram of elementary unit.

of primitive modules are: Register $A_{REG} = 0.6A_{FA}$; buffer $A_{BUFF} = 0.4A_{FA}$; MUX $A_{MUX} = 0.45A_{FA}$; multiple generator MG $A_{MG} = 0.45A_{FA}$; [4:2] adder $A_{[4:2]} = 2.3A_{FA}$; SEL $A_{SEL} = 5A_{FA}$, and on-the-fly converters $A_{OFC} = 2A_{MUX} + 2A_{REG} = 2.1A_{FA}$.

6 Computing Consecutive Integer Powers of Complex Argument

The scheme proposed for evaluating complex polynomials can be used in evaluating integer powers of a complex argument [10]. Let us first consider this computation in the real domain. Consecutive integer powers $x^2, x^3, ..., x^k$ are computed in parallel by solving the corresponding linear system as indicated next.

$$\begin{pmatrix} 1 & -x & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -x & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 1 & -x \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} s_0 \\ s_1 \\ \vdots \\ s_{k-1} \\ s_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ x \end{pmatrix}$$

and the integer powers are obtained as

$$s_0 = x^k, \quad s_1 = x^{k-1}, \ldots, s_{n-1} = x^2$$

The mapping in the complex domain is shown next. The complex argument is $z = x + iy$.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & -x & y & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -y & -x & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & -x & y & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & -y & -x & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 & 0 & -x & y \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 1 & -y & -x \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The components of the solution s of the linear system

$$\mathbf{A} \times \left(s_0^r, s_0^i, s_1^r, s_1^i, \ldots, s_{k-1}^r, s_{k-1}^i, s_k^r, s_k^i\right)^T = (0, 0, 0, 0, \ldots, 0, 0, x, y)^T \tag{18}$$

are equal to the integer powers of z.

The residual recurrences are

$$\begin{aligned} w_{h,r}^{(j)} &= 2\left[w_{h,r}^{(j-1)} - d_{h,r}^{(j-1)} + x d_{h+1,r}^{(j-1)} - y d_{h+1,i}^{(j-1)}\right] \\ w_{h,i}^{(j)} &= 2\left[w_{h,i}^{(j-1)} - d_{h,i}^{(j-1)} + y d_{h+1,r}^{(j-1)} + x d_{h+1,i}^{(j-1)}\right] \end{aligned} \tag{19}$$

for $h = k$,

$$w_{k,r}^{(j)} = 2\left[w_{k,r}^{(j-1)} - d_{k,r}^{(j-1)}\right]$$

$$w_{k,i}^{(j)} = 2\left[w_{k,i}^{(j-1)} - d_{k,i}^{(j-1)}\right]$$

with the initial conditions for $h = 0, \ldots, k-1$

$$w_{h,r}^{(0)} = 0, \; w_{h,i}^{(0)} = 0$$

and

$$w_{k,r}^{(0)} = x, \; w_{k,i}^{(0)} = y$$

The following conditions need to be satisfied for the convergence of iterations

$$|x| + |y| \leq \alpha \tag{20}$$

From condition Eq. 14 and using $\Delta = 1/8$, we get $\alpha \leq 3/16$ and $|x|, \; |y| \leq 3/32$ which guarantee convergence of the algorithm. When needed, a range reduction can be achieved by scaling of the initial values [3].

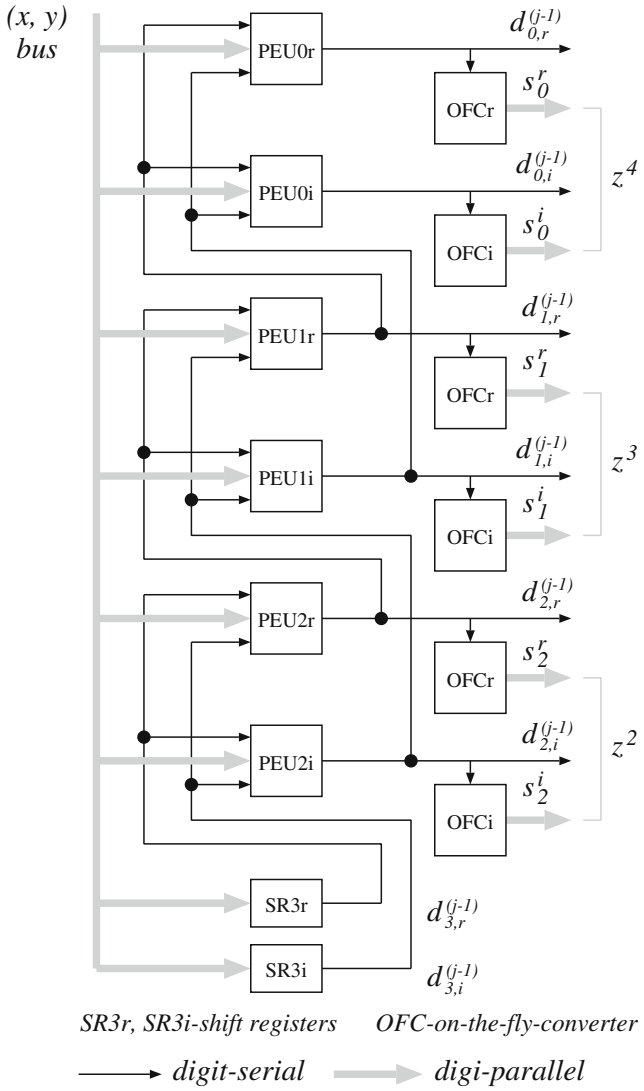

Figure 4 Scheme for CIP computation.

A scheme for implementing the computation of consecutive integer powers of a complex argument (CIP scheme) is shown in Fig. 4. The corresponding elementary unit is similar to the EU of the complex polynomial scheme, illustrated in Fig. 3. The cycle time and cost are similar. Again, a bit-parallel bus transmits x and y parts of the complex argument $z = x + iy$ in a broadcast mode. The load phase is simpler than for complex polynomials because there is no need to load complex coefficients. A total cost of an m-bit $CIPk$ scheme for evaluating $z^2,\ z^3, \ldots, z^k$ is

$$A_{CIPk}(m) = (k-1) \times A_{PEU}(m) + 2 \times (m+2) A_{REG}$$

If the results are needed in bit-parallel, conventional form, $2(k-1)$ on-the-fly converters (OFC) are used at a cost of $2(k-1) \times [2(m+2)(A_{REG} + A_{MUX})]$.

7 Possible Applications of the Method

In [1] the author discusses how in control system theory, the frequency response of a linear continuous time invariant system is obtained by substituting $s = i\omega$ in the system transfer function $H(s) = q(s)/p(s)$ where q and p are one-variable polynomials. Evaluating such polynomials would be easily done using our method. Moreover, we could easily take into account the fact that s has no real part: Eq. 19 would be simplified, since variable x of that equation would disappear.

Another application of complex polynomials is related power amplifiers used in communications systems. In [21] implementation of predistortion is based on a 5-th order polynomial evaluated in the analog domain. Our technique maybe applicable for implementations in the digital domain. A time-domain adaptive algorithm for compensation of nonlinear distortions in high power amplifiers using complex polynomials is presented in [15] .

In [13] a technique is introduced to obtain coefficients in rational function approximations to satisfy the convergence conditions of the E-method in the real domain. Extending this idea to the complex domain appears plausible and attractive in reducing the cost of implementation. Generalizing that technique to the complex domain is rather simple.

In a paper [4] a digit-recurrence algorithm for performing complex divisions is presented. The algorithm requires a prescaling step to get an approximation to the reciprocal of the divisor. In that paper, this is done using a table that can be very large. Instead of that, we propose to approximate the reciprocal of $1 + z$ (with, say, $|z| \leq 1/2$, with reduction to this rather large domain easily done) by the truncated series

$$1 - z + z^2 - z^3 + \cdots + (-1)^k z^k$$

Using k terms, we have an error bounded by 2^{-k}. Moreover, a brief examination of [4] and this paper shows that much hardware can be common to both methods.

8 Summary

With the exception of complex addition and multiplication, complex operations are typically not implemented in hardware. Recently, hardware-oriented methods for complex division and square root have been introduced [4–6, 11]. The method for evaluating complex polynomials discussed in this paper can be extended to evaluation of complex rational functions and complex operators such as multiply-add and sum of products

[8, 10]. Complex online arithmetic algorithms for basic operations and certain linear-algebra computations, suitable for FPGA implementation, have been developed in [18] and presented in several papers [19, 20].

In this paper we extend a class of algorithms for complex arithmetic suitable for hardware implementation. We presented a method for evaluating complex polynomials by solving diagonally-dominant linear systems in complex domain by a digit-recurrence algorithm. The latency is roughly m cycles for m bits of precision, independent of the order of the system. This does not take into account potentially needed scaling steps. The cycle time is independent of m. We discussed the transform from real to complex numbers, the iteration, and the convergence conditions. Implementation is given at a high level with estimates of the cost and latency. Consecutive integer powers of a complex argument, which can be computed using the proposed polynomial scheme, are also discussed.

Acknowledgement We thank the reviewers for comments and suggestions.

References

1. Benmahammed, K. (1994). Evaluation of complex polynomials in one and two variables. *Multidimensional Systems and Signal Processing*, *5*, 245–261.
2. Ercegovac, M. D. (1975). *A general method for evaluation of functions and computation in a digital computer*. PhD thesis, Dept. of Computer Science, University of Illinois, Urbana-Champaign.
3. Ercegovac, M. D. (1977). A general hardware-oriented method for evaluation of functions and computations in a digital computer. *IEEE Transactions on Computers*, *C-26*(7), 667–680.
4. Ercegovac, M. D., & Muller, J.-M. (2003). Complex division with prescaling of operands. In *IEEE international conference on application-specific systems, architectures and processors* (pp. 293–303).
5. Ercegovac, M. D., & Muller, J.-M. (2004). Design of a complex divider. In *Proc. SPIE on advanced signal processing algorithms, architectures, and implementations XII* (pp. 51–59).
6. Ercegovac, M. D., & Muller, J.-M. (2004). Complex square root with operand prescaling. In *IEEE international conference on application-specific systems, architectures and processors* (pp. 293–303).
7. Ercegovac, M. D., & Lang, T. (2004). *Digital arithmetic*. San Francisco: Morgan Kaufmann (an Imprint of Elsewier Science).
8. Ercegovac, M. D., & Muller, J.-M. (2007). *Solving systems of linear equations in complex omain: Complex e-method. LIP report no. 2007-2*. Lyon: École Normale Supérieure de Lyon.
9. Ercegovac, M. D., & Muller, J.-M. (2007). A hardware-oriented method for evaluating complex polynomials. In *IEEE international conference on application-specific systems, architectures and processors*.
10. Ercegovac, M. D., & Muller, J.-M. (2007). Complex multiply-add and other related operators. In *Proc. SPIE on advanced signal processing algorithms, architectures, and implementations XII* (pp. 1–11).
11. Ercegovac, M. D., & Muller, J.-M. (2007). Complex square root with operand prescaling. *Journal of VLSI Signal Processing*, *49*, 19–30
12. Ercegovac, M. D., & Muller, J.-M. (2006). Arithmetic processor for solving tridiagonal systems of linear equations. *Proc. 40th asilomar conference on signals, systems and computers* (pp. 337–340).
13. Brisebarre, N., & Muller, J.-M. (2004). Functions approximable by E-fractions. In *38th asilomar conference on signals, systems and computers, pacific grove, California* (Nov).
14. Nutall, A. H. (1987). Efficient evaluation of polynomials and exponentials of polynomials for equispaced arguments. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *ASSP-35*, 1486–1487.
15. Ohmori, Y. D., & Sano, H. (2003). Time-domain adaptive compensation for nonlinear distortion of high power amplifiers. In *Proc. of the American control conference* (Vol. 1, pp. 356 – 361) (June).
16. Olver, F. W. J. (1986). Error bounds for polynomial evaluation and complex arithmetic. *IMA Journal of Numerical Analysis 6*, 373–379.
17. Reif, J. H. (1997). Approximate complex polynomial evaluation in near constant work per point. In *STOC 97* (pp. 30–39).
18. McIlhenny, R. (2002). *Complex number on-line arithmetic for reconfigurable hardware: Algorithms, implementations, and applications*. PhD. Dissertation, University of California, Los Angeles.
19. McIlhenny, R., & Ercegovac, M. D. (2005). On the design of an on-line complex matrix inversion unit. In *Proc. 39th asilomar conference on signals, systems and computers* (pp. 1172–1176).
20. McIlhenny, R., & Ercegovac, M. D. (2006). On the design of an on-line complex householder transform. In *Proc. 40th asilomar conference on signals, systems and computers* (pp. 318–322).
21. Westesson, E., & Sundstrom, L. (1999). A complex polynomial predistorter chip in CMOS for baseband or IF linearization of RF power amplifiers. In *Proceedings of the 1999 IEEE international symposium on circuits and systems* (Vol. 1, pp. 206 – 209).

Miloš D. Ercegovac is a Professor and a former Chair in the Computer Science Department of the Henry Samueli School of Engineering and Applied Science, University of California at Los Angeles, where he has been on the faculty since 1975. He earned his MS (Õ72) and PhD (Õ75) in computer science from the University of Illinois, Urbana-Champaign, and BS in electrical engineering (Õ65) from the University of Belgrade, Serbia. Dr. Ercegovac has specialized for over 30 years in research and teaching in digital arithmetic, digital and computer system design, and parallel architectures. His dedication to teaching and research has also resulted in several co-authored books: two in the area of digital design (Digital Systems and Hardware/Firmware Algorithms, Wiley & Sons, 1985, and Introduction to Digital Design, Wiley & Sons, 1999), and two in digital arithmetic (Division and Square Root: Digit-Recurrence Algorithms and Implementations, Kluwer Academic Publishers, 1994, and Digital Arithmetic, Morgan Kaufmann Publishers - a Division of Elsevier, 2004.) Dr. Ercegovac has been involved in organizing the IEEE Symposia on Computer Arithmetic since 1978. He served as an associate editor of the IEEE Transactions on Computers 1988–1992 and as a subject area editor for the Journal of Parallel and Distributed Computing 1986–1993. Dr. Ercegovac's work has been recognized by his election in 2003 to IEEE Fellow and to Foreign Member of the Serbian Academy of Sciences and Arts in Belgrade, Serbia. He is also a member of the ACM and of the IEEE Computer Society.

Jean-Michel Muller was born in Grenoble, France, in 1961. He received his Ph.D. degree in 1985 from the Institut National Polytechnique de Grenoble. He is Directeur de Recherches (senior researcher) at CNRS, France, and he is the former head of the LIP laboratory (LIP is a joint laboratory of CNRS, Ecole Normale Supérieure de Lyon, INRIA and Université Claude Bernard Lyon 1). His research interests are in Computer Arithmetic. Dr. Muller was co-program chair of the 13th IEEE Symposium on Computer Arithmetic (Asilomar, USA, June 1997), general chair of SCAN'97 (Lyon, France, Sept. 1997), general chair of the 14th IEEE Symposium on Computer Arithmetic (Adelaide, Australia, April 1999). He is the author of several books, including "Elementary Functions, Algorithms and Implementation" (2nd edition, Birkhäuser Boston, 2006). He served as associate editor of the IEEE Transactions on Computers from 1996 to 2000. He is a senior member of the IEEE.